

## ■ 데이터 관리하는 모델

- ORM (Object Relational Mapping)
- 데이터의 저장/조회를 위해 데이터베이스 명령어인 SQL을 사용해야 하지만 장고의 모델 기능을 사용하면 함수 사용으로 SQL을 대체하는 것이 가능
- ORM의 장점
  - 생산성 증가  
함수(메소드) 사용으로 객체지향적 접근만 고려하면 되므로 생산성 증가
  - 유지보수 편의성 증가  
기존 객체와 독립적인 객체로 코드가 구성되므로 재사용/리팩토링 등 유지보수가 편리함
  - DBMS 의존도 낮아짐  
DBMS에 따라 조금씩 다른 SQL이 사용되지만 ORM은 추상화가 잘 되어 있어서 특정 DBMS에 종속적이지 않음

## ■ 모델 사용을 위한 설정

### ● 데이터베이스 정보 입력

- config/settings.py

```
# SQLite3
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```
# MySQL (MariaDB)
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django',
        'USER': 'root',
        'PASSWORD': '1234',
        'HOST': 'localhost',
        'PORT': 3306
    }
}
```

## ■ 모델 사용을 위한 설정

### ● SQL 로그 출력

- config/settings.py

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
        }
    },
    'loggers': {
        'django.db.backends': {
            'handlers': ['console'],
            'level': 'DEBUG',
        },
    },
}
```

## ■ 모델 클래스

- [App]/models.py 내의 클래스로 작성
- django.db.models.Model 클래스를 상속 받아서 구현

```
from django.db import models

class 모델클래스(models.Model):
    속성1 = models.CharField(max_length=30)
    속성2 = models.IntegerField()
```

- 모델 클래스를 프로젝트에 반영하기 위해 App 등록

```
INSTALLED_APPS = [
    '추가할 App', 프로젝트에 사용될 App명 등록, 목록의 상단에 입력 (가끔 static 오류 발생)
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

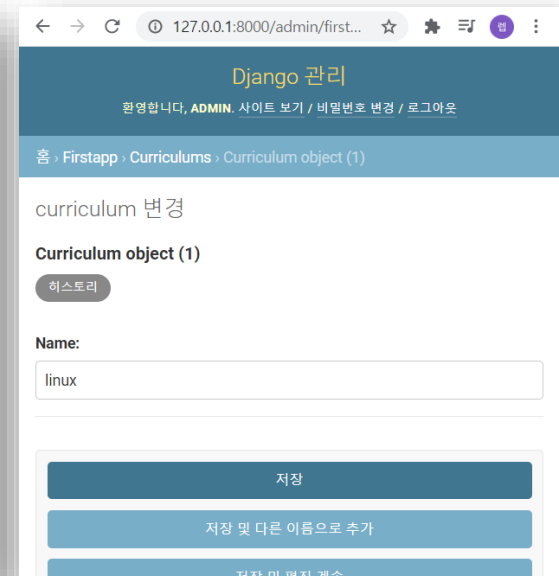
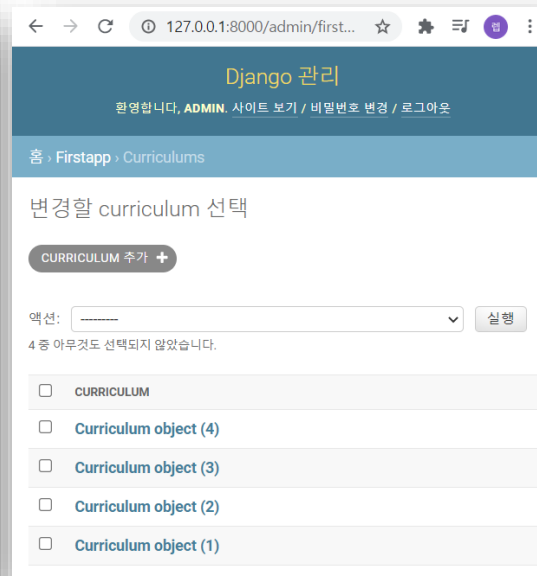
## ■ 모델 클래스

### ● 관리자 사이트에서 데이터 제어를 위해 클래스 등록

- [app]/admin.py

```
from django.contrib import admin
from .models import 모델클래스

admin.site.register(모델클래스)
```



