**Lucene全文检索技术项目开发记录**

**指导老师：高华玲**

**组长：张科**

**组员：邹洲、赵祥文、李天杰**

**项目名：Lucene全文检索技术**

**2021/5/14**

目录

**[1、 数据准备 3](#_Toc20884)**

[1.1插入mysql数据 3](#_Toc28673)

[1.2建表语句 4](#_Toc20915)

**[2、 导出数据 4](#_Toc7263)**

[2.1创建Sku对象 4](#_Toc14597)

[2.2连接mysql数据库，导出数据 5](#_Toc18758)

[2.3创建接口函数 5](#_Toc4439)

**[3、 创建索引库 6](#_Toc3161)**

[3.1mysql数据采集 6](#_Toc10404)

[3.2取出数据，放入Documen 6](#_Toc22273)

[3.3创建分词器，创建Directory目录对象 6](#_Toc163)

[3.4指定分词器与保存目录 6](#_Toc27201)

[3.5查看索引库文件 7](#_Toc11212)

**[4、 配置文内容（pom.xml） 8](#_Toc5676)**

**[5、 配置resources文件夹 10](#_Toc2935)**

[5.1配置applicatio.yml文件 10](#_Toc2563)

[5.2 resources文件 10](#_Toc16387)

**[6、 实现在前端的搜索功能 10](#_Toc25157)**

[6.1定义启动函数Application 11](#_Toc25384)

[6.2定义查询函数searchServiceImpl 11](#_Toc19510)

[6.3创建接口SearchService 13](#_Toc9693)

[6.4定义与页面的交互函数SearchController 13](#_Toc4773)

**[7、 简单介绍前端页面的js部分 14](#_Toc30337)**

[7.1初始化删选条件 14](#_Toc17926)

[7.2提交表单 14](#_Toc14622)

[7.3价格删选 14](#_Toc526)

**[8、 项目启动截图 15](#_Toc11015)**

[8.1启动项目 15](#_Toc9585)

[8.2前端页面 15](#_Toc5151)

[8.3搜索测试 16](#_Toc13443)

