

Topic: 16. DRF Searching

Speaker: Personal / Notebook: API Development using Django Framework



We can search through the database fields like BLOGS' title or comments.

1. Go to APIVIEWS.PY:

```
File Edit Selection View Go ... DjangoREST_APIProject
admin.py students urls.py django_rest_main urls.py api views.py blogs
DJANGOREST_APIPROJECT
  api
    > __pycache__
    > migrations
    > _init_.py
    > admin.py
    > apps.py
    > models.py
    > paginations.py
    > serializers.py
    > tests.py
    > urls.py
    > views.py
  > blogs
  > django_rest_main
  > employees
  > env
  > students
  db.sqlite3
api > views.py > ...
1 from django.shortcuts import render, get_object_or_404
2 # from django.http import JsonResponse
3 from students.models import Student
4 from employees.models import Employee
5
6 from .serializers import StudentSerializer, EmployeeSerializer
7 from rest_framework.response import Response
8 from rest_framework import status
9 from rest_framework.decorators import api_view
10 from rest_framework.views import APIView
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 from rest_framework.filters import SearchFilter
19
20
21
```

```
File Edit Selection View Go ... < -> DjangoREST_APIProject

EXPLORER
  DJANGOREST_APIPROJECT
    api
      __pycache__
      migrations
      __init__.py
      admin.py
      apps.py
      models.py
      paginations.py
      serializers.py
      tests.py
      urls.py
      views.py
      blogs

api > views.py > CommentsView
213 class EmployeeViewSet(viewsets.ModelViewSet):
222     # uses the customized / case-insensitive class by employee
223     filterset_class = EmployeeFilter
224
225
226     # uses nested serializers
227     class BlogView(generics.ListCreateAPIView):
228         queryset = Blog.objects.all()
229         serializer_class = BlogSerializer
230         filter_backends = [SearchFilter]
231         search_fields = ['blog_title']
232
233     class CommentsView(generics.ListCreateAPIView):
234         queryset = Comment.objects.all()
235         serializer_class = CommentSerializer
236
```

