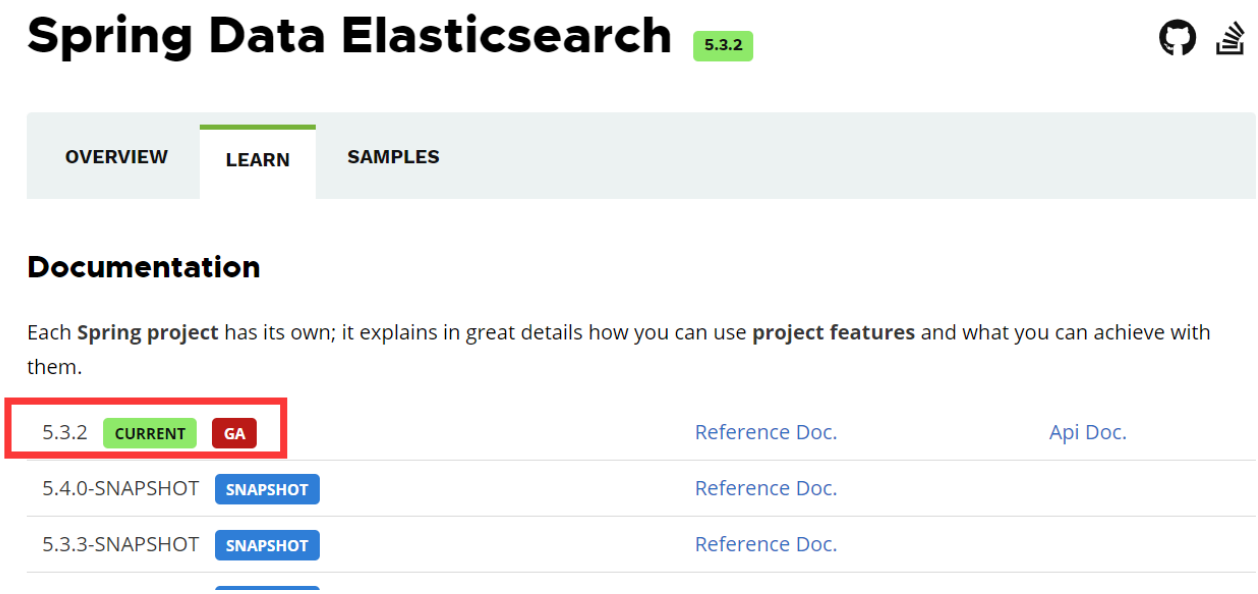
> 主讲老师：Fox
> 有道云笔记地址：https://note.youdao.com/s/dhwCycLz

1. Spring Data Elasticsearch的介绍

Spring Data Elasticsearch 基于 spring data API 简化 Elasticsearch 操作，将原始操作Elasticsearch 的客户端 API 进行封装 。Spring Data 为 Elasticsearch 项目提供集成搜索引擎。Spring Data Elasticsearch POJO 的关键功能区域为中心的模型与 Elastichsearch 交互文档和轻松地编写一个存储索引库数据访问层。

> 官方网站: https://spring.io/projects/spring-data-elasticsearch



2. Spring Boot整合Spring Data Elasticsearch

1）版本选型

Elasticsearch 8.14.x 对应依赖 Spring Data Elasticsearch 5.3.x，对应Spring6.1.x，Spring Boot版本可以选择3.3.x

2）引入依赖

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
</dependency>
```

如果Spring Boot版本选择3.3.2，对应的Spring Data Elasticsearch为5.3.2

3）配置ElasticSearch

Spring Boot中有两种配置ElasticSearch的方式，选择一种即可。

方式1：yml配置

```yaml
spring:
  elasticsearch:
    uris: http://localhost:9200
    connection-timeout: 3s
```

方式2： @Configuration配置

```java
@Configuration
public class MyESClientConfig extends ElasticsearchConfiguration {

