Elasticsearch for Django REST Framework Documentation

Release 0.2.0

Yaroslav Muravskyi

Dec 09, 2019

Contents:

CHAPTER 1

Basic Usage

Let's take a look at a quick example of using Django REST Elasticsearch to build a simple application. In the example, we'll build a simple blogging system

1.1 Create model

First of all, we create a model.

```
class Blog(models.Model):
    title = models.CharField(_('Title'), max_length=1000)
    created_at = models.DateTimeField(_('Created at'), auto_now_add=True)
    body = models.TextField(_('Body'))
    tags = ArrayField(models.CharField(max_length=200), blank=True, null=True)
    is_published = models.BooleanField(_('Is published'), default=False)

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

1.2 Create the mappings in Elasticsearch

Then, we have to create a model-like wrapper around our Django model.

```
class BlogIndex(DocType):

    pk = Integer()
    title = Text(fields={'raw': Keyword()})
    created_at = Date()
    body = Text()
    tags = Keyword(multi=True)
    is_published = Boolean()
```

(continues on next page)

(continued from previous page)

```
    class Meta:
        index = 'blog'
```

After we need to create the mappings in Elasticsearch. For that you can create the mappings directly by calling the init class method:

```
BlogIndex.init()
```

In the Document lifecycle documentation, you can find the full explanation how to work with the document manually.

1.3 Make the index updatable when new data is added, updated or deleted

We want to have a consistent data in the ElasticSearch, that is why we need to create, update or delete a document when we change anything in the model. The best way to do it add a Django signal dispatcher. Before adding signals, let's create a serializer to create, update and delete an elasticsearch document.

```
from rest_framework_elasticsearch.es_serializer import ElasticModelSerializer
from .models import Blog
from .search_indexes import BlogIndex

class ElasticBlogSerializer(ElasticModelSerializer):
    class Meta:
        model = Blog
        es_model = BlogIndex
        fields = ('pk', 'title', 'created_at', 'tags', 'body', 'is_published')
```

After we need to create a *signals.py* file and add this code:

```
from django.db.models.signals import pre_save, post_delete
from django.dispatch import receiver
from .serializers import Blog, ElasticBlogSerializer

@receiver(pre_save, sender=Blog, dispatch_uid="update_record")
def update_es_record(sender, instance, **kwargs):
    obj = ElasticBlogSerializer(instance)
    obj.save()

@receiver(post_delete, sender=Blog, dispatch_uid="delete_record")
def delete_es_record(sender, instance, *args, **kwargs):
    obj = ElasticBlogSerializer(instance)
    obj.delete(ignore=404)
```

1.4 Simple django REST framework search view

Finally, let's make a simple search view to find all posts filtered by a tag and search by a word in a title:

```
from elasticsearch import Elasticsearch, RequestsHttpConnection
from rest_framework_elasticsearch import es_views, es_pagination, es_filters
from .search_indexes import BlogIndex
```

(continues on next page)

(continued from previous page)

```
class BlogView(es_views.ListElasticAPIView):
    es_client = Elasticsearch(hosts=['elasticsearch:9200/'],
                              connection_class=RequestsHttpConnection)
    es_model = BlogIndex
    es_filter_backends = (
        es_filters.ElasticFieldsFilter,
        es_filters.ElasticSearchFilter
    )
    es_filter_fields = (
        es_filters.ESFieldFilter('tag', 'tags'),
    )
    es_search_fields = (
        'tags',
        'title',
    )
```

That's all, we can start using it.

```
http://example.com/blogs/api/list?search=elasticsearch
http://example.com/blogs/api/list?tag=opensource
http://example.com/blogs/api/list?tag=opensource,aws
```

CHAPTER 2

Pagination

Example of the pagination response

```
class BlogView(es_views.ListElasticAPIView):
    es_client = Elasticsearch(hosts=['elasticsearch:9200/'],
                              connection_class=RequestsHttpConnection)

    es_pagination_class = es_pagination.ElasticLimitOffsetPagination

    es_model = BlogIndex
    es_filter_backends = (
        es_filters.ElasticFieldsFilter,
        es_filters.ElasticSearchFilter
    )
    es_filter_fields = (
        es_filters.ESFieldFilter('tag', 'tags'),
    )
    es_search_fields = (
        'tags',
        'title',
    )
```

CHAPTER 3

Ordering

Example of ordering

```
class BlogView(es_views.ListElasticAPIView):
    es_client = Elasticsearch(hosts=['elasticsearch:9200/'],
                              connection_class=RequestsHttpConnection)

    es_pagination_class = es_pagination.ElasticLimitOffsetPagination

    es_filter_backends = (
        es_filters.ElasticFieldsFilter,
        es_filters.ElasticSearchFilter,
        es_filters.ElasticOrderingFilter,
    )
    es_ordering_fields = (
        "created_at",
        ("title.raw", "title")
    )
    es_filter_fields = (
        es_filters.ESFieldFilter('tag', 'tags'),
    )
    es_search_fields = (
        'tags',
        'title',
    )
```

CHAPTER 4

About

Django REST Elasticsearch provides the easy way for integration Django REST Framework and Elasticsearch. The library uses Elasticsearch DSL library, it is a high-level library to the official low-level client.

CHAPTER 5

Requirements

- Django REST Framework 3.5 and above
- elasticsearch-dsl>=5.0.0,<7.0.0 (*Elasticsearch 5.x*)

CHAPTER 6

Installation

```
pip install django-rest-elasticsearch
```

CHAPTER 7

Changelog

7.1 v0.4

Release date: 2018-03-03

- Add tests

7.2 v0.3.4

Release date: 2018-01-20

- Fixed small bugs

7.3 v0.3.3

Release date: 2017-11-30

- Added a support of the API schema
- Remove None values from gte/lte range filter

7.4 v0.3.2

Release date: 2017-09-06

- Fixed the bug with the paginator class
- Remove Django<1.11 from the requirement

7.5 v0.3.0

Release date: 2017-06-26

- Fixed the bug with filtering by boolean values
- Added validation

7.6 v0.2.0

Release date: 2017-04-07

- Initial release.