
Django

웹 프레임워크 설치 방법과 사용법 교안

Django 웹프레임워크

1. Django 웹프레임워크 소개

장고(Django, FAQ 발음으로는 "쟁고"(IPA: ['dʒæŋɡoʊ])^[2])는 파이썬으로 작성된 오픈 소스 웹 애플리케이션 프레임워크로, 모델-뷰-컨트롤러(MVC) 패턴을 따르고 있다.

2. Django 웹프레임워크 특징

장고는 웹 개발에서 번거로운 요소들을 새로 개발할 필요 없이 내장된 기능만을 이용해 빠른 개발을 할 수 있다는 장점이 있다.

구성

장고는 파이썬으로 코딩한 모델을 [관계형 데이터베이스](#)로 구축해주는 모델(Model), HTTP 요청을 처리하는 웹 템플릿 시스템인 뷰(View), URL의 라우팅을 처리하는 URL 컨트롤러 (Controller) 로 구성된 [MVC 디자인 패턴](#)을 따른다.

하지만 전통적인 MVC 디자인 패턴에서 이야기하는 컨트롤러의 기능을 프레임워크를 자체에서 하기 때문에 모델(Model), 템플릿(Template), 뷰(View)로 분류해 MTV 프레임워크라고 보기도 한다.

모델

모델은 데이터에 관한 정보를 담는다. 데이터에 대한 접근, 검증, 작동과 데이터 사이의 관계를 정의하는데, 일반적으로 각각의 모델은 데이터베이스에서 테이블에 해당한다.

장고에서는 모델을 정의할 때 필드의 종류를 지정해줘야 하는데, 이것이 데이터베이스에게 컬럼 타입을 알려주고 HTML 폼으로 표시 될 때의 입력 타입도 내포하는 역할을 한다. 또한 장고의 폼 자동 생성 API 를 이용할 때 데이터 검증에 쓰이기도 한다.

뷰

어떤 데이터가 표시될 것인지를 정의한다. 뷰는 HTTP 응답(response)를 반환해야 하며 응답의 종류는 웹 페이지, 리디렉션, 문서 등의 다양한 형태가 가능하다.

장고에는 자주 사용되는 형태의 뷰를 패턴화하여 추상화 해둔 재사용 가능한 뷰들을 내장해 놓았는데, 이들을 제네릭 뷰(generic view) 라고 하며 원하는 제네릭 뷰를 상속한 클래스 뷰를 생성하여 사용할 수 있다.

템플릿

데이터가 어떻게 표시되는 지를 정의한다. 템플릿은 사용자에게 실제로 보여지는 웹 페이지나 문서를 다룬다.

흔히 HTML 에 기반해서 템플릿을 만들며, HTML 에 동적인 요소를 추가하기 위해 파이썬의 일부 기능을 쓰게 도와주는 장고 템플릿 태그가 존재한다.

내장 애플리케이션

장고에는 아래와 같은 웹 개발에서 자주 쓰이는 애플리케이션이 'contrib' 패키지에 내장되어있다.

- 확장가능한 사용자 인증 시스템

- 동적 관리자 인터페이스
- RSS 또는 아토를 위한 피드 생성
- 사이트맵 생성
- 사이트 간 요청 위조(CSRF), 사이트 간 스크립팅(XSS), SQL 인젝션과 같은 해킹 시도에 대한 보안 대책

3.Django 설치 및 개발 환경 구축

아나콘다를 사용해야 하는 이유

1. 특정 라이브러리가 파이썬 버전에 종속되어 있을 경우

Ex) tf-pose-estimation 라이브러리는 tensorflow 1.10 버전을 지원한다. Tensorflow 1.10 버전은 아직까지 파이썬 3.7 버전의 빌드는 제공하지 않는다.

