Filtering

The simplest way to filter the queryset of any view that subclasses GenericAPIView is to override the .get_queryset() method.

Filtering against the current user

Generic Filtering

REST framework also includes support for generic filtering backends that allow you to easily construct complex searches and filters.

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’,
]
```

https://django-filter.readthedocs.io/en/latest/index.html

Global Setting

Settings.py

```
REST_FRAMEWORK = {
    'DEFAULT_FILTER_BACKENDS':
['django_filters.rest_framework.DjangoFilterBackend']
}
```

Per View Setting

You can set the filter backends on a per-view, or per-viewset basis, using the GenericAPIView class-based views.

```
from django_filters.rest_framework import DjangoFilterBackend
class StudentListView(ListAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
    filter_backends = [DjangoFilterBackend]
```

DjangoFilterBackend

If all you need is simple equality-based filtering, you can set a filterset_fields attribute on the view, or viewset, listing the set of fields you wish to filter against.

```
class StudentList(ListAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
    filter_backends = [DjangoFilterBackend]
    filterset_fields = [‘name', ‘city’]
```

http://127.0.0.1:8000/studentapi/?name=Sonam&city=Ranchi

SearchFilter

The SearchFilter class supports simple single query parameter based searching, and is based on the Django admin's search functionality.

The SearchFilter class will only be applied if the view has a *search_fields* attribute set. The search_fields attribute should be a list of names of text type fields on the model, such as CharField or TextField.

SearchFilter

```
from rest_framework.filters import SearchFilter
class StudentListView(ListAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
    filter_backends = [SearchFilter]
    search_fields = [‘city’]
```

http://127.0.0.1:8000/studentapi/?search=Ranchi

SearchFilter

- '^' Starts-with search.
- '=' Exact matches.
- '@' Full-text search. (Currently only supported Django's PostgreSQL backend.)
- '$' Regex search.

Example:-

search_fields = [‘^name’,]

http://127.0.0.1:8000/studentapi/?search=r

OrderingFilter

The OrderingFilter class supports simple query parameter controlled ordering of results.

http://127.0.0.1:8000/studentapi/?ordering=name

The client may also specify reverse orderings by prefixing the field name with '-', like so:

http://127.0.0.1:8000/studentapi/?ordering=-name

Multiple orderings may also be specified:

http://example.com/api/users?ordering=account,username

OrderingFilter

It's recommended that you explicitly specify which fields the API should allowing in the ordering filter. You can do this by setting an ordering_fields attribute on the view, like so:

```
class StudentListView(generics.ListAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
    filter_backends = [OrderingFilter]
    ordering_fields = [‘name']
    ordering_fields = [‘name', ‘city’]
    ordering_fields = '__all__'
```