lucene&solr

**第三天**

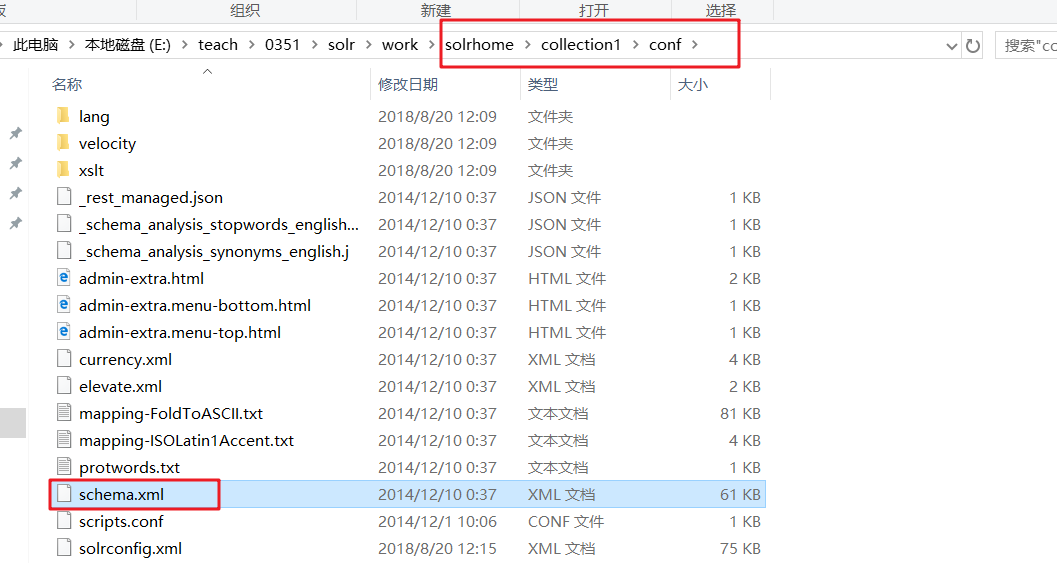
# 学习目标

* 能够安装IK分词器
* 能够完成导入数据库数据到索引库
* 能够应用solr实现复杂查询
* 能够完成JD搜索案例

# solr使用

## schema.xml

说明：schema.xml文件在solrCore/conf目录。主要用于配置域的类型（FieldType）和配置域（Field）。在solr中域要先配置再使用。



### Field标签

作用：配置域。

|  |
| --- |
| <field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" /> |

* name：域的名称
* type：域的类型
* indexed：是否索引
* stored：是否存储
* required：是否必须（在solr如果某个域指定是必须的，那么在添加索引的时候，文档对象中必须要包含该域）
* mulitValued：是否多值（在solr中可以允许一个域保存多个值，比如保存多个QQ好友Id）

### DynamicField标签

作用：配置动态域。

|  |
| --- |
| <dynamicField name="\*\_i" type="int" indexed="true" stored="true"/> |

* name：动态域的名称

说明：动态的名称是一个表达式，\*表示匹配任意字符串。在使用的时候只需要域的名称与表达式匹配即可。比如product\_i:手机，直接使用product\_i就可以，不需要在额外定义。

### UniqueKey标签

作用：指定唯一约束域，相当于关系数据库中的主键。

|  |
| --- |
| <uniqueKey>id</uniqueKey> |

说明：在solr中，id域是必须的。

### CopyField标签

作用：配置复制域。

|  |
| --- |
| <copyField source="cat" dest="text"/> |

* source：源域的名称
* dest：目标域的名称

说明：在solr中，允许将多个域的值复制给一个域，这样的目的是方便执行搜索。solr在创建索引的时候，会把源域的内容复制给目标域。（目标域的定义必须要包含一个属性：multiValued="true"）

### FieldType标签

作用：配置域的类型。

|  |
| --- |
| <fieldType name="text\_general" class="solr.TextField" positionIncrementGap="100">  <analyzer type="index">  <tokenizer class="solr.StandardTokenizerFactory"/>  <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />  <!-- in this example, we will only use synonyms at query time  <filter class="solr.SynonymFilterFactory" synonyms="index\_synonyms.txt" ignoreCase="true" expand="false"/>  -->  <filter class="solr.LowerCaseFilterFactory"/>  </analyzer>  <analyzer type="query">  <tokenizer class="solr.StandardTokenizerFactory"/>  <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />  <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>  <filter class="solr.LowerCaseFilterFactory"/>  </analyzer>  </fieldType> |

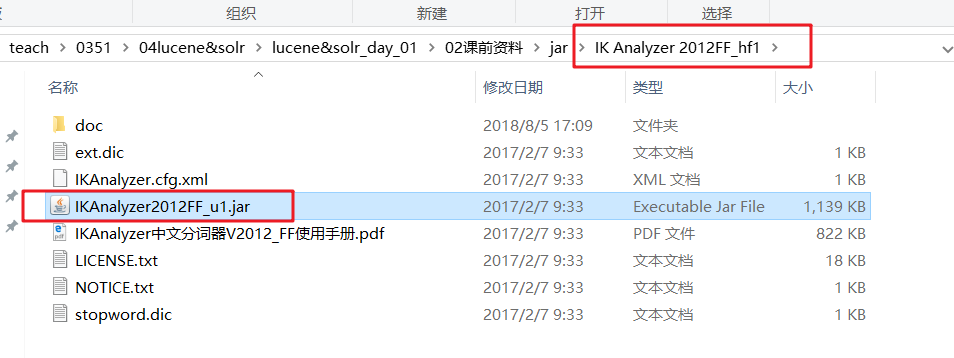
* name：域的类型名称
* class：域的类型的solr类型
* analyzer ：指定分析器，index是索引流程；query是检索流程
* tokenizer ：具体使用的分词器
* filter ：具体使用的过滤器

