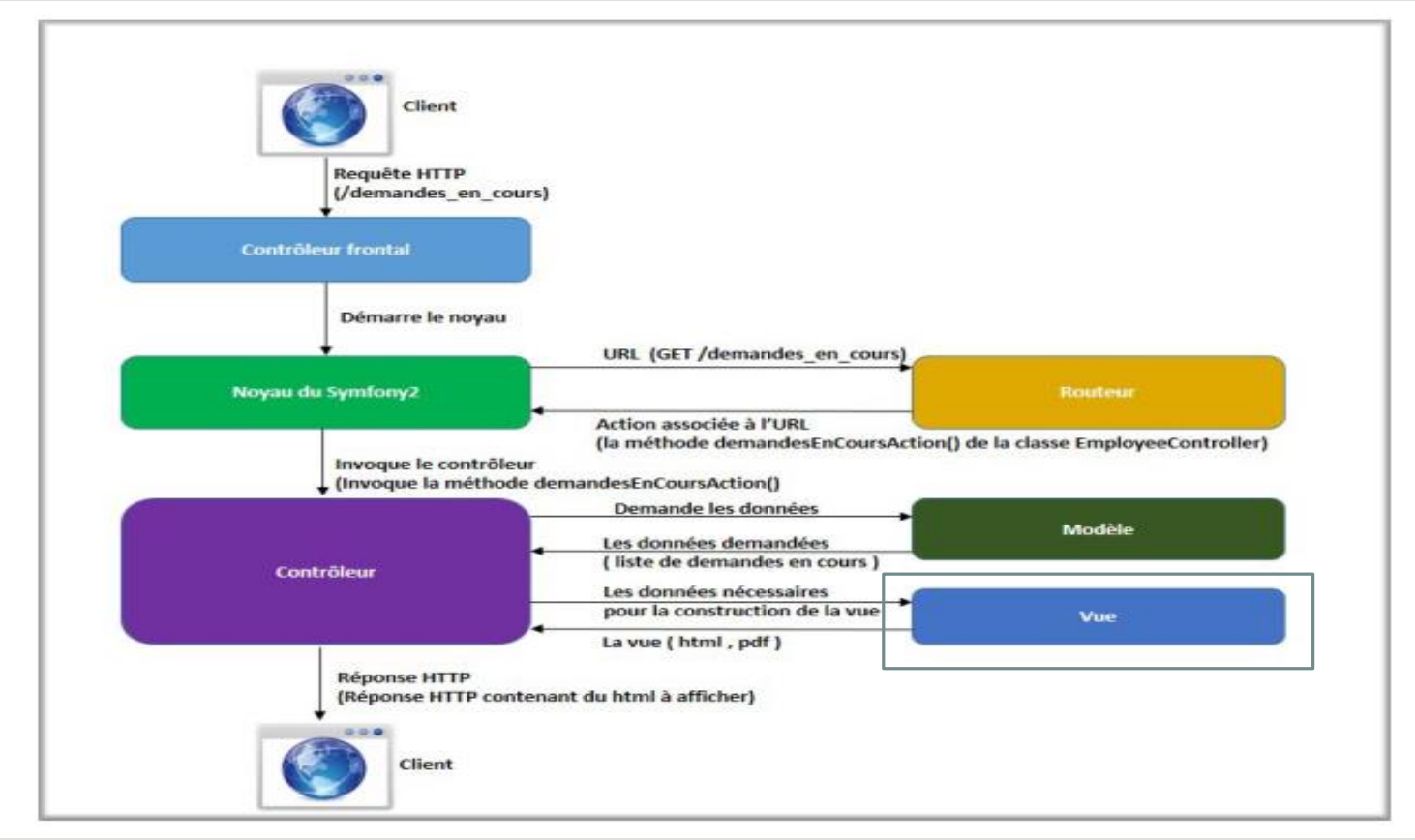
Symfony2
Les TWIGS

AYMEN SELLAOUTI

Introduction (1)



Introduction (2)

- Moteur de Template PHP.
- Développé par l'équipe de Sensio Labs,
- Directement intégré dans Symfony (pas besoin de l’installer).

