Simon GAUTIER

France

Twig

Variables, conditions, boucles

Filtres, opérateurs, fonctions, tests

Inclusion de template, héritage de template, macros

Intégration de Twig dans Symfony

Créer une extension pour Twig

Qu’est-ce que Twig ?

- Twig est un moteur de templating PHP
  - Ne nuit pas aux performances car le langage est transformé en PHP avant exécution
  - Sécurisé : par défaut les variables sont échappées avant affichage
- Pourquoi un moteur de templating ?
  - Faire de l’intégration en s’affranchissant du langage (en l’occurrence PHP)
  - Avoir du code HTML « propre », proche d’une maquette statique
- Site officiel : twig.symfony.com

PHP vs. Twig

- Afficher une variable (sécurisé par défaut avec Twig) :

```php
<?php
echo htmlspecialchars(
    $my_variable,
    ENT_QUOTES,
    'UTF-8'
)
?>
```

```
{{ my_variable }}
```

- Twig : des structures à haut niveau :
  - Beaucoup moins verbeux que l'équivalent PHP

```
{% for product in products %}
    * {{ product.name }}
{% else %}
    No products have been found.
{% endfor %}
```

Principe général – les tags en Twig

```
{# ... Commenter quelque chose ... #}

{% ... Faire quelque chose ... %}

{{ ... Afficher quelque chose ... }}
```

Affichage d'une variable

- Variable simple : `{{ my_variable }}`

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/display_variables.html.twig
https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DisplayVariablesController.php

Tester : https://cours-symfony.argetis.com/twig/display_variables

- Variable complexe : `{{ my_variable.name }}`
  - Plusieurs équivalent PHP, dans l'ordre de priorité suivant :

```php
echo $my_variable['name'];
echo $my_variable->name;
echo $my_variable->name();
echo $my_variable->getName();
echo $my_variable->isName();
echo $my_variable->hasName();
```

- Paramétrage de Twig (**`config/packages/twig.yaml`**)
  - Si strict_variables à true, une exception est déclenchée si la variable n'existe pas

```yaml
twig:
    strict_variables: '%kernel.debug%'
```

Les filtres

- Liste de tous les filtres standards : cf. documentation

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

```
{# Afficher une date correctement formatée #}
{{ createdAt|date('d/m/Y') }}<br><br>

{# Traitement de chaînes #}
{{ name|lower }}<br>
{{ name|upper }}<br>
{{ name|capitalize }}<br>
{{ name|title }}<br><br>

{# Afficher une valeur si la variable n'est pas définie #}
{{ author|default('No author') }}<br><br>

{# Cumuler les filtres #}
{{ tags|sort|join(', ') }}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/filters.html.twig
https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/FiltersController.php

Tester : https://cours-symfony.argetis.com/twig/filters

Les fonctions

- Liste de toutes les fonctions standards : cf. documentation

  attribute, block, constant, cycle, date, dump, include, max, min, parent, random, range, source, template_from_string

- Très utile pour débuguer : fonction dump
  - Installer le package var-dumper (affichage optimisé) : **`$ composer req debug-pack`**
  - Utile également pour les tests (vu dans un prochain chapitre)

- Exemple d'utilisation de fonctions :

```
{% set start_year = date() | date('Y') %}
{% set end_year = start_year + 5 %}

{% for year in start_year..end_year %}
    {{ cycle(['odd', 'even'], year) }}
{% endfor %}
```

Remarque : **`set`** permet de définir une variable dans Twig

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/functions.html.twig

Tester : https://cours-symfony.argetis.com/twig/functions

if / elseif / else

```
{% if stock > 10 %}
   Produit en stock
{% elseif stock > 1 %}
   Plus que {{ stock }} produits disponibles
{% elseif stock == 1 %}
   Dernier produit disponible
{% else %}
   Produit épuisé
{% endif %}
```

Tester : https://cours-symfony.argetis.com/twig/conditions/20
https://cours-symfony.argetis.com/twig/conditions/5
https://cours-symfony.argetis.com/twig/conditions/1
https://cours-symfony.argetis.com/twig/conditions/0

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/conditions.html.twig

```
{% for product in products %}
   {% if product.active %}
      <h2>{{ product.name }}</h2>
      <p>{{ product.description }}</p>
   {% endif %}
{% else %}
   Aucun produit disponible
{% endfor %}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/conditions-for.html.twig

