



Django WebApp

장고란?

파이썬 기반으로 작성된 오픈소스 웹 어플리케이션 프레임워크
프레임워크 :

빠대, 골조라고도 하며 프로그램을 개발하는 데에 있어서 사용되는 기본 개념 구조
즉, 파이썬 프로그래밍 언어를 기반으로 한 동적인 웹을 작성하는 데에 있어 장고라는
기본 개념 구조 요소를 이용하여 개발

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

공식 사이트에 개발에 관련한 기술 문서들을 제공합니다.

장고의 특징

웹 개발에 있어서 번거로운 요소들을 새로 개발할 필요 없이 내장된
기능만을 이용해 빠른 개발을 할 수 있다는 장점

MTV패턴 – M(Model),(Template) , V(View)

MVC패턴 – M(model),V(Template), C(Controller-views.py)

장고를 사용하는 사이트

- 인스타그램
- NASA
- Disqus
- 모질라

MVT 패턴

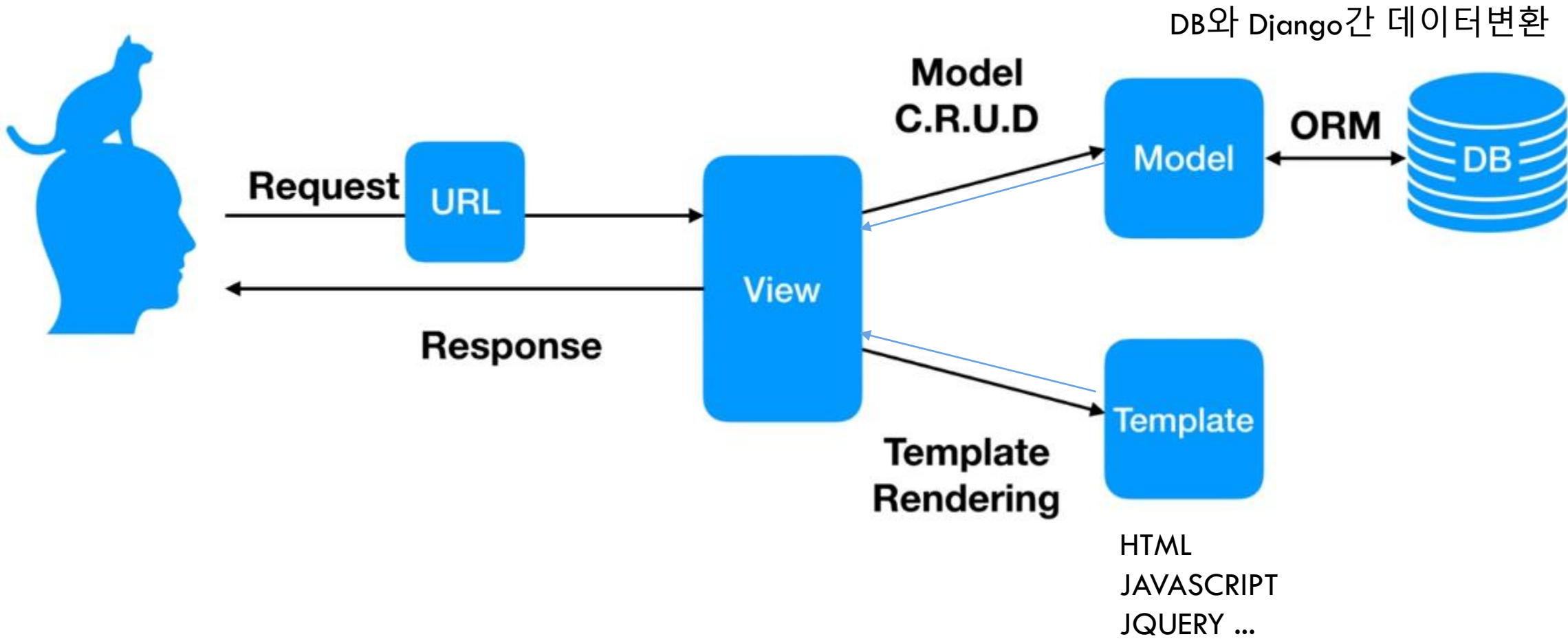
Model + View + Template

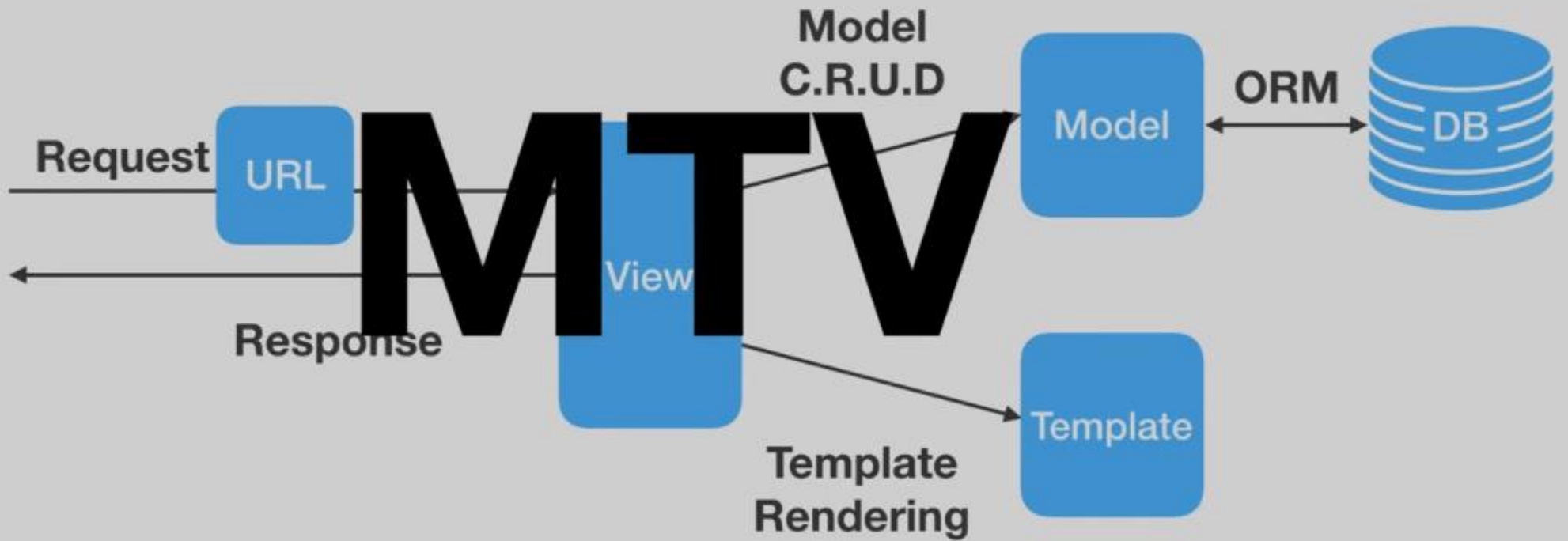
MTV 패턴이라고도 함

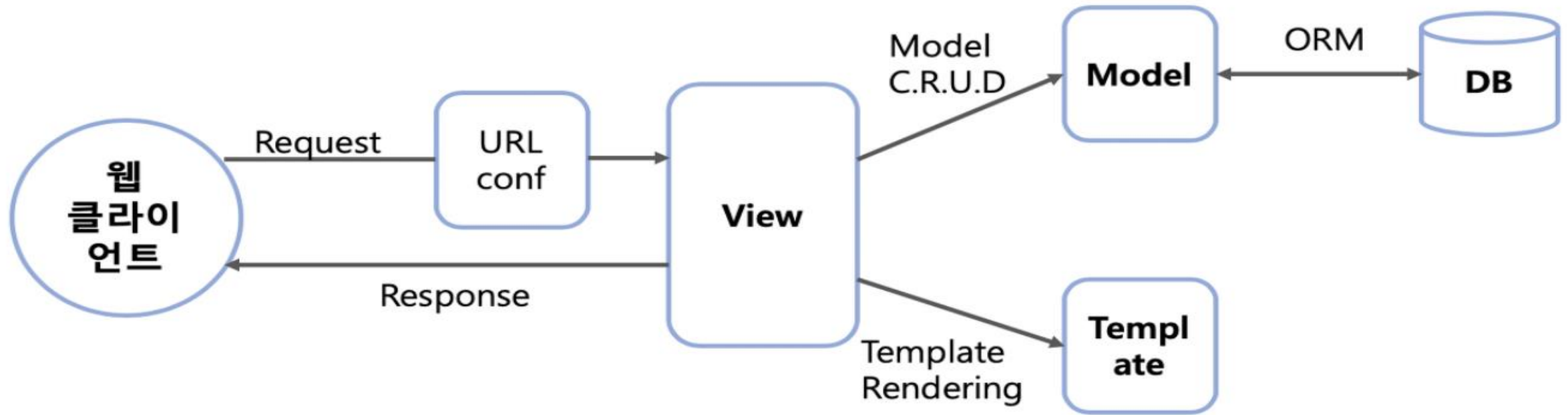
웹 어플리케이션을 개발하는 데에 있어서 영역을 크게 위의 3가지로 나눈 것

➔ 독립적으로 개발

➔ 3가지로 나누어서 개발을 하는 데에 있어서 딱히 정해진 순서는 없음



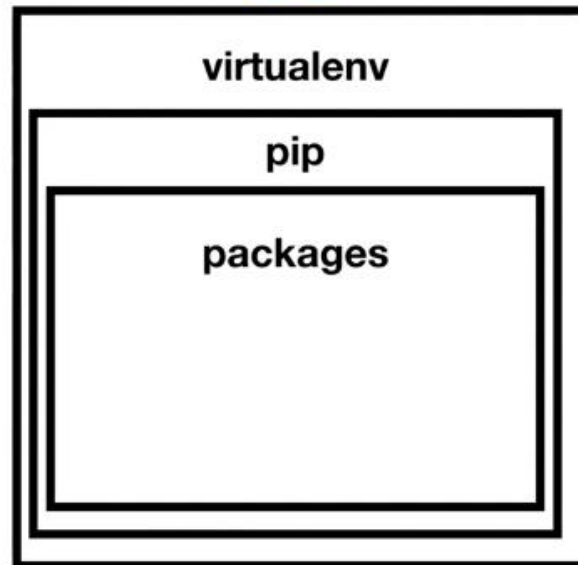




웹 클라이언트의 요청을 받고, 장고에서 MVT 패턴에 따라 처리하는 과정을 요약하면 다음과 같다.

1. 클라이언트로부터 요청을 받으면 URLconf를 이용하여 URL을 분석한다.
2. URL 분석 결과를 통해 해당 URL에 대한 처리를 담당할 뷰를 결정한다.
3. 뷰는 자신의 로직을 실행하면서 만일 데이터 베이스 처리가 필요하면 모델을 통해 처리하고 그 결과를 반환받는다.
4. 뷰는 자신의 로직 처리가 끝나면 템플릿을 사용하여 클라이언트에 전송할 HTML 파일을 생성한다.
5. 뷰는 최종 결과로 HTML 파일을 클라이언트에게 보내 응답한다.

