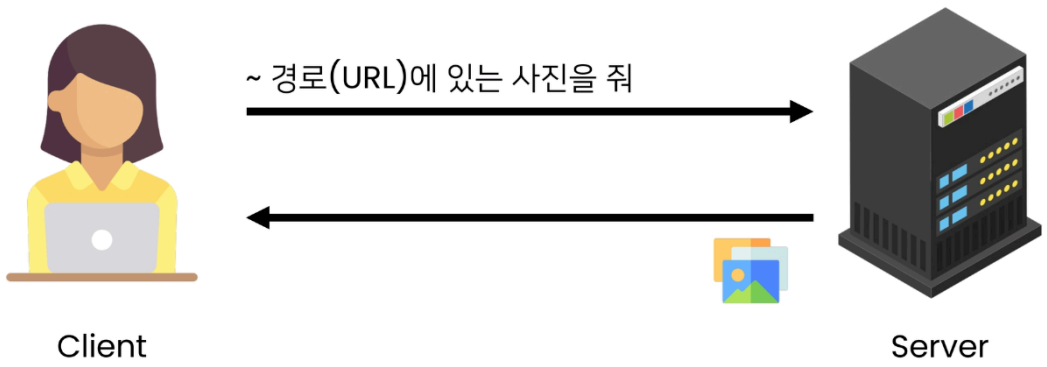
Day7 static file & media file

자료	Django
구분	Django
과목	

정적 파일이란? (static file)

- HTML을 제외하고 웹 페이지를 렌더링 할 때 필요한 추가적인 파일을 의미한다.
- Django에서 정적파일은 아래의 두가지 파일로 분류될 수 있다.
  - **Static 파일** : 서비스 안에 이미지, CSS처럼 개발자가 서버를 개발할 때 미리 넣어 준비해 놓은 파일로, 서비스 중에 추가되거나 수정되지 않는 파일
  - **Media 파일** : 유저가 업로드 할 수 있는 파일로 언제 어떤 파일이 제공될 지는 예측할 수 없는 파일들을 의미한다. 예를들자면 프로필 사진 등이 있겠다.

웹 서버와 정적 파일



Static files 구성하기

Django에서 정적 파일을 구성하고 사용하기 위한 몇 가지 단계가 있다.

1. `INSTALLED_APPS` 에 `django.contrib.staticfiles` 가 포함되어 있는지 확인하기
2. `settings.py` 에서 `STATIC_URL` 을 정의하기

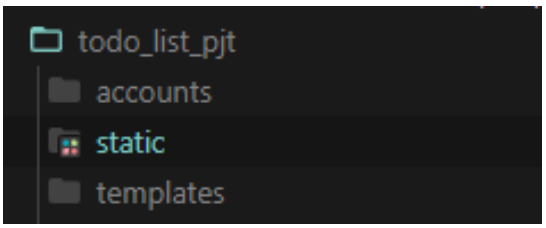
```
# settings.py

INSTALLED_APPS = [
    'articles',
    'django.contrib.admin',
```

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

...

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

3. Static 폴더 생성 후

   이미지 파일 (정적파일) 을 폴더 안에 넣기

   - ex) `최상단 프로젝트폴더/static/sample_img.jpg`



4. 파일을 불러올 템플릿에서

   static 템플릿 태그를 사용하여 지정된 경로에 있는 정적 파일의 URL 만들기

   (나는 예시로 index.html 파일에서 불러와 보겠다)

```
{% load static %}

<img src="{% static "ssafy.jpg" %}" alt="ssafy">
```

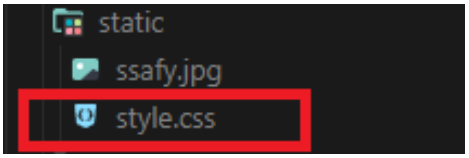
[Main] [Create] [Login]

