Développement web côté serveur avec Symfony

Module 03 - Vues et Twig

Objectifs

- Comprendre le fonctionnement de Twig
- Comprendre l'intérêt d'utiliser Twig
- Savoir utiliser les fonctionnalités principales de Twig
- Connaître les spécificités de Twig avec Symfony

Les délimiteurs

Twig, du HTML, mais pas seulement

- Un fichier HTML normal est un fichier Twig valide
- 3 délimiteurs permettent d'exécuter du code
- Compilés en PHP
- Avantages de Twig :
  - Minimaliste
  - Lisible
  - Système d'héritage
  - Performant

Les délimiteurs

Twig, une originalité critiquable

- Twig est très semblable à ...
- Nunjucks sous JavaScript qui est très semblable à ...
- Jinja sous Python qui est très semblable au ...
- DTL de Django qui est très semblable à ...
- Jtwig sous Java qui est très semblable à ...
- ...
- Mais tant mieux ☺

Les délimiteurs

Twig est minimaliste

- La référence Twig est disponible ici : https://twig.symfony.com/doc/
- Tout ce qu'on peut faire avec Twig tient sur un écran
- Symfony ajoute des fonctionnalités à Twig

Vues et Twig

Les délimiteurs

- Servent à séparer le code Twig du code HTML
- Composés d'un délimiteur ouvrant et d'un fermant
- Il est possible d'utiliser plusieurs délimiteurs sur une même page

Premier délimiteur Twig : {# #}

- `{# ... #}` permet d'écrire des commentaires
- Commentaires invisibles et multilignes
- Équivalent à `<?php /* un commentaire */ ?>` en PHP

Vues et Twig

Premier délimiteur Twig : {# #}

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Accueil</title>
</head>
<body>
    {#
        Ceci est un commentaire Twig !
        Ne s'affiche pas dans le navigateur \o/
    #}
    <h1>Home avec Twig !</h1>
</body>
</html>
```

Deuxième délimiteur Twig : {{ }}

- `{{ ... }}` affiche quelque chose
- Équivalent à l'echo PHP : `<?php echo "du texte"; ?>`
- On y inclut habituellement une variable

Troisième délimiteur Twig : {% %}

- `{% ... %}` exécute du code
- Équivalent à `<?php ?>`
- Permet de :
  - Créer des variables
  - Réaliser une boucle
  - Tester une condition
  - Exécuter des fonctions
  - ...
- Utilisé aussi pour les balises Twig

Vues et Twig

Utilisation de {{ }} et de {% %}

```twig
{# initialise une variable et lui affecte une valeur #}
{% set city = "Ouagadougou" %}

{# affiche la valeur de la variable #}
<p>{{ city }}</p>

{# teste la valeur de la variable #}
{% if city == "Ouagadougou" %}
    <p>Condition remlie !</p>
{% else %}
    <p>Ce n'est pas Ouaga</p>
{% endif %}
```

Les balises Twig (tags)

- apply
- autoescape
- **block**
- deprecated
- do
- embed
- **extends**
- flush
- **for**
- from
- **if**
- import
- **include**
- macro
- sandbox
- **set**
- use
- verbatim
- with

Vues et Twig

La balise `if`

- Permet d'exécuter des alternatives
- Se termine par `endif`
- Les opérateurs de comparaison habituels existent pour la plupart
- On utilise les mots-clefs `and` et `or`
- `else` et `elseif` pour créer des branches

Vues et Twig

La balise `set` et les variables

- `set` crée une variable et affecte une valeur
- Pas de `$`
- Les chaînes sont entre guillemets simples ou doubles : `"une chaîne"`
- Les nombres sont saisis sans guillemets : `2042`
- Les booléens aussi : `true`
- Les tableaux sont entre crochets : `["bleu", "jaune", "vert"]`
- Les objets entre accolades : `{'name': 'Johnny', 'age': 74}`

Vues et Twig

La balise if

