

- solr
- lucene
 - 倒排索引
 - 实际举例
 - lucene API 介绍
 - 创建索引
 - 新建 maven 项目,添加依赖
 - 创建测试类,添加以下代码
 - 查看索引
 - 运行 luke
 - 查看文档
 - 指定分词器,并测试分词
 - 查询测试
 - 从索引查询
- solr 安装
 - 解压 solr
 - 启动 solr
 - 浏览器访问 solr 控制台
- 创建 core
 - 复制默认配置
 - 创建名为 pd 的 core
- 中文分词测试
 - 中文分词工具 - ik-analyzer
 - 使用 ik-analyzer 对中文进行分词测试
 - 设置停止词
- 准备 mysql 数据库数据
- 从 mysql 导入商品数据
 - 设置字段
 - Copy Field 副本字段
 - Data Import Handler 配置
 - 导入数据
 - 查询测试
 - 在复制字段 `_text_` 中查找 电脑
 - 在标题中查找 电脑
 - 用双引号查找完整词 "笔记本"
 - 搜索 `+lenovo +电脑`
 - 搜索 `+lenovo -电脑`
 - 统计 cid
 - 价格范围
 - 多字段统计

- 拼多多商城实现商品的全文检索
 - 修改 hosts 文件, 添加 www.pd.com 映射
 - eclipse 导入 pd-web 项目
 - 修改数据库连接配置
 - 启动项目, 访问 www.pd.com
 - 商品检索调用分析
 - pom.xml 添加 solr 和 lombok 依赖
 - application.yml 添加 solr 连接信息
 - Item 实体类
 - SearchService 业务接口
 - SearchServiceImpl 业务实现类
 - SearchController 控制器

solr

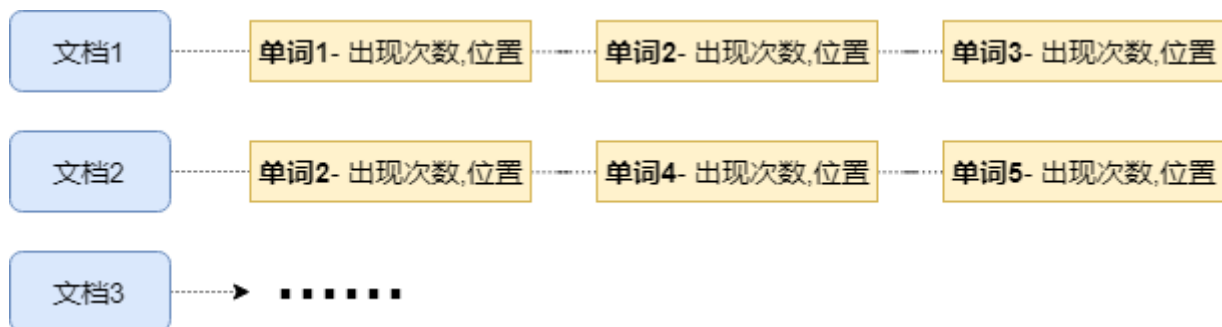
Solr是一个高性能, 基于Lucene的全文搜索服务器。同时对其进行了扩展, 提供了比Lucene更为丰富的查询语言, 同时实现了可配置、可扩展, 并对查询性能进行了优化, 并且提供了一个完善的功能管理界面, 是一款非常优秀的全文搜索引擎。

lucene

Lucene是apache jakarta项目的一个子项目, 是一个开放源代码的全文检索引擎开发工具包, 但它不是一个完整的全文检索引擎, 而是一个全文检索引擎的架构, 提供了完整的查询引擎和索引引擎, 部分文本分析引擎。Lucene的目的是为软件开发人员提供一个简单易用的工具包, 以方便的在目标系统中实现全文检索的功能, 或者是以此为基础建立起完整的全文检索引擎。

倒排索引

我们一般情况下,先找到文档,再在文档中找出包含的词;



倒排索引则是这个过程反过来,用词,来找出它出现的文档.



实际举例

文档编号	文档内容
1	全文检索引擎工具包
2	全文检索引擎的架构
3	查询引擎和索引引擎

分词结果

文档编号	分词结果集
1	{全文,检索,引擎,工具,包}
2	{全文,检索,引擎,的,架构}
3	{查询,引擎,和,索引,引擎}

倒排索引

编号	单词	文档编号列表
1	全文	1,2
2	检索	1,2
3	引擎	1,2,3
4	工具	1
5	包	1
6	架构	2
7	查询	3
8	索引	3

lucene API 介绍

创建索引

新建 maven 项目,添加依赖

```
<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>cn.tedu</groupId>
  <artifactId>lucene-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>luceme-demo</name>

  <dependencies>
    <dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-core</artifactId>
      <version>8.1.1</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
    </dependency>

    <dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-analyzers-smartcn</artifactId>
      <version>8.1.1</version>
    </dependency>

  </dependencies>
</project>
```