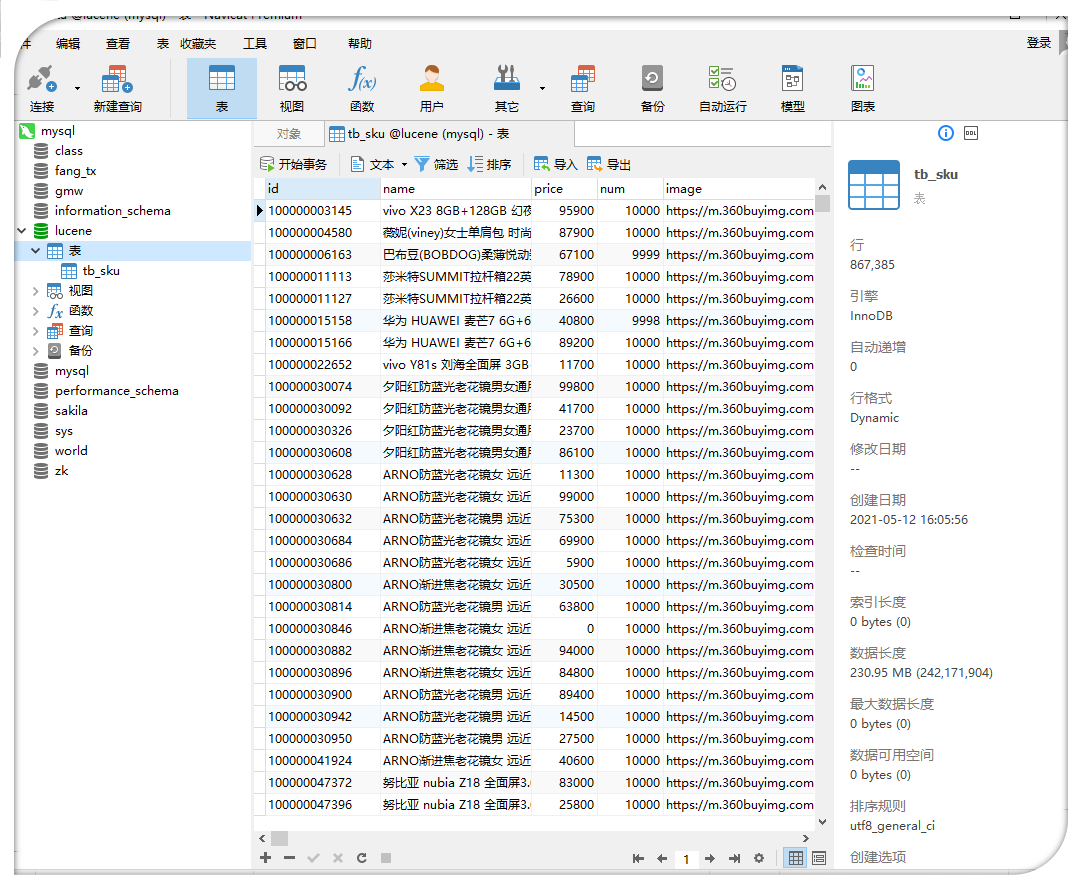
**[9、 项目结语 17](#_Toc15233)**

**[10、 附zk.iml的配置 17](#_Toc9655)**

1. **数据准备**

**1.1插入mysql数据**

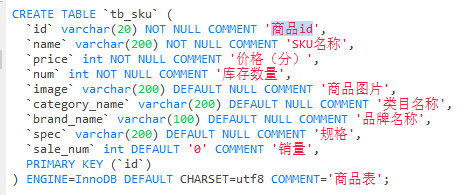
将准备的mysql语句，进行直接的插入，在使用之前，可以线创建一个数据库，在mysql建库语句里面插入use ‘database\_name’，才能在mysql workbench上面运行（在Navicat Premium上直接运行不需要加入，当然加入也没有影响。）



mysql数据展示

一共是938864条数据，数据的价格有问题。在使用mysql workbench运行时，由于mysql的建库语句太多，数据太大，导致导入的时候卡死，不建议使用。

**1.2建表语句**



表的信息内容

1. **导出数据**

**2.1创建Sku对象**

首先是建立一个对象，用于存放mysql导出的每一条数据。Sku类

//商品主键id

private String id;

//商品名称

private String name;

//价格

private Integer price;

//库存数量

private Integer num;

//图片

private String image;

//分类名称

private String categoryName;

//品牌名称

private String brandName;

//规格

private String spec;

//销量

private Integer saleNum;

再创建属于这个类的get和set方法，就是取和赋值（略）

**2.2连接mysql数据库，导出数据**

在就是连接数据库，以及查询数据，获得查询的数据，将每一行数据都赋值类一个Sku类，在将每个Sku类都添加到list中。

while (resultSet.next()) {

Sku sku = new Sku();

sku.setId(resultSet.getString("id"));

sku.setName(resultSet.getString("name"));

sku.setSpec(resultSet.getString("spec"));

sku.setBrandName(resultSet.getString("brand\_name"));

sku.setCategoryName(resultSet.getString("category\_name"));

sku.setImage(resultSet.getString("image"));

sku.setNum(resultSet.getInt("num"));

sku.setPrice(resultSet.getInt("price"));

sku.setSaleNum(resultSet.getInt("sale\_num"));

// 添加到集合中

list.add(sku);

}

} catch (Exception e) {

e.printStackTrace();

}

// 返回所有集合，大概100万条数据，即100万个sku类型的数据

return list;

**2.3创建接口函数**

创建一个接口函数，启动querySkuList()这个函数，将返回的list导出。

此时的list里面应该包含了938864个Sku类。package cn.zk.dao;

import cn.zk.pojo.Sku;

import java.util.List;

public interface SkuDao {

public List<Sku> querySkuList();

}

1. **创建索引库**

其实创建索引库就是将文本或者标题进行分词后，形成的一系列的一个文件。

在更目录下创建一个dir，将创建的索引库导入到dir中。

**3.1mysql数据采集**

对mysql导出的数据进行采集

//采集数据

SkuDao skuDao = new SkuDaoImpl();

List<Sku> skuList = skuDao.querySkuList();

**3.2取出数据，放入Documen**

使用for循环将list的Sku类都拿出来，放到Document中

//文档集合

List<Document> docList = new ArrayList<>();

// 循环取出所有的 Sku 对象

for (Sku sku : skuList) {

//2. 创建文档对象

Document document = new Document();

document.add(new StringField("id", sku.getId(), Field.Store.YES));

document.add(new TextField("name", sku.getName(), Field.Store.YES));

document.add(new IntPoint("price", sku.getPrice()));

document.add(new StoredField("price", sku.getPrice()));

document.add(new StoredField("image", sku.getImage()));

document.add(new StringField("categoryName", sku.getCategoryName(), Field.Store.YES));

document.add(new StringField("brandName", sku.getBrandName(), Field.Store.YES));

//将文档对象放入到文档集合中

docList.add(document);

}

**3.3创建分词器，创建Directory目录对象**

创建一个分词器

Analyzer analyzer = new StandardAnalyzer();

创建Directory目录对象，目录对象表示索引库的位置

Directory dir = FSDirectory.open(Paths.get("F:\\dir"));

**3.4指定分词器与保存目录**

分配指定的分词器，并将结果保存到F://dir中

//创建IndexWriter输出流对象, 指定输出的位置和使用的config初始化对象

IndexWriter indexWriter = new IndexWriter(dir, config);

//写入文档到索引库

for (Document doc : docList) {

indexWriter.addDocument(doc);