## ■ 모델 클래스

### ● 속성 종류 (Field Type)

- Field Type

```
from django.db import models

class 모델클래스(models.Model):
    속성1 = models.CharField(max_length=30)
    속성2 = models.IntegerField()
```

| 타입                                     | 설명                                   |
|--|--------------------------------------|
| AutoField                              | 자동증가 타입 (기본키와 함께 지정)                 |
| CharField                              | 제한된 문자열 타입, max_length 옵션으로 최대 길이 지정 |
| IntegerField                           | 정수                                   |
| FloatField                             | 실수                                   |
| DateTimeField                          | 날짜 / 시간 (파이썬의 datetime.datetime)     |
| BooleanField                           | 논리 (True / False)                    |
| Text, FilePath, Email, Image, URL, ... |                                      |

<https://docs.djangoproject.com/ko/3.2/ref/models/fields/#field-types>

## ■ 모델 클래스

### ● 속성 종류 (Field Type)

- Field Option

```
from django.db import models

class 모델클래스(models.Model):
    속성1 = models.CharField(max_length=30)
    속성2 = models.IntegerField()
```

| 타입  | 설명                                   |
|---|--------------------------------------|
| null  | 자동증가 타입 (기본키와 함께 지정)                 |
| blank   | 제한된 문자열 타입, max_length 옵션으로 최대 길이 지정 |
| primary_key   | 정수                                   |
| unique  | 실수                                   |
| default   | 날짜 / 시간 (파이썬의 datetime.datetime)     |
| db_column   | 논리 (True / False)                    |
| db_index, db_tablespace, help_text, verbose_name, validators, ... |                                      |

## ■ 모델 클래스

### ● Manager 속성

- 모든 모델은 매니저 속성을 가져야 되며,  
매니저 속성을 정의하지 않았다면 objects 라는 기본 속성을 가짐

모델의 매니저 속성 사용 예) 모델클래스.objects.all()

- 매니저 속성 정의

```
from django.db import models

class SecondManager(models.Manager):
    def get_queryset(self):
        return super(SecondManager, self).get_queryset().filter(name__contains='kim')

class Curriculum(models.Model):
    name = models.CharField(max_length=255)

    objects = models.Manager()
    second_objects = SecondManager()

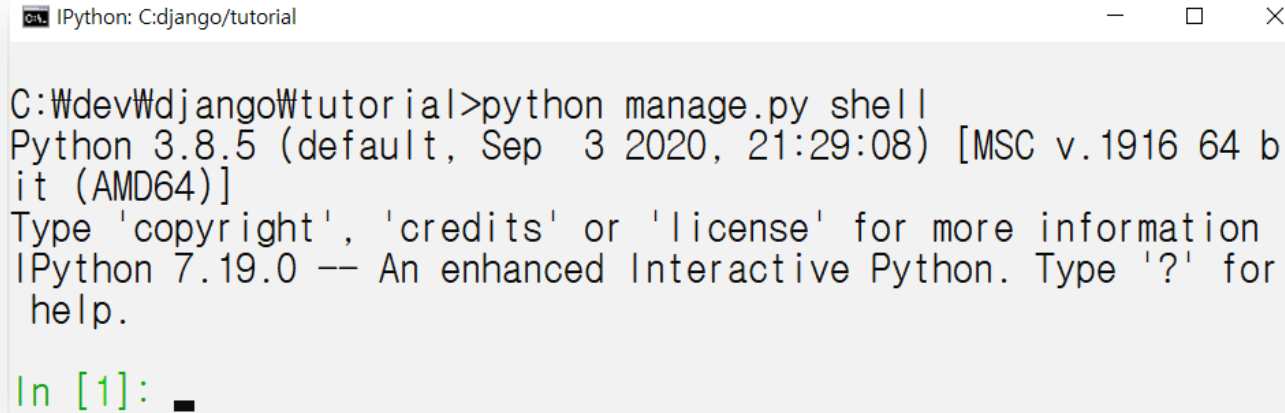
    def __str__(self):
        return self.name
```



