# django-elasticsearch-dsl-drf Documentation

***Release 0.14***

**Artur Barseghyan <artur.barseghyan@gmail.com>**

**Aug 06, 2018**

# Contents


**1 Prerequisites** **3**

**2 Dependencies** **5**

**3 Documentation** **7**

**4 Main features and highlights** **9**

**5 Installation** **11**

**6 Quick start** **13**

**7 Testing** **15**

**8 Writing documentation** **17**

**9 License** **19**

**10 Support** **21**

**11 Author** **23**

**12 Project documentation** **25**

12.1 Installing Elasticsearch . . . 25
12.1.1 Docker . . . 26
12.1.1.1 5.x . . . 26
12.1.1.2 6.x . . . 26
12.1.2 Vagrant . . . 26
12.1.2.1 2.x . . . 26
12.2 Quick start . . . 26
12.2.1 Installation . . . 28
12.2.2 Example app . . . 28
12.2.2.1 Sample models . . . 29
12.2.2.1.1 Required imports . . . 29
12.2.2.1.2 Book statuses . . . 29
12.2.2.1.3 Publisher model . . . 29
12.2.2.1.4 Author model . . . 30
12.2.2.1.5 Tag model . . . 31

12.2.2.1.6 Book model . . . 31
12.2.2.2 Admin classes . . . 32
12.2.2.3 Create database tables . . . 33
12.2.2.4 Fill in some data . . . 33
12.2.2.5 Sample document . . . 33
12.2.2.5.1 Required imports . . . 34
12.2.2.5.2 Index definition . . . 34
12.2.2.5.2.1 Settings . . . 34
12.2.2.5.2.2 Document index . . . 34
12.2.2.5.3 Custom analyzers . . . 35
12.2.2.5.4 Document definition . . . 35
12.2.2.6 Syncing Django's database with Elasticsearch indexes . . . 36
12.2.2.6.1 Full database sync . . . 36
12.2.2.6.2 Sample partial sync (using custom signals) . . . 37
12.2.2.6.2.1 Required imports . . . 37
12.2.2.6.2.2 Update book index on related model change . . . 37
12.2.2.6.2.3 Update book index on related model removal . . . 37
12.2.2.7 Sample serializer . . . 38
12.2.2.7.1 Required imports . . . 38
12.2.2.7.2 Serializer definition . . . 39
12.2.2.8 ViewSet definition . . . 40
12.2.2.8.1 Required imports . . . 40
12.2.2.8.2 ViewSet definition . . . 41
12.2.2.9 URLs . . . 43
12.2.2.9.1 Required imports . . . 43
12.2.2.9.2 Router definition . . . 43
12.2.2.9.3 URL patterns . . . 43
12.2.2.10 Check what you've done so far . . . 43
12.2.2.10.1 URLs . . . 44
12.2.2.10.2 Test in browser . . . 44
12.2.3 Development and debugging . . . 44
12.2.3.1 Profiling tools . . . 44
12.2.3.1.1 Installation . . . 44
12.2.3.1.2 Configuration . . . 44
12.2.3.2 Debugging . . . 45
12.3 Filter usage examples . . . 45
12.3.1 Search . . . 46
12.3.1.1 Search in all fields . . . 46
12.3.1.2 Search a single term on specific field . . . 47
12.3.1.3 Search for multiple terms . . . 47
12.3.1.4 Search for multiple terms in specific fields . . . 47
12.3.2 Filtering . . . 47
12.3.2.1 Supported lookups . . . 47
12.3.2.1.1 Native . . . 47
12.3.2.1.1.1 term . . . 47
12.3.2.1.1.2 terms . . . 48
12.3.2.1.1.3 range . . . 48
12.3.2.1.1.4 exists . . . 48
12.3.2.1.1.5 prefix . . . 48
12.3.2.1.1.6 wildcard . . . 48
12.3.2.1.1.7 ids . . . 48
12.3.2.1.2 Functional . . . 49
12.3.2.1.2.1 contains . . . 49
12.3.2.1.2.2 in . . . 49

12.3.2.1.2.3 gt . . . 49
12.3.2.1.2.4 gte . . . 49
12.3.2.1.2.5 lt . . . 49
12.3.2.1.2.6 lte . . . 50
12.3.2.1.2.7 startswith . . . 50
12.3.2.1.2.8 endswith . . . 50
12.3.2.1.2.9 isnull . . . 50
12.3.2.1.2.10 exclude . . . 50
12.3.3 Usage examples . . . 50
12.4 Search backends . . . 51
12.4.1 Compound search filter backend . . . 51
12.4.1.1 Sample view . . . 51
12.4.1.2 Sample request . . . 51
12.4.1.3 Generated query . . . 52
12.4.2 Multi match search filter backend . . . 52
12.4.2.1 Sample view . . . 52
12.4.2.2 Sample request . . . 53
12.4.2.3 Generated query . . . 53
12.4.2.4 Options . . . 54
12.4.2.4.1 Type options . . . 54
12.4.2.4.2 Operator options . . . 54
12.4.3 Simple query string filter backend . . . 54
12.4.3.1 Sample view . . . 54
12.4.3.2 Sample request 1 . . . 55
12.4.3.3 Generated query 1 . . . 55
12.4.3.4 Sample request 2 . . . 56
12.4.3.5 Generated query 2 . . . 56
12.4.3.6 Sample request 3 . . . 56
12.4.3.7 Generated query 3 . . . 57
12.4.3.8 Options . . . 57
12.4.3.8.1 Default Operator options . . . 57
12.5 Basic usage examples . . . 58
12.5.1 Example app . . . 58
12.5.1.1 Sample models . . . 58
12.5.1.2 Sample document . . . 59
12.5.1.3 Sample serializer . . . 60
12.5.1.4 Sample view . . . 61
12.5.1.5 Usage example . . . 62
12.5.1.5.1 Sample queries . . . 62
12.5.1.5.1.1 Search . . . 62
12.5.1.5.1.2 Filtering . . . 63
12.5.1.5.1.3 Ordering . . . 64
12.6 Advanced usage examples . . . 64
12.6.1 Example app . . . 66
12.6.1.1 Sample models . . . 66
12.6.1.2 Sample document . . . 68
12.6.1.2.1 Index definition . . . 68
12.6.1.2.1.1 Settings . . . 69
12.6.1.2.1.2 Document index . . . 69
12.6.1.3 Sample serializer . . . 71
12.6.1.4 Sample view . . . 72
12.6.1.5 Usage example . . . 74
12.6.1.5.1 Search . . . 74
12.6.1.5.2 Filtering . . . 75

12.6.1.5.3 Ordering . . . 75
12.6.1.6 Ids filter . . . 76
12.6.1.7 Faceted search . . . 76
12.6.1.8 Post-filter . . . 77
12.6.1.8.1 Sample view . . . 77
12.6.1.8.2 Sample queries . . . 79
12.6.1.9 Geo-spatial features . . . 79
12.6.1.9.1 Filtering . . . 80
12.6.1.9.2 Ordering . . . 80
12.6.1.10 Suggestions . . . 80
12.6.1.10.1 Completion suggesters . . . 80
12.6.1.10.1.1 Document definition . . . 80
12.6.1.10.1.2 Serializer definition . . . 82
12.6.1.10.1.3 ViewSet definition . . . 82
12.6.1.10.1.4 Sample requests/responses . . . 84
12.6.1.10.1.5 Suggestions on Array/List field . . . 85
12.6.1.10.1.6 Sample requests/responses . . . 86
12.6.1.10.2 Term and Phrase suggestions . . . 86
12.6.1.10.2.1 Document definition . . . 86
12.6.1.10.2.2 ViewSet definition . . . 88
12.6.1.10.2.3 Sample requests/responses . . . 91
12.6.1.10.2.4 Completion . . . 92
12.6.1.10.2.5 Term . . . 93
12.6.1.10.2.6 Phrase . . . 94
12.6.1.11 Functional suggestions . . . 95
12.6.1.11.1 Document definition . . . 95
12.6.1.11.2 ViewSet definition . . . 96
12.6.1.12 Highlighting . . . 97
12.6.1.13 Pagination . . . 99
12.6.1.13.1 Page number pagination . . . 99
12.6.1.13.2 Limit/offset pagination . . . 100
12.6.1.13.3 Customisations . . . 100
12.7 Nested fields usage examples . . . 101
12.7.1 Example app . . . 102
12.7.1.1 Sample models . . . 102
12.7.1.2 Sample document . . . 106
12.7.1.2.1 Index definition . . . 106
12.7.1.2.1.1 Settings . . . 106
12.7.1.2.1.2 Document index . . . 107
12.7.1.3 Sample serializer . . . 109
12.7.1.4 Sample view . . . 110
12.7.1.5 Usage example . . . 112
12.7.1.5.1 Sample queries . . . 112
12.7.1.5.1.1 Search . . . 112
12.7.1.5.1.2 Nested filtering . . . 113
12.7.1.5.1.3 Nested search . . . 113
12.7.1.5.1.4 Sample models . . . 113
12.7.1.5.1.5 Sample document . . . 114
12.7.1.5.1.6 Sample view . . . 116
12.7.1.5.1.7 Sample request . . . 117
12.7.1.5.1.8 Filtering . . . 117
12.7.1.5.1.9 Ordering . . . 118
12.7.1.6 Suggestions . . . 118
12.7.1.7 Nested aggregations/facets . . . 118

12.8 More like this . . . 122
12.8.1 Usage example . . . 122
12.8.1.1 Sample view . . . 122
12.8.1.2 Sample request . . . 123
12.8.1.3 Generated query . . . 123
12.8.1.4 Options . . . 123
12.9 Release history and notes . . . 124
12.9.1 0.14 . . . 124
12.9.2 0.13.2 . . . 124
12.9.3 0.13.1 . . . 124
12.9.4 0.13 . . . 124
12.9.5 0.12 . . . 125
12.9.6 0.11 . . . 125
12.9.7 0.10 . . . 126
12.9.8 0.9 . . . 126
12.9.9 0.8.4 . . . 126
12.9.10 0.8.3 . . . 127
12.9.11 0.8.2 . . . 127
12.9.12 0.8.1 . . . 127
12.9.13 0.8 . . . 127
12.9.14 0.7.2 . . . 127
12.9.15 0.7.1 . . . 128
12.9.16 0.7 . . . 128
12.9.17 0.6.4 . . . 128
12.9.18 0.6.3 . . . 128
12.9.19 0.6.2 . . . 128
12.9.20 0.6.1 . . . 128
12.9.21 0.6 . . . 129
12.9.22 0.5.1 . . . 129
12.9.23 0.5 . . . 129
12.9.24 0.4.4 . . . 129
12.9.25 0.4.3 . . . 129
12.9.26 0.4.2 . . . 130
12.9.27 0.4.1 . . . 130
12.9.28 0.4 . . . 130
12.9.29 0.3.12 . . . 130
12.9.30 0.3.11 . . . 130
12.9.31 0.3.10 . . . 130
12.9.32 0.3.9 . . . 131
12.9.33 0.3.8 . . . 131
12.9.34 0.3.7 . . . 131
12.9.35 0.3.6 . . . 131
12.9.36 0.3.5 . . . 131
12.9.37 0.3.4 . . . 131
12.9.38 0.3.3 . . . 131
12.9.39 0.3.2 . . . 131
12.9.40 0.3.1 . . . 132
12.9.41 0.3 . . . 132
12.9.42 0.2.6 . . . 132
12.9.43 0.2.5 . . . 132
12.9.44 0.2.4 . . . 132
12.9.45 0.2.3 . . . 132
12.9.46 0.2.2 . . . 132
12.9.47 0.2.1 . . . 132

12.9.48 0.2 . . . 133
12.9.49 0.1.8 . . . 133
12.9.50 0.1.7 . . . 133
12.9.51 0.1.6 . . . 133
12.9.52 0.1.5 . . . 133
12.9.53 0.1.4 . . . 133
12.9.54 0.1.3 . . . 134
12.9.55 0.1.2 . . . 134
12.9.56 0.1.1 . . . 134
12.9.57 0.1 . . . 134
12.10 django_elasticsearch_dsl_drf package . . . 134
12.10.1 Subpackages . . . 134
12.10.1.1 django_elasticsearch_dsl_drf.fields package . . . 134
12.10.1.1.1 Submodules . . . 134
12.10.1.1.2 django_elasticsearch_dsl_drf.fields.common module . . . 134
12.10.1.1.3 django_elasticsearch_dsl_drf.fields.helpers module . . . 135
12.10.1.1.4 django_elasticsearch_dsl_drf.fields.nested_fields module . . . 136
12.10.1.1.5 Module contents . . . 138
12.10.1.2 django_elasticsearch_dsl_drf.filter_backends package . . . 141
12.10.1.2.1 Subpackages . . . 141
12.10.1.2.1.1 django_elasticsearch_dsl_drf.filter_backends.aggregations package 141
12.10.1.2.1.2 Submodules . . . 141
12.10.1.2.1.3 django_elasticsearch_dsl_drf.filter_backends.aggregations.bucket_aggregations module . . . 141
12.10.1.2.1.4 django_elasticsearch_dsl_drf.filter_backends.aggregations.metrics_aggregations module . . . 141
12.10.1.2.1.5 django_elasticsearch_dsl_drf.filter_backends.aggregations.pipeline_aggregations module . . . 141
12.10.1.2.1.6 Module contents . . . 141
12.10.1.2.1.7 django_elasticsearch_dsl_drf.filter_backends.filtering package . . 141
12.10.1.2.1.8 Submodules . . . 141
12.10.1.2.1.9 django_elasticsearch_dsl_drf.filter_backends.filtering.common module . . . 141
12.10.1.2.1.10 django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial module . . . 149
12.10.1.2.1.11 django_elasticsearch_dsl_drf.filter_backends.filtering.ids module . 153
12.10.1.2.1.12 django_elasticsearch_dsl_drf.filter_backends.filtering.nested module . . . 154
12.10.1.2.1.13 django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter module . . . 156
12.10.1.2.1.14 Module contents . . . 158
12.10.1.2.1.15 django_elasticsearch_dsl_drf.filter_backends.ordering package . . 172
12.10.1.2.1.16 Submodules . . . 172
12.10.1.2.1.17 django_elasticsearch_dsl_drf.filter_backends.ordering.common module . . . 172
12.10.1.2.1.18 django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial module . . . 174
12.10.1.2.1.19 Module contents . . . 176
12.10.1.2.1.20 django_elasticsearch_dsl_drf.filter_backends.suggester package . 180
12.10.1.2.1.21 Submodules . . . 180
12.10.1.2.1.22 django_elasticsearch_dsl_drf.filter_backends.suggester.functional module . . . 180
12.10.1.2.1.23 django_elasticsearch_dsl_drf.filter_backends.suggester.native module . . . 184

12.10.1.2.1.24 Module contents . . . 188
12.10.1.2.2 Submodules . . . 193
12.10.1.2.3 django_elasticsearch_dsl_drf.filter_backends.faceted_search module . . . 193
12.10.1.2.4 django_elasticsearch_dsl_drf.filter_backends.highlight module . . . 196
12.10.1.2.5 django_elasticsearch_dsl_drf.filter_backends.mixins module . . . 197
12.10.1.2.6 django_elasticsearch_dsl_drf.filter_backends.search module . . . 198
12.10.1.2.7 Module contents . . . 202
12.10.1.3 django_elasticsearch_dsl_drf.tests package . . . 202
12.10.1.3.1 Submodules . . . 202
12.10.1.3.2 django_elasticsearch_dsl_drf.tests.base module . . . 202
12.10.1.3.3 django_elasticsearch_dsl_drf.tests.data_mixins module . . . 203
12.10.1.3.4 django_elasticsearch_dsl_drf.tests.test_faceted_search module . . . 203
12.10.1.3.5 django_elasticsearch_dsl_drf.tests.test_filtering_common module . . . 203
12.10.1.3.6 django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial module . . . 206
12.10.1.3.7 django_elasticsearch_dsl_drf.tests.test_filtering_nested module . . . 207
12.10.1.3.8 django_elasticsearch_dsl_drf.tests.test_filtering_post_filter module . . . 209
12.10.1.3.9 django_elasticsearch_dsl_drf.tests.test_functional_suggesters module . . . 211
12.10.1.3.10 django_elasticsearch_dsl_drf.tests.test_helpers module . . . 212
12.10.1.3.11 django_elasticsearch_dsl_drf.tests.test_highlight module . . . 212
12.10.1.3.12 django_elasticsearch_dsl_drf.tests.test_ordering_common module . . . 212
12.10.1.3.13 django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial module . . . 213
12.10.1.3.14 django_elasticsearch_dsl_drf.tests.test_pagination module . . . 213
12.10.1.3.15 django_elasticsearch_dsl_drf.tests.test_search module . . . 214
12.10.1.3.16 django_elasticsearch_dsl_drf.tests.test_serializers module . . . 214
12.10.1.3.17 django_elasticsearch_dsl_drf.tests.test_suggesters module . . . 215
12.10.1.3.18 django_elasticsearch_dsl_drf.tests.test_views module . . . 215
12.10.1.3.19 django_elasticsearch_dsl_drf.tests.test_wrappers module . . . 216
12.10.1.3.20 Module contents . . . 216
12.10.2 Submodules . . . 229
12.10.3 django_elasticsearch_dsl_drf.analyzers module . . . 229
12.10.4 django_elasticsearch_dsl_drf.apps module . . . 229
12.10.5 django_elasticsearch_dsl_drf.compat module . . . 229
12.10.6 django_elasticsearch_dsl_drf.constants module . . . 230
12.10.7 django_elasticsearch_dsl_drf.helpers module . . . 230
12.10.8 django_elasticsearch_dsl_drf.pagination module . . . 231
12.10.9 django_elasticsearch_dsl_drf.serializers module . . . 232
12.10.10 django_elasticsearch_dsl_drf.utils module . . . 233
12.10.11 django_elasticsearch_dsl_drf.views module . . . 234
12.10.12 django_elasticsearch_dsl_drf.viewsets module . . . 234
12.10.13 django_elasticsearch_dsl_drf.wrappers module . . . 235
12.10.14 Module contents . . . 235

**13 Indices and tables** **237**

**Python Module Index** **239**

Integrate Elasticsearch DSL with Django REST framework in the shortest way possible, with least efforts possible.

Package provides views, serializers, filter backends, pagination and other handy add-ons.

You are expected to use django-elasticsearch-dsl for defining your Elasticsearch documents.

CHAPTER 1

# Prerequisites

- Django 1.8, 1.9, 1.10, 1.11, 2.0 and 2.1.
- Python 2.7, 3.4, 3.5, 3.6, 3.7
- Elasticsearch 2.x, 5.x, 6.x

CHAPTER 2

# Dependencies

**elasticsearch and elasticsearch-dsl**

Depending on your Elasticsearch version (either 2.x, 5.x or 6.x) you should use 2.x, 5.x or 6.x versions of the `elasticsearch` and `elasticsearch-dsl` packages accordingly.

**django-elasticsearch-dsl**

You are advised to use the latest version of django-elasticsearch-dsl.

**djangorestframework**

Initial version of this package was written for djangorestframework 3.6.2.

CHAPTER 3

# Documentation

Documentation is available on Read the Docs.

# CHAPTER 4

## Main features and highlights

- *Dynamic serializer for Documents*.
- *Search filter backend*.
- *Ordering filter backend*.
- *Filtering filter backend* (big variety of native- and functional- query lookups, such as `gt`, `gte`, `lt`, `lte`, `endswith`, `contains`, `wildcard`, `exists`, `exclude`, `isnull`, `range`, `in`, `prefix` (same as `startswith`), `term` and `terms` is implemented.
- *Geo-spatial filtering filter backend* (the following filters implemented: `geo_distance`, `geo_polygon` and `geo_bounding_box`).
- *Geo-spatial ordering filter backend* (the following filters implemented: `geo_distance`).
- *Faceted search filter backend*.
- *Post-filter filter backend*.
- *Nested filtering filter backend*.
- *Highlight backend*.
- *Suggester filter backend*.
- *Functional suggester filter backend*.
- *Pagination (Page number and limit/offset pagination)*.
- *Ids filter backend*.
- *Multi match search filter backend*.
- *Simple search query search filter backend*.
- *More-like-this support (detail action)*.

CHAPTER 5

# Installation

1. Install latest stable version from PyPI:

```
pip install django-elasticsearch-dsl-drf
```

   or latest stable version from GitHub:

```
pip install https://github.com/barseghyanartur/django-elasticsearch-dsl-drf/
↪archive/stable.tar.gz
```

   or latest stable version from BitBucket:

```
pip install https://bitbucket.org/barseghyanartur/django-elasticsearch-dsl-drf/
↪get/stable.tar.gz
```

2. Add `rest_framework`, `django_elasticsearch_dsl` and `django_elasticsearch_dsl_drf` to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    # REST framework
    'rest_framework',

    # Django Elasticsearch integration
    'django_elasticsearch_dsl',

    # Django REST framework Elasticsearch integration (this package)
    'django_elasticsearch_dsl_drf',
    # ...
)
```

CHAPTER 6

# Quick start

Perhaps the easiest way to get acquainted with `django-elasticsearch-dsl-drf` is to read the *quick start tutorial*.

See it as a guide of diving into integration of Elasticsearch with Django with very low knowledge entry level.

CHAPTER 7

# Testing

Project is covered with tests.

To test with all supported Python/Django versions type:

```
tox
```

To test against specific environment, type:

```
tox -e py37-django21
```

To test just your working environment type:

```
./runtests.py
```

To run a single test in your working environment type:

```
./runtests.py src/django_elasticsearch_dsl_drf/tests/test_filtering.py
```

Or:

```
./manage.py test django_elasticsearch_dsl_drf.tests.test_ordering
```

It's assumed that you have all the requirements installed. If not, first install the test requirements:

```
pip install -r examples/requirements/test.txt
```

CHAPTER 8

# Writing documentation

Keep the following hierarchy.

```
=====
title
=====

header
======

sub-header
----------

sub-sub-header
~~~~~~~~~~~~~~

sub-sub-sub-header
^^^^^^^^^^^^^^^^^^

sub-sub-sub-sub-header
++++++++++++++++++++++

sub-sub-sub-sub-sub-header
**************************
```

CHAPTER 9

# License

GPL 2.0/LGPL 2.1

CHAPTER 10

# Support

For any issues contact me at the e-mail given in the *Author* section.

CHAPTER 11

# Author

Artur Barseghyan <artur.barseghyan@gmail.com>

CHAPTER 12

# Project documentation

Contents:

**Table of Contents**


- *django-elasticsearch-dsl-drf*
  - *Prerequisites*
  - *Dependencies*
  - *Documentation*
  - *Main features and highlights*
  - *Installation*
  - *Quick start*
  - *Testing*
  - *Writing documentation*
  - *License*
  - *Support*
  - *Author*
  - *Project documentation*
  - *Indices and tables*


## 12.1 Installing Elasticsearch

For development and testing purposes, it's often handy to be able to quickly switch between different Elasticsearch versions. Since this packages supports 2.x, 5.x and 6.x branches, you could make use of the following boxes/containers

for development and testing.

For all containers/boxes mentioned below, no authentication is required (for Elasticsearch).

### 12.1.1 Docker

#### 12.1.1.1 5.x

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:5.5.3
docker run -p 9200:9200 -e "discovery.type=single-node" -e "xpack.security.
↪enabled=false" docker.elastic.co/elasticsearch/elasticsearch:5.5.3
```

#### 12.1.1.2 6.x

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:6.3.1
docker run -p 9200:9200 -e "discovery.type=single-node" -e "xpack.security.
↪enabled=false" docker.elastic.co/elasticsearch/elasticsearch:6.3.1
```

### 12.1.2 Vagrant

#### 12.1.2.1 2.x

```
./scripts/vagrant_start.sh
```

## 12.2 Quick start

The best way to get acquainted with `django-elasticsearch-dsl-drf`.

See it as a guide of diving into integration of Elasticsearch with Django with very low knowledge entry level.

Contents:

**Table of Contents**


- *Quick start*
  - *Installation*
  - *Example app*
    - *Sample models*
      - *Required imports*
      - *Book statuses*
      - *Publisher model*
      - *Author model*
      - *Tag model*

        - *Book model*
    - *Admin classes*
    - *Create database tables*
    - *Fill in some data*
    - *Sample document*
        - *Required imports*
        - *Index definition*
        - *Settings*
        - *Document index*
        - *Custom analyzers*
        - *Document definition*
    - *Syncing Django's database with Elasticsearch indexes*
        - *Full database sync*
        - *Sample partial sync (using custom signals)*
        - *Required imports*
        - *Update book index on related model change*
        - *Update book index on related model removal*
    - *Sample serializer*
        - *Required imports*
        - *Serializer definition*
    - *ViewSet definition*
        - *Required imports*
        - *ViewSet definition*
    - *URLs*
        - *Required imports*
        - *Router definition*
        - *URL patterns*
    - *Check what you've done so far*
        - *URLs*
        - *Test in browser*
- *Development and debugging*
    - *Profiling tools*
        - *Installation*
        - *Configuration*
    - *Debugging*

### 12.2.1 Installation

1. Install latest stable version from PyPI:

```
pip install django-elasticsearch-dsl-drf
```

2. Add `rest_framework`, `django_elasticsearch_dsl` and `django_elasticsearch_dsl_drf` to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    # REST framework
    'rest_framework',

    # Django Elasticsearch integration
    'django_elasticsearch_dsl',

    # Django REST framework Elasticsearch integration (this package)
    'django_elasticsearch_dsl_drf',
    # ...
)
```

3. Basic Django REST framework and `django-elasticsearch-dsl` configuration:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ),
    'DEFAULT_PAGINATION_CLASS':
        'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 100,
    'ORDERING_PARAM': 'ordering',
}

# Elasticsearch configuration
ELASTICSEARCH_DSL = {
    'default': {
        'hosts': 'localhost:9200'
    },
}
```

### 12.2.2 Example app

To get started, let's imagine we have a simple book register with a couple of models.

- *Publisher model*: The book publisher model. Each book might have only one publisher (`ForeignKey` relation).
- *Author model*: The book author model. Each book might have unlimited number of authors (`ManyToMany` relation).
- *Tag model*: The tag model. Each book might have unlimited number of tags (`ManyToMany` relation).
- *Book model*: The book model.

To keep things separate, our Django models will reside in the `books` app. Elasticsearch documents and Django REST framework views will be defined in a `search_indexes` app. Both of the apps should be added to the

INSTALLED_APPS.

```
INSTALLED_APPS = (
    # ...
    'books',  # Books application
    'search_indexes',  # Elasticsearch integration with the Django
                       # REST framework
    # ...
)
```

#### 12.2.2.1 Sample models

Content of the `books/models.py` file. Additionally, see the code comments.

##### 12.2.2.1.1 Required imports

Imports required for model definition.

*books/models/book.py*

```
import json

from django.conf import settings
from django.db import models
from django.utils.translation import ugettext, ugettext_lazy as _

from six import python_2_unicode_compatible
```

##### 12.2.2.1.2 Book statuses

*books/models/book.py*

```
# States indicate the publishing status of the book. Publishing might
# be in-progress, not yet published, published, rejected, etc.
BOOK_PUBLISHING_STATUS_PUBLISHED = 'published'
BOOK_PUBLISHING_STATUS_NOT_PUBLISHED = 'not_published'
BOOK_PUBLISHING_STATUS_IN_PROGRESS = 'in_progress'
BOOK_PUBLISHING_STATUS_CANCELLED = 'cancelled'
BOOK_PUBLISHING_STATUS_REJECTED = 'rejected'
BOOK_PUBLISHING_STATUS_CHOICES = (
    (BOOK_PUBLISHING_STATUS_PUBLISHED, "Published"),
    (BOOK_PUBLISHING_STATUS_NOT_PUBLISHED, "Not published"),
    (BOOK_PUBLISHING_STATUS_IN_PROGRESS, "In progress"),
    (BOOK_PUBLISHING_STATUS_CANCELLED, "Cancelled"),
    (BOOK_PUBLISHING_STATUS_REJECTED, "Rejected"),
)
BOOK_PUBLISHING_STATUS_DEFAULT = BOOK_PUBLISHING_STATUS_PUBLISHED
```

##### 12.2.2.1.3 Publisher model

*books/models/book.py*

```
@python_2_unicode_compatible
class Publisher(models.Model):
    """Publisher."""

    name = models.CharField(max_length=30)
    address = models.CharField(max_length=50)
    city = models.CharField(max_length=60)
    state_province = models.CharField(max_length=30)
    country = models.CharField(max_length=50)
    website = models.URLField()
    latitude = models.DecimalField(null=True,
                                   blank=True,
                                   decimal_places=15,
                                   max_digits=19,
                                   default=0)
    longitude = models.DecimalField(null=True,
                                    blank=True,
                                    decimal_places=15,
                                    max_digits=19,
                                    default=0)

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

#### 12.2.2.1.4 Author model

*books/models/author.py*

```
@python_2_unicode_compatible
class Author(models.Model):
    """Author."""

    salutation = models.CharField(max_length=10)
    name = models.CharField(max_length=200)
    email = models.EmailField()
    headshot = models.ImageField(upload_to='authors', null=True, blank=True)

    class Meta(object):
        """Meta options."""
```

(continues on next page)

(continued from previous page)

```
        ordering = ["id"]

    def __str__(self):
        return self.name
```

#### 12.2.2.1.5 Tag model

*books/models/tag.py*

```
class Tag(models.Model):
    """Simple tag model."""

    title = models.CharField(max_length=255, unique=True)

    class Meta(object):
        """Meta options."""

        verbose_name = _("Tag")
        verbose_name_plural = _("Tags")

    def __str__(self):
        return self.title
```

#### 12.2.2.1.6 Book model

*books/models/book.py*

```
@python_2_unicode_compatible
class Book(models.Model):
    """Book."""

    title = models.CharField(max_length=100)
    description = models.TextField(null=True, blank=True)
    summary = models.TextField(null=True, blank=True)
    authors = models.ManyToManyField('books.Author', related_name='books')
    publisher = models.ForeignKey(Publisher, related_name='books')
    publication_date = models.DateField()
    state = models.CharField(max_length=100,
                             choices=BOOK_PUBLISHING_STATUS_CHOICES,
                             default=BOOK_PUBLISHING_STATUS_DEFAULT)
    isbn = models.CharField(max_length=100, unique=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    pages = models.PositiveIntegerField(default=200)
    stock_count = models.PositiveIntegerField(default=30)
    tags = models.ManyToManyField('books.Tag',
                                  related_name='books',
                                  blank=True)

    class Meta(object):
        """Meta options."""

        ordering = ["isbn"]
```

(continues on next page)

(continued from previous page)

```
    def __str__(self):
        return self.title

    # The only publisher information we're going to need in our document
    # is the publisher name. Since publisher isn't a required field,
    # we define a properly on a model level to avoid indexing errors on
    # non-existing relation.
    @property
    def publisher_indexing(self):
        """Publisher for indexing.

        Used in Elasticsearch indexing.
        """
        if self.publisher is not None:
            return self.publisher.name

    # As of tags, again, we only need a flat list of tag names, on which
    # we can filter. Therefore, we define a properly on a model level,
    # which will return a JSON dumped list of tags relevant to the
    # current book model object.
    @property
    def tags_indexing(self):
        """Tags for indexing.

        Used in Elasticsearch indexing.
        """
        return [tag.title for tag in self.tags.all()]
```

#### 12.2.2.2 Admin classes

This is just trivial. A couple of correspondent admin classes in order to ba able to fill some data.

*books/admin.py*

```
from django.contrib import admin

from .models import *


@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    """Book admin."""

    list_display = ('title', 'isbn', 'price', 'publication_date')
    search_fields = ('title',)
    filter_horizontal = ('authors', 'tags',)


@admin.register(Author)
class AuthorAdmin(admin.ModelAdmin):
    """Author admin."""

    list_display = ('name', 'email',)
    search_fields = ('name',)
```

(continues on next page)

(continued from previous page)

```
@admin.register(Publisher)
class PublisherAdmin(admin.ModelAdmin):
    """Publisher admin."""

    list_display = ('name',)
    search_fields = ('name',)


@admin.register(Tag)
class TagAdmin(admin.ModelAdmin):
    """Tag admin."""

    list_display = ('title',)
    search_fields = ('title',)
```

#### 12.2.2.3 Create database tables

For now, just run the migrations to create the database tables.

```
./manage.py makemigrations books
./manage.py migrate books
```

#### 12.2.2.4 Fill in some data

If you have followed the instructions, you should now be able to log into the Django admin and create a dozen of Book/Author/Publisher/Tag records in admin.

```
http://localhost:8000/admin/books/publisher/
http://localhost:8000/admin/books/author/
http://localhost:8000/admin/books/tag/
http://localhost:8000/admin/books/book/
```

Once you've done that, proceed to the next step.

#### 12.2.2.5 Sample document

In Elasticsearch, a document is a basic unit of information that can be indexed. For example, you can have a document for a single customer, another document for a single product, and yet another for a single order. This document is expressed in JSON (JavaScript Object Notation) which is an ubiquitous internet data interchange format.

Within an index/type, you can store as many documents as you want. Note that although a document physically resides in an index, a document actually must be indexed/assigned to a type inside an index.

Simply said, you could see an Elasticsearch index as a database and a document as a database table (which makes a Document definition in Elasticsearch DSL similar to a Django Model definition).

Often, complex SQL model structures are flatterned in Elasticsearch indexes/documents. Nested relations are denormalized.

In our example, all 4 models (`Author`, `Publisher`, `Tag`, `Book`) would be flatterned into a single `BookDocument`, which would hold all the required information.

Content of the `search_indexes/documents/book.py` file. Additionally, see the code comments.

#### 12.2.2.5.1 Required imports

*search_indexes/documents/book.py*

```
from django.conf import settings
from django_elasticsearch_dsl import DocType, Index, fields
from elasticsearch_dsl import analyzer

from books.models import Book
```

#### 12.2.2.5.2 Index definition

To separate dev/test/staging/production indexes, the following approach is recommended.

##### 12.2.2.5.2.1 Settings

*settings/base.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'book',
    'search_indexes.documents.publisher': 'publisher',
}
```

*settings/testing.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'test_book',
    'search_indexes.documents.publisher': 'test_publisher',
}
```

*settings/production.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'prod_book',
    'search_indexes.documents.publisher': 'prod_publisher',
}
```

##### 12.2.2.5.2.2 Document index

*search_indexes/documents/books.py*

```
# Name of the Elasticsearch index
INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)
```

#### 12.2.2.5.3 Custom analyzers

```
html_strip = analyzer(
    'html_strip',
    tokenizer="standard",
    filter=["standard", "lowercase", "stop", "snowball"],
    char_filter=["html_strip"]
)
```

#### 12.2.2.5.4 Document definition

*search_indexes/documents/book.py*

```
@INDEX.doc_type
class BookDocument(DocType):
    """Book Elasticsearch document."""

    id = fields.IntegerField(attr='id')

    title = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    description = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    summary = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    publisher = fields.StringField(
        attr='publisher_indexing',
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    publication_date = fields.DateField()

    state = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
```

(continues on next page)

(continued from previous page)

```
    )

    isbn = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    price = fields.FloatField()

    pages = fields.IntegerField()

    stock_count = fields.IntegerField()

    tags = fields.StringField(
        attr='tags_indexing',
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword', multi=True),
            'suggest': fields.CompletionField(multi=True),
        },
        multi=True
    )

    class Meta(object):
        """Meta options."""

        model = Book  # The model associate with this DocType
```

#### 12.2.2.6 Syncing Django's database with Elasticsearch indexes

So far, we have a couple of Django models and a single (decentralized) Elasticsearch index/document (Book).

##### 12.2.2.6.1 Full database sync

The excellent `django-elasticsearch-dsl` library makes a good job of keeping the Book index fresh. It makes use of signals, so whenever the Book model is changed, the correspondent BookDocument indexes would be updated.

To simply run the full sync between Django's database and Elasticsearch, do as follows:

1. Create Elasticsearch indexes:

```
./manage.py search_index --create -f
```

2. Sync the data:

```
./manage.py search_index --populate -f
```

However, in case if a Tag, Publisher or Author models change, the Book index would not be automatically updated.

#### 12.2.2.6.2 Sample partial sync (using custom signals)

In order to keep indexes fresh, you will have to write a couple of simple lines of code (using Django's signals). Whenever a change is made to any of the Tag, Publisher or Author models, we're going to update the correspondent BookDocument index.

