Les assets

A l'installation de notre projet Symfony, un package **symfony/asset** a été installé automatiquement. Grâce à ce package et la fonction `asset` qu'il nous fournit dans les templates Twig, on va pouvoir gérer l'inclusion de fichiers statiques au sein de nos pages (CSS, JS, images...).

Si on veut créer une feuille de style CSS par exemple, on va créer un dossier `css` dans le dossier `public` situé à la racine du projet.

> **LE DOSSIER `public/`**
>
> Le dossier `public/`, à la racine du projet, est la **racine** exposée par notre serveur, quand il est lancé.
>
> Par exemple, si je tape `http://localhost:8000/css/styles.css`, j'accèderai au fichier qui se trouve dans `public/css/styles.css`

Dans ce dossier `public/css`, on va créer un fichier `index.css`, et pour tester le bon fonctionnement, y inscrire un code CSS de test :

```
body {
  background-color: #ddd;
}
```

Enfin, pour utiliser le composant `asset` de Symfony, on va utiliser une extension Twig dans le template de base de notre application pour y intégrer notre feuille de style. Dans le fichier `base.html.twig`, dans le bloc `stylesheets` :

```
{% block stylesheets %}
  {# En-dessous de la balise qui intègre Bootstrap... #}
  <link rel="stylesheet" href="{{ asset('css/index.css') }}">
{% endblock %}
```

On peut à présent relancer le serveur avec `symfony serve --no-tls`, et vérifier que l'arrière-plan de la page d'accueil est bien gris clair. Le composant asset inclut automatiquementle fichier en se basant sur le répertoire `public/`.

Grouper les assets

On peut grouper des assets ensemble, dans un **package**, et leur définir un répertoire de base par exemple :

> config/packages/framework.yaml

```
framework:
  assets:
    packages:
      css:
        base_path: "/css"
      images:
        base_path: "/images"
```

Ensuite, on peut indiquer depuis le template Twig pour quel package on souhaite inclure l'asset :

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

Versionner les assets

Il est possible de configurer une politique de versioning dans le fichier `config/packages/framework.yaml`, ainsi que le format dans lequel la version doit être intégrée à l'URL de l'asset :

```yaml
framework:
  assets:
    version: "v1"
    version_format: "%%s?v=%%s"
```

Lors d'une mise à jour, on pourra donc changer la version incluse à l'URL pour forcer le rechargement de nos feuilles de style par exemple.

- Plus d'informations sur le composant Asset ici
- Plus d'informations sur les paramètres de configuration ici

> **WEBPACK ENCORE**
>
> Pour gérer le front au sein d'une application Symfony, un package complet **Webpack Encore** a été développé par les contributeurs de Symfony. Ce package met en oeuvre des outils Javascript, ce n'est donc pas la priorité de ce module.
>
> Par ailleurs, si on utilise une librairie ou un framework JS pour gérer notre front-end, nous n'aurons besoin de Symfony que pour gérer la partie API, et non l'interface. C'est aussi pour cette raison qu'on ne s'attarde pas là-dessus.
>
> Si ça vous intéresse, prenez le temps de regarder tout de même !