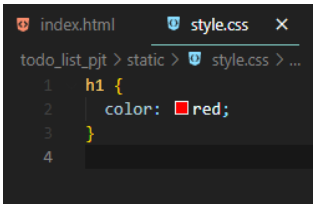
할 일 목록 페이지

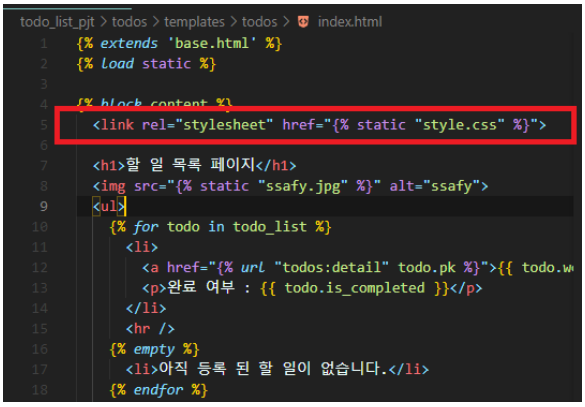
CSS를 적용 할 때에도 방법은 같다.

static 폴더에 style.css 파일 하나 만들고

적용하고 싶은 html 파일에서 {% *load* static %} 이후에 css파일을 불러와 사용하면 된다.







Django template tag

{% load %}

- 특정 라이브러리, 패키지에 등록된 모든 템플릿 태그와 필터를 불러 올 때 사용하는 tag 이다.

{% static '' %}

- STATIC_ROOT에 저장된 정적 파일에 연결할 때 사용하는 tag 이다.

사실 Static 파일을 불러오는 방법은 2가지가 있다.

위에서 우리가 실습한 것은 settings.py 들어가서

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

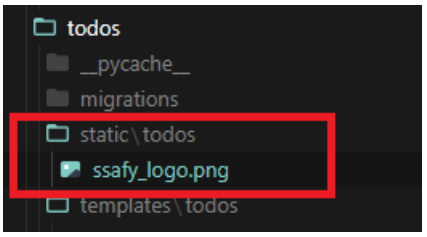
을 적어 줌으로써 static 파일을 관리할 폴더를 따로 만들어서 경로를 우리가 따로 지정해 준 케이스 이다. 프로젝트의 규모가 크거나 복잡한 프로젝트 같은경우 추가경로를 적극적으로 사용하여 개발을 한다.

하지만, 소규모 개인 개발에서는 static 파일을 별도로 (전역으로) 관리를 하기 보다는, 각 앱별로 static 파일을 따로 관리하는 경우도 많이 있다.

앞서서 sataic 파일을 추가경로를 지정해서 사용하는 것은 위에서 이미 해 보았기 때문에 이번에는 기본경로, 즉 앱 폴더 안에 static 파일을 만들어서 불러오는 것을 해보자.

이번에는 앱 안에 static폴더를 만들고 template 폴더를 만들었던 것 처럼

static 폴더 안에 앱이름으로 폴더를 하나 더 만들어 보자.



그리고 아무 이미지 파일을 폴더 안에 넣어주자.

끝이다. 앱 안에 static 폴더를 만들었을 경우에는

아까와 같이 settigs.py에 따로 설정을 해 줄 것이 없다.

원하는 곳에서 불러와보자.

(나는 create_todo.html 에서 불러와 보겠다.)

migrations / migrate를 하고 다시 서버를 켜서 확인해 보자.

이곳에 할 일을 생성합니다.

Work:

자 정리를 해보자.

static 파일을 이용하는 방법은 2가지가 있다.

1. 기본경로 - 각 앱 안에 static 파일 만들어서 사용

   소규모 개발 또는 각 앱안에 적용할 이미지 또는 css를 개별적으로 적용시킬 때 사용

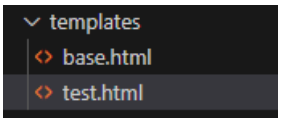
2. 추가경로 - settings.py에 경로 추가 후 사용

   대규모 개발 또는 프로젝트 전역으로 적용할 이미지 또는 css를 사용할때 이용한다.

