Django - ORM과 쿼리셋

장고를 데이터베이스에 연결, 데이터를 저장하는 방법에 대해서 알아보자.

쿼리셋이란?

쿼리셋(QuerySet)은 전달받은 모델의 객체 목록이다. 쿼리셋은 데이터베이스로부터 데이터를 읽고, 필터를 걸거나 정렬을 할 수 있다.

장고 쉘(Shell)

```
command-line

(myvenv) ~/djangogirls$ python manage.py shell
```

실행하면 아래처럼 인터랙티브쉘 모드처럼 바뀐다.

```
command-line

(InteractiveConsole)
>>>
```

> 파이썬 프롬프트와 비슷하지만 장고 인터랙티브콘솔은 장고의 기능도 사용가능하고 파이썬의 기능도 사용 가능하다.

모든 객체 조회하기

```
>>> Post.objects.all()
Traceback (most recent call last):
      File "<console>", line 1, in <module>
NameError: name 'Post' is not defined
```

> import를 안했기때문에 에러가 발생됨

```
>>> from blog.models import Post
```

> `blog.models`에서 `Post`객체를 불러왔다.

```
>>> Post.objects.all()
<QuerySet [<Post: notice>, <Post: TEST>]>
```

> 예제와 다를수도 있지만 아까 관리자 페이지에서 생성한 글에 관한 객체를 볼수 있다.

객체 생성하기

데이터베이스에 새 글 객체를 저장하는 방법에 대해 알아보자.

```
>>> Post.objects.create(author=me, title='Sample title', text='Test')

NameError: name 'me' is not defined
```

> 작성자로서 `User(사용자)` 모델의 인스턴스를 가져와 전달해줘야 한다.

먼저 User 모델을 불러온다.

```
>>> from django.contrib.auth.models import User
```

아래는 데이터베이스에서 유저가 어떤일을 하는지 알 수 있다.

```
>>> User.objects.all()
Out[6]: <QuerySet [<User: ksyong1234>]>
```

> 슈퍼유저로 등록했던 사용자가 나온다.

이 사용자의 인스턴스(instance)를 가져와 보자. :

```
>>> me = User.objects.get(username='ksyong1234')
```

> `사용자이름(username)`이 `'ksyong1234'`인 User 인스턴스를 받아왔다. 사용자 이름을 바꿨다면, 바뀐 이름을 넣어야한다.

아래처럼 입력하여 게시물을 생성해보자

```
>>> Post.objects.create(author=me, title='Sample title', text='Test')
Out[8]: <Post: Sample title>

>>> Post.objects.all()
Out[9]: <QuerySet [<Post: notice>, <Post: TEST>, <Post: Sample title>]>
```

> `Post.objects.create`로 생성한 게시물이 확인된다.

필터링 하기

쿼리셋의 중요한 기능은 데이터를 필터링하는 것이다.
예를 들어, 우리는 ksyong1234라는 사용자가 작성한 모든 글을 찾고 싶다고 해볼게요. 이런 경우 `Post.objects.all()`에서 `all`대신, `filter`를 사용한다.
쿼리셋 안에 있는 괄호 안에 원하는 조건을 넣어주면 된다.

```
>>> Post.objects.filter(author=me)
Out[10]: <QuerySet [<Post: notice>, <Post: TEST>, <Post: Sample title>]>
```

> 아까 `me`는 `'ksyong1234'`라는 인스턴스를 받아왔기 때문에 `me`는 `ksyong1234`가 된다

제목(title)에 'title'이라는 글자가 들어간 글들만을 뽑아내서 보고 싶다면?

```
>>> Post.objects.filter(title__contains='title')
Out[11]: <QuerySet [<Post: Sample title>]>
```

> **NOTE** `title`와 `contains` 사이에 있는 밑줄(*1이/2개(_)*이다. 장고 ORM은 필드 이름("title")과 연산자과 필터("contains")를 밑줄 2개를 사용해 구분한다. 밑줄 1개만 입력한다면, `FieldError:`

> `Cannot resolve keyword title_contains` 라는 오류가 뜰 것이다.

게시일(published_date)로 과거에 작성한 글을 필터링하면 목록을 불러올 수 있다.

```
>>> from django.utils import timezone
>>> Post.objects.filter(published_date__lte=timezone.now())
Out[13]: <QuerySet [<Post: notice>, <Post: TEST>]>
```

> 아까 관리자페이지에서 작성한 글들만 나온다.
> 게시하려는 게시물의 인스턴스를 얻으면 가능

```
>>> post = Post.objects.get(title="Sample title")
```

그리고 publish메서드를 사용해서 게시한다.

```
>>> post.publish()
```

```
>>> Post.objects.filter(published_date__lte=timezone.now())
Out[16]: <QuerySet [<Post: notice>, <Post: TEST>, <Post: Sample title>]>
```

정렬하기

쿼리셋은 객체 목록의 정렬도 가능하다.

```
>>> Post.objects.order_by('created_date')
Out[17]: <QuerySet [<Post: notice>, <Post: TEST>, <Post: Sample title>]>
```

-을 맨 앞에 붙여주면 내림차순 정렬도 가능하다

```
>>> Post.objects.order_by('-created_date')
Out[18]: <QuerySet [<Post: Sample title>, <Post: TEST>, <Post: notice>]>
```

쿼리셋 연결하기

필터링과 정렬을 연결해서 사용 할 수 있다.

```
>>> Post.objects.filter(published_date__lte=timezone.now()).order_by('publ
ished_date')
Out[19]: <QuerySet [<Post: notice>, <Post: TEST>, <Post: Sample title>]>
```

종료하기

```
>>> exit()
$
```