ESEngine

Release latest

January 07, 2016

Contents

ESEngine is an ODM (Object Document Mapper) it maps Python classes in to Elasticsearch index/doc_type and object instances() in to Elasticsearch documents and also provides a Pythonic query builder.

Install with `pip install esengine`

CHAPTER 1

Payload

The main `Payload` class allows you to build Elasticsearch queries

```
from esngine import Payload, Query, Aggregate

# Create a new payload
p = Payload()

# Match something
p.query(Query.match('some_field', 'some_text'))

# Aggregate something
p.aggregate(Aggregate.terms('my_terms', 'some_field'))
```

1.1 API

1.2 Exceptions

CHAPTER 2

Queries

Note that all Query calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

2.1 Query.span_or

```
Query.span_or([Query])
```

2.2 Query.terms

```
Query.terms(field, [value])
```

2.3 Query.has_child

```
Query.has_child(type, filter=Filter, query=Query)
```

2.4 Query.span_first

```
Query.span_first(Query)
```

2.5 Query.prefix

```
Query.prefix(field, value, boost=None)
```

2.6 Query.term

```
Query.term(field, value, boost=None)
```

2.7 Query.fuzzy

```
Query.fuzzy(field, value, boost=None, fuzziness=None, prefix_length=None, max_expansions=None)
```

2.8 Query.nested

```
Query.nested(path, Query)
```

2.9 Query.dis_max

```
Query.dis_max([Query])
```

2.10 Query.query_string

```
Query.query_string(query, fields=[])
```

2.11 Query.fuzzy_like_this

```
Query.fuzzy_like_this([fields], like_text)
```

2.12 Query.has_parent

```
Query.has_parent(parent_type, filter=Filter, query=Query)
```

2.13 Query.function_score

```
Query.function_score([functions], filter=Filter, query=Query)
```

2.14 Query.geo_shape

```
Query.geo_shape(field, type=None, coordinates=[])
```

2.15 Query.fuzzy_like_this_field

```
Query.fuzzy_like_this_field(field, like_text, max_query_terms=None, ignore_tf=None, fuzziness=None, p
```

2.16 Query.span_multi

```
Query.span_multi(Query)
```

2.17 Query.match_all

```
Query.match_all(boost=None)
```

2.18 Query.span_near

```
Query.span_near([Query])
```

2.19 Query.simple_query_string

```
Query.simple_query_string(query, fields=[])
```

2.20 Query.multi_match

```
Query.multi_match([fields], query)
```

2.21 Query.span_term

```
Query.span_term(field, value, boost=None)
```

2.22 Query.regexp

```
Query.regexp(field, value, boost=None, flags=None)
```

2.23 Query.ids

```
Query.ids([values], type=None)
```

2.24 Query.more_like_this

```
Query.more_like_this([fields], like_text)
```

2.25 Query.range

```
Query.range(field, gte=None, gt=None, lte=None, lt=None)
```

2.26 Query.bool

```
Query.bool(must=[Query], must_not=[Query], should=[Query])
```

2.27 Query.common

```
Query.common(query)
```

2.28 Query.wildcard

```
Query.wildcard(field, value, boost=None)
```

2.29 Query.indices

```
Query.indices([indices], query=Query, no_match_query=Query)
```

2.30 Query.filtered

```
Query.filtered(filter=Filter, query=Query)
```

2.31 Query.span_not

```
Query.span_not(include=Query, exclude=Query)
```

2.32 Query.boost

```
Query.boost(positive=None, negative=None)
```

2.33 Query.constant_score

```
Query.constant_score(filter=Filter, query=Query)
```

2.34 Query.match

```
Query.match(field, query, operator=None, zero_terms_query=None, cutoff_frequency=None, boost=None)
```

