2022

응용소프트웨어개발 7주차 장고로 정적파일 처리하기

JEONBUK NATIONAL UNIVERSITY

페이지 만들기: FBV vs CBV

FBV와 CBV

FBV(Function based view)는 말 그대로 함수에 기반을 둔 방법입니다. 함수를 직접 만들어서 원하는 기능을 직접 구현할 수 있는 장점이 있습니다. CBV(Class based view)는 장고가 제공하는 클래스를 활용해 구현하는 방법입니다. 장고는 웹 개발을 할 때 반복적으로 많이 구현하는 것들을 클래스로 미리 만들어서 제공하고 있습니다. 이 클래스들을 활용하는 방법입니다.

FBV와 CBV 중 어느 것이 낫다고 단정할 수는 없습니다. 필요에 따라 선택하면 됩니다. 필자는 CBV를 더 선호하지만 장고의 원리를 이해하기 위해 FBV를 먼저 살펴보겠습니다.

FBV를 CBV로 변환하기

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post

# def index(request):
#     posts = Post.objects.all()
#
#     return render(
#         request,
#         'blog/index.html',
```

주석 처리하거나 삭제!

FBV를 CBV로 변환하기

02단계 urls.py 수정하기

이제 blog/urls.py를 열어 URL 끝이 /blog/일 때는 PostList 클래스로 처리하도록 수정합니다. 기존의 path('', views.index)는 주석 처리하거나 삭제합니다.

실습 파일: blog/urls.py

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

urlpatterns = [
    path('', views.PostList.as_view()),
    # path('', views.index),
    path('<int:pk>/', views.single_post_page),
]
```

