Django advanced project #week_4(final)

Writer : Taeyeong Kim, DGIST LikeLion

Date: 2019.10.09

아래 자료를 공부하기 전에, https://github.com/lizard-kim/LikeLion_study_Fall_Semester에 있는 코드를 clone하여 참고하기 바란다. 코드에 대한 자세한 정보는 주석을 통해 알 수 있으니, 이곳에서는 대략적인 설명만 할 것이다.

Let`s make our Django API

Install REST Framework & Create your project

프로젝트를 생성하기 이전에, django_restframework를 설치하자

```
$ pip install djangorestframeowork
```

이때, 프로젝트의 settings.py에 'rest_framework' 라는 앱을 우리가 생성한 앱과 같이 등록해주자!

그 다음, 프로젝트를 생성하자.

```
$ django-admin startproject project
$ python manage.py startapp mystorage # 이후 settings.py에 app을 추가해주자
```

기본 틀 만들기

project/urls.py

```
from django.contrib import admin
from django.urls import path, include # urls을 따로 관리할 겁니다.
from mystorage import urls

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('mystorage.urls')),
]
```

먼저 project의 url을 설정해 줍시다. mystorage의 url을 include했으니, mystorage/urls.py를 작성해야겠죠?

project/urls.py

```
from rest_framework.routers import DefaultRouter #router를 기반으로 작성할겁니다.
from django.urls import path, include
from mystorage import views # views import 해주기

router = DefaultRouter()
router.register('essay', views.PostViewSet) #127.0.0.1:8000/essay/ 를 views.py의
PostViewSet함수와 연결

urlpatterns = [
    path('', include(router.urls)),
]
```

우리는 router를 이용하여 url을 설정해줄 겁니다. DefaultRouter의 register라는 method에 첫번째 인자로 url의 주소를 넣고, views의 함수를 연결해줍니다.

> 왜 router를 사용해서 url을 mapping하나요?
>
> - 기존에는 as_view() 함수를 사용해서 관례적으로 사용되던 class의 method, get / post 등과 같은 method를 mapping하였습니다. 이 과정이 반복되다보니 router의 DefaultRouter()를 사용하여 간단하게 작성하는 것입니다. DefaultRouter() 에는 이러한 method가 이미 구현이 되어있거든요!

views.py

```
from rest_framework import viewsets
from .models import Essay
from .serializers import EssaySerialize

class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer
```

이전에 배운 viewsets을 사용한 모습입니다. 지난수업에서 배웠듯이 queryset과 serializer_class를 넘겨주는 모습입니다. 이제 Essay model과 EssaySerializer serializer class를 만들어 봅시다!

models.py

```
from django.db import models
from django.conf import settings

class Essay(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, default=1,
on_delete=models.CASCADE) # user에 맞게 정해지도록 설정합니다.
    title = models.CharField(max_length=30)
    body = models.TextField()
```

