django-zombodb Documentation

Release 0.3.0

Flávio Juvenal

Oct 02, 2019

USER GUIDE

1 Installation and Configuration 3
1.1 Example 3
1.2 Requirements 3
1.3 Installation 3
1.4 Settings 3

2 Integrating with Elasticsearch 5
2.1 Installing ZomboDB extension 5
2.2 Activating ZomboDB extension 5
2.3 Creating an index 6
2.4 Field mapping 7

3 Searching 9
3.1 query_string_search 9
3.2 dsl_search 9
3.3 dict_search 9
3.4 Validation 10
3.5 Sorting by score 10
3.6 Limiting 10
3.7 Lazy and Chainable 10
3.8 Missing features 11

4 django_zombodb package 13
4.1 Submodules 13
4.2 django_zombodb.admin_mixins module 13
4.3 django_zombodb.apps module 13
4.4 django_zombodb.base_indexes module 13
4.5 django_zombodb.exceptions module 14
4.6 django_zombodb.helpers module 14
4.7 django_zombodb.indexes module 14
4.8 django_zombodb.operations module 15
4.9 django_zombodb.querysets module 15
4.10 django_zombodb.serializers module 15
4.11 Module contents 15

5 Change Log 17
5.1 0.3.0 (2019-07-18) 17
5.2 0.2.1 (2019-06-13) 17
5.3 0.2.0 (2019-03-01) 17
5.4 0.1.0 (2019-02-01) 17

6 Contributing **19**

6.1 Types of Contributions 19

6.2 Get Started! 20

6.3 Pull Request Guidelines 20

6.4 Tips 20

7 Credits **23**

7.1 Development Lead 23

7.2 Contributors 23

Python Module Index **25**

Index **27**

Easy Django integration with Elasticsearch through ZomboDB Postgres Extension. Thanks to ZomboDB, **your Django models are synced with Elasticsearch at transaction time**! Searching is also very simple: you can make Elasticsearch queries by just calling one of the search methods on your querysets. Couldn't be easier!

CHAPTER

ONE

INSTALLATION AND CONFIGURATION

1.1 Example

You can check a fully configured Django project with django-zombodb at https://github.com/vintasoftware/django-zombodb/tree/master/example

1.2 Requirements

- **Python**: 3.5, 3.6, 3.7
- **Django**: 2.0, 2.1

1.3 Installation

Install django-zombodb:

```
pip install django-zombodb
```

1.4 Settings

Set `ZOMBODB_ELASTICSEARCH_URL` on your settings.py. That is the URL of the ElasticSearch cluster used by ZomboDB.

```
ZOMBODB_ELASTICSEARCH_URL = 'http://localhost:9200/'
```

Move forward to learn how to integrate your models with Elasticsearch.

CHAPTER

TWO

INTEGRATING WITH ELASTICSEARCH

ZomboDB integrates Postgres with Elasticsearch through Postgres indexes. If you don't know much about ZomboDB, please read its tutorial before proceeding.

2.1 Installing ZomboDB extension

Since ZomboDB is a Postgres extension, you must install and activate it. Follow the official ZomboDB installation instructions.

2.2 Activating ZomboDB extension

django-zombodb provides a Django migration operation to activate ZomboDB extension on your database. To run it, please make sure your database user is a superuser:

```
psql -d your_database -c "ALTER USER your_database_user SUPERUSER"
```

Then create an empty migration on your "main" app (usually called "core" or "common"):

```
python manage.py makemigrations core --empty
```

Add the *django_zombodb.operations.ZomboDBExtension* operation to the migration you've just created:

```
import django_zombodb.operations

class Migration(migrations.Migration):

    dependencies = [
        ('restaurants', '0001_initial'),
    ]

    operations = [
        django_zombodb.operations.ZomboDBExtension(),
        ...
    ]
```

Alternatively, you can activate the extension manually with a command. But you should **avoid** this because you'll need to remember to run this on production, on tests, and on the machines of all your co-workers:

```
psql -d django_zombodb -c "CREATE EXTENSION zombodb"
```

2.3 Creating an index

Imagine you have the following model:

```
class Restaurant(models.Model):
    name = models.TextField()
    street = models.TextField()
```

To integrate it with Elasticsearch, we need to add a *ZomboDBIndex* to it:

```
from django_zombodb.indexes import ZomboDBIndex

class Restaurant(models.Model):
    name = models.TextField()
    street = models.TextField()

    class Meta:
        indexes = [
            ZomboDBIndex(fields=[
                'name',
                'street',
            ]),
        ]
```

