Topic: 15. DRF Filtering

Speaker: Personal | Notebook: API Development using Django Framework

---

Django Filtering:

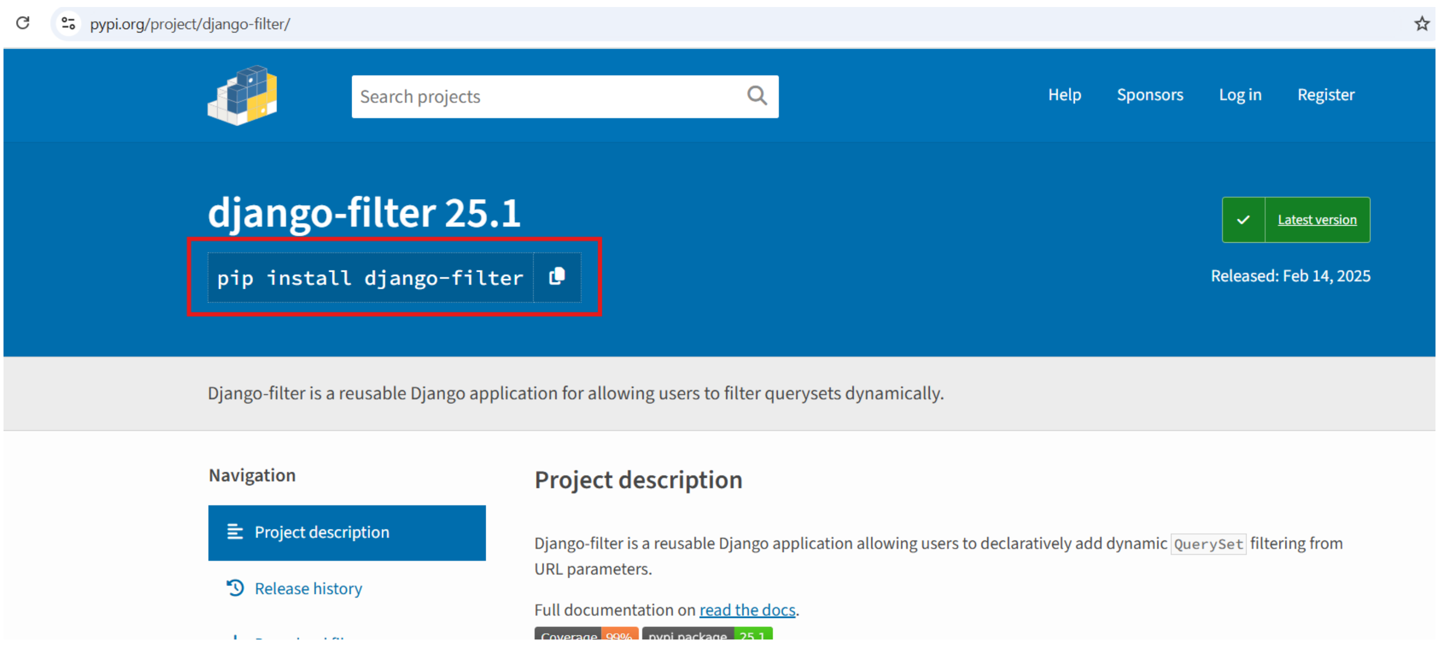
1. Filtering allows searching for certain record(s) based on given criteria. According to [Django Rest API Framework documentation:]()

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

```
pip install django-filter
```

Latest version

Released: Feb 14, 2025

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

Navigation

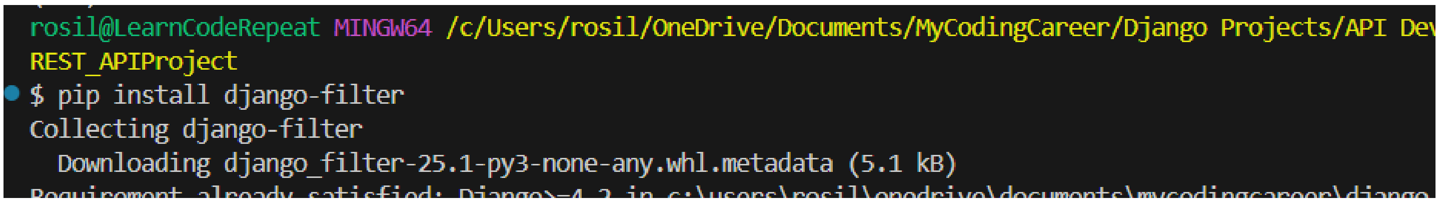
Project description

Release history

Project description

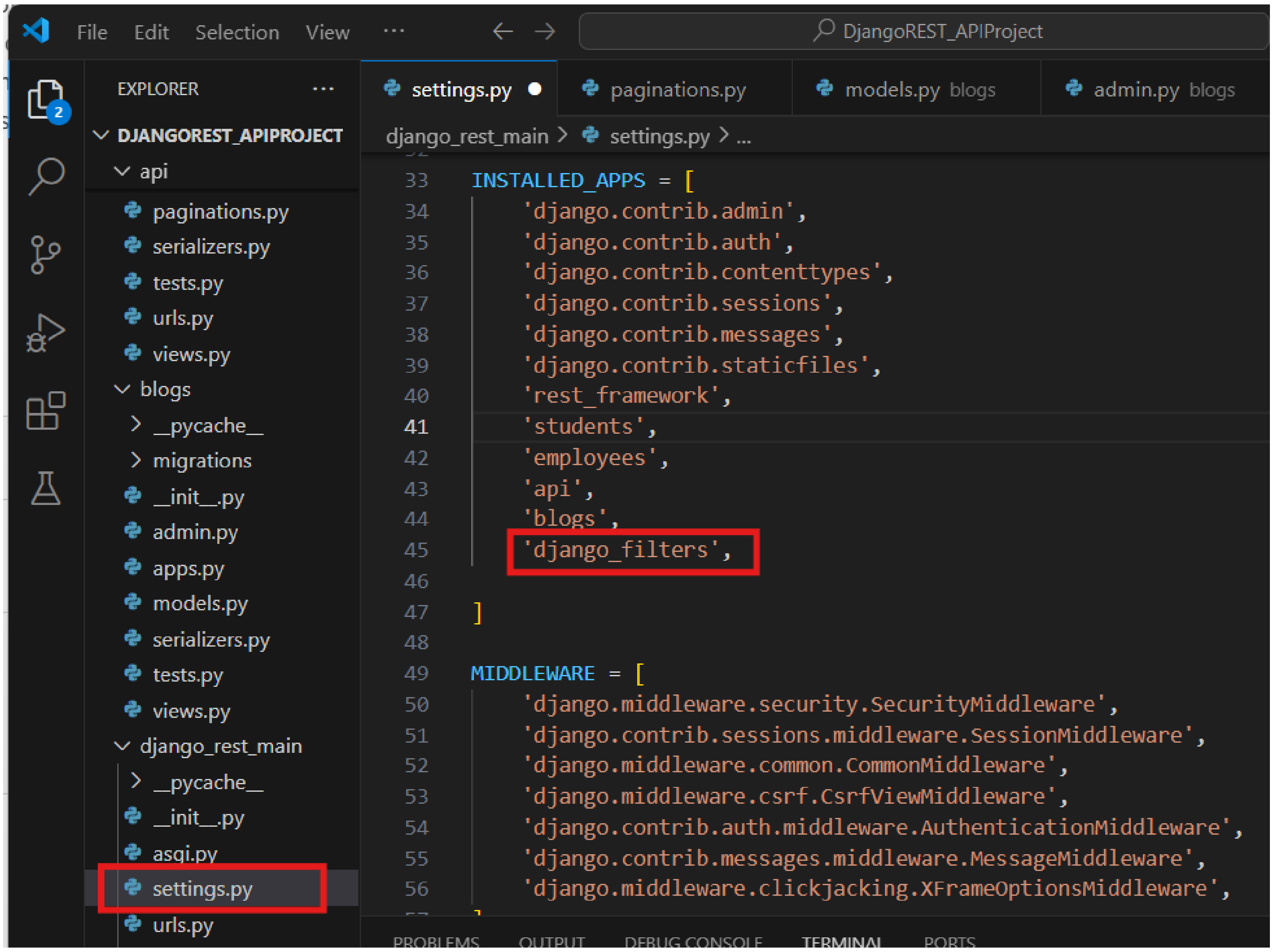
Django-filter is a reusable Django application allowing users to declaratively add dynamic `QuerySet` filtering from URL parameters.

Full documentation on [read the docs]().