    @Override
    public ClientConfiguration clientConfiguration() {
        return ClientConfiguration.builder().connectedTo("localhost:9200").build();
    }
}
```

4）Java代码实现

方式1：使用*ElasticsearchRepository*

ElasticsearchRepository 是Spring Data Elasticsearch项目中的一个接口，用于简化对Elasticsearch集群的CRUD操作以及其他高级搜索功能的集成。这个接口允许开发者通过声明式编程模型来执行数据持久化操作，从而避免直接编写复杂的REST API调用代码。

创建实体类

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Document(indexName = "employees")
public class Employee {
    @Id
    private Long id;
    @Field(type= FieldType.Keyword)
    private String name;
    private int sex;
    private int age;
    @Field(type= FieldType.Text,analyzer="ik_max_word")
    private String address;
    private String remark;
}
```

实现ElasticsearchRepository接口

该接口是框架封装的用于操作Elastsearch的高级接口

```
@Repository
public interface EmployeeRepository extends ElasticsearchRepository<Employee, Long> {
    List<Employee> findByName(String name);
}
```

测试

```java
@Autowired
EmployeeRepository employeeRepository;

@Test
public void testDocument() {

    Employee employee = new Employee(10L, "fox666", 1, 32, "长沙麓谷", "java architect");
    //插入文档
    employeeRepository.save(employee);

    //根据id查询
    Optional<Employee> result = employeeRepository.findById(10L);
    if (!result.isEmpty()){
        log.info(String.valueOf(result.get()));
    }


    //根据name查询
    List<Employee> list = employeeRepository.findByName("fox666");
    if(!list.isEmpty()){
        log.info(String.valueOf(list.get(0)));
    }

}
```

更多实现参考官方文档：https://docs.spring.io/spring-data/elasticsearch/reference/elasticsearch/repositories/elasticsearch-repository-queries.html

方式2：使用ElasticsearchTemplate

ElasticsearchTemplate模板类，封装了便捷操作Elasticsearch的模板方法，包括 索引 / 映射 / 文档 CRUD 等底层操作和高级操作。

```java
@Autowired
ElasticsearchTemplate elasticsearchTemplate;
```

> 从 Java Rest Client 7.15.0 版本开始，Elasticsearch 官方决定将 RestHighLevelClient 标记为废弃的，并推荐使用新的 Java API Client，即 ElasticsearchClient. Spring Data ElasticSearch对ElasticsearchClient做了进一步的封装，成了新的客户端 ElasticsearchTemplate

测试

```java
@Slf4j
public class ElasticsearchClientTest extends VipEsDemoApplicationTests{

    @Autowired
    ElasticsearchTemplate elasticsearchTemplate;


    @Test
    public void testCreateIndex(){

        //索引是否存在
        boolean exist = elasticsearchTemplate.indexOps(Employee.class).exists();
        if(exist){
            //删除索引
            elasticsearchTemplate.indexOps(Employee.class).delete();
        }
        //创建索引
        //1）配置settings
        Map<String, Object> settings = new HashMap<>();
        //"number_of_shards": 1,
        //"number_of_replicas": 1
        settings.put("number_of_shards",1);
        settings.put("number_of_replicas",1);
        //2) 配置mapping
        String json = "{\n" +
                "      \"properties\": {\n" +
                "        \"_class\": {\n" +
                "          \"type\": \"text\",\n" +
                "          \"fields\": {\n" +
                "            \"keyword\": {\n" +
                "              \"type\": \"keyword\",\n" +
                "              \"ignore_above\": 256\n" +
                "            }\n" +
                "          }\n" +
                "        },\n" +
                "        \"address\": {\n" +
                "          \"type\": \"text\",\n" +
                "          \"fields\": {\n" +
                "            \"keyword\": {\n" +
```

```
                "              \"type\": \"keyword\"\n" +
                "            }\n" +
                "          },\n" +
                "          \"analyzer\": \"ik_max_word\"\n" +
                "        },\n" +
                "        \"age\": {\n" +
                "          \"type\": \"integer\"\n" +
                "        },\n" +
                "        \"id\": {\n" +
                "          \"type\": \"long\"\n" +
                "        },\n" +
                "        \"name\": {\n" +
                "          \"type\": \"keyword\"\n" +
                "        },\n" +
                "        \"remark\": {\n" +
                "          \"type\": \"text\",\n" +
                "          \"fields\": {\n" +
                "            \"keyword\": {\n" +
                "              \"type\": \"keyword\"\n" +
                "            }\n" +
                "          },\n" +
                "          \"analyzer\": \"ik_smart\"\n" +
                "        },\n" +
                "        \"sex\": {\n" +
                "          \"type\": \"integer\"\n" +
                "        }\n" +
                "      }\n" +
                "    }";
        Document mapping = Document.parse(json);
        //3)创建索引
        elasticsearchTemplate.indexOps(Employee.class)
                .create(settings,mapping);

        //查看索引mappings信息
        Map<String, Object> mappings =
elasticsearchTemplate.indexOps(Employee.class).getMapping();
        log.info(mappings.toString());


    }