After that, create and run the migrations:

```
python manage.py makemigrations
python manage.py migrate
```

Warning: During the migration, *ZomboDBIndex* reads the value at `settings.ZOMBODB_ELASTICSEARCH_URL`. That means if `settings.ZOMBODB_ELASTICSEARCH_URL` changes after the *ZomboDBIndex* migration, **the internal index stored at Postgres will still point to the old URL**. If you wish to change the URL of an existing *ZomboDBIndex*, change both `settings.ZOMBODB_ELASTICSEARCH_URL` and issue a `ALTER INDEX index_name SET (url='http://some.new.url');` (preferably inside a `migrations.RunSQL` in a new migration).

Now the `Restaurant` model will support Elasticsearch queries for both `name` and `street` fields. But to perform those searches, we need it to use the custom queryset *SearchQuerySet*:

```
from django_zombodb.indexes import ZomboDBIndex
from django_zombodb.querysets import SearchQuerySet

class Restaurant(models.Model):
    name = models.TextField()
    street = models.TextField()

    objects = models.Manager.from_queryset(SearchQuerySet)()

    class Meta:
        indexes = [
            ZomboDBIndex(fields=[
                'name',
                'street',
            ]),
        ]
```

Note: If you already have a custom queryset on your model, make it inherit from *SearchQuerySetMixin*.

2.4 Field mapping

From Elasticsearch documentation:

> "Mapping is the process of defining how a document, and the fields it contains, are stored and indexed. For instance, use mappings to define:
>
> - which string fields should be treated as full text fields.
> - which fields contain numbers, dates, or geolocations.
> - whether the values of all fields in the document should be indexed into the catch-all _all field.
> - the format of date values.
> - custom rules to control the mapping for dynamically added fields."

If you don't specify a mapping for your *ZomboDBIndex*, django-zombodb uses ZomboDB's default mappings, which are based on the Postgres type of your model fields.

To customize mapping, specify a `field_mapping` parameter to your *ZomboDBIndex* like below:

```
from django_zombodb.indexes import ZomboDBIndex
from django_zombodb.querysets import SearchQuerySet

class Restaurant(models.Model):
    name = models.TextField()
    street = models.TextField()

    objects = models.Manager.from_queryset(SearchQuerySet)()

    class Meta:
        indexes = [
            ZomboDBIndex(
                fields=[
                    'name',
                    'street',
                ],
                field_mapping={
                    'name': {"type": "text",
                             "copy_to": "zdb_all",
                             "analyzer": "fulltext_with_shingles",
                             "search_analyzer": "fulltext_with_shingles_search"},
                    'street': {"type": "text",
                               "copy_to": "zdb_all",
                               "analyzer": "brazilian"},
                }
            )
        ]
```

Note: You probably wish to have `"copy_to": "zdb_all"` on your textual fields to match ZomboDB default behavior. From ZomboDB docs: "`zdb_all` is ZomboDB's version of Elasticsearch's "_all" field, except `zdb_all` is enabled for all versions of Elasticsearch. It is also configured as the default search field for every ZomboDB index". For more info, read Elasticsearch docs take on the "_all" field.

Move forward to learn how to perform Elasticsearch queries through your model.

CHAPTER

THREE

SEARCHING

On models with *ZomboDBIndex*, use methods from *SearchQuerySet*/*SearchQuerySetMixin* to perform various kinds of Elasticsearch queries:

3.1 query_string_search

The *query_string_search()* method implements the simplest type of Elasticsearch queries: the ones with the query string syntax. To use it, just pass as an argument a string that follows the query string syntax.

```
Restaurant.objects.query_string_search("brasil~ AND steak*")
```

3.2 dsl_search

The query string syntax is user-friendly, but it's limited. For supporting all kinds of Elasticsearch queries, the recommended way is to use the *dsl_search()* method. It accepts arguments of elasticsearch-dsl-py `Query` objects. Those objects have the same representation power of the Elasticsearch JSON Query DSL. You can do "match", "term", and even compound queries like "bool".