```
{% set currentWeather = "sunny" %}
{% set currentTemperature = 22 %}

{% if currentWeather == "rainy" %}
    <p>On regarde un film !</p>
{% elseif currentWeather == "snowy" %}
    <p>Bataille de boules de neige !</p>
{% elseif currentWeather == "sunny" and currentTemperature > 20 %}
    <p>Tous à la playa ! Il fait {{ currentTemperature }} degrés !</p>
{% else %}
    <p>Parfait pour une balade.</p>
{% endif %}
```

La balise `for`

- `for` réalise une itération, une boucle
- Unique boucle en Twig
- Permet de :
  - Boucler sur un tableau ou un objet, avec ses clefs ou pas
  - Boucler sur une liste de nombres ou de lettres
  - Boucler en fonction de conditions
  - Avoir accès à des informations sur la boucle en cours

Vues et Twig

La balise `for` et les tableaux

```
{% set vegetables = ["carotte", "tomate", "patate"] %}
<h1>{{ products[2] }}</h1>

{% for vegetable in vegetables %}
    <p>{{ vegetable }}</p>
{% else %}
    <p>Pas de légumes ici.</p>
{% endfor %}
```

Vues et Twig

La balise `for` et les objets

```
{%
    set user = {
        'name': 'bingo',
        'age': 66,
        'isActive': true
    }
%}

<h2>{{ user.name }}</h2>
<dl>
    {% for key, val in user %}
        <dt>{{ key }}</dt><dd>{{ val }}</dd>
    {% endfor %}
</dl>
```

Les délimiteurs et balises Twig

Démonstration

Les filtres Twig

- Modifient une variable
- Peuvent s'enchaîner
- Ont parfois des options, des arguments
- Utilisés avec un | (*pipe*) à la suite d'une variable

Vues et Twig

Quelques filtres

- abs
- batch
- capitalize
- convert_encoding
- **date**
- date_modify
- default
- escape
- first
- **format**
- **join**
- json_encode
- keys
- last
- **length**
- **lower**
- markdown_to_html
- merge
- **nl2br**
- **number_format**
- **raw**
- replace
- reverse
- **round**
- slice
- sort
- split
- striptags
- title
- trim
- **upper**
- url_encode

Les principaux filtres et leur utilité

Filtre	Fonction	Exemple
date	Convertit un objet DateTime en chaîne	article.publishedDate|date("d/m/Y")
format	Remplace un marqueur par une chaîne	"%d notes sur un %s"|format(88, "piano")
join	Fusionne les valeurs d'un tableau en une chaîne	["a", "b", "c"]|join("/")
length	Retourne la longueur d'une variable	["a", "b", "c"]|length
lower	Convertit la chaîne en minuscule	"Mercredi"|lower
nl2br	Convertit les sauts de ligne en balise 	"un texte avec sauts de lignes"|nl2br
number_format	Formatte un nombre	1200.123|number_format(2, ',', ' ')
raw	Désactive l'échappement automatique	" le h1 est interprété "|raw
round	Arrondit un nombre	123.456|round(2)
upper	Convertit la chaîne en majuscule	"gros titre"|upper

Les filtres Twig

Démonstration

Vues et Twig

Blocs et héritage

- Permettent de créer un gabarit réutilisable
- Évitent la répétition de code

Vues et Twig

Blocs et héritage

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="img/favicon.png">
    <title>Première page</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>Première page</h1>
        <p>lorem ipsum</p>
    </main>

    <footer>
        ...
    </footer>
</body>
</html>
```

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="img/favicon.png">
    <title>Deuxième page</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <h1>Deuxième page</h1>
        <p>dolor sit amet</p>
    </main>

    <footer>
        ...
    </footer>
</body>
</html>
```

Solution des *include()* en PHP

```php
<?php include('top.php'); ?>

    <main>
        <h1>Deuxième page</h1>
        <p>dolor sit amet</p>
    </main>

<?php include('bottom.php'); ?>
```

Vues et Twig

Solution *gabarit* en PHP

```php
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="img/favicon.png">
    <title>Deuxième page</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <?php include('page_2.php'); ?>
    </main>