2. 패키지를 로컬로 분리하고 싶을 경우

일반적으로 pip 명령어는 글로벌로 설치가 된다.

```
pip freeze > requirements.txt
```

위의 명령어로 환경에 저장된 패키지를 한번에 requirements.txt 에 저장할 수 있는데,

패키지를 같이 배포하고 싶을 경우 문제가 생길 수 있다.

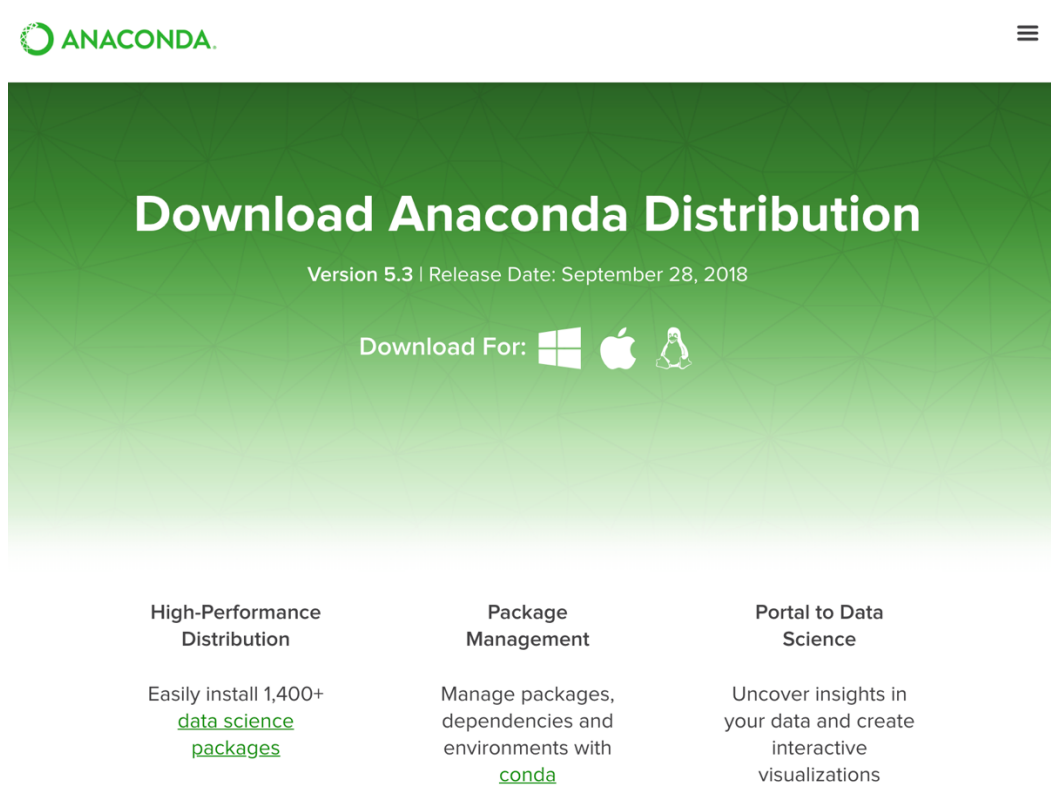
3. 저장공간을 절약하고 싶은 경우

생성한 가상환경 별로 필요 여부에 따라 생성과 삭제가 가능하므로 추후 필요 없는 가상환경을 삭제해 저장공간을 절약할 수 있다.

1. 아나콘다 설치

<https://www.anaconda.com/download/>

위의 링크로 들어가서 파이썬 3.x version 64-Bit (x86) installer 를 다운로드 한다.





2. 장고 설치

```
$ conda activate 생성한 가상 환경 이름
```

우선, 생성한 가상 환경을 활성화하고

conda 명령어가 안 먹힐 경우,

```
$ vi ~/.bashrc
```

로 .bashrc 파일에 들어가서

```
export PATH="/home/<user>/anaconda3/bin:$PATH"
```

위의 경로를 추가해주고 anaconda3 디렉토리가 있는 경로를 본인 ubuntu pc 에 맞게 설치해주면 된다.

```
$ source ~/.bashrc
```

경로 추가 해주고 나면, 터미널에서 source 명령어로 최신 쉘을 로드한다.

