

## 스 크 린

.....

박규리

권가영

.....

여현정

박지예

.....

조희원

손재윤

.....

김은서

장다연

.....

김나영

이가은

.....

고희주

이규민

.....

장다연

# 정적 파일 및 미디어 파일 관리

12기 덕성여자대학교 멋쟁이사자처럼 7차 세션

# 오늘 배울 내용

- 01 정적 파일
- 02 Static 파일
- 03 Media 파일

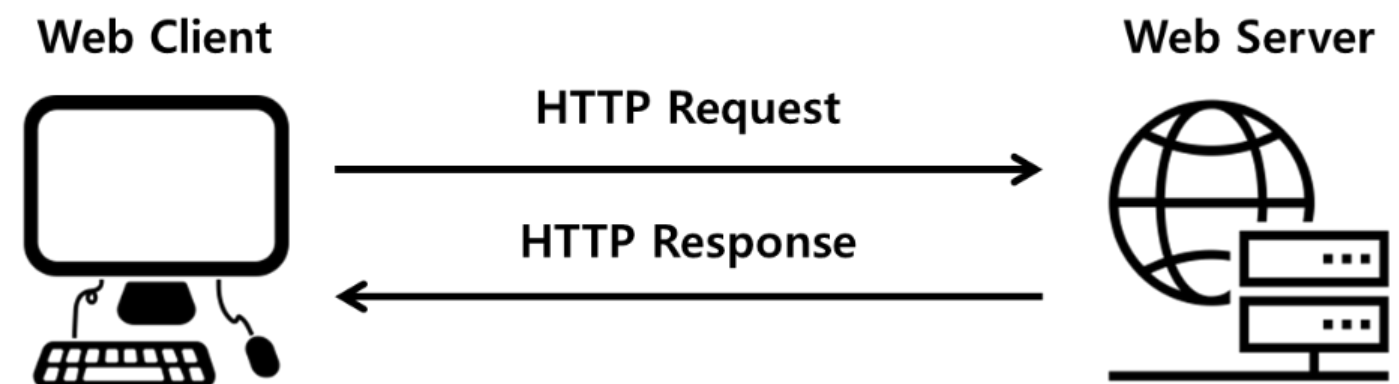
# 정적 파일이란?

HTML을 제외하고 웹 페이지를 렌더링 할 때 필요한 추가적인 파일을 의미한다.

- 서버에 저장되어 있어야하며, 서버에 저장된 그대로를 서비스한다.
- django에서 정적파일은 아래의 두가지 파일로 분류될 수 있다.

Static 파일

Media 파일



# Static 파일이란?

Image, 자바스크립트(js), CSS 파일을 Static 파일이라고 한다.

- 개발자가 미리 서버에 저장해 놓은 파일을 의미한다.
  - 파일 자체가 고정되어 있고, 서비스 중에도 추가되거나 변경되지 않는다.
- 외부 환경에 관계 없이 일정한 결과값을 제공해주는 걸 의미한다.
- Static 웹페이지
  - 사용자는 서버에 저장된 데이터가 변경되지 않는 한 고정된 웹 페이지를 보게 된다.

# MTV 구조 이해

이전 주차에서도 설명했듯이, 장고는 MTV 구조로 동작한다.

즉, 앱 폴더(우리 실습에서는 blog 폴더)에 있는 **templates** 폴더의 html 파일은 정적 파일이 아니다.

지금까지 만든 html 파일만 봐도 form을 활용하여 사용자가 채워야하는 빈칸들이 많고,  
django는 views.py에 정의한 내용에 따라 그 빈칸을 채워 사용자에게 정보를 제공한다.  
따라서, **templates** 폴더에 **css, js** 파일을 함께 넣어도 해당 파일에 접근할 수 없다.

