BackEnd Workshop

Clarusway

Serializers

Task

- Create serializers for:
    - Artist,
    - Album (artists with the names),
    - Song (artist and album with the name).
- Create nested serializers for:
    - Songs with lyrics,
    - Albums with the songs and `number of the songs` in that album.
- Create views for each serializer (CRUD for first three serializer; only list for last two)
- Create urls for each view

Solution

Setup

```
py -m venv env
.\env\Scripts\activate
pip install djangorestframework
pip install pillow
pip freeze > requirements.txt
```

```
django-admin startproject main .
python manage.py startapp song
```

- Update settings.py:

```
INSTALLED_APPS = [
    'song',
    'rest_framework',
]
```

- Add models.py:

```
from django.db import models


class Artist(models.Model):
    first_name = models.CharField(max_length=50, null = False, blank = True)
    last_name = models.CharField(max_length=50, null = True, blank = True)
    artist_pic = models.ImageField(upload_to='artists')
    num_stars = models.IntegerField(default=0, blank = True)

    def __str__(self):
        return self.first_name


class Album(models.Model):
    artist = models.ManyToManyField(Artist)
    name = models.CharField(max_length=100)
    released = models.DateField(null = True, blank=True)
    cover = models.ImageField(upload_to='covers')

    def __str__(self):
        return self.name


class Lyric(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField(null = False, blank=True)

    def __str__(self):
        return self.title


class Song(models.Model):
    album = models.ForeignKey(Album, on_delete=models.CASCADE,
related_name='songs')
    artist = models.ForeignKey(Artist, on_delete=models.CASCADE)
    lyric = models.OneToOneField(Lyric, on_delete=models.CASCADE, null=True)
    name = models.CharField(max_length=100)
```

```
    released = models.DateField(null = True, blank=True)

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

- Register models:

```
from .models import Album, Artist, Song, Lyric

admin.site.register(Album)
admin.site.register(Artist)
admin.site.register(Song)
admin.site.register(Lyric)
```

- Setup urls.py files:
    - Create song/urls.py
    - Add it main/urls.py
- Handle image files:
    - serve image files locally urlpatterns

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

urlpatterns = [
    # ... the rest of your URLconf goes here ...
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

- Add to settings.py:

```
MEDIA_ROOT = BASE_DIR / 'media/'
MEDIA_URL = '/media/'
```

- Apply migrations:

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
python manage.py runserver
```

- Create objects on admin panel.

Serializers:

- Serializers.py:

```
from rest_framework import serializers
from .models import (
    Album,
    Artist,
    Lyric,
    Song,
    )


class ArtistSerializer(serializers.ModelSerializer):
    class Meta:
        model = Artist
        fields = '__all__'


class AlbumSerializer(serializers.ModelSerializer):
    artist = serializers.StringRelatedField(many=True)
    class Meta:
        model = Album
        fields = '__all__'


class SongSerializer(serializers.ModelSerializer):
    album = serializers.StringRelatedField()
    artist = serializers.StringRelatedField()
    class Meta:
        model = Song
        fields = '__all__'


class LyricSerializer(serializers.ModelSerializer):
    song = serializers.StringRelatedField()
    class Meta:
        model = Lyric
        fields = '__all__'


class SongWithLyricsSerializer(serializers.ModelSerializer):
    album = serializers.StringRelatedField()
    artist = serializers.StringRelatedField()
    lyric = LyricSerializer()
    class Meta:
        model = Song
        fields = ['artist', 'album', 'name', 'released', 'lyric']


class AlbumWithSongsSerializer(serializers.ModelSerializer):
    songs = SongSerializer(many=True)
```

```
    num_of_songs = serializers.SerializerMethodField()
    artist = serializers.StringRelatedField(many=True)
    class Meta:
        model = Album
        fields = ('name', 'artist', 'num_of_songs', 'released', 'songs')

    def get_num_of_songs(self, obj):
        return Song.objects.filter(album_id=obj.id).count()
```

Views

- Views.py

```
from django.shortcuts import render
from rest_framework.decorators import api_view
from .models import (
    Album,
    Song,
    Artist,
)
from .serializers import (
    AlbumSerializer,
    SongSerializer,
    ArtistSerializer,
    SongWithLyricsSerializer,
    AlbumWithSongsSerializer,
    )
from rest_framework.generics import (
    ListAPIView,
    ListCreateAPIView,
    RetrieveUpdateDestroyAPIView,
    )


class ArtistListCreateView(ListCreateAPIView):
    queryset = Artist.objects.all()
    serializer_class = ArtistSerializer


class ArtistRUDView(RetrieveUpdateDestroyAPIView):
    queryset = Artist.objects.all()
    serializer_class = ArtistSerializer


class AlbumListCreateView(ListCreateAPIView):
    queryset = Album.objects.all()
    serializer_class = AlbumSerializer


class AlbumRUDView(RetrieveUpdateDestroyAPIView):
    queryset = Album.objects.all()
```

```
    serializer_class = AlbumSerializer


class SongListCreateView(ListCreateAPIView):
    queryset = Song.objects.all()
    serializer_class = SongSerializer


class SongRUDView(RetrieveUpdateDestroyAPIView):
    queryset = Song.objects.all()
    serializer_class = SongSerializer


class SongWithLyricsListView(ListAPIView):
    queryset = Song.objects.all()
    serializer_class = SongWithLyricsSerializer


class AlbumWithSongListView(ListAPIView):
    queryset = Album.objects.all()
    serializer_class = AlbumWithSongsSerializer
```

Urls

- Main.urls.py:

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

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

- Song/urls.py:

```
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static
from .views import (
    ArtistListCreateView,
    ArtistRUDView,
    AlbumListCreateView,
    AlbumRUDView,
    SongListCreateView,
    SongRUDView,
    SongWithLyricsListView,
    AlbumWithSongListView,
)
```

```
urlpatterns = [
    path('artists/', ArtistListCreateView.as_view()),
    path('artists/<int:pk>/', ArtistRUDView.as_view()),
    path('albums/', AlbumListCreateView.as_view()),
    path('albums/<int:pk>/', AlbumRUDView.as_view()),
    path('songs/', SongListCreateView.as_view()),
    path('songs/<int:pk>/', SongRUDView.as_view()),
    path('lyricsofsong/', SongWithLyricsListView.as_view()),
    path('songsofalbums/', AlbumWithSongListView.as_view()),
]  + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

☺ **Thanks for Attending** ✍

Clarusway