    <footer>
        ...
    </footer>
</body>
</html>
```

Vues et Twig

Solution de Twig 1/2

- Les blocs !
- Chaque bloc est personnalisable

```twig
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="img/favicon.png">
    <title>{% block title %}Un titre par défaut{% endblock %}</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
            </ul>
        </nav>
    </header>

    <main>
        {% block body %}{% endblock %}
    </main>

    <footer>
        ...
    </footer>
</body>
</html>
```

Vues et Twig

Solution de Twig 2/2

- L'héritage !

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

{% block body %}
    <p>Ce contenu remplace le block body !</p>
{% endblock %}

{% block title %}Deuxième page{% endblock %}
```

```twig
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="img/favicon.png">
    <title>{% block title %}Un titre par défaut{% endblock %}</title>
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">1</a></li>
                <li><a href="#">2</a></li>
                <li><a href="#">3</a></li>
                <li><a href="#">4</a></li>
                <li><a href="#">5</a></li>
            </ul>
        </nav>
    </header>

    <main>
        {% block body %}{% endblock %}
    </main>

    <footer>
        ...
    </footer>
</body>
</html>
```

Vues et Twig

Inclusion

- Beaucoup moins utilisé avec Twig
- Équivalent au `include()` de PHP
- Utile pour éviter une répétition de codes, présents sur plusieurs pages
- `{% include('fichier.html.twig') %}`

Vues et Twig

Blocs, héritage et inclusion

Démonstration

Gestion des assets : problème

- Fichiers .js, .css, images, documents .pdf, etc.
- Réécriture d'URL et liens relatifs

```
<link rel="stylesheet" href="css/style.css">
```

http://mon-site.com/ => http://mon-site.com/css/style.css => ok

http://mon-site.com/catalogue/ => http://mon-site.com/catalogue/css/style.css => pas ok

Gestion des assets : solution

- Utiliser {% asset("css/style.css") %}

```
<link rel="stylesheet" href="{{ asset('css/style.css') }}">
```

- Générera la bonne URL

Vues et Twig

Gestion des URLs internes

- Même problème que pour les assets
- Souplesse du système de routage
- Toujours utiliser `{{ path('home') }}` ou `{{ url('home') }}`

