# pyramid_es Documentation

***Release 0.2.1***

**Scott Torborg**

March 01, 2014

Scott Torborg - Cart Logic

`pyramid_es` is a pattern and set of utilities for integrating the elasticsearch search engine with a Pyramid web app. It is intended to make it easy to index a set of persisted objects and search those documents inside Pyramid views.

CHAPTER 1

# Example Usage

```
client = get_client(request)
result = client.query(Movie).\
    filter_term('year', 1987).\
    order_by('rating').\
    execute()
```

CHAPTER 2

# Contents

## 2.1 Quick Start

### 2.1.1 Install

Install with pip:

```
$ pip install pyramid_es
```

### 2.1.2 Integrate with a Pyramid App

Include pyramid_es, by calling `config.include('pyramid_es')` or adding pyramid_es to `pyramid.includes`.

Configure the following settings:

- `elastic.servers`
- `elastic.timeout`
- `elastic.index`
- `elastic.disable_indexing`

### 2.1.3 Add the Mixin Class to a Model

Add `ElasticMixin` to a model class. For example:

```
from pyramid_es.mixin import ElasticMixin

class Article(Base, ElasticMixin):
    ...
```

Then implement the `elastic_mapping()` class method:

```
from pyramid_es.mixin import ElasticMixin, ESMapping, ESString, ESField

class Article(Base, ElasticMixin):
    ...

    @classmethod
    def elastic_mapping(cls):
```

```
        return ESMapping(
            return ESMapping(
                analyzer='content',
                properties=ESMapping(
                    ESString('title', boost=5.0),
                    ESString('body'),
                    ESField('pubdate'))))
```

You can customize the exact behavior of the mapping and document creation by adjusting the `elastic_mapping(cls)` class method and the `elastic_document(self)` instance method.

### 2.1.4 Access the Client

To interact with the elasticsearch server, use the client instance maintained by `pyramid_es`. You can access it like:

```
from pyramid_es import get_client

client = get_client(registry)
```

All operations–index maintenance, diagnostics, indexing, and querying–are performed via methods on this instance.

### 2.1.5 Index a Document

After the model class is prepared, index a document with:

```
client.index_object(article)
```

This call will create or update the elasticsearch backend state for this model object, so you can simply call it any time the object is created or updated. If the object is deleted, call:

```
client.delete_object(article)
```

### 2.1.6 Execute a Search Query

Search queries are formed generatively, much like SQLAlchemy. Here's an example:

```
q = client.query(Article)
q = q.filter_term('title', 'Introduction')
q = q.order_by('pubdate', desc=True)
results = q.execute()

for result in results:
    print result.title, result.pubdate
```

To make a keyword search, add the `q` argument to `client.query()`:

```
q = client.query(Article, q='kittens')
```

Calling a query method like `.filter_term()` or `.order_by()` will create a totally new query instance, and not modify the original.

You can use query methods to:

- Add filters on specific fields, range filters, or anything else supported by elasticsearch
- Sort by fields

- Add search facets

### 2.1.7 The Result Object

Calling `.execute()` on a query issues the query to the backend and returns a special result object. This object behaves similar to a dict, but supports iteration and a few special properties.

## 2.2 API Reference

### 2.2.1 Client

pyramid_es.**client_from_config**(*settings*, *prefix='elastic.'*)
Instantiate and configure an Elasticsearch from settings.

pyramid_es.**get_client**(*request*)
Get the registered Elasticsearch client. The supplied argument can be either a `Request` instance or a `Registry`.

**class** pyramid_es.client.**ElasticClient**(*servers*, *index*, *timeout=1.0*, *disable_indexing=False*)
A handle for interacting with the Elasticsearch backend.

**analyze**(*text*, *analyzer*)
Preview the result of analyzing a block of text using a given analyzer.

**delete_index**()
Delete the index on the ES server.

**delete_mapping**(*cls*)
Delete the mapping corresponding to `cls` on the server. Does not delete subclass mappings.

**ensure_all_mappings**(*base_class*, *recreate=False*)
Initialize explicit mappings for all subclasses of the specified SQLAlcehmy declarative base class.

**ensure_index**(*recreate=False*)
Ensure that the index exists on the ES server, and has up-to-date settings.

**ensure_mapping**(*cls*, *recreate=False*)
Put an explicit mapping for the given class if it doesn't already exist.

**get**(*obj*, *routing=None*)
Retrieve the ES source document for a given object or (document type, id) pair.

**get_mappings**(*cls=None*)
Return the object mappings currently used by ES.

**index_document**(*id*, *doc_type*, *doc*, *parent=None*)
Add or update the indexed document from a raw document source (not an object).

**index_object**(*obj*)
Add or update the indexed document for an object.

**index_objects**(*objects*)
Add multiple objects to the index.

**query**(**classes*, ***kw*)
Return an ElasticQuery against the specified class.

**refresh**()
Refresh the ES index.

**search**(*body*, *classes=None*, *fields=None*, ***query_params*)
Run ES search using default indexes.

**subtype_names**(*cls*)
Return a list of document types to query given an object class.

### 2.2.2 Model Mixin

Utility classes intended to make it easier to specify Elastic Search mappings for model objects.