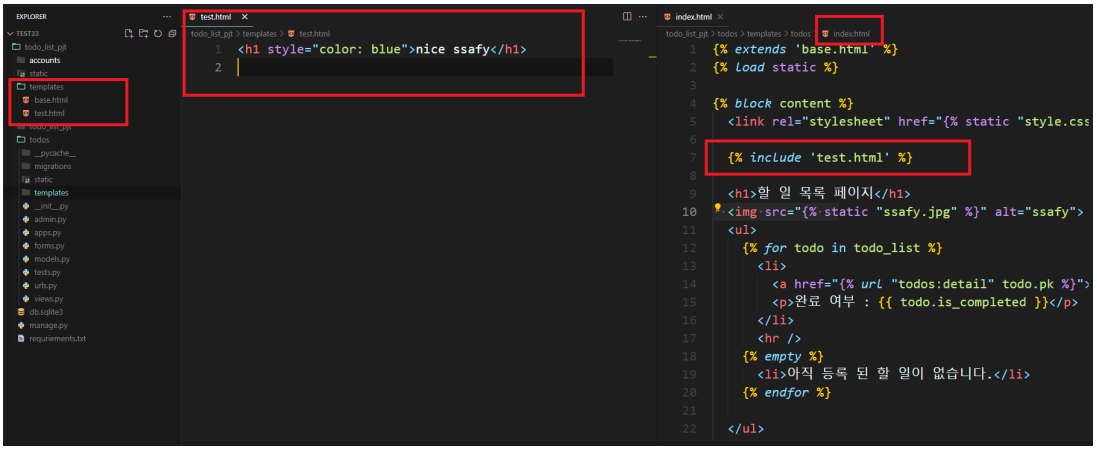
참고로 두 경로가 충돌할 경우 기본경로로 불러온 파일이 우선 적용된다.

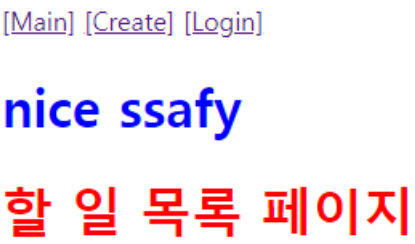
[추가] html 문서안에 다른 html 문서 불러오기!

1. 전역으로 만들어놓은 templates 폴더 안에 test.html 파일 생성



2. test.html 파일 안에는 h1 태그 그리고 “nice ssafy”를 작성
3. index.html 파일에서 test.html 파일 불러 올 것임
4. 원하는 위치에 {% *include* 'test.html' %} 넣기 <끝>





[공식문서]

https://docs.djangoproject.com/en/5.0/ref/templates/builtins/#include

여기까지 static 파일도 해보고 추가적으로 html 파일도 불러오는 것을 해 보았다.

정리 해보자.

Static Files 정리

기본경로

1. 앱폴더 안에 static/앱네임 폴더에 이미지 파일 넣기
2. {% load static %}
3. img stc="{% static 경로 %}

추가경로

1. settings.py에 STATIC_URL = 'static/'
2. STATICFILES_DIRS = [
   BASE_DIR / 'static',
   ]
3. static 폴더 만들고 파일넣기

이번에는 유저가 이미지 파일을 업로드 할 수 있도록 media 파일을 구현해 보자.

Media Files 시작

ImageField()

- 이미지 업로드에 사용되는 모델 필드
- 사용자에 의해 업로드 된 객체가 유효한 이미지인지 검사기능도 포함되어 있음

ImageField 를 사용하기

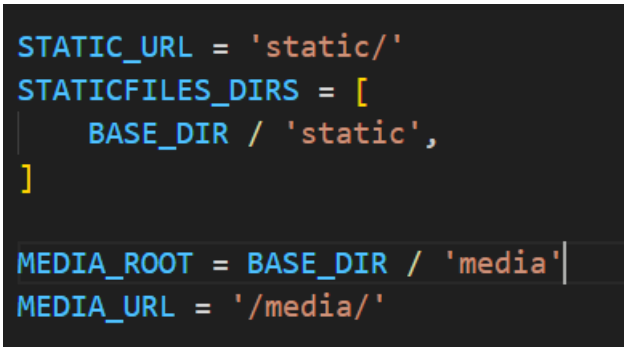
단계 1. settings.py에 `MEDIA_ROOT` , `MEDIA_URL` 설정

