# Djagno 기반의 웹프로그래밍

이 소 영

yisy0703@naver.com

Web Frameworks for Python(https://wiki.python.org/moin/WebFrameworks)
장고 공식 사이트(https://www.djangoproject.com/)
장고 공식 소스 저장소(http://github.com/django/django)
장고 참조 문서 (https://docs.djangoproject.com/ko/5.2/)

# Agenda

Django(장고) 기반의 파이썬 웹 프로그래밍

# Ch 02 Django App

1. Django Project
2. Model
3. View

Chapter 02. Django App

- 기본 converter
  (https://github.com/django/django/blob/5.2/django/urls/converters.py)

- HttpRequest 소스
  (https://docs.djangoproject.com/en/5.0/_modules/django/http/request/#HttpRequest )

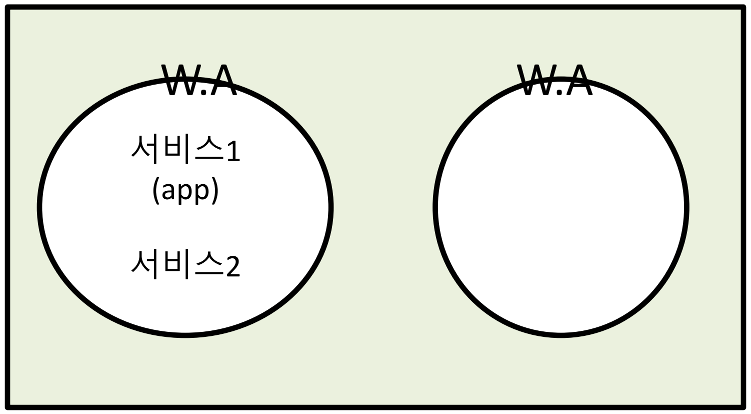
- HttpRequest 객체
  (https://docs.djangoproject.com/ko/5.2/ref/request-response  )

- HttpResponse 소스
  (https://github.com/django/django/blob/5.2/django/http/response.py )

- HttpResponse 객체
  (https://docs.djangoproject.com/ko/5.2/ref/request-response )

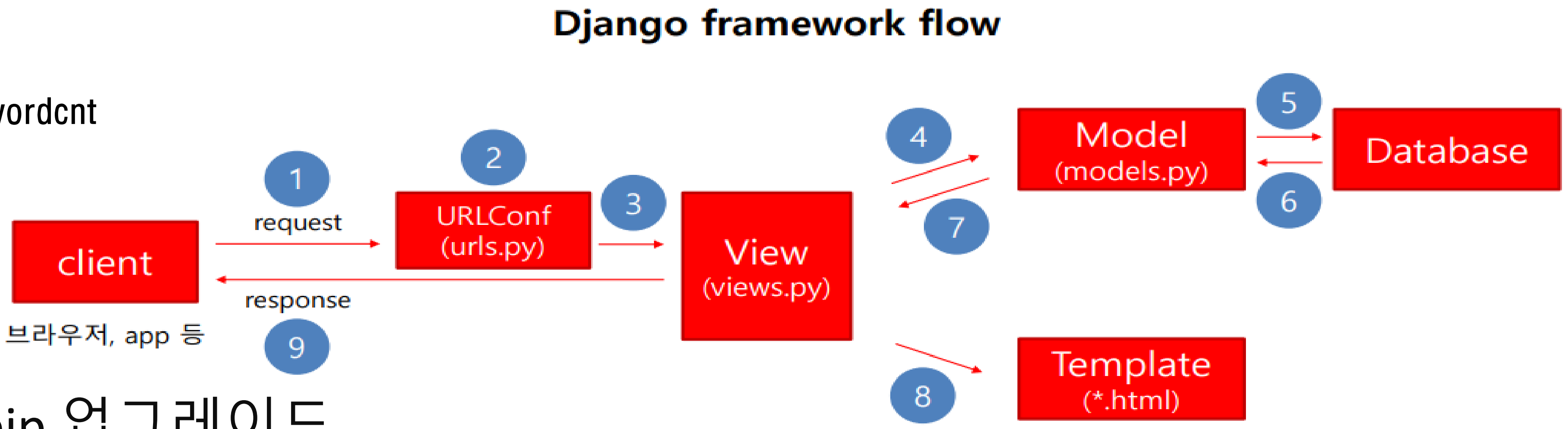
# 1. Django Project

Django Project = Web Application = web site



## 실습

ch02_wordcnt



1. pip 업그레이드
   - pip --version
   - python –m pip install --upgrade pip
   - pip --version
2. Django install
   - pip install django
3. 프로젝트 폴더 생성(venv 가상환경 생성) 후
   - django-admin startproject ch02  .
4. SECRET_KEY 숨기기(.gitignore생성)
   - .env생성**(.gitignore에 .env추가)**
   - pip install python-decouple
   - settings.py 를 수정하기

     from decouple import config

     SECRET_KEY = config('SECRET_KEY')

## 실습

ch02_wordcnt

5. home, wordcnt app 추가하기
   - python manage.py startapp home
   - python manage.py startapp wordcnt
   - Settings.py에 home과 wordcnt app 등록
6. 요청 url

```
admin/
''                : name=index
test/             : name=test
showId/숫자/   : name=showIntId
showId/문자/   : name=showStrId
wordcnt/         : name=wordcnt:wordinput
wordcnt/about/: name=wordcnt:abount
wordcnt/result/: name=wordcnt:result
```

7. URLconf
   - ch02/urls.py

```
urlpatterns = [
    path("admin/", admin.site.urls),
    path('', views.index, name='index'),
    path('test/', views.test, name='test'),
    path('showId/<int:id>/', views.showIntId, name='showIntId'),
    path('showId/<str:id>/', views.showStrId, name='showStrId'),
    path('wordcnt/', include('wordcnt.urls')),
]
```

## 실습

ch02_wordcnt

- wordcnt/urls.py

```
# wordcnt 패키지 안의 urls.py :
#  /wordcnt/ : text 입력
#  /wordcnt/result : 입력된 text wordcount
#  /wordcnt/about  : 도움말 페이지
from django.urls import path
import wordcnt.views
app_name = 'wordcnt'
urlpatterns = [
  path('', wordcnt.views.wordinput, name='wordinput'),
  path('about/', wordcnt.views.about, name='about'),
  path('result/', wordcnt.views.result, name='result'),
]
```

# 실습

ch02_wordcnt

## 7. home/views

```python
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def index(request):
  context = {'msg':'wordCount welcome page'}
  return render(request,
              'home/index.html',
              context=context)
def test(request):
  return HttpResponse('<h1>테스트  페이지</h1>'+
                '<button onclick="location=\'/\'">뒤로</button>')
def intId(request, id):
  msg = '숫자 ID는 ' + str(id)
  type = '숫자'
  return render(request,
              template_name='home/showId.html',
              context={'msg':msg, 'type':type})

def strId(request, id):
  msg = '문자 ID는 ' + str(id)
  type = '문자'
  return render(request,
              template_name='home/showId.html',
              context={'msg':msg, 'type':type})
```

# 실습

ch02_wordcnt

## 7. wordcnt/views

```
from django.shortcuts import render

# text 입력
def wordinput(request):
 return render(request, 'wordcnt/wordinput.html')
def about(request):
 return render(request, 'wordcnt/about.html')

def result(request):
 #print(request.POST)
 # full = request.POST['fulltext']
 #full = request.POST.get('fulltext','')
 full = request.GET['fulltext']
 strlength = len(full) # 글자수
  words = full.split()
 wordcnt = len(words) # 단어  갯수
  words_dic = dict() # 빈  딕셔너리
  for word in words:
  if word in words_dic.keys():
   words_dic[word] += 1 # words_dic['hong'] = 2
  else:
   words_dic[word] = 1
 print('★', full, wordcnt)
 context = {
  'full':full,
  'strlength':strlength,
  'wordcnt':wordcnt,
  'dict':words_dic.items() # [('hong',2),('good',1),('luck',1)]
 }
 return render(request,
        'wordcnt/result.html',
        context=context)
```