```

```
80
81      @Test
82      public void testBulkBatchInsert(){
83          List<Employee> employees = new ArrayList<>();
84          employees.add(new Employee(2L,"张三",1,25,"广州天河公园","java developer"));
85          employees.add(new Employee(3L,"李四",1,28,"广州荔湾大厦","java assistant"));
86          employees.add(new Employee(4L,"小红",0,26,"广州白云山公园","php developer"));
87
88          List<IndexQuery> bulkInsert = new ArrayList<>();
89          for (Employee employee : employees) {
90              IndexQuery indexQuery = new IndexQuery();
91              indexQuery.setId(String.valueOf(employee.getId()));
92              String json = JSONObject.toJSONString(employee);
93              indexQuery.setSource(json);
94              bulkInsert.add(indexQuery);
95          }
96          //bulk批量插入文档
97          elasticsearchTemplate.bulkIndex(bulkInsert,Employee.class);
98      }
99
100
101     @Test
102     public void testDocument(){
103
104         //根据id删除文档
105         //对应:  DELETE /employee/_doc/12
106         elasticsearchTemplate.delete(String.valueOf(12L),Employee.class);
107
108         Employee employee = new Employee(12L,"张三三",1,25,"广州天河公园","java
    developer");
109         //插入文档
110         elasticsearchTemplate.save(employee);
111
112         //根据id查询文档
113         //对应: GET /employee/_doc/12
114         Employee emp = elasticsearchTemplate.get(String.valueOf(12L),Employee.class);
115         log.info(String.valueOf(emp));
116
117     }
118
119
```

```
120
121     @Test
122     public void testQueryDocument(){
123         //条件查询
124         /* 查询姓名为张三的员工信息
125         GET /employee/_search
126         {
127             "query": {
128             "term": {
129                 "name": {
130                     "value": "张三"
131                 }
132             }
133         }
134         }*/
135
136         //第一步：构建查询语句
137         //方式1：StringQuery
138 //        Query query = new StringQuery("{\n" +
139 //                "            \"term\": {\n" +
140 //                "                \"name\": {\n" +
141 //                "                    \"value\": \"张三\"\n" +
142 //                "                }\n" +
143 //                "            }\n" +
144 //                "        }");
145         //方式2：NativeQuery
146         Query query = NativeQuery.builder()
147                 .withQuery(q -> q.term(
148                         t -> t.field("name").value("张三")))
149                 .build();
150
151
152         //第二步：调用search查询
153         SearchHits<Employee> search = elasticsearchTemplate.search(query,
    Employee.class);
154         //第三步：解析返回结果
155         List<SearchHit<Employee>> searchHits = search.getSearchHits();
156         for (SearchHit hit: searchHits){
157             log.info("返回结果："+hit.toString());
158         }
159
```

```
160
161     }
162
163
164     @Test
165     public void testMatchQueryDocument(){
166         //条件查询
167         /*最少匹配广州，公园两个词
168         GET /employee/_search
169         {
170             "query": {
171             "match": {
172                 "address": {
173                     "query": "广州公园",
174                      "minimum_should_match": 2
175                 }
176             }
177         }
178         }*/
179
180         //第一步：构建查询语句
181         //方式1：StringQuery
182 //        Query query = new StringQuery("{\n" +
183 //                "            \"match\": {\n" +
184 //                "                \"address\": {\n" +
185 //                "                    \"query\": \"广州公园\",\n" +
186 //                "                     \"minimum_should_match\": 2\n" +
187 //                "                }\n" +
188 //                "            }\n" +
189 //                "        }");
190         //方式2：NativeQuery
191         Query query = NativeQuery.builder()
192                 .withQuery(q -> q.match(
193                         m -> m.field("address").query("广州公园")
194                                 .minimumShouldMatch("2")))
195                 .build();
196
197
198         //第二步：调用search查询
199         SearchHits<Employee> search = elasticsearchTemplate.search(query,
    Employee.class);
```

```
        //第三步：解析返回结果
        List<SearchHit<Employee>> searchHits = search.getSearchHits();
