API Server 구현하기

**Python + Django Project**

# **Django 실습하기 – Server 실행**

## **Django 프로젝트 뼈대 구성**

$ django-admin startproject mysite # mysite 라는 project 생성

$ python manage.py startapp polls # polls라는 애플리케이션 생성

$ python manage.py migrate # 데이터베이스에 변경사항을 반영

$ python manage.py runserver # 현재까지의 작업을 웹 서버로 확인

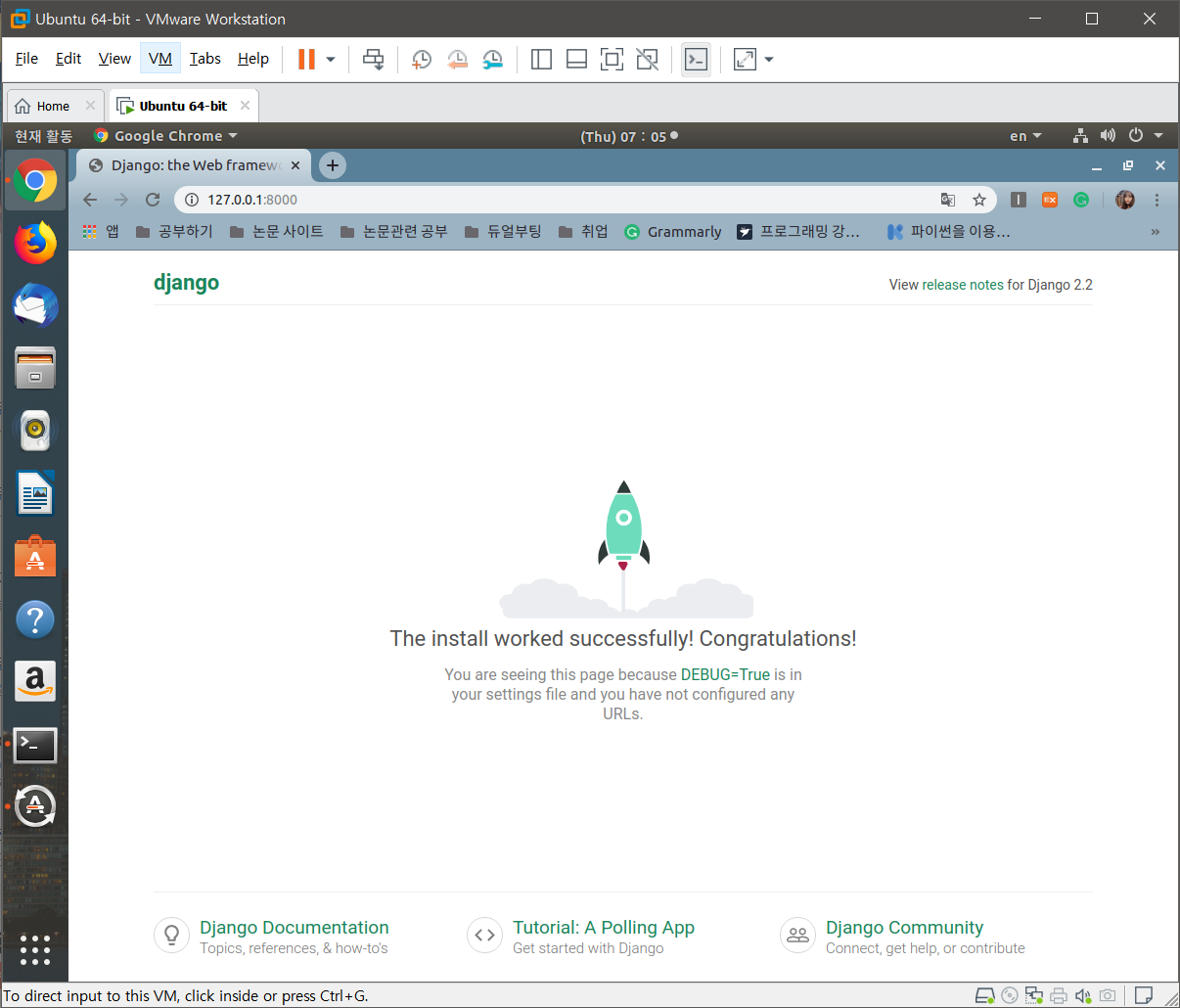
## **서버 실행하기**

$ python manage.py runserver # default(IP주소와 포트번호 지정X): 127.0.0.1, 8000 port

$ python manage.py runserver 8888 # 8888번 포트 이용

$ python manage.py runserver 0.0.0.0:8000 &

## **Web Site에서 확인**



# **References**

\*\* 스타일 보기 : 보기 => 표시 => 탐색창

1. 파이썬 웹 프로그래밍, 김석훈, 한빛미디어
2. Django 공식 문서

<https://docs.djangoproject.com/ko/2.0/topics/testing/overview/#the-test-database>

1. Django REST framework로 간단한 API 만들기

<https://jamanbbo.tistory.com/43>

1. REST API 제대로 알고 사용하기

<https://meetup.toast.com/posts/92>

1. 나만의 웹 크롤러 만들기

<https://beomi.github.io/2017/03/01/HowToMakeWebCrawler-Save-with-Django/>