- MEDIA_ROOT란?
  - 사용자가 업로드 한 파일들을 보관할 폴더의 경로를 지정하는 설정이다.
  - Django는 성능을 보장하기 위해 업로드한 파일은 데이터베이스에 저장하지 않는다. 데이터베이스에 저장되는 것은 파일의 경로다. (문자열)
  - `MEDIA_ROOT` 는 `STATIC_ROOT` 와 반드시 경로가 달라야 한다. 같으면 안된다.
  - 설정해보자. settings.py 파일의 하단에 아래 코드를 추가하자.

```
# settings.py

# media
MEDIA_ROOT = BASE_DIR / 'media'
```

- MEDIA_URL이란?
  - 업로드된 파일을 **웹에서 접근할 수 있도록 하는 URL 경로**를 지정하는 설정이다.
  - 비어 있는 값으로 설정할 것이 아니라면 반드시 slash(/)로 끝나야 한다.
  - MEDIA_URL은 STATIC_URL과 반드시 다른 경로로 지정해야 한다.
  - 아래 코드도 추가하자.

```
# settings.py

# media
MEDIA_URL = '/media/'
```



단계 2. 개발 단계에서 사용자가 업로드한 미디어 파일 제공하기

- 사용자로부터 업로드 된 파일이 프로젝트에 업로드 되고나서, 실제로 사용자에게 제공하기 위해서는 업로드된 파일의 URL 필요하다.
  - 업로드 된 파일의 URL == `settings.MEDIA_URL`
  - 위 URL을 통해 참조하는 파일의 실제 위치 == `settings.MEDIA_ROOT`
  - 즉, 사용자가 파일을 업로드하면:
    1. Django는 `MEDIA_ROOT`에 파일을 저장한다.
    2. 사용자는 `MEDIA_URL`을 통해 파일을 불러올 수 있다.

todo_list_pjt 폴더 안에 있는 urls.py에 코드를 추가하자.

```
# 프로젝트폴더의/urls.py

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/', include('articles.urls')),
] + static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```

Media File 사용하기

CREATE

ImageField 작성

- 기존 컬럼 사이에 작성해도 실제 테이블에 추가 될 때는 가장 우측(뒤)에 추가됨

```
# todos/models.py

from django.db import models
```

```
class Todo(models.Model):
    work = models.CharField(max_length=100)
    content = models.TextField()
    is_completed = models.BooleanField()
    image = models.ImageField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Model field option - `blank`

- 유효성 검사에서 사용 된다. (is_valid)
  - 필드에 `blank=True` 가 있으면 form 유효성 검사에서 빈 값을 입력 받을 수 있다.
  - True인 경우 필드를 비워 둘 수 있음 → DB에는 ''(빈 문자열) 저장

Migrations

- ImageField를 사용하려면 반드시 Pillow 라이브러리가 필요하다.
  - Pillow 설치 없이는 makemigrations 실행 불가하다.

```
$ pip install pillow
$ pip freeze > requirements.txt

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

- [참고] Pillow
  - 광범위한 파일 형식 지원, 이미지 처리 기능을 제공하는 라이브러리

enctype 속성 변경

- 파일 또는 이미지 업로드 시에는 form 태그에 enctype 속성을 다음과 같이 변경해야 한다.
- enctype이란?

  폼 데이터를 서버로 전송할 때의 인코딩 방식을 지정하는 것을 말한다. 이미지와 같은 파일을 업로드 할때에는 form 태그에서 기존의 데이터를 인코딩 하는 방식으로 파일을 전송 할 수 없기 때문에 속성을 하나 추가 해 주는 것이다.

