| 구분 | Django |
| --- | --- |



# Media Files

## ImageField()

- 이미지 업로드에 사용되는 모델 필드
- FileField를 상속받는 서브 클래스이기 때문에 FileField의 모든 속성 및 메서드를 사용 가능
- 더해서 사용자에 의해 업로드 된 객체가 유효한 이미지인지 검사
- ImageField 인스턴스는 최대 길이가 100자인 문자열로 DB에 생성되며, max_length 인자를 사용하여 최대 길이를 변경할 수 있음

## FileField()

- `FileField(upload_to='', storage=None, max_length=100, **options)`
- 파일 업로드에 사용하는 모델 필드
- 2개의 선택 인자를 가지고 있음
  1. upload_to
  2. storage

## FileField / ImageField 를 사용하기 위한 단계

1. settings.py에 `MEDIA_ROOT`, `MEDIA_URL` 설정
2. `upload_to` 속성을 정의하여 업로드된 파일에 사용할 MEDIA_ROOT의 하위 경로를 지정 (선택사항)

## MEDIA_ROOT

- Default : '' (Empty string)
- 사용자가 업로드 한 파일(미디어 파일)들을 보관할 디렉토리의 절대 경로
- Django는 성능을 위해 업로드한 파일은 데이터베이스에 저장하지 않음
  - 데이터베이스에 저장 되는 것은 파일 경로
- `MEDIA_ROOT`는 `STATIC_ROOT`와 반드시 다른 경로로 지정해야 함

```
# settings.py

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

## MEDIA_URL

- Default : '' (Empty string)
- MEDIA_ROOT에서 제공되는 미디어 파일을 처리하는 URL
- 업로드 된 파일의 주소(URL)를 만들어 주는 역할
  - 웹 서버가 사용자가 사용하는 public URL
- 비어 있지 않은 값으로 설정한다면 반드시 slash(/)로 끝나야 함
- MEDIA_URL은 STATIC_URL과 반드시 다른 경로로 지정해야 함

```
# settings.py

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

## 개발 단계에서 사용자가 업로드한 미디어 파일 제공하기

- 사용자로부터 업로드 된 파일이 프로젝트에 업로드 되고나서, 실제로 사용자에게 제공하기 위해서는 업로드된 파일의 URL 필요
  - 업로드 된 파일의 URL == `settings.MEDIA_URL`
  - 위 URL을 통해 참조하는 파일의 실제 위치 == `settings.MEDIA_ROOT`

```
# crud/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

```
# articles/models.py

from django.db import models

# Create your models here.
class Article(models.Model):
    title = models.CharField(max_length=30)
    content = models.TextField()
    image = models.ImageField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

### Model field option (1) - `blank`

- Defalut : False
- True인 경우 필드를 비워 둘 수 있음
  - 이럴 경우 DB에는 ‘’(빈 문자열) 저장
- 유효성 검사에서 사용 됨(is_valid)
  - “Validation-related”
  - 필드에 `blank=True` 가 있으면 form 유효성 검사에서 빈 값을 입력받을 수 있음

### Model field option (2) - `null`

- Defalut : False
- True 인 경우 Django는 빈 값을 DB에 NULL로 저장함

    - “Database-related”
- 주의사항 : ChartField, TextField와 같은 문자열 기빈 필드에는 null 옵션 사용을 피해야 함
    - 문자열 기반 필드에 `null=True`로 설정 시 데이터 없음에 대한 표현에 “빈 문자열”과 “NULL” 2가지 모두 가능하게 됨
    - “데이터 없음”에 대한 표현에 두 개의 가능한 값을 갖는 것은 좋지 않음
    - Django는 문자열 기반 필드에서 NULL이 아닌 빈 문자열을 사용하는 것이 규칙

## Migrations

- ImageField를 사용하려면 반드시 Pillow 라이브러리가 필요
    - Pillow 설치 없이는 makemigrations 실행 불가

```
$ pip install pillow

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

$ pip freeze > requirements.txt
```