Introduction (3)

- Permet de gérer de l'héritage entre Templates et layout
- L'objectif principal de Twig est de permettre aux développeurs de séparer la couche de présentation (Vue) dans des Templates dédiés, afin de favoriser la maintenabilité du code.
- Idéal pour les graphistes qui ne connaissent pas forcément le langage PHP et qui s'accommoderont parfaitement des instructions natives du moteur, relativement simples à maîtriser.
- Il y a quelques fonctionnalités en plus, comme l'héritage de templates.
- Il sécurise vos variables automatiquement (htmlentities(), addslashes())

Syntaxe de bases de Twig

{{ maVar }} : Les doubles accolades (équivalent de l'echo dans php ou du <%= %> pour les jsp) permettent d'imprimer une valeur, le résultat d'une fonction...

{% code %}: Les accolades pourcentage permettent d'exécuter une fonction, définir un bloc...

{# Les commentaires #}: Les commentaires.

Comment afficher une variable dans TWIGS (1)

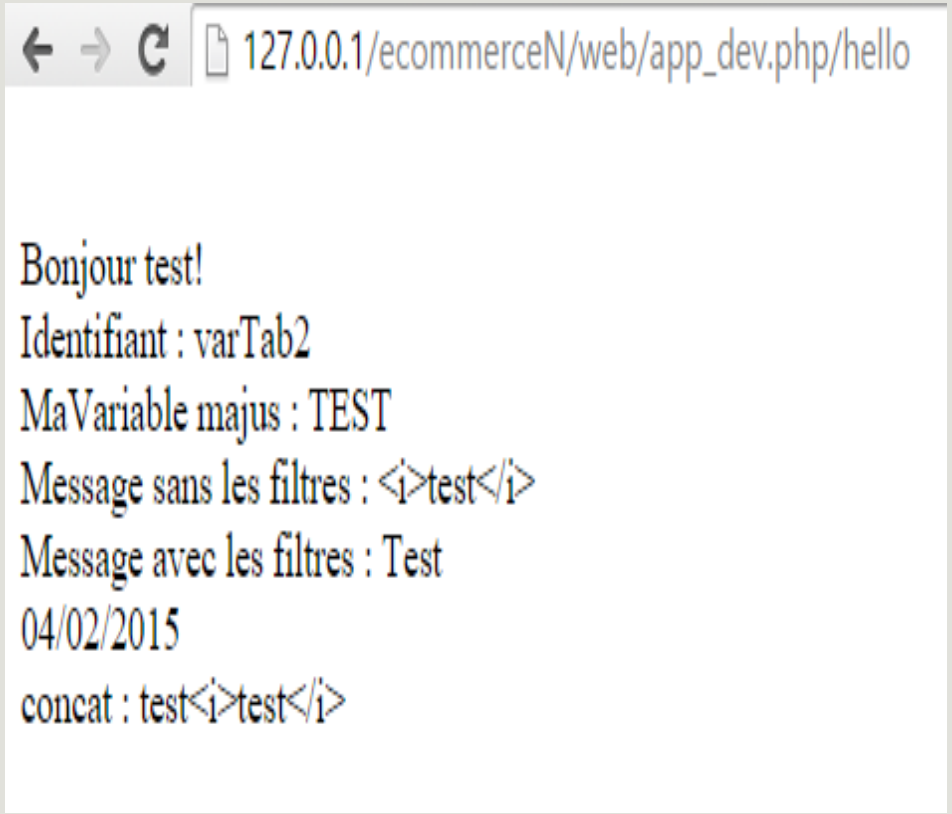
Fonctionnalité	Exemple Twig	Équivalent PHP
Afficher une variable	Variable : {{ MaVariable }}	Pseudo : <?php echo $MaVariable; ?>
Afficher le contenu d'une case d'un tableau	Identifiant : {{ tab['id'] }}	Identifiant : <?php echo $tab['id']; ?>
Afficher l'attribut d'un objet, dont le getter respecte la convention $objet->getAttribut()	Identifiant : {{ user.id }}	Identifiant : <?php echo $user->getId(); ?>
Afficher une variable en lui appliquant un filtre. Ici, « upper » met tout en majuscules :	MaVariable majus : {{ MaVariable|upper }}	MaVariable majus: <?php echo strtoupper($MaVariable); ?>

