

노트북: 첫 번째 노트북

만든 날짜: 2020-10-16 오전 8:22

URL: <https://continuous-development.tistory.com/92?category=805760>

---

Django

## [Django] #3 - Django ORM을 통한 데이터 관리

2020. 9. 18. 19:16 수정 삭제 공개

# ORM이란?

Object Relational Mapping, 객체-관계 매핑로서  
MODEL에 있는 것을 DB에 있는 것으로 똑같이 매핑을 시키는 작업이다.

객체와 관계형 데이터베이스의 데이터를 자동으로 매핑(연결)해주는 것을 말한다.

객체 지향 프로그래밍은 클래스를 사용하고, 관계형 데이터베이스는 테이블을 사용한다.

일반적인 모형에서는 객체 모델과 관계형 모델 간에 불일치가 존재한다.  
Django에서는 ORM을 통해 객체 간의 관계를 바탕으로 SQL을 자동으로 생성하여 불일치를 해결한다.

데이터베이스 데이터 <—매핑—> Object 필드  
객체를 통해 간접적으로 데이터베이스 데이터를 다룬다.

# ORM의 장단점

## 장점

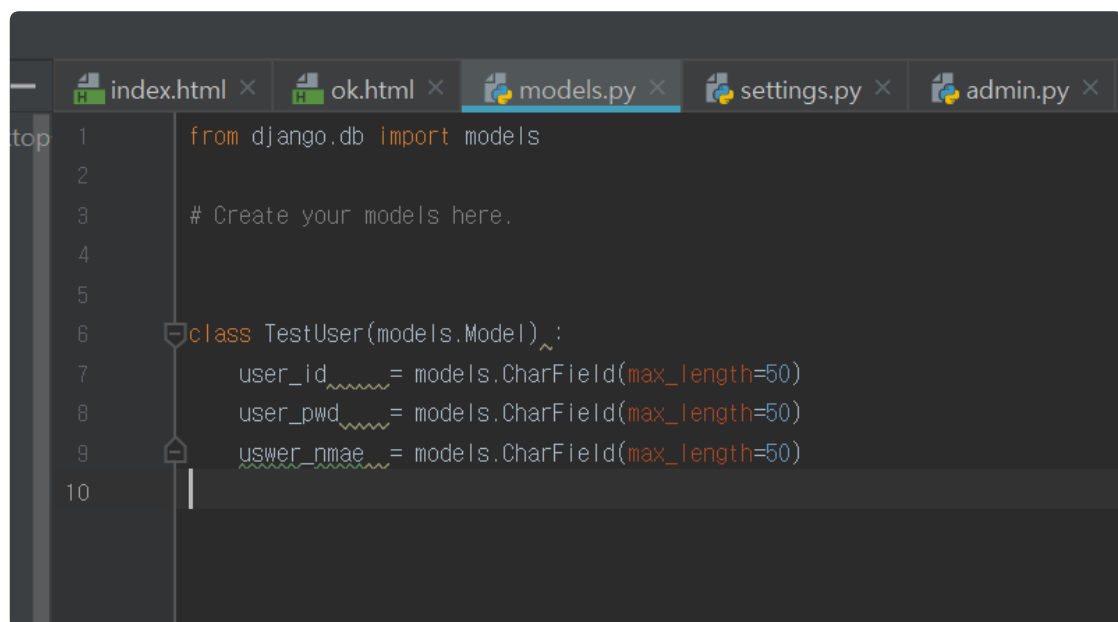
객체 지향적인 코드로 인해 더 직관적이고 비즈니스 로직에 더 집중할 수 있게 도와준다.