Tester : https://cours-symfony.argetis.com/twig/conditions-for/0
https://cours-symfony.argetis.com/twig/conditions-for/5

Exécuté si la collection est vide

Itération – variable de contexte **`loop`**

- **`loop`** est une variable auto-déclarée au sein d'une boucle
- Si boucles imbriquées, le « **`loop`** » est lié à la boucle parente la plus proche

```
{% set vars = [1, 2, 'test'] %}
{% for var in vars %}
   <li>{{ loop.index }} loop.index : affiche l'indice courant de l'itération (de 1 à 3)</li>
   <li>{{ loop.index0 }} loop.index0 : affiche l'indice courant de l'itération (de 0 à 2)</li>
   <li>{{ loop.revindex }} loop.revindex : affiche le nombre d'itérations restantes avant la fin (de 3 à 1)</li>
   <li>{{ loop.revindex0 }} loop.revindex0 : affiche le nombre d'itérations restantes avant la fin (de 2 à 0)</li>
   <li>{{ loop.first }} loop.first : booléen vrai seulement à la première itération</li>
   <li>{{ loop.last }} loop.last : booléen vrai seulement à la dernière itération</li>
   <li>{{ loop.length }} loop.length : nombre total d'itérations (3)</li>
   <li>loop.parent : objet correspondant au contexte parent</li>
   <li>---</li>
{% endfor %}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/loop.html.twig

Tester : https://cours-symfony.argetis.com/twig/loop

Les opérateurs

```
{{ 2+3 }}{# 5 #}<br>
{{ 5-2 }}{# 3 #}<br>
{{ 2*3 }}{# 6 #}<br>
{{ 3/2 }}{# 1.5 #}<br>
{{ 2**3 }}{# 8 #}<br>
{{ 20//3 }}{# 6 : résultat de la division de 20 par 3 arrondi à l'entier inférieur #}<br>
{{ 3%2 }}{# 1 (modulo) #}<br><br>
```

- Opérateurs mathématiques : **`+ - * / ** // %`**
- Opérateurs logiques :
  `and or not (...)`
  `b-and b-xor b-or`

```
{% if(1 and 1 or (0 and not(0 or 1))) %}Cette expression est vraie{% endif %}<br><br>

{{ 12 b-or 9 }}<br>{# 1100 b-or 1001 ==> 1101 ==> 13 #}
{{ 12 b-xor 9 }}<br>{# 1100 b-xor 1001 ==> 0101 ==> 5 #}
{{ 12 b-and 9 }}<br><br>{# 1100 b-xor 1001 ==> 1000 ==> 8 #}
```

- Opérateurs de comparaison : **`== != < > <= >=`**
- Autres opérateurs de comparaison :
  `starts with`
  `ends with`
  `matches`

```
{% set url = 'https://www.google.fr' %}
{% set phone_number = '05.49.49.49.49' %}
{% if url starts with 'https://' %}Oui l'URL commence par https://{% endif %}<br>
{% if url ends with '.fr' %}Oui l'URL termine par .fr{% endif %}<br>
{% if phone_number matches '/^[\\d\\.]+$/' %}Oui le numéro de téléphone est valide{% endif %}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/operators.html.twig

Tester : https://cours-symfony.argetis.com/twig/operators

Les opérateurs

- Opérateur de concaténation : **~**
- Opérateur d'interpolation : **#{ ... }**
- Opérateur d'appartenance : **in** / **not in**
- Autres opérateurs : **is** / **is not**

```
{% set first_name = 'John' %}
{% set last_name = 'Doe' %}
{{ first_name ~ ' ' ~ last_name }}<br>
{{ "Hello #{ first_name }" }}<br><br>
```

```
{% set values = [1, 2, 4, 5] %}
{% if 3 not in values %}Elément non trouvé{% endif %}<br><br>

{% set login = 'my_login' %}
{% set password = 'log' %}
{% if password in login %}
    Ne pas utiliser un mot de passe contenu dans le login !<br><br>
{% endif %}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/operators.html.twig

Tester : https://cours-symfony.argetis.com/twig/operators

```
{% set number = 11 %}
{% if number is odd %}Nombre impair{% endif %}<br>
{% if number is not divisible by(3) %}Nombre non divisible par 3{% endif %}<br><br>
```

Les opérateurs

- Equivalent à la fonction **`range()`** : « **`..`** »

