mall整合Elasticsearch实现商品搜索

原创 梦想de星空 macrozheng 2019-05-20 08:31

收录于合集
#mall学习教程（架构篇） 10个

> 本文主要讲解mall整合Elasticsearch的过程，以实现商品信息在Elasticsearch中的导入、查询、修改、删除为例。

项目使用框架介绍

Elasticsearch

> Elasticsearch 是一个分布式、可扩展、实时的搜索与数据分析引擎。 它能从项目一开始就赋予你的数据以搜索、分析和探索的能力，可用于实现全文搜索和实时数据统计。

Elasticsearch的安装和使用

1.下载Elasticsearch6.2.2的zip包，并解压到指定目录，下载地址：
https://www.elastic.co/cn/downloads/past-releases/elasticsearch-6-2-2

2.安装中文分词插件，在elasticsearch-6.2.2\bin目录下执行以下命令：elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v6.2.2/elasticsearch-analysis-ik-6.2.2.zip

3.运行bin目录下的elasticsearch.bat启动Elasticsearch

4.下载Kibana,作为访问Elasticsearch的客户端，请下载6.2.2版本的zip包，并解压到指定目录，下载地址：https://artifacts.elastic.co/downloads/kibana/kibana-6.2.2-windows-x86_64.zip

5.运行bin目录下的kibana.bat，启动Kibana的用户界面

6.访问http://localhost:5601 即可打开Kibana的用户界面

Spring Data Elasticsearch

> Spring Data Elasticsearch是Spring提供的一种以Spring Data风格来操作数据存储的方式，它可以避免编写大量的样板代码。

常用注解

@Document

```
//标示映射到Elasticsearch文档上的领域对象
public @interface Document {
  //索引库名次，mysql中数据库的概念
    String indexName();
  //文档类型，mysql中表的概念
    String type() default "";
  //默认分片数
    short shards() default 5;
  //默认副本数量
    short replicas() default 1;

}
```

@Id

```
//表示是文档的id，文档可以认为是mysql中表行的概念
public @interface Id {
}
```

@Field

```
public @interface Field {
  //文档中字段的类型
    FieldType type() default FieldType.Auto;
  //是否建立倒排索引
    boolean index() default true;
  //是否进行存储
    boolean store() default false;
  //分词器名次
    String analyzer() default "";
}
```

```
//为文档自动指定元数据类型
public enum FieldType {
    Text,//会进行分词并建了索引的字符类型
    Integer,
    Long,
    Date,
    Float,
    Double,
```

```
    Boolean,
    Object,
    Auto,//自动判断字段类型
    Nested,//嵌套对象类型
    Ip,
    Attachment,
    Keyword//不会进行分词建立索引的类型
}
```

Sping Data方式的数据操作

继承ElasticsearchRepository接口可以获得常用的数据操作方法

可以使用衍生查询

> 在接口中直接指定查询方法名称便可查询，无需进行实现，如商品表中有商品名称、标题和关键字，直接定义以下查询，就可以对这三个字段进行全文搜索。

```
    /**
     * 搜索查询
```

```
    * 搜索查询
    *
    * @param name              商品名称
    * @param subTitle          商品标题
    * @param keywords          商品关键字
    * @param page              分页信息
    * @return
    */
   Page<EsProduct> findByNameOrSubTitleOrKeywords(String name, String subTitle, String
```

> 在idea中直接会提示对应字段

使用@Query注解可以用Elasticsearch的DSL语句进行查询

```
@Query("{"bool" : {"must" : {"field" : {"name" : "?0"}}}}")
Page<EsProduct> findByName(String name,Pageable pageable);
```

项目使用表说明

- pms_product：商品信息表
- pms_product_attribute：商品属性参数表
- pms_product_attribute_value：存储产品参数值的表

整合Elasticsearch实现商品搜索

在pom.xml中添加相关依赖

```
<!--Elasticsearch相关依赖-->
```

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

修改SpringBoot配置文件

> 修改application.yml文件，在spring节点下添加Elasticsearch相关配置。

```
data:
  elasticsearch:
    repositories:
      enabled: true
    cluster-nodes: 127.0.0.1:9300 # es的连接地址及端口号
    cluster-name: elasticsearch # es集群的名称
```