## ■ Django Shell을 이용한 모델의 기능 연습

### ● Django Shell 실행

- python manage.py shell

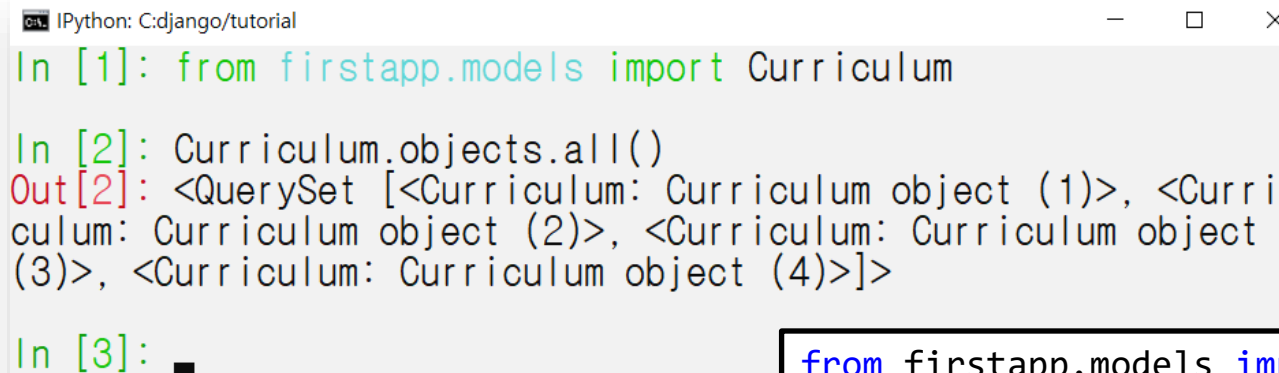


```
IPython: C:\django\tutorial

C:\Wdev\Wdjango\Wtutorial>python manage.py shell
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: _
```

### ● 데이터 전체 조회 : 모델클래스.objects.all()



```
IPython: C:\django\tutorial

In [1]: from firstapp.models import Curriculum

In [2]: Curriculum.objects.all()
Out[2]: <QuerySet [<Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (3)>, <Curriculum: Curriculum object (4)>]>

In [3]: _
```

```
from firstapp.models import Curriculum

Curriculum.objects.all()
```

## ■ Django Shell을 이용한 모델의 기능 연습

### ● 데이터 조회 (1개) : 모델클래스.objects.get(속성=검색어)

```
IPython: C:\django\tutorial
In [11]: Curriculum.objects.get(id=2)
Out[11]: <Curriculum: Curriculum object (2)>

In [12]: Curriculum.objects.get(name='django')
Out[12]: <Curriculum: Curriculum object (4)>

In [13]: Curriculum.objects.get(pk=1)
Out[13]: <Curriculum: Curriculum object (1)>
```

```
Curriculum.objects.get(id=2)
```

```
Curriculum.objects.get(name='django')
```

```
Curriculum.objects.get(pk=1)
```

### ● 데이터 조회 (N개) : 모델클래스.objects.filter(속성=검색어)

```
IPython: C:\django\tutorial
In [15]: Curriculum.objects.filter(name__contains='go')
Out[15]: <QuerySet [<Curriculum: Curriculum object (4)>]>

In [16]: _
```

```
Curriculum.objects.filter(name__contains='go')
```

## ■ Django Shell을 이용한 모델의 기능 연습

- 데이터 제외 (N개) : 모델클래스.objects.exclude(속성=검색어)

```
IPython: C:\django\tutorial
In [17]: Curriculum.objects.exclude(name='python')
Out[17]: <QuerySet [<Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (4)>]>

In [18]: Curriculum.objects.exclude(pk=3)
Out[18]: <QuerySet [<Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (4)>]>
```

`Curriculum.objects.exclude(name='python')`

`Curriculum.objects.exclude(pk=3)`

- 데이터 개수 : 모델클래스.objects.count()

```
IPython: C:\django\tutorial
In [22]: Curriculum.objects.count()
Out[22]: 4

In [23]:
```

`Curriculum.objects.count()`

## ■ Django Shell을 이용한 모델의 기능 연습

### ● 오름차순 정렬 : 모델클래스.objects.order\_by(속성)

```
IPython: C:\django\tutorial
In [25]: Curriculum.objects.filter(name__contains='n').order_by('id')
Out[25]: <QuerySet [<Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (3)>, <Curriculum: Curriculum object (4)>]>

In [26]: Curriculum.objects.filter(name__contains='n').order_by('-id')
Out[26]: <QuerySet [<Curriculum: Curriculum object (4)>, <Curriculum: Curriculum object (3)>, <Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (1)>]>
```

```
Curriculum.objects.filter(name__contains='n').order_by('id')
Curriculum.objects.filter(name__contains='n').order_by('-id')
```

