

Symfony - Bases - 3 - Twig

⌚ Cours	Symfony
⌚ Type	Guide
⌚ Supports	https://twig.symfony.com/
⌚ Difficulté	Très Facile

Twig est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le Framework Symfony.

Syntaxe

- {{ say something }} permet d'afficher le contenu demandé. Il est possible d'afficher des variables, des fonctions ou un template.
- {# ça, c'est le commentaire #}
- {% do something %} permet d'effectuer des commandes comme des conditions, des boucles etc.

Essayez donc, dans votre fichier `contact.html.twig` d'afficher le titre (normalement vous devriez avoir réussi à faire ça dans l'exercice de la partie 2)

```
{# bloc HTML, on dicte à twig qu'on fait un bloc, print du titre, on dicte à twig qu'on ferme le bloc, fin du bloc html #}
<title>{% block title %}{{title}}{% endblock %}</title>
```

Essayez ça maintenant :

```
<title>{% block title %}{{title}}{% endblock %}</title>
<form>
    Contact :

    {% for data in form %}
        <label for="{{ data }}">{{ data }}</label>
        <input type="text" name="{{ data }}"/>
    {% endfor %}

</form>
```

```
#[Route('/contact')]
public function showContact() : Response
{
```

```

$form = [
    'Nom',
    'Prénom',
    'Adresse',
    'email'
];
return $this->render('contact/contact.html.twig', [
    'title' => 'Contact',
    'form' => $form,
]);
}

```

Vous venez de parcourir un tableau et de créer des cases de formulaires pour index du tableau ! Félicitations, joli copier/coller 

Ce que vous venez de faire avec Twig est assez simple. Vous avez créé une boucle avec `{% for valeur in tableau %}` suivi d'un simple affichage de chaque valeur avec `{{ valeur }}`. Rien de bien sorcier pour le moment .

Ce qui est cool avec Twig, c'est que tout est assez facile à intégrer. La syntaxe sera toujours la même et vous avez juste à appeler ce qu'il vous faut !

Vous pouvez effectuer beaucoup de choses avec la syntaxe : `{% %}`. Vous pouvez trouver la liste de [toutes les commandes ici](#) (scrollez un tout petit peu, c'est tous les petits mots que vous voyez partout)

Je vais passer les if, for etc. que vous connaissez déjà et vous donner quelques exemples faciles à utiliser :

Upper : Permet de transformer votre texte en majuscules

```

{{ 'test'|upper}}
Sortie : TEST

```

Lower : Transforme votre texte en minuscules évidemment

```

{{ 'TEST'|lower}}
Sortie : test

```

Capitalize : Première lettre de la phrase en majuscule

```

{{ 'test de test'|capitalize}}
Sortie : Test de test

```

Set : Permet d'assigner une valeur à une variable

```

{% set maVariable = 'quelque chose' %}

```

Inheritance

Le principe le plus important de Twig est celui de l'héritage.

Essayez d'inspecter le code d'une de vos pages. Vous allez vite voir que ça ressemble à ceci :

```

...<html> == $0
  ▼<head>
    <title>Contact</title>
  </head>
  ▼<body>
    ▼<form>
      " Contact : "
      ▶<label for="Nom">...</label>
      ▶<label for="Prénom">...</label>
      ▶<label for="Adresse">...</label>
      ▶<label for="email">...</label>
    </form>
  </body>
</html>

```

Rien ne vous choque ici ? Il n'y a pas de Doctype !

i C'est parce que Twig possède un système d'intégrations de composants et c'est exactement ce qu'on va faire dès maintenant.

Dirigez-vous sur le fichier `base.html.twig` que nous avons laissé de côté jusqu'à présent et lisez le code devant vous :

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>% block title %}Welcome!{% endblock %}</title>
    <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22><text y=%221.2
    {% block stylesheets %}
    {% endblock %}

    {% block javascripts %}
    {% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
  </body>
</html>

```

C'est votre composant par défaut, d'où son nom : BASE !

On va "Override" (écraser) le `{% block body %}{% endblock %}` ensemble ! Car c'est là qu'on souhaite afficher notre page.

Dirigeons-nous dans notre précédente page (la page contact) et ajoutons-y ces deux blocs précédés par un `extends`

```

{% extends 'base.html.twig' %}
{% block body %}
<title>% block title %}{{title}}{% endblock %}</title>
<form>
  Contact :

  {% for data in form %}
  <label for="{{ data }}">{{ data }}
    <input type="text" name="{{ data }}"/>
  </label>
  {% endfor %}

</form>
{% endblock %}

```

Cette syntaxe ne vous rappelle rien ? C'est un peu comme le `include` de PHP. On a réussi à importer une page par défaut dans notre page actuellement. Un bon moyen de faire apparaître le même header et le même footer partout ! 😊

Doctype est revenu par magie ! 🎉

D'ailleurs, vous l'avez peut-être compris, mais maintenant vous pouvez retirer la ligne ;

```
<title>{% block title %}{ {title}}{% endblock %}</title>
```

de votre page contact et simplement préciser le contenu de ce bloc pour l'override :

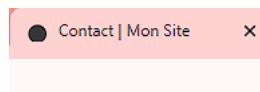
```
{% block title %}Contact{% endblock %}
```

On peut même aller encore plus loin, je viens de vous parler d'héritage ! Essayez donc dans base.html.twig d'écrire :

```
<title>{% block title %}Mon site{% endblock %}</title>
```

Puis dans votre page contact d'override le bloc comme ceci :

```
{% block title %}Contact | {{ parent() }}{% endblock %}
```



Tadaa !!

i Pour comprendre un peu plus ce que vous venez de faire. Pensez donc à la POO ! Chaque template est similaire à une classe, et chaque block est similaire à une méthode !