添加商品文档对象EsProduct

> 不需要中文分词的字段设置成@Field(type = FieldType.Keyword)类型，需要中文分词的设置成@Field(analyzer = "ikmaxword",type = FieldType.Text)类型。

```
package com.macro.mall.tiny.nosql.elasticsearch.document;

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

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.List;

/**
 * 搜索中的商品信息
 * Created by macro on 2018/6/19.
 */
@Document(indexName = "pms", type = "product",shards = 1,replicas = 0)
public class EsProduct implements Serializable {
    private static final long serialVersionUID = -1L;
    @Id
    private Long id;
    @Field(type = FieldType.Keyword)
    private String productSn;
    private Long brandId;
    @Field(type = FieldType.Keyword)
    private String brandName;
    private Long productCategoryId;
```

```
    @Field(type = FieldType.Keyword)
    private String productCategoryName;
    private String pic;
    @Field(analyzer = "ik_max_word",type = FieldType.Text)
    private String name;
    @Field(analyzer = "ik_max_word",type = FieldType.Text)
    private String subTitle;
    @Field(analyzer = "ik_max_word",type = FieldType.Text)
    private String keywords;
    private BigDecimal price;
    private Integer sale;
    private Integer newStatus;
    private Integer recommandStatus;
    private Integer stock;
    private Integer promotionType;
    private Integer sort;
    @Field(type =FieldType.Nested)
    private List<EsProductAttributeValue> attrValueList;

    //省略了所有getter和setter方法
}
```

添加EsProductRepository接口用于操作Elasticsearch

> 继承ElasticsearchRepository接口，这样就拥有了一些基本的Elasticsearch数据操作方法，同时定义了一个衍生查询方法。

```
package com.macro.mall.tiny.nosql.elasticsearch.repository;

import com.macro.mall.tiny.nosql.elasticsearch.document.EsProduct;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.elasticsearch.repository.ElasticsearchRepository;

/**
 * 商品ES操作类
 * Created by macro on 2018/6/19.
 */
public interface EsProductRepository extends ElasticsearchRepository<EsProduct, Long> {
    /**
     * 搜索查询
     *
     * @param name              商品名称
     * @param subTitle          商品标题
     * @param keywords          商品关键字
     * @param page              分页信息
     * @return
     */
    Page<EsProduct> findByNameOrSubTitleOrKeywords(String name, String subTitle, String
```

```
}
```

添加EsProductService接口

```
package com.macro.mall.tiny.service;

import com.macro.mall.tiny.nosql.elasticsearch.document.EsProduct;
import org.springframework.data.domain.Page;

import java.util.List;

/**
 * 商品搜索管理Service
 * Created by macro on 2018/6/19.
 */
public interface EsProductService {
    /**
     * 从数据库中导入所有商品到ES
     */
    int importAll();

    /**
     * 根据id删除商品
     */
    void delete(Long id);

    /**
     * 根据id创建商品
     */
    EsProduct create(Long id);

    /**
     * 批量删除商品
     */
    void delete(List<Long> ids);

    /**
     * 根据关键字搜索名称或者副标题
     */
    Page<EsProduct> search(String keyword, Integer pageNum, Integer pageSize);

}
```

添加EsProductService接口的实现类EsProductServiceImpl

```
package com.macro.mall.tiny.service.impl;

import com.macro.mall.tiny.dao.EsProductDao;
import com.macro.mall.tiny.nosql.elasticsearch.document.EsProduct;
import com.macro.mall.tiny.nosql.elasticsearch.repository.EsProductRepository;
import com.macro.mall.tiny.service.EsProductService;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.util.CollectionUtils;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;


/**
 * 商品搜索管理Service实现类
 * Created by macro on 2018/6/19.
 */
@Service
public class EsProductServiceImpl implements EsProductService {
    private static final Logger LOGGER = LoggerFactory.getLogger(EsProductServiceImpl.cl
    @Autowired
    private EsProductDao productDao;
    @Autowired
    private EsProductRepository productRepository;

    @Override
    public int importAll() {
        List<EsProduct> esProductList = productDao.getAllEsProductList(null);
        Iterable<EsProduct> esProductIterable = productRepository.saveAll(esProductList)
        Iterator<EsProduct> iterator = esProductIterable.iterator();
        int result = 0;
        while (iterator.hasNext()) {
            result++;
            iterator.next();
        }
        return result;
    }

    @Override
    public void delete(Long id) {
        productRepository.deleteById(id);
    }

    @Override
    public EsProduct create(Long id) {
        EsProduct result = null;
        List<EsProduct> esProductList = productDao.getAllEsProductList(id);
        if (esProductList.size() > 0) {
            EsProduct esProduct = esProductList.get(0);
            result = productRepository.save(esProduct);
        }
        return result;
    }

    @Override
```

```
    public void delete(List<Long> ids) {
        if (!CollectionUtils.isEmpty(ids)) {
            List<EsProduct> esProductList = new ArrayList<>();
            for (Long id : ids) {
                EsProduct esProduct = new EsProduct();
                esProduct.setId(id);
                esProductList.add(esProduct);
            }
            productRepository.deleteAll(esProductList);
        }
    }

    @Override
    public Page<EsProduct> search(String keyword, Integer pageNum, Integer pageSize) {
        Pageable pageable = PageRequest.of(pageNum, pageSize);
        return productRepository.findByNameOrSubTitleOrKeywords(keyword, keyword, keywor
    }


}
```

添加EsProductController定义接口

```
package com.macro.mall.tiny.controller;

import com.macro.mall.tiny.common.api.CommonPage;
import com.macro.mall.tiny.common.api.CommonResult;
import com.macro.mall.tiny.nosql.elasticsearch.document.EsProduct;
import com.macro.mall.tiny.service.EsProductService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.List;


/**
 * 搜索商品管理Controller
 * Created by macro on 2018/6/19.
 */
@Controller
@Api(tags = "EsProductController", description = "搜索商品管理")
@RequestMapping("/esProduct")
public class EsProductController {
    @Autowired
    private EsProductService esProductService;

    @ApiOperation(value = "导入所有数据库中商品到ES")
    @RequestMapping(value = "/importAll", method = RequestMethod.POST)
    @ResponseBody
```

```
    public CommonResult<Integer> importAllList() {
        int count = esProductService.importAll();
        return CommonResult.success(count);
    }

    @ApiOperation(value = "根据id删除商品")
    @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
    @ResponseBody
    public CommonResult<Object> delete(@PathVariable Long id) {
        esProductService.delete(id);
        return CommonResult.success(null);
    }

    @ApiOperation(value = "根据id批量删除商品")
    @RequestMapping(value = "/delete/batch", method = RequestMethod.POST)
    @ResponseBody
    public CommonResult<Object> delete(@RequestParam("ids") List<Long> ids) {
        esProductService.delete(ids);
        return CommonResult.success(null);
    }

    @ApiOperation(value = "根据id创建商品")
    @RequestMapping(value = "/create/{id}", method = RequestMethod.POST)
    @ResponseBody
    public CommonResult<EsProduct> create(@PathVariable Long id) {
        EsProduct esProduct = esProductService.create(id);
        if (esProduct != null) {
            return CommonResult.success(esProduct);
        } else {
            return CommonResult.failed();
        }
    }

    @ApiOperation(value = "简单搜索")
    @RequestMapping(value = "/search/simple", method = RequestMethod.GET)
    @ResponseBody
    public CommonResult<CommonPage<EsProduct>> search(@RequestParam(required = false) St
                                                      @RequestParam(required = false, de
                                                      @RequestParam(required = false, de
        Page<EsProduct> esProductPage = esProductService.search(keyword, pageNum, pageSi
        return CommonResult.success(CommonPage.restPage(esProductPage));
    }

}
```

进行接口测试

将数据库中数据导入到Elasticsearch

进行商品搜索

项目源码地址

https://github.com/macrozheng/mall-learning/tree/master/mall-tiny-06

推荐阅读

- mall架构及功能概览
- mall学习所需知识点（推荐资料）
- mall整合SpringBoot+MyBatis搭建基本骨架
- mall整合Swagger-UI实现在线API文档
- mall整合Redis实现缓存功能

- mall整合SpringSecurity和JWT实现认证和授权（一）
- mall整合SpringSecurity和JWT实现认证和授权（二）
- mall整合SpringTask实现定时任务

欢迎关注，点个在看

收录于合集 #mall学习教程（架构篇） 10

上一篇
mall整合SpringTask实现定时任务

下一篇
mall整合Mongodb实现文档操作

阅读原文

喜欢此内容的人还喜欢

项目中到底该不该用Lombok?
macrozheng