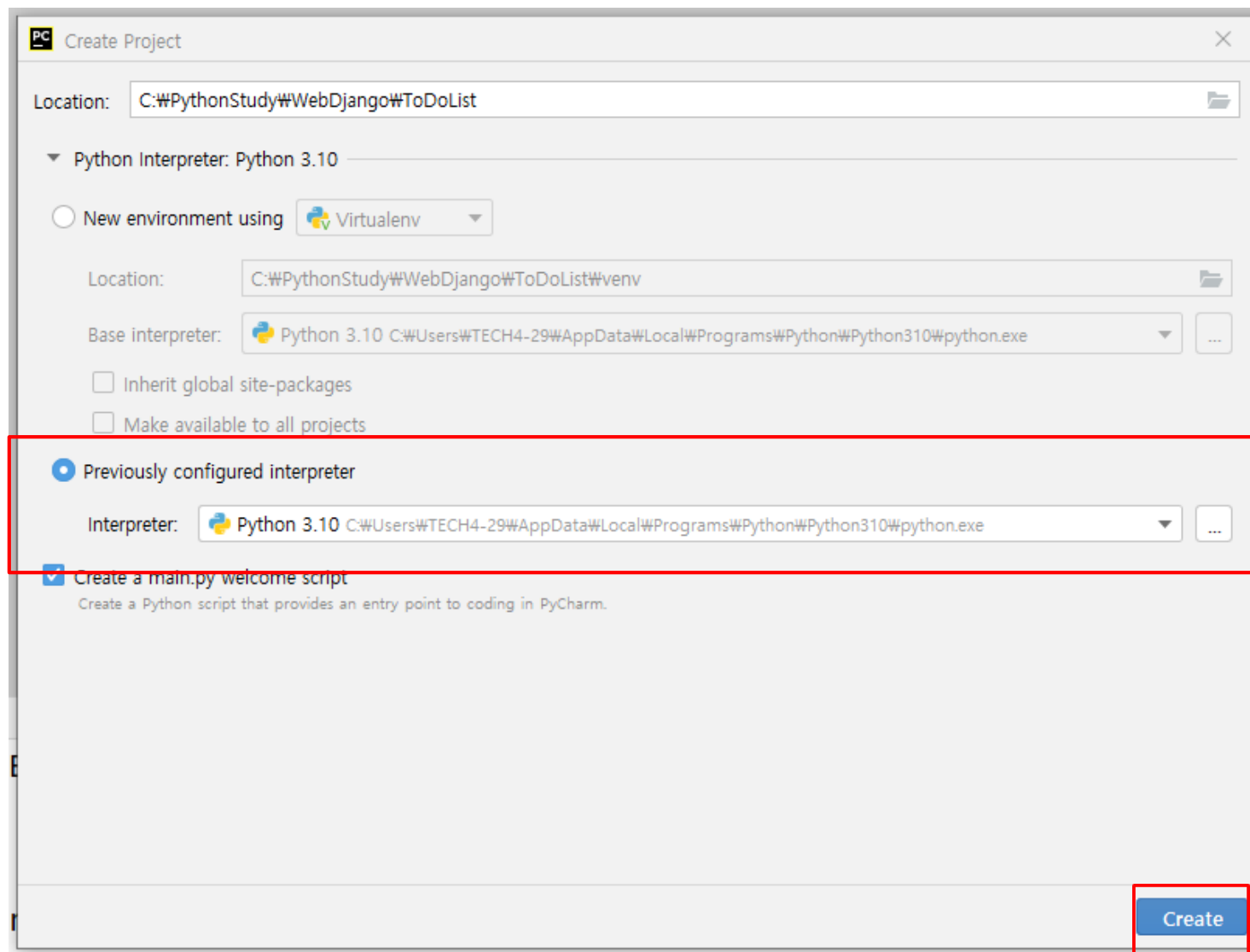
환경 설정





무조건 생성해보는 Django 프로젝트

프로젝트 생성



The image shows the 'Create Project' dialog box in PyCharm. The 'Location' field is set to 'C:\PythonStudy\WebDjango\ToDoList'. Under 'Python Interpreter: Python 3.10', the 'New environment using' option is selected with 'Virtualenv' as the backend. The 'Location' for the new environment is 'C:\PythonStudy\WebDjango\ToDoList\venv'. The 'Base interpreter' is 'Python 3.10 C:\Users\TECH4-29\AppData\Local\Programs\Python\Python310\python.exe'. The 'Previously configured interpreter' option is selected, and its 'Interpreter' is also 'Python 3.10 C:\Users\TECH4-29\AppData\Local\Programs\Python\Python310\python.exe'. The 'Create a main.py welcome script' checkbox is checked. The 'Create' button is at the bottom right.

PC Create Project

Location: C:\PythonStudy\WebDjango\ToDoList

Python Interpreter: Python 3.10

☐ New environment using Virtualenv

Location: C:\PythonStudy\WebDjango\ToDoList\venv

Base interpreter: Python 3.10 C:\Users\TECH4-29\AppData\Local\Programs\Python\Python310\python.exe

☐ Inherit global site-packages

☐ Make available to all projects

☒ Previously configured interpreter

Interpreter: Python 3.10 C:\Users\TECH4-29\AppData\Local\Programs\Python\Python310\python.exe