FBV를 CBV로 변환하기

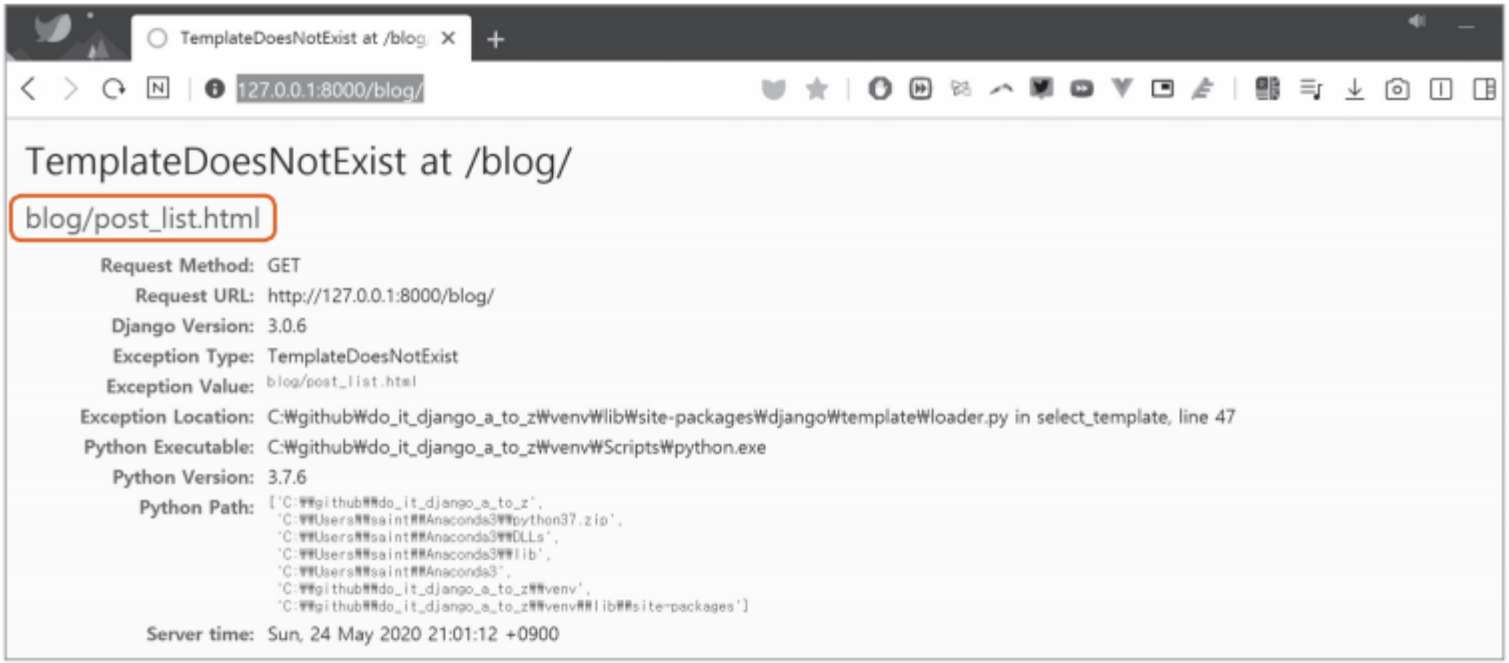

그림 8-20 ListView를 활용한 클래스에서 템플릿이 지정되지 않아 TemplateDoesNotExist 오류 발생

장고가 제공하는 ListView는 모델명 뒤에 '_list'가 붙은 html 파일을 기본 템플릿으로 사용하도록 설정되어 있습니다. 즉, Post 모델을 사용하면 post_list.html이 필요합니다. 그래서 이 오류는 다음 두 가지 방법으로 해결할 수 있습니다. 하나는 PostList 클래스에서 template_name을 직접 지정하는 방법이고 다른 하나는 post_list.html을 바로 만드는 것이죠.

CBV에서 템플릿파일 지정방식#1 - 직접지정

실습 파일: blog/views.py

```
from django.shortcuts import render
from .models import Post
from django.views.generic import ListView

class PostList(ListView):
    model = Post
    template_name = 'blog/index.html'
(...생략...)
```

앞에서 FBV를 사용할 때는 index() 함수에서 Post.objects.all() 함수로 가져온 Post 레코드를 posts 딕셔너리로 명명했습니다. 그리고 템플릿 파일에서 for 문으로 posts에 담긴 Post 레코드를 하나씩 나열했습니다. 템플릿에서 ListView로 만든 클래스의 모델 객체를 가져오려면 object_list 명령어를 사용하면 됩니다. 또는 Post 모델을 사용했으니 post_list라고 써도 자동으로 인식합니다. blog/index.html에 posts라고 써 있는 부분을 object_list 또는 post_list로 바꿔줍시다. 여기서는 post_list로 수정했습니다. 이제 웹 브라우저에서 새로고침을 하면 이전처럼 잘 나올 겁니다.

실습 파일: blog/templates/blog/index.html

```
(...생략...)
<body>
<h1>Blog</h1>

{% for p in post_list %}
    <hr/>
    <h2><a href="{{ p.get_absolute_url }}">{{ p.title }}</a></h2>
    <h4>{{ p.created }}</h4>
    <p>{{ p.content }}</p>
{% endfor %}
</body>
</html>
```

CBV에서 템플릿파일 지정방식#2 - 관례

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post
    template_name = 'blog/index.html'
(...생략...)
```

그리고 blog/templates/blog/index.html의 파일명을 post_list.html로 수정하면 됩니다. 파이참에서 index.html 파일을 마우스 오른쪽 버튼으로 클릭합니다. 그리고 [Refactor]를 선택한 후 [Rename]을 선택하면 됩니다.

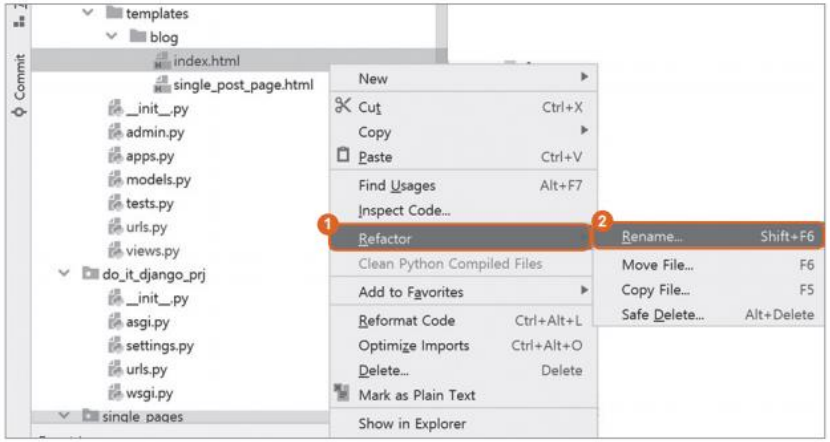

그림 8-22 index.html을 post_list.html로 수정

CBV에서 최신파일 보여주기

04단계 최신 포스트부터 보여주기

앞서 말했듯이 대부분의 블로그 웹 사이트는 최신 글을 맨 위에 배치합니다. CBV로 만드는 페이지에도 적용해 보겠습니다. 다행히 장고의 ListView는 이런 기능도 구현해 줍니다. 다음과 같이 ordering = '-pk'를 추가하세요. Post 레코드 중 pk 값이 작은 순서대로 보여달라는 뜻입니다.

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post
    ordering = '-pk'
(...생략...)
```