→ css, js 파일은 Static 파일이기 때문

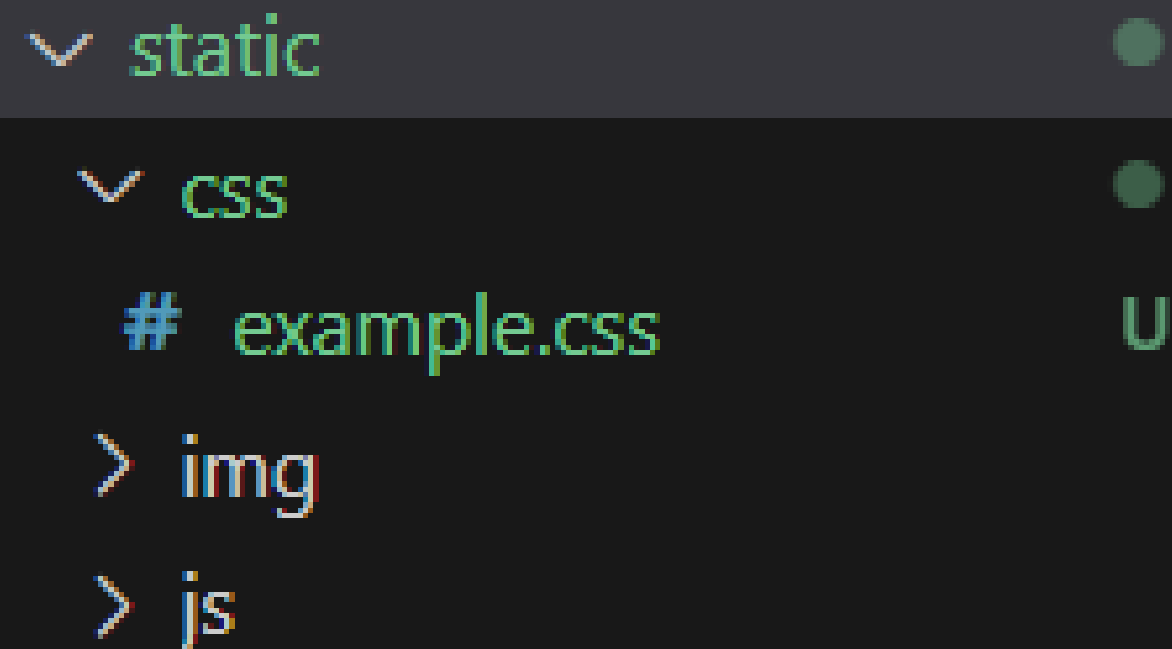
css, js 파일은 templates 폴더의 html과 달리 고정된 내용만 제공한다.

따라서 최종적으로 웹 서버를 운영할 때는

**특정 URL로 접근을 하면 해당 css, js 파일을 제공할 수 있도록 설정**해야 한다.

# Static 파일 사용법

## static 폴더 만들기



```
▼ static
  ▼ css
    # example.css
  > img
  > js
```

static 폴더를 만들고 사용자에게 보여줄

css, js와 같은 static 파일을 넣는다.

이때, 코드의 깔끔한 정리를 위해 css, js, 이미지 파일들을 관리해 줄 각각의 폴더(css,js,img)를 static 폴더 하위에 생성해 주는 것이 좋다.

# Static 파일 사용법

## settings.py 파일에 static 폴더 경로 생성

```
STATIC_URL = '/static/'  
  
STATICFILES_DIRS = [  
    BASE_DIR / 'static',  
]
```

static 파일이 어디에 저장되어 있는지 django에게 위치를 알려주어야 한다.

STATIC\_URL은 웹 페이지에서 사용할 정적 파일의 최상위 URL 경로이다. STATIC\_URL을 `'/static/'`로 설정해 `'/static/'` 경로로 요청이 들어오면 static 파일로 처리하도록 한다.

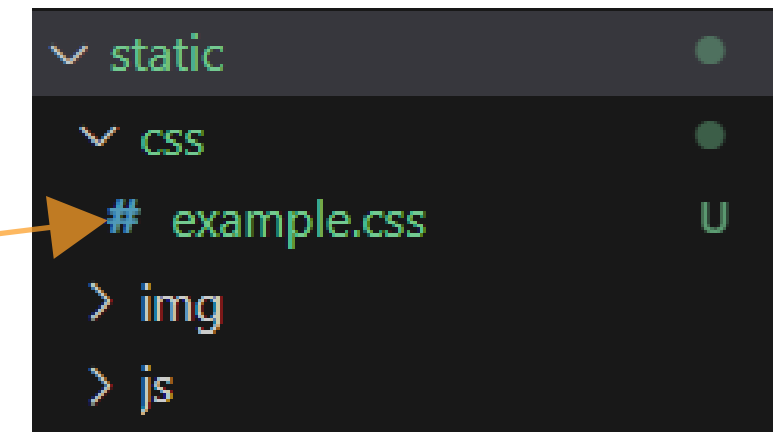
STATICFILES\_DIRS는 개발 단계에서 사용하는 정적 파일이 위치한 경로들을 지정하는 설정 항목이다. django가 static 파일을 찾을 디렉터리들을 리스트로 나타낸다.



