

## 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.

pypi.org/project/django-filter/

Search projects

Help Sponsors Log in Register

# django-filter 25.1

Latest version

Released: Feb 14, 2025

`pip install django-filter`

Django-filter is a reusable Django application for allowing users to filter querysets dynamically.

### Navigation

- Project description
- Release history

### Project description

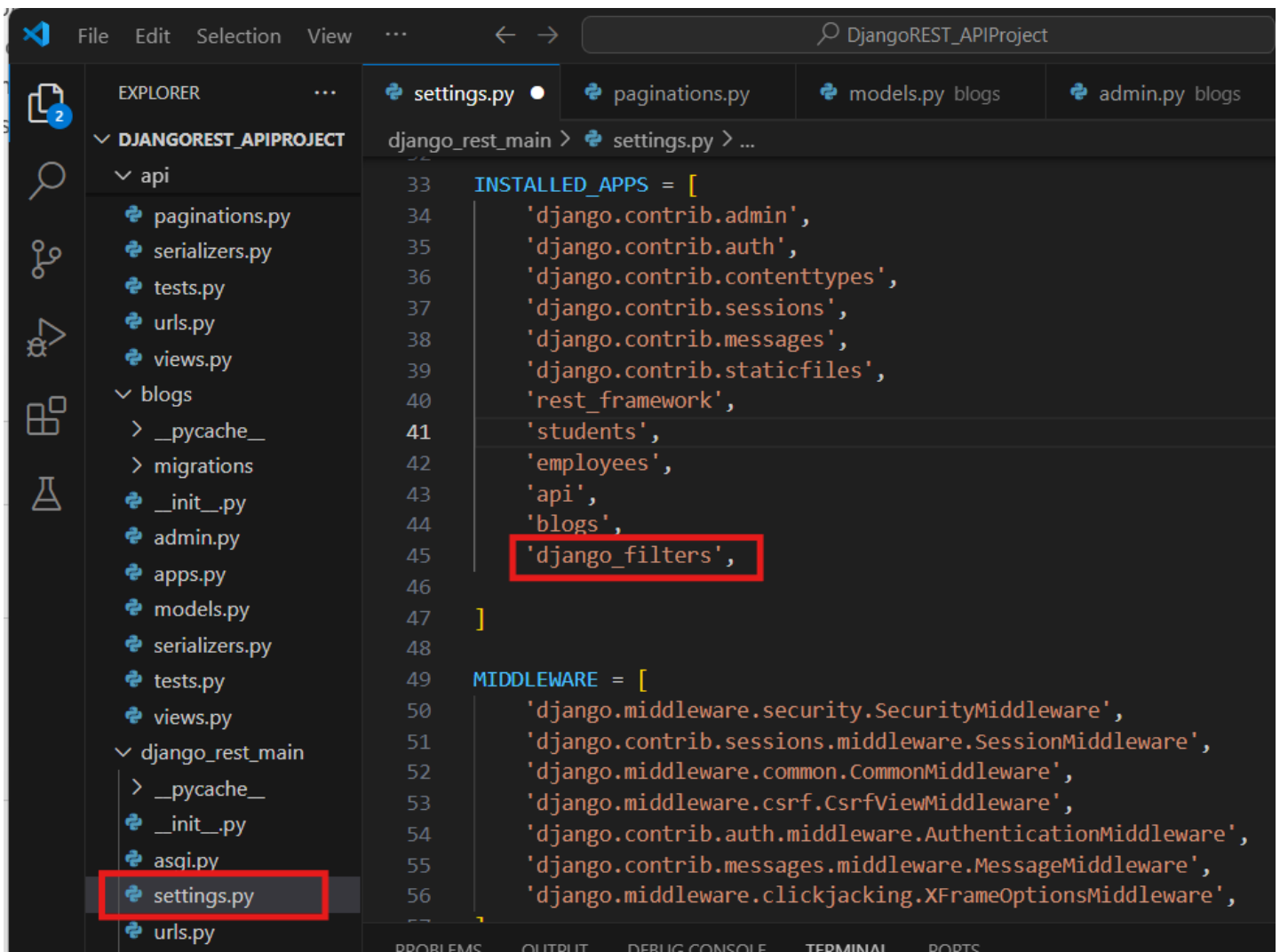
Django-filter is a reusable Django application allowing users to declaratively add dynamic `QuerySet` filtering from URL parameters.

