Spring Data Elasticsearch

BioMed Central Development Team

Version 1.1.4.RELEASE
2015-10-14

Table of Contents

Preface

The Spring Data Elasticsearch project applies core Spring concepts to the development of solutions using the Elasticsearch Search Engine. We have povided a "template" as a high-level abstraction for storing,querying,sorting and faceting documents. You will notice similarities to the Spring data solr and mongodb support in the Spring Framework.

Chapter 1. Project Metadata

- Version Control - https://github.com/spring-projects/spring-data-elasticsearch
- Bugtracker - https://jira.spring.io/browse/DATAES
- Release repository - https://repo.spring.io/libs-release
- Milestone repository - https://repo.spring.io/libs-milestone
- Snapshot repository - https://repo.spring.io/libs-snapshot

Chapter 2. Requirements

Requires Elasticsearch 0.20.2 and above or optional dependency or not even that if you are using Embedded Node Client

Reference Documentation

Chapter 3. Elasticsearch Repositories

This chapter includes details of the Elasticsearch repository implementation.

3.1. Introduction

3.1.1. Spring Namespace

The Spring Data Elasticsearch module contains a custom namespace allowing definition of repository beans as well as elements for instantiating a `ElasticsearchServer` .

Using the `repositories` element looks up Spring Data repositories as described in [repositories.create-instances] .

Example 1. Setting up Elasticsearch repositories using Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
http://www.springframework.org/schema/data/elasticsearch
http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch-
1.0.xsd">

    <elasticsearch:repositories base-package="com.acme.repositories" />

</beans>
```

Using the `Transport Client` or `Node Client` element registers an instance of `Elasticsearch Server` in the context.

Example 2. Transport Client using Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
http://www.springframework.org/schema/data/elasticsearch
http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch-
1.0.xsd">

    <elasticsearch:transport-client id="client" cluster-nodes=
"localhost:9300,someip:9300" />

</beans>
```

Example 3. Node Client using Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:elasticsearch="http://www.springframework.org/schema/data/elasticsearch"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
http://www.springframework.org/schema/data/elasticsearch
http://www.springframework.org/schema/data/elasticsearch/spring-elasticsearch-
1.0.xsd">

    <elasticsearch:node-client id="client" local="true"" />

</beans>
```

3.1.2. Annotation based configuration

The Spring Data Elasticsearch repositories support cannot only be activated through an XML namespace but also using an annotation through JavaConfig.

Example 4. Spring Data Elasticsearch repositories using JavaConfig

```
@Configuration
@EnableElasticsearchRepositories(basePackages =
"org/springframework/data/elasticsearch/repositories")
static class Config {

    @Bean
    public ElasticsearchOperations elasticsearchTemplate() {
        return new ElasticsearchTemplate(nodeBuilder().local(true).node().client());
    }
}
```

The configuration above sets up an `Embedded Elasticsearch Server` which is used by the `ElasticsearchTemplate` . Spring Data Elasticsearch Repositories are activated using the `@EnableElasticsearchRepositories` annotation, which essentially carries the same attributes as the XML namespace does. If no base package is configured, it will use the one the configuration class resides in.

3.1.3. Elasticsearch Repositores using CDI

The Spring Data Elasticsearch repositories can also be set up using CDI functionality.

Example 5. Spring Data Elasticsearch repositories using JavaConfig

```
class ElasticsearchTemplateProducer {

    @Produces
    @ApplicationScoped
    public ElasticsearchOperations createElasticsearchTemplate() {
        return new ElasticsearchTemplate(nodeBuilder().local(true).node().client());
    }
}

class ProductService {

    private ProductRepository repository;

    public Page<Product> findAvailableBookByName(String name, Pageable pageable) {
        return repository.findByAvailableTrueAndNameStartingWith(name, pageable);
    }

    @Inject
    public void setRepository(ProductRepository repository) {
        this.repository = repository;
    }
}
```

3.2. Query methods

3.2.1. Query lookup strategies

The Elasticsearch module supports all basic query building feature as String,Abstract,Criteria or have it being derived from the method name.

Declared queries

Deriving the query from the method name is not always sufficient and/or may result in unreadable method names. In this case one might make either use of `@Query` annotation (see Using @Query Annotation).

3.2.2. Query creation

Generally the query creation mechanism for Elasticsearch works as described in [repositories.query-methods] . Here's a short example of what a Elasticsearch query method translates into:

Example 6. Query creation from method names

```
public interface BookRepository extends Repository<Book, String>
{
    List<Book> findByNameAndPrice(String name, Integer price);
}
```

The method name above will be translated into the following Elasticsearch json query