Comment afficher une variable dans TWIG (2)

Fonctionnalité	Exemple Twig	Équivalent PHP
Afficher une variable en combinant les filtres. « striptags » supprime les balises HTML. « title » met la première lettre de chaque mot en majuscule.	Message : {{ news.texte|striptags|title }}	Message : <?php echo ucwords(strip_tags($news-> getTexte())); ?>
Utiliser un filtre avec des arguments. Attention, il faut que date soit un objet de type Datetime ici.	Date : {{ date|date('d/m/Y') }}	Date : <?php echo $date->format('d/m/Y'); ?>
Concaténer	Identité : {{ nom ~ " " ~ prenom }}	Identité : <?php echo $nom.' '. $prenom; ?>

Comment afficher une variable dans les TWIGS : Exemple :

```
{# set permet de déclarer des varaibles #}
{% set MaVariable = 'test' %}
Bonjour {{ MaVariable }}!
<br>
{# On affiche l'indexe du tableau tab envoyé par le controlleur#}
Identifiant : {{ tab['1'] }}
<br>
{# Application de quelques filtres #}
MaVariable majus : {{ MaVariable|upper }}
<br>
{% set texte = '<i>test</i>' %}
Message sans les filtres : {{ texte }}
<br>
Message avec les filtres : {{ texte|striptags|title }}
<br>
{# now nous donne la date système #}
{{ "now"|date("m/d/Y") }}
<br>
{# Concaténer #}
concat : {{ MaVariable ~""~ texte }}
```



Twig et l'affichage d'un attribut

Lorsqu'on veut accéder à un attribut d'un objet avec twig on a le fonctionnement suivant pour l'exemple {{objet.attribut}} :

- ✓Vérification si l'objet est un tableau si oui vérifier que nous avons un index valide dans ce cas elle affiche le contenu de objet['attribut']
- ✓Sinon, si objet est un objet, elle vérifie si attribut est un attribut valide public. Si c'est le cas, elle affiche objet->attribut.
- ✓Sinon, si objet est un objet, elle vérifie si attribut() est une méthode valide publique. Si c'est le cas, elle affiche objet->attribut().
- ✓Sinon, si objet est un objet, elle vérifie si getAttribut() est une méthode valide. Si c'est le cas, elle affiche objet->getAttribut().
- ✓Sinon, et si objet est un objet, elle vérifie si isAttribut() est une méthode valide. Si c'est le cas, elle affiche objet->isAttribut().
- ✓Sinon, elle n'affiche rien et retourne null.

Les filtres

Filtre	Description	Exemple Twig
Upper	Met toutes les lettres en majuscules.	{{ var|upper }}
Striptags	Supprime toutes les balises XML.	{{ var|striptags }}
Date	Formate la date selon le format donné en argument. La variable en entrée doit être une instance de Datetime.	{{ date|date('d/m/Y') }} Date d'aujourd'hui : {{ "now"|date('d/m/Y') }}
Format	Insère des variables dans un texte, équivalent à printf.	{{ "Il y a %s pommes et %s poires"|format(153, nb_poires) }}
Length	Retourne le nombre d'éléments du tableau, ou le nombre de caractères d'une chaîne.	Longueur de la variable : {{ texte|length }} Nombre d'éléments du tableau : {{ tableau|length }}

Tests

is defined l'équivalent de isset en php

Rôle vérifie l'existence d'une variable

Exemple:{% if MaVariable is defined %} J'existe {% endif%}

even

Rôle vérifie si la variable est pair

Exemple : {% if MaVariable is even %}Pair{% endif%}

odd

Rôle vérifie si la variable est impair

Exemple : {% if MaVariable is odd%}Impair{% endif%}

Structures de contrôle

Les structures que ce soit séquentielles ou itératives sont souvent très proches du langage naturel.

