



Création et utilisation des templates

Lotfi KHEDIRI

Un template
est le meilleur
moyen
d'organiser et
de rendre le
HTML depuis
votre
application

Les templates
dans Symfony
sont créés avec
Twig

Twig : un
moteur de
template php
flexible, rapide
et sécurisé

Twig propose
un langage du
même nom
utilisé dans les
templates au
lieu du php.



Introduction

Twig Templating Language



Le langage de création de template **Twig** permet d'écrire des template concis et lisibles qui sont plus conviviaux pour les concepteurs Web et, à plusieurs égards, plus puissants que les modèles PHP.



La syntaxe de Twig est basée sur ces trois constructions:

`{{ ... }}`, utilisé pour afficher le contenu d'une variable ou le résultat de l'évaluation d'une expression;

`{% ... %}`, utilisé pour exécuter une logique, telle qu'une conditionnelle ou une boucle;

`{# ... #}`, utilisé pour ajouter des commentaires au template (contrairement aux commentaires HTML, ces commentaires ne sont pas inclus dans la page rendue).



Twig est livré avec une longue liste de balises , **filtres** et **fonctions** disponibles par défaut



Twig Templating Language

```
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome to Symfony!</title>
  </head>
  <body>
    <h1>{{ page_title }}</h1>

    {% if user.isLoggedIn %}
      Hello {{ user.name }}!
    {% endif %}

    {# ... #}
  </body>
</html>
```

créer un nouveau fichier dans
le répertoire templates/ pour stocker le
contenu du template



Ensuite, créez un *contrôleur* qui restitue
ce template et lui transmet les variables
nécessaires (dans un array)



```
$this->render("chemin du template",  
[param1,param2,...])
```



Création
de
templates

Variables de template

- Un besoin courant de modèles est **d'afficher** les valeurs stockées dans les modèles transmis par le contrôleur ou le service
- Les variables stockent généralement des **objets** et des **tableaux** au lieu de chaînes, de **nombres** et de valeurs **booléennes**
- **Twig** fournit un accès rapide aux variables PHP complexes

```
<p>{{ user.name }} added this comment on  
{{ comment.publishedAt|date }}</p>
```

Lien vers des pages

- Utilisez la fonction Twig **path()** pour créer un lien vers ces pages et passez le nom de la route comme premier argument et les paramètres de la route comme deuxième argument facultatif

```
<a href="{ { path('blog_index') } }">Homepage</a>
```

```
{# ... #}
```

```
{% for post in blog_posts %}
```

```
    <h1>
```

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

```
    </h1>
```

```
    <p>{{ post.excerpt }}</p>
```

```
{% endfor %}
```

Lien vers les ressources CSS, JavaScript et image

- ▢ Si un template doit être lié à un actif statique (par exemple une image), Symfony fournit une fonction Twig **asset()** pour aider à générer cette URL.
- ▢ Cette fonction permet de rendre l'application plus portable.

```
{# the image lives at "public/images/logo.png" #}  
  
  
{# the CSS file lives at "public/css/blog.css" #}  
<link href="{{ asset('css/blog.css') }}" rel="stylesheet" />
```


La variable globale **app** de Twig

- ▮ app : Objet contextuel injecté automatiquement dans chaque modèle Twig
- ▮ Donne accès à certaines informations de l'application.
- ▮

```
<p>Username: {{ app.user.username ?? 'Anonymous user' }}</p>
```
- ▮

```
{% if app.debug %}
```
- ▮

```
<p>Request method: {{ app.request.method }}</p>
```
- ▮

```
<p>Application Environment: {{ app.environment }}</p>
```
- ▮

```
{% endif %}
```

La variable globale **app** de Twig

- La variable **app** (qui est une instance de [AppVariable](#)) donne accès à ces variables:

variable	Rôle
app.user	L' objet utilisateur actuel ou null si l'utilisateur n'est pas authentifié.
app.request	Stocke les données de demande en cours
app.session	Objet Session qui représente <i>la session de l' utilisateur</i> courant ou null s'il n'y en a pas.
app.flashes	Un tableau de tous les messages flash stockés dans la session. Vous pouvez également obtenir uniquement les messages d'un certain type (par exemple <code>app.flashes('notice')</code>).
app.environment	Le nom de l' environnement de configuration courant (dev, prod, etc.).
app.debug	Vrai si en mode débogage . Faux sinon.
app.token	Un objet TokenInterface représentant le jeton de sécurité.