## 配置分词器

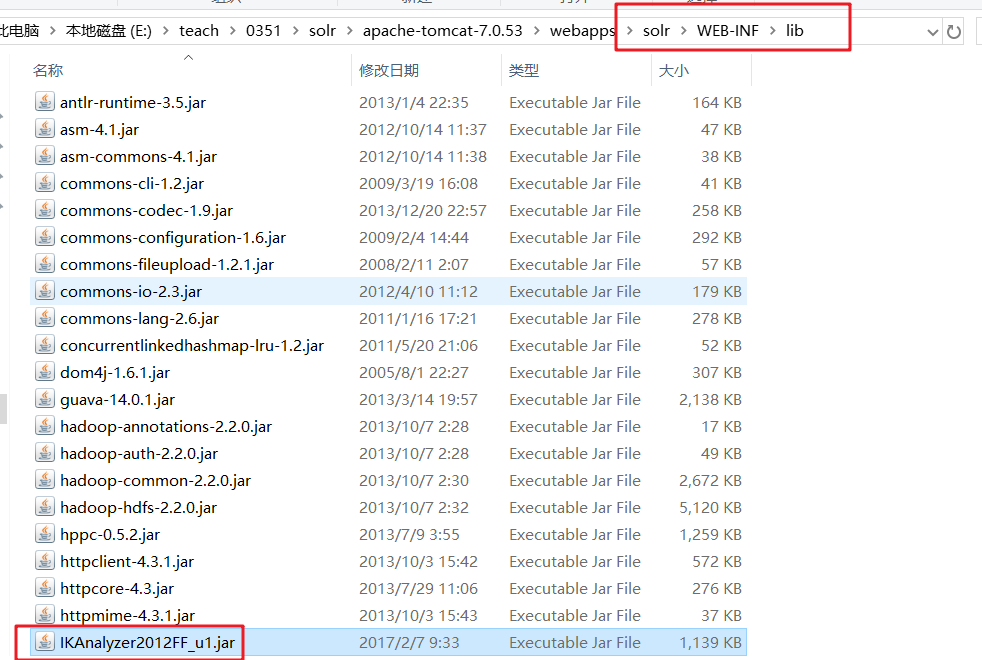
说明：配置ik中文分词器。

### 导入ik分词器jar包

复制ik分词器解压目录的jar包：

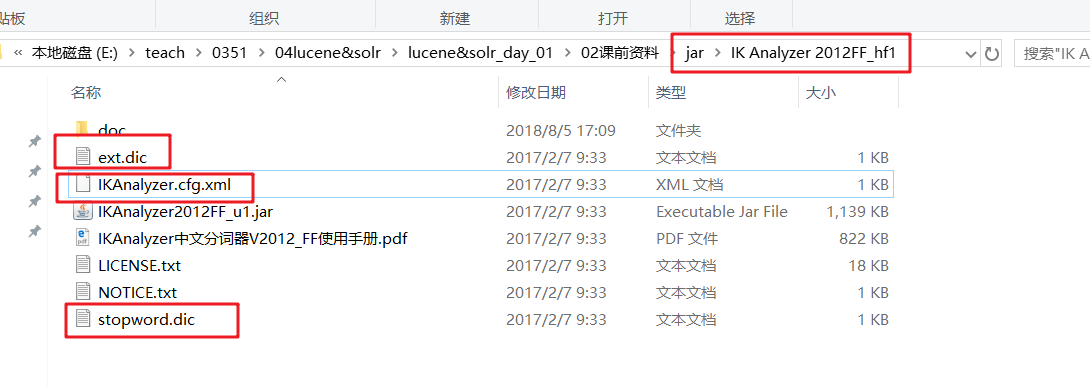


粘贴到solr/WEB-INF/lib目录：

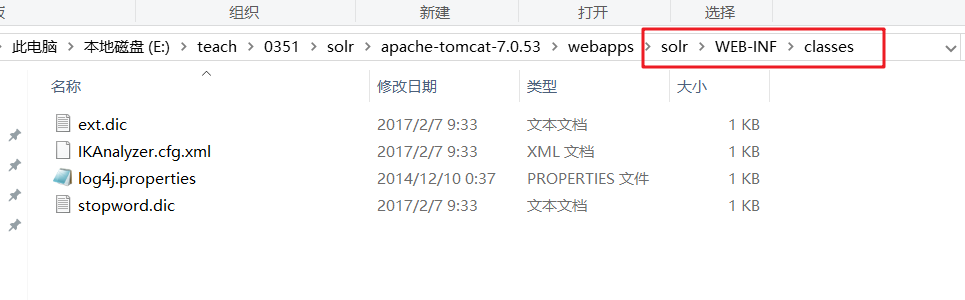


### 导入ik分词器的配置文件

复制ik解压目录中的配置文件：

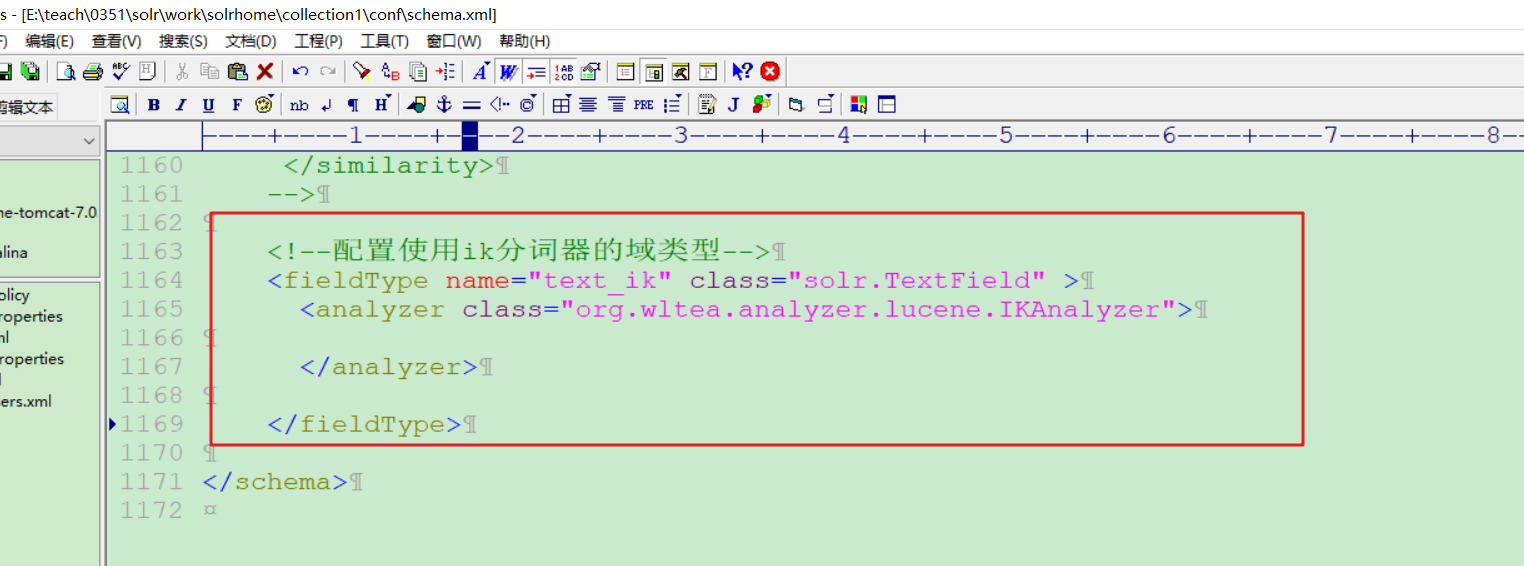


粘贴到solr/WEB-INF/classes目录：



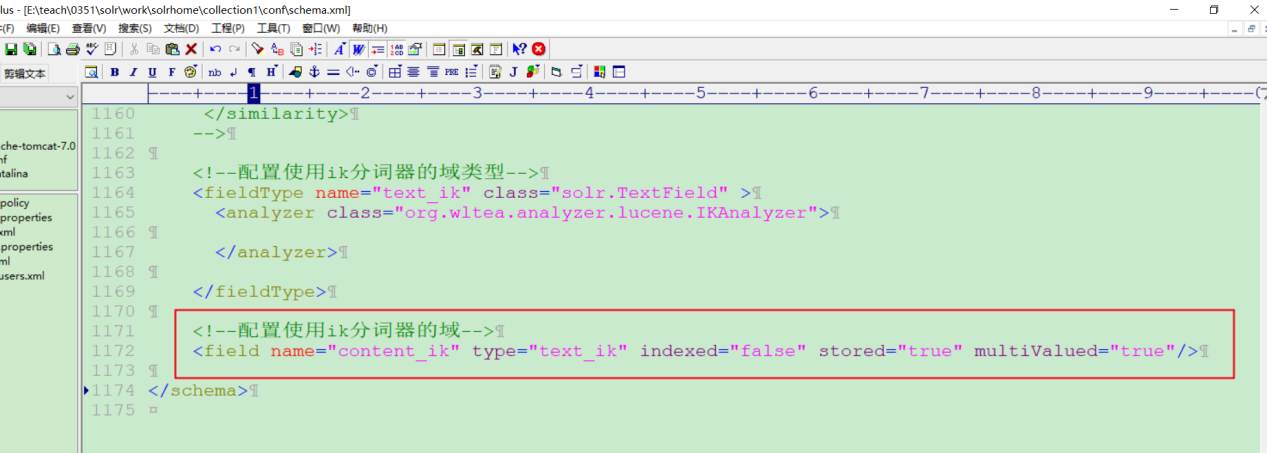
### 配置schema.xml

#### 通过FieldType标签，配置使用ik分词器的域类型



|  |
| --- |
| <!--配置使用ik分词器的域类型-->  <fieldType name="text\_ik" class="solr.TextField" >  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer">  </analyzer>  </fieldType> |

#### 通过Field标签，配置使用ik分词器的域



|  |
| --- |
| <!--配置使用ik分词器的域-->  <field name="content\_ik" type="text\_ik" indexed="false" stored="true" multiValued="true"/> |

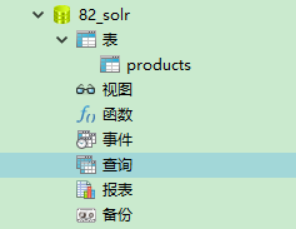
### 重新启动tomcat

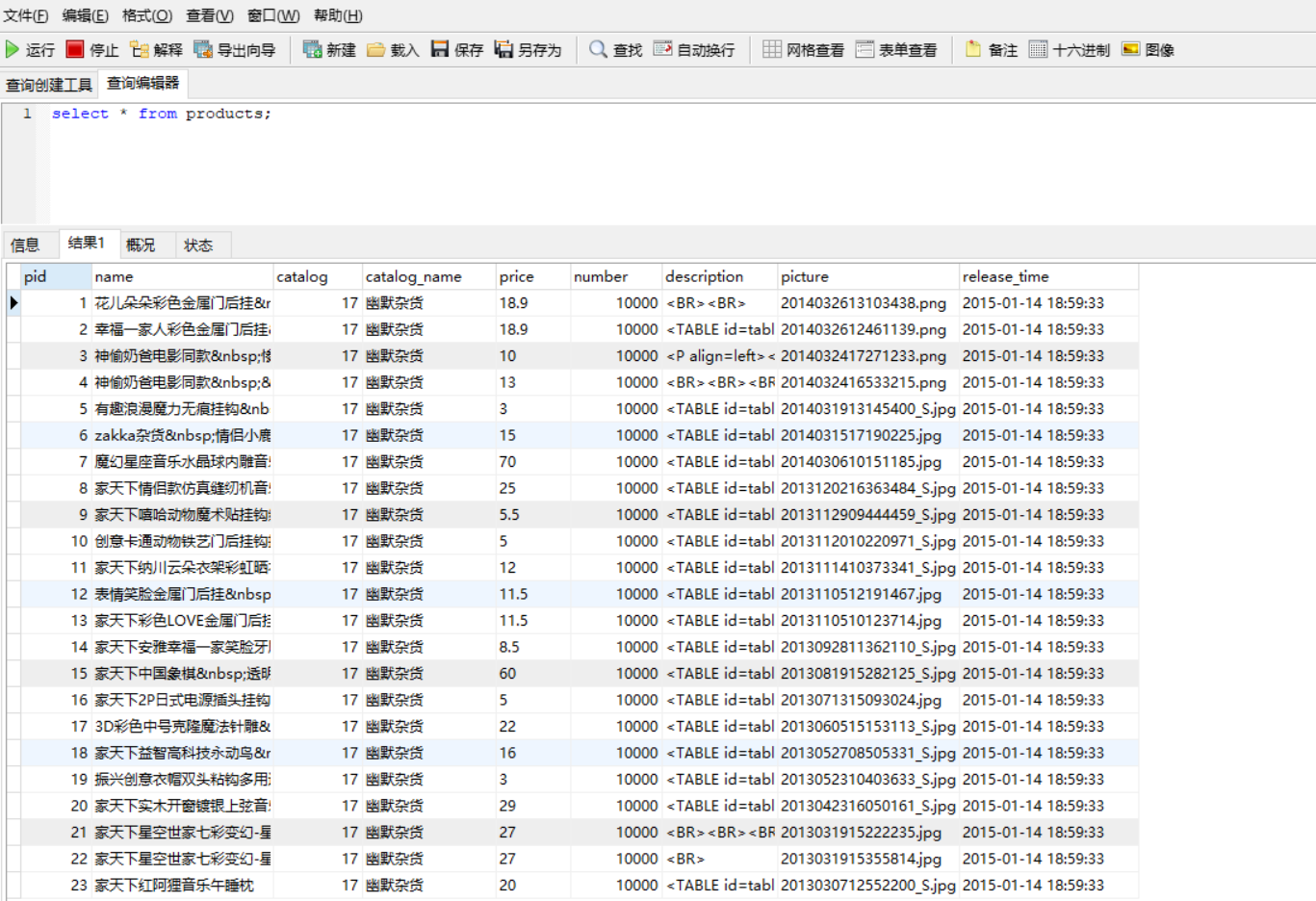


## 配置业务域

说明：使用solr实现商品数据的搜索，需要把保存在关系数据库中的商品数据，导入solr索引库。

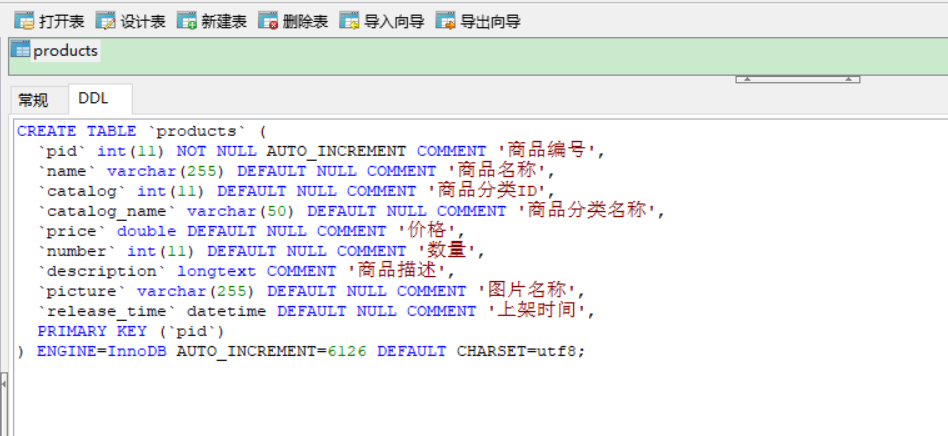
### 准备商品数据

`



### 配置商品业务域

说明：分析商品表，确定哪些字段需要在solr中建立索引和存储。



确定字段：pid，name，catalog，catalog\_name，price，description，picture

商品编号（商品编号是主键，直接solr提供的id域）：

|  |
| --- |
| <field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" /> |

商品名称：

|  |
| --- |
| <field name="product\_name" type="text\_ik" indexed="true" stored="true" /> |

商品分类id：

|  |
| --- |
| <field name="product\_catalog" type="int" indexed="true" stored="true" /> |

商品分类名称（商品分类名称不需要分词，使用string类型）：

|  |
| --- |
| <field name="product\_catalog\_name" type="string" indexed="true" stored="true" /> |

商品价格：

|  |
| --- |
| <field name="product\_price" type="double" indexed="true" stored="true" /> |

商品描述：

|  |
| --- |
| <field name="product\_description" type="text\_ik" indexed="true" stored="false" /> |

