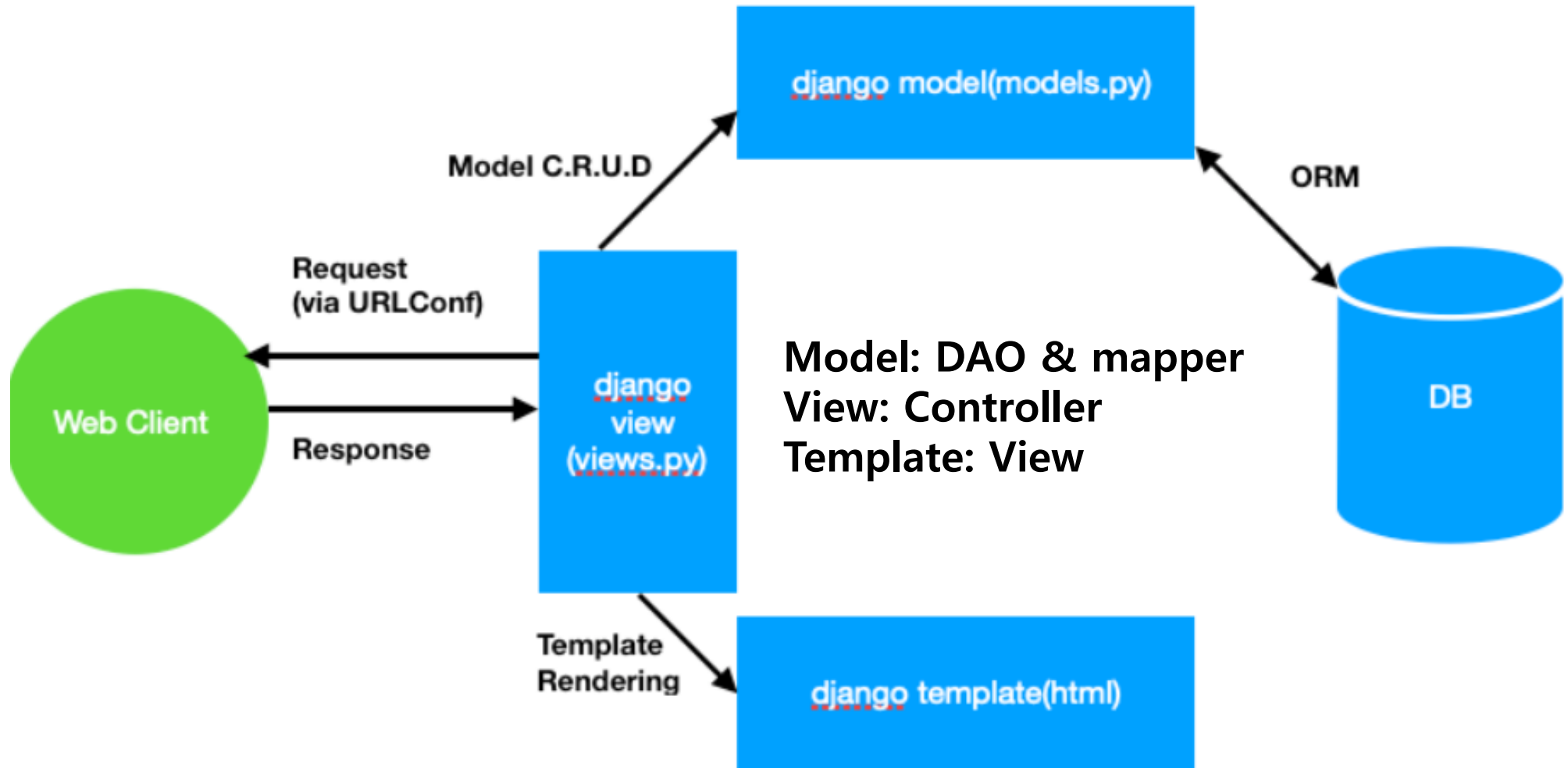