Here we're using the elasticsearch-dsl-py `Q` shortcut to create `Query` objects:

```
from elasticsearch_dsl import Q as ElasticsearchQ

query = ElasticsearchQ(
    'bool',
    must=[
        ElasticsearchQ('match', name='pizza'),
        ElasticsearchQ('match', street='school')
    ]
)
Restaurant.objects.dsl_search(query)
```

3.3 dict_search

If you already have a Elasticsearch JSON query mounted as a `dict`, use the *dict_search()* method. The `dict` will be serialized using the `JSONSerializer` of elasticsearch-py, the official Python Elasticsearch client. This means dict values of `date`, `datetime`, `Decimal`, and `UUID` types will be correctly serialized.

3.4 Validation

If you're receiving queries from the end-user, particularly query string queries, you should call the search methods with `validate=True`. This will perform Elasticsearch-side validation through the Validate API. When doing that, *InvalidElasticsearchQuery* may be raised.

```
from django_zombodb.exceptions import InvalidElasticsearchQuery

queryset = Restaurant.objects.all()
try:
    queryset = queryset.query_string_search("AND steak*", validate=True)
except InvalidElasticsearchQuery:
    messages.error(request, "Invalid search query. Not filtering by search.")
```

3.5 Sorting by score

By default, the resulting queryset from the search methods is unordered. You can get results ordered by Elasticsearch's score passing `sort=True`.

```
Restaurant.objects.query_string_search("brasil~ AND steak*", sort=True)
```

Alternatively, if you want to combine with your own `order_by`, you can use the method *annotate_score()*:

```
Restaurant.objects.query_string_search(
    "brazil* AND steak*"
).annotate_score(
    attr='zombodb_score'
).order_by('-zombodb_score', 'name', 'pk')
```

3.6 Limiting

It's a good practice to set a hard limit to the number of search results. For most search use cases, you shouldn't need more than a certain number of results, either because users will only consume some of the high scoring results, or because documents with lower scores aren't relevant to your process. To limit the results, use the `limit` parameter on search methods:

```
Restaurant.objects.query_string_search("brasil~ AND steak*", limit=1000)
```

3.7 Lazy and Chainable

The search methods are like the traditional `filter` method: they return a regular Django `QuerySet` that supports all operations, and that's lazy and chainable. Therefore, you can do things like:

```
Restaurant.objects.filter(
    name__startswith='Pizza'
).query_string_search(
    'name:Hut'
).filter(
```

(continues on next page)

(continued from previous page)

```
    street__contains='Road'
)
```

Warning: It's fine to call `filter`/`exclude`/etc. before and after search. If possible, the best would be using only a Elasticsearch query. However, it's definitely **slow** to call search methods multiple times on the same queryset! **Please avoid this**:

```
Restaurant.objects.query_string_search(
    'name:Pizza'
).query_string_search(
    'name:Hut'
)
```

While that may work as expected, it's extremely inneficient. Instead, use compound queries like "bool". They'll be much faster. Note that "bool" queries might be quite confusing to implement. Check tutorials about them, like this one.

3.8 Missing features

Currently django-zombodb doesn't support ZomboDB's offset and sort functions that work on the Elasticsearch side. Regular SQL LIMIT/OFFSET/ORDER BY works fine, therefore traditional `QuerySet` operations work, but aren't as performant as doing the same on ES side.

CHAPTER

FOUR

DJANGO_ZOMBODB PACKAGE

4.1 Submodules

4.2 django_zombodb.admin_mixins module

class django_zombodb.admin_mixins.**ZomboDBAdminMixin**

Bases: object

get_list_display(*request*)

get_ordering(*request*)

get_queryset(*request*)

get_search_fields(*request*)

get_search_fields is unnecessary if ZomboDBAdminMixin is used. But since search_form.html uses this, we'll return a placeholder tuple

get_search_results(*request*, *queryset*, *search_term*)

max_search_results = None

4.3 django_zombodb.apps module

class django_zombodb.apps.**DjangoZomboDBConfig**(*app_name*, *app_module*)

Bases: django.apps.config.AppConfig

name = 'django_zombodb'

4.4 django_zombodb.base_indexes module

class django_zombodb.base_indexes.**PostgresIndex**(**, fields=(), name=None, db_tablespace=None, opclasses=(), condition=None*)

Bases: django.db.models.indexes.Index

create_sql(*model*, *schema_editor*, *using="*)

get_with_params()

max_name_length

4.5 django_zombodb.exceptions module

exception django_zombodb.exceptions.**InvalidElasticsearchQuery**

Bases: `Exception`

4.6 django_zombodb.helpers module

django_zombodb.helpers.**get_zombodb_index_from_model**(*model*)

django_zombodb.helpers.**validate_query_dict**(*model*, *query*)

django_zombodb.helpers.**validate_query_string**(*model*, *query*)

