

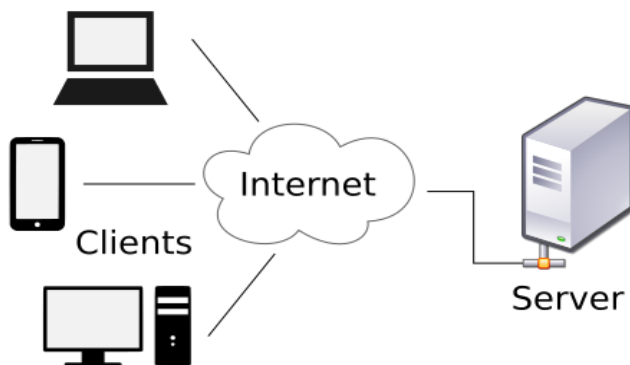
[Django(장고) 프로그래밍]

웹의 기본 이해

웹은 보통 **월드 와이드 웹(WWW)**이라고도 하며, 인터넷이라는 연결을 통해 여러 컴퓨터가 연결되어 서로 정보를 나누는 연결망들을 의미합니다. 이 웹이라는 것을 어떤 관점에서 보냐에 따라서 다음과 같은 개념이 생긴다.

1. 정보를 제공하는 컴퓨터와 정보를 받아가는 컴퓨터

서버	정보를 제공하기 위해 고정된 주소(도메인, 고정 IP 등을 차지하며 방문하는 컴퓨터들에게 필요한 정보를 제공한다. 보통의 우리가 접속하는 웹 사이트들이 이에 해당한다.
클라이언트	정보를 제공받기 위해 서버를 찾아 접속하는 컴퓨터입니다. 보통은 고정된 주소 없이 인터넷 연결을 통해 서버에 접근한다. 일반적으로 우리가 웹 브라우저를 위해 쓰고 있는 컴퓨터들이 이에 해당한다.



2. 웹 프로그램의 구조 별 개념

프론트엔드	프론트엔드는 보통 사용자들이 보는 화면의 모습을 결정한다. HTML, CSS, JavaScript 프로그래밍으로 보통 구성되고 클라이언트의 웹 브라우저에서 코드가 실행되거나 그려진다.
백엔드	백엔드는 사용자가 접속하면 그 에 맞는 데이터를 보내주기 위해 여러가지 처리를 하는 부분을 담당하는 로직을 구성한다. 우리가 배울 Django 기반의 Python 프로그래밍이나 데이터베이스 등이 여기에 속한다. 프론트엔드 코드 조각들을 여기서 가지고 있다가, 사용자에게 적절하게 처리해서 보내주는 역할도 한다. 보통 서버에서 프로그램이 실행된다.(CGI, PHP, ASP, Servlet, JSP, NodeJS, Spring MVC)

3. 서버-클라이언트 간 통신 방향 별 개념

요청(Request)	클라이언트가 서버로 원하는 정보를 받기 위해 필요한 정보를 보내는 과정을 요청 이라고 한다. 보통은 클라이언트의 IP와 접속하고 있는 브라우저 프로그램이나 모바일 여부, 요청하는 URL과 방식, 그리고 필요한 경우 서버가 처리하기 위한 데이터들을 보낸다.
응답(Response)	요청을 받은 서버가 받은 데이터를 처리하여 사용자에게 정보를 내려주는 것을 응답 이라고 한다. 응답에는 여러가지 형태의 데이터가 내려갈 수 있는데, 우리가 보통 웹 사이트에 접속했을 때 보여지는 화면들은 응답에 HTML, CSS, JavaScript 코드가 포함되어 그걸 웹 브라우저가 해석하고 실행하여 화면에 그려지게 되는 케이스가 속한다.(JSON, XML...)

HTTP 프로토콜(통신규약)

HTTP(Hypertext Transfer Protocol)는 웹을 개발하는 사람이라면 누구나 다 알아야 하는 통신 프로토콜이다. 프로토콜이란 상호 간에 정의한 규칙을 의미하며 특정 기기 간에 데이터를 주고받기 위해 정의되었다. 통신 프로토콜을 쉽게 풀어보면 “나는 이렇게 줄 테니 넌 이렇게 받고 난 너가 준거 그렇게 받을게” 를 의미한다.

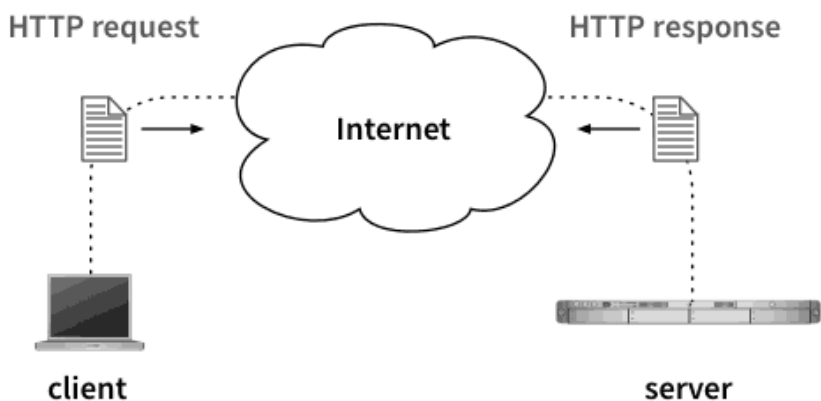
웹에서는 브라우저와 서버 간에 데이터를 주고받기 위한 방식으로 HTTP 프로토콜을 사용하고 있다.

- HTTP 프로토콜의 특징

HTTP 프로토콜은 상태가 없는(stateless) 프로토콜입니다. 여기서 ‘상태가 없다’ 라는 말은 데이터를 주고 받기 위한 각각의 데이터 요청이 서로 독립적으로 관리가 된다는 말이다. 좀 더 쉽게 말해서 이전 데이터 요청과 다음 데이터 요청이 서로 관련이 없다는 것이다. 이러한 특징 덕분에 서버는 세션과 같은 별도의 추가 정보를 관리하지 않아도 되고, 다수의 요청 처리 및 서버의 부하를 줄일 수 있는 성능 상의 이점이 생긴다. HTTP 프로토콜은 일반적으로 TCP/IP 통신 위에서 동작하며 기본 포트는 80번이다.

- HTTP Request & HTTP Response

HTTP 프로토콜로 데이터를 주고받기 위해서는 아래와 같이 요청(Request)을 보내고 응답(Response)을 받아야 한다.



그리고 요청과 응답을 이해하기 위해서는 먼저 클라이언트(Client)와 서버(Server)를 이해해야 한다. 클라이언트란 요청을 보내는 쪽을 의미하며 웹 관점에서는 브라우저를 의미한다. 서버란 요청을 받는 쪽을 의미하며 데이터를 보내주는 원격지의 컴퓨터를 의미한다.

- URL

URL(Uniform Resource Locators)은 개발자가 아니더라도 이미 우리에게 익숙한 용어이다. 서버에 자원을 요청하기 위해 입력하는 영문 주소이며 URL 구조는 아래와 같다.(resource path 가 생략되면 기본파일을 달라는 의미로 해석 - index.html)



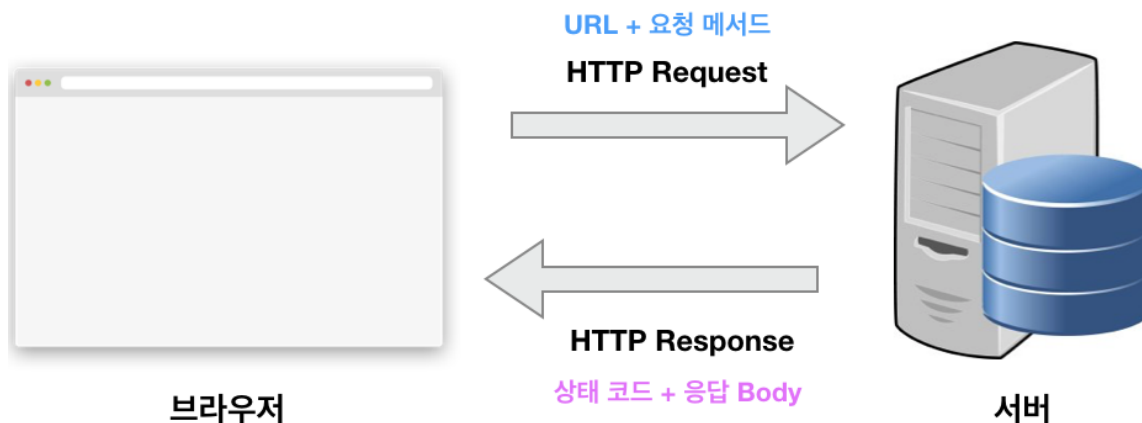
- HTTP 요청 메서드

앞에서 살펴본 URL을 이용하면 서버에 특정 데이터를 요청할 수 있다. 여기서 요청하는 데이터에 특정 동작을 수행하고 싶으면 HTTP 요청 메서드(HTTP Request Methods)를 이용한다.(GET 이 기본)

GET : 존재하는 자원에 대한 요청	PUT : 존재하는 자원에 대한 변경
POST : 새로운 자원을 생성	DELETE : 존재하는 자원에 대한 삭제

- HTTP 상태 코드(200, 404, 500)

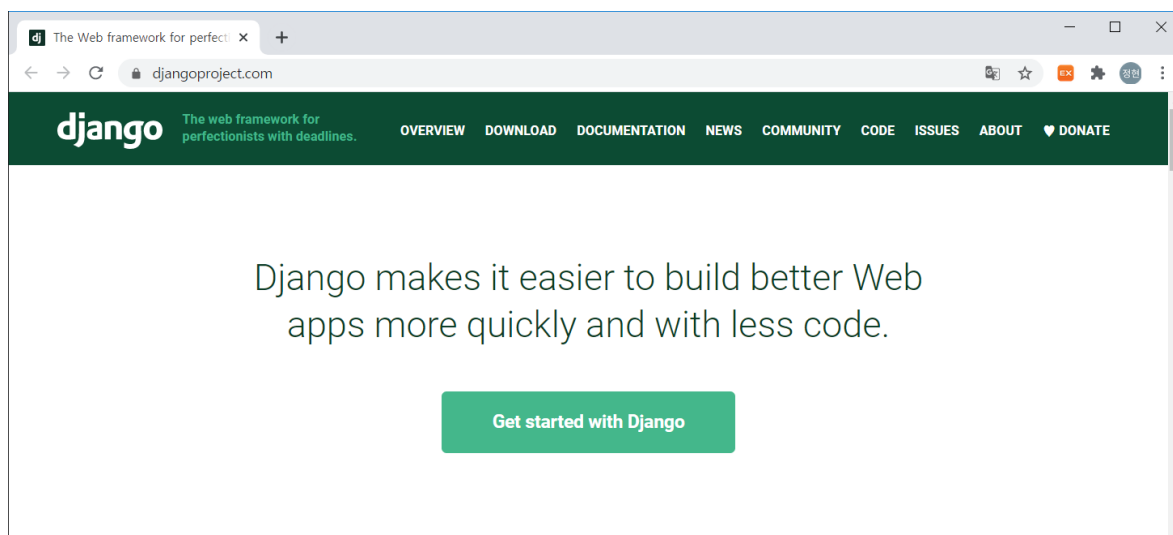
2xx - 성공, 3xx - 리다이렉션, 4xx - 클라이언트 에러, 5xx - 서버 에러



Django(장고)

장고는 파이썬으로 작성된 **오픈 소스 웹 애플리케이션 프레임워크**로, 모델-뷰-컨트롤러 패턴을 따르고 있다. 현재는 **장고 소프트웨어 재단**에 의해 관리되고 있다. 고도의 데이터베이스 기반 웹사이트를 작성하는 데 있어서 수고를 덜어 주는 것이 장고의 주된 목표이다.

