Django Package Documentation

Release 0.1.5

Alexander Beach

Mar 04, 2018

Contents

Contents:

CHAPTER 1

DRF Elasticsearch DSL

DRF Elasticsearch DSL is losely based on django-haystack and provides a `ModelSerializerDocument` which supports all of the field types provided by elastic-search-dsl persistence.. `ModelSerializerDocument` is losely based on the `DocType` class provided by `elasticsearch-dsl.py`

The purpose of this libraray is to allow definition of elasticsearch documents with DRF's ModelSerializer class while optionally providing support for async document updates and deletes with celery.

1.1 Documentation

The full documentation is at https://drf-elasticsearch-dsl.readthedocs.io.

1.2 Quickstart

Install Django Package:

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

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'drf_elasticsearch_dsl.apps.DrfElasticsearchDsl',
    ...
)
```

Configure DRF_SERIALIZER_ELASTICSERACH_SETTTINGS in your settings.py file with your elasticsearch url(s)

```
DRF_SERIALIZER_ELASTICSERACH_SETTTINGS = {
    'elasticsearch_hosts': ['localhost']
}
```

Create a Model

```
from django.db import models


class Contact(models.Model):

    first_name = models.CharField(max_length=32, null=False, blank=False)
    last_name = models.CharField(max_length=32, null=False, blank=False)
    url = models.URLField(null=False, blank=False)
    email = models.EmailField(max_length=254, null=False, blank=False)
    bio = models.TextField(null=False, blank=False)
    birthday = models.DateField(null=False, blank=False)
```

Create a ModelSerializer

```
from rest_framework import serializers

class ContactSerializer(serializers.ModelSerializer):

    class Meta:
        model = Contact
        fields = '__all__'
```

Create a `search_indexes.py`, which should be in the root of the application. Add your `ModelSerializerDocument` classes here. The specificed index will have its mapping updated for this document.

```
from drf_elasticsearch_dsl.documents import ModelSerializerDocument
from elasticsearch_dsl import Date, Keyword, Text, String
from .serialziers import ContactSerializer


class ContactSerializerDocument(ModelSerializerDocument):
    first_name = String()
    last_name = String()
    url = Keyword()
    email = Keyword()
    bio = Text()
    birthday = Date()

    class Meta:
        index = 'myapp'
        serializer = ContactSerializer
        doc_type = 'myapp.contact'
```

Finally, sync your database with elasticsearch by running:

```
$ python manage.py update_index
```

1.3 Features

1.3.1 Celery Support

By default, dr-elasticsearch-dsl does not setup signals to sync models on save or delete. To enable celery support, add the following to your settings.py configuration:

```
DRF_SERIALIZER_ELASTICSERACH_SETTTINGS = {
    ...
    'signal_processor_class': 'drf_elasticsearch_dsl.signals.CelerySignalProcessor',
}
```

See the celery documentation for details setting up celery with django

1.4 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install -r requirements_test.txt
(myenv) $ tox
```

1.5 TODO:

- Add search URLS to be automatically added to all `ModelSerializerDocument` added to `search_indexes.py`
- Better documentation
- Better test coverage

1.6 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install drf-elasticsearch-dsl
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv drf-elasticsearch-dsl
$ pip install drf-elasticsearch-dsl
```

CHAPTER 3

Usage

3.1 TODO

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/ajbeach2/drf-elasticsearch-dsl/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

Django Package could always use more documentation, whether as part of the official Django Package docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ajbeach2/drf-elasticsearch-dsl/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *drf-elasticsearch-dsl* for local development.

1. Fork the *drf-elasticsearch-dsl* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/drf-elasticsearch-dsl.git
```

3. Install your local copy into a virtualenv:

```
$ virtualenv env && source env/bin/activate
$ pip install -r requirements_dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 drf_elasticsearch_dsl tests
$ make test
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/ajbeach2/drf-elasticsearch-dsl/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_drf_elasticsearch_dsl
```

CHAPTER 5

Credits

5.1 Development Lead

- Alexander Beach <ajbeach2@gmail.com>

5.2 Contributors

None yet. Why not be the first?