##### 12.2.2.6.2.1 Required imports

*search_indexes/signals.py*

```
from django.db.models.signals import post_save, post_delete
from django.dispatch import receiver

from django_elasticsearch_dsl.registries import registry
```

##### 12.2.2.6.2.2 Update book index on related model change

*search_indexes/signals.py*

```
@receiver(post_save)
def update_document(sender, **kwargs):
    """Update document on added/changed records.

    Update Book document index if related `books.Publisher` (`publisher`),
    `books.Author` (`authors`), `books.Tag` (`tags`) fields have been updated
    in the database.
    """
    app_label = sender._meta.app_label
    model_name = sender._meta.model_name
    instance = kwargs['instance']

    if app_label == 'book':
        # If it is `books.Publisher` that is being updated.
        if model_name == 'publisher':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)

        # If it is `books.Author` that is being updated.
        if model_name == 'author':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)

        # If it is `books.Tag` that is being updated.
        if model_name == 'tag':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)
```

##### 12.2.2.6.2.3 Update book index on related model removal

*search_indexes/signals.py*

```
@receiver(post_delete)
def delete_document(sender, **kwargs):
    """Update document on deleted records.

    Updates Book document from index if related `books.Publisher`
    (`publisher`), `books.Author` (`authors`), `books.Tag` (`tags`) fields
    have been removed from database.
    """
    app_label = sender._meta.app_label
    model_name = sender._meta.model_name
    instance = kwargs['instance']

    if app_label == 'books':
        # If it is `books.Publisher` that is being updated.
        if model_name == 'publisher':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)
                # registry.delete(_instance, raise_on_error=False)

        # If it is `books.Author` that is being updated.
        if model_name == 'author':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)
                # registry.delete(_instance, raise_on_error=False)

        # If it is `books.Tag` that is being updated.
        if model_name == 'tag':
            instances = instance.books.all()
            for _instance in instances:
                registry.update(_instance)
                # registry.delete(_instance, raise_on_error=False)
```

#### 12.2.2.7 Sample serializer

At this step we're going to define a serializer to be used in the Django REST framework ViewSet.

Content of the `search_indexes/serializers.py` file. Additionally, see the code comments.

##### 12.2.2.7.1 Required imports

*search_indexes/serializers/book.py*

```
import json

from rest_framework import serializers
from django_elasticsearch_dsl_drf.serializers import DocumentSerializer

from .documents import BookDocument
```

#### 12.2.2.7.2 Serializer definition

Simplest way to create a serializer, is to just specify which fields are needed to be serialized and leave it further to the dynamic serializer.

*search_indexes/serializers/book.py*

```
class BookDocumentSerializer(DocumentSerializer):
    """Serializer for the Book document."""

    class Meta(object):
        """Meta options."""

        # Specify the correspondent document class
        document = BookDocument

        # List the serializer fields. Note, that the order of the fields
        # is preserved in the ViewSet.
        fields = (
            'id',
            'title',
            'description',
            'summary',
            'publisher',
            'publication_date',
            'state',
            'isbn',
            'price',
            'pages',
            'stock_count',
            'tags',
        )
```

However, if dynamic serializer doesn't work for your or you want to customize too many things, you are free to use standard `Serializer` class of the Django REST framework.

*search_indexes/serializers/book.py*

```
class BookDocumentSerializer(serializers.Serializer):
    """Serializer for the Book document."""

    id = serializers.IntegerField(read_only=True)

    title = serializers.CharField(read_only=True)
    description = serializers.CharField(read_only=True)
    summary = serializers.CharField(read_only=True)

    publisher = serializers.CharField(read_only=True)
    publication_date = serializers.DateField(read_only=True)
    state = serializers.CharField(read_only=True)
    isbn = serializers.CharField(read_only=True)
    price = serializers.FloatField(read_only=True)
    pages = serializers.IntegerField(read_only=True)
    stock_count = serializers.IntegerField(read_only=True)
    tags = serializers.SerializerMethodField()

    class Meta(object):
        """Meta options."""
```

(continues on next page)

(continued from previous page)

```
        # List the serializer fields. Note, that the order of the fields
        # is preserved in the ViewSet.
        fields = (
            'id',
            'title',
            'description',
            'summary',
            'publisher',
            'publication_date',
            'state',
            'isbn',
            'price',
            'pages',
            'stock_count',
            'tags',
        )

    def get_tags(self, obj):
        """Get tags."""
        if obj.tags:
            return list(obj.tags)
        else:
            return []
```

#### 12.2.2.8 ViewSet definition

At this step, we're going to define Django REST framework ViewSets.

Content of the `search_indexes/viewsets.py` file. Additionally, see the code comments.

##### 12.2.2.8.1 Required imports

*search_indexes/viewsets/book.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_TERMS,
    LOOKUP_FILTER_RANGE,
    LOOKUP_FILTER_PREFIX,
    LOOKUP_FILTER_WILDCARD,
    LOOKUP_QUERY_IN,
    LOOKUP_QUERY_GT,
    LOOKUP_QUERY_GTE,
    LOOKUP_QUERY_LT,
    LOOKUP_QUERY_LTE,
    LOOKUP_QUERY_EXCLUDE,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    FilteringFilterBackend,
    IdsFilterBackend,
    OrderingFilterBackend,
    DefaultOrderingFilterBackend,
    SearchFilterBackend,
)
```

(continues on next page)

(continued from previous page)

```
from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet

from .documents import BookDocument, PublisherDocument
from .serializers import BookDocumentSerializer
```

#### 12.2.2.8.2 ViewSet definition

*search_indexes/viewsets/book.py*

```
class BookDocumentView(BaseDocumentViewSet):
    """The BookDocument view."""

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        IdsFilterBackend,
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
    ]
    # Define search fields
    search_fields = (
        'title',
        'description',
        'summary',
    )
    # Define filter fields
    filter_fields = {
        'id': {
            'field': 'id',
            # Note, that we limit the lookups of id field in this example,
            # to `range`, `in`, `gt`, `gte`, `lt` and `lte` filters.
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'title': 'title.raw',
        'publisher': 'publisher.raw',
        'publication_date': 'publication_date',
        'state': 'state.raw',
        'isbn': 'isbn.raw',
        'price': {
            'field': 'price.raw',
            # Note, that we limit the lookups of `price` field in this
            # example, to `range`, `gt`, `gte`, `lt` and `lte` filters.
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_GT,
```

(continues on next page)

(continued from previous page)

```
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'pages': {
            'field': 'pages',
            # Note, that we limit the lookups of `pages` field in this
            # example, to `range`, `gt`, `gte`, `lt` and `lte` filters.
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'stock_count': {
            'field': 'stock_count',
            # Note, that we limit the lookups of `stock_count` field in
            # this example, to `range`, `gt`, `gte`, `lt` and `lte`
            # filters.
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'tags': {
            'field': 'tags',
            # Note, that we limit the lookups of `tags` field in
            # this example, to `terms, `prefix`, `wildcard`, `in` and
            # `exclude` filters.
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
        'tags.raw': {
            'field': 'tags.raw',
            # Note, that we limit the lookups of `tags.raw` field in
            # this example, to `terms, `prefix`, `wildcard`, `in` and
            # `exclude` filters.
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
    }
```

(continues on next page)

(continued from previous page)

```
    # Define ordering fields
    ordering_fields = {
        'id': 'id',
        'title': 'title.raw',
        'price': 'price.raw',
        'state': 'state.raw',
        'publication_date': 'publication_date',
    }
    # Specify default ordering
    ordering = ('id', 'title', 'price',)
```

#### 12.2.2.9 URLs

At this step, we're going to define url patterns.

Content of the `search_indexes/urls.py` file. Additionally, see the code comments.

##### 12.2.2.9.1 Required imports

*search_indexes/urls.py*

```
from django.conf.urls import url, include
from rest_framework.routers import DefaultRouter

from .views import BookDocumentView
```

##### 12.2.2.9.2 Router definition

*search_indexes/urls.py*

```
router = DefaultRouter()
books = router.register(r'books',
                        BookDocumentView,
                        base_name='bookdocument')
```

##### 12.2.2.9.3 URL patterns

*search_indexes/urls.py*

```
urlpatterns = [
    url(r'^', include(router.urls)),
]
```

#### 12.2.2.10 Check what you've done so far

At this point, you are one step away from a working example of integrating Elasticsearch DSL with Django.

#### 12.2.2.10.1 URLs

If you didn't add the urls of the `search_indexes` example application to your project's global url patterns, make sure to do it now.

```
from django.conf.urls import include, url
from search_indexes import urls as search_index_urls

urlpatterns = [
    # ...
    # Search URLs
    url(r'^search/', include(search_index_urls)),
    # ...
]
```

#### 12.2.2.10.2 Test in browser

Open the following URL in your browser.

```
http://localhost:8000/search/books/
```

Perform a number of lookups:

```
http://localhost:8001/search/books/?ids=54|55|56
http://localhost:8001/search/books/?summary__contains=photography
http://localhost:8001/search/books/?tags__contains=ython
http://localhost:8001/search/books/?state=published
http://localhost:8001/search/books/?pages__gt=10&pages__lt=30
```

### 12.2.3 Development and debugging

#### 12.2.3.1 Profiling tools

Looking for profiling tools for Elasticsearch?

Try django-elasticsearch-debug-toolbar package. It's implemented as a panel for the well known Django Debug Toolbar and gives you full insights on what's happening on the side of Elasticsearch.

##### 12.2.3.1.1 Installation

```
pip install django-debug-toolbar
pip install django-elasticsearch-debug-toolbar
```

##### 12.2.3.1.2 Configuration

Change your development settings in the following way:

*settings/dev.py*

```
MIDDLEWARE_CLASSES += (
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    'debug_toolbar_force.middleware.ForceDebugToolbarMiddleware',
)

INSTALLED_APPS += (
    'debug_toolbar',
    'elastic_panel',
)

DEBUG_TOOLBAR_CONFIG = {
    'INTERCEPT_REDIRECTS': False,
}

DEBUG_TOOLBAR_PANELS = (
    # Defaults
    'debug_toolbar.panels.versions.VersionsPanel',
    'debug_toolbar.panels.timer.TimerPanel',
    'debug_toolbar.panels.settings.SettingsPanel',
    'debug_toolbar.panels.headers.HeadersPanel',
    'debug_toolbar.panels.request.RequestPanel',
    'debug_toolbar.panels.sql.SQLPanel',
    'debug_toolbar.panels.staticfiles.StaticFilesPanel',
    'debug_toolbar.panels.templates.TemplatesPanel',
    'debug_toolbar.panels.cache.CachePanel',
    'debug_toolbar.panels.signals.SignalsPanel',
    'debug_toolbar.panels.logging.LoggingPanel',
    'debug_toolbar.panels.redirects.RedirectsPanel',
    # Additional
    'elastic_panel.panel.ElasticDebugPanel',
)
```

#### 12.2.3.2 Debugging

Although (the unbeatable) Kibana is strongly recommended for data analyses, there are other good tools worth mentioning. One of them is elasticsearch-head Elasticsearch 2.x plugin or a correspondent Chrome extension of the same plugin. You may find it very useful for quick data preview or testing Elasticsearch queries.

## 12.3 Filter usage examples

Example usage of filtering backends.

Contents:

**Table of Contents**


- *Filter usage examples*
  - *Search*
    - *Search in all fields*
    - *Search a single term on specific field*

        * *Search for multiple terms*
        * *Search for multiple terms in specific fields*
    - *Filtering*
        * *Supported lookups*
            · *Native*
            · *term*
            · *terms*
            · *range*
            · *exists*
            · *prefix*
            · *wildcard*
            · *ids*
            · *Functional*
            · *contains*
            · *in*
            · *gt*
            · *gte*
            · *lt*
            · *lte*
            · *startswith*
            · *endswith*
            · *isnull*
            · *exclude*
    - *Usage examples*


### 12.3.1 Search

Query param name reserved for search is `search`. Make sure your models and documents do not have it as a field or attribute.

Multiple search terms are joined with `OR`.

Let's assume we have a number of Book items with fields `title`, `description` and `summary`.

#### 12.3.1.1 Search in all fields

Search in all fields (`name`, `address`, `city`, `state_province` and `country`) for word "reilly".

```
http://127.0.0.1:8080/search/publisher/?search=reilly
```

#### 12.3.1.2 Search a single term on specific field

In order to search in specific field (`name`) for term "reilly", add the field name separated with `:` to the search term.

```
http://127.0.0.1:8080/search/publisher/?search=name:reilly
```

#### 12.3.1.3 Search for multiple terms

In order to search for multiple terms "reilly", "bloomsbury" add multiple `search` query params.

```
http://127.0.0.1:8080/search/publisher/?search=reilly&search=bloomsbury
```

#### 12.3.1.4 Search for multiple terms in specific fields

In order to search for multiple terms "reilly", "bloomsbury" in specific fields add multiple `search` query params and field names separated with `:` to each of the search terms.

```
http://127.0.0.1:8080/search/publisher/?search=name:reilly&search=city:london
```

### 12.3.2 Filtering

#### 12.3.2.1 Supported lookups

##### 12.3.2.1.1 Native

The following native (to Elasticsearch) filters/lookups are implemented:

- *term*
- *terms*
- *range*
- *exists*
- *prefix*
- *wildcard*
- *regexp*
- *fuzzy*
- *type*
- *ids*

###### 12.3.2.1.1.1 term

Find documents which contain the exact term specified in the field specified.

```
http://127.0.0.1:8080/search/books/?tags__term=education&tags__term=economy
```

#### 12.3.2.1.1.2 terms

Find documents which contain any of the exact terms specified in the field specified. Note, that multiple values are separated with double underscores __.

```
http://localhost:8000/api/articles/?id=1&id=2&id=3
http://localhost:8000/api/articles/?id__terms=1__2__3
```

#### 12.3.2.1.1.3 range

Find documents where the field specified contains values (dates, numbers, or strings) in the range specified.

**From, to**

```
http://localhost:8000/api/users/?age__range=16__67
```

**From, to, boost**

```
http://localhost:8000/api/users/?age__range=16__67__2.0
```

#### 12.3.2.1.1.4 exists

Find documents where the field specified contains any non-null value.

```
http://localhost:8000/api/articles/?tags__exists=true
```

#### 12.3.2.1.1.5 prefix

Find documents where the field specified contains terms which begin with the exact prefix specified.

```
http://localhost:8000/api/articles/?tags__prefix=bio
```

#### 12.3.2.1.1.6 wildcard

Find documents where the field specified contains terms which match the pattern specified, where the pattern supports single character wildcards (?) and multi-character wildcards (*)

```
http://localhost:8000/api/articles/?title__wildcard=*elusional*
```

#### 12.3.2.1.1.7 ids

Find documents with the specified type and IDs.

```
http://localhost:8000/api/articles/?ids=68__64__58
http://localhost:8000/api/articles/?ids=68&ids=64&ids=58
```

#### 12.3.2.1.2 Functional

The following functional (non-native to Elasticsearch, but common in Django) filters/lookups are implemented:

- *contains*
- *in*
- *gt*
- *gte*
- *lt*
- *lte*
- *startswith*
- *endswith*
- *isnull*
- *exclude*

##### 12.3.2.1.2.1 contains

Case-insensitive containment test.

##### 12.3.2.1.2.2 in

In a given list.

```
http://localhost:8000/api/articles/?id__in=1__2__3
```

##### 12.3.2.1.2.3 gt

Greater than.

```
http://localhost:8000/api/users/?id__gt=10
```

##### 12.3.2.1.2.4 gte

Greater than or equal to.

```
http://localhost:8000/api/users/?id__gte=10
```

##### 12.3.2.1.2.5 lt

Less than.

```
http://localhost:8000/api/users/?id__lt=10
```

#### 12.3.2.1.2.6 lte

Less than or equal to.

```
http://localhost:8000/api/users/?id__lte=10
```

#### 12.3.2.1.2.7 startswith

Case-sensitive starts-with.

http://localhost:8000/api/articles/?tags__startswith=bio

#### 12.3.2.1.2.8 endswith

Case-sensitive ends-with.

```
http://localhost:8000/api/articles/?state__endswith=lished
```

#### 12.3.2.1.2.9 isnull

Takes either True or False.

**True**

```
http://localhost:8000/api/articles/?null_field__isnull=true
```

**False**

```
http://localhost:8000/api/articles/?tags__isnull=false
```

#### 12.3.2.1.2.10 exclude

Returns a new query set of containing objects that do not match the given lookup parameters.

```
http://localhost:8000/api/articles/?tags__exclude=children
http://localhost:8000/api/articles/?tags__exclude=children__python
```

### 12.3.3 Usage examples

See the example project for sample models/views/serializers.

- models
- documents
- serializers
- views

Additionally, see:

- Basic usage examples

- Advanced usage examples
- Misc usage examples

## 12.4 Search backends

### 12.4.1 Compound search filter backend

Compound search filter backend aims to replace old style *SearchFilterBackend*.

#### 12.4.1.1 Sample view

```
from django_elasticsearch_dsl_drf.filter_backends import (
    DefaultOrderingFilterBackend,
    CompoundSearchFilterBackend,
    OrderingFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from .documents import BookDocument
from .serializers import BookDocumentSerializer

class BookCompoundSearchBackendDocumentViewSet(DocumentViewSet):

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'

    filter_backends = [
        # ...
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        CompoundSearchFilterBackend,
        # ...
    ]

    multi_match_search_fields = (
        'title',
        'description',
        'summary',
    )

    ordering = ('_score', 'id', 'title', 'price',)
```

#### 12.4.1.2 Sample request

```
http://localhost:8000/search/books-compound-search-backend/?search=enim
```

#### 12.4.1.3 Generated query

```
{
  "from": 0,
  "sort": [
    "id",
    "title",
    "price"
  ],
  "size": 23,
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "title": {
              "query": "enim"
            }
          }
        },
        {
          "match": {
            "description": {
              "query": "enim"
            }
          }
        },
        {
          "match": {
            "summary": {
              "query": "enim"
            }
          }
        }
      ]
    }
  }
}
```

### 12.4.2 Multi match search filter backend

Document and serializer definition are trivial (there are lots of examples in other sections).

#### 12.4.2.1 Sample view

```
from django_elasticsearch_dsl_drf.filter_backends import (
    DefaultOrderingFilterBackend,
    MultiMatchSearchFilterBackend,
    OrderingFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from .documents import BookDocument
from .serializers import BookDocumentSerializer
```

(continues on next page)

(continued from previous page)

```
class BookMultiMatchSearchFilterBackendDocumentViewSet(DocumentViewSet):

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'

    filter_backends = [
        # ...
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        MultiMatchSearchFilterBackend,
        # ...
    ]

    multi_match_search_fields = {
        'title': {'boost': 4},
        'summary': {'boost': 2},
        'description': None,
    }

    ordering = ('_score', 'id', 'title', 'price',)
```

#### 12.4.2.2 Sample request

**Note:** Multiple search params (*search_multi_match*) are not supported. Even if you provide multiple search params, the first one would be picked, having the rest simply ignored.

```
http://localhost:8000/search/books-multi-match-search-backend/?search_multi_
↪match=debitis%20enim
```

#### 12.4.2.3 Generated query

```
{
  "from": 0,
  "query": {
    "multi_match": {
      "query": "debitis enim",
      "fields": [
        "summary^2",
        "description",
        "title^4"
      ]
    }
  },
  "size": 38,
  "sort": [
    "_score",
    "id",
    "title",
```

(continues on next page)

(continued from previous page)

```
    "price"
  ]
}
```

#### 12.4.2.4 Options

All standard multi match query options are available/tunable with help of `multi_match_options` view property.

Selective list of available options:

- operator
- type
- analyzer
- tie_breaker

##### 12.4.2.4.1 Type options

See the Elasticsearch docs for detailed explanation.

- best_fields
- most_fields
- cross_fields
- phrase
- phrase_prefix

**Example**

```
class BookMultiMatchSearchFilterBackendDocumentViewSet(DocumentViewSet):

    # ...

    multi_match_options = {
        'type': 'phrase'
    }
```

##### 12.4.2.4.2 Operator options

Can be either `and` or `or`.

### 12.4.3 Simple query string filter backend

Document and serializer definition are trivial (there are lots of examples in other sections).

#### 12.4.3.1 Sample view

```
from django_elasticsearch_dsl_drf.filter_backends import (
    DefaultOrderingFilterBackend,
    SimpleQueryStringSearchFilterBackend,
    OrderingFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from .documents import BookDocument
from .serializers import BookDocumentSerializer


class BookSimpleQueryStringSearchFilterBackendDocumentViewSet(DocumentViewSet):

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'

    filter_backends = [
        # ...
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SimpleQueryStringSearchFilterBackend,
        # ...
    ]

    simple_query_string_search_fields = {
        'title': {'boost': 4},
        'summary': {'boost': 2},
        'description': None,
    }

    ordering = ('_score', 'id', 'title', 'price',)
```

#### 12.4.3.2 Sample request 1

**Note:** Multiple search params (*search_simple_query_string*) are not supported. Even if you provide multiple search params, the first one would be picked, having the rest simply ignored.

```
http://localhost:8000/search/books-simple-query-string-search-backend/?search_simple_
↪query_string="chapter%20II"%20%2Bfender
```

#### 12.4.3.3 Generated query 1

```
{
  "query": {
    "simple_query_string": {
      "query": "\"chapter II\" +fender",
      "default_operator": "and",
      "fields": [
        "title",
        "description",
        "summary"
```

(continues on next page)

(continued from previous page)

```
      ]
    }
  },
  "sort": [
    "_score",
    "id",
    "title",
    "price"
  ],
  "from": 0,
  "size": 1
}
```

#### 12.4.3.4 Sample request 2

**Note:** Multiple search params (*search_simple_query_string*) are not supported. Even if you provide multiple search params, the first one would be picked, having the rest simply ignored.

```
http://localhost:8000/search/books-simple-query-string-search-backend/?search_simple_
↪query_string="chapter%20II"%20%2B(shutting%20|%20fender)
```

#### 12.4.3.5 Generated query 2

```
{
  "query": {
    "simple_query_string": {
      "query": "\"chapter II\" +(shutting | fender)",
      "default_operator": "and",
      "fields": [
        "title",
        "description",
        "summary"
      ]
    }
  },
  "sort": [
    "_score",
    "id",
    "title",
    "price"
  ],
  "from": 0,
  "size": 2
}
```

#### 12.4.3.6 Sample request 3

**Note:** Multiple search params (*search_simple_query_string*) are not supported. Even if you provide multiple search params, the first one would be picked, having the rest simply ignored.

```
http://localhost:8000/search/books-simple-query-string-search-backend/?search_simple_
↪query_string=%22Pool%20of%20Tears%22%20-considering
```

#### 12.4.3.7 Generated query 3

```
{
  "query": {
    "simple_query_string": {
      "query": "\"Pool of Tears\" -considering",
      "default_operator": "and",
      "fields": [
        "title",
        "description",
        "summary"
      ]
    }
  },
  "sort": [
    "_score",
    "id",
    "title",
    "price"
  ],
  "from": 0,
  "size": 1
}
```

#### 12.4.3.8 Options

All standard multi match query options are available/tunable with help of `simple_query_string_options` view property.

Selective list of available options:

- default_operator

##### 12.4.3.8.1 Default Operator options

Can be either `and` or `or`.

**Example**

```
class BookSimpleQueryStringSearchFilterBackendDocumentViewSet(DocumentViewSet):

    # ...

    simple_query_string_options = {
        "default_operator": "and",
    }
```

## 12.5 Basic usage examples

Basic Django REST framework integration example

See the example project for sample models/views/serializers.

- models
- documents
- serializers
- views

Contents:

**Table of Contents**


- *Basic usage examples*
  - *Example app*
    - *Sample models*
    - *Sample document*
    - *Sample serializer*
    - *Sample view*
    - *Usage example*
      - *Sample queries*
      - *Search*
      - *Filtering*
      - *Ordering*


### 12.5.1 Example app

#### 12.5.1.1 Sample models

*books/models/publisher.py*

```
class Publisher(models.Model):
    """Publisher."""

    name = models.CharField(max_length=30)
    address = models.CharField(max_length=50)
    city = models.CharField(max_length=60)
    state_province = models.CharField(max_length=30)
    country = models.CharField(max_length=50)
    website = models.URLField()
    latitude = models.DecimalField(null=True,
                                   blank=True,
                                   decimal_places=15,
                                   max_digits=19,
```

(continues on next page)

(continued from previous page)

```
                                  default=0)
    longitude = models.DecimalField(null=True,
                                    blank=True,
                                    decimal_places=15,
                                    max_digits=19,
                                    default=0)

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

#### 12.5.1.2 Sample document

*search_indexes/documents/publisher.py*

```
from django_elasticsearch_dsl import DocType, Index, fields
from elasticsearch_dsl import analyzer

from books.models import Publisher

# Name of the Elasticsearch index
PUBLISHER_INDEX = Index('publisher')
# See Elasticsearch Indices API reference for available settings
PUBLISHER_INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)


@PUBLISHER_INDEX.doc_type
class PublisherDocument(DocType):
    """Publisher Elasticsearch document."""

    id = fields.IntegerField(attr='id')

    name = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
```

(continues on next page)

(continued from previous page)

```
    info = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
    address = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
    city = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
    state_province = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
    country = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )
    website = fields.StringField()

    # Location
    location = fields.GeoPointField(attr='location_field_indexing')

    class Meta(object):
        """Meta options."""

        model = Publisher  # The model associate with this DocType
```

#### 12.5.1.3 Sample serializer

*search_indexes/serializers/book.py*

```
import json

from django_elasticsearch_dsl_drf.serializers import DocumentSerializer

class PublisherDocumentSerializer(DocumentSerializer):
    """Serializer for Publisher document."""

    location = serializers.SerializerMethodField()

    class Meta(object):
        """Meta options."""

        # Note, that since we're using a dynamic serializer,
        # we only have to declare fields that we want to be shown. If
        # somehow, dynamic serializer doesn't work for you, either extend
        # or declare your serializer explicitly.
```

(continues on next page)

(continued from previous page)

```
        fields = (
            'id',
            'name',
            'info',
            'address',
            'city',
            'state_province',
            'country',
            'website',
        )

    def get_location(self, obj):
    """Represent location value."""
    try:
        return obj.location.to_dict()
    except:
        return {}
```

#### 12.5.1.4 Sample view

*search_indexes/views/publisher.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_GEO_DISTANCE,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    FilteringFilterBackend,
    OrderingFilterBackend,
    SearchFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

# Example app models
from search_indexes.documents.publisher import PublisherDocument
from search_indxes.serializers import PublisherDocumentSerializer

class PublisherDocumentView(DocumentViewSet):
    """The PublisherDocument view."""

    document = PublisherDocument
    serializer_class = PublisherDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
    ]
    # Define search fields
    search_fields = (
        'name',
        'info',
        'address',
        'city',
        'state_province',
```

(continues on next page)

(continued from previous page)

```
        'country',
    )
    # Define filtering fields
    filter_fields = {
        'id': None,
        'name': 'name.raw',
        'city': 'city.raw',
        'state_province': 'state_province.raw',
        'country': 'country.raw',
    }
    # Define ordering fields
    ordering_fields = {
        'id': None,
        'name': None,
        'city': None,
        'country': None,
    }
    # Specify default ordering
    ordering = ('id', 'name',)
    # Define geo-spatial filtering fields
    geo_spatial_filter_fields = {
        'location': {
            'lookups': [
                LOOKUP_FILTER_GEO_DISTANCE,
            ],
        },
    }
```

#### 12.5.1.5 Usage example

Considering samples above, you should be able to perform the search, sorting and filtering actions described below.

##### 12.5.1.5.1 Sample queries

###### 12.5.1.5.1.1 Search

Query param name reserved for search is `search`. Make sure your models and documents do not have it as a field or attribute.

Multiple search terms are joined with `OR`.

Let's assume we have a number of Book items with fields `title`, `description` and `summary`.

**Search in all fields**

Search in all fields (`name`, `address`, `city`, `state_province` and `country`) for word "reilly".

```
http://127.0.0.1:8080/search/publisher/?search=reilly
```

**Search a single term on specific field**

In order to search in specific field (`name`) for term "reilly", add the field name separated with `:` to the search term.

```
http://127.0.0.1:8080/search/publisher/?search=name:reilly
```

**Search for multiple terms**

In order to search for multiple terms "reilly", "bloomsbury" add multiple `search` query params.

```
http://127.0.0.1:8080/search/publisher/?search=reilly&search=bloomsbury
```

**Search for multiple terms in specific fields**

In order to search for multiple terms "reilly", "bloomsbury" in specific fields add multiple `search` query params and field names separated with `:` to each of the search terms.

```
http://127.0.0.1:8080/search/publisher/?search=name:reilly&search=city:london
```

#### 12.5.1.5.1.2 Filtering

Let's assume we have a number of Publisher documents with in cities (Yerevan, Groningen, Amsterdam, London).

Multiple filter terms are joined with `AND`.

**Filter documents by single field**

Filter documents by field (`city`) "yerevan".

```
http://127.0.0.1:8080/search/publisher/?city=yerevan
```

**Filter documents by multiple fields**

Filter documents by `city` "Yerevan" and "Groningen".

```
http://127.0.0.1:8080/search/publisher/?city__in=yerevan__groningen
```

**Filter document by a single field**

Filter documents by (field `country`) "Armenia".

```
http://127.0.0.1:8080/search/publisher/?country=armenia
```

**Filter documents by multiple fields**

Filter documents by multiple fields (field `city`) "Yerevan" and "Amsterdam" with use of functional `in` query filter.

```
http://127.0.0.1:8080/search/publisher/?city__in=yerevan__amsterdam
```

You can achieve the same effect by specifying multiple filters (`city`) "Yerevan" and "Amsterdam". Note, that in this case multiple filter terms are joined with `OR`.

```
http://127.0.0.1:8080/search/publisher/?city=yerevan&city=amsterdam
```

If you want the same as above, but joined with `AND`, add `__term` to each lookup.

```
http://127.0.0.1:8080/search/publisher/?city__term=education&city__term=economy
```

**Filter documents by a word part of a single field**

Filter documents by a part word part in single field (`city`) "ondon".

```
http://127.0.0.1:8080/search/publisher/?city__wildcard=*ondon
```

**Geo-distance filtering**

Filter documents by radius of 100000km from the given location.

```
http://127.0.0.1:8000/search/publishers/?location__geo_distance=100000km__12.04__-63.
↪93
```

#### 12.5.1.5.1.3 Ordering

The − prefix means ordering should be descending.

**Order documents by field (ascending)**

Filter documents by field `city` (ascending).

```
http://127.0.0.1:8080/search/publisher/?search=country:armenia&ordering=city
```

**Order documents by field (descending)**

Filter documents by field `country` (descending).

```
http://127.0.0.1:8080/search/publisher/?ordering=-country
```

**Order documents by multiple fields**

If you want to order by multiple fields, use multiple ordering query params. In the example below, documents would be ordered first by field `country` (descending), then by field `city` (ascending).

```
http://127.0.0.1:8080/search/publisher/?ordering=-country&ordering=city
```

## 12.6 Advanced usage examples

Advanced Django REST framework integration examples.

See the example project for sample models/views/serializers.

- models
- documents
- serializers
- viewsets

Contents:

**Table of Contents**


- *Advanced usage examples*
  - *Example app*
    - *Sample models*
    - *Sample document*
      - *Index definition*

        - *Settings*
        - *Document index*
    - *Sample serializer*
    - *Sample view*
    - *Usage example*
        - *Search*
        - *Filtering*
        - *Ordering*
    - *Ids filter*
    - *Faceted search*
    - *Post-filter*
        - *Sample view*
        - *Sample queries*
    - *Geo-spatial features*
        - *Filtering*
        - *Ordering*
    - *Suggestions*
        - *Completion suggesters*
        - *Document definition*
        - *Serializer definition*
        - *ViewSet definition*
        - *Sample requests/responses*
        - *Suggestions on Array/List field*
        - *Sample requests/responses*
        - *Term and Phrase suggestions*
        - *Document definition*
        - *ViewSet definition*
        - *Sample requests/responses*
        - *Completion*
        - *Term*
        - *Phrase*
    - *Functional suggestions*
        - *Document definition*
        - *ViewSet definition*
    - *Highlighting*

        * *Pagination*
            - *Page number pagination*
            - *Limit/offset pagination*
            - *Customisations*


### 12.6.1 Example app

#### 12.6.1.1 Sample models

*books/models/publisher.py*

```
import json

from django.conf import settings
from django.db import models
from django.utils.translation import ugettext, ugettext_lazy as _

from six import python_2_unicode_compatible

BOOK_PUBLISHING_STATUS_PUBLISHED = 'published'
BOOK_PUBLISHING_STATUS_NOT_PUBLISHED = 'not_published'
BOOK_PUBLISHING_STATUS_IN_PROGRESS = 'in_progress'
BOOK_PUBLISHING_STATUS_CANCELLED = 'cancelled'
BOOK_PUBLISHING_STATUS_REJECTED = 'rejected'
BOOK_PUBLISHING_STATUS_CHOICES = (
    (BOOK_PUBLISHING_STATUS_PUBLISHED, "Published"),
    (BOOK_PUBLISHING_STATUS_NOT_PUBLISHED, "Not published"),
    (BOOK_PUBLISHING_STATUS_IN_PROGRESS, "In progress"),
    (BOOK_PUBLISHING_STATUS_CANCELLED, "Cancelled"),
    (BOOK_PUBLISHING_STATUS_REJECTED, "Rejected"),
)
BOOK_PUBLISHING_STATUS_DEFAULT = BOOK_PUBLISHING_STATUS_PUBLISHED


@python_2_unicode_compatible
class Publisher(models.Model):
    """Publisher."""

    name = models.CharField(max_length=30)
    info = models.TextField(null=True, blank=True)
    address = models.CharField(max_length=50)
    city = models.CharField(max_length=60)
    state_province = models.CharField(max_length=30)
    country = models.CharField(max_length=50)
    website = models.URLField()
    latitude = models.DecimalField(null=True,
                                   blank=True,
                                   decimal_places=15,
                                   max_digits=19,
                                   default=0)
    longitude = models.DecimalField(null=True,
                                    blank=True,
                                    decimal_places=15,
```

(continues on next page)

(continued from previous page)

```
                                    max_digits=19,
                                    default=0)

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

*books/models/author.py*

```
@python_2_unicode_compatible
class Author(models.Model):
    """Author."""

    salutation = models.CharField(max_length=10)
    name = models.CharField(max_length=200)
    email = models.EmailField()
    headshot = models.ImageField(upload_to='authors', null=True, blank=True)

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name
```

*books/models/tag.py*

```
class Tag(models.Model):
    """Simple tag model."""

    title = models.CharField(max_length=255, unique=True)

    class Meta(object):
        """Meta options."""

        verbose_name = _("Tag")
        verbose_name_plural = _("Tags")

    def __str__(self):
        return self.title
```

*books/models/book.py*

```
@python_2_unicode_compatible
class Book(models.Model):
    """Book."""

    title = models.CharField(max_length=100)
    description = models.TextField(null=True, blank=True)
    summary = models.TextField(null=True, blank=True)
    authors = models.ManyToManyField('books.Author', related_name='books')
    publisher = models.ForeignKey(Publisher, related_name='books')
    publication_date = models.DateField()
    state = models.CharField(max_length=100,
                             choices=BOOK_PUBLISHING_STATUS_CHOICES,
                             default=BOOK_PUBLISHING_STATUS_DEFAULT)
    isbn = models.CharField(max_length=100, unique=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    pages = models.PositiveIntegerField(default=200)
    stock_count = models.PositiveIntegerField(default=30)
    tags = models.ManyToManyField('books.Tag',
                                  related_name='books',
                                  blank=True)

    class Meta(object):
        """Meta options."""

        ordering = ["isbn"]

    def __str__(self):
        return self.title

    @property
    def publisher_indexing(self):
        """Publisher for indexing.

        Used in Elasticsearch indexing.
        """
        if self.publisher is not None:
            return self.publisher.name

    @property
    def tags_indexing(self):
        """Tags for indexing.

        Used in Elasticsearch indexing.
        """
        return [tag.title for tag in self.tags.all()]
```

#### 12.6.1.2 Sample document

##### 12.6.1.2.1 Index definition

To separate dev/test/staging/production indexes, the following approach is recommended.