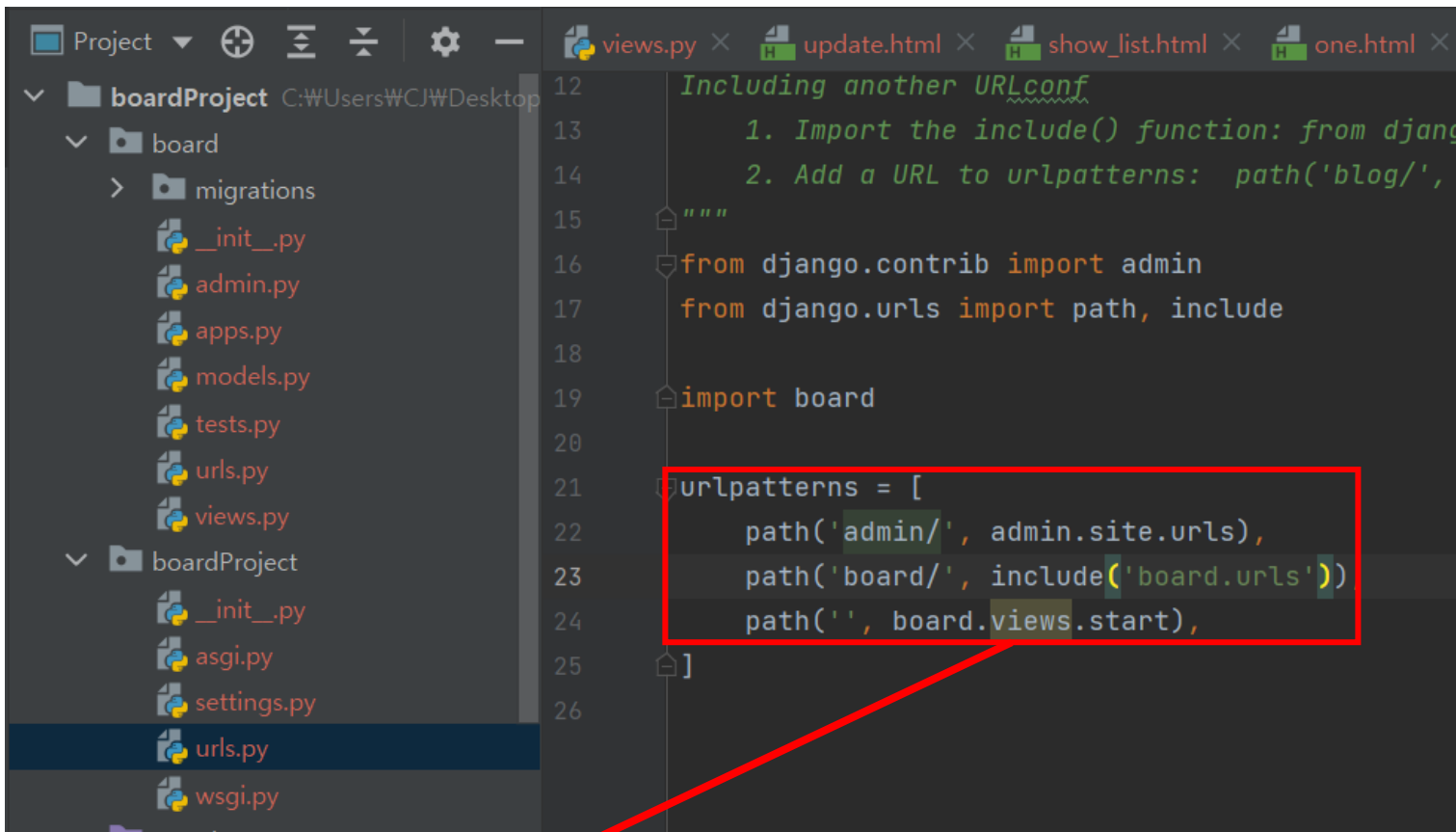