```
{% set number = 5 %}
{% if number in 1..10 %}Oui 5 est compris entre 1 et 10{% endif %}<br>
{% for letter in 'a'..'z' %}{{ letter }}{% endfor %}<br><br>
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/operators.html.twig

Tester : https://cours-symfony.argetis.com/twig/operators

- ? / ?: / ??

```
{% set article = {"published": true, 'title': 'Harry Potter', 'author': null} %}
{{ article.published ? 'yes' : 'no' }}<br>
{{ article.author ?: 'Anonyme' }}<br>{# affiche article.author si renseigné, sinon "Anonyme" #}
{{ article.title ? article.title }}<br>{# affiche article.title si renseigné, sinon rien #}
{{ article.price ?? 19 }}<br><br>{# affiche article.price si défini et renseigné, sinon 0 #}
```

Les tests standards dans Twig

```
{% set number = 10 %}
{% set firstname = 'John' %}
{% set tab = [1, 2, 3] %}
{% if number is constant('\App\\Controller\\Twig\\DefaultController::TOTO') %}...{% endif %}<br>
{% if lastname is defined %}{{ lastname }}{% endif %}<br>
{% if firstname|length is divisible by(3) %}Divisible{% endif %}<br>
{% if firstname is empty %}Empty{% endif %}<br>
{% if number is even %}Pair{% endif %}<br>
{% if number is odd %}Impair{% endif %}<br>
{% if tab is iterable %}tab est itérable{% endif %}<br>
{% if firstname is null %}null{% endif %}<br>
{% if firstname is same as ('John') %}Egalité en valeur et en type (=== PHP){% endif %}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/tests.html.twig

Tester : https://cours-symfony.argetis.com/twig/tests

Pour aller plus loin

Contrôle des espaces

- **`spaceless`** : suppression des espaces entre les tags HTML, mais pas les espaces au sein des balises HTML ou au sein d'un texte brut

```
{% set firstname = 'John' %}
{% spaceless %}
    <p>
        Hello <strong>{{ firstname }}</strong>!
    </p>
{% endspaceless %}
```

```
<p>
        Hello <strong>John</strong>!
    </p>
```

```
{% spaceless %}
    <ul>
        {% for i in 1..3 %}
            <li>{{ i }}</li>
        {% endfor %}
    </ul>
{% endspaceless %}
```

```
<ul><li>1</li><li>2</li><li>3</li></ul>
```

Pour aller plus loin

Contrôle des espaces

```
{% set firstname = 'John' %}
<p>
    Hello <strong>    {{- firstname }}    </strong>!
</p>
```

```
<p>
    Hello <strong>John    </strong>!
</p>
```

```
<ul>
    {%- for i in 1..3 %}
        <li>{{ i }}</li>
    {% endfor %}
</ul>
```

```
<ul>        <li>1</li>
            <li>2</li>
            <li>3</li>
    </ul>
```

```
<ul>
    {%- for i in 1..3 -%}
        <li>{{ i }}</li>
    {% endfor %}
</ul>
```

```
<ul><li>1</li>
    <li>2</li>
    <li>3</li>
    </ul>
```

```
<ul>
    {%- for i in 1..3 -%}
        <li>{{ i }}</li>
    {%- endfor %}
    </ul>
```

```
<ul><li>1</li><li>2</li><li>3</li>    </ul>
```

```
<ul>
    {%- for i in 1..3 -%}
        <li>{{ i }}</li>
    {%- endfor -%}
    </ul>
```

```
<ul><li>1</li><li>2</li><li>3</li></ul>
```

Inclusion de template

- Inclusion simple : `{{ include('menu.html.twig') }}` `{% include 'menu.html.twig' %}`
- Inclusion sans transmission du contexte au template appelé :

```
{{ include('menu.html.twig', with_context = false) }}
```

```
{% include 'menu.html.twig' only %}
```

- Inclusion avec passage de variables en plus :

```
{{ include('menu.html.twig', {'firstname': 'John'}) }}
```

```
{% include 'menu.html.twig' with {'firstname': 'John'} %}
```

- Possibilité de mixer les 2 :

```
{% include 'menu.html.twig' with {'firstname': 'John'} only %}
```

```
{{ include('menu.html.twig', {'firstname': 'John'}, with_context = false) }}
```

Héritage de template

- Problématique : la plupart des pages d’un site ont la même forme :
  - Header / Menu / Contenu / Footer / …
- Il serait possible de définir chacun de nos templates en incluant chacun de ces templates à chaque fois, mais c’est fastidieux

- Solution : héritage de templates. Principe :
  - Un template parent qui définit des zones
  - Un template fils qui fait références à ces zones en les définissant / complétant / remplaçant

Héritage de templates – template parent

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
    </head>
    <body class="{% block body_class %}{% endblock %}">
        {% block h1 %}<h1>Mon site</h1>{% endblock %}
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/inheritance/parent.html.twig

Héritage de templates – template fils