商品图片：

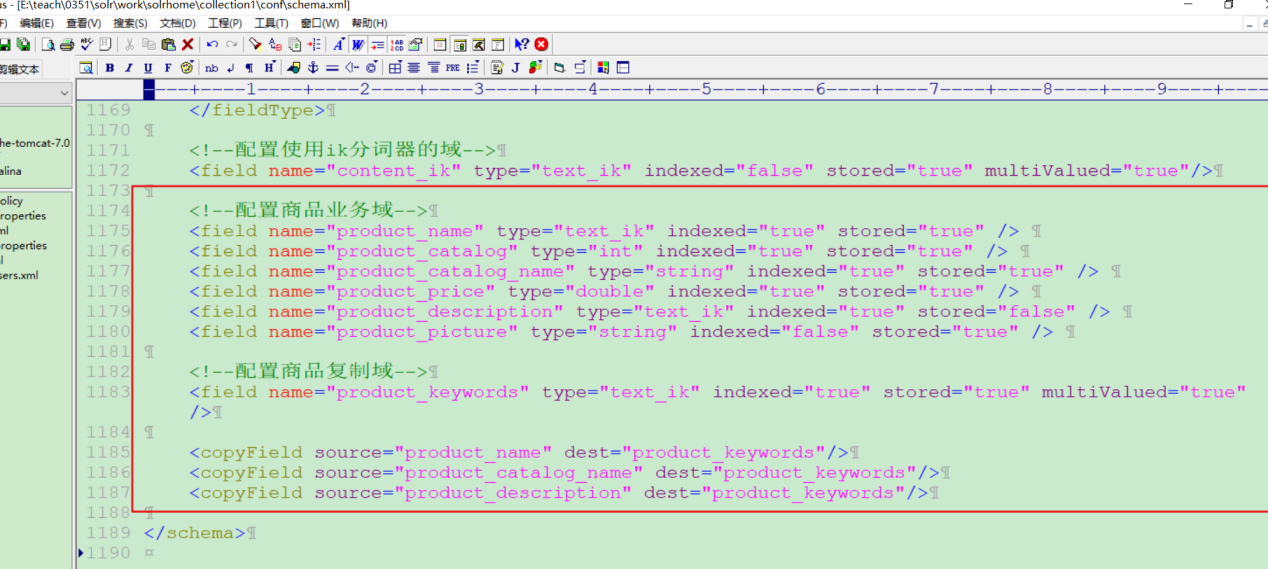
|  |
| --- |
| <field name="product\_picture" type="string" indexed="false" stored="true" /> |

定义商品复制域（在实际项目中复制域不需要存储，这里存储是为了给大家看到数据效果）：

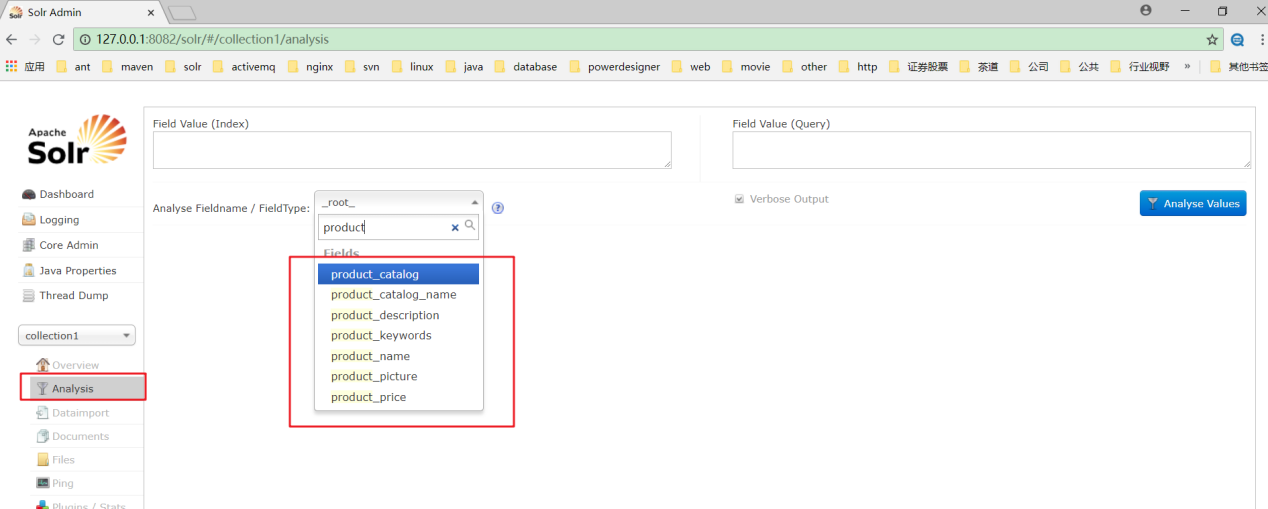
|  |
| --- |
| <field name="product\_keywords" type="text\_ik" indexed="true" stored="true" multiValued="true" /> |

|  |
| --- |
| <copyField source="product\_name" dest="product\_keywords"/>  <copyField source="product\_catalog\_name" dest="product\_keywords"/>  <copyField source="product\_description" dest="product\_keywords"/> |