# Django\_MVT - MVT(Model-View-Template)



1. 클라이언트 요청을 받으면 URLconf 이용하여 URL 분석
2. URL 분석 결과하고 해당 URL에 대한 처리를 담당할 view 결정
3. view 로직 실행, db처리 필요 시 model 통해 처리하고 결과 받음
4. view 로직 처리 후, template 사용하여 클라이언트에 전송할 html 파일 생성
5. view 최종 결과로 html 파일을 클라이언트에게 보내 응답

# Django\_MVT - Settings

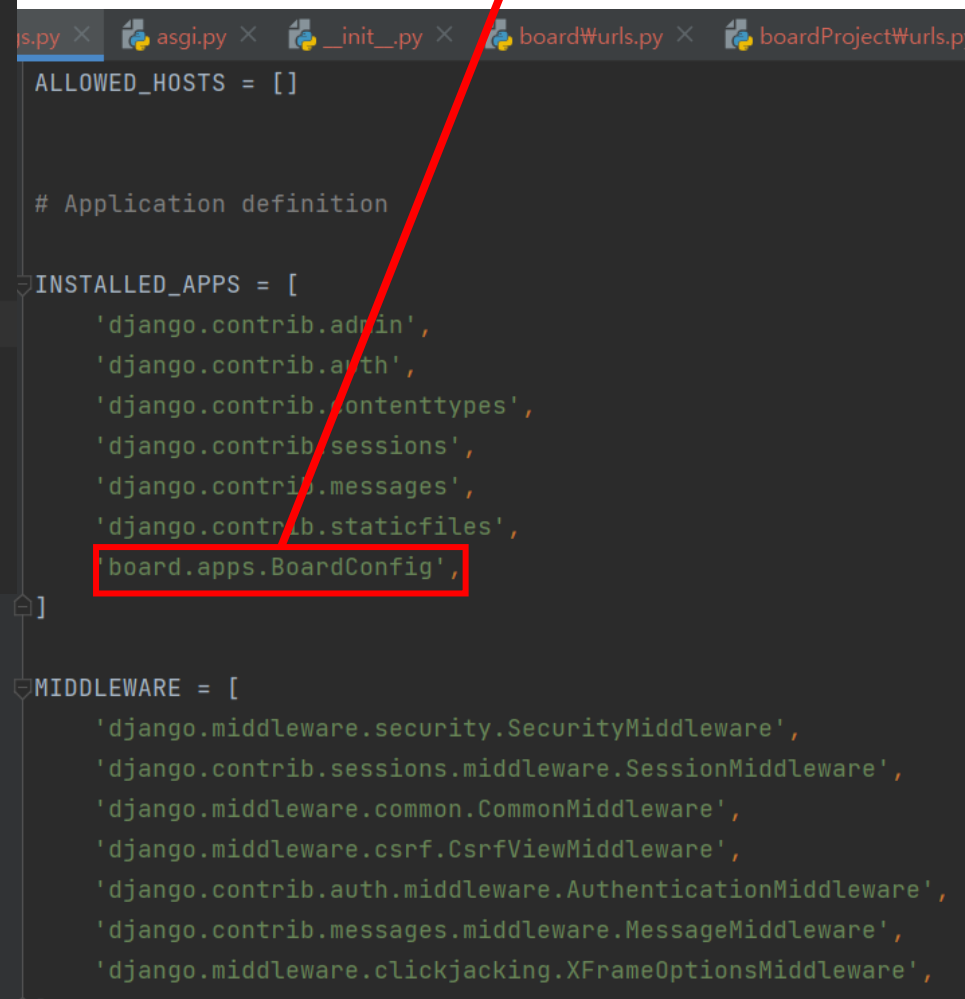


The screenshot shows the Django project file explorer on the left and the `views.py` file in the editor. The file explorer shows the `boardProject` directory with subdirectories `migrations` and `boardProject`. The `boardProject` directory contains files `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, `views.py`, `asgi.py`, `settings.py`, `urls.py`, and `wsgi.py`. The `views.py` file in the editor contains the following code:

```
12 Including another URLconf
13 1. Import the include() function: from django
14 2. Add a URL to urlpatterns: path('blog/',
15
16 from django.contrib import admin
17 from django.urls import path, include
18
19 import board
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('board/', include('board.urls')),
24     path('', board.views.start),
25 ]
26
```

A red box highlights the `urlpatterns` list, and a red arrow points from it to the text below.