재사용 및 유지보수의 편리성이 증가

DMBS에 대한 종속성이 줄어든다

## ORM 구현하기

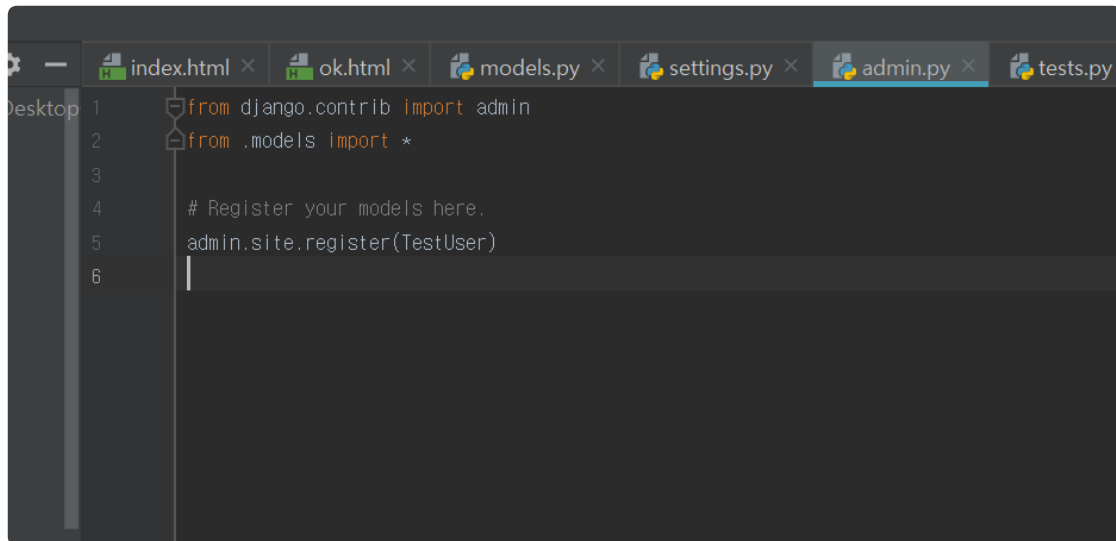
models.py 에서 데이터베이스와 연관된 것을 만들 때는 models를 상속 받고 orm을 통해 데이터베이스와 매핑할 수 있다.



```
1 from django.db import models
2
3 # Create your models here.
4
5
6 class TestUser(models.Model):
7     user_id = models.CharField(max_length=50)
8     user_pwd = models.CharField(max_length=50)
9     user_name = models.CharField(max_length=50)
10
```

위와 같이 models.py 파일에서 TestUser라는 클래스를 만든다.

이 클래스는 DB의 테이블 역할을 한다고 생각하면 된다.



```
1 from django.contrib import admin
2 from .models import *
3
4 # Register your models here.
5 admin.site.register(TestUser)
6
```

그다음 admin 부분에서 `admin.site.register(TestUser)`에 등록함으로써, Django는 디폴트 폼 표현을 구성할 수 있었습니다. 즉 해당 클래스의 형태로 객체를 구현한다. 여기서는 이 객체는 데이터베이스의 하나의 행이라고 생각하면 된다.

## # 모델 마이그레이션

# 사용자 모델 -> DB (테이블로) 만드는 명령어

```
python manage.py makemigrations
```

모델 변경내역 히스토리 관리로서 `migrations`를 한다,  
위와 같이 `make`를 같이서 지금 현재의 모델에 대한 마이그레이션 파일을 만든다.

```
Terminal: Local × +
C:\Users\hwang in beom\Desktop\djangoweb>python manage.py makemigrations
Migrations for 'helloApp':
  helloApp\migrations\0001_initial.py
    - Create model TestUser
```

여기서는 이제 위에서 만든 마이 그레이트 파일을 통해 지금 현재 프로젝트에 마이그레이션을 적용한다.

```
python manage.py migrate
```

그걸 적용하는 부분이다.

```
C:\Users\hwang in beom\Desktop\djangoweb>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, helloApp, sessions
Running migrations:
```