```twig
{% extends 'twig/inheritance/parent.html.twig' %} {# Cette instruction DOIT être la première du template fils #}
{# Un template ne peut hériter que d'un seul parent #}
{# Il n'y a pas de contrainte de niveaux : un template peut hériter d'un parent qui hérite d'un autre etc. #}

{# Il est possible de compléter un bloc déjà défini par le parent.
   Si on ne fait pas appel à parent(), le contenu est remplacé #}
{% block title %}Ma page - {{ parent() }}{% endblock %}

{# Il est possible vider un bloc défini par le parent #}
{% block h1 %}{% endblock %}

{# Il est possible d'utiliser une notation raccourcie, sans le endblock #}
{% block body_class 'my_class' %}

{% block body %}
    {# Il est possible d'inclure un template #}
    {{ include('twig/inheritance/menu.html.twig') }}

    {# Il est possible d'appeler via la fonction block() le contenu d'un bloc existant #}
    <h2>{{ block('title') }}</h2>

    Contenu de ma page

    {# Il est possible de déclarer de nouveaux blocs qui pourront à leur tour être redéfinis par un template fils #}
    {% block body_bottom %}Pied de page{% endblock %}
{% endblock %}

{# Il n'est pas obligatoire de redéfinir tous les blocs du parent #}

{# Il est interdit d'écrire du contenu hors des balises block ==> erreur sinon #}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/inheritance/child.html.twig

Héritage de templates – testons !

```php
public function inheritanceAction()
{
    // Dans le contrôleur, on appelle le template fils
    return $this->render('twig/inheritance/child.html.twig');
}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php

Tester : https://cours-symfony.argetis.com/twig/inheritance

Résultat (code source généré)

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ma page - Welcome!</title>
</head>
<body class="my_class">
<ul>
    <li><a href="#">Item 1</a></li>
    <li><a href="#">Item 2</a></li>
    <li><a href="#">Item 3</a></li>
</ul>

<h2>Ma page - Welcome!</h2>

Contenu de ma page

Pied de page
<!-- Le profiler s'affichera ici, s'il est activé -->
</body>
</html>
```

Macros

- Objectif : factoriser le code pour des éléments récurrents (typiquement : balise input d'un formulaire)
- Équivalent aux fonctions dans un langage de programmation classique
- Les macros doivent être importées pour pouvoir être utilisées

Macro – définition et utilisation

```twig
{% macro input(name, value, type='text', size=20) %}
    <input type="{{ type }}"
           name="{{ name }}"
           value="{{ value|escape }}"
           size="{{ size }}" />
{% endmacro %}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/macro/declaration.html.twig

```twig
{% import "twig/macro/declaration.html.twig" as utils %}
<form>
    {{ utils.input('firstname') }}
    {{ utils.input('lastname') }}
    {{ utils.input('login') }}
    {{ utils.input('password', null, 'password') }}
</form>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/macro/usage.html.twig

- Pour une utilisation dans le fichier qui définit la macro :

```twig
{% macro input(name, value, type='text', size=20) %}
    <input type="{{ type }}"
           name="{{ name }}"
           value="{{ value|escape }}"
           size="{{ size }}" />
{% endmacro %}
{% import _self as utils %}
<form>
    {{ utils.input('firstname') }}
    {{ utils.input('lastname') }}
    {{ utils.input('login') }}
    {{ utils.input('password', null, 'password') }}
</form>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/macro/declaration_and_usage.html.twig

Tester : https://cours-symfony.argetis.com/twig/macro
https://cours-symfony.argetis.com/twig/macro-bis

Intégration de Twig dans Symfony

- Variables globales :
  - Définies dans **`config/packages/twig.yaml`**
  - Utilisables comme n'importe quelle variable dans tous les templates Twig
  - Définition :