1. Rest Framework. Python+Django 서버 구축하기

<https://javafa.gitbooks.io/python-django/content/chapter4.html>

1. 기존 DB를 Django로 Migrate 하기

<https://tbang.tistory.com/65>

1. Python Pandas DataFrame 저장 방법 5가지

<https://tariat.tistory.com/583>

1. [pandas] 데이터 선택하기

<https://stickie.tistory.com/80>

1. 10

# **Django Project 생성하기**

## **Django-admin startproject API\_Server**

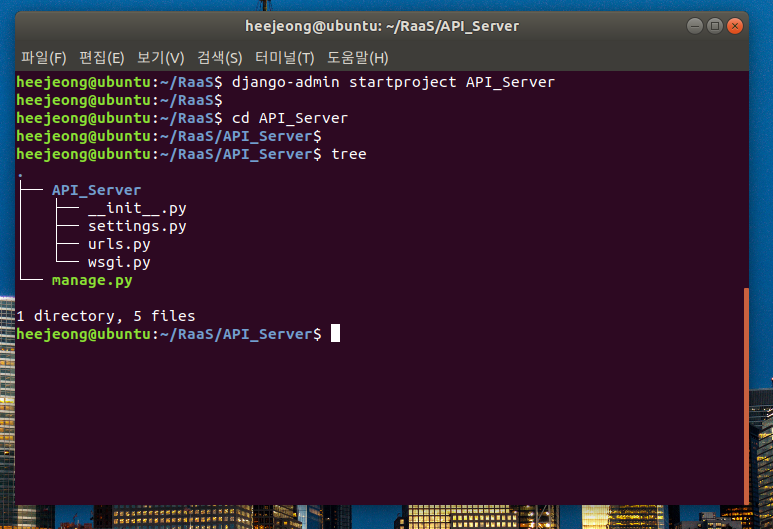
**$ pip install Django** #장고(Django) 설치

**$ django-admin startproject project명** #project 생성하기

**RaaS (Recommendation as a Service)**

**Project명: API\_Server**

**$ Django-admin startproject API\_Server**



## **Project Structure**

Project: API\_Server

**API\_Server Directory 생성 – 4 files created**

1. \_\_init\_\_.py (파이썬 패키지 인식 파일)
2. settings.py: 프로젝트 설정 파일
3. urls.py: 프로젝트 레벨의 URL패턴을 정의하는 최상위 URLConf, 보통은 하위 애플리케이션 마다 urls.py 파일이 있음/
4. wsgi.py: Apache와 같은 상용 웹 서버와 WSGI 규격으로 연동하기 위한 파일