#### 12.6.1.2.1.1 Settings

*settings/base.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'book',
    'search_indexes.documents.publisher': 'publisher',
}
```

*settings/testing.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'test_book',
    'search_indexes.documents.publisher': 'test_publisher',
}
```

*settings/production.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.book': 'prod_book',
    'search_indexes.documents.publisher': 'prod_publisher',
}
```

#### 12.6.1.2.1.2 Document index

*search_indexes/documents/book.py*

```
from django.conf import settings
from django_elasticsearch_dsl import DocType, Index, fields
from elasticsearch_dsl import analyzer

from books.models import Book

# Name of the Elasticsearch index
INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)

html_strip = analyzer(
    'html_strip',
    tokenizer="standard",
    filter=["standard", "lowercase", "stop", "snowball"],
    char_filter=["html_strip"]
)


@INDEX.doc_type
class BookDocument(DocType):
    """Book Elasticsearch document."""
```

(continues on next page)

(continued from previous page)

```
    id = fields.IntegerField(attr='id')

    title = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    description = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    summary = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    publisher = fields.StringField(
        attr='publisher_indexing',
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    publication_date = fields.DateField()

    state = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    isbn = fields.StringField(
        analyzer=html_strip,
        fields={
            'raw': fields.StringField(analyzer='keyword'),
        }
    )

    price = fields.FloatField()

    pages = fields.IntegerField()

    stock_count = fields.IntegerField()

    tags = fields.StringField(
        attr='tags_indexing',
        analyzer=html_strip,
```

(continues on next page)

(continued from previous page)

```
        fields={
            'raw': fields.StringField(analyzer='keyword', multi=True),
            'suggest': fields.CompletionField(multi=True),
        },
        multi=True
    )

    class Meta(object):
        """Meta options."""

        model = Book  # The model associate with this DocType
```

#### 12.6.1.3 Sample serializer

*search_indexes/serializers/tag.py*

```
import json

from rest_framework import serializers

class TagSerializer(serializers.Serializer):
    """Helper serializer for the Tag field of the Book document."""

    title = serializers.CharField()

    class Meta(object):
        """Meta options."""

        fields = ('title',)
        read_only_fields = ('title',)
```

*search_indexes/serializers/book.py*

```
class BookDocumentSerializer(serializers.Serializer):
    """Serializer for the Book document."""

    id = serializers.SerializerMethodField()

    title = serializers.CharField(read_only=True)
    description = serializers.CharField(read_only=True)
    summary = serializers.CharField(read_only=True)

    publisher = serializers.CharField(read_only=True)
    publication_date = serializers.DateField(read_only=True)
    state = serializers.CharField(read_only=True)
    isbn = serializers.CharField(read_only=True)
    price = serializers.FloatField(read_only=True)
    pages = serializers.IntegerField(read_only=True)
    stock_count = serializers.IntegerField(read_only=True)
    tags = serializers.SerializerMethodField()

    class Meta(object):
        """Meta options."""

        fields = (
```

(continues on next page)

(continued from previous page)

```
            'id',
            'title',
            'description',
            'summary',
            'publisher',
            'publication_date',
            'state',
            'isbn',
            'price',
            'pages',
            'stock_count',
            'tags',
        )
        read_only_fields = fields

    def get_tags(self, obj):
        """Get tags."""
        if obj.tags:
            return list(obj.tags)
        else:
            return []
```

#### 12.6.1.4 Sample view

*search_indexes/viewsets/book.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_TERMS,
    LOOKUP_FILTER_RANGE,
    LOOKUP_FILTER_PREFIX,
    LOOKUP_FILTER_WILDCARD,
    LOOKUP_QUERY_IN,
    LOOKUP_QUERY_EXCLUDE,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    FilteringFilterBackend,
    OrderingFilterBackend,
    DefaultOrderingFilterBackend,
    SearchFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

# Example app models
from search_indexes.documents.book import BookDocument
from search_indxes.serializers import BookDocumentSerializer


class BookDocumentView(DocumentViewSet):
    """The BookDocument view."""

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
```

(continues on next page)

(continued from previous page)

```
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
    ]
    # Define search fields
    search_fields = (
        'title',
        'summary',
        'description',
    )
    # Define filtering fields
    filter_fields = {
        'id': {
            'field': '_id',
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_IN,
            ],
        },
        'publisher': 'publisher.raw',
        'publication_date': 'publication_date',
        'isbn': 'isbn.raw',
        'tags': {
            'field': 'tags',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
        'tags.raw': {
            'field': 'tags.raw',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
    }
    # Define ordering fields
    ordering_fields = {
        'id': 'id',
        'title': 'title.raw',
        'price': 'price.raw',
        'state': 'state.raw',
        'publication_date': 'publication_date',
    }
    # Specify default ordering
    ordering = ('id', 'title',)
```

#### 12.6.1.5 Usage example

Considering samples above, you should be able to perform the search, sorting and filtering actions described below.

##### 12.6.1.5.1 Search

Query param name reserved for search is `search`. Make sure your models and documents do not have it as a field or attribute.

Multiple search terms are joined with `OR`.

Let's assume we have a number of Book items with fields `title`, `description` and `summary`.

**Search in all fields**

Search in all fields (`title`, `description` and `summary`) for word "education".

```
http://127.0.0.1:8080/search/books/?search=education
```

**Search a single term on specific field**

In order to search in specific field (`title`) for term "education", add the field name separated with `:` to the search term.

```
http://127.0.0.1:8080/search/books/?search=title:education
```

**Search for multiple terms**

In order to search for multiple terms "education", "technology" add multiple `search` query params.

```
http://127.0.0.1:8080/search/books/?search=education&search=technology
```

**Search for multiple terms on specific fields**

In order to search for multiple terms "education", "technology" in specific fields add multiple `search` query params and field names separated with `:` to each of the search terms.

```
http://127.0.0.1:8080/search/books/?search=title:education&search=summary:technology
```

**Search with boosting**

It's possible to boost search fields. In order to do that change the *search_fields* definition of the *DocumentViewSet* as follows:

```
class BookDocumentView(DocumentViewSet):
    """The BookDocument view."""

    # ...

    # Define search fields
    search_fields = {
        'title': {'boost': 4},
        'summary': {'boost': 2},
        'description': None,
    }

    # Order by `_score` first.
    ordering = ('_score', 'id', 'title', 'price',)
```

(continues on next page)

(continued from previous page)

```
    # ...
```

Note, that we are ordering results by *_score* first.

#### 12.6.1.5.2 Filtering

Let's assume we have a number of Book documents with the tags (education, politics, economy, biology, climate, environment, internet, technology).

Multiple filter terms are joined with `AND`.

**Filter documents by field**

Filter documents by field (`state`) "published".

```
http://127.0.0.1:8080/search/books/?state=published
```

**Filter documents by multiple fields**

Filter documents by field (`states`) "published" and "in_progress".

```
http://127.0.0.1:8080/search/books/?state__in=published__in_progress
```

**Filter document by a single field**

Filter documents by (field `tag`) "education".

```
http://127.0.0.1:8080/search/books/?tag=education
```

**Filter documents by multiple fields**

Filter documents by multiple fields (field `tags`) "education" and "economy" with use of functional `in` query filter.

```
http://127.0.0.1:8080/search/books/?tags__in=education__economy
```

You can achieve the same effect by specifying multiple fields (`tags`) "education" and "economy". Note, that in this case multiple filter terms are joined with `OR`.

```
http://127.0.0.1:8080/search/books/?tags=education&tags=economy
```

If you want the same as above, but joined with `AND`, add `__term` to each lookup.

```
http://127.0.0.1:8080/search/books/?tags__term=education&tags__term=economy
```

**Filter documents by a word part of a single field**

Filter documents by a part word part in single field (`tags`). Word part should match both "technology" and "biology".

```
http://127.0.0.1:8080/search/books/?tags__wildcard=*logy
```

#### 12.6.1.5.3 Ordering

The − prefix means ordering should be descending.

**Order documents by field (ascending)**

Order documents by field `price` (ascending).

```
http://127.0.0.1:8080/search/books/?search=title:lorem&ordering=price
```

**Order documents by field (descending)**

Order documents by field `price` (descending).

```
http://127.0.0.1:8080/search/books/?search=title:lorem&ordering=-price
```

**Order documents by multiple fields**

If you want to order by multiple fields, use multiple ordering query params. In the example below, documents would be ordered first by field `publication_date` (descending), then by field `price` (ascending).

```
http://127.0.0.1:8080/search/books/?search=title:lorem&ordering=-publication_date&
↪ordering=price
```

#### 12.6.1.6 Ids filter

Filters documents that only have the provided ids.

```
http://127.0.0.1:8000/api/articles/?ids=68__64__58
```

Or, alternatively:

```
http://127.0.0.1:8000/api/articles/?ids=68&ids=64&ids=58
```

#### 12.6.1.7 Faceted search

In order to add faceted search support, we would have to extend our view set in the following way:

*search_indexes/viewsets/book.py*

```
# ...

from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    FacetedSearchFilterBackend,
)

# ...

from elasticsearch_dsl import (
    DateHistogramFacet,
    RangeFacet,
    TermsFacet,
)

# ...

class BookDocumentView(DocumentViewSet):
    """The BookDocument view."""

    # ...
```

(continues on next page)

(continued from previous page)

```
    filter_backends = [
        # ...
        FacetedSearchFilterBackend,
    ]

    # ...

    faceted_search_fields = {
        'state': 'state.raw',  # By default, TermsFacet is used
        'publisher': {
            'field': 'publisher.raw',
            'facet': TermsFacet,  # But we can define it explicitly
            'enabled': True,
        },
        'publication_date': {
            'field': 'publication_date',
            'facet': DateHistogramFacet,
            'options': {
                'interval': 'year',
            }
        },
        'pages_count': {
            'field': 'pages',
            'facet': RangeFacet,
            'options': {
                'ranges': [
                    ("<10", (None, 10)),
                    ("11-20", (11, 20)),
                    ("20-50", (20, 50)),
                    (">50", (50, None)),
                ]
            }
        },
    }

    # ...
```

Note, that none of the facets is enabled by default, unless you explicitly specify it to be enabled. That means, that you will have to add a query string *facet={facet_field_name}* for each of the facets you want to see in results.

In the example below, we show results with faceted `state` and `pages_count` facets.

```
http://127.0.0.1:8000/search/books/?facet=state&facet=pages_count
```

#### 12.6.1.8 Post-filter

The *post_filter* is very similar to the common filter. The only difference is that it doesn't affect facets. So, whatever post-filters applied, the numbers in facets will remain intact.

##### 12.6.1.8.1 Sample view

**Note:** Note the `PostFilterFilteringFilterBackend` and `post_filter_fields` usage.

*search_indexes/viewsets/book.py*

```
# ...

from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    PostFilterFilteringFilterBackend,
)

# ...

class BookDocumentView(DocumentViewSet):
    """The BookDocument view."""

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
        PostFilterFilteringFilterBackend,
    ]
    # Define search fields
    search_fields = (
        'title',
        'summary',
        'description',
    )
    # Define filtering fields
    filter_fields = {
        'id': {
            'field': '_id',
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_IN,
            ],
        },
        'publisher': 'publisher.raw',
        'publication_date': 'publication_date',
        'isbn': 'isbn.raw',
        'tags': {
            'field': 'tags',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
        'tags.raw': {
            'field': 'tags.raw',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
```

(continues on next page)

(continued from previous page)

```
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
    }
    # Define post-filter filtering fields
    post_filter_fields = {
        'publisher_pf': 'publisher.raw',
        'isbn_pf': 'isbn.raw',
        'state_pf': 'state.raw',
        'tags_pf': {
            'field': 'tags',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
    }
    # Define ordering fields
    ordering_fields = {
        'id': 'id',
        'title': 'title.raw',
        'price': 'price.raw',
        'state': 'state.raw',
        'publication_date': 'publication_date',
    }
    # Specify default ordering
    ordering = ('id', 'title',)
```

#### 12.6.1.8.2 Sample queries

**Filter documents by field**

Filter documents by field (`state`) "published".

```
http://127.0.0.1:8080/search/books/?state_pf=published
```

**Filter documents by multiple fields**

Filter documents by field (`states`) "published" and "in_progress".

```
http://127.0.0.1:8080/search/books/?state_pf__in=published__in_progress
```

### 12.6.1.9 Geo-spatial features

For testing the boundaries the following online services might be helpful:

- geojson.io
- Bounding Box Tool

#### 12.6.1.9.1 Filtering

**Geo-distance filtering**

Filter documents by radius of 100000km from the given location.

```
http://localhost:8000/search/publishers/?location__geo_distance=100000km__12.04__-63.
↪93
```

**Geo-polygon filtering**

Filter documents that are located in the given polygon.

```
http://localhost:8000/search/publishers/?location__geo_polygon=40,-70__30,-80__20,-90
```

**Geo-bounding-box filtering**

Filter documents that are located in the given bounding box.

```
http://localhost:8000/search/publishers/?location__geo_bounding_box=44.87,40.07__43.
↪87,41.11
```

#### 12.6.1.9.2 Ordering

**Geo-distance ordering**

```
http://localhost:8000/search/publishers/?ordering=location__48.85__2.30__km__plane
```

### 12.6.1.10 Suggestions

The suggest feature suggests similar looking terms based on a provided text by using a suggester.

---

**Note:** The `SuggesterFilterBackend` filter backend can be used in the `suggest` custom view action/route only. Usages outside of the are `suggest` action/route are restricted.

---

There are three options available here: `term`, `phrase` and `completion`.

---

**Note:** Suggestion functionality is exclusive. Once you have queried the `SuggesterFilterBackend`, the latter will transform your current search query into suggestion search query (which is very different). Therefore, always add it as the very last filter backend.

---

#### 12.6.1.10.1 Completion suggesters

##### 12.6.1.10.1.1 Document definition

To make use of suggestions, you should properly index relevant fields of your documents using `fields.CompletionField`.

*search_indexes/documents/publisher.py*

```
from django.conf import settings

from django_elasticsearch_dsl import DocType, Index, fields

from books.models import Publisher

# Name of the Elasticsearch index
INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)


@INDEX.doc_type
class PublisherDocument(DocType):
    """Publisher Elasticsearch document."""

    id = fields.IntegerField(attr='id')

    name = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
            'suggest': fields.CompletionField(),
        }
    )

    info = fields.StringField()

    address = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword')
        }
    )

    city = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
            'suggest': fields.CompletionField(),
        }
    )

    state_province = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
            'suggest': fields.CompletionField(),
        }
    )

    country = fields.StringField(
        fields={
            'raw': fields.StringField(analyzer='keyword'),
            'suggest': fields.CompletionField(),
        }
    )
```

(continues on next page)

(continued from previous page)

```
    website = fields.StringField()

    # Location
    location = fields.GeoPointField(attr='location_field_indexing')

    class Meta(object):
        """Meta options."""

        model = Publisher  # The model associate with this DocType
```

After that the `name.suggest`, `city.suggest`, `state_province.suggest` and `country.suggest` fields would be available for suggestions feature.

#### 12.6.1.10.1.2 Serializer definition

This is how publisher serializer would look like.

*search_indexes/serializers/publisher.py*

```
import json

from django_elasticsearch_dsl_drf.serializers import DocumentSerializer

class PublisherDocumentSerializer(DocumentSerializer):
    """Serializer for Publisher document."""

    class Meta(object):
        """Meta options."""

        # Note, that since we're using a dynamic serializer,
        # we only have to declare fields that we want to be shown. If
        # somehow, dynamic serializer doesn't work for you, either extend
        # or declare your serializer explicitly.
        fields = (
            'id',
            'name',
            'info',
            'address',
            'city',
            'state_province',
            'country',
            'website',
        )
```

#### 12.6.1.10.1.3 ViewSet definition

In order to add suggestions support, we would have to extend our view set in the following way:

*search_indexes/viewsets/publisher.py*

```
# ...

from django_elasticsearch_dsl_drf.constants import SUGGESTER_COMPLETION
```

(continues on next page)

(continued from previous page)

```
from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    SuggesterFilterBackend,
)

# ...

class PublisherDocumentViewSet(DocumentViewSet):
    """The PublisherDocument view."""

    document = PublisherDocument

    # ...

    filter_backends = [
        # ...
        SuggesterFilterBackend,
    ]

    # ...

    # Suggester fields
    suggester_fields = {
        'name_suggest': {
            'field': 'name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'city_suggest': {
            'field': 'city.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'state_province_suggest': {
            'field': 'state_province.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'country_suggest': {
            'field': 'country.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
    }

    # Geo-spatial filtering fields
    geo_spatial_filter_fields = {
        'location': {
            'lookups': [
                LOOKUP_FILTER_GEO_DISTANCE,
            ],
        },
    }
```

In the example below, we show suggestion results (auto-completion) for `country` field.

#### 12.6.1.10.1.4 Sample requests/responses

Once you have extended your view set with `SuggesterFilterBackend` functionality, you can make use of the `suggest` custom action of your view set.

**Request**

```
GET http://127.0.0.1:8000/search/publishers/suggest/?country_suggest__completion=Ar
```

**Response**

```
{
    "_shards": {
        "failed": 0,
        "successful": 1,
        "total": 1
    },
    "country_suggest__completion": [
        {
            "options": [
                {
                    "score": 1.0,
                    "text": "Armenia"
                },
                {
                    "score": 1.0,
                    "text": "Argentina"
                }
            ],
            "offset": 0,
            "length": 2,
            "text": "Ar"
        }
    ]
}
```

You can also have multiple suggesters per request.

**Request**

```
GET http://127.0.0.1:8000/search/publishers/suggest/?name_suggest__completion=B&
↪country_suggest__completion=Ar
```

**Response**

```
{
    "_shards": {
        "successful": 1,
        "total": 1,
        "failed": 0
    },
    "country_suggest__completion": [
        {
            "text": "Ar",
            "options": [
```

(continues on next page)

(continued from previous page)


```
                {
                    "score": 1.0,
                    "text": "Armenia"
                },
                {
                    "score": 1.0,
                    "text": "Argentina"
                }
            ],
            "offset": 0,
            "length": 2
        }
    ],
    "name_suggest__completion": [
        {
            "text": "B",
            "options": [
                {
                    "score": 1.0,
                    "text": "Book Works"
                },
                {
                    "score": 1.0,
                    "text": "Brumleve LLC"
                },
                {
                    "score": 1.0,
                    "text": "Booktrope"
                },
                {
                    "score": 1.0,
                    "text": "Borman, Post and Wendt"
                },
                {
                    "score": 1.0,
                    "text": "Book League of America"
                }
            ],
            "offset": 0,
            "length": 1
        }
    ]
}
```


#### 12.6.1.10.1.5 Suggestions on Array/List field

Suggestions on Array/List fields (typical use case - tags, where Tag model would be a many-to-many relation to a Book model) work almost the same.

Before checking the *Sample requests/responses*, do have in mind the following:

- `Book` (see the *Sample models*)
- `BookSerializer` (see the *Sample serializer*)
- `BookDocumentView` (see the **'Sample view'_**)

#### 12.6.1.10.1.6 Sample requests/responses

Once you have extended your view set with `SuggesterFilterBackend` functionality, you can make use of the `suggest` custom action of your view set.

**Request**

```
GET http://127.0.0.1:8000/search/books/suggest/?tag_suggest__completion=bio
```

**Response**

```
{
    "_shards": {
        "failed": 0,
        "successful": 1,
        "total": 1
    },
    "country_suggest__completion": [
        {
            "options": [
                {
                    "score": 1.0,
                    "text": "Biography"
                },
                {
                    "score": 1.0,
                    "text": "Biology"
                }
            ],
            "offset": 0,
            "length": 2,
            "text": "bio"
        }
    ]
}
```

### 12.6.1.10.2 Term and Phrase suggestions

While for the `completion` suggesters to work the `CompletionField` shall be used, the `term` and `phrase` suggesters work on common text fields.

#### 12.6.1.10.2.1 Document definition

*search_indexes/documents/book.py*

```
from django.conf import settings

from django_elasticsearch_dsl import DocType, Index, fields

from books.models import Book

# Name of the Elasticsearch index
INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

```

(continues on next page)

(continued from previous page)

```
# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)

@INDEX.doc_type
class BookDocument(DocType):
    """Book Elasticsearch document."""
    # ID
    id = fields.IntegerField(attr='id')

    title = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
        }
    )

    description = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
        }
    )

    summary = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField()
        }
    )

    # Publisher
    publisher = StringField(
        attr='publisher_indexing',
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
        }
    )

    # Publication date
    publication_date = fields.DateField()

    # State
    state = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
        }
    )

    # ISBN
    isbn = StringField(
```

(continues on next page)

(continued from previous page)

```
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
        }
    )

    # Price
    price = fields.FloatField()

    # Pages
    pages = fields.IntegerField()

    # Stock count
    stock_count = fields.IntegerField()

    # Tags
    tags = StringField(
        attr='tags_indexing',
        analyzer=html_strip,
        fields={
            'raw': KeywordField(multi=True),
            'suggest': fields.CompletionField(multi=True),
        },
        multi=True
    )

    null_field = fields.StringField(attr='null_field_indexing')

    class Meta(object):
        """Meta options."""

        model = Book  # The model associate with this DocType
```

#### 12.6.1.10.2.2 ViewSet definition

**Note:** The suggester filter backends shall come as last ones.

Suggesters for the view are configured in `suggester_fields` property.

In the example below, the `title_suggest` is the name of the GET query param which points to the `title.suggest` field of the `BookDocument` document. For the `title_suggest` the allowed suggesters are `SUGGESTER_COMPLETION`, `SUGGESTER_TERM` and `SUGGESTER_PHRASE`.

URL shall be constructed in the following way:

```
/search/books/suggest/?{QUERY_PARAM}__{SUGGESTER_NAME}={VALUE}
```

Example for `completion` suggester:

```
GET http://127.0.0.1:8000/search/books/suggest/?title_suggest__completion=temp
```

However, since we have `default_suggester` defined we can skip the `__{SUGGESTER_NAME}` part (if we want `completion` suggester functionality). Thus, it might be written as short as:

```
GET http://127.0.0.1:8000/search/books/suggest/?title_suggest=temp
```

Example for `term` suggester:

```
GET http://127.0.0.1:8000/search/books/suggest/?title_suggest__term=tmeporus
```

Example for `phrase` suggester:

```
GET http://127.0.0.1:8000/search/books/suggest/?title_suggest__phrase=tmeporus
```

*search_indexes/viewsets/book.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_PREFIX,
    LOOKUP_FILTER_RANGE,
    LOOKUP_FILTER_TERMS,
    LOOKUP_FILTER_WILDCARD,
    LOOKUP_QUERY_EXCLUDE,
    LOOKUP_QUERY_GT,
    LOOKUP_QUERY_GTE,
    LOOKUP_QUERY_IN,
    LOOKUP_QUERY_IN,
    LOOKUP_QUERY_ISNULL,
    LOOKUP_QUERY_LT,
    LOOKUP_QUERY_LTE,
    SUGGESTER_COMPLETION,
    SUGGESTER_PHRASE,
    SUGGESTER_TERM,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    SuggesterFilterBackend,
)

class BookDocumentViewSet(DocumentViewSet):
    """The BookDocument view."""

    document = BookDocument
    # serializer_class = BookDocumentSerializer
    serializer_class = BookDocumentSimpleSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
        SuggesterFilterBackend,  # This should be the last backend
    ]
    # Define search fields
    search_fields = (
        'title',
        'description',
        'summary',
    )
    # Define filter fields
    filter_fields = {
        'id': {
```

(continues on next page)

(continued from previous page)

```
            'field': 'id',
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
                LOOKUP_FILTER_TERMS,
            ],
        },
        'title': 'title.raw',
        'publisher': 'publisher.raw',
        'publication_date': 'publication_date',
        'state': 'state.raw',
        'isbn': 'isbn.raw',
        'price': {
            'field': 'price.raw',
            'lookups': [
                LOOKUP_FILTER_RANGE,
            ],
        },
        'pages': {
            'field': 'pages',
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'stock_count': {
            # 'field': 'stock_count',
            'lookups': [
                LOOKUP_FILTER_RANGE,
                LOOKUP_QUERY_GT,
                LOOKUP_QUERY_GTE,
                LOOKUP_QUERY_LT,
                LOOKUP_QUERY_LTE,
            ],
        },
        'tags': {
            'field': 'tags',
            'lookups': [
                LOOKUP_FILTER_TERMS,
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
                LOOKUP_QUERY_ISNULL,
            ],
        },
        'tags.raw': {
            'field': 'tags.raw',
            'lookups': [
                LOOKUP_FILTER_TERMS,
```

(continues on next page)

(continued from previous page)

```
                LOOKUP_FILTER_PREFIX,
                LOOKUP_FILTER_WILDCARD,
                LOOKUP_QUERY_IN,
                LOOKUP_QUERY_EXCLUDE,
            ],
        },
        # This has been added to test `exists` filter.
        'non_existent_field': 'non_existent_field',
        # This has been added to test `isnull` filter.
        'null_field': 'null_field',
    }
    # Define ordering fields
    ordering_fields = {
        'id': 'id',
        'title': 'title.raw',
        'price': 'price.raw',
        'state': 'state.raw',
        'publication_date': 'publication_date',
    }
    # Specify default ordering
    ordering = ('id', 'title', 'price',)

    # Suggester fields
    suggester_fields = {
        'title_suggest': {
            'field': 'title.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
                SUGGESTER_TERM,
                SUGGESTER_PHRASE,
            ]
            'default_suggester': SUGGESTER_COMPLETION,
        },
        'publisher_suggest': 'publisher.suggest',
        'tag_suggest': 'tags.suggest',
        'summary_suggest': 'summary',
    }
```

#### 12.6.1.10.2.3 Sample requests/responses

Once you have extended your view set with `SuggesterFilterBackend` functionality, you can make use of the `suggest` custom action of your view set.

Let's considering, that one of our books has the following text in the summary:

```
Twas brillig, and the slithy toves
Did gyre and gimble in the wabe.
All mimsy were the borogoves
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"
```

(continues on next page)

(continued from previous page)

```
He took his vorpal sword in his hand,
Long time the manxome foe he sought --
So rested he by the Tumtum tree,
And stood awhile in thought.
```

#### 12.6.1.10.2.4 Completion

**Request**

```
GET http://127.0.0.1:8000/search/books/suggest/?title_suggest__completion=temp
```

**Response**


```
{
    "_shards": {
        "successful": 1,
        "total": 1,
        "failed": 0
    },
    "title_suggest": [
        {
            "length": 4,
            "text": "temp",
            "options": [
                {
                    "text": "Tempora voluptates distinctio facere ",
                    "_index": "book",
                    "_score": 1.0,
                    "_id": "1000087",
                    "_type": "book_document",
                    "_source": {
                        "description": null,
                        "summary": "Veniam dolores recusandae maxime laborum earum.",
                        "id": 1000087,
                        "state": "cancelled",
                        "authors": [
                            "Jayden van Luyssel",
                            "Yassin van Rooij",
                            "Florian van 't Erve",
                            "Mats van Nimwegen",
                            "Wessel Keltenie"
                        ],
                        "title": "Tempora voluptates distinctio facere."
                    }
                },
                {
                    "text": "Tempore sapiente repellat alias ad corrupti",
                    "_index": "book",
                    "_score": 1.0,
                    "_id": "29",
                    "_type": "book_document"
                    "_source": {
                        "description": null,
                        "summary": "Dolores minus architecto iure fugit qui sed.",
                        "id": 29,
```


(continues on next page)

(continued from previous page)


```
                        "state": "canelled",
                        "authors": [
                            "Wout van Northeim",
                            "Lenn van Vliet-Kuijpers",
                            "Tijs Mulder"
                        ],
                        "title": "Tempore sapiente repellat alias ad."
                    },

                },
                {
                    "text": "Temporibus exercitationem minus expedita",
                    "_index": "book",
                    "_score": 1.0,
                    "_id": "17",
                    "_type": "book_document",
                    "_source": {
                        "description": null,
                        "summary": "A laborum alias voluptates tenetur sapiente modi.
↪",
                        "id": 17,
                        "state": "canelled",
                        "authors": [
                            "Juliette Estey",
                            "Keano de Keijzer",
                            "Koen Scheffers",
                            "Florian van 't Erve",
                            "Tara Oversteeg",
                            "Mats van Nimwegen"
                        ],
                        "title": "Temporibus exercitationem minus expedita."
                    }
                }
            ],
            "offset": 0
        }
    ]
}
```


#### 12.6.1.10.2.5 Term

**Request**

```
GET http://127.0.0.1:8000/search/books/suggest/?summary_suggest__term=tovse
```

**Response**


```
{
    "_shards": {
        "failed": 0,
        "total": 1,
        "successful": 1
    },
    "summary_suggest__term": [
        {
```


(continues on next page)

(continued from previous page)


```
            "text": "tovs",
            "offset": 0,
            "options": [
                {
                    "text": "tove",
                    "score": 0.75,
                    "freq": 1
                },
                {
                    "text": "took",
                    "score": 0.5,
                    "freq": 1
                },
                {
                    "text": "twas",
                    "score": 0.5,
                    "freq": 1
                }
            ],
            "length": 5
        }
    ]
}
```


#### 12.6.1.10.2.6 Phrase

**Request**

```
GET http://127.0.0.1:8000/search/books/suggest/?summary_suggest__phrase=slith%20tovs
```

**Response**


```
{
    "summary_suggest__phrase": [
        {
            "text": "slith tovs",
            "offset": 0,
            "options": [
                {
                    "text": "slithi tov",
                    "score": 0.00083028956
                }
            ],
            "length": 10
        }
    ],
    "_shards": {
        "failed": 0,
        "total": 1,
        "successful": 1
    }
}
```

#### 12.6.1.11 Functional suggestions

If native suggestions are not good enough for you, use functional suggesters.

Configuration is very similar to native suggesters.

##### 12.6.1.11.1 Document definition

Obviously, different filters require different approaches. For instance, when using functional completion prefix filter, the best approach is to use keyword field of the Elasticsearch. While for match completion, Ngram fields work really well.

The following example indicates Ngram analyzer/filter usage.

*search_indexes/documents/book.py*

```
from django.conf import settings
from django_elasticsearch_dsl import DocType, Index, fields

from elasticsearch_dsl import analyzer
from elasticsearch_dsl.analysis import token_filter

from books.models import Book

edge_ngram_completion_filter = token_filter(
    'edge_ngram_completion_filter',
    type="edge_ngram",
    min_gram=1,
    max_gram=20
)


edge_ngram_completion = analyzer(
    "edge_ngram_completion",
    tokenizer="standard",
    filter=["lowercase", edge_ngram_completion_filter]
)

INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)

@INDEX.doc_type
class BookDocument(DocType):
    """Book Elasticsearch document."""

    # In different parts of the code different fields are used. There are
    # a couple of use cases: (1) more-like-this functionality, where `title`,
    # `description` and `summary` fields are used, (2) search and filtering
    # functionality where all of the fields are used.

    # ID
    id = fields.IntegerField(attr='id')
```

(continues on next page)

(continued from previous page)

```
    # ************************************************************************
    # ********************** Main data fields for search ***********************
    # ************************************************************************

    title = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
            'edge_ngram_completion': StringField(
                analyzer=edge_ngram_completion
            ),
        }
    )

    # ...

    class Meta(object):
        """Meta options."""

        model = Book  # The model associate with this DocType
```

#### 12.6.1.11.2 ViewSet definition

**Note:** The suggester filter backends shall come as last ones.

Functional suggesters for the view are configured in `functional_suggester_fields` property.

In the example below, the `title_suggest` is the name of the GET query param which points to the `title.raw` field of the `BookDocument` document. For the `title_suggest` the allowed suggester is `FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX`. For Ngram match we have the `title_suggest_match` field, which points to `title.edge_ngram_completion` field of the same document. For `title_suggest_match` the allowed suggester is `FUNCTIONAL_SUGGESTER_COMPLETION_MATCH`.

URL shall be constructed in the following way:

```
/search/books/functional_suggest/?{QUERY_PARAM}__{SUGGESTER_NAME}={VALUE}
```

Example for `completion_prefix` suggester:

```
GET http://localhost:8000/search/books/functional_suggest/?title_suggest_prefix__
↪completion_prefix=Temp
```

However, since we have `default_suggester` defined we can skip the `__{SUGGESTER_NAME}` part (if we want `completion_prefix` suggester functionality). Thus, it might be written as short as:

```
GET http://localhost:8000/search/books/functional_suggest/?title_suggest_prefix=Temp
```

Example for `completion_match` suggester:

```
GET http://localhost:8000/search/books/functional_suggest/?title_suggest_match__
↪completion_match=Temp
```

However, since we have `default_suggester` defined we can skip the `__{SUGGESTER_NAME}` part (if we want `completion_match` suggester functionality). Thus, it might be written as short as:

```
GET http://localhost:8000/search/books/functional_suggest/?title_suggest_match=Temp
```

*search_indexes/viewsets/book.py*

```
from django_elasticsearch_dsl_drf.constants import (
    # ...
    FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
    FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    SuggesterFilterBackend,
)

class BookDocumentViewSet(DocumentViewSet):
    """The BookDocument view."""

    document = BookDocument
    serializer_class = BookDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        IdsFilterBackend,
        OrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SearchFilterBackend,
        FacetedSearchFilterBackend,
        HighlightBackend,
        FunctionalSuggesterFilterBackend,  # This should come as last
    ]

    # ...

    # Functional suggester fields
    functional_suggester_fields = {
        'title_suggest': {
            'field': 'title.raw',
            'suggesters': [
                FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
            ],
            'default_suggester': FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
        },
        'title_suggest_match': {
            'field': 'title.edge_ngram_completion',
            'suggesters': [FUNCTIONAL_SUGGESTER_COMPLETION_MATCH],
            'default_suggester': FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
        }
    }
```

#### 12.6.1.12 Highlighting

Highlighters enable you to get highlighted snippets from one or more fields in your search results so you can show users where the query matches are.

**ViewSet definition**

```
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet
from django_elasticsearch_dsl_drf.filter_backends import (
    # ...
    HighlightBackend,
)

from ..documents import BookDocument
from ..serializers import BookDocumentSimpleSerializer


class BookDocumentViewSet(BaseDocumentViewSet):
"""The BookDocument view."""

    document = BookDocument
    # serializer_class = BookDocumentSerializer
    serializer_class = BookDocumentSimpleSerializer
    lookup_field = 'id'
    filter_backends = [
        # ...
        HighlightBackend,
    ]

    # ...

    # Define highlight fields
    highlight_fields = {
        'title': {
            'enabled': True,
            'options': {
                'pre_tags': ["<b>"],
                'post_tags': ["</b>"],
            }
        },
        'summary': {
            'options': {
                'fragment_size': 50,
                'number_of_fragments': 3
            }
        },
        'description': {},
    }

    # ...
```

**Request**

```
GET http://127.0.0.1:8000/search/books/?search=optimisation&highlight=title&
↪highlight=summary
```

**Response**

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "facets": {
        "_filter_publisher": {
```

(continues on next page)

(continued from previous page)


```
            "publisher": {
                "buckets": [
                    {
                        "key": "Self published",
                        "doc_count": 1
                    }
                ],
                "doc_count_error_upper_bound": 0,
                "sum_other_doc_count": 0
            },
            "doc_count": 1
        }
    },
    "results": [
        {
            "id": 999999,
            "title": "Performance optimisation",
            "description": null,
            "summary": "Ad animi adipisci libero facilis iure totam
                        impedit. Facilis maiores quae qui magnam dolores.
                        Veritatis quia amet porro voluptates iure quod
                        impedit. Dolor voluptatibus maiores at libero
                        magnam.",
            "authors": [
                "Artur Barseghyan"
            ],
            "publisher": "Self published",
            "publication_date": "1981-04-29",
            "state": "cancelled",
            "isbn": "978-1-7372176-0-2",
            "price": 40.51,
            "pages": 162,
            "stock_count": 30,
            "tags": [
                "Guide",
                "Poetry",
                "Fantasy"
            ],
            "highlight": {
                "title": [
                    "Performance <b>optimisation</b>"
                ]
            },
            "null_field": null
        }
    ]
}
```


#### 12.6.1.13 Pagination

##### 12.6.1.13.1 Page number pagination

By default, the `PageNumberPagination` class is used on all view sets which inherit from `DocumentViewSet`.

Example:

```
http://127.0.0.1:8000/search/books/?page=4
http://127.0.0.1:8000/search/books/?page=4&page_size=100
```

#### 12.6.1.13.2 Limit/offset pagination

In order to use a different `pagination_class`, for instance the `LimitOffsetPagination`, specify it explicitly in the view.

*search_indexes/viewsets/book.py*

```
# ...

from django_elasticsearch_dsl_drf.pagination import LimitOffsetPagination

# ...

class BookDocumentView(DocumentViewSet):
    """The BookDocument view."""

    # ...

    pagination_class = LimitOffsetPagination

    # ...
```

Example:

```
http://127.0.0.1:8000/search/books/?limit=100
http://127.0.0.1:8000/search/books/?offset=400&limit=100
```

#### 12.6.1.13.3 Customisations

If you want to add additional data to the paginated response, for instance, the page size, subclass the correspondent pagination class and add your modifications in the `get_paginated_response_context` method as follows:

```
from django_elasticsearch_dsl_drf.pagination import PageNumberPagination


class CustomPageNumberPagination(PageNumberPagination):
    """Custom page number pagination."""

    def get_paginated_response_context(self, data):
        __data = super(
            CustomPageNumberPagination,
            self
        ).get_paginated_response_context(data)
        __data.append(
            ('current_page', int(self.request.query_params.get('page', 1)))
        )
        __data.append(
            ('page_size', self.get_page_size(self.request))
        )

        return sorted(__data)
```

Same applies to the customisations of the `LimitOffsetPagination`.