Full documentation on [read the docs](#).

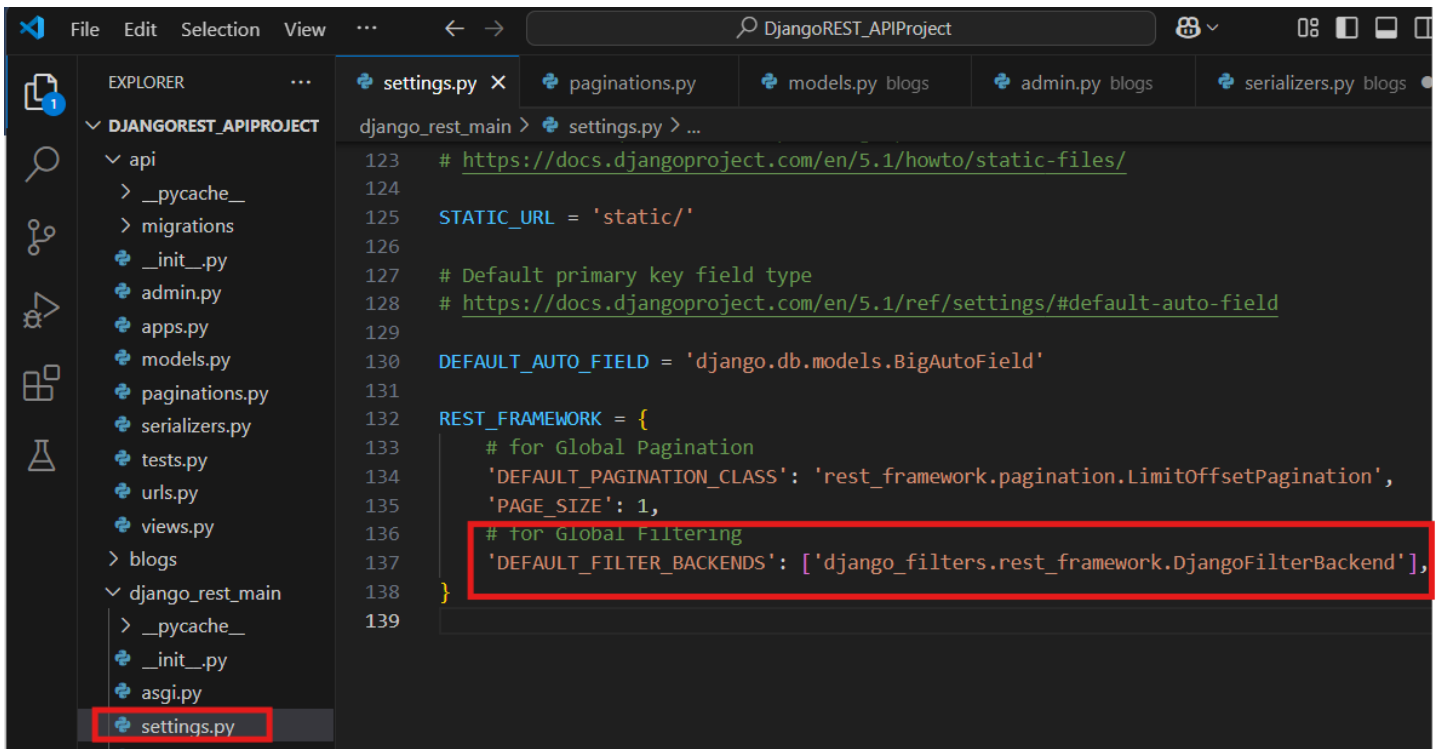
Coverage: 99% | PyPI package: 25.1

```
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API Dev
REST_APIProject
$ pip install django-filter
Collecting django-filter
  Downloading django_filter-25.1-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: Django>=4.2 in c:\users\rosil\onedrive\documents\mycodingcareer\django
```