```yaml
twig:
    # ...
    globals:
        my_global_var: 12345
```

https://github.com/sgautier/cours_symfony/blob/master/config/packages/twig.yaml
https://github.com/sgautier/cours_symfony/blob/master/templates/twig/global_variable.html.twig

Tester : https://cours-symfony.argetis.com/twig/global-variable

- Objet global auto-déclaré : app
  - **`{{ app.request }}`**
  - **`{{ app.session }}`**
  - **`{{ app.user }}`**
  - **`{{ app.debug }}`** ➔ true si le mode debug est activé, false sinon
  - **`{{ app.environment }}`** ➔ environnement courant : dev, prod, …

Intégration de Twig dans Symfony

- Génération d'urls en utilisant le nom d'une route :
  - Relatives :
  - Absolues :

```
<a href="{{ path('hello') }}">Lien vers page Hello World</a>
<a href="{{ url('twig_conditions') }}">Lien vers page des conditions Twig</a>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/urls.html.twig

- Remarque : générer une URL dans un contrôleur :

```
$this->get('router')->generate('nom-de-ma-route-6', ['year' => 2018, 'month' => '01', 'filename' => 'test']);
$this->generateUrl('nom-de-ma-route-6', ['year' => 2018, 'month' => '01', 'filename' => 'test']);
$this->generateUrl('nom-de-ma-route-6', ['year' => 2018, 'month' => '01', 'filename' => 'test'], UrlGeneratorInterface::ABSOLUTE_URL);
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php

- Génération d'urls pour les assets (les assets doivent être présents dans le répertoire « **`public`** » de Symfony) :
  - Nécessite l'installation du recipe associé : **`$ composer req asset`**

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

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/urls.html.twig

Tester : https://cours-symfony.argetis.com/twig/urls

Intégration de Twig dans Symfony

- Emplacement des templates : répertoire « **`templates`** »

- Pour hiérarchiser ses templates, on peut par exemple (conseillé) créer un répertoire par contrôleur + nommer son template en fonction de l'action à laquelle il est lié
  - Rien d'obligatoire ➔ respecter les contraintes du projet quand elles existent pour faciliter le travail des développeurs

- Invoquer un template depuis une action d'un contrôleur :

```php
public function totoAction()
{
    return $this->render('toto.html.twig');
}
```

```php
public function totoAction()
{
    return $this->render('toto.html.twig', ['key1' => 'val1', /*...*/]);
}
```

Paramétrage des assets – quelques bases

- /!\ Ce cours ne traite pas de la création d'un frontend, il y aurait de quoi y consacrer plusieurs heures/jours

- Problématique souvent constatée : comment forcer le rechargement des assets côté client lorsqu'ils ont été modifiés par le développeur ? ➔ Symfony propose, via le composant « asset », un mécanisme de versionning qui permet la génération des URLs et la gestion de leurs versions :
  - **`$ composer require symfony/asset`**
  - Si on change de version, on est certain que le navigateur va tout re-télécharger ➔ pas besoin d'appeler tous vos utilisateurs pour faire un « Ctrl+F5 » ☺
  - Fichier framework.yaml pour gérer la version

```yaml
framework:
    # ...
    assets:
        version: "v1"
```

```
{{ asset('images/logo.png') }}
```

```
/images/logo.png?v1
```

https://github.com/sgautier/cours_symfony/blob/master/config/packages/framework.yaml

Tester : https://cours-symfony.argetis.com/twig/urls

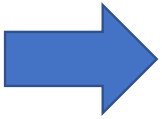
Paramétrage des assets – quelques bases

- Définition d'une URL de base pour les assets (exemple : CDN) :
  - Possibilité de définir plusieurs URLs ➔ Symfony en prend une au hasard à chaque appel de la fonction **`asset()`**

```
framework:
    # ...
    assets:
        base_urls:
            - '%env(CDN_URL_1)%'
            - '%env(CDN_URL_2)%'
```

https://github.com/sgautier/cours_symfony/blob/master/config/packages/framework.yaml

