Django - 모델

객체(Object)란

객체란 속성과 행동을 모아놓은 것.
예를 들어 고양이(Cat)라는 객체를 모델링 한다고 해본다. 이 고양이는 여러 속성을 가지고 있다: 색깔, 나이, 분위기(착한, 나쁜, 졸려 하는), 주인(주인이 사람일 수도 있지만, 길고양이면 주인이 없으니 속성이 빈 값이 될 수 있다.) 등이 될 수 있다.

또 고양이는 특정 행동을 할 수 있다: 야옹야옹하기, 긁기, 또는 먹기 등이 있다. (맛과, 고양이에게 고양이 먹이는 행동하는 객체가 다르다.)

기본적으로 객체지향설계 개념은 현실에 존재하는 것을 속성과 행위로 나타내는 것이다. 여기서 속성은 객체 속성(properties), 행위는 메서드(methods)로 구현된다.

```
Post(게시글)
--------
title(제목)
text(내용)
author(글쓴이)
created_date(작성일)
published_date(게시일)
```

> 블로그 기본 폼

블로그 글로 할 수 있는 것은 어떤 것들이 있을까? 글을 출판하는 `메서드(method)`가 있으면 좋을 듯 하다.
그래서 우리는 `publish`메서드도 만들어야 한다.

장고 모델

장고 안의 모델은 객체의 특별한 종류이다. 이 모델을 저장하면 그 내용이 데이터베이스에 저장되는 것이 특별한 점이다.

애플리케이션 만들기

잘 정돈된 상태에서 시작하기 위해, 프로젝트 내부에 별도의 애플리케이션을 만들어볼 것이다.
애플리케이션을 만들기 위해 콘솔 창(djangogirls 디렉토리에서 manage.py 파일)에서 아래 명령어를 실행.

`Windows`

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

`macOS`

```
(myvenv) ~/djangogirls$ python3 manage.py startapp blog
```

이제 blog디렉터리가 생성되고 그 안에 여러 파일도 같이 들어있는 것을 알 수 있다.

```
djangogirls
├── mysite
|       __init__.py
|       settings.py
|       urls.py
|       wsgi.py
├── manage.py
└── blog
    ├── migrations
    |       __init__.py
    ├── __init__.py
    ├── admin.py
    ├── models.py
    ├── tests.py
    └── views.py
```

애플리케이션을 생성한 후 장고에 사용해야 한다고 알려줘야 한다. 이 역할을 하는 파일이 `mysite/settings.py` 이다.
이 파일 안에서 `INSTALLED_APPS` 를 열어, `]` 바로 위에 `'blog'` 를 추가한다. 최종 결과물은 아래와 같다.

```
mysite/settings.py

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
    'blog',
]
```

블로그 글 모델 만들기

모든 `Model` 객체는 `blog/models.py` 파일에 선언하여 모델을 만든다. 이 파일에 블로그 글 모델도 정의할 것이다.

`blog/models.py` 파일을 열어서 안에 모든 내용을 삭제한 후 아래 코드를 추가

```
blog/models.py

from django.conf import settings
from django.db import models
from django.utils import timezone


class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models
.CASCADE)
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(
            default=timezone.now)
    published_date = models.DateTimeField(
            blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.title
```

> `str` 양 옆에 언더스코어(_) 를 두 개씩 넣었는지 다시 확인. 파이썬에서 자주 사용되는데, "던더(dunder; 더블-언더스코어의 준말)"라고도 불림.

`from` 또는 `import` 로 시작하는 부분은 다른 파일에 있는 것을 추가하라는 뜻
`class Post(models.Model):` 는 모델을 정의하는 코드

- `class` 는 특별한 키워드로, 객체를 정의한다는 것을 알려준다.
- `Post` 는 모델의 이름. (특수문자와 공백 제외한다면) 다른 이름을 붙일 수도 있다. 항상 클래스 이름의 첫 글자는 대문자로 써야 한다.

- `models`은 **Post가 장고 모델임을 의미**. 이 코드 때문에 장고는 Post가 데이터베이스에 저장되어야 한다고 알게 된다.

속성을 정의하는 것
속성을 정의하기 위해, 필드마다 어떤 종류의 데이터 타입을 가지는지를 정해야 함.
텍스트, 숫자, 날짜, 사용자 같은 다른 객체 참조 등이 있다.

- `models.CharField` - 글자 수가 제한된 텍스트를 정의할 때 사용한다. 글 제목같이 짧은 문자열 정보를 저장할 때 사용.
- `models.TextField` - 글자 수에 제한이 없는 긴 텍스트를 위한 속성
- `models.DateTimeField` - 날짜와 시간을 의미
- `models.ForeignKey` - 다른 모델에 대한 링크를 의미

`__str__` 메서드를 보면, `__str__`를 호출하면 Post 모델의 제목 텍스트(string)를 얻게 될 거에요.

데이터베이스에 모델을 위한 테이블 만들기

데이터베이스에 Post 모델을 추가할 것이다. 장고 모델에 (방금 만든) 몇 가지 변화가 생겼다는 걸 알게 해줘야 한다.
커맨드 창에 입력

```
command-line

(myvenv) ~/djangogirls$ python manage.py makemigrations blog
Migrations for 'blog':
  blog/migrations/0001_initial.py:
  - Create model Post
```

장고는 데이터베이스에 지금 반영할수 있도록 마이그레이션 파일(migration file)이라는 것이 있다. 이제 `python manage.py migrate blog` 명령을 실행해, 실제 데이터베이스에 모델 추가를 반영

```
command-line

(myvenv) ~/djangogirls$ python manage.py migrate blog
Operations to perform:
  Apply all migrations: blog
Running migrations:
  Applying blog.0001_initial... OK
```