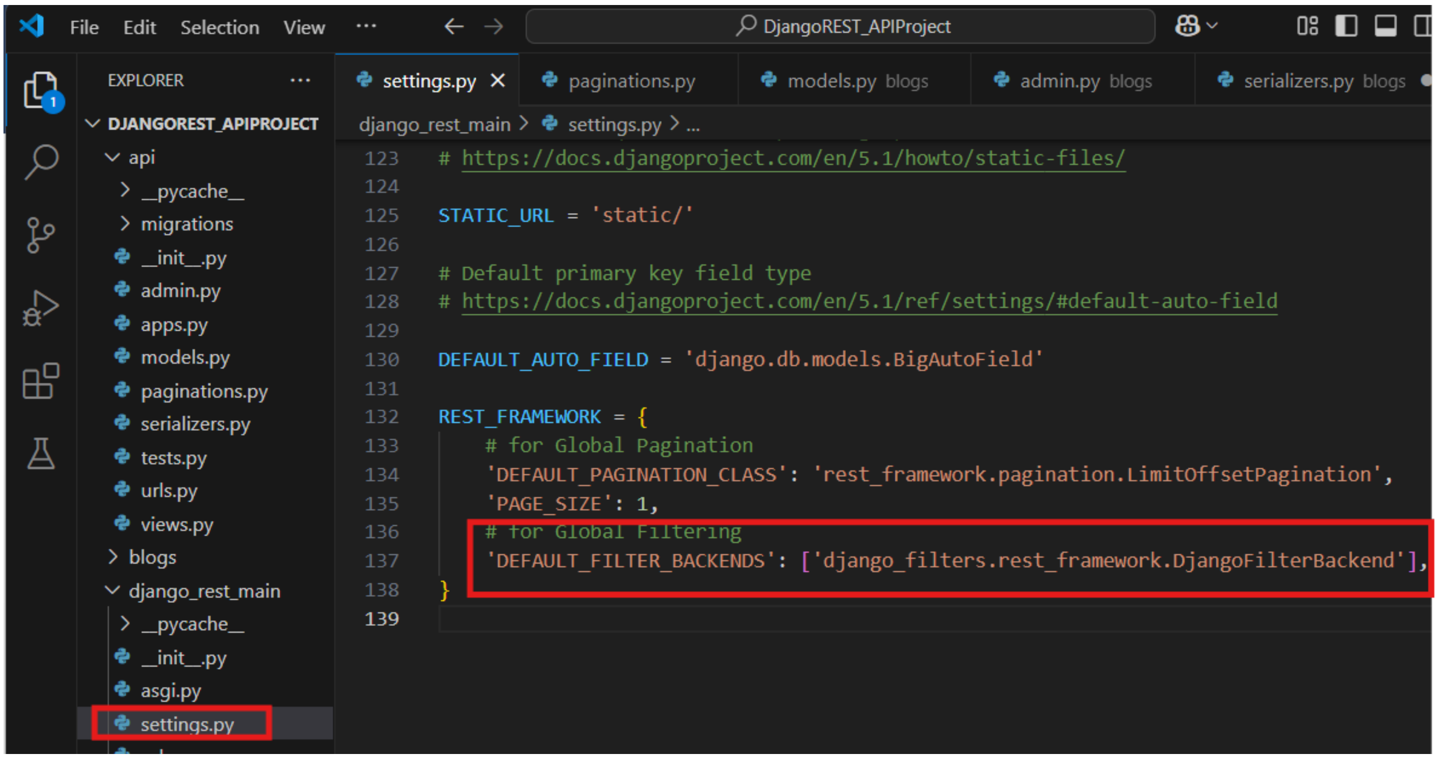
```
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API Dev
REST_APIProject
$ pip install django-filter
Collecting django-filter
  Downloading django_filter-25.1-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: Django>=4.2 in c:\users\rosil\onedrive\documents\mycodingcareer\django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

3. Register your Django package in SETTINGS.PY



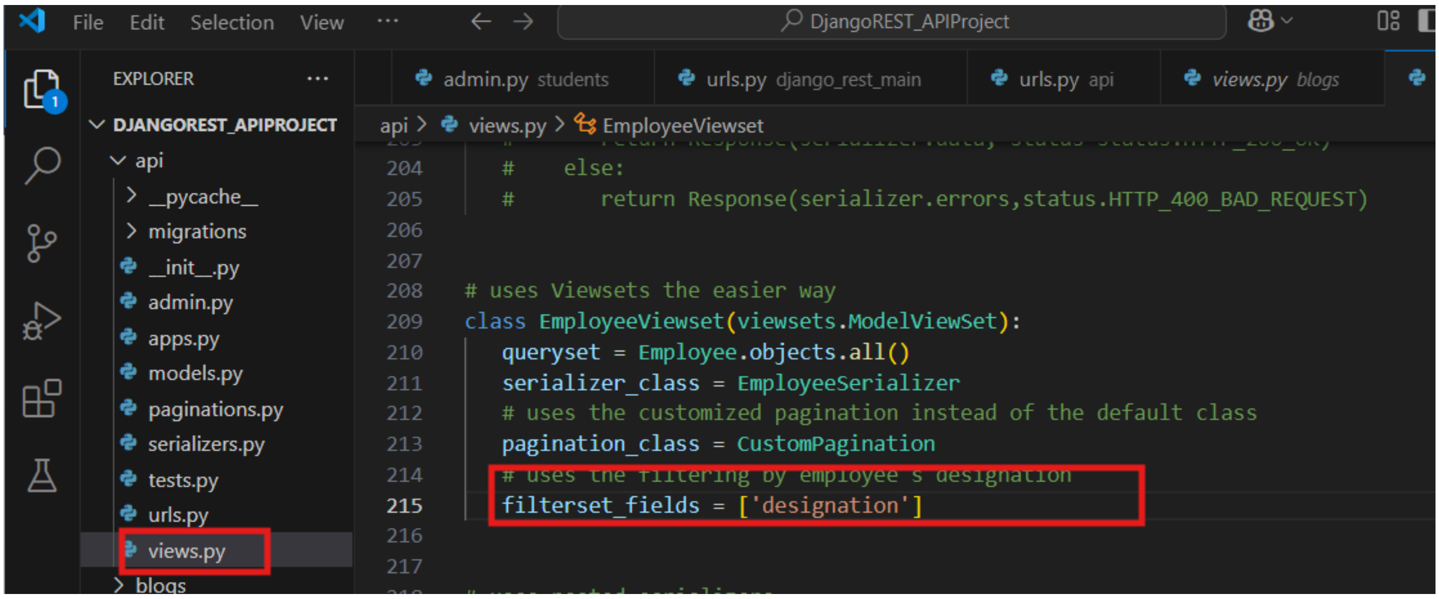
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'students',
    'employees',
    'api',
    'blogs',
    'django_filters',

]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

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

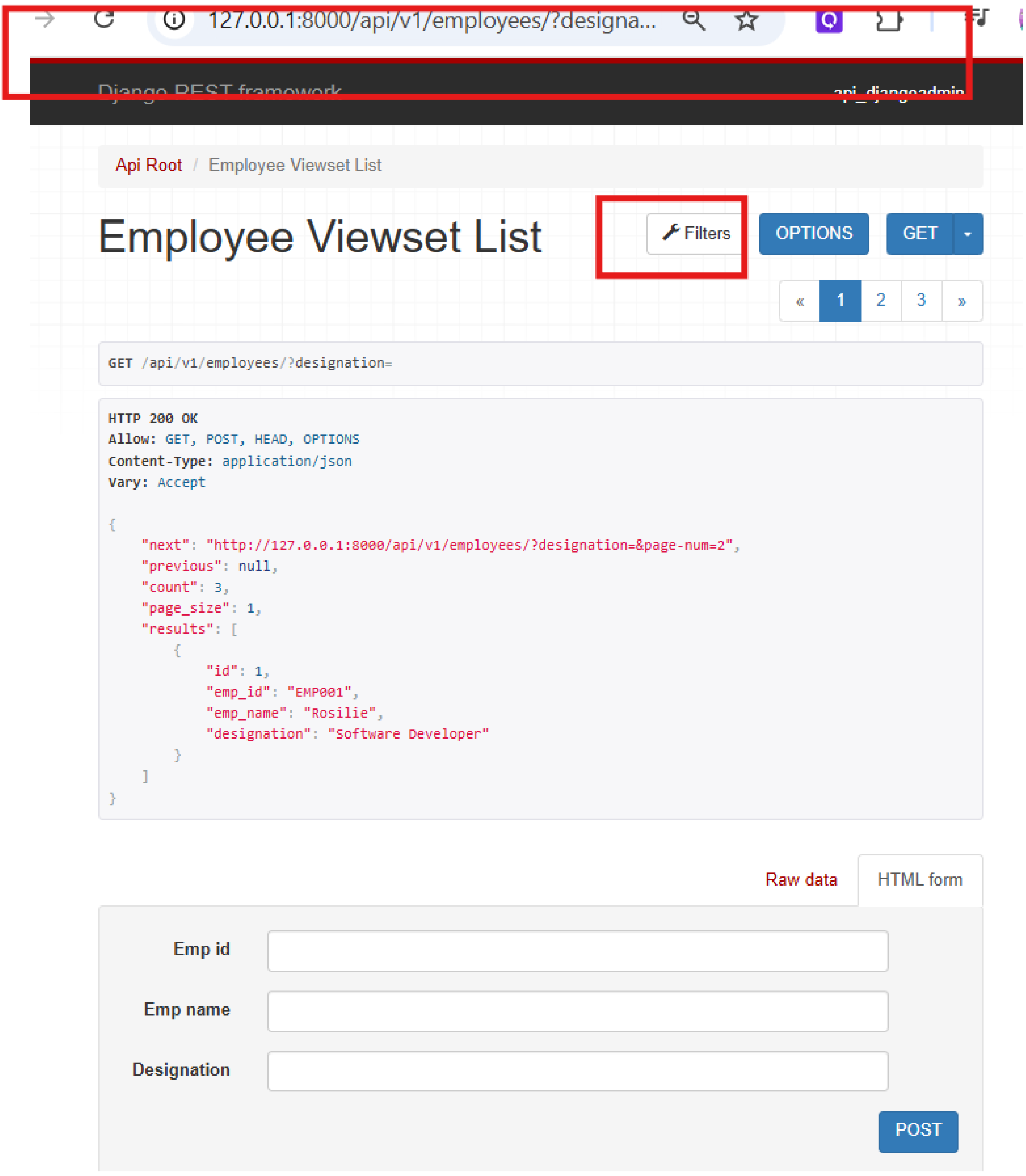