```
{{ asset('images/logo.png') }}
```



```
https://localhost:8000/images/logo.png

OU

https://127.0.0.1:8000/images/logo.png
```

```
CDN_URL_1='https://localhost:8000'
CDN_URL_2='https://127.0.0.1:8000'
```

https://github.com/sgautier/cours_symfony/blob/master/.env

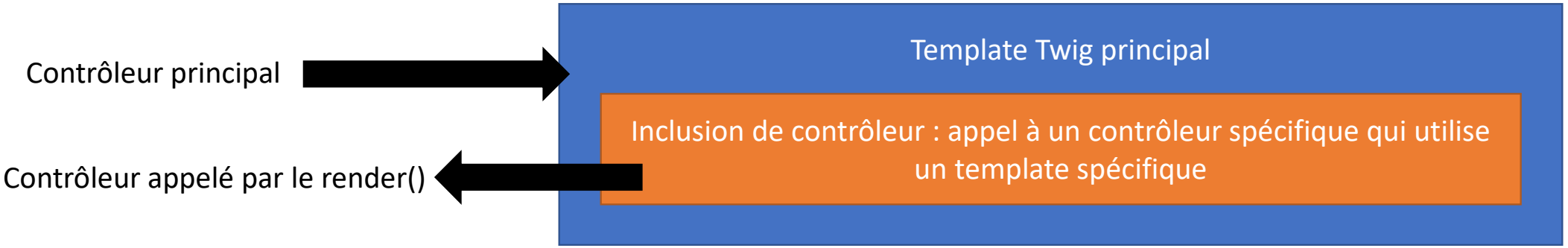
Tester : https://cours-symfony.argetis.com/twig/urls

- La suite : voir documentation pour ceux qui veulent aller plus loin sur les concepts frontend :
  - https://symfony.com/doc/current/components/asset.html
  - https://symfony.com/doc/current/frontend.html

Inclusion de contrôleur

- Exemple d'utilisation : afficher la liste des X derniers articles publiés
  - L'inclusion simple ne suffit pas car ce bloc a besoin par exemple de se connecter à la base de données ➔ pour récupérer le contenu des articles
  - Si ce bloc est inclus dans un layout général, quid du contexte nécessaire ? Il est en effet exclus que chaque action de notre application fasse en plus de ses traitements le calcul des X derniers articles publiés et retourne le résultat à twig !

- Contraintes :
  - Attention aux performances car une sous-requête Symfony est lancée



Inclusion de contrôleur – Exemple

- 1. Le contrôleur principal invoque un template :

```php
public function renderAction()
{
    return $this->render('twig/use_render.html.twig');
}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/DefaultController.php

- 2. Dans twig, faire appel à l'action du contrôleur souhaité :

```twig
{{ render(controller(
    'App\\Controller\\Twig\\RenderExampleController::bestSalesAction',
    { 'nbProducts': 3 }
)) }}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/use_render.html.twig

- 3. Le 2ème contrôleur est appelé, et charge son propre template :

```php
public function bestSalesAction($nbProducts)
{
    // Evidemment, dans la vraie vie, les produits sont chargés depuis la base de données par exemple
    $products = [];
    for($i=1 ; $i<=$nbProducts ; $i++) {
        $products[] = ['name' => 'Product ' . $i, 'price' => 25+$i,];
    }
    return $this->render('twig/best_sales.html.twig', ['products' => $products]);
}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Controller/Twig/RenderExampleController.php

```twig
{% if(products|length > 0) %}
    <ul>
    {% for product in products %}
        <li>{{ product.name }} ({{ product.price }} €)</li>
    {% endfor %}
    </ul>
{% endif %}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/best_sales.html.twig

Tester : https://cours-symfony.argetis.com/twig/render

Fonctions pour les formulaires

- Cf. détails dans le chapitre dédié aux formulaires
- Nécessite l'installation d'un recipe : **`$ composer req form`**
- Doc : symfony.com/doc/current/reference/twig_reference.html

```
{% form_theme form 'form/fields.html.twig' %}

{{ form_start(form) }}
{{ form_errors(form) }}
<div>
    {{ form_label(form.firstname) }}
    {{ form_errors(form.firstname) }}
    {{ form_widget(form.firstname) }}

    {{ form_row(form.lastname) }}
    {{ form_widget(form.save) }}

    {{ form_rest(form) }}
</div>
{{ form_end(form) }}
```

Fonctions pour la sécurité

- Cf. détails dans le chapitre dédié à la sécurité
- Nécessite l'installation d'un recipe : **`$ composer req security`**
- Doc : symfony.com/doc/current/reference/twig_reference.html

```
{% if is_granted('ROLE_ADMIN') %}...{% endif %}
<a href="{{ logout_path() }}">Se déconnecter</a>
<a href="{{ logout_url() }}">Se déconnecter</a>
```

Fonctions pour la traduction

- Cf. détails dans le chapitre dédié aux traductions
- Doc : symfony.com/doc/current/reference/twig_reference.html