## SUPERUSER 만들기 ( 관리자 )

```
python manage.py createsuperuser
```

우리가 DB를 들어가서 봐야 하는데 이 작업들은 권한이 있는 자만 볼 수 있게 해야 한다. 그 작업을 하기 위해 어드민 계정을 생성하는 작업이다.

```

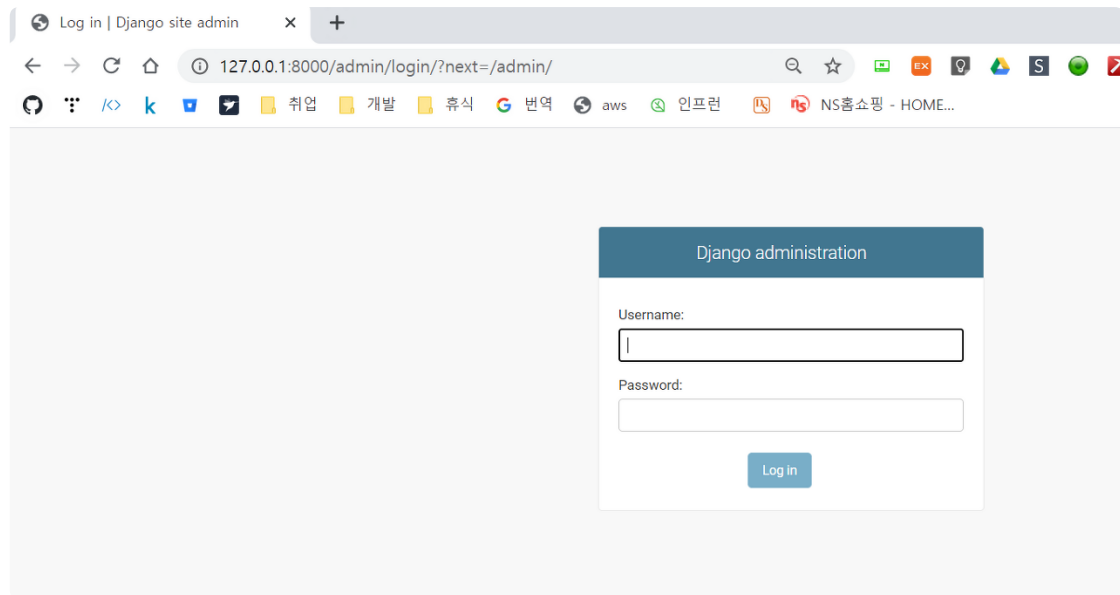
C:\Users\hwang in beom\Desktop\djangoweb>python manage.py createsuperuser
Username (leave blank to use 'hwanginbeom'): hwanginbeom
Email address: rydn2004@naver.com
Password:
Password (again):
This password is too common.
bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

C:\Users\hwang in beom\Desktop\djangoweb>

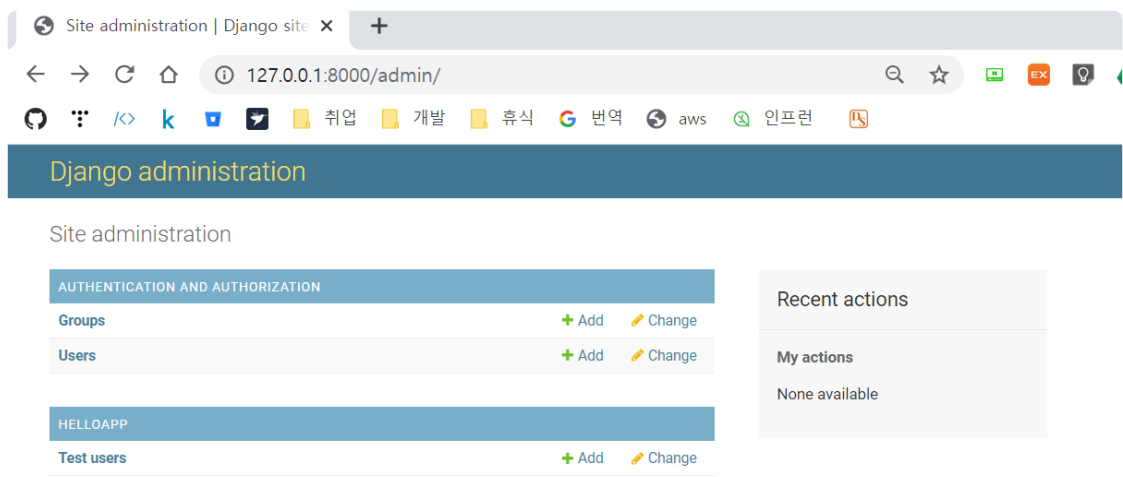
```

여기에 address / id / password를 입력하고 y를 눌러 생성한다.