Setting.py에 board config가 app으로  
등록되어 있는지 확인



The screenshot shows the `settings.py` file in the editor. The file contains the following code:

```
ALLOWED_HOSTS = []

# Application definition

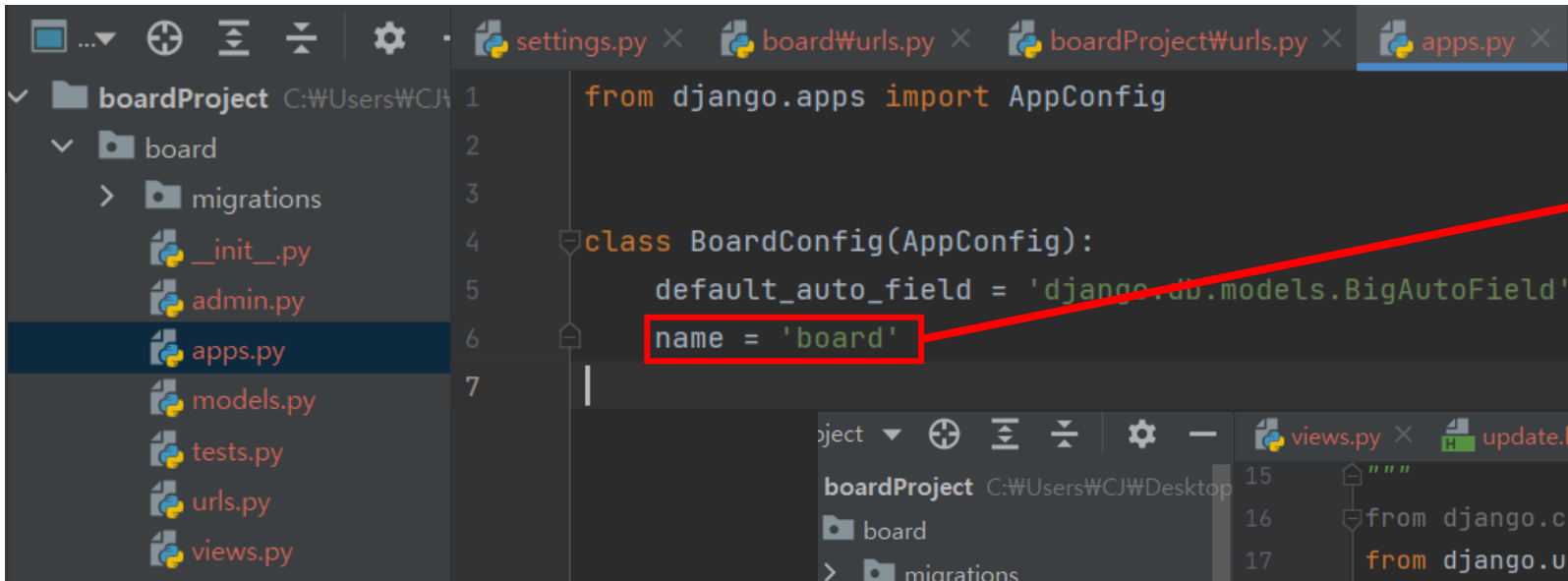
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'board.apps.BoardConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

A red box highlights the `'board.apps.BoardConfig',` line in the `INSTALLED_APPS` list, and a red arrow points from it to the text above.

urlpatterns에 admin(사용자관리계정)과  
include를 이용하여 board app경로를 지정

# Django\_MVT – apps & urls

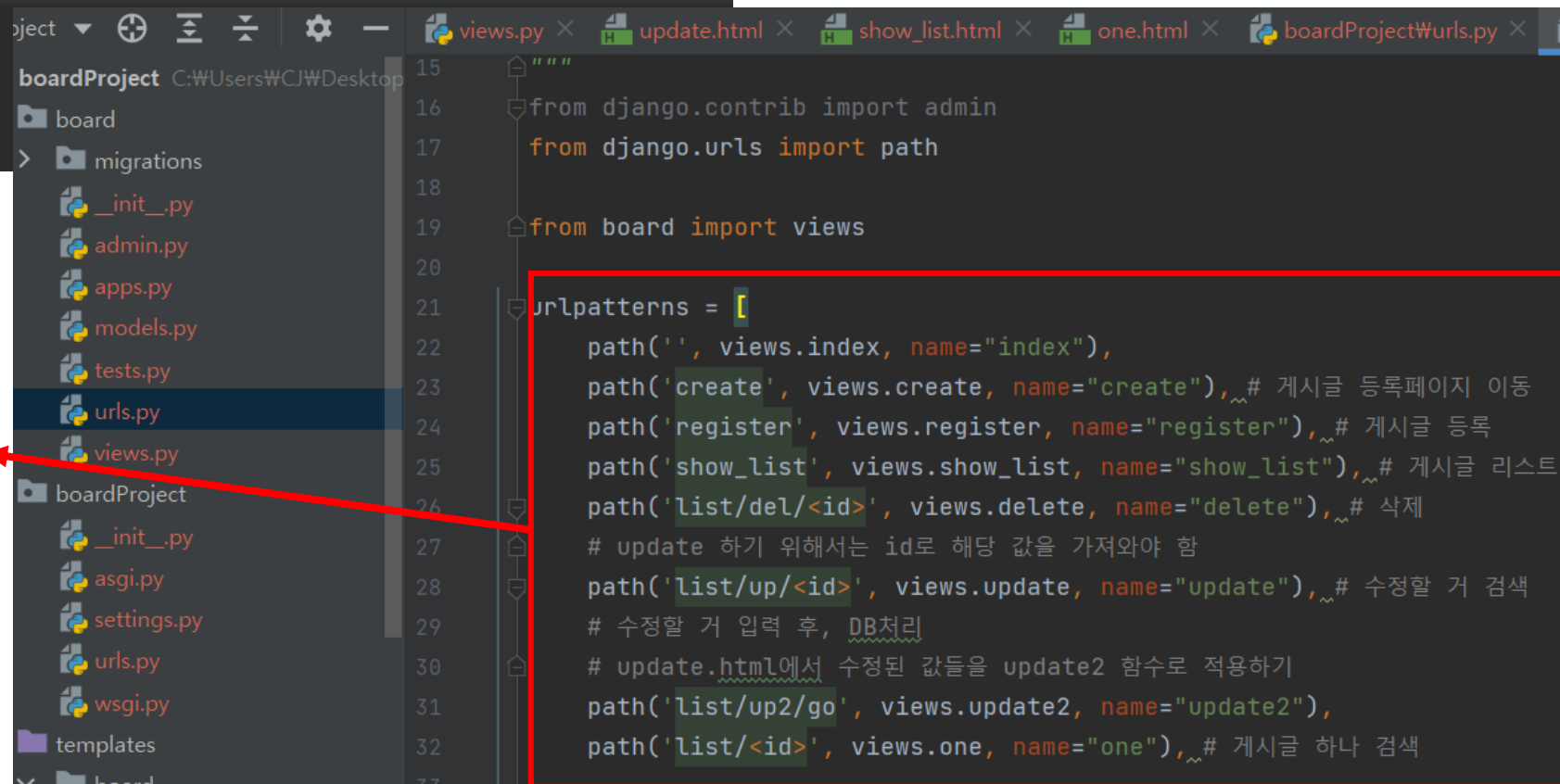


The screenshot shows the Django IDE interface. On the left, the file explorer displays the 'boardProject' directory structure, including 'board' and 'migrations' subdirectories. The 'apps.py' file is selected. The main editor shows the content of 'apps.py', which includes a red box around the line `name = 'board'`. A red arrow points from this line to the Korean text on the right.