```
# https://docs.djangoproject.com/en/5.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

REST_FRAMEWORK = {
    # for Global Pagination
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.LimitOffsetPagination',
    'PAGE_SIZE': 1,
    # for Global Filtering
    'DEFAULT_FILTER_BACKENDS': ['django_filters.rest_framework.DjangoFilterBackend'],
}
```

5. Update our API\VIEWS.PY:



```
    #     else:
    #         return Response(serializer.errors,status.HTTP_400_BAD_REQUEST)


# uses Viewsets the easier way
class EmployeeViewset(viewsets.ModelViewSet):
    queryset = Employee.objects.all()
    serializer_class = EmployeeSerializer
    # uses the customized pagination instead of the default class
    pagination_class = CustomPagination
    # uses the filtering by employee's designation
    filterset_fields = ['designation']
```

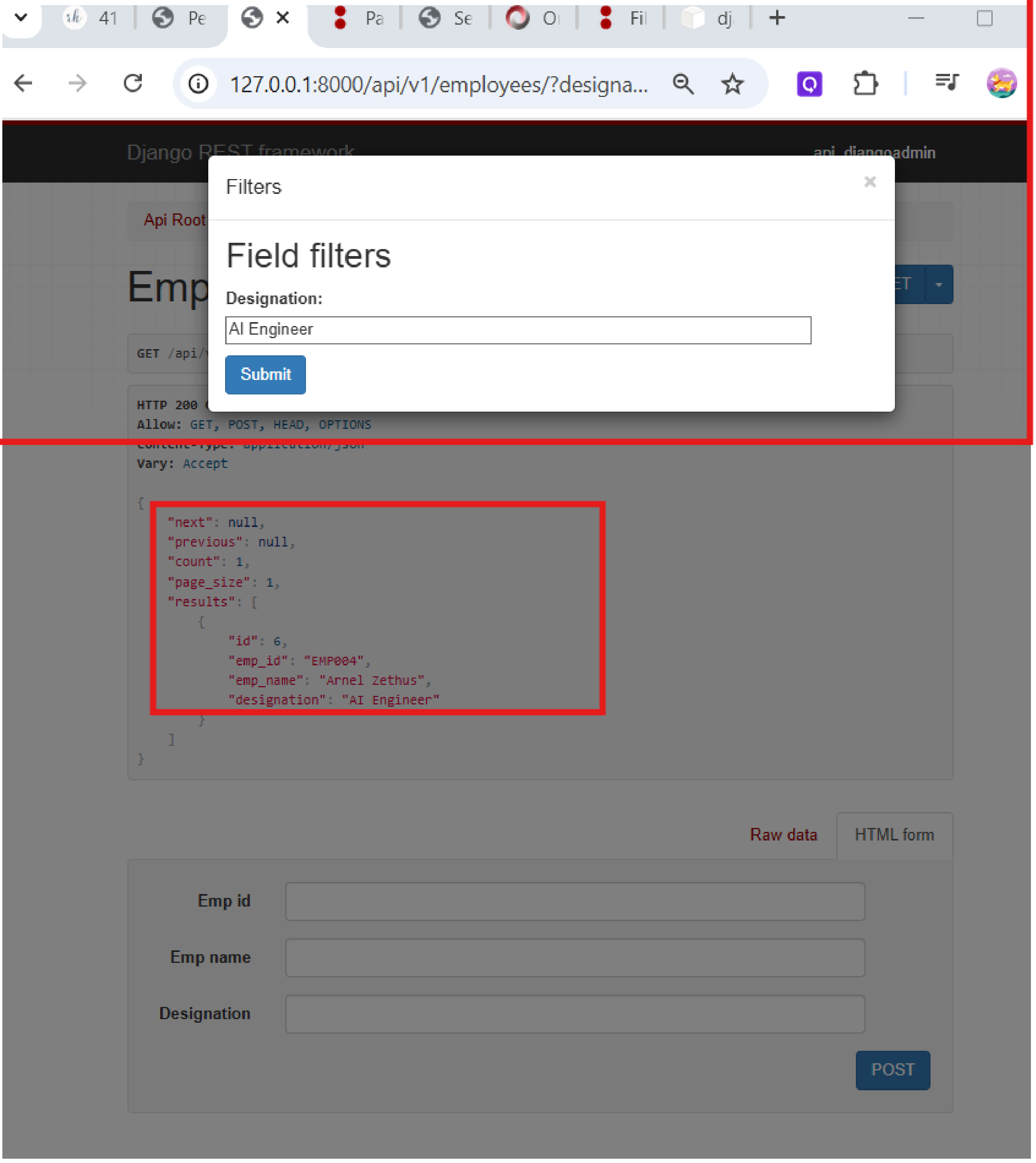


127.0.0.1:8000/api/v1/employees/?designa...

Employee Viewset List

```
GET /api/v1/employees/?designation=
```

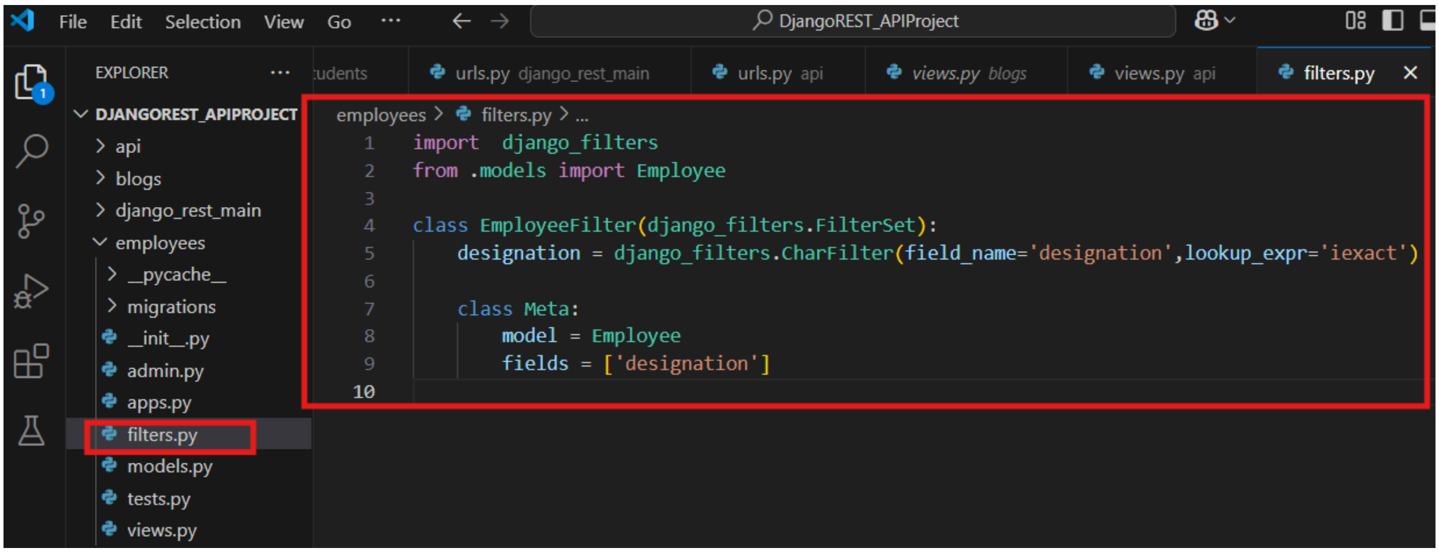
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "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"
        }
    ]
}
```