## 12.7 Nested fields usage examples

Advanced Django REST framework integration examples with object/nested fields.

See the example project for sample models/views/serializers.

- models
- documents
- serializers
- viewsets

Contents:

**Table of Contents**


- *Nested fields usage examples*
  - *Example app*
    - *Sample models*
    - *Sample document*
      - *Index definition*
      - *Settings*
      - *Document index*
    - *Sample serializer*
    - *Sample view*
    - *Usage example*
      - *Sample queries*
      - *Search*
      - *Nested filtering*
      - *Nested search*
      - *Sample models*
      - *Sample document*
      - *Sample view*
      - *Sample request*
      - *Filtering*
      - *Ordering*
    - *Suggestions*
    - *Nested aggregations/facets*

### 12.7.1 Example app

#### 12.7.1.1 Sample models

*books/models/continent.py*

```
from django.db import models

from six import python_2_unicode_compatible

@python_2_unicode_compatible
class Continent(models.Model):
    """Continent."""

    name = models.CharField(max_length=255)
    info = models.TextField(null=True, blank=True)
    latitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )
    longitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

*books/models/country.py*

```
@python_2_unicode_compatible
class Country(models.Model):
    """Country."""

    name = models.CharField(max_length=255)
```

(continues on next page)

(continued from previous page)

```
    info = models.TextField(null=True, blank=True)
    continent = models.ForeignKey(
        'books.Continent',
        on_delete=models.CASCADE
    )
    latitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )
    longitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native
        filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

*books/models/city.py*

```
@python_2_unicode_compatible
class City(models.Model):
    """City."""

    name = models.CharField(max_length=255)
    info = models.TextField(null=True, blank=True)
    country = models.ForeignKey('books.Country', on_delete=models.CASCADE)
    latitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )
    longitude = models.DecimalField(
```

(continues on next page)

(continued from previous page)

```
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )

    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return self.name

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native
        filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }
```

*books/models/address.py*

```
from django.db import models
from django_elasticsearch_dsl_drf.wrappers import dict_to_obj

from six import python_2_unicode_compatible

@python_2_unicode_compatible
class Address(models.Model):
    """Address."""

    street = models.CharField(max_length=255)
    house_number = models.CharField(max_length=60)
    appendix = models.CharField(max_length=30, null=True, blank=True)
    zip_code = models.CharField(max_length=60)
    city = models.ForeignKey('books.City', on_delete=models.CASCADE)
    latitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )
    longitude = models.DecimalField(
        null=True,
        blank=True,
        decimal_places=15,
        max_digits=19,
        default=0
    )
```

(continues on next page)

(continued from previous page)

```
    class Meta(object):
        """Meta options."""

        ordering = ["id"]

    def __str__(self):
        return "{} {} {} {}".format(
            self.street,
            self.house_number,
            self.appendix,
            self.zip_code
        )

    @property
    def location_field_indexing(self):
        """Location for indexing.

        Used in Elasticsearch indexing/tests of `geo_distance` native
        filter.
        """
        return {
            'lat': self.latitude,
            'lon': self.longitude,
        }

    @property
    def country_indexing(self):
        """Country data (nested) for indexing.

        Example:

        >>> mapping = {
        >>>     'country': {
        >>>         'name': 'Netherlands',
        >>>         'city': {
        >>>             'name': 'Amsterdam',
        >>>         }
        >>>     }
        >>> }

        :return:
        """
        wrapper = dict_to_obj({
            'name': self.city.country.name,
            'city': {
                'name': self.city.name
            }
        })

        return wrapper

    @property
    def continent_indexing(self):
        """Continent data (nested) for indexing.

        Example:
```

(continues on next page)

(continued from previous page)

```
        >>> mapping = {
        >>>     'continent': {
        >>>         'name': 'Asia',
        >>>         'country': {
        >>>             'name': 'Netherlands',
        >>>             'city': {
        >>>                 'name': 'Amsterdam',
        >>>             }
        >>>         }
        >>>     }
        >>> }

        :return:
        """
        wrapper = dict_to_obj({
            'name': self.city.country.continent.name,
            'country': {
                'name': self.city.country.name,
                'city': {
                    'name': self.city.name,
                }
            }
        })

        return wrapper
```

#### 12.7.1.2 Sample document

##### 12.7.1.2.1 Index definition

To separate dev/test/staging/production indexes, the following approach is recommended.

###### 12.7.1.2.1.1 Settings

*settings/base.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.address': 'address',
}
```

*settings/testing.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
    'search_indexes.documents.address': 'test_address',
}
```

*settings/production.py*

```
# Name of the Elasticsearch index
ELASTICSEARCH_INDEX_NAMES = {
```

(continues on next page)

(continued from previous page)

```
    'search_indexes.documents.address': 'prod_address',
}
```

#### 12.7.1.2.1.2 Document index

*search_indexes/documents/address.py*

```
from django.conf import settings

from django_elasticsearch_dsl import DocType, Index, fields
from django_elasticsearch_dsl_drf.compat import KeywordField, StringField

from books.models import Address

from .analyzers import html_strip


INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)

@INDEX.doc_type
class AddressDocument(DocType):
    """Address Elasticsearch document."""

    # In different parts of the code different fields are used. There are
    # a couple of use cases: (1) more-like-this functionality, where `title`,
    # `description` and `summary` fields are used, (2) search and filtering
    # functionality where all of the fields are used.

    # ID
    id = fields.IntegerField(attr='id')

    # ************************************************************************
    # ********************** Main data fields for search *********************
    # ************************************************************************

    street = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
        }
    )

    house_number = StringField(analyzer=html_strip)

    appendix = StringField(analyzer=html_strip)

    zip_code = StringField(
        analyzer=html_strip,
```

(continues on next page)

(continued from previous page)

```
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
        }
    )

    # ************************************************************
    # ********** Additional fields for search and filtering **********
    # ************************************************************

    # City object
    city = fields.ObjectField(
        properties={
            'name': StringField(
                analyzer=html_strip,
                fields={
                    'raw': KeywordField(),
                    'suggest': fields.CompletionField(),
                }
            ),
            'info': StringField(analyzer=html_strip),
            'location': fields.GeoPointField(attr='location_field_indexing'),
            'country': fields.ObjectField(
                properties={
                    'name': StringField(
                        analyzer=html_strip,
                        fields={
                            'raw': KeywordField(),
                            'suggest': fields.CompletionField(),
                        }
                    ),
                    'info': StringField(analyzer=html_strip),
                    'location': fields.GeoPointField(
                        attr='location_field_indexing'
                    )
                }
            )
        }
    )

    # Country object
    country = fields.NestedField(
        attr='country_indexing',
        properties={
            'name': StringField(
                analyzer=html_strip,
                fields={
                    'raw': KeywordField(),
                    'suggest': fields.CompletionField(),
                }
            ),
            'city': fields.ObjectField(
                properties={
                    'name': StringField(
                        analyzer=html_strip,
                        fields={
                            'raw': KeywordField(),
```

(continues on next page)

(continued from previous page)

```
                        },
                    ),
                },
            ),
        },
    )

    # Continent object
    continent = fields.NestedField(
        attr='continent_indexing',
        properties={
            'name': StringField(
                analyzer=html_strip,
                fields={
                    'raw': KeywordField(),
                    'suggest': fields.CompletionField(),
                }
            ),
            'country': fields.NestedField(
                properties={
                    'name': StringField(
                        analyzer=html_strip,
                        fields={
                            'raw': KeywordField(),
                        }
                    ),
                    'city': fields.NestedField(
                        properties={
                            'name': StringField(
                                analyzer=html_strip,
                                fields={
                                    'raw': KeywordField(),
                                }
                            )
                        }
                    )
                }
            )
        }
    )

    location = fields.GeoPointField(attr='location_field_indexing')

    class Meta(object):
        """Meta options."""

        model = Address  # The model associate with this DocType
```

#### 12.7.1.3 Sample serializer

*search_indexes/serializers/address.py*

```
from django_elasticsearch_dsl_drf.serializers import DocumentSerializer

from ..documents import AddressDocument
```

(continues on next page)

(continued from previous page)

```
class AddressDocumentSerializer(DocumentSerializer):
    """Serializer for address document."""

    class Meta(object):
        """Meta options."""

        document = AddressDocument
        fields = (
            'id',
            'street',
            'house_number',
            'appendix',
            'zip_code',
            'city',
            'country',
            'continent',
            'location',
        )
```

#### 12.7.1.4 Sample view

*search_indexes/viewsets/address.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_GEO_DISTANCE,
    LOOKUP_FILTER_GEO_POLYGON,
    LOOKUP_FILTER_GEO_BOUNDING_BOX,
    SUGGESTER_COMPLETION,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    DefaultOrderingFilterBackend,
    FacetedSearchFilterBackend,
    FilteringFilterBackend,
    GeoSpatialFilteringFilterBackend,
    GeoSpatialOrderingFilterBackend,
    NestedFilteringFilterBackend,
    OrderingFilterBackend,
    SearchFilterBackend,
    SuggesterFilterBackend,
)
from django_elasticsearch_dsl_drf.pagination import LimitOffsetPagination
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from ..documents import AddressDocument
from ..serializers import AddressDocumentSerializer

class AddressDocumentViewSet(DocumentViewSet):
    """The AddressDocument view."""

    document = AddressDocument
    serializer_class = AddressDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
```

(continues on next page)

(continued from previous page)

```
        FacetedSearchFilterBackend,
        FilteringFilterBackend,
        OrderingFilterBackend,
        SearchFilterBackend,
        GeoSpatialFilteringFilterBackend,
        GeoSpatialOrderingFilterBackend,
        NestedFilteringFilterBackend,
        DefaultOrderingFilterBackend,
        SuggesterFilterBackend,
    ]
    pagination_class = LimitOffsetPagination
    # Define search fields
    search_fields = (
        'street',
        'zip_code',
        'city.name',
        'city.country.name',
    )
    # Define filtering fields
    filter_fields = {
        'id': None,
        'city': 'city.name.raw',
    }
    # Nested filtering fields
    nested_filter_fields = {
        'continent_country': {
            'field': 'continent.country.name.raw',
            'path': 'continent.country',
        },
        'continent_country_city': {
            'field': 'continent.country.city.name.raw',
            'path': 'continent.country.city',
        },
    }
    # Define geo-spatial filtering fields
    geo_spatial_filter_fields = {
        'location': {
            'lookups': [
                LOOKUP_FILTER_GEO_BOUNDING_BOX,
                LOOKUP_FILTER_GEO_DISTANCE,
                LOOKUP_FILTER_GEO_POLYGON,

            ],
        },
    }
    # Define ordering fields
    ordering_fields = {
        'id': None,
        'street': None,
        'city': 'city.name.raw',
        'country': 'city.country.name.raw',
        'zip_code': None,
    }
    # Define ordering fields
    geo_spatial_ordering_fields = {
        'location': None,
    }
```

(continues on next page)

(continued from previous page)

```
    # Specify default ordering
    ordering = (
        'id',
        'street.raw',
        'city.name.raw',
    )
    # Suggester fields
    suggester_fields = {
        'street_suggest': {
            'field': 'street.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'city_suggest': {
            'field': 'city.name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'country_suggest': {
            'field': 'city.country.name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        }
    }

    # Facets
    faceted_search_fields = {
        'city': {
            'field': 'city.name.raw',
            'enabled': True,
        },
        'country': {
            'field': 'city.country.name.raw',
            'enabled': True,
        },
    }
```

#### 12.7.1.5 Usage example

Considering samples above, you should be able to perform the search, sorting and filtering actions described below.

##### 12.7.1.5.1 Sample queries

###### 12.7.1.5.1.1 Search

Just a couple of examples, because searching in nested fields doesn't differ from searching in simple fields.

**Search in all fields**

Search in all fields (`street`, `zip_code` and `city`, `country`) for word "Picadilly".

```
http://127.0.0.1:8000/search/addresses/?search=Piccadilly
```

**Search a single term on specific field**

In order to search in specific field (`country`) for term "Armenia", add the field name separated with `|` to the search term.

```
http://127.0.0.1:8000/search/addresses/?search=city.country.name:Armenia
```

#### 12.7.1.5.1.2 Nested filtering

**Filter documents by nested field**

Filter documents by field (`continent.country`) "Armenia".

```
http://127.0.0.1:8000/search/addresses/?continent_country=Armenia
```

Filter documents by field (`continent.country.city`) "Amsterdam".

```
http://127.0.0.1:8000/search/addresses/?continent_country_city=Amsterdam
```

#### 12.7.1.5.1.3 Nested search

For nested search, let's have another example.

#### 12.7.1.5.1.4 Sample models

*books/models/city.py*

```
from django.db import models
from six import python_2_unicode_compatible

@python_2_unicode_compatible
class City(models.Model):
    """City."""

    name = models.CharField(max_length=255)
    info = models.TextField(null=True, blank=True)
    country = models.ForeignKey('books.Country')
    latitude = models.DecimalField(null=True,
                                   blank=True,
                                   decimal_places=15,
                                   max_digits=19,
                                   default=0)
    longitude = models.DecimalField(null=True,
                                    blank=True,
                                    decimal_places=15,
                                    max_digits=19,
                                    default=0)
```

*books/models/country.py*

```
from django.db import models
from six import python_2_unicode_compatible

@python_2_unicode_compatible
class Country(models.Model):
    """Country."""

    name = models.CharField(max_length=255)
    info = models.TextField(null=True, blank=True)
    latitude = models.DecimalField(null=True,
                                   blank=True,
                                   decimal_places=15,
                                   max_digits=19,
                                   default=0)
    longitude = models.DecimalField(null=True,
                                    blank=True,
                                    decimal_places=15,
                                    max_digits=19,
                                    default=0)
```

#### 12.7.1.5.1.5 Sample document

*documents/city.py*

```
from django.conf import settings

from django_elasticsearch_dsl import DocType, Index, fields
from django_elasticsearch_dsl_drf.compat import KeywordField, StringField

from books.models import City

from .analyzers import html_strip

INDEX = Index(settings.ELASTICSEARCH_INDEX_NAMES[__name__])

# See Elasticsearch Indices API reference for available settings
INDEX.settings(
    number_of_shards=1,
    number_of_replicas=1
)


@INDEX.doc_type
class CityDocument(DocType):
    """City Elasticsearch document.

    This document has been created purely for testing out complex fields.
    """

    # ID
    id = fields.IntegerField(attr='id')

    # ****************************************************************
    # ******************** Main data fields for search ***************
    # ****************************************************************
```

(continues on next page)

(continued from previous page)

```
    name = StringField(
        analyzer=html_strip,
        fields={
            'raw': KeywordField(),
            'suggest': fields.CompletionField(),
        }
    )

    info = StringField(analyzer=html_strip)

    # ********************************************************************
    # ************** Nested fields for search and filtering **************
    # ********************************************************************

    # City object
    country = fields.NestedField(
        properties={
            'name': StringField(
                analyzer=html_strip,
                fields={
                    'raw': KeywordField(),
                    'suggest': fields.CompletionField(),
                }
            ),
            'info': StringField(analyzer=html_strip),
            'location': fields.GeoPointField(attr='location_field_indexing'),
        }
    )

    location = fields.GeoPointField(attr='location_field_indexing')

    # ********************************************************************
    # ********** Other complex fields for search and filtering ***********
    # ********************************************************************

    boolean_list = fields.ListField(
        StringField(attr='boolean_list_indexing')
    )

    datetime_list = fields.ListField(
        StringField(attr='datetime_list_indexing')
    )
    float_list = fields.ListField(
        StringField(attr='float_list_indexing')
    )
    integer_list = fields.ListField(
        StringField(attr='integer_list_indexing')
    )

    class Meta(object):
        """Meta options."""

        model = City  # The model associate with this DocType
```

#### 12.7.1.5.1.6 Sample view

*viewsets/city.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_GEO_DISTANCE,
    LOOKUP_FILTER_GEO_POLYGON,
    LOOKUP_FILTER_GEO_BOUNDING_BOX,
    SUGGESTER_COMPLETION,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    FilteringFilterBackend,
    DefaultOrderingFilterBackend,
    OrderingFilterBackend,
    SearchFilterBackend,
    SuggesterFilterBackend,
    GeoSpatialFilteringFilterBackend,
    GeoSpatialOrderingFilterBackend,
)
from django_elasticsearch_dsl_drf.pagination import LimitOffsetPagination
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from ..documents import CityDocument
from ..serializers import CityDocumentSerializer

class CityDocumentViewSet(BaseDocumentViewSet):
    """The CityDocument view."""

    document = CityDocument
    serializer_class = CityDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FilteringFilterBackend,
        OrderingFilterBackend,
        SearchFilterBackend,
        GeoSpatialFilteringFilterBackend,
        GeoSpatialOrderingFilterBackend,
        DefaultOrderingFilterBackend,
        SuggesterFilterBackend,
    ]
    pagination_class = LimitOffsetPagination
    # Define search fields
    search_fields = (
        'name',
        'info',
    )

    search_nested_fields = {
        'country': ['name'],
    }

    # Define filtering fields
    filter_fields = {
        'id': None,
        'name': 'name.raw',
        'country': 'country.name.raw',
    }
```

(continues on next page)

(continued from previous page)

```
    # Define geo-spatial filtering fields
    geo_spatial_filter_fields = {
        'location': {
            'lookups': [
                LOOKUP_FILTER_GEO_BOUNDING_BOX,
                LOOKUP_FILTER_GEO_DISTANCE,
                LOOKUP_FILTER_GEO_POLYGON,

            ],
        },
    }
    # Define ordering fields
    ordering_fields = {
        'id': None,
        'name': None,
        'country': 'country.name.raw',
    }
    # Define ordering fields
    geo_spatial_ordering_fields = {
        'location': None,
    }
    # Specify default ordering
    ordering = (
        'id',
        'name.raw',
        'country.name.raw',
    )

    # Suggester fields
    suggester_fields = {
        'name_suggest': {
            'field': 'name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'country_suggest': {
            'field': 'country.name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        }
    }
```

#### 12.7.1.5.1.7 Sample request

**Request**

```
GET http://127.0.0.1:8000/search/cities/?search=Switzerland
```

#### 12.7.1.5.1.8 Filtering

**Filter documents by field**

Filter documents by field (`city`) "Dublin".

```
http://127.0.0.1:8000/search/addresses/?city=Dublin
```

**Filter documents by multiple fields**

Filter documents by field (`states`) "published" and "in_progress".

```
http://127.0.0.1:8000/search/addresses/?city__in=Yerevan__Dublin
```

#### 12.7.1.5.1.9 Ordering

The − prefix means ordering should be descending.

**Order documents by field (descending)**

Order documents by field `country` (ascending).

```
http://127.0.0.1:8000/search/addresses/?ordering=-country
```

### 12.7.1.6 Suggestions

The suggest feature suggests similar looking terms based on a provided text by using a suggester.

---

**Note:** The `SuggesterFilterBackend` filter backend can be used in the `suggest` custom view action/route only. Usages outside of the are `suggest` action/route are restricted.

---

There are three options available here: `term`, `phrase` and `completion`.

---

**Note:** Suggestion functionality is exclusive. Once you have queried the `SuggesterFilterBackend`, the latter will transform your current search query into suggestion search query (which is very different). Therefore, always add it as the very last filter backend.

---

Suggest completion for field `country`.

```
http://127.0.0.1:8000/search/addresses/suggest/?country_suggest__completion=Ar
```

Suggest completion for field `city`.

```
http://127.0.0.1:8000/search/addresses/suggest/?city_suggest__completion=Ye
```

### 12.7.1.7 Nested aggregations/facets

At the moment, nested aggregations/facets are not supported out of the box. Out of the box support will surely land in the package one day, but for now, there's a simple and convenient way of implementing nested aggregations/facets with minimal efforts. Consider the following example.

*search_indexes/backends/nested_continents.py*

```
from django_elasticsearch_dsl_drf.filter_backends.mixins import (
    FilterBackendMixin,
)
from rest_framework.filters import BaseFilterBackend

class NestedContinentsBackend(BaseFilterBackend, FilterBackendMixin):
    """Adds nesting to continents."""

    faceted_search_param = 'nested_facet'

    def get_faceted_search_query_params(self, request):
        """Get faceted search query params.

        :param request: Django REST framework request.
        :type request: rest_framework.request.Request
        :return: List of search query params.
        :rtype: list
        """
        query_params = request.query_params.copy()
        return query_params.getlist(self.faceted_search_param, [])

    def filter_queryset(self, request, queryset, view):
        """Filter the queryset.
        :param request: Django REST framework request.
        :param queryset: Base queryset.
        :param view: View.
        :type request: rest_framework.request.Request
        :type queryset: elasticsearch_dsl.search.Search
        :type view: rest_framework.viewsets.ReadOnlyModelViewSet
        :return: Updated queryset.
        :rtype: elasticsearch_dsl.search.Search
        """
        facets = self.get_faceted_search_query_params(request)

        if 'continent' in facets:
            queryset \
                .aggs\
                .bucket('continents',
                        'nested',
                        path='continent') \
                .bucket('continent_name',
                        'terms',
                        field='continent.name.raw',
                        size=10) \
                .bucket('counties',
                        'nested',
                        path='continent.country') \
                .bucket('country_name',
                        'terms',
                        field='continent.country.name.raw',
                        size=10) \
                .bucket('city',
                        'nested',
                        path='continent.country.city') \
                .bucket('city_name',
                        'terms',
                        field='continent.country.city.name.raw',
```

(continues on next page)

(continued from previous page)

```
                        size=10)

        return queryset
```

The view will look as follows:

*search_indexes/viewsets/address.py*

```
from django_elasticsearch_dsl_drf.constants import (
    LOOKUP_FILTER_GEO_DISTANCE,
    LOOKUP_FILTER_GEO_POLYGON,
    LOOKUP_FILTER_GEO_BOUNDING_BOX,
    SUGGESTER_COMPLETION,
)
from django_elasticsearch_dsl_drf.filter_backends import (
    DefaultOrderingFilterBackend,
    FacetedSearchFilterBackend,
    FilteringFilterBackend,
    GeoSpatialFilteringFilterBackend,
    GeoSpatialOrderingFilterBackend,
    NestedFilteringFilterBackend,
    OrderingFilterBackend,
    SearchFilterBackend,
    SuggesterFilterBackend,
)
from django_elasticsearch_dsl_drf.pagination import LimitOffsetPagination
from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet

from ..backends import NestedContinentsBackend
from ..documents import AddressDocument
from ..serializers import AddressDocumentSerializer

class AddressDocumentViewSet(DocumentViewSet):
    """The AddressDocument view."""

    document = AddressDocument
    serializer_class = AddressDocumentSerializer
    lookup_field = 'id'
    filter_backends = [
        FacetedSearchFilterBackend,
        FilteringFilterBackend,
        OrderingFilterBackend,
        SearchFilterBackend,
        GeoSpatialFilteringFilterBackend,
        GeoSpatialOrderingFilterBackend,
        NestedContinentsBackend,
        NestedFilteringFilterBackend,
        DefaultOrderingFilterBackend,
        SuggesterFilterBackend,
    ]
    pagination_class = LimitOffsetPagination
    # Define search fields
    search_fields = (
        'street',
        'zip_code',
        'city.name',
        'city.country.name',
```

(continues on next page)

(continued from previous page)

```
    )
    # Define filtering fields
    filter_fields = {
        'id': None,
        'city': 'city.name.raw',
    }
    # Nested filtering fields
    nested_filter_fields = {
        'continent_country': {
            'field': 'continent.country.name.raw',
            'path': 'continent.country',
        },
        'continent_country_city': {
            'field': 'continent.country.city.name.raw',
            'path': 'continent.country.city',
        },
    }
    # Define geo-spatial filtering fields
    geo_spatial_filter_fields = {
        'location': {
            'lookups': [
                LOOKUP_FILTER_GEO_BOUNDING_BOX,
                LOOKUP_FILTER_GEO_DISTANCE,
                LOOKUP_FILTER_GEO_POLYGON,

            ],
        },
    }
    # Define ordering fields
    ordering_fields = {
        'id': None,
        'street': None,
        'city': 'city.name.raw',
        'country': 'city.country.name.raw',
        'zip_code': None,
    }
    # Define ordering fields
    geo_spatial_ordering_fields = {
        'location': None,
    }
    # Specify default ordering
    ordering = (
        'id',
        'street.raw',
        'city.name.raw',
    )
    # Suggester fields
    suggester_fields = {
        'street_suggest': {
            'field': 'street.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        },
        'city_suggest': {
            'field': 'city.name.suggest',
            'suggesters': [
```

(continues on next page)

(continued from previous page)

```
                SUGGESTER_COMPLETION,
            ],
        },
        'country_suggest': {
            'field': 'city.country.name.suggest',
            'suggesters': [
                SUGGESTER_COMPLETION,
            ],
        }
    }

    # Facets
    faceted_search_fields = {
        'city': {
            'field': 'city.name.raw',
            'enabled': True,
        },
        'country': {
            'field': 'city.country.name.raw',
            'enabled': True,
        },
    }
```

## 12.8 More like this

More like this functionality.

### 12.8.1 Usage example

#### 12.8.1.1 Sample view

```
from django_elasticsearch_dsl_drf.filter_backends import (
    FilteringFilterBackend,
    IdsFilterBackend,
    OrderingFilterBackend,
    PostFilterFilteringFilterBackend,
    SearchFilterBackend,
)
from django_elasticsearch_dsl_drf.viewsets import (
    DocumentViewSet,
    MoreLikeThisMixin,
)

class BookMoreLikeThisDocumentViewSet(DocumentViewSet,
                                      MoreLikeThisMixin):
    """Same as BookDocumentViewSet, with more-like-this and no facets."""

    filter_backends = [
        # ...
        FilteringFilterBackend,
        PostFilterFilteringFilterBackend,
        IdsFilterBackend,
```

(continues on next page)

(continued from previous page)

```
        OrderingFilterBackend,
        SearchFilterBackend,
        # ...
    ]

    # More-like-this options
    more_like_this_options = {
        'fields': (
            'title.mlt',
            'summary.mlt',
            'description.mlt',
        )
    }
```

#### 12.8.1.2 Sample request

```
http://localhost:8000/search/books-more-like-this-no-options/1007587/more_like_this/
```

#### 12.8.1.3 Generated query

```
{
  "query": {
    "more_like_this": {
      "fields": [
        "title.mlt",
        "summary.mlt",
        "description.mlt"
      ],
      "like": {
        "_index": "book",
        "_id": "1007587",
        "_type": "book_document"
      }
    }
  },
  "from": 0,
  "size": 14,
  "sort": [
    "_score"
  ]
}
```

#### 12.8.1.4 Options

Pretty much all more-like-this options available in Elasticsearch. You might be particularly interested in the following:

- min_term_freq
- max_query_terms
- unlike
- stop_words

## 12.9 Release history and notes

Sequence based identifiers are used for versioning (schema follows below):

```
major.minor[.revision]
```

- It's always safe to upgrade within the same minor version (for example, from 0.3 to 0.3.4).
- Minor version changes might be backwards incompatible. Read the release notes carefully before upgrading (for example, when upgrading from 0.3.4 to 0.4).
- All backwards incompatible changes are mentioned in this document.

### 12.9.1 0.14

2018-08-06

- More like this support through detail action.

### 12.9.2 0.13.2

2018-08-03

- Successfully tested against Python 3.7 and Django 2.1.
- Unified the base `BaseSearchFilterBackend` class.
- Minor clean up and fixes in docs.
- Upgrading test suite to modern versions (`pytest`, `tox`, `factory_boy`, `Faker`). Removing unused dependencies from requirements (`drf-extensions`).
- Fixed missing PDF generation in offline documentation (non ReadTheDocs). The `rst2pdf` package (which does not support Python 3) has been replaced with `rinohtype` package (which does support Python 3).

### 12.9.3 0.13.1

2018-07-26

- Minor fix in suggesters on Elasticsearch 6.x.

### 12.9.4 0.13

2018-07-23

---

**Note:** Release dedicated to Guido van Rossum, the former Python BDFL, who resigned from his BDFL position recently. Guido knew it better than we all do. His charisma, talent and leadership will be certainly missed a lot by the community. Thumbs up again for the best BDFL ever.

---

- The `SimpleQueryStringSearchFilterBackend` backend has been implemented.
- Minor fixes in the `MultiMatchSearchFilterBackend` backend.

### 12.9.5 0.12

2018-07-21

- New-style Search Filter Backends. Old style `SearchFilterBackend` is still supported (until at least version 0.16), but is deprecated. Migrate to `CompoundSearchFilterBackend`. `MultiMatchSearchFilterBackend` introduced (the name speaks for itself).
- From now on, your views would also work with model- and object-level permissions of the Django REST Framework (such as `DjangoModelPermissions`, `DjangoModelPermissionsOrAnonReadOnly` and `DjangoObjectPermissions`). Correspondent model or object would be used for that. If you find it incorrect in your case, write custom permissions and declare the explicitly in your view-sets.
- Fixed geo-spatial `geo_distance` ordering for Elastic 5.x. and 6.x.
- Fixes occasionally failing tests.

### 12.9.6 0.11

2018-07-15

**Note:** This release contains backwards incompatible changes. You should update your Django code and front-end parts of your applications that were relying on the complex queries using `|` and `:` chars in the GET params.

**Note:** If you have used custom filter backends using `SEPARATOR_LOOKUP_VALUE`, `SEPARATOR_LOOKUP_COMPLEX_VALUE` or `SEPARATOR_LOOKUP_COMPLEX_MULTIPLE_VALUE` constants or `split_lookup_complex_value` helper method of the `FilterBackendMixin`, you most likely want to run your functional tests to see if everything still works.

**Note:** Do not keep things as they were in your own fork, since new search backends will use the `|` and `:` symbols differently.

**Examples of old API requests vs new API requests**

**Note:** Note, that `|` and `:` chars were mostly replaced with `__` and `,`.

*Old API requests*

```
http://127.0.0.1:8080/search/publisher/?search=name|reilly&search=city|london
http://127.0.0.1:8000/search/publishers/?location__geo_distance=100000km|12.04|-63.93
http://localhost:8000/api/articles/?id__terms=1|2|3
http://localhost:8000/api/users/?age__range=16|67|2.0
http://localhost:8000/api/articles/?id__in=1|2|3
http://localhost:8000/api/articles/?location__geo_polygon=40,-70|30,-80|20,-90|_
↪name:myname|validation_method:IGNORE_MALFORMED
```

*New API requests*

```
http://127.0.0.1:8080/search/publisher/?search=name:reilly&search=city:london
http://127.0.0.1:8000/search/publishers/?location__geo_distance=100000km__12.04__-63.
↪93
```

(continues on next page)

(continued from previous page)

```
http://localhost:8000/api/articles/?id__terms=1__2__3
http://localhost:8000/api/users/?age__range=16__67__2.0
http://localhost:8000/api/articles/?id__in=1__2__3
http://localhost:8000/api/articles/?location__geo_polygon=40,-70__30,-80__20,-90___
↪name,myname__validation_method,IGNORE_MALFORMED
```

- `SEPARATOR_LOOKUP_VALUE` has been removed. Use `SEPARATOR_LOOKUP_COMPLEX_VALUE` and `SEPARATOR_LOOKUP_COMPLEX_MULTIPLE_VALUE` instead.
- `SEPARATOR_LOOKUP_NAME` has been added.
- The method `split_lookup_complex_value` has been removed. Use `split_lookup_complex_value` instead.
- Default filter lookup option is added. In past, if no specific lookup was provided and there were multiple values for a single field to filter on, by default `terms` filter was used. The `term` lookup was used by default in similar situation for a single value to filter on. It's now possible to declare default lookup which will be used when no lookup is given.
- Removed deprecated `views` module. Import from `viewsets` instead.
- Removed undocumented `get_count` helper from `helpers` module.

### 12.9.7 0.10

2018-07-06

- Elasticsearch 6.x support.
- Minor fixes.

### 12.9.8 0.9

2018-07-04

- Introduced `post_filter` support.
- Generalised the `FilteringFilterBackend` backend. Both `PostFilterFilteringFilterBackend` and `NestedFilteringFilterBackend` backends are now primarily based on it.
- Reduced Elastic queries from 3 to 2 when using `LimitOffsetPagination`.

### 12.9.9 0.8.4

2018-06-27

**Note:** Release supported by Goldmund, Wyldebeast & Wunderliebe.

- Added `NestedFilteringFilterBackend` backend.
- Documentation updated with examples of implementing a nested aggregations/facets.

### 12.9.10 0.8.3

2018-06-25

- It's possible to retrieve original dictionary from `DictionaryProxy` object.
- Added helper wrappers and helper functions as a temporary fix for issues in the `django-elasticsearch-dsl`.

### 12.9.11 0.8.2

2018-06-05

- Minor fixes.

### 12.9.12 0.8.1

2018-06-05

- Fixed wrong filter name in functional suggesters results into an error on Django 1.10 (and prior).
- Documentation improvements.

### 12.9.13 0.8

2018-06-01

---

**Note:** Release supported by Goldmund, Wyldebeast & Wunderliebe.

---

---

**Note:** This release contain minor backwards incompatible changes. You should update your code.

- 1. `BaseDocumentViewSet` (which from now on does not contain `suggest` functionality) has been renamed to `DocumentViewSet` (which does contain `suggest` functionality).
- 2. You should no longer import from `django_elasticsearch_dsl_drf.views`. Instead, import from `django_elasticsearch_dsl_drf.viewsets`.

---

- Deprecated `django_elasticsearch_dsl_drf.views` in favour of `django_elasticsearch_dsl_drf.viewsets`.
- Suggest action/method has been moved to `SuggestMixin` class.
- `FunctionalSuggestMixin` class introduced which resembled functionality of the `SuggestMixin` with several improvements/additions, such as advanced filtering and context-aware suggestions.
- You can now define a default suggester in `suggester_fields` which will be used if you do not provide suffix for the filter name.

### 12.9.14 0.7.2

2018-05-09

**Note:** Release dedicated to the Victory Day, the victims of the Second World War and Liberation of Shushi.

- Django REST framework 3.8.x support.

### 12.9.15 0.7.1

2018-04-04

**Note:** Release supported by Goldmund, Wyldebeast & Wunderliebe.

- Add query *boost* support for search fields.

### 12.9.16 0.7

2018-03-08

**Note:** Dear ladies, congratulations on International Women's Day

- CoreAPI/CoreSchema support.

### 12.9.17 0.6.4

2018-03-05

- Minor fix: explicitly use DocType in the ViewSets.

### 12.9.18 0.6.3

2018-01-03

- Minor fix in the search backend.
- Update the year in the license and code.

### 12.9.19 0.6.2

2017-12-29

- Update example project (and the tests that are dependant on the example project) to work with Django 2.0.
- Set minimal requirement for `django-elasticsearch-dsl` to 3.0.

### 12.9.20 0.6.1

2017-11-28

- Documentation fixes.

### 12.9.21 0.6

2017-11-28

- Added highlight backend.
- Added nested search functionality.

### 12.9.22 0.5.1

2017-10-18

- Fixed serialization of complex nested structures (lists of nested objects).
- Documentation fixes.

### 12.9.23 0.5

2017-10-05

---

**Note:** This release contains changes that might be backwards incompatible for your project. If you have used dynamic document serializer `django_elasticsearch_dsl_drf.serializers.DocumentSerializer` with customisations (with use of `serializers.SerializerMethodField`, having the value parsed to JSON), just remove the custom parts.

---

- Support for `ObjectField`, `NestedField`, `GeoPointField`, `ListField`, `GeoShapeField` (and in general, nesting fields either as a dictionary or list should not be a problem at all).
- Dynamic serializer has been made less strict.
- Added `get_paginated_response_context` methods to both `PageNumberPagination` and `LimitOffsetPagination` pagination classes to simplify customisations.

### 12.9.24 0.4.4

2017-10-02

- Documentation improvements (Elasticsearch suggestions).
- More tests (term and phrase suggestions).
- Code style fixes.

### 12.9.25 0.4.3

2017-09-28

- Documentation fixes.
- Fixes in tests.
- Improved factories.

### 12.9.26 0.4.2

2017-09-28

- Added `geo_bounding_box` query support to the geo-spatial features.

### 12.9.27 0.4.1

2017-09-26

- Fixes in docs.