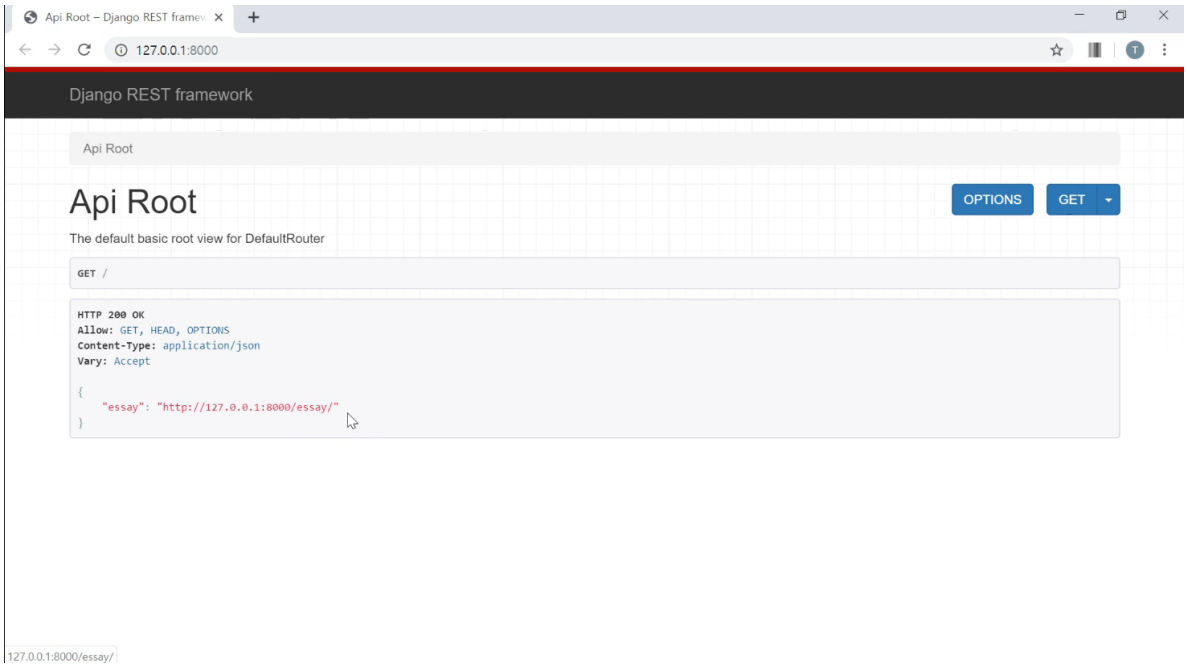
model을 작성한 이후 migrate하는것 잊지마세요! 또한 `$ python manage.py createsuperuser`를 통해 admin도 만들어줍시다!

serializers.py

```
from .models import Essay, Album, Files
from rest_framework import serializers

class EssaySerializer(serializers.ModelSerializer):

    class Meta:
        model = Essay
        fields = '__all__'
```

Run server

서버를 돌려 확인해봅시다!



> 잘 동작합니다. /admin/에 접속해서 user1, 2를 추가해 주고, /essay/에 들어가서 글도 써봅시다!

User를 자동으로 저장하기

serializers.py 수정하기

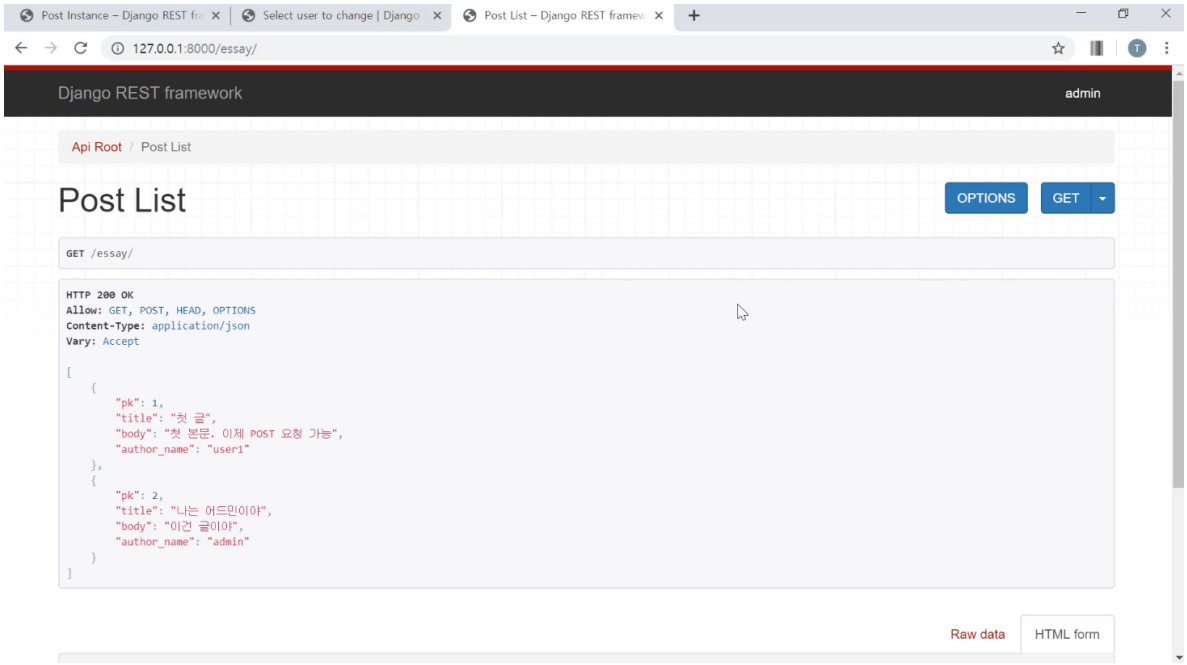
글을 작성할 때 글쓴이를 직접 지정해주는 과정이 귀찮습니다. 로그인한 회원정보를 받아 자동으로 입력되게 코드를 수정해 봅시다.

```
from .models import Essay, Album, Files
from rest_framework import serializers
from rest_framework.parsers import MultiPartParser, FormParser

class EssaySerializer(serializers.ModelSerializer):
    # author를 자동으로 지정
    author_name = serializers.ReadOnlyField(source='author.username')
    #건들지 못하게 readonly로 지정하기

    class Meta:
        model = Essay
        fields = ('pk', 'title', 'body', 'author_name')
```