        for (SearchHit hit: searchHits){
            log.info("返回结果："+hit.toString());
        }

    }

    @Test
    public void testQueryDocument3(){
        // 分页排序高亮
        /*
        GET /employee/_search
        {
          "from": 0,
          "size": 3,
          "query": {
            "match": {
              "remark": {
                "query": "JAVA"
              }
            }
          },
          "highlight": {
            "pre_tags": ["<font color='red'>"],
            "post_tags": ["<font/>"],
            "require_field_match": "false",
            "fields": {
              "*":{}
            }
          },
          "sort": [
            {
              "age": {
                "order": "desc"
              }
            }
          ]
        }*/
        //第一步：构建查询语句
```

```
        Query query = new StringQuery("{\n" +
                "        \"match\": {\n" +
                "          \"remark\": {\n" +
                "            \"query\": \"JAVA\"\n" +
                "          }\n" +
                "        }\n" +
                "      }");
        //分页  注意：from = pageNumber（页码，从0开始，） * pageSize（每页的记录数）
        query.setPageable(PageRequest.of(0, 3));
        //排序
        query.addSort(Sort.by(Order.desc("age")));
        //高亮
        HighlightField highlightField = new HighlightField("*");
        HighlightParameters highlightParameters = new
HighlightParameters.HighlightParametersBuilder()
                .withPreTags("<font color='red'>")
                .withPostTags("<font/>")
                .withRequireFieldMatch(false)
                .build();
        Highlight highlight = new
Highlight(highlightParameters,Arrays.asList(highlightField));
        HighlightQuery highlightQuery = new HighlightQuery(highlight,Employee.class);

        query.setHighlightQuery(highlightQuery);


        //第二步：调用search查询
        SearchHits<Employee> search = elasticsearchTemplate.search(query,
Employee.class);
        //第三步：解析返回结果
        List<SearchHit<Employee>> searchHits = search.getSearchHits();
        for (SearchHit hit: searchHits){
            log.info("返回结果："+hit.toString());
        }
    }


    @Test
    public void testBoolQueryDocument(){
        //条件查询
        /*
```

```
278         GET /employee/_search
279         {
280           "query": {
281             "bool": {
282               "must": [
283                 {
284                   "match": {
285                     "address": "广州"
286                   }
287                 },{
288                   "match": {
289                     "remark": "java"
290                   }
291                 }
292               ]
293             }
294           }
295         }
296          */
297
298         //第一步：构建查询语句
299         //方式1：StringQuery
300 //        Query query = new StringQuery("{\n" +
301 //                "            \"bool\": {\n" +
302 //                "              \"must\": [\n" +
303 //                "                {\n" +
304 //                "                  \"match\": {\n" +
305 //                "                    \"address\": \"广州\"\n" +
306 //                "                  }\n" +
307 //                "                },{\n" +
308 //                "                  \"match\": {\n" +
309 //                "                    \"remark\": \"java\"\n" +
310 //                "                  }\n" +
311 //                "                }\n" +
312 //                "              ]\n" +
313 //                "            }\n" +
314 //                "          }");
315         //方式2：NativeQuery
316         Query query = NativeQuery.builder()
317                 .withQuery(q -> q.bool(
```

```
                        m -> m.must(
                          QueryBuilders.match( q1 -> q1.field("address").query("广州")),
                          QueryBuilders.match( q2 -> q2.field("remark").query("java"))
                        )))
                .build();

        //第二步：调用search查询
        SearchHits<Employee> search = elasticsearchTemplate.search(query, Employee.class);
        //第三步：解析返回结果
        List<SearchHit<Employee>> searchHits = search.getSearchHits();
        for (SearchHit hit: searchHits){
            log.info("返回结果："+hit.toString());
        }

    }

}
```