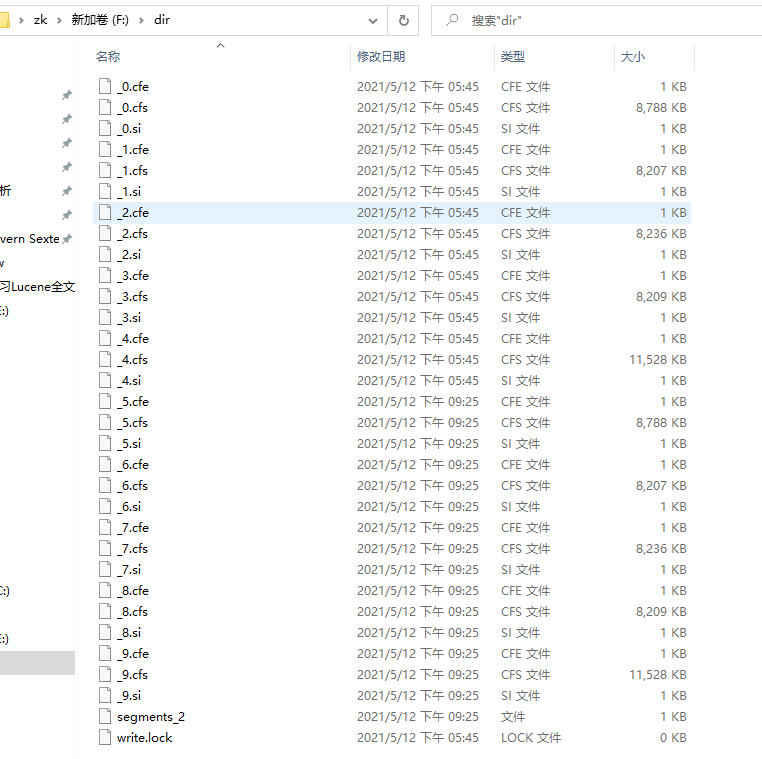
}

释放资源，防止内存溢出

indexWriter.close();

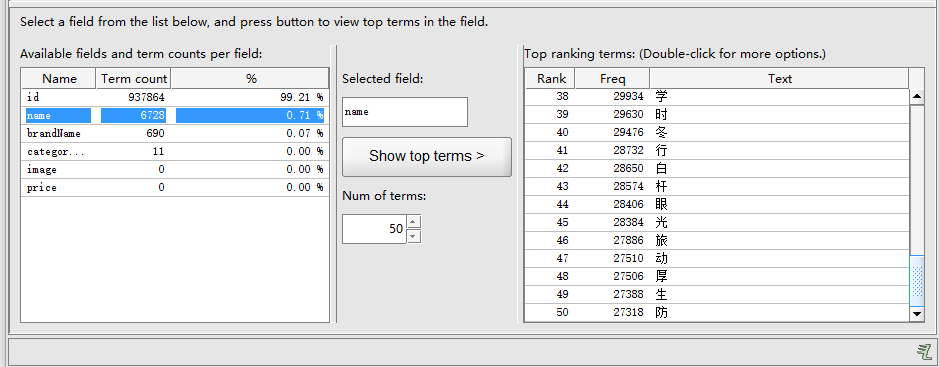
**3.5查看索引库文件**

目录：（F://dir），这里注意dir保存的路径

****

索引库建立成功图

可以使用luke.bat进行查看。



使用luke查看索引库内容

至此，索引库创建完成！

1. **配置文内容（pom.xml）**

<?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>

<!--配置文件-->

<groupId>cn.zk</groupId>

<artifactId>zk</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

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

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

<skipTests>true</skipTests>

</properties>

<parent>

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

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

<version>2.1.4.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-core</artifactId>

<version>7.7.2</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-analyzers-common</artifactId>

<version>7.7.2</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-queryparser</artifactId>

<version>7.7.2</version>

</dependency>

<!-- 测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<!-- mysql数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.48</version>

</dependency>

<!--web起步依赖-->

<dependency>

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

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

</dependency>

<!-- 引入thymeleaf -->

<dependency>

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

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

</dependency>

<!-- Json转换工具 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.51</version>

</dependency>

</dependencies>

</project>

1. **配置resources文件夹**

**5.1配置applicatio.yml文件**

先配置一个applicatio.yml文件

#关掉缓冲

spring:

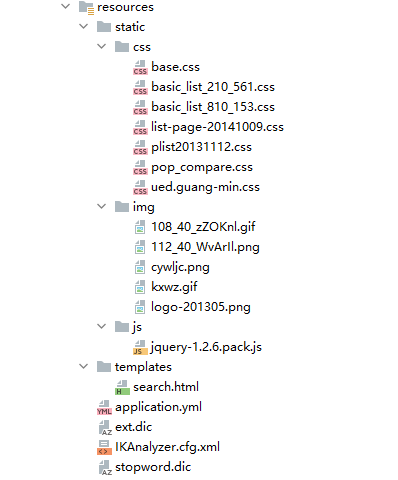
thymeleaf:

cache: false

关闭缓冲，否则在几次的搜索后，回导致内存的溢出。

**5.2 resources文件**

导入前段需要的css代码、img以及js等。目录如下：



resources文件夹目录展示

在static里面保存的是css文件，，以及要用的图片和js

Search.html是前端的页面html代码，这里不对前端多做描述。

1. **实现在前端的搜索功能**

**6.1定义启动函数Application**

先定义一个项目的启动函数（Application）

package cn.zk;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

/\*\*

\*项目的启动类，通过这个类，启动这个项目

\*/

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

**6.2定义查询函数searchServiceImpl**

1. 、定义每页的显示数量（需要与网页的一页的数量进行匹配）

//每页查询20条数据

public final static Integer PAGE\_SIZE = 20;

1. 、在定义一个函数query（查询函数）先要知道网页的是第几页的数据，防止重复显示，先定义开始的显示数据是第几条，在设置结束的数据是第几条

//需要使用的对象封装

ResultModel resultModel = new ResultModel();

//从第几条开始查询

int start = (page - 1) \* PAGE\_SIZE;

//查询到多少条为止

Integer end = page \* PAGE\_SIZE;

//创建分词器

Analyzer analyzer = new StandardAnalyzer();

1. 、创建查询对象，导入前端输入的关键字（这里需要与前端的变量名进行同步，防止获取不到），这里需要判断用户输入的关键字是否为空，是空，这返回所有的数据库的文件，不是这等于用户输入的关键字。

//根据查询关键字封装查询对象

QueryParser queryParser = new QueryParser("name", analyzer);

Query query1 = null;

//判断传入的查询关键字是否为空, 如果为空查询所有, 如果不为空, 则根据关键字查询

if (StringUtils.isEmpty(queryString)) {

query1 = queryParser.parse("\*:\*");

} else {

query1 = queryParser.parse(queryString);

}

//将关键字查询对象, 封装到组合查询对象中

builder.add(query1, BooleanClause.Occur.MUST);

1. 、看用户是否用价格方面的要求，对搜索到的内容进行价格的删选

//根据价格范围封装查询对象

if (!StringUtils.isEmpty(price)) {

String[] split = price.split("-");

Query query2 = IntPoint.newRangeQuery("price", Integer.parseInt(split[0]), Integer.parseInt(split[1]));

//将价格查询对象, 封装到组合查询对象中

builder.add(query2, BooleanClause.Occur.MUST);

}

1. 、指定索引库的位置，创建相关对象，将搜索的条数返回，促成最大的页数，将查询的结果集返回，并且在后台返回消耗的时间（可以删去，对页面没有影响）

//创建Directory目录对象, 指定索引库的位置

/\*\*

\* 使用MMapDirectory消耗的查询时间

\* ====消耗时间为=========324ms

\* ====消耗时间为=========18ms

\*/

Directory directory = FSDirectory.open(Paths.get("F:\\dir"));

//创建输入流对象

IndexReader reader = DirectoryReader.open(directory);

//创建搜索对象

IndexSearcher indexSearcher = new IndexSearcher(reader);

//搜索并获取搜索结果

TopDocs topDocs = indexSearcher.search(builder.build(), end);

//获取查询到的总条数

resultModel.setRecordCount(topDocs.totalHits);

//获取查询到的结果集

ScoreDoc[] scoreDocs = topDocs.scoreDocs;

long endTime = System.currentTimeMillis();

System.out.println("====消耗时间为=========" + (endTime - startTime) + "ms");

6）、遍历返回的每一条数据，将其封装到每一个Sku对象中，方便后面的页面显示

//遍历结果集封装返回的数据

List<Sku> skuList = new ArrayList<>();

if (scoreDocs != null) {

for (int i = start; i < end; i ++) {

//通过查询到的文档编号, 找到对应的文档对象

Document document = reader.document(scoreDocs[i].doc);

//封装Sku对象

Sku sku = new Sku();

sku.setId(document.get("id"));

sku.setPrice(Integer.parseInt(document.get("price")));

sku.setImage(document.get("image"));

sku.setName(document.get("name"));

sku.setBrandName(document.get("brandName"));

sku.setCategoryName(document.get("categoryName"));

skuList.add(sku);

}

}

**6.3创建接口SearchService**

创建SearchServiceImpl函数的接口SearchService

package cn.zk.service;

import cn.zk.pojo.ResultModel;

public interface SearchService {

public ResultModel query(String queryString, String price, Integer page) throws Exception;

}

**6.4定义与页面的交互函数SearchController**

1）、处理页面的当前页

//处理当前页

if (StringUtils.isEmpty(page)) {

page = 1;

}

if (page <= 0) {

page = 1;

}

1. 、调用service查询，使用model对象去承接返回值

ResultModel resultModel = searchService.query(queryString, price, page);

model.addAttribute("result", resultModel);

1. 将查询到的页面，返回到前端页面中，这里要注意前端页面定义的相关值的函数名，一定要一致，否则无法赋值成功。

model.addAttribute("queryString", queryString);

model.addAttribute("price", price);

model.addAttribute("page", page);

1. **简单介绍前端页面的js部分**

**7.1初始化删选条件**

在查询的时候一定要清空之前的查询条件，如价格、当前的页面码等，防止查询的时候查询数据与查询的内容不符合或者查询出来的数据少的尴尬情况。

function query() {

//执行关键词查询时清空过滤条件

document.getElementById("price").value="";

document.getElementById("page").value="";

//执行查询

queryList();

}

**7.2提交表单**

提交表单，也就是前后端交互，将用户输入的关键字导入到java代码中（这里一定要注意函数名，防止不一致，无法传递）

function queryList() {

//提交表单

document.getElementById("actionForm").submit();

}

在页面定位搜索框，是以post方式发送的

<form th:id="actionForm" th:action="list" th:method="POST">

<div class="form">

<input th:type="text" class="text"

th:name="queryString" th:id="key" th:value="${queryString }">

<input type="button" value="搜索" class="button" onclick="query()">

</div>

<input th:type="hidden" th:name="price" th:id="price" th:value="${price }"/>

<input th:type="hidden" th:name="page" th:id="page" th:value="${result.curPage }"/>

</form>

**7.3价格删选**

页面的价格删选，当用户点击价格删选后，将值的范围传递到java代码中，这里要注意的还是变量名，以及它传入的方式，是传入的字符串还是整数。

function filter(key, value) {

document.getElementById(key).value=value;

queryList();

}

在页面找到对于的选择框所在地，查看发现是字符串，在处理的时候使用字符串处理的方法。

<li><a href="javascript:filter('price','0-500')">0-500</a></li>

<li><a href="javascript:filter('price','500-1000')">500-1000</a></li>

<li><a href="javascript:filter('price','1000-5000')">1000-5000</a></li>

<li><a href="javascript:filter('price','5000-10000')">5000-10000</a></li>

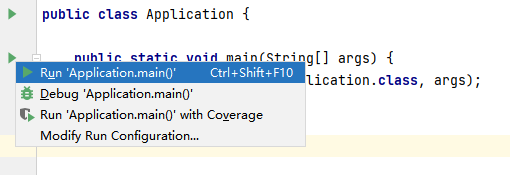
<li><a href="javascript:filter('price','10000-50000')">10000-50000</a></li>

