

淘淘商城

第八天

**讲师：传智.入云龙**

# 课程计划

* Solr服务的搭建
* 搜索功能的实现

## 系统架构



# Solr概述

什么是Solr

Solr 是Apache下的一个顶级开源项目，采用Java开发，它是基于Lucene的全文搜索服务器。Solr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，并对索引、搜索性能进行了优化。

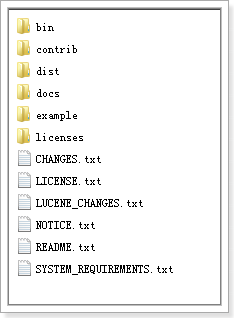
Solr可以独立运行，运行在Jetty、Tomcat等这些Servlet容器中，Solr 索引的实现方法很简单，用 POST 方法向 Solr 服务器发送一个描述 Field 及其内容的 XML 文档，Solr根据xml文档添加、删除、更新索引 。Solr 搜索只需要发送 HTTP GET 请求，然后对 Solr 返回Xml、json等格式的查询结果进行解析，组织页面布局。Solr不提供构建UI的功能，Solr提供了一个管理界面，通过管理界面可以查询Solr的配置和运行情况。

## 下载

从Solr官方网站（http://lucene.apache.org/solr/ ）下载Solr4.10.3，根据Solr的运行环境，Linux下需要下载lucene-4.10.3.tgz，windows下需要下载lucene-4.10.3.zip。

Solr使用指南可参考：https://wiki.apache.org/solr/FrontPage。

下载lucene-4.10.3.zip并解压：



bin：solr的运行脚本

contrib：solr的一些贡献软件/插件，用于增强solr的功能。

dist：该目录包含build过程中产生的war和jar文件，以及相关的依赖文件。

docs：solr的API文档

example：solr工程的例子目录：

* example/solr：

该目录是一个包含了默认配置信息的Solr的Core目录。

* example/multicore：

该目录包含了在Solr的multicore中设置的多个Core目录。

* example/webapps：

该目录中包括一个solr.war，该war可作为solr的运行实例工程。

licenses：solr相关的一些许可信息

# Solr的安装及配置

## 运行环境

solr 需要运行在一个Servlet容器中，Solr4.10.3要求jdk使用1.7以上，Solr默认提供Jetty（java写的Servlet容器），本教程使用Tocmat作为Servlet容器，环境如下：

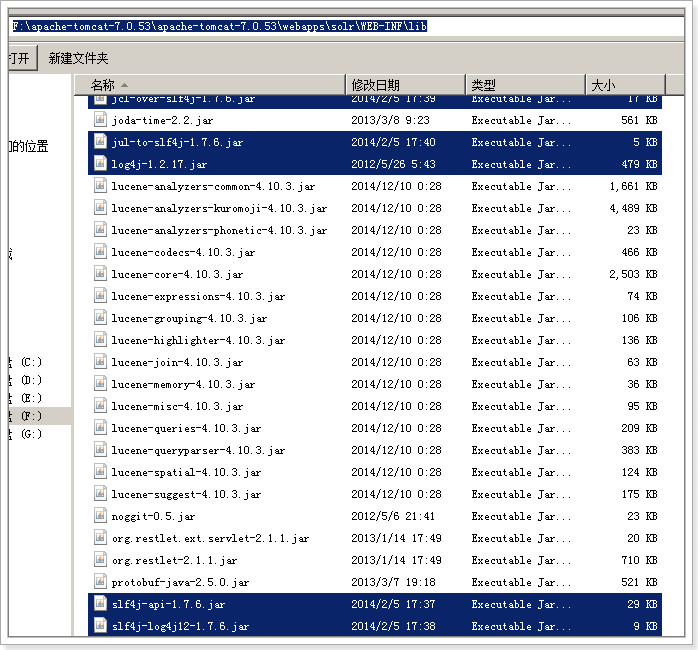
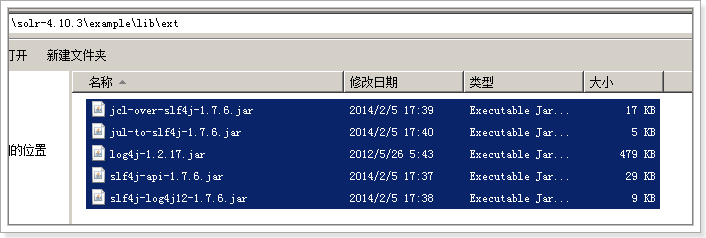
Solr：Solr4.10.3

