Templates Django

Il est temps d'afficher des données ! Pour nous aider, Django fournit des balises de gabarit (template tags) qui sont intégrées au framework. Pour le reste du tutoriel, nous utiliserons plutôt le mot template, bien plus répandu que sa traduction "gabarit".

Qu'est-ce que c'est que des balises de template ?

En HTML, vous ne pouvez pas mettre directement du code Python car les navigateurs seraient incapables de le comprendre. Les navigateurs ne connaissent que le HTML. Nous vous avons signalé précédemment que HTML est du genre statique, alors que Python est bien plus dynamique.

Les Balises de template Django nous permettent de transférer des choses ressemblant à du Python dans du HTML afin de nous permettre de construire des sites web plus rapidement. Cool, non ?

Template d'affichage de la liste des billets

Dans le chapitre précédent, nous avons donné à notre template une liste de billets à l'aide de la variable billets. Nous allons maintenant les afficher en HTML.

Afin d'afficher une variable dans un template Django, nous utiliserons des doubles accolades avec le nom de la variable à l'intérieur. Ça ressemble à ceci :

blog/templates/blog/post_list.html

```
{{ billets }}
```

Essayez de faire la même chose avec votre template blog/templates/blog/post_list.html. Ouvrez-le avec votre éditeur et remplacez tout ce qui se trouve entre la seconde balise

jusqu'au troisième

avec la ligne

billets

. Sauvegardez votre fichier et rafraîchissez votre page pour voir le résultat :

https://tutorial.djangogirls.org/fr/django_templates/images/step1.png

Comme vous pouvez le voir, tout ce que nous avons, c'est ceci :

blog/templates/blog/post_list.html

```
<QuerySet [<Billet: My second post>, <Billet: My first post>]>
```

Cela signifie que Django l'interprète comme une liste d'objets. Essayez de vous rappeler comment afficher des listes en Python. Dans un template Django, vous pouvez les écrire de la façon suivante :

blog/templates/blog/post_list.html

```
{% for billet in billets %}
```

```
    {{ billet }}
{% endfor %}
```

Essayez ceci dans votre template.

[Image: Figure 13.2](image_link)

Ça marche ! Cependant, nous aimerions plutôt afficher les billets à la manière des billets statiques, comme lorsque nous les avions créés dans le chapitre Introduction au HTML. Vous pouvez mélanger HTML et balises de template. Notre <body> ressemble maintenant à ceci :

blog/templates/blog/post_list.html

```
<div>
    <h1><a href="/">Django Girls Blog</a></h1>
</div>

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

Tout ce qui se situe entre '{% for %}' et '{% endfor %}' va être répété pour chaque objet présent dans la liste. Rafraîchissez votre page :

[Image: Figure 13.3](image_link)

Avez-vous remarqué que nous avons utilisé une notation légèrement différente cette fois-ci (

billet.title

ou

billet.text

) ? Nous accédons aux données associées à chaque champ défini dans notre modèle Billet. De même, le |linebreaksbr nous permet de rediriger le texte des billets à travers un filtre qui convertit automatiquement les fins de lignes en paragraphes.

Ça a marché ? Bravo ! Éloignez vous un peu de votre clavier maintenant : vous avez mérité de faire une pause. :)

[Figure 13.4](image_link)

Templates Django

Il est temps d'afficher des données ! Pour nous aider, Django fournit des balises de gabarit (template tags) qui sont intégrées au framework. Pour le reste du tutoriel, nous utiliserons plutôt le mot template, bien plus répandu que sa traduction "gabarit". Qu'est-ce que c'est que des balises de template ?

En HTML, vous ne pouvez pas mettre directement du code Python car les navigateurs seraient incapables de le comprendre. Les navigateurs ne connaissent que le HTML. Nous vous avons signalé précédemment que HTML est du genre statique, alors que Python est bien plus dynamique.

Les Balises de template Django nous permettent de transferer des choses ressemblant à du Python dans du HTML afin de nous permettre de construire des sites web plus rapidement. Cool, non ?
Template d'affichage de la liste des billets

Dans le chapitre précédent, nous avons donné à notre template une liste de billets à l'aide de la variable billets. Nous allons maintenant les afficher en HTML.

Afin d'afficher une variable dans un template Django, nous utiliserons des doubles accolades avec le nom de la variable à l'intérieur. Ça ressemble à ceci :

blog/templates/blog/post_list.html

billets

Essayez de faire la même chose avec votre template blog/templates/blog/post_list.html. Ouvrez-le avec votre éditeur et remplacez tout ce qui se trouve entre la seconde balise

jusqu'au troisième

avec la ligne

billets
. Sauvegardez votre fichier et rafraichissez votre page pour voir le résultat :

Figure 13.1

Comme vous pouvez le voir, tout ce que nous avons, c'est ceci :

blog/templates/blog/post_list.html

<QuerySet [<Billet: My second post>, <Billet: My first post>]>

Cela signifie que Django l'interprète comme une liste d'objets. Essayez de vous rappeler comment afficher des listes en Python. Dans un template Django, vous pouvez les écrire de la façon suivante :

blog/templates/blog/post_list.html

{% for billet in billets %}

billet

{% endfor %}

Essayez ceci dans votre template.

Figure 13.2

Ça marche ! Cependant, nous aimerions plutôt afficher les billets à la manière des billets statiques, comme lorsque nous les avions créés dans le chapitre Introduction au HTML. Vous pouvez mixer HTML

et balises de template. Notre body ressemble maintenant à ceci :

blog/templates/blog/post_list.html

<h1>Django Girls Blog</h1>

{% for billet in billets %}

published:

billet.published_date

 <h2>

billet.title

</h2> linebreaksbr

{% endfor %}

Tout ce qui se situe entre {% for %} et {% endfor %} va être répété pour chaque objet présent dans la liste. Rafraichissez votre page :

Figure 13.3

Avez-vous remarqué que nous avons utilisé une notation légèrement différente cette fois-ci (

billet.title

ou

billet.text

) ? Nous accédons aux données associées à chaque champ défini dans notre modèle Billet. De même, le |linebreaksbr nous permet de rediriger le texte des billets à travers un filtre qui convertit automatiquement les fins de lignes en paragraphes.

Ça a marché ? Bravo ! Éloignez vous un peu de votre clavier maintenant : vous avez mérité de faire une pause. :)

Figure 13.4

</html> 13. Le CSS