장고 공식 웹 사이트 URL : <https://www.djangoproject.com/>



- 장고로 개발된 웹 사이트



Instagram



Pinterest



Udemy



Sentry



Coursera



MIT

[라이브러리와 프레임워크]

- 라이브러리(API)란

재사용이 필요한 기능으로 반복적인 코드 작성을 없애기 위해 언제든지 필요한 곳에서 호출하여 사용할 수 있도록 Class나 Function으로 만들어진 것이다. 사용 여부는 코드 작성자 선택 사항이며 새로운 라이브러리 제작 시에도 엄격한 규칙이 존재하지 않는다. 제작 의도에 맞게 작성하면 된다.

간략 설명: 프로그램 제작 시 필요한 기능

비교 설명: 자동차 바퀴, 자동차 헤드라이트, 자동차 에어백

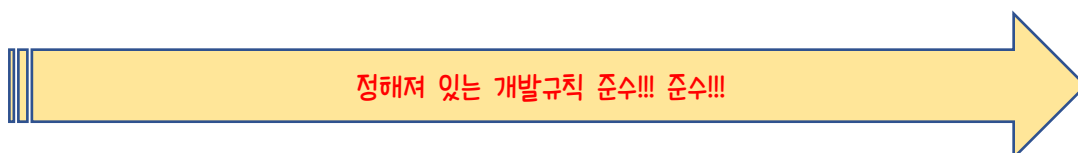
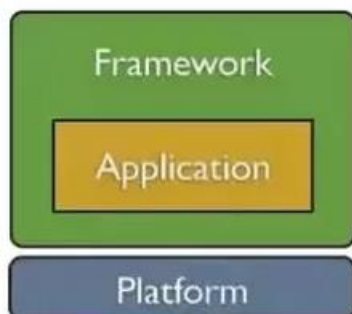


- 프레임워크란

원하는 기능 구현에만 집중하여 빠르게 개발 할 수 있도록 기본적으로 필요한 기능을 갖추고 있는 것으로 위에서 설명한 라이브러리가 포함되어 있다. 프레임워크만으로는 실행되지 않으며 기능 추가를 해야 되고 프레임워크에 의존하여 개발해야 되며 프레임워크가 정의한 규칙을 준수해야 한다.

간략 설명: 프로그램 기본 구조(빠대)

비교 설명: 자동차 프레임



- 장고 프로그램의 개발 패턴

장고 프로그래밍에서 사용되는 개발 패턴은 MVC(Model View Controller) 패턴이다. MVC는 SW 공학에서 사용되는 SW 개발 패턴으로 이 패턴을 성공적으로 사용하면 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적인 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향없이 쉽게 유지 보수할 수 있는 애플리케이션을 만들 수 있다.

MVC에서 Model은 애플리케이션에서 처리하는 데이터를 나타내며 View는 사용자 인터페이스 요소를 나타내고 Controller는 데이터와 비즈니스 로직 사이의 상호동작을 관리한다. 요청과 응답의 동작으로 이루어지는 웹은 MVC 패턴을 기반으로 웹 애플리케이션 개발 시 요청 로직과 응답 로직을 나누어 개발해서 요청 로직은 Controller로 응답 로직은 View로 그리고 처리 데이터는 Model로 개발하여 확장성과 유지보수성을 향상시킬 수 있게 되어 이미 많은 사이트들이 이 MVC 패턴을 적용하여 개발되고 운용되고 있다.

장고 프레임 워크에서는 View를 Template, Controller는 View라고 표현하며, MVC를 MTV라고 한다. Model은 데이터베이스에 저장되는 데이터의 영역 Template은 사용자에게 보여지는 영역 View는 실질적으로 프로그램 로직이 동작하여 적절한 처리 결과를 Template에게 전달하는 역할을 수행한다.

소프트웨어 디자인 패턴
MVC

django

M_{odel}

M_{odel}

V_{iew}

T_{emplate}

C_{ontroller}

V_{iew}

Model

Template

View

M

T

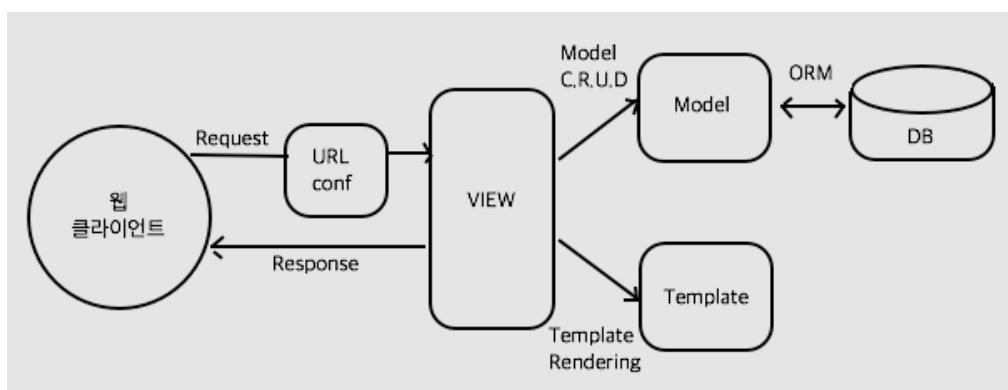
V

데이터 관리
(DB 동작)

사용자가 보는 화면
(인터페이스)

중간 관리자
(상호 동작)

- 장고의 처리 흐름



웹 클라이언트의 요청을 받아 장고에서 MTV 모델에 따라 처리하는 과정이다

- 클라이언트로부터 요청을 받으면 URLconf 모듈을 이용하여 URL을 분석한다.
- URL 분석 결과를 통해 해당 URL에 매칭되는 뷰를 실행한다.
- 뷰는 자신의 로직을 실행하고, 데이터베이스 처리가 필요하면 모델을 통해 처리하고 그 결과를 반환 받는다.

- 뷰는 자신의 로직 처리가 끝나면 템플릿을 사용하여 클라이언트에 전송할 HTML 파일을 생성한다.
- 뷰는 최종 결과로 HTML 파일을 클라이언트에게 보내 응답한다.
- 장고 개발환경 구축

가상 환경(virtual environment)

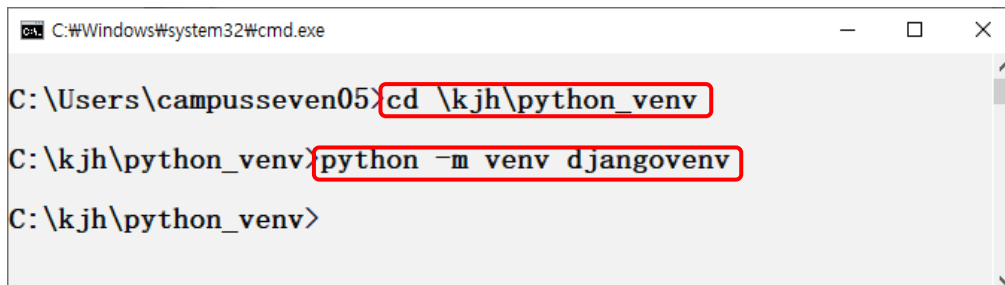
파이썬에서 가상 환경(virtual environment)은 하나의 PC에서 프로젝트 별로 독립된 파이썬 실행 환경(runtime/interpreter)을 사용할 수 있도록 해준다. 가상 환경을 사용하지 않으면 PC 내의 모든 프로젝트에서 운영체제에 설치된 하나의 파이썬 런타임을 사용하게 되고 동일한 위치에 외부 패키지를 설치하고 서로 공유하게 된다. 이럴 경우, 하나의 프로젝트에서 설치한 패키지의 버전이 다른 프로젝트에서 설치한 동일 패키지의 다른 버전과 충돌을 일으킬 소지가 생기기 때문에, 프로젝트 별로 독립된 가상 환경을 구성하여 사용하는 것이 권장된다.

파이썬에서 외부 패키지를 설치할 때는 일반적으로 pip이라는 패키지 매니저를 사용하는데, 기본적으로 운영체제에 파이썬이 설치된 위치의 site-packages 디렉터리에 안에 설치된다. 파이썬의 가상 환경을 이용하면 프로젝트 별로 따로 패키지를 설치하고, 다른 프로젝트로 부터 격리시킬 수 있기 때문에 시스템 전역 패키지 설치로 인한 불필요한 이슈를 방지할 수 있다. 파이썬 3.3 부터는 venv 모듈이 내장되기 때문에 별도 패키지 설치없이 파이썬만 설치되어 있으면 바로 가상 환경 구성이 가능하게 되었다.

[가상환경 만들기과 가상환경 안에 장고 개발 환경 설치하기]

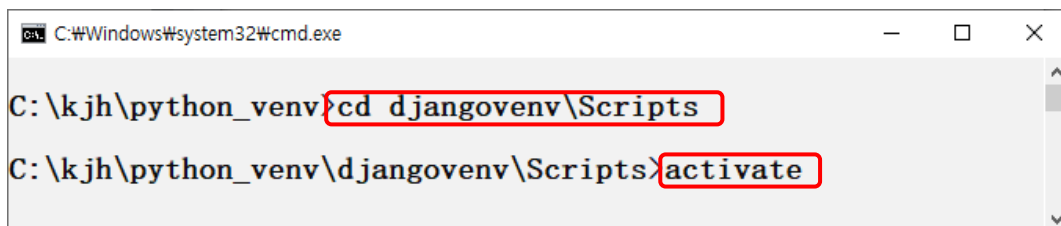
(1) c:\xxx\python_venv 라는 폴더를 생성한다.

