14. Héritage de template

Django vous réserve encore bien des surprises : une assez géniale est l'héritage de template. Qu'est ce que ça signifie ? C'est une fonctionnalité qui vous permet de réutiliser certains morceaux de HTML dans différentes pages de votre site web.

Les modèles permettent d'utiliser les mêmes informations ou mises en page en plusieurs endroits. Vous n'avez pas à vous répéter dans chaque fichier. Et si vous voulez changer quelque chose, vous n'avez pas à le faire dans chaque modèle, mais juste un! Créer un template de base

Un template de base est le template le plus simple que vous pouvez faire hériter à chaque page de votre site web.

Créons le fichier base.html dans le dossier blog/templates/blog/ :

```
blog
└── templates
    └── blog
        base.html
        post_list.html
```

Ensuite, ouvrez ce fichier dans l'éditeur de code et collez-y tout ce qui se trouve dans le fichier post_list.html. Ça devrait ressembler à ça :

`blog/templates/blog/base.html`

```
{% load static %}
<html>
    <head>
        <title>Django Girls blog</title>
        <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
        <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-
theme.min.css">
        <link
href='//fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext'
rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="{% static 'css/blog.css' %}">
    </head>
    <body>
        <div class="page-header">
            <h1><a href="/">Django Girls Blog</a></h1>
        </div>

        <div class="content container">
            <div class="row">
                <div class="col-md-8">
                {% for billet in billets %}
                    <div class="billet">
                        <div class="date">
```

```
                            {{ billet.published_date }}
                        </div>
                        <h2><a href="">{{ billet.title }}</a></h2>
                        <p>{{ billet.text|linebreaksbr }}</p>
                    </div>
                {% endfor %}
                </div>
            </div>
        </div>
    </body>
</html>
```

Puis, dans le fichier `base.html`, remplacez tout ce qui se trouve dans <body> (de <body> à </body>) par ceci :

`blog/templates/blog/base.html`

```
<body>
    <div class="page-header">
        <h1><a href="/">Django Girls Blog</a></h1>
    </div>
    <div class="content container">
        <div class="row">
            <div class="col-md-8">
            {% block content %}
            {% endblock %}
            </div>
        </div>
    </div>
</body>
```

Vous pouvez remarquer qu'on vient de remplacer tout ce qui était entre {% for billet in billets %} et {% endfor %} avec :

`blog/templates/blog/base.html`

```
{% block content %}
{% endblock %}
```

Mais pourquoi? Vous venez de créer un block ! Vous avez utilisé la balise de modèle `{% block %}` pour créer une partie qui va contenir du HTML. Ce HTML viendra d'un autre modèle qui étendra ce modèle-ci (`base.html`). Nous vous expliquerons comment faire cela dans un instant.

Enregistrez base.html et ouvrez votre `blog/templates/blog/post_list.html` à nouveau dans l'éditeur de code. Vous allez tout supprimer au-dessus de `{% for billet in billets %}` et en dessous de `{% endfor %}`. Lorsque vous avez terminé, le fichier ressemblera à ceci :

`blog/templates/blog/post_list.html`

```
{% for billet in billets %}
    <div class="billet">
        <div class="date">
            {{ billet.published_date }}
        </div>
        <h2><a href="">{{ billet.title }}</a></h2>
        <p>{{ billet.text|linebreaksbr }}</p>
    </div>
{% endfor %}
```

Nous voulons utiliser ceci dans le cadre de notre modèle pour tous les blocs 'content'. Il est temps d'ajouter ajouter des balises block à ce fichier !

Vous voulez que votre balise block corresponde à la balise dans votre fichier base.html . Vous voulez aussi qu'il inclue tout le code qui appartient à vos blocs de contenu. Pour faire cela, entourez tout le code de `{% block content %}` et `{% endblock %}`. Comme ceci :

`blog/templates/blog/post_list.html`

```
{% block content %}
    {% for billet in billets %}
        <div class="billet">
            <div class="date">
                {{ billet.published_date }}
            </div>
            <h2><a href="">{{ billet.title }}</a></h2>
            <p>{{ billet.text|linebreaksbr }}</p>
        </div>
    {% endfor %}
{% endblock %}
```

Seule une chose reste à faire. Nous devons connecter ces deux modèles ensemble. C'est ça, étendre des modèles ! Nous allons le faire en ajoutant une balise 'extends' au début du fichier. Comme cela :

`blog/templates/blog/post_list.html`

```
{% extends 'blog/base.html' %}

{% block content %}
    {% for billet in billets %}
        <div class="billet">
            <div class="date">
                {{ billet.published_date }}
            </div>
            <h2><a href="">{{ billet.title }}</a></h2>
            <p>{{ billet.text|linebreaksbr }}</p>
        </div>
    {% endfor %}
{% endblock %}
```

Et voilà ! Enregistrez le fichier et vérifiez que votre site fonctionne toujours correctement. :)

Si vous obtenez l'erreur `TemplateDoesNotExist`, cela signifie qu'il n'y a pas de fichier `blog/base.html` et que vous avez runserver en cours d'exécution dans la console. Arrêtez-le (en appuyant simultanément sur Ctrl+C, les touches Control et C de votre clavier) et relancez-le en tapant la commande python manage.py runserver.

15. Finaliser votre application