☒ Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

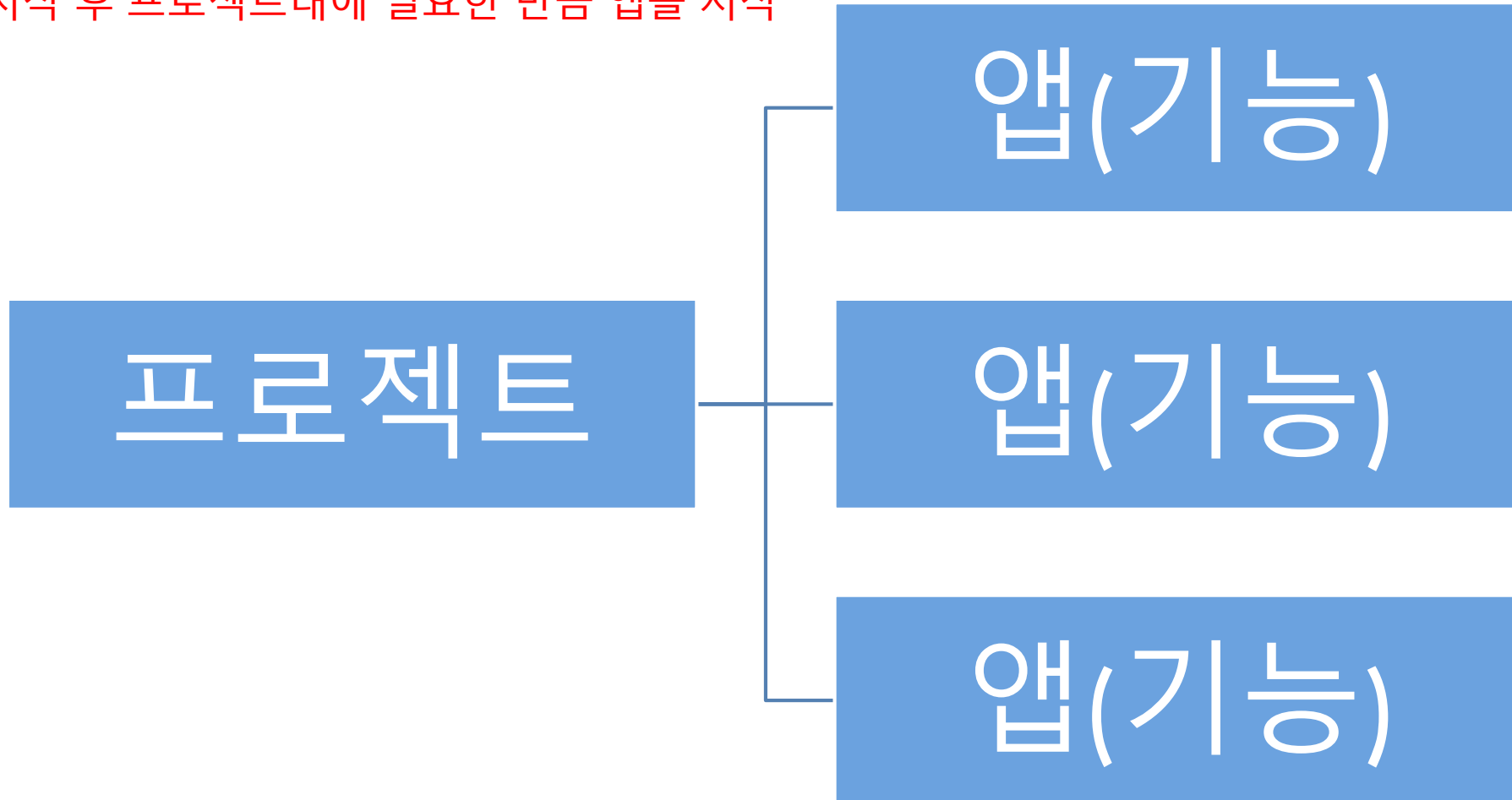
Create

Django install

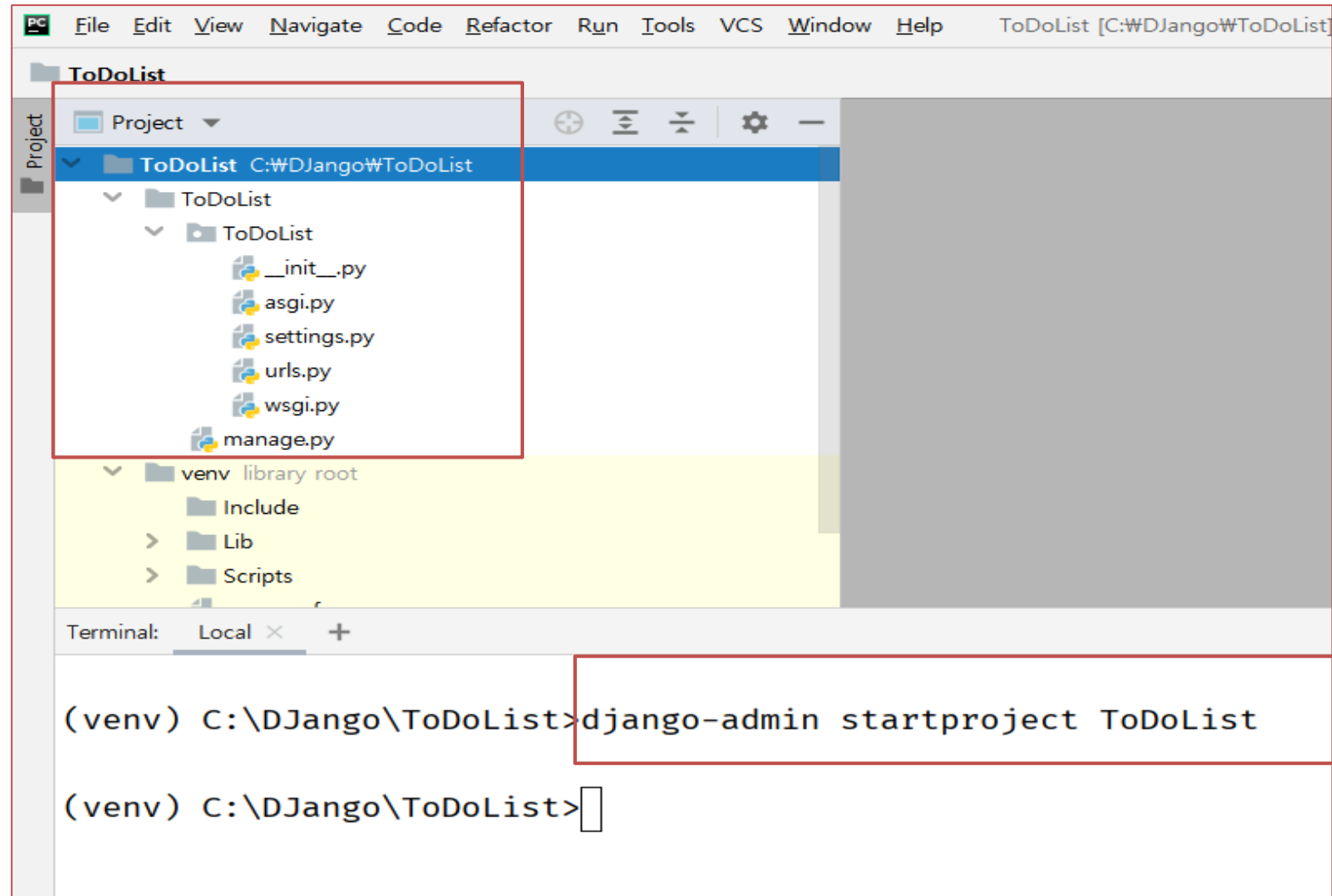
```
Terminal: Local x +  
  
(venv) C:\DJango\ToDoList>pip install django  
Collecting django  
  Downloading Django-3.2.6-py3-none-any.whl (7.9 MB)  
    |████████████████████████████████████████| 7.9 MB 1.1 MB/s  
Collecting pytz  
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)  
Collecting asgiref<4,>=3.3.2  
  Using cached asgiref-3.4.1-py3-none-any.whl (25 kB)  
Collecting sqlparse>=0.2.2  
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)  
Installing collected packages: sqlparse, pytz, asgiref, django  
Successfully installed asgiref-3.4.1 django-3.2.6 pytz-2021.1 sqlparse-0.4.1  
  
(venv) C:\DJango\ToDoList>
```