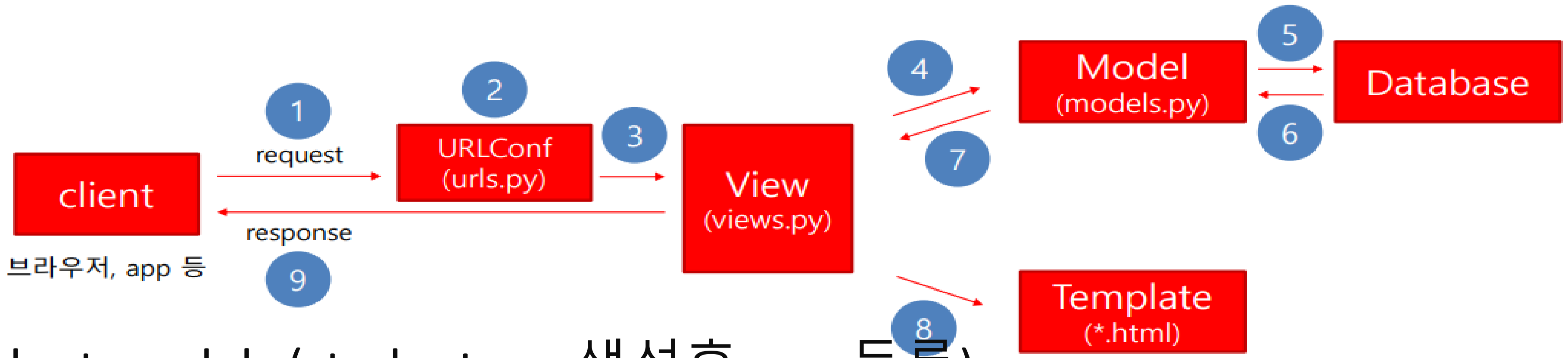
# 실습

Ch02_wordcnt

## 8. templates/home

```
<head>
  <meta charset="UTF-8">
  <!-- 파비콘 설정 : 아이콘 설정 (http://www.flaticon.com) -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  {% load static %}
  <link href="{% static 'img/brand.png' %}" rel="icon">
  <link href="{% static 'css/ex.css' %}" rel="stylesheet">
  <title>Title</title>
</head>
<body>
<div class="p-5 mb-4 bg-body-tertiary rounded-3">
  <h2>템플릿을 index.html로 사용해 보았어요</h2>
  <h1 onclick="location='{% url 'wordcnt:wordinput' %}'">{{msg}}</h1>
  <img src="{% static 'img/django.png' %}" alt="django 그림"><br>
  <a href="test">TEST</a><br>
  <a href="{% url 'test' %}">TEST</a><br>
  <button class="btn btn-outline-danger" onclick="location='{% url 'test'%}'">TEST</button>
  <button class="btn btn-outline-danger" onclick="location='/test'">TEST</button>
  <hr>
  <a href="showId/123">showIntId/123</a><br>
  <a href="{% url 'showIntId' 124 %}">showIntId/124</a><br>
  <button class="btn btn-outline-warning" onclick="location='showId/123'">숫자ID</button>
  <button class="btn btn-outline-warning" onclick="location='{% url 'showIntId' 124 %}'">숫자ID</button>
  <hr>
  <a href="showId/aab">showIntId/123</a><br>
  <a href="{% url 'showStrId' 'abc' %}">showIntId/124</a><br>
  <button class="btn btn-outline-info" onclick="location='showId/abc'">문자ID</button>
  <button class="btn btn-outline-info" onclick="location='{% url 'showStrId' 'abc' %}'">문자ID</button>
</div>
</body>
```

## Django framework flow



- Student.models (student app생성후 app 등록)

```
from django.db import models

class Student(models.Model): # student_student(테이블명 : 앱명_클래스명소문자)
  id = models.AutoField(primary_key=True)
  name = models.CharField(max_length=100, unique=True)
  major = models.CharField(max_length=100, null=True, blank=True) # 기본이
null=False
  age = models.IntegerField(default=0)
  grade = models.IntegerField(default=1)
  def __str__(self):
    return "{}:{}({}, {}세 {}학년)".format(self.id,
                                        self.name,
                                        self.major,
                                        self.age,
                                        self.grade)
```

## 실습

Ch03_orm

- python manage.py makemigrations : 변경사항이 있는지 migrations 모듈 생성
- python manage.py migrate : 사용자 및 그룹 테이블 생성
- Python manage.py sqlmigrate student 0001 : 테이블 생성 sql 보여줌
- python manage.py createsuperuser : 관리자 계정 생성(auth_user)
- python manage.py runserver 관리자 계정 로그인 확인
- python manage.py shell : 장고 shell 모두 실행

**CRATE**

```
from student.models import Student
st = Student(name='hong', major='computer', age=22, grade=2)
st.save()
st = Student(name='kim', major='bigdata', age=21, grade=2)
st.save()
Student.objects.create(name='lee', major='ai', age=23, grade=3)
qs = Student.objects.all() # 전체 데이터 읽기READ
qs[0].name, qs[1]
for s in qs:
    print(s)
```

# 실습

Ch03_orm

qs = Student.objects.get(name='kim') # 조건에 맞는 한행 읽기 **READ**
Print(qs)
qs =Student.objects.filter(age__lt=30) # 필터로 읽기
__lt : 보다 작다
__lte : 보다 작거나 같다
__gt : ~보다 크다
__gte : ~보다 크거나 같다
__isnull : null인 자료
__contains : 특정 문자열 포함 major__contains = 'i'
qs = Student.objects.order_by('age') # age 필터 기준으로 오름차순 정렬 가져옴
qs = Student.objects.order_by('-age') # age 필터 기준으로 내림차순 정렬 가져옴
qs = Student.objects.get(name='kim')
qs.age = 40 # 데이터 수정 **UPDATE**
qs.save()
qs = Student.objects.get(name='hong')
qs.delete() # 데이터 삭제 **DELETE**
Student.objects.filter(age__lt=30).delete()

# Django Project 생성

- 장고 프로젝트 생성

  django-admin startproject myproject .

```
myproject : manage.py
   └ myproject : __init__.py : 패키지로 만들어짐
                 settings.py : 장고 프로젝트 설정
                 urls.py     : 들어온 요청과 view 연결
                 wsgi.py     : 실제 서버 배포시 사용
```

- Django 규칙에 따라 디렉터리, 파일 자동 생성

# Django Project 생성

- Model을 DB에 반영(디폴트가 SQLite가 기본 설정)

  D:\src\ django\myproject> python manage.py migrate : models의 내용을 DB에 반영(migrations폴더참조)

  D:\src\ django\myproject> python manage.py createsuperuser : admin로그인 id 추가

- 개발 서버 구동(실습용 개발 서버)

  D:\src\django\myproject> python manage.py runserver

  http://127.0.0.1:8000으로 Webserver 연결

# Django Project 구조

1. Django Project

- 장고 프로젝트는 여러 개의 App을 가짐
- App 생성 : python manage.py <app_name>

```
D:\src\django\myproject> python manage.py –help : 도움말
D:\src\django\myproject> python manage.py startapp blog : 앱생성. 서비스 구현은 앱 안에.
```

- App의 구조

```
myproject : manage.py
   └ myproject : __init__.py, settings.py, urls.py, wsgi.py
   └ blog : migrations
            __init__.py
            admin.py
            apps.py
            models.py
            tests.py
            views.py
```

# Blog 앱 작성

1. Django Project

- myproject/settings.py

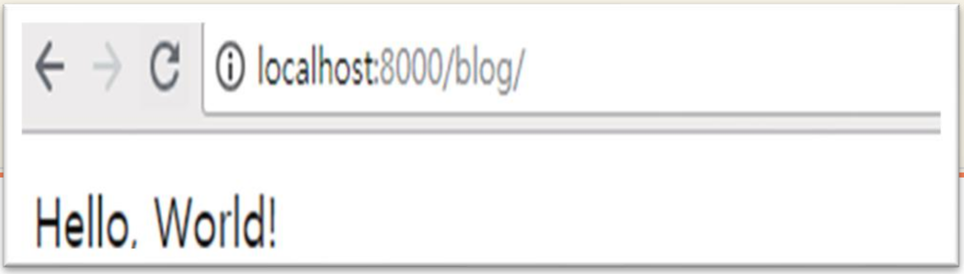
```
INSTALLED_APPS = [
  ~생략~ # 장고에서 미리 만들어 놓은 기능(앱) 있음
'blog',     # 앱 등록
]
```

- blog/views.py

```
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, World!")
```

# Blog 앱 작성

1. Django Project


localhost:8000/blog/

Hello, World!