다음과 같이 하면, conda 를 적용할 수 있게 된다.

```
$ pip install Django
```

Pip 명령어로 장고 설치를 한다.

```
$ python3 -m Django --version
```

으로 버전 확인을 할 수 있다.

설치가 되었는지 확인하기 위해서 샘플 프로젝트를 생성해보자.

```
$ mkdir Project_test
```

```
$ cd Project_test
```

장고 프로젝트 생성을 위해 디렉토리로 이동한 후,

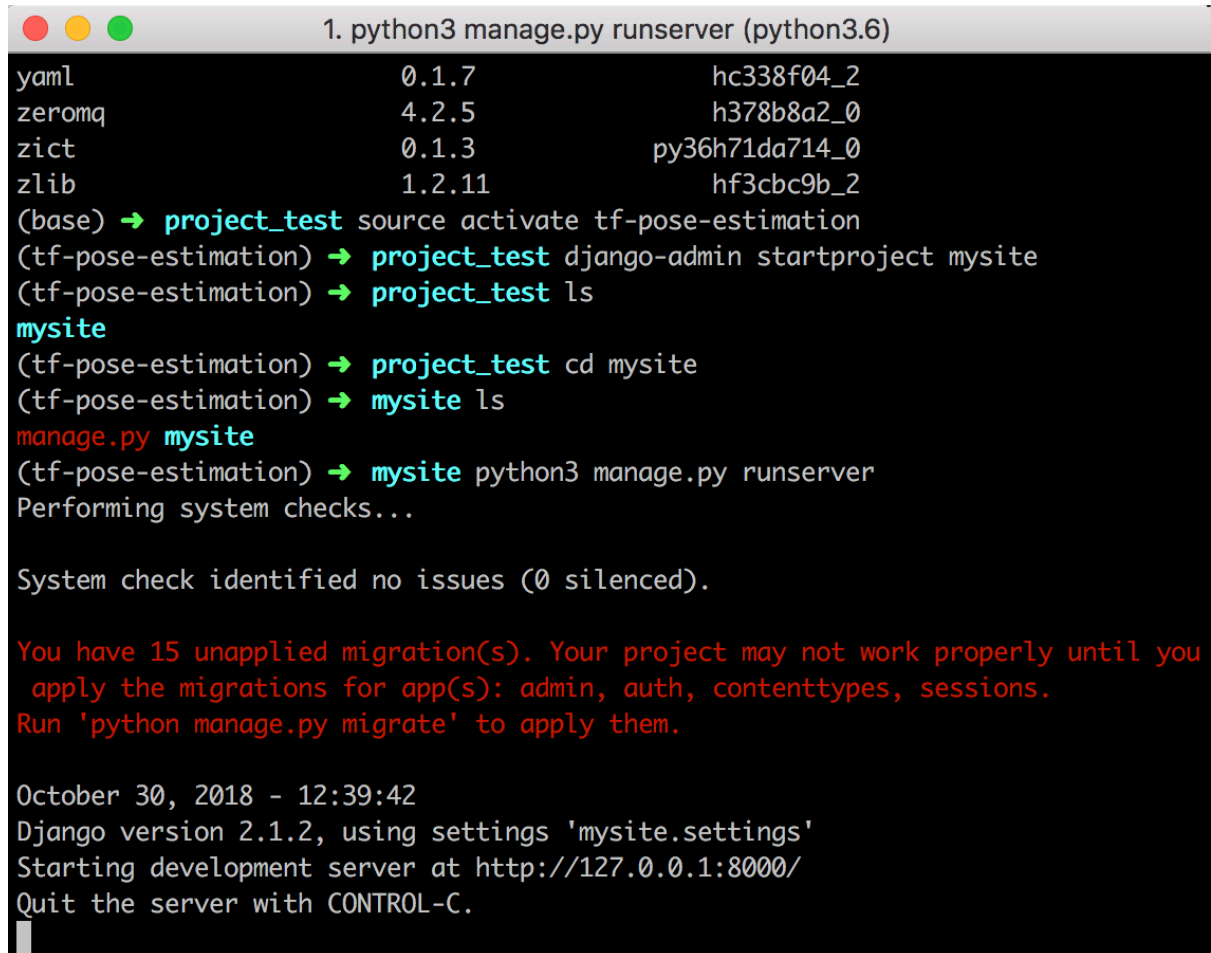
```
$ django-admin startproject mysite
```

```
$ cd mysite
```