方式3：使用ElasticsearchClient

从 Java Rest Client 7.15.0 版本开始，Elasticsearch 官方决定将 RestHighLevelClient 标记为废弃的，并推荐使用新的 Java API Client，即 ElasticsearchClient.

官网文档：https://www.elastic.co/guide/en/elasticsearch/client/java-api-client/8.14/getting-started-java.html

测试

```java
@Autowired
ElasticsearchClient elasticsearchClient;

String indexName = "employee_demo";

@Test
public void testCreateIndex() throws IOException {

    //索引是否存在
    BooleanResponse exist = elasticsearchClient.indices()
            .exists(e->e.index(indexName));
    if(exist.value()){
        //删除索引
        elasticsearchClient.indices().delete(d->d.index(indexName));
    }
    //创建索引
    elasticsearchClient.indices().create(c->c.index(indexName)
            .settings(s->s.numberOfShards("1").numberOfReplicas("1"))
            .mappings(m-> m.properties("name",p->p.keyword(k->k))
                    .properties("sex",p->p.long_(l->l))
                    .properties("address",p->p.text(t->t.analyzer("ik_max_word")))
            )
    );

    //查询索引
    GetIndexResponse getIndexResponse = elasticsearchClient.indices().get(g ->
g.index(indexName));
    log.info(getIndexResponse.result().toString());

}


@Test
public void testBulkBatchInsert() throws IOException {
    List<Employee> employees = new ArrayList<>();
    employees.add(new Employee(2L,"张三",1,25,"广州天河公园","java developer"));
    employees.add(new Employee(3L,"李四",1,28,"广州荔湾大厦","java assistant"));
    employees.add(new Employee(4L,"小红",0,26,"广州白云山公园","php developer"));

    List<IndexQuery> bulkInsert = new ArrayList<>();
```

```
    for (Employee employee : employees) {
        IndexQuery indexQuery = new IndexQuery();
        indexQuery.setId(String.valueOf(employee.getId()));
        String json = JSONObject.toJSONString(employee);
        indexQuery.setSource(json);
        bulkInsert.add(indexQuery);
    }
    List<BulkOperation> list = new ArrayList<>();
    for (Employee employee : employees) {
        BulkOperation bulkOperation = new BulkOperation.Builder()
                .create(c->c.id(String.valueOf(employee.getId()))
                        .document(employee)
                )
                .build();

        list.add(bulkOperation);
    }

    //bulk批量插入文档
    elasticsearchClient.bulk(b->b.index(indexName).operations(list));
}

@Test
public void testDocument() throws IOException {
    Employee employee = new Employee(12L,"张三三",1,25,"广州天河公园","java developer");

    IndexRequest<Employee> request = IndexRequest.of(i -> i
            .index(indexName)
            .id(employee.getId().toString())
            .document(employee)
    );

    IndexResponse response = elasticsearchClient.index(request);

    log.info("response:"+response);
}


@Test
public void testQuery() throws IOException {
```

```
    SearchRequest searchRequest = SearchRequest.of(s -> s
            .index(indexName)
            .query(q -> q.match(m -> m.field("name").query("张三三"))
     ));

    log.info("构建的DSL语句:"+ searchRequest.toString());

    SearchResponse<Employee> searchResponse = elasticsearchClient.search(searchRequest, Employee.class);

    List<Hit<Employee>> hits = searchResponse.hits().hits();
    hits.stream().map(Hit::source).forEach(employee -> {
        log.info("员工信息:"+employee);
    });

}

@Test
public void testBoolQueryDocument() throws IOException {
    //条件查询
    /*
    GET /employee/_search
    {
      "query": {
        "bool": {
          "must": [
            {
              "match": {
                "address": "广州"
              }
            },{
              "match": {
                "remark": "java"
              }
            }
          ]
        }
      }
     */

```

```
    //第一步：构建查询语句
    BoolQuery.Builder boolQueryBuilder = new BoolQuery.Builder();
    boolQueryBuilder.must(m->m.match(q->q.field("address").query("广州")))
            .must(m->m.match(q->q.field("remark").query("java")));

    SearchRequest searchRequest = new SearchRequest.Builder()
            .index("employee")
            .query(q->q.bool(boolQueryBuilder.build()))
            .build();

    //第二步：调用search查询
    SearchResponse<Employee> searchResponse = elasticsearchClient.search(searchRequest,
Employee.class);
    //第三步：解析返回结果
    List<Hit<Employee>> list = searchResponse.hits().hits();
    for(Hit<Employee> hit: list){
        //返回source
        log.info(String.valueOf(hit.source()));
    }

}
```