### 配置schema.xml



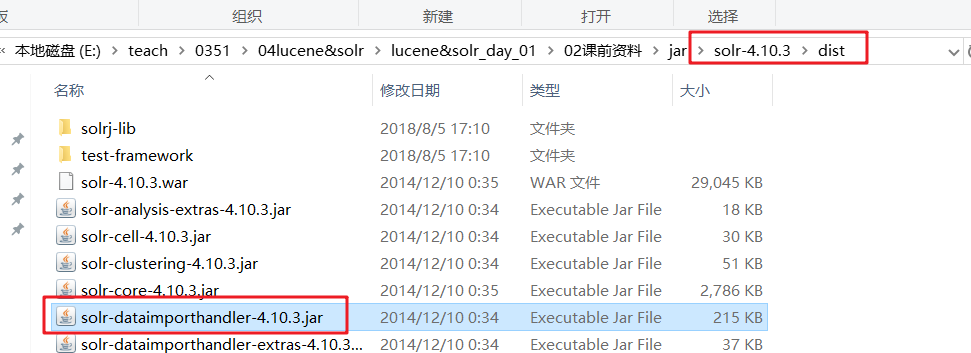
### 重新启动tomcat



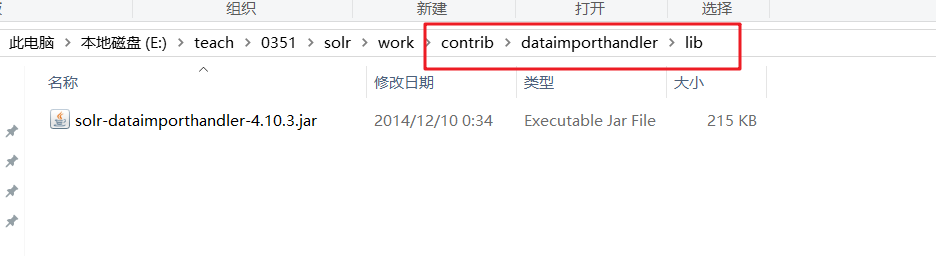
## 配置导入数据的请求处理器插件

### 导入请求处理器的插件包

复制solr解压目录dist目录中的jar包：

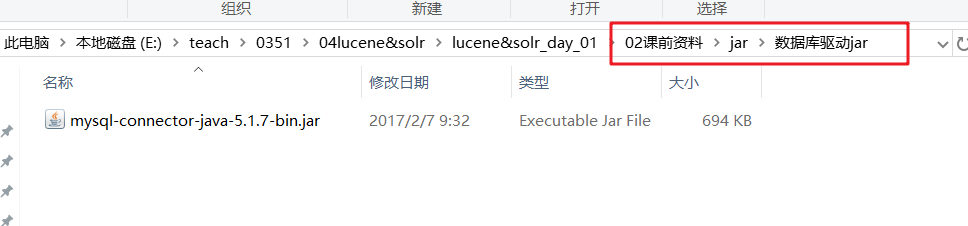


粘贴到contrib/dataimporthandler/lib（lib手动创建）：

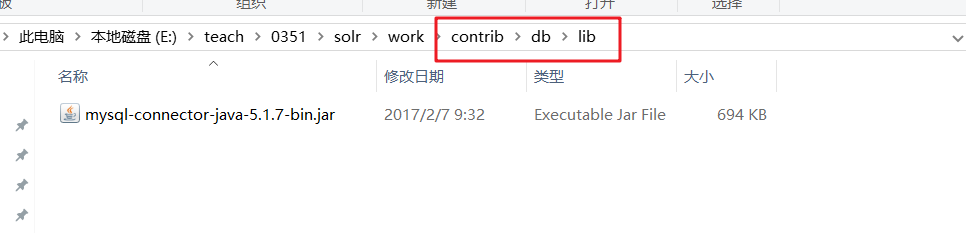


### 导入数据库的驱动包

复制课前资料中的数据库驱动包：

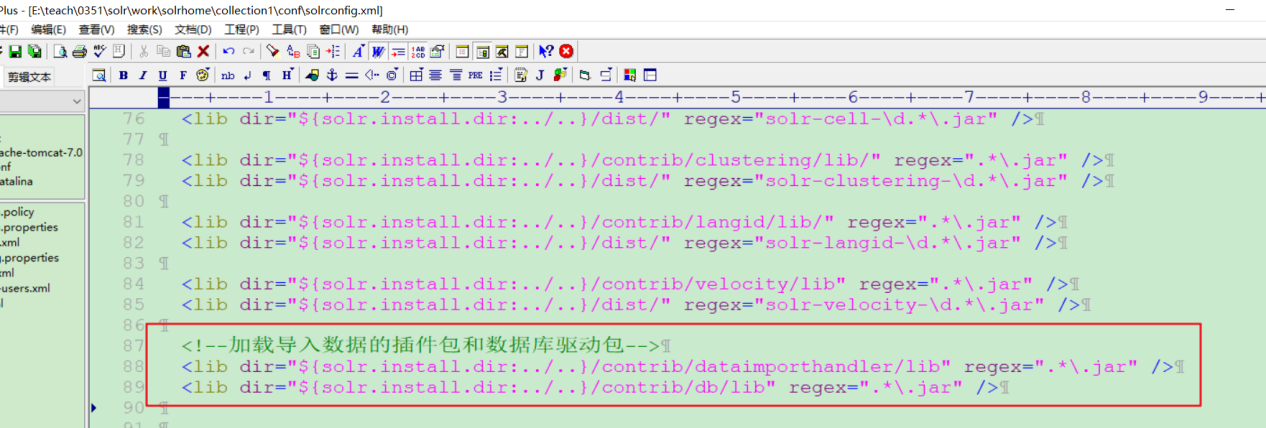


粘贴到contrib/db/lib(db/lib手动创建)：

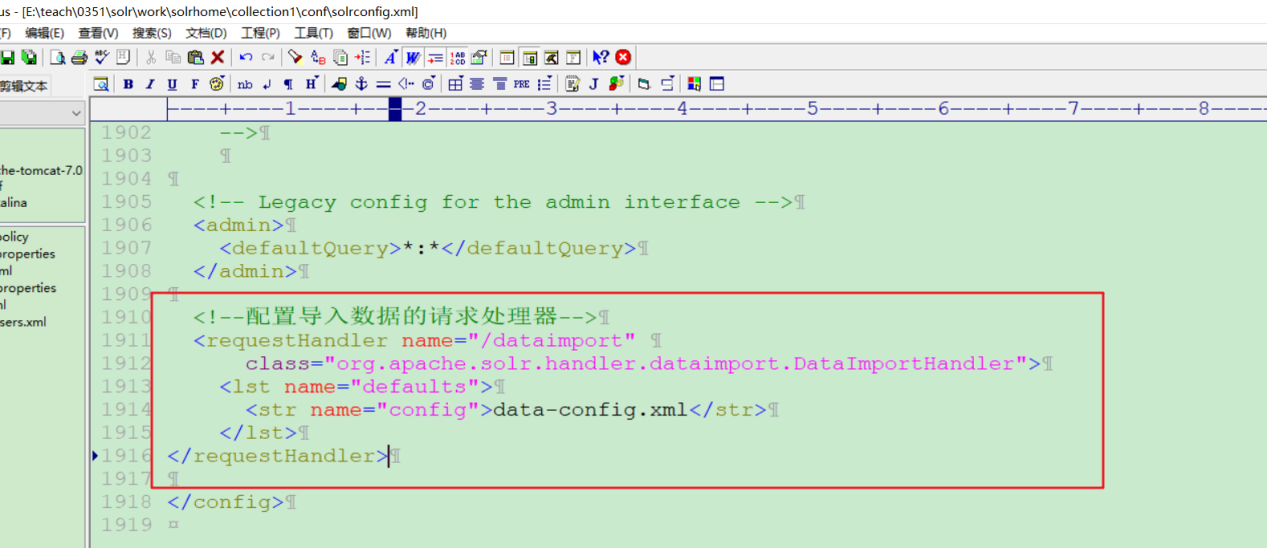


### 配置solrconfig.xml

#### 通过lib标签，加载jar包

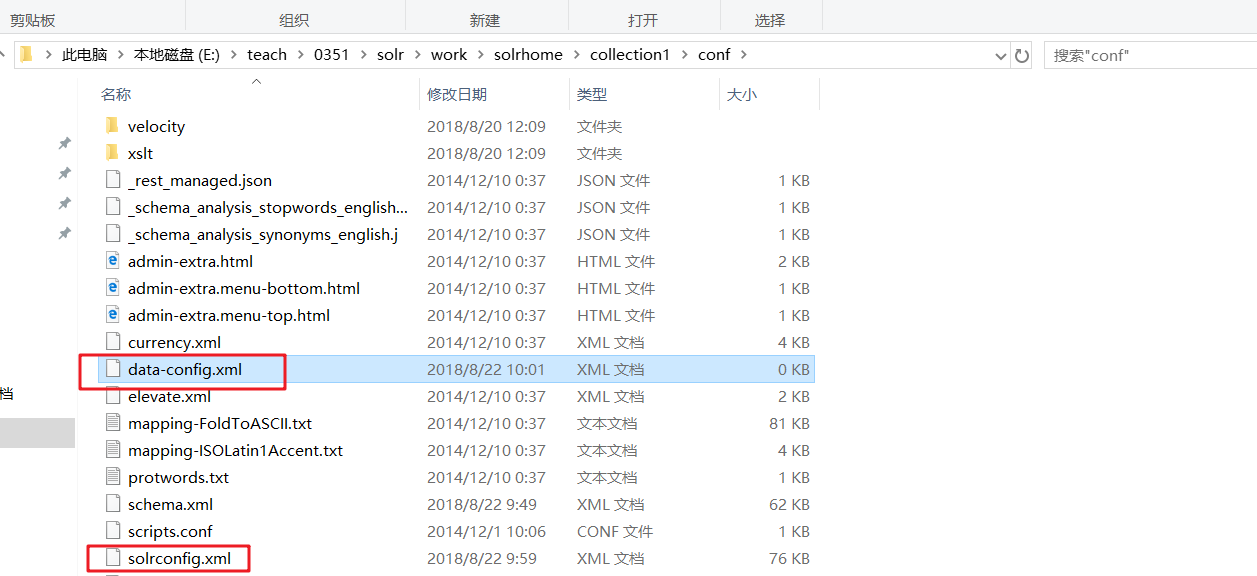


#### 通过Requesthandler标签，配置导入数据的请求处理器



|  |
| --- |
| <!--配置导入数据的请求处理器-->  <requestHandler name="/dataimport"  class="org.apache.solr.handler.dataimport.DataImportHandler">  <lst name="defaults">  <str name="config">data-config.xml</str>  </lst>  </requestHandler> |

#### 创建data-config.xml

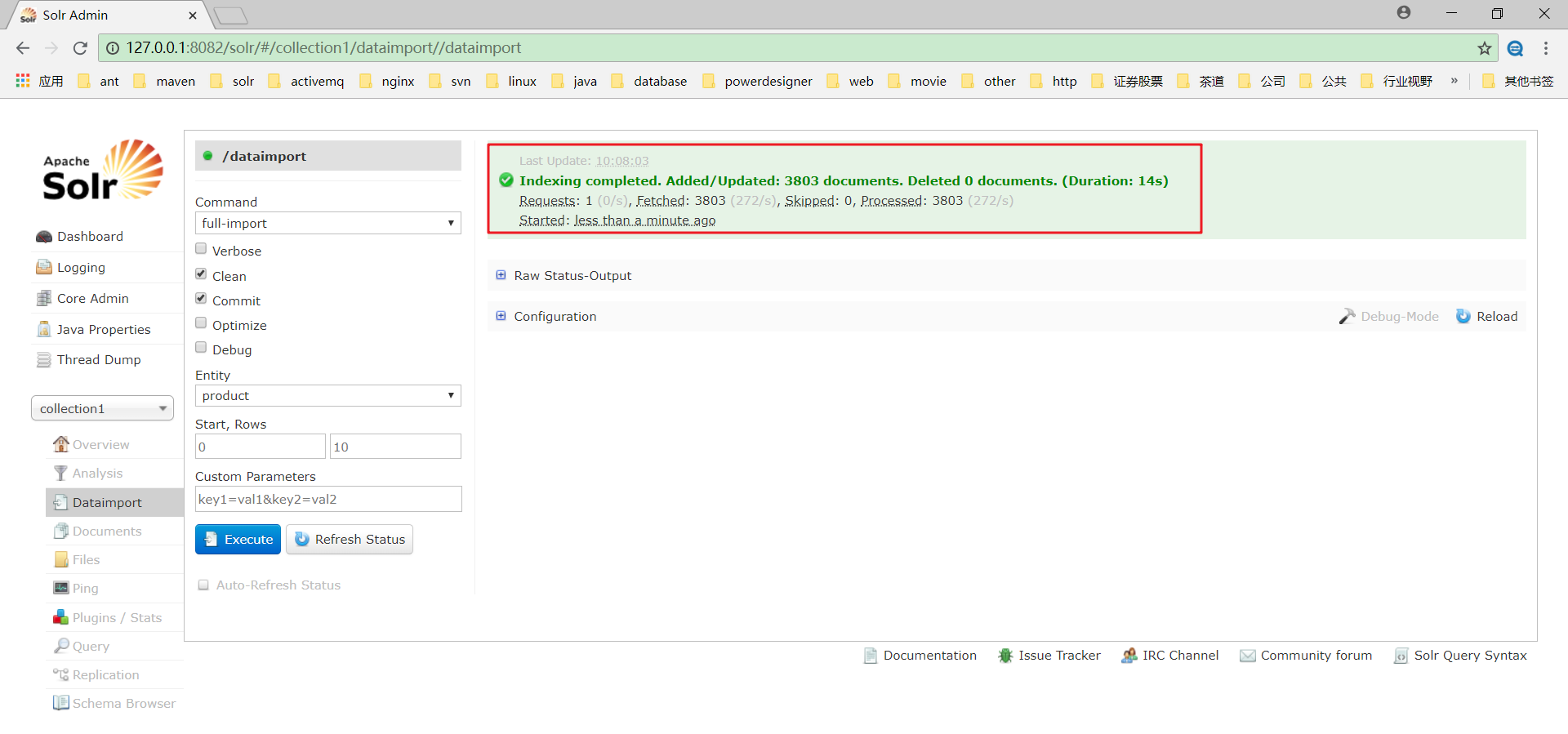


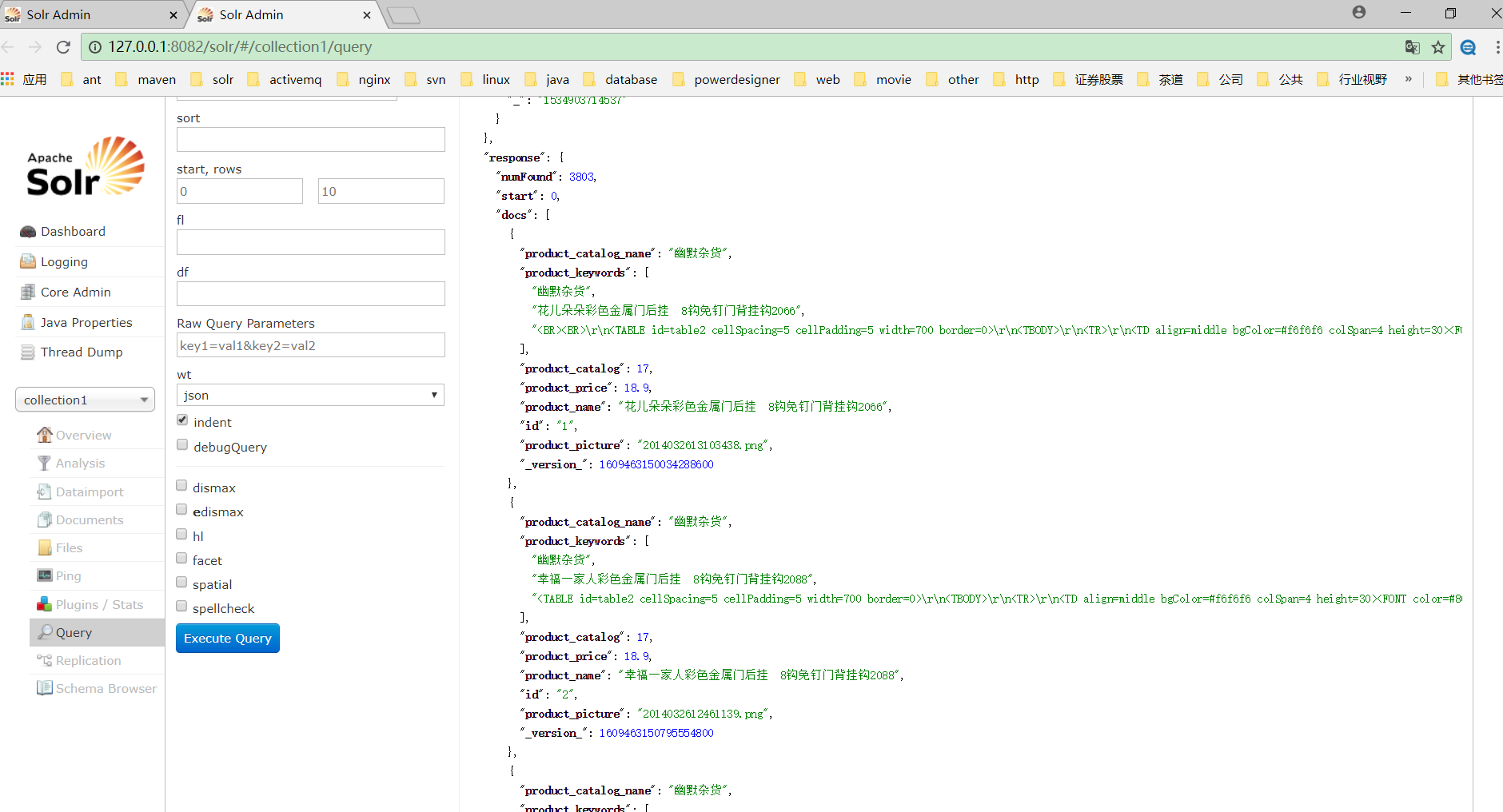
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <dataConfig>  <!--配置连接数据库信息-->  <dataSource type="JdbcDataSource"  driver="com.mysql.jdbc.Driver"  url="jdbc:mysql://127.0.0.1:3306/82\_solr"  user="root"  password="admin"/>  <document>  <!--配置sql语句中的字段，与solr索引库的域的对应关系-->  <entity name="product" query="SELECT pid,name,catalog,catalog\_name,price,description,picture FROM products">  <field column="pid" name="id"/>  <field column="name" name="product\_name"/>  <field column="catalog" name="product\_catalog"/>  <field column="catalog\_name" name="product\_catalog\_name"/>  <field column="price" name="product\_price"/>  <field column="description" name="product\_description"/>  <field column="picture" name="product\_picture"/>  </entity>  </document>  </dataConfig> |