django-admin 명령어로 새로운 사이트를 만들 수 있다.

사이트를 만들면, manage.py 스크립트가 생길 것이다.

```
$ python3 manage.py runserver
```



```
1. python3 manage.py runserver (python3.6)
yaml                                0.1.7                                hc338f04_2
zeromq                              4.2.5                                h378b8a2_0
zict                                 0.1.3                                py36h71da714_0
zlib                                 1.2.11                               hf3cbc9b_2
(base) → project_test source activate tf-pose-estimation
(tf-pose-estimation) → project_test django-admin startproject mysite
(tf-pose-estimation) → project_test ls
mysite
(tf-pose-estimation) → project_test cd mysite
(tf-pose-estimation) → mysite ls
manage.py mysite
(tf-pose-estimation) → mysite python3 manage.py runserver
Performing system checks...

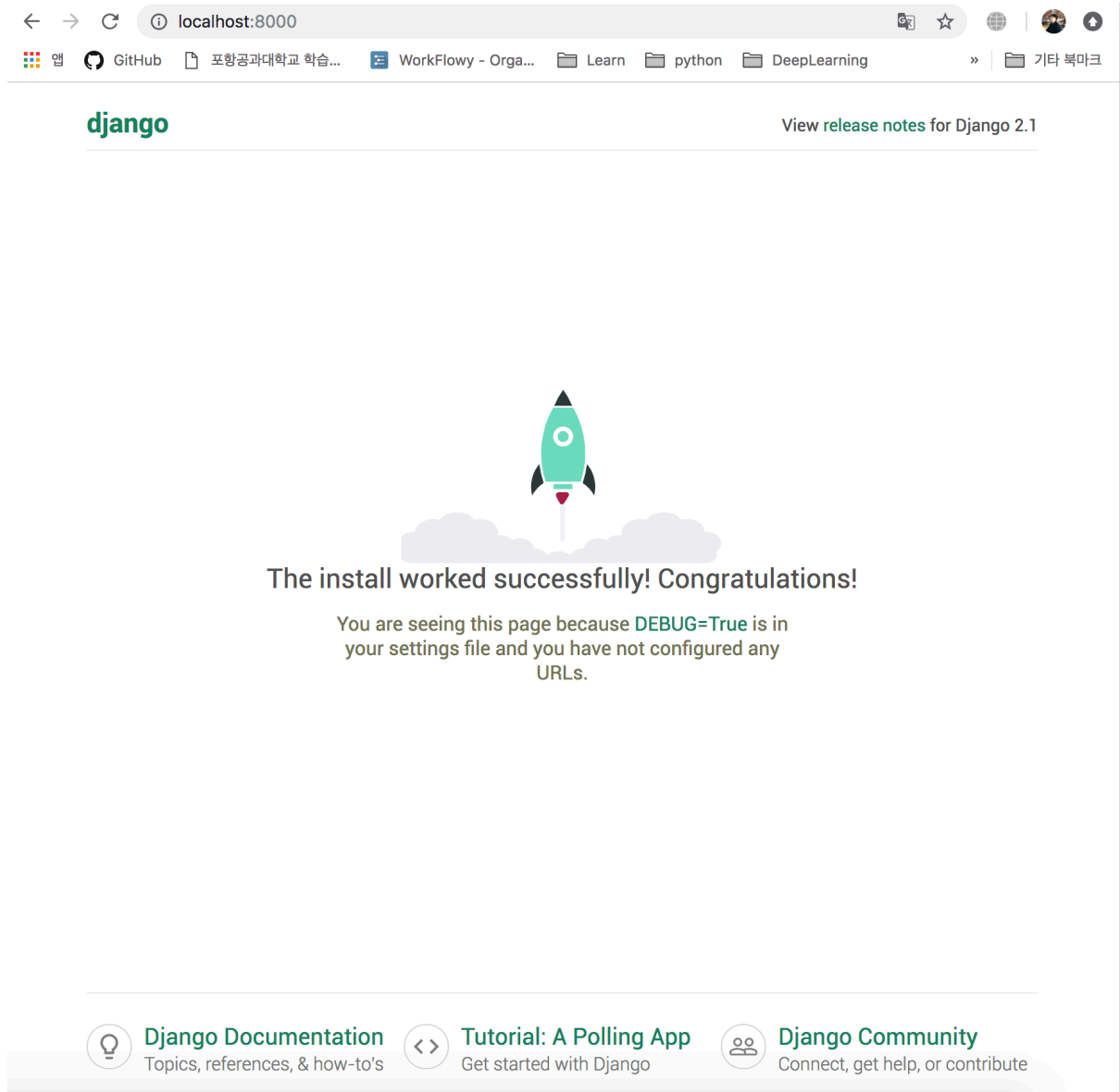
System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you
  apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

October 30, 2018 - 12:39:42
Django version 2.1.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

맥을 사용했기 때문에 터미널의 이미지는 다를 수 있음.

그리고, 웹 브라우저(chrome 또는 firefox)로 <http://127.0.0.1:8000/>
or <http://localhost:8000/> 로 접속하면,



다음과 같은 화면이 뜨게 된다.

이렇게 화면이 나올 경우, 장고는 정상적으로 설치 완료.

AI 교육생분들은 장고를 통해, 프로젝트 UI를 잘 구성해보고자 이 교안을 볼 것이기 때문에 실질적으로 더 나아가본다.

startproject 에서 무엇이 생성되는지 확인해 봅시다.