# Static 파일 사용법

## css 파일 경로 지정하기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  {%load static%}
  <link href="{% static 'css/example.css' %}" rel="stylesheet" />
```



css 파일을 적용하고 싶은 html에서 <!DOCTYPE html> 바로 아래에 **{% load static %}**를 추가하여 static 파일을 사용하겠다고 선언해야 한다.

그리고 <head>태그 안에 위 코드처럼 css 파일 링크를 작성한다.

# 미디어 파일이란?

웹 사용자가 웹에서 업로드한 정적 파일을 미디어 파일이라고 한다.

- 파일 자체는 고정되었지만, 언제 어떤 파일이 정적 파일로 제공되고 준비되는지 예측할 수 없다.
- FileField, ImageField를 통해 파일을 저장한다.

# Static file VS Media file

## 공통점

- 정적 파일이다.

## 차이점

### Static File

- 서비스에 필요한 정적 파일을 개발자가 미리 준비해둔다.
  - 즉, 개발 단계에서 업로드해서 보여주는 파일

### Media File

- 웹 사용자가 파일을 업로드하며, 언제 어떤 파일이 추가될 지 모른다.
  - 즉, 사용자가 웹에 업로드한 파일

# 미디어 파일 사용법

## 이미지 폴더 지정하기

django는 이미지 업로드를 위한 **ImageField**를 제공한다.

ImageField를 사용하려면 사용자가 업로드한 이미지를 어디에 저장할지 먼저 설정해야 한다.

그리고 업로드 된 이미지들이 모여 있는 폴더의 URL을 어떻게 할지도 설정해야 한다.

# 미디어 파일 사용법

## 이미지 폴더 지정하기

Blog/lionblog/settings.py

```
from pathlib import Path  
import os ①
```

```
# 미디어 파일  
MEDIA_URL = "/media/" ②  
MEDIA_ROOT = os.path.join(BASE_DIR, "media")
```

① os모듈을 사용하므로 import.os을 추가한다.

② 이미지 파일은 프로젝트 폴더 아래 'media'라는 이름의 폴더를 만들고 그 안에 저장되도록 설정한다.

MEDIA\_URL은 /media/로 지정하여 웹 브라우저에서 도메인 뒤에 media/라는 경로가 따라오면 미디어 파일을 사용하도록 한다.

# 미디어 파일 사용법

## ImageField / FileField 필드 추가

blog/models.py

```
class Post(models.Model):
    title = models.CharField(max_length=50) # 글의 제목, 최대 길이는 50
    content = models.TextField() # 글의 본문
    created_at = models.DateTimeField(auto_now_add=True) # 생성일자, 해당 모델이 최초로 create() 될 때 최신 날짜 값을 채워줌
    author = models.ForeignKey(to = User, on_delete = models.CASCADE, related_name = "posts")
    category = models.ManyToManyField(to = Category, through = "PostCategory", related_name = "posts")
    like = models.ManyToManyField(to = User, through = "Like", related_name="liked_posts")
    image = models.ImageField(upload_to = upload_filepath, blank = True) # 이미지 업로드 ①
    video = models.FileField(upload_to = upload_filepath, blank = True ) # 영상 업로드

    def __str__(self):
        return f'[{self.id}] {self.title}'
```

- ① ImageField로 image를 추가하고 FileField로 video를 추가한다.

ImageField는 이미지 저장을 지원하는 모델 필드이며, FileField는 파일 저장을 지원하는 필드이다.

# 미디어 파일 사용법

## 파일 경로 설정

blog/models.py

```
from django.db import models
from users.models import User
import os
from uuid import uuid4
from django.utils import timezone
```

```
# uuid라는 고유 식별자 생성기를 통해서 파일 경로의 중복을 막음
```

```
def upload_filepath(instance, filename):
    today_str = timezone.now().strftime("%Y%m%d")
    file_basename = os.path.basename(filename)
    return f'{instance._meta.model_name}/{today_str}/{str(uuid4())}_{file_basename}'
```

# 미디어 파일 사용법

## 파일 경로 설정

blog/models.py