- myproject/urls.py

```python
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
  path('blog/', include('blog.urls'))
]
```

- blog/urls.py

```python
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index)
]
```

# 새로운 앱 작성

0. 프로젝트 생성
1. 앱 생성
2. 프로젝트/settings.py의 INSTALLED_APPS에 앱 등록
3. View 작성
4. 앱이름/urls.py 파일 생성
5. 프로젝트/urls.py에 include 적용

# Django의 요청처리

1. Django Project

1. root URLConf : settings.py의 ROOT_URLCONF
```
ROOT_URLCONF = 'myproject.urls'
```
2. ROOT_URLCONF 모듈 로드 후 urlpatterns 변수 검색
3. ROOT_URLCONF의 include를 통해 TREE 구조로 확장
4. Tree 구조로 확장된 urlpatterns의 path()또는 re_path()들을 검색 리스트에 포함
5. 작성된 리스트에서 URL 패턴을 순차 검색
6. 요청된 URL과 일치하는 패턴을 찾으면 검색 중단
7. 일치된 패턴의 뷰 함수 호출
8. 뷰 함수에 다음의 인자를 전달
   - HttpRequest인스턴스
   - 이름이 지정되지 않은 인자는 위치 기반으로 전달
   - 키워드 인자는 kwargs 값에 설정되어 전달

## url pattern

1. Django Project

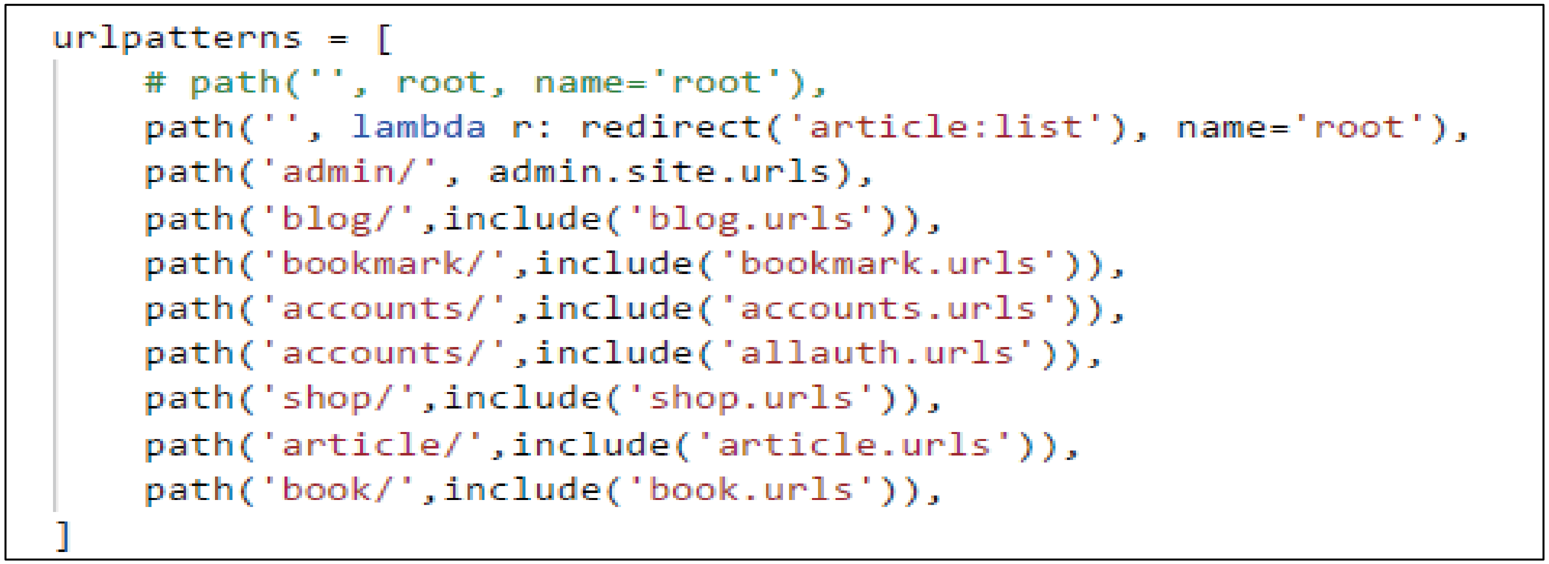
- URL 패턴의 끝은 “/”로 끝남

- 첫번째 /는 내부적으로 추가되기 때문에 지정하지 않음
  - “/articles/” 대신 “articles/”로 지정

- View의 인자로 사용되는 값은 꺽쇄괄호 <변수이름>를 사용

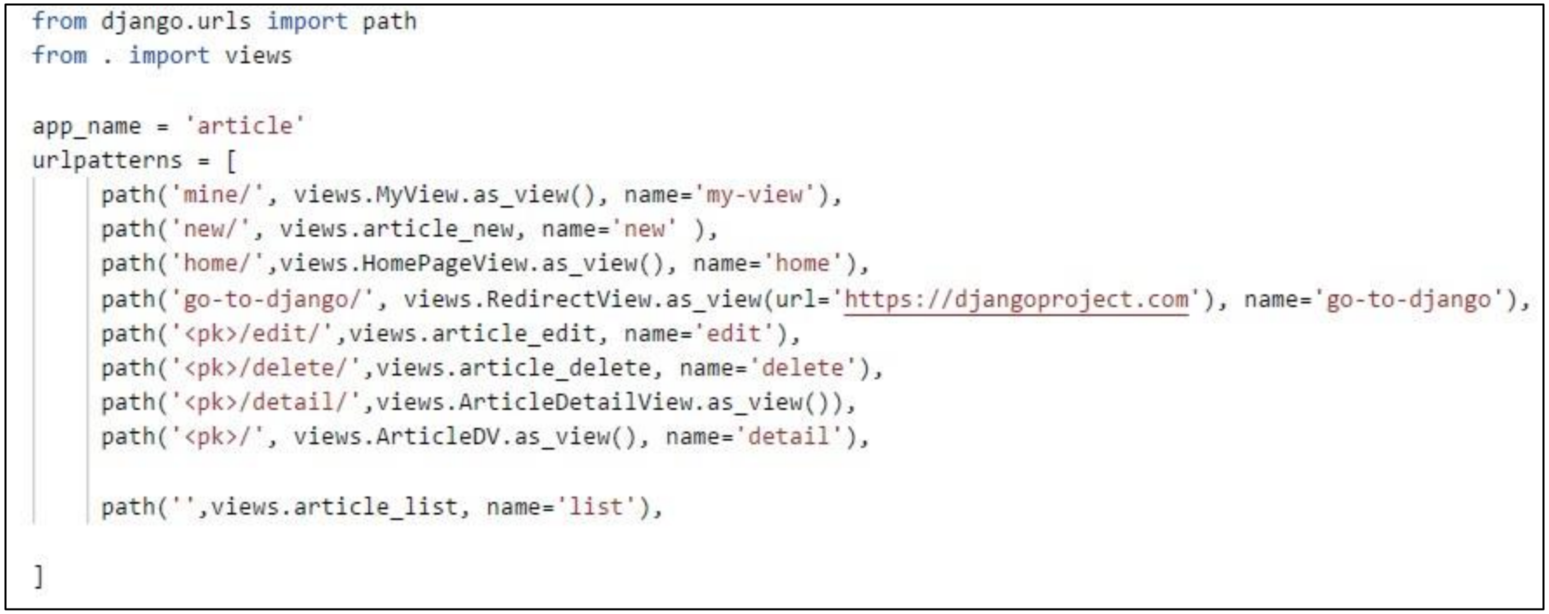
- View의 인자로 사용되는 값의 타입 지정 시 <데이터 타입:변수이름>으로 지정

- 프로젝트/urls.py



```
urlpatterns = [
    # path('', root, name='root'),
    path('', lambda r: redirect('article:list'), name='root'),
    path('admin/', admin.site.urls),
    path('blog/',include('blog.urls')),
    path('bookmark/',include('bookmark.urls')),
    path('accounts/',include('accounts.urls')),
    path('accounts/',include('allauth.urls')),
    path('shop/',include('shop.urls')),
    path('article/',include('article.urls')),
    path('book/',include('book.urls')),
]
```