views.py

```
from rest_framework import viewsets
from .models import Essay, Album, Files
from .serializers import EssaySerializer

class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    filter_backends = [SearchFilter]
    search_fields = ('title', 'body')

    def perform_create(self, serializer):
        serializer.save(author=self.request.user) #유저를 자동으로 저장
```

글정보에도 글쓴이가 자동으로 저장되도록 `perform_create`를 작성해줍시다.

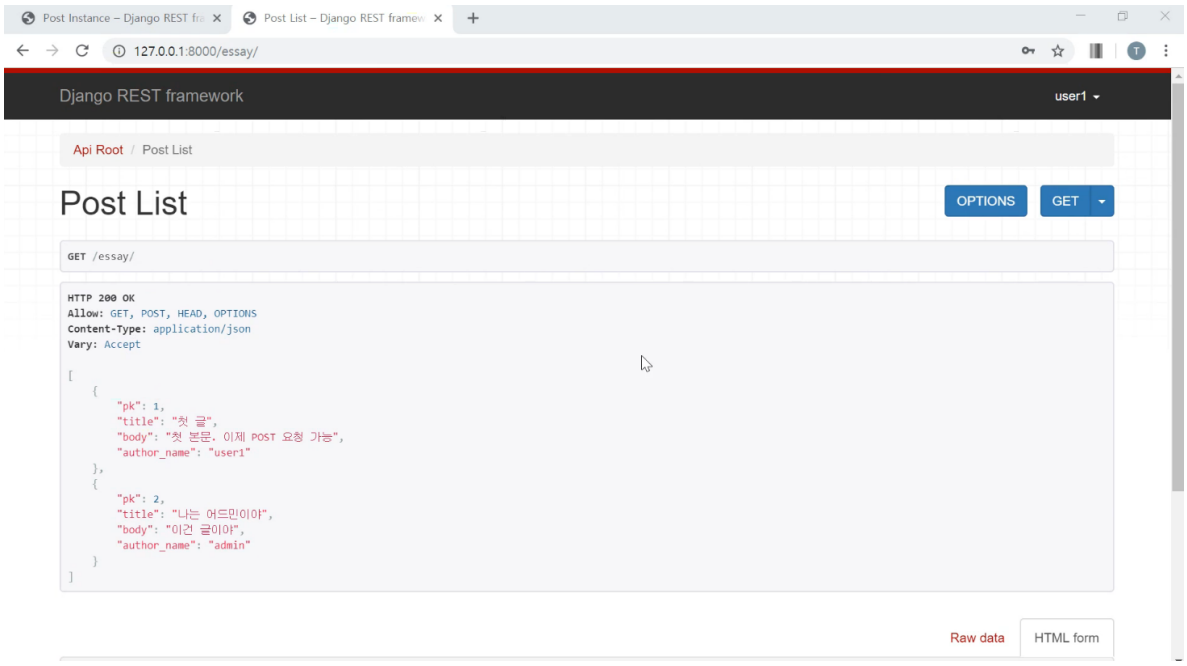


> author_name에 글쓴이가 표시되는 것을 확인할 수 있습니다.