(2) c:\xxx\python_venv 폴더로 옮겨서 다음 명령을 입력하고 실행시켜 가상 환경, djangoenv 를 생성한다.



```
C:\Windows\system32\cmd.exe
C:\Users\campussevent05>cd \kjh\python_venv
C:\kjh\python_venv>python -m venv djangoenv
C:\kjh\python_venv>
```

(3) 생성된 djangoenv 폴더의 Scripts 라는 폴더로 옮겨서 activate 프로그램을 실행하여 가상환경을 활성화 한다.



```
C:\Windows\system32\cmd.exe
C:\kjh\python_venv>cd djangoenv\Scripts
C:\kjh\python_venv\djangoenv\Scripts>activate
```

(4) djangoenv라는 가상환경을 활성화하면 다음과 같이 명령 프롬프트 맨 앞에 (djangoenv)가 붙은 것을 볼 수 있다. 이렇게 가상환경을 활성화한 상태에서 이 가상환경 안에 장고 개발 환경을 설치하기 위해 pip 명령을 실행한다.



```
C:\Windows\system32\cmd.exe
(djangoenv) C:\kjh\python_venv\djangoenv\Scripts>pip install django
```



```
C:\Windows\system32\cmd.exe

(djangoven) C:\kjh\python_venv\djangoven\Scripts>pip install django
Collecting django
  Downloading Django-3.1.5-py3-none-any.whl (7.8 MB)
    |████████████████████████████████████████████████████████████████████████████████| 7.8 MB 6.4 MB/s
Collecting pytz
  Downloading pytz-2020.5-py2.py3-none-any.whl (510 kB)
    |████████████████████████████████████████████████████████████████████████████████| 510 kB ...
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.1-py3-none-any.whl (42 kB)
    |████████████████████████████████████████████████████████████████████████████████| 42 kB 420 kB/s
Collecting asgiref<4, >=3.2.10
  Downloading asgiref-3.3.1-py3-none-any.whl (19 kB)
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.3.1 django-3.1.5 pytz-2020.5 sqlparse-0.4.1
WARNING: You are using pip version 20.2.3; however, version 21.0 is available.
You should consider upgrading via the 'c:\kjh\python_venv\djangoven\scripts\python.exe -m pip install --upgrade pip' command.

(djangoven) C:\kjh\python_venv\djangoven\Scripts>
```

위의 화면은 장고 개발환경이 설치되는 화면이다. 그런데 제일 아래에 보면 우리 시스템에 설치된 pip 의 버전 관련해서 경고 오류가 출력되는 것을 볼 수 있다. 다음 명령을 실행시켜 업그레이드 해놓는다.(명령어 : `python -m pip install --upgrade pip`)

```
C:\Windows\system32\cmd.exe

(djangoven) C:\kjh\python_venv\djangoven\Scripts>python -m pip install --upgrade pip
Collecting pip
  Downloading pip-21.0-py3-none-any.whl (1.5 MB)
    |████████████████████████████████████████████████████████████████████████████████| 1.5 MB 1.6 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
  Successfully installed pip-21.0

(djangoven) C:\kjh\python_venv\djangoven\Scripts>
```

[장고 프로젝트 만들기]

- (1) `c:\xxx\DJANGOexam` 이라는 폴더를 생성한다.
- (2) `c:\xxx\DJANGOexam` 폴더로 이동해서 다음 명령을 입력하고 실행시켜 학습용 장고 프로젝트를 생성한다.
(이동하는 명령 : `cd c:\xxx\DJANGOexam`)

```
C:\Windows\system32\cmd.exe

(djangoven) C:\kjh\DJANGOexam>django-admin startproject studyproject

(djangoven) C:\kjh\DJANGOexam>
```

