クエリセットを理解しよう

Djangoでは、`モデル.objects`とコードを書くことで、データベースに簡単にアクセスすることができます。objectsは様々なメソッドを持っていて、メソッドを使えば指定した条件でデータを取得することができます。

取得したデータのリストのことを`クエリセット（QuerySet）`といいます。

言葉だけだとよくわからないと思うので、実際に使ってみましょう。最初はエディタにコードを書くのではなく、コンソール画面で使ってみます。

`python manage.py shell`というコマンドを打ってください。すると以下のように、コマンドを打てる画面になります。`python`というコマンドでも似たような画面が起動することはすでに環境構築のレッスンで学習済みですが、このコマンドではPythonコードに加えてDjangoのコードも扱えるコマンドプロンプトが立ち上がります。

ここで、Blogモデルを扱いたいのですが、まずはblogsアプリケーション内のmodels.pyファイルからBlogモデルをインポートする必要があります。これをしないと次の作業でエラーが出てしまうので忘れないでください。

ターミナル

```
# blogs/models.pyのBlogモデルをインポート
>>> from blogs.models import Blog
```

インポートできたら、実際にデータベースにアクセスしてBlogsテーブルからデータを取得してみましょう。まずは今データベースに保存されている全てのBlogインスタンスを取得してみます。データを全て取得するためにはall( )メソッドを使います。

ターミナル

```
>>> Blog.objects.all()
```

これで、さっき作ったブログ記事を一覧として表示させることができました。このように取得した一覧のことをクエリセットと呼びます。

objectsは、all( )以外にも様々なメソッドを持っていますので紹介します。

get() 条件に合う１つのインスタンスを取得

```
# idが1のインスタンスを取得
>>> Blog.objects.get(id=1)
```

filter() 条件に合うインスタンスを全て取得

```
# titleフィールドに文字列"Django"を含む記事を全て取得
>>> Blog.objects.filter(title__contains="Django")
```

order_by() 引数に指定したフィールドを基準にして昇順で取得

```
# idの順に取得
>>> Blog.objects.order_by('id')
# フィールド名の前に - をつけると降順で取得できる
>>> Blog.objects.order_by('-id')
```

複数のメソッドを組み合わせて使うこともできます。

filter()とorder_by()を組み合わせて使う

```
# titleフィールドに"Django"を含む記事を"id"の降順で取得
>>> Blog.objects.filter(title__contains="Django").order_by('-id')
```

他にもクエリセットを取得する方法はたくさんあり、あらゆる形で条件を指定してインスタンスを取得することができます。クエリセットの取得方法については、他のチュートリアルやブログでも説明しますので、徐々に知識をつけていきましょう。

次に、create( )メソッドを紹介します。これを使えば、コマンドプロンプトからインスタンスを作成することができます。

create( ) インスタンスを作る

```
# 各フィールドに値を指定してBlogインスタンスを作成
>>> Blog.objects.create(title="コマンドから作られたブログ", text="createメソッドでブログを作ってみました。")
```

日時に関するフィールドは、自動で値が入るので指定する必要はありません。これでコマンドラインからインスタンスを作ることができました！Adminページから確認することもできますし、もちろん先ほど学習したall( )メソッドでも表示されます。

クエリセットを変数に代入することも可能です。また、「インスタンス.フィールド名」とコードを書けば、プロパティを出力することもできます。

id=1のBlogインスタンスを変数blogに代入

```
>>> blog = Blog.objects.get(id=1)
# id=1のBlogインスタンスのタイトルを出力
>>> blog.title
```

複数のBlogインスタンスを代入するときは、変数名をblogsと複数形にするとわかりやすくなります。

全てのBlogインスタンスを変数blogsに代入

```
>>> blogs = Blog.objects.all()
# blogsリストの１番目のインスタンスの本文を表示
```

```
>>> blogs[0].text
```

テンプレートタグを使ってクエリセットをHTMLで表示

ここまでで、データベースに保管されたインスタンスを自分が指定した条件で取得できるようになりました。あとは、取得したインスタンスをHTMLで表示させるだけです！

「アプリケーションで機能を作ろう」のレッスンでも少し紹介しましたが、HTMLで表示する内容はViewで定義することができます。さっきターミナル画面に直接打ったようなコードをViewに書いてクエリセットを取得し、そのクエリセットをHTMLファイルに渡して表示させるという流れになります。