CBV형식으로 블로그 상세 페이지 구현하기

실습 파일: blog/views.py

```
~~from django.shortcuts import render~~
from django.views.generic import ListView, DetailView
from .models import Post

class PostList(ListView):
    model = Post
    ordering = '-pk'

class PostDetail(DetailView):
    model = Post
(...생략...)
# def single_post_page(request, pk):
#     post = Post.objects.get(pk=pk)
#
#     return render(
#         request,
#         'blog/single_post_page.html',
#         {
#             'post': post,
#         }
#     )
```

실습 파일: blog/urls.py

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

urlpatterns = [
    path('<int:pk>/', views.PostDetail.as_view()),
    path('', views.PostList.as_view()),
    # path('<int:pk>/', views.single_post_page),
    # path('', views.index),
]
```

CBV형식으로 블로그 상세 페이지 구현하기 - 템플릿 파일

그림 8-23 포스트 상세 페이지의 템플릿 파일을 지정하지 않아 TemplateDoesNotExist 오류 발생

정적 파일 처리하기

03단계 static 폴더 만들고 css 파일 넣기

각 앱 폴더 아래에 static 폴더를 만들고 css, js와 같은 정적 파일을 넣겠습니다. 먼저 blog/static/blog/bootstrap 폴더를 만드세요. 이전에 부트스트랩을 배울 때 내려받았던 bootstrap.min.css 파일과 bootstrap.min.css.map 파일을 그 안에 넣어주면 됩니다.

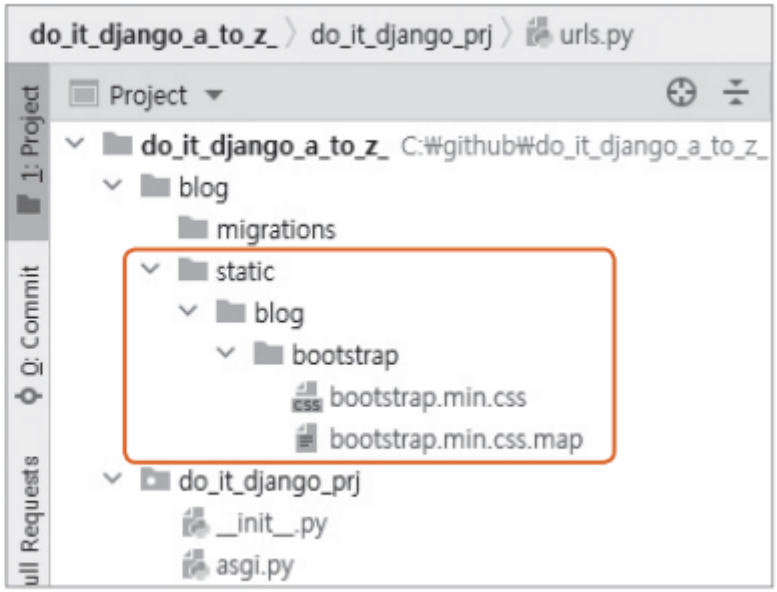

그림 9-3 bootsrtap.min.css와 bootstrap.min.css.map 추가

정적 파일 처리하기

실습 파일: blog/templates/blog/post_list.html

```
<!DOCTYPE html>
{% load static %}
<html lang="ko">
<head>
    <title>Blog</title>
    <link rel="stylesheet" href="{% static 'blog/bootstrap/bootstrap.min.css' %}"
media="screen">

    <script src="https://kit.fontawesome.com/**********.js"
crossorigin="anonymous"></script>
</head>
(...생략...)
```

블로그 목록 페이지 완성하기

실습 파일: blog/templates/blog/post_list.html

```
(...생략...)
{% for p in post_list %}
<!-- Blog Post -->
<div class="card mb-4">
    <img class="card-img-top" src="http://placehold.it/750x300" alt="Card image cap">
    <div class="card-body">
        <h2 class="card-title">{{ p.title }}</h2>
        <p class="card-text">{{ p.content }}</p>
        <a href="{{ p.get_absolute_url }}" class="btn btn-primary">Read More &rarr;</a>
    </div>
    <div class="card-footer text-muted">
        Posted on {{ p.created_at }} by
        <a href="#">작성자명 쓸 위치(개발예정)</a>
    </div>
</div>
{% endfor %}
(...생략...)
```

알찬대학 따뜻한동행

2022

응용소프트웨어개발 7주차
Streamlit

JEONBUK NATIONAL UNIVERSITY

전북대학교
JEONBUK NATIONAL UNIVERSITY

streamlit

https://docs.streamlit.io/library/get-started/create-an-app
https://zzsza.github.io/mlops/2021/02/07/python-streamlit-dashboard/

streamlit

https://taegon-streamlit-softeng-2022-streamlit-app-iojt1i.streamlitapp.com/