<li><a href="javascript:filter('price','50000-100000')">50000-100000</a></li>

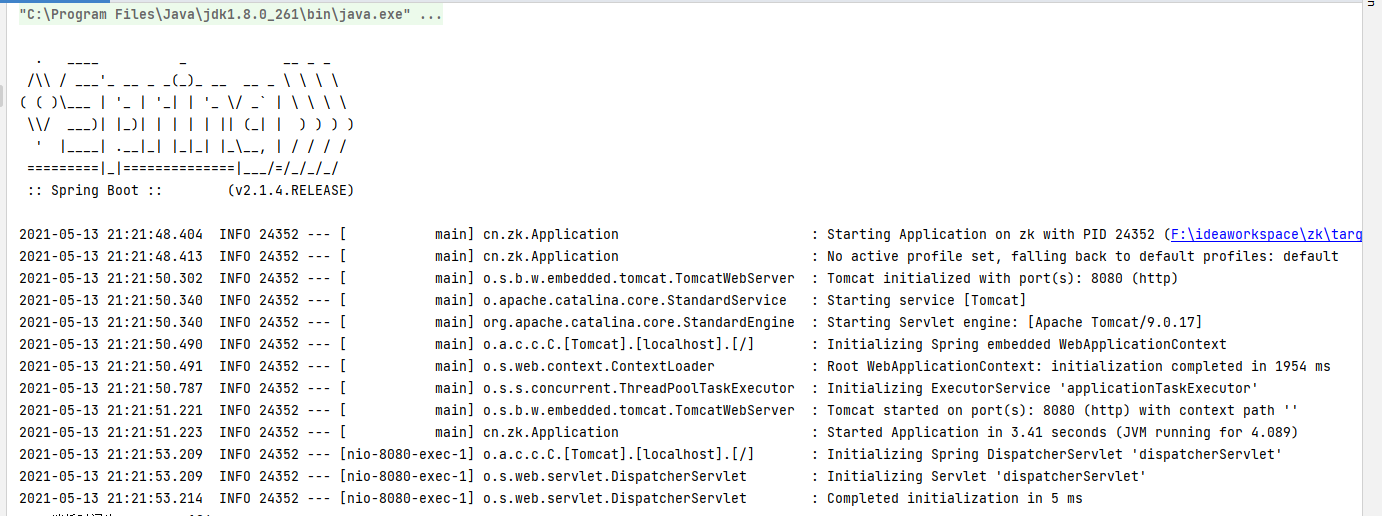
1. **项目启动截图**

**8.1启动项目**

在Application中启动项目



项目启动截图

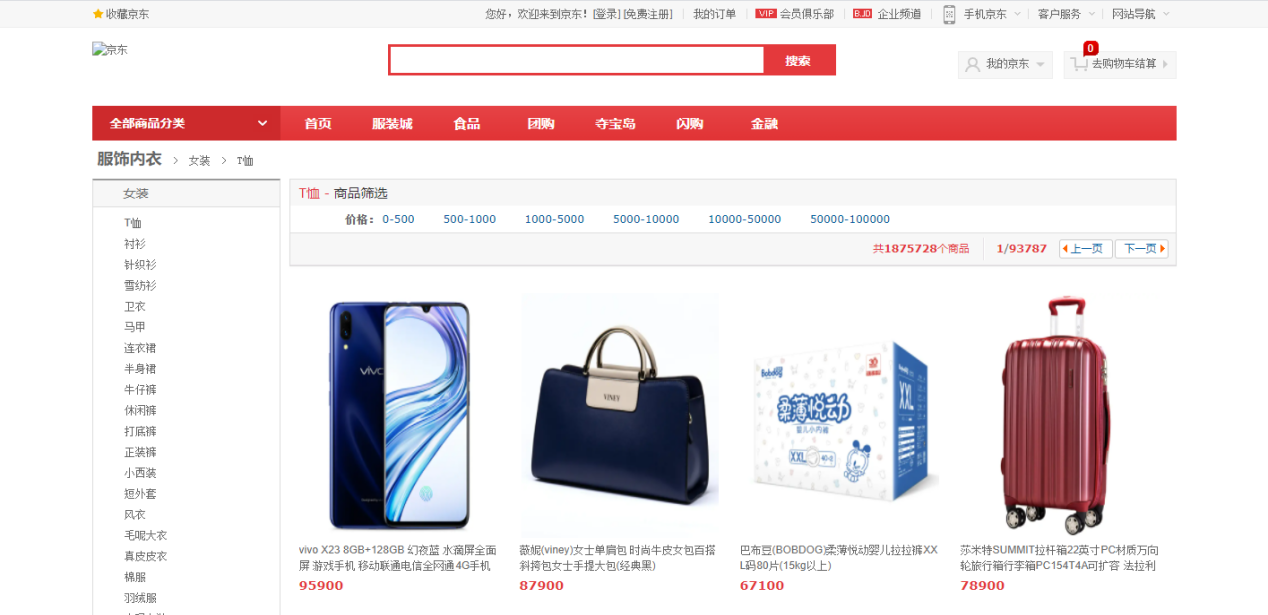


启动完成示例

到这里，项目启动完成！

**8.2前端页面**

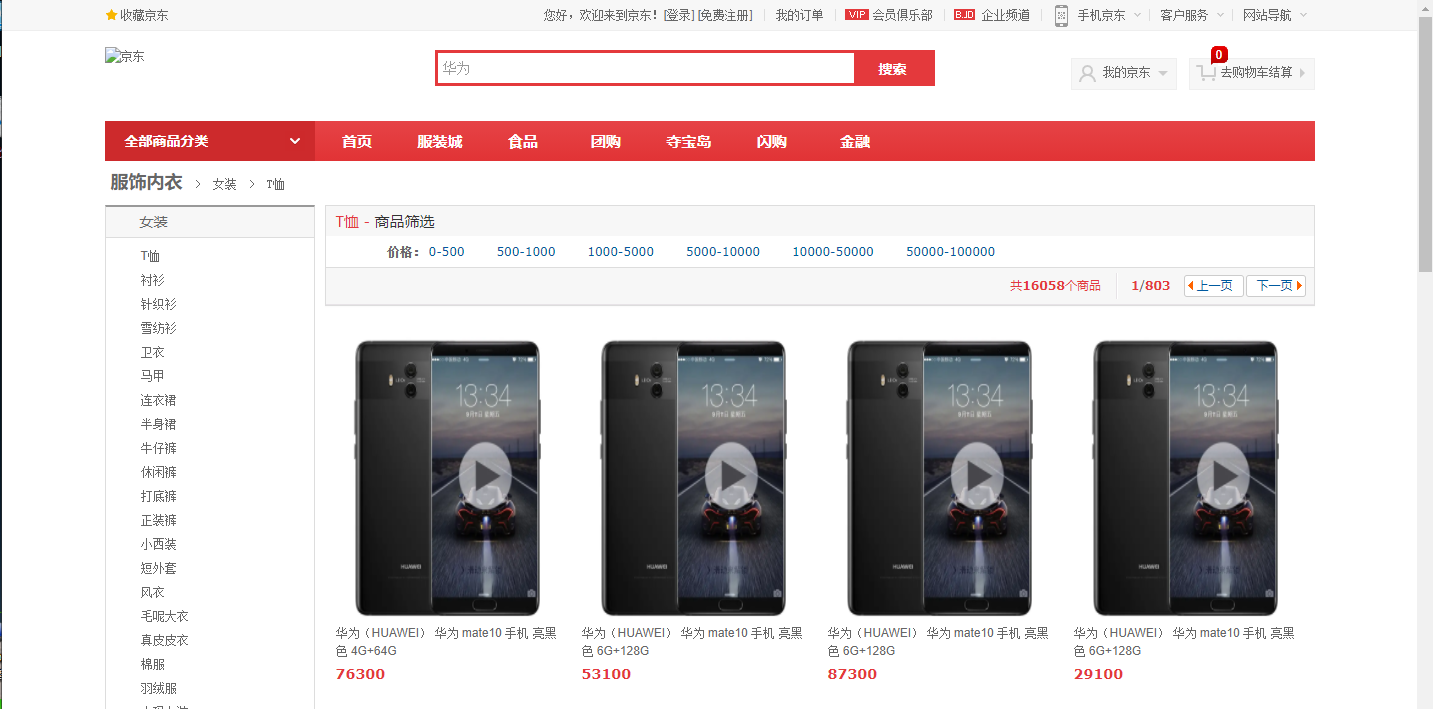
前端显示页面



前端显示示例

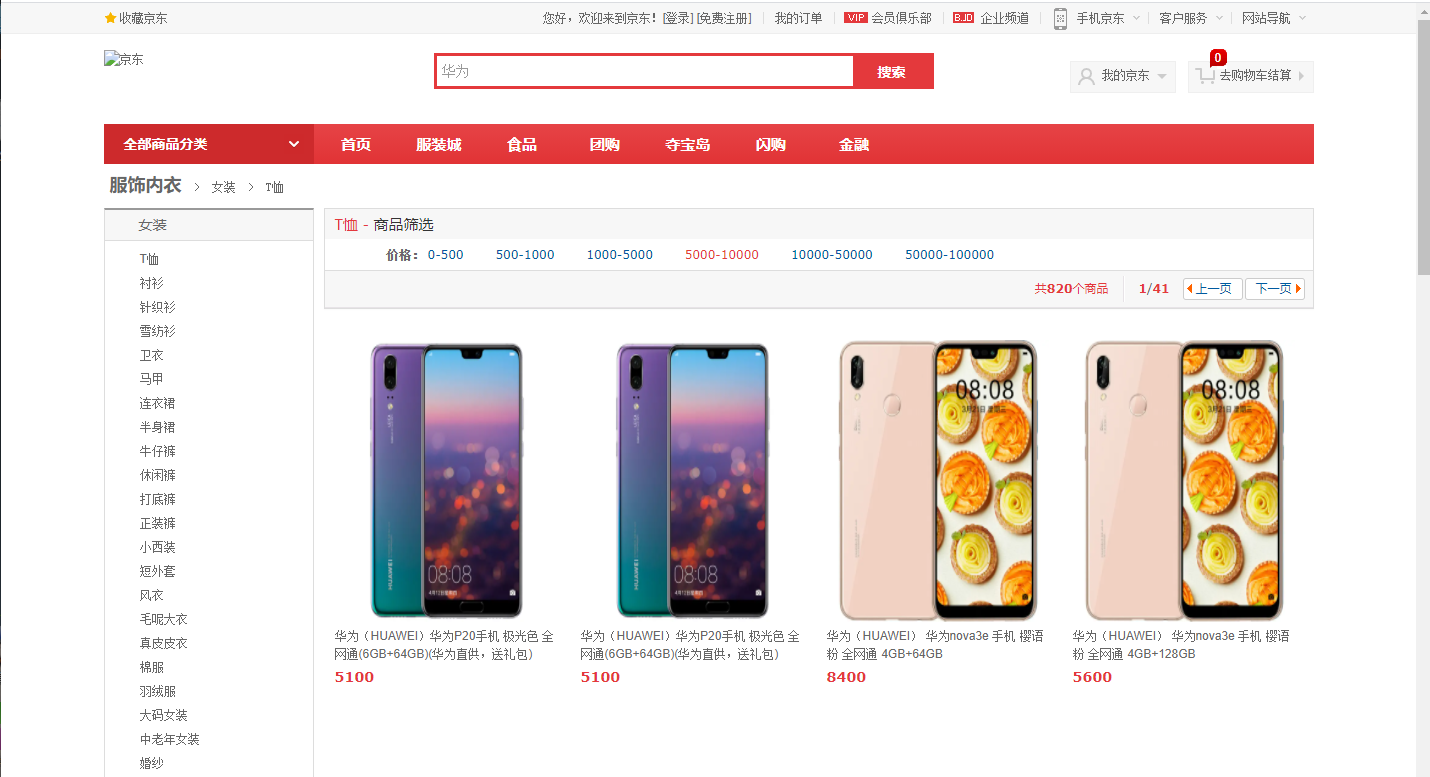
**8.3搜索测试**

搜索华为页面截图



搜索‘华为’显示示例

进行价格的删选（5000-10000）



价格删选显示示例

1. **项目结语**

在这次项目中，要注意一下几点：

1. 、在实验过程中，由于第三方的分词库无法使用，所以直接使用的standardAnalyzer()分词器
2. 、需要注意直接的dir文件的保存路径，由于索引库只创建一次，后面都是直接进行使用的，所以导入的时候，要注意路径的问题。
3. 、推荐使用本地的java包，不然在网上下载会影响项目启动的速度。

到此，学习该项目的学习记录到此结束，后附加zk.iml的配置内容，以及修改后的代码的下载地址：

<https://github.com/xx20000722/Search-Engines.git>

1. **附zk.iml的配置**

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

<module org.jetbrains.idea.maven.project.MavenProjectsManager.isMavenModule="true" type="JAVA\_MODULE" version="4">