```
1 from django.apps import AppConfig
2
3
4 class BoardConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'board'
7
```

Board Package.apps에 사용할 app name(board) 명시되어 있어야 함

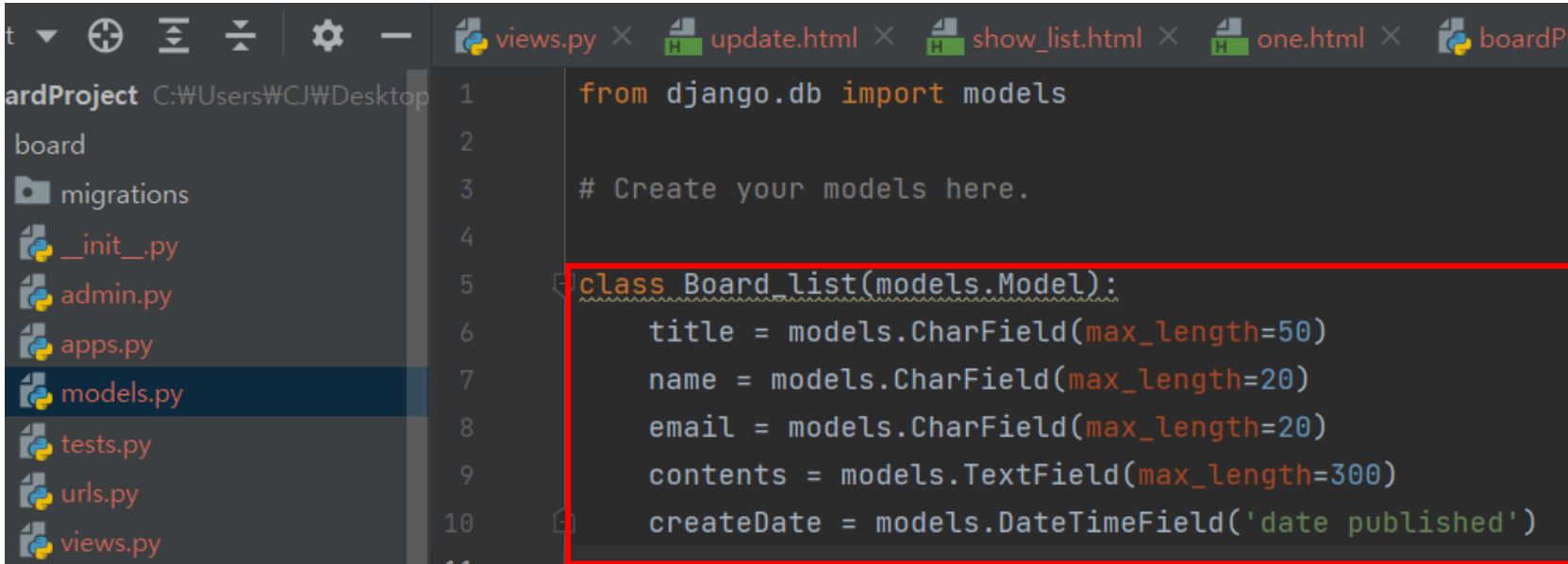
클라이언트로부터 해당 로직에 대한 (경로) 요청이 들어왔을 때, 처리할 view(Controller역할)의 함수를 지정 /를 추가해서 경로가 board 아래 있음을 명시해 줌



The screenshot shows the Django IDE interface with the 'urls.py' file selected. The main editor shows the content of 'urls.py', which includes a red box around the 'urlpatterns' list. A red arrow points from the Korean text on the left to this list.