2.35 Query.top_children

```
Query.top_children(type, query=Query)
```

CHAPTER 3

Aggregates

Note that all Aggregate calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

3.1 Aggregate.geo_bounds

```
Aggregate.geo_bounds(name, field)
```

3.2 Aggregate.date_histogram

```
Aggregate.date_histogram(name, field, interval)
```

3.3 Aggregate.global

```
Aggregate.global(name)
```

3.4 Aggregate.nested

```
Aggregate.nested(name, path)
```

3.5 Aggregate.ip_range

```
Aggregate.ip_range(name, field, [ranges])
```

3.6 Aggregate.filters

```
Aggregate.filters(name, [Filter])
```

3.7 Aggregate.avg

```
Aggregate.avg(name, field)
```

3.8 Aggregate.children

```
Aggregate.children(name, type)
```

3.9 Aggregate.stats

```
Aggregate.stats(name, field)
```

3.10 Aggregate.scripted_metric

```
Aggregate.scripted_metric(name)
```

3.11 Aggregate.min

```
Aggregate.min(name, field)
```

3.12 Aggregate.sum

```
Aggregate.sum(name, field)
```

3.13 Aggregate.extended_stats

```
Aggregate.extended_stats(name, field)
```

3.14 Aggregate.value_count

```
Aggregate.value_count(name, field)
```

3.15 Aggregate.percentiles

```
Aggregate.percentiles(name, field)
```

3.16 Aggregate.terms

```
Aggregate.terms(name, field)
```

3.17 Aggregate.missing

```
Aggregate.missing(name, field)
```

3.18 Aggregate.max

```
Aggregate.max(name, field)
```

3.19 Aggregate.histogram

```
Aggregate.histogram(name, field, interval)
```

3.20 Aggregate.date_range

```
Aggregate.date_range(name, field, [ranges])
```

3.21 Aggregate.cardinality

```
Aggregate.cardinality(name, field)
```

3.22 Aggregate.geohash_grid

```
Aggregate.geohash_grid(name, field)
```

3.23 Aggregate.geo_distance

```
Aggregate.geo_distance(name, field, origin, [ranges])
```

3.24 Aggregate.filter

```
Aggregate.filter(name, Filter)
```

3.25 Aggregate.percentile_ranks

```
Aggregate.percentile_ranks(name, field)
```

3.26 Aggregate.range

```
Aggregate.range(name, field, [ranges])
```

3.27 Aggregate.significant_terms

```
Aggregate.significant_terms(name, field)
```

3.28 Aggregate.top_hits

```
Aggregate.top_hits(name)
```

3.29 Aggregate.reverse_nested

```
Aggregate.reverse_nested(name)
```

CHAPTER 4

Suggesters

Note that all Suggester calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

4.1 Suggester.completion

```
Suggester.completion(field, size=None)
```

4.2 Suggester.phrase

```
Suggester.phrase(field, gram_size=None, real_word_error_likelihood=None, confidence=None, max_errors=
```

4.3 Suggester.term

```
Suggester.term(field, analyzer=None, size=None, sort=None, suggest_mode=None)
```

CHAPTER 5

Filters

Note that all Filter calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

5.1 Filter.geohash_shell

```
Filter.geohash_shell(field, lat=None, lon=None)
```

5.2 Filter.geo_polygon

```
Filter.geo_polygon(field, [points])
```

5.3 Filter.exists

```
Filter.exists(field)
```

5.4 Filter.not_

```
Filter.not_(filter=Filter, query=Query)
```

5.5 Filter.nested

```
Filter.nested(path, Filter)
```

5.6 Filter.prefix

```
Filter.prefix(field, value)
```

5.7 Filter.has_parent

```
Filter.has_parent(parent_type, filter=Filter, query=Query)
```

5.8 Filter.geo_distance_range

```
Filter.geo_distance_range(field, lat=None, lon=None)
```

5.9 Filter.script

```
Filter.script(script)
```

5.10 Filter.bool

```
Filter.bool(must=[Filter], must_not=[Filter], should=[Filter])
```

5.11 Filter.type

```
Filter.type(value)
```

5.12 Filter.terms

```
Filter.terms(field, [value])
```

5.13 Filter.has_child

```
Filter.has_child(type, filter=Filter, query=Query)
```

5.14 Filter.missing

```
Filter.missing(field)
```

5.15 Filter.term

```
Filter.term(field, value)
```

5.16 Filter.geo_shape

```
Filter.geo_shape(field, type=None, coordinates=[])
```

5.17 Filter.regexp

```
Filter.regexp(field, value, flags=None, max_determinized_states=None)
```

5.18 Filter.or_

```
Filter.or_([Filter])
```

5.19 Filter.match_all

```
Filter.match_all(None)
```

5.20 Filter.geo_distance

```
Filter.geo_distance(field, lat=None, lon=None)
```

5.21 Filter.geo_bounding_box

```
Filter.geo_bounding_box(field, top_left=None, bottom_right=None)
```

5.22 Filter.and_

```
Filter.and_([Filter])
```

5.23 Filter.ids

```
Filter.ids([values], type=None)
```

5.24 Filter.range

```
Filter.range(field, gte=None, gt=None, lte=None, lt=None)
```

5.25 Filter.limit

```
Filter.limit(value)
```

5.26 Filter.indices

```
Filter.indices([indices], filter=Filter, no_match_filter=Filter)
```

CHAPTER 6

Meta

- genindex
- modindex

CHAPTER 7

Credits

This library is created and maintained by http://catholabs.com/authors/ team

- Contributors: https://github.com/catholabs/esengine/graphs/contributors

A lot of Open Source software were used to develop and inspire this project:

- PyCharm (IDE)
- Py.Test (test suit)
- TravisCI (Continuous Integration)
- Landscape.io (Code Quality Assurance)
- Elasticsearch-py (Base Client)
- Elasticsearch-dsl (inspiration for models)
- ElasticQuery by "Nick Barrett/@Fizzadar" (inspiration for query builder)