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

**Session-19**

**Filtering in DRF:**

* The default behavior of REST framework's generic list views is to return the entire queryset for a model manager. Often you will want your API to restrict the items that are returned by the queryset.
* The simplest way to filter the queryset of any view that subclasses GenericAPIView is to override the get\_queryset() method.
* Overriding this method allows you to customize the queryset returned by the view in a number of different ways.

**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',  
]

**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)  
 trainedby=models.CharField(max\_length=20)

**admin.py :**

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

**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 and add few users with the name user1 and user2 with staff status

Create a new python file with the name serializers.py.

**serializer.py:**

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

**views.py :**

from myapp.serializers import StudentSerializer  
from rest\_framework.generics import ListAPIView  
from myapp.models import Student

class StudentList(ListAPIView):  
 queryset = Student.objects.all()  
 *#queryset = Student.objects.filter(trainedby="user1")* serializer\_class = StudentSerializer

**urls.py:**

from django.contrib import admin  
from django.urls import path  
from myapp 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/>

**Filtering against the current user:**

* You might want to filter the queryset to ensure that only results relevant to the currently authenticated user making the request are returned.
* You can do so by filtering based on the value of request.user

**Views.py:**

from django.shortcuts import render  
from myapp.serializers import StudentSerializer  
from rest\_framework.generics import ListAPIView  
from myapp.models import Student  
  
*# Create your views here.*class StudentList(ListAPIView):  
 queryset = Student.objects.all()serializer\_class = StudentSerializer  
  
 def get\_queryset(self):  
 user=self.request.user  
 return Student.objects.filter(trainedby=user)

**Generic Filtering:**

* As well as being able to override the default queryset, REST framework also includes support for generic filtering backends that allow you to easily construct complex searches and filters.
* Generic filters can also present themselves as HTML controls in the browsable API and admin API.

**DjangoFilterBackend:**

* The django-filter library includes a DjangoFilterBackend class which supports highly customizable field filtering for REST framework.
* To use DjangoFilterBackend, first install django-filter.
* **pip install django-filter**
* Then add 'django\_filters' to Django's INSTALLED\_APPS:
* INSTALLED\_APPS = [

...

'django\_filters',

...

]

* You should now either add the filter backend to your settings:

REST\_FRAMEWORK = {

'DEFAULT\_FILTER\_BACKENDS': ['django\_filters.rest\_framework.DjangoFilterBackend']

}

* Or add the filter backend to an individual View or ViewSet.

**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',  
 'django\_filters',  
]

REST\_FRAMEWORK={  
 'DEFAULT\_FILTER\_BACKENDS':['django\_filters.rest\_framework.DjangoFilterBackend']  
}

**Views.py:**

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

* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin/studentapi/>?saddress=hyderabad
* **Now you can remove default filter backends from settings.py file and try below code .**

**Views.py:**

from myapp.serializers import StudentSerializer  
from rest\_framework.generics import ListAPIView  
from myapp.models import Student  
from django\_filters.rest\_framework import DjangoFilterBackend  
  
*# Create your views here.*class StudentList(ListAPIView):  
 queryset = Student.objects.all()  
 serializer\_class = StudentSerializer  
 filter\_backends = [DjangoFilterBackend]  
 filterset\_fields=['saddress','trainedby']

* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin/studentapi/>?saddress=hyderabad&trainedby=user1