Elles sont introduites entre {% %}

Généralement elle se termine par une expression de fin de block

Structure conditionnelle

IF

Syntaxe :

{% if cnd %}

Block de traitement

{%endif%}

Exemple

{% if employee.salaire < 250 %}

ce salaire est inférieur au smig

{%endif%}

Structure conditionnelle

IF else elseif

Syntaxe :

{% if cnd %}

block traitement

{% elseif cnd2 %}

block traitement

{% else %}

block traitement

{% endif %}

Exemple

{% if maison.tempertature <0%}

Très Froid

{% elseif maison.tempertature <10 %}

Froid

{% else %}

Bonne température

{% endif %}

Structure itérative

```
{% for valeur in valeurs %}
    block à répéter
{% else %}
    Traitements à faire si il n'y
    a aucune valeur
{% endfor %}
```

Exemple

```
La formation de l'équipe A est : <br>
<ol>
{% for joueur in joueurs %}
        <li> {{player.nom}} </li>
  {% else %}
      La liste n'a pas encore été renseignée
  {% endfor %}
</ol>
```

Structure itérative

La boucle for définit une variable loop ayant les attributs suivants :

Variable	Description
{{ loop.index }}	Le numéro de l'itération courante (en commençant par 1).
{{ loop.index0 }}	Le numéro de l'itération courante (en commençant par 0).
{{ loop.revindex }}	Le nombre d'itérations restantes avant la fin de la boucle (en finissant par 1).
{{ loop.revindex0 }}	Le nombre d'itérations restantes avant la fin de la boucle (en finissant par 0).
{{ loop.first }}	true si c'est la première itération, false sinon.
{{ loop.last }}	true si c'est la dernière itération, false sinon.
{{ loop.length }}	Le nombre total d'itérations dans la boucle.

Accès aux Template

Les Templates doivent se trouver dans l'un des deux emplacements suivants :

- app/ressources/views
- /ressources/views d'un bundle

Afin d'accéder aux Template, une convention de nommage est définit :

NomBundle:DossierFichier:pagetwig

Exemples

::index.html.twig // ce fichier se trouve directement dans app/ressources/views

FormationBundle::index.html.twig // ce fichier se trouve dans src/FormationBundle/ressources/views

FormationBundle:Multimedia:index.html.twig // ce fichier se trouve dans src/FormationBundle/Multimedia/ressources/views

Nommage des pages TWIG

Par convention le nommage des vues dans symfony se fait selon la convention suivante :