* **manage.py 파일 생성;** 장고의 명령어를 처리하는 파일

# **Django App 만들기**

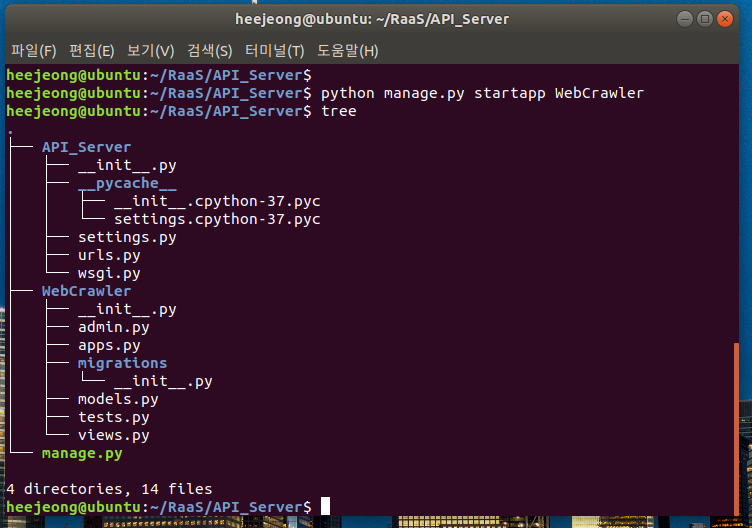
## 앱이란?

* Django는 프로젝트와 그 안의 앱들로 관리함. 앱은 하나의 기능을 담당하는 단위

## 앱 생성: python manage.py startapp Crawler

* 앱은 manage.py 파일을 통해 startapp 라는 명령어로 생성.

**$ python manage.py startapp WebCrawler**



* **프로젝트 안에서 앱을 관리하려면, 프로젝트 폴더 안의 settings.py 파일의 INSTALLED\_APPS에 app 명 추가해줘야 한다**

# **Oracle DB로 바꾸기**

## Settings.py 파일 바꾸기

|  |
| --- |
| DATABASES = {  ‘default’:{  ‘ENGINE’ : ‘django.db.backends.oracle’,  ‘NAME’ : ‘xe’,  ‘USER’ : ‘django’,  ‘PASSWORD’ : ‘onlyroot’  }  } |

## 오류의 원인

1. Version : Oracle 11g XE

Django.db.migrateions.exceiptions.MigrationSchemaMissing: Unable to create the Django\_migrations table (ORA-02000 : missing ALWAYS keyword)

* <https://stackoverflow.com/questions/49229664/configure-the-django-with-oracle-11g-data-base-issue>
* Django 2.x support Oracle 12.1 or higher
* Django downgrade 1.11

django.db.utils.DatabaseError: ORA-00955: name is already used by an existing object

* $ python manage.py makemigrations
* $ python manage.py migrate –fake-initial

File "/home/heejeong/anaconda3/lib/python3.7/site-packages/django/db/backends/oracle/operations.py", line 245, in fetch\_returned\_insert\_id

return int(cursor.\_insert\_id\_var.getvalue())

* <https://stackoverflow.com/questions/52402184/typeerror-when-python-manage-py-migrate>
* cx\_Oracle version 6 down

1. Version : Oracle 18c XE

warning: oracle-database-xe-18c-1.0-1.x86\_64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY

* Oracle is not supported on any Debian distro, period. Read this: <https://docs.oracle.com/en/database/...C-C7319ED934F0>. Only Oracle Linux, Red at or SUSE

1. Oracle 12c SE

* **Oracle 12c Version Install in Ubuntu 18.04LTS**

<https://tutorialforlinux.com/2018/05/09/how-to-install-oracle-12c-r2-database-on-ubuntu-18-04-bionic-64bit-easy-guide/>