```
mysite/
  manage.py
  mysite/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

이 파일들은,

- **mysite/** 디렉토리 바깥의 디렉토리는 단순히 프로젝트를 담는 공간입니다. 이 이름은 Django 와 아무 상관이 없으니, 원하는 이름으로 변경하셔도 됩니다.
 - **manage.py**: Django 프로젝트와 다양한 방법으로 상호작용 하는 커맨드라인의 유틸리티 입니다. **manage.py** 에 대한 자세한 정보는 [django-admin and manage.py](#) 에서 확인할 수 있습니다.
 - **mysite/** 디렉토리 내부에는 project 를 위한 실제 Python 패키지들이 저장됩니다. 이 디렉토리 내의 이름을 이용하여, (**mysite.urls** 와 같은 식으로) project 어디서나 Python 패키지들을 import 할 수 있습니다.
 - **mysite/__init__.py**: Python 으로 하여금 이 디렉토리를 패키지 처럼 다루라고 알려 주는 용도의 단순한 빈 파일입니다. Python 초심자라면, Python 공식 홈페이지의 [more about packages](#) 를 읽어보십시오.
 - **mysite/settings.py**: 현재 Django project 의 환경/구성을 저장합니다. [Django settings](#) 에서 환경 설정이 어떻게 동작하는지 확인할 수 있습니다.
 - **mysite/urls.py**: 현재 Django project 의 URL 선언을 저장합니다. Django 로 작성된 사이트의 "목차" 라고 할 수 있습니다. [URL dispatcher](#) 에서 URL 에 대한 자세한 내용을 읽어보세요.
 - **mysite/wsgi.py**: 현재 project 를 서비스 하기 위한 WSGI 호환 웹 서버의 진입점 입니다. [How to deploy with WSGI](#) 를 읽어보세요.
-