创建测试类,添加以下代码

```
package test;

import java.io.File;

import org.apache.lucene.analysis.cn.smart.SmartChineseAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field.Store;
import org.apache.lucene.document.LongPoint;
import org.apache.lucene.document.StoredField;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.FSDirectory;
import org.junit.Test;
```

```

public class Test1 {
    String[] a = {
        "3, 华为 - 华为电脑, 爆款",
        "4, 华为手机, 旗舰",
        "5, 联想 - Thinkpad, 商务本",
        "6, 联想手机, 自拍神器"
    };

    @Test
    public void test1() throws Exception {
        // 存储索引文件的路径
        File path = new File("d:/abc/");
        FSDirectory d = FSDirectory.open(path.toPath());
        // Lucene 提供的中文分词器
        SmartChineseAnalyzer analyzer = new SmartChineseAnalyzer();
        // 通过配置对象来指定分词器
        IndexWriterConfig cfg = new IndexWriterConfig(analyzer);
        // 索引输出工具
        IndexWriter writer = new IndexWriter(d, cfg);

        for (int i = 0; i < a.length; i++) {
            String[] strs = a[i].split(",");

            // 创建文档, 文档中包含的是要索引的字段
            Document doc = new Document();
            doc.add(new LongPoint("id", Long.parseLong(strs[0])));
            doc.add(new StoredField("id", Long.parseLong(strs[0])));
            doc.add(new TextField("title", strs[1], Store.YES));
            doc.add(new TextField("sellPoint", strs[2], Store.YES));

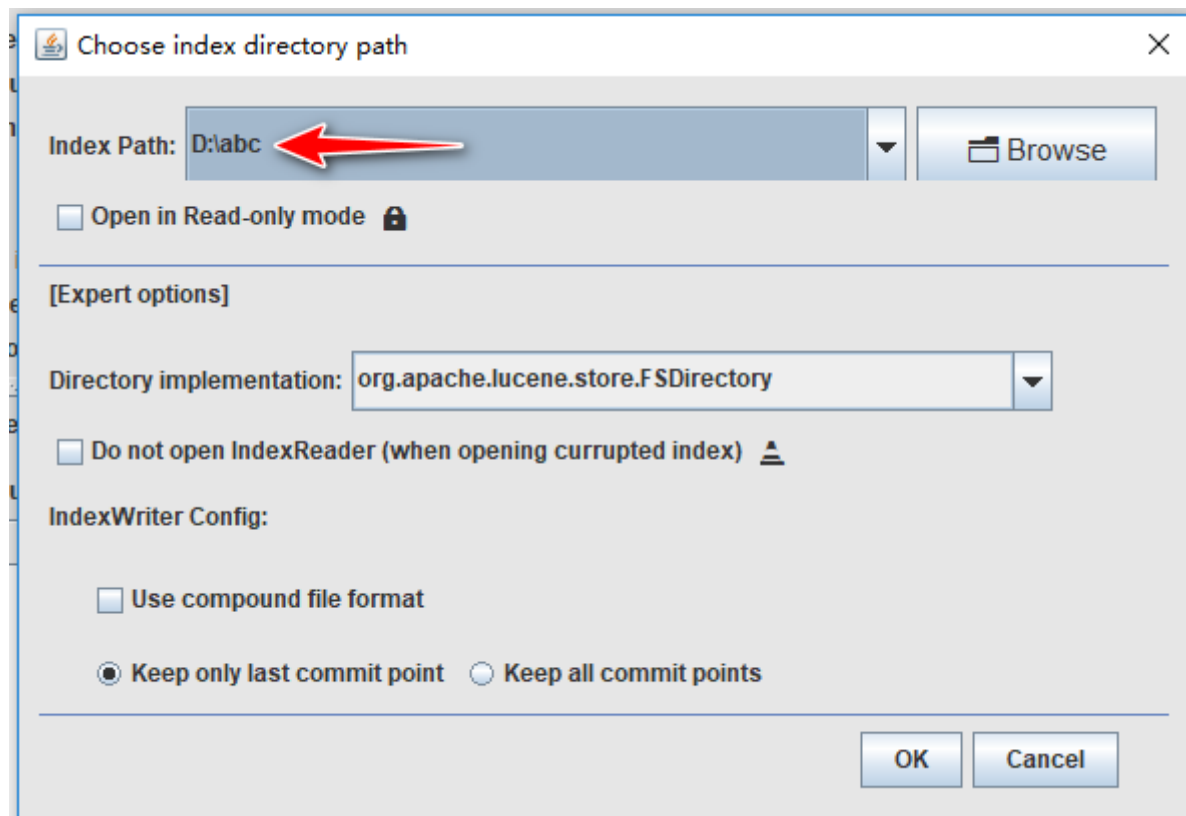
            // 将文档写入磁盘索引文件
            writer.addDocument(doc);
        }
        writer.close();
    }
}