### 重新启动tomcat



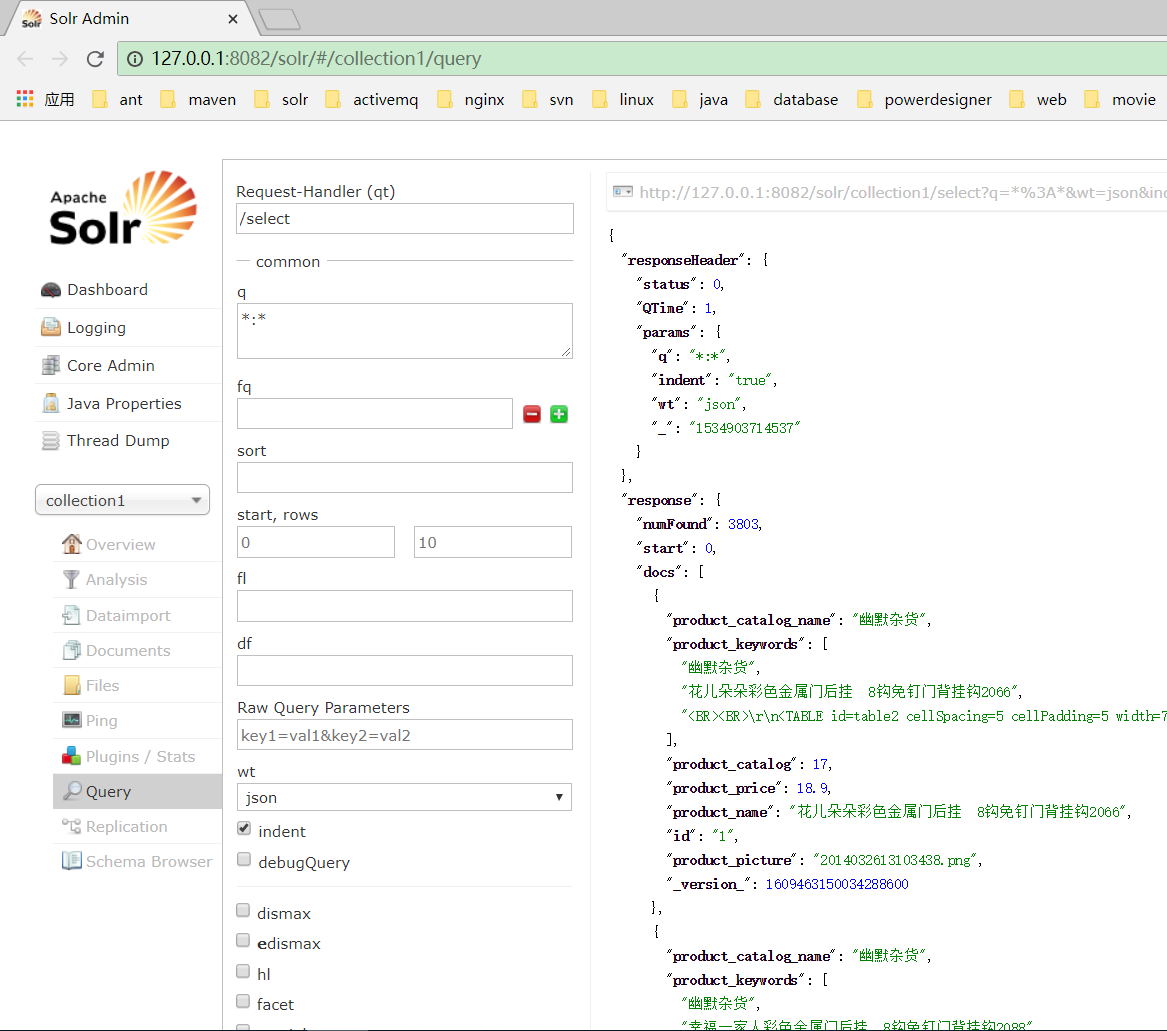
### 执行导入





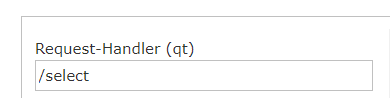
# solrj高级查询

## solr查询语法



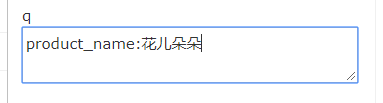
### select

说明：查询请求处理器。



### q

说明：指定搜索表达式。

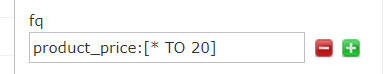


\*:\*，表示搜索全部

product\_name:花儿朵朵，表示搜索商品名称域中包含有花儿朵朵

### fq

说明：指定搜索的过滤条件。



product\_price:[\* TO 20]，表示搜索20以内的商品，包含20

product\_price:[\* TO 20}，表示搜索20以内的商品，不包含20

说明：

1. fq是一个数组，可以指定多个
2. 使用fq，前提是先使用q查询出结果

### sort

说明：指定搜索结果排序。

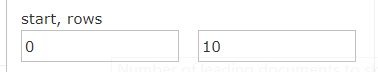


product\_price asc，表示按照商品价格升序排序

product\_price desc，表示按照商品价格降序排序

### start，rows

说明：指定分页。



start：开始索引

rows：页面大小

### fl

说明：指定搜索结果显示的域列表



id,product\_name,product\_price,product\_catalog，显示商品id、商品名称、商品价格、商品分类id

### df

说明：指定默认搜索域。



注意事项：默认搜索域只能指定一个。

### wt

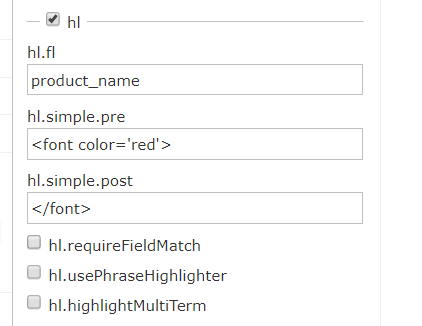
说明：指定搜索结果的响应格式



常用格式：json/xml/csv。

### hl

说明：指定高亮显示



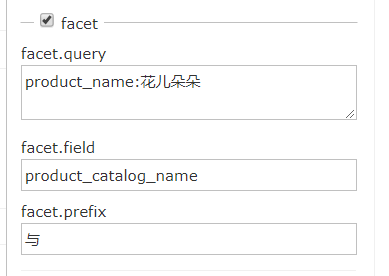
hl.fl：指定高亮显示的域名称

hl.simple.pre：指定高亮显示的html标签的开始部分

hl.simple.post：指定高亮显示的html标签的结束部分

### facet

说明：分片统计，相当于关系数据库中的group by 子句



facet.query：指定统计条件

facet.field：指定分组统计的域

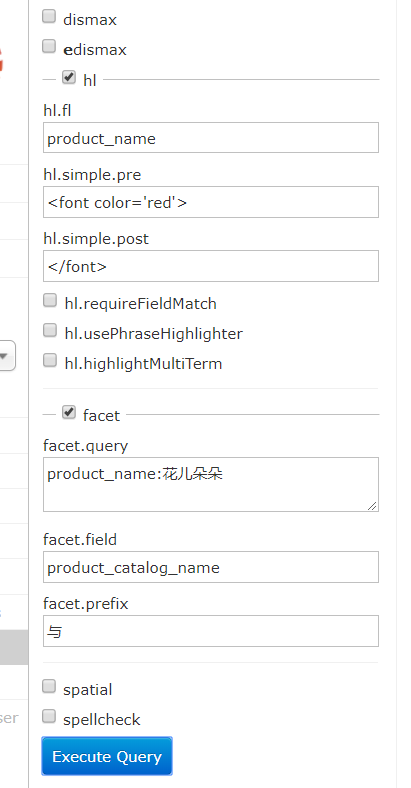
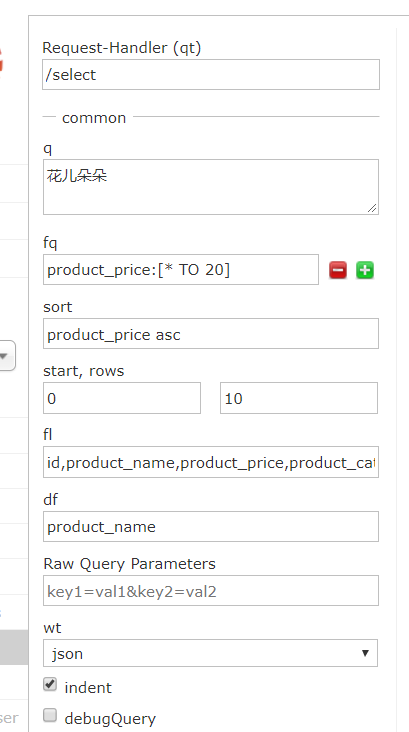
facet.prefix：指定组过滤条件