2. Now search thru BLOGS model:

Django REST framework

Api Root

Blog

Filters

Search

Apple Search

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

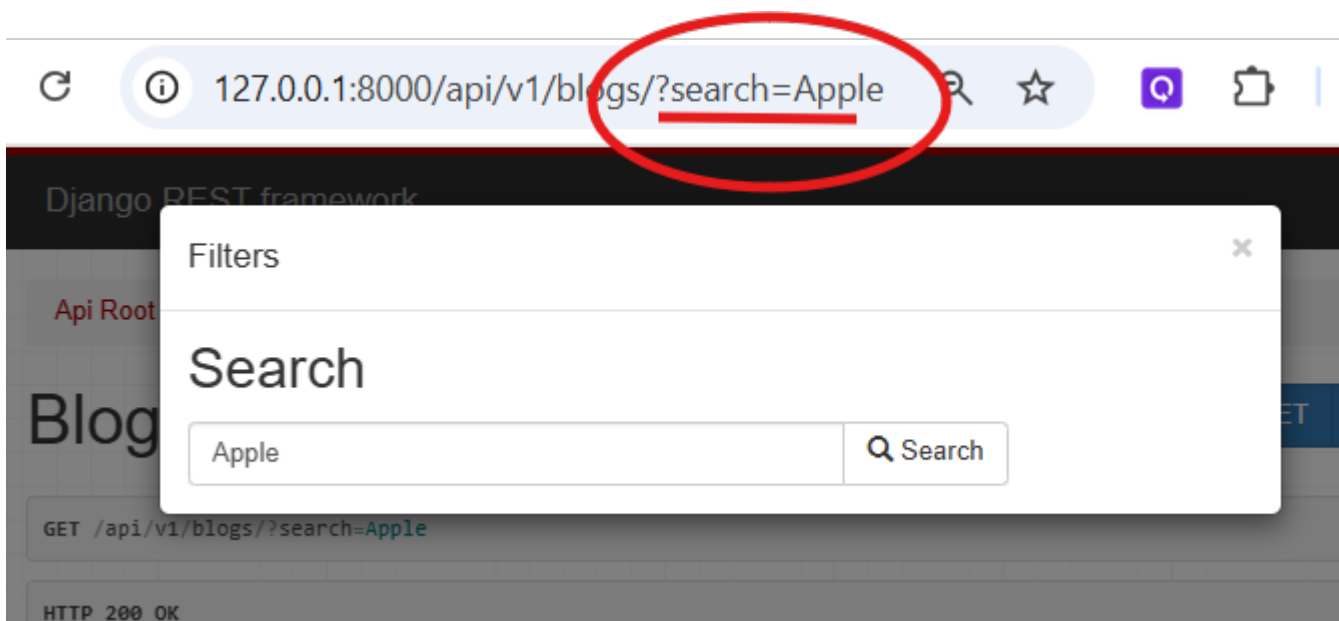
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 4,
      "comments": [
        {
          "id": 9,
          "comment": "Gosh! Expensive!",
          "blog": 4
        },
        {
          "id": 10,
          "comment": "I am not buying that!",
          "blog": 4
        },
        {
          "id": 11,
          "comment": "That's crazy!"
          "blog": 4
        }
      ],