(3) 생성된 studyproject 폴더로 이동해서 `python manage.py runserver` 를 입력하고 실행시켜 장고에 내장된 서버를 기동시킨다. (이동하는 명령 : `cd studyproject`)

```
C:\Windows\system32\cmd.exe - python manage.py runserver

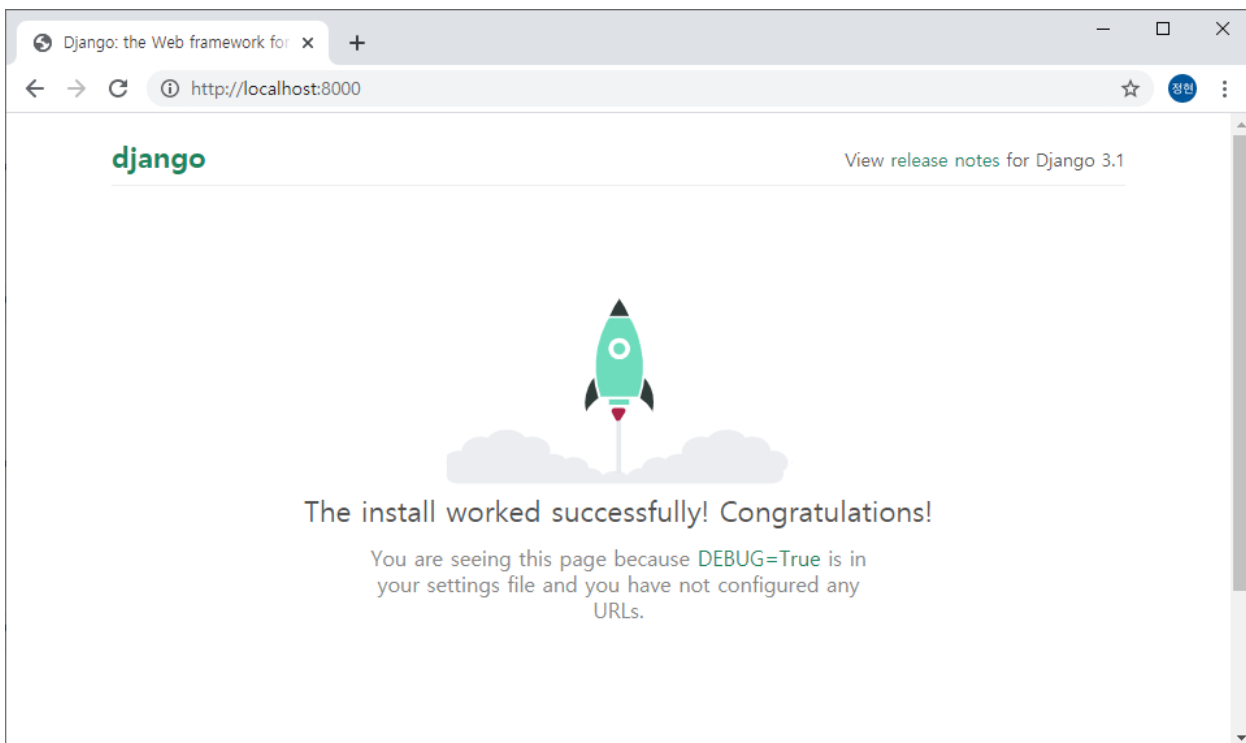
(djangovenv) C:\kjh\DJANGOexam>cd studyproject

(djangovenv) C:\kjh\DJANGOexam\studyproject>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 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.
January 25, 2021 - 17:40:59
Django version 3.1.5, using settings 'studyproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

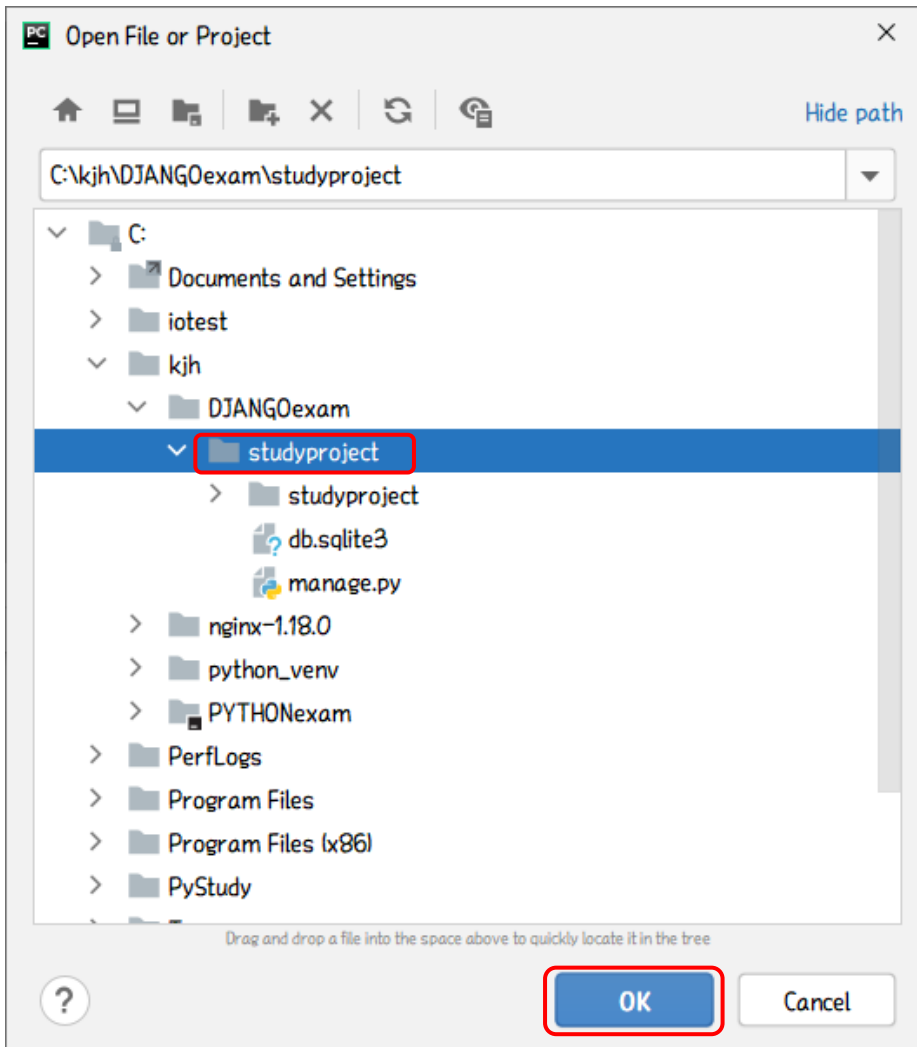
(4) 크롬 브라우저에서 `http://localhost:8000/` 입력하여 요청하면 장고의 기본 웹 페이지가 출력되는 것을 볼 수 있다.



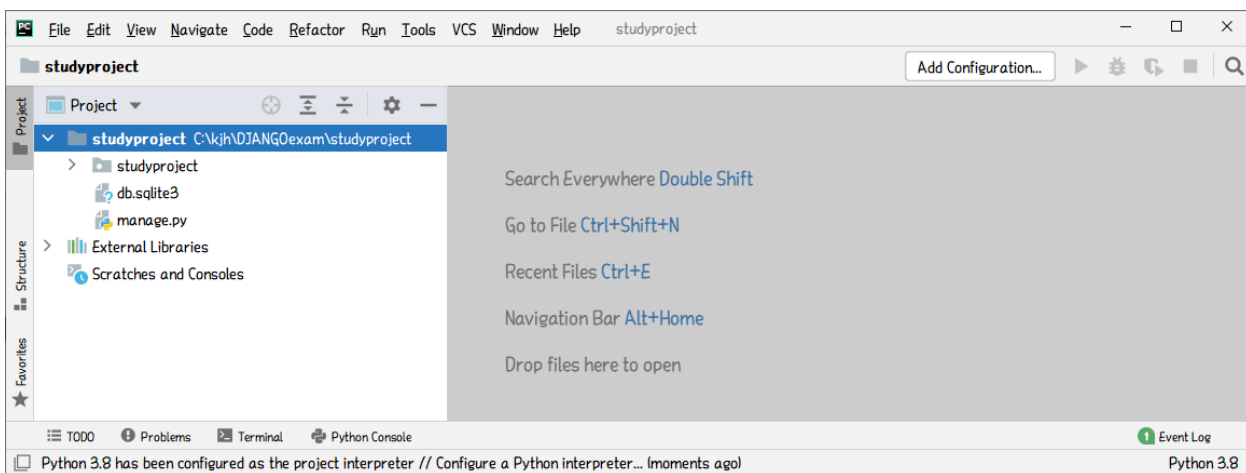
-----→ 이 화면이 잘 보인다면!!! 추카추카!!

cmd 창에서 `ctrl+c`를 입력하여 서버를 종료하고 이제부터는 생성된 프로젝트를 파이썬으로 데리고 들어간다.

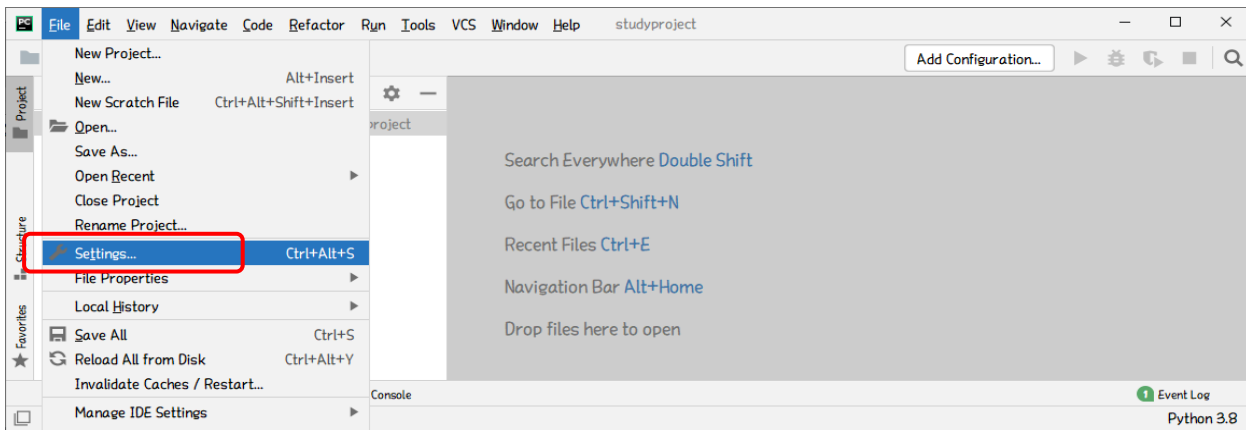
(1) 파이참에서 **File>Open** 을 선택하고 **c:\xxx\DJANGOexam\studyproject**을 엽니다. 다음 화면을 참조한다.



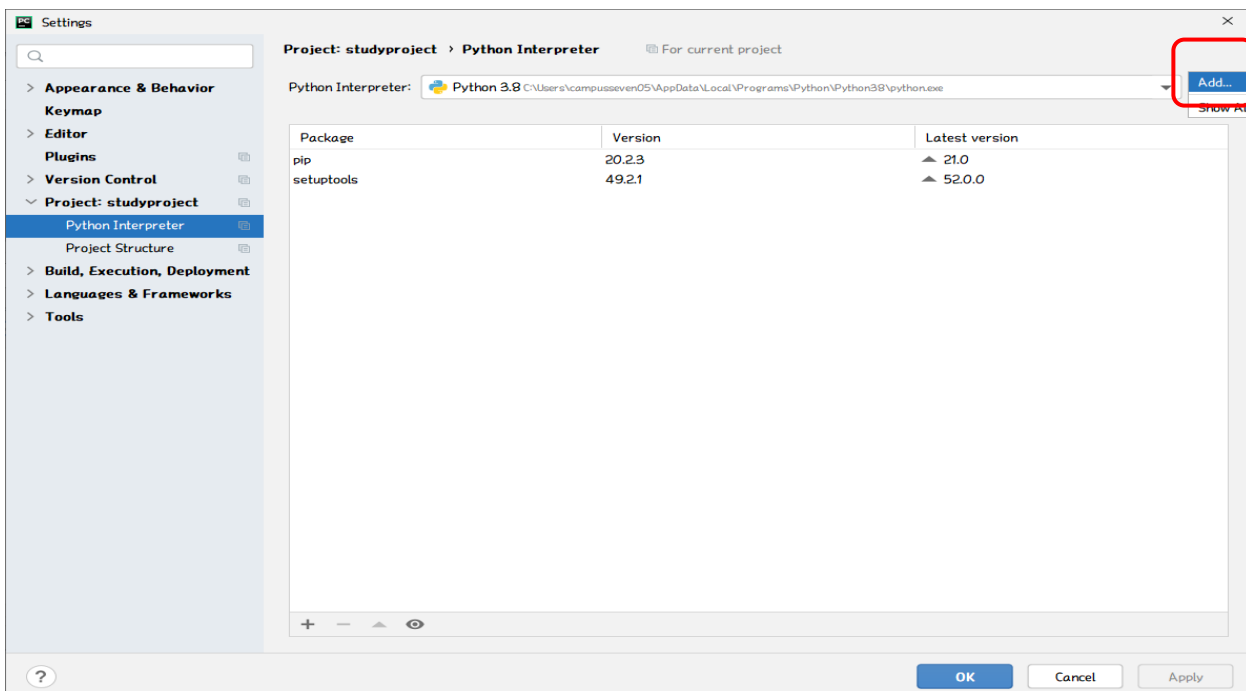
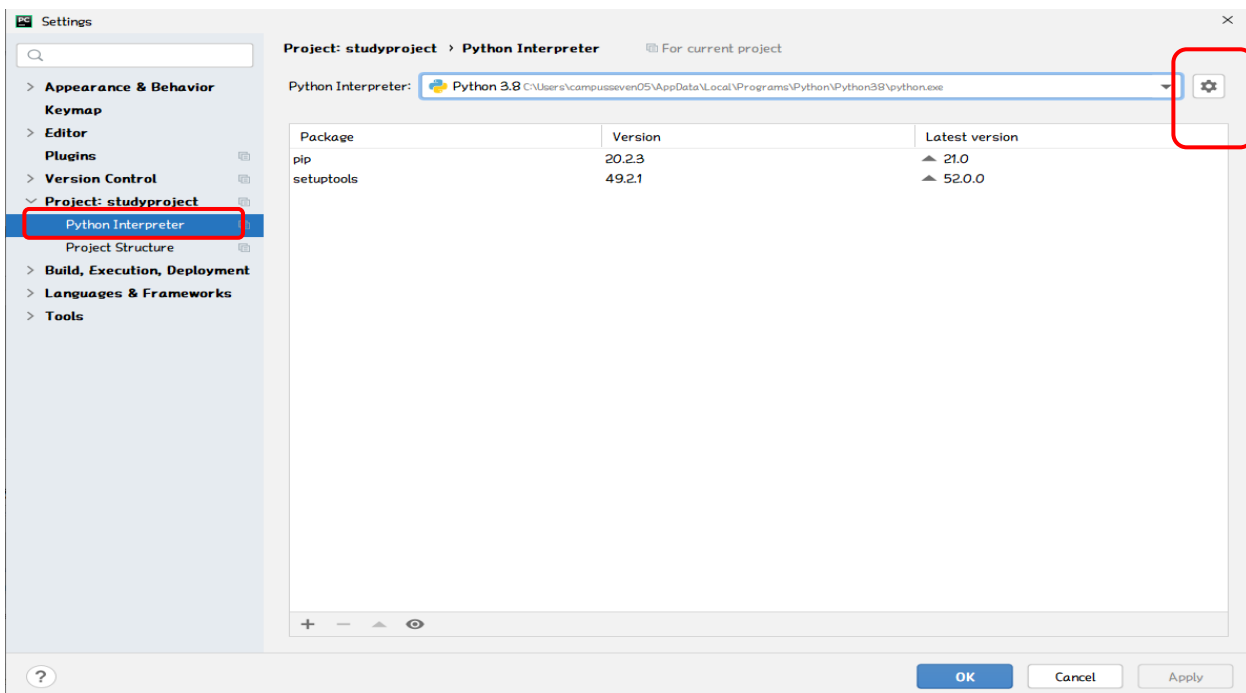
오픈이 잘 진행되면 다음과 같이 **studyproject** 가 인식되는 것을 볼 수 있다.