Jdk：jdk1.7.0\_72

Tomcat：apache-tomcat-7.0.53

## Solr整合tomcat

1. 将dist\solr-4.10.3.war拷贝到Tomcat的webapp目录下改名为solr.war
2. 启动tomcat后，solr.war自动解压，将原来的solr.war删除。
3. 拷贝example\lib\ext 目录下所有jar包到Tomcat的webapp\solr\WEB-INF\lib目录下



1. 拷贝log4j.properties文件

在 Tomcat下webapps\solr\WEB-INF目录中创建文件 classes文件夹，

复制Solr目录下example\resources\log4j.properties至Tomcat下webapps\solr\WEB-INF\classes目录

1. 创建solrhome及配置solrcore的solrconfig.xml文件
2. 修改Tomcat目录 下webapp\solr\WEB-INF\web.xml文件，如下所示：

设置Solr home

<!--配置jndi告诉solr工程我们的solrhome的位置-->

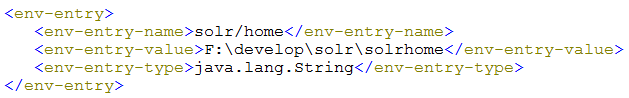
<env-entry>

<env-entry-name>solr/home</env-entry-name>

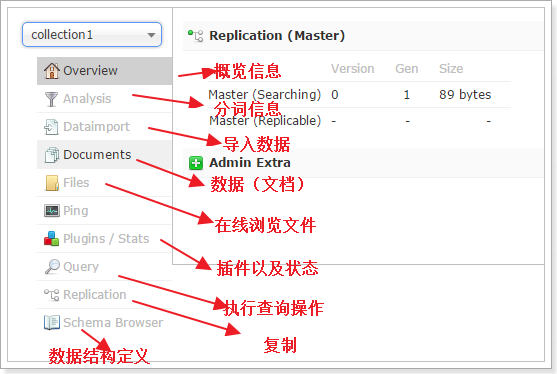
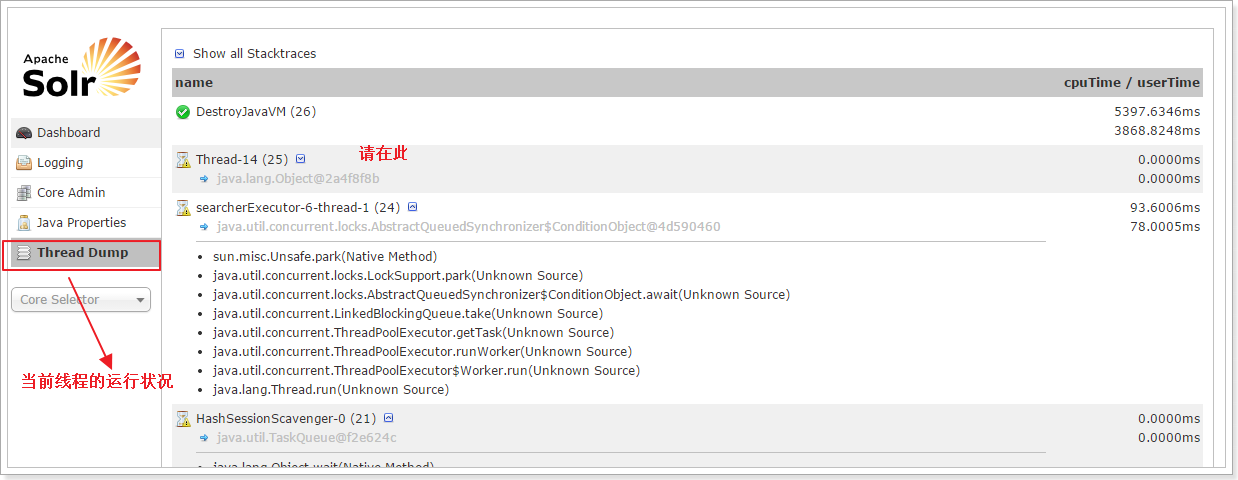
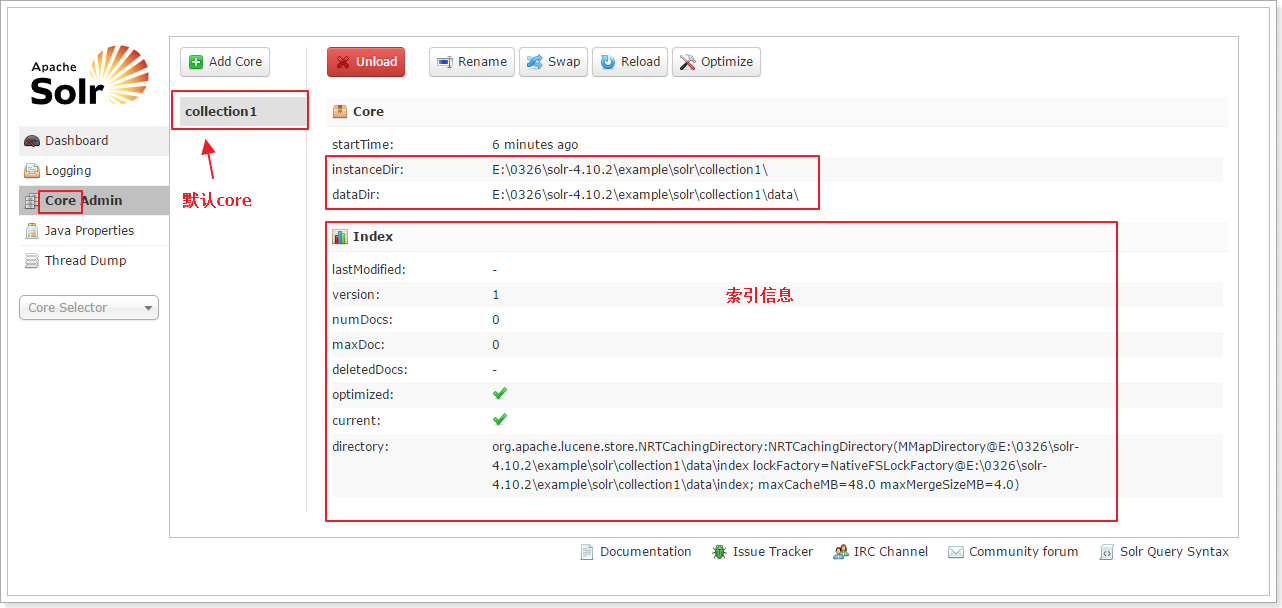
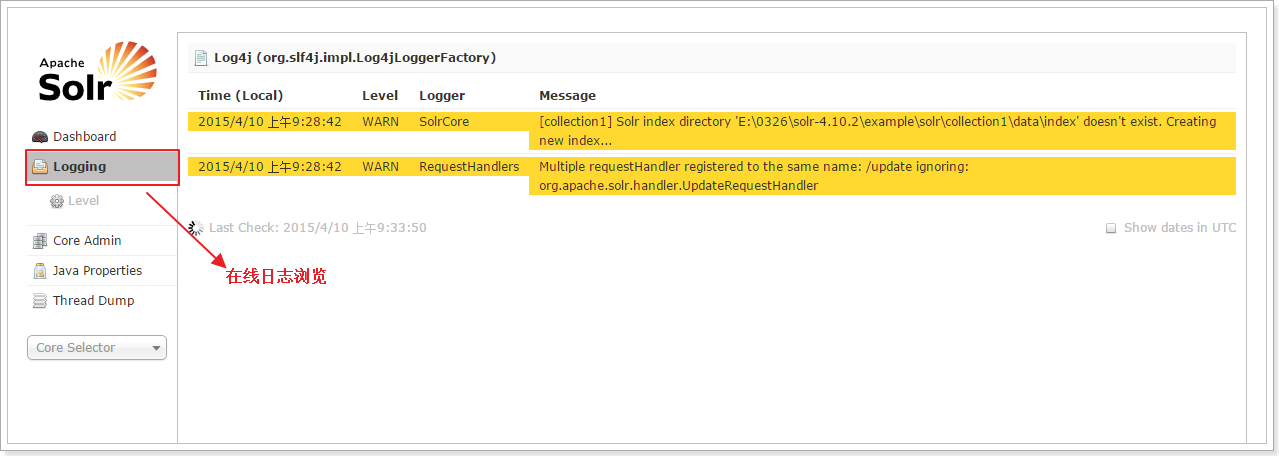
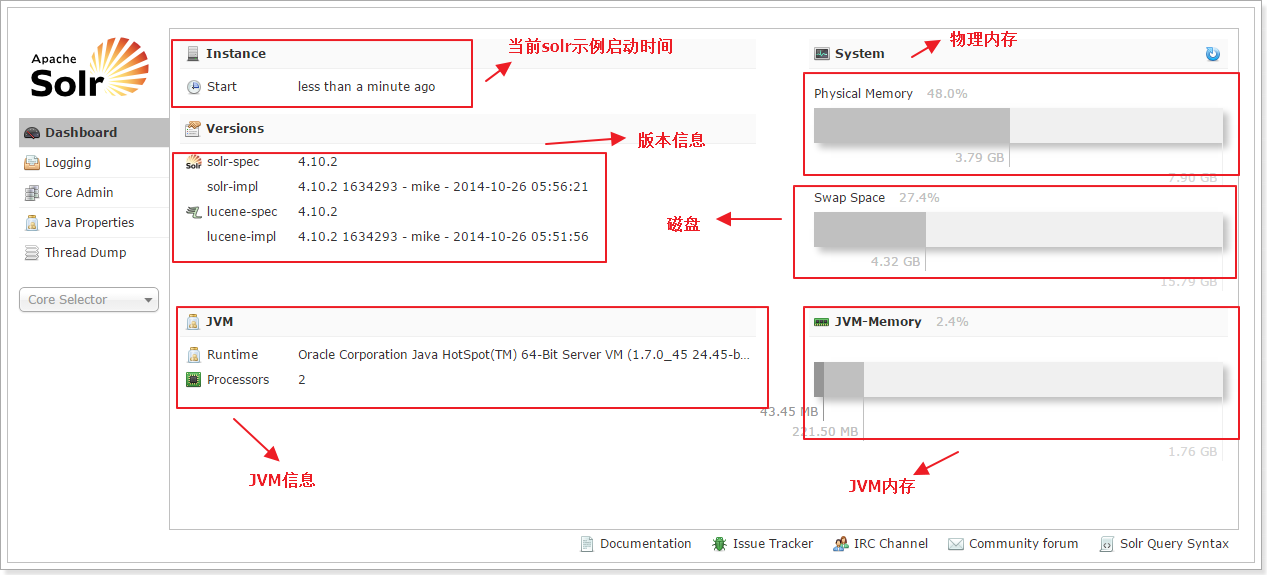
<env-entry-value>D:/temp/solr/solrhome</env-entry-value>