djang 프로젝트 구조

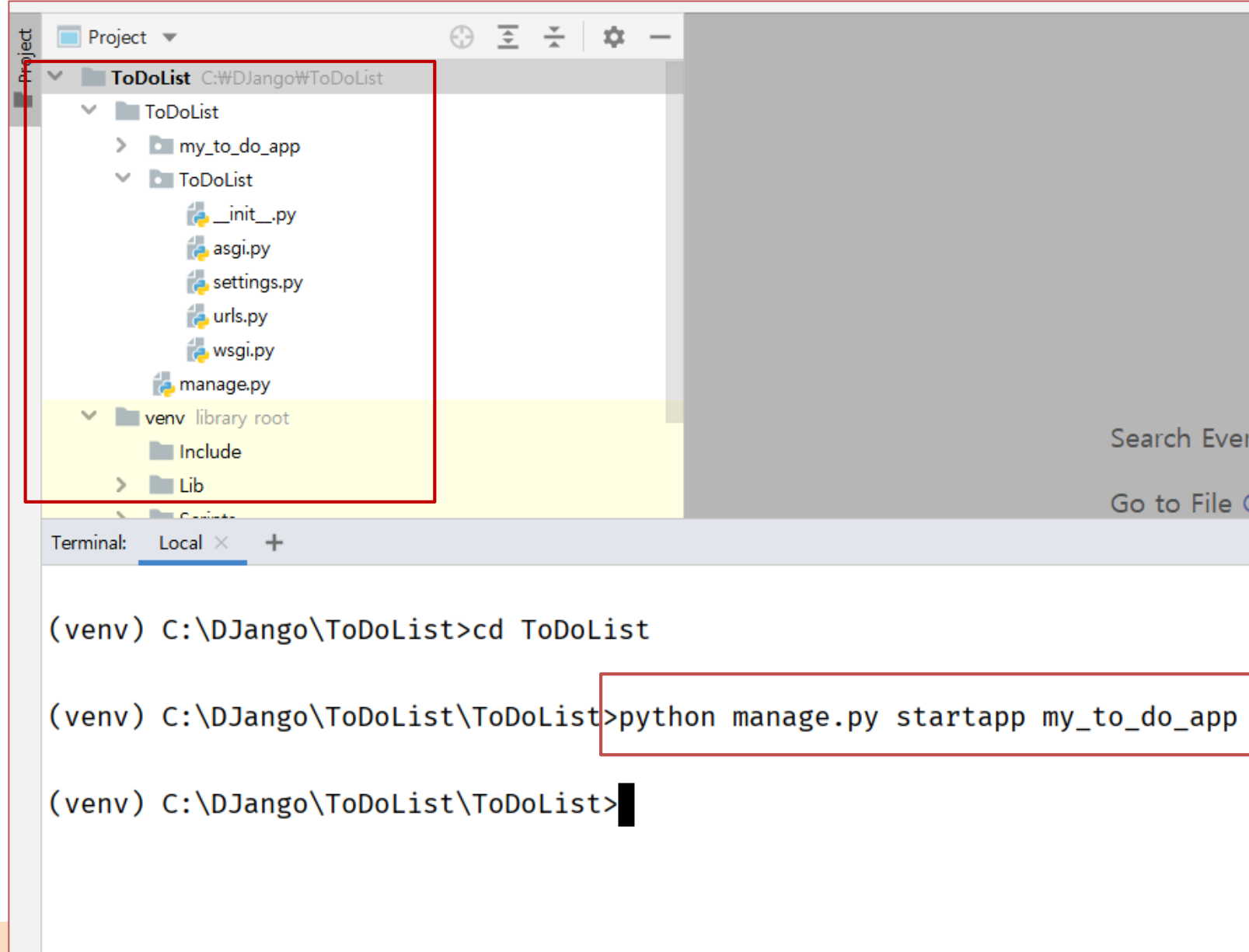
프로젝트 시작 후 프로젝트내에 필요한 만큼 앱을 시작



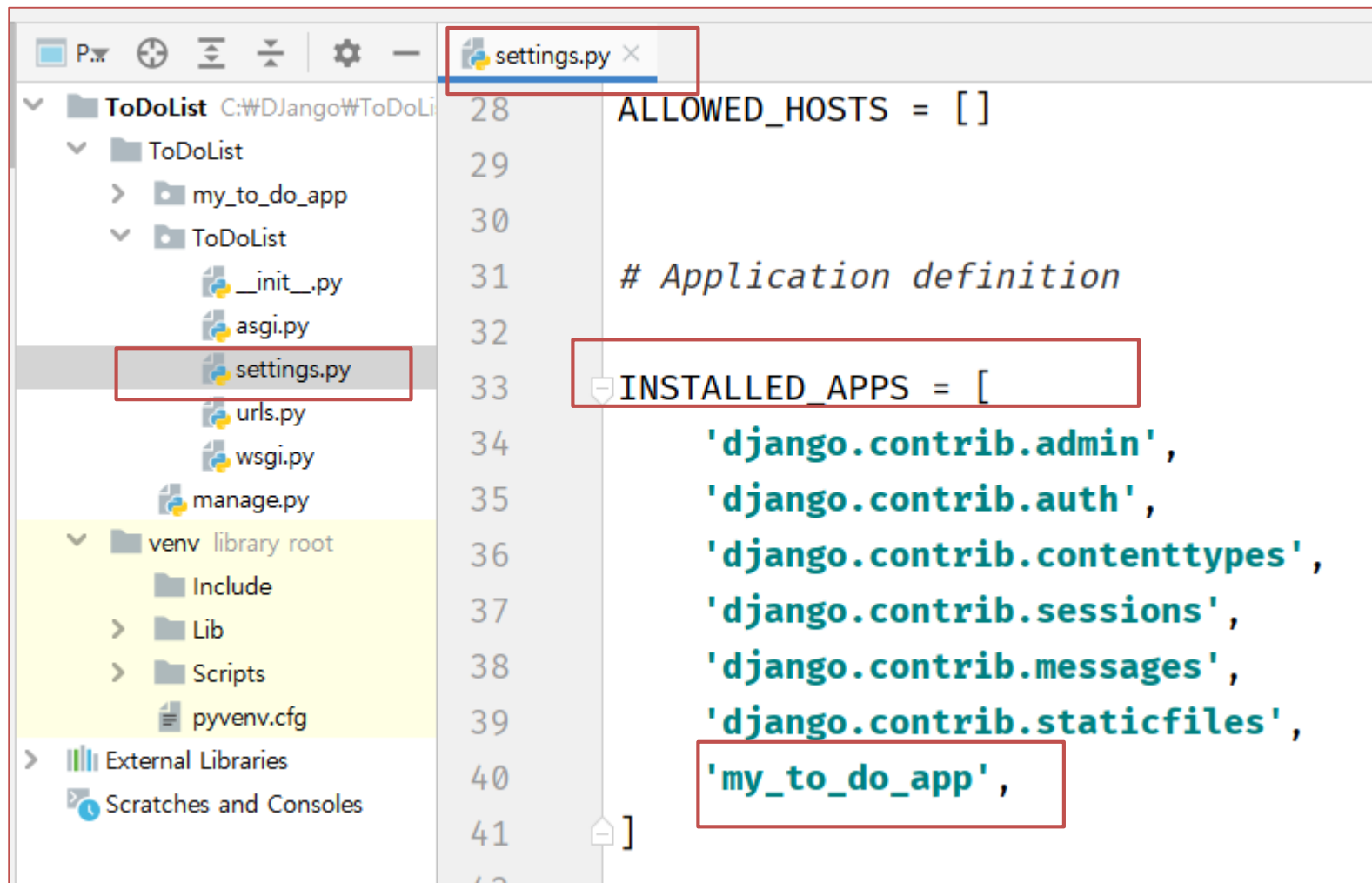
django 프로젝트 시작



Django app 시작



setting.py url에 앱 추가-my_to_do_app



Django migrate

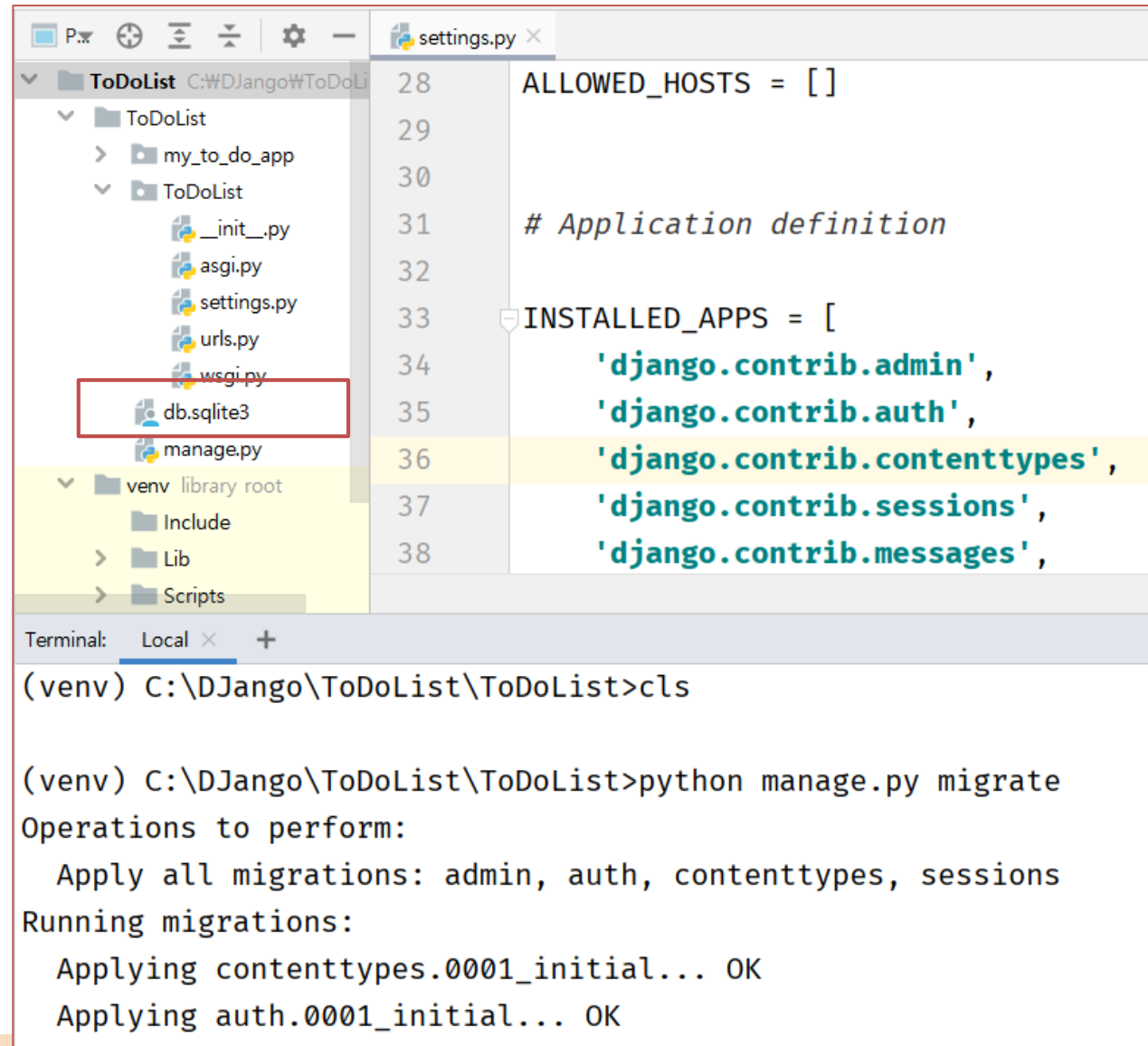
- ▶ 마이그레이션(Migration)이란 데이터베이스의 스키마(Schema)를 관리하기 위한 방법
- ▶ 데이터베이스에 테이블, 필드 등의 변경이 발생했을 때 지정된 데이터베이스에 적용하는 과정을 의미
- ▶ 현재 모델(model.py)은 정의만 되어있을 뿐, 데이터베이스를 생성하고 적용하지 않았음
- ▶ 마이그레이션을 통해 데이터베이스를 생성하고 모델의 생성, 변경, 삭제 등에 따라 작업 내역을 관리하고 데이터베이스를 최신화 할 수 있다

migrate

프로젝트 생성 후 처음 실행하는 기본 마이그레이션

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
```

db 자동 추가



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'ToDoList' with a subdirectory 'my_to_do_app' containing a 'ToDoList' folder. Inside this folder, 'db.sqlite3' is highlighted with a red box. The main editor window shows the 'settings.py' file with the following code:

```
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
```

The terminal window shows the following commands and output:

```
(venv) C:\DJango\ToDoList\ToDoList>cls

(venv) C:\DJango\ToDoList\ToDoList>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
```

migrate

- ▶ `python manage.py migrate`으로 현재 프로젝트의 마이그레이션을 진행함
- ▶ 마이그레이션 진행 시, 장고 프로젝트에서 사용하는 11개의 기본 테이블이 생성됨
- ▶ `sqlite3` 데이터베이스의 경우 생성되는 테이블은 아래와 같다
 - ▶ `auth_group`, `auth_group_permissions`, `auth_permission`, `auth_user`, `auth_user_groups`,
`auth_user_user_permissions`, `django_admin_log`, `django_content_type`, `django_migrations`,
`django_session`, `sqlite_sequence`

기본 db 파일 생성

C:\Users\mkm05\PycharmProjects\first_django 디렉터리

2021-01-20	오전	12:16	<DIR>	.
2021-01-20	오전	12:16	<DIR>	..
2021-01-20	오전	12:22	<DIR>	.idea
2021-01-19	오후	11:27	<DIR>	bookmark
2021-01-20	오전	12:23	<DIR>	config
2021-01-19	오후	11:27		131,072 db.sqlite3
2021-01-19	오후	11:19		553 manage.py
2021-01-20	오전	12:26	<DIR>	polls
2021-01-19	오후	11:14	<DIR>	venv
2개 파일				131,625 바이트
7개 디렉터리				74,359,238,656 바이트 남음

migrate/migrations

- ▶ `model.py`에 정의된 모델의 생성/변경 내역을 히스토리 관리, 데이터베이스에 적용 등과 같은 기능을 제공하여 손쉽게 데이터베이스의 구조를 바꿀 수 있음
- ▶ 장고에서는 데이터베이스를 제공하고 있기 때문에 제공되는 데이터베이스(sqlite)를 사용할 수 있으며, 해당 db 에서 사용할 테이블 생성

Django 서버 실행

Terminal: Local × +

```
(venv) C:\Users\mkm05\PycharmProjects\first_django>python manage.py runserver  
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
January 19, 2021 - 23:49:40
```

```
Django version 2.1, using settings 'config.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CTRL-BREAK.
```



Django 디폴트페이지 확인

127.0.0.1:8000

django

[View release notes for Django](#)



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

프로젝트 환경설정

setting.py

```
25  # SECURITY WARNING: don't run with debug turned on in production!
26  DEBUG = True #True : 개발모드 False :운영모드
27
28  # ALLOWED_HOSTS = ['localhost', '127.0.0.1', '192.168.56.101'] # 기동할 서버의 ip나열
29  # 위에 나열된 ip로 서버가 운영될 수 있음
30  # ALLOWED_HOSTS = ['*'] # 모든 ip 로 서버 운영 가능
31  ALLOWED_HOSTS = [] #모든 ip 서버 운영 가능
32
33  # Application definition
```

환경설정 2

```
33 # Application definition
```

```
34
```

```
35 INSTALLED_APPS = [
```

```
36     'django.contrib.admin',
```

```
37     'django.contrib.auth',
```

```
38     'django.contrib.contenttypes',
```

```
39     'django.contrib.sessions',
```

```
40     'django.contrib.messages',
```

```
41     'django.contrib.staticfiles',
```

```
42     'bookmark',
```

```
43     'polls.apps.PollsConfig',
```

```
44 ]
```

django에서 사용할 app

개발자가 추가한 app

프로젝트에 포함되는 모든 app는 모두 설정파일에 등록해야 함

환경설정 3 (db)

```
77 # Database
78 # https://docs.djangoproject.com/en/2.1/ref/settings/#databases
79
80 DATABASES = {
81     'default': {
82         'ENGINE': 'django.db.backends.sqlite3',
83         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
84     }
85 }
```

장고는 default로 sqlite3을 사용하도록 설정하고 있음. 자동 설치됨
다른 db사용하고 싶으면 여기서 변경

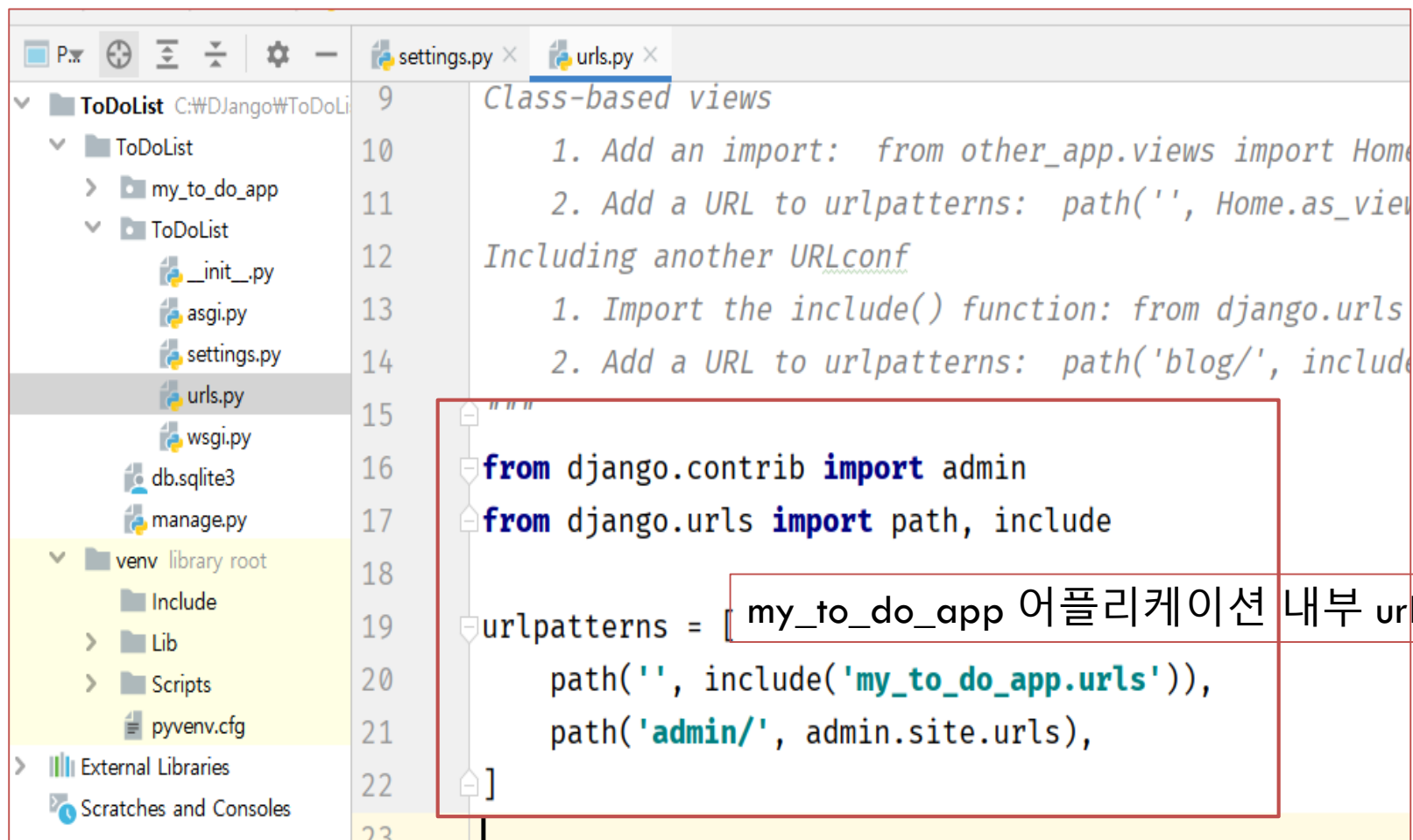
환경설정 4 (타임존지정)

