Le Templating | Symfony 4

Nous allons mettre en place le thème de notre projet TechNews.
Mais avant cela, nous allons ajouter à notre projet Twig.

> Doc de référence :
> https://symfony.com/doc/current/templating.html
> https://twig.symfony.com

Présentation de Twig

Les templates vont nous permettre de séparer le code PHP du code HTML/XML/Text, etc. Seulement, pour faire du HTML de présentation, on a toujours besoin d'un peu de code dynamique : faire une boucle pour afficher toutes les annonces de notre plateforme, créer des conditions pour afficher un menu différent pour les utilisateurs authentifiés ou non, etc. Pour faciliter ce code dynamique dans les templates, le moteur de templates Twig offre son pseudo-langage à lui.
Source : Openclassroom.

Mise en Place de Twig

Nous allons lancer dans notre console :

> composer require twig

Ce qui aura pour effet de charger dans notre projet SF4 tous le nécessaire au fonctionnement de twig !

Nous allons maintenant procéder aux étapes suivantes :

- Création de notre structure HTML de base : **base.html.twig** ;
- Hériter du **Controller** Symfony ;
- Rendu du template Twig sur notre page d'Accueil.

> Doc de Référence
> https://symfony.com/doc/current/controller.html#the-base-controller-classes-services et
> https://symfony.com/doc/current/controller.html#rendering-templates

```
class IndexController extends Controller
...
public function index() {
    return $this->render('base.html.twig');
}
```

Une fois notre thème de base mis en place, nous allons mettre en place notre thème TechNews.

Mise en Place de Encore Webpack

> Doc de référence : https://symfony.com/doc/current/best_practices/web-assets.html
> https://symfony.com/doc/current/templating.html#linking-to-assets

Depuis SF4, nous avons la possibilité d'intégrer facilement Webpack à Symfony grâce à Encore.

Présentation de Webpack

Pas besoin de s'appeler Einstein pour comprendre les intentions de Webpack en interprétant le nom : web + pack. Faire un pack prêt pour le web.

Il vous permettra de faire bien des choses : utiliser un serveur local, utiliser le Live Reload, minifier vos fichiers CSS, JS, mais aussi et surtout compiler tous vos fichiers pour les regrouper en un seul. Pratique pour bien des raisons, notamment pour la performance de votre app !

> Tuto à lire :
> http://putaindecode.io/fr/articles/js/webpack/
> https://www.alsacreations.com/tuto/lire/1754-debuter-avec-webpack.html

Présentation de Encore

> Doc de Référence : https://symfony.com/doc/current/frontend.html#encore-toc

"Encore", est le nouveau système de gestion des assets (JS, CSS) basé sur Webpack de Symfony.

Webpack Encore est une façon simple d'intégrer WebPack dans votre projet symfony.

Installation de Encore

> Doc de Référence :
> https://symfony.com/doc/current/frontend/encore/installation.html

Vu que nous travaillons avec Symfony Flex (v4) nous allons utiliser composer + yarn | npm.

> *NOTA BENE : Explication de Yarn / Npm*

Dans notre console :

```
composer require encore asset
npm install
```

Nous pouvons ensuite finaliser la configuration de nos assets.

> Doc de Référence :
> https://symfony.com/doc/current/frontend/encore/simple-example.html

Configuration de Encore / Webpack

Nous allons maintenant ouvrir le fichier **webpack.config.js**
Symfony nous à déjà mis en place une configuration de base que nous allons adapter.

Nous allons aussi rajouter tout nos fichiers CSS et JS.
Il seront ensuite compilé et regroupé en un seul fichier CSS et JS.

```
// -- Chargement du CSS
require('./css/main.scss');

// -- Chargement du JS
require('./js/jquery.min');
require('./js/bootstrap.js');
require('./js/owl.carousel.min.js');
require('./js/main.js');
```

Ensuite via notre console nous allons lancer la compilation de nos assets :

```
yarn run encore dev
```

Vous devriez maintenant voir dans votre dossier **public** un nouveua dossier **build** contenant nos assets avec un fichier **app.css** et **app.js**

C'est deux fichiers contiennent le regroupement de l'ensemble de nos fichiers CSS et JS.

Mise en Place de notre Thème

Nous allons mettre en place l'architecture nécessaire au fonctionnement de notre thème TechNews en nous appuyant sur Twig.

1. Création du Menu **_nav_menu.html.twig**
2. Création du Footer **_footer.html.twig**
3. Création de la Sidebar **_sidebar.html.twig**
4. Création d'un Layout **layout.html.twig**
5. Création de notre Vue Accueil **index.html.twig**

Nous allons maintenant créer nos différentes vues :

Nous avons maintenant mis en place notre architecture de base, nous allons pouvoir entrer en détail dans nos vues. **Nous allons mettre en place les views suivantes :**

- Affichage de la Page d'Accueil
- Affichage des Articles d'une Catégorie
- Affichage d'un Article du site.

Génération des URLs :

> Doc de Référence : https://symfony.com/doc/current/templating.html#linking-to-pages

Nous allons maintenant mettre en place la navigation inter-pages de notre site.

```
<a href="{{ url('index') }}">Accueil</a>
```

Ici, la fonction **url()** va demander à Symfony de créer une URL absolue pour la route index. Le résultat après compilation sera alors :

```
<a href="http://localhost:8000/">Accueil</a>
```

Lorsqu'il y à des paramètres, nous procédons de la façons suivantes :

```
<a href="{{ url('index_categorie', { libellecategorie : 'business' }) }}">Business</a>
```

Ce qui donnera :

```
<a href="http://localhost:8000/categorie/business">Business</a>
```

Written with ❤ by Hugo LIEGEARD.