3. Register your Django package in SETTINGS.PY



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



5. Update our APIVIEWS.PY:

The image shows a code editor interface for a Django REST API project. The Explorer panel on the left shows the project structure, with the `views.py` file in the `api` directory selected. The main editor displays the `EmployeeViewSet` class in `views.py`. The class is a `ModelViewSet` that uses `EmployeeSerializer` and `CustomPagination`. The `filterset_fields` attribute is set to `['designation']`, which is highlighted with a red box. The code is as follows:

```
203 # return response(serializer.data, status=status.HTTP_200_OK)
204 # else:
205 #     return Response(serializer.errors, status.HTTP_400_BAD_REQUEST)
206
207
208 # uses Viewsets the easier way
209 class EmployeeViewSet(viewsets.ModelViewSet):
210     queryset = Employee.objects.all()
211     serializer_class = EmployeeSerializer
212     # uses the customized pagination instead of the default class
213     pagination_class = CustomPagination
214     # uses the filtering by employee's designation
215     filterset_fields = ['designation']
216
217
218 # return response(serializer.data, status=status.HTTP_200_OK)
```

Api Root / Employee Viewset List

## Employee Viewset List

Filters

OPTIONS

GET

«

1

2

3

»

GET /api/v1/employees/?designation=

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "next": "http://127.0.0.1:8000/api/v1/employees/?designation=&page-num=2",
  "previous": null,
  "count": 3,
  "page_size": 1,
  "results": [
    {
      "id": 1,
      "emp_id": "EMP001",
      "emp_name": "Rosilie",
      "designation": "Software Developer"
    }
  ]
}
```

Raw data

HTML form

Emp id

Emp name

Designation

POST

Filters

Field filters

Designation:

AI Engineer

Submit