### 12.9.28 0.4

2017-09-26

---

**Note:** This release contains changes that might be backwards incompatible for your project. Make sure to add the `DefaultOrderingFilterBackend` everywhere you have used the `OrderingFilterBackend`, right after the latter.

---

- `GeoSpatialFilteringFilterBackend` filtering backend, supporting `geo_distance` and `geo_polygon` geo-spatial queries.
- `GeoSpatialOrderingFilterBackend` ordering backend, supporting ordering of results for `geo_distance` filter.
- `OrderingFilterBackend` no longer provides defaults when no ordering is given. In order to take care of the defaults include the `DefaultOrderingFilterBackend` in the list of `filter_backends` (after all other ordering backends).

### 12.9.29 0.3.12

2017-09-21

- Added `geo_distance` filter. Note, that although functionally the filter would not change its' behaviour, it is likely to be moved to a separate backend (`geo_spatial`). For now use as is.
- Minor fixes.

### 12.9.30 0.3.11

2017-09-21

- Added `query` argument to `more_like_this` helper.

### 12.9.31 0.3.10

2017-09-20

- Minor fixes.
- Simplified Elasticsearch version check.

### 12.9.32 0.3.9

2017-09-12

- Python 2.x compatibility fix.

### 12.9.33 0.3.8

2017-09-12

- Fixes tests on some environments.

### 12.9.34 0.3.7

2017-09-07

- Docs fixes.

### 12.9.35 0.3.6

2017-09-07

- Fixed suggestions test for Elasticsearch 5.x.
- Added *compat* module for painless testing of Elastic 2.x to Elastic 5.x transition.

### 12.9.36 0.3.5

2017-08-24

- Minor fixes in the ordering backend.
- Improved tests and coverage.

### 12.9.37 0.3.4

2017-08-23

- Minor fixes in the ordering backend.

### 12.9.38 0.3.3

2017-07-13

- Minor fixes and improvements.

### 12.9.39 0.3.2

2017-07-12

- Minor fixes and improvements.

### 12.9.40 0.3.1

2017-07-12

- Minor Python2 fixes.
- Minor documentation fixes.

### 12.9.41 0.3

2017-07-11

- Add suggestions support (`term`, `phrase` and `completion`).

### 12.9.42 0.2.6

2017-07-11

- Minor fixes.
- Fixes in documentation.

### 12.9.43 0.2.5

2017-07-11

- Fixes in documentation.

### 12.9.44 0.2.4

2017-07-11

- Fixes in documentation.

### 12.9.45 0.2.3

2017-07-11

- Fixes in documentation.

### 12.9.46 0.2.2

2017-07-11

- Fixes in documentation.

### 12.9.47 0.2.1

2017-07-11

- Fixes in documentation.

### 12.9.48 0.2

2017-07-11

- Initial faceted search support.
- Pagination support.

### 12.9.49 0.1.8

2017-06-26

- Python2 fixes.
- Documentation and example project improvements.

### 12.9.50 0.1.7

2017-06-25

- Dynamic serializer for Documents.
- Major improvements in documentation.

### 12.9.51 0.1.6

2017-06-23

- Implemented `gt`, `gte`, `lt` and `lte` functional query lookups.
- Implemented `ids` native filter lookup.

### 12.9.52 0.1.5

2017-06-22

- Implemented `endswith` and `contains` functional filters.
- Added tests for `wildcard`, `exists`, `exclude` and `isnull` filters. Improved `range` filter tests.
- Improve `more_like_this` helper test.
- Improve ordering tests.
- Two additional arguments added to the `more_like_this` helper: `min_doc_freq` and `max_doc_freq`.
- Minor documentation improvements.

### 12.9.53 0.1.4

2017-06-22

- Added tests for `in`, `term` and `terms` filters.
- Minor documentation fixes.

#### 12.9.54 0.1.3

2017-06-21

- Added tests for `more_like_this` helper, `range` and `prefix` filters.
- Minor documentation improvements.

#### 12.9.55 0.1.2

2017-06-20

- Minor fixes in tests.

#### 12.9.56 0.1.1

2017-06-20

- Fixes in `more_like_this` helper.
- Tiny documentation improvements.

#### 12.9.57 0.1

2017-06-19

- Initial beta release.

## 12.10 django_elasticsearch_dsl_drf package

### 12.10.1 Subpackages

#### 12.10.1.1 django_elasticsearch_dsl_drf.fields package

##### 12.10.1.1.1 Submodules

##### 12.10.1.1.2 django_elasticsearch_dsl_drf.fields.common module

Common fields.

**class** django_elasticsearch_dsl_drf.fields.common.**BooleanField**(***kwargs*)
Bases: `rest_framework.fields.BooleanField`

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.common.**CharField**(***kwargs*)
Bases: `rest_framework.fields.CharField`

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.common.**DateField**(*format=<class 'rest_framework.fields.empty'>*, *input_formats=None*, **args*, ***kwargs*)
Bases: rest_framework.fields.DateField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.common.**FloatField**(***kwargs*)
Bases: rest_framework.fields.FloatField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.common.**IntegerField**(***kwargs*)
Bases: rest_framework.fields.IntegerField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.common.**IPAddressField**(*protocol='both'*, ***kwargs*)
Bases: rest_framework.fields.IPAddressField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

#### 12.10.1.1.3 django_elasticsearch_dsl_drf.fields.helpers module

Helpers.

django_elasticsearch_dsl_drf.fields.helpers.**to_representation**(*value*)
To representation.

#### 12.10.1.1.4 django_elasticsearch_dsl_drf.fields.nested_fields module

Nested fields.

**class** django_elasticsearch_dsl_drf.fields.nested_fields.**GeoPointField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Geo point field.

**class** django_elasticsearch_dsl_drf.fields.nested_fields.**GeoShapeField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Geo shape field.

**class** django_elasticsearch_dsl_drf.fields.nested_fields.**NestedField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Nested field.

**class** django_elasticsearch_dsl_drf.fields.nested_fields.**ObjectField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: rest_framework.fields.Field

Object field.

**get_value**(*dictionary*)
Get value.

**to_internal_value**(*data*)
To internal value.

**to_representation**(*value*)

To representation.

**class** django_elasticsearch_dsl_drf.fields.nested_fields.**ListField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: rest_framework.fields.Field

List field.

**get_value**(*dictionary*)
Get value.

**to_internal_value**(*data*)
To internal value.

**to_representation**(*value*)
To representation.

#### 12.10.1.1.5 Module contents

Fields.

**class** django_elasticsearch_dsl_drf.fields.**BooleanField**(***kwargs*)
Bases: rest_framework.fields.BooleanField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**CharField**(***kwargs*)
Bases: rest_framework.fields.CharField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**DateField**(*format=<class 'rest_framework.fields.empty'>*, *input_formats=None*, **args*, ***kwargs*)
Bases: rest_framework.fields.DateField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**FloatField**(***kwargs*)
Bases: rest_framework.fields.FloatField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**GeoPointField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)
Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Geo point field.

**class** django_elasticsearch_dsl_drf.fields.**GeoShapeField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Geo shape field.

**class** django_elasticsearch_dsl_drf.fields.**IntegerField**(***kwargs*)
Bases: rest_framework.fields.IntegerField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**IPAddressField**(*protocol='both'*, ***kwargs*)
Bases: rest_framework.fields.IPAddressField

Object field.

**get_value**(*dictionary*)
Get value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**ListField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)
Bases: rest_framework.fields.Field

List field.

**get_value**(*dictionary*)
Get value.

**to_internal_value**(*data*)
To internal value.

**to_representation**(*value*)
To representation.

**class** django_elasticsearch_dsl_drf.fields.**NestedField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)
Bases: *django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField*

Nested field.

**class** django_elasticsearch_dsl_drf.fields.**ObjectField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class 'rest_framework.fields.empty'>*, *initial=<class 'rest_framework.fields.empty'>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: rest_framework.fields.Field

Object field.

**get_value**(*dictionary*)
Get value.

**to_internal_value**(*data*)
To internal value.

**to_representation**(*value*)
To representation.

#### 12.10.1.2 django_elasticsearch_dsl_drf.filter_backends package

##### 12.10.1.2.1 Subpackages

###### 12.10.1.2.1.1 django_elasticsearch_dsl_drf.filter_backends.aggregations package

###### 12.10.1.2.1.2 Submodules

###### 12.10.1.2.1.3 django_elasticsearch_dsl_drf.filter_backends.aggregations.bucket_aggregations module

###### 12.10.1.2.1.4 django_elasticsearch_dsl_drf.filter_backends.aggregations.metrics_aggregations module

###### 12.10.1.2.1.5 django_elasticsearch_dsl_drf.filter_backends.aggregations.pipeline_aggregations module

###### 12.10.1.2.1.6 Module contents

###### 12.10.1.2.1.7 django_elasticsearch_dsl_drf.filter_backends.filtering package

###### 12.10.1.2.1.8 Submodules

###### 12.10.1.2.1.9 django_elasticsearch_dsl_drf.filter_backends.filtering.common module

Common filtering backend.

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.common.**FilteringFilterBackend**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     FilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [FilteringFilterBackend,]
>>>     filter_fields = {
>>>         'title': 'title.raw',
>>>         'state': {
>>>             'field': 'state.raw',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>             'default_lookup': LOOKUP_FILTER_WILDCARD,
>>>         }
>>> }
```

**classmethod apply_filter_prefix**(*queryset*, *options*, *value*)

Apply *prefix* filter.

Syntax:

/endpoint/?field_name__prefix={value}

Example:

http://localhost:8000/api/articles/?tags__prefix=bio

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_range**(*queryset*, *options*, *value*)

Apply *range* filter.

Syntax:

/endpoint/?field_name__range={lower}__{upper}__{boost} /endpoint/?field_name__range={lower}__{upper}

Example:

http://localhost:8000/api/users/?age__range=16__67__2.0 http://localhost:8000/api/users/?age__range=16__67 http://localhost:8000/api/users/?age__range=16

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_term**(*queryset*, *options*, *value*)

Apply *term* filter.

Syntax:

/endpoint/?field_name={value}

Example:

http://localhost:8000/api/articles/?tags=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_terms**(*queryset*, *options*, *value*)

Apply *terms* filter.

Syntax:

/endpoint/?field_name__terms={value1}__{value2} /endpoint/?field_name__terms={value1}

Note, that number of values is not limited.

Example:

http://localhost:8000/api/articles/?tags__terms=children__python http://localhost:8000/api/articles/?tags__terms=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*mixed: either str or iterable (list, tuple)*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_contains**(*queryset*, *options*, *value*)

Apply *contains* filter.

Syntax:

/endpoint/?field_name__contains={value}

Example:

http://localhost:8000/api/articles/?state__contains=lis

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_endswith**(*queryset*, *options*, *value*)

Apply *endswith* filter.

Syntax:

/endpoint/?field_name__endswith={value}

Example:

http://localhost:8000/api/articles/?tags__endswith=dren

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_exclude**(*queryset*, *options*, *value*)

Apply *exclude* functional query.

Syntax:

/endpoint/?field_name__isnull={value1}__{value2} /endpoint/?field_name__exclude={valu1}

Note, that number of values is not limited.

Example:

http://localhost:8000/api/articles/?tags__exclude=children__python http://localhost:8000/api/articles/?tags__exclude=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_exists**(*queryset*, *options*, *value*)

Apply *exists* filter.

Syntax:

/endpoint/?field_name__exists=true /endpoint/?field_name__exists=false

Example:

http://localhost:8000/api/articles/?tags__exists=true http://localhost:8000/api/articles/?tags__exists=false

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_gt**(*queryset*, *options*, *value*)

Apply *gt* functional query.

Syntax:

/endpoint/?field_name__gt={value}__{boost} /endpoint/?field_name__gt={value}

Example:

http://localhost:8000/api/articles/?id__gt=1__2.0 http://localhost:8000/api/articles/?id__gt=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_gte**(*queryset*, *options*, *value*)
Apply *gte* functional query.

Syntax:

/endpoint/?field_name__gte={value}__{boost} /endpoint/?field_name__gte={value}

Example:

http://localhost:8000/api/articles/?id__gte=1__2.0 http://localhost:8000/api/articles/?id__gte=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_in**(*queryset*, *options*, *value*)
Apply *in* functional query.

Syntax:

/endpoint/?field_name__in={value1}__{value2} /endpoint/?field_name__in={value1}

Note, that number of values is not limited.

Example:

http://localhost:8000/api/articles/?tags__in=children__python

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_isnull**(*queryset*, *options*, *value*)
Apply *isnull* functional query.

Syntax:

/endpoint/?field_name__isnull=true /endpoint/?field_name__isnull=false

Example:

http://localhost:8000/api/articles/?tags__isnull=true http://localhost:8000/api/articles/?tags__isnull=false

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_lt**(*queryset*, *options*, *value*)
Apply *lt* functional query.

Syntax:

/endpoint/?field_name__lt={value}__{boost} /endpoint/?field_name__lt={value}

Example:

http://localhost:8000/api/articles/?id__lt=1__2.0 http://localhost:8000/api/articles/?id__lt=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_lte**(*queryset*, *options*, *value*)
Apply *lte* functional query.

Syntax:

/endpoint/?field_name__lte={value}__{boost} /endpoint/?field_name__lte={value}

Example:

http://localhost:8000/api/articles/?id__lte=1__2.0 http://localhost:8000/api/articles/?id__lte=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_wildcard**(*queryset*, *options*, *value*)
Apply *wildcard* filter.

Syntax:

/endpoint/?field_name__wildcard={value}* /endpoint/?field_name__wildcard=*{value} /endpoint/?field_name__wildcard=*{value}*

Example:

http://localhost:8000/api/articles/?tags__wildcard=child*

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.

- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_coreschema_field**(*field*)

**get_filter_query_params**(*request*, *view*)
Get query params to be filtered on.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.

**Return type** dict

**classmethod get_gte_lte_params**(*value*, *lookup*)
Get params for *gte*, *gt*, *lte* and *lt* query.

Syntax:

/endpoint/?field_name__gt={lower}__{boost} /endpoint/?field_name__gt={lower}

Example:

http://localhost:8000/api/articles/?id__gt=1 http://localhost:8000/api/articles/?id__gt=1__2.0

**Parameters**

- **value** (*str*) –
- **lookup** (*str*) –

**Returns** Params to be used in *range* query.

**Return type** dict

**classmethod get_range_params**(*value*)
Get params for *range* query.

Syntax:

/endpoint/?field_name__range={lower}__{upper}__{boost} /endpoint/?field_name__range={lower}__{upper}

Example:

http://localhost:8000/api/users/?age__range=16__67__2.0 http://localhost:8000/api/users/?age__range=16__67 http://localhost:8000/api/users/?age__range=16

**Parameters** **value** –

**Type** str

**Returns** Params to be used in *range* query.

**Return type** dict

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)

Prepare filter fields.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Filtering options.

**Return type** dict

#### 12.10.1.2.1.10 django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial module

Geo spatial filtering backend.

Elasticsearch supports two types of geo data:

- geo_point fields which support lat/lon pairs
- geo_shape fields, which support points, lines, circles, polygons, multi-polygons etc.

The queries in this group are:

- geo_shape query: Find document with geo-shapes which either intersect, are contained by, or do not intersect with the specified geo-shape.
- geo_bounding_box query: Finds documents with geo-points that fall into the specified rectangle.
- geo_distance query: Finds document with geo-points within the specified distance of a central point.
- geo_distance_range query: Like the geo_distance query, but the range starts at a specified distance from the central point. Note, that this one is deprecated and this isn't implemented.
- geo_polygon query: Find documents with geo-points within the specified polygon.

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial.**GeoSpatialFilteri**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Geo-spatial filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_GEO_DISTANCE,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
```

(continues on next page)

(continued from previous page)

```
>>>     GeoSpatialFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [GeoSpatialFilteringFilterBackend,]
>>>     geo_spatial_filter_fields = {
>>>         'loc': 'location',
>>>         'location': {
>>>             'field': 'location',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_GEO_DISTANCE,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_query_geo_bounding_box**(*queryset*, *options*, *value*)
Apply *geo_bounding_box* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_geo_distance**(*queryset*, *options*, *value*)
Apply *geo_distance* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_geo_polygon**(*queryset*, *options*, *value*)
Apply *geo_polygon* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.

- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**filter_queryset** (*request*, *queryset*, *view*)
Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_filter_query_params** (*request*, *view*)
Get query params to be filtered on.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.

**Return type** dict

**classmethod get_geo_bounding_box_params** (*value*, *field*)
Get params for *geo_bounding_box* query.

Example:

/api/articles/?location__geo_bounding_box=40.73,-74.1__40.01,-71.12

Example:

**/api/articles/?location__geo_polygon=40.73,-74.1** __40.01,-71.12 ___name,myname __validation_method,IGNORE_MALFORMED __type,indexed

Elasticsearch:

**{**

**"query": {**

**"bool"** [{]

**"must"** [{] "match_all" : {}

}, "filter" : {

**"geo_bounding_box"** [{]

**"person.location"** [{]

**"top_left"** [{] "lat" : 40.73, "lon" : -74.1

}, "bottom_right" : {

"lat" : 40.01, "lon" : -71.12

}

}

}

}

}

}

}

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_bounding_box* query.

**Return type** dict

**classmethod get_geo_distance_params**(*value*, *field*)

Get params for *geo_distance* query.

Example:

/api/articles/?location__geo_distance=2km__43.53__-12.23

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.

**Return type** dict

**classmethod get_geo_polygon_params**(*value*, *field*)

Get params for *geo_polygon* query.

Example:

/api/articles/?location__geo_polygon=40,-70__30,-80__20,-90

Example:

**/api/articles/?location__geo_polygon=40,-70** __30,-80 __20,-90 ___name,myname __validation_method,IGNORE_MALFORMED

Elasticsearch:

**{**

**“query”: {**

**“bool”** [{]

**“must”** [{] “match_all” : {}

}, “filter” : {

**“geo_polygon”** [{]

**“person.location”** [{]

**“points”** [[] {“lat” : 40, “lon” : -70}, {“lat” : 30, “lon” : -80}, {“lat” : 20, “lon” : -90}

]

}

}

}

}

}

}

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.

**Return type** dict

**classmethod prepare_filter_fields**(*view*)

Prepare filter fields.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Filtering options.

**Return type** dict

#### 12.10.1.2.1.11 django_elasticsearch_dsl_drf.filter_backends.filtering.ids module

Ids filtering backend.

Filters documents that only have the provided ids. Note, this query uses the *_uid* field.

Elastic query:

{

"query": {

"ids": { "type": "book_document", "values": ["68", "64", "58"]

}

}

}

REST framework request equivalent:

- http://localhost:8000/api/articles/?ids=68__64__58
- http://localhost:8000/api/articles/?ids=68&ids=64&ids=58

Official Elastic docs:

- https://www.elastic.co/guide/en/elasticsearch/reference/current/ query-dsl-ids-query.html

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.ids.**IdsFilterBackend**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Ids filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     IdsFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
```

(continues on next page)

(continued from previous page)

```
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [IdsFilterBackend]
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_ids_query_params**(*request*)

Get search query params.

**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.

**Returns** List of search query params.

**Return type** list

**get_ids_values**(*request*, *view*)

Get ids values for query.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**ids_query_param = 'ids'**

#### 12.10.1.2.1.12 django_elasticsearch_dsl_drf.filter_backends.filtering.nested module

Nested filtering backend.

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.nested.**NestedFilteringFilterBa**

Bases: *django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend*

Nested filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_TERM,
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     NestedFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [NestedFilteringFilterBackend,]
>>>     nested_filter_fields = {
>>>         'country': {
>>>             'field': 'continent.country.name.raw',
>>>             'path': 'continent.country',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_TERM,
>>>                 LOOKUP_FILTER_TERMS,
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_filter**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply filter.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod apply_query**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply query.

**Parameters**

- **queryset** –

- **options** –
- **args** –
- **kwargs** –

**Returns**

**get_coreschema_field**(*field*)

**get_filter_field_nested_path**(*filter_fields*, *field_name*)

Get filter field path to be used in nested query.

**Parameters**

- **filter_fields** –
- **field_name** –

**Returns**

**get_filter_query_params**(*request*, *view*)

Get query params to be filtered on.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.

**Return type** dict

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)

Prepare filter fields.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Filtering options.

**Return type** dict

#### 12.10.1.2.1.13 django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter module

The `post_filter` filtering backend.

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter.**PostFilterFilteri**

Bases: *django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend*

The `post_filter` filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     PostFilterFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
```

(continues on next page)

(continued from previous page)

```
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [PostFilterFilteringFilterBackend,]
>>>     post_filter_fields = {
>>>         'title': 'title.raw',
>>>         'state': {
>>>             'field': 'state.raw',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_filter**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply filter.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod apply_query**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply query.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**get_coreschema_field**(*field*)

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)

Prepare filter fields.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Filtering options.

**Return type** dict

#### 12.10.1.2.1.14 Module contents

Term level filtering and `post_filter` backends.

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.**FilteringFilterBackend**
Bases: `rest_framework.filters.BaseFilterBackend`, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     FilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [FilteringFilterBackend,]
>>>     filter_fields = {
>>>         'title': 'title.raw',
>>>         'state': {
>>>             'field': 'state.raw',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>             'default_lookup': LOOKUP_FILTER_WILDCARD,
>>>         }
>>> }
```

**classmethod apply_filter_prefix**(*queryset*, *options*, *value*)
Apply *prefix* filter.

Syntax:
/endpoint/?field_name__prefix={value}
Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.

- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_range**(*queryset*, *options*, *value*)

Apply *range* filter.

Syntax:
/endpoint/?field_name__range={lower}__{upper}__{boost} /endpoint/?field_name__range={lower}__{upper}

Example:
http://localhost:8000/api/users/?age__range=16__67__2.0 http://localhost:8000/api/users/?age__range=16__67 http://localhost:8000/api/users/?age__range=16

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_term**(*queryset*, *options*, *value*)

Apply *term* filter.

Syntax:
/endpoint/?field_name={value}

Example:
http://localhost:8000/api/articles/?tags=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_filter_terms**(*queryset*, *options*, *value*)

Apply *terms* filter.

Syntax:
/endpoint/?field_name__terms={value1}__{value2} /endpoint/?field_name__terms={value1}
Note, that number of values is not limited.

Example:
http://localhost:8000/api/articles/?tags__terms=children__python http://localhost:8000/api/articles/?tags__terms=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.

- **value** (*mixed: either str or iterable (list, tuple)*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_contains**(*queryset*, *options*, *value*)

Apply *contains* filter.

Syntax:

/endpoint/?field_name__contains={value}

Example:

http://localhost:8000/api/articles/?state__contains=lis

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_endswith**(*queryset*, *options*, *value*)

Apply *endswith* filter.

Syntax:

/endpoint/?field_name__endswith={value}

Example:

http://localhost:8000/api/articles/?tags__endswith=dren

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_exclude**(*queryset*, *options*, *value*)

Apply *exclude* functional query.

Syntax:

/endpoint/?field_name__isnull={value1}__{value2} /endpoint/?field_name__exclude={valu1}

Note, that number of values is not limited.

Example:

http://localhost:8000/api/articles/?tags__exclude=children__python http://localhost:8000/api/articles/?tags__exclude=children

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_exists**(*queryset*, *options*, *value*)

Apply *exists* filter.

Syntax:

/endpoint/?field_name__exists=true /endpoint/?field_name__exists=false

Example:

http://localhost:8000/api/articles/?tags__exists=true http://localhost:8000/api/articles/?tags__exists=false

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_gt**(*queryset*, *options*, *value*)

Apply *gt* functional query.

Syntax:

/endpoint/?field_name__gt={value}__{boost} /endpoint/?field_name__gt={value}

Example:

http://localhost:8000/api/articles/?id__gt=1__2.0 http://localhost:8000/api/articles/?id__gt=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_gte**(*queryset*, *options*, *value*)

Apply *gte* functional query.

Syntax:

/endpoint/?field_name__gte={value}__{boost} /endpoint/?field_name__gte={value}

Example:

http://localhost:8000/api/articles/?id__gte=1__2.0 http://localhost:8000/api/articles/?id__gte=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_in**(*queryset*, *options*, *value*)
Apply *in* functional query.

Syntax:
/endpoint/?field_name__in={value1}__{value2} /endpoint/?field_name__in={value1}
Note, that number of values is not limited.

Example:
http://localhost:8000/api/articles/?tags__in=children__python

**Parameters**
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_isnull**(*queryset*, *options*, *value*)
Apply *isnull* functional query.

Syntax:
/endpoint/?field_name__isnull=true /endpoint/?field_name__isnull=false
Example:
http://localhost:8000/api/articles/?tags__isnull=true http://localhost:8000/api/articles/?tags__isnull=false

**Parameters**
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_lt**(*queryset*, *options*, *value*)
Apply *lt* functional query.

Syntax:
/endpoint/?field_name__lt={value}__{boost} /endpoint/?field_name__lt={value}
Example:
http://localhost:8000/api/articles/?id__lt=1__2.0 http://localhost:8000/api/articles/?id__lt=1

**Parameters**
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_lte**(*queryset*, *options*, *value*)
Apply *lte* functional query.

Syntax:

/endpoint/?field_name__lte={value}__{boost} /endpoint/?field_name__lte={value}

Example:

http://localhost:8000/api/articles/?id__lte=1__2.0 http://localhost:8000/api/articles/?id__lte=1

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_wildcard**(*queryset*, *options*, *value*)

Apply *wildcard* filter.

Syntax:

/endpoint/?field_name__wildcard={value}* /endpoint/?field_name__wildcard=*{value} /endpoint/?field_name__wildcard=*{value}*

Example:

http://localhost:8000/api/articles/?tags__wildcard=child*

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_coreschema_field**(*field*)

**get_filter_query_params**(*request*, *view*)

Get query params to be filtered on.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.

**Return type** dict

**classmethod get_gte_lte_params**(*value*, *lookup*)
Get params for *gte*, *gt*, *lte* and *lt* query.

Syntax:
/endpoint/?field_name__gt={lower}__{boost} /endpoint/?field_name__gt={lower}
Example:
http://localhost:8000/api/articles/?id__gt=1 http://localhost:8000/api/articles/?id__gt=1__2.0

**Parameters**
- **value** (*str*) –
- **lookup** (*str*) –

**Returns** Params to be used in *range* query.
**Return type** dict

**classmethod get_range_params**(*value*)
Get params for *range* query.

Syntax:
/endpoint/?field_name__range={lower}__{upper}__{boost} /endpoint/?field_name__range={lower}__{upper}
Example:
http://localhost:8000/api/users/?age__range=16__67__2.0 http://localhost:8000/api/users/?age__range=16__67 http://localhost:8000/api/users/?age__range=16

**Parameters** **value** –
**Type** str
**Returns** Params to be used in *range* query.
**Return type** dict

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.**GeoSpatialFilteringFilterBacke**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Geo-spatial filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_GEO_DISTANCE,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     GeoSpatialFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
```

(continues on next page)

(continued from previous page)

```
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [GeoSpatialFilteringFilterBackend,]
>>>     geo_spatial_filter_fields = {
>>>         'loc': 'location',
>>>         'location': {
>>>             'field': 'location',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_GEO_DISTANCE,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_query_geo_bounding_box**(*queryset*, *options*, *value*)

Apply *geo_bounding_box* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_geo_distance**(*queryset*, *options*, *value*)

Apply *geo_distance* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_query_geo_polygon**(*queryset*, *options*, *value*)

Apply *geo_polygon* query.

**Parameters**

- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_filter_query_params**(*request*, *view*)
Get query params to be filtered on.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.
**Return type** dict

**classmethod get_geo_bounding_box_params**(*value*, *field*)
Get params for *geo_bounding_box* query.

Example:
/api/articles/?location__geo_bounding_box=40.73,-74.1__40.01,-71.12
Example:
**/api/articles/?location__geo_polygon=40.73,-74.1** __40.01,-71.12 ___name,myname __validation_method,IGNORE_MALFORMED __type,indexed
Elasticsearch:

```
{
    "query": {
        "bool" [{]
            "must" [{] "match_all" : {}
            }, "filter" : {
                "geo_bounding_box" [{]
                    "person.location" [{]
                        "top_left" [{] "lat" : 40.73, "lon" : -74.1
                        }, "bottom_right" : {
                            "lat" : 40.01, "lon" : -71.12
                        }
                    }
                }
            }
        }
    }
```

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_bounding_box* query.
**Return type** dict

**classmethod get_geo_distance_params**(*value*, *field*)
Get params for *geo_distance* query.

Example:
/api/articles/?location__geo_distance=2km__43.53__-12.23

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.
**Return type** dict

**classmethod get_geo_polygon_params**(*value*, *field*)
Get params for *geo_polygon* query.

Example:
/api/articles/?location__geo_polygon=40,-70__30,-80__20,-90
Example:
**/api/articles/?location__geo_polygon=40,-70** __30,-80 __20,-90 ___name,myname __validation_method,IGNORE_MALFORMED
Elasticsearch:
**{**

**“query”: {**

**“bool”** [{]

**“must”** [{] “match_all” : {}

}, “filter” : {

**“geo_polygon”** [{]

**“person.location”** [{]

**“points”** [[] {“lat” : 40, “lon” : -70}, {“lat” : 30, “lon” : -80}, {“lat” : 20, “lon” : -90}

]

}

}

}

}

}

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.

**Return type** dict

**classmethod prepare_filter_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.**IdsFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Ids filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     IdsFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [IdsFilterBackend]
```

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_ids_query_params**(*request*)
Get search query params.
**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.
**Returns** List of search query params.
**Return type** list

**get_ids_values**(*request*, *view*)
Get ids values for query.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.

- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**ids_query_param = 'ids'**

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.**NestedFilteringFilterBackend**
Bases: *django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend*

Nested filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_TERM,
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     NestedFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [NestedFilteringFilterBackend,]
>>>     nested_filter_fields = {
>>>         'country': {
>>>             'field': 'continent.country.name.raw',
>>>             'path': 'continent.country',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_TERM,
>>>                 LOOKUP_FILTER_TERMS,
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_filter**(*queryset*, *options=None*, *args=None*, *kwargs=None*)
Apply filter.

**Parameters**

- **queryset** –

- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod apply_query**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply query.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**get_coreschema_field**(*field*)

**get_filter_field_nested_path**(*filter_fields*, *field_name*)

Get filter field path to be used in nested query.

**Parameters**

- **filter_fields** –
- **field_name** –

**Returns**

**get_filter_query_params**(*request*, *view*)

Get query params to be filtered on.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.

**Return type** dict

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)

Prepare filter fields.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Filtering options.

**Return type** dict

**class** django_elasticsearch_dsl_drf.filter_backends.filtering.**PostFilterFilteringFilterBacke**

Bases: *django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend*

The `post_filter` filtering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     LOOKUP_FILTER_PREFIX,
>>>     LOOKUP_FILTER_WILDCARD,
>>>     LOOKUP_QUERY_EXCLUDE,
```

(continues on next page)

(continued from previous page)

```
>>>     LOOKUP_QUERY_ISNULL,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     PostFilterFilteringFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [PostFilterFilteringFilterBackend,]
>>>     post_filter_fields = {
>>>         'title': 'title.raw',
>>>         'state': {
>>>             'field': 'state.raw',
>>>             'lookups': [
>>>                 LOOKUP_FILTER_PREFIX,
>>>                 LOOKUP_FILTER_WILDCARD,
>>>                 LOOKUP_QUERY_EXCLUDE,
>>>                 LOOKUP_QUERY_ISNULL,
>>>             ],
>>>         }
>>> }
```

**classmethod apply_filter**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply filter.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod apply_query**(*queryset*, *options=None*, *args=None*, *kwargs=None*)

Apply query.

**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**get_coreschema_field**(*field*)

**get_schema_fields**(*view*)

**classmethod prepare_filter_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

#### 12.10.1.2.1.15 django_elasticsearch_dsl_drf.filter_backends.ordering package

#### 12.10.1.2.1.16 Submodules

#### 12.10.1.2.1.17 django_elasticsearch_dsl_drf.filter_backends.ordering.common module

Ordering backend.

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.common.**DefaultOrderingFilterBa**
Bases: rest_framework.filters.BaseFilterBackend

Default ordering filter backend for Elasticsearch.

Make sure this is your last ordering backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     DefaultOrderingFilterBackend,
>>>     OrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [
>>>         DefaultOrderingFilterBackend,
>>>         OrderingFilterBackend,
>>>     ]
>>>     ordering_fields = {
>>>         'id': 'id',
>>>         'title': 'title.raw',
>>>         'date_submitted': 'date_submitted',
>>>         'state': {
>>>             'field': 'state.raw',
>>>         }
>>>     }
>>>     ordering = 'name'
```

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**classmethod get_default_ordering_params**(*view*)
Get the default ordering params for the view.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.
**Returns** Ordering params to be used for ordering.
**Return type** list

**get_ordering_query_params**(*request*, *view*)
Get ordering query params.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.
**Return type** list

**ordering_param = 'ordering'**

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.common.**OrderingFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend

Ordering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     OrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [OrderingFilterBackend,]
>>>     ordering_fields = {
>>>         'id': 'id',
>>>         'title': 'title.raw',
>>>         'date_submitted': 'date_submitted',
>>>         'state': {
>>>             'field': 'state.raw',
```

(continues on next page)

(continued from previous page)

```
>>>         }
>>>     }
>>>     ordering = ('id', 'name',)
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_ordering_query_params**(*request*, *view*)

Get ordering query params.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**get_schema_fields**(*view*)

**ordering_param = 'ordering'**

#### 12.10.1.2.1.18 django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial module

Geo-spatial ordering backend.

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial.**GeoSpatialOrderingF**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Geo-spatial ordering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     GeoSpatialOrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
```

(continues on next page)

(continued from previous page)

```
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [GeoSpatialOrderingFilterBackend,]
>>>     geo_spatial_ordering_fields = {
>>>         'location': {
>>>             'field': 'location',
>>>         }
>>>     }
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod get_geo_distance_params**(*value*, *field*)

Get params for *geo_distance* ordering.

Example:

/api/articles/?ordering=-location__45.3214__-34.3421__km__planes

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.

**Return type** dict

**get_geo_spatial_field_name**(*request*, *view*, *name*)

Get geo-spatial field name.

We have to deal with a couple of situations here:

Example 1:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': None,
>>> }
```

Example 2:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': 'location',
>>> }
```

Example 3:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': {
```

(continues on next page)

(continued from previous page)

```
>>>         'field': 'location'
>>>     },
>>> }
```

**Parameters**

- **request** –
- **view** –
- **name** –

**Returns**

**get_ordering_query_params**(*request*, *view*)

Get ordering query params.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**ordering_param = 'ordering'**

#### 12.10.1.2.1.19 Module contents

Ordering backends.

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.**DefaultOrderingFilterBackend**

Bases: rest_framework.filters.BaseFilterBackend

Default ordering filter backend for Elasticsearch.

Make sure this is your last ordering backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     DefaultOrderingFilterBackend,
>>>     OrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [
>>>         DefaultOrderingFilterBackend,
```

(continues on next page)

(continued from previous page)

```
>>>         OrderingFilterBackend,
>>>     ]
>>>     ordering_fields = {
>>>         'id': 'id',
>>>         'title': 'title.raw',
>>>         'date_submitted': 'date_submitted',
>>>         'state': {
>>>             'field': 'state.raw',
>>>         }
>>>     }
>>>     ordering = 'name'
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod get_default_ordering_params**(*view*)

Get the default ordering params for the view.

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**get_ordering_query_params**(*request*, *view*)

Get ordering query params.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**ordering_param = 'ordering'**

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.**GeoSpatialOrderingFilterBacken**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Geo-spatial ordering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     GeoSpatialOrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
```

(continues on next page)

(continued from previous page)

```
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [GeoSpatialOrderingFilterBackend,]
>>>     geo_spatial_ordering_fields = {
>>>         'location': {
>>>             'field': 'location',
>>>         }
>>>     }
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod get_geo_distance_params**(*value*, *field*)

Get params for *geo_distance* ordering.

Example:

/api/articles/?ordering=-location__45.3214__-34.3421__km__planes

**Parameters**

- **value** (*str*) –
- **field** –

**Returns** Params to be used in *geo_distance* query.

**Return type** dict

**get_geo_spatial_field_name**(*request*, *view*, *name*)

Get geo-spatial field name.