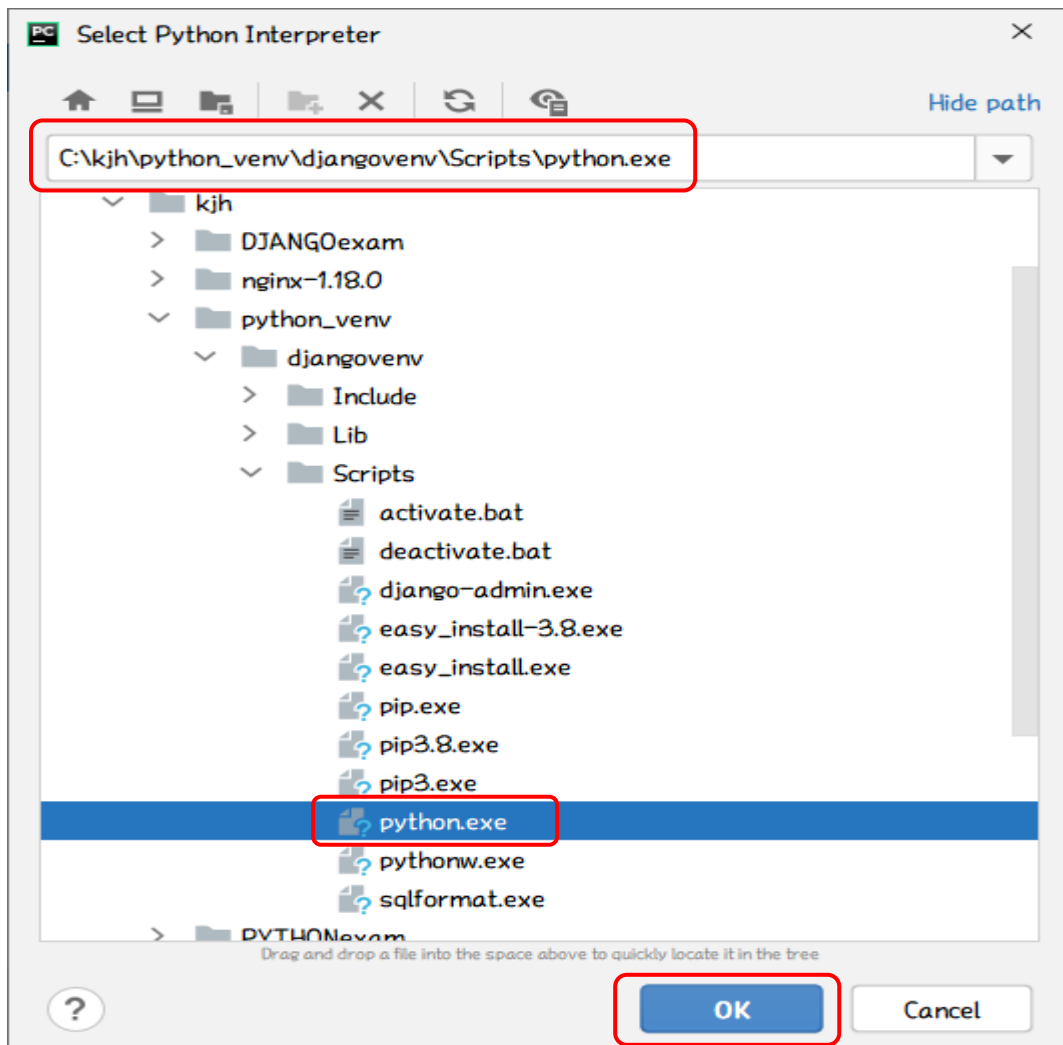
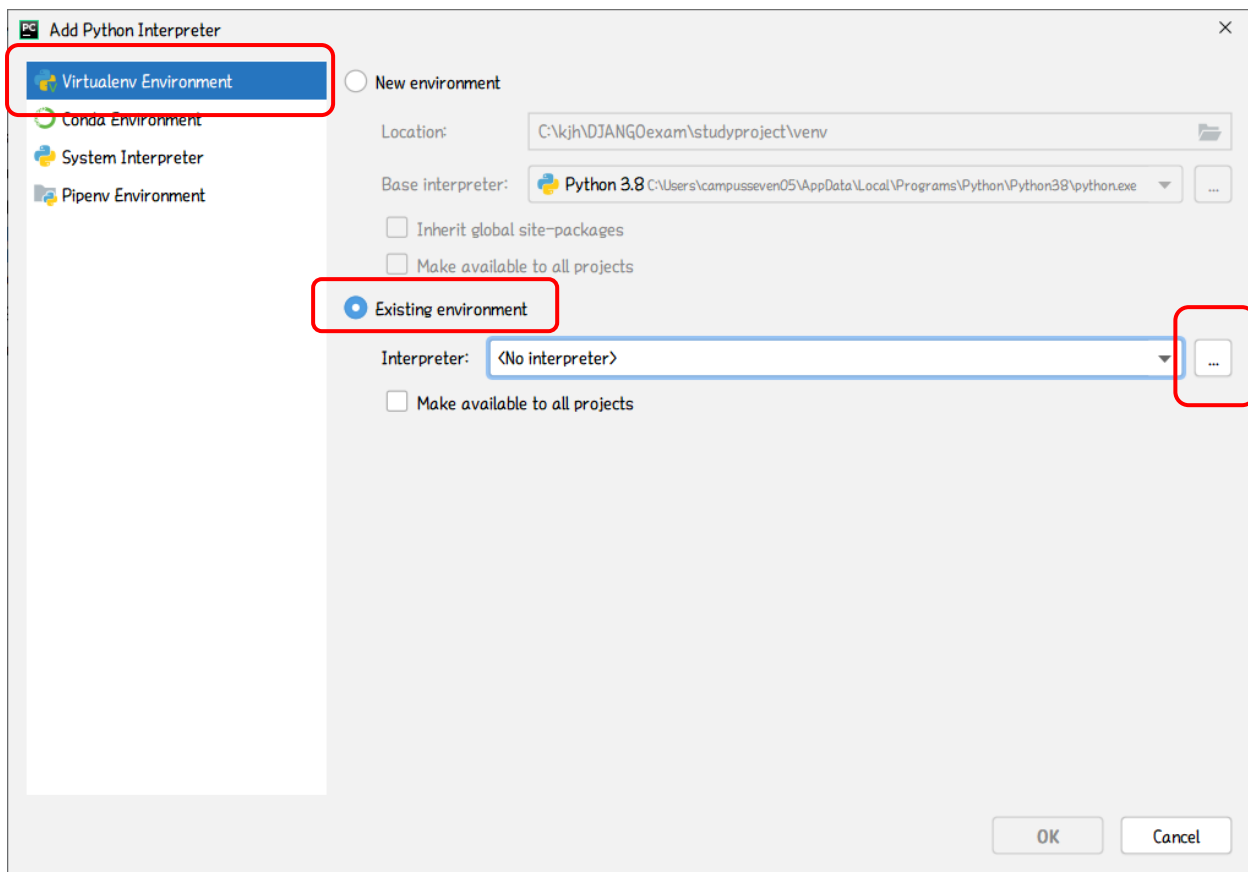


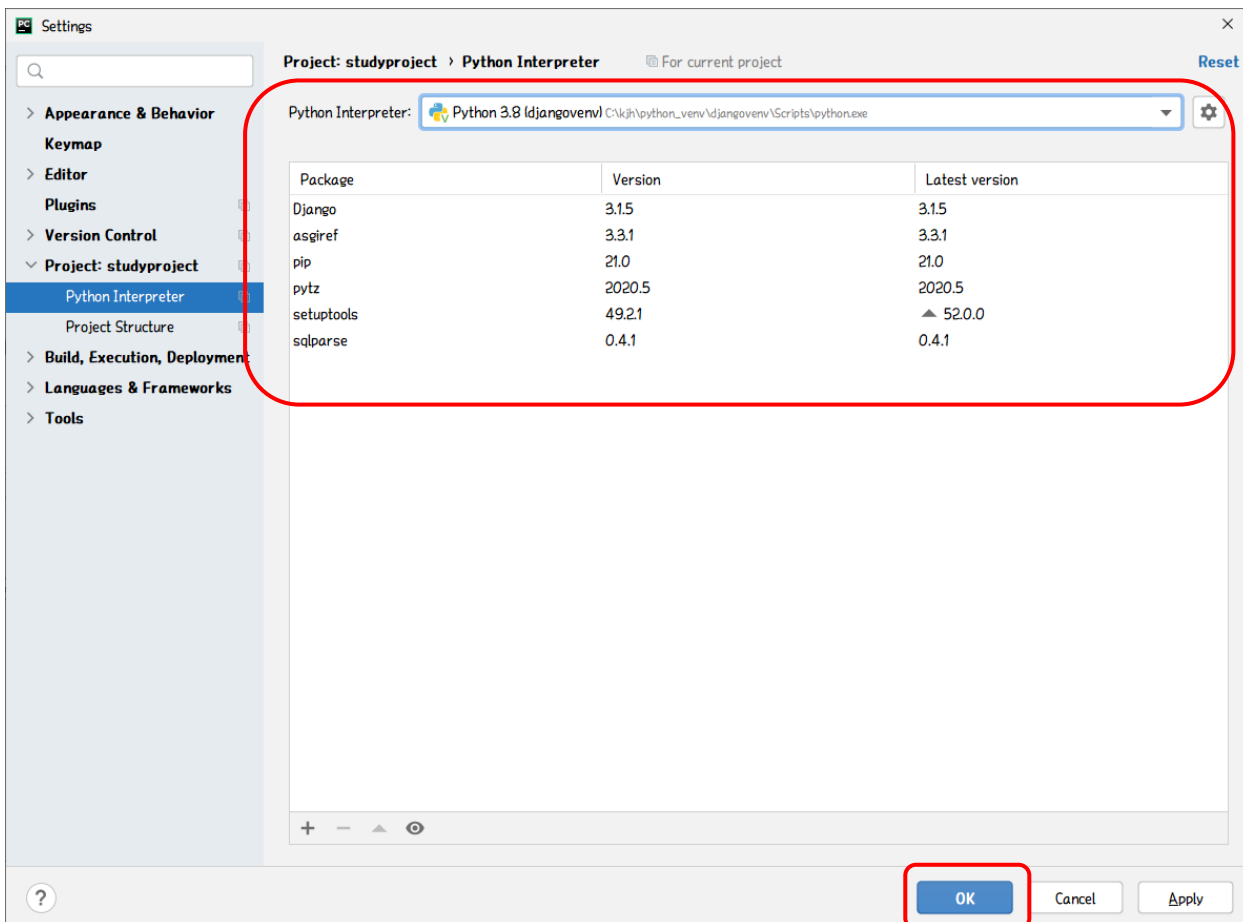
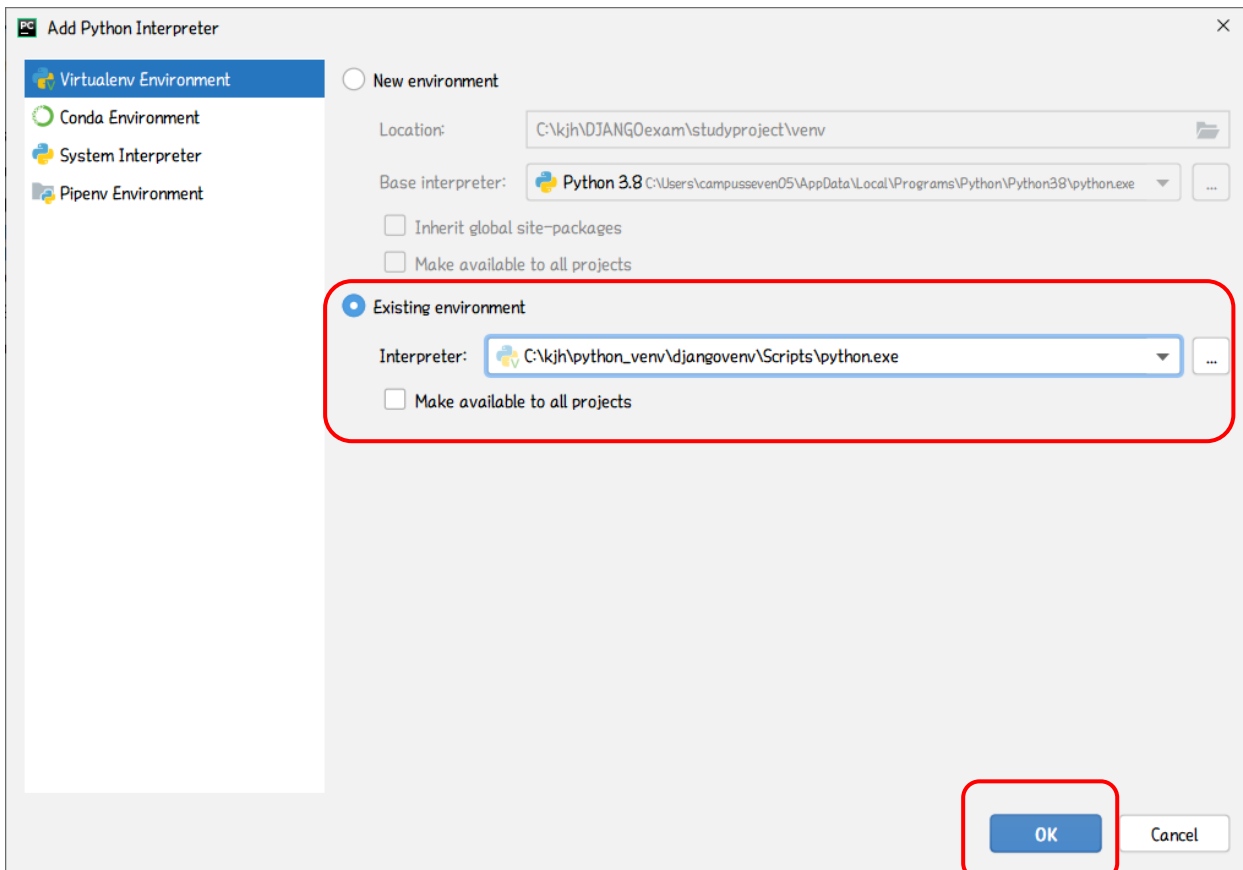
(2) **File>Settings** 를 선택하고 **studyproject**의 프로젝트 인터프리터를 가상환경으로 선택한다. 다음 화면과 같이 선택하고 OK 버튼을 클릭한다.



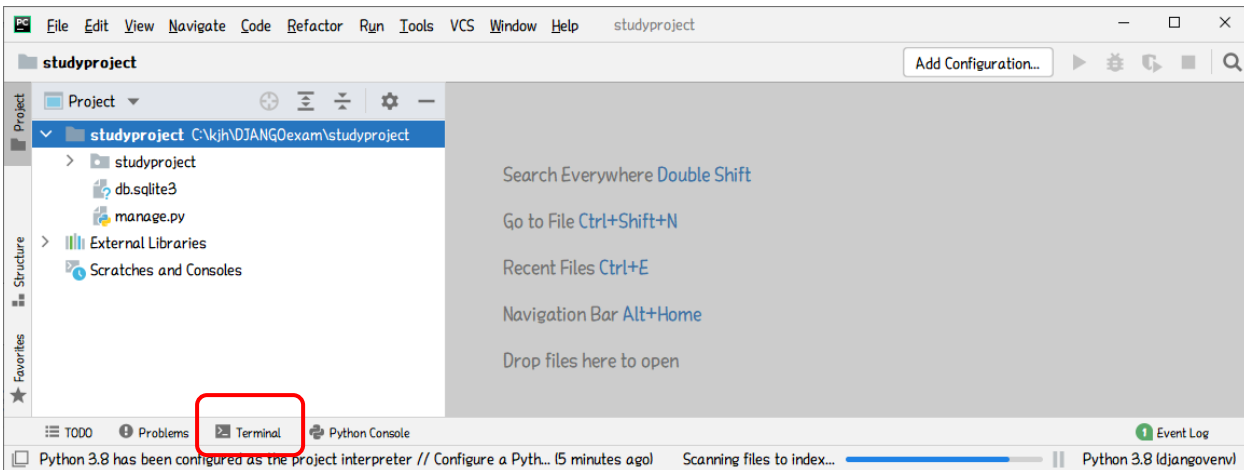
다음과 같은 서브윈도우가 출력되면 **Project: studyproject** 의 **Python Interpreter** 를 선택한다.



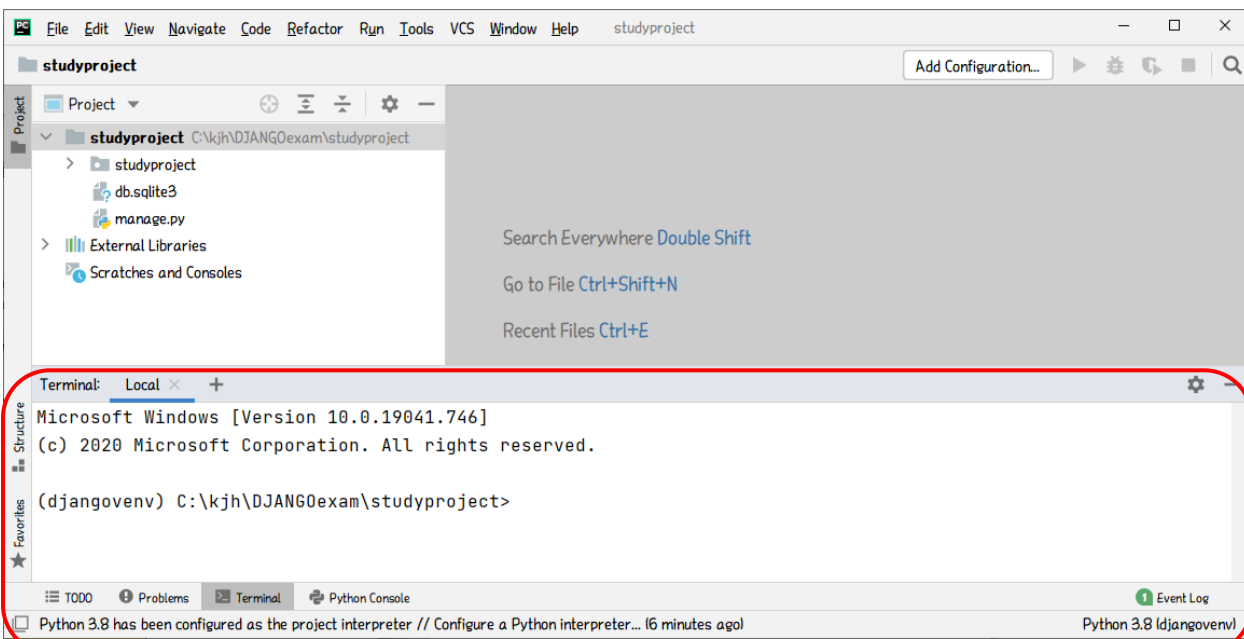




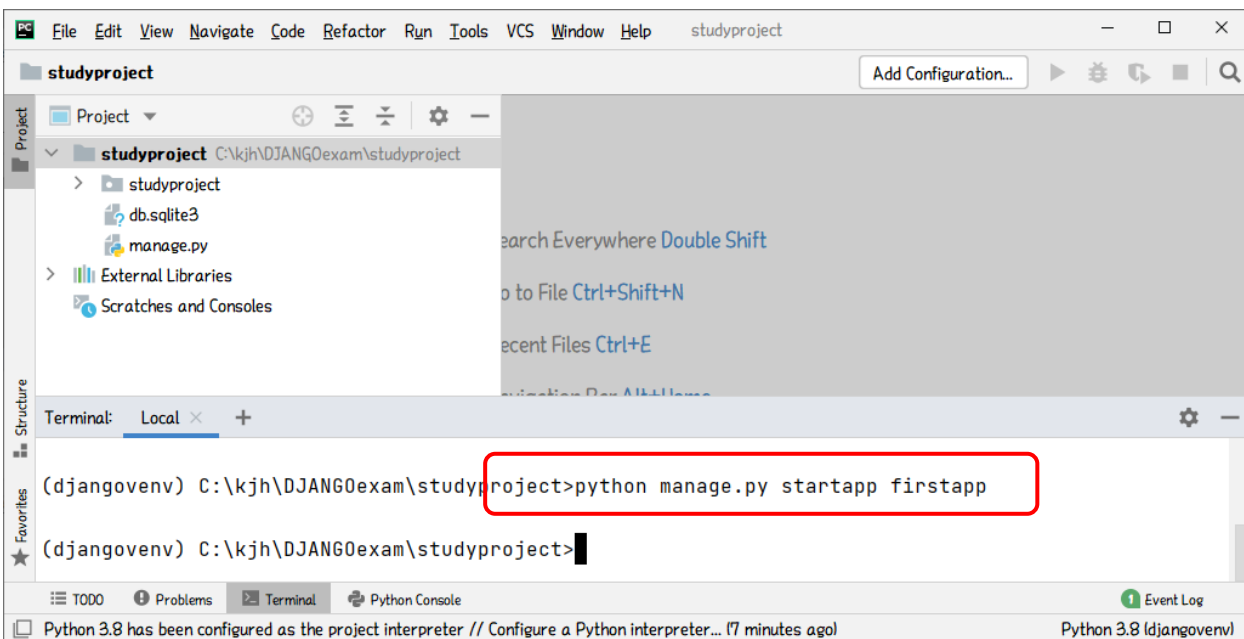
(3) 파이참 하단 왼쪽에 있는 Terminal을 선택하여 cmd 창과 비슷한 창을 출력한다.



그러면 다음과 같이 하단에 쉘 창하고 비슷하게 명령을 입력할 수 있는 화면이 출력된다..



(4) 학습용 소스들을 작성할 **firstapp** 이라는 장고 앱을 하나 생성한다.



(5) firstapp 이라는 장고 앱을 생성하면 다음과 같은 디렉토리 구조가 생성된다. 표시된 파일들을 제시된 내용으로 수정한다.

from django.http import HttpResponse
def welcome(request):
 return HttpResponse("<h1>장고 공부를 재미있게 합시다!!</h1>")

106~108 에서 다음 내용으로 수정한다.
LANGUAGE_CODE = 'ko-kr'
TIME_ZONE = 'Asia/Seoul'

import firstapp.views
urlpatterns = [
,
 path('welcome/', firstapp.views.welcome),
]

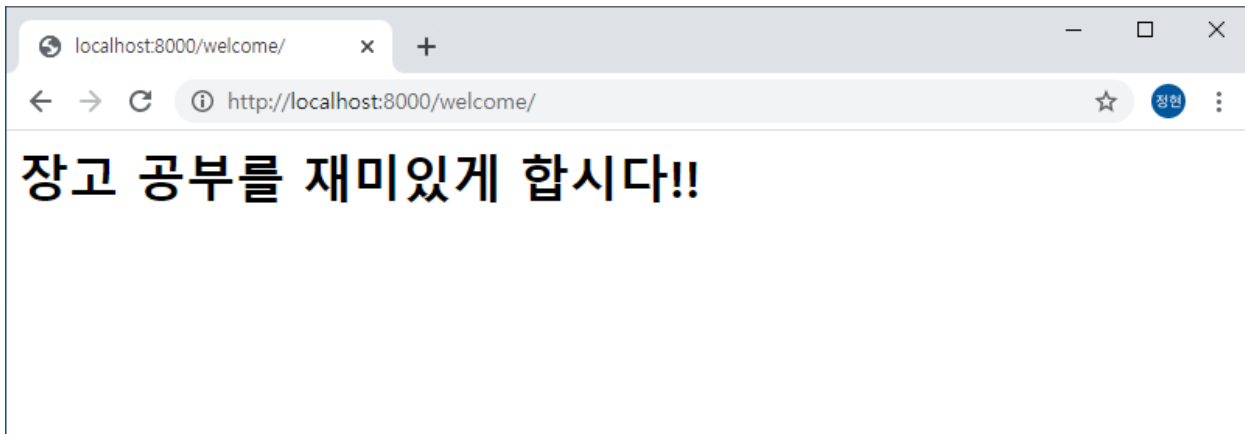
(6) settings.py, views.py, urls.py 이 세 개의 파일의 수정이 성공적으로 된 것을 확인한 후에 서버를 기동시킨다.