USER: oracle, PW: onlyroot

* 설치가 어려움

## 차선책: 구현을 위해 검증된 sqlite3 사용하기

**Django의 기본 데이터베이스인 Sqlite3를 사용한다.**

* Sqlite3 설치하기

$ sudo apt-get install sqlite3

$ sudo apt-get install sqlitebrower # 데이터베이스 탐색기

# **DB 관리자 확인**

**$ cd Project/**

**$ python manage.py createsuperuser**

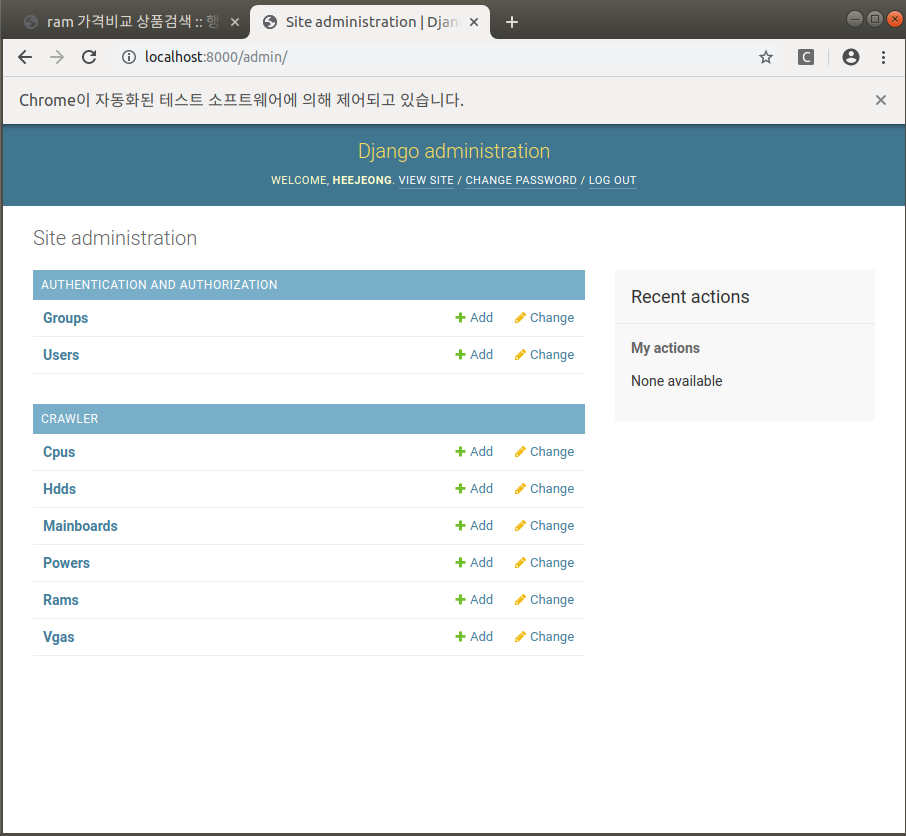
## App에 등록한 DB 보기

* **models.py : 데이터베이스 정의**
* **admin.py : 관리자 사이트에 모델 등; admin.site.register(Model)**

## 서버에서 확인

**$ python manage.py runsever**

**Web: localhost:8000/admin**



# **Rest API의 이해와 설계**

REST는 웹의 창시자(HTTP) 중의 한 사람인 Roy Fielding의 2000년 논문에 의해서 소개되었다.

\*\* 웹의 장점을 최대 활용할 수 있는 네트워크 기반의 아키텍쳐: **Representational safe transfer(REST)**

## **REST의 요소**

* 리소스, 메소드, 메시지

“이름이 Terry인 사용자를 생성한다”

|  |
| --- |
| HTTP POST , <http://myweb/users/>  {  “users”:{  “name”:”terry”  }  } |

생성한다는 의미를 갖는 메서드는 HTTP Post 메서드,

생성하고자 하는 대상이 되는 리소스URI로 표현,

생성하고자 하는 디테일한 내용을 JSON 문서로 표현

## **HTTP 메서드**

REST는 행위에 대한 메서드를 HTTP 메서드를 그대로 사용한다.

REST에서는 CRUD(Create, Read, Update, Delete)에 해당하는 4가지 메서드만 사용한다.

|  |  |  |
| --- | --- | --- |
| 메서드 | 의미 | Idempotent |
| POST | Create | No |
| GET | Select | Yes |
| PUT | Update | Yes |
| DELETE | Delete | Yes |