```

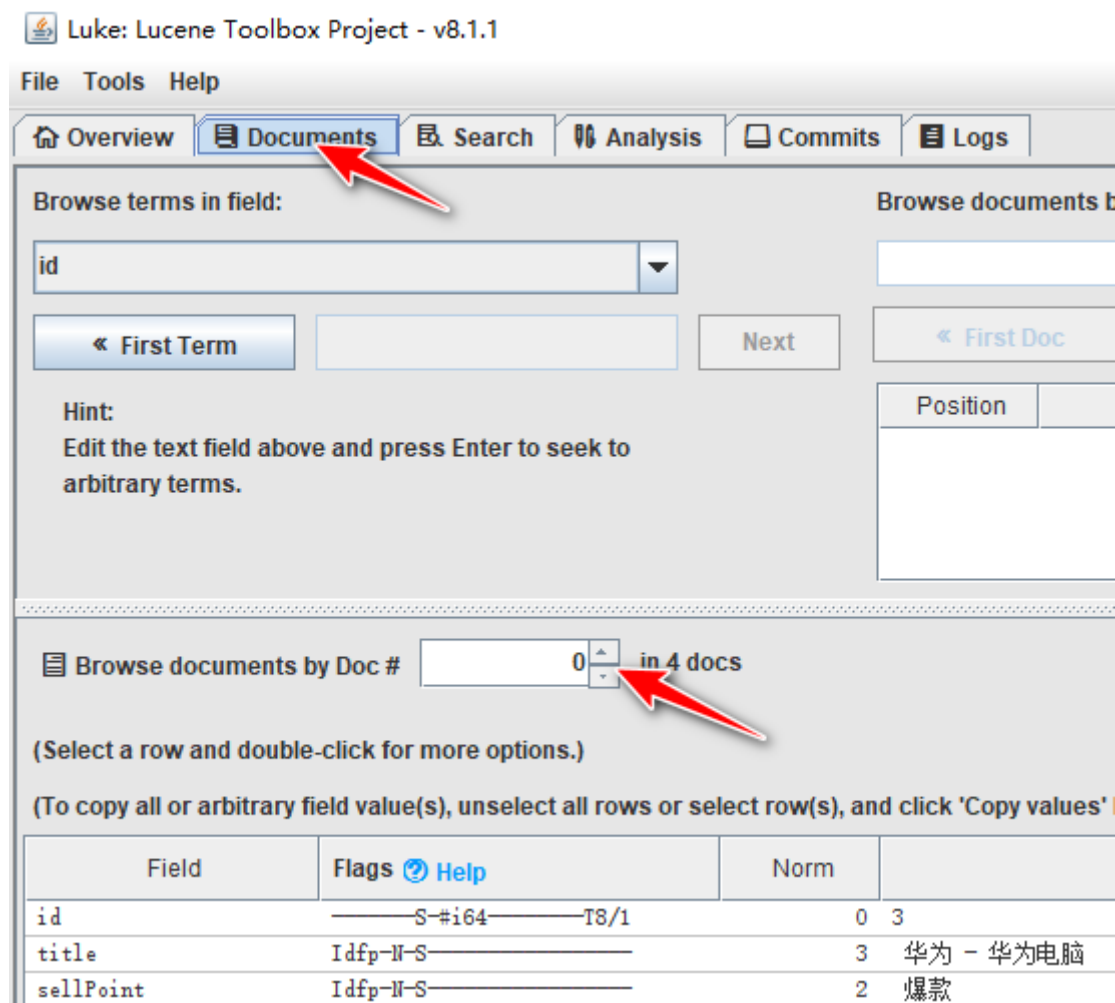
查看索引

运行 luke

运行 lucene 8.1.1 中的 luke 应用程序, 指定索引的存放目录



查看文档



指定分词器,并测试分词

Luke: Lucene Toolbox Project - v8.1.1

File Tools Help

Overview Documents Search **Analysis** Commits Logs

☒ Preset ☐ Custom

Preset analyzers:

org.apache.lucene.analysis.cn.smart.SmartChineseAnalyzer

Selected Analyzer: org.apache.lucene.analysis.cn.smart.SmartChineseAnalyzer

华为手机

Test Analyzer

Hint: Double click the row to show all token attributes.

Term	Attributes
华为	term=华为, bytes=[e5 8d 8e e4 b8 ba], startOffset=0, endOffset=2, positionIncrement=1, positionLe...
手机	term=手机, bytes=[e6 89 8b e6 9c ba], startOffset=2, endOffset=4, positionIncrement=1, positionLe...

查询测试

Luke: Lucene Toolbox Project - v8.1.1

File Tools Help

Overview Documents **Search** Analysis Commits Logs

Query settings

Query Parser Analyzer Similarity Sort Field Values More Like This

☒ StandardQueryParser ☐ Classic QueryParser

Default field: title Default operator: OR

☒ Enable position increments ☐ Allow leading wildcard (*)

☐ Split on whitespace

Phrase query:

☐ Generate phrase query

☐ Generate multi term synonyms phrase query

Phrase slop: 0 (integer value required)

Query expression: 华为

Parsed query:

Parse

Search

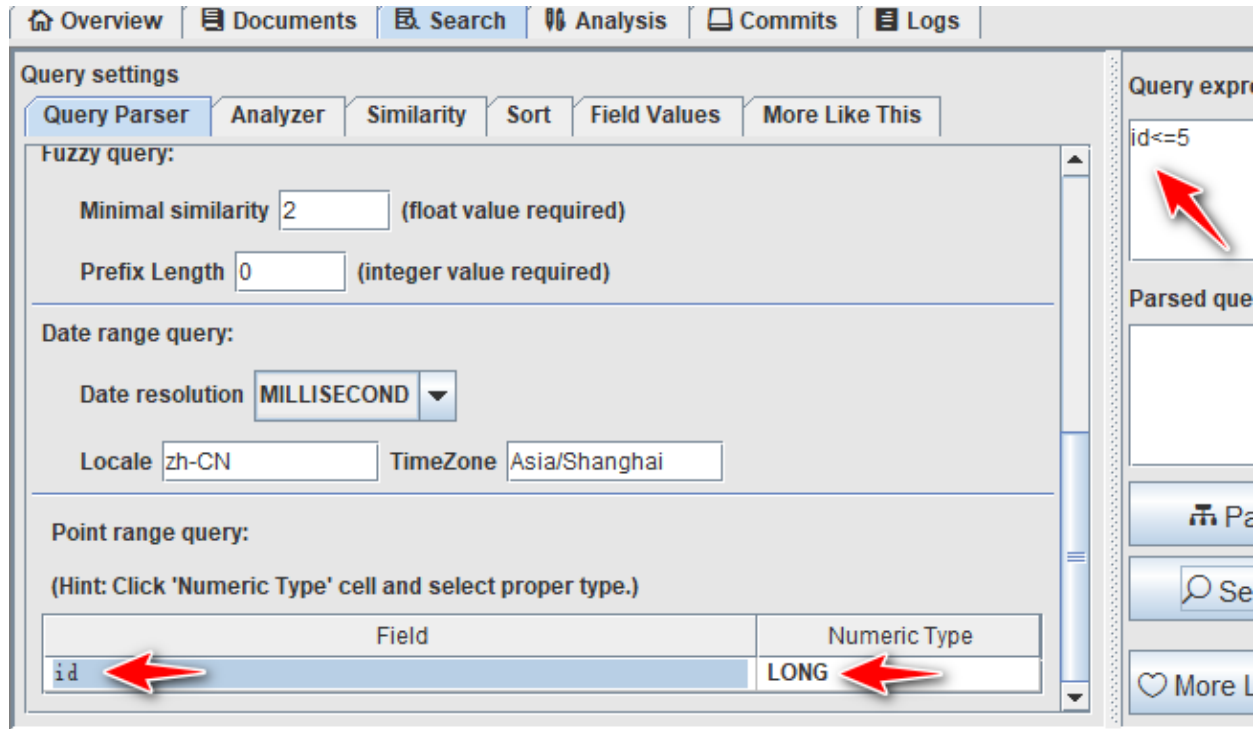
More Like This

Search Results: Total docs: 2 hits 1 ~

(Select a row and double-click for more options.)