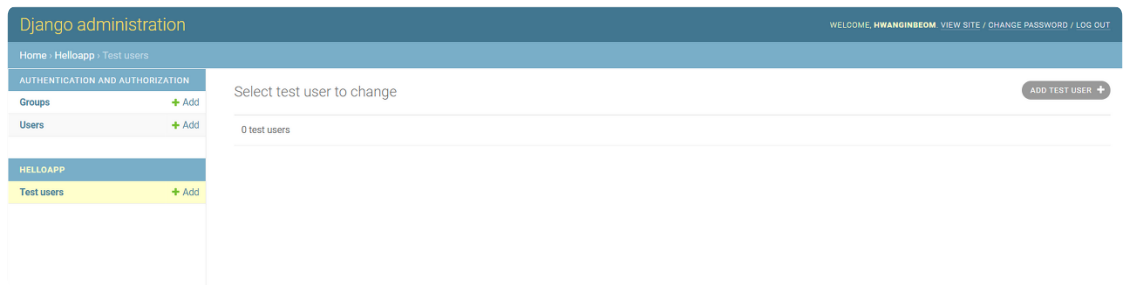
이제 서버를 다 시키고 (python manage.py runserver)  
admin으로 들어가 보자  
localhost/admin으로 들어가면 아래와 같이 창이 뜬다.



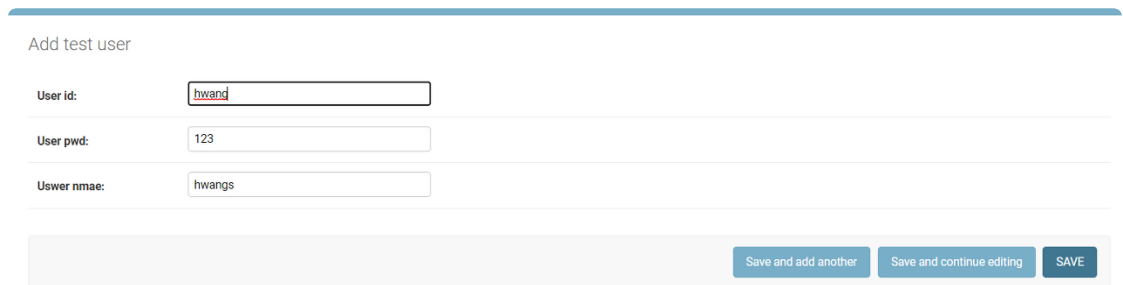
이 곳에서 데이터에 대한 관리를 한다. 아까 만들었던 superuser의 아이디 비밀번호를 치고 들어가면



이렇게 Test users 가 생긴 걸 볼 수 있다. 이 TestUsers는 우리가 model 에서 생성했던 클래스의 형태로 테이블이 생겨난 것이다.



해당 테이블을 클릭하고



add를 눌러 데이터를 추가할 수 있다.

이런 방식으로 브라우저로 db를 관리할 수 있다.

✓ The test user "TestUser object (2)" was added successfully.

Select test user to change ADD TEST USER +

Action:  Go 0 of 2 selected

<input type="checkbox"/>	TEST USER
<input type="checkbox"/>	TestUser object (2)
<input type="checkbox"/>	TestUser object (1)