We have to deal with a couple of situations here:

Example 1:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': None,
>>> }
```

Example 2:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': 'location',
>>> }
```

Example 3:

```
>>> geo_spatial_ordering_fields = {
>>>     'location': {
>>>         'field': 'location'
>>>     },
>>> }
```

**Parameters**

- **request** –
- **view** –
- **name** –

**Returns**

**get_ordering_query_params**(*request*, *view*)

Get ordering query params.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**ordering_param = 'ordering'**

**class** django_elasticsearch_dsl_drf.filter_backends.ordering.**OrderingFilterBackend**

Bases: rest_framework.filters.BaseFilterBackend

Ordering filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     OrderingFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [OrderingFilterBackend,]
>>>     ordering_fields = {
>>>         'id': 'id',
```

(continues on next page)

(continued from previous page)

```
>>>         'title': 'title.raw',
>>>         'date_submitted': 'date_submitted',
>>>         'state': {
>>>             'field': 'state.raw',
>>>         }
>>>     }
>>>     ordering = ('id', 'name',)
```

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_ordering_query_params**(*request*, *view*)

Get ordering query params.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Ordering params to be used for ordering.

**Return type** list

**get_schema_fields**(*view*)

**ordering_param = 'ordering'**

#### 12.10.1.2.1.20 django_elasticsearch_dsl_drf.filter_backends.suggester package

#### 12.10.1.2.1.21 Submodules

#### 12.10.1.2.1.22 django_elasticsearch_dsl_drf.filter_backends.suggester.functional module

Functional suggesters backend.

It's assumed, that fields you're planning to query suggestions for have been properly indexed using `fields.CompletionField`.

Example:

```
>>> from django_elasticsearch_dsl import DocType, Index, fields
>>>
>>> from books.models import Publisher
>>>
>>> # Name of the Elasticsearch index
```

(continues on next page)

(continued from previous page)

```
>>> PUBLISHER_INDEX = Index(PUBLISHER_INDEX_NAME)
>>> # See Elasticsearch Indices API reference for available settings
>>> PUBLISHER_INDEX.settings(
>>>     number_of_shards=1,
>>>     number_of_replicas=1
>>> )
>>>
>>> @PUBLISHER_INDEX.doc_type
>>> class PublisherDocument(DocType):
>>>     "Publisher Elasticsearch document."
>>>
>>>     id = fields.IntegerField(attr='id')
>>>
>>>     name = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     info = fields.StringField()
>>>
>>>     address = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword')
>>>         }
>>>     )
>>>
>>>     city = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     state_province = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     country = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     website = fields.StringField()
>>>
>>>     class Meta(object):
>>>         "Meta options."
>>>
>>>         model = Publisher  # The model associate with this DocType
```

**class** django_elasticsearch_dsl_drf.filter_backends.suggester.functional.**FunctionalSuggester**

Bases: `rest_framework.filters.BaseFilterBackend`, *`django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin`*

Suggester filter backend for Elasticsearch.

Suggestion functionality is exclusive. Once you have queried the `FunctionalSuggesterFilterBackend`, the latter will transform your current search query into another search query (altered). Therefore, always add it as the very last filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>     FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>     FUNCTIONAL_SUGGESTER_PHRASE_MATCH,
>>>     FUNCTIONAL_SUGGESTER_PHRASE_MATCH,
>>>     FUNCTIONAL_SUGGESTER_TERM_MATCH,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     FunctionalSuggesterFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet
>>>
>>> # Local PublisherDocument definition
>>> from .documents import PublisherDocument
>>>
>>> # Local PublisherDocument serializer
>>> from .serializers import PublisherDocumentSerializer
>>>
>>> class PublisherDocumentView(DocumentViewSet):
>>>
>>>     document = PublisherDocument
>>>     serializer_class = PublisherDocumentSerializer
>>>     filter_backends = [
>>>         # ...
>>>         FunctionalSuggesterFilterBackend,
>>>     ]
>>>     # Suggester fields
>>>     suggester_fields = {
>>>         'name_suggest': {
>>>             'field': 'name.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>         'city_suggest': {
>>>             'field': 'city.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>         'state_province_suggest': {
>>>             'field': 'state_province.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>             ],
>>>         },
>>>         'country_suggest': {
```

(continues on next page)

(continued from previous page)

```
>>>             'field': 'country.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>     }
```

**classmethod apply_suggester_completion_match**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester match.

This is effective when used with Ngram fields.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_completion_prefix**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester prefix.

This is effective when used with Keyword fields.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**clean_queryset**(*queryset*)

Clean the queryset.

- Remove aggregations.
- Remove highlight.
- Remove sorting options.

**Parameters** **queryset** –

**Returns**

**extract_field_name**(*field_name*)

Extract field name.

For instance, “name.suggest” or “name.raw” becomes “name”.

**Parameters** **field_name** –

**Returns**

**Return type** str

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_suggester_query_params**(*request*, *view*)
Get query params to be for suggestions.
**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.
**Return type** dict

**classmethod prepare_suggester_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

**serialize_queryset**(*queryset*, *suggester_name*, *value*, *serializer_field*)
Serialize queryset.

This shall be done here, since we don't want to delegate it to pagination.
**Parameters**

- **queryset** –
- **suggester_name** –
- **value** –
- **serializer_field** –

**Returns**

#### 12.10.1.2.1.23 django_elasticsearch_dsl_drf.filter_backends.suggester.native module

Suggesters backend.

It's assumed, that fields you're planning to query suggestions for have been properly indexed using `fields.CompletionField`.

Example:

```
>>> from django_elasticsearch_dsl import DocType, Index, fields
>>>
>>> from books.models import Publisher
>>>
>>> # Name of the Elasticsearch index
>>> PUBLISHER_INDEX = Index(PUBLISHER_INDEX_NAME)
```

(continues on next page)

(continued from previous page)

```
>>> # See Elasticsearch Indices API reference for available settings
>>> PUBLISHER_INDEX.settings(
>>>     number_of_shards=1,
>>>     number_of_replicas=1
>>> )
>>>
>>> @PUBLISHER_INDEX.doc_type
>>> class PublisherDocument(DocType):
>>>     "Publisher Elasticsearch document."
>>>
>>>     id = fields.IntegerField(attr='id')
>>>
>>>     name = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     info = fields.StringField()
>>>
>>>     address = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword')
>>>         }
>>>     )
>>>
>>>     city = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     state_province = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     country = fields.StringField(
>>>         fields={
>>>             'raw': fields.StringField(analyzer='keyword'),
>>>             'suggest': fields.CompletionField(),
>>>         }
>>>     )
>>>
>>>     website = fields.StringField()
>>>
>>>     class Meta(object):
>>>         "Meta options."
>>>
>>>         model = Publisher  # The model associate with this DocType
```

**class** django_elasticsearch_dsl_drf.filter_backends.suggester.native.**SuggesterFilterBackend**

Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.*

*filter_backends.mixins.FilterBackendMixin*

Suggester filter backend for Elasticsearch.

Suggestion functionality is exclusive. Once you have queried the `SuggesterFilterBackend`, the latter will transform your current search query into suggestion search query (which is very different). Therefore, always add it as the very last filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     SUGGESTER_TERM,
>>>     SUGGESTER_PHRASE,
>>>     SUGGESTER_COMPLETION,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     SuggesterFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet
>>>
>>> # Local PublisherDocument definition
>>> from .documents import PublisherDocument
>>>
>>> # Local PublisherDocument serializer
>>> from .serializers import PublisherDocumentSerializer
>>>
>>> class PublisherDocumentView(BaseDocumentViewSet):
>>>
>>>     document = PublisherDocument
>>>     serializer_class = PublisherDocumentSerializer
>>>     filter_backends = [
>>>         # ...
>>>         SuggesterFilterBackend,
>>>     ]
>>>     # Suggester fields
>>>     suggester_fields = {
>>>         'name_suggest': {
>>>             'field': 'name.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_TERM,
>>>                 SUGGESTER_PHRASE,
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'city_suggest': {
>>>             'field': 'city.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'state_province_suggest': {
>>>             'field': 'state_province.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'country_suggest': {
>>>             'field': 'country.suggest',
>>>             'suggesters': [
```

(continues on next page)

(continued from previous page)

```
>>>                     SUGGESTER_COMPLETION,
>>>                 ],
>>>             },
>>>         }
```

**classmethod apply_suggester_completion**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_phrase**(*suggester_name*, *queryset*, *options*, *value*)

Apply *phrase* suggester.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_term**(*suggester_name*, *queryset*, *options*, *value*)

Apply *term* suggester.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_suggester_query_params**(*request*, *view*)
Get query params to be for suggestions.
**Parameters**
- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.
**Return type** dict

**classmethod prepare_suggester_fields**(*view*)
Prepare filter fields.
**Parameters** **view**(*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

#### 12.10.1.2.1.24 Module contents

Suggester filtering backends.

**class** django_elasticsearch_dsl_drf.filter_backends.suggester.**SuggesterFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Suggester filter backend for Elasticsearch.

Suggestion functionality is exclusive. Once you have queried the SuggesterFilterBackend, the latter will transform your current search query into suggestion search query (which is very different). Therefore, always add it as the very last filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     SUGGESTER_TERM,
>>>     SUGGESTER_PHRASE,
>>>     SUGGESTER_COMPLETION,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     SuggesterFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet
>>>
>>> # Local PublisherDocument definition
>>> from .documents import PublisherDocument
>>>
>>> # Local PublisherDocument serializer
>>> from .serializers import PublisherDocumentSerializer
>>>
>>> class PublisherDocumentView(BaseDocumentViewSet):
>>>
>>>     document = PublisherDocument
>>>     serializer_class = PublisherDocumentSerializer
>>>     filter_backends = [
>>>         # ...
```

(continues on next page)

(continued from previous page)

```
>>>         SuggesterFilterBackend,
>>>     ]
>>>     # Suggester fields
>>>     suggester_fields = {
>>>         'name_suggest': {
>>>             'field': 'name.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_TERM,
>>>                 SUGGESTER_PHRASE,
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'city_suggest': {
>>>             'field': 'city.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'state_province_suggest': {
>>>             'field': 'state_province.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>         'country_suggest': {
>>>             'field': 'country.suggest',
>>>             'suggesters': [
>>>                 SUGGESTER_COMPLETION,
>>>             ],
>>>         },
>>>     }
```

**classmethod apply_suggester_completion**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_phrase**(*suggester_name*, *queryset*, *options*, *value*)

Apply *phrase* suggester.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_term**(*suggester_name*, *queryset*, *options*, *value*)
Apply *term* suggester.
**Parameters**
- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.
**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.
**Parameters**
- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_suggester_query_params**(*request*, *view*)
Get query params to be for suggestions.
**Parameters**
- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.
**Return type** dict

**classmethod prepare_suggester_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

**class** django_elasticsearch_dsl_drf.filter_backends.suggester.**FunctionalSuggesterFilterBacke**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Suggester filter backend for Elasticsearch.

Suggestion functionality is exclusive. Once you have queried the FunctionalSuggesterFilterBackend, the latter will transform your current search query into another search query (altered). Therefore, always add it as the very last filter backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.constants import (
>>>     FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>     FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>     FUNCTIONAL_SUGGESTER_PHRASE_MATCH,
>>>     FUNCTIONAL_SUGGESTER_PHRASE_MATCH,
>>>     FUNCTIONAL_SUGGESTER_TERM_MATCH,
>>> )
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     FunctionalSuggesterFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.viewsets import DocumentViewSet
>>>
>>> # Local PublisherDocument definition
>>> from .documents import PublisherDocument
>>>
>>> # Local PublisherDocument serializer
>>> from .serializers import PublisherDocumentSerializer
>>>
>>> class PublisherDocumentView(DocumentViewSet):
>>>
>>>     document = PublisherDocument
>>>     serializer_class = PublisherDocumentSerializer
>>>     filter_backends = [
>>>         # ...
>>>         FunctionalSuggesterFilterBackend,
>>>     ]
>>>     # Suggester fields
>>>     suggester_fields = {
>>>         'name_suggest': {
>>>             'field': 'name.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>         'city_suggest': {
>>>             'field': 'city.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>         'state_province_suggest': {
>>>             'field': 'state_province.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_MATCH,
>>>             ],
>>>         },
>>>         'country_suggest': {
>>>             'field': 'country.suggest',
>>>             'suggesters': [
>>>                 FUNCTIONAL_SUGGESTER_COMPLETION_PREFIX,
>>>             ],
>>>         },
>>>     }
```

**classmethod apply_suggester_completion_match**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester match.

This is effective when used with Ngram fields.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**classmethod apply_suggester_completion_prefix**(*suggester_name*, *queryset*, *options*, *value*)

Apply *completion* suggester prefix.

This is effective when used with Keyword fields.

**Parameters**

- **suggester_name** (*str*) –
- **queryset** (*elasticsearch_dsl.search.Search*) – Original queryset.
- **options** (*dict*) – Filter options.
- **value** (*str*) – value to filter on.

**Returns** Modified queryset.

**Return type** elasticsearch_dsl.search.Search

**clean_queryset**(*queryset*)

Clean the queryset.

- Remove aggregations.
- Remove highlight.
- Remove sorting options.

**Parameters** **queryset** –

**Returns**

**extract_field_name**(*field_name*)

Extract field name.

For instance, "name.suggest" or "name.raw" becomes "name".

**Parameters** **field_name** –

**Returns**

**Return type** str

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_suggester_query_params**(*request*, *view*)
Get query params to be for suggestions.
**Parameters**
- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Request query params to filter on.
**Return type** dict

**classmethod prepare_suggester_fields**(*view*)
Prepare filter fields.
**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Filtering options.
**Return type** dict

**serialize_queryset**(*queryset*, *suggester_name*, *value*, *serializer_field*)
Serialize queryset.

This shall be done here, since we don't want to delegate it to pagination.
**Parameters**
- **queryset** –
- **suggester_name** –
- **value** –
- **serializer_field** –

**Returns**

#### 12.10.1.2.2 Submodules

#### 12.10.1.2.3 django_elasticsearch_dsl_drf.filter_backends.faceted_search module

Faceted search backend.

**class** django_elasticsearch_dsl_drf.filter_backends.faceted_search.**FacetedSearchFilterBacke**
Bases: rest_framework.filters.BaseFilterBackend

Faceted search backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     FacetedSearchFilterBackend
>>> )
>>> from elasticsearch_dsl import TermsFacet, DateHistogramFacet
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
```

(continues on next page)

(continued from previous page)

```
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [FacetedSearchFilterBackend,]
>>>     faceted_search_fields = {
>>>         'title': 'title.raw',  # Uses `TermsFacet` by default
>>>         'state': {
>>>             'field': 'state.raw',
>>>             'facet': TermsFacet,
>>>         },
>>>         'publisher': {
>>>             'field': 'publisher.raw',
>>>             'facet': TermsFacet,
>>>             'enabled': False,
>>>         },
>>>         'date_published': {
>>>             'field': 'date_published.raw',
>>>             'facet': DateHistogramFacet,
>>>             'options': {
>>>                 'interval': 'month',
>>>             },
>>>             'enabled': True,
>>>         },
>>>
>>>     }
```

Facets make queries to be more heavy. That's why by default all facets are disabled and enabled only explicitly either in the filter options (*enabled* set to True) or via query params *?facet=state&facet=date_published*.

**aggregate**(*request*, *queryset*, *view*)

Aggregate.

**Parameters**

- **request** –
- **queryset** –
- **view** –

**Returns**

**construct_facets**(*request*, *view*)

Construct facets.

Turns the following structure:

```
>>> {
>>>     'publisher': {
>>>         'field': 'publisher.raw',
>>>         'facet': TermsFacet,
>>>         'enabled': False,
>>>     }
>>>     'date_published': {
>>>         'field': 'date_published',
>>>         'facet': DateHistogramFacet,
>>>         'options': {
>>>             'interval': 'month',
>>>         },
>>>         'enabled': True,
```

(continues on next page)

(continued from previous page)

```
>>>     },
>>> }
```

Into the following structure:

```
>>> {
>>>     'publisher': TermsFacet(field='publisher.raw'),
>>>     'publishing_frequency': DateHistogramFacet(
>>>         field='date_published.raw',
>>>         interval='month'
>>>     ),
>>> }
```

**faceted_search_param = 'facet'**

**filter_queryset**(*request*, *queryset*, *view*)

Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.

**Return type** elasticsearch_dsl.search.Search

**get_faceted_search_query_params**(*request*)

Get faceted search query params.

**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.

**Returns** List of search query params.

**Return type** list

**classmethod prepare_faceted_search_fields**(*view*)

Prepare faceted search fields.

Prepares the following structure:

```
>>> {
>>>     'publisher': {
>>>         'field': 'publisher.raw',
>>>         'facet': TermsFacet,
>>>         'enabled': False,
>>>     }
>>>     'date_published': {
>>>         'field': 'date_published.raw',
>>>         'facet': DateHistogramFacet,
>>>         'options': {
>>>             'interval': 'month',
>>>         },
>>>         'enabled': True,
>>>     },
>>> }
```

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –

**Returns** Faceted search fields options.

**Return type** dict

#### 12.10.1.2.4 django_elasticsearch_dsl_drf.filter_backends.highlight module

Highlight backend.

**class** django_elasticsearch_dsl_drf.filter_backends.highlight.**HighlightBackend**
Bases: rest_framework.filters.BaseFilterBackend

Highlight backend.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     HighlightBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [HighlightBackend,]
>>>     highlight_fields = {
>>>         'author.name': {
>>>             'enabled': False,
>>>             'options': {
>>>                 'fragment_size': 150,
>>>                 'number_of_fragments': 3
>>>             }
>>>         }
>>>         'title': {
>>>             'options': {
>>>                 'pre_tags' : ["<em>"],
>>>                 'post_tags' : ["</em>"]
>>>             },
>>>             'enabled': True,
>>>         },
>>>     }
```

Highlight make queries to be more heavy. That's why by default all highlights are disabled and enabled only explicitly either in the filter options (*enabled* set to True) or via query params *?highlight=author.name&highlight=title*.

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.

- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_highlight_query_params**(*request*)
Get highlight query params.
**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.
**Returns** List of search query params.
**Return type** list

**highlight_param = 'highlight'**

**classmethod prepare_highlight_fields**(*view*)
Prepare faceted search fields.

Prepares the following structure:

```
>>> {
>>>     'author.name': {
>>>         'enabled': False,
>>>         'options': {
>>>             'fragment_size': 150,
>>>             'number_of_fragments': 3
>>>         }
>>>     }
>>>     'title': {
>>>         'options': {
>>>             'pre_tags' : ["<em>"],
>>>             'post_tags' : ["</em>"]
>>>         },
>>>         'enabled': True,
>>>     },
>>> }
```

**Parameters** **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) –
**Returns** Highlight fields options.
**Return type** dict

#### 12.10.1.2.5 django_elasticsearch_dsl_drf.filter_backends.mixins module

Mixins.

**class** django_elasticsearch_dsl_drf.filter_backends.mixins.**FilterBackendMixin**
Bases: object

Filter backend mixin.

**classmethod apply_filter**(*queryset*, *options=None*, *args=None*, *kwargs=None*)
Apply filter.
**Parameters**

- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod apply_query**(*queryset*, *options=None*, *args=None*, *kwargs=None*)
Apply query.
**Parameters**
- **queryset** –
- **options** –
- **args** –
- **kwargs** –

**Returns**

**classmethod split_lookup_complex_multiple_value**(*value*, *maxsplit=-1*)
Split lookup complex multiple value.
**Parameters**
- **value** (*str*) – Value to split.
- **maxsplit** (*int*) – The *maxsplit* option of *string.split*.

**Returns** Lookup filter split into a list.
**Return type** list

**classmethod split_lookup_complex_value**(*value*, *maxsplit=-1*)
Split lookup complex value.
**Parameters**
- **value** (*str*) – Value to split.
- **maxsplit** (*int*) – The *maxsplit* option of *string.split*.

**Returns** Lookup filter split into a list.
**Return type** list

**classmethod split_lookup_filter**(*value*, *maxsplit=-1*)
Split lookup filter.
**Parameters**
- **value** (*str*) – Value to split.
- **maxsplit** (*int*) – The *maxsplit* option of *string.split*.

**Returns** Lookup filter split into a list.
**Return type** list

**classmethod split_lookup_name**(*value*, *maxsplit=-1*)
Split lookup value.
**Parameters**
- **value** (*str*) – Value to split.
- **maxsplit** (*int*) – The *maxsplit* option of *string.split*.

**Returns** Lookup value split into a list.
**Return type** list

#### 12.10.1.2.6 django_elasticsearch_dsl_drf.filter_backends.search module

Search filter backends.

**class** django_elasticsearch_dsl_drf.filter_backends.search.**BaseSearchFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Base search filter backend.

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.
**Parameters**
- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_coreschema_field**(*field*)

**get_query_backends**(*request*, *view*)
Get query backends.
**Returns**

**get_schema_fields**(*view*)

**get_search_query_params**(*request*)
Get search query params.
**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.
**Returns** List of search query params.
**Return type** list

**matching = 'should'**

**query_backends = []**

**search_param = 'search'**

**class** django_elasticsearch_dsl_drf.filter_backends.search.**CompoundSearchFilterBackend**
Bases: django_elasticsearch_dsl_drf.filter_backends.search.base.BaseSearchFilterBackend

Compound search backend.

**query_backends = [<class 'django_elasticsearch_dsl_drf.filter_backends.search.query_ba**

**class** django_elasticsearch_dsl_drf.filter_backends.search.**MultiMatchSearchFilterBackend**
Bases: django_elasticsearch_dsl_drf.filter_backends.search.base.BaseSearchFilterBackend

Multi match search filter backend.

**matching = 'must'**

**query_backends = [<class 'django_elasticsearch_dsl_drf.filter_backends.search.query_ba**

**search_param = 'search_multi_match'**

**class** django_elasticsearch_dsl_drf.filter_backends.search.**SearchFilterBackend**
Bases: rest_framework.filters.BaseFilterBackend, *django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin*

Search filter backend for Elasticsearch.

Example:

```
>>> from django_elasticsearch_dsl_drf.filter_backends import (
>>>     SearchFilterBackend
>>> )
>>> from django_elasticsearch_dsl_drf.views import BaseDocumentViewSet
>>>
>>> # Local article document definition
>>> from .documents import ArticleDocument
>>>
>>> # Local article document serializer
>>> from .serializers import ArticleDocumentSerializer
>>>
>>> class ArticleDocumentView(BaseDocumentViewSet):
>>>
>>>     document = ArticleDocument
>>>     serializer_class = ArticleDocumentSerializer
>>>     filter_backends = [SearchFilterBackend,]
>>>     search_fields = (
>>>         'title',
>>>         'content',
>>>     )
>>>     search_nested_fields = {
>>>         'state': ['name'],
>>>         'documents.author': ['title', 'description'],
>>>     }
```

**construct_nested_search** (*request*, *view*)

Construct nested search.

We have to deal with two types of structures:

Type 1:

```
>>> search_nested_fields = {
>>>     'country': {
>>>         'path': 'country',
>>>         'fields': ['name'],
>>>     },
>>>     'city': {
>>>         'path': 'country.city',
>>>         'fields': ['name'],
>>>     },
>>> }
```

Type 2:

```
>>> search_nested_fields = {
>>>     'country': {
>>>         'path': 'country',
>>>         'fields': [{'name': {'boost': 2}}]
>>>     },
>>>     'city': {
>>>         'path': 'country.city',
>>>         'fields': [{'name': {'boost': 2}}]
>>>     },
>>> }
```

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST

framework request.

- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**construct_search**(*request*, *view*)
Construct search.

We have to deal with two types of structures:

Type 1:

```
>>> search_fields = (
>>>     'title',
>>>     'description',
>>>     'summary',
>>> )
```

Type 2:

```
>>> search_fields = {
>>>     'title': {'boost': 2},
>>>     'description': None,
>>>     'summary': None,
>>> }
```

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**filter_queryset**(*request*, *queryset*, *view*)
Filter the queryset.

**Parameters**

- **request** (*rest_framework.request.Request*) – Django REST framework request.
- **queryset** (*elasticsearch_dsl.search.Search*) – Base queryset.
- **view** (*rest_framework.viewsets.ReadOnlyModelViewSet*) – View.

**Returns** Updated queryset.
**Return type** elasticsearch_dsl.search.Search

**get_coreschema_field**(*field*)

**get_schema_fields**(*view*)

**get_search_query_params**(*request*)
Get search query params.

**Parameters** **request** (*rest_framework.request.Request*) – Django REST framework request.
**Returns** List of search query params.
**Return type** list

**search_param = 'search'**

**class** django_elasticsearch_dsl_drf.filter_backends.search.**SimpleQueryStringSearchFilterBack**
Bases: django_elasticsearch_dsl_drf.filter_backends.search.base.BaseSearchFilterBackend

Simple query string search filter backend.

**matching = 'must'**

**query_backends = [<class 'django_elasticsearch_dsl_drf.filter_backends.search.query_ba**

**search_param = 'search_simple_query_string'**

#### 12.10.1.2.7 Module contents

All filter backends.

### 12.10.1.3 django_elasticsearch_dsl_drf.tests package

#### 12.10.1.3.1 Submodules

### 12.10.1.3.2 django_elasticsearch_dsl_drf.tests.base module

Base tests.

**class** django_elasticsearch_dsl_drf.tests.base.**BaseRestFrameworkTestCase**(*methodName='runTest'*)
Bases: django.test.testcases.TransactionTestCase

Base REST framework test case.

**authenticate**()
Helper for logging in Genre Coordinator user.
**Returns**

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**class** django_elasticsearch_dsl_drf.tests.base.**BaseTestCase**(*methodName='runTest'*)
Bases: django.test.testcases.TransactionTestCase

Base test case.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

#### 12.10.1.3.3 django_elasticsearch_dsl_drf.tests.data_mixins module

Data mixins.

**class** django_elasticsearch_dsl_drf.tests.data_mixins.**AddressesMixin**
Bases: object

Addresses mixin.

**classmethod created_addresses**()
Create addresses.
**Returns**

**class** django_elasticsearch_dsl_drf.tests.data_mixins.**BooksMixin**
Bases: object

Books mixin.

**classmethod create_books**()
Create books.
**Returns**

#### 12.10.1.3.4 django_elasticsearch_dsl_drf.tests.test_faceted_search module

Test faceted search backend.

**class** django_elasticsearch_dsl_drf.tests.test_faceted_search.**TestFacetedSearch**(*methodName='r*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test faceted search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_list_results_with_facets**()
Test list results with facets.

#### 12.10.1.3.5 django_elasticsearch_dsl_drf.tests.test_filtering_common module

Test filtering backend.

**class** django_elasticsearch_dsl_drf.tests.test_filtering_common.**TestFilteringCommon**(*methodNa*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*, *django_elasticsearch_dsl_drf.tests.data_mixins.BooksMixin*

Test filtering common.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_default_filter_lookup**()
Test default filter lookup.

Example:
**http://localhost:8000/search/books-default-filter-lookup/** ?authors=Robin&authors=Luc

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_gt**()
Field filter gt.

Example:
http://localhost:8000/api/users/?id__gt=10

**Returns**

**test_field_filter_gt_with_boost**()
Field filter gt with boost.

Example:
http://localhost:8000/api/users/?id__gt=10__2.0

**Returns**

**test_field_filter_gte**()
Field filter gte.

Example:
http://localhost:8000/api/users/?id__gte=10

**Returns**

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1__2__3

**test_field_filter_isnull_false**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?tags__isnull=false

**test_field_filter_isnull_true**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?null_field__isnull=true

**test_field_filter_lt**()
Field filter lt.

Example:
http://localhost:8000/api/users/?id__lt=10

**Returns**

**test_field_filter_lt_with_boost**()
Field filter lt with boost.

Example:
http://localhost:8000/api/users/?id__lt=10__2.0

**Returns**

**test_field_filter_lte**()
Field filter lte.

Example:
http://localhost:8000/api/users/?id__lte=10

**Returns**

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67__2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1__2__3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_ids_filter**()
Test ids filter.

Example:
http://localhost:8000/api/articles/?ids=68__64__58 http://localhost:8000/api/articles/?ids=68&ids=64&ids=58

**test_nested_field_filter_term**()
Nested field filter term.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_various_complex_fields**()
Test various complex fields.
**Returns**

#### 12.10.1.3.6 django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial module

Test geo-spatial filtering backend.

**class** django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.**TestFilteringGeoSpatial**
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test filtering geo-spatial.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_geo_bounding_box**()
Test field filter geo-bounding-box.
**Returns**

**test_field_filter_geo_bounding_box_fail_test**()
Test field filter geo-bounding-box (fail test).
**Returns**

**test_field_filter_geo_distance**()
Field filter geo-distance.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549__2.3000

**Returns**

**test_field_filter_geo_distance_distance_type_arc**()
Field filter geo-distance.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549__2.3000__arc

**Returns**

**test_field_filter_geo_distance_not_enough_args_fail**()
Field filter geo-distance. Fail test on not enough args.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549

**Returns**

**test_field_filter_geo_polygon**()
Test field filter geo-polygon.
**Returns**

**test_field_filter_geo_polygon_fail_test**()
Test field filter geo-polygon (fail test).
**Returns**

**test_field_filter_geo_polygon_string_options**()
Test field filter geo-polygon.
**Returns**

**test_field_filter_geo_polygon_string_options_fail_test**()
Test field filter geo-polygon (fail test).
**Returns**

#### 12.10.1.3.7 django_elasticsearch_dsl_drf.tests.test_filtering_nested module

Test nested filtering backend.

**class** django_elasticsearch_dsl_drf.tests.test_filtering_nested.**TestFilteringNested**(*methodNa*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test filtering nested.

**base_url**

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1__2__3

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67__2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1__2__3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

#### 12.10.1.3.8 django_elasticsearch_dsl_drf.tests.test_filtering_post_filter module

Test filtering *post_filter* backend.

**class** django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.**TestFilteringPostFilter**
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*, *django_elasticsearch_dsl_drf.tests.data_mixins.BooksMixin*

Test filtering *post_filter*.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_gt**()
Field filter gt.

Example:
http://localhost:8000/api/users/?id__gt=10

**Returns**

**test_field_filter_gt_with_boost**()
Field filter gt with boost.

Example:
http://localhost:8000/api/users/?id__gt=10;2.0

**Returns**

**test_field_filter_gte**()
Field filter gte.

Example:
http://localhost:8000/api/users/?id__gte=10

**Returns**

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1;2;3

**test_field_filter_isnull_false**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?tags__isnull=false

**test_field_filter_isnull_true**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?null_field__isnull=true

**test_field_filter_lt**()
Field filter lt.

Example:
http://localhost:8000/api/users/?id__lt=10

**Returns**

**test_field_filter_lt_with_boost**()
Field filter lt with boost.

Example:
http://localhost:8000/api/users/?id__lt=10;2.0

**Returns**

**test_field_filter_lte**()
Field filter lte.

Example:
http://localhost:8000/api/users/?id__lte=10

**Returns**

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16;67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16;67;2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1;2;3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_ids_filter**()
Test ids filter.

Example:
http://localhost:8000/api/articles/?ids=68;64;58 http://localhost:8000/api/articles/?ids=68&ids=64&ids=58

**test_list_results_with_facets**()
Test list results with facets.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_various_complex_fields**()
Test various complex fields.
**Returns**

#### 12.10.1.3.9 django_elasticsearch_dsl_drf.tests.test_functional_suggesters module

Test functional suggestions backend.

**class** django_elasticsearch_dsl_drf.tests.test_functional_suggesters.**TestFunctionalSuggester**
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test functional suggesters.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_suggesters_completion**()
Test suggesters completion.

**test_suggesters_completion_no_args_provided**()
Test suggesters completion with no args provided.

#### 12.10.1.3.10 django_elasticsearch_dsl_drf.tests.test_helpers module

Test helpers.

**class** django_elasticsearch_dsl_drf.tests.test_helpers.**TestHelpers**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseTestCase*

Test helpers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_filter_by_field**()
Filter by field.

#### 12.10.1.3.11 django_elasticsearch_dsl_drf.tests.test_highlight module

Test highlight backend.

**class** django_elasticsearch_dsl_drf.tests.test_highlight.**TestHighlight**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test highlight.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_list_results_with_highlights**()
Test list results with facets.

#### 12.10.1.3.12 django_elasticsearch_dsl_drf.tests.test_ordering_common module

Test ordering backend.

**class** django_elasticsearch_dsl_drf.tests.test_ordering_common.**TestOrdering**(*methodName='runTest*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test ordering.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_author_default_order_by**()
Author order by default.

**test_author_order_by_field_id_ascending**()
Order by field *name* ascending.

**test_author_order_by_field_id_descending**()
Order by field *id* descending.

**test_author_order_by_field_name_ascending**()
Order by field *name* ascending.

**test_author_order_by_field_name_descending**()
Order by field *name* descending.

**test_book_default_order_by**()
Book order by default.

**test_book_order_by_field_id_ascending**()
Order by field *id* ascending.

**test_book_order_by_field_id_descending**()
Order by field *id* descending.

**test_book_order_by_field_title_ascending**()
Order by field *title* ascending.

**test_book_order_by_field_title_descending**()
Order by field *title* descending.

**test_book_order_by_non_existent_field**()
Order by non-existent field.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

#### 12.10.1.3.13 django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial module

Test geo-spatial ordering filter backend.

**class** django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial.**TestOrderingGeoSpatial**(*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test ordering geo-spatial.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_geo_distance**()
Field filter geo_distance.

Example:
http://localhost:8000 /api/publisher/?ordering=location;48.85;2.30;km;plane

#### 12.10.1.3.14 django_elasticsearch_dsl_drf.tests.test_pagination module

Test pagination.

**class** django_elasticsearch_dsl_drf.tests.test_pagination.**TestPagination**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test pagination.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_pagination**()
Test pagination.

#### 12.10.1.3.15 django_elasticsearch_dsl_drf.tests.test_search module

Test search backend.

**class** django_elasticsearch_dsl_drf.tests.test_search.**TestSearch**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_compound_search_boost_by_field**()

**test_compound_search_by_field**()

**test_compound_search_by_field_multi_terms**()

**test_compound_search_by_nested_field**()

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_search_boost**(*url=None*, *search_field='search'*)
Search boost.
**Returns**

**test_search_boost_compound**(*search_field='search'*)

**test_search_by_field**(*url=None*, *search_field='search'*)
Search by field.

**test_search_by_field_multi_terms**(*url=None*, *search_field='search'*)
Search by field, multiple terms.

**test_search_by_nested_field**(*url=None*)
Search by field.

#### 12.10.1.3.16 django_elasticsearch_dsl_drf.tests.test_serializers module

Test serializers.

**class** django_elasticsearch_dsl_drf.tests.test_serializers.**TestSerializers**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test serializers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**test_serializer_document_equals_to_none**()
Test serializer no document specified.

**test_serializer_fields_and_exclude**()
Test serializer fields and exclude.

**test_serializer_meta_del_attr**()
Test serializer set attr.

**test_serializer_meta_set_attr**()
Test serializer set attr.

**test_serializer_no_document_specified**()
Test serializer no document specified.

#### 12.10.1.3.17 django_elasticsearch_dsl_drf.tests.test_suggesters module

Test suggestions backend.

**class** django_elasticsearch_dsl_drf.tests.test_suggesters.**TestSuggesters**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test suggesters.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_nested_fields_suggesters_completion**()
Test suggesters completion for nested fields.

**test_suggesters_completion**()
Test suggesters completion.

**test_suggesters_completion_no_args_provided**()
Test suggesters completion with no args provided.

**test_suggesters_phrase**()
Test suggesters phrase.

**test_suggesters_term**()
Test suggesters term.

**class** django_elasticsearch_dsl_drf.tests.test_suggesters.**TestSuggestersEmptyIndex**(*methodName*
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test suggesters on empty index.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_suggesters_on_empty_index**()
Test suggesters phrase.

#### 12.10.1.3.18 django_elasticsearch_dsl_drf.tests.test_views module

Test views.

**class** django_elasticsearch_dsl_drf.tests.test_views.**TestViews**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test views.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_detail_view**()
Test detail view.

**test_listing_view**()
Test listing view.

#### 12.10.1.3.19 django_elasticsearch_dsl_drf.tests.test_wrappers module

Test wrappers.

**class** django_elasticsearch_dsl_drf.tests.test_wrappers.**TestWrappers**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test helpers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={})]**

**classmethod setUpClass**()
Hook method for setting up class fixture before running tests in the class.

**test_dict_to_obj**()
Test *dict_to_obj*.
**Returns**

**test_obj_to_dict**()
Test *obj_to_dict*.
**Returns**

**test_wrapper_as_json**()
Test **:Wrapper:‘as_json‘** property.

#### 12.10.1.3.20 Module contents

Tests.

**class** django_elasticsearch_dsl_drf.tests.**TestFacetedSearch**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test faceted search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_list_results_with_facets**()
Test list results with facets.

**class** django_elasticsearch_dsl_drf.tests.**TestFilteringCommon**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*, *django_elasticsearch_dsl_drf.tests.data_mixins.BooksMixin*

Test filtering common.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_default_filter_lookup**()
Test default filter lookup.

Example:
**http://localhost:8000/search/books-default-filter-lookup/** ?authors=Robin&authors=Luc

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_gt**()
Field filter gt.

Example:
http://localhost:8000/api/users/?id__gt=10

**Returns**

**test_field_filter_gt_with_boost**()
Field filter gt with boost.

Example:
http://localhost:8000/api/users/?id__gt=10__2.0

**Returns**

**test_field_filter_gte**()
Field filter gte.

Example:
http://localhost:8000/api/users/?id__gte=10

**Returns**

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1__2__3

**test_field_filter_isnull_false**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?tags__isnull=false

**test_field_filter_isnull_true**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?null_field__isnull=true

**test_field_filter_lt**()
Field filter lt.

Example:
http://localhost:8000/api/users/?id__lt=10

**Returns**

**test_field_filter_lt_with_boost**()
Field filter lt with boost.

Example:
http://localhost:8000/api/users/?id__lt=10__2.0

**Returns**

**test_field_filter_lte**()
Field filter lte.

Example:
http://localhost:8000/api/users/?id__lte=10

**Returns**

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67__2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1__2__3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_ids_filter**()
Test ids filter.

Example:
http://localhost:8000/api/articles/?ids=68__64__58 http://localhost:8000/api/articles/?ids=68&ids=64&ids=58

**test_nested_field_filter_term**()
Nested field filter term.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_various_complex_fields**()
Test various complex fields.
**Returns**

**class** django_elasticsearch_dsl_drf.tests.**TestFilteringGeoSpatial**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test filtering geo-spatial.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_geo_bounding_box**()
Test field filter geo-bounding-box.
**Returns**

**test_field_filter_geo_bounding_box_fail_test**()
Test field filter geo-bounding-box (fail test).
**Returns**

**test_field_filter_geo_distance**()
Field filter geo-distance.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549__2.3000

**Returns**

**test_field_filter_geo_distance_distance_type_arc**()
Field filter geo-distance.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549__2.3000__arc

**Returns**

**test_field_filter_geo_distance_not_enough_args_fail**()
Field filter geo-distance. Fail test on not enough args.

Example:
http://localhost:8000 /api/publisher/?location__geo_distance=1km__48.8549

**Returns**

**test_field_filter_geo_polygon**()
Test field filter geo-polygon.
**Returns**

**test_field_filter_geo_polygon_fail_test**()
Test field filter geo-polygon (fail test).
**Returns**

**test_field_filter_geo_polygon_string_options**()
Test field filter geo-polygon.
**Returns**

**test_field_filter_geo_polygon_string_options_fail_test**()
Test field filter geo-polygon (fail test).
**Returns**

**class** django_elasticsearch_dsl_drf.tests.**TestFilteringNested**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test filtering nested.

**base_url**

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1__2__3

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16__67__2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1__2__3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**class** django_elasticsearch_dsl_drf.tests.**TestFilteringPostFilter**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*, *django_elasticsearch_dsl_drf.tests.data_mixins.BooksMixin*

Test filtering *post_filter*.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_contains**()
Test filter contains.

Example:
http://localhost:8000/api/articles/?state__contains=lishe

**test_field_filter_endswith**()
Test filter endswith.

Example:
http://localhost:8000/api/articles/?state__endswith=lished

**test_field_filter_exclude**()
Test filter exclude.

Example:
http://localhost:8000/api/articles/?tags__exclude=children

**test_field_filter_exists_false**()
Test filter exists.

Example:
http://localhost:8000/api/articles/?non_existent__exists=false

**test_field_filter_exists_true**()
Test filter exists true.

Example:
http://localhost:8000/api/articles/?tags__exists=true

**test_field_filter_gt**()
Field filter gt.

Example:
http://localhost:8000/api/users/?id__gt=10

**Returns**

**test_field_filter_gt_with_boost**()
Field filter gt with boost.

Example:
http://localhost:8000/api/users/?id__gt=10;2.0

**Returns**

**test_field_filter_gte**()
Field filter gte.

Example:
http://localhost:8000/api/users/?id__gte=10

**Returns**

**test_field_filter_in**()
Test filter in.

Example:
http://localhost:8000/api/articles/?id__in=1;2;3

**test_field_filter_isnull_false**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?tags__isnull=false

**test_field_filter_isnull_true**()
Test filter isnull true.

Example:
http://localhost:8000/api/articles/?null_field__isnull=true

**test_field_filter_lt**()
Field filter lt.

Example:
http://localhost:8000/api/users/?id__lt=10

**Returns**

**test_field_filter_lt_with_boost**()
Field filter lt with boost.

Example:
http://localhost:8000/api/users/?id__lt=10;2.0

**Returns**

**test_field_filter_lte**()
Field filter lte.

Example:
http://localhost:8000/api/users/?id__lte=10

**Returns**

**test_field_filter_prefix**()
Test filter prefix.

Example:
http://localhost:8000/api/articles/?tags__prefix=bio

**test_field_filter_range**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16;67

**test_field_filter_range_with_boost**()
Field filter range.

Example:
http://localhost:8000/api/users/?age__range=16;67;2.0

**test_field_filter_term**()
Field filter term.

**test_field_filter_term_explicit**()
Field filter term.

**test_field_filter_terms_list**()
Test filter terms.

**test_field_filter_terms_string**()
Test filter terms.

Example:
http://localhost:8000/api/articles/?id__terms=1;2;3

**test_field_filter_wildcard**()
Test filter wildcard.

Example:
http://localhost:8000/api/articles/?title__wildcard=*elusional*

**test_ids_filter**()
Test ids filter.

Example:
http://localhost:8000/api/articles/?ids=68;64;58 http://localhost:8000/api/articles/?ids=68&ids=64&ids=58

**test_list_results_with_facets**()
Test list results with facets.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_various_complex_fields**()
Test various complex fields.
**Returns**

**class** django_elasticsearch_dsl_drf.tests.**TestFunctionalSuggesters**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test functional suggesters.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_suggesters_completion**()
Test suggesters completion.

**test_suggesters_completion_no_args_provided**()
Test suggesters completion with no args provided.

**class** django_elasticsearch_dsl_drf.tests.**TestHelpers**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseTestCase*

Test helpers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_filter_by_field**()
Filter by field.

**class** django_elasticsearch_dsl_drf.tests.**TestHighlight**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test highlight.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_list_results_with_highlights**()
Test list results with facets.

**class** django_elasticsearch_dsl_drf.tests.**TestMultiMatchSearch**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test multi match search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_search**(*url=None*)
Search.

**test_search_boost**(*url=None*)
Search boost.
**Returns**

**test_search_boost_selected_fields**(*url=None*)
Search boost.
**Returns**

**test_search_selected_fields**(*url=None*)
Search boost.
**Returns**

**class** django_elasticsearch_dsl_drf.tests.**TestSimpleQueryStringSearch**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test simple query string search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_search_boost_selected_fields**(*url=None*)
Search boost.
**Returns**

**test_search_selected_fields**(*url=None*)
Search boost.
**Returns**

**test_search_with_quotes**(*url=None*)
Search with quotes.

**test_search_with_quotes_alternative**()
Test search by field.
**Parameters** **url** –
**Returns**

**test_search_with_quotes_boost**(*url=None*)
Search boost.
**Returns**

**test_search_with_quotes_boost_alternative**()
Search boost.
**Returns**

**test_search_without_quotes**(*url=None*)
Test search without quotes. This does not work on Elasticsearch 6.x.
**Parameters** **url** –
**Returns**

**test_search_without_quotes_boost**(*url=None*)
Search boost without quotes. Does not work on Elasticsearch 6.x.
**Returns**

**class** django_elasticsearch_dsl_drf.tests.**TestOrdering**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test ordering.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_author_default_order_by**()
Author order by default.

**test_author_order_by_field_id_ascending**()
Order by field *name* ascending.

**test_author_order_by_field_id_descending**()
Order by field *id* descending.

**test_author_order_by_field_name_ascending**()
Order by field *name* ascending.

**test_author_order_by_field_name_descending**()
Order by field *name* descending.

**test_book_default_order_by**()
Book order by default.

**test_book_order_by_field_id_ascending**()
Order by field *id* ascending.

**test_book_order_by_field_id_descending**()
Order by field *id* descending.

**test_book_order_by_field_title_ascending**()
Order by field *title* ascending.

**test_book_order_by_field_title_descending**()
Order by field *title* descending.

**test_book_order_by_non_existent_field**()
Order by non-existent field.

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**class** django_elasticsearch_dsl_drf.tests.**TestOrderingGeoSpatial**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test ordering geo-spatial.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up.

**test_field_filter_geo_distance**()
Field filter geo_distance.

Example:
http://localhost:8000 /api/publisher/?ordering=location;48.85;2.30;km;plane

**class** django_elasticsearch_dsl_drf.tests.**TestPagination**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test pagination.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_pagination**()
Test pagination.

**class** django_elasticsearch_dsl_drf.tests.**TestSearch**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test search.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUp**()
Hook method for setting up the test fixture before exercising it.

**test_compound_search_boost_by_field**()

**test_compound_search_by_field**()

**test_compound_search_by_field_multi_terms**()

**test_compound_search_by_nested_field**()

**test_schema_field_not_required**()
Test schema fields always not required

**test_schema_fields_with_filter_fields_list**()
Test schema field generator

**test_search_boost**(*url=None*, *search_field='search'*)
Search boost.
**Returns**

**test_search_boost_compound**(*search_field='search'*)

**test_search_by_field**(*url=None*, *search_field='search'*)
Search by field.

**test_search_by_field_multi_terms**(*url=None*, *search_field='search'*)
Search by field, multiple terms.

**test_search_by_nested_field**(*url=None*)
Search by field.

**class** django_elasticsearch_dsl_drf.tests.**TestSerializers**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test serializers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**test_serializer_document_equals_to_none**()
Test serializer no document specified.

**test_serializer_fields_and_exclude**()
Test serializer fields and exclude.

**test_serializer_meta_del_attr**()
Test serializer set attr.

**test_serializer_meta_set_attr**()
Test serializer set attr.

**test_serializer_no_document_specified**()
Test serializer no document specified.

**class** django_elasticsearch_dsl_drf.tests.**TestSuggesters**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*, *django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin*

Test suggesters.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_nested_fields_suggesters_completion**()
Test suggesters completion for nested fields.

**test_suggesters_completion**()
Test suggesters completion.

**test_suggesters_completion_no_args_provided**()
Test suggesters completion with no args provided.

**test_suggesters_phrase**()
Test suggesters phrase.

**test_suggesters_term**()
Test suggesters term.

**class** django_elasticsearch_dsl_drf.tests.**TestViews**(*methodName='runTest'*)
Bases: *django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase*

Test views.

**pytestmark = [Mark(name='django_db', args=(), kwargs={}), Mark(name='django_db', args=**

**classmethod setUpClass**()
Set up class.

**test_detail_view**()
Test detail view.

**test_listing_view**()
Test listing view.

**class** django_elasticsearch_dsl_drf.tests.**TestWrappers**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Test helpers.

**pytestmark = [Mark(name='django_db', args=(), kwargs={})]**

**classmethod setUpClass**()
Hook method for setting up class fixture before running tests in the class.

**test_dict_to_obj**()
Test *dict_to_obj*.
**Returns**

**test_obj_to_dict**()
Test *obj_to_dict*.
**Returns**

**test_wrapper_as_json**()
Test **:Wrapper:'as_json'** property.

### 12.10.2 Submodules

### 12.10.3 django_elasticsearch_dsl_drf.analyzers module

Analyzers.

### 12.10.4 django_elasticsearch_dsl_drf.apps module

Apps.

**class** django_elasticsearch_dsl_drf.apps.**Config**(*app_name*, *app_module*)
Bases: django.apps.config.AppConfig

Config.

**label = 'django_elasticsearch_dsl_drf'**

**name = 'django_elasticsearch_dsl_drf'**

### 12.10.5 django_elasticsearch_dsl_drf.compat module

Transitional compatibility module. Contains various field wrappers and helpers for painless (testing of) Elastic 2.x to Elastic 5.x transition. This module is not supposed to solve all transition issues for you. Better move to Elastic 5.x as soon as possible.

django_elasticsearch_dsl_drf.compat.**get_elasticsearch_version**(*default=(2, 0, 0)*)
Get Elasticsearch version.

**Parameters** **default** (*tuple*) – Default value. Mainly added for building the docs when Elasticsearch is not running.
**Returns**
**Return type** list

django_elasticsearch_dsl_drf.compat.**KeywordField**(***kwargs*)
Keyword field.
**Parameters** **kwargs** –
**Returns**

django_elasticsearch_dsl_drf.compat.**StringField**(***kwargs*)
String field.
**Parameters** **kwargs** –
**Returns**

### 12.10.6 django_elasticsearch_dsl_drf.constants module

Constants module. Contains Elasticsearch constants, lookup constants, functional constants, suggesters, etc.

### 12.10.7 django_elasticsearch_dsl_drf.helpers module

Helpers.

django_elasticsearch_dsl_drf.helpers.**get_document_for_model**(*model*)
Get document for model given.
**Parameters** **model** (Subclass of *django.db.models.Model*.) – Model to get document index for.
**Returns** Document index for the given model.
**Return type** Subclass of *django_elasticsearch_dsl.DocType*.

django_elasticsearch_dsl_drf.helpers.**get_index_and_mapping_for_model**(*model*)
Get index and mapping for model.
**Parameters** **model** (Subclass of *django.db.models.Model*.) – Django model for which to get index and mapping for.
**Returns** Index and mapping values.
**Return type** tuple.

django_elasticsearch_dsl_drf.helpers.**more_like_this**(*obj*, *fields*, *max_query_terms=25*, *min_term_freq=2*, *min_doc_freq=5*, *max_doc_freq=0*, *query=None*)
More like this.

https://www.elastic.co/guide/en/elasticsearch/reference/current/ query-dsl-mlt-query.html
**Parameters**
- **obj** (Instance of *django.db.models.Model* (sub-classed) model.) – Django model instance for which similar objects shall be found.
- **fields** (*list*) – Fields to search in.
- **max_query_terms** (*int*) –
- **min_term_freq** (*int*) –
- **min_doc_freq** (*int*) –
- **max_doc_freq** (*int*) –
- **query** (*elasticsearch_dsl.query.Q*) – Q query

**Returns** List of objects.
**Return type** elasticsearch_dsl.search.Search

Example:

```
>>> from django_elasticsearch_dsl_drf.helpers import more_like_this
>>> from books.models import Book
>>> book = Book.objects.first()
>>> similar_books = more_like_this(
>>>     book,
>>>     ['title', 'description', 'summary']
>>> )
```

django_elasticsearch_dsl_drf.helpers.**sort_by_list**(*unsorted_dict*, *sorted_keys*)

Sort an OrderedDict by list of sorted keys.

**Parameters**

- **unsorted_dict** (*collections.OrderedDict*) – Source dictionary.
- **sorted_keys** (*list*) – Keys to sort on.

**Returns** Sorted dictionary.

**Return type** collections.OrderedDict

### 12.10.8 django_elasticsearch_dsl_drf.pagination module

Pagination.

**class** django_elasticsearch_dsl_drf.pagination.**LimitOffsetPagination**(**args*, ***kwargs*)

Bases: rest_framework.pagination.LimitOffsetPagination

A limit/offset pagination.

Example:
http://api.example.org/accounts/?limit=100 http://api.example.org/accounts/?offset=400&limit=100

**get_count**(*es_response*)

**get_facets**(*facets=None*)

Get facets.

**Parameters** **facets** –

**Returns**

**get_paginated_response**(*data*)

Get paginated response.

**Parameters** **data** –

**Returns**

**get_paginated_response_context**(*data*)

Get paginated response data.

**Parameters** **data** –

**Returns**

**paginate_queryset**(*queryset*, *request*, *view=None*)

**class** django_elasticsearch_dsl_drf.pagination.**Page**(*object_list*, *number*, *paginator*, *facets*)

Bases: django.core.paginator.Page

Page for Elasticsearch.

**class** django_elasticsearch_dsl_drf.pagination.**PageNumberPagination**(**args*, ***kwargs*)

Bases: rest_framework.pagination.PageNumberPagination

Page number pagination.

A simple page number based style that supports page numbers as query parameters.

Example:
http://api.example.org/accounts/?page=4 http://api.example.org/accounts/?page=4&page_size=100

**django_paginator_class**
alias of *Paginator*

**get_count**(*es_response*)

**get_facets**(*page=None*)
Get facets.
**Parameters** **page** –
**Returns**

**get_paginated_response**(*data*)
Get paginated response.
**Parameters** **data** –
**Returns**

**get_paginated_response_context**(*data*)
Get paginated response data.
**Parameters** **data** –
**Returns**

**paginate_queryset**(*queryset*, *request*, *view=None*)
Paginate a queryset.

Paginate a queryset if required, either returning a page object, or *None* if pagination is not configured for this view.
**Parameters**
- **queryset** –
- **request** –
- **view** –

**Returns**

**class** django_elasticsearch_dsl_drf.pagination.**Paginator**(*object_list*, *per_page*, *orphans=0*, *allow_empty_first_page=True*)
Bases: django.core.paginator.Paginator

Paginator for Elasticsearch.

**page**(*number*)
Returns a Page object for the given 1-based page number.
**Parameters** **number** –
**Returns**

### 12.10.9 django_elasticsearch_dsl_drf.serializers module

Serializers.

**class** django_elasticsearch_dsl_drf.serializers.**DocumentSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

A dynamic DocumentSerializer class.

**create**(*validated_data*)

Create.

Do nothing.

**Parameters** **validated_data** –

**Returns**

**get_fields**()

Get the required fields for serializing the result.

**update**(*instance*, *validated_data*)

Update.

Do nothing.

**Parameters**

- **instance** –
- **validated_data** –

**Returns**

**class** django_elasticsearch_dsl_drf.serializers.**DocumentSerializerMeta**

Bases: rest_framework.serializers.SerializerMetaclass

Metaclass for the DocumentSerializer.

Ensures that all declared subclasses implemented a Meta.

**class** django_elasticsearch_dsl_drf.serializers.**Meta**

Bases: type

Template for the DocumentSerializerMeta.Meta class.

**exclude = ()**

**field_aliases = {}**

**field_options = {}**

**fields = ()**

**ignore_fields = ()**

**index_aliases = {}**

**index_classes = ()**

**search_fields = ()**

**serializers = ()**

### 12.10.10 django_elasticsearch_dsl_drf.utils module

Utils.

**class** django_elasticsearch_dsl_drf.utils.**DictionaryProxy**(*mapping*)
Bases: object

Dictionary proxy.

**to_dict**()
To dict.
**Returns**

**class** django_elasticsearch_dsl_drf.utils.**EmptySearch**(**args*, ***kwargs*)
Bases: object

Empty Search.

**execute**(**args*, ***kwargs*)

**highlight**(**args*, ***kwargs*)

**sort**(**args*, ***kwargs*)

**to_dict**(**args*, ***kwargs*)

### 12.10.11 django_elasticsearch_dsl_drf.views module

### 12.10.12 django_elasticsearch_dsl_drf.viewsets module

Base ViewSets.

**class** django_elasticsearch_dsl_drf.viewsets.**BaseDocumentViewSet**(**args*, ***kwargs*)
Bases: rest_framework.viewsets.ReadOnlyModelViewSet

Base document ViewSet.

**document = None**

**document_uid_field = 'id'**

**get_object**()
Get object.

**get_queryset**()
Get queryset.

**pagination_class**
alias of *django_elasticsearch_dsl_drf.pagination.PageNumberPagination*

**class** django_elasticsearch_dsl_drf.viewsets.**DocumentViewSet**(**args*, ***kwargs*)
Bases: *django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet*, *django_elasticsearch_dsl_drf.viewsets.SuggestMixin*, *django_elasticsearch_dsl_drf.viewsets.FunctionalSuggestMixin*

DocumentViewSet with suggest and functional-suggest mix-ins.

**class** django_elasticsearch_dsl_drf.viewsets.**FunctionalSuggestMixin**
Bases: object

Functional suggest mixin.

**functional_suggest**(*request*)
Functional suggest functionality.
**Parameters** **request** –
**Returns**

**class** django_elasticsearch_dsl_drf.viewsets.**MoreLikeThisMixin**
Bases: object

More-like-this mixin.

**more_like_this**(*request*, *pk=None*, *id=None*)
More-like-this functionality detail view.
**Parameters** **request** –
**Returns**

**class** django_elasticsearch_dsl_drf.viewsets.**SuggestMixin**
Bases: object

Suggest mixin.

**suggest**(*request*)
Suggest functionality.

### 12.10.13 django_elasticsearch_dsl_drf.wrappers module

django_elasticsearch_dsl_drf.wrappers.**dict_to_obj**(*mapping*)
dict to obj mapping.
**Parameters** **mapping**(*dict*) –
**Returns**
**Return type** *Wrapper*

django_elasticsearch_dsl_drf.wrappers.**obj_to_dict**(*obj*)
Wrapper to dict.
**Parameters** **obj** (**'obj':Wrapper:**) –
**Returns**
**Return type** dict

**class** django_elasticsearch_dsl_drf.wrappers.**Wrapper**
Bases: object

Wrapper.

Example: >>> from django_elasticsearch_dsl_drf.wrappers import dict_to_obj >>> >>> mapping = { >>> 'country': { >>> 'name': 'Netherlands', >>> 'province': { >>> 'name': 'North Holland', >>> 'city': { >>> 'name': 'Amsterdam', >>> } >>> } >>> } >>> } >>> >>> wrapper = dict_to_obj(mapping) >>> wrapper.country.name >>> "Netherlands" >>> wrapper.country.province.name >>> "North Holland" >>> wrapper.country.province.city.name >>> "Amsterdam" >>> wrapper.as_dict >>> { >>> 'country': { >>> 'name': 'Netherlands', >>> 'province': { >>> 'name': 'North Holland', >>> 'city': { >>> 'name': 'Amsterdam', >>> } >>> } >>> } >>> } >>> str(wrapper) >>> "Netherlands"

**as_dict**
As dict.
**Returns**
**Return type** dict

**as_json**
As JSON.
**Returns**
**Return type** str

### 12.10.14 Module contents

Integrate Elasticsearch DSL with Django REST framework.

CHAPTER 13

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## d


django_elasticsearch_dsl_drf, 235
django_elasticsearch_dsl_drf.analyzers, 229
django_elasticsearch_dsl_drf.apps, 229
django_elasticsearch_dsl_drf.compat, 229
django_elasticsearch_dsl_drf.constants, 230
django_elasticsearch_dsl_drf.fields, 138
django_elasticsearch_dsl_drf.fields.common, 134
django_elasticsearch_dsl_drf.fields.helpers, 135
django_elasticsearch_dsl_drf.fields.nested_fields, 136
django_elasticsearch_dsl_drf.filter_backends, 202
django_elasticsearch_dsl_drf.filter_backends.aggregations, 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.bucket_aggregations, 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.metrics_aggregations, 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.pipeline_aggregations, 141
django_elasticsearch_dsl_drf.filter_backends.faceted_search, 193
django_elasticsearch_dsl_drf.filter_backends.filtering, 158
django_elasticsearch_dsl_drf.filter_backends.filtering.common, 141
django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial, 149
django_elasticsearch_dsl_drf.filter_backends.filtering.ids, 153
django_elasticsearch_dsl_drf.filter_backends.filtering.nested, 154
django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter, 156
django_elasticsearch_dsl_drf.filter_backends.highli 196
django_elasticsearch_dsl_drf.filter_backends.mixins 197
django_elasticsearch_dsl_drf.filter_backends.orderi 176
django_elasticsearch_dsl_drf.filter_backends.orderi 172
django_elasticsearch_dsl_drf.filter_backends.orderi 174
django_elasticsearch_dsl_drf.filter_backends.search 198
django_elasticsearch_dsl_drf.filter_backends.sugges 188
django_elasticsearch_dsl_drf.filter_backends.sugges 180
django_elasticsearch_dsl_drf.filter_backends.sugges 184
django_elasticsearch_dsl_drf.helpers, 230
django_elasticsearch_dsl_drf.pagination, 231
django_elasticsearch_dsl_drf.serializers, 232
django_elasticsearch_dsl_drf.tests, 216
django_elasticsearch_dsl_drf.tests.base, 202
django_elasticsearch_dsl_drf.tests.data_mixins, 203
django_elasticsearch_dsl_drf.tests.test_faceted_sea 203
django_elasticsearch_dsl_drf.tests.test_filtering_c 203
django_elasticsearch_dsl_drf.tests.test_filtering_g 206
django_elasticsearch_dsl_drf.tests.test_filtering_n 207
django_elasticsearch_dsl_drf.tests.test_filtering_p 209
django_elasticsearch_dsl_drf.tests.test_functional_

211
django_elasticsearch_dsl_drf.tests.test_helpers,
212
django_elasticsearch_dsl_drf.tests.test_highlight,
212
django_elasticsearch_dsl_drf.tests.test_ordering_common,
212
django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial,
213
django_elasticsearch_dsl_drf.tests.test_pagination,
213
django_elasticsearch_dsl_drf.tests.test_search,
214
django_elasticsearch_dsl_drf.tests.test_serializers,
214
django_elasticsearch_dsl_drf.tests.test_suggesters,
215
django_elasticsearch_dsl_drf.tests.test_views,
215
django_elasticsearch_dsl_drf.tests.test_wrappers,
216
django_elasticsearch_dsl_drf.utils, 233
django_elasticsearch_dsl_drf.viewsets,
234
django_elasticsearch_dsl_drf.wrappers,
235

# Index

## A


AddressesMixin (class in django_elasticsearch_dsl_drf.tests.data_mixins), 203
aggregate() (django_elasticsearch_dsl_drf.filter_backends.faceted_search.FacetedSearchFilterBackend method), 194
apply_filter() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.NestedFilteringFilterBackend class method), 155
apply_filter() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilteringFilterBackend class method), 169
apply_filter() (django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter.PostFilterFilteringFilterBackend class method), 157
apply_filter() (django_elasticsearch_dsl_drf.filter_backends.filtering.PostFilterFilteringFilterBackend class method), 171
apply_filter() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackendMixin class method), 197
apply_filter_prefix() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 142
apply_filter_prefix() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 158
apply_filter_range() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 143
apply_filter_range() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 159
apply_filter_term() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 143
apply_filter_term() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 159
apply_filter_terms() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 143
apply_filter_terms() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 159
apply_query() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.NestedFilteringFilterBackend class method), 155
apply_query() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilteringFilterBackend class method), 170
apply_query() (django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter.PostFilterFilteringFilterBackend class method), 157
apply_query() (django_elasticsearch_dsl_drf.filter_backends.filtering.PostFilterFilteringFilterBackend class method), 171
apply_query() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterB class method), 198
apply_query_contains() (django_elasticsearch_dsl_drf.filter_backends.filter class method), 144
apply_query_contains() (django_elasticsearch_dsl_drf.filter_backends.filter class method), 160
apply_query_endswith() (django_elasticsearch_dsl_drf.filter_backends.filte class method), 144
apply_query_endswith() (django_elasticsearch_dsl_drf.filter_backends.filte class method), 160
apply_query_exclude() (django_elasticsearch_dsl_drf.filter_backends.filteri class method), 144
apply_query_exclude() (django_elasticsearch_dsl_drf.filter_backends.filteri class method), 160
apply_query_exists() (django_elasticsearch_dsl_drf.filter_backends.filtering class method), 145
apply_query_exists() (django_elasticsearch_dsl_drf.filter_backends.filtering class method), 161
apply_query_geo_bounding_box() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia class method), 150
apply_query_geo_bounding_box() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia class method), 165
apply_query_geo_distance() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia class method), 150
apply_query_geo_distance() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia class method), 165
apply_query_geo_polygon() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia class method), 150
apply_query_geo_polygon() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia class method), 165
apply_query_gt() (django_elasticsearch_dsl_drf.filter_backends.filtering.co class method), 145
apply_query_gt() (django_elasticsearch_dsl_drf.filter_backends.filtering.Fil

class method), 161
apply_query_gte() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 145
apply_query_gte() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 161
apply_query_in() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 146
apply_query_in() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 161
apply_query_isnull() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 146
apply_query_isnull() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 162
apply_query_lt() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 147
apply_query_lt() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 162
apply_query_lte() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 147
apply_query_lte() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 162
apply_query_wildcard() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 147
apply_query_wildcard() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 163
apply_suggester_completion() (django_elasticsearch_dsl_drf.filter_backends.suggester.native.SuggesterFilterBackend class method), 187
apply_suggester_completion() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterBackend class method), 189
apply_suggester_completion_match() (django_elasticsearch_dsl_drf.filter_backends.suggester.functional.FunctionalSuggesterFilterBackend class method), 183
apply_suggester_completion_match() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend class method), 191
apply_suggester_completion_prefix() (django_elasticsearch_dsl_drf.filter_backends.suggester.functional.FunctionalSuggesterFilterBackend class method), 183
apply_suggester_completion_prefix() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend class method), 192
apply_suggester_phrase() (django_elasticsearch_dsl_drf.filter_backends.suggester.native.SuggesterFilterBackend class method), 187
apply_suggester_phrase() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterBackend class method), 189
apply_suggester_term() (django_elasticsearch_dsl_drf.filter_backends.suggester.native.SuggesterFilterBackend class method), 187
apply_suggester_term() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterBackend class method), 190
as_dict (django_elasticsearch_dsl_drf.wrappers.Wrapper attribute), 235
as_json (django_elasticsearch_dsl_drf.wrappers.Wrapper attribute), 235
authenticate() (django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase method), 202

## B

base_url (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested attribute), 207
base_url (django_elasticsearch_dsl_drf.tests.TestFilteringNested attribute), 220
BaseDocumentViewSet (class in django_elasticsearch_dsl_drf.viewsets), 234
BaseRestFrameworkTestCase (class in django_elasticsearch_dsl_drf.tests.base), 202
BaseSearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.search), 198
BaseTestCase (class in django_elasticsearch_dsl_drf.tests.base), 202
BooksMixin (class in django_elasticsearch_dsl_drf.tests.data_mixins), 203
BooleanField (class in django_elasticsearch_dsl_drf.fields), 138
BooleanField (class in django_elasticsearch_dsl_drf.fields.common), 134

## C

CharField (class in django_elasticsearch_dsl_drf.fields), 138
CharField (class in django_elasticsearch_dsl_drf.fields.common), 134
clean_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.functional.FunctionalSuggesterFilterBackend method), 183
clean_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend method), 192
CompoundSearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.search), 199
Config (class in django_elasticsearch_dsl_drf.apps), 229
construct_facets() (django_elasticsearch_dsl_drf.filter_backends.faceted_search.FacetedSearchFilterBackend method), 194
construct_nested_search() (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend method), 200
construct_search() (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend method), 201
create() (django_elasticsearch_dsl_drf.serializers.DocumentSerializer method), 233
create_books() (django_elasticsearch_dsl_drf.tests.data_mixins.BooksMixin class method), 203

created_addresses() (django_elasticsearch_dsl_drf.tests.data_mixins.AddressesMixin class method), 203

## D

DateField (class in django_elasticsearch_dsl_drf.fields), 139
DateField (class in django_elasticsearch_dsl_drf.fields.common), 135
DefaultOrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering), 176
DefaultOrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering_common), 172
dict_to_obj() (in module django_elasticsearch_dsl_drf.wrappers), 235
DictionaryProxy (class in django_elasticsearch_dsl_drf.utils), 233
django_elasticsearch_dsl_drf (module), 235
django_elasticsearch_dsl_drf.analyzers (module), 229
django_elasticsearch_dsl_drf.apps (module), 229
django_elasticsearch_dsl_drf.compat (module), 229
django_elasticsearch_dsl_drf.constants (module), 230
django_elasticsearch_dsl_drf.fields (module), 138
django_elasticsearch_dsl_drf.fields.common (module), 134
django_elasticsearch_dsl_drf.fields.helpers (module), 135
django_elasticsearch_dsl_drf.fields.nested_fields (module), 136
django_elasticsearch_dsl_drf.filter_backends (module), 202
django_elasticsearch_dsl_drf.filter_backends.aggregations (module), 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.bucket_aggregations (module), 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.metrics_aggregations (module), 141
django_elasticsearch_dsl_drf.filter_backends.aggregations.pipeline_aggregations (module), 141
django_elasticsearch_dsl_drf.filter_backends.faceted_search (module), 193
django_elasticsearch_dsl_drf.filter_backends.filtering (module), 158
django_elasticsearch_dsl_drf.filter_backends.filtering.common (module), 141
django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial (module), 149
django_elasticsearch_dsl_drf.filter_backends.filtering.ids (module), 153
django_elasticsearch_dsl_drf.filter_backends.filtering.nested (module), 154
django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter (module), 156
django_elasticsearch_dsl_drf.filter_backends.highlight (module), 196
django_elasticsearch_dsl_drf.filter_backends.mixins (module), 197
django_elasticsearch_dsl_drf.filter_backends.ordering (module), 176
django_elasticsearch_dsl_drf.filter_backends.ordering.common (module), 172
django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial (module), 174
django_elasticsearch_dsl_drf.filter_backends.search (module), 198
django_elasticsearch_dsl_drf.filter_backends.suggester (module), 188
django_elasticsearch_dsl_drf.filter_backends.suggester.functional (module), 180
django_elasticsearch_dsl_drf.filter_backends.suggester.native (module), 184
django_elasticsearch_dsl_drf.helpers (module), 230
django_elasticsearch_dsl_drf.pagination (module), 231
django_elasticsearch_dsl_drf.serializers (module), 232
django_elasticsearch_dsl_drf.tests (module), 216
django_elasticsearch_dsl_drf.tests.base (module), 202
django_elasticsearch_dsl_drf.tests.data_mixins (module), 203
django_elasticsearch_dsl_drf.tests.test_faceted_search (module), 203
django_elasticsearch_dsl_drf.tests.test_filtering_common (module), 203
django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial (module), 206
django_elasticsearch_dsl_drf.tests.test_filtering_nested (module), 207
django_elasticsearch_dsl_drf.tests.test_filtering_post_filter (module), 209
django_elasticsearch_dsl_drf.tests.test_functional_suggesters (module), 211
django_elasticsearch_dsl_drf.tests.test_helpers (module), 212
django_elasticsearch_dsl_drf.tests.test_highlight (module), 212
django_elasticsearch_dsl_drf.tests.test_ordering_common (module), 212
django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial (module), 213
django_elasticsearch_dsl_drf.tests.test_pagination (module), 213
django_elasticsearch_dsl_drf.tests.test_search (module), 214
django_elasticsearch_dsl_drf.tests.test_serializers (module), 214
django_elasticsearch_dsl_drf.tests.test_suggesters (mod-

ule), 215
django_elasticsearch_dsl_drf.tests.test_views (module), 215
django_elasticsearch_dsl_drf.tests.test_wrappers (module), 216
django_elasticsearch_dsl_drf.utils (module), 233
django_elasticsearch_dsl_drf.viewsets (module), 234
django_elasticsearch_dsl_drf.wrappers (module), 235
django_paginator_class (django_elasticsearch_dsl_drf.pagination.PageNumberPagination attribute), 232
document (django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet attribute), 234
document_uid_field (django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet attribute), 234
DocumentSerializer (class in django_elasticsearch_dsl_drf.serializers), 232
DocumentSerializerMeta (class in django_elasticsearch_dsl_drf.serializers), 233
DocumentViewSet (class in django_elasticsearch_dsl_drf.viewsets), 234

## E

EmptySearch (class in django_elasticsearch_dsl_drf.utils), 234
exclude (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
execute() (django_elasticsearch_dsl_drf.utils.EmptySearch method), 234
extract_field_name() (django_elasticsearch_dsl_drf.filter_backends.suggester.functional.FunctionalSuggesterFilterBackend method), 183
extract_field_name() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterBackend method), 192

## F

faceted_search_param (django_elasticsearch_dsl_drf.filter_backends.faceted_search.FacetedSearchFilterBackend attribute), 195
FacetedSearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.faceted_search), 193
field_aliases (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
field_options (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
fields (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.faceted_search.FacetedSearchFilterBackend method), 195
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend method), 148
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend method), 163
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_ method), 151
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.Geo method), 165
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.ids.I method), 154
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsF method), 168
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.highlight.Hig method), 196
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.com method), 172
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.com method), 174
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.Def method), 177
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.geo method), 175
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.Geo method), 178
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.ordering.Ord method), 180
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.search.BaseS method), 199
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.search.Searc method), 201
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.fu method), 183
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.Fu method), 192
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.na method), 187
filter_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggester.Su method), 190
FilterBackendMixin (class in django_elasticsearch_dsl_drf.filter_backends.mixins), 197
FilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering), 158
FilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering.common), 141
FloatField (class in django_elasticsearch_dsl_drf.fields), 139
FloatField (class in django_elasticsearch_dsl_drf.fields.common), 135
functional_suggest() (django_elasticsearch_dsl_drf.viewsets.FunctionalSug method), 234
FunctionalSuggesterFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.suggester), 190
FunctionalSuggesterFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.suggester.functional

181
FunctionalSuggestMixin (class in django_elasticsearch_dsl_drf.viewsets), 234

## G

GeoPointField (class in django_elasticsearch_dsl_drf.fields), 139
GeoPointField (class in django_elasticsearch_dsl_drf.fields.nested_fields), 136
GeoShapeField (class in django_elasticsearch_dsl_drf.fields), 139
GeoShapeField (class in django_elasticsearch_dsl_drf.fields.nested_fields), 136
GeoSpatialFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering), 164
GeoSpatialFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatial), 149
GeoSpatialOrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering), 177
GeoSpatialOrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial), 174
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend method), 148
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend method), 163
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.NestedFilteringFilterBackend method), 156
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilteringFilterBackend method), 170
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter.PostFilterFilteringFilterBackend method), 157
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.filtering.PostFilterFilteringFilterBackend method), 171
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend method), 199
get_coreschema_field() (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend method), 201
get_count() (django_elasticsearch_dsl_drf.pagination.LimitOffsetPagination method), 231
get_count() (django_elasticsearch_dsl_drf.pagination.PageNumberPagination method), 232
get_default_ordering_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.common.DefaultOrderingFilterBackend class method), 173
get_default_ordering_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.DefaultOrderingFilterBackend class method), 177
get_document_for_model() (in module django_elasticsearch_dsl_drf.helpers), 230
get_elasticsearch_version() (in module django_elasticsearch_dsl_drf.compat), 229
get_faceted_search_query_params() (django_elasticsearch_dsl_drf.filter_backends.faceted_search.Fac method), 195
get_facets() (django_elasticsearch_dsl_drf.pagination.LimitOffsetPaginatio method), 231
get_facets() (django_elasticsearch_dsl_drf.pagination.PageNumberPaginati method), 232
get_fields() (django_elasticsearch_dsl_drf.serializers.DocumentSerializer method), 233
get_filter_field_nested_path() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.Nes method), 156
get_filter_field_nested_path() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilt method), 170
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.F method), 148
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFi method), 163
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia method), 151
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia method), 166
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.Nes method), 156
get_filter_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilt method), 170
get_geo_bounding_box_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia class method), 151
get_geo_bounding_box_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia class method), 166
get_geo_distance_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.geo_spatia class method), 152
get_geo_distance_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatia class method), 167
get_geo_distance_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatia class method), 175
get_geo_distance_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatia

class method), 178
get_geo_polygon_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatialFilteringFilterBackend class method), 152
get_geo_polygon_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.GeoSpatialFilteringFilterBackend class method), 167
get_geo_spatial_field_name() (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatialOrderingFilterBackend method), 175
get_geo_spatial_field_name() (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatialOrderingFilterBackend method), 178
get_gte_lte_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.RangeFilteringFilterBackend class method), 148
get_gte_lte_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.RangeFilteringFilterBackend class method), 163
get_highlight_query_params() (django_elasticsearch_dsl_drf.filter_backends.highlighting.HighlightFilterBackend method), 197
get_ids_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend method), 154
get_ids_query_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend method), 168
get_ids_values() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend method), 154
get_ids_values() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend method), 168
get_index_and_mapping_for_model() (in module django_elasticsearch_dsl_drf.helpers), 230
get_object() (django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet method), 234
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.CommonOrderingFilterBackend method), 173
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatialOrderingFilterBackend method), 174
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.NestedOrderingFilterBackend method), 177
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.OrderingFilterBackend method), 176
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.SimpleFieldOrderingFilterBackend method), 179
get_ordering_query_params() (django_elasticsearch_dsl_drf.filter_backends.ordering.SimpleFieldOrderingFilterBackend method), 180
get_paginated_response() (django_elasticsearch_dsl_drf.pagination.LimitOffsetPagination method), 231
get_paginated_response() (django_elasticsearch_dsl_drf.pagination.PageNumberPagination method), 232
get_paginated_response() (django_elasticsearch_dsl_drf.pagination.LimitOffsetPagination method), 231
get_paginated_response() (django_elasticsearch_dsl_drf.pagination.PageNumberPagination method), 232
get_query() (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend method), 199
get_queryset() (django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet method), 234
get_range_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.RangeFilteringFilterBackend class method), 148
get_range_params() (django_elasticsearch_dsl_drf.filter_backends.filtering.RangeFilteringFilterBackend class method), 164
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.CompoundSearchFilterBackend method), 149
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.DefaultLookupFilterBackend method), 164
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.FacetedSearchFilterBackend method), 156
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend method), 170
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend method), 157
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.NestedFilteringFilterBackend method), 171
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.CommonOrderingFilterBackend method), 174
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatialOrderingFilterBackend method), 180
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend method), 199
get_schema_fields() (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend method), 201
get_search_query_params() (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend method), 199
get_search_query_params() (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend method), 201
get_suggester_query_params() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend method), 184
get_suggester_query_params() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend method), 192
get_suggester_query_params() (django_elasticsearch_dsl_drf.filter_backends.suggester.native.SuggesterFilterBackend method), 188
get_suggestion_query_params() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterBackend method), 190

get_value() (django_elasticsearch_dsl_drf.fields.BooleanField method), 138
get_value() (django_elasticsearch_dsl_drf.fields.CharField method), 138
get_value() (django_elasticsearch_dsl_drf.fields.common.BooleanField method), 134
get_value() (django_elasticsearch_dsl_drf.fields.common.CharField method), 134
get_value() (django_elasticsearch_dsl_drf.fields.common.DateField method), 135
get_value() (django_elasticsearch_dsl_drf.fields.common.FloatField method), 135
get_value() (django_elasticsearch_dsl_drf.fields.common.IntegerField method), 135
get_value() (django_elasticsearch_dsl_drf.fields.common.IPAddressField method), 135
get_value() (django_elasticsearch_dsl_drf.fields.DateField method), 139
get_value() (django_elasticsearch_dsl_drf.fields.FloatField method), 139
get_value() (django_elasticsearch_dsl_drf.fields.IntegerField method), 140
get_value() (django_elasticsearch_dsl_drf.fields.IPAddressField method), 140
get_value() (django_elasticsearch_dsl_drf.fields.ListField method), 140
get_value() (django_elasticsearch_dsl_drf.fields.nested_fields.ListField method), 138
get_value() (django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField method), 137
get_value() (django_elasticsearch_dsl_drf.fields.ObjectField method), 141

## H

highlight() (django_elasticsearch_dsl_drf.utils.EmptySearch method), 234
highlight_param (django_elasticsearch_dsl_drf.filter_backends.highlight.HighlightBackend attribute), 197
HighlightBackend (class in django_elasticsearch_dsl_drf.filter_backends.highlight), 196

## I

ids_query_param (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend attribute), 154
ids_query_param (django_elasticsearch_dsl_drf.filter_backends.filtering.IdsFilterBackend attribute), 169
IdsFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering), 168
IdsFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering.ids), 153
ignore_fields (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
index_aliases (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
index_classes (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
IntegerField (class in django_elasticsearch_dsl_drf.fields), 140
IntegerField (class in django_elasticsearch_dsl_drf.fields.common), 135
IPAddressField (class in django_elasticsearch_dsl_drf.fields), 140
IPAddressField (class in django_elasticsearch_dsl_drf.fields.common), 135

## K

KeywordField() (in module django_elasticsearch_dsl_drf.compat), 230

## L

label (django_elasticsearch_dsl_drf.apps.Config attribute), 229
LimitOffsetPagination (class in django_elasticsearch_dsl_drf.pagination), 231
ListField (class in django_elasticsearch_dsl_drf.fields), 140
ListField (class in django_elasticsearch_dsl_drf.fields.nested_fields), 138

## M

matching (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchF attribute), 199
matching (django_elasticsearch_dsl_drf.filter_backends.search.MultiMatchS attribute), 199
matching (django_elasticsearch_dsl_drf.filter_backends.search.SimpleQuery attribute), 202
Meta (class in django_elasticsearch_dsl_drf.serializers), 233
more_like_this() (django_elasticsearch_dsl_drf.viewsets.MoreLikeThisMix method), 235
more_like_this() (in module django_elasticsearch_dsl_drf.helpers), 230
MoreLikeThisMixin (class in django_elasticsearch_dsl_drf.viewsets), 234
MultiMatchSearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.search), 199

## N

name (django_elasticsearch_dsl_drf.apps.Config attribute), 229

NestedField (class in django_elasticsearch_dsl_drf.fields), 140
NestedField (class in django_elasticsearch_dsl_drf.fields.nested_fields), 136
NestedFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering), 169
NestedFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering.nested), 154

## O

obj_to_dict() (in module django_elasticsearch_dsl_drf.wrappers), 235
ObjectField (class in django_elasticsearch_dsl_drf.fields), 141
ObjectField (class in django_elasticsearch_dsl_drf.fields.nested_fields), 137
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.common.DjangoOrderingFilterBackend attribute), 173
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.common.OrderingFilterBackend attribute), 174
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.DefaultOrderingFilterBackend attribute), 177
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial.GeoSpatialOrderingFilterBackend attribute), 176
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.GeoSpatialOrderingFilterBackend attribute), 179
ordering_param (django_elasticsearch_dsl_drf.filter_backends.ordering.OrderingFilterBackend attribute), 180
OrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering), 179
OrderingFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.ordering.common), 173

## P

Page (class in django_elasticsearch_dsl_drf.pagination), 231
page() (django_elasticsearch_dsl_drf.pagination.Paginator method), 232
PageNumberPagination (class in django_elasticsearch_dsl_drf.pagination), 231
paginate_queryset() (django_elasticsearch_dsl_drf.pagination.LimitOffsetPagination method), 231
paginate_queryset() (django_elasticsearch_dsl_drf.pagination.PageNumberPagination method), 232
pagination_class (django_elasticsearch_dsl_drf.viewsets.BaseDocumentViewSet attribute), 234
Paginator (class in django_elasticsearch_dsl_drf.pagination), 232
PostFilterFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering), 170
PostFilterFilteringFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter), 156
prepare_faceted_search_fields() (django_elasticsearch_dsl_drf.filter_backends.faceted_search.FacetedSearchFilterBackend class method), 195
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.FilteringFilterBackend class method), 149
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.common.FilteringFilterBackend class method), 164
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.nested.NestedFilteringFilterBackend class method), 153
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.filtering.post_filter.PostFilterFilteringFilterBackend class method), 168
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.common.DjangoOrderingFilterBackend class method), 156
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.common.OrderingFilterBackend class method), 170
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.DefaultOrderingFilterBackend class method), 157
prepare_filter_fields() (django_elasticsearch_dsl_drf.filter_backends.ordering.geo_spatial.GeoSpatialOrderingFilterBackend class method), 171
prepare_highlight_fields() (django_elasticsearch_dsl_drf.filter_backends.highlight.HighlightFilterBackend class method), 187
prepare_suggester_fields() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterBackend class method), 184
prepare_suggester_fields() (django_elasticsearch_dsl_drf.filter_backends.suggester.FunctionalSuggesterFilterMixin class method), 193
prepare_suggester_fields() (django_elasticsearch_dsl_drf.filter_backends.suggester.native.SuggesterFilterBackend class method), 188
prepare_suggester_fields() (django_elasticsearch_dsl_drf.filter_backends.suggester.SuggesterFilterMixin class method), 190
pytestmark (django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase attribute), 202
pytestmark (django_elasticsearch_dsl_drf.tests.base.BaseTestCase attribute), 202
pytestmark (django_elasticsearch_dsl_drf.tests.test_faceted_search.TestFacetedSearch attribute), 203
pytestmark (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon attribute), 203
pytestmark (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestGeoSpatialFiltering attribute), 206
pytestmark (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested attribute), 207
pytestmark (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestPostFilterFiltering attribute), 209

pytestmark (django_elasticsearch_dsl_drf.tests.test_functional_suggesters.TestFunctionalSuggesters attribute), 211
pytestmark (django_elasticsearch_dsl_drf.tests.test_helpers.TestHelpers attribute), 212
pytestmark (django_elasticsearch_dsl_drf.tests.test_highlight.TestHighlight attribute), 212
pytestmark (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering attribute), 212
pytestmark (django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial.TestOrderingGeoSpatial attribute), 213
pytestmark (django_elasticsearch_dsl_drf.tests.test_pagination.TestPagination attribute), 213
pytestmark (django_elasticsearch_dsl_drf.tests.test_search.TestSearch attribute), 214
pytestmark (django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers attribute), 214
pytestmark (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters attribute), 215
pytestmark (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggestersEmptyIndex attribute), 215
pytestmark (django_elasticsearch_dsl_drf.tests.test_views.TestViews attribute), 215
pytestmark (django_elasticsearch_dsl_drf.tests.test_wrappers.TestWrappers attribute), 216
pytestmark (django_elasticsearch_dsl_drf.tests.TestFacetedSearch attribute), 216
pytestmark (django_elasticsearch_dsl_drf.tests.TestFilteringCommon attribute), 216
pytestmark (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial attribute), 219
pytestmark (django_elasticsearch_dsl_drf.tests.TestFilteringNested attribute), 220
pytestmark (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter attribute), 221
pytestmark (django_elasticsearch_dsl_drf.tests.TestFunctionalSuggesters attribute), 224
pytestmark (django_elasticsearch_dsl_drf.tests.TestHelpers attribute), 224
pytestmark (django_elasticsearch_dsl_drf.tests.TestHighlight attribute), 224
pytestmark (django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch attribute), 225
pytestmark (django_elasticsearch_dsl_drf.tests.TestOrdering attribute), 226
pytestmark (django_elasticsearch_dsl_drf.tests.TestOrderingGeoSpatial attribute), 227
pytestmark (django_elasticsearch_dsl_drf.tests.TestPagination attribute), 227
pytestmark (django_elasticsearch_dsl_drf.tests.TestSearch attribute), 227
pytestmark (django_elasticsearch_dsl_drf.tests.TestSerializers attribute), 228
pytestmark (django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch attribute), 225
pytestmark (django_elasticsearch_dsl_drf.tests.TestSuggesters attribute), 228
pytestmark (django_elasticsearch_dsl_drf.tests.TestViews attribute), 228
pytestmark (django_elasticsearch_dsl_drf.tests.TestWrappers attribute), 229

## Q

query_backends (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend attribute), 199
query_backends (django_elasticsearch_dsl_drf.filter_backends.search.CompoundSearchFilterBackend attribute), 199
query_backends (django_elasticsearch_dsl_drf.filter_backends.search.MultiMatchSearchFilterBackend attribute), 199
query_backends (django_elasticsearch_dsl_drf.filter_backends.search.SimpleQueryStringSearchFilterBackend attribute), 202

## S

search_fields (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
search_param (django_elasticsearch_dsl_drf.filter_backends.search.BaseSearchFilterBackend attribute), 199
search_param (django_elasticsearch_dsl_drf.filter_backends.search.MultiMatchSearchFilterBackend attribute), 199
search_param (django_elasticsearch_dsl_drf.filter_backends.search.SearchFilterBackend attribute), 202
search_param (django_elasticsearch_dsl_drf.filter_backends.search.SimpleQueryStringSearchFilterBackend attribute), 202
SearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.search), 199
serialize_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggesters.base.BaseSuggesterFilterBackend method), 184
serialize_queryset() (django_elasticsearch_dsl_drf.filter_backends.suggesters.completion.CompletionSuggesterFilterBackend method), 193
serializers (django_elasticsearch_dsl_drf.serializers.Meta attribute), 233
setUp() (django_elasticsearch_dsl_drf.tests.test_faceted_search.TestFacetedSearch class method), 203
setUp() (django_elasticsearch_dsl_drf.tests.test_highlight.TestHighlight class method), 212
setUp() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch class method), 214
setUp() (django_elasticsearch_dsl_drf.tests.TestFacetedSearch class method), 216
setUp() (django_elasticsearch_dsl_drf.tests.TestHighlight class method), 224
setUp() (django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch class method), 225
setUp() (django_elasticsearch_dsl_drf.tests.TestSearch class method), 227
setUp() (django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch class method), 225

setUpClass() (django_elasticsearch_dsl_drf.tests.base.BaseRestFrameworkTestCase class method), 202
setUpClass() (django_elasticsearch_dsl_drf.tests.base.BaseTestCase class method), 202
setUpClass() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon class method), 203
setUpClass() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial class method), 206
setUpClass() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested class method), 207
setUpClass() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter class method), 209
setUpClass() (django_elasticsearch_dsl_drf.tests.test_functional_suggesters.TestFunctionalSuggesters class method), 211
setUpClass() (django_elasticsearch_dsl_drf.tests.test_helpers.TestHelpers class method), 212
setUpClass() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering class method), 212
setUpClass() (django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial.TestOrderingGeoSpatial class method), 213
setUpClass() (django_elasticsearch_dsl_drf.tests.test_pagination.TestPagination class method), 213
setUpClass() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters class method), 215
setUpClass() (django_elasticsearch_dsl_drf.tests.test_suggesters_mixin.TestSuggesterMixin class method), 215
setUpClass() (django_elasticsearch_dsl_drf.tests.test_views.TestViews class method), 216
setUpClass() (django_elasticsearch_dsl_drf.tests.test_wrappers.TestWrappers class method), 216
setUpClass() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon class method), 216
setUpClass() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial class method), 219
setUpClass() (django_elasticsearch_dsl_drf.tests.TestFilteringNested class method), 220
setUpClass() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter class method), 222
setUpClass() (django_elasticsearch_dsl_drf.tests.TestFunctionalSuggesters class method), 224
setUpClass() (django_elasticsearch_dsl_drf.tests.TestHelpers class method), 224
setUpClass() (django_elasticsearch_dsl_drf.tests.TestOrdering class method), 226
setUpClass() (django_elasticsearch_dsl_drf.tests.TestOrderingGeoSpatial class method), 227
setUpClass() (django_elasticsearch_dsl_drf.tests.TestPagination class method), 227
setUpClass() (django_elasticsearch_dsl_drf.tests.TestSuggesters class method), 228
setUpClass() (django_elasticsearch_dsl_drf.tests.TestViews class method), 228
setUpClass() (django_elasticsearch_dsl_drf.tests.TestWrappers class method), 229
SimpleQueryStringSearchFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.search), 202
sort() (django_elasticsearch_dsl_drf.utils.EmptySearch method), 234
sort_by_list() (in module django_elasticsearch_dsl_drf.helpers), 231
split_lookup_complex_multiple_value() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackend class method), 198
split_lookup_complex_value() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackend class method), 198
split_lookup_filter() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackend class method), 198
split_lookup_name() (django_elasticsearch_dsl_drf.filter_backends.mixins.FilterBackend class method), 198
StringField() (in module django_elasticsearch_dsl_drf.compat), 230
suggest() (django_elasticsearch_dsl_drf.viewsets.SuggestMixin method), 235
SuggesterFilterBackend (class in django_elasticsearch_dsl_drf.filter_backends.suggester), 188
SuggesterFilterBackendEmptyIndex (class in django_elasticsearch_dsl_drf.filter_backends.suggester.native), 185
SuggestMixin (class in django_elasticsearch_dsl_drf.viewsets), 235

## T

test_author_default_order_by() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 212
test_author_default_order_by() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_author_order_by_field_id_ascending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 212
test_author_order_by_field_id_ascending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_author_order_by_field_id_descending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 212
test_author_order_by_field_id_descending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_author_order_by_field_name_ascending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 212
test_author_order_by_field_name_ascending() (django_elasticsearch_dsl_drf.tests.TestOrdering

method), 226
test_author_order_by_field_name_descending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 212
test_author_order_by_field_name_descending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_default_order_by() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_default_order_by() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_order_by_field_id_ascending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_order_by_field_id_ascending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_order_by_field_id_descending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_order_by_field_id_descending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_order_by_field_title_ascending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_order_by_field_title_ascending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_order_by_field_title_descending() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_order_by_field_title_descending() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_book_order_by_non_existent_field() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_book_order_by_non_existent_field() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 226
test_compound_search_boost_by_field() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch method), 214
test_compound_search_boost_by_field() (django_elasticsearch_dsl_drf.tests.TestSearch method), 227
test_compound_search_by_field() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch method), 214
test_compound_search_by_field() (django_elasticsearch_dsl_drf.tests.TestSearch method), 227
test_compound_search_by_field_multi_terms() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch method), 214
test_compound_search_by_field_multi_terms() (django_elasticsearch_dsl_drf.tests.TestSearch method), 227
test_compound_search_by_nested_field() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch method), 214
test_compound_search_by_nested_field() (django_elasticsearch_dsl_drf.tests.TestSearch method), 227
test_default_filter_lookup() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFil method), 203
test_default_filter_lookup() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_detail_view() (django_elasticsearch_dsl_drf.tests.test_views.TestViews method), 216
test_detail_view() (django_elasticsearch_dsl_drf.tests.TestViews method), 229
test_dict_to_obj() (django_elasticsearch_dsl_drf.tests.test_wrappers.TestWr method), 216
test_dict_to_obj() (django_elasticsearch_dsl_drf.tests.TestWrappers method), 229
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFil method), 203
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilte method), 207
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestF method), 209
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 220
test_field_filter_contains() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFil method), 204
test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilte method), 207
test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestF method), 209

test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 220
test_field_filter_endswith() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 207
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 209
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 220
test_field_filter_exclude() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 207
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 209
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 220
test_field_filter_exists_false() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 209
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 220
test_field_filter_exists_true() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_geo_bounding_box() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 206
test_field_filter_geo_bounding_box() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 219
test_field_filter_geo_bounding_box_fail_test() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 206
test_field_filter_geo_bounding_box_fail_test() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 219
test_field_filter_geo_distance() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 206
test_field_filter_geo_distance() (django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial.TestOrderingGeoSpatial method), 213
test_field_filter_geo_distance() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 219
test_field_filter_geo_distance() (django_elasticsearch_dsl_drf.tests.TestOrderingGeoSpatial method), 227
test_field_filter_geo_distance_distance_type_arc() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 206
test_field_filter_geo_distance_distance_type_arc() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 219
test_field_filter_geo_distance_not_enough_args_fail() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 207
test_field_filter_geo_distance_not_enough_args_fail() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 220
test_field_filter_geo_polygon() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 207
test_field_filter_geo_polygon() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 220
test_field_filter_geo_polygon_fail_test() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial.TestFilteringGeoSpatial method), 207

test_field_filter_geo_polygon_fail_test() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 220
test_field_filter_geo_polygon_string_options() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial method), 207
test_field_filter_geo_polygon_string_options() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 220
test_field_filter_geo_polygon_string_options_fail_test() (django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial method), 207
test_field_filter_geo_polygon_string_options_fail_test() (django_elasticsearch_dsl_drf.tests.TestFilteringGeoSpatial method), 220
test_field_filter_gt() (django_elasticsearch_dsl_drf.tests.test_filtering_common method), 204
test_field_filter_gt() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter method), 209
test_field_filter_gt() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_gt() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_gt_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_gt_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 209
test_field_filter_gt_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_gt_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_gte() (django_elasticsearch_dsl_drf.tests.test_filtering_common method), 204
test_field_filter_gte() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter method), 209
test_field_filter_gte() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_gte() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.test_filtering_common method), 204
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter method), 208
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter method), 210
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 217
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_in() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_isnull_false() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_isnull_false() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_isnull_false() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_isnull_false() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 222
test_field_filter_isnull_true() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 204
test_field_filter_isnull_true() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_isnull_true() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_isnull_true() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_lt() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_lt() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_lt() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_lt() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_lt_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_lt_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_lt_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_lt_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_lte() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_lte() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_lte() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_lte() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.test_filtering_co

method), 205
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringNested method), 208
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_prefix() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_range() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_range_with_boost() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 210
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_term() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_term_explicit() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_terms_list() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 218
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_terms_string() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 223
test_field_filter_wildcard()

(django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 205
test_field_filter_wildcard() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_field_filter_wildcard() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_field_filter_wildcard() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 219
test_field_filter_wildcard() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_field_filter_wildcard() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 224
test_filter_by_field() (django_elasticsearch_dsl_drf.tests.test_helpers.TestHelpers method), 212
test_filter_by_field() (django_elasticsearch_dsl_drf.tests.TestHelpers method), 224
test_ids_filter() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 206
test_ids_filter() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_ids_filter() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 219
test_ids_filter() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 224
test_list_results_with_facets() (django_elasticsearch_dsl_drf.tests.test_faceted_search.TestFacetedSearch method), 203
test_list_results_with_facets() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_list_results_with_facets() (django_elasticsearch_dsl_drf.tests.TestFacetedSearch method), 216
test_list_results_with_facets() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 224
test_list_results_with_highlights() (django_elasticsearch_dsl_drf.tests.test_highlight.TestHighlight method), 212
test_list_results_with_highlights() (django_elasticsearch_dsl_drf.tests.TestHighlight method), 225
test_listing_view() (django_elasticsearch_dsl_drf.tests.test_views.TestViews method), 216
test_listing_view() (django_elasticsearch_dsl_drf.tests.TestViews method), 229
test_nested_field_filter_term() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 206
test_nested_field_filter_term() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 219
test_nested_fields_suggesters_completion() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_nested_fields_suggesters_completion() (django_elasticsearch_dsl_drf.tests.TestSuggesters method), 228
test_obj_to_dict() (django_elasticsearch_dsl_drf.tests.test_wrappers.TestWrappers method), 216
test_obj_to_dict() (django_elasticsearch_dsl_drf.tests.TestWrappers method), 229
test_pagination() (django_elasticsearch_dsl_drf.tests.test_pagination.TestPagination method), 213
test_pagination() (django_elasticsearch_dsl_drf.tests.TestPagination method), 227
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 206
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFilteringNested method), 208
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering method), 213
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch method), 214
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 219
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestFilteringNested method), 221
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 224
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch method), 225
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestOrdering method), 227
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestSearch method), 227
test_schema_field_not_required() (django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch method), 228
test_schema_fields_with_filter_fields_list()

(django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFiltering
method), 206
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.test_filtering_nested.TestFiltering
method), 208
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFiltering
method), 211
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.test_ordering_common.TestOrdering
method), 213
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestFilteringCommon
method), 219
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestFilteringNested
method), 221
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter
method), 224
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch
method), 225
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestOrdering
method), 227
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestSearch
method), 227
test_schema_fields_with_filter_fields_list()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 225
test_search() (django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch
method), 225
test_search_boost() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_search_boost() (django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch
method), 225
test_search_boost() (django_elasticsearch_dsl_drf.tests.TestSearch
method), 227
test_search_boost_compound()
(django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_search_boost_compound()
(django_elasticsearch_dsl_drf.tests.TestSearch
method), 227
test_search_boost_selected_fields()
(django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch
method), 225
test_search_boost_selected_fields()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 225
test_search_by_field() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFiltering
method), 206
test_search_by_field() (django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_search_by_field() (django_elasticsearch_dsl_drf.tests.TestSearch
method), 228
test_search_by_field_multi_terms()
(django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFiltering
method), 206
test_search_by_field_multi_terms()
(django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_search_by_field_multi_terms()
(django_elasticsearch_dsl_drf.tests.TestSearch
method), 228
test_search_by_nested_field()
(django_elasticsearch_dsl_drf.tests.test_search.TestSearch
method), 214
test_search_by_nested_field()
(django_elasticsearch_dsl_drf.tests.TestSearch
method), 228
test_search_selected_fields()
(django_elasticsearch_dsl_drf.tests.TestMultiMatchSearch
method), 225
test_search_selected_fields()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 225
test_search_with_quotes()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 225
test_search_with_quotes_alternative()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 225
test_search_with_quotes_boost()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 226
test_search_with_quotes_boost_alternative()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 226
test_search_without_quotes()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 226
test_search_without_quotes_boost()
(django_elasticsearch_dsl_drf.tests.TestSimpleQueryStringSearch
method), 226
test_serializer_document_equals_to_none()
(django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers
method), 214
test_serializer_document_equals_to_none()
(django_elasticsearch_dsl_drf.tests.TestSerializers
method), 228
test_serializer_fields_and_exclude()
(django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers
method), 214
test_serializer_fields_and_exclude()
(django_elasticsearch_dsl_drf.tests.TestSerializers
method), 228
test_simple_search_meta_del_attr()

(django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers method), 214
test_serializer_meta_del_attr() (django_elasticsearch_dsl_drf.tests.TestSerializers method), 228
test_serializer_meta_set_attr() (django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers method), 215
test_serializer_meta_set_attr() (django_elasticsearch_dsl_drf.tests.TestSerializers method), 228
test_serializer_no_document_specified() (django_elasticsearch_dsl_drf.tests.test_serializers.TestSerializers method), 215
test_serializer_no_document_specified() (django_elasticsearch_dsl_drf.tests.TestSerializers method), 228
test_suggesters_completion() (django_elasticsearch_dsl_drf.tests.test_functional_suggesters.TestFunctionalSuggesters method), 211
test_suggesters_completion() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_suggesters_completion() (django_elasticsearch_dsl_drf.tests.TestFunctionalSuggesters method), 224
test_suggesters_completion() (django_elasticsearch_dsl_drf.tests.TestSuggesters method), 228
test_suggesters_completion_no_args_provided() (django_elasticsearch_dsl_drf.tests.test_functional_suggesters.TestFunctionalSuggesters method), 211
test_suggesters_completion_no_args_provided() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_suggesters_completion_no_args_provided() (django_elasticsearch_dsl_drf.tests.TestFunctionalSuggesters method), 224
test_suggesters_completion_no_args_provided() (django_elasticsearch_dsl_drf.tests.TestSuggesters method), 228
test_suggesters_on_empty_index() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_suggesters_phrase() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_suggesters_phrase() (django_elasticsearch_dsl_drf.tests.TestSuggesters method), 228
test_suggesters_term() (django_elasticsearch_dsl_drf.tests.test_suggesters.TestSuggesters method), 215
test_suggesters_term() (django_elasticsearch_dsl_drf.tests.TestSuggesters method), 228
test_various_complex_fields() (django_elasticsearch_dsl_drf.tests.test_filtering_common.TestFilteringCommon method), 206
test_various_complex_fields() (django_elasticsearch_dsl_drf.tests.test_filtering_post_filter.TestFilteringPostFilter method), 211
test_various_complex_fields() (django_elasticsearch_dsl_drf.tests.TestFilteringCommon method), 219
test_various_complex_fields() (django_elasticsearch_dsl_drf.tests.TestFilteringPostFilter method), 224
test_wrapper_as_json() (django_elasticsearch_dsl_drf.tests.test_wrappers.TestWrappers method), 216
test_wrapper_as_json() (django_elasticsearch_dsl_drf.tests.TestWrappers method), 229
TestFacetedSearch (class in django_elasticsearch_dsl_drf.tests), 216
TestFacetedSearch (class in django_elasticsearch_dsl_drf.tests.test_faceted_search), 203
TestFunctionalSuggesters (class in django_elasticsearch_dsl_drf.tests), 216
TestFilteringCommon (class in django_elasticsearch_dsl_drf.tests.test_filtering_common), 203
TestFilteringGeoSpatial (class in django_elasticsearch_dsl_drf.tests), 219
TestFilteringGeoSpatial (class in django_elasticsearch_dsl_drf.tests.test_filtering_geo_spatial), 206
TestFilteringNested (class in django_elasticsearch_dsl_drf.tests), 220
TestFilteringNested (class in django_elasticsearch_dsl_drf.tests.test_filtering_nested), 207
TestFilteringPostFilter (class in django_elasticsearch_dsl_drf.tests), 221
TestFilteringPostFilter (class in django_elasticsearch_dsl_drf.tests.test_filtering_post_filter), 209
TestFunctionalSuggesters (class in django_elasticsearch_dsl_drf.tests), 224
TestFunctionalSuggesters (class in django_elasticsearch_dsl_drf.tests.test_functional_suggesters), 211
TestHelpers (class in django_elasticsearch_dsl_drf.tests), 224
TestHelpers (class in django_elasticsearch_dsl_drf.tests.test_helpers), 212
TestHighlight.TestSuggesters (class in django_elasticsearch_dsl_drf.tests), 224
TestHighlight (class in django_elasticsearch_dsl_drf.tests.test_highlight), 212
TestMultiTermFilteringCommon (class in

django_elasticsearch_dsl_drf.tests), 225
TestOrdering (class in django_elasticsearch_dsl_drf.tests), 226
TestOrdering (class in django_elasticsearch_dsl_drf.tests.test_ordering_common), 212
TestOrderingGeoSpatial (class in django_elasticsearch_dsl_drf.tests), 227
TestOrderingGeoSpatial (class in django_elasticsearch_dsl_drf.tests.test_ordering_geo_spatial), 213
TestPagination (class in django_elasticsearch_dsl_drf.tests), 227
TestPagination (class in django_elasticsearch_dsl_drf.tests.test_pagination), 213
TestSearch (class in django_elasticsearch_dsl_drf.tests), 227
TestSearch (class in django_elasticsearch_dsl_drf.tests.test_search), 214
TestSerializers (class in django_elasticsearch_dsl_drf.tests), 228
TestSerializers (class in django_elasticsearch_dsl_drf.tests.test_serializers), 214
TestSimpleQueryStringSearch (class in django_elasticsearch_dsl_drf.tests), 225
TestSuggesters (class in django_elasticsearch_dsl_drf.tests), 228
TestSuggesters (class in django_elasticsearch_dsl_drf.tests.test_suggesters), 215
TestSuggestersEmptyIndex (class in django_elasticsearch_dsl_drf.tests.test_suggesters), 215
TestViews (class in django_elasticsearch_dsl_drf.tests), 228
TestViews (class in django_elasticsearch_dsl_drf.tests.test_views), 215
TestWrappers (class in django_elasticsearch_dsl_drf.tests), 229
TestWrappers (class in django_elasticsearch_dsl_drf.tests.test_wrappers), 216
to_dict() (django_elasticsearch_dsl_drf.utils.DictionaryProxy method), 234
to_dict() (django_elasticsearch_dsl_drf.utils.EmptySearch method), 234
to_internal_value() (django_elasticsearch_dsl_drf.fields.ListField method), 140
to_internal_value() (django_elasticsearch_dsl_drf.fields.nested_fields.ListField method), 138
to_internal_value() (django_elasticsearch_dsl_drf.fields.nested_fields.ObjectField method), 137
to_internal_value() (django_elasticsearch_dsl_drf.fields.ObjectField method), 141
to_representation() (django_elasticsearch_dsl_drf.fields.BooleanField method), 138
to_representation() (django_elasticsearch_dsl_drf.fields.CharField method), 138
to_representation() (django_elasticsearch_dsl_drf.fields.common.BooleanFi method), 134
to_representation() (django_elasticsearch_dsl_drf.fields.common.CharField method), 135
to_representation() (django_elasticsearch_dsl_drf.fields.common.DateField method), 135
to_representation() (django_elasticsearch_dsl_drf.fields.common.FloatField method), 135
to_representation() (django_elasticsearch_dsl_drf.fields.common.IntegerFie method), 135
to_representation() (django_elasticsearch_dsl_drf.fields.common.IPAddress method), 135
to_representation() (django_elasticsearch_dsl_drf.fields.DateField method), 139
to_representation() (django_elasticsearch_dsl_drf.fields.FloatField method), 139
to_representation() (django_elasticsearch_dsl_drf.fields.IntegerField method), 140
to_representation() (django_elasticsearch_dsl_drf.fields.IPAddressField method), 140
to_representation() (django_elasticsearch_dsl_drf.fields.ListField method), 140
to_representation() (django_elasticsearch_dsl_drf.fields.nested_fields.ListFi method), 138
to_representation() (django_elasticsearch_dsl_drf.fields.nested_fields.Objec method), 137
to_representation() (django_elasticsearch_dsl_drf.fields.ObjectField method), 141
to_representation() (in module django_elasticsearch_dsl_drf.fields.helpers), 135

## U

update() (django_elasticsearch_dsl_drf.serializers.DocumentSerializer method), 233

## W

Wrapper (class in django_elasticsearch_dsl_drf.wrappers), 235