```
(djangoenv) C:\kjh\DJANGOexam\studypoint>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

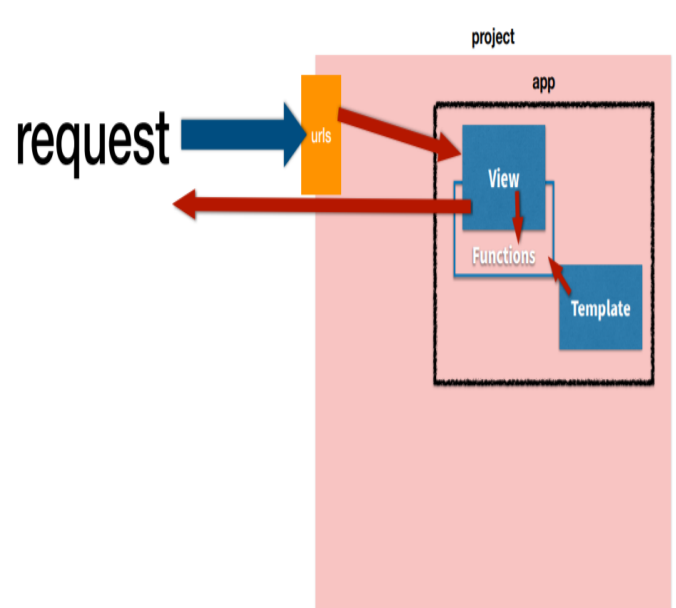
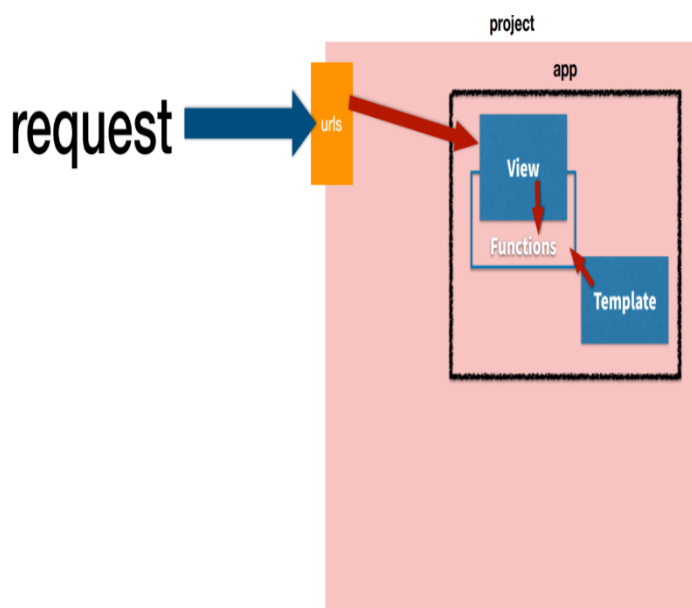
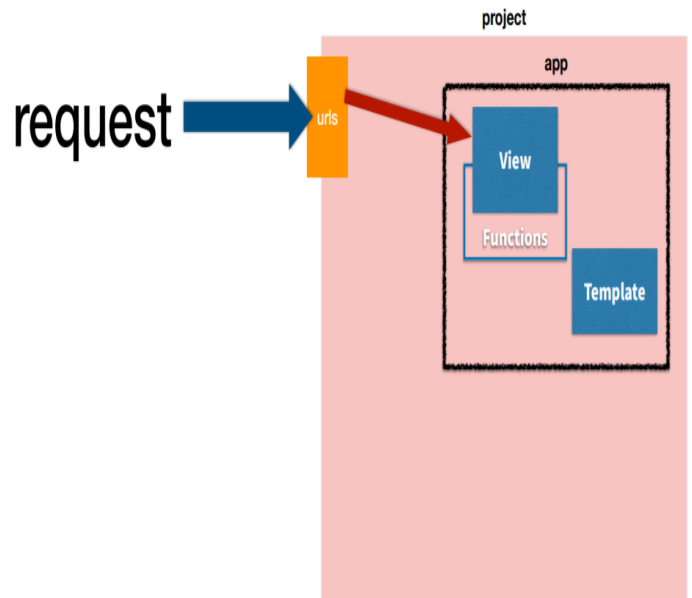
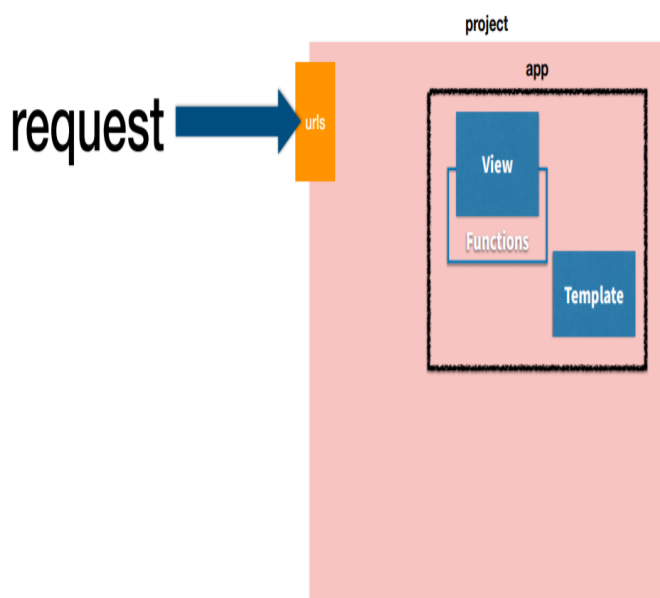
System check identified no issues (0 silenced).

