5 minutes

Syntaxes de base Twig

Les bases des syntaxes `Twig`

Les trois syntaxes de base de `Twig` sont :

`{{ expression }}` : qui permet d'évaluer une expression, par exemple une variable `PHP` et de l'afficher.

`{% condition %}` : qui permet d'évaluer une structure de contrôle (par exemple `if` ou `else`, ou encore une boucle `for`).

`{# commentaire #}` : qui permet d'écrire des commentaires qui ne seront pas affichés dans le code source `HTML` retourné au client.

Utilisation de variables avec `Twig`

Les variables passées par un contrôleur `Symfony` à un `template Twig` sont appelées **variables de `template`** .

`Twig` permet un accès facilité aux propriétés ou méthodes des tableaux associatifs et des objets `PHP` en utilisant la notation `variable.prop`.

`Twig` va automatiquement essayer d'accéder à `variable['prop']` (pour les tableaux associatifs), puis à `variable->prop` (pour les propriété publiques des objets), puis à `variable->prop()` (pour les méthodes publiques des objets), puis à `variable->getProp()` (pour les accesseurs des objets).

Cela permet de pouvoir utiliser les mêmes `templates` que vous choisissiez de développer en `POO` ou non certaines entités.

Voici un exemple utilisant plusieurs structures de contrôle et une variable de `template` :

```
{# Structures de contrôle #}

{% if users %}
  <ul>
  {% for user in users %}
    <li>{{ user.username }}</li>
  {% endfor %}
  </ul>
{% endif %}
```

Copier

Le `if` vérifie qu'il y a au moins un élément dans le tableau `users`.

Ensuite nous itérons sur `users` pour créer un élément de liste par utilisateur avec une boucle `for`.

Pour passer la variable au `template`, il suffit de passer un tableau associatif avec les variables.

Les clés sont les noms des variables qui seront utilisés côté `template Twig` et les valeurs sont les valeurs assignées à ces variables.

Dans notre exemple, nous pourrions ainsi avoir :

```
<?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
  {
    $users = [
      ['username' => 'Bob'],
      ['username' => 'Paul'],
      ['username' => 'Jean'],
    ];
    return $this->render('test.html.twig', ['users' => $users]);
  }
}
```

Copier

Assigner des variables de `template` avec `set`

`set` permet d'assigner des valeurs à des variables de `template`.

Par exemple :

```
{% set mavar = 42 %}

{# Affiche 42 : #}
{{ mavar }}
```

Copier

Fonctionnalités avancées des conditions avec `Twig`

`Twig` propose les mots clés suivants pour être utilisés avec les conditions :

`not` : pour inverser le test d'une condition.

`and` : pour évaluer deux conditions cumulatives.

`or` : pour évaluer deux conditions alternatives.

Par exemple :

```
{% if user.subscribed and not user.locked %}
  Abonné
{% elseif user.subscribed and user.locked %}
  Bloqué !
{% else %}
  Pas abonné !
{% endif %}
```

Copier

Utilisation de boucles avec `Twig`

Utiliser des boucles dans un `template Twig` est très simple :

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

Copier

Vous pouvez itérer sur les clés et les valeurs :

```
<h1>Utilisateurs</h1>
<ul>
  {% for cle, user in users %}
    <li>{{ cle }}: {{ user.username }}</li>
  {% endfor %}
</ul>
```

Copier

Ou utiliser plusieurs valeurs mises à disposition par `Twig` sur la variable `loop` :

```
<h1>Utilisateurs</h1>
<ul>
  {% for user in users %}
    <li>{{ loop.index }} - {{ user.username }}</li>
  {% endfor %}
</ul>
```

Copier

Les principales valeurs sont : `loop.index` (numéro de l'itération, commençant à 1), `loop.first` (vaut `true` si c'est le premier élément), `loop.last` (vaut `true` si c'est le dernier élément) et `loop.length` (nombre d'éléments).

Exemple de la vidéo

Voici le code du `template test.html.twig` :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  </head>
  <body>
  {# Je suis un commentaire #}

  <h2>Nom:
    {{product.name}}</h2>
  <h2>Prix:
    {{product.price}}</h2>
  {% if product.price > 10000 %}
    <h3>C'est cher</h3>
  {% else %}
    <h3>Ce n'est pas cher</h3>
  {% endif %}

  <ul>
    {% for item in [1,2,3] %}
    <li>{{item}}</li>
```

```
    {% endfor %}
  </ul>

  <h2>Maj:
    {{product.date}}</h2>
  </body>
</html>
```

Copier

Voici le code du `DefaultController.php` :

```
<?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
  {
    $product = [
      'name' => 'Voiture Tesla',
      'price' => 50000,
      'date' => strtotime('yesterday')
    ];
    return $this->render('test.html.twig', ['product' => $product]);
  }
}
```

Copier