### ● 내림차순 정렬 : 모델클래스.objects.order\_by(-속성)

```
IPython: C:\django\tutorial
In [28]: Curriculum.objects.filter(name__contains='n').order_by('name')
Out[28]: <QuerySet [<Curriculum: Curriculum object (4)>, <Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (3)>]>

In [29]: Curriculum.objects.filter(name__contains='n').order_by('-name')
Out[29]: <QuerySet [<Curriculum: Curriculum object (2)>, <Curriculum: Curriculum object (3)>, <Curriculum: Curriculum object (1)>, <Curriculum: Curriculum object (4)>]>
```

```
Curriculum.objects.filter(name__contains='n').order_by('name')
Curriculum.objects.filter(name__contains='n').order_by('-name')
```

## ■ Django Shell을 이용한 모델의 기능 연습

- 처음 데이터 조회 : 모델클래스.objects.order\_by(속성).first()

```
IPython: C:\django\tutorial
In [30]: Curriculum.objects.order_by('-id').first()
Out[30]: <Curriculum: Curriculum object (4)>
```

Curriculum.objects.order\_by('-id').first()

- 마지막 데이터 조회 : 모델클래스.objects.order\_by(속성).last()

```
IPython: C:\django\tutorial
In [31]: Curriculum.objects.order_by('-id').last()
Out[31]: <Curriculum: Curriculum object (1)>
```

Curriculum.objects.order\_by('-id').last()

## ■ Django Shell을 이용한 모델의 기능 연습

- 데이터 입력 : 모델클래스.objects.create(속성1=값1, 속성2=값2, ...)

```
IPython: C:\django\tutorial
In [32]: Curriculum.objects.create(name='java')
Out[32]: <Curriculum: Curriculum object (5)>
```

```
Curriculum.objects.create(name='java')
```

- 데이터 입력 : 모델 객체 생성 후 save()

```
IPython: C:\django\tutorial
In [34]: c = Curriculum(name='kotlin')
In [35]: c.save()
```

```
c = Curriculum(name='kotlin')
c.save()
```

## ■ Django Shell을 이용한 모델의 기능 연습

- 데이터 수정 : 데이터 조회 후 속성 값을 변경하여 저장

```
IPython: C:\django\tutorial
In [36]: data = Curriculum.objects.get(id=3)
In [37]: data.name = 'javascript'
In [38]: data.save()
In [39]: _
```

```
data = Curriculum.objects.get(id=3)
data.name = 'javascript'
data.save()
```

|   | id  | name   |
|---|-----|--------|
|   | ... | 필터     |
| 1 | 1   | linux  |
| 2 | 2   | python |
| 3 | 3   | python |
| 4 | 4   | django |
| 5 | 5   | java   |
| 6 | 6   | kotlin |

|   | id  | name       |
|---|-----|------------|
|   | ... | 필터         |
| 1 | 1   | linux      |
| 2 | 2   | python     |
| 3 | 3   | javascript |
| 4 | 4   | django     |
| 5 | 5   | java       |
| 6 | 6   | kotlin     |

## ■ Django Shell을 이용한 모델의 기능 연습

### ● 데이터 삭제 : 데이터 조회 후 삭제

```
IPython: C:\django\tutorial
In [39]: data = Curriculum.objects.get(id=6)
In [40]: data.delete()
Out[40]: (1, {'firstapp.Curriculum': 1})
```

```
data = Curriculum.objects.get(id=6)
data.delete()
```

|   | id  | name   |
|---|-----|--------|
|   | ... | 필터     |
| 1 | 1   | linux  |
| 2 | 2   | python |
| 3 | 3   | python |
| 4 | 4   | django |
| 5 | 5   | java   |
| 6 | 6   | kotlin |

|   | id  | name       |
|---|-----|------------|
|   | ... | 필터         |
| 1 | 1   | linux      |
| 2 | 2   | python     |
| 3 | 3   | javascript |
| 4 | 4   | django     |
| 5 | 5   | java       |
|   |     | 삭제         |