Doc ID	Score	Field Values
0	0.396	sellPoint= 爆款; id=3; title= 华为 - 华为电脑;
1	0.33	sellPoint= 旗舰; id=4; title= 华为手机;

- id的查询



从索引查询

在测试类中添加 test2() 测试方法

```
package test;

import java.io.File;

import org.apache.lucene.analysis.cn.smart.SmartChineseAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field.Store;
import org.apache.lucene.document.LongPoint;
import org.apache.lucene.document.StoredField;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.Term;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TermQuery;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.FSDirectory;
import org.junit.Test;

public class Test1 {
    String[] a = {
        "3, 华为 - 华为电脑, 爆款",
        "4, 华为手机, 旗舰",
        "5, 联想 - Thinkpad, 商务本",
        "6, 联想手机, 自拍神器"
    };
}
```



```

};

@Test
public void test1() throws Exception {
    // 存储索引文件的路径
    File path = new File("d:/abc/");
    FSDirectory d = FSDirectory.open(path.toPath());
    // Lucene提供的中文分词器
    SmartChineseAnalyzer analyzer = new SmartChineseAnalyzer();
    // 通过配置对象来指定分词器
    IndexWriterConfig cfg = new IndexWriterConfig(analyzer);
    // 索引输出工具
    IndexWriter writer = new IndexWriter(d, cfg);

    for (int i = 0; i < a.length; i++) {
        String[] strs = a[i].split(",");

        // 创建文档, 文档中包含的是要索引的字段
        Document doc = new Document();
        doc.add(new LongPoint("id", Long.parseLong(strs[0])));
        doc.add(new StoredField("id", Long.parseLong(strs[0])));
        doc.add(new TextField("title", strs[1], Store.YES));
        doc.add(new TextField("sellPoint", strs[2], Store.YES));

        // 将文档写入磁盘索引文件
        writer.addDocument(doc);
    }
    writer.close();
}

@Test
public void test2() throws Exception {
    // 索引数据的保存目录
    File path = new File("d:/abc");
    FSDirectory d = FSDirectory.open(path.toPath());
    // 创建搜索工具对象
    DirectoryReader reader = DirectoryReader.open(d);
    IndexSearcher searcher = new IndexSearcher(reader);

    // 关键词搜索器, 我们搜索 "title: 华为"
    TermQuery q = new TermQuery(new Term("title", "华为"));
    // 执行查询, 并返回前20条数据
    TopDocs docs = searcher.search(q, 20);

    // 遍历查询到的结果文档并显示
    for (ScoreDoc scoreDoc : docs.scoreDocs) {
        Document doc = searcher.doc(scoreDoc.doc);
        System.out.println(doc.get("id"));
        System.out.println(doc.get("title"));
        System.out.println(doc.get("sellPoint"));
        System.out.println("-----");
    }
}
}

```

solr 安装

下面我们来安装 solr 服务器

解压 solr

```
cd /usr/local

# 上传 solr-8.1.1.tgz 到 /usr/local 目录
# 并解压缩
tar -xzf solr-8.1.1.tgz
```

启动 solr

```
cd /usr/local/solr-8.1.1

# 不建议使用管理员启动 solr, 加 -force 强制启动
bin/solr start -force

# 开放 8983 端口
firewall-cmd --zone=public --add-port=8983/tcp --permanent
firewall-cmd --reload
```

浏览器访问 solr 控制台

<http://192.168.64.170:8983>

- 注意修改地址



Dashboard

Logging



Core Admin

Java Properties

Instance

Start 5 minutes ago

Versions

 solr-spec	8.1.1
solr-impl	8.1.1 fcbe46c28cef11l
 lucene-spec	8.1.1

创建 core

数据库中 pd_item 表中的商品数据, 在 solr 中保存索引数据, 一类数据, 在 solr 中创建一个 core 保存索引数据

创建一个名为 pd 的 core, 首先要准备以下目录结构:

```
# solr目录/server/solr/  
# pd/  
# conf/  
# data/  
  
mkdir server/solr/pd  
mkdir server/solr/pd/conf  
mkdir server/solr/pd/data
```

conf 目录是 core 的配置目录, 存储一组配置文件, 我们以默认配置为基础, 后续逐步修改

复制默认配置

```
cp -r server/solr/configsets/_default/conf server/solr/pd
```

创建名为 pd 的 core



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

No cores available
Go and create one

+ Add Core

name: pd

instanceDir: pd

dataDir: data

config: solrconfig.xml

schema: schema.xml

instanceDir and dataDir need to exist before you can create the core

Add Core

Cancel

中文分词测试



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

pd

Overview

Analysis

Dataimport

Documents

Field Value (Index)

并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

Field Value (Query)

Analyse Fieldname / FieldType: text_general

Schema Browser

Verbose Output

Analyse Values

ST	text	Solr	是	—	个
	raw_bytes	[53 6f 6c 72]	[e6 98 af]	[e4 b8 80]	[e4 b8 aa]
	start	0	4	5	6
	end	4	5	6	7
	positionLength	1	1	1	1
	type	<ALPHANUM>	<IDEOGRAPHIC>	<IDEOGRAPHIC>	<IDEOG
	termFrequency	1	1	1	1
	position	1	2	3	4
SF	text	Solr	是	—	个

填入以下文本, 观察分词结果:

Solr是一个高性能, 采用Java5开发, 基于Lucene的全文搜索服务器。同时对其进行了扩展, 提供了比Lucene更为丰富的查询语言, 同时实现了可配置、可扩展并对查询性能进行了优化, 并且提供了一个完善的功能管理界面, 是一款非常优秀的全文搜索引擎。

中文分词工具 - ik-analyzer

<https://github.com/mage/ik-analyzer-solr>

简介

适配最新版本solr 7&8;

扩展IK原有词库:

分词工具	词库中词的数量	最后更新时间
ik	27.5万	2012年
mmseg	15.7万	2017年
word	64.2万	2014年
jieba	58.4万	2012年
jcesg	16.6万	2018年
sougou词库	115.2万	2019年

- 下载 ik-analyzer 分词 jar 文件,传到 `solr目录/server/solr-webapp/webapp/WEB-INF/lib`
- 复制6个文件到 `solr目录/server/solr-webapp/webapp/WEB-INF/classes`

```
# classes目录不存在,需要创建该目录
mkdir /usr/local/solr-8.1.1/server/solr-webapp/webapp/WEB-INF/classes
```

这6个文件复制到 classes 目录下

```
resources/
  IKAnalyzer.cfg.xml
  ext.dic
  stopword.dic
  stopwords.txt
  ik.conf
  dynamicdic.txt
```

- 配置 managed-schema

修改 `solr目录/server/solr/pd/conf/managed-schema` ,添加 ik-analyzer 分词器

```
<!-- ik分词器 -->
<fieldType name="text_ik" class="solr.TextField">
```

```

<analyzer type="index">
  <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false"
  conf="ik.conf"/>
  <filter class="solr.LowerCaseFilterFactory"/>
</analyzer>
<analyzer type="query">
  <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="true"
  conf="ik.conf"/>
  <filter class="solr.LowerCaseFilterFactory"/>
</analyzer>
</fieldType>