```
{% extends 'base.html' %}

{% load static %}

{% block content %}

<h1>이곳에 할 일을 생성합니다.</h1>
```

```
<img src="{% static "todos/sample.png" %}" alt="test">

<form action="{% url 'todos:create_todo' %}" method="POST"  enctype="multipart/form-data">
 {% csrf_token %}
 {{ form.as_p }}
 <input type="submit" value="제출하기" />
</form>
{% endblock %}
```

자 이제 views.py 에 있는 함수에서 해당 이미지 파일을 받아보자.

request.FILES

- 파일 및 이미지는 request의 POST 속성 값으로 넘어가지 않고 FILES 속성 값에 담겨서 넘어간다.

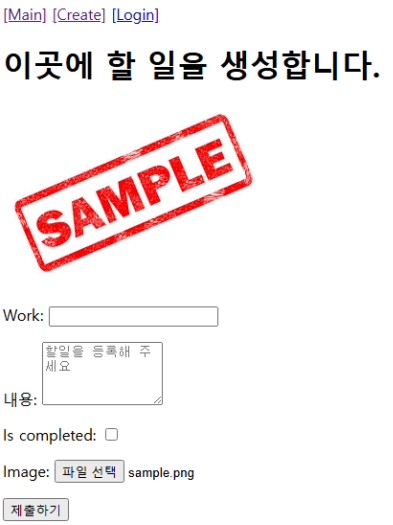
```
# todos/views.py

def create_todo(request):
    if request.method == 'POST':
        form = TodoForm(request.POST, request.FILES)
    ...
```

이미지 첨부하기

migrations, migrate를 한 후에 서버를 켜보자.

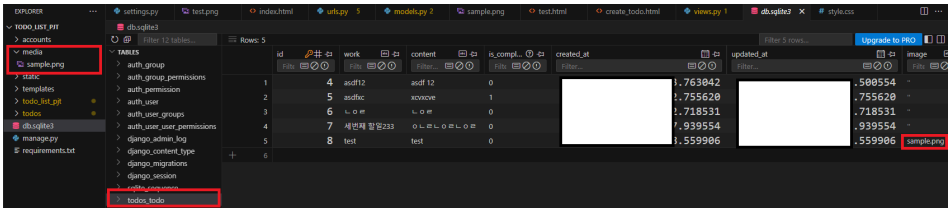
create 파트로 가서 브라우저에서 파일을 업로드 하고

새로운 게시글을 생성해 보자. 그리고 DB를 확인해 보자.



- 이미지를 첨부하지 않으면 blank=True 속성으로 인해 빈 문자열 저장이 될 것이고 이미지를 첨부한 경우는 MEDIA_ROOT 경로에 이미지가 업로드 될 것이다.

  파일 자체가 업로드 되는 것이 아니라 파일의 경로가 저장된다는 사실을 기억하자.



READ

업로드 이미지 출력하기

- 업로드 된 파일의 상대 URL은 Django가 제공하는 url 속성을 통해 얻을 수 있다

```
<!-- todos/detail.hmtl -->

<h1>{{ todo.work }}의 상세 페이지 입니다.</h1>
<hr />

{% if todo.image %}
  <img src="{{todo.image.url}}">
{% endif %}
```

- `todo.image.url` - 업로드 파일의 경로
- `todo.image` - 업로드 파일의 파일 이름
- 이미지를 업로드 하지 않은 게시물을 detail 템플릿을 출력할 수 없는 문제가 발생 할 수가 있다. 따라서 이미지 데이터가 있는 경우에만 이미지 출력할 수 있도록 if문 처리 했다.

서버 켜서 확인 해보자.

UPDATE

업로드 이미지 수정하기

- enctype 속성 값 추가

```
<!-- todos/update_todo.hmtl -->

{% extends 'base.html' %} {% block content %}
<h1>이 곳에서 할 일을 수정합니다.</h1>

<form action="{% url 'todos:update_todo' todo.pk %}"
                                    method="POST" enctype="multipart/form-data" >
...
```

- 이미지 파일이 담겨 있는 request.FILES 추가 작성

```
# todos/views.py

def update_todo(request, todo_pk):
    todo = Todo.objects.get(pk=todo_pk)
    if request.method == 'POST':
        form = TodoForm(request.POST, request.FILES, instance=todo)
```

서버 켜서 수정 잘 되는지 확인해보자.

[참고]
Django의 imageField 또는 fileField에서 upload_to 옵션을 사용하면 파일이 저장될 경로를 동적으로 지정할 수도 있다는 것을 알아두자.