- 앱/urls.py



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

app_name = 'article'
urlpatterns = [
    path('mine/', views.MyView.as_view(), name='my-view'),
    path('new/', views.article_new, name='new' ),
    path('home/',views.HomePageView.as_view(), name='home'),
    path('go-to-django/', views.RedirectView.as_view(url='https://djangoproject.com'), name='go-to-django'),
    path('<pk>/edit/',views.article_edit, name='edit'),
    path('<pk>/delete/',views.article_delete, name='delete'),
    path('<pk>/detail/',views.ArticleDetailView.as_view()),
    path('<pk>/', views.ArticleDV.as_view(), name='detail'),

    path('',views.article_list, name='list'),

]
```

## 2. Model

# 데이터 베이스 설정

2. Model

- myproject/settings.py

```python
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
  }
}

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

USE_TZ = False
```

## Django Model

2. Model

- Django 내장 ORM

- SQL문 없이 장고 모델을 통해 DB CRUD 작업 가능

- 파이썬 클래스와 DB 테이블
  - ◆ Model = DB Table
  - ◆ Model instance = Table의 1개 row

# Model 정의

2. Model

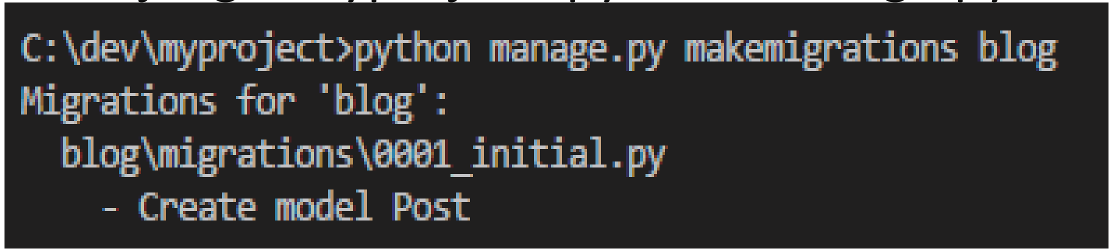
• blog/models.py

```
from django.db import models

class Post(models.Model):
  title = models.CharField(max_length=100)
  content = models.TextField()
  create_at = models.DateField(auto_now_add=True)
  updated_at = models.DateTimeField(auto_now=True)
  def __str__(self):
      return self.title
```

# Model의 활성화

2. Model

- D:\src\django\myproject>python manage.py makemigrations blog

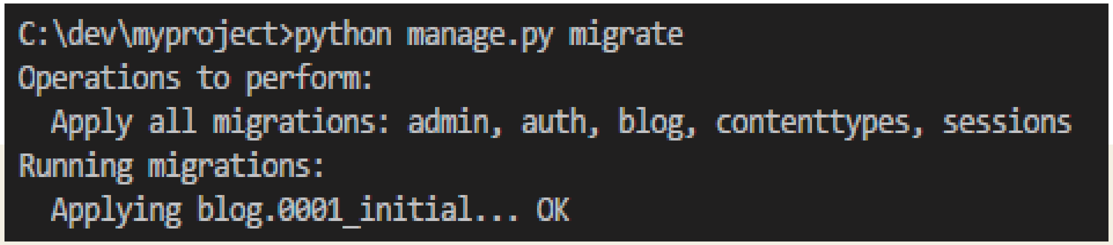


```
C:\dev\myproject>python manage.py makemigrations blog
Migrations for 'blog':
  blog\migrations\0001_initial.py
    - Create model Post
```

- D:\src\django\myproject>python manage.py sqlmigrate blog 0001

```
BEGIN;
--
-- Create model Post
--
CREATE TABLE "blog_post" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
             "title" varchar(100) NOT NULL, "content" text NOT NULL,  "create
             _at" date NOT NULL, "updated_at" datetime NOT NULL);
COMMIT;
```

- D:\src\django\myproject>python manage.py migrate



```
C:\dev\myproject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying blog.0001_initial... OK
```

# DB API

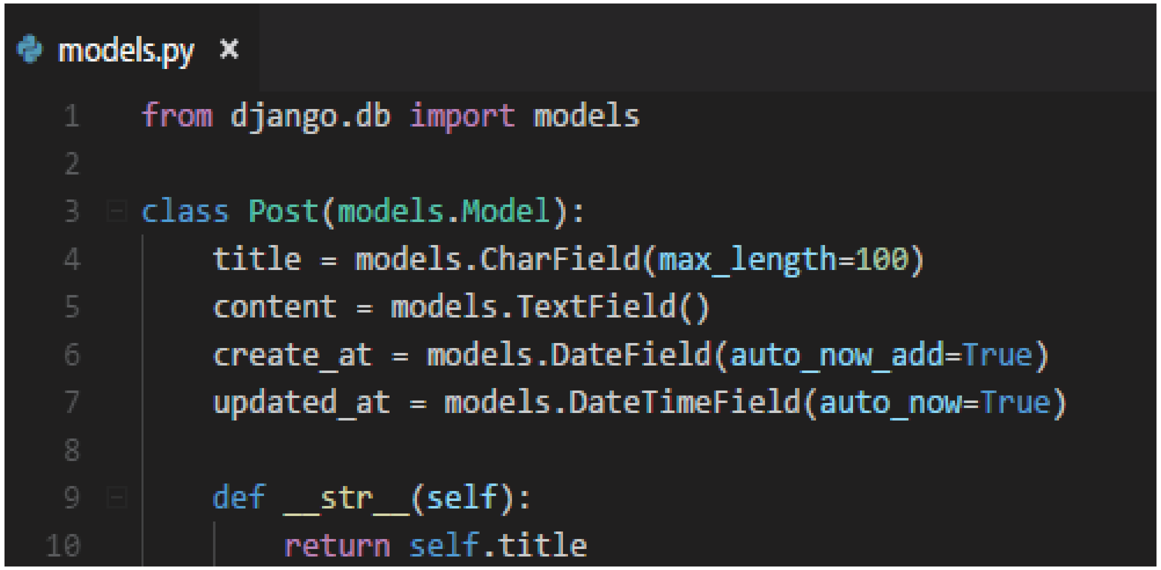
2. Model

```
C:\dev\myproject> python manage.py shell (기본쉘은 장고 프로젝트 인식. 장고쉘은 환경값
인식하는 쉘)

ln[1] : from blog.models import Post
ln[2] : Post.objects.all()  #<QuerySet []>
ln[3] : p1 = Post(title='아이스하키', content='빙판에서 5명이 퍽을 가지고 하는 경기')
ln[4] : p1.save() # 실제 DB에 save(insert)
ln[5] : p1.id   #1
ln[6] : p1.title
ln[7] : p1.content
ln[8] : p1.create_at
ln[9] :  p2 = Post(); p2.title='농구‘
Ln[10] :  p2.content='지상에서 5명이 농구공을 가지고 하는 경기'
ln[10] : p2.save()
ln[11] : p2.id  # 2

ln[12] : Post.objects.all() #<QuerySet [<Post: Post object (1)>, <Post: Post object
(2)>]>
```

# 객체 표현

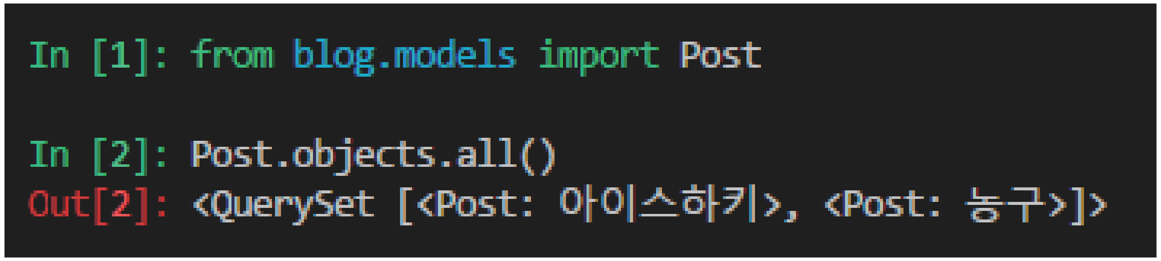
2. Model

- __str__ 메소드 오버라이딩