```

- 重启 solr 服务

```

cd /usr/local/solr-8.1.1

bin/solr restart -force

```

使用 ik-analyzer 对中文进行分词测试

填入以下文本, 选择使用 `text_ik` 分词器, 观察分词结果:

Solr是一个高性能, 采用Java5开发, 基于Lucene的全文搜索服务器。同时对其进行了扩展, 提供了比Lucene更为丰富的查询语言, 同时实现了可配置、可扩展并对查询性能进行了优化, 并且提供了一个完善的功能管理界面, 是一款非常优秀的全文搜索引擎。

The screenshot shows the Solr Admin UI. On the left is a sidebar with navigation links: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, Overview, Analysis (highlighted with a red arrow), Dataimport, Documents, and Files. The main area is titled 'Field Value (Index)' and 'Field Value (Query)'. Below this, there's a section for 'Analyse Fieldname / FieldType:' with a dropdown menu set to 'text_ik' (indicated by a red arrow) and a 'Schema Browser' link. A 'Verbose Output' checkbox is checked. A blue 'Analyse Values' button (indicated by a red arrow) is on the right. The results are displayed in a table with columns for the field name, raw bytes, start/end positions, and the analyzed tokens. The table shows the analysis of the text 'Solr是一个高性能, 采用Java5开发, 基于Lucene的全文搜索服务器。同时对其进行了扩展' using the 'text_ik' tokenizer. The results are grouped by field name (IKT and LCE) and show the raw bytes, start/end positions, and the analyzed tokens.

Field	raw_bytes	start	end	positionLength	type	termFrequency	position
IKT	solr	0	4	1	ENGLISH	1	1
IKT	是一个	4	7	1	CN_WORD	1	2
IKT	是	6	7	1	CN_WORD	1	3
IKT	一个	7	10	1	CN_WORD	1	4
LCE	solr	0	4	1	ENGLISH	1	1
LCE	是一个	4	7	1	CN_WORD	1	2
LCE	是	6	7	1	CN_WORD	1	3
LCE	一个	7	10	1	CN_WORD	1	4

设置停止词

上传停止词配置文件到 solr目录/server/solr-webapp/webapp/WEB-INF/classes

```
stopword.dic  
stopwords.txt
```

- 重启服务,观察分词结果中,停止词被忽略

```
bin/solr restart -force
```

准备 mysql 数据库数据

- 用 sqlyog 执行 pd.sql
- 授予 root 用户 跨网络访问权限
注意: 此处设置的是远程登录的 root 用户,本机登录的 root 用户密码不变

```
grant all on *.* to 'root'@'%' identified by 'root';
```

随机修改30%的商品,让商品下架,以便后面做查询测试

```
UPDATE pd_item SET STATUS=0 WHERE RAND()<0.3
```

从 mysql 导入商品数据

设置字段

- title text_ik
- sellPoint text_ik
- price plong
- barcode string
- image string
- cid plong
- status pint
- created pdate
- updated pdate



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

pd

Overview

Analysis

Dataimport

Documents

Files

Ping

Plugins / Stats

Query

Replication

Schema

Segments info

Add Field

Add Dynamic Field

Add Copy Field

name: title

field type: text_ik

default: enter a default value if needed

- ☒ stored
- ☒ indexed
- ☒ uninvertible
- ☐ docValues
- ☐ multiValued
- ☐ required

Show omit options ☒

Show term vector options ☒

Show sort options ☒

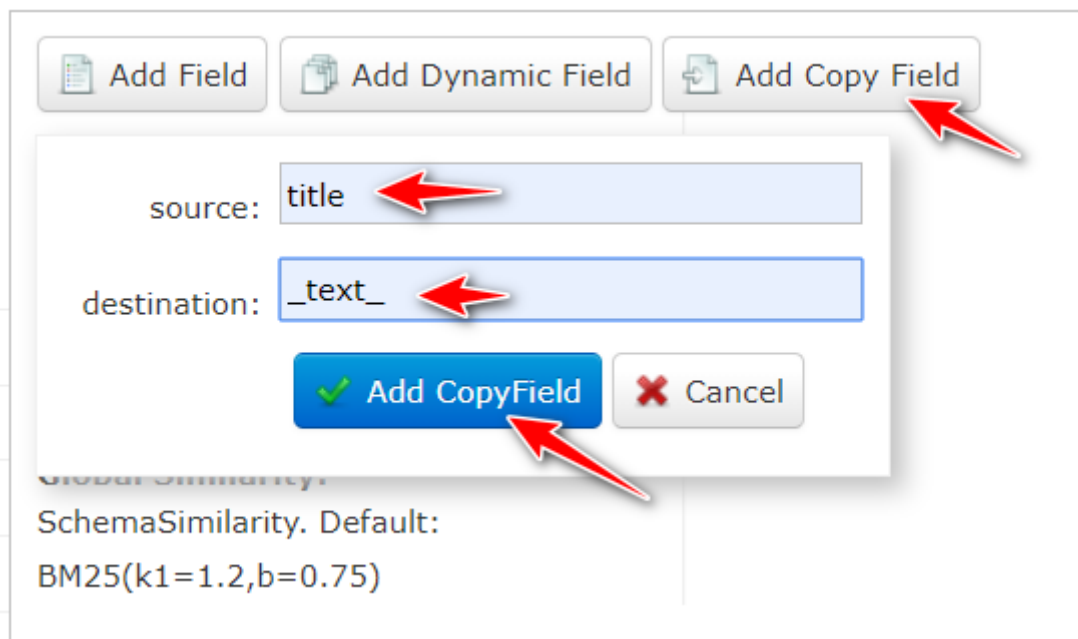
Add Field

Cancel

Copy Field 副本字段

查询时需要按字段查询,例如 `title:电脑`, 可以将多个字段的值合并到一个字段进行查询,默认查询字段 `_text_`

将 `title` 和 `sellPoint` 复制到 `_text_` 字段



Data Import Handler 配置

- 添加 jar 文件

Data Import Handler 的 jar 文件存放在 `solr目录/dist` 目录下

```
solr-dataimporthandler-8.1.1.jar  
solr-dataimporthandler-extras-8.1.1.jar
```