```
from django.db import models
from users.models import User
import os
from uuid import uuid4
from django.utils import timezone

# uuid라는 고유 식별자 생성기를 통해서 파일 경로의 중복을 막음
def upload_filepath(instance, filename):
    today_str = timezone.now().strftime("%Y%m%d")
    file_basename = os.path.basename(filename)
    return f'{instance._meta.model_name}/{today_str}/{str(uuid4())}_{file_basename}'
```

파일을 저장할 폴더의 경로 규칙을 지정한다.  
여기서는 파일이 업로드 된 날짜에 따라  
서로 다른 폴더에 저장되도록 설정했다.

`timezone.now().strftime("%Y%m%d")` : 현재 시간을 가져와서 연도, 월, 일을 문자열로 변환한다.

`os.path.basename(filename)` : 파일의 기본 이름만 사용한다. (파일 이름에서 경로 제외)

`str(uuid4())` : UUID는 중복될 가능성이 낮은 고유한 식별자이다. UUID를 파일 이름에 추가하여 파일 이름이 중복되지 않도록 한다.



# 미디어 파일 사용법

## migrate

models.py를 변경했으면 터미널창에 아래 두 가지 명령어 실행한다.

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

# 미디어 파일 사용법

## Pillow 라이브러리 설치

```
$ python manage.py makemigrations
SystemCheckError: System check identified some issues:

ERRORS:
blog.Post.image: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
users.User.profile: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
```

모델을 변경했으니 마이그레이션을 하면 위와 같은 오류가 발생한다.

오류 메시지를 잘 읽어보면 'Pillow가 설치되어 있지 않아 ImageField를 사용할 수 없다'고 적혀 있다.

# 미디어 파일 사용법

## Pillow 라이브러리 설치

Pillow는 파이썬에서 이미지를 처리하기 위한 라이브러리이다.

```
python -m pip install Pillow
```

```
$ python -m pip install Pillow
Collecting Pillow
  Downloading pillow-10.2.0-cp311-cp311-win_amd64.whl (2.6 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 6.0 MB/s eta 0:00:00
Installing collected packages: Pillow
Successfully installed Pillow-10.2.0

[notice] A new release of pip available: 22.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv)
```

오류를 해결하기 위해

```
python -m pip install Pillow
```

 입력하여

Pillow를 설치한다.

# 미디어 파일 사용법

## Pillow 라이브러리 설치

```
$ pip list
Package      Version
-----
pillow       10.2.0
pip          22.3.1
setuptools   65.5.0

[notice] A new release of pip available: 22.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv)
```

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```


## pip list

설치가 끝나면 pip list 명령어로 Pillow가 설치되었는지 확인하고 다시 마이그레이션을 한다.

# 미디어 URL 지정

## 미디어 파일을 위한 URL 지정

Blog/lionblog/blog/urls.py

Blog > lionblog > blog >  urls.py > ...

```
1  from django.urls import path
2  from .views import *
3  from django.conf import settings
4  from django.conf.urls.static import static
5
6  app_name = 'blog'
7  urlpatterns = [
8      path('', list, name = "list"),
9      path('create/', create, name = "create"),
10     path('detail/<int:id>', detail, name = "detail"),
11     path('update/<int:id>', update, name = "update"),
12     path('delete/<int:id>', delete, name = "delete"),
13     path('create-comment/<int:post_id>', create_comment, name = "create-comment"),
14     path('add-like/<int:post_id>', add_like, name="add-like"),
15     path('remove-like/<int:post_id>', remove_like, name="remove-like"),
16     path('my-like/', mylike, name="my-like")
17 ]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

# 미디어 URL 지정

## 미디어 파일을 위한 URL 지정

urls.py에서 media URL에 대한 설정을 한 것이다.

```
from django.urls import path
from .views import *
from django.conf import settings
from django.conf.urls.static import static
```

static 폴더와 settings.py 파일을 사용해야 하므로  
왼쪽 코드를 추가한다.

```
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

⇒ 장고에서 미디어 파일에 대한 URL을 제공하는 역할을 한다.

만약, 미디어 파일에 대한 URL 설정을 하지 않으면 Page not Found 오류가 발생할 것이다.

# views.py 수정

## views.py create 함수 수정

blog/views.py