You have 18 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.
January 25, 2021 - 18:03:10
Django version 3.1.5, using settings 'studypoint.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```


(7) 브라우저에서 다음 URL 로 요청해 본다.



[Views와 Templates]



urls.py - 프로젝트 폴더(메인)와 앱폴더(서브)

views.py - 앱폴더

templates\xxxx.html - 앱폴더

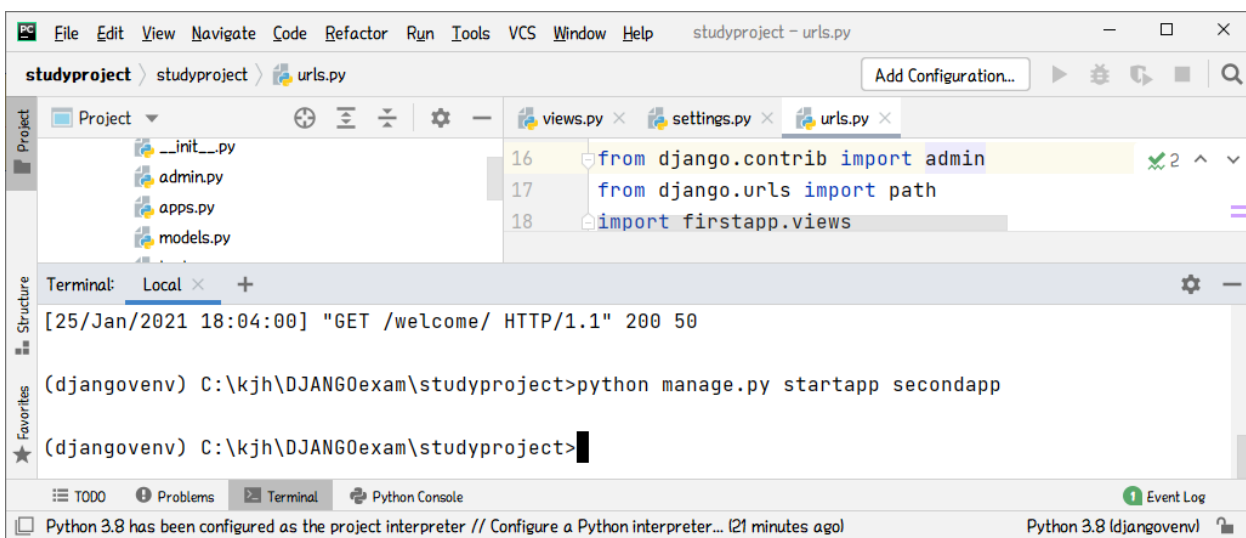
[장고를 이용한 웹 서버 프로그래밍]

요청 시 사용될 URL 문자열의 패스와 호출될 뷰의 함수를 결정한다. - 라우팅을 정한다고 한다.	urls.py
GET 방식 요청을 처리할 것인지, POST 방식 요청을 처리할 것인지 결정하고 요청에 대한 기능을 수행하도록 위에서 설정한 함수에 로직을 구현한다.	views.py
수행한 결과를 클라이언트로 응답하는 기능의 템플릿을 구현한다.	templates/xxx.html

[두 번째 장고 앱 생성]

(1) 파이참 하단 왼쪽에 있는 Terminal을 선택하여 cmd 창과 비슷한 창을 출력한다.(터미널 창이라고 한다.)

(2) 학습용 소스들을 작성할 secondapp 이라는 장고 앱을 하나 생성한다.



<body>

<h1>안녕? 템프릿(html)을 통해서 응답해요</h1>

</body>

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    path'', views.exam1, name='exam1'),
```

```
]
```

```
from django.shortcuts import render
```

```
from django.http import HttpResponse
```

```
from django.template import loader
```

```
def exam1(request):
```

```
    template = loader.get_template('exam1.html')
```

```
    return HttpResponse(template.render(None, request))
```

```
from django.contrib import admin
```

```
from django.urls import path, include
```

```
import firstapp.views
```

```
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
```

```
    path('welcome/', firstapp.views.welcome ),
```

```
    path('secondapp/', include('secondapp.urls')),
```

```
]
```

33행

```
INSTALLED_APPS = [
```

```
    'secondapp',
```

```
]
```

안녕? 템프릿(html)을 통해서 응답해요

[secondapp/views.py]

```
def exam2(request):
    template = loader.get_template('exam2.html')
    if request.method == 'GET':
        msg = "GET방식으로 했군요...ㅎ"
    else:
        msg = "POST방식으로 했군요...ㅎ"
    context = {'result': msg}
    return HttpResponse(template.render(context, request))

def exam2_1(request):
    template = loader.get_template('exam2_1.html')
    if request.method == 'GET':
        msg = request.GET.get("info1", "없음") + "-" + request.GET.get("info2", "없음") + "-" + request.GET.get("info3", "없음")
    else:
        msg = request.POST.get("info1", "없음") + "-" + request.POST.get("info2", "없음") + "-" +
            request.POST.get("info3", "없음")
    context = {'result': msg}
    return HttpResponse(template.render(context, request))
```

[secondapp/urls.py]

```
path('exam2/', views.exam2, name='exam2'),
path('exam2_1/', views.exam2_1, name='exam2_1'),
```

[exam2.html]

```
<body>
<h1>HTTP 요청 방식을 파악하자</h1>
<hr>
<h2>이번 요청은 {{ result }} </h2>
<hr>
<a href="http://localhost:8000/secondapp/exam2">하이퍼 링크 텍스트로 요청하기</a>
<hr>
<form method="GET" method="/secondapp/exam2">
    <input type="hidden" name="info1" value="django">
    <input type="hidden" name="info2" value="css">
```

```

    <input type="hidden" name="info3" value="javascript">
    <input type="submit" value="<form>태그로 GET방식 요청">
</form>
<hr>
<a href="http://localhost:8000/secondapp/exam2"></img></a>
<hr>
<form method="POST" method="/secondapp/exam2">
    {% csrf_token %}
    <input type="hidden" name="info1" value="django">
    <input type="hidden" name="info2" value="css">
    <input type="hidden" name="info3" value="javascript">
    <input type="submit" value="<form>태그로 POST방식 요청">
</form>
<hr>
</body>

```

[exam2_1.html]

