1. **SpringBoot整合elasticsearch**

在这一篇文章开始之前，你需要先安装一个ElasticSearch，如果你是mac或者linux可以参考[www.jianshu.com/p/e47b45137…](https://link.juejin.im/?target=https%3A%2F%2Fwww.jianshu.com%2Fp%2Fe47b451375ea)，如果是windows可以自定百度一下。

下面是官方套话介绍elasticsearch：

**ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。 我们建立一个网站或应用程序，并要添加搜索功能，但是想要完成搜索工作的创建是非常困难的。我们希望搜索解决方案要运行速度快，我们希望能有一个零配置和一个完全免费的搜索模式，我们希望能够简单地使用JSON通过HTTP来索引数据，我们希望我们的搜索服务器始终可用，我们希望能够从一台开始并扩展到数百台，我们要实时搜索，我们要简单的多租户，我们希望建立一个云的解决方案。因此我们利用Elasticsearch来解决所有这些问题及可能出现的更多其它问题。**

具体elasticsearch相关问题可以去[elastic中文社区](https://link.juejin.im/?target=https%3A%2F%2Felasticsearch.cn%2F)查看。

这一篇文章将介绍springboot如何简单整合elasticsearch。

1.启动elasticsearch。  
2.新建项目，pom文件中加入elasticsearch依赖，完整pom如下：

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.dalaoyang</groupId>

<artifactId>springboot\_elasticsearch</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springboot\_elasticsearch</name>

<description>springboot\_elasticsearch</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.9.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-elasticsearch</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

复制代码

配置文件如下：

##端口号

server.port=8888

##es地址

spring.data.elasticsearch.cluster-nodes = 127.0.0.1:9300

复制代码

新建一个商品实体类GoodsInfo，其中需要注意：  
indexName：索引名称 可以理解为数据库名 必须为小写 不然会报org.elasticsearch.indices.InvalidIndexNameException异常  
type：类型 可以理解为表名

package com.dalaoyang.entity;

import org.springframework.data.elasticsearch.annotations.Document;

import java.io.Serializable;

/\*\*

\* @author dalaoyang

\* @Description

\* @project springboot\_learn

\* @package com.dalaoyang.entity

\* @email yangyang@dalaoyang.cn

\* @date 2018/5/4

\*/

@Document(indexName = "testgoods",type = "goods")

//indexName索引名称 可以理解为数据库名 必须为小写 不然会报org.elasticsearch.indices.InvalidIndexNameException异常

//type类型 可以理解为表名

public class GoodsInfo implements Serializable {

private Long id;

private String name;

private String description;

public Long **getId**() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String **getName**() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String **getDescription**() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public GoodsInfo(Long id, String name, String description) {

this.id = id;

this.name = name;

this.description = description;

}

public **GoodsInfo**() {

}

}

复制代码

创建GoodsRepository，继承ElasticsearchRepository，代码如下：

package com.dalaoyang.repository;

import com.dalaoyang.entity.GoodsInfo;

import org.springframework.data.elasticsearch.repository.ElasticsearchRepository;

import org.springframework.stereotype.Component;

/\*\*

\* @author dalaoyang

\* @Description

\* @project springboot\_learn

\* @package com.dalaoyang.repository

\* @email yangyang@dalaoyang.cn

\* @date 2018/5/4

\*/

@Component

public interface GoodsRepository extends ElasticsearchRepository<GoodsInfo,Long> {

}

复制代码

最后新建一个controller进行测试，其中包含简单的增删改查以及一个搜索，增删改查这里不做解释了。需要注意es的第一页页码是0

## 关键

package com.dalaoyang.controller;

import com.dalaoyang.entity.GoodsInfo;

import com.dalaoyang.repository.GoodsRepository;

import org.elasticsearch.index.query.QueryBuilders;

import org.elasticsearch.index.query.functionscore.FunctionScoreQueryBuilder;

import org.elasticsearch.index.query.functionscore.ScoreFunctionBuilders;

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.data.elasticsearch.core.query.NativeSearchQueryBuilder;

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

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

/\*\*

\* @author dalaoyang

\* @Description

\* @project springboot\_learn

\* @package com.dalaoyang.controller

\* @email yangyang@dalaoyang.cn

\* @date 2018/5/4

\*/

@RestController

public class GoodsController {

@Autowired

private GoodsRepository goodsRepository;

//http://localhost:8888/save

@GetMapping("save")

public String **save**(){

GoodsInfo goodsInfo = new GoodsInfo(System.currentTimeMillis(),

"商品"+System.currentTimeMillis(),"这是一个测试商品");

goodsRepository.save(goodsInfo);

return "success";

}

//http://localhost:8888/delete?id=1525415333329

@GetMapping("delete")

public String delete(long id){

goodsRepository.delete(id);

return "success";

}

//http://localhost:8888/update?id=1525417362754&name=修改&description=修改

@GetMapping("update")

public String update(long id,String name,String description){

GoodsInfo goodsInfo = new GoodsInfo(id,

name,description);

goodsRepository.save(goodsInfo);

return "success";

}

//http://localhost:8888/getOne?id=1525417362754

@GetMapping("getOne")

public GoodsInfo getOne(long id){

GoodsInfo goodsInfo = goodsRepository.findOne(id);

return goodsInfo;

}

//每页数量

private Integer PAGESIZE=10;

//http://localhost:8888/getGoodsList?query=商品

//http://localhost:8888/getGoodsList?query=商品&pageNumber=1

//根据关键字"商品"去查询列表，name或者description包含的都查询

@GetMapping("getGoodsList")

public List<GoodsInfo> getList(Integer pageNumber,String query){

**if**(pageNumber==null) pageNumber = 0;

//es搜索默认第一页页码是0

SearchQuery searchQuery=getEntitySearchQuery(pageNumber,PAGESIZE,query);

Page<GoodsInfo> goodsPage = goodsRepository.search(searchQuery);

return goodsPage.getContent();

}

private SearchQuery getEntitySearchQuery(int pageNumber, int pageSize, String searchContent) {

FunctionScoreQueryBuilder **function**ScoreQueryBuilder = QueryBuilders.functionScoreQuery()

.add(QueryBuilders.matchPhraseQuery("name", searchContent),

ScoreFunctionBuilders.weightFactorFunction(100))

.add(QueryBuilders.matchPhraseQuery("description", searchContent),

ScoreFunctionBuilders.weightFactorFunction(100))

//设置权重分 求和模式

.scoreMode("sum")

//设置权重分最低分

.setM**in**Score(10);

// 设置分页

Pageable pageable = new PageRequest(pageNumber, pageSize);

return new NativeSearchQueryBuilder()

.withPageable(pageable)

.withQuery(**function**ScoreQueryBuilder).build();

}

}

复制代码