2 test users

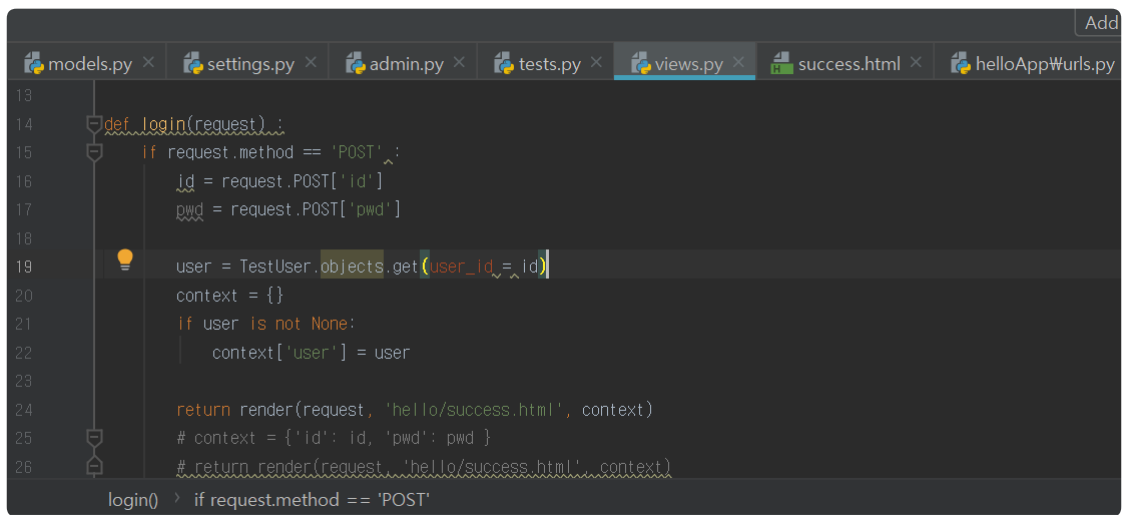
# 데이터 사용하기

view 단에서 user\_id를 가져오는 작업을 하려고 한다.

```
3 from .models import *
4 # Create your views here.
5
6 def index(request):
7     # HttpResponse는 문자, 바이트, 응답, render는 템플릿 통해 응답
8     # return HttpResponse('<div align=center>설리와 함께하는 장고</div>')
9     return render(request, 'hello/index.html')
10
11 def hi(request):
12     context = {'msg': '여기까지 왔습니다!'}
13     return render(request, 'hello/ok.html', context)
14
15 def login(request):
16     if request.method == 'POST':
17         id = request.POST['id']
18         pwd = request.POST['pwd']
19
20         user = TestUser.objects.get(user_id=id)
21         # context = {'id': id, 'pwd': pwd}
22         # return render(request, 'hello/success.html', context)
```

해당 user\_id를 통해 TestUser의 객체를 가져와 user에 넣는다.

그다음 context라는 dict에 넣어 render 할 때 같이 보내준다.



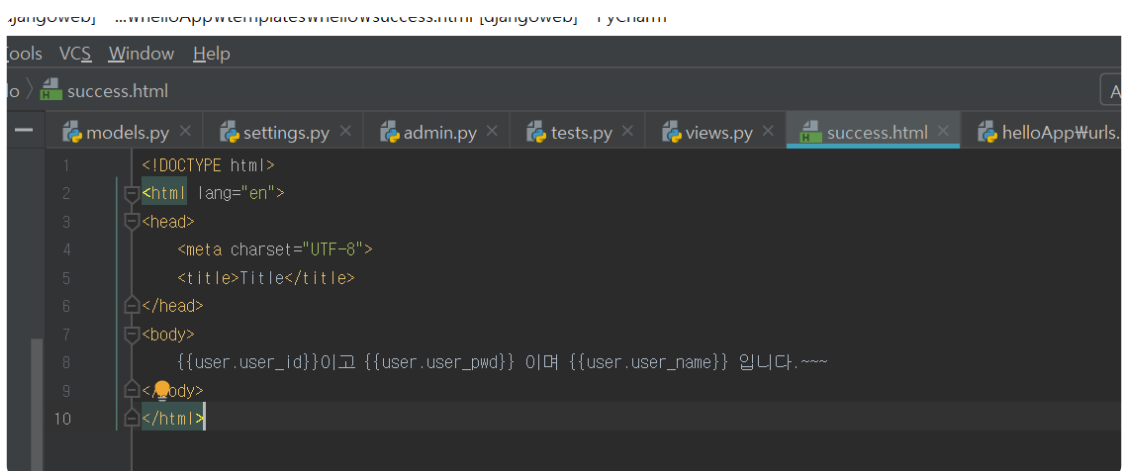
```
13
14 def login(request):
15     if request.method == 'POST':
16         id = request.POST['id']
17         pwd = request.POST['pwd']
18
19         user = TestUser.objects.get(user_id=id)
20         context = {}
21         if user is not None:
22             context['user'] = user
23
24         return render(request, 'hello/success.html', context)
25     # context = {'id': id, 'pwd': pwd }
26     # return render(request, 'hello/success.html', context)
login() > if request.method == 'POST'
```

데이터를 꺼낼 때는 위와 같은 식으로 해당 클래스. object.get이라는 명령어와 괄호 안에 id값을 가져온다.