      "blog_title": "A $2,300 Apple iPhone? Trump tariffs could make that happen.",
      "blog_body": "Tariffs could increase iPhone prices by up to 43% - Rosenblatt Securities\r\nSamsu"
    }
  ]
}
```

3. If you want to change the search keyword 'search' to 'q,' then update the SETTINGS.PY

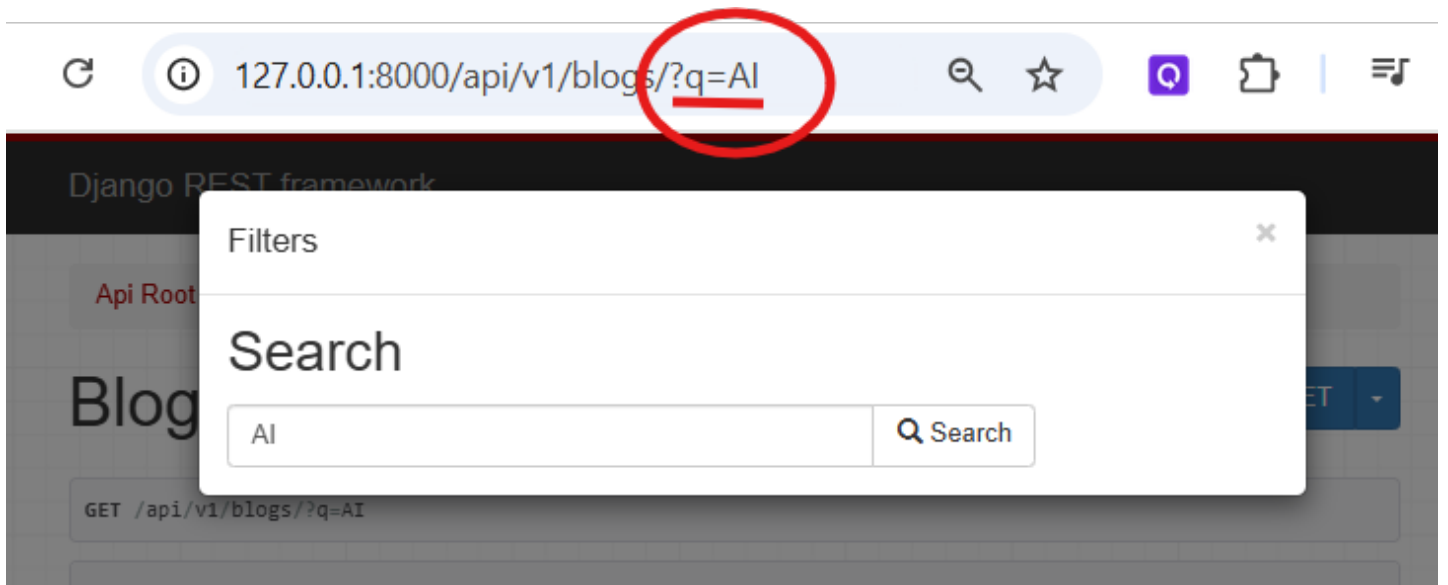
```
131
132 REST_FRAMEWORK = {
133     # for Global Pagination
134     'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.LimitOffsetPagination',
135     'PAGE_SIZE': 5,
136     # for Global Filtering
137     'DEFAULT_FILTER_BACKENDS': ['django_filters.rest_framework.DjangoFilterBackend'],