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

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

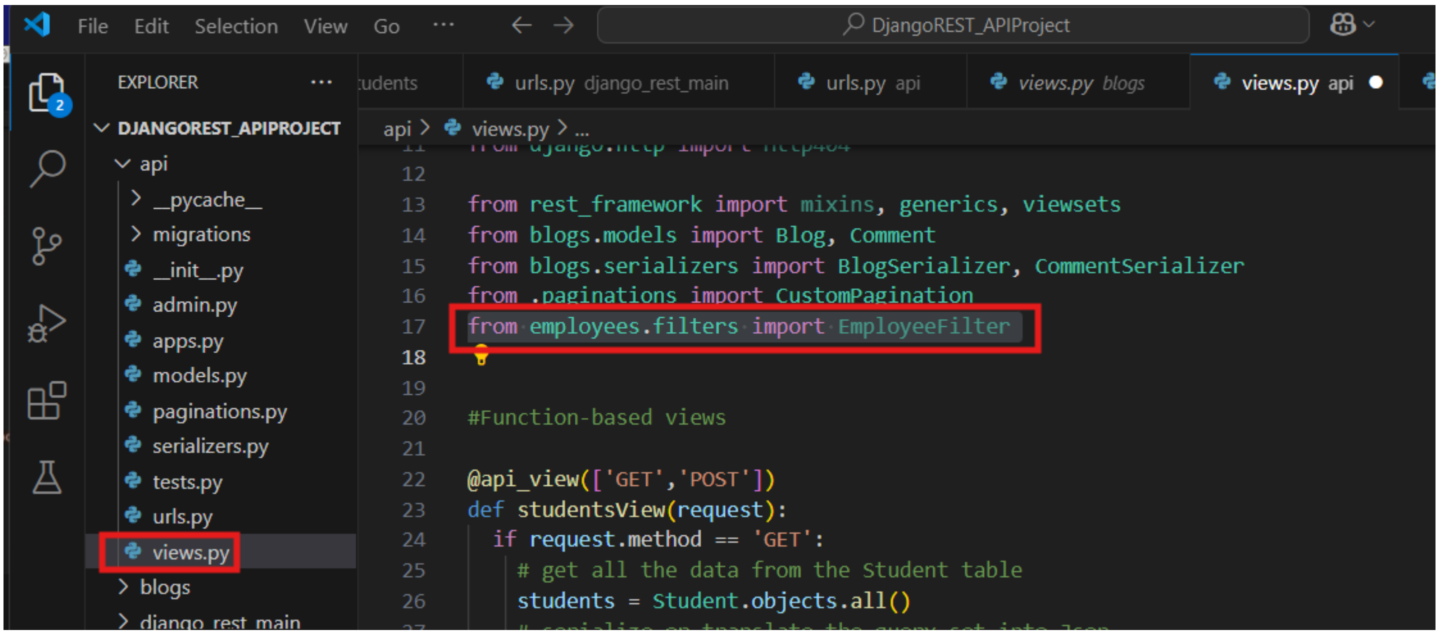
* You need to type the exact keyword to find an exact match, so this is a case-sensitive filter. To solve this, we use custom filters instead.

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.



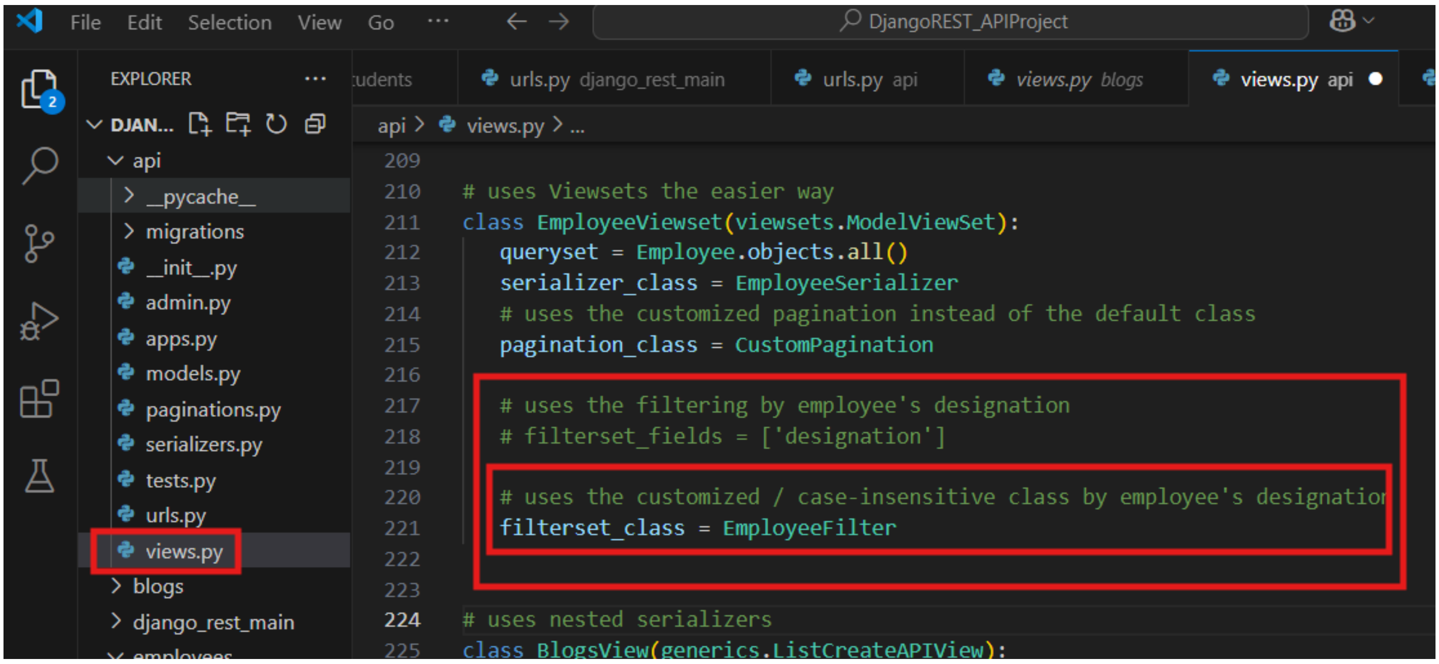
```python
import django_filters
from .models import Employee

class EmployeeFilter(django_filters.FilterSet):
    designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')

    class Meta:
        model = Employee
        fields = ['designation']
```