```
def create(request):
    # 존재하는 모든 카테고리 확인할 수 있도록
    categories = Category.objects.all()

    if request.method == "POST":
        title = request.POST.get('title')
        content = request.POST.get('content')
        video = request.FILES.get('video')
        image = request.FILES.get('image')

        category_ids = request.POST.getlist('category')
        category_list = [get_object_or_404(Category, id =

        post = Post.objects.create(
            title = title,
            content = content,
            author = request.user,
            image = image,
            video = video
        )
```

# views.py 수정

## views.py create 함수 수정

blog/views.py

```
def create(request):  
    # 존재하는 모든 카테고리 확인할 수 있도록  
    categories = Category.objects.all()  
  
    if request.method == "POST":  
        title = request.POST.get('title')  
        content = request.POST.get('content')  
        video = request.FILES.get('video')  
        image = request.FILES.get('image')  
  
        category_ids = request.POST.getlist('category')  
        category_list = [get_object_or_404(Category, id =  
  
        post = Post.objects.create(  
            title = title,  
            content = content,  
            author = request.user,  
            image = image,  
            video = video  
        )
```

- ① 사용자가 POST 요청을 보낼 때 전송한 비디오 파일과 이미지 파일 받아오기
- ② 새로운 게시물의 image와 video를 지정



# views.py 수정

## views.py update 함수 수정

blog/views.py

```
def update(request, id):  
    post = get_object_or_404(Post, id = id)  
    if request.method == "POST":  
        post.title = request.POST.get('title')  
        post.content = request.POST.get('content')  
        video = request.FILES.get('video')  
        image = request.FILES.get('image')  
  
        if video:  
            post.video.delete()  
            post.video = video  
        if image:  
            post.image.delete()  
            post.image = image  
  
    post.save()
```

# views.py 수정

## views.py update 함수 수정

blog/views.py

```
def update(request, id):  
    post = get_object_or_404(Post, id = id)  
    if request.method == "POST":  
        post.title = request.POST.get('title')  
        post.content = request.POST.get('content')  
        video = request.FILES.get('video')  
        image = request.FILES.get('image')  
  
        if video:  
            post.video.delete()  
            post.video = video  
        if image:  
            post.image.delete()  
            post.image = image  
  
    post.save()
```

- ① 파일 필드로 전송된 데이터 받아오기
- ② 새로운 이미지나 비디오 파일을 업로드했다면,  
이전 파일 삭제 후 새로운 파일로 업데이트

# html 수정

## create.html 수정

blog/templates/blog/create.html

```
<body>
<h2>글 작성</h2>
<form method="POST" action="{%url 'blog:create'%}" enctype="multipart/form-data">
    {%csrf_token%} 제목 - <input type="text" name="title" /><br /><br />
    내용 - <textarea name="content"></textarea><br /><br />
    동영상 - <input type="file" name="video"><br><br>
    이미지 - <input type="file" name="image"><br><br>
    카테고리 -
    <select name="category" multiple>
```

# html 수정

## create.html 수정

blog/templates/blog/create.html

```
<body>
<h2>글 작성</h2>
<form method="POST" action="{% url 'create' %}" enctype="multipart/form-data">
    {% csrf_token %}
    제목 - <input type="text" name="title"><br><br>
    내용 - <textarea name="content"></textarea><br><br>
    동영상 - <input type="file" name="video"><br><br>
    이미지 - <input type="file" name="image"><br><br>
    카테고리 -
    <select name="category" multiple>
        {% for category in categories %}
```

- ① `enctype` 폼 데이터가 서버로 제출될 때 해당 데이터가 인코딩 되는 방법을 명시  
`"multipart/form-data"` 모든 문자를 인코딩 하지 않으며, 파일이나 이미지를 서버로 전송할 때 주로 사용
- ② `<input type="file">` 태그를 활용하여 파일을 선택할 수 있는 파일 입력란 생성  
`name="video/image"` 폼 데이터에서 이 파일을 식별할 때 사용되는 이름

# html 수정

## detail.html 수정

blog/templates/blog/detail.html

```
<body>
<h2>제목 - {{ post.title }}</h2>
<p>내용 - {{ post.content }}</p>
<p>작성일 - {{ post.created_at }}</p>
<p>작성자 - {{ post.author.nickname }}</p>

