Twig

Dans ce chapitre, nous allons voir comment :

- Accéder à un service depuis twig
- Créer nos propres extensions (filtres) twig.
- Générer le rendu d'un contrôleur depuis le template
- Afficher un Fil d'Ariane dans notre menu

Finalisation du Layout

Pour la finalisation de notre thème, nous allons :

1. Afficher les catégories de notre site dans le menu
2. Générer le contenu de notre sidebar via la fonction render()

Accéder à un service depuis Twig

Il nous est tout à fait possible d'accéder à un de nos services directement depuis notre template twig.

Nous allons tout d'abord déclarer notre service dans twig :

config/packages/twig.yaml :

```
twig:
   ...
    globals:
      categorie_service: '@App\Repository\CategorieRepository'
```

Nous pouvons maintenant accéder à notre service depuis notre thème :
Créons une variable categories : dans **layout.html.twig** de cette façon la variable sera accessible dans toutes les vues.

```
{% set categories = categorie_service.findAll() %}
{{ dump(categories) }}
```

Vous devriez voir toutes les catégories du site s'afficher !
Une boucle et le tour est joué !

> Mettons à jour notre menu et notre pied de page.

Créer un filtre twig

Pour la génération de nos urls, nous aurons besoin d'un slug. Nous aurons également besoin de générer une accroche de 170 caractères pour nos articles.

Pour cela nous allons mettre en place une extension twig !

> Doc de Référence : https://symfony.com/doc/current/templating/twig_extension.html

Dans notre console :

```
composer require twig/extensions
```

Créons ensuite notre classe AppExtension qui hérite de la classe AbstractExtension dans **Service/Twig/AppExtension.php**

> Atelier : Création de AppExtention.php

Grâce à l'autowiring de SF4, notre extension est automatiquement chargé !

Nous pouvons maintenant finaliser la générations des URLs dans toutes les pages. *(Index, Categorie, Article, Nav et Footer)*

Générer le rendu d'un contrôleur depuis le template

Nous allons attaquer la dernière partie de notre site, la sidebar.

> Doc de Référence : https://symfony.com/doc/current/templating/embedding_controllers.html

Dans certains cas, nous avons besoin d'inclure plus qu'un simple template dans notre vue. Par exemple, dans le cas de notre sidebar, nous avons besoin de récupérer les 5 derniers articles de la base, mais aussi les articles en position "special".

Vous comprenez vite qu'il ne sera pas possible de faire notre requête directement depuis notre template…

La solution est alors la suivante, nous allons inclure dans notre template, le résultat du rendu d'un contrôleur.

Dans **IndexController**, créons une fonction **sidebar()** :

```
public function sidebar() {

        # Récupération du Répository
        $repository = $this->getDoctrine()->getRepository(Article::class);

        # Récupération des 5 derniers articles
        $articles   = $repository->findLastFiveArticle();

        # Récupération des articles à la position "special"
        $special    = $repository->findSpecialArticles();

        return $this->render('components/_sidebar.html.twig', [
            'articles'  => $articles,
            'special'   => $special
        ]);

    }
```

Depuis notre **_sidebar.html.twig** nous pouvons récupérer les variables :

```
{{ dump(articles) }}
{{ dump(special) }}
```

Nous pouvons maintenant faire nos boucles…

Pour afficher ensuite notre sidebar, dans nos vues :

```
{{ render(controller('App\\Controller\\TechNews\\IndexController::sidebar')) }}
```

Comprenez ici que symfony va executer la fonction **sidebar()** de notre **IndexController**. Nous auront alors le rendu de **_sidebar.html.twig** via **sidebar()** dans notre vue !

Affichage du current

Il est souvent pratique sur un site internet d'afficher à l'utilisateur un Fil d'Ariane. C'est à dire lui indiquer ou il se trouve sur notre site.

Notre thème dispose d'une classe **"current"** à appliquer sur la balise **"li"** pour indiquer à l'utilisateur sur quelle page il se trouve.

Pour mettre en place cela, nous allons simplement vérifier si la page sur laquelle l'utilisateur se trouve correspond à l'une des catégories du menu.

```
{% for categorie in categories %}
     <li {% if(active == categorie.libelle) %}class="current"{%endif%}>...</li>
 {% endfor %}
```

Nous allons ensuite créer une variable **"active"** via twig dans notre page catégorie **(categorie.html.twig)** :

```
{% set active = app.request.get('libellecategorie') | capitalize %}
```

De cette façon lorsque PHP parcours les catégories, s'il trouve une correspondance il ajoutera la classe current sur le "li" correspondant.

Pour la page d'accueil, nous mettrons manuellement la déclaration :

```
<li {% if(active == 'index') %}class="current"{%endif%}></li>
```

Dans la vue :

```
{% set active = 'index' %}
```

Nous pourrons faire pareil par la suite pour les autres pages...

```
{% set active = '...' %}
```

Vous devriez avoir maintenant un site internet entièrement fonctionnelle côté front !

Written with ❤ by Hugo LIEGEARD.