11. Données dynamiques dans les templates

Nous avons différents morceaux en place : le modèle Billet qui est défini dans le fichier models.py, la vue post_list dans views.py et nous venons de créer notre template. Mais comment allons-nous faire pour faire apparaître nos billets dans notre template HTML ? Car au final, n'est-ce pas le but que nous souhaiterions atteindre ? Nous aimerions prendre du contenu, en l'occurrence notre modèle sauvegardé dans notre base de données, et réussir à joliment l'afficher dans notre template.

C'est à ça que servent les vues : connecter les modèles et les templates. Dans notre vue post_list, nous allons avoir besoin de prendre les modèles dont nous avons besoin et de les passer au template. C'est dans la vue que nous allons décider ce qui va s'afficher (quel modèle) dans un template.

Ok, et sinon, on fait comment ?

Nous allons avoir besoin d'ouvrir le fichier blog/views.py dans l'éditeur de code. Pour l'instant, la vue post_list ressemble à ceci :

blog/views.py

from django.shortcuts import render

def post_list(request):

```
  return render(request, 'blog/post_list.html', {})
```

Est-ce que vous vous souvenez de comment rajouter des morceaux de code écrits dans d'autres fichiers ? Nous en avons parlé dans un chapitre précédent. Nous allons devoir importer notre modèle qui est défini dans le fichier models.py. Pour cela, nous allons ajouter la ligne from .models import Billet de la façon suivante :

blog/views.py

from django.shortcuts import render from .models import Billet

Le point avant models signifie "dossier courant" ou "application courante". Les fichiers views.py et models.py sont dans le même répertoire. Cela signifie que nous pouvons utiliser . suivi par le nom du fichier (sans .py). Ensuite, nous importons le modèle (Billet).

Ok, et après ? Afin de pouvoir aller chercher les véritables billets de blog de notre modèle Billet, nous avons besoin de quelque chose qui s'appelle un QuerySet.

QuerySet

Normalement, ce mot doit vous évoquer quelque chose. Nous en avons un peu parlé dans la section Django ORM (QuerySets).

Maintenant, nous allons nous intéresser à comment publier les billets classés par date de publication (published_date). Ça tombe bien, on a déjà fait ça dans la section sur les QuerySets !

blog/views.py

Billet.objects.filter(published_datelte=timezone.now()).order_by('published_date') Ouvrons donc le fichier blog/views.py dans l'éditeur de code et ajoutons le morceau de code suivant à la fonction def

post_list(request). N'oubliez pas d'ajouter d'abord from django.utils import timezone. blog/views.py from django.shortcuts import render from django.utils import timezone from .models import Billet def post_list(request): billets = Billet.objects.filter(published_datelte=timezone.now()).order_by('published_date')

```
  return render(request, 'blog/post_list.html', {'billets': billets})
```

Il nous manque encore un petit quelque chose : passer notre QuerySet billets à notre template. Pas d'inquiétude, nous allons voir cet aspect prochainement.

Veuillez noter que nous créons une variable pour notre QuerySet : billets. Considérez que c'est le nom de notre QuerySet. À partir de maintenant, nous allons pouvoir faire référence à notre QuerySet en utilisant ce nom.

Dans la fonction render, nous avons un paramètre request, qui désigne tout ce que nous recevons d'un utilisateur par l'intermédiaire d'Internet, et un autre qui signale le fichier template ('blog/post_list.html'). Le dernier paramètre, {}, va nous permettre de glisser des informations que notre template va utiliser. Nous devons donner des noms à ces informations (nous allons rester sur 'billets' pour le moment). :) Ça va ressembler à ça : {'billets': billets}. La partie située avant : est une chaîne de caractères ; vous devez donc l'entourer de guillemets : ''.

Au final, notre fichier blog/views.py doit ressembler à ceci maintenant :

blog/views.py

from django.shortcuts import render from django.utils import timezone from .models import Billet

def post_list(request):

```
  billets =
Billet.objects.filter(published_date__lte=timezone.now()).order_by('publishe
d_date')
  return render(request, 'blog/post_list.html', {'billets': billets})
```

Et voilà, c'est bon ! Nous allons retourner du côté de notre template pour que notre QuerySet puisse s'afficher correctement !

Si vous voulez en savoir plus sur les QuerySets, n'hésitez pas à consulter la documentation officielle du framework : https://docs.djangoproject.com/en/2.2/ref/models/querysets/