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



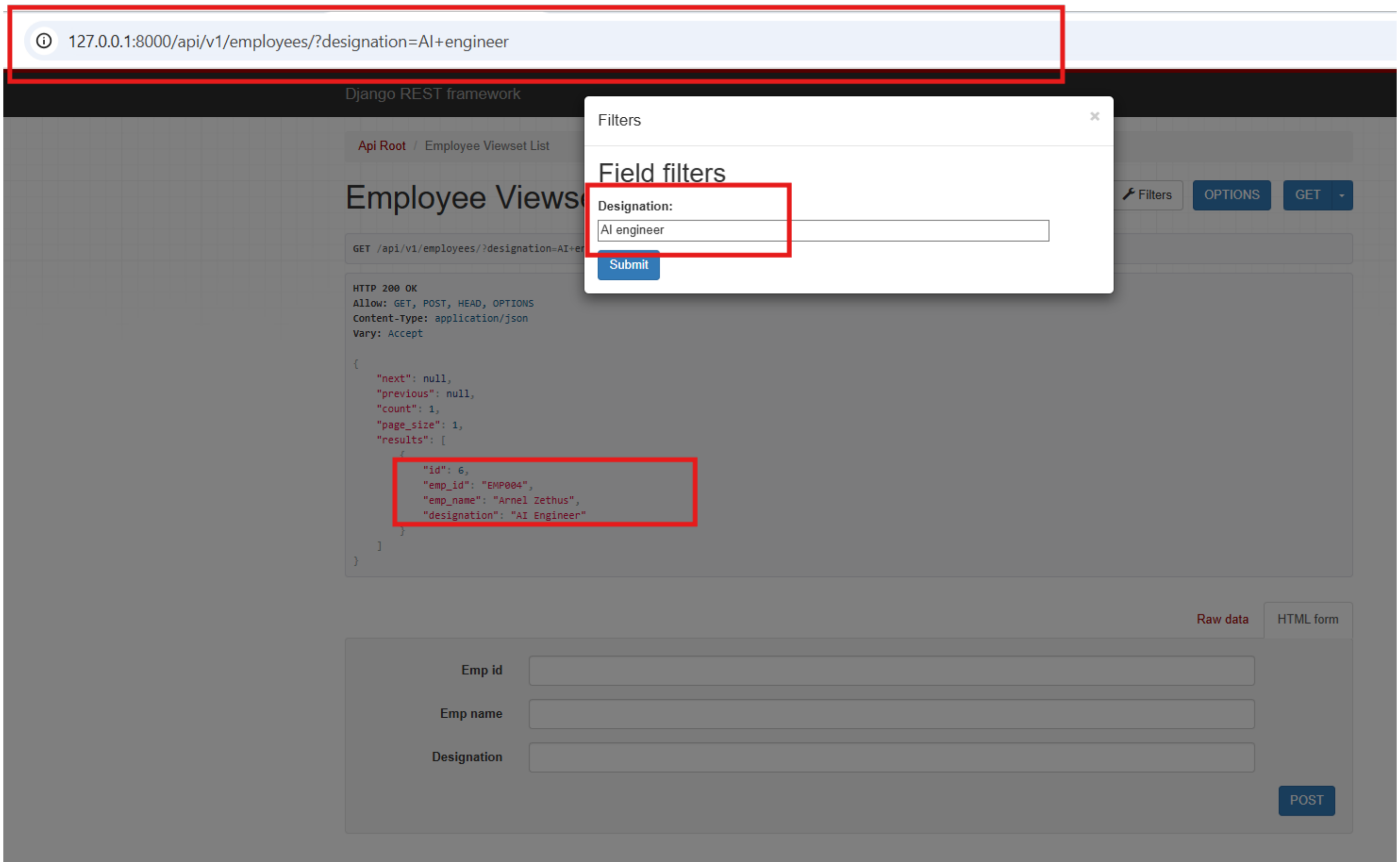
```python

from rest_framework import mixins, generics, viewsets
from blogs.models import Blog, Comment
from blogs.serializers import BlogSerializer, CommentSerializer
from .paginations import CustomPagination
from employees.filters import EmployeeFilter


#Function-based views

@api_view(['GET','POST'])
def studentsView(request):
    if request.method == 'GET':
        # get all the data from the Student table
        students = Student.objects.all()
```

To use the new filter class:



```python

# uses Viewsets the easier way
class EmployeeViewset(viewsets.ModelViewSet):
    queryset = Employee.objects.all()
    serializer_class = EmployeeSerializer
    # uses the customized pagination instead of the default class
    pagination_class = CustomPagination

    # uses the filtering by employee's designation
    # filterset_fields = ['designation']

    # uses the customized / case-insensitive class by employee's designatio
    filterset_class = EmployeeFilter


# uses nested serializers
class BlogsView(generics.ListCreateAPIView):
```