그 후 context라는 딕셔너리에 user라는 키값을 넣고 user라는 값을 넣어준다.

이제 render 부분에서 request(요청) , template , context를 함께 넣어서 보내준다.

context에 담아서 보내준 값을 html에서 받으려면 이렇게 받으면 된다.

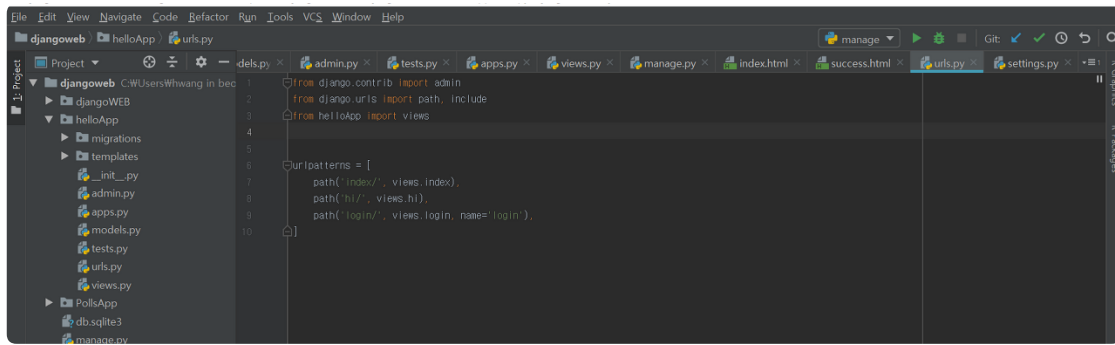


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     {{user.user_id}}이고 {{user.user_pwd}} 이며 {{user.user_name}} 입니다.~~~
9 </body>
10 </html>
```

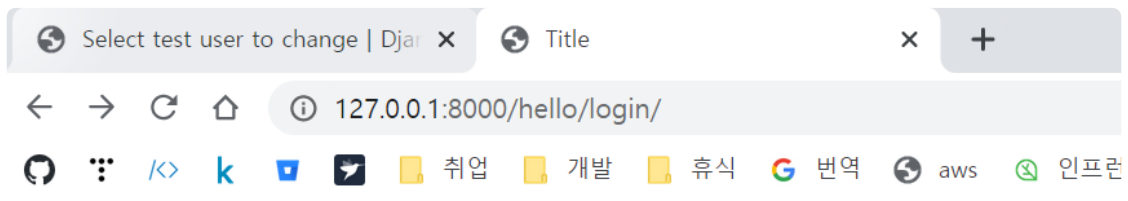
위에 context에 담아놔던 user라는 키값 사용하고 그 안에 들어있는 id, pwd, name을 사용한다.

위에 값에 맞춰 urls 부분도 구현한 뒤





127.0.0.1:8000/hello/login 부분으로 이동한 다음 login을 하게 되면 아래와 같은 창이 뜬다.



test이고 123 이며 inbeom 입니다.~~~

## 'Django' 카테고리의 다른 글

[Django] #6 - Django 를 통한 사용자 등록 구현하기

[Django] #5 - Django 를 통한 static 사용하기

[Django] #4 - Django ORM을 통한 데이터 관리 예제

**[Django] #3 - Django ORM을 통한 데이터 관리**

[Django] #2 - Django 개발 환경 세팅 및 서버 실행까지

[Django] #1 - Django 란?

django ORM

Django 데이터

ORM

장고 orm



꾸까꾸

혼자 끄적끄적하는 블로그 입니다.