```
{
  "next": null,
  "previous": null,
  "count": 1,
  "page_size": 1,
  "results": [
    {
      "id": 6,
      "emp_id": "EMP004",
      "emp_name": "Arnel Zethus",
      "designation": "AI Engineer"
    }
  ]
}
```

Raw data HTML form

Emp id

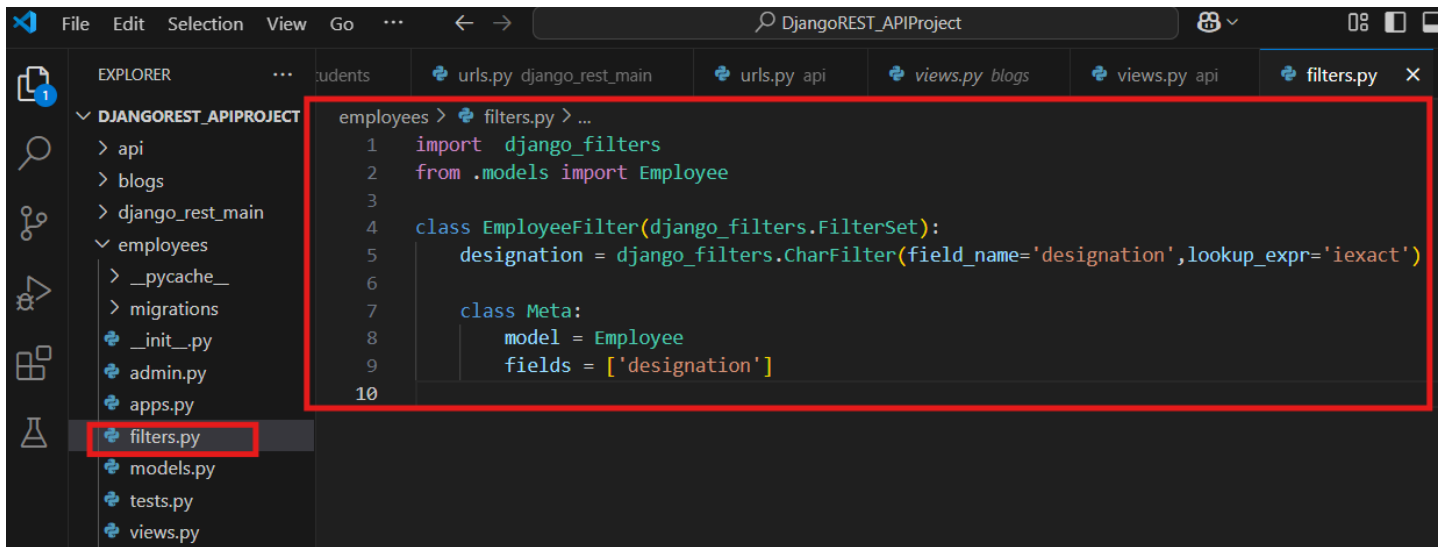
Emp name

Designation

POST

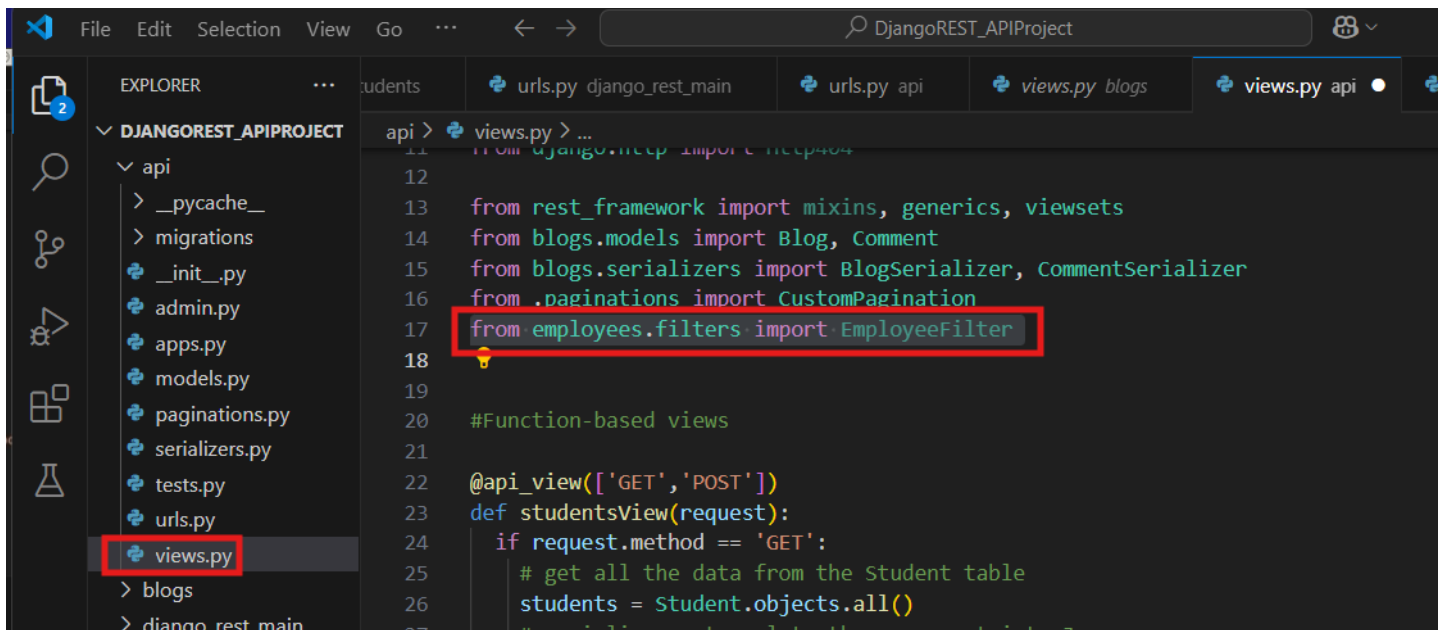
\* You need to type the exact keyword to find an exact match, so this is a case-sensitive filter. To solve this, we use custom filters instead.

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.



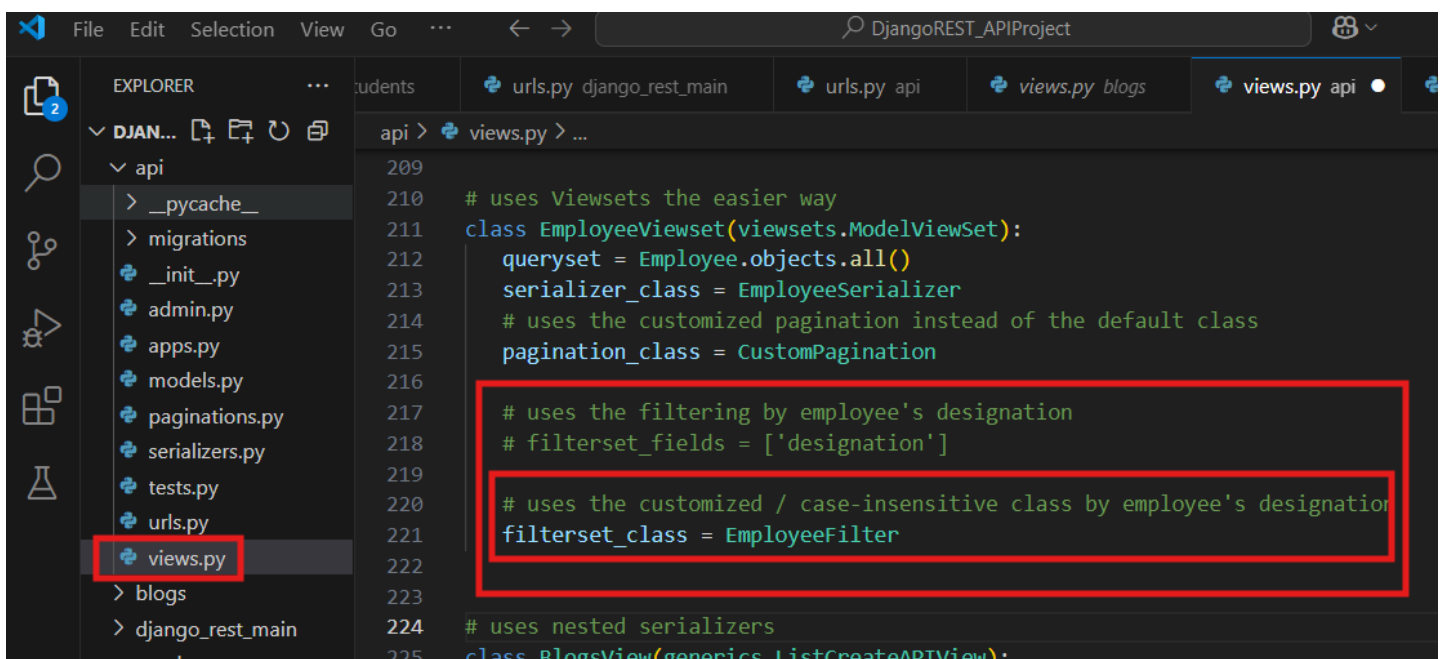
```
employees > filters.py > ...
1  import django_filters
2  from .models import Employee
3
4  class EmployeeFilter(django_filters.FilterSet):
5      designation = django_filters.CharFilter(field_name='designation', lookup_expr='iexact')
6
7      class Meta:
8          model = Employee
9          fields = ['designation']
10
```

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



