Topic: 15. DRF Filtering

Speaker: Personal | Notebook: API Development using Django Framework



Django Filtering:

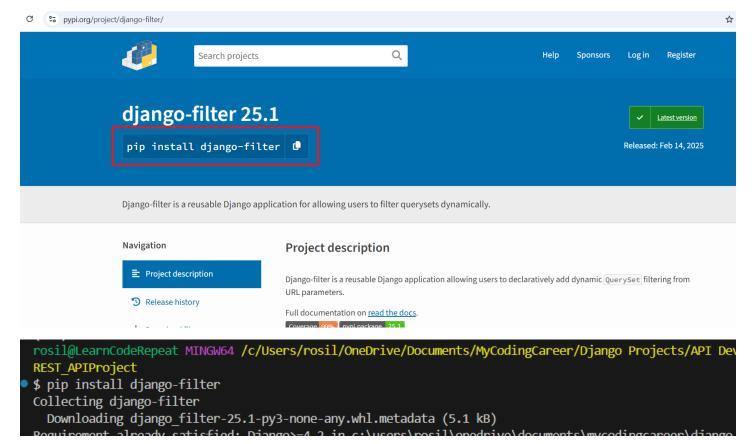
1. Filtering allows searching for certain record(s) based on given criteria. According to Django Rest API Framework documentation:

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.