```
{{ message|trans }}
{{ message|transchoice(5) }}
{{ message|trans({'%name%': 'John'}, "app") }}
{{ message|transchoice(5, {'%name%': 'John'}, 'app') }}
```

Ligne de commande

- Vérifier la syntaxe des templates :

  `php bin/console lint:twig templates`

- Afficher la liste des fonctions, filtres, tests et variables globales :

  `php bin/console debug:twig`

Gestion des pages d’erreurs

- Symfony génère pour chaque type d’erreur une exception
  - Exemple : pour une erreur 404 : NotFoundHttpException
- Les pages d’erreur sont affichées grâce à un contrôleur inclus dans le TwigBundle : ExceptionController
- Sélection du template en fonction de l’erreur (le premier trouvé est pris) :
  - error + status code + format + twig ➔ error404.html.twig, error500.xml.twig
  - error + format + twig ➔ error.html.twig, error.json.twig, error.xml.twig
  - error.html.twig

Personnaliser les pages d'erreur

- Il faut surcharger les templates par défaut définis dans le TwigBundle
- Mettre les templates de surcharge dans le répertoire suivant :
  `templates/bundles/[NOM-DU-BUNDLE]/[CHEMIN-VERS-TEMPLATE]`

➔ Pour Twig : **`templates/bundles/TwigBundle/`**

- Exemple pour les erreur 404 :
  `templates/bundles/TwigBundle/Exception/error404.html.twig`

- Voir doc : https://symfony.com/doc/current/controller/error_pages.html

Personnaliser les pages d'erreur

- Exemple : **`templates/bundles/TwigBundle/Exception/error404.html.twig`**
- Pour tester (en dev uniquement !) : https://127.0.0.1:8000/_error/404

```twig
<!DOCTYPE html>
<html>
<head>
    <meta charset="{{ _charset }}" />
    <title>An Error Occurred: {{ status_text }}</title>
</head>
<body>
<h1>Zut ! Une erreur s'est produite</h1>
<h2>Le serveur a retourné une erreur "{{ status_code }} {{ status_text }}".</h2>

<div>
    Toutes nos excuses pour la gêne occasionnée
</div>
</body>
</html>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/bundles/TwigBundle/Exception/error404.html.twig

Tester : https://cours-symfony.argetis.com/_error/404
https://cours-symfony.argetis.com/_error/401

Créer une extension pour Twig

- Objectif : créer ses propres filtres / fonctions / opérateurs / …
- Recipe à installer : **`$ composer require twig/extensions`**
- Rappels :
  - Une fonction est un appelable PHP qui génère du contenu pour un template ➔ permet notamment d'éviter de complexifier les templates avec une logique métier lourde
  - Un filtre permet de transformer une information avant son affichage (ex : passage en majuscules)
- Pour créer une extension : créer une classe qui étend de **`\Twig\Extension\AbstractExtension`**
  - Pas besoin de déclarer la classe comme service si autowiring et autoconfigure activés (cf. chapitre sur les services)

```php
<?php
namespace App;
use Twig\Extension\AbstractExtension;

class MyTwigExtension extends AbstractExtension
{
    // Ceci est une extension Twig !
}
```

Créer une extension pour Twig

Créer / utiliser sa propre fonction

```php
<?php
namespace App\Twig\Extension;
use Twig\Extension\AbstractExtension;
use Twig\TwigFunction;

class Gravatar extends AbstractExtension
{
    public function getFunctions(): array
    {
        // Il est possible de définir plusieurs fonction Twig
        return [
            new TwigFunction('gravatar', [$this, 'getGravatarUri']),
        ];
    }

    public function getGravatarUri($email, $size='80', $rating='g')
    {
        $url = 'https://gravatar.com/avatar/%s?size=%u&rating=%s';
        $hash = md5(strtolower(trim($email)));
        return sprintf($url, $hash, (int) $size, $rating);
    }
}
```

```twig
<img src="{{ gravatar('moi@example.com', 64) }}" alt="..."/>
<img src="{{ gravatar('lui@example.com', 120, 'r') }}" alt="..."/>
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/extension.html.twig

https://github.com/sgautier/cours_symfony/blob/master/src/Twig/Extension/Gravatar.php

Tester : https://cours-symfony.argetis.com/twig/extension

Créer une extension pour Twig

Créer / utiliser son propre filtre