```
api > views.py > ...
11  from django.http import Http404
12
13  from rest_framework import mixins, generics, viewsets
14  from blogs.models import Blog, Comment
15  from blogs.serializers import BlogSerializer, CommentSerializer
16  from .paginations import CustomPagination
17  from employees.filters import EmployeeFilter
18
19
20  #Function-based views
21
22  @api_view(['GET', 'POST'])
23  def studentsView(request):
24      if request.method == 'GET':
25          # get all the data from the Student table
26          students = Student.objects.all()
27
```

To use the new filter class:



```
api > views.py > ...
209
210  # uses Viewsets the easier way
211  class EmployeeViewSet(viewsets.ModelViewSet):
212      queryset = Employee.objects.all()
213      serializer_class = EmployeeSerializer
214      # uses the customized pagination instead of the default class
215      pagination_class = CustomPagination
216
217      # uses the filtering by employee's designation
218      # filterset_fields = ['designation']
219
220      # uses the customized / case-insensitive class by employee's designation
221      filterset_class = EmployeeFilter
222
223
224  # uses nested serializers
225  class BlogsView(generics.ListCreateAPIView):
```

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

The screenshot shows a web browser with the URL `127.0.0.1:8000/api/v1/employees/?designation=AI+engineer` in the address bar. The page displays the 'Employee Views' section of a Django REST framework API. A 'Filters' modal is open, showing a 'Field filters' section with a 'Designation:' field containing 'AI engineer' and a 'Submit' button. The main content area shows the HTTP response for the GET request, which is a JSON object containing employee data. A specific employee record is highlighted with a red box:

```
{
  "id": 6,
  "emp_id": "EMP004",
  "emp_name": "Arnel Zethus",
  "designation": "AI Engineer"
}
```

Below the JSON response, there are input fields for 'Emp id', 'Emp name', and 'Designation', and a 'POST' button.

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.

The screenshot shows a code editor with the file `filters.py` open. The code defines a custom filter class `EmployeeFilter` that inherits from `django_filters.FilterSet`. The `designations` and `emp_name` filters are defined using `CharFilter` with `lookup_expr='iexact'` and `lookup_expr='icontains'` respectively. The `Meta` class specifies the `model = Employee` and the `fields = ['designation', 'emp_name']`.