```
15
16 from django.contrib import admin
17 from django.urls import path
18
19 from board import views
20
21 urlpatterns = [
22     path('', views.index, name="index"),
23     path('create', views.create, name="create"), # 게시글 등록페이지 이동
24     path('register', views.register, name="register"), # 게시글 등록
25     path('show_list', views.show_list, name="show_list"), # 게시글 리스트
26     path('list/del/<id>', views.delete, name="delete"), # 삭제
27     # update 하기 위해서는 id로 해당 값을 가져와야 함
28     path('list/up/<id>', views.update, name="update"), # 수정할 거 검색
29     # 수정할 거 입력 후, DB처리
30     # update.html에서 수정된 값들을 update2 함수로 적용하기
31     path('list/up2/go', views.update2, name="update2"),
32     path('list/<id>', views.one, name="one"), # 게시글 하나 검색
33
```

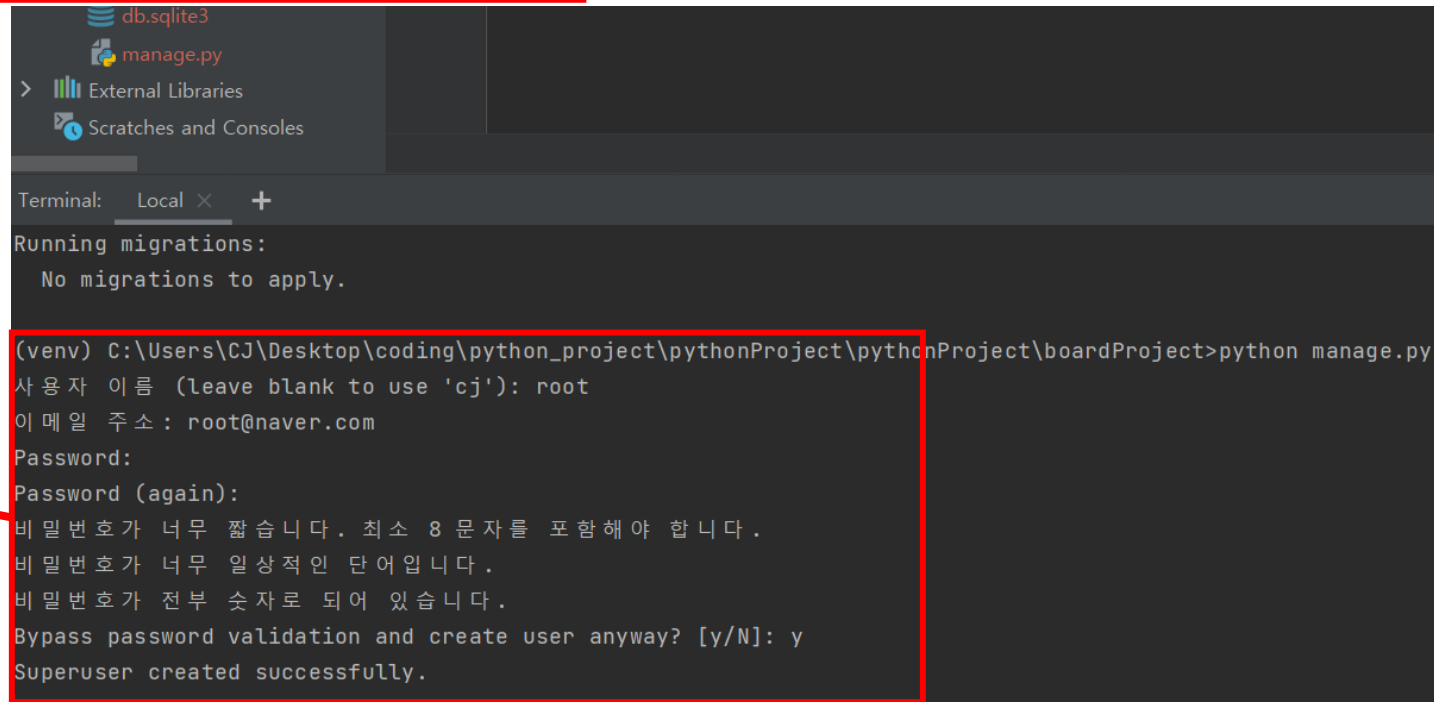
# Django\_MVT – DB & admin