138     # overrides the query parameter instead of the default label 'search'
139     'SEARCH_PARAM': 'q',
140 }
141
```

The Explorer sidebar on the left shows the project structure with 'settings.py' highlighted in the 'django_rest_main' folder.

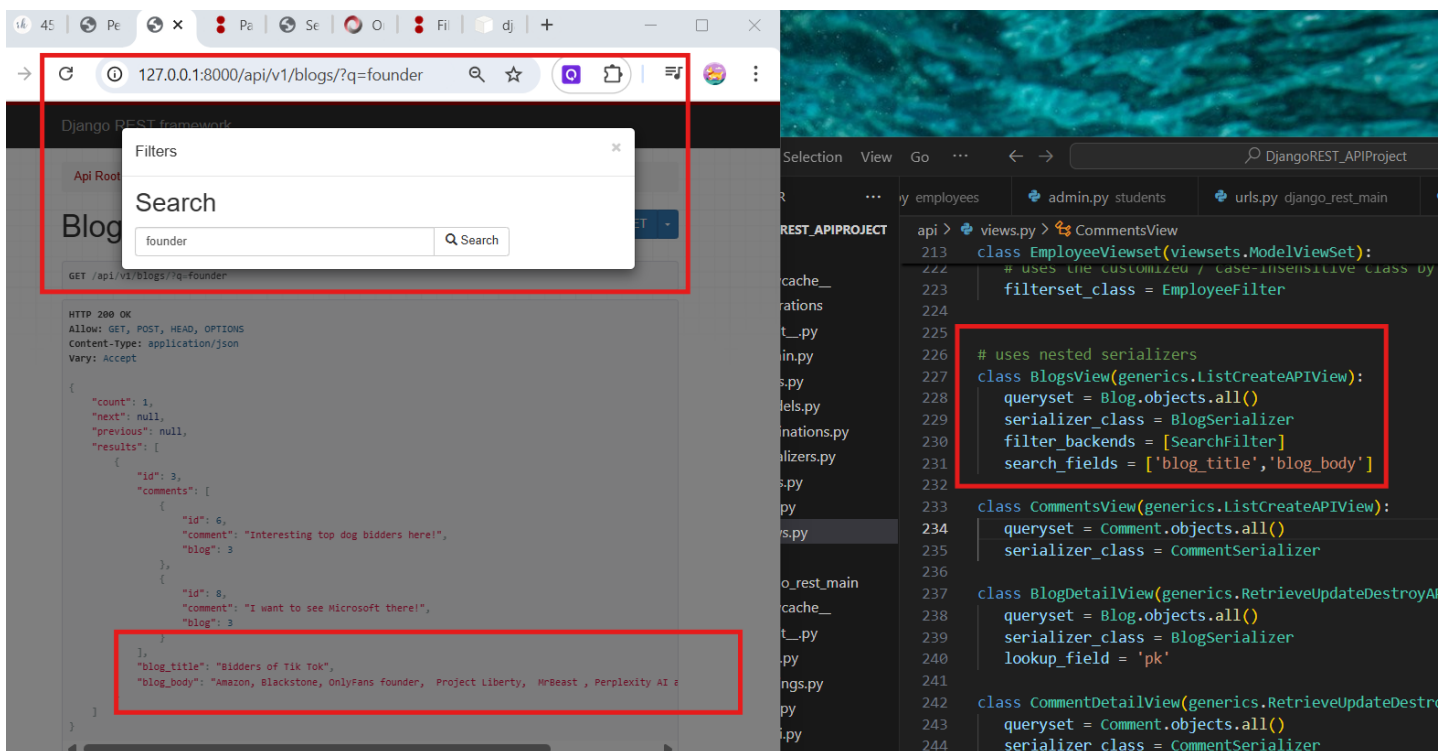
FROM:



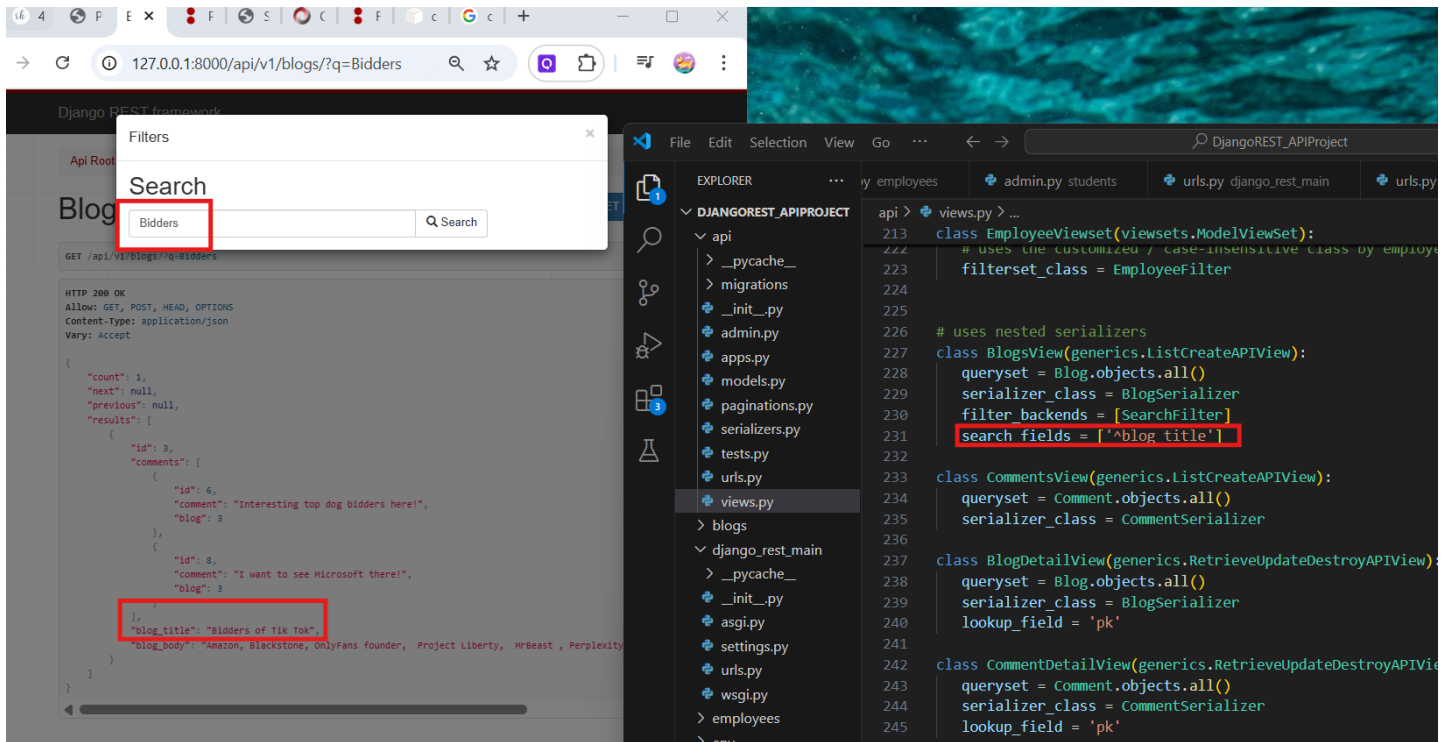
TO:



4. To search through the BLOG_BODY:



5. To search for a blog title that starts with a specific keyword, use the caret symbol (^).



But when you use a keyword, not the actual starting word, it won't show any result.