それでは、クエリセットをHTMLで表示させる準備をしていきます。views.pyを開いて、このように書き換えてください。

django_blog/blogs/views.py

```
from django.shortcuts import render
from .models import Blog

def index(request):
    blogs = Blog.objects.order_by('-created_datetime')
    return render(request, 'blogs/index.html', {'blogs': blogs})
```

これまでのコードと比べて、3箇所追加されています。

1箇所目は、２行目でBlogモデルをmodels.pyからインポートしている点です。これはさっきのコンソール画面でも１番最初にやっていますよね。インポートすることでこのファイルでBlogモデルを利用できるようにしています。

2箇所目は、`blogs = Blog.objects.order_by('-created_datetime')`の部分です。さっき覚えたばかりのクエリセットが使われています！データベースに保存されているBlogインスタンスを作成日順に並べたクエリセットを取得し、blogsという変数に代入していますね。

3つ目は、`{'blogs': blogs}`の部分です。これは初めて見るものですね。

このようにrender関数の中で`{ }`と書かれている部分では、テンプレート（HTML）に渡したいデータを自由に定義することができます。Pythonの辞書型と同様に定義しますので、`キー`と`値`を書かなければなりません。今回の場合、キーは`'blogs'`で、値は`変数blogs（つまり、１つ上の行で指定したクエリセット）`となります。この記述により、index.htmlでは、「blogs」と書くだけでクエリセットを表示させることができるようになります。

今回は、たまたまキーと値の名前が一緒でしたが、もちろんキーの名前は自由に定義することができます。注意しなければならないのは、キーは文字列なのでシングルクォートで囲み、値は今回の場合変数なのでクォートでは囲まず変数名をそのまま書きます。

これで、クエリセットのデータをテンプレートに渡すことができるので、実際に表示させてみましょう。HTMLファイルを開いてこのように書いてください。

index.html

```
<h1>ブログサイト</h1>
<p>ここはトップページです。</p>
{{ blogs }}
```

このように、Viewから渡されたデータはキーに指定した文字列を`{{ }}`で囲むことで表示することができます。`{{ }}`のことを`テンプレートタグ`といいます。

コードを書いたら、保存してトップページを表示してみましょう。このように、クエリセットが表示されるはずです。

ブログのクエリセットを表示させることができましたが、これだとBlogインスタンスのリストを表示させているだけです。リストの中の１つ１つの要素をきちんと表示させたいですよね？ここで使うのがPythonのfor文です。以下のコードのように、DjangoではHTMLに直接Pythonコードを書くことができます。このfor文では、変数「blog」に、blogsリストのインスタンスを順番に代入して表示させています。

index.html

```
<h1>ブログサイト</h1>
<p>ここはトップページです。</p>
{% for blog in blogs %}
  {{ blog.title }}
{% endfor %}
```

ここでまた新しいものが出てきましたね。`{% %}`です。これもテンプレートタグというのですが、`{{ }}`と違って、このタグの中身は実際のページには表示されません。