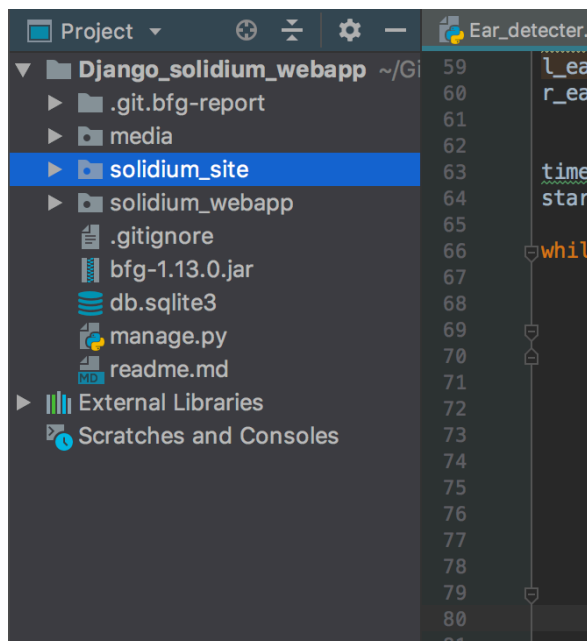
장고 Docs 엔 이렇게 설명 되어있다.

<https://docs.djangoproject.com/ko/2.1/intro/tutorial01/>

하지만, 프로젝트에서 실제 사용하는 부분들을 간략하게 경험을 비롯해 설명해보면,

startproject 에서 무엇이 생성되는지 확인해 봅시다.

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py
```



왼쪽에 solidium_site 가
하위 mysite 디렉토리와 같고 이
안에 __init.py, settings.py,
urls.py, wsgi.py 가 있다.

media 는 미디어 파일을 저장하는
디렉토리인데, settings.py 에

media 의 경로를 저장하고

Settings.py

```
127
128     #이미지 파일을 저장할 폴더 위치 저장
129     MEDIA_URL = '/media/'
130     MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
131     MEDIA_ROOT_URL = '.'
132
```

Webapp 안에 존재하는 views.py 같은 로직 처리 부분에서 settings.py 를 import 하여 media 디렉토리로 접근한다.

간략한 settings.py 에 관한 설명을 마치고,

```
$ python3 manage.py startapp myapp
```

위의 명령어로 웹 애플리케이션 디렉토리를 생성해보도록 한다.

```
(tf-pose-estimation) → mysite python3 manage.py startapp myapp
(tf-pose-estimation) → mysite ls
db.sqlite3 manage.py myapp mysite
(tf-pose-estimation) → mysite cd myapp
(tf-pose-estimation) → myapp ls
__init__.py apps.py models.py views.py
admin.py migrations tests.py
(tf-pose-estimation) → myapp
```

```
myapp/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

웹 애플리케이션 디렉토리가 생성되었다.

mysite/settings.py 에서

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

INSTALLED_APPS 리스트 안에 myapp 을 추가해준다.

(마지막에 , 써주는 것을 잊지말자 !)

이 디렉토리가 실제 프로젝트에서 구현하고자 하는 앱 애플리케이션에서 가장 중요한 부분.

그리고 127.0.0.1:8000/으로 들어오는 모든 접속 요청을

myapp.urls 로 전송하기 위해서는,

두가지를 해야한다.