Injecter des variables automatiquement dans tous les modèles Twig

- ❑ Twig permet d'injecter automatiquement une ou plusieurs variables dans tous les modèles.
- ❑ Ces variables globales sont définies dans l'option **twig.globals** à l'intérieur du fichier de configuration principal de Twig:

```
# config/packages/twig.yaml
twig:
    # ...
    globals:
        ga_tracking: 'UA-xxxxx-x'
```

<p>The Google tracking code is:
{{ ga_tracking }}</p>

Rendu d'un template dans les contrôleurs

- ▢ Si votre contrôleur s'étend du AbstractController , utilisez la méthode **render()**:

```
// the `render()` method returns a `Response` object with the  
// contents created by the template  
return $this->render('product/index.html.twig', [  
    'category' => '...',  
    'promotions' => ['...', '...'],  
]);
```

- ▢ Si le contrôleur ne dérive pas de AbstractController, vous devrez récupérer les services dans votre contrôleur et utiliser la méthode render() du service twig.

L'utilitaire Dump de Twig

- ❑ Symfony fournit une fonction **dump ()** comme alternative améliorée à la fonction PHP `var_dump()` .
- ❑ Cette fonction est utile pour inspecter le contenu de n'importe quelle variable et vous pouvez également l'utiliser dans les templates Twig.
- ❑ Pour l'utiliser, installer le composant `composer require symfony/var-dumper` l'application:

```
{% dump articles %}  
  
{% for article in articles %}  
    {# the contents of this variable are dumped inside the  
page contents and they are visible on the web page #}  
    {{ dump(article) }}
```

Inclusion de templates

- Si certains codes Twig sont répétés dans plusieurs templates, vous pouvez les extraire dans un seul "fragment de template" et les inclure dans d'autres templates

```
{# templates/blog/index.html.twig #}  
  
{# ... #}  
{{ include('blog/_user_profile.html.twig') }}
```

- La fonction `include()` de Twig prend comme argument le chemin du template à inclure.
- Le template inclus a accès à toutes les variables du template qui l'inclut

Incorporer le résultat de l'exécution d'un contrôleur dans un template (imbrication)

```
{# templates/base.html.twig #}

{# ... #}
<div id="sidebar">

    {# if the controller is associated with a route, use the path() or url() functions #}
    {{ render(path('latest_articles', {max: 3})) }}
    {{ render(url('latest_articles', {max: 3})) }}

    {# if you don't want to expose the controller with a public URL,
       use the controller() function to define the controller to execute #}
    {{ render(controller(
        'App\\Controller\\BlogController::recentArticles', {max: 3}
    )) }}


</div>
```

Pour plus de détails :

<https://symfony.com/doc/current/templates.html#embedding-controllers>

Héritage et mises en page des templates

- Le concept d' héritage de modèle Twig est similaire à l'héritage de classe PHP.
- Définir un **template parent** à partir duquel d'autres templates peuvent s'étendre et les templates enfants peuvent remplacer des parties du template parent.
- En pratique, le template **base.html.twig** ressemblerait à ceci:



```
{# templates/base.html.twig #}
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title> {% block title %} My Application{% endblock %} </title>
    {% block stylesheets %}
      <link rel="stylesheet" type="text/css" href="/css/base.css"/>
    {% endblock %}
  </head>
  <body>
    {% block body %}
      <div id="sidebar">
        {% block sidebar %}
          <ul>
            <li><a href="{{ path('homepage') }}">Home</a></li>
            <li><a href="{{ path('blog_index') }}">Blog</a></li>
          </ul>
        {% endblock %}
      </div>
      <div id="content">
        {% block content %}{% endblock %}
      </div>
    {% endblock %}
  </body>
```