```python
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    create_at = models.DateField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title
```

- __str__ 메소드에서 반환 값 출력



```
In [1]: from blog.models import Post

In [2]: Post.objects.all()
Out[2]: <QuerySet [<Post: 아이스하키>, <Post: 농구>]>
```

# Django Admin App

- Super User 계정 생성

```
D:\src\django\myproject> python manage.py createsuperuser
```

- 개발 서버 구동

```
D:\src\django\myproject> python manage.py runserver
```

- Admin 페이지에서 blog 모델

```python
# blog/admin.py

from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

- Admin 페이지 접속
  http://localhost:8000/admin

## 3. View

## View

3. View

- HttpRequest
  - ◆ View의 첫번째 인자로 전달, client의 요청 정보를 가짐
- HttpResponse
  - ◆ View의 반환 객체, client로 전달되는 응답 정보를 가짐
  - ◆ 문서 : https://docs.djangoproject.com/en/3.2/ref/request-response/#httpresponse-objects
  - ◆ 소스 : https://github.com/django/django/blob/3.2/django/http/response.py
- View 종류
  - ◆ FBV(Function Based View) ; 함수 기반 뷰
    - 호출 가능한 객체
  - ◆ CBV(Class Based View) ; 클래스 기반 뷰
    - 클래스이름.as_view()를 통해 호출 가능한 객체를 생성/반환

# View의 인자

3. View

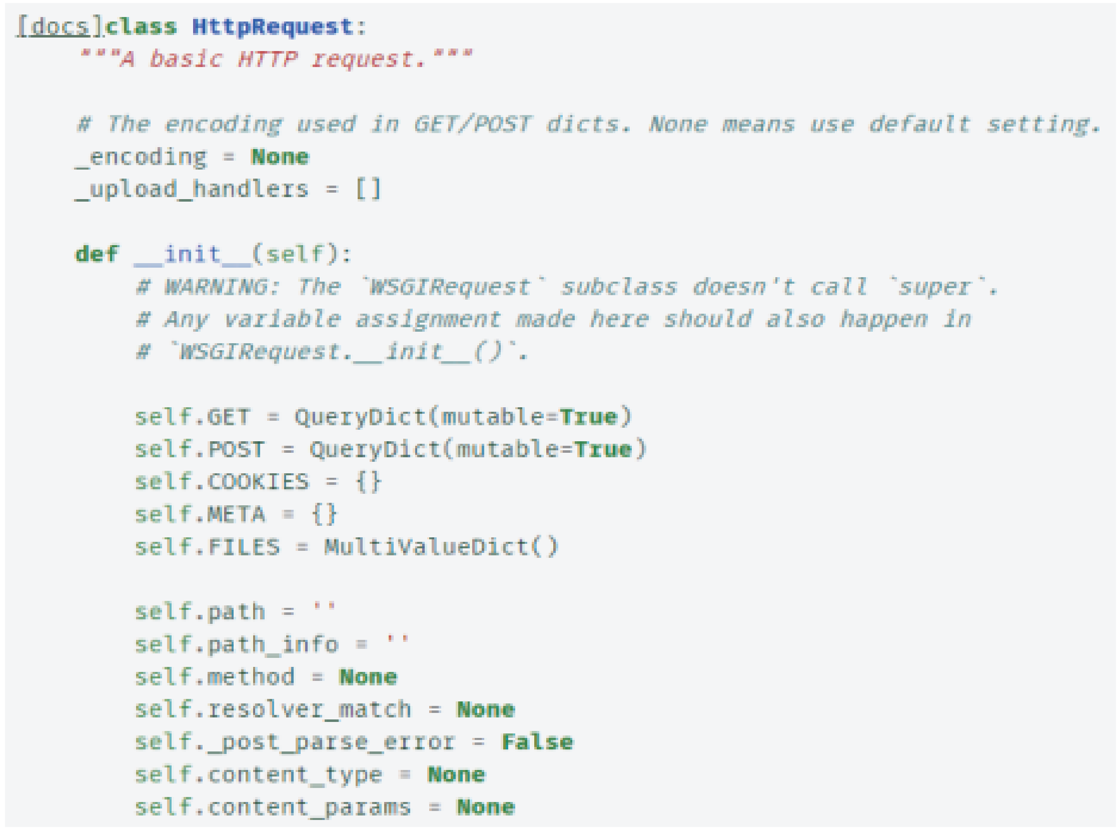
- 1번째 인자
  - ◆ HttpRequest ; client 의 요청 정보를 가짐
  - ◆ 문서 : https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects
  - ◆ 소스 : https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects

- 2번째이후
  - ◆ 요청 URL로부터 capture된 문자열들. Client가 넘겨준 데이터
  - ◆ url(), re_path()를 통한 모든 인자는 str타입으로 전달
  - ◆ Path를 통한 인자는 매핑된 Converter의 to_python()에서 반환된 타입으로 전달

# HttpRequest

3. View

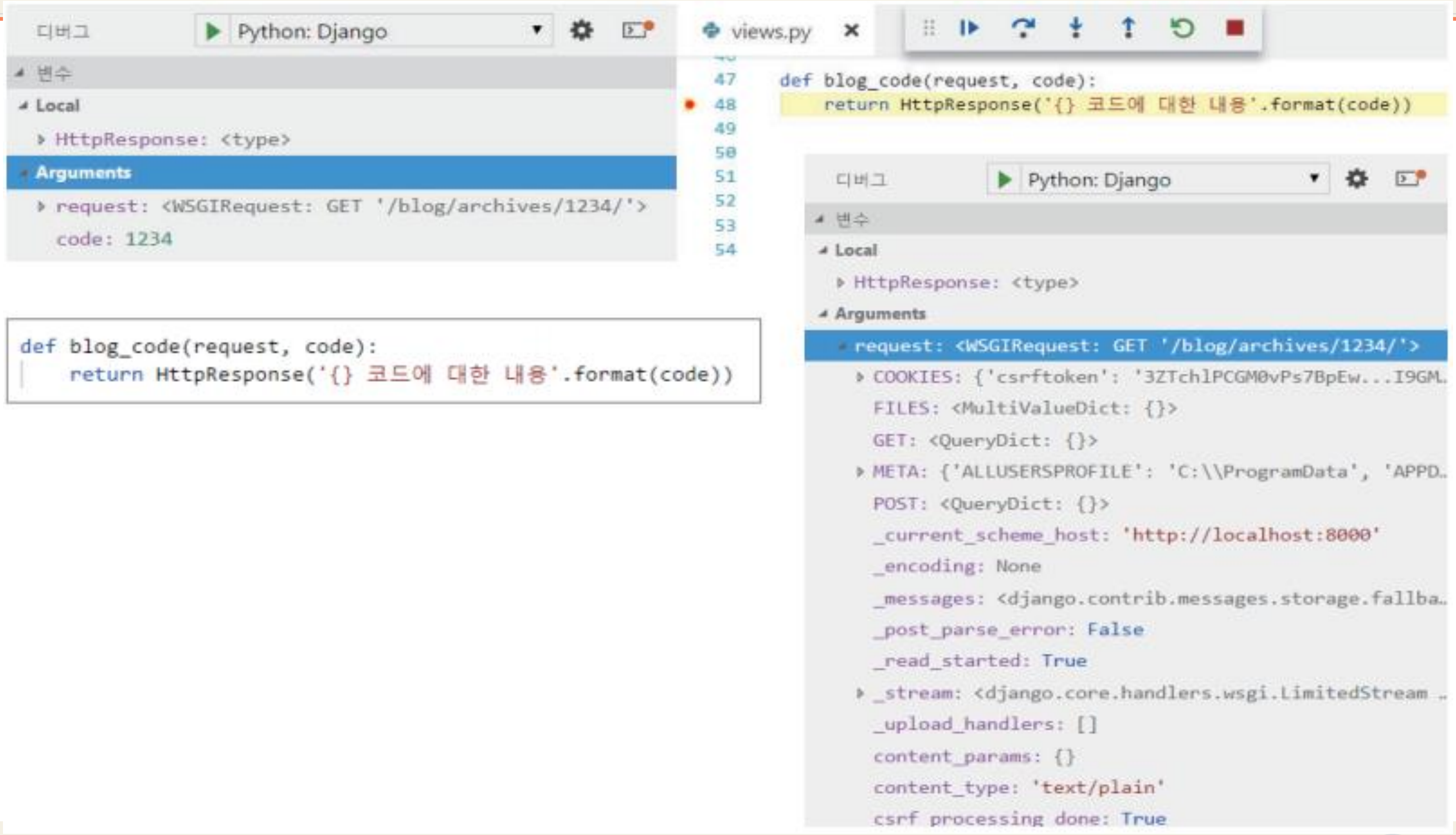
- 문서
  - https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects
- 속성
  - HttpRequest.body
  - HttpRequest.path
  - HttpRequest.path_info
  - HttpRequest.method
  - HttpRequest.encoding
  - HttpRequest.content_type
  - HttpRequest.content_params
  - HttpRequest.GET
  - HttpRequest.POST
  - HttpRequest.COOKIES
  - HttpRequest.FILES
  - HttpRequest.META



```
[docs]class HttpRequest:
    """A basic HTTP request."""

    # The encoding used in GET/POST dicts. None means use default setting.
    _encoding = None
    _upload_handlers = []

    def __init__(self):
        # WARNING: The `WSGIRequest` subclass doesn't call `super`.
        # Any variable assignment made here should also happen in
        # `WSGIRequest.__init__()`.

        self.GET = QueryDict(mutable=True)
        self.POST = QueryDict(mutable=True)
        self.COOKIES = {}
        self.META = {}
        self.FILES = MultiValueDict()

        self.path = ''
        self.path_info = ''
        self.method = None
        self.resolver_match = None
        self._post_parse_error = False
        self.content_type = None
        self.content_params = None
```