NomPage.FormatFinal.MoteurDeTemplate

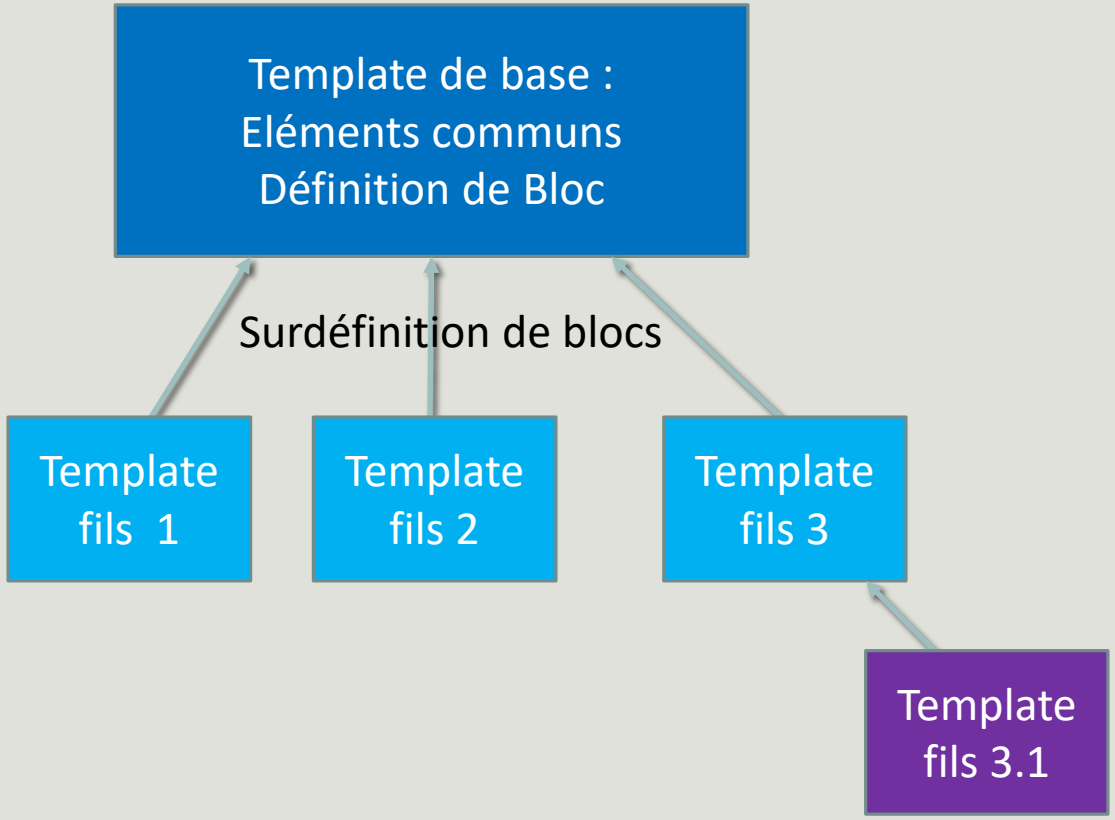
Exemple

index.html.twig

Le nom du fichier est index, le format final sera du html et le moteur de template suivi est le TWIG