4.7 django_zombodb.indexes module

class django_zombodb.indexes.**ZomboDBIndex**(*, *shards=None*, *replicas=None*, *alias=None*, *refresh_interval=None*, *type_name=None*, *bulk_concurrency=None*, *batch_size=None*, *compression_level=None*, *llapi=None*, *field_mapping=None*, ***kwargs*)

Bases: `django.contrib.postgres.indexes.PostgresIndex`

create_sql(*model*, *schema_editor*, *using=''*)

deconstruct()

get_with_params()

remove_sql(*model*, *schema_editor*)

suffix = 'zombodb'

class django_zombodb.indexes.**ZomboDBIndexCreateStatementAdapter**(*statement*, *model*, *schema_editor*, *fields*, *field_mapping*, *row_type*)

Bases: `object`

references_column(**args*, ***kwargs*)

references_table(**args*, ***kwargs*)

rename_column_references(**args*, ***kwargs*)

rename_table_references(**args*, ***kwargs*)

template = 'CREATE INDEX %(name)s ON %(table)s USING zombodb ((ROW(%(columns)s)::%(row_

class django_zombodb.indexes.**ZomboDBIndexRemoveStatementAdapter**(*statement*, *row_type*)

Bases: `object`

references_column(**args*, ***kwargs*)

references_table(**args*, ***kwargs*)

rename_column_references(**args*, ***kwargs*)

rename_table_references(**args*, ***kwargs*)

4.8 django_zombodb.operations module

class `django_zombodb.operations.`**ZomboDBExtension**

Bases: `django.contrib.postgres.operations.CreateExtension`

4.9 django_zombodb.querysets module

class `django_zombodb.querysets.`**SearchQuerySet**(*model=None*, *query=None*, *using=None*, *hints=None*)

Bases: *django_zombodb.querysets.SearchQuerySetMixin*, `django.db.models.query.QuerySet`

class `django_zombodb.querysets.`**SearchQuerySetMixin**

Bases: `object`

annotate_score(*attr='zombodb_score'*)

dict_search(*query*, *validate=False*, *sort=False*, *score_attr='zombodb_score'*, *limit=None*)

dsl_search(*query*, *validate=False*, *sort=False*, *score_attr='zombodb_score'*, *limit=None*)

order_by_score(*score_attr='zombodb_score'*)

query_string_search(*query*, *validate=False*, *sort=False*, *score_attr='zombodb_score'*, *limit=None*)

4.10 django_zombodb.serializers module

4.11 Module contents

CHAPTER FIVE

CHANGE LOG

5.1 0.3.0 (2019-07-18)

- Support for custom Elasticsearch mappings through `field_mapping` parameter on `ZomboDBIndex`.
- Support to `limit` parameter on search methods.

5.2 0.2.1 (2019-06-13)

- Dropped support for Python 3.4.
- Added missing imports to docs.

5.3 0.2.0 (2019-03-01)

- Removed parameter `url` from `ZomboDBIndex`. This simplifies the support of multiple deployment environments (local, staging, production), because the ElasticSearch URL isn't copied to inside migrations code (see Issue #17).

5.4 0.1.0 (2019-02-01)

- First release on PyPI.

CHAPTER

SIX

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/vintasoftware/django-zombodb/issues. Please fill the fields of the issue template.

6.1.2 Fix Bugs

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

6.1.3 Implement Features

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

6.1.4 Write Documentation

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

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vintasoftware/django-zombodb/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 :)

6.2 Get Started!

Ready to contribute? Here's how to set up *django-zombodb* for local development.

1. Fork the *django-zombodb* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/django-zombodb.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv django-zombodb
   $ cd django-zombodb/
   $ pip install -e .
   $ make install_requirements
   ```

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 the linters and the tests, including testing other Python versions with tox:

   ```
   $ make lint
   $ make test
   $ make test-all
   ```

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.

6.3 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.
3. The pull request should pass CI. Check https://travis-ci.org/vintasoftware/django-zombodb/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ python runtests.py tests.test_apps
```

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Flávio Juvenal <flavio@vinta.com.br>

7.2 Contributors

None yet. Why not be the first?