# HttpRequest

3. View



```
디버그    Python: Django
변수
Local
  HttpResponse: <type>
Arguments
  request: <WSGIRequest: GET '/blog/archives/1234/'>
  code: 1234
```

```python
def blog_code(request, code):
    return HttpResponse('{} 코드에 대한 내용'.format(code))
```

```python
def blog_code(request, code):
    return HttpResponse('{} 코드에 대한 내용'.format(code))
```

```
디버그    Python: Django
변수
Local
  HttpResponse: <type>
Arguments
  request: <WSGIRequest: GET '/blog/archives/1234/'>
    COOKIES: {'csrftoken': '3ZTchlPCGM0vPs7BpEw...I9GM…
    FILES: <MultiValueDict: {}>
    GET: <QueryDict: {}>
    META: {'ALLUSERSPROFILE': 'C:\\ProgramData', 'APPD…
    POST: <QueryDict: {}>
    _current_scheme_host: 'http://localhost:8000'
    _encoding: None
    _messages: <django.contrib.messages.storage.fallba…
    _post_parse_error: False
    _read_started: True
    _stream: <django.core.handlers.wsgi.LimitedStream …
    _upload_handlers: []
    content_params: {}
    content_type: 'text/plain'
    csrf_processing_done: True
```

# View의 반환값

- View는 반드시 HttpResponse 객체를 리턴해야 함
- 문서
  - https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects
  - 소스 : https://docs.djangoproject.com/en/3.2/ref/request-response/#httprequest-objects
- 속성
  - HttpResponse.content
  - HttpResponse.charset
  - HttpResponse.status_code
  - HttpResponse.reason_phrase
  - HttpResponse.streaming
  - HttpResonse.closed

## View의 반환값

3. View

```
#https://github.com/django/django/blob/3.2/django/http/response.py
class HttpResponseBase:
  status_code = 200
  def_init_(self, content_type=None, status=None, reason=None, charset=None):
    self._headers = {}
    self._closable_objects = []
    self._handler_class = None
    self.cookies = SimpleCookie()
    self.closed = False
    if status is not None:
      try:
         self.status_code = int(status)
      except (ValueError, TypeError):
        raise TypeError('HTTP status code must be an integer.')

      if not 100 <= self.status_code <= 599:
        raise ValueError('HTTP status code must be an integer from 100 to 599.')
    self._reason_phrase = reason
    self._charset = charset
    if content_type is None:
      content_type = '%s; charset=%s'% (settings.DEFAULT_CONTENT_TYPE,self.charset)
    self['Content-Type'] = content_type
```

# View 내 function의 반환값

3. View

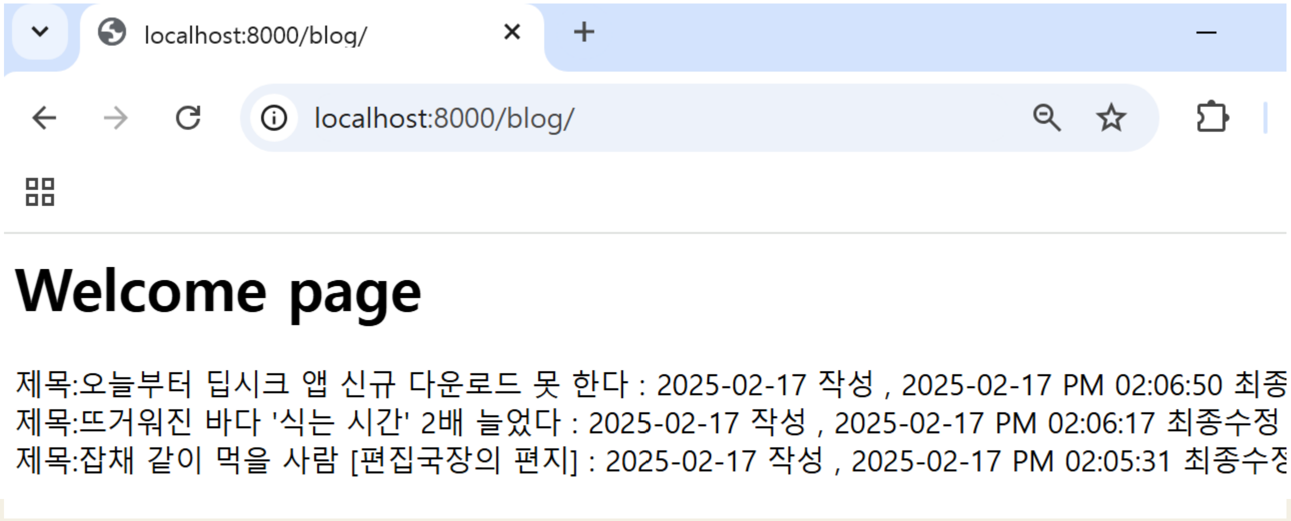
```
class HttpResponse(HttpResponseBase):
  streaming = False

  def_init_(self, content=b'', *args, **kwargs):
    super().__init__(*args, **kwargs)
    self.content = content

class StreamingHttpResponse(HttpResponseBase):
class FileResponse(StreamingHttpResponse):
class HttpResponseRedirectBase(HttpResponse):
class HttpResponseRedirect(HttpResponseRedirectBase):
class HttpResponseNotModified(HttpResponse):
class HttpResponseBadRequest(HttpResponse):
class HttpResponseNotFound(HttpResponse):
class HttpResponseForbidden(HttpResponse):
class HttpResponseNotAllowed(HttpResponse):
class HttpResponseGone(HttpResponse):
class HttpResponseServerError(HttpResponse):
 class HttpResponseServerError(HttpResponse):
 class Http404(Exception):
class JsonResponse(HttpResponse):
```

# 응답 정보 직접 처리

3. View

- blog/views.py

```
from django.http import HttpResponse
from .models import Post

def index(request):
  post_list = Post.objects.all()
  output= ' <br>'.join([p.__str__() for p in post_list])
  return HttpResponse("<h1>Welcome page</h1>" + output)
```



Welcome page

제목:오늘부터 딥시크 앱 신규 다운로드 못 한다 : 2025-02-17 작성 , 2025-02-17 PM 02:06:50 최종
제목:뜨거워진 바다 '식는 시간' 2배 늘었다 : 2025-02-17 작성 , 2025-02-17 PM 02:06:17 최종수정
제목:잡채 같이 먹을 사람 [편집국장의 편지] : 2025-02-17 작성 , 2025-02-17 PM 02:05:31 최종수정

return

# Template 사용

3. View

```
# blog/views.py
from django.shortcuts import render
def index(request):
  post_list = Post.objects.all()
  return render(request, 'blog/index.html', {'post_list':post_list})
                                         # 템플릿페이지에서 사용할 이름
```

```
# blog/templates/blog/index.html
{% if post_list %}
  <ul>
    {%for post in post_list %}
        <li><a href="/blog/{{post.id}}/">{{post.title}}</a></li>
    {% endfor %}
  </ul>
{% else %}
  <p> No posts are available.</p>
{% endif %}
```