2. Install the library for Django Filter, go here. Go for the the latest version.



3. Register your Django package in SETTINGS.PY

```
×
         Edit
                                                paginations.py
                                                                    models.py blogs
        EXPLORER
                               🕏 settings.py 🔍
                                                                                          admin.py blogs
C<sub>2</sub>

✓ DJANGOREST APIPROJECT

                               django_rest_main > 🕏 settings.py > ...

√ api

                                       INSTALLED APPS = [
                                           'django.contrib.admin',
        paginations.py
                                           'django.contrib.auth',
        serializers.py
                                           'django.contrib.contenttypes',
        tests.py
                                           'django.contrib.sessions',
        urls.py
                                           'django.contrib.messages',
        views.py
                                           'django.contrib.staticfiles',

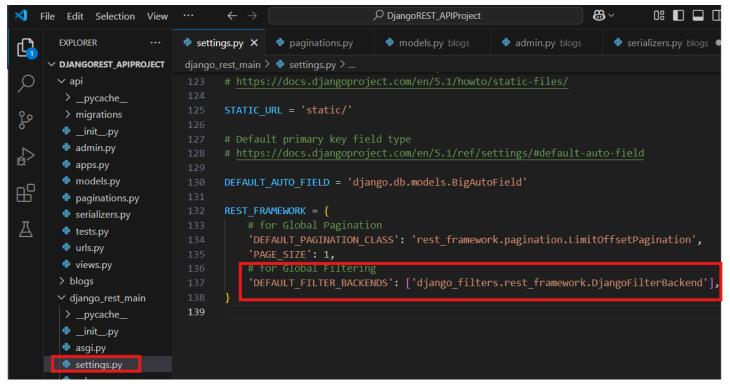
✓ blogs

                                           'rest framework',
                                           'students',
         > _pycache_
                                 41
                                           'employees',
         > migrations
                                           'api',
        __init__.py
                                           'blogs',
        admin.py
                                          'django_filters',
        apps.py
        models.py
        serializers.py
                                      MIDDLEWARE = [
        tests.py
                                           'django.middleware.security.SecurityMiddleware',
        views.py
                                           'django.contrib.sessions.middleware.SessionMiddleware',

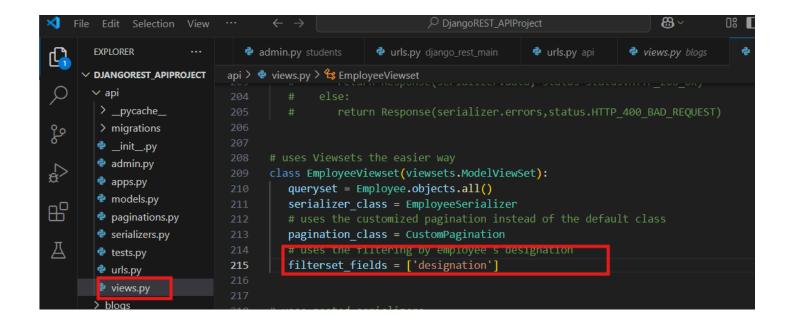
✓ django_rest_main

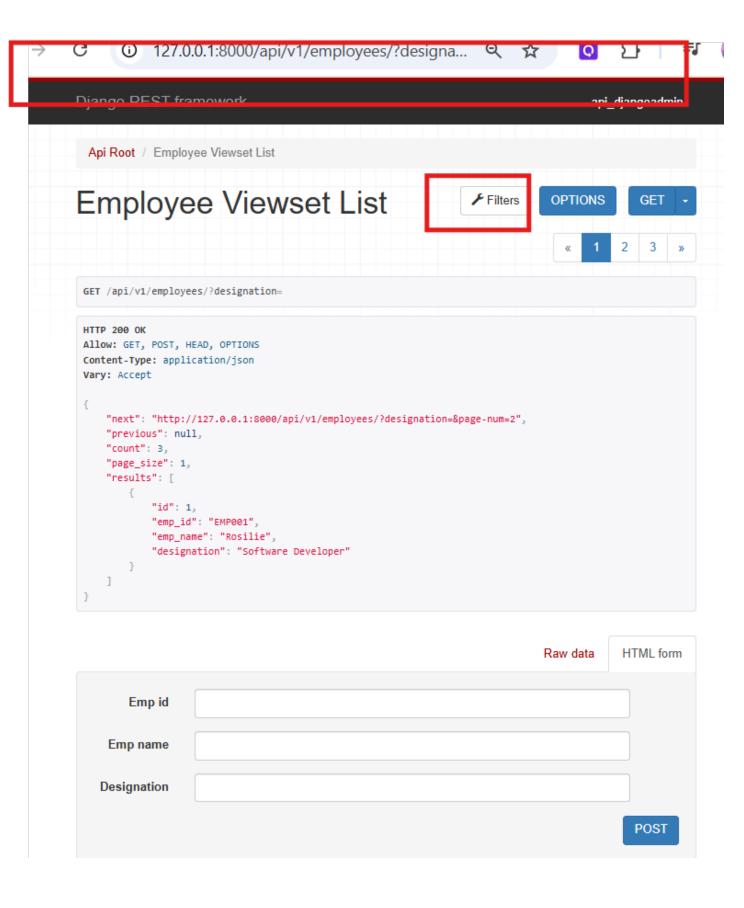
                                           'django.middleware.common.CommonMiddleware',
         _pycache_
                                           'django.middleware.csrf.CsrfViewMiddleware',
        🕏 __init__.py
                                           'django.contrib.auth.middleware.AuthenticationMiddleware',
        🕏 asgi.py
                                           'django.contrib.messages.middleware.MessageMiddleware',
                                           'django.middleware.clickjacking.XFrameOptionsMiddleware',
        settings.py
        urls.py
```

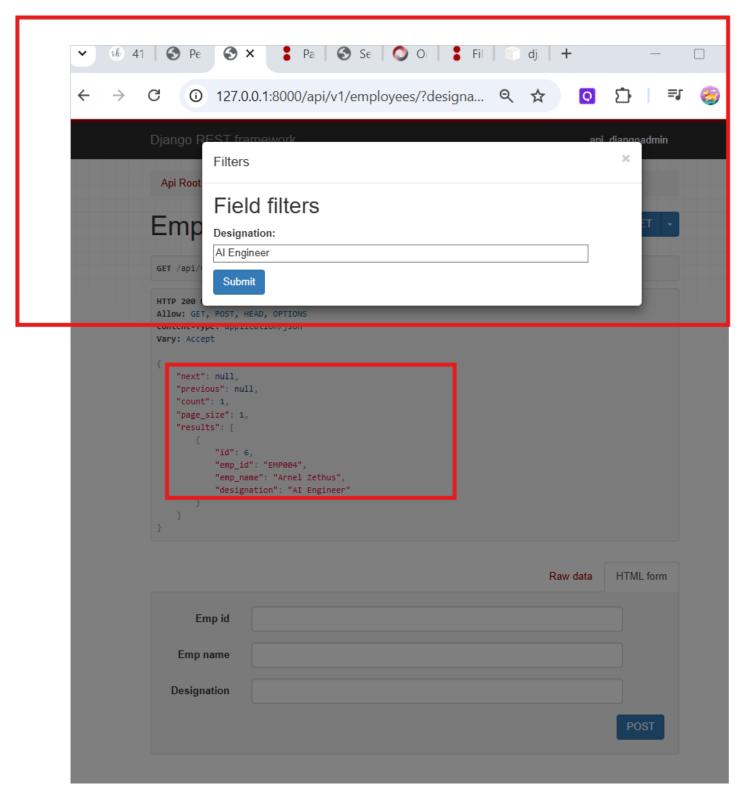
4. To set the GLOBAL FILTER, do this just like how you did the GLOBAL PAGINATION.