```
107 # Internationalization
108 # https://docs.djangoproject.com/en/2.1/topics/i18n/
109
110 LANGUAGE_CODE = 'en-us'
111
112 # TIME_ZONE = 'UTC'
113 TIME_ZONE = 'Asia/Seoul'
114
115 USE_I18N = True
116
117 USE_L10N = True
118
119 USE_TZ = True
```

세계 표준시에서 seoul로

url 설정

- ▶ ToDoList -> urls.py (프로젝트 전체 url을 담당하는 파일)



The screenshot shows a code editor with two tabs: 'settings.py' and 'urls.py'. The 'urls.py' tab is active, displaying the following code:

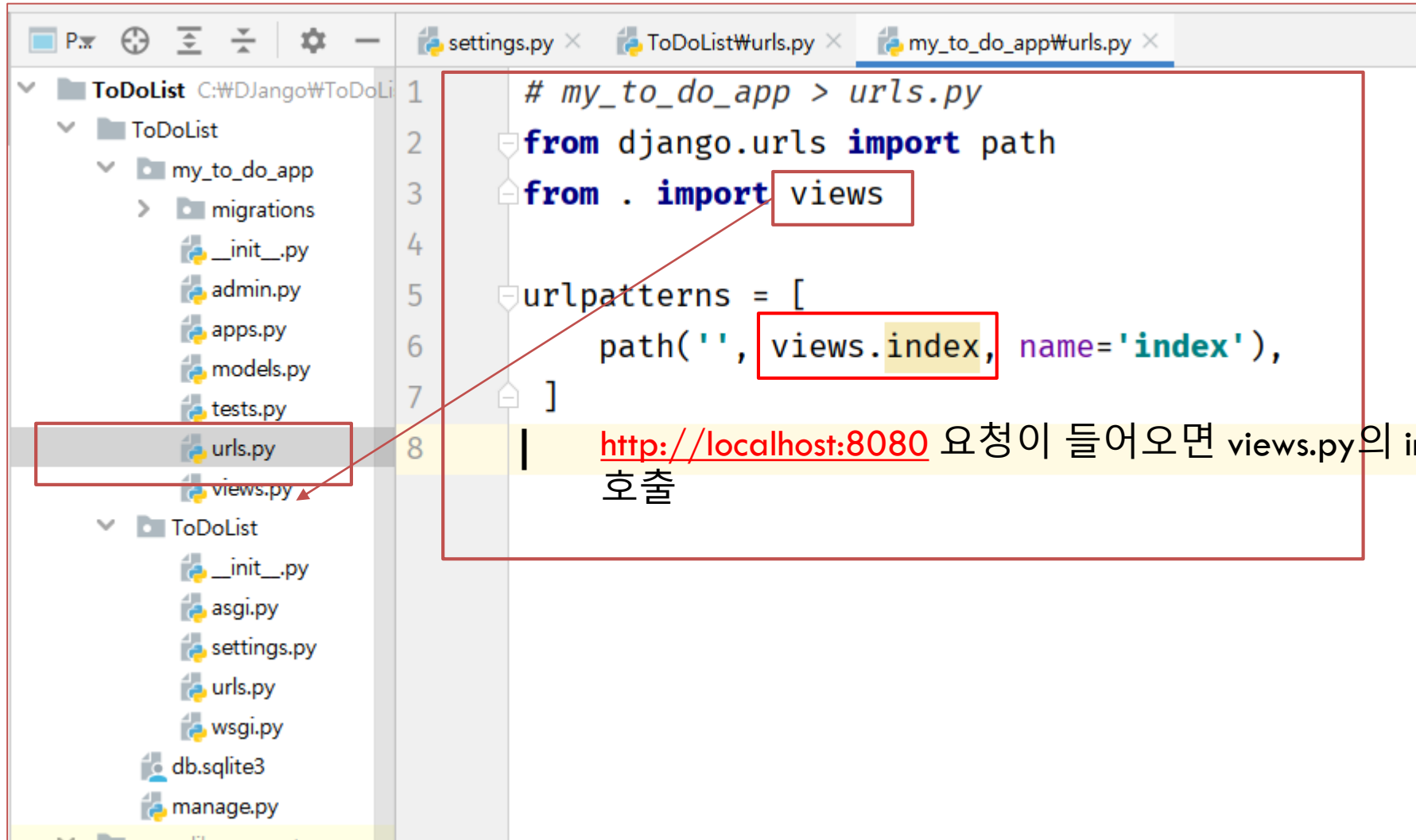
```
9  Class-based views
10      1. Add an import: from other_app.views import Home
11      2. Add a URL to urlpatterns: path('', Home.as_view)
12  Including another URLconf
13      1. Import the include() function: from django.urls
14      2. Add a URL to urlpatterns: path('blog/', include
15
16  """
17  from django.contrib import admin
18  from django.urls import path, include
19  urlpatterns = [
20      path('', include('my_to_do_app.urls')),
21      path('admin/', admin.site.urls),
22  ]
23
```

The left sidebar shows the project structure:

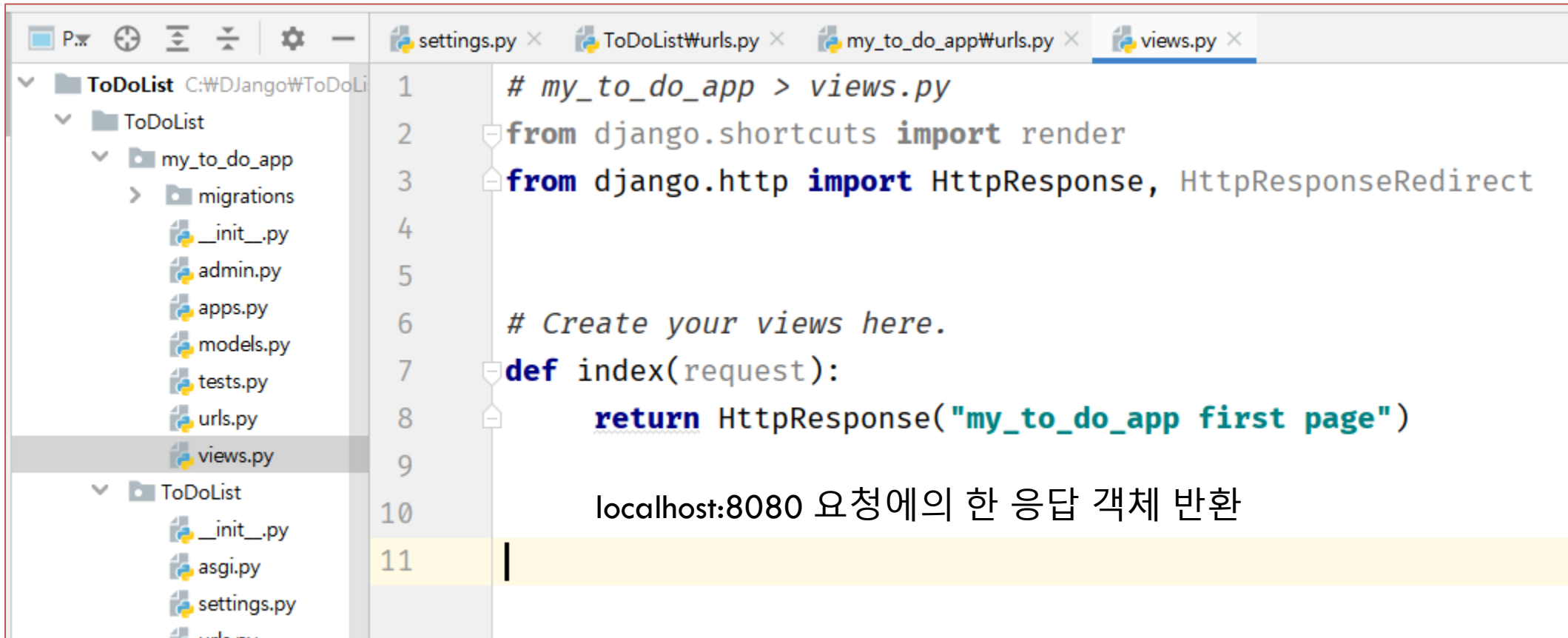
- ToDoList C:\#Django\ToDoList
 - ToDoList
 - my_to_do_app
 - ToDoList
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - db.sqlite3
 - manage.py
 - venv library root
 - Include
 - Lib
 - Scripts
 - pyvenv.cfg
 - External Libraries
 - Scratches and Consoles

A red box highlights the code in lines 16-22, which is annotated with the text: "my_to_do_app 어플리케이션 내부 urls.py를 포함시키는 코드".

my_to_do_app urls.py 추가



view 연결(views.py)



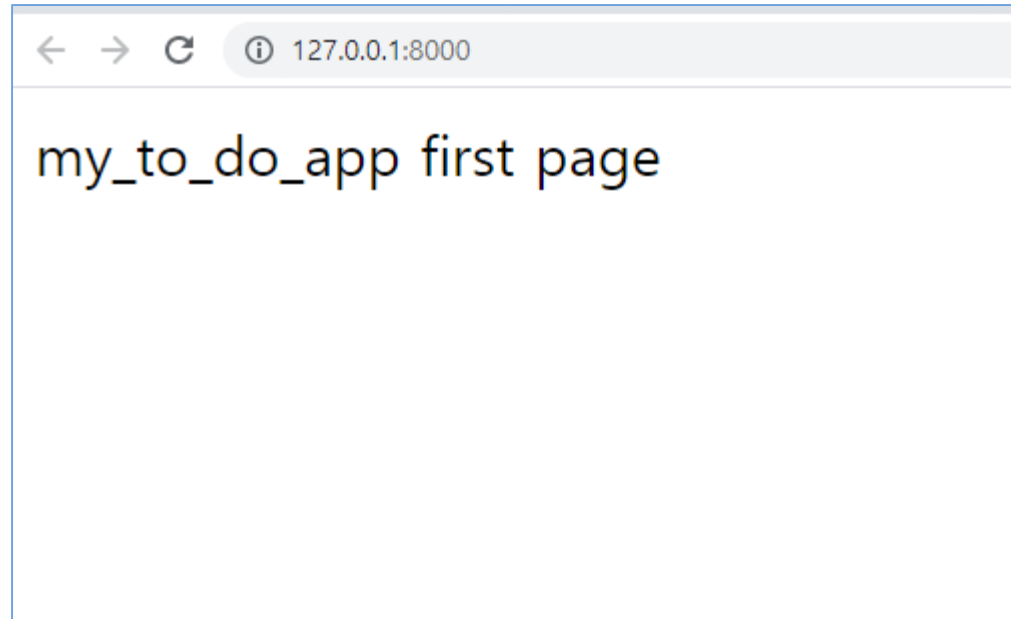
```
1 # my_to_do_app > views.py
2 from django.shortcuts import render
3 from django.http import HttpResponse, HttpResponseRedirect
4
5
6 # Create your views here.
7 def index(request):
8     return HttpResponse("my_to_do_app first page")
9
10
11
```

localhost:8080 요청에의 한 응답 객체 반환

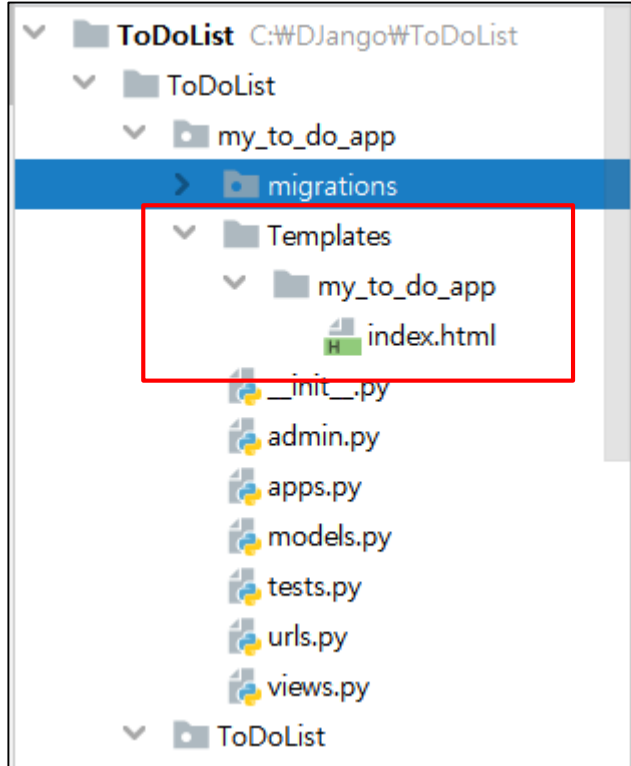
서버 실행 후 확인