# 템플릿페이지가 있는 곳은 앱/templates

# Rander() 함수 – 주로 많이 씀

3. View

```
# blog/converters.py
class CodeConverter:
  regex = '\d{1,4}'

  def to_python(self, value):
    return int(value)

  def to_url(self, value):
    return str(value)
```

```
# blog/urls.py
from django.urls import path, register_converter
from . import views
from .converters import CodeConverter
register_converter(CodeConverter, 'dddd')
app_name="blog"
urlpatterns = [
  path('', views.index, name='index'), # url은 위에서부터 찾음
  #path('<int:post_id>/', views.detail, name='detail'),
  path('<dddd:post_id>/', views.detail, name='detail'),]
```

## detail View 작성

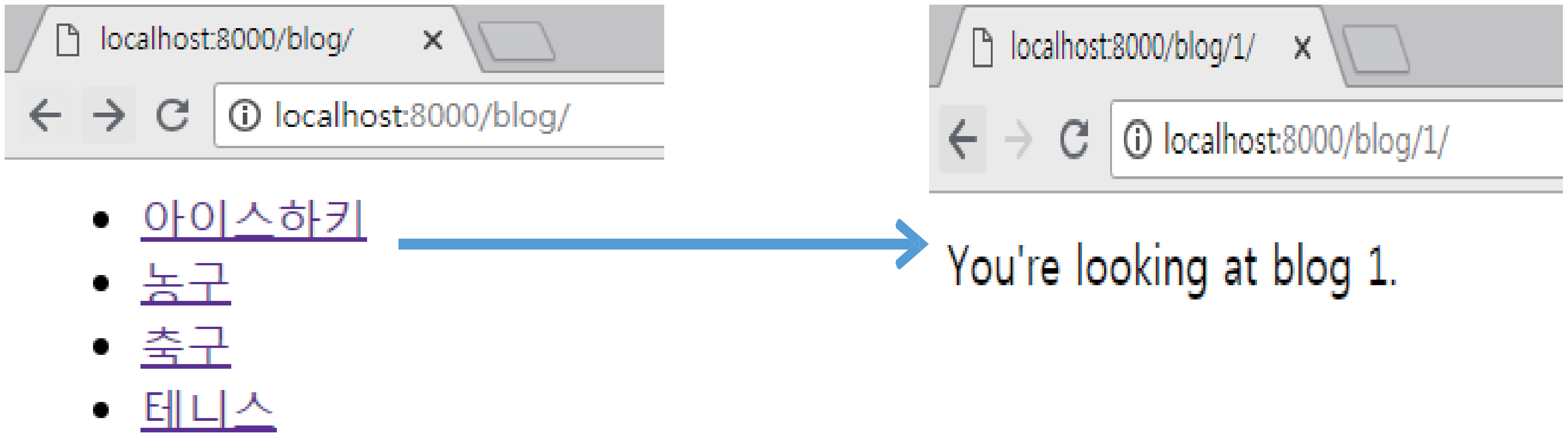
3. View

- blog/views.py

```
def detail(request, post_id):
  return HttpResponse("You're looking at blog %s." %post_id)
```

- blog/urls.py https://github.com/django/django/blob/3.2/django/urls/converters.py

```
path('<int:post_id>/', views.detail)
```



- 아이스하키
- 농구
- 축구
- 테니스

You're looking at blog 1.

# Converters

https://github.com/django/django/blob/3.2/django/urls/converters.py

- blog/converters.py

```
class Codeconverter:
  regex = "\d{1,4}"

  def to_python(self, value):
    return int(value)

  def to_url(self, value):
    return str(value)
```

- blog/templates/blog/detail.html

```
{{ post.content }}
```

## 404 오류 발생

3. View

- blog/views.py

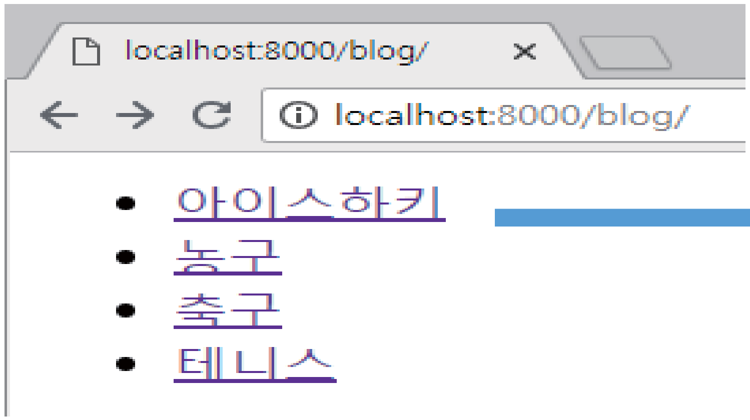
```
from django.http import Http404

def detail(request, post_id):
  try:
    post = Post.objects.get(pk=post_id)
  except Post.DoesNotExist:
    raise Http404("Post does not exist")
  return render(request, 'blog/detail.html', {'post':post})
```

- blog/templates/blog/detail.html

```
{{ post.content }}
```

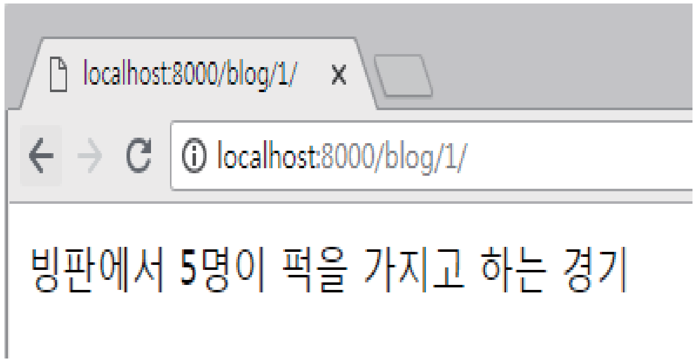
# 404오류 발생

3. View



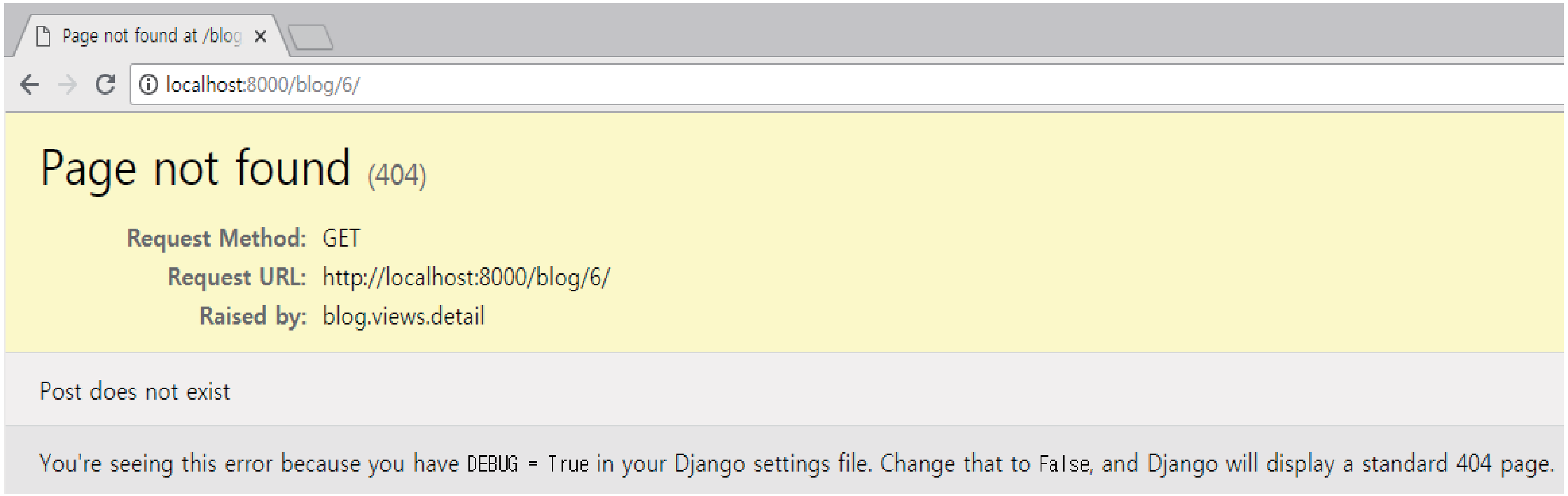
localhost:8000/blog/

- 아이스하키
- 농구
- 축구
- 테니스



localhost:8000/blog/1/

빙판에서 5명이 퍽을 가지고 하는 경기



Page not found at /blog

localhost:8000/blog/6/

Page not found (404)

```
Request Method:  GET
   Request URL:  http://localhost:8000/blog/6/
     Raised by:  blog.views.detail
```