```
1 from django.db import models
2
3 # Create your models here.
4
5 class Board_list(models.Model):
6     title = models.CharField(max_length=50)
7     name = models.CharField(max_length=20)
8     email = models.CharField(max_length=20)
9     contents = models.TextField(max_length=300)
10    createDate = models.DateTimeField('date published')
```

Board\_list DB를 생성하고  
컬럼을 정의해주고 admin.py  
로 이동하여 관리자 DB 접근할  
수 있도록 admin.site.register(DB  
명) 작성

Terminal → python manage.py migrate 하여  
Database 생성 후 Python manage.py  
createsuperuser으로 관리자 계정 생성



```
db.sqlite3
manage.py
> External Libraries
Scratches and Consoles

Terminal: Local X +
Running migrations:
  No migrations to apply.

(venv) C:\Users\CJ\Desktop\coding\python_project\pythonProject\pythonProject\boardProject>python manage.py
createsuperuser
사용자 이름 (leave blank to use 'cj'): root
이메일 주소: root@naver.com
Password:
Password (again):
비밀번호가 너무 짧습니다. 최소 8 문자를 포함해야 합니다.
비밀번호가 너무 일상적인 단어입니다.
비밀번호가 전부 숫자로 되어 있습니다.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

# Django\_MVT - views

```
def start(request):
    return HttpResponse(
        '<h1>Hello, this is homepage.</h1><br><hr color="red">' +
        "<h1><a href=admin>Admin Page</a></h1>"
    )

def register(request):
    # data = request.POST
    # 입력값을 딕셔너리 형태로 넣어줌
    # context = {"data": data}
    return render(request, "board/register.html")

def index(request):
    return HttpResponse(
        "<h1><a href=register>Create board</a>"
    )

def create(request):
    data = request.POST
    title2 = data.get('title')
    name2 = data.get('name')
    email2 = data.get('email')
    contents2 = data.get('contents')
    b_data = Board_list(title=title2, name=name2, email=email2, contents=contents2)
    b_data.save()
    return redirect('/board/show_list')

def delete(request, id):
    data = Board_list.objects.get(id=id)
    data.delete()
    return redirect('/board/show_list')

def update(request, id):
    # Get 방식으로 파라미터 id 값을 넘겨 받음
    # id를 검색 해서 보여주는 역할
    # member/urls.py에서 입력한 views의 함수를 찾아서 실행
    # request 객체가 member/페이지의 input에서 입력한
    data = Board_list.objects.get(id=id)
    context = {"data": data}
    # update.html로 context 값(id)을 넘긴다.
    # render(입력값이 할당된 변수, "클라이언트가 요청한 정보를 보여줄 페이지 경로")
    return render(request, 'board/update.html', context)

# Post 방식으로 update.html에서 수정된 값들을 전송
def update2(request):
    data = request.POST
    # 전송받은 data의 id를 one 변수에 저장 후
    # member의 test db에 저장
    one = Board_list.objects.get(id=data.get('id'))
    one.title = data.get('title')
    one.name = data.get('name')
    one.email = data.get('email')
    one.contents = data.get('contents')
    one.save()
    return redirect('/board/show_list')

def show_list(request):
    # name 역순으로 order_by 묶어서 보여줌
    board_list = Board_list.objects.order_by('-id')[:10]
    context = {'board_list': board_list}
    return render(request, 'board/show_list.html', context)

def one(request, id):
    # request 객체가 member/페이지의 input에서 입력한 정보를 받음
    data = Board_list.objects.get(id=id)
    context = {"data": data}
    # update.html로 context 값(id)을 넘긴다.
    # render(입력값이 할당된 변수, "클라이언트가 요청한 정보를 보여줄 페이지 경로")
    return render(request, 'board/one.html', context)
```

views.py 사용되는 모든 함수를 정의해야 함 (MVC의 Controller 역할로, model or template으로 보내줌)

