**每日作业卷**

**Lucene&Solr第3天**

# 关卡1

## 训练案例1

### 训练描述

使用课上资料，实现以下功能：

要求：

1. 配置安装IK分词器到solr服务
2. 配置使用IK分词器
3. 配置扩展词汇和停用词汇
4. 根据MySQL数据库表结构，配置业务域
5. 配置solr的DataImport功能实现数据导入到索引库

### 操作步骤描述答案

1. 配置安装IK分词器到solr服务

复制IK分词器的jar包到solr服务中

复制IK分词器的三个配置文件到solr服务中

1. 配置使用IK分词器

需要配置FieldType和Field

<!—配置实用IK分词器的FieldType -->

<fieldType name="text\_ik" class="solr.TextField">

<analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>

</fieldType>

<!—配置Field-->

<field name="content\_ik" type="text\_ik" indexed="true" stored="true" />

1. 配置扩展词汇和停用词汇

修改IK分词器的配置文件ext.dic实现添加扩展词汇

修改IK分词器的配置文件stopword.dic实现添加停用词汇

1. 根据MySQL数据库表结构，配置业务域

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

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

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

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

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

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

1. 配置solr的DataImport功能实现数据导入到索引库

添加jar包

复制dist目录中的solr-dataimporthandler-4.10.3.jar，粘贴到contrib\dataimporthandler\lib

复制mysql-connector-java-5.1.7-bin.jar，粘贴到contrib\db\lib

修改solrconfig.xml，配置刚刚复制的jar包

添加以下配置即可：

<lib dir="${solr.install.dir:../..}/contrib/dataimporthandler/lib" regex=".\*\.jar" />

<lib dir="${solr.install.dir:../..}/contrib/db/lib" regex=".\*\.jar" />

修改solrconfig.xml，添加requestHandler配置

<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配置文件

在collection1\conf\目录下创建data-config.xml文件

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

<dataConfig>

<dataSource type="JdbcDataSource"

driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://127.0.0.1:3306/solr"

user="root"

password="root"/>

<document>

<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"/>

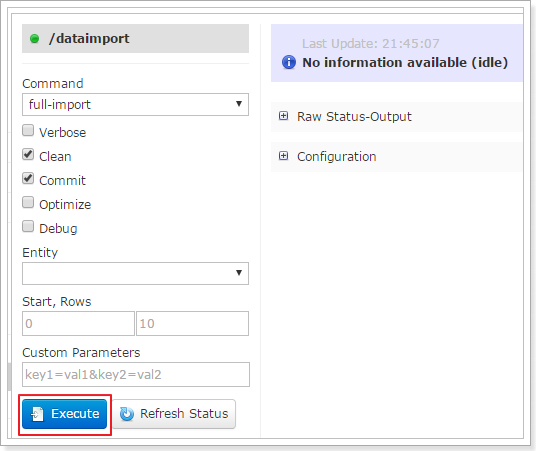
<field column="picture" name="product\_picture"/>

</entity>

</document>

</dataConfig>

重启tomcat

进入solr管理界面的DataImport功能，点击“execute”按钮导入数据

## 训练案例2

### 训练描述

要求：

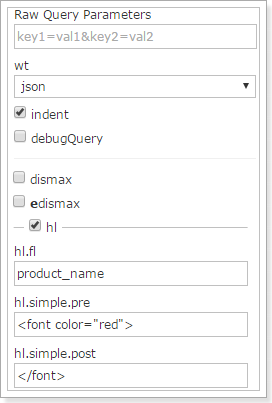
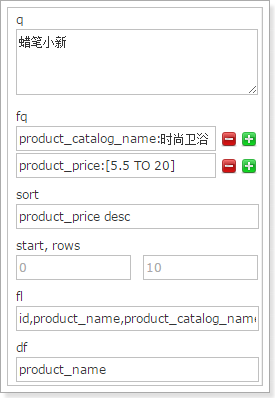
1. 使用solr浏览器管理界面完成复杂查询
2. 使用solrJ完成对索引库的复杂查询

### 操作步骤描述答案

要求：

1. 应用solr浏览器管理界面完成复杂查询

使用管理界面的Query功能完成复杂查询

查询条件包括：查询语句，过滤条件，排序，分页，设置返回哪些数据，设置默认搜索字段，设置输出格式，设置高亮

1. 使用solrJ完成对索引库的复杂查询

使用solrj完成和要求1一样的复杂查询

查询条件包括：查询语句，过滤条件，排序，分页，设置返回哪些数据，设置默认搜索字段，设置输出格式，设置高亮

@Test

**public** **void** testQuery2() **throws** Exception {

// 创建SolrQuery查询对象

SolrQuery solrQuery = **new** SolrQuery();

// 设置查询语句:挂钩

solrQuery.setQuery("挂钩");

// 设置过滤查询fq,product\_price:{4 TO 18.9],product\_catalