ElasticSearch -5

1. ES &text

text

1 field keyword

```
POST /index_customer/_mapping
{
    "properties": {
        "nickname": {
            "analyzer": "ik_max_word",
            "type": "text",
            "fields": {
                "pinyin": {
                    "analyzer": "pinyin",
                    "type": "text"
                },
                "keyword": {
                    "ignore_above": 256,
                    "type": "keyword"
                }
            }
        }
    }
}
#         index
POST /index_customer/_update_by_query
```

2 field fielddata

```
# fielddata text
#   false
POST /index_customer/_mapping
{
    "properties": {
        "nickname": {
            "analyzer": "ik_max_word",
            "type": "text",
            "fielddata": true,
            "fields": {
                "pinyin": {
                    "analyzer": "pinyin",
                    "type": "text",
                    "fielddata": true
                },
                "keyword": {
```

```
                    "ignore_above": 256,
                    "type": "keyword"
                }
            }
        }
    }
}
```

- doc_count_error_upper_bound
- sum_other_doc_count
  - 10
  - size
  -
- buckets

size

```
POST /index_customer/_search
{
    "query": {
        "match_all": {}
    },
    "size": 0,
    "aggs": {
        "nickname_term": {
            "terms": {
                "field": "nickname",
                "size": 20
            }
        }
    }
}
```

doc field

```
#         doc
# "eager_global_ordinals": true
POST /index_customer/_mapping
{
    "properties": {
        "nickname": {
            "analyzer": "ik_max_word",
            "type": "text",
            "fielddata": true,
            "eager_global_ordinals": true,
```

```
            "fields": {
                "pinyin": {
                    "analyzer": "pinyin",
                    "type": "text",
                    "fielddata": true,
                    "eager_global_ordinals": true
                },
                "keyword": {
                    "ignore_above": 256,
                    "type": "keyword"
                }
            }
        }
    }
}


# cardinality
POST /index_customer/_search
{
    "query": {
        "match_all": {}
    },
    "size": 0,
    "aggs": {
        "nickname_term": {
            "cardinality": {
                "field": "nickname"
            }
        }
    }
}
```

2. range

```
POST POST /index_customer/_search
{
    "query": {
        "match_all": {}
    },
    "size": 0,
    "sort": [
        {
            "consume": "desc"
```

```
        }
    ],
    "aggs": {
        "city_count": {
            "terms": {
                "field": "city"
            }
        },
        "consume_range": {
            "range": {
                "field": "consume",
                "ranges": [
                    {
                        "to": 3000
                    },
                    {
                        "from": 3000,
                        "to": 6000
                    },
                    {
                        "from": 6000,
                        "to": 9000
                    },
                    {
                        "from": 9000
                    }
                ]
            }
        }
    }
}


POST /index_customer/_search
{
    "query": {
        "match_all": {}
    },
    "size": 0,
    "sort": [
        {
            "consume": "desc"
        }
    ],
    "aggs": {
        "city_count": {
```

```
            "terms": {
                "field": "city"
            }
        },
        "consume_histogram": {
            "histogram": {
                "field": "consume",
                "interval": 2000,
                "extended_bounds": {
                    "min": 0,
                    "max": 20000
                }
            }
        }
    }
}
```

3. Pipeline

pipeline

```
#
GET /index_customer/_search
{
    "query": {
        "match_all": {}
    },
    "size": 0,
    "sort": [
        {
            "consume": "desc"
        }
    ],
    "aggs": {
        "city_count": {
            "terms": {
                "field": "city"
            },
            "aggs": {
                "avg_consume": {
                    "avg": {
                        "field": "consume"
                    }
                }
            }
        },
```

```
        "min_consume_by_city": {
            "min_bucket": {
                "buckets_path": "city_count>avg_consume"
            }
        }
    }
}
# min_bucket / buckets_path
# max_bucket / min_bucket / avg_bucket / sum_bucket / stats_bucket
```

4. Springboot Elasticsearch

POM

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

yaml

```
#    9300
spring:
  data:
    elasticsearch:
      cluster-name: icoding-es
      cluster-nodes: 47.92.163.109:9300
```

po

```
package com.icodingedu.po;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.elasticsearch.annotations.Document;
import org.springframework.data.elasticsearch.annotations.Field;

//indexName
//type
@Data
@Document(indexName = "index_user",type = "_doc",shards = 3,replicas = 1)
public class UserBo {
    //index doc id  id
    @Id
    private String id;

    //
    @Field(store = true,index = true,analyzer = "ik_max_word",searchAnalyzer = "ik_max_word'
```

```
    private String nickname;

    @Field(store = true)
    private Integer sex;

    @Field(store = true)
    private Double consume;

    @Field(store = true,index = true,analyzer = "ik_max_word",searchAnalyzer = "ik_max_word"
    private String review;
}
```

controller

```
package com.icodingedu.controller;

import com.icodingedu.po.UserBo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.elasticsearch.core.ElasticsearchTemplate;
import org.springframework.data.elasticsearch.core.query.IndexQuery;
import org.springframework.data.elasticsearch.core.query.IndexQueryBuilder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class ESUserController {

    @Autowired
    ElasticsearchTemplate elasticsearchTemplate;

    @GetMapping("/create_index")
    @ResponseBody
    public String createIndex(){
        UserBo userBo = new UserBo();
        userBo.setId("1001");
        userBo.setConsume(1899.66);
        userBo.setNickname("  ");
        userBo.setReview("icoding edu          ");
        userBo.setSex(1);

        IndexQuery indexQuery = new IndexQueryBuilder()
                .withObject(userBo)
                .build();
        elasticsearchTemplate.index(indexQuery);
        return "index/mapping/document    ";
    }
```

```
}
```

mapping

```
//   po
//
//    elasticsearchTemplate  po
// elasticsearchTemplate.index(indexQuery)    index
```

index

```
@GetMapping("/delete-index")
@ResponseBody
public String deleteIndex(){
  elasticsearchTemplate.deleteIndex(UserBo.class);
  return "   ";
}
```

ElasticsearchTemplate

- index mapping json
- ET

5. Springboot ES

document

```
    @GetMapping("/update")
    @ResponseBody
    public String updateIndex(){

        Map<String,Object> data = new HashMap<String,Object>();
        data.put("username","jackwang");
        data.put("consume",7888.99);

        IndexRequest indexRequest = new IndexRequest();
        indexRequest.source(data);

        UpdateQuery updateQuery = new UpdateQueryBuilder()
                .withClass(UserBo.class)
                .withId("1001")
                .withIndexRequest(indexRequest)
                .build();

        elasticsearchTemplate.update(updateQuery);
        return "   ";
    }
```

document

```
    @GetMapping("/delete/{id}")
    @ResponseBody
    public String deleteDocument(@PathVariable("id") String uid){
        elasticsearchTemplate.delete(UserBo.class,uid);
        return " id:"+uid;
    }
```

id doc

```
    @GetMapping("/get/{id}")
    @ResponseBody
    public String getIndex(@PathVariable("id") String uid){

        GetQuery query = new GetQuery();
        query.setId(uid);

        UserBo userBo = elasticsearchTemplate.queryForObject(query,UserBo.class);
        return userBo.toString();
    }
```

6. Springboot ES

```
// ES  index
package com.icodingedu.po;

import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.elasticsearch.annotations.Document;
import org.springframework.data.elasticsearch.annotations.Field;

@Data
@Document(indexName = "index_customer",type = "_doc")
public class CustomerPo {
    @Id
    private String id;

    @Field(store=true)
    private Integer age;

    @Field(store=true)
    private String username;

    @Field(store=true)
    private String nickname;
    @Field(store=true)
    private Float consume;
```

```
    @Field(store=true)
    private String desc;

    @Field(store=true)
    private Integer sex;

    @Field(store=true)
    private String birthday;

    @Field(store=true)
    private String city;

    @Field(store=true)
    private String faceimg;
}
```

controller

```
    @GetMapping("/list")
    @ResponseBody
    public String getList(){
        //3.
        Pageable pageable = PageRequest.of(0,2);
        //2. query
        SearchQuery query = new NativeSearchQueryBuilder()
                .withQuery(QueryBuilders.matchQuery("desc","    "))
                .withPageable(pageable)
                .build();
        //1.
        AggregatedPage<CustomerPo> customerPos = elasticsearchTemplate.queryForPage(query,Cu
        System.out.println("  "+customerPos.getTotalPages());
        System.out.println("   "+customerPos.getTotalElements());
        List<CustomerPo> customerPoList = customerPos.getContent();
        for (CustomerPo customerPo:customerPoList) {
            System.out.println(customerPo.toString());

        }
        return "  ";
    }
```

7. Springboot ES

```
//
    @GetMapping("/listhiglight")
    @ResponseBody
    public String getListHighLight(){
        //4.
```

```
    String preTag = "<font color='red'>";
    String postTag = "</font>";
    //3.
    Pageable pageable = PageRequest.of(0,2);
    //2. query
    SearchQuery query = new NativeSearchQueryBuilder()
            .withQuery(QueryBuilders.matchQuery("desc","    "))
            .withHighlightFields(new HighlightBuilder.Field("desc").preTags(preTag).post
            .withPageable(pageable)
            .build();
    //1.  ,
    AggregatedPage<CustomerPo> customerPos = elasticsearchTemplate.queryForPage(query, C
        @Override
        public <T> AggregatedPage<T> mapResults(SearchResponse searchResponse, Class<T>
            return null;
        }

        @Override
        public <T> T mapSearchHit(SearchHit searchHit, Class<T> aClass) {
            return null;
        }
    });
    System.out.println("  "+customerPos.getTotalPages());
    System.out.println("   "+customerPos.getTotalElements());
    List<CustomerPo> customerPoList = customerPos.getContent();
    for (CustomerPo customerPo:customerPoList) {
        System.out.println(customerPo.toString());

    }
    return "  ";
}


@GetMapping("/listhiglight")
@ResponseBody
public String getListHighLight(){
    //4.
    String preTag = "<font color='red'>";
    String postTag = "</font>";
    //3.
    Pageable pageable = PageRequest.of(0,2);
    //2. query
    SearchQuery query = new NativeSearchQueryBuilder()
            .withQuery(QueryBuilders.matchQuery("desc","    "))
            .withHighlightFields(new HighlightBuilder.Field("desc").preTags(preTag).post
            .withPageable(pageable)
```

```
            .build();
    //1.  ,
    AggregatedPage<CustomerPo> customerPos = elasticsearchTemplate.queryForPage(query, C
        @Override
        public <T> AggregatedPage<T> mapResults(SearchResponse searchResponse, Class<T>
            List<CustomerPo> customerPoList = new ArrayList<CustomerPo>();
            SearchHits searchHits = searchResponse.getHits();
            for (SearchHit h: searchHits) {
                HighlightField highlightField = h.getHighlightFields().get("desc");
                String desc = highlightField.fragments()[0].toString();
                CustomerPo customerPoHighlight = new CustomerPo();
                customerPoHighlight.setAge((Integer)h.getSourceAsMap().get("age"));
                customerPoHighlight.setBirthday(h.getSourceAsMap().get("birthday").toStr
                customerPoHighlight.setCity(h.getSourceAsMap().get("city").toString());
                customerPoHighlight.setConsume(Float.valueOf(h.getSourceAsMap().get("cor
                customerPoHighlight.setDesc(desc);//
                customerPoHighlight.setFaceimg(h.getSourceAsMap().get("faceimg").toStrir
                customerPoHighlight.setId(h.getSourceAsMap().get("id").toString());
                customerPoHighlight.setNickname(h.getSourceAsMap().get("nickname").toStr
                customerPoHighlight.setSex((Integer)h.getSourceAsMap().get("sex"));
                customerPoHighlight.setUsername(h.getSourceAsMap().get("username").toStr
                customerPoList.add(customerPoHighlight);
            }
            if(customerPoList.size()>0){
                return new AggregatedPageImpl<>((List<T>) customerPoList);
            }
            return null;
        }

        @Override
        public <T> T mapSearchHit(SearchHit searchHit, Class<T> aClass) {
            return null;
        }
    });
    System.out.println("   "+customerPos.getTotalPages());
    System.out.println("    "+customerPos.getTotalElements());
    List<CustomerPo> customerPoList = customerPos.getContent();
    for (CustomerPo customerPo:customerPoList) {
        System.out.println(customerPo.toString());

    }
    return "   ";
}
```

8. Springboot ES

ok

```
@GetMapping("/list")
@ResponseBody
public String getList(){
    //4.
    SortBuilder sortBuilder1 = new FieldSortBuilder("consume")
            .order(SortOrder.DESC);
    SortBuilder sortBuilder2 = new FieldSortBuilder("age")
            .order(SortOrder.ASC);

    //3.
    Pageable pageable = PageRequest.of(0,6);
    //2. query
    SearchQuery query = new NativeSearchQueryBuilder()
            .withQuery(QueryBuilders.matchQuery("desc"," "))
            .withPageable(pageable)
            .withSort(sortBuilder1)
            .withSort(sortBuilder2)
            .build();
    //1.
    AggregatedPage<CustomerPo> customerPos = elasticsearchTemplate.queryForPage(query,Cu
    System.out.println("  "+customerPos.getTotalPages());
    System.out.println("   "+customerPos.getTotalElements());
    List<CustomerPo> customerPoList = customerPos.getContent();
    for (CustomerPo customerPo:customerPoList) {
        System.out.println(customerPo.toString());

    }
    return "  ";
}
```