**REST API (Django Rest Framework-DRF)**

**Session-22**

**Working with LimitOffsetPagination:**

* This pagination style mirrors the syntax used when looking up multiple database records. The client includes both a "limit" and an "offset" query parameter.
* The limit indicates the maximum number of items to return, and is equivalent to the page\_size in other styles. The offset indicates the starting position of the query in relation to the complete set of unpaginated items.
* To set this globally write the following code in settings.py

REST\_FRAMEWORK={

'DEFAULT\_PAGINATION\_CLASS':'rest\_framework.pagination.LimitOffsetPagination'

}

**Settings.py :**

INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'myapp.apps.MyappConfig',  
 'rest\_framework',  
 'paginationapp',  
]

**models.py :**

from django.db import models  
  
*# Create your models here.*class Student(models.Model):  
 name=models.CharField(max\_length=20)  
 address=models.CharField(max\_length=20)  
 age=models.IntegerField()

**admin.py :**

from django.contrib import admin  
from paginationapp.models import Student  
*# Register your models here.*@admin.register(Student)  
class StudentAdmin(admin.ModelAdmin):  
 list\_display = ['id','name','address','age']

**Now go to terminal and then type the following commands**

* Python manage.py makemigrations
* Python manage.py migrate
* Python manage.py createsuperuser
* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin>
* Insert few records
* Create a new python file with the name serializers.py.

**serializer.py:**

from rest\_framework import serializers  
from paginationapp.models import Student  
  
class StudentSerializer(serializers.ModelSerializer):  
 class Meta:  
 model=Student  
 fields=['id','name','address','age']

**views.py :**

from django.shortcuts import render  
from paginationapp.serializers import StudentSerializer  
from paginationapp.models import Student  
from rest\_framework.generics import ListAPIView  
  
  
*# Create your views here.*class StudentList(ListAPIView):  
 queryset = Student.objects.all()  
 serializer\_class = StudentSerializer

**urls.py:**

from django.contrib import admin  
from django.urls import path  
from paginationapp import views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('studentapi/',views.StudentList.as\_view()),  
]

* Now run server: python manage.py runserver
* Now go to browser: http://127.0.0.1:8000/admin/studentapi/

**Pagination.py:**

from rest\_framework.pagination import LimitOffsetPagination  
  
class MyPagination(LimitOffsetPagination):  
 default\_limit = 5  
 limit\_query\_param = 'pagelimit'  
 offset\_query\_param = 'pageoffset'  
 max\_limit = 4

**views.py :**

from django.shortcuts import render  
from paginationapp.serializers import StudentSerializer  
from paginationapp.models import Student  
from rest\_framework.generics import ListAPIView  
from paginationapp.pagination import MyPagination  
  
*# Create your views here.*class StudentList(ListAPIView):  
 queryset = Student.objects.all()  
 serializer\_class = StudentSerializer  
 pagination\_class = MyPagination

* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin/studentapi/>
* <http://127.0.0.1:8000/studentapi/?limit=4&offset=7>

**Working with CursorPagination:**

* This pagination style only presents forward and reverse controls, and does not allow the client to navigate to arbitrary positions.
* Cursor based pagination requires that there is a unique, unchanging ordering of items in the result set.
* This ordering might typically be a creation timestamp on the records, as this presents a consistent ordering to paginate against.
* Cursor based pagination is more complex than other schemes. It also requires that the result set presents a fixed ordering, and does not allow the client to arbitrarily index into the result set. However it does provide the following benefits:
* Provides a consistent pagination view. When used properly CursorPagination ensures that the client will never see the same item twice when paging through records, even when new items are being inserted by other clients during the pagination process.

**Pagination.py:**

from rest\_framework.pagination import CursorPagination  
  
class MyPagination(CursorPagination):  
 page\_size = 3  
 ordering = 'name'  
 cursor\_query\_param = 'c'

**Working with Hyperlink Model Serializer:**

* The HyperlinkedModelSerializer class is similar to the ModelSerializer class except that it uses hyperlinks to represent relationships, rather than primary keys.
* By default the serializer will include a url field instead of a primary key field.
* The url field will be represented using a HyperlinkedIdentityField serializer field, and any relationships on the model will be represented using a HyperlinkedRelatedField serializer field.

**Models.py:**

from django.db import models  
  
*# Create your models here.*class Student(models.Model):  
 name=models.CharField(max\_length=20)  
 sid=models.IntegerField()  
 saddress=models.CharField(max\_length=20)

**admin.py:**

from django.contrib import admin  
from myapp1.models import Student  
*# Register your models here.*@admin.register(Student)  
class StudentAdmin(admin.ModelAdmin):  
 list\_display = ['id','name','sid','saddress']

**serializers.py:**

from rest\_framework import serializers  
from myapp1.models import Student  
  
class StudentSerializer(serializers.HyperlinkedModelSerializer):  
 class Meta:  
 model=Student  
 fields=['id','url','name','sid','saddress']

**views.py:**

from django.shortcuts import render  
from myapp1.serializers import StudentSerializer  
from myapp1.models import Student  
from rest\_framework import viewsets  
  
*# Create your views here.*class StudentModelViewset(viewsets.ModelViewSet):  
 queryset = Student.objects.all()  
 serializer\_class = StudentSerializer

**urls.py:**

from django.contrib import admin  
from django.urls import path,include  
from myapp1 import views  
from rest\_framework.routers import DefaultRouter  
  
  
router=DefaultRouter()  
  
router.register('studentapi',views.StudentModelViewset,basename='student')  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('',include(router.urls)),  
]

* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin/studentapi/>