PYTHON MODULE INDEX

d

django_zombodb, 15
django_zombodb.admin_mixins, 13
django_zombodb.apps, 13
django_zombodb.base_indexes, 13
django_zombodb.exceptions, 14
django_zombodb.helpers, 14
django_zombodb.indexes, 14
django_zombodb.operations, 15
django_zombodb.querysets, 15
django_zombodb.serializers, 15

INDEX

A

annotate_score() (*django_zombodb.querysets.SearchQuerySetMixin method*), 15

C

create_sql() (*django_zombodb.base_indexes.PostgresIndex method*), 13

create_sql() (*django_zombodb.indexes.ZomboDBIndex method*), 14

D

deconstruct() (*django_zombodb.indexes.ZomboDBIndex method*), 14

dict_search() (*django_zombodb.querysets.SearchQuerySetMixin method*), 15

django_zombodb (*module*), 15

django_zombodb.admin_mixins (*module*), 13

django_zombodb.apps (*module*), 13

django_zombodb.base_indexes (*module*), 13

django_zombodb.exceptions (*module*), 14

django_zombodb.helpers (*module*), 14

django_zombodb.indexes (*module*), 14

django_zombodb.operations (*module*), 15

django_zombodb.querysets (*module*), 15

django_zombodb.serializers (*module*), 15

DjangoZomboDBConfig (*class in django_zombodb.apps*), 13

dsl_search() (*django_zombodb.querysets.SearchQuerySetMixin method*), 15

G

get_list_display() (*django_zombodb.admin_mixins.ZomboDBAdminMixin method*), 13

get_ordering() (*django_zombodb.admin_mixins.ZomboDBAdminMixin method*), 13

get_queryset() (*django_zombodb.admin_mixins.ZomboDBAdminMixin method*), 13

get_search_fields() (*django_zombodb.admin_mixins.ZomboDBAdminMixin method*), 13

get_search_results() (*django_zombodb.admin_mixins.ZomboDBAdminMixin method*), 13

get_with_params() (*django_zombodb.base_indexes.PostgresIndex method*), 13

get_with_params() (*django_zombodb.indexes.ZomboDBIndex method*), 14

get_zombodb_index_from_model() (*in module django_zombodb.helpers*), 14

I

InvalidElasticsearchQuery, 14

M

max_name_length (*django_zombodb.base_indexes.PostgresIndex attribute*), 13

max_search_results (*django_zombodb.admin_mixins.ZomboDBAdminMixin attribute*), 13

N

name (*django_zombodb.apps.DjangoZomboDBConfig attribute*), 13

O

order_by_score() (*django_zombodb.querysets.SearchQuerySetMixin method*), 15

P

PostgresIndex (*class in django_zombodb.base_indexes*), 13

Q

query_string_search() (*django_zombodb.querysets.SearchQuerySetMixin method*), 15

R

references_column() (*django_zombodb.indexes.ZomboDBIndexCreateStatementAdapte method*), 14

references_column() (*django_zombodb.indexes.ZomboDBIndexRemoveStatementAdapter method*), 14
references_table() (*django_zombodb.indexes.ZomboDBIndexCreateStatementAdapter method*), 14
references_table() (*django_zombodb.indexes.ZomboDBIndexRemoveStatementAdapter method*), 14
remove_sql() (*django_zombodb.indexes.ZomboDBIndex method*), 14
rename_column_references() (*django_zombodb.indexes.ZomboDBIndexCreateStatementAdapter method*), 14
rename_column_references() (*django_zombodb.indexes.ZomboDBIndexRemoveStatementAdapter method*), 14
rename_table_references() (*django_zombodb.indexes.ZomboDBIndexCreateStatementAdapter method*), 14
rename_table_references() (*django_zombodb.indexes.ZomboDBIndexRemoveStatementAdapter method*), 14

S

SearchQuerySet (*class in django_zombodb.querysets*), 15
SearchQuerySetMixin (*class in django_zombodb.querysets*), 15
suffix (*django_zombodb.indexes.ZomboDBIndex attribute*), 14

T

template (*django_zombodb.indexes.ZomboDBIndexCreateStatementAdapter attribute*), 14

V

validate_query_dict() (*in module django_zombodb.helpers*), 14
validate_query_string() (*in module django_zombodb.helpers*), 14

Z

ZomboDBAdminMixin (*class in django_zombodb.admin_mixins*), 13
ZomboDBExtension (*class in django_zombodb.operations*), 15
ZomboDBIndex (*class in django_zombodb.indexes*), 14
ZomboDBIndexCreateStatementAdapter (*class in django_zombodb.indexes*), 14
ZomboDBIndexRemoveStatementAdapter (*class in django_zombodb.indexes*), 14