5. Update our API\VIEWS.PY:







- * You need to type the exact keyword to find an exact match, so this is a case-sensitive filter. To solve this, we use custom filters instead.
- 6. To allow for case-insensitive filter, create a new file FILTERS.PY in the EMPLOYEES app or you can have this in API folder.

```
88 \
    File Edit Selection
                                                                       DjangoREST_APIProject
                       View
                             Go
                                                                                                                          de filters.py
       EXPLORER
                                        diango_rest_main
                                                                                                        views.py api

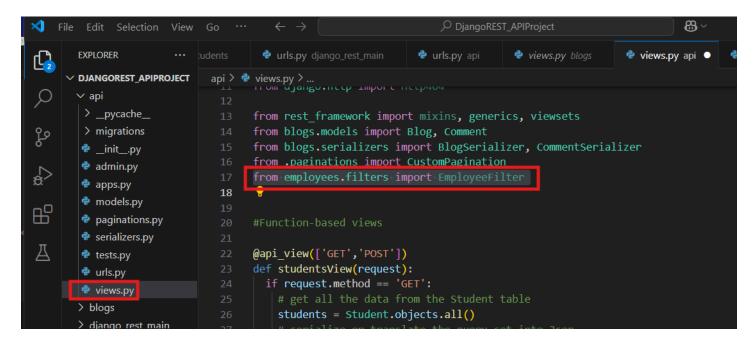
✓ DJANGOREST_APIPROJECT

                               employees > 🕏 filters.py > ...
                                      import django_filters
       > api
                                      from .models import Employee
       > blogs
       > django_rest_main
وړ
                                      class EmployeeFilter(django_filters.FilterSet):

✓ employees

                                          designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
        > _pycache_
        > migrations
                                               model = Employee
        __init__.py
ß
                                               fields = ['designation']
        admin.py
                                 10
        apps.py
       🕏 filters.py
        models.py
        tests.py
        views.py
```