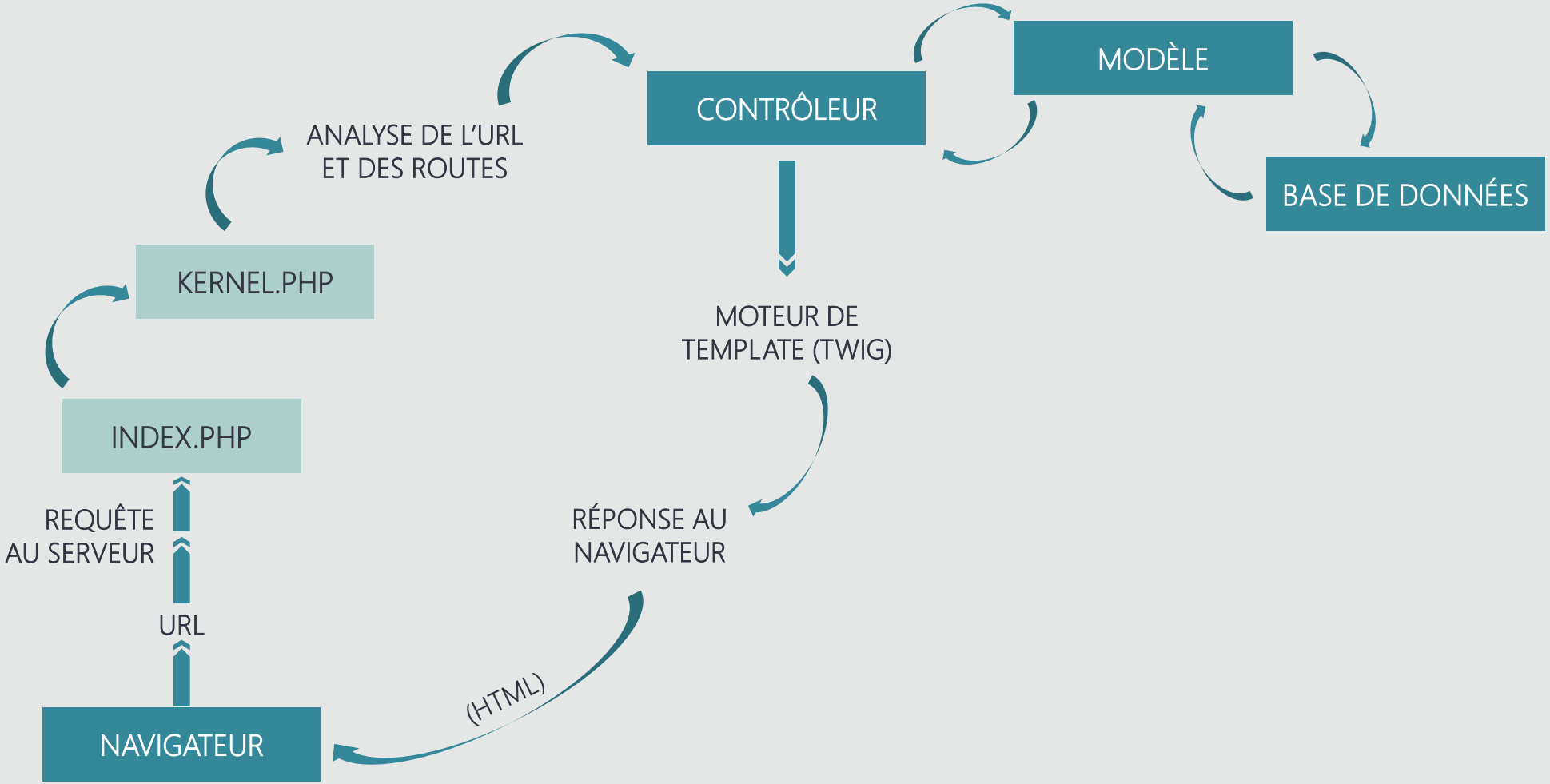
Vues et Twig

Gestion des assets et des URLs

Démonstration

Vues et Twig

Passer des variables à la vue



Vues et Twig

Passer des variables à la vue

```php
// dans un contrôleur

/**
 * @Route("/", name="main_home")
 */
public function home()
{
    //j'ai 2 données disponibles dans mon contrôleur
    $productCount = 222;
    $username = "Bertrand";

    return $this->render('main/home.html.twig', [
        "productCount" => $productCount,
        "username" => $username,
    ]);
}
```

Vues et Twig

Passer des variables à la vue

```php
public function home()
{
    //j'ai 2 données disponibles dans mon contrôleur
    $productCount = 222;
    $username = "Bertrand";

    //version plus compacte avec compact()
    return $this->render('main/home.html.twig', compact("productCount", "username"));
}
```

Vues et Twig

Passer des variables à la vue

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

{% block body %}
    <p>Bonjour {{ username }}</p>
    <div>{{ productCount }} produits disponibles !</div>
{% endblock %}

{% block title %}Deuxième page{% endblock %}
```

Vues et Twig

Passer des variables à la vue

Démonstration

Les attaques XSS

- Cross-Site Scripting
- Injection de code malveillant dans notre HTML
- Protection en PHP avec `<?php echo htmlentities($myVar); ?>`

Les attaques XSS sous Twig

- Protection automatique avec `{{ myVar }}` !

- Filtre `raw` pour désactiver la protection
- Filtre `striptags`
- Au besoin, utiliser HTMLPurifier ou autres

Attaques XSS

Démonstration

Conclusion

- Vous avez une bonne idée des fonctionnalités et avantages de Twig
- Vous savez comment utiliser Twig, avec ou sans Symfony