```
(venv) C:\DJango\ToDoList\ToDoList>manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 07, 2021 - 22:39:57
Django version 3.2.6, using settings 'ToDoList.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



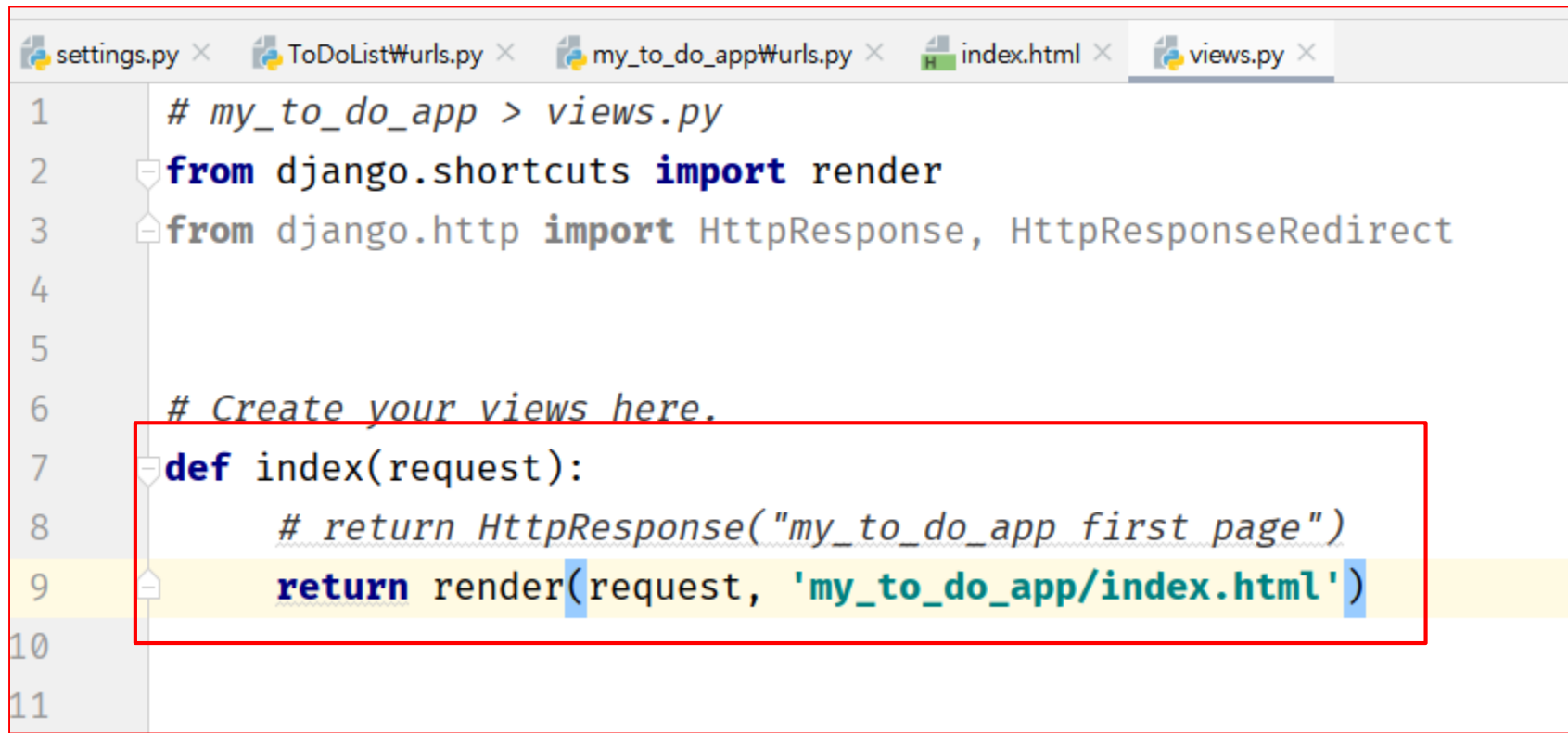
html 템플릿 사용하기(front-end 파일)



제공파일 사용
front-end 관련 파일들은 app>Templates>app>*.html
의 경로를 따른다(규칙)

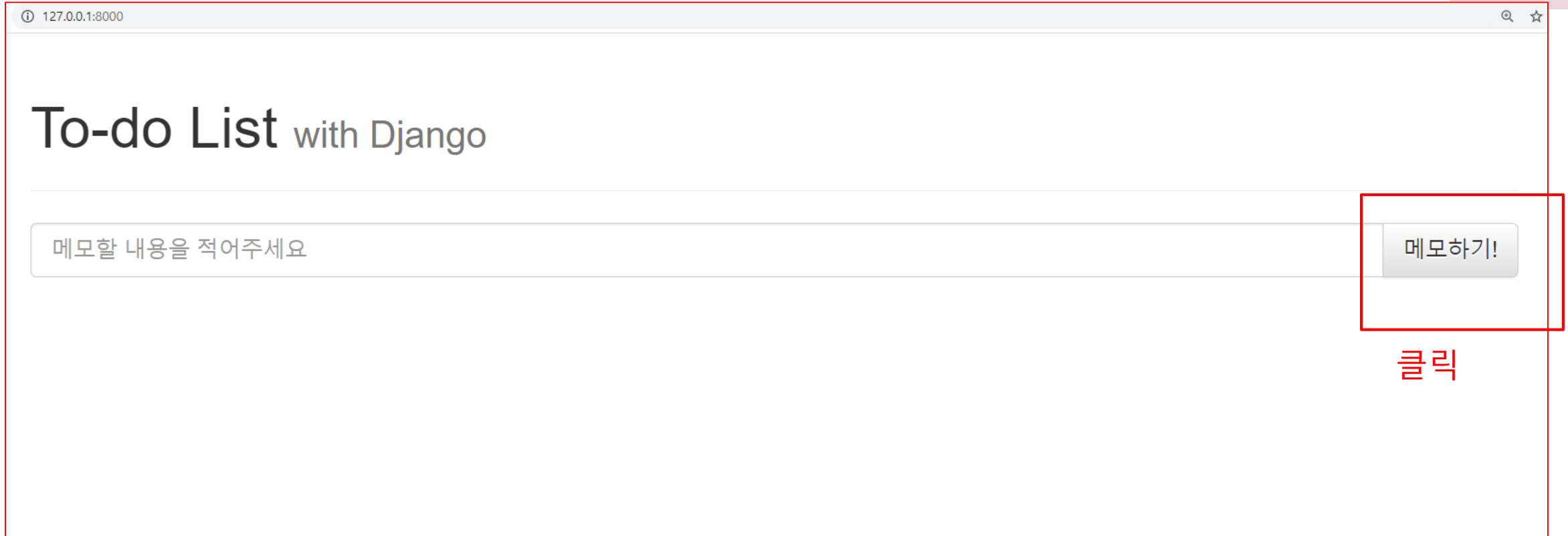
호출 시 index.html 파일로 응답 반환

▶ views.py 수정



```
settings.py × ToDoListWurls.py × my_to_do_appWurls.py × index.html × views.py ×
1      # my_to_do_app > views.py
2      from django.shortcuts import render
3      from django.http import HttpResponse, HttpResponseRedirect
4
5
6      # Create your views here.
7      def index(request):
8          # return HttpResponse("my_to_do_app first page")
9          return render(request, 'my_to_do_app/index.html')
10
11
```

서버 새로고침 후 확인



결과확인

Page not found (404)

Request Method: POST

Request URL: `http://127.0.0.1:8000/createTodo/`

버튼에 의해 요청된 url

404 에러는 해당 url이 서버에 존재하지 않는다는 의미

url을 등록해야 함

Using the URLconf defined in `ToDoList.urls`, Django tried these URL patterns, in this order:

1. `[name='index']`
2. `admin/`

The current path, `createTodo/`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to


index.html 확인

```
settings.py x ToDoListWurls.py x my_to_do_appWurls.py x index.html x views.py x
34 </head>
35 <body>
36   <div class="container">
37     <div class="header">
38       <div class="page-header">
39         <h1>To-do List <small>with Django</small></h1>
40       </div>
41     </div>
42     <div class="content">
43       <div class="messageDiv">
44         <form action="./createTodo/" method="POST">{% csrf_token %}
45           <div class="input-group">
46             <input id="todoContent" name="todoContent" type="text" class="form-control" p
47             <span class="input-group-btn">
48               <button class="btn btn-default" type="submit">메모하기!</button>
49             </span>
50           </div>
51         </form>
52       </div>
53
```

도메인 /createTodo/를 요청

<http://127.0.0.1:8000/createTodo/> url 등록

▶ my_to_do_app urls.py 에 해당 url 등록



The screenshot shows a Django project's file explorer on the left and a code editor on the right. The file explorer shows the project structure, with the `my_to_do_app` folder selected. The code editor shows the `my_to_do_app/urls.py` file, which contains the following code:

```
1 # my_to_do_app > urls.py
2 from django.urls import path
3 from . import views
4
5 urlpatterns = [
6     path('', views.index, name='index'),
7     path('createTodo/', views.createTodo, name='createTodo'),
8 ]
```

The `path('createTodo/', views.createTodo, name='createTodo')` line is highlighted with a red box. Below the code editor, there is a note: `views.py` 에 `createTodo` 함수 생성해야 함.

views.py 수정

```
settings.py x ToDoListWurls.py x my_to_do_appWurls.py x index.html x views.py x
1      # my_to_do_app > views.py
2      from django.shortcuts import render
3      from django.http import HttpResponse, HttpResponseRedirect
4
5
6      # Create your views here.
7      def index(request):
8          # return HttpResponse("my_to_do_app first page")
9          return render(request, 'my_to_do_app/index.html')
10
11      def createTodo(request):
12          return HttpResponse("createTodo 작성 합시다!!!!")
13
14
```

확인

← → ↻ ⓘ 127.0.0.1:8000/createTodo/

createTodo 작성 합시다!!!!

ⓘ 127.0.0.1:8000

🔍 ☆

To-do List with Django

메모할 내용을 적어주세요

메모하기!

클릭

createToDo 함수 기능 추가

```
settings.py x ToDoListWurls.py x my_to_do_appWurls.py x index.html x views.py x
34 </head>
35 <body>
36   <div class="container">
37     <div class="header">
38       <div class="page-header">
39         <h1>To-do List <small>with Django</small></h1>
40       </div>
41     </div>
42     <div class="content">
43       <div class="messageDiv">
44         <form action="/createToDo/" method="POST">{% c
45         <div class="input-group">
46           <input id="todoContent" name="todoContent" type="text" class="form-control" p
47           <span class="input-group-btn">
48             <button class="btn btn-default" type="submit">메모하기!</button>
49           </span>
50         </div>
51       </form>
52     </div>
53
```

사용자가 입력한 memo 값을 가져와서
처리해야 함 => name속성의 변수로 전달됨

createToDo 함수 기능 추가

```
settings.py x ToDoListUrls.py x my_to_do_appUrls.py x index.html x views.py x
1  # my_to_do_app > views.py
2  from django.shortcuts import render
3  from django.http import HttpResponseRedirect
4
5
6  # Create your views here.
7  def index(request):
8      # return HttpResponseRedirect("my_to_do_app first page")
9      return render(request, 'my_to_do_app/index.html')
10
11  def createToDo(request):
12      user_input_str = request.POST['todoContent']
13      return HttpResponseRedirect("createToDo 작성 합시다!!!! =>" + user_input_str)
14
15
16
```

input tag name 속성 속성값

결과 확인

← → ↻ ⓘ 127.0.0.1:8000/createTodo/

createTodo 작성 합시다!!!! => todoContent name에 담겨서 서버로 넘어갑니다

← → ↻ ⓘ 127.0.0.1:8000

🔍 ☆

To-do List with Django

todoContent name에 담겨서 서버로 넘어갑니다

메모하기!

사용자가 입력한 문자열을 가공하여 클라이언트에게 반환 => 브라우저 출력

DB 처리

- ▶ 사용자가 입력한 문자열을 데이터베이스에 저장하기
- ▶ 데이터베이스에 저장된 내용을 보여주기

DB에 저장하기 (views.py)

```
1 # my_to_do_app > views.py
2 from django.shortcuts import render
3 from django.http import HttpResponse, HttpResponseRedirect
4 from django.urls import reverse
5 from .models import *
6
7 # Create your views here.
8 def index(request):
9     # return HttpResponse("my_to_do_app first page")
10     return render(request, 'my_to_do_app/index.html')
11
12 def createTodo(request):
13     user_input_str = request.POST['todoContent']
14     new_todo = Todo(content=user_input_str)
15     new_todo.save()
16     return HttpResponse("DB에 저장되었어요 =>" + user_input_str)
```

model에 대입하고
db에 저장



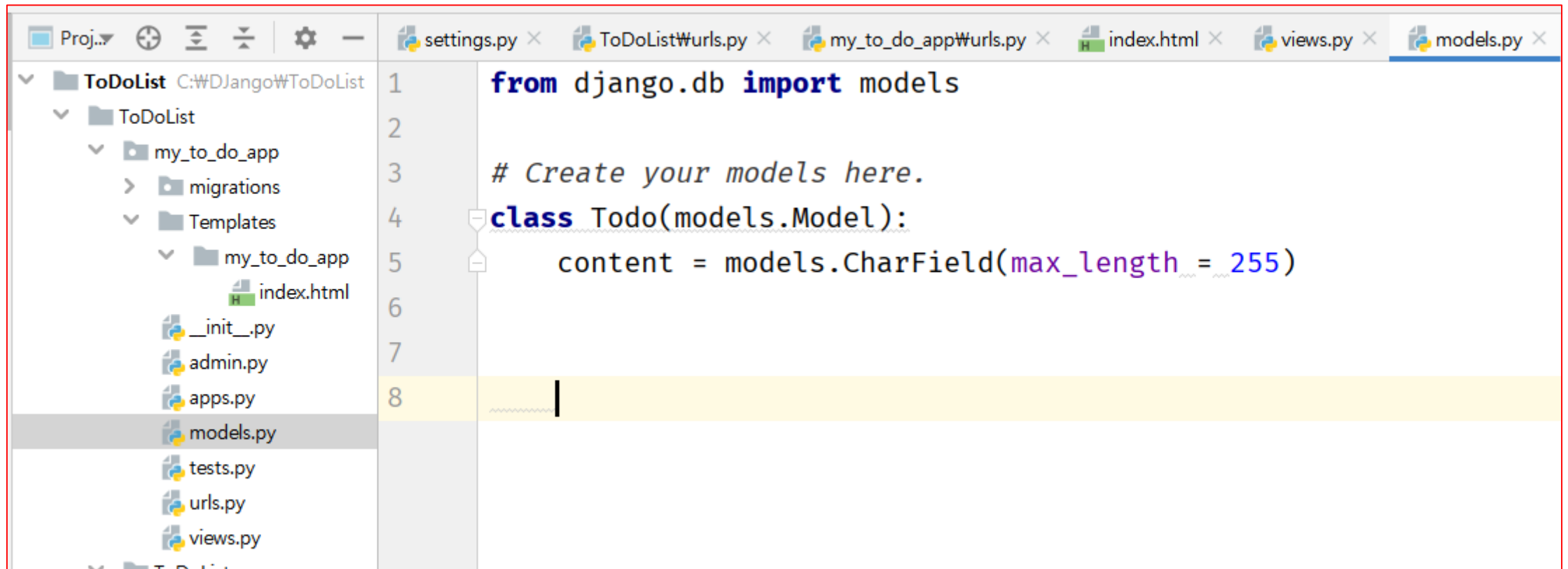
DB 저장(모델관련 설정)

model 관련 코딩

- ▶ `models.py` – 테이블 정의
- ▶ `admins.py` – 정의된 테이블이 admin 화면에 보이게 함
- ▶ `manage.py makemigrations` – 데이터베이스에 변경이 필요한 사항을 추출
- ▶ `manage.py migrate` – 데이터베이스에 변경사항을 반영
- ▶ `manage.py runserver` – 현재까지 작업 개발을 웹서버로 확인

모델 생성(db 생성)

▶ models.py



```
1 from django.db import models
2
3 # Create your models here.
4 class Todo(models.Model):
5     content = models.CharField(max_length = 255)
6
7
8
```

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'ToDoList' folder containing 'my_to_do_app' and its sub-files. The 'models.py' file is selected. The code editor shows the Django model definition for 'Todo'.

models.py에서 정의할 수 있는 데이터

- ▶ CharField : 문자열
- ▶ DateField : 날짜
- ▶ EmailField : e-mail =>
 - ▶ EmailValidator라는 것을 통해 입력되는 문자열이 이메일 형식인지를 자동 체크해 줌
 - ▶ 형식에서 어긋하면 저장과정에서 오류 발생
- ▶ FileField : 파일을 저장할 수 있는 데이터 타입
 - ▶ 파일의 이름을 저장하며 실제 파일은 upload_to라는 옵션에 지정되는 위치에 저장
- ▶ TextField : 글자수의 제한이 없음
- ▶ IntegerField : 숫자
- ▶ BooleanField : T/F

DB에 반영

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py makemigrations
```

```
Migrations for 'my_to_do_app':
```

```
  my_to_do_app\migrations\0001_initial.py
```

```
    - Create model Todo
```

```
(venv) C:\DJango\ToDoList\ToDoList>
```

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py migrate
```

```
Operations to perform:
```

```
  Apply all migrations: admin, auth, contenttypes, my_to_do_app, sessions
```

```
Running migrations:
```

```
  Applying my_to_do_app.0001_initial... OK
```

DB shell 에서 확인

```
Terminal: Local x +
(venv) C:\DJango\ToDoList\ToDoList>python manage.py dbshell
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
sqlite> .tables
auth_group                      django_admin_log
auth_group_permissions          django_content_type
auth_permission                 django_migrations
auth_user                      django_session
auth_user_groups               my_to_do_app_todo
auth_user_user_permissions
sqlite> PRAGMA table_info(my_to_do_app_todo)
...> ;
0|id|integer|1|1
1|content|varchar(255)|1|0
sqlite> select * from my_to_do_app_todo
...> ;
sqlite> █
```

실행확인

To-do List with Django

todoContent name에 담겨서 서버로 넘어갑니다

메모하기!