urls.py

오른쪽 상단에 user이름이 뜨는것은 좋지만, 거기에 로그인과 로그아웃 기능을 더하고 싶습니다. 만들어봅시다.

```
from django.contrib import admin
from django.urls import path, include
from mystorage import urls
from rest_framework import urls

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('mystorage.urls')),
    path('api-auth/', include('rest_framework.urls')), #rest_framework에 구현되어있는 기능을 사용합시다.
]
```



> 오른쪽 상단의 버튼을 눌러보면 잘 구현되있음을 알 수 있습니다.

List Filtering & Searching

접속한 user에 따라 list에 본인의 글만 보일 수 있도록 만들어 봅시다.

views.py

```
from rest_framework import viewsets
from .models import Essay, Album, Files
from .serializers import EssaySerializer
from rest_framework.filters import SearchFilter


class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    filter_backends = [SearchFilter] # search
    search_fields = ('title', 'body')
```

```
    def perform_create(self, serializer):
        serializer.save(author=self.request.user) #유저를 자동으로 저장

    def get_queryset(self): # 본인의 글만 보이게 한다.
        qs = super().get_queryset()

        if self.request.user.is_authenticated:
            qs = qs.filter(author = self.request.user)
        else:
            qs = qs.none()

        return qs
```

> 서버를 돌려보면 잘 작동하는 것을 알 수 있습니다. filtering, searching의 원리는 강의를 참고해주세요!

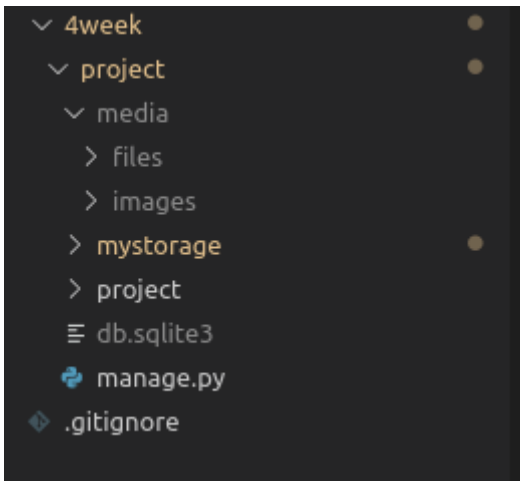
Media File Upload

이제 우리가 만든 서버에 이미지, 파일을 올릴 수 있도록 만들어 봅시다. 이전 django강의의 연장선이니 편하게 따라해 보세요!

settings.py

```
STATIC_URL = '/static/'

# now adding~~~~
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

settings.py에서 media 경로를 지정해 줍시다. 만든 이후, 우리의 프로젝트 안에 media 폴더를 만들어 줍시다.



> 사진은 images에, 그 외의 파일은 files에 저장할 수 있도록 하위폴더를 만들었습니다.

project/urls.py

```
from django.contrib import admin
from django.urls import path, include
```

```
from mystorage import urls
from rest_framework import urls
from django.conf import settings
from django.conf.urls.static import static # static 함수 사용


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('mystorage.urls')),
    path('api-auth/', include('rest_framework.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) #
urlpatterns에 media가 추가될 수 있도록, 인자로는 media파일의 url과 경로를 지정한다.
```

> 이전 장고 강의에서 다룬 내용입니다. 기억나시나요?

models.py

media파일을 업로드 해야하니까 그에 맞게 model을 정의해야겠죠?