分组统计的应用场景：

|  |
| --- |
| 电子产品（5000） 服装（80000） 食品（200） |

## 代码实现

说明：使用代码实现solr界面的查询效果。要求搜索参数与搜索结果跟页面一致。



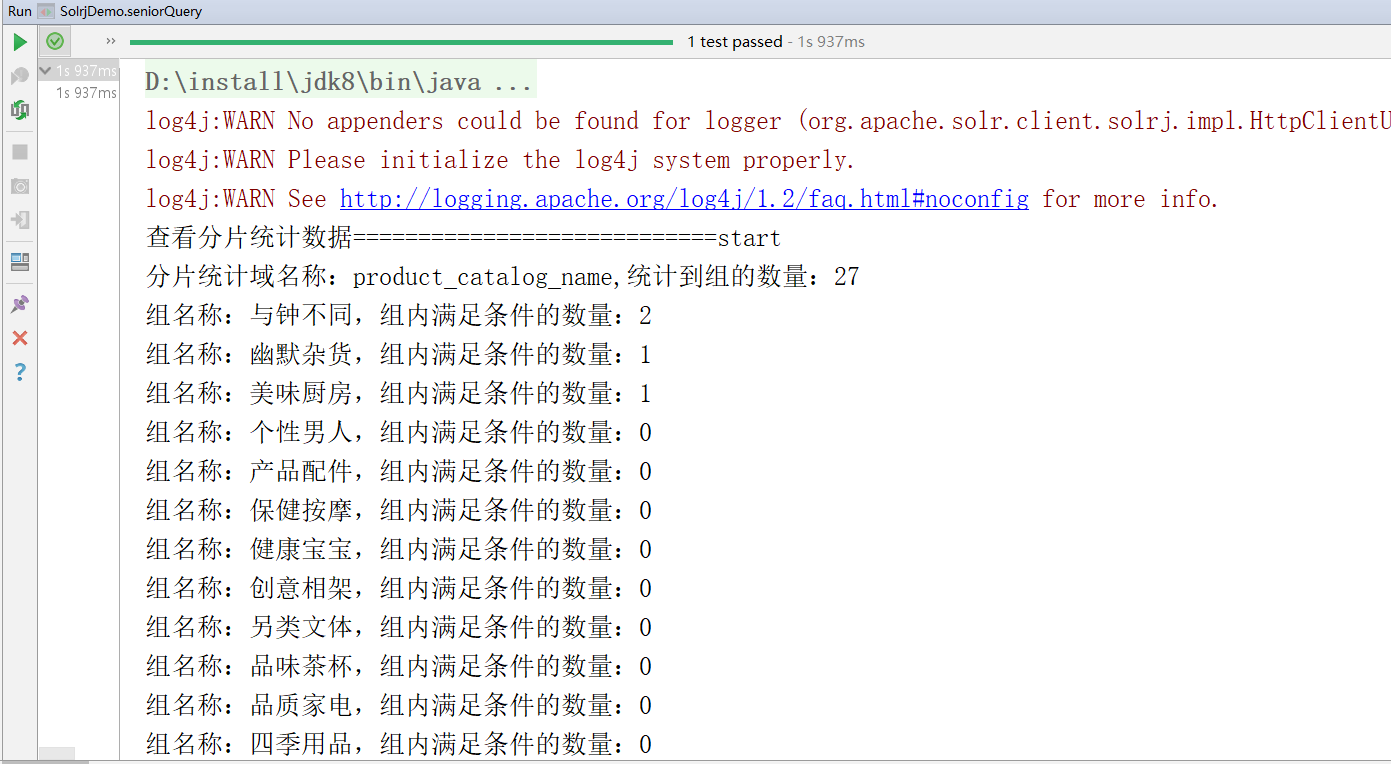
1. 建立HttpSolrServer对象，连接solr服务
2. 建立查询对象（SolrQuery）
3. 使用HttpSolrServer对象执行搜索，返回QueryResponse
4. 从QueryResponse中获取搜索结果集（SolrDocumentList）
5. 处理结果集

|  |
| --- |
| */\*\*  \* lucene&solr第三天课程内容，solrj高级查询============================  \*/* @Test  **public void** seniorQuery() **throws** Exception{ *// 1.建立HttpSolrServer对象，连接solr服务* HttpSolrServer server = **new** HttpSolrServer(**"http://127.0.0.1:8082/solr"**); *// 2.建立查询对象（SolrQuery）* SolrQuery sq = **new** SolrQuery();  *// 设置查询表达式q* sq.setQuery(**"花儿朵朵"**);  *// 设置过滤条件fq* sq.setFilterQueries(**"product\_price:[\* TO 20]"**);  *// 设置排序sort* sq.setSort(**"product\_price"**, SolrQuery.ORDER.***asc***);  *// 设置分页 start rows* sq.setStart(0);  sq.setRows(10);   *// 设置显示域列表fl  // id,product\_name,product\_price,product\_catalog* sq.setFields(**"id"**,**"product\_name"**,**"product\_price"**,**"product\_catalog"**);  *// 设置默认搜索域df* sq.set(**"df"**,**"product\_name"**);  *// 设置响应格式wt* sq.set(**"wt"**,**"json"**);  *// 设置高亮hl* sq.setHighlight(**true**);*//开启高亮显示* sq.addHighlightField(**"product\_name"**);*// 添加高亮显示域* sq.setHighlightSimplePre(**"<font color='red'>"**);*// 设置高亮显示的html标签的开始部分* sq.setHighlightSimplePost(**"</font>"**);*// 设置高亮显示的html标签的结束部分   // 设置分片统计facet* sq.setFacet(**true**);*// 开启分片统计* sq.addFacetField(**"product\_catalog\_name"**);*//添加分片统计的域的名称  // 3.使用HttpSolrServer对象执行搜索，返回QueryResponse* QueryResponse queryResponse = server.query(sq);  *// 4.从QueryResponse中获取搜索结果集（SolrDocumentList）  // 4.1.取出搜索结果集* SolrDocumentList results = queryResponse.getResults();  *// 4.2.获取高亮数据  /\*\*  \* "highlighting": {  "1": {  "product\_name": [  "<font color='red'>花儿</font><font color='red'>朵朵</font>彩色金属门后挂 8钩免钉门背挂钩2066"  ]  },  \*/* Map<String, Map<String, List<String>>> highlighting = queryResponse.getHighlighting();   *// 4.3.获取分片统计的数据* List<FacetField> facetFields = queryResponse.getFacetFields();  *// 5.处理结果集  // 5.1.实际搜索到的结果数量* System.***out***.println(**"实际搜索到的结果数量："**+results.getNumFound());  *// 5.2.打印文档内容* **for**(SolrDocument doc:results){  System.***out***.println(**"--------------华丽丽分割线------------------------"**);  *// 商品Id，商品名称，商品价格，商品分类Id  // 商品Id* String pid = doc.get(**"id"**).toString();   *// 商品名称* String pname = **""**;  List<String> list = highlighting.get(pid).get(**"product\_name"**);  **if**(list != **null** &&  list.size()>0){  pname = list.get(0);  }**else**{  pname = doc.get(**"product\_name"**).toString();  }   *// 商品价格* String pprice = doc.get(**"product\_price"**).toString();   *// 商品分类Id* String pcatalog = doc.get(**"product\_catalog"**).toString();   System.***out***.println(**"商品Id："**+pid);  System.***out***.println(**"商品名称："**+pname);  System.***out***.println(**"商品价格："**+pprice);  System.***out***.println(**"商品分类Id："**+pcatalog);   }  } |



查看分片统计数据：

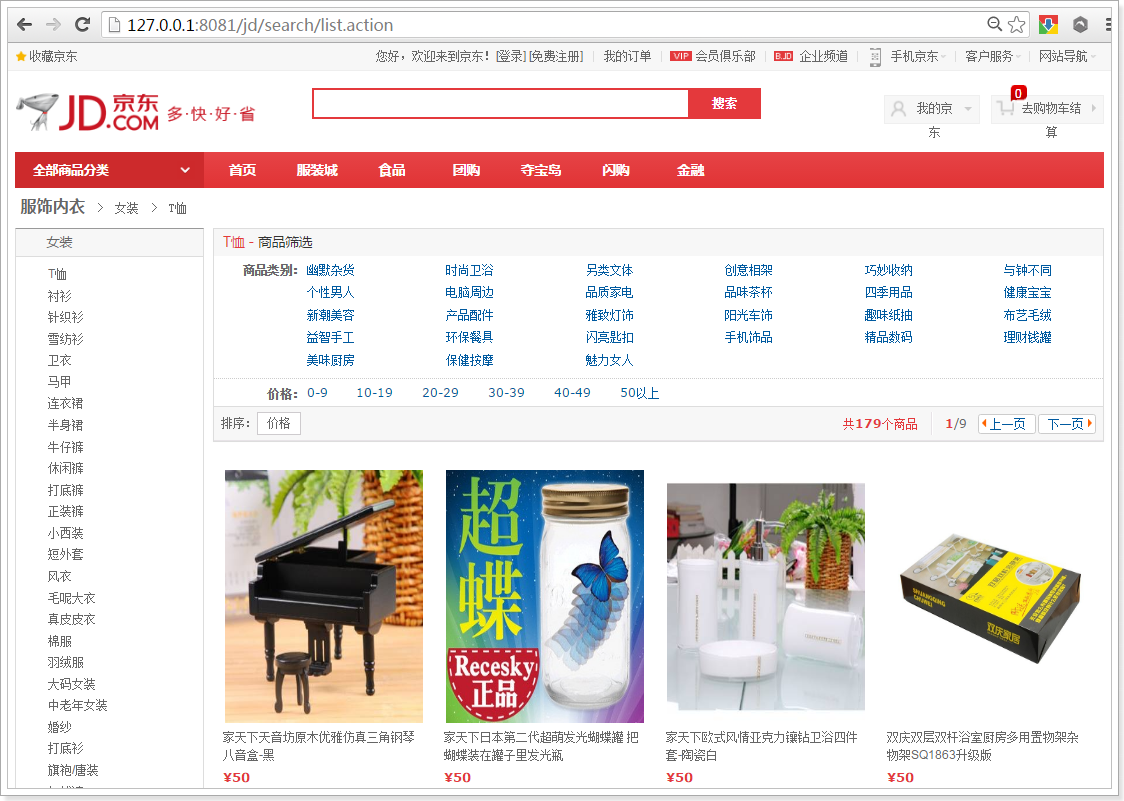
|  |
| --- |
| */\*\*  \* lucene&solr第三天课程内容，solrj高级查询============================  \*/* @Test  **public void** seniorQuery() **throws** Exception{ *// 1.建立HttpSolrServer对象，连接solr服务* HttpSolrServer server = **new** HttpSolrServer(**"http://127.0.0.1:8082/solr"**); *// 2.建立查询对象（SolrQuery）* SolrQuery sq = **new** SolrQuery();  *// 设置查询表达式q* sq.setQuery(**"花儿朵朵"**);  *// 设置过滤条件fq* sq.setFilterQueries(**"product\_price:[\* TO 20]"**);  *// 设置排序sort* sq.setSort(**"product\_price"**, SolrQuery.ORDER.***asc***);  *// 设置分页 start rows* sq.setStart(0);  sq.setRows(10);   *// 设置显示域列表fl  // id,product\_name,product\_price,product\_catalog* sq.setFields(**"id"**,**"product\_name"**,**"product\_price"**,**"product\_catalog"**);  *// 设置默认搜索域df* sq.set(**"df"**,**"product\_name"**);  *// 设置响应格式wt* sq.set(**"wt"**,**"json"**);  *// 设置高亮hl* sq.setHighlight(**true**);*//开启高亮显示* sq.addHighlightField(**"product\_name"**);*// 添加高亮显示域* sq.setHighlightSimplePre(**"<font color='red'>"**);*// 设置高亮显示的html标签的开始部分* sq.setHighlightSimplePost(**"</font>"**);*// 设置高亮显示的html标签的结束部分   // 设置分片统计facet* sq.setFacet(**true**);*// 开启分片统计* sq.addFacetField(**"product\_catalog\_name"**);*//添加分片统计的域的名称  // 3.使用HttpSolrServer对象执行搜索，返回QueryResponse* QueryResponse queryResponse = server.query(sq);  *// 4.从QueryResponse中获取搜索结果集（SolrDocumentList）  // 4.1.取出搜索结果集* SolrDocumentList results = queryResponse.getResults();  *// 4.2.获取高亮数据  /\*\*  \* "highlighting": {  "1": {  "product\_name": [  "<font color='red'>花儿</font><font color='red'>朵朵</font>彩色金属门后挂 8钩免钉门背挂钩2066"  ]  },  \*/* Map<String, Map<String, List<String>>> highlighting = queryResponse.getHighlighting();   *// 4.3.获取分片统计的数据* List<FacetField> facetFields = queryResponse.getFacetFields();  *// 查看分片统计数据============================start* System.***out***.println(**"查看分片统计数据============================start"**);  **for**(FacetField f:facetFields){  System.***out***.println(**"分片统计域名称："**+f.getName()+**",统计到组的数量："**+f.getValueCount());  *// 查看组内信息* List<FacetField.Count> values = f.getValues();  **for**(FacetField.Count c:values){  System.***out***.println(**"组名称："**+c.getName()+**"，组内满足条件的数量："**+c.getCount());  }  }   System.***out***.println(**"查看分片统计数据============================end"**);  *// 查看分片统计数据============================end // 5.处理结果集  // 5.1.实际搜索到的结果数量* System.***out***.println(**"实际搜索到的结果数量："**+results.getNumFound());  *// 5.2.打印文档内容* **for**(SolrDocument doc:results){  System.***out***.println(**"--------------华丽丽分割线------------------------"**);  *// 商品Id，商品名称，商品价格，商品分类Id  // 商品Id* String pid = doc.get(**"id"**).toString();   *// 商品名称* String pname = **""**;  List<String> list = highlighting.get(pid).get(**"product\_name"**);  **if**(list != **null** &&  list.size()>0){  pname = list.get(0);  }**else**{  pname = doc.get(**"product\_name"**).toString();  }   *// 商品价格* String pprice = doc.get(**"product\_price"**).toString();   *// 商品分类Id* String pcatalog = doc.get(**"product\_catalog"**).toString();   System.***out***.println(**"商品Id："**+pid);  System.***out***.println(**"商品名称："**+pname);  System.***out***.println(**"商品价格："**+pprice);  System.***out***.println(**"商品分类Id："**+pcatalog);   }  } |