- [참고] Pillow
    - 광범위한 파일 형식 지원, 효율적이고 강력한 이미지 처리 기능을 제공하는 라이브러리
    - 이미지 처리 도구를 위한 견고한 기반 제공

## enctype 속성 변경

- 파일 또는 이미지 업로드 시에는 form 태그에 enctype 속성을 다음과 같이 변경해야 함.

```
<!-- article/create.hmtl -->

{% extends 'base.html' %}

{% block content %}
  <h1>글작성</h1>
  <hr>

  <form action="{% url 'articles:create' %}" method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {{form.as_p}}
    <input type="submit">
  </form>

{% endblock content %}
```

## request.FILES

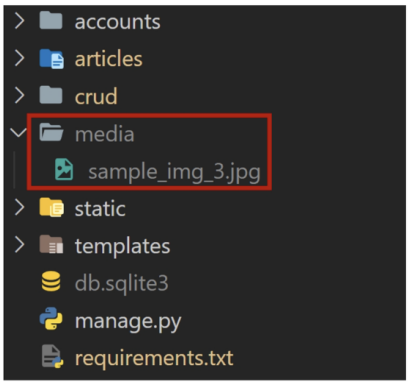
- 파일 및 이미지는 request의 POST 속성 값으로 넘어가지 않고 FILES 속성 값에 담겨 넘어감

```
# articles/views.py

def create(request):
    if request.method == 'POST':
        form = ArticleForm(request.POST, request.FILES)
  ...
```

## 이미지 첨부하기

- 이미지를 첨부하지 않으면 blank=True 속성으로 인해 빈 문자열 저장
- 이미지를 첨부한 경우는 MEDIA_ROOT 경로에 이미지가 업로드 됨

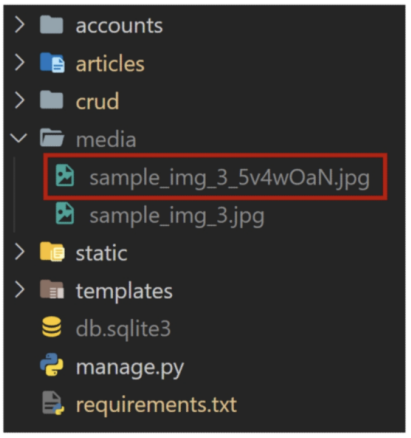



| id | title | content | created_at | updated_at | user_id | image |
|---|---|---|---|---|---|---|
| 1 | 이미지 | 올려보기 | 2022-10-05 15:41:46.363805 | 2022-10-05 15:41:46.363805 | 1 | sample_img_3.jpg |
| 2 | 이미지 없이 | 작성해보기 | 2022-10-05 15:43:40.937291 | 2022-10-05 15:43:40.937291 | 1 | |

❖ 파일 자체가 아닌 "경로"가 저장 된다는 것을 잊지 말 것

89

- 만약 같은 이름의 파일을 업로드 한다면 Django는 파일 이름 끝에 임의의 난수 값을 붙여 저장함.




# READ

## 업로드 이미지 출력하기

- 업로드 된 파일의 상대 URL은 Django가 제공하는 url 속성을 통해 얻을 수 있음
  - `article.image.url` - 업로드 파일의 경로
  - `article.image` - 업로드 파일의 파일 이름

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

<img src="{{article.image.url}}">
```

- 이미지를 업로드 하지 않은 게시물을 detail 템플릿을 출력할 수 없는 문제 해결
  - 이미지 데이터가 있는 경우에만 이미지 출력할 수 있도록 처리

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

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

# UPDATE

## 업로드 이미지 수정하기

- enctype 속성 값 추가

```
<!-- article/update.hmtl -->

{% extends 'base.html' %}

{% block content %}
  <h1>글수정</h1>
  <hr>

  <form action="{% url 'articles:update' article.pk %}" enctype="multipart/form-data" method="POST">
...
```

- 이미지 파일이 담겨 있는 request.FILES 추가 작성

```
# articles/views.py

def update(request, pk):
    article = Article.objects.get(pk=pk)
    if request.method == 'POST':
        form = ArticleForm(request.POST, request.FILES, instance=article)
```