<env-entry-type>java.lang.String</env-entry-type>

</env-entry>



# Solr界面功能



# 安装中文分词器

## 安装步骤

### 第一步：配置IKAnalyzer的jar包

拷贝IKAnalyzer的文件到Tomcat下Solr目录中

将IKAnalyzer2012FF\_u1.jar拷贝到 Tomcat的webapps/solr/WEB-INF/lib 下。

### 第二步：IKAnalyzer的配置文件

在Tomcat的webapps/solr/WEB-INF/下创建classes目录

将IKAnalyzer.cfg.xml、ext\_stopword.dic mydict.dic copy到 Tomcat的

webapps/solr/WEB-INF/classes

**注意：ext\_stopword.dic 和mydict.dic必须保存成无BOM的utf-8类型。**

### 第三步：修改schema.xml文件

修改schema.xml文件

修改Solr的schema.xml文件，添加FieldType：

|  |
| --- |
| <fieldType name="text\_ik" class="solr.TextField">  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>  </fieldType> |

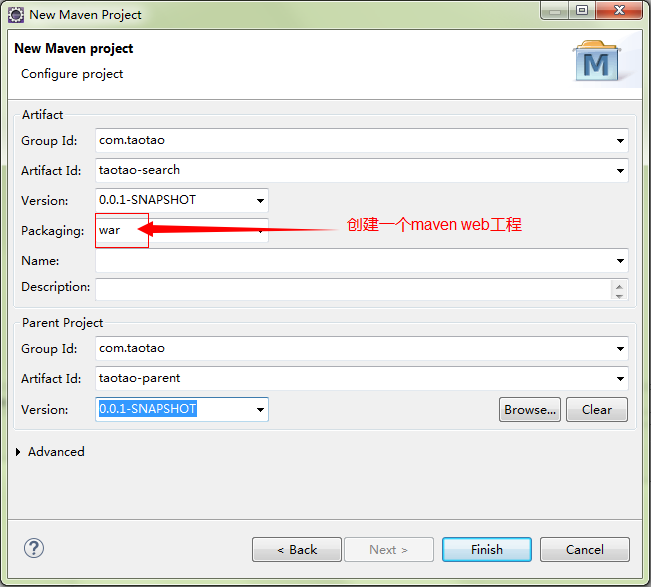
### 第四步：设置业务系统Field

设置业务系统Field

|  |
| --- |
| <field name="item\_title" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_sell\_point" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_price" type="long" indexed="true" stored="true"/>  <field name="item\_image" type="string" indexed="false" stored="true" />  <field name="item\_category\_name" type="string" indexed="true" stored="true" />  <field name="item\_desc" type="text\_ik" indexed="true" stored="false" />  <field name="item\_keywords" type="text\_ik" indexed="true" stored="false" multiValued="true"/>  <copyField source="item\_title" dest="item\_keywords"/>  <copyField source="item\_sell\_point" dest="item\_keywords"/>  <copyField source="item\_category\_name" dest="item\_keywords"/>  <copyField source="item\_desc" dest="item\_keywords"/> |

# 搭建taotao-search服务

## 创建工程



## 配置工程

工程配置参考taotao-rest，基本相同，不需要引入Jedis 的jar包，需要sorlJ的jar包。其他都相同。

### 添加solrj的依赖关系

|  |
| --- |
| <!-- solr客户端 -->  <dependency>  <groupId>org.apache.solr</groupId>  <artifactId>solr-solrj</artifactId>  </dependency> |

## 商品信息导入索引库

### 需求分析

需要把数据库中的商品信息导入索引库，需要商品的id、商品的名称、商品的卖点、商品的价格、商品的图片、商品的分类名称。

原则：需要展示给用户的字段、需要搜索的字段需要添加到索引库。

### Sql语句

|  |
| --- |
| SELECT  a.id,  a.title,  a.sell\_point,  a.price,  a.image,  b. NAME category\_name  FROM  tb\_item a  LEFT JOIN tb\_item\_cat b ON a.cid = b.id |

### POJO定义

|  |
| --- |
| /\*\*  \* 商品信息POJO  \* <p>Title: Item</p>  \* <p>Description: </p>  \* <p>Company: www.itcast.com</p>  \* **@author** 入云龙  \* **@date** 2015年7月28日下午3:06:16  \* **@version** 1.0  \*/  **public** **class** Item {  **private** Long id;  **private** String title;  **private** String sell\_point;  **private** Long price;  **private** String image;  **private** String category\_name;  } |

### Mapper

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >  <mapper namespace=*"com.taotao.search.mapper.ItemMapper"* >  <select id=*"getItemList"* resultType=*"com.taotao.search.pojo.Item"*>  SELECT  a.id,  a.title,  a.sell\_point,  a.price,  a.image,  b. NAME category\_name  FROM  tb\_item a  LEFT JOIN tb\_item\_cat b ON a.cid = b.id  </select>  </mapper> |

### Service

|  |
| --- |
| @Service  **public** **class** ItemServiceImpl **implements** ItemService {  @Autowired  **private** ItemMapper itemMapper;  @Autowired  **private** SolrServer solrServer;    @Override  **public** TaotaoResult importItemToIndex() **throws** Exception {  //查询商品列表  List<Item> itemList = itemMapper.getItemList();  //将商品列表导入solr  **for** (Item item : itemList) {  SolrInputDocument document = **new** SolrInputDocument();  document.addField("id", item.getId());  document.addField("item\_title", item.getTitle());  document.addField("item\_sell\_point", item.getSell\_point());  document.addField("item\_price", item.getPrice());  document.addField("item\_image", item.getImage());  document.addField("item\_category\_name", item.getCategory\_name());  //将文档写入索引库  solrServer.add(document);  }  //提交修改  solrServer.commit();  **return** TaotaoResult.*ok*();  }  } |