```
{ "bool" :
    { "must" :
        [
            { "field" : {"name" : "?"} },
            { "field" : {"price" : "?"} }
        ]
    }
}
```

A list of supported keywords for Elasticsearch is shown below.

Table 1. Supported keywords inside method names

Keyword	Sample	Elasticsearch Query String
And	findByNameAndPrice	{"bool" : {"must" : [{"field" : {"name" : "?"}}, {"field" : {"price" : "?"}}]}}
Or	findByNameOrPrice	{"bool" : {"should" : [{"field" : {"name" : "?"}}, {"field" : {"price" : "?"}}]}}
Is	findByName	{"bool" : {"must" : {"field" : {"name" : "?"}}}}
Not	findByNameNot	{"bool" : {"must_not" : {"field" : {"name" : "?"}}}}
Between	findByPriceBetween	{"bool" : {"must" : {"range" : {"price" : {"from" : ?,"to" : ?,"include_lower" : true,"include_upper" : true}}}}}
LessThanEqual	findByPriceLessThan	{"bool" : {"must" : {"range" : {"price" : {"from" : null,"to" : ?,"include_lower" : true,"include_upper" : true}}}}

Keyword	Sample	Elasticsearch Query String
GreaterThanEqual	findByPriceGreaterThan	{"bool" : {"must" : {"range" : {"price" : {"from" : ?,"to" : null,"include_lower" : true,"include_upper" : true}}}}}
Before	findByPriceBefore	{"bool" : {"must" : {"range" : {"price" : {"from" : null,"to" : ?,"include_lower" : true,"include_upper" : true}}}}}
After	findByPriceAfter	{"bool" : {"must" : {"range" : {"price" : {"from" : ?,"to" : null,"include_lower" : true,"include_upper" : true}}}}}
Like	findByNameLike	{"bool" : {"must" : {"field" : {"name" : {"query" : "?*","analyze_wildcard" : true}}}}}
StartingWith	findByNameStartingWith	{"bool" : {"must" : {"field" : {"name" : {"query" : "?*","analyze_wildcard" : true}}}}}
EndingWith	findByNameEndingWith	{"bool" : {"must" : {"field" : {"name" : {"query" : "*?","analyze_wildcard" : true}}}}}
Contains/Containing	findByNameContaining	{"bool" : {"must" : {"field" : {"name" : {"query" : "?","analyze_wildcard" : true}}}}}
In	findByNameIn(Collection<String> names)	{"bool" : {"must" : {"bool" : {"should" : [{"field" : {"name" : "?"}}, {"field" : {"name" : "?"}}]}}}}
NotIn	findByNameNotIn(Collection<String>names)	{"bool" : {"must_not" : {"bool" : {"should" : {"field" : {"name" : "?"}}}}}}
Near	findByStoreNear	Not Supported Yet !
True	findByAvailableTrue	{"bool" : {"must" : {"field" : {"available" : true}}}}
False	findByAvailableFalse	{"bool" : {"must" : {"field" : {"available" : false}}}}
OrderBy	findByAvailableTrueOrderByNameDesc	{"sort" : [{ "name" : {"order" : "desc"} }],"bool" : {"must" : {"field" : {"available" : true}}}}

3.2.3. Using @Query Annotation

Example 7. Declare query at the method using the `@Query` annotation.

```
public interface BookRepository extends ElasticsearchRepository<Book, String> {
    @Query("{"bool" : {"must" : {"field" : {"name" : "?0"}}}}")
    Page<Book> findByName(String name,Pageable pageable);
}
```

Chapter 4. Miscellaneous Elasticsearch Operation Support

This chapter covers additional support for Elasticsearch operations that cannot be directly accessed via the repository interface. It is recommended to add those operations as custom implementation as described in [repositories.custom-implementations] .

4.1. Filter Builder

Filter Builder improves query speed.

```
private ElasticsearchTemplate elasticsearchTemplate;

SearchQuery searchQuery = new NativeSearchQueryBuilder()
    .withQuery(matchAllQuery())
    .withFilter(boolFilter().must(termFilter("id", documentId)))
    .build();

Page<SampleEntity> sampleEntities =
    elasticsearchTemplate.queryForPage(searchQuery,SampleEntity.class);
```

4.2. Using Scan And Scroll For Big Result Set

Elasticsearch has scan and scroll feature for getting big result set in chunks. `ElasticsearchTemplate` has scan and scroll methods that can be used as below.

Example 8. Using Scan and Scroll

```
SearchQuery searchQuery = new NativeSearchQueryBuilder()
    .withQuery(matchAllQuery())
    .withIndices("test-index")
    .withTypes("test-type")
    .withPageable(new PageRequest(0,1))
    .build();
String scrollId = elasticsearchTemplate.scan(searchQuery,1000,false);
List<SampleEntity> sampleEntities = new ArrayList<SampleEntity>();
boolean hasRecords = true;
while (hasRecords){
    Page<SampleEntity> page = elasticsearchTemplate.scroll(scrollId, 5000L , new
ResultsMapper<SampleEntity>()
    {
        @Override
        public Page<SampleEntity> mapResults(SearchResponse response) {
            List<SampleEntity> chunk = new ArrayList<SampleEntity>();
            for(SearchHit searchHit : response.getHits()){
                if(response.getHits().getHits().length <= 0) {
                    return null;
                }
                SampleEntity user = new SampleEntity();
                user.setId(searchHit.getId());
                user.setMessage((String)searchHit.getSource().get("message"));
                chunk.add(user);
            }
            return new PageImpl<SampleEntity>(chunk);
        }
    });
    if(page != null) {
        sampleEntities.addAll(page.getContent());
        hasRecords = page.hasNextPage();
    }
    else{
        hasRecords = false;
    }
    }
}
```

Appendix