← → ↻ ⓘ 127.0.0.1:8000/createTodo/

DB에 저장되었어요 => todoContent name에 담겨서 서버로 넘어갑니다

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py dbshell
SQLite version 3.33.0 2020-08-14 13:23:32
Enter ".help" for usage hints.
sqlite> .tables
auth_group                django_admin_log
auth_group_permissions    django_content_type
auth_permission           django_migrations
auth_user                 django_session
auth_user_groups          my_to_do_app_todo
auth_user_user_permissions

sqlite> select * from my_to_do_app_todo;
1|todoContent name에 담겨서 서버로 넘어갑니다
sqlite>
```

데이터베이스에 저장된 내용을 보여주기

▶ 어디에 보여 줄 것인가?

To-do List with Django

메모할 내용을 적어주세요

메모하기!

완료

추가

저장 후 출력 FLOW

1. 사용자가 메모 입력 후 메모하기 클릭
2. 서버로 메모(todoContent)파라미터와 함께 서버에 요청
3. 서버는 해당 요청을 받은 후 <http://127.0.0.1:8000/createTodo/> 의 url을 확인함
 1. `path('createTodo', views.createTodo, name='createTodo')`,
 2. `views.py`의 `createTodo` 함수를 실행
 3. `createTodo`함수는 전송된 파라미터를 DB에 저장 후
 4. `views.index`는 DB에 있는 모든 메모를 갖고 와서 파라미터를 통해 `template(index.html)`에게 전달
 5. `template`은 전달된 파라미터로 동적 코드를 생성(rendering)
4. `index.html`코드 응답 반환: 위 모든 처리를 끝낸 후 `index.html`을 반환해야 함
 1. `views.index`에서 처리 후 `index.html` 반환(이 처리를 편하게 하기 위해 `redirect` 기능을 이용함)

views.py redirect 처리 기능 추가

```
settings.py x ToDoListUrls.py x my_to_do_appUrls.py x index.html x index(copy).html x views.py x models.py x
1 # my_to_do_app > views.py
2 from django.shortcuts import render
3 from django.http import HttpResponseRedirect, HttpResponseRedirect
4 from django.urls import reverse
5 from .models import *
6
7 # Create your views here.
8 def index(request):
9     # return HttpResponseRedirect("my_to_do_app first page")
10     return render(request, 'my_to_do_app/index.html')
11
12 def createTodo(request):
13     user_input_str = request.POST['todoContent']
14     new_todo = Todo(content=user_input_str)
15     new_todo.save()
16     return HttpResponseRedirect(reverse('index'))
17     # return HttpResponseRedirect("DB에 저장되었습니다 =>" + user_input_str)
18
```

확인

현재 화면이 지속 됨 : redirect 된 화면임

127.0.0.1:8000

To-do List with Django

메모할 내용을 적어주세요

메모하기!

완료

데이터베이스에 저장된 내용을 보여주기 db에서 입력된 memo 추출하기

▶ index.html로 redirect했으므로 views.py의 index 함수 수정

```
1 # my_to_do_app > views.py
2 from django.shortcuts import render
3 from django.http import HttpResponseRedirect, HttpResponseRedirect
4 from django.urls import reverse
5 from .models import * # models.py에서 생성된 객체 모두 import(class Todo)
6
7 # Create your views here.
8 def index(request):
9     todos = Todo.objects.all() # Todo 클래스 이용해 db에 저장된 모든 레코드 select
10    content = {'todos': todos} #content dict 생성
11    return render(request, 'my_to_do_app/index.html', content) #index.html 파일로 전달
12    # return HttpResponseRedirect("my_to_do_app first page")
13    # return render(request, 'my_to_do_app/index.html')
```

데이터베이스에 저장된 내용을 보여주기

html 수정

```
54 <div class="ToDoDiv">
55   <ul class="list-group">
56     {% for todo in todos %} 저장된 메모가 DICT형태로 반환 : 반복문 이용
57     <form action="/doneTodo/" method="GET">
58       <div class="input-group" name='todo1'>
59         <li class="list-group-item">{{ todo.content| }}</li>
60         <input type="hidden" id="todoNum" name="todoNum" value="{{ todo.id }}"></input>
61         <span class="input-group-addon">
62           <button type="submit" class="custom-btn btn btn-danger">완료</button>
63         </span>
64       </div>
65     </form>
66     {% endfor %} 마지막 data면 종료
67   </ul>
68 </div>
69 </div>
```

DB로부터 전송된 데이터를 html 파일로 전달 Django는 html파일의 {% %}안의 코드를 실행해 모두 html로 변환한 후 클라이언트로 전송함

{ } 템플릿 태그

- ▶ { } : html에 파이썬 코드를 추가할 수 있는 태그
 - ▶ {% %} : 파이썬 문법
 - ▶ {{ }} : 사용자에게 직접 보여주는 값

html 템플릿 확인

- View에서 처리된 data는 content라는 특수 딕셔너리 형태로 html 파일에 데이터를 넘겨줌

```
<ul class="list-group">
  {% for todo in todos %}
    {% if todo.isDone == False %}
      <form action="/doneTodo/" method="GET">
        <div class="input-group" name='todo1'>
          <li class="list-group-item">{{ todo.content }}</li>
          <input type="hidden" id="todoNum" name="todoNum" value="{{ todo.id }}"></input>
          <span class="input-group-addon">
            <button type="submit" class="custom-btn btn btn-danger">완료</button>
          </span>
        </div>
      </form>
    {% else %}
    {% endif %}
  {% endfor %}
```

실행 확인

To-do List with Django

메모할 내용을 적어주세요

메모하기!

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

CRUD

▶ Create

- ▶ 데이터를 만드는 기능 => 데이터 생성해 데이터베이스에 저장
 - ▶ 메모하기 버튼을 누르면 사용자가 입력한 글이 DB에 저장

DB 저장 확인

▶ dbshell에서 확인

```
sqlite> select * from my_to_do_app_todo;
1|todoContent name에 담겨서 서버로 넘어갑니다
2|todoContent name에 담겨서 서버로 넘어갑니다
3|todoContent name에 담겨서 서버로 넘어갑니다
4|
5|todoContent name에 담겨서 서버로 넘어갑니다
6|todoContent name에 담겨서 서버로 넘어갑니다
7|todoContent name에 담겨서 서버로 넘어갑니다
8|올림픽이 끝났습니다
sqlite> █
```

메모 삭제하기

To-do List with Django

메모할 내용을 적어주세요

메모하기!

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

클릭시 메모 삭제

template 확인

▶ index.html

```
<div class="ToDoDiv">
  <ul class="list-group">
    {% for todo in todos %}
      <form action="/deleteTodo/" method="GET">
        <div class="input-group" name='todo1'>
          <li class="list-group-item">{{ todo.content }}</li>
          <input type="hidden" id="todoNum" name="todoNum" value="{{ todo.id }}"></input>
          <span class="input-group-addon">
            <button type="submit" class="custom-btn btn btn-danger">완료</button>
          </span>
        </div>
      </form>
    {% endfor %}
  </ul>
</div>
```

urls.py

```
settings.py x  ToDoListWurls.py x  my_to_do_appWurls.py x  index2.html x  index.html x  views.py x
1      # my_to_do_app > urls.py
2      from django.urls import path
3      from . import views
4
5      urlpatterns = [
6          path('', views.index, name='index'),
7          path('createTodo/', views.createTodo, name='createTodo'),
8          path('deleteTodo/', views.deleteTodo, name='doneTodo')
9      ]
10
```

views.py

▶ doneTodo 함수 생성

```
22 def deleteTodo(request):
23     done_todo_id = request.GET['todoNum']
24     print("완료한 todo의 id", done_todo_id)
25     todo = Todo.objects.get(id=done_todo_id)
26     # todo.isDone = True
27     todo.delete()
28     return HttpResponseRedirect(reverse('index'))
29
```

확인

To-do List with Django

메모할 내용을 적어주세요

메모하기!

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

클릭 시 메모 삭제

완료 버튼 삭제=>숨기기

▶ 완료 버튼을 삭제 대신 숨기기 버튼으로 변경

모델 변경

```
settings.py x ToDoListWurls.py x my_to_do_appWurls.py x index2.html x index.html x views.py x apps.py x models.py x
1  from django.db import models
2
3  # Create your models here.
4  class Todo(models.Model):
5      content = models.CharField(max_length = 255)
6      isDone = models.BooleanField(default=False) #완료버튼에 사용
7
```

변경 모델 DB에 반영

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py makemigrations
```

```
Migrations for 'my_to_do_app':
```

```
my_to_do_app\migrations\0002_todo_isdone.py
```

```
- Add field isDone to todo
```

```
(venv) C:\DJango\ToDoList\ToDoList>python manage.py migrate
```

```
Operations to perform:
```

```
Apply all migrations: admin, auth, contenttypes, my_to_do_app, sessions
```

```
Running migrations:
```

```
Applying my_to_do_app.0002_todo_isdone... OK
```

template 확인 =>

todo.isDone 변수 값에 따라 표현할 것 인가 결정

▶ index.html

```
54 <div class="todoDiv">
55   <ul class="list-group">
56     {% for todo in todos %}
57     {% if todo.isDone == False %}
58     <form action="/doneTodo/" method="GET">
59       <div class="input-group" name='todo1'>
60         <li class="list-group-item">{{ todo.content }}</li>
61         <input type="hidden" id="todoNum" name="todoNum" value="{{ todo.id }}"></input>
62         <span class="input-group-addon">
63           <button type="submit" class="custom-btn btn btn-danger">완료</button>
64         </span>
65       </div>
66     </form>
67     {% else %}
68     {% endif %}
69     {% endfor %}
70   </ul>
```

urls.py

```
settings.py x  ToDoListWurls.py x  my_to_do_appWurls.py x  index2.html x  index.html x  views.py x
1      # my_to_do_app > urls.py
2      from django.urls import path
3      from . import views
4
5      urlpatterns = [
6          path('', views.index, name='index'),
7          path('createTodo/', views.createTodo, name='createTodo'),
8          path('deleteTodo/', views.deleteTodo, name='doneTodo')
9      ]
10
```


views.py

▶ doneTodo 함수 생성

```
22  def deleteTodo(request):
23      done_todo_id = request.GET['todoNum']
24      print("완료한 todo의 id", done_todo_id)
25      todo = Todo.objects.get(id=done_todo_id)
26      # todo.isDone = True
27      todo.delete()
28      return HttpResponseRedirect(reverse('index'))
29
```

확인

To-do List with Django

메모할 내용을 적어주세요

메모하기!

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

todoContent name에 담겨서 서버로 넘어갑니다

완료

클릭 시 메모 숨김

삭제가 아니고 숨겨진 건지 확인

```
sqlite> select * from my_to_do_app_todo;
11|처음 메모 입니다 |1
12|test 중 입니다 |1
13|올림픽이 끝났습니다 |1
14|처음 메모 입니다 |1
15||1
16|test 중 입니다 |1
17|처음 메모 입니다 |0
18|올림픽이 끝났습니다 |0
sqlite> █
```