# 京东案例

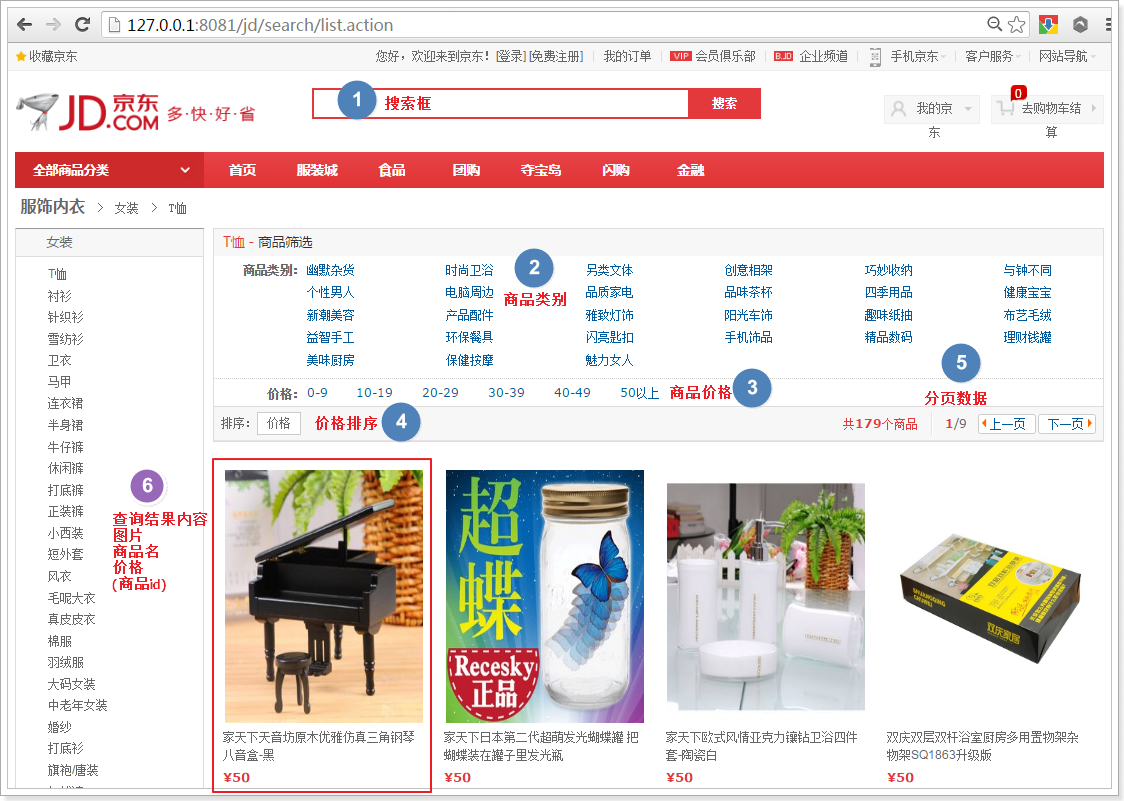
## 需求

使用solr实现电商网站中商品信息搜索功能。根据关键字搜索商品信息，根据商品分类名称，商品价格过滤搜索结果。根据价格进行排序，实现分页。界面如下：



## 分析

### UI分析



### 项目架构分析

应用服务器：

表现层：springmvc

业务层：serivce

持久层：dao

数据库：mysql

搜索服务器：

部署solr服务

索引库

从service发起索引和搜索请求

## 准备项目环境

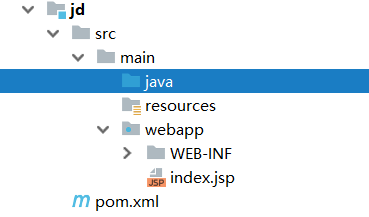
jdk：1.8

ide：idea

solr：4.10.3

web容器：tomcat

## 创建项目



## 配置pom.xml，导入依赖包

* spring（包括springmvc）
* solrj客户端
* jstl标签库包
* log4j.properties

|  |
| --- |
| *<?***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**>cn.itheima</**groupId**>  <**artifactId**>jd</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  <**packaging**>war</**packaging**>   <**name**>jd Maven Webapp</**name**>  <**url**>http://www.example.com</**url**>   <**properties**>  *<!-- spring版本号 -->* <**spring.version**>4.3.8.RELEASE</**spring.version**>  *<!-- jstl标签版本 -->* <**jstl.version**>1.2</**jstl.version**>  *<!-- commons-lang版本 -->* <**commons.lang.version**>2.6</**commons.lang.version**>  *<!-- solrj版本 -->* <**solrj.version**>4.10.3</**solrj.version**>  *<!-- log4j日志版本 -->* <**slf4j.version**>1.7.7</**slf4j.version**>  <**log4j.version**>1.2.17</**log4j.version**>  *<!-- jcl版本 -->* <**jcl.version**>1.7.6</**jcl.version**>  </**properties**>   <**dependencies**>  *<!-- springmvc依赖包 -->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aop</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- JSTL标签类 -->* <**dependency**>  <**groupId**>jstl</**groupId**>  <**artifactId**>jstl</**artifactId**>  <**version**>${jstl.version}</**version**>  </**dependency**>   *<!-- solrj依赖 -->* <**dependency**>  <**groupId**>org.apache.solr</**groupId**>  <**artifactId**>solr-solrj</**artifactId**>  <**version**>${solrj.version}</**version**>  </**dependency**>  *<!-- log4j日志包 -->* <**dependency**>  <**groupId**>log4j</**groupId**>  <**artifactId**>log4j</**artifactId**>  <**version**>${log4j.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-api</**artifactId**>  <**version**>${slf4j.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>slf4j-log4j12</**artifactId**>  <**version**>${slf4j.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>jcl-over-slf4j</**artifactId**>  <**version**>${jcl.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.slf4j</**groupId**>  <**artifactId**>jul-to-slf4j</**artifactId**>  <**version**>${jcl.version}</**version**>  </**dependency**>  <**dependency**>  <**groupId**>commons-lang</**groupId**>  <**artifactId**>commons-lang</**artifactId**>  <**version**>${commons.lang.version}</**version**>  </**dependency**>  *<!-- jsp依赖包，只在编译时需要 -->* <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>jsp-api</**artifactId**>  <**version**>2.0</**version**>  <**scope**>provided</**scope**>  </**dependency**>  </**dependencies**>   <**build**>  <**finalName**>jd</**finalName**>  <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.1</**version**>  <**configuration**>  *<!-- tomcat 的端口号 -->* <**port**>8080</**port**>  *<!-- 访问应用的路径 -->* <**path**>/jd</**path**>  *<!-- URL按UTF-8进行编码，解决中文参数乱码 -->* <**uriEncoding**>UTF-8</**uriEncoding**>  *<!-- tomcat名称 -->* <**server**>tomcat7</**server**>  </**configuration**>  </**plugin**>  </**plugins**>  </**build**> </**project**> |

## 编写配置文件

### springmvc.xml

* 配置包扫描controller
* 配置处理器映射器
* 配置处理器适配器
* 配置视图解析器

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd"**>   *<!--配置包扫描controller-->* <**context:component-scan base-package="cn.itheima.jd.controller"**/>   *<!--注解驱动方式配置处理器映射器和适配器-->* <**mvc:annotation-driven**/>   *<!--配置视图解析器-->* <**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  *<!--配置视图的公共目录路径（前缀）-->* <**property name="prefix" value="/WEB-INF/jsp/"**/>  *<!--配置视图的扩展名称（后缀）-->* <**property name="suffix" value=".jsp"**/>  </**bean**>   </**beans**> |

### applicationContext-service.xml

* 配置包扫描service

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>   *<!--配置包扫描service-->* <**context:component-scan base-package="cn.itheima.jd.service"**/>  </**beans**> |

### applicationContext-solr.xml

* 配置HttpSolrServer对象

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd"**>   *<!--配置HttpSolrServer对象-->* <**bean id="httpSolrServer" class="org.apache.solr.client.solrj.impl.HttpSolrServer"**>  *<!--注入solr服务器地址-->* <**constructor-arg name="baseURL" value="http://127.0.0.1:8082/solr"**/>  </**bean**>  </**beans**> |

### log4j.properties

|  |
| --- |
| # Global logging configuration  log4j.rootLogger=INFO, stdout  # Console output...  log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n |

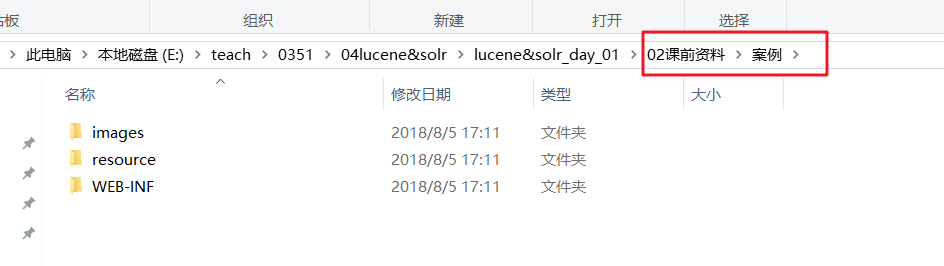
### web.xml

* 配置加载spring配置文件
* 配置spring监听器（ConetxtLoaderListener）
* 配置字符集编码过滤器（CharacterEncodingFilter）
* 配置前端控制器（DispatcherServlet）

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5"**>   <**display-name**>jd</**display-name**>   *<!--加载spring配置文件-->* <**context-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext-\*.xml</**param-value**>  </**context-param**>    *<!--配置spring的监听器-->* <**listener**>  <**listener-class**>org.springframework.web.context.ContextLoaderListener</**listener-class**>  </**listener**>   *<!--配置字符集编码过滤器-->* <**filter**>  <**filter-name**>encodingFilter</**filter-name**>  <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>   *<!--指定使用的字符集-->* <**init-param**>  <**param-name**>encoding</**param-name**>  <**param-value**>UTF-8</**param-value**>  </**init-param**>  </**filter**>   <**filter-mapping**>  <**filter-name**>encodingFilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**>  </**filter-mapping**>   *<!--配置前端控制器-->* <**servlet**>  <**servlet-name**>jd</**servlet-name**>  <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>   *<!--加载springmvc主配置文件-->* <**init-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:springmvc.xml</**param-value**>  </**init-param**>   <**load-on-startup**>1</**load-on-startup**>  </**servlet**>    <**servlet-mapping**>  <**servlet-name**>jd</**servlet-name**>  <**url-pattern**>\*.action</**url-pattern**>  </**servlet-mapping**>  </**web-app**> |