### Controller

|  |
| --- |
| @Controller  @RequestMapping("/manager")  **public** **class** ItemController {  @Autowired  **private** ItemService itemService;    @RequestMapping("/importall")  @ResponseBody  **public** TaotaoResult importAll() {  TaotaoResult result = **null**;  **try** {  result = itemService.importItemToIndex();  } **catch** (Exception e) {  e.printStackTrace();  **return** TaotaoResult.*build*(500, ExceptionUtil.*getStackTrace*(e));  }  **return** result;  }  } |

### Pom.xml

需要在pom文件的build节点中添加如下信息，否则mapper映射文件不会被发布，从而发生异常。

|  |
| --- |
| <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  <resource>  <directory>src/main/resources</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  </resources> |

### applicationContext-dao.xml

扫描包添加taotao-search工程中的包：

|  |
| --- |
| <!-- 加载mapper文件 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.taotao.mapper,com.taotao.search.mapper"*></property>  </bean> |

# 实现商品搜索功能

实现商品搜索功能需要两步实现：

第一步：在taotoa-search工程中发布服务

第二步：在taotao-portal中调用服务并展示结果。

## 发布搜索服务

### Dao

返回值SearchResult：

|  |
| --- |
| **public** **class** SearchResult {  **private** Long recordCount;  **private** List<Item> itemList;  **private** Integer pageCount;  **private** Integer curPage;    } |

Dao

|  |
| --- |
| @Service  **public** **class** ItemSearchDaoImpl **implements** ItemSearchDao {    @Autowired  **private** SolrServer solrServer;  @Override  **public** SearchResult searchItem(SolrQuery solrQuery) **throws** Exception {  //根据查询条件搜索索引库  QueryResponse response = solrServer.query(solrQuery);  //取商品列表  SolrDocumentList documentList = response.getResults();  //商品列表  List<Item> itemList = **new** ArrayList<>();  **for** (SolrDocument solrDocument : documentList) {  Item item = **new** Item();  item.setId((Long) solrDocument.get("id"));  //取高亮显示  Map<String, Map<String, List<String>>> highlighting = response.getHighlighting();  List<String> list = highlighting.get(solrDocument.get("id")).get("item\_title");  String title = "";  **if** (**null** != list && !list.isEmpty()) {  title = list.get(0);  } **else** {  title = (String) solrDocument.get("item\_title");  }  item.setTitle(title);  item.setPrice((Long) solrDocument.get("item\_price"));  item.setSell\_point((String) solrDocument.get("item\_sell\_point"));  item.setImage((String) solrDocument.get("item\_image"));  item.setCategory\_name((String) solrDocument.get("item\_category\_name"));    itemList.add(item);  }  SearchResult result = **new** SearchResult();  //商品列表  result.setItemList(itemList);  //总记录数据  result.setRecordCount(documentList.getNumFound());    **return** result;  }  } |