1. mysite 안의 urls.py 에 다음과 같이 코드를 삽입한다.

mysite/urls.py

```
from django.conf.urls import include, url  
from django.contrib import admin  
  
urlpatterns = [  
    url(r'^admin/', admin.site.urls),  
    url(r'', include('myapp.urls')),  
]
```

2. myapp 디렉토리에서 urls.py 를 생성하고 다음과 같이 코드를 삽입한다.

```
$ touch urls.py
```

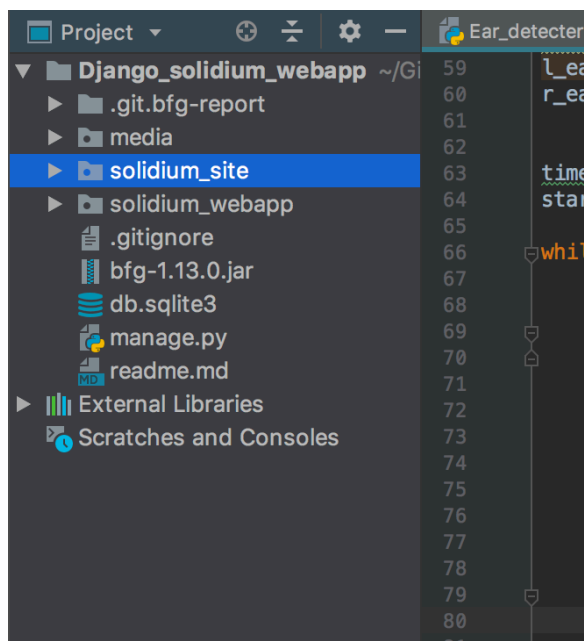
myapp/urls.py

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.main, name='main'),
]
```

<http://127.0.0.1:8000/> 로 접속할 경우, 위의 views.py 에 있는 main()을 호출한다는 의미이다.

이 2 개를 하고 나면, site 로 접속하는 모든 요청이 mysite.urls 에서 myapp.urls 로 전송되고, myapp.urls 의 urlpatterns 리스트에 있는 url 을 통해 views.py 에 있는 클래스나 함수를 호출하게 된다.



실제 장고에서 사용하는 MTV 패턴(자바 Spring 이나 Jsp 의 MVC 패턴과 동일한 개념)은 웹 애플리케이션 디렉토리 안에서 충분히 이루어진다.

왼쪽의 solidium_webapp 부분이 myapp 과 같은 부분이다.

```
models.py
tests.py
views.py
```

views.py 는 애플리케이션의 로직이 되는 부분이다.

Ex) url 에서 main()이라는 함수를 호출하면 로직이 되는 views.py 에서 models.py 에서 만든 모델에서 필요한 정보를 받아와 template 에 전달하는 역할을 한다.

url 에서 호출된 함수를 views.py 에서 간단하게 다음과 같이 만든다.

myapp/views.py

```
from django.shortcuts import render

# Create your views here.

def main(request):
    return render(request, 'main.html', {})
```

templates 디렉토리는 myapp 디렉토리안에 생성하고,

그 안에 main.html 이라는 새파일을 만들어
myapp/templates/main.html

```
<html>
  <p>Hi Postech!</p>
  <p>It works!</p>
</html>
```

위의 내용을 넣으면 된다.

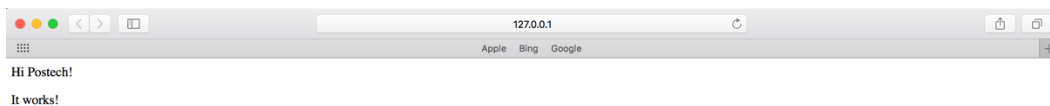
그리고 파이참 또는 터미널 상에서 코드를 run 하면,

```
/Users/donghoon/anaconda3/envs/tf-pose-estimation/bin/python3.6 /Users/donghoon/pro
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you ap
Run 'python manage.py migrate' to apply them.
October 30, 2018 - 14:29:24
Django version 2.1.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

정상적으로 실행 되고,



온전하게 실행되는 것을 볼 수 있다.

Django 를 쉽고 단계적으로 가르쳐주는 페이지는

<https://tutorial.djangogirls.org/ko/deploy/>가 있고,

<https://docs.djangoproject.com/ko/2.1/intro/tutorial01/>도

포괄적인 Django 의 구조를 설명하며 튜토리얼을 진행한다.