Post does not exist

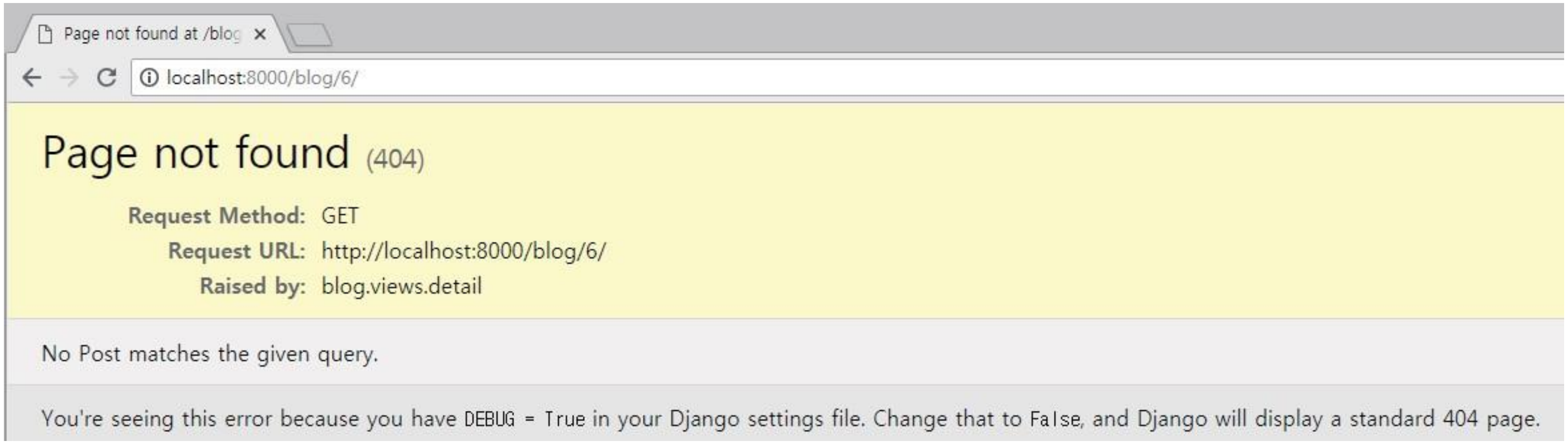
You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

# get_object_or_404() 함수

3. View

- blog/views.py

```
from django.shortcuts import render, get_object_or_404

def detail(request, post_id):
  post = get_object_or_404(Post, pk=post_id)
  return render(request, 'blog/detail.html', {'post':post})
```



Page not found (404)

```
Request Method:  GET
   Request URL:  http://localhost:8000/blog/6/
     Raised by:  blog.views.detail
```

No Post matches the given query.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

# JSON 데이터 응답

HttpResponse 객체의 HttpResponse subclasses 참조

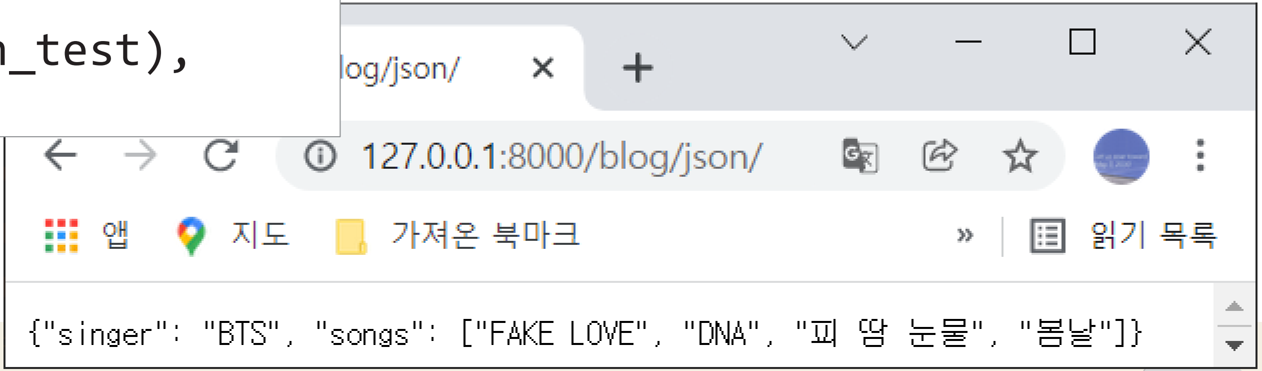
3. View

- blog/views.py

```
from django.http import JsonResponse # HttpResponse 하위클래스

def json_test(request):
  music = {'singer':'BTS','songs': ['FAKE LOVE','DNA','피 땀 눈물','봄날'
  ]}
  return JsonResponse(music, json_dumps_params={'ensure_ascii':False}) #
  한글 깨지지 않으려
```

- blog/urls.py

```
urlpatterns = [
  path('json/', views.json_test),
]
```



```
{"singer": "BTS", "songs": ["FAKE LOVE", "DNA", "피 땀 눈물", "봄날"]}
```

## JSON 데이터 응답

3. View

- blog/views.py

```
import os
from django.conf import settings

def excel_download(request):
  filepath = os.path.join(settings.BASE_DIR, 'demo.xlsx')
  filename = ' myproject.xlsx'
  with open(filepath, 'rb') as f:
    response = HttpResponse(f, content_type='application/vnd.ms-excel')
    response["Content-Disposition"] =
                "attachment; filename={}".format(filename)
  return response
```

- blog/urls.py

```
urlpatterns = [
    path('excel/', views.excel_download),
]
```

# Pandas를 통해 CSV 응답

3. View

```
import pandas as pd
from io import StringIO
from urllib.parse import quote
from django.http import HttpResponse

def pandas_csv_download(request):
   df = pd.DataFrame([
     [100, 110, 120],
     [200, 210, 220],
     [300, 310, 320],
   ])

   io = StringIO()
   df.to_csv(io, index=None)
   io.seek(0)

   filename = quote('pandas_csv.csv')
   response = HttpResponse(io, content_type="text/csv")
   response['Content-Disposition'] =
               "attachment; filename={}".format(filename)
   return response
```

# Pandas를 통해 excel 응답

3. View

```
import pandas as pd
from io import BytesIO
from urllib.parse import quote
from django.http import HttpResponse
# pip install openpyxl
def pandas_excel_download(request):
   df = pd.DataFrame([
     [100, 110, 120],
     [200, 210, 220],
     [300, 310, 320],
   ])

   io = BytesIO()
   df.to_excel(io)
   io.seek(0)

   filename = quote('pandas_excel.xlsx')
   response = HttpResponse(io, content_type="application/vnd.ms-excel")
   response['Content-Disposition'] =
            "attachment; filename={}".format(filename)
   return response
```

# Redirect

3. View

```
from django.shortcuts import redirect

def get_redirect1(request):
    return redirect('/blog/') # redirect('blog:index')
def get_redirect2(request):
    return redirect('http://google.com')
```

```
from django.urls import path, register_converter
from . import views
from .converter import CodeConverter
Register_converter(CodeConverter, ‘dddd’)
app_name="blog"
urlpatterns = [
  path('', views.index, name="index"),
  path('<dddd:post_id>/', views.detail, name='detail'),
  path('<int:post_id>/', views.detail, name='detail'), # int 컨버터를 써달라는 얘기
  path('json/', views.json_test),
  path('excel/', views.excel_download),
  path('csv/', views.pandas_csv_download),
  path('csvexcel/', views.pandas_excel_download),
  path('re1/', views.get_redirect1),
  path('re2/', views.get_redirect2),
]
```