Java第四阶段\_day01\_ElasticSearch答案

1. 简述题：简述什么是ES，应用场景有哪些。

是一个分布式的restful风格的搜索引擎，主要用在当数据量庞大项目例如

全文搜索引擎：电商网站，检索商品。

数据分析：日志数据分析，logstash采集日志，ElasticSearch进行复杂的数据分析

1. 编程题：Spring Data Elasticsearch完成对索引增删改查。

package com.bigdata.es.demo;

import com.bigdata.EsApplication;

import org.elasticsearch.search.aggregations.\*;

import com.bigdata.es.pojo.Item;

import com.bigdata.es.repository.ItemRepository;

import org.elasticsearch.index.query.QueryBuilders;

import org.elasticsearch.search.aggregations.bucket.terms.StringTerms;

import org.elasticsearch.search.sort.SortBuilders;

import org.elasticsearch.search.sort.SortOrder;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.data.elasticsearch.core.ElasticsearchTemplate;

import org.springframework.data.elasticsearch.core.aggregation.AggregatedPage;

import org.springframework.data.elasticsearch.core.query.FetchSourceFilter;

import org.springframework.data.elasticsearch.core.query.NativeSearchQueryBuilder;

import org.springframework.test.context.junit4.SpringRunner;

import java.util.ArrayList;

import java.util.List;

@RunWith(SpringRunner.class)//让测试在Spring容器环境下执行

@SpringBootTest(classes = EsApplication.class,webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)//配置文件属性的读取

public class EsTest {

@Autowired

public ElasticsearchTemplate template;

@Autowired

public ItemRepository itemRepository;

@Test

public void testCreate(){

// 创建索引，会根据Item类的@Document注解信息来创建

template.createIndex(Item.class);

// 配置映射，会根据Item类中的id、Field等字段来自动完成映射

template.putMapping(Item.class);

}

@Test

public void deleteIndex() {

//template.deleteIndex(Item.class);

template.deleteIndex("big");

}

@Test

public void index() {

Item item = new Item(1L, "小米手机7", " 手机",

"小米", 3499.00, "http://image.bigdata.com/13123.jpg");

itemRepository.save(item);

}

@Test

public void indexList() {

List<Item> list = new ArrayList<>();

//list.add(new Item(1L, "小米手机7", "手机", "小米", 3299.00, "http://image.bigdata.com/13123.jpg"));

list.add(new Item(2L, "坚果手机R1", "手机", "锤子", 3699.00, "http://image.bigdata.com/13123.jpg"));

list.add(new Item(3L, "华为META10", "手机", "华为", 4499.00, "http://image.bigdata.com/13123.jpg"));

list.add(new Item(4L, "小米Mix2S", "手机", "小米", 4299.00, "http://image.bigdata.com/13123.jpg"));

list.add(new Item(5L, "荣耀V10", "手机", "华为", 2799.00, "http://image.bigdata.com/13123.jpg"));

// 接收对象集合，实现批量新增

itemRepository.saveAll(list);

}

@Test

public void delete(){

itemRepository.deleteById(2L);

}

@Test

public void testFind(){

Iterable<Item> all=itemRepository.findAll();

for (Item item:all){

System.out.println("item="+item);

}

}

@Test

public void query(){

// 查询全部，并按照价格降序排序

Iterable<Item> items = this.itemRepository.findAll(Sort.by("price").descending());

for (Item item : items) {

System.out.println("item = " + item);

}

}

@Test

public void testFindBy(){

List<Item> items=this.itemRepository.findByPriceBetween(2000d,4000d);

for (Item item : items) {

System.out.println("item = " + item);

}

}

@Test

public void search(){

// 构建查询条件

NativeSearchQueryBuilder queryBuilder = new NativeSearchQueryBuilder();

// 添加基本分词查询

queryBuilder.withQuery(QueryBuilders.matchQuery("title", "小米手机"));

// 搜索，获取结果

Page<Item> items = this.itemRepository.search(queryBuilder.build());

// 总条数

long total = items.getTotalElements();

System.out.println("total = " + total);

for (Item item : items) {

System.out.println(item);

}

}

@Test

public void searchByPage(){

// 构建查询条件

NativeSearchQueryBuilder queryBuilder = new NativeSearchQueryBuilder();

// 添加基本分词查询

queryBuilder.withQuery(QueryBuilders.termQuery("category", "手机"));

// 分页：从0开始

int page = 0;

int size = 2;

queryBuilder.withPageable(PageRequest.of(page,size));

// 搜索，获取结果

Page<Item> items = this.itemRepository.search(queryBuilder.build());

// 总条数

System.out.println("总条数 = " + items.getTotalElements());

// 总页数

System.out.println("总页数 = " + items.getTotalPages());

// 当前页

System.out.println("当前页：" + items.getNumber());

// 每页大小

System.out.println("每页大小：" + items.getSize());

for (Item item : items) {

System.out.println(item);

}

}

@Test

public void searchAndSort(){

// 构建查询条件

NativeSearchQueryBuilder queryBuilder = new NativeSearchQueryBuilder();

// 添加基本分词查询

queryBuilder.withQuery(QueryBuilders.termQuery("category", "手机"));

// 排序

queryBuilder.withSort(SortBuilders.fieldSort("price").order(SortOrder.ASC));

// 搜索，获取结果

Page<Item> items = this.itemRepository.search(queryBuilder.build());

// 总条数

long total = items.getTotalElements();

System.out.println("总条数 = " + total);

for (Item item : items) {

System.out.println(item);

}

}

}

1. 面试题：ES和Lucene区别。

Elasticsearch基于Lucene构建，Elasticsearch利用Lucene做实际的工作

ELasticsearch中的每个分片都是一个分离的Lucene实例.

Elasticsearch在Lucene基础上（即利用Lucene的功能）提供了一个分布式的、基于JSON的REST API 来更方便地使用 Lucene的功能。