```
from django.db import models
from django.conf import settings

class Essay(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, default=1,
on_delete=models.CASCADE) # user에 맞게 정해지도록 설정합니다.
    title = models.CharField(max_length=30)
    body = models.TextField()

class Album(models.Model): # 사진
    author = models.ForeignKey(settings.AUTH_USER_MODEL, default=1,
on_delete=models.CASCADE) # 위와 마찬가지로 user에 맞게 정해지도록 합니다.
    image = models.ImageField(upload_to="images") # images 폴더에 업로드 됩니다.
    desc = models.CharField(max_length=100)

class Files(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, default=1,
on_delete=models.CASCADE)
    myfile = models.FileField(blank=False, null=False, upload_to="files") #
files 폴더에 업로드 됩니다.
    desc = models.CharField(max_length=100)
```

model을 작성한 이후, `$ pip install Pillow`를 통해 이미지 파일을 잘 다룰 수 있도록 합시다.

Pillow까지 설치가 완료되었다면 migrate해줍시다!

mystoage/urls.py

이미지와 파일을 띄울 수 있도록 url을 추가로 설정해줍시다.

```
from rest_framework.routers import DefaultRouter #router를 기반으로 작성할겁니다.
from django.urls import path, include
from mystorage import views

router = DefaultRouter()
router.register('essay', views.PostViewSet)  #127.0.0.1:8000/essay/ 를 views.py의
PostViewSet함수와 연결
router.register('album', views.ImgViewSet)
router.register('files', views.FileViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

views.py

urls.py에 연결한 함수를 정의해주어야 겠죠?

```
from rest_framework import viewsets
from .models import Essay, Album, Files
from .serializers import EssaySerializer, AlbumSerializer, FilesSerializer
from rest_framework.filters import SearchFilter

class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    filter_backends = [SearchFilter]
    search_fields = ('title', 'body')

    def perform_create(self, serializer):
        serializer.save(author=self.request.user) #유저를 자동으로 저장

    def get_queryset(self):
        qs = super().get_queryset()

        if self.request.user.is_authenticated:
            qs = qs.filter(author = self.request.user)
        else:
            qs = qs.none()

        return qs

class ImgViewSet(viewsets.ModelViewSet):
    queryset = Album.objects.all()
    serializer_class = AlbumSerializer


class FileViewSet(viewsets.ModelViewSet):
    queryset = Files.objects.all()
    serializer_class = FilesSerializer
```