# **Django RESTful API 서버 구현**

## **REST framework 설치**

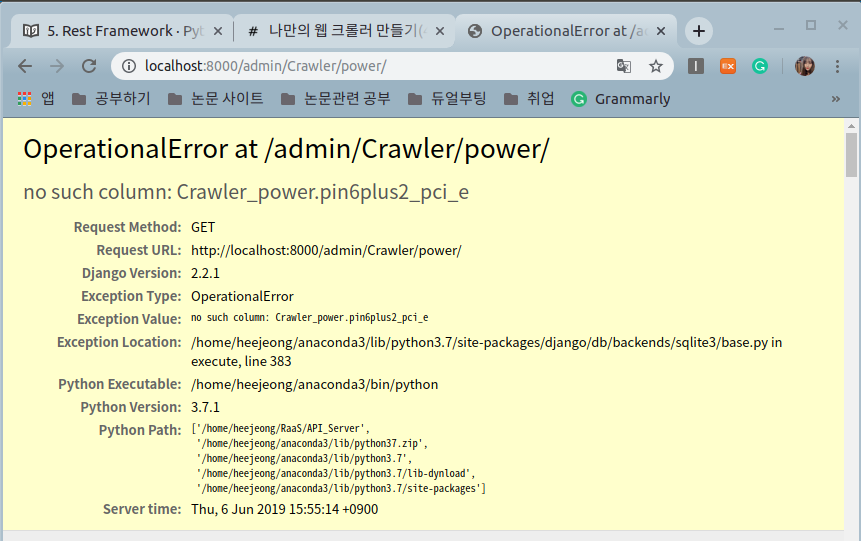
$ pip install djangorestframework

DRF(Django Rest Framework) 라고 부름

## Error 사항 정리

### OperationalError

* 에러창



* 문제점: DB 변경사항 저장 안됨: Column not found
* 해결책: Serializer의 속성 확인

## **Django REST Framework 사용법**

### **Model 생성**

* 생성된 모델은 ORM 기능으로 테이블로 변환됨.

from django.db import models

class Movie(models.Model):

title = models.CharField(max\_length=30) # 제목

genre = models.CharField(max\_length=15) # 장르

year = models.IntegerField() # 제작 년도

def \_\_str\_\_(self):

return self.title

* 모델 생성 후 migration, migrate를 해준다

$ python manage.py makemigrations

$ python manage.py migrate

### **Serializer 생성**

* Serializer란 queryset과 모델 인스턴스와 같은 복잡한 데이터를 json, xml 또는 다른 콘텐츠 유형으로 쉽게 변환 가능
* 받은 데이터의 유효성을 검사한 다음 복잡한 타입으로 형 변환하도록 serialization을 제공

from rest\_framework import serializers

from .models import Movie

**class MovieSerializer(serializers.ModelSerializer):**

**class Meta: model = Movie # 모델 설정**

**fields = ('id','title','genre','year') # 필드 설정**

### **views.py 작성**

* DRF는 사람들이 자주 사용하는 **view 로직을 그룹화 한 viewset**을 제공
* CRUD를 직접 짜지 않아도 이 기능들을 사용 가능해진다.

**from rest\_framework import viewsets**

**from .serializers import MovieSerializer**

from .models import Movie

class MovieViewSet(viewsets.ModelViewSet):

queryset = Movie.objects.all()

serializer\_class = MovieSerializer

### **urls.py 작성**

* DRF는 **url을 자동으로 맵핑 해주는 router를 제공**.
* 앞서 생성한 viewset을 router에 연결하면 자동으로 url 맵팽

from django.conf.urls import url,include

from django.contrib import admin

**from rest\_framework import routers**

**from movies.views import MovieViewSet**

**router = routers.DefaultRouter()**

**router.register('movies',MovieViewSet) # prefix = movies , viewset = MovieViewSet**

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^',include(router.urls)),

]