7. Then update our VIEWS.PY. Import the class EmployeeFilter from the Employees\Filters.py



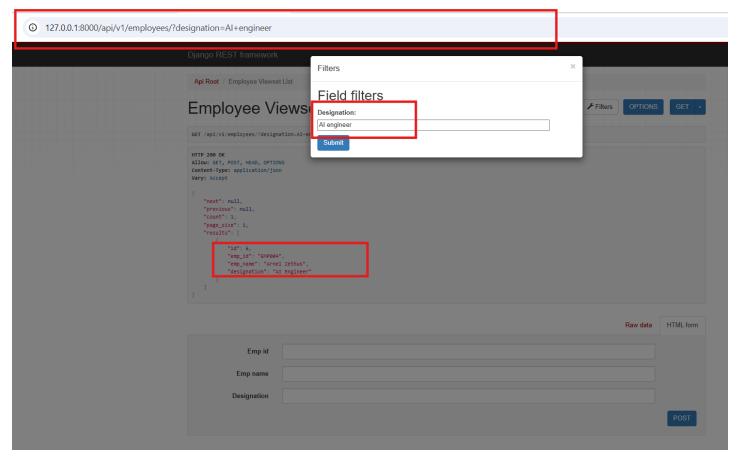
To use the new filter class:

```
83
       EXPLORER
                                        urls.py django_rest_main
                                                                   urls.py api
                                                                                    views.py blogs
                                                                                                        views.py api
C)
      ∨ DJAN... 🖺 🛱 ひ 🗗
                               api > 🕏 views.py > ...

✓ api

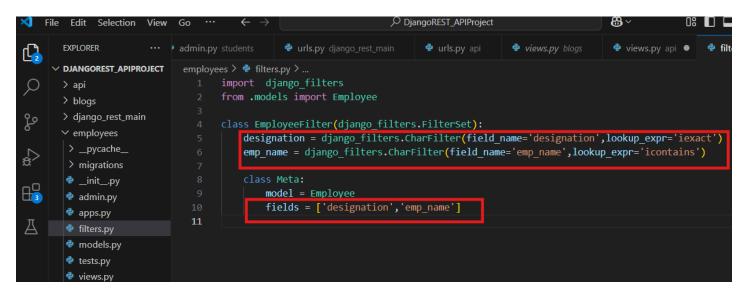
        > _pycache_
                                      class EmployeeViewset(viewsets.ModelViewSet):
        > migrations
                                         queryset = Employee.objects.all()
        🕏 __init__.py
                                         serializer_class = EmployeeSerializer
        admin.py
        apps.py
                                         pagination_class = CustomPagination
        models.py
船
        paginations.py
        serializers.py
Д
        🕏 tests.py
        🕏 urls.py
                                         filterset_class = EmployeeFilter
        🕏 views.py
         blogs
                                      # uses nested serializers
                               224
       > django_rest_main
                                            BlogsView(generics.ListCreateAPTView
```

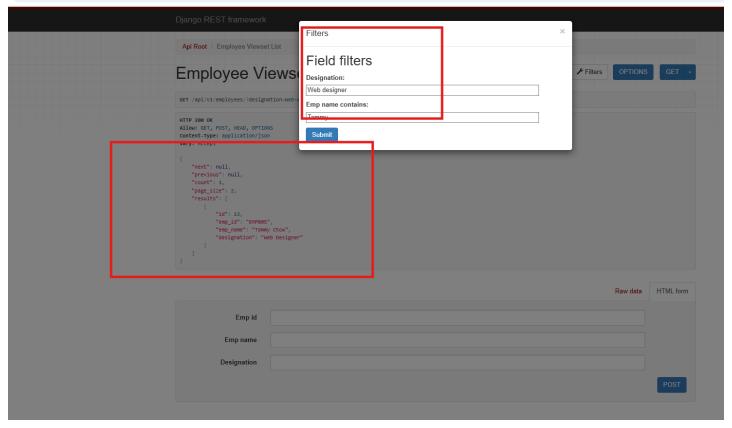
8. Now even if we search without minding the lowercase or uppercase, we can still find the record:

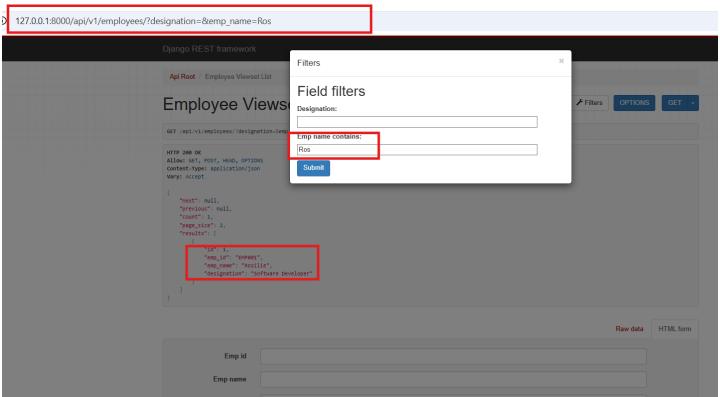


9. To add filters like by name and ID, we update our FILTERS.PY:

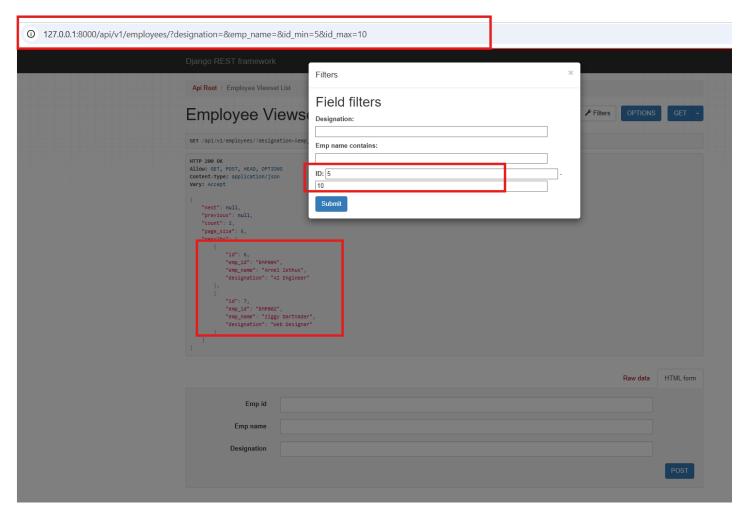
If we add the 2 filter keywords, we search the record for these 2 criteria.



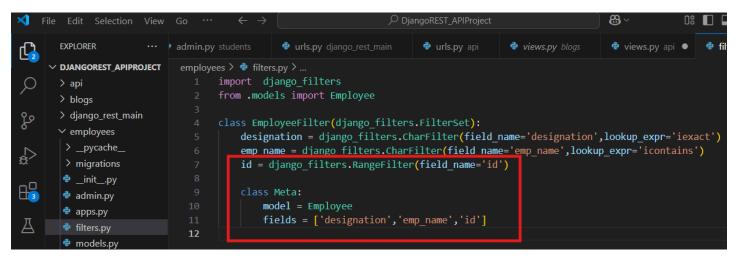




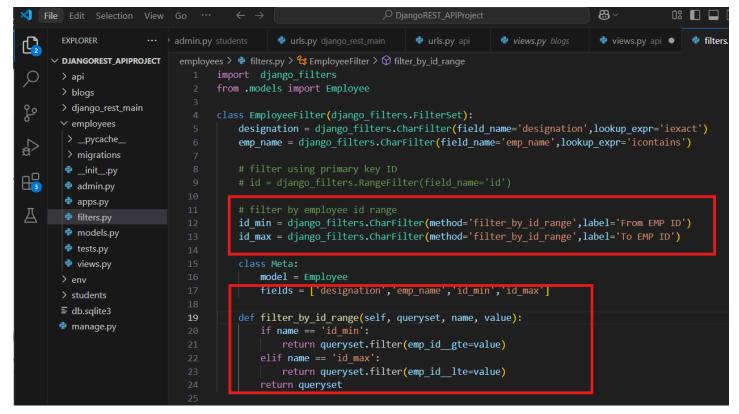
10. To retrieve the employee records within the given ID range for example: PK or ID 5 - 10: $\frac{10}{10}$



We update our FILTERS.PY as:



11. But to use the EMP_ID (ex. EMP005-EMP008) with CharField as our range filter, we update the FILTERS.PY as:



Before the Filter by Emp_ID:

Django REST framework

Api Root / Employee Viewset List

Employee Viewset List



GET /api/v1/employees/

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
    "next": "http://127.0.0.1:8000/api/v1/employees/?page-num=2",
    "previous": null,
    "count": 6,
    "page_size": 5,
    "results": [
           "id": 1,
           "emp_id": "EMP001",
           "emp_name": "Rosilie",
            "designation": "Software Developer"
           "id": 6,
            "emp_id": "EMP004",
            "emp_name": "Arnel Zethus",
            "designation": "AI Engineer"
           "id": 7,
            "emp_id": "EMP005",
            "emp_name": "Ziggy DartVader",
            "designation": "Web Designer"
           "id": 11,
           "emp_id": "EMP008",
           "emp_name": "Dylan",
            "designation": "Network Engineer"
            "id": 12,
            "emp_id": "EMP009",
            "emp_name": "Lexine",
            "designation": "Graphics Designer"
```







