Django Gum Documentation

Release 1.1.0

Marcos Gabarda

April 11, 2016

Contents

Django Gum provides an easy way to integrate Django with **Elasticsearch 1.7.X**. It can be use for indexing models, handle mappings and make queries.

The design was made assuming the user has some knowledge about how Elasticsearch works, therefore, this package **DOES NOT** give you API like Django ORM.

CHAPTER 1

Getting Started

1.1 Installation

Use your PyPI to install the app:

```
pip install django-gum
```

1.2 Configuration

1.2.1 Add Gum to `INSTALLED_APPS`

As with most Django applications, you should add Gum to the `INSTALLED_APPS` within your settings file (usually `settings.py`).

Example:

```
INSTALLED_APPS += ('gum',)
```

1.2.2 Modify your `settings.py`

You have to add to your settings file where is the Elasticsearch server you want to use and which'll be the default index.

Example:

```
GUM_ELASTICSEARCH_URLS = ["http://127.0.0.1:9200/"]
GUM_ELASTICSEARCH_INDEX = ".gum-tests"
```

1.3 Handling data

1.3.1 Linking models and mapping

Each model have to has a Mapping Type associated whit it. To do this, you have to create an `index.py` file inside your app and create a `MappingType` class, and register this class with the model.

Example:

```
from gum.indexer import MappingType, indexer

class PostMappingType(MappingType):

    def document(self, instance):
        tags_text = " ".join(map(lambda x: x.label, instance.tags.all()))
        return {
            "title": instance.title,
            "content": instance.content,
            "text": "{} {} {}".format(instance.title, instance.content, tags_text)
        }

    def mapping(self):
        return {
            "properties": {
                "title": {
                    "type": "string",
                    "store": True,
                },
                "content": {
                    "type": "string",
                    "store": True,
                },
                "text": {
                    "type": "string",
                    "store": True,
                }
            }
        }

indexer.register(Post, PostMappingType)
```

1.3.2 Updating index

You can use this command to update all registers models:

```
./manage.py gum --update-index
```

Or you can only update specified models:

```
./manage.py gum --update-index blog.Post
```

1.3.3 Making queries

You can perform Elasticsearch searches (accessing `search` method) using `elasticseaech` model attribute.

Example:

```
response = Post.elasticsearch.search(body={
    "query": {
        "match_all": {}
    }
})
```

CHAPTER 2

Configuration and defaults

2.1 Gum options

2.1.1 GUM_DEBUG

Activates/deactivates the debug output.

Defaults to `False`.

2.1.2 GUM_USE_CELERY

Boolean setting to activate/desactivate the use of Celery to update a model document when it is saved.

Defaults to `False`.

2.1.3 GUM_INDEX_FILES

You can use this variable to give a list of index files to be checked by Gum in order to find registered mappings, in addition to index.py files.

Defaults to `tuple()`.

2.2 Elasticsearch options

2.2.1 GUM_ELASTICSEARCH_URLS

A list of address of Elasticsearch servers.

Defaults to `["http://127.0.0.1:9200/"]`.

2.2.2 GUM_ELASTICSEARCH_CONNECTION_PARAMS

Parameters given to Elasticsearch class to connect with Elasticsearch server.

Defatils to:

```
{
    "timeout": 5
}
```

2.2.3 GUM_ELASTICSEARCH_INDEX

Index name to use by default.

Defaults to `"_all"`.

2.2.4 GUM_DEFAULT_ELASTICSEARCH_SETTINGS

Defines the settings of all indices created by Gum.

Defaults to:

```
{
    'settings': {
        "analysis": {
            "analyzer": {
                "ngram_analyzer": {
                    "type": "custom",
                    "tokenizer": "lowercase",
                    "filter": ["gum_ngram"]
                },
                "edgengram_analyzer": {
                    "type": "custom",
                    "tokenizer": "lowercase",
                    "filter": ["gum_edgengram"]
                }
            },
            "tokenizer": {
                "gum_ngram_tokenizer": {
                    "type": "nGram",
                    "min_gram": 3,
                    "max_gram": 15,
                },
                "gum_edgengram_tokenizer": {
                    "type": "edgeNGram",
                    "min_gram": 2,
                    "max_gram": 15,
                    "side": "front"
                }
            },
            "filter": {
                "gum_ngram": {
                    "type": "nGram",
                    "min_gram": 3,
                    "max_gram": 15
                },
                "gum_edgengram": {
                    "type": "edgeNGram",
                    "min_gram": 2,
                    "max_gram": 15
                }
            }
```

```
            }
        }
}
```

CHAPTER 3

Management commands

3.1 gum

Base command to manage Gum based indices. It accepts the following arguments:

--update-index (<app_name.model> <app_name.model> ...): By default, it updates all registered models indices. It can be provided a list of models to restrict the update.

--update-settings: It updates the settings of the indices.

--remove: Deletes all indices.

Examples:

```
# Update everything.
./manage.py gum --update-index
# Update a model.
./manage.py gum --update-index blog.Post
```

Note: This package was initialy developed for Onpublico project.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search