### **만든 API로 CRUD 요청**

GET /movies/ 영화리스트 조회

POST /movies/ 영화 객체 추가

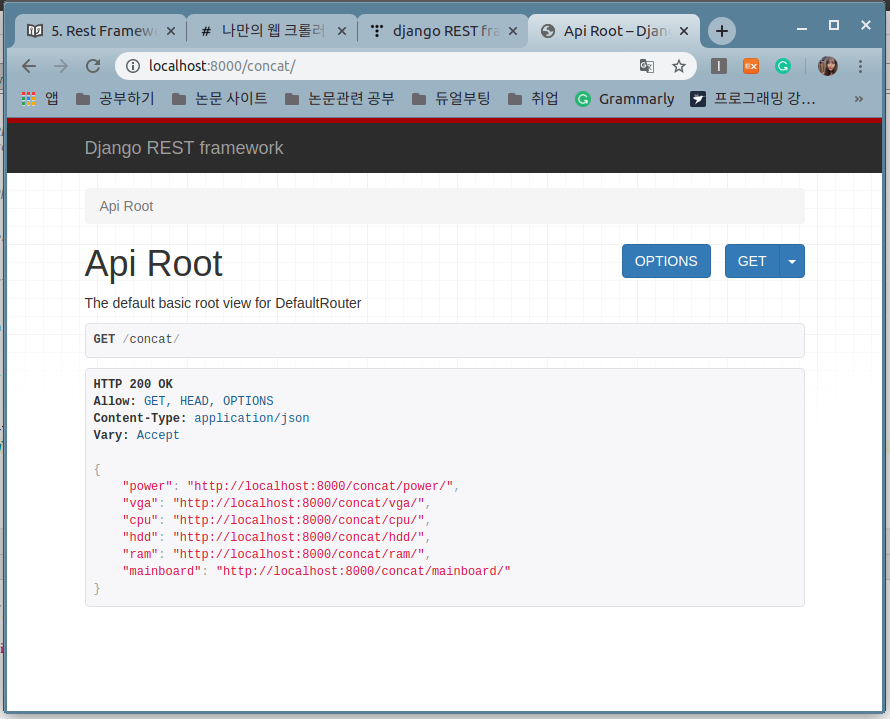
GET /movies/{pk} 영화 객체 조회 (한 개)

PUT / movies/{pk} 영화 객체 수정

DELETE /movies/{pk} 영화 객체 삭제

### **Crawler API 생성**

* concat에 맵핑하도록 해놓았다.



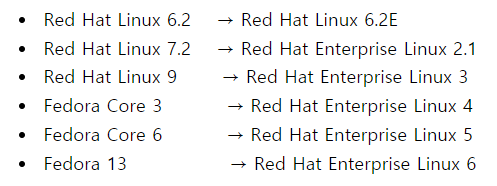
* **소스는 Github에 저장할 것이다.**

# **Appendix A. Fedora, RedHat, CentOS**

<https://ttend.tistory.com/65>

1. **Red Hat vs Fedora**

2003년 레드햇 버전 9까지는 오픈 라이선스로 배포, 그 이후 버전부터는 레드햇 엔터프라이즈 (RHEL, Red Hat Enterprise Linux) 라는 이름으로, 상용으로 변경됨. 레드햇 엔터프라이즈 에디션도 배포 라이선스 자체는 무료지만, 기술지원에 대해 계약기간 동안 비용을 지불해야한다.



(Fedora는 Core 제품군과 Extras 제품군으로 나뉘어 있는데 Fedora 7버전부터 통합되면서 제품명에 Core가 붙지 않는다.)

1. **CentOS**

레드햇 엔터프라이즈 리눅스(RHEL)가 사용으로 전환되었지만, 라이선스는 GNU GPL을 따르고 있기 때문에 레드햇 FTP를 통해 소스 대부분을 공개하고 있다. (자체 프로그램 제외)

이러한 소스에서 레드햇의 상표를 지우고, 다시 컴파일하여 배포판으로 만든 대표적인 제품이 CentOS이다. 이러한 재배포 제품 중 오라클 엔터프라이즈 리눅스도 있다.

1. **Ubuntu**

데비안 계열 Linux; 데비안은 온라인 커뮤니티에서 제작하여 레드햇보다 더 먼저 배포되어 시장을 선점하였음. 이 데비안에서 파생되어진 OS를 데비안 계열이라 부름. 사후 지원과 배포, 내장 유틸들의 성능이 레드햇 계열에 비해 부족하여 오랫동안 밀렸음

우분투; 영국의 캐노니컬이라는 회사에서 만든 배포판으로 쉽고 편한 설치와 이용법.

CentOS vs Ubuntu ? 서버는 CentOS, 데스크탑은 우분투.

대부분의 회사는 CentOS를 사용함.