클라이언트가 요청하는 로직에 대한 모든 요청에 대한 기능을 보내주는 역할을 담당함

# Django\_MVT – Create

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hi2</title>
</head>
<body bgcolor="#ffc0cb">
    <h3>Create Board</h3>
    <hr>
    <form action="create" method="post">
        {# form를 보낼때 서버로 부터 승인받은 키를 받아 데이터를 전달할 수 있음 #}
        {% csrf_token %}
        <h4>제목 <input type="text" name="title" value="test"></h4>
        <h4>이름 <input type="text" name="name" value="test"></h4>
        <h4>이메일 <input type="text" name="email" value="test"></h4>
        <h4>게시글</h4><textarea cols=100 rows=10
                                name="contents">test</textarea>
        <input type="submit" value="게시글등록">
    </form>
</body>
</html>
```

## Create Board

제목

이름

이메일

게시글

오늘은 수요일입니다.  
시간이 후딱 가네요

게시글등록

<QuerySet [<Board\_list: Board\_list object (5)>, <Board\_list: Board\_list object (3)>, <Board\_list: Board\_list object

- 5 "게시글 제목은" - 오늘은 수요일, "게시자 이름은" - Wednesday, "게시자 이메일주소는" - wed@naver.com
- 3 "게시글 제목은" - tomato, "게시자 이름은" - tom, "게시자 이메일주소는" - tomato
- 2 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test
- 1 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test

id	title	name	email	contents	createDate
1	test	test	test	test	2021-08-07 15:00:00
2	test	test	test	test	2021-08-07 15:00:00
3	tomato	tom	tomato	tomato tomato	2021-06-08 15:00:00
5	오늘은 수요일	Wednesday	wed@naver.com	오늘은 수요일입니다	2021-06-08 15:00:00

Create Board page로 이동하게 되면 CRUD-Create에 해당하는 register.html 페이지가 출력되고  
입력한 값은 views.py – create 함수로 이동하여 db에 저장되게 된다.



# Django\_MVT – Read & Delete

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>search page</title>
</head>
<body>
  {{ board_list }}
  <hr color="red">
  {% if data %}
    "번호" - {{ data.id }}<br>
    "제목" - {{ data.title }}<br>
    "이름" - {{ data.name }}<br>
    "이메일주소" - {{ data.email }}<br>
    "게시글내용" -{{ data.contents }}<br>
  {% else %}
    <p>nothing!!!</p>
  {% endif %}
  <a href="/board/list/del/{{ data.id }}">DELETE</a>
  <a href="/board/list/up/{{ data.id }}">UPDATE</a>
```

"번호" - 5

"제목" - 오늘은 수요일

"이름" - Wednesday

"이메일주소" - wed@naver.com

"게시글내용" -오늘은 수요일입니다. 시간이 후딱 가네요

[DELETE](#) [UPDATE](#)

<QuerySet [<Board\_list: Board\_list object (3)>, <Board\_list: Board\_list object (2)>, <Boar

- 3 "게시글 제목은" - tomato, "게시자 이름은" - tom, "게시자 이메일주소는" - tomato
- 2 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test
- 1 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test

1	test	test	test	test	2021-08-07 15:00:00
2	test	test	test	test	2021-08-07 15:00:00
3	tomato	tom	tomato	tomato tomato	2021-06-08 15:00:00

게시글 번호 click 하게 되면, parameter로 id를 받으며 Read 기능을 담당하는 one 함수가 실행된다.

DELETE 클릭 시, Delete 기능 실행되며 id를 parameter 변수로 받아 게시물이 삭제된다.

# Django\_MVT - Settings

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>hi2</title>
</head>
<body bgcolor="#ff8c00">
  <h3>This is update page.!!</h3>
  <hr>
  <form action="/board/list/up2/go" method="post">
    {# form를 보낼때 서버로 부터 승인받은 키를 받아 데이터를 전달할 수 있음 #}
    {% csrf_token %}
    <h4>번호 <input type="text" name="id" value="{{ data.id }}"><br></h4>
    <h4>제목 <input type="text" name="title" value="{{ data.title }}"></h4>
    <h4>이름 <input type="text" name="name" value="{{ data.name }}"></h4>
    <h4>이메일 <input type="text" name="email" value="{{ data.email }}"></h4>
    <h4>게시글</h4><textarea cols=100 rows=10 name="contents"
      value="{{ data.contents }}"></textarea>
    <input type="submit" value="게시글수정">
  </form>
</body>
</html>
```

This is update page.!!

번호

제목

이름

이메일

게시글

<QuerySet [  
<Board\_list: Board\_list object (3)>, <Board\_list: Board\_list object (2)>, <Board\_list: Board\_list object (1)>]>

- 3 "게시글 제목은" - potato is good, "게시자 이름은" - potato, "게시자 이메일주소는" - po@tato.com
- 2 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test
- 1 "게시글 제목은" - test, "게시자 이름은" - test, "게시자 이메일주소는" - test

id	title	name	email	contents	createDate
1	test	test	test	test	2021-08-07 15:00:00
2	test	test	test	test	2021-08-07 15:00:00
3	potato is good	potato	po@tato.com	감자 너무 맛있어요	2021-06-08 15:00:00

UPDATE 클릭 시, Update 기능에 실행을 위해 id 검색을 담당하는 update 함수가 실행되며

Update.html template으로 이동되며 게시글 수정하게 되면 update2 실행하며 수정된 게시글이 DB에 등록됨