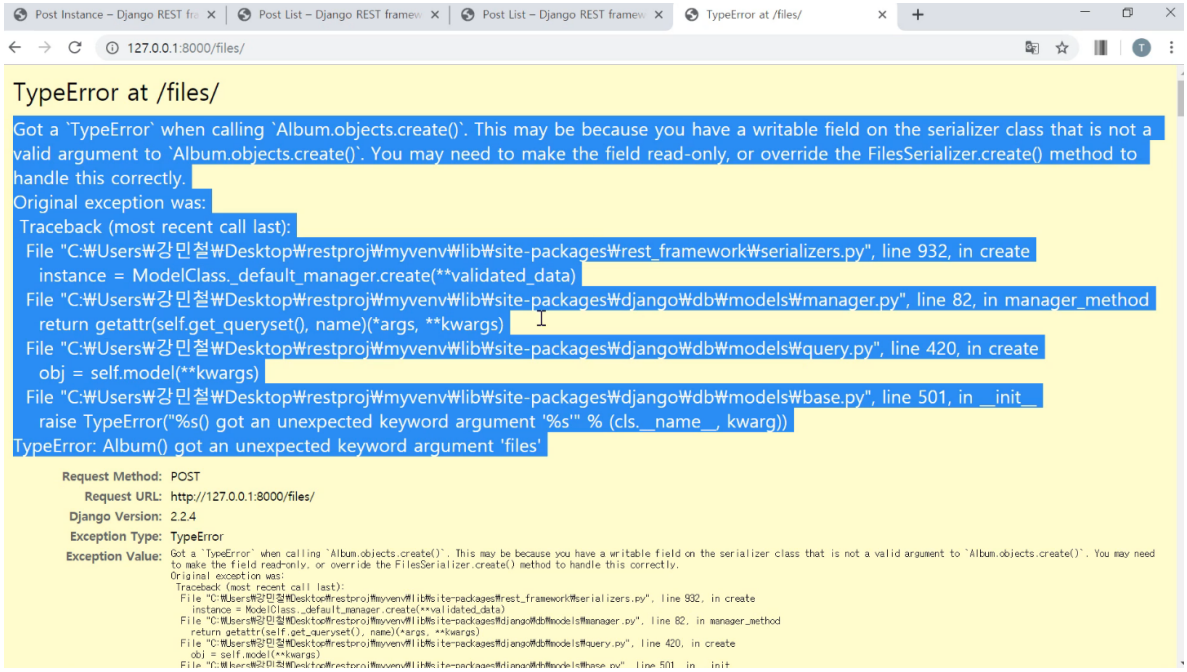
serializers.py

views에서 사용한 serializer를 정의해봅시다.

```
from .models import Essay, Album, Files
from rest_framework import serializers
from rest_framework.parsers import MultiPartParser, FormParser

class EssaySerializer(serializers.ModelSerializer):

    author_name = serializers.ReadOnlyField(source='author.username')

    class Meta:
        model = Essay
        fields = ('pk', 'title', 'body', 'author_name')


class AlbumSerializer(serializers.ModelSerializer):

    author_name = serializers.ReadOnlyField(source='author.username')
    image = serializers.ImageField(use_url=True)

    class Meta:
        model = Album
        fields = ('pk', 'author_name', 'image', 'desc')

class FilesSerializer(serializers.ModelSerializer):

    author_name = serializers.ReadOnlyField(source='author.username')
    myfiles = serializers.FileField(use_url=True)

    class Meta:
        model = Files
        fields = ('pk', 'author_name', 'myfiles', 'desc')
```

> run server로 확인해보고, 이미지와 파일을 올려봅시다.

what the ....



> pdf 파일을 올리는데 문제가 생깁니다. 해결해 봅시다.

Let`s Solve File ERROR

파일 업로드 에러를 해결하기 위해서는 `parser_class`와 `create()`오버라이딩이 필요합니다. 한 번 사용해 봅시다.

views.py

```
from rest_framework import viewsets
from .models import Essay, Album, Files
from .serializers import EssaySerializer, AlbumSerializer, FilesSerializer
from rest_framework.filters import SearchFilter
from rest_framework.parsers import MultiPartParser, FormParser


class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    filter_backends = [SearchFilter]
    search_fields = ('title', 'body')

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)

    def get_queryset(self):
        qs = super().get_queryset()

        if self.request.user.is_authenticated:
            qs = qs.filter(author = self.request.user)
        else:
            qs = qs.none()
```

```
        return qs

class ImgViewSet(viewsets.ModelViewSet):
    queryset = Album.objects.all()
    serializer_class = AlbumSerializer


from rest_framework.response import Response # import 하는 이유는 APIView 강의를 다시 확인해 봅시다.
from rest_framework import status


class FileViewSet(viewsets.ModelViewSet):
    queryset = Files.objects.all()
    serializer_class = FilesSerializer

    parser_classes = (MultiPartParser, FormParser) # 다양한 media파일을 인코딩할 수 있도록 한다.

    # create() -> post()

    def post(self, request, *args, **kwargs): #우리의 입맛대로 post를 새로 정의하자
        serializer = FilesSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=HTTP_201_CREATED)
        else:#유효성 검사를 통과하지 못하면 bad request를 응답한다.
            return Response(serializer.error, status=HTTP_400_BAD_REQUEST)
```

Run server && Test!!!

> 다시 파일과 이미지를 올려보세요! 잘 동작하는 것을 확인할 수 있을 겁니다.

References

classlion project 실습 강의