<component name="FacetManager">

<facet type="Spring" name="Spring">

<configuration />

</facet>

<facet type="web" name="Web">

<configuration>

<webroots />

<sourceRoots>

<root url="file://$MODULE\_DIR$/src/main/java" />

<root url="file://$MODULE\_DIR$/src/main/resources" />

</sourceRoots>

</configuration>

</facet>

</component>

<component name="NewModuleRootManager" LANGUAGE\_LEVEL="JDK\_1\_8">

<output url="file://$MODULE\_DIR$/target/classes" />

<output-test url="file://$MODULE\_DIR$/target/test-classes" />

<content url="file://$MODULE\_DIR$">

<sourceFolder url="file://$MODULE\_DIR$/src/main/java" isTestSource="false" />

<sourceFolder url="file://$MODULE\_DIR$/src/main/resources" type="java-resource" />

<sourceFolder url="file://$MODULE\_DIR$/src/test/java" isTestSource="true" />

<excludeFolder url="file://$MODULE\_DIR$/target" />

</content>

<orderEntry type="jdk" jdkName="1.8" jdkType="JavaSDK" />

<orderEntry type="sourceFolder" forTests="false" />

<orderEntry type="library" name="Maven: commons-io:commons-io:2.6" level="project" />

<orderEntry type="library" name="Maven: org.apache.lucene:lucene-core:7.7.2" level="project" />

<orderEntry type="library" name="Maven: org.apache.lucene:lucene-analyzers-common:7.7.2" level="project" />

<orderEntry type="library" name="Maven: org.apache.lucene:lucene-queryparser:7.7.2" level="project" />

<orderEntry type="library" name="Maven: org.apache.lucene:lucene-queries:7.7.2" level="project" />

<orderEntry type="library" name="Maven: org.apache.lucene:lucene-sandbox:7.7.2" level="project" />

<orderEntry type="library" scope="TEST" name="Maven: junit:junit:4.12" level="project" />

<orderEntry type="library" scope="TEST" name="Maven: org.hamcrest:hamcrest-core:1.3" level="project" />

<orderEntry type="library" name="Maven: mysql:mysql-connector-java:5.1.48" level="project" />

<orderEntry type="library" name="Maven: org.wltea.ik-analyzer:ik-analyzer:8.1.0" level="project" />

<orderEntry type="library" name="Maven: ch.qos.logback:logback-classic:1.2.3" level="project" />

<orderEntry type="library" name="Maven: ch.qos.logback:logback-core:1.2.3" level="project" />

<orderEntry type="library" name="Maven: org.slf4j:slf4j-api:1.7.26" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter-web:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-autoconfigure:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter-logging:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.apache.logging.log4j:log4j-to-slf4j:2.11.2" level="project" />

<orderEntry type="library" name="Maven: org.apache.logging.log4j:log4j-api:2.11.2" level="project" />

<orderEntry type="library" name="Maven: org.slf4j:jul-to-slf4j:1.7.26" level="project" />

<orderEntry type="library" name="Maven: javax.annotation:javax.annotation-api:1.3.2" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-core:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-jcl:5.1.6.RELEASE" level="project" />

<orderEntry type="library" scope="RUNTIME" name="Maven: org.yaml:snakeyaml:1.23" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter-json:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.core:jackson-databind:2.9.8" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.core:jackson-annotations:2.9.0" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.core:jackson-core:2.9.8" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.9.8" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.9.8" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.9.8" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter-tomcat:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.apache.tomcat.embed:tomcat-embed-core:9.0.17" level="project" />

<orderEntry type="library" name="Maven: org.apache.tomcat.embed:tomcat-embed-el:9.0.17" level="project" />

<orderEntry type="library" name="Maven: org.apache.tomcat.embed:tomcat-embed-websocket:9.0.17" level="project" />

<orderEntry type="library" name="Maven: org.hibernate.validator:hibernate-validator:6.0.16.Final" level="project" />

<orderEntry type="library" name="Maven: javax.validation:validation-api:2.0.1.Final" level="project" />

<orderEntry type="library" name="Maven: org.jboss.logging:jboss-logging:3.3.2.Final" level="project" />

<orderEntry type="library" name="Maven: com.fasterxml:classmate:1.4.0" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-web:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-beans:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-webmvc:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-aop:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-context:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework:spring-expression:5.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.springframework.boot:spring-boot-starter-thymeleaf:2.1.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.thymeleaf:thymeleaf-spring5:3.0.11.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.thymeleaf:thymeleaf:3.0.11.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.attoparser:attoparser:2.0.5.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.unbescape:unbescape:1.1.6.RELEASE" level="project" />

<orderEntry type="library" name="Maven: org.thymeleaf.extras:thymeleaf-extras-java8time:3.0.4.RELEASE" level="project" />

<orderEntry type="library" name="Maven: com.alibaba:fastjson:1.2.51" level="project" />

</component>

</module>