Héritage 1- Définition

- Vision proche de l'héritage dans l'orienté objet



Héritage 2 – Exemple de Template père

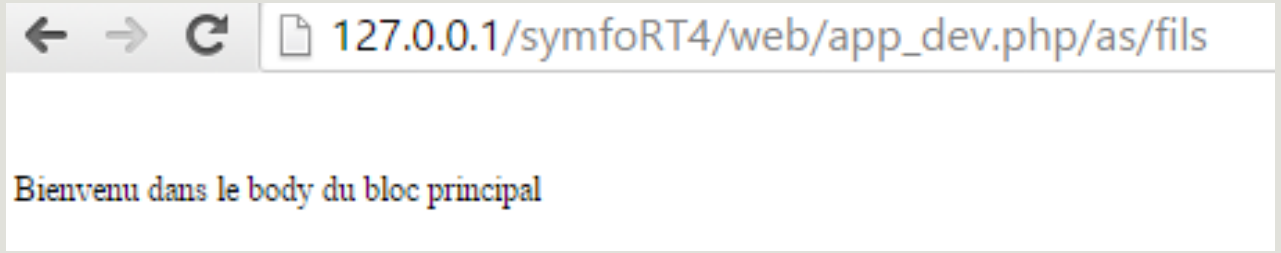
- Exemple de Template de base

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Bonjour je suis le bloc principal!{% endblock %}</title>
        {% block stylesheets %}{#ici je vais définir mes fichiers css#}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" /> {#ici c'est le favicon de mon appli#}
    </head>
    <body>
        {% block body %}
        {#ici c'est le contenu du Body#}
        Bienvenu dans le body du bloc principal
        {% endblock %}
        {% block javascripts %}{#ici je vais définir mes fichiers js#}{% endblock %}
    </body>
</html>
```

Héritage 3- Syntaxe

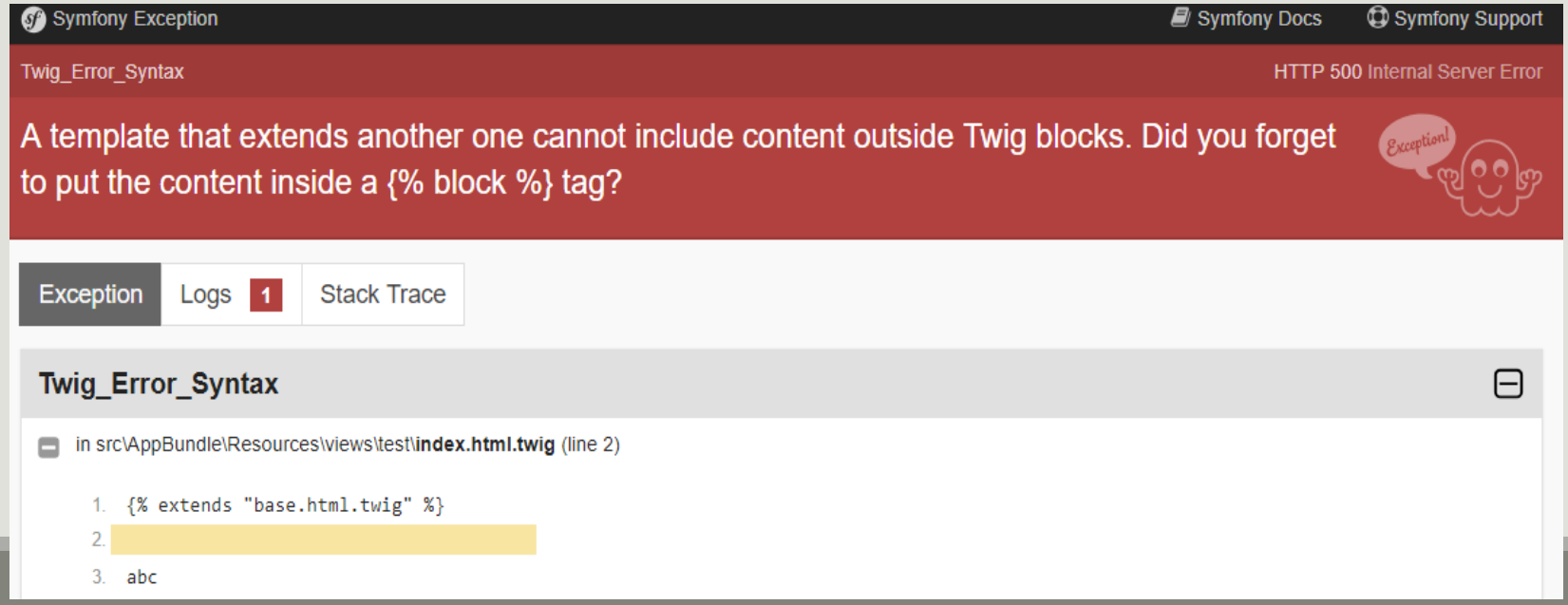
- Afin d'hériter d'un Template père il faut étendre ce dernier avec la syntaxe suivante :

- {% extends 'TemplateDeBase' %}
- Exemple :
- {% extends 'base.html.twig' %}
- Si le fils ne surcharge aucun des blocs et n'ajoute rien on aura le même affichage que pour la base



Héritage 4- Le Bloc

- L'élément de base de l'héritage est le bloc
- Un bloc est définit comme suit : {% block nomBloc%} contenu du bloc{%endblock%}
- Pour changer le contenu de la page il faut sur-définir le bloc cible
- En héritant d'une page et si vous écrivez du code à l'extérieur des blocs vous aurez le message suivant qui est très explicite.



Héritage 5- Récupérer le contenu d'un bloc parent

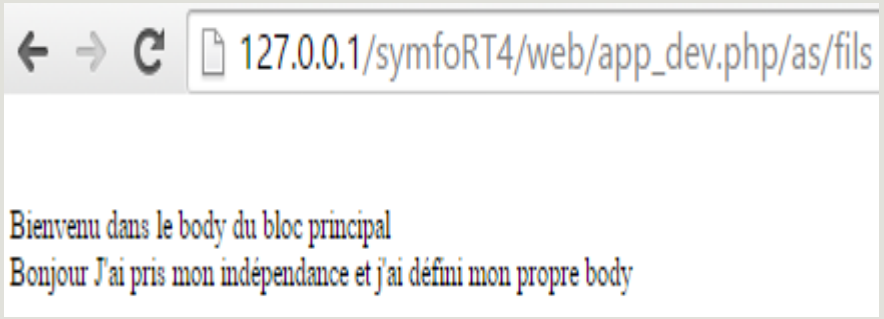
- Pour récupérer le contenu d'un bloc père il suffit d'utiliser la méthode parent()

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

{% block body %}

    {{ parent() }}<br>
    Bonjour J'ai pris mon indépendance et j'ai défini mon propre body

{% endblock %}
```