### Service

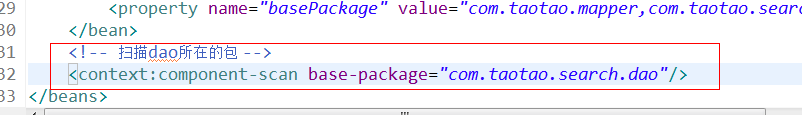
|  |
| --- |
| @Service  **public** **class** ItemSearchServiceImpl **implements** ItemSearchService {  @Value("${SEARCH\_RESULT\_PAGE\_SIZE}")  **private** Integer PAGE\_SIZE;  @Autowired  **private** ItemSearchDao itemSearchDao;    @Override  **public** SearchResult searchItem(String queryString, Integer page) **throws** Exception {  //创建一个查询对象  SolrQuery solrQuery = **new** SolrQuery();  //查询条件  **if** (StringUtils.*isBlank*(queryString)) {  solrQuery.setQuery("\*:\*");  } **else** {  solrQuery.setQuery(queryString);  }  //分页条件  **if** (page == **null**) {  page = 1;  }  solrQuery.setStart((page -1) \* PAGE\_SIZE);  solrQuery.setRows(PAGE\_SIZE);  //高亮显示  solrQuery.setHighlight(**true**);  //设置高亮显示的域  solrQuery.addHighlightField("item\_title");  //高亮显示前缀  solrQuery.setHighlightSimplePre("<em style=\"color:red\">");  //后缀  solrQuery.setHighlightSimplePost("</em>");  //设置默认搜索域  solrQuery.set("df", "item\_keywords");    //执行查询  SearchResult result = itemSearchDao.searchItem(solrQuery);  //计算分页  Long recordCount = result.getRecordCount();  **int** pageCount = (**int**) (recordCount / PAGE\_SIZE);  **if** (recordCount % PAGE\_SIZE > 0) {  pageCount++;  }  result.setPageCount(pageCount);  result.setCurPage(page);    **return** result;  }  } |

### Controller

|  |
| --- |
| @Controller  **public** **class** ItemSearchController {    @Autowired  **private** ItemSearchService itemSearchService;  @RequestMapping("/q")  @ResponseBody  **public** TaotaoResult search(@RequestParam(value = "kw") String queryString,  @RequestParam(value = "page", defaultValue = "1") Integer page) {    **if** (StringUtils.*isBlank*(queryString)) {  **return** TaotaoResult.*build*(400, "查询条件是必须的参数");  }  SearchResult result = **null**;  **try** {  result = itemSearchService.searchItem(queryString, page);    } **catch** (Exception e) {  e.printStackTrace();  **return** TaotaoResult.*build*(500, ExceptionUtil.*getStackTrace*(e));  }    **return** TaotaoResult.*ok*(result);  }  } |

### 扫描dao所在的包

在applicationContext-dao.xml文件中添加如下内容：



### 解决get请求乱码问题

在controller中添加字符串编码转换逻辑：

Tomcat默认的编码为ISO8859-1，需要转换成utf-8的编码。



## 调用服务实现搜索功能

### 功能分析

Taotao-portal展示首页，用户在首页输入查询内容提交至taotao-portal，taotao-portal调用taotao-search提供的搜索服务，得到商品列表。在taotao-portal中渲染商品列表展示搜索结果页面。

### Dao层

木有

### Service层

使用HttpClientUtil工具类调用搜索服务，返回一个json数据，需要把json数据转换成TaotaoResult对象。

|  |
| --- |
| @Service  **public** **class** SearchServiceImpl **implements** SearchService {  @Value("${SEARCH\_BASE\_URL}")  **private** String SEARCH\_BASE\_URL;  @Override  **public** SearchResult searchItemList(String queryString, Integer page) **throws** Exception {  //查询参数  Map<String, String> param = **new** HashMap<>();  param.put("kw", queryString);  param.put("page", page==**null**?"1":page.toString());  //调用taotao-search提供的搜索服务  String resultString = HttpClientUtil.*doGet*(SEARCH\_BASE\_URL, param);  //转换成taotaoResult对象  TaotaoResult taotaoResult = TaotaoResult.*formatToPojo*(resultString, SearchResult.**class**);  SearchResult searchResult = **null**;  //查询成功  **if** (taotaoResult.getStatus() == 200) {  //取查询结果  searchResult = (SearchResult) taotaoResult.getData();  }    **return** searchResult;  }  } |

### Controller

|  |
| --- |
| @Controller  **public** **class** SearchController {  @Autowired  **private** SearchService searchService;    @RequestMapping("/search")  **public** String searchItemList(@RequestParam(value="q")String queryString, Integer page, Model model) **throws** Exception {  //字符串转码  queryString = **new** String(queryString.getBytes("ISO8859-1"), "UTF-8");    SearchResult searchResult = searchService.searchItemList(queryString, page);  model.addAttribute("itemList", searchResult.getItemList());  model.addAttribute("query", queryString);  model.addAttribute("totalPages", searchResult.getPageCount());  model.addAttribute("page", searchResult.getCurPage());  model.addAttribute("pages", searchResult.getPageCount());  **return** "search";  }  } |