```
1 import django_filters
2 from .models import Employee
3
4 class EmployeeFilter(django_filters.FilterSet):
5     designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
6     emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')
7
8     class Meta:
9         model = Employee
10        fields = ['designation', 'emp_name']
11
```

127.0.0.1:8000/api/v1/employees/?designation=Web+designer&emp\_name=Tammy

Django REST framework

Api Root / Employee Viewset List

## Employee Views

GET /api/v1/employees/?designation=Web+designer&emp\_name=Tammy

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "next": null,
  "previous": null,
  "count": 1,
  "page_size": 2,
  "results": [
    {
      "id": 13,
      "emp_id": "EMP005",
      "emp_name": "Tammy Chow",
      "designation": "Web Designer"
    }
  ]
}
```

Filters

Field filters

Designation: Web designer

Emp name contains: Tammy

Submit

Raw data HTML form

Emp id

Emp name

Designation

POST

127.0.0.1:8000/api/v1/employees/?designation=&emp\_name=Ros

Django REST framework

Api Root / Employee Viewset List

## Employee Views

GET /api/v1/employees/?designation=&emp\_name=Ros

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "next": null,
  "previous": null,
  "count": 1,
  "page_size": 2,
  "results": [
    {
      "id": 1,
      "emp_id": "EMP001",
      "emp_name": "Roslie",
      "designation": "Software Developer"
    }
  ]
}
```

Filters

Field filters

Designation:

Emp name contains: Ros

Submit

Raw data HTML form

Emp id

Emp name

10. To retrieve the employee records within the given ID range for example: PK or ID 5 - 10:



127.0.0.1:8000/api/v1/employees/?designation=&emp\_name=&id\_min=5&id\_max=10

Django REST framework

Api Root / Employee Viewset List

## Employee Views

GET /api/v1/employees/?designation=&emp\_name=&id\_min=5&id\_max=10

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "next": null,
  "previous": null,
  "count": 2,
  "page_size": 5,
  "results": [
    {
      "id": 6,
      "emp_id": "EMP004",
      "emp_name": "Arnel Zethus",
      "designation": "AI Engineer"
    },
    {
      "id": 7,
      "emp_id": "EMP002",
      "emp_name": "Ziggy Dartvader",
      "designation": "Web Designer"
    }
  ]
}
```

Filters

Field filters

Designation:

Emp name contains:

ID: 5

10

Submit

Raw data HTML form

Emp id

Emp name

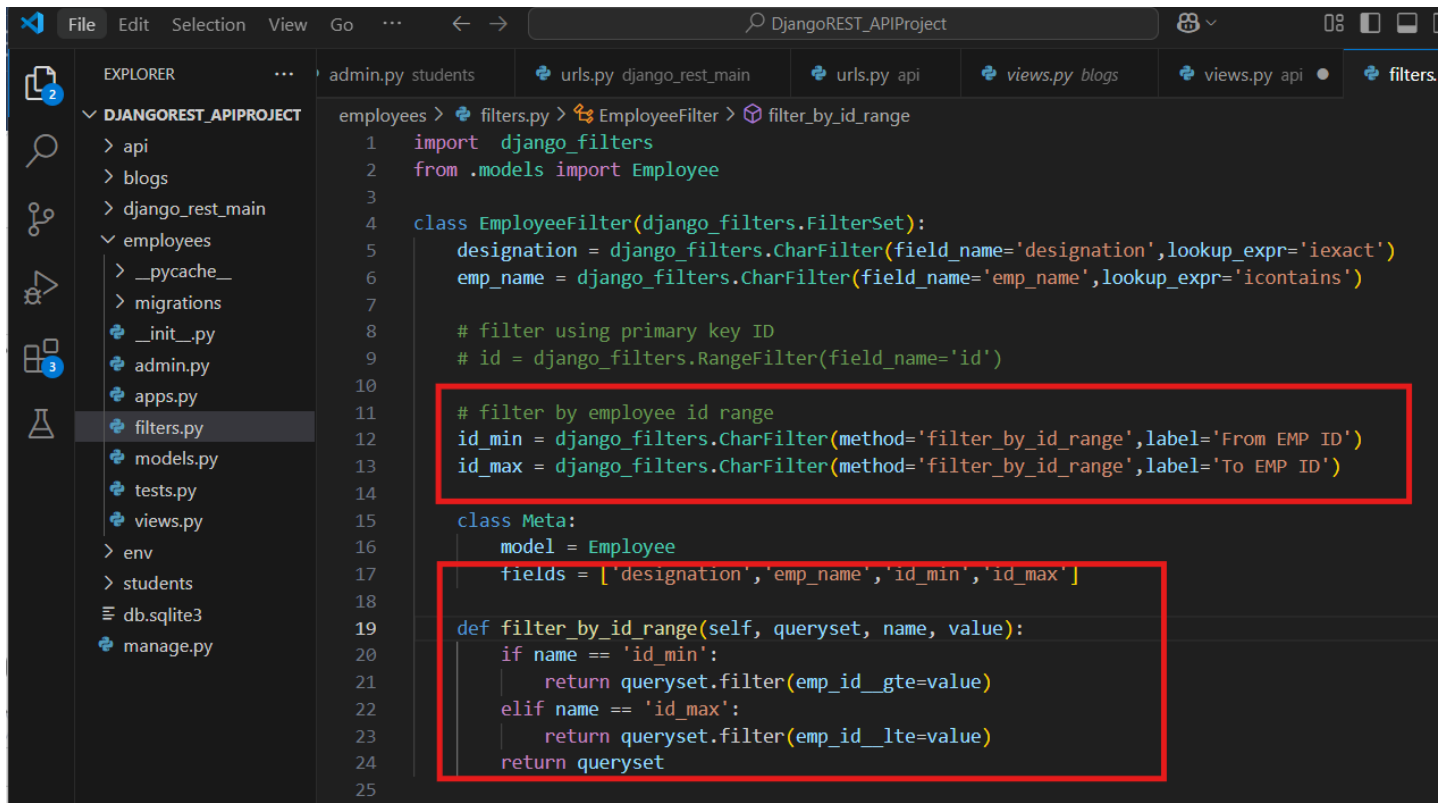
Designation

POST

We update our FILTERS.PY as:

```
1 import django_filters
2 from .models import Employee
3
4 class EmployeeFilter(django_filters.FilterSet):
5     designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
6     emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')
7     id = django_filters.RangeFilter(field_name='id')
8
9     class Meta:
10         model = Employee
11         fields = ['designation','emp_name','id']
12
```

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



```
1 import django_filters
2 from .models import Employee
3
4 class EmployeeFilter(django_filters.FilterSet):
5     designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
6     emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')
7
8     # filter using primary key ID
9     # id = django_filters.RangeFilter(field_name='id')
10
11     # filter by employee id range
12     id_min = django_filters.CharFilter(method='filter_by_id_range',label='From EMP ID')
13     id_max = django_filters.CharFilter(method='filter_by_id_range',label='To EMP ID')
14
15     class Meta:
16         model = Employee
17         fields = ['designation','emp_name','id_min','id_max']
18
19     def filter_by_id_range(self, queryset, name, value):
20         if name == 'id_min':
21             return queryset.filter(emp_id__gte=value)
22         elif name == 'id_max':
23             return queryset.filter(emp_id__lte=value)
24         return queryset
25
```

Before the Filter by Emp\_ID:

# Employee Viewset List

Filters

OPTIONS

GET

« 1 2 »

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"
    }
  ]
}
```

After:

Django REST framework

Api Root

Emp

GET /api/

HTTP 200

Allow: GET

Content-Type

Vary: Accept

{

"next"

"prev"

"count"

"page"

"result"

}

Filters

Field filters

Designation:

Emp name contains:

From EMP ID:

EMP003

To EMP ID:

EMP009

Submit

{

"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"

}

}