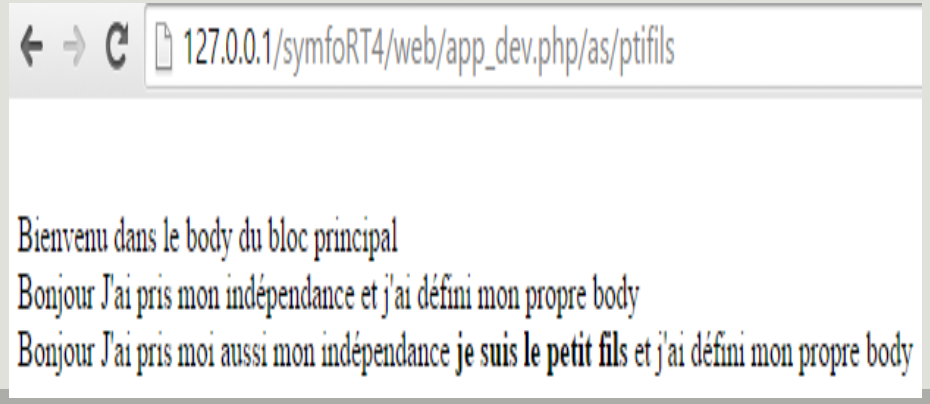
- Ceci peut être aussi appliqué pour toute la hiérarchie

```
{% extends 'Rt4AsBundle:Default:fils.html.twig' %}

{% block body %}

    {{ parent() }}<br>
    Bonjour J'ai pris moi aussi mon indépendance <b> je suis le petit fils </b> et j'ai défini mon propre body

{% endblock %}
```



Inclusion de Template

Afin d'inclure un Template ou un fragment de code dans un autre template on utilise la syntaxe suivante :

{{ include('NomBundle:Dossier:nomTemplate', {'labelParam1': param1,'labelParam2': param2,... })

Exemple :

{{ include('Rt4AsBundle:Default:section.html.twig', {'Section': section})

Inclusion de Contrôleur

Que faire si on veut inclure un template dynamique ? Un template des meilleurs ventes un template des articles les plus vus, les derniers cours postés …

La solution consiste à afficher la valeur de retour de la fonction render qui prend en paramètres le controlleur à appeler et les attributs qu'il attend de recevoir.

Inclure un contrôleur sans ou avec des paramètres.

Syntaxe :

{{ render(controller('NomBundle:NomController:NomAction', {'labelParam1': param1,'labelParam2': param2,... })) }}

Exemple

{{ render(controller('Rt4AsBundle:Student:Top', { 'nb': 10 })) }}

Génération de liens avec TWIG

Twig permet de générer des liens à partir des noms de root en utilisant la fonction path.

```
rt4_as_homepage:
    path:     /hello/{name}
    defaults: { _controller: Rt4AsBundle:Default:index }

rt4_as_base:
    path:     /base
    defaults: { _controller: Rt4AsBundle:Default:base }

rt4_as_fils:
    path:     /fils
    defaults: { _controller: Rt4AsBundle:Default:fils }

rt4_as_ptifils:
    path:     /ptifils
    defaults: { _controller: Rt4AsBundle:Default:petitFils }
```

```
{% extends 'Rt4AsBundle:Default:fils.html.twig' %}

{% block body %}

    {{ parent() }}<br>
    Bonjour J'ai pris moi aussi mon indépendance <b> je suis le petit fils </b> et j'ai défini mon propre body <br>

    <a href="{{ path('rt4_as_base') }}">mon grand père est ici</a><br>
    <a href="{{ path('rt4_as_fils') }}"> mon père est ici </a><br>
{% endblock %}
```

Génération de liens paramétrable avec TWIG

Si le lien contient des paramètres, on garde la même syntaxe de la fonction path et on y ajoute comme deuxième paramètre un tableau contenant les paramètres passer à la route.

Syntaxe : {{ path('root', { 'param1': param1, 'param2': param2,... }) }}

```
rt4_as_homepage:
    path:     /hello/{name}
    defaults: { _controller: Rt4AsBundle:Default:index }

rt4_as_base:
    path:     /base
    defaults: { _controller: Rt4AsBundle:Default:base }

rt4_as_fils:
    path:     /fils
    defaults: { _controller: Rt4AsBundle:Default:fils }

rt4_as_ptifils:
    path:     /ptifils
    defaults: { _controller: Rt4AsBundle:Default:petitFils }
```

```
{% extends 'Rt4AsBundle:Default:fils.html.twig' %}

{% block body %}

    {{ parent() }}<br>
    Bonjour J'ai pris moi aussi mon indépendance <b> je suis le petit fils </b> et j'ai défini mon propre body <br>

     <a href="{{ path('rt4_as_base') }}">mon grand père est ici</a><br>
    <a href="{{ path('rt4_as_fils') }}"> mon père est ici </a><br>
    <a href="{{ path('rt4_as_homepage', {name: 'aymen'}) }}"> Bonjour </a><br>

{% endblock %}
```