```php
<?php
namespace App\Twig\Extension;
use Twig\Extension\AbstractExtension;
use Twig\TwigFilter;

class Truncate extends AbstractExtension
{
    public function getFilters()
    {
        return [
            new TwigFilter('truncate', [$this, 'truncate']),
        ];
    }

    public function truncate($text, $max = 30)
    {
        if (mb_strlen($text) > $max) {
            $text = mb_substr($text, 0, $max);
            $text = mb_substr($text, 0, mb_strrpos($text,' '));
            $text.= '...';
        }
        return $text;
    }
}
```

https://github.com/sgautier/cours_symfony/blob/master/src/Twig/Extension/Truncate.php

```twig
{% set text = "Ceci est un texte assez long avec lequel nous allons tester notre filtre" %}
{{ text|truncate }}
{{ text|truncate(15) }}
```

https://github.com/sgautier/cours_symfony/blob/master/templates/twig/extension.html.twig

Tester : https://cours-symfony.argetis.com/twig/extension

TP – Travailler avec Twig 1/3

- Créer un layout général avec la structure HTML de base (reprendre le fichier **`base.html.twig`**)
  - Définir un bloc de title avec un titre de page par défaut
  - Définir un bloc de titre h1 avec un titre par défaut
  - Définir un bloc de menu avec 2 liens :
    - « Accueil » : doit pointer à la racine de notre site Symfony (pas la racine du blog, mais bien du site entier donc URL « / » ➔ créer une action dans un nouveau contrôleur qui affiche le template de layout général)
    - « Blog » qui est l'accueil de notre blog. L'accueil du blog est la liste des articles ➔ URL : « /blog/list »
  - Définir un bloc de menu de gauche et y faire une inclusion de contrôleur pour afficher les X (paramètre de l'action) derniers articles du blog (titre de l'article seulement + lien vers l'article). Créer un template **`last_articles.html.twig`** pour ce bloc.
  - Afficher dans ce template les messages flash s'il y en a
  - Définir enfin un bloc de contenu (avec pour contenu ce qui sera affiché pour la page d'accueil)
- Créer le template principal de notre blog (hérite du layout général)
  - Compléter le title en y ajoutant une mention à notre blog
  - Modifier le titre h1 : « Bienvenue sur le blog ! »
  - Ajouter un lien « Ajouter un article » au menu
  - Définir comme contenu :
    - Un nouveau bloc de contenu vide pour l'instant (qui sera complété par les templates de contenu)
    - En dessous, une phrase d'accroche de notre blog qui sera toujours affichée au sein du blog

TP – Travailler avec Twig 2/3

- Créer tous les templates de contenu, les appeler dans les actions du contrôleur :
  - list.html.twig : Utile pour la liste des articles (afficher pour chaque article maxi X caractères du contenu de l'article + lien vers la page de détail de l'article) ➔ X : variable globale Twig
  - view.html.twig : consultation d'un article + bouton suppression / modification
  - add.html.twig, edit.html.twig : ajout / modification d'un article. Ces 2 templates doivent inclure un template form.html.twig qui sera le formulaire (mettre dans form.html.twig juste un texte pour l'instant, la notion de formulaire sera vue plus tard)
- Redéfinir le h1 pour les templates :
  - view.html.twig, add.html.twig, edit.html.twig
- Définir un contenu pour le bloc défini dans le template du blog pour les templates :
  - list.html.twig, view.html.twig, add.html.twig, edit.html.twig
- Pour afficher des données, vos actions de contrôleurs peuvent retourner des données factices dans des tableaux / des objets PHP stdClass

TP bonus – Travailler avec Twig 3/3

Faire beau sans se fatiguer !

Rappel : nous ne faisons pas un cours sur les concepts frontend

- Pour avoir un site joli sans y passer trop de temps, intégrer le framework Bootstrap à votre site Symfony !
- Invoquer les CSS / JS depuis les CDN :
  - https://www.bootstrapcdn.com/
  - https://developers.google.com/speed/libraries/#jquery
- Créer des blocs js et css dans le layout et invoquer les fichiers nécessaires (cf. exemple ci-dessous)
- Trouver un exemple de HTML ici : http://getbootstrap.com/getting-started/#examples

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Mon site</title>
    {% block css %}
        <link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    {% endblock %}
</head>
<body class="{% block body_class %}{% endblock %}">
{% block h1 %}<h1>Mon site</h1>{% endblock %}
{% block body %}{% endblock %}
{% block javascripts %}{% endblock %}
</body>

{% block js %}
    <script src="//ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
{% endblock %}
</html>
```