复制这两个文件和 mysql 的 jar 文件到 `solr目录/server/solr-webapp/webapp/WEB-INF/lib`

- `dih-config.xml` 修改 mysql 的 ip 地址,传到 `solr目录/server/solr/pd/conf`
- `solrconfig.xml` 中添加 DIH 配置

```
<requestHandler name="/dataimport"  
class="org.apache.solr.handler.dataimport.DataImportHandler">  
  <lst name="defaults">  
    <str name="config">dih-config.xml</str>  
  </lst>  
</requestHandler>
```

- 重启 solr

```
bin/solr restart -force
```

导入数据

重启 solr 后导入数据,确认导入的文档数量为 3160

The screenshot displays the Solr Admin interface. On the left, a sidebar contains navigation links: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, and a dropdown menu with 'pd'. Below these are links for Overview, Analysis, Dataimport (highlighted with a red arrow), Documents, Files, and Ping. The main content area is titled '/dataimport'. It features a 'Command' dropdown set to 'full-import' (with a red arrow), checkboxes for 'Verbose', 'Clean' (checked, with a red arrow), 'Commit', 'Optimize', and 'Debug'. Below these is an 'Entity' dropdown. The 'Start, Rows' section shows '0' and '10'. The 'Custom Parameters' field contains 'key1=val1&key2=val2'. At the bottom of this section are 'Execute' (with a red arrow) and 'Refresh Status' buttons. On the right, a green status box indicates 'Last Update: 11:33:34' and 'Indexing completed. Added/Updated: 3160 documents.' (with a red arrow). Below this are links for 'Raw Status-Output' and 'Configuration'.

查询测试

在复制字段 `_text_` 中查找 电脑



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

pd

Overview

Analysis

Dataimport

Documents

Files

Ping

Plugins / Stats

Query

Replication

Request-Handler (qt)

/select

common

q

电脑

fq

sort

start, rows

0

10

fl

df

Raw Query Parameters

key1=val1&key2=val2

wt

☐ indent off

http://192.168.64.170:8983/solr/pd

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 2,
    "params": {
      "q": "电脑",
      "_": "1561558870827"
    }
  },
  "response": {
    "numFound": 1756, "start": 0,
    {
      "image": "/images/server/images/po:
      "sellPoint": "给你满载而归的喜悦!",
      "price": 89,
      "created": "2017-01-16T16:13:42Z",
      "id": "10000043",
      "title": "乐尚书包 电脑包 bag黑色",
      "updated": "2017-01-13T16:16:57Z",
      "cid": 917,
      "status": 1,
      "_version_": 1637413026271330305
    },
    {
      "image": "/images/server/images/po:
      "sellPoint": "给你满载而归的喜悦!",
      "price": 89,
      "created": "2017-01-16T16:13:45Z",
      "id": "10000044",
      "title": "乐尚书包 电脑包 bag粉色"
```

在标题中查找 电脑

q

title:电脑

fq

sort

start, rows

0

10

fl

```
"status": 0,
"QTime": 25,
"params": {
  "q": "title:电脑",
  "_": "1561991032747"
},
"response": {
  "numFound": 10, "start": 0, "docs": [
    {
      "image": "/images/server/images/portal/22_LEXON_LNE602E
      "sellPoint": "给你满载而归的喜悦!",
      "price": 89,
      "created": "2017-01-16T16:13:42Z",
      "id": "10000043",
      "title": "乐尚书包 电脑包 bag黑色",
      "updated": "2017-01-13T16:16:57Z",
      "cid": 917
```

用双引号查找完整词 "笔记本"

q
"笔记本"

fq

sort

start, rows
0 10

fl

```
status :0,
"QTime":2,
"params":{
  "q":"\"笔记本\"",
  "_":"1561991032747"}},
"response":{"numFound":18,"start":0,"docs":[
{
  "image":"/images/mobileImage/02.png",
  "sellPoint":"经典回顾！超值特惠！",
  "price":49,
  "created":"2018-11-12T16:29:47Z",
  "id":"830972",
  "title":"生活笔记本",
  "updated":"2015-03-08T21:28:30Z",
  "cid":163
```

搜索 +lenovo +电脑

q
+lenovo +电脑

fq

sort

start, rows
0 10

fl

```
status :0,
"QTime":1,
"params":{
  "q":"+lenovo +电脑",
  "_":"1561991032747"}},
"response":{"numFound":8,"start":0,"docs":[
{
  "image":"/images/server/images/portal/17Lenovo) xiaoxinAir13Pro_gold/collect.png",
  "sellPoint":"青春的活力 青年专属",
  "price":6439,
  "created":"2017-01-13T16:16:57Z",
  "id":"10000037",
  "title":"联想(Lenovo) 小新Air13 Pro 13.3英寸14.8mm超轻薄笔记本电脑",
  "updated":"2017-01-13T16:16:57Z",
  "cid":163,
  "static":1
```

搜索 +lenovo -电脑

q
+lenovo -电脑

fq

sort

start, rows
0 10

fl

```
status :0,
"QTime":0,
"params":{
  "q":"+lenovo -电脑",
  "_":"1561991032747"}},
"response":{"numFound":25,"start":0,"docs":[
{
  "image":"/images/server/images/portal/13LenovoIdeaPad310_black/collect.png",
  "sellPoint":"清仓！仅北京，武汉仓有货！",
  "price":5119,
  "created":"2017-01-13T16:16:57Z",
  "id":"10000020",
  "title":"联想(Lenovo) IdeaPad310低配版",
  "updated":"2017-01-13T16:16:57Z",
  "cid":163,
  "static":1
```

统计 cid

☒ facet

facet.query

facet.field

facet.prefix

☐ spatial

☐ spellcheck

Execute Query