```

<body>
<h1>HTTP 요청 방식과 Query 추출을 파악하자</h1>
<hr>
<h2>전달된 Query : {{ result }}</h2>
<hr>
<a href="http://localhost:8000/secondapp/exam2_1">하이퍼 링크 텍스트로 요청하기</a>
<hr>
<form method="GET" method="/secondapp/exam2_1">
    <input type="hidden" name="info1" value="django">
    <input type="hidden" name="info2" value="css">
    <input type="hidden" name="info3" value="javascript">
    <input type="submit" value="<form>태그로 GET방식 요청">
</form>
<hr>
{% load static %}
<a href="http://localhost:8000/secondapp/exam2_1">
    </img></a>
<hr>
<form method="POST" method="/secondapp/exam2_1">

```

{% csrf_token %}

<input type="hidden" name="info1" value="django">

<input type="hidden" name="info2" value="css">

<input type="hidden" name="info3" value="javascript">

<input type="submit" value="<form>태그로 POST방식 요청">

</form>

<hr>

</body>

HttpRequest와 HttpResponse

장고는 request와 response 객체로 서버와 클라이언트가 정보를주고 받습니다. 이를 위해 장고는 django.http 모듈에서 HttpRequest와 HttpResponse API를 제공한다.

서버-클라이언트 통신 시 아래와 같은 절차로 데이터가 오고 간다.

- 1) 특정 페이지가 요청(리퀘스트)되면, 장고는 요청 시 메타데이터(여러 다양한 정보를 포함하는 HttpRequest 객체를 생성
- 2) 장고는 urls.py에서 정의한 특정 View 함수에 첫 번째 인자로 해당 객체(request)를 전달
- 3) 해당 View는 결과값을 HttpResponse 혹은 JsonResponse 객체에 담아 전달

- HttpRequest 객체

1) 주요 속성(Attribute)

HttpRequest.body # request의 body 객체

HttpRequest.headers # request의 headers 객체

HttpRequest.COOKIEs # 모든 쿠키를 담고 있는 딕셔너리 객체

HttpRequest.method # request의 메소드 타입

HttpRequest.GET # GET 파라미터를 담고 있는 딕셔너리 같은 객체

HttpRequest.POST # POST 파라미터를 담고 있는 딕셔너리 같은 객체

2) 주요 메소드(Methods)

HttpRequest.read

HttpRequest.get_host()

HttpRequest.get_port()

[사용 사례]

request.method

if request.method == 'GET':

do_something()

elif request.method == 'POST':

do_something_else()

request.headers 가져오기

'User-Agent' in request.headers

request.headers['User-Agent']

request.headers.get('User-Agent')

- HttpResponseRedirect

HttpResponse(data, content_type)

response를 반환하는 가장 기본적인 함수로서 주로 html을 반환

string 전달하기

HttpResponse("Here's the text of the Web page.")

html 태그 전달하기

response = HttpResponse()

response.write("<p>Here's the text of the Web page.</p>")

- HttpResponseRedirect

HttpResponseRedirect(url)

별다른 response를 하지는 않고, 지정된 url페이지로 redirect를 함

첫 번째 인자로 url를 반드시 지정해야 하며, 경로는 절대경로 혹은 상대경로를 이용할 수 있음

- Render

render(request(필수), template_name(필수), context=None, content_type=None, status=None, using=None)

render는 http response 객체를 반환하는 함수로 template을 context와 엮어 HttpResponseRedirect로 쉽게 반환해 주는 함수임

template_name에는 불러오고 싶은 템플릿명을 적음

context에는 View에서 사용하던 변수(dictionary 자료형)를 html 템플릿에서 전달하는 역할을 함.

key 값이 템플릿에서 사용할 변수이름, value값이 파이썬 변수 값이 됨

views.py

from django.shortcuts import render

def my_view(request):

 name = "joey"

 return render(request, 'app/index.html', {'name': name})

- JsonResponse

JsonResponse(data, encoder=DjangoJSONEncoder, safe=True, json_dumps_params=None, **kwargs)

HttpResponse의 subclass로, JSON-encoded response를 생성할 수 있게 해 줌.

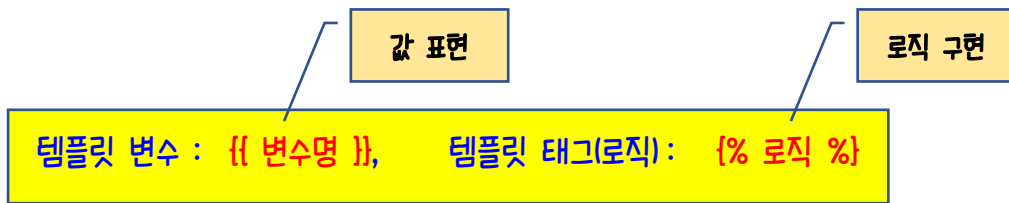
대부분의 기능은 superclass에서 상속받음 첫 번째 인자로는 전달할 데이터로서 반드시 dictionary 객체여야 함.

디폴트 Content-type 헤더는 application/json임 encoder는 데이터를 serialize할 때 이용됨.

json_dumps_params는 json.dumps()에 전달할 디렉터리의 keyword arguments임

Serializing non-dictionary objects

response = JsonResponse([1, 2, 3], safe=False)



Django의 템플릿 언어(template language)는 강력함과 편리함 사이의 균형을 잡고자 설계되었다. 템플릿 언어를 사용하면 HTML 작업을 훨씬 수월하게 할 수 있다. 변수, 필터, 태그, 주석 등 4가지 기능을 제공한다.

1. 템플릿 변수

템플릿변수를 사용하면 뷰에서 템플릿으로 객체를 전달할 수 있다. {{ 변수 }}와 같이 사용한다.

점(.) 은 변수의 속성에 접근할 때 사용한다. {{ section.title }} 뷰에서 보내온 section객체의 title 속성을 출력한다.

변수명으로 데이터 값 추출이 안되는 경우 공백으로 처리된다.

2. 템플릿 필터

템플릿필터는 변수의 값을 특정 형식으로 변환할 때 사용한다. 변수 다음에 파이프(|)를 넣은 다음 적용하고자 하는 필터를 명시한다.

여러 개의 필터를 연속적으로 사용할 수 있다. {{ text|escapelinebreaks }}는 텍스트 콘텐츠를 이스케이프한 다음, 행 바꿈을 <p> 태그로 바꾸기 위해 종종 사용되곤 한다.

몇몇 필터는 : 문자를 통해 인자를 취한다.

필터 인자는 {{bioltruncatewords:30 }}과 같이 사용하는데, 이것은 bio 변수의 처음 30 단어를 보여준다.

필터 인자에 공백이 포함된 경우에는 반드시 따옴표로 묶는다.

장고는 30개 정도의 내장 템플릿 필터를 제공하는데, 자주 사용되는 템플릿 필터를 다음과 같다.

- default

변수가 false 또는 비어 있는 경우, 지정된 default를 사용한다.

```
{{ value|default:"nothing" }}
```

value가 제공되지 않았거나 비어 있는 경우, 위에서는 “nothing“을 출력한다.

- length

값의 길이를 반환한다. 문자열과 목록에 대하여 사용할 수 있다.

```
{{ value|length }}
```

value가 ['a', 'b', 'c', 'd']라면, 결과는 4가 된다.

- upper

```
{{ story.headline|upper }}
```

'story.headline'의 값을 대문자 형식으로 변환한다.

3. 템플릿 태그

`{% tag %}` 또는 `{% tag %} ... tag contents ... {% endtag %}`

HTML 자체는 프로그래밍 로직을 구현할 수 없지만, 템플릿 태그를 사용하면 if문, for문처럼 흐름을 제어할 수 있다.

`{% tag %}` 로 구성한다. `{% extends %}`와 같이 단독으로 사용할 수 있는 템플릿 태그들도 있지만, `{% if %}` 처럼 뒤에 `{% endif %}` 템플릿 태그를 반드시 달아주어야 하는 것들도 있다.

장고에는 20개가 넘는 템플릿 태그가 내장되어 있으며 내장 태그 레퍼런스에서 읽어볼 수 있다.

- `{% csrf_token %}`

CSRF 란

사이트 간 요청 위조(또는 크로스 사이트 요청 위조, 영어: Cross-site request forgery, CSRF, XSRF)는 웹사이트 취약점 공격의 하나로, 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 공격을 말한다. 예를 들자면 아래와 같은 공격이다.

사용자가 xxx.xxx.xxx에 login을 한다. sessionid등의 쿠키값을 정상적으로 발급받아 login된다.

공격자가 mail이나 게시판등을 이용해 악의적인 http request의 주소를 사용자쪽으로 전달한다. 예를 든다면 `http://xxx.xxx.xxx/changepassword?password=abc`와 같은 request를 서버로 전송하게끔.

사용자가 해당 주소를 실행하여 원하지 않는 request를 전송하게 된다. 서버입장에선 로그인을 거친 정상적인 client가 request를 수행한 것이니 해당 프로세스를 수행한다. 위 문제를 해결 하기 위해선 changepassword 페이지의 form전달 값에 특정한 값을 추가하면 된다. 예를 들어 사용자로 부터 captcha를 값을 입력받게 하여, 정상적인 페이지에서의 요청인지 확인하면 되는 것이다. django에서는 간단히 csrf token을 발행하여 처리하게 한다.

- for

배열의 각 원소에 대하여 반복처리

``

`{% for student in student_list %}`

`{{ student.name }}`

`{% endfor %}`

``

- if / else

변수가 true이면 블록의 콘텐츠를 표시. if 태그 내에 템플릿 필터 및 각종 연산자를 사용할 수 있다.

```
{% if student_list %}
```

```
    총 학생 수 : {{ student_listlength }}
```

```
{% else %}
```

```
    학생이 없어요!
```

```
{% endif %}
```

```
{% if athlete_list %}
```

```
    Number of athletes: {{ athlete_listlength }}
```

```
{% elif athlete_in_locker_room_list %}
```

```
    Athletes should be out of the locker room soon!
```

```
{% else %}
```

```
    No athletes.
```

```
{% endif %}
```

- block 및 extentds

중복되는 html 파일 내용을 반복해서 작성해야 하는 번거로움을 줄여준다.

```
{% extends "base_generic.html" %}
```

```
{% block title %}{{ section.title }}{% endblock %}
```

```
{% block content %}
```

```
    <h1>{{ section.title }}</h1>
```

```
{% for story in story_list %}
```

```
    <h2>
```

```
        <a href="{{ story.get_absolute_url }}">
```

```
            {{ story.headlinelupper }}
```

```
        </a>
```

```
    </h2>
```

```
    <p>{{ story.teaseltruncatewords:"100" }}</p>
```

```
{% endfor %}
```

```
{% endblock %}
```

4. 템플릿 코멘트

HTML 문서 상에서 주석이 필요할 때 사용하며 장고에서는 두 가지 형식의 코멘트 형식을 제공한다.

한 줄

```
{# 주석 내용 #}
```

개행 허용되지 않음.

여러 줄

```
{% comment %}
```

주석 내용

```
{% endcomment %}
```