<!-- 영상 & 이미지 -->
{% if post.image %}
    
{% endif %}
{% if post.video %}
    <video width="320" height="240" controls>
        <source src="{{ post.video.url }}" type="video/mp4">
    </video>
{% endif %}
```

①

- ① 조건문을 통해 게시물에 이미지 또는 동영상이 첨부되어 있는지 확인 후, 이미지 또는 동영상이 첨부되어 있을 경우 해당 파일을 표시

# html 수정

## update.html 수정

blog/templates/blog/update.html

```
<h2>글 수정</h2>
<form method="POST" action="{% url 'blog:update' post.id %}" enctype = "multipart/form-data">
    {%csrf_token%}
    제목 - <input type="text" value="{{post.title}}" name="title" /><br /><br />
    내용 - <textarea name="content">{{post.content}}</textarea><br /><br />

    {% if post.image %}
        기존이미지 : <br><br>
    {%endif%}
    이미지 변경 - <input type="file" name="image"><br><br>

    {% if post.video %}
        기존 비디오 : <video width="320" height="240" controls>
            <source src="{{post.video.url}}" type="video/mp4"><br><br>
        </video>
    {% endif %}
    비디오 변경 - <input type="file" name="video"><br><br>

    <input type="submit" value="수정하기" />
```

# views.py 수정

## update.html 수정

blog/templates/blog/update.html