**class** pyramid_es.mixin.**ESField**(*name*, *filter=None*, *attr=None*, ***kwargs*)
A leaf property that doesn't emit a mapping definition.

This behavior is useful if you want to allow Elastic Search to automatically construct an appropriate mapping while indexing.

**class** pyramid_es.mixin.**ESMapping**(**args*, ***kwargs*)
ESMapping defines a tree-like DSL for building Elastic Search mappings.

Calling dict(es_mapping_object) produces an Elastic Search mapping definition appropriate for pyes.

Applying an ESMapping to another object returns an Elastic Search document.

**properties**
Return the dictionary {name: property, ...} describing the top-level properties in this mapping, or None if this mapping is a leaf.

**update**(*m*)
Return a copy of the current mapping merged with the properties of another mapping. update merges just one level of hierarchy and uses simple assignment below that.

**class** pyramid_es.mixin.**ESProp**(*name*, *filter=None*, *attr=None*, ***kwargs*)
A leaf property.

**class** pyramid_es.mixin.**ESString**(*name*, ***kwargs*)
A string property.

**class** pyramid_es.mixin.**ElasticMixin**
Mixin for SQLAlchemy classes that use ESMapping.

**elastic_document**()
Apply the class ES mapping to the current instance.

**classmethod elastic_mapping**()
Return an ES mapping for the current class. Should basically be some form of `return ESMapping(...)`.

**class** pyramid_es.mixin.**ElasticParent**
Descriptor to return the parent document type of a class or the parent document ID of an instance.

The child class should specify a property:

__elastic_parent__ = ('ParentDocType', 'parent_id_attr')

### 2.2.3 Queries

**class** pyramid_es.query.**ElasticQuery**(*client*, *classes=None*, *q=None*)
Represents a query to be issued against the ES backend.

**add_facet**(**args*, ***kwargs*)
Add a query facet, to return data used for the implementation of faceted search (e.g. returning result counts for given possible sub-queries).

The facet should be supplied as a dict in the format that ES uses for representation.

It is recommended to use the helper methods `add_term_facet()` or `add_range_facet()` where possible.

**add_range_facet**(*name*, *field*, *ranges*)
Add a range facet.

ES will return data about documetn counts for the top sub-queries (by document count) inw hich the results are filtered by a given numerical range.

**add_term_facet**(*name*, *size*, *field*)
Add a term facet.

ES will return data about document counts for the top sub-queries (by document count) in which the results are filtered by a given term.

**count**()
Execute this query to determine the number of documents that would be returned, but do not actually fetch documents. Returns an int.

**execute**(*start=None*, *size=None*, *fields=None*)
Execute this query and return a result set.

**filter_term**(**args*, ***kwargs*)
Filter for documents where the field `term` matches `value`.

**filter_terms**(**args*, ***kwargs*)
Filter for documents where the field `term` matches one of the elements in `value` (which should be a sequence).

**filter_value_lower**(**args*, ***kwargs*)
Filter for documents where term is numerically more than `lower`.

**filter_value_upper**(**args*, ***kwargs*)
Filter for documents where term is numerically less than `upper`.

**limit**(**args*, ***kwargs*)
When returning results, stop at document `n`.

**static match_all_query**()
Static method to return a filter dict which will match everything. Can be overridden in a subclass to customize behavior.

**offset**(**args*, ***kwargs*)
When returning results, start at document `n`.

**order_by**(**args*, ***kwargs*)
Sort results by the field `key`. Default to ascending order, unless `desc` is True.

**size**(**args*, ***kwargs*)
When returning results, stop at document `n`.

**start**(**args*, ***kwargs*)
When returning results, start at document `n`.

**static text_query**(*phrase*, *operator='and'*)
Static method to return a filter dict to match a text search. Can be overridden in a subclass to customize behavior.

pyramid_es.query.**filters**(*f*)
A convenience decorator to wrap query methods that are adding filters. To use, simply make a method that returns a filter dict in elasticsearch's JSON object format.

Should be used inside @generative (listed after in decorator order).

pyramid_es.query.**generative**(*f*)
A decorator to wrap query methods to make them automatically generative.

**class** pyramid_es.result.**ElasticResult**(*raw*)
Wrapper for an Elasticsearch result set. Provides access to the documents, result aggregate data (like total count), and facets.

**facets**
Return the facets returned by this seach query.

**total**
Return the total number of docs which would have been matched by this query. Note that this is not necessarily the same as the number of document result records associated with this object, because the query may have a start / size applied.

**class** pyramid_es.result.**ElasticResultRecord**(*raw*)
Wrapper for an Elasticsearch result record. Provides access to the indexed document, ES result data (like score), and the mapped object.

## 2.3 Contributing

Patches and suggestions are strongly encouraged! GitHub pull requests are preferred, but other mechanisms of feedback are welcome.

pyramid_es has a comprehensive test suite with 100% line and branch coverage, as reported by the excellent `coverage` module. To run the tests, simply run in the top level of the repo:

```
$ tox
```

This will also ensure that the Sphinx documentation builds correctly, and that there are no PEP8 or Pyflakes warnings in the codebase.

Any pull requests should preserve all of these things.

CHAPTER 3

# Indices and Tables

- *genindex*
- *modindex*

## p