Héritage et mises en page des templates

- ▮ La balise de bloc Twig définit les sections de page qui peuvent être **remplacées** dans les templates enfants.
- ▮ Ils peuvent être **vides**, comme le bloc "**content**" ou définir un contenu par défaut, comme le bloc "**title**", qui s'affiche lorsque les templates enfants ne les remplacent pas.
- ▮ Un template enfant "**layout.html.twig**" pourrait ressembler à ceci:
- ▮ ***{# templates/blog/layout.html.twig #}***

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

```
{% block title%} <h1>Blog</h1> {% endblock %}
```

```
{% block content %}
```

```
<p>qsn,qhjgsgdghjsqgdgsqdgjqj</p>
```

```
{% endblock %}
```

Output Escaping

- ▢ Pour éviter les attaques XSS ([Cross-Site Scripting](#)), utilisez "output escaping" pour transformer les caractères qui ont une signification spéciale (par exemple, remplacer < par l'entité HTML <)
- ▢ Les applications Symfony sont sécurisées par défaut car elles effectuent un échappement de sortie automatique grâce à l' [option Twig autoescape](#)

```
<p>Hello {{ name }}</p>
{# if 'name' is '<script>alert('hello!')</script>', Twig will output this:
   '<p>Hello &lt;script&gt;alert(&#39;hello!&#39;)&lt;/script&gt;</p>' #}
```

- ▢ Si vous rendez une variable approuvée et **contenant du contenu HTML**, utilisez le [filtre row](#) de Twig pour désactiver l'échappement de sortie pour cette variable

```
<h1>{{ product.title|raw }}</h1>
{# if 'product.title' is 'Lorem <strong>Ipsum</strong>', Twig will output
   exactly that instead of 'Lorem &lt;strong&gt;Ipsum&lt;/strong&gt;' #}
```

[Pour plus d'information sur l'échappement de sortie de Twig](#)

Template Namespaces

- ❑ La plupart des applications stockent leurs templates dans le répertoire "**templates/**" par défaut.
- ❑ On peut stocker tout ou une partie d'entre eux dans des répertoires différents.
- ❑ Utilisez l'option **twig.paths** pour configurer ces répertoires supplémentaires.
- ❑ Chaque chemin est défini comme une paire **key: value** où key est le répertoire du template et value l'espace de noms Twig, ce qui est expliqué plus loin:

```
# config/packages/twig.yaml
twig:
  # ...
  paths:
    # directories are relative to the project root dir (but you
    # can also use absolute directories)
    'email/default/templates': ~
    'backend/templates': ~
```


Bundle Templates

- ❑ Si on installe des packages / bundles dans une application, ils peuvent **inclure** leurs propres templates Twig (dans le répertoire **Resources/views/** de chaque bundle).
- ❑ Pour éviter un conflit avec nos propres templates, Symfony ajoute les templates du bundle sous un espace de noms créé automatiquement après le nom du bundle.
- ❑ Par exemple, les templates d'un bundle appelé **AcmeFooBundle** sont disponibles sous l'espace de noms **AcmeFoo**.
- ❑ Si ce bundle comprend le template **<your-project>/vendor/acmefoo-bundle/Resources/views/user/profile.html.twig**,
On peut s'y référer en tant que **@AcmeFoo/user/profile.html.twig**.



Learn More

- ▢ [How to Inject Variables Automatically into all Templates](#)
- ▢ [How to Embed Asynchronous Content with include.js](#)
- ▢ [How to Write a custom Twig Extension](#)