Génération de liens absolus

Pour générer l'url absolue nous utilisons la fonction url. Elle prend en paramètre le nom de la root souhaitée. Cet fonctionnalité peut être utile par exemple si vous envoyer un lien par mail. Ce lien ne peut pas être relatif sinon il sera traité relativement la ou il est exécuté et non relativement à votre serveur.

Syntaxe : {{ url('root') }}

```
{% extends 'Rt4AsBundle:Default:fils.html.twig' %}

{% block body %}

    {{ parent() }}<br>
    Bonjour J'ai pris moi aussi mon indépendance <b> je suis le petit fils </b> et j'ai défini mon propre body <br>

     <a href="{{ path('rt4_as_base') }}">mon grand père est ici</a><br>
    <a href="{{ path('rt4_as_fils') }}"> mon père est ici </a><br>
    <a href="{{ path('rt4_as_homepage', {name: 'aymen'}) }}"> Bonjour </a><br>

    url relative : {{ path('rt4_as_fils') }} <br>
    url absolue  : {{ url('rt4_as_fils') }}

{% endblock %}
```

Bienvenu dans le body du bloc principal
Bonjour J'ai pris mon indépendance et j'ai défini mon propre body
mon père est ici

Bonjour J'ai pris moi aussi mon indépendance **je suis le petit fils** et j'ai défini mon propre body
mon grand père est ici
mon père est ici
Bonjour
url relative : /symfoRT4/web/app_dev.php/as/fils
url absolue : http://127.0.0.1/symfoRT4/web/app_dev.php/as/fils

Asset

Pour générer le path d'un fichier (img, css, js,…) nous utilisons la fonction asset.

Cette fonction permet la portabilité de l'application en permettant une génération automatique du path du fichier indépendamment de l'emplacement de l'application.

Exemple :

- Si l'application est hébergé directement dans la racine de l'hôte alors le path des images est /images/nomImg.
- Si l'application est hébergé dans un sous répertoire de l'hôte alors le path des images est /nomApp/images/nomImg.

Syntaxe :

asset('ressource')

Exemples

```
<img src="{{ asset('images/logo.png') }}" alt="Symfony!" />
<link href="{{ asset('css/blog.css') }}" rel="stylesheet" type="text/css" />
```

Surcharge de Template

Afin de surcharger les Template d'un Bundle, nous nous basons sur le fonctionnement de base de symfony2.

Lorsque le contrôleur fait appel à un Template dans un Bundle Symfony2 cherche dans 2 emplacements selon l'ordre suivant :

1) app/Resources/NomBundle/views/DossierTemplate/nomTemplate.html.twig
2) src/NomAppli/NomBundle/Resources/views/DossierTemplate/nomTemplate.html.twig

Pour surcharger le template il suffit donc de copier ce dernier vers le dossier app/Resources/NomBundle/views/DossierTemplate/ que vous devez créer vous-même et personnaliser le Template à votre guise.

Accéder aux variables globales

A chaque requête, symfony vous fournit une variables globale app dans votre Template. Cette variable vous permet d'accéder à plusieurs informations très utiles.

1. app.session que nous avons vu dans le skill Controller nous permet de récupérer la session courante. Elle a comme valeur null en cas d'absence de session.
2. app.user permet de récupérer l'utilisateur courant connecté et null si aucun utilisateur n'est connecté.
3. app.environment : permet de récupérer l'environnement courant, e.g. dev, prod, test.
4. app.debug retourne true si on est dans le debug mode false sinon.

Débugger avec dump

Afin de débugger les variables dans votre page TWIG vous pouvez utiliser la fonction dump.

Exemple :

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

{% block body %}

    {{ dump(app) }}

{% endblock %}
```