## 导入静态资源

### 解压课前资料中的“案例.zip”



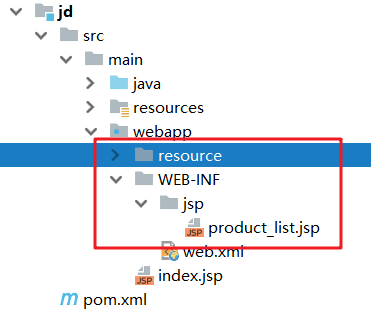
说明：

images：商品图片目录

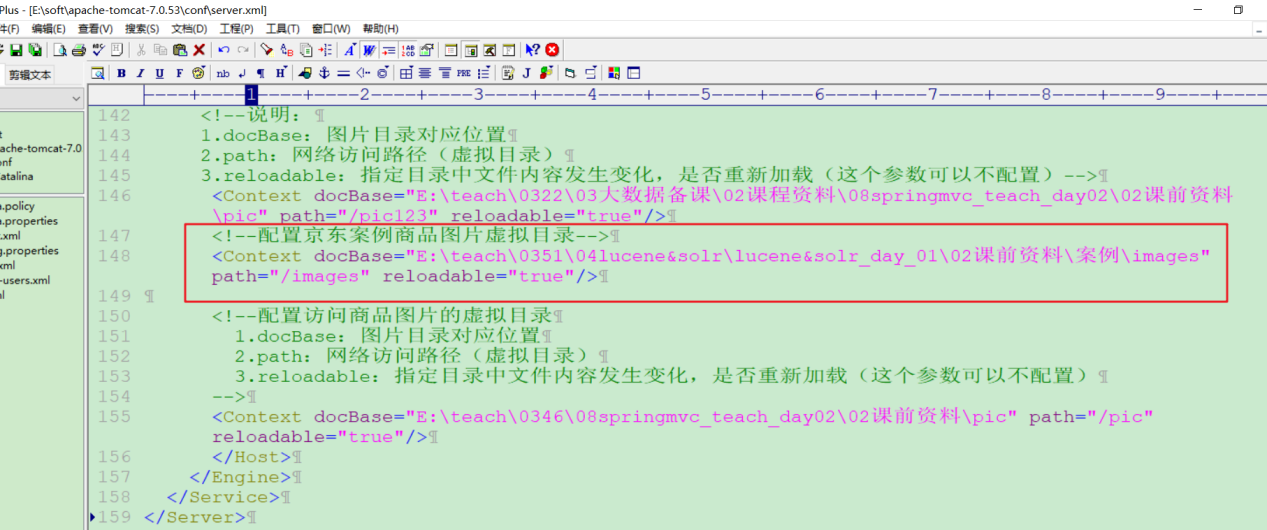
resourece：css样式文件

WEB-INF：jsp页面

### 复制resource和WEB-INF目录，粘贴到项目的webapp下



### 配置虚拟目录



## 实现商品页面的跳转

### 编写搜索controller

|  |
| --- |
| **package** cn.itheima.jd.controller;  **import** org.springframework.stereotype.Controller; **import** org.springframework.web.bind.annotation.RequestMapping;  */\*\*  \* 搜索处理器  \*/* @Controller **public class** SearchController {   */\*\*  \* 搜索商品  \* action="list.action"  \*/* @RequestMapping(**"list"**)  **public** String list(){   **return "product\_list"**;  }  } |

### 启动执行



## 实现商品搜索功能

### 编写商品实体类

说明：根据页面分析，包含有商品id，商品图片，商品名称，商品价格属性。

|  |
| --- |
| **package** cn.itheima.jd.po;  */\*\*  \* 商品实体类对象  \*/* **public class** Product {   *// 商品Id* **private** String **pid**;  *// 商品名称* **private** String **name**;  *// 商品价格* **private** String **price**;  *// 商品图片* **private** String **picture**;   **public** String getPid() {  **return pid**;  }   **public void** setPid(String pid) {  **this**.**pid** = pid;  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public** String getPrice() {  **return price**;  }   **public void** setPrice(String price) {  **this**.**price** = price;  }   **public** String getPicture() {  **return picture**;  }   **public void** setPicture(String picture) {  **this**.**picture** = picture;  } } |

### 编写搜索结果实体类

说明：根据页面分析，包含当前页，页数，总记录数，搜索结果商品结合list。

|  |
| --- |
| **package** cn.itheima.jd.po;  **import** java.util.List;  */\*\*  \* 搜索结果实体类  \*/* **public class** Result {   *// 当前页* **private** Integer **curPage**;  *// 页数* **private** Integer **pageCount**;  *// 总记录数* **private** Integer **recordCount**;   *// 商品的结果集合list* **private** List<Product> **productList**;   **public** Integer getCurPage() {  **return curPage**;  }   **public void** setCurPage(Integer curPage) {  **this**.**curPage** = curPage;  }   **public** Integer getPageCount() {  **return pageCount**;  }   **public void** setPageCount(Integer pageCount) {  **this**.**pageCount** = pageCount;  }   **public** Integer getRecordCount() {  **return recordCount**;  }   **public void** setRecordCount(Integer recordCount) {  **this**.**recordCount** = recordCount;  }   **public** List<Product> getProductList() {  **return productList**;  }   **public void** setProductList(List<Product> productList) {  **this**.**productList** = productList;  } } |

### 编写商品搜索service接口

|  |
| --- |
| **package** cn.itheima.jd.service;  **import** cn.itheima.jd.po.Result;  */\*\*  \* 搜索service接口  \*/* **public interface** SearchService {   */\*\*  \* 搜索商品  \* 参数确定：根据提交的搜索表单确定  \*/* Result searchProduct(String queryString,String catalog\_name,String price,Integer page,String sort); } |

### 编写商品搜索service实现类【重点】

|  |
| --- |
| **package** cn.itheima.jd.service.impl;  **import** cn.itheima.jd.po.Product; **import** cn.itheima.jd.po.Result; **import** cn.itheima.jd.service.SearchService; **import** com.sun.media.sound.SoftLanczosResampler; **import** org.apache.commons.lang.StringUtils; **import** org.apache.solr.client.solrj.SolrQuery; **import** org.apache.solr.client.solrj.SolrServerException; **import** org.apache.solr.client.solrj.impl.HttpSolrServer; **import** org.apache.solr.client.solrj.response.QueryResponse; **import** org.apache.solr.common.SolrDocument; **import** org.apache.solr.common.SolrDocumentList; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Service;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.Map; **import** java.util.Properties;  */\*\*  \* 搜索商品service实现类  \*/* @Service **public class** SearchServiceImpl **implements** SearchService {   *// 注入HttpSolrServer* @Autowired  **private** HttpSolrServer **httpSolrServer**;   */\*\*  \* 搜索商品  \* 参数确定：根据提交的搜索表单确定  \*/* **public** Result searchProduct(String queryString, String catalog\_name, String price, Integer page, String sort) {   *// 1.建立查询对象（SolrQuery）* SolrQuery sq = **new** SolrQuery();   *// 1.1.设置搜索关键词(如果搜索关键词为空，搜索全部)* **if**(StringUtils.*isNotBlank*(queryString)){  sq.setQuery(queryString);  }**else**{  sq.setQuery(**"\*:\*"**);  }   *// 1.2.搜索默认搜索域* sq.set(**"df"**,**"product\_keywords"**);   *// 1.3.设置过滤条件  // 商品分类名称* **if**(StringUtils.*isNotBlank*(catalog\_name)){  catalog\_name=**"product\_catalog\_name:"**+catalog\_name;  }  *// 商品价格0-9* **if**(StringUtils.*isNotBlank*(price)){  String[] arr = price.split(**"-"**);  price = **"product\_price:["**+arr[0]+**" TO "**+arr[1]+**"]"**;  }  sq.setFilterQueries(catalog\_name,price);   *// 1.4.设置分页  // 默认搜索第一页* **if**(page== **null**){  page=1;  }   **int** pageSize =10;*// 每一页显示10条* sq.setStart((page-1)\*pageSize);  sq.setRows(pageSize);   *// 1.5.设置排序  // 如果是1，是升序，其它降序* **if**(**"1"**.equals(sort)){  sq.setSort(**"product\_price"**, SolrQuery.ORDER.***asc***);  }**else**{  sq.setSort(**"product\_price"**, SolrQuery.ORDER.***desc***);  }   *// 1.6.设置高亮显示* sq.setHighlight(**true**);*//开启高亮显示* sq.addHighlightField(**"product\_name"**);  sq.setHighlightSimplePre(**"<font color='red'>"**);*// 设置高亮显示的html标签的开始部分* sq.setHighlightSimplePost(**"</font>"**);*// 设置高亮显示的html标签的结束部分   // 2.执行搜索，返回QueryResponse* QueryResponse queryResponse = **null**;  **try** {  queryResponse = **httpSolrServer**.query(sq);  } **catch** (SolrServerException e) {  e.printStackTrace();  }   *// 3.从QueryResposne中获取数据  // 3.1.获取搜索结果集* SolrDocumentList searchResult = queryResponse.getResults();   *// 3.2.获取高亮数据* Map<String, Map<String, List<String>>> highlighting = queryResponse.getHighlighting();   *// 4.处理结果集  // 4.1.创建Result* Result result = **new** Result();   *// 4.2.设置当前页* result.setCurPage(page);   *// 4.3.设置页数  /\*\*  \* 总记录数%页面大小；如果余数为0；pageCount=toltals/pageSize;  \* 如果余数不为0：pageCount=(toltals/pageSize)+1;  \*/* **int** toltals = (**int**)searchResult.getNumFound();  **int** pageCount=0;  **if**(toltals%pageSize==0){  pageCount=toltals/pageSize;  }**else**{  pageCount=(toltals/pageSize)+1;  }  result.setPageCount(pageCount);   *// 4.4.设置总记录数* result.setRecordCount(toltals);   *// 4.5.封装商品集合list* List<Product> productList = **new** ArrayList<Product>();  **for**(SolrDocument doc:searchResult){  *// 商品Id，商品名称，商品图片，商品价格* String pid = doc.get(**"id"**).toString();   String pname = **""**;  List<String> list = highlighting.get(pid).get(**"product\_name"**);  **if**(list != **null** && list.size()>0){  pname = list.get(0);  }**else**{  pname = doc.get(**"product\_name"**).toString();  }   String picture = doc.get(**"product\_picture"**).toString();   String pprice = doc.get(**"product\_price"**).toString();   *// 创建商品对象* Product product = **new** Product();  product.setPid(pid);  product.setName(pname);  product.setPicture(picture);  product.setPrice(pprice);   productList.add(product);  }   result.setProductList(productList);   **return** result;  } } |

### 改造搜索controller，增加搜索商品

|  |
| --- |
| */\*\*  \* 搜索商品  \* action="list.action"  \*/* @RequestMapping(**"list"**) **public** String list(Model model,String queryString,String catalog\_name,String price,Integer page,String sort){   *// 1.搜索商品* Result result = **searchService**.searchProduct(queryString, catalog\_name, price, page, sort);   *// 2.响应数据* model.addAttribute(**"result"**,result);   *// 3.参数数据回显* model.addAttribute(**"queryString"**,queryString);  model.addAttribute(**"catalog\_name"**,catalog\_name);  model.addAttribute(**"price"**,price);  model.addAttribute(**"sort"**,sort);   **return "product\_list"**; } |