テンプレートタグの使い分けとして、`{{ }}`にはページに表示させたいもの、`{% %}`にはページに表示させたくない処理コードを記述することになります。

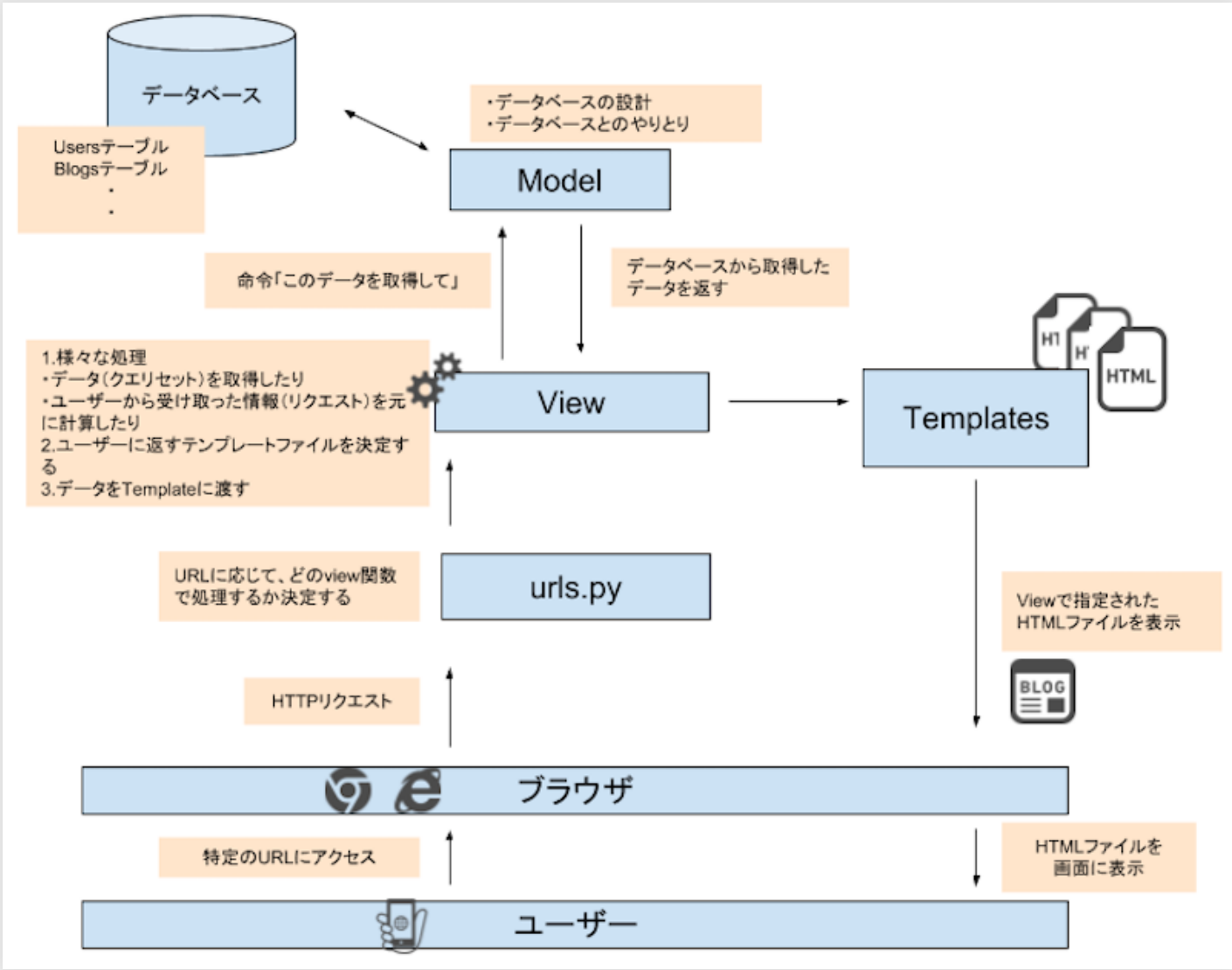
また、通常のPythonコードと違い、for文の終わりを表す`{% endfor %}`というコードが必要になるので、忘れないように注意してください。

上のコードでは、`{{ blog.title }}`のように`「インスタンス.フィールド名」`と書くことで、各インスタンスのタイトルを表示させています。`{{ blog.text }}`と書けば、各記事の本文が表示されるようになります。

なお、フィールド名を省略して、`{{ blog }}`とだけ書くこともできます。この場合、models.pyファイルの`def __str__(self):`で`return self.title`と指定しているので、タイトルが表示されるようになります。

これでクエリセットのデータをちゃんと１つ１つ表示できましたね！お疲れ様でした！

ここまでの流れを図にすると以下のようになります。これが理解できていれば、Djangoの基礎的な流れはもう完璧です！



最後に、今のままだとちょっと見栄えが悪いので、綺麗に表示されるようにHTMLタグで各要素を囲んであげましょう。本当はCSSを使えばもっと綺麗にデザインできますが、このチュートリアルでは簡単化のためにCSSの設定をせず、HTMLタグにstyleを指定することで最低限デザインを整えるだけにします。

ついでに、`{{ blog.text }}`とコードを追加して各記事の本文も表示させます。実際の下のコードには、`{{ blog.text |truncatechars:100 }}`と書いてあります。`|truncatechars:100`の部分は`テンプレートタグフィルター`と呼ばれるもので、条件をつけて表示させることができます。ここで使っているフィルターは表示する文字数を制限するもので、ここでは本文の最初の１００文字だけを表示するように条件をつけています。フィルターは他にもありますので、別の機会に紹介します。

django_blog/blogs/templates/blogs/index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>ブログ管理サイト</title>
</head>
<body>
  <h1 style="text-align: center;">My Blog</h1>
  <div style="width: 70%;margin: 0px auto;">
    <hr />
    {% for blog in blogs %}
    <div>
      <h3>{{ blog.title }}</h3>
      <div>{{ blog.text | truncatechars:100 }}</div>
    </div>
    <hr />
    {% endfor %}
  </div>
</body>
</html>
```

綺麗に表示されましたか？いよいよ次は最後のレッスンです。各記事の詳細ページを作って、このサイトを完成させましょう！