실전 프로그램 개발
- 콘텐츠 편집 기능 -
(파일 업로드, 다운로드)

파일 업로드

❖ 파일 업로드를 위한 설정

- settings.py
  - MEDIA : url 상의 경로
  - MEDIA_ROOT : 실제 파일 시스템의 경로

❖ mysite/settings.py

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- <프로젝트 디렉토리>/media 디렉토리 생성

파일 업로드

❖ 업로드 파일을 위한 모델

- models.FileField
  - 파일 업로드를 위한 필드
  - form에서 <input type="file">로 대응
  - upload_to 인수로 저장할 위치(디렉토리) 설정
    - upload_to="%Y/%m/%d"
      업로드 당시의 년/월/일 디렉토리 생성하고 파일 업로드

- mime-type(content type)
  - 파일의 종류를 설명하는 문자열(파일명의 확장자 개념)
    - text/text
    - text/html
    - image/jpeg
    - image/gif

파일 업로드

❖ blog/models.py

```
class PostAttachFile(models.Model):
  post         = models.ForeignKey(Post, on_delete=models.CASCADE,
                                   related_name="files",
                                   verbose_name='Post', blank=True, null=True)
  upload_file  = models.FileField(upload_to="%Y/%m/%d",
                                   null=True, blank=True, verbose_name='파일')
  filename     = models.CharField(max_length=64, null=True,
                                   verbose_name='첨부파일명')
  content_type = models.CharField(max_length=128, null=True,
                                   verbose_name='MIME TYPE')
  size         = models.IntegerField('파일 크기')

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

파일 업로드

❖ 데이터베이스 반영

- `python manage.py makemigrations blog`
- `python manage.py migrate`

파일 업로드

❖ 파일 업로드를 위한 form enctype 설정

- form의 enctype 속성 설정
  - enctype="application/x-www-form-urlencoded"
    - 디폴트
    - name1=value1&name2=value2&...
  - enctype="application/json"
    - JSON 포맷
    - { "name1" : "value1", "name2":"value2", ... }
  - enctype="multipart/form-data"
    - 각각의 요소마다 header와 body로 구성 각 요소는 빈줄로 구분
    - 바이너리 데이터 전송시 사용

파일 업로드

❖ blog/templates/blog/blog_form.html

```
   :
  <form action="." method="post" enctype="multipart/form-data"
    class="card pt-3">{% csrf_token %}
   :
    <div class="form-group row">
      <label for="files"
      class="col-form-label col-sm-2 ml-3 font-weight-bold">첨부 파일 :</label>
      <div class="col-sm-8 ">
        <input type="file" id="files" name="files" multiple>
      </div>
    </div>

    <div class="form-group row">
      {{ form.content|add_label_class:"col-form-label col-sm-2 ml-3 font-
weight-bold" }}
      <div class="col-sm-8">
        {{ form.content|add_class:"form-control" }}
      </div>
    </div>
   :
```

파일 업로드

❖ 뷰에서 파일 업로드 처리

- PostAttachFile를 저장하기 위해서는 ForeignKey를 위한 연관 Post 객체 필요
- CreateView의 form_valid(form) 메서드
  - form의 정보를 기반으로 모델 객체를 저장
  - self.object를 통해 저장된 모델 객체(Post) 접근 가능
  - HttpResponse 객체를 반환

파일 업로드

❖ 뷰에서 파일 업로드 처리

- self.request.FILES에 업로드된 파일 정보 관리
- 개별 업로드 파일 정보 얻기
  - <input type="file" id="files" **name="files" multiple**>
  - self.request.FILES.getlist("요소명")
    - multiple 설정이 있으면 리스트로 반환
    - multiple이 없으면 단일 항목으로 리턴
    - 요소 타입 : InMemoryUploadedFile
      name, size, content_type

파일 업로드

❖ blog/views.py

```
class PostCreateView(LoginRequiredMixin, CreateView):
  model = Post
  fields = ['title', 'description', 'content', 'tags']
  success_url = reverse_lazy('blog:index')

  def form_valid(self, form):
    form.instance.owner = self.request.user
    form.instance.modify_dt = timezone.now()
    # return HttpResponseRedirect(self.get_success_url())
    response = super().form_valid(form) # Post 모델 저장, self.object

    # 업로드 파일 얻기
    files = self.request.FILES.getlist("files")
    for file in files:
      attach_file = PostAttachFile(post= self.object, filename = file.name,
                          size = file.size, content_type = file.content_type,
                          upload_file = file)
      attach_file.save()

    return response
```

파일 다운로드

❖ blog/views.py

```
:
from django.http import FileResponse
import os
from django.conf import settings

def download(request, id):
  file = PostAttachFile.objects.get(id=id)
  file_path = os.path.join(settings.MEDIA_ROOT, str(file.upload_file))

  return FileResponse(open(file_path, 'rb'))
```

파일 다운로드

❖ blog/urls.py

```
    :

urlpatterns = [
    :
     # Example: /blog/99/delete/
     path('<int:pk>/delete/', PostDeleteView.as_view(), name="delete"),

     path('download/<int:id>', download, name="download"),
]
```

파일 다운로드

❖ Post의 첨부파일 얻기

```
class PostAttachFile(models.Model):
  post         = models.ForeignKey(Post, on_delete=models.CASCADE,
                                   related_name="files",
                                   verbose_name='Post', blank=True, null=True)
:
```

- related_name
  - 상대 모델에 지정한 이름으로 관계 Manager가 지정됨
  - 디폴트 이름
    - 모델명_set
    - related_name으로 이름 변경 가능
  - post.files
    - 해당 포스트와 관계된 첨부 파일 관리
    - all(), get(), filter() 등 사용 가능

파일 다운로드

❖ blog/urls.py

```
   :
 <div class="text-right">작성자: {{post.owner}} / 죄회수: {{post.read_cnt}} /
수정일: {{ post.modify_dt|date:"Y년 m월 d일" }}</div>

  <div class="text-right my-3">
    {% for file in post.files.all %}
      <a href="{%url 'blog:download' file.id%}" class="ml-4">
          <i class="fas fa-download"></i>
          {{file.filename}} ({{file.size | filesizeformat}}) </a>
    {% endfor %}
  </div>
   :
```