```
      "title": "联想 \\\\LENOVO / 笔记本\\\\经典版",
      "updated": "2017-01-11T13:09:32Z",
      "cid": 163,
      "status": 1,
      "_version_": 1637413024282181632}]
},
"facet_counts": {
  "facet_queries": {},
  "facet_fields": {
    "cid": [
      "560", 2433,
      "76", 640,
      "163", 65,
      "238", 9,
      "241", 4,
      "298", 3,
      "236", 2,
      "917", 2,
      "3", 1,
      "925", 1]],
    "facet_ranges": {},
    "facet_intervals": {},
    "facet_heatmaps": {}}}
```

Raw Query Parameters

facet.mincount=50

wt

☐ indent off

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☒ facet

facet.query

facet.field

cid

facet.prefix

☐ spatial

☐ spellcheck

Execute Query

```

      "title": "联想 (LENOVO) 小新S10经典版 ",
      "updated": "2017-01-11T13:09:32Z",
      "cid": 163,
      "status": 1,
      "_version_": 1637413024282181632}]
},
"facet_counts": {
  "facet_queries": {},
  "facet_fields": {
    "cid": [
      "560", 2433,
      "76", 640,
      "163", 65]],
    "facet_ranges": {},
    "facet_intervals": {},
    "facet_heatmaps": {}]}

```

价格范围

在 Raw Query Parameters 输入框中填入以下内容:

```
facet.range=price&facet.range.start=0&facet.range.end=10000&facet.range.gap=2000
```

Raw Query Parameters

facet.range=price&facet.range.start=0&f;

wt

- ☐ indent off
- ☐ debugQuery

- ☐ dismax
- ☐ edismax
- ☐ hl

☒ facet

facet.query

facet.field

facet.prefix

- ☐ spatial
- ☐ spellcheck

Execute Query


```

      "title": "联想 (Lenovo) 小新310经典版",
      "updated": "2017-01-11T13:09:32Z",
      "cid": 163,
      "status": 1,
      "_version_": 1637413024282181632}]
},
"facet_counts": {
  "facet_queries": {},
  "facet_fields": {},
  "facet_ranges": {
    "price": {
      "counts": [
        "0", 29,
        "2000", 4,
        "4000", 36,
        "6000", 54,
        "8000", 118],
      "gap": 2000,
      "start": 0,
      "end": 10000}],
    "facet_intervals": {},
    "facet_heatmaps": {}
  }
}

```

多字段统计

在 Raw Query Parameters 输入框中填入以下内容:
 facet.pivot=cid,status

Raw Query Parameters

wt

▼

☐ indent off

☐ debugQuery

☐ dismax

☐ edismax

☐ hl

☒ facet

facet.query

facet.field

facet.prefix

☐ spatial

☐ spellcheck

Execute Query

```

        title : 联想 (Lenovo) 小新310经典版 ,
        "updated": "2017-01-11T13:09:32Z",
        "cid":163,
        "status":0,
        "_version_":1637871095801446402]]
    },
    "facet_counts": {
        "facet_queries": {},
        "facet_fields": {},
        "facet_ranges": {},
        "facet_intervals": {},
        "facet_heatmaps": {},
        "facet_pivot": {
            "cid,status": [ {
                "field": "cid",
                "value": 560,
                "count": 2433,
                "pivot": [ {
                    "field": "status",
                    "value": 1,
                    "count": 1686},
                    {
                        "field": "status",
                        "value": 0,
                        "count": 747}]]},
            {
                "field": "cid",
                "value": 163,
                "count": 1637871095801446402
            }
        ]
    }
}

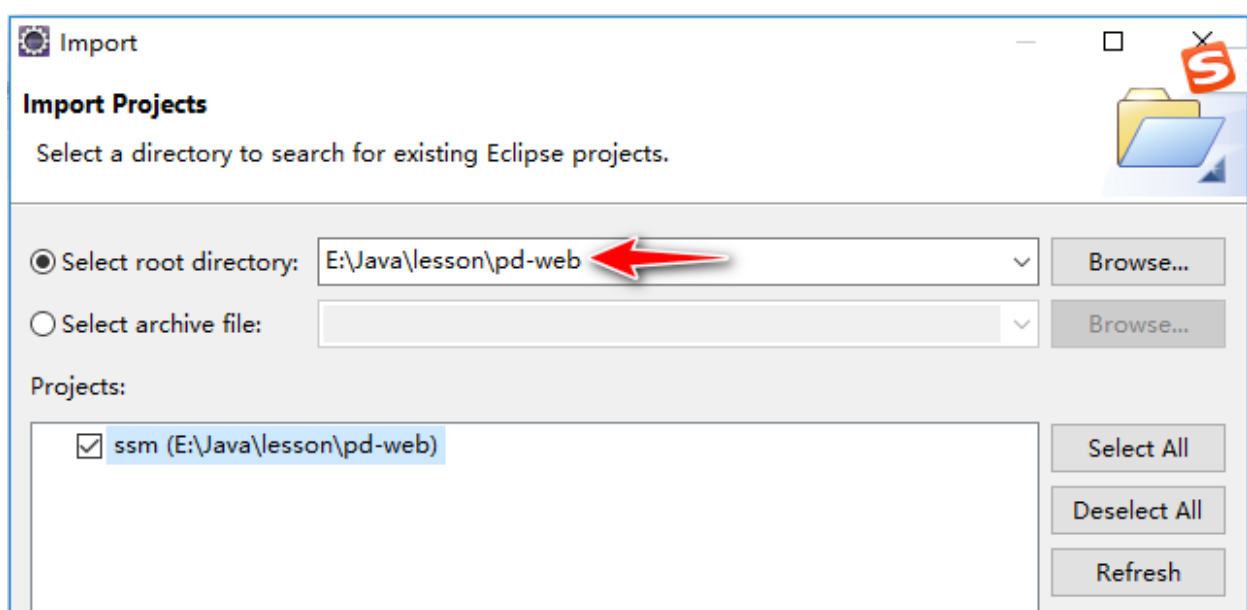
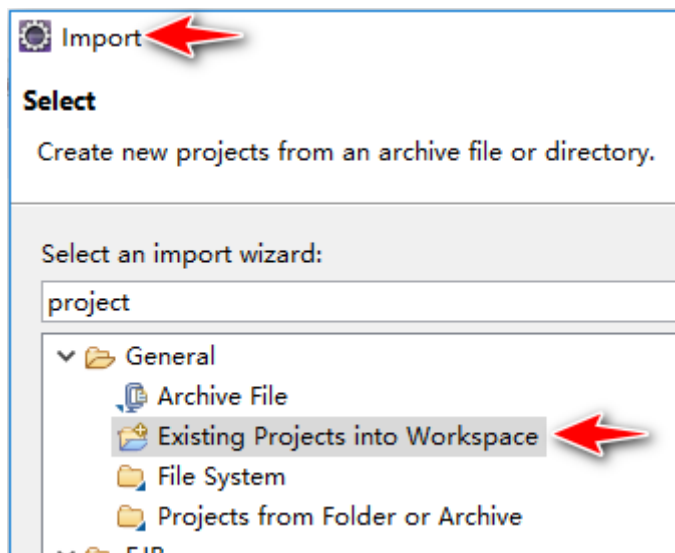
```

拼多多商城实现商品的全文检索

修改 hosts 文件, 添加 www.pd.com 映射

```
127.0.0.1    www.pd.com
```

eclipse 导入 pd-web 项目



修改数据库连接配置

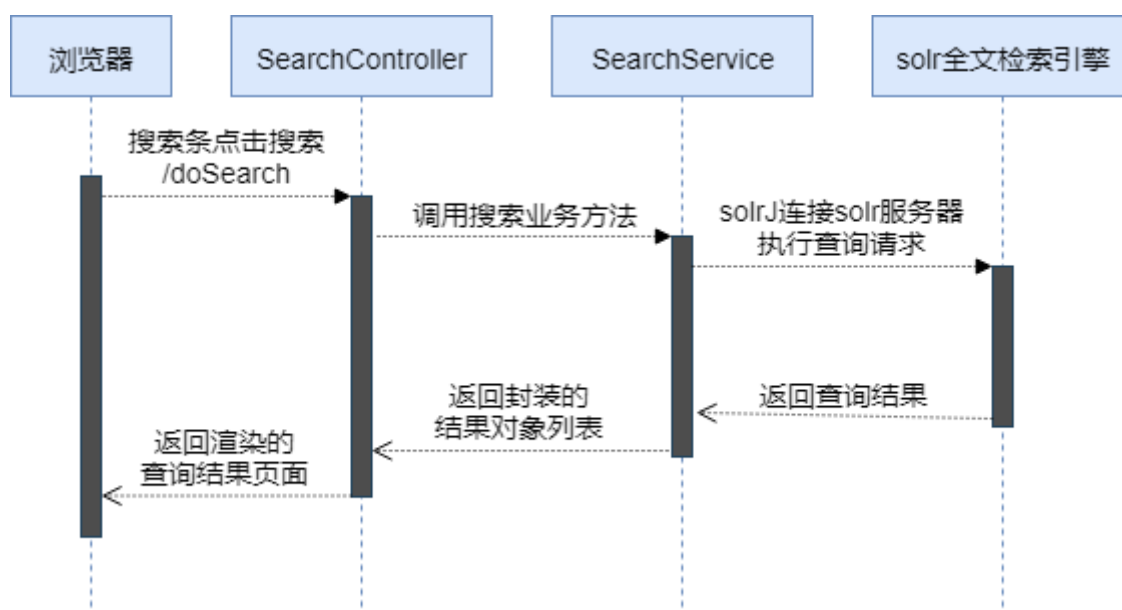
application.yml 配置文件中,修改连接配置

```
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/pd_store?useUnicode=true&characterEncoding=UTF-8
    username: root
    password: root
```

启动项目, 访问 www.pd.com



商品检索调用分析



pom.xml 添加 solr 和 lombok 依赖

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

<dependency>
  <groupId>org.projectlombok</groupId>
```

```
<artifactId>lombok</artifactId>
</dependency>
```

application.yml 添加 solr 连接信息

```
spring:
  data:
    solr:  #注意修改ip地址
      host: http://192.168.64.170:8983/solr/pd
```

Item 实体类

```
package com.pd.pojo;

import java.io.Serializable;

import org.apache.solr.client.solrj.beans.Field;

import lombok.Data;

@Data
public class Item implements Serializable {
    private static final long serialVersionUID = 1L;

    @Field("id")
    private String id;
    @Field("title")
    private String title;
    @Field("sellPoint")
    private String sellPoint;
    @Field("price")
    private Long price;
    @Field("image")
    private String image;
}
```

SearchService 业务接口

```
package com.pd.service;

import java.util.List;

import com.pd.pojo.Item;

public interface SearchService {
```

```
List<Item> findItemByKey(String key) throws Exception;  
}
```

SearchServiceImpl 业务实现类

```
package com.pd.service.impl;  
  
import java.util.List;  
  
import org.apache.solr.client.solrj.SolrClient;  
import org.apache.solr.client.solrj.SolrQuery;  
import org.apache.solr.client.solrj.response.QueryResponse;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.stereotype.Service;  
  
import com.pd.pojo.Item;  
import com.pd.service.SearchService;  
  
@Service  
public class SearchServiceImpl implements SearchService {  
  
    /*  
     * SolrClient实例是在 SolrAutoConfiguration 类中创建的  
     *  
     * SolrAutoConfiguration添加了@Configuration注解,  
     * 是spring boot自动配置类,其中的solrClient()方法中创建了SolrClient实例  
     */  
    @Autowired  
    private SolrClient solrClient;  
  
    @Override  
    public List<Item> findItemByKey(String key) throws Exception {  
        // 封装查询的关键词  
        // 也可以封装其他的查询参数,比如指定字段,facet设置等  
        SolrQuery query = new SolrQuery(key);  
        // 查询前多少条数据  
        query.setStart(0);  
        query.setRows(20);  
  
        // 执行查询并得到查询结果  
        QueryResponse qr = solrClient.query(query);  
        // 把查询结果转成一组商品实例  
        List<Item> beans = qr.getBeans(Item.class);  
        return beans;  
    }  
}
```

SearchController 控制器

```
package com.pd.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import com.pd.pojo.Item;
import com.pd.service.SearchService;

@Controller
public class SearchController {
    @Autowired
    private SearchService searchService;

    @GetMapping("/search/toSearch.html")
    public String search(String key, Model model) throws Exception {
        List<Item> itemList = searchService.findItemByKey(key);
        model.addAttribute("list", itemList);
        return "/search.jsp";
    }
}
```