```
<body>
<h2>글 수정</h2>
<form method="POST" action="{% url 'update' post.id %}" enctype="multipart/form-data">
    {% csrf_token %}
    제목 - <input type="text" value="{ {{ post.title }}" name="title"><br><br>
    내용 - <textarea name="content">{{ post.content }}</textarea><br><br>
    {% if post.image %}
        기존 이미지 : <br><br>
    {% endif %}
    이미지 변경 - <input type="file" name="image"><br><br>
    {% if post.video %}
        기존 비디오 : <video width="320" height="240" controls>
            <source src="{ {{ post.video.url }}" type="video/mp4"><br><br>
        </video>
    {% endif %}<br><br>
    비디오 변경 - <input type="file" name="video"><br><br>
    <input type="submit" value="수정 하기">
</form>
</body>
```

① 기존에 파일이 첨부되어 있을 경우, 기존 파일을 표시한다.

또한, 이미지/비디오 변경을 위한 파일 입력란도 제공하고 새 파일을 선택하여 업로드 할 수 있도록 한다.



# 실행 결과

← → ↺ ⓘ 127.0.0.1:8000/create/

NAVER Keep 검사 파파고 유튜브

## 글 작성

제목 - 축제 재미있다!

내용 - ㅎㅎ

동영상 - 파일 선택 IMG\_8901.MOV

이미지 - 파일 선택 duksung\_symbol.png

카테고리 - 음식  
의류  
잡화

작성하기

← → ↺ ⓘ 127.0.0.1:8000/detail/2/

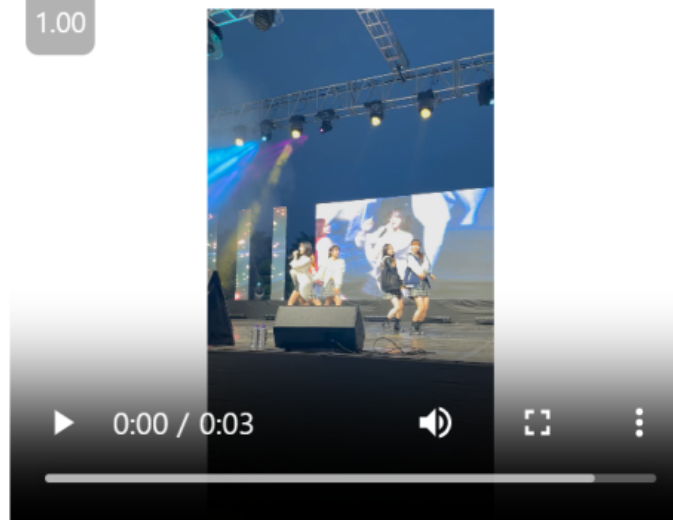
NAVER Keep 검사 파파고 유튜브 PPT DSWU 블로그 대외활동

## 제목 - 축제 재미있다!

내용 - ㅎㅎ

작성일 - 2024년 6월 20일 5:13 오후

작성자 -



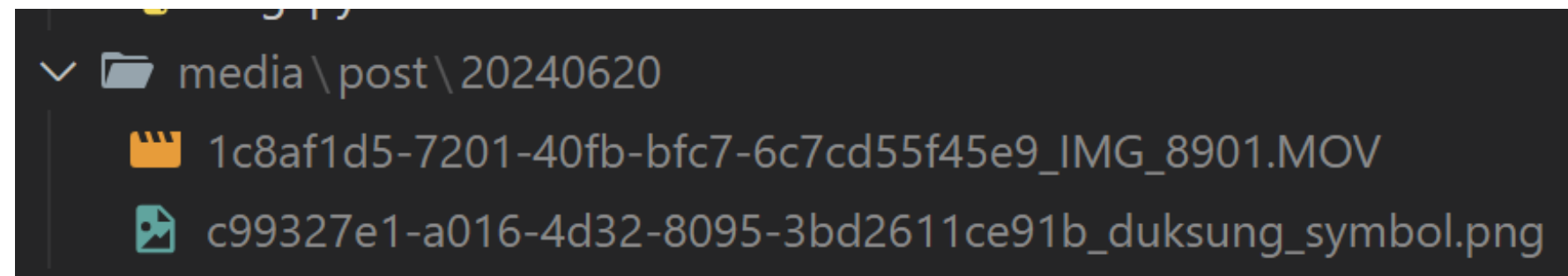
좋아요 - 0 [좋아요](#)

카테고리 - [음식]

동영상과 이미지 파일을 선택할 수 있도록  
변경되어 있음을 확인할 수 있다.



## 실행 결과



이미지 파일을 업로드 한 후, vscode 탐색기를 확인해보면 media 폴더가 생성되어 있고, 앞서 설정한 upload\_filepath 경로대로 이미지 파일이 저장되어 있음을 확인할 수 있다.

# 세션 과제 공지

과제 (마감기한 : ~ 6/26(수) 23:59)

01

## 실습 내용 업로드

1. 동영상과 이미지 파일을 올린 상태의 `create.html`, `detail.html`, `update.html` 페이지  
캡처해서 노션에 업로드
2. 실습 코드는 깃허브에 올리기

# 스터디 공지

과제 (마감기한 : ~ 6/30 (일) 23:59)

# 01 에브리타임 수정하기

← → ↺ ⓘ 127.0.0.1:8000/cate1/

NAVER Keep 검사 파파고 유튜브 PPT DSWU

[메인페이지로 돌아가기](#)

[cate1](#) [cate2](#)

## cate1

[로그아웃](#) [마이페이지](#)

제목

내용내용

☐ 익명

동영상

파일 선택

IMG\_8901.MOV

이미지

파일 선택

IMG\_9814.png

작성 완료

동영상, 이미지 업로드  
버튼 만들기

## cate1

[로그아웃](#) [마이페이지](#)

제목 입력

본문 입력

☐ 익명

동영상

파일 선택

선택된 파일 없음

이미지

파일 선택

선택된 파일 없음

작성 완료

제목

내용내용

지금

댓글 0 공감 0개

글 목록은 수정 X

01

# 에브리타임 수정하기

← → ↺ 127.0.0.1:8000/4/

NAVER Keep 검사 파파고 유튜브 PPT DSWU 블로그 대외활동 멧사

뒤로가기

제목

내용내용

2024년 6월 20일 6:39 오후

admin

1.00



0:00 / 0:03



수정하기 삭제하기

공감 0개

스크랩 0개

공감

detail 페이지에서는  
업로드한 사진과 영상을  
볼 수 있음

← → ↺ 127.0.0.1:8000/update/4/

NAVER Keep 검사 파파고 유튜브 PPT DSWU 블로그 대외활동

제목

내용내용

BookMate

기존 이미지 :

이미지 변경 - 파일 선택 선택된 파일 없음

1.00



0:00 / 0:03



기존 비디오 :

비디오 변경 - 파일 선택 선택된 파일 없음

작성 완료

update 페이지에서는  
업로드한 사진과 영상을  
수정할 수 있음

01

# 에브리타임 수정하기

1. 동영상과 이미지 파일을 올린 상태의 글 작성 페이지, 글 세부 조회 페이지, 수정 페이지  
캡처해서 노선에 업로드
2. 실습 코드는 깃허브에 올리기

# 정적 파일 및 미디어 파일 관리

12기 덕성여자대학교 멋쟁이사자처럼 7차 세션