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


127.0.0.1:8000/api/v1/employees/?designation=AI+engineer

Designation: AI engineer

```
GET /api/v1/employees/?designation=AI+e
```

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

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

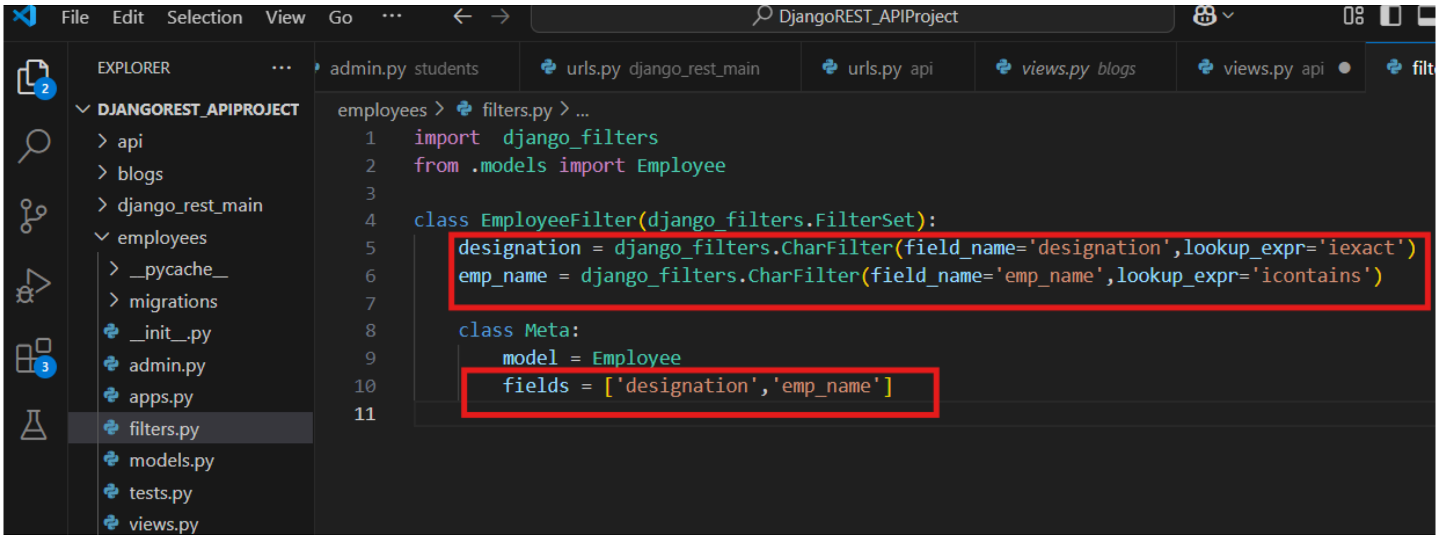
Emp id

Emp name

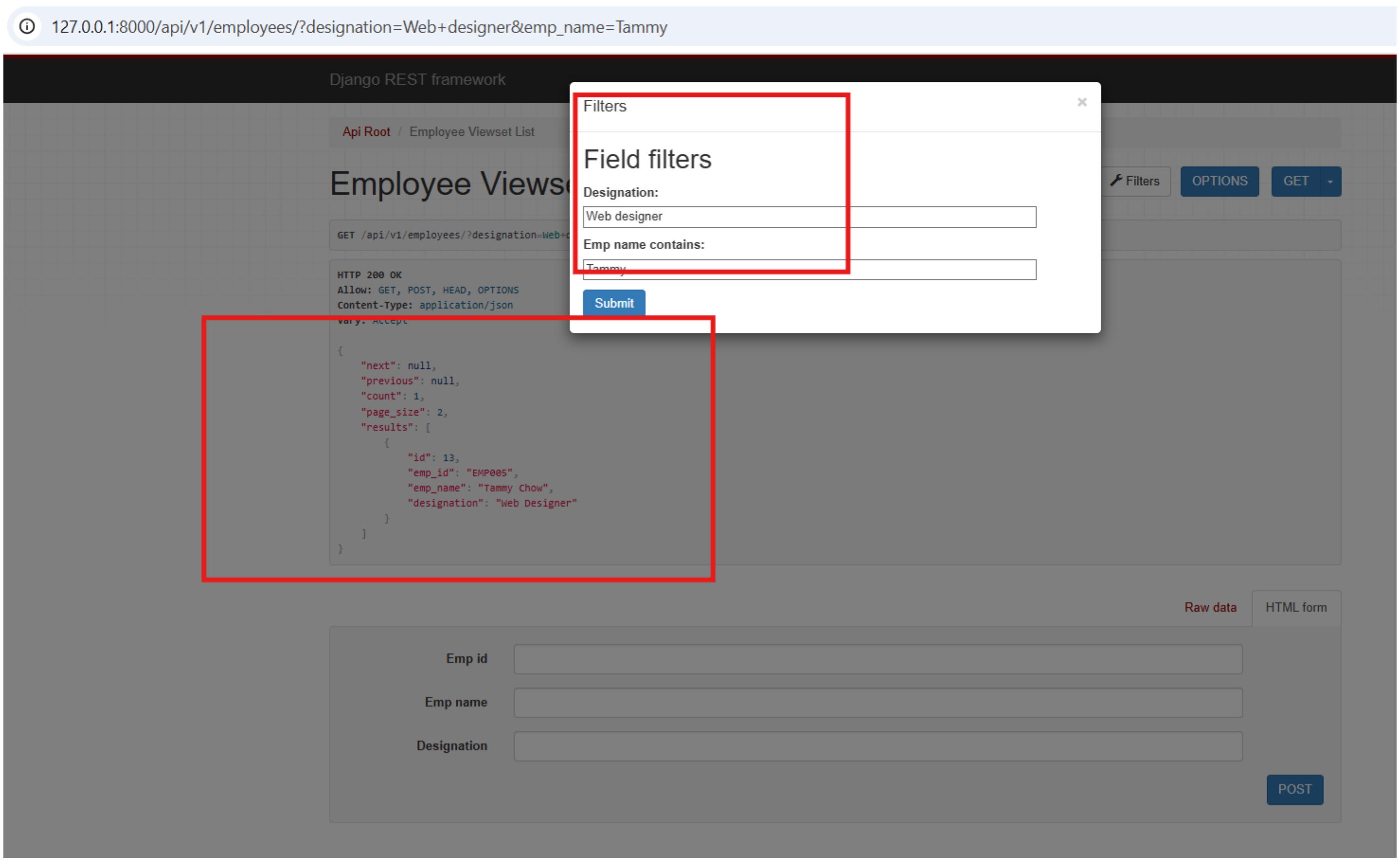
Designation

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.



```python
import  django_filters
from .models import Employee

class EmployeeFilter(django_filters.FilterSet):
    designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
    emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')

    class Meta:
        model = Employee
        fields = ['designation','emp_name']
```


127.0.0.1:8000/api/v1/employees/?designation=Web+designer&emp_name=Tammy

Field filters

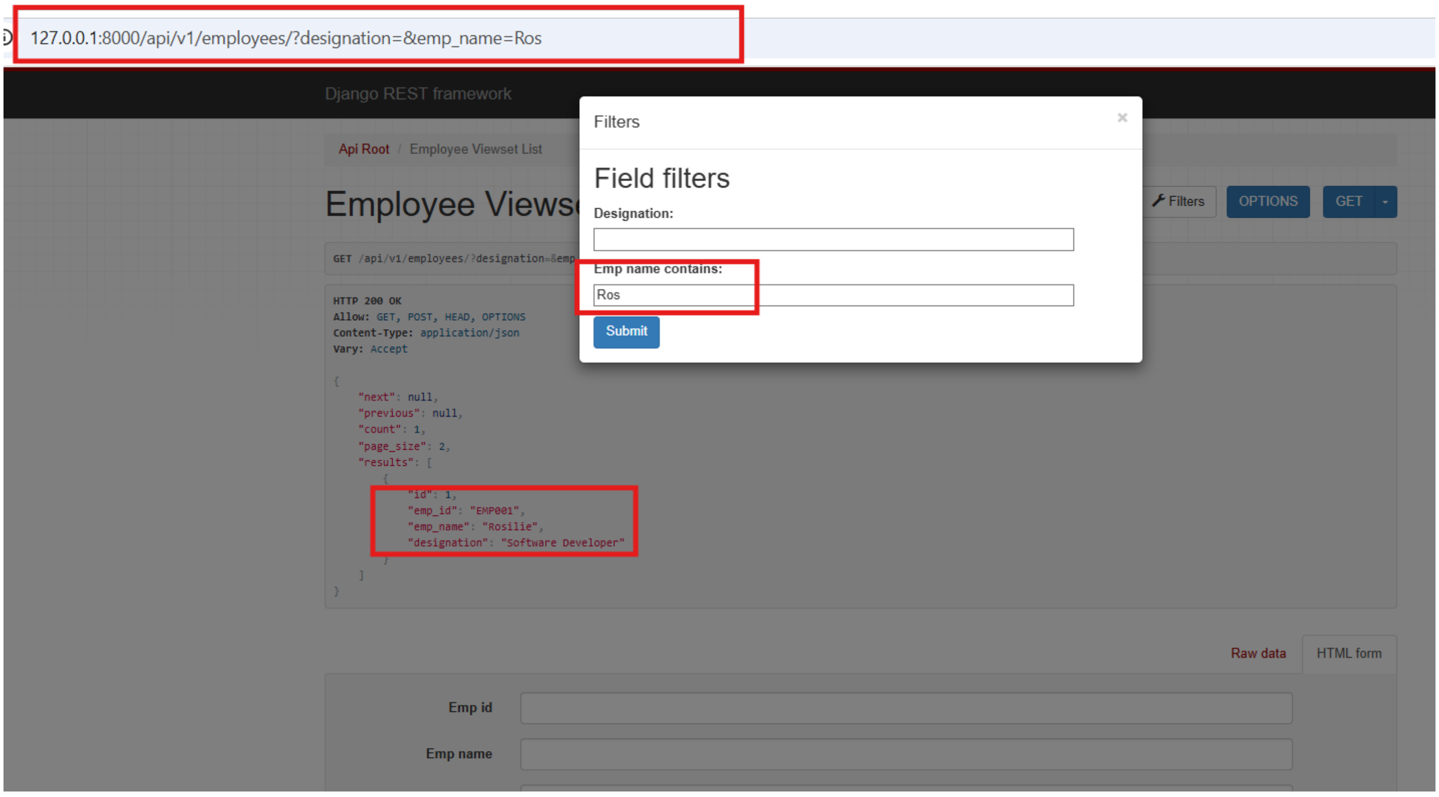
Designation: Web designer

Emp name contains: Tammy

```
GET /api/v1/employees/?designation=Web+d
```

```
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"
        }
    ]
}
```

Emp id	
Emp name	
Designation	


127.0.0.1:8000/api/v1/employees/?designation=&emp_name=Ros

Field filters

Designation:

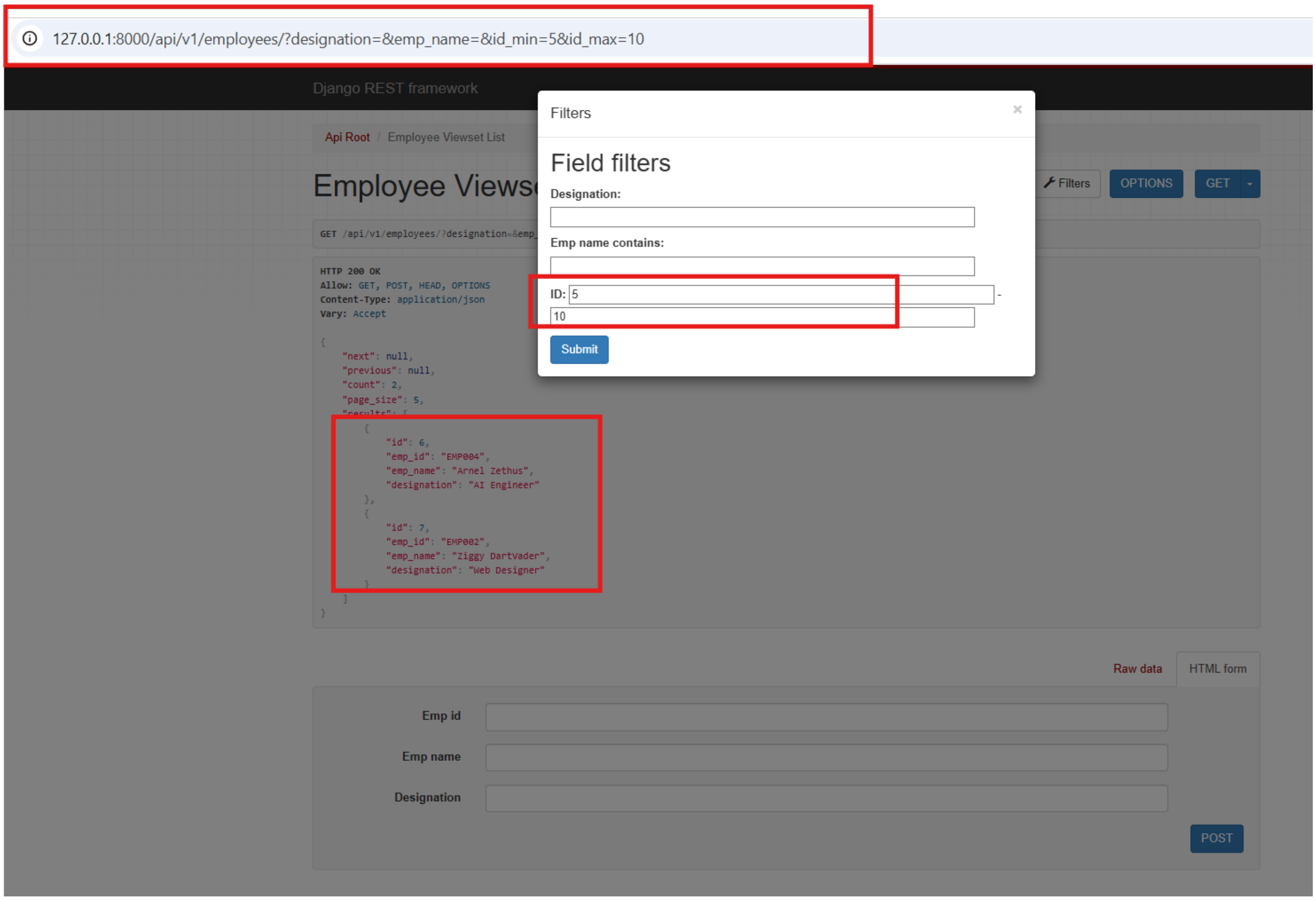
Emp name contains: Ros

```
GET /api/v1/employees/?designation=&emp
```

```
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": "Rosilie",
            "designation": "Software Developer"
        }
    ]
}
```

Emp id	
Emp name	

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



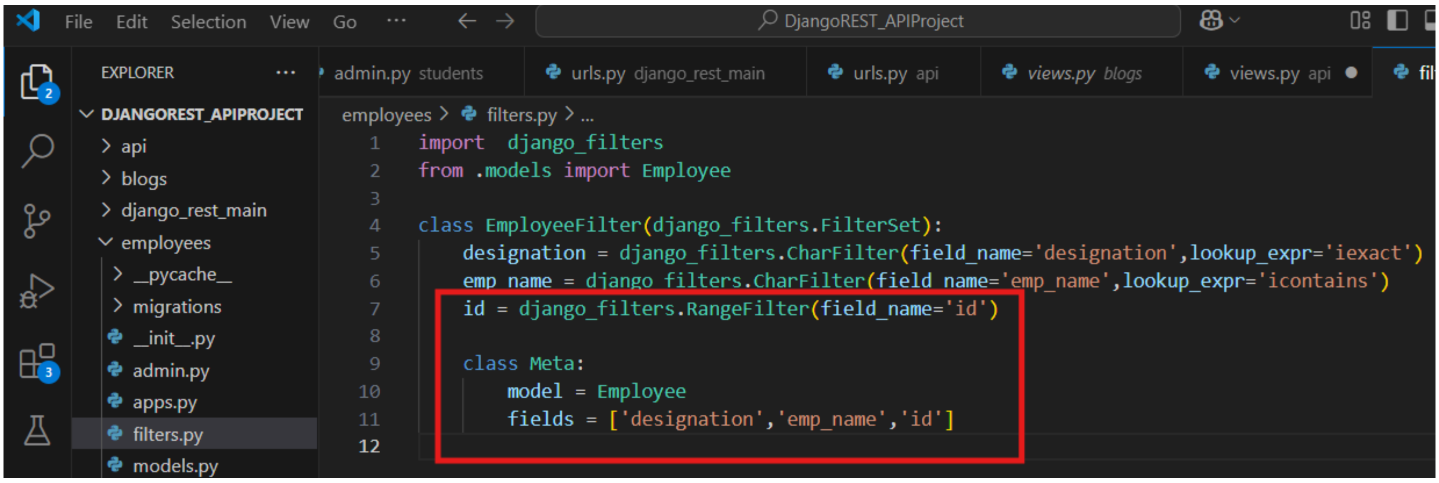
```
127.0.0.1:8000/api/v1/employees/?designation=&emp_name=&id_min=5&id_max=10
```

ID: 5 - 10

```
GET /api/v1/employees/?designation=&emp_
```

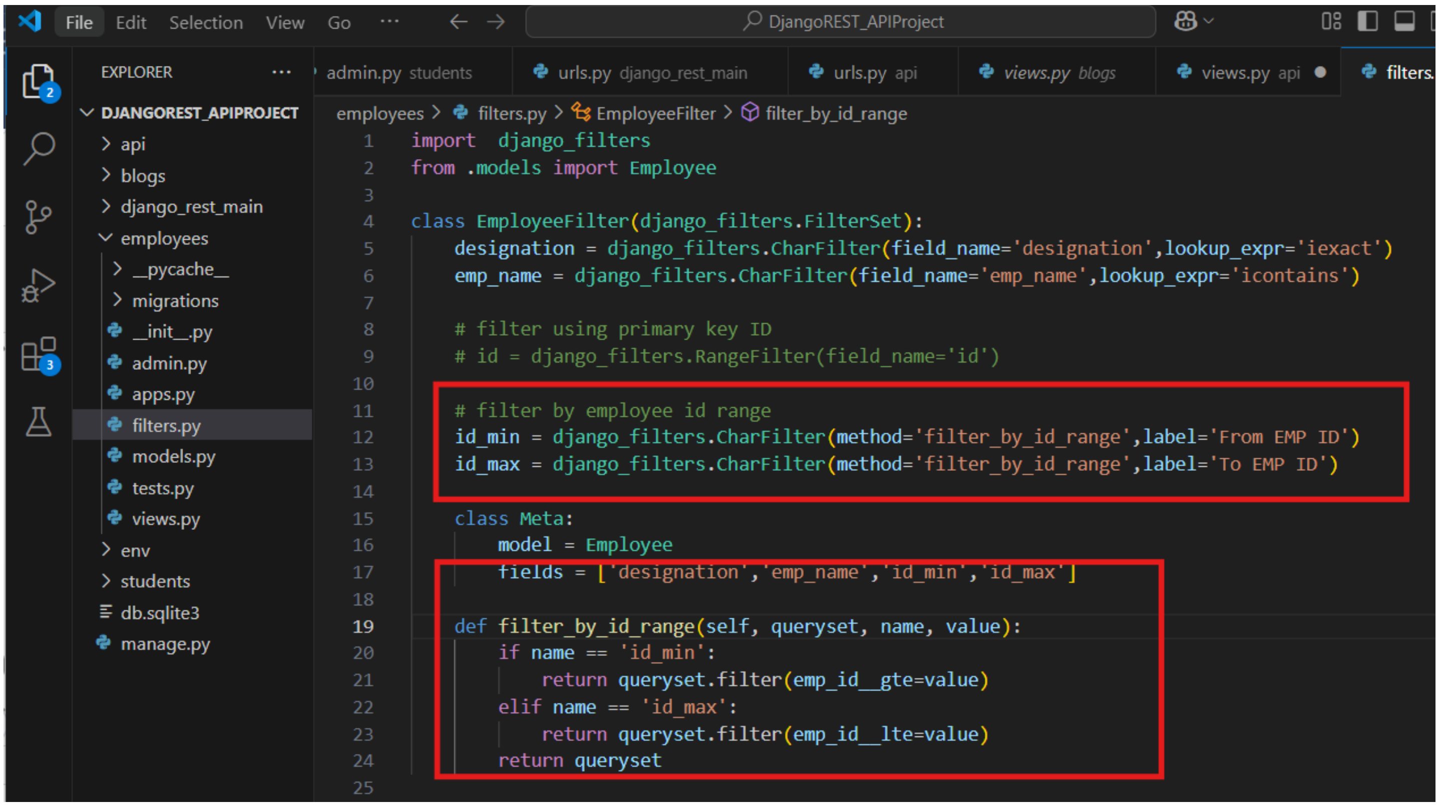
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "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"
        }
    ]
}
```

We update our FILTERS.PY as:



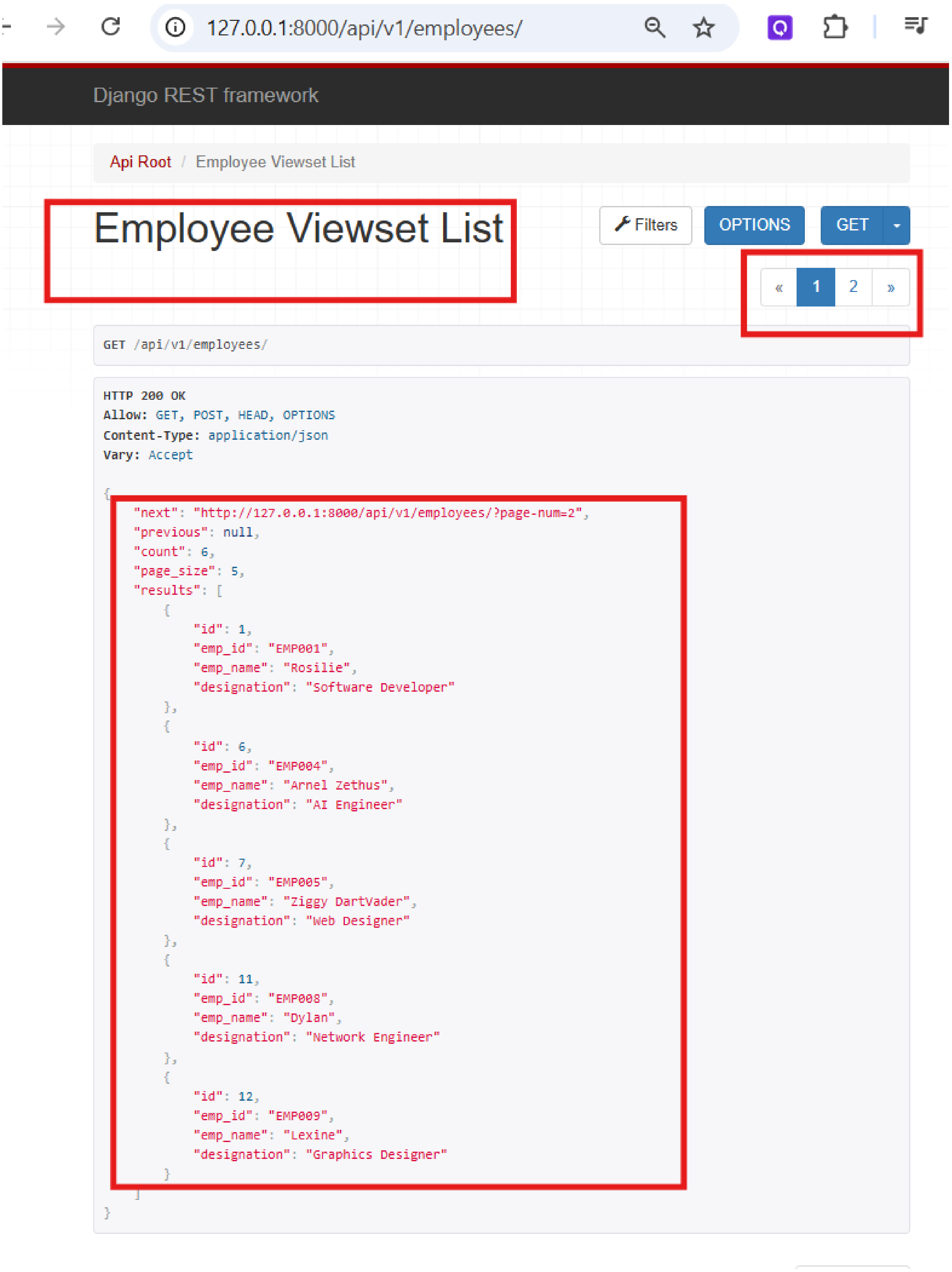
```python
import  django_filters
from .models import Employee

class EmployeeFilter(django_filters.FilterSet):
    designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
    emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')
    id = django_filters.RangeFilter(field_name='id')

    class Meta:
        model = Employee
        fields = ['designation','emp_name','id']
```

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



```python
import  django_filters
from .models import Employee

class EmployeeFilter(django_filters.FilterSet):
    designation = django_filters.CharFilter(field_name='designation',lookup_expr='iexact')
    emp_name = django_filters.CharFilter(field_name='emp_name',lookup_expr='icontains')

    # filter using primary key ID
    # id = django_filters.RangeFilter(field_name='id')

    # filter by employee id range
    id_min = django_filters.CharFilter(method='filter_by_id_range',label='From EMP ID')
    id_max = django_filters.CharFilter(method='filter_by_id_range',label='To EMP ID')

    class Meta:
        model = Employee
        fields = ['designation','emp_name','id_min','id_max']

    def filter_by_id_range(self, queryset, name, value):
        if name == 'id_min':
            return queryset.filter(emp_id__gte=value)
        elif name == 'id_max':
            return queryset.filter(emp_id__lte=value)
        return queryset

```

Before the Filter by Emp_ID:

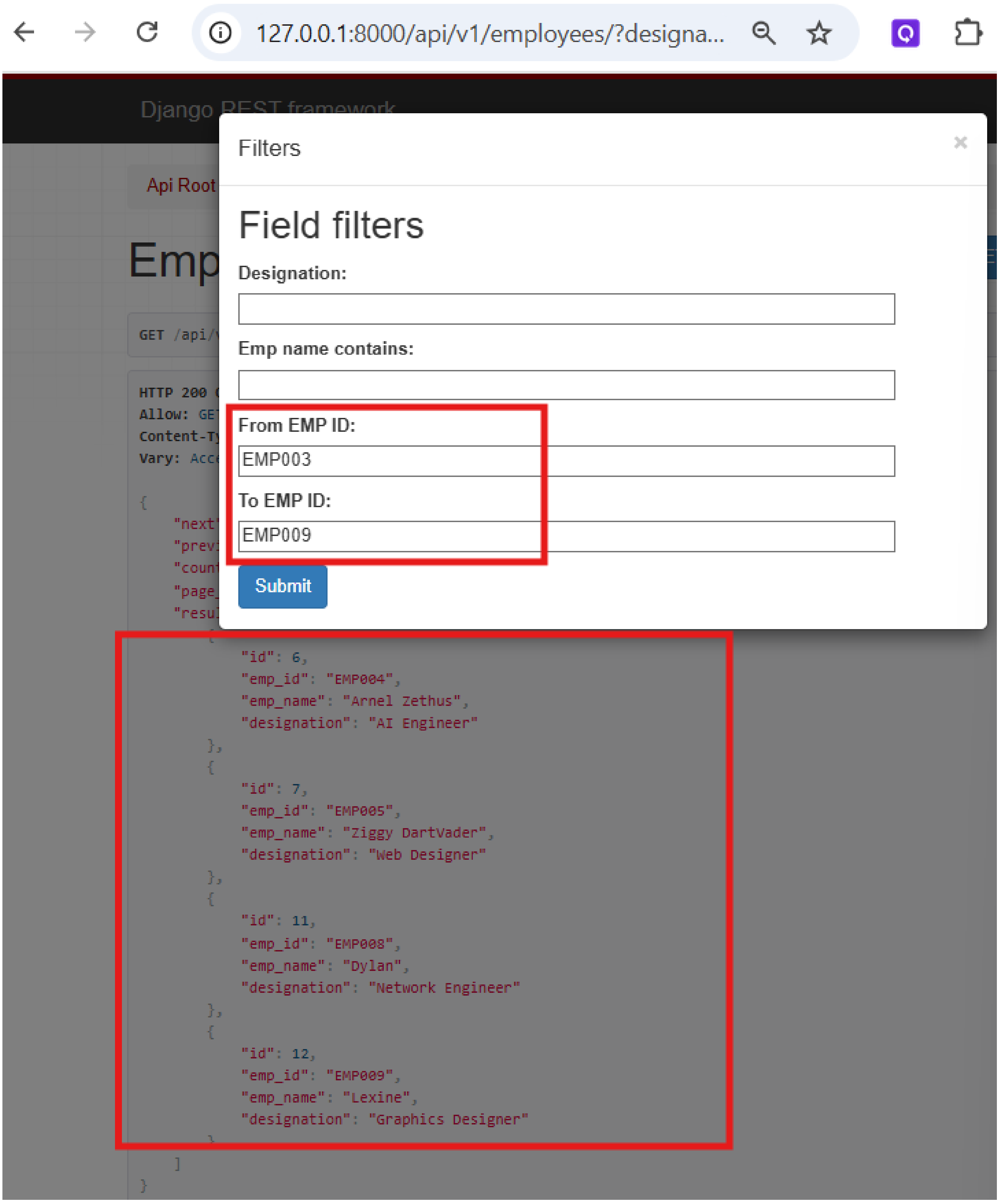

127.0.0.1:8000/api/v1/employees/
Django REST framework
Api Root / Employee Viewset List
Filters
OPTIONS
GET
« 1 2 »

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"
        }
    ]
}
```

After:



Designation:

Emp name contains:

From EMP ID: EMP003

To EMP ID: EMP009

```
GET /api/

HTTP 200
Allow: GE
Content-Ty
Vary: Acce

{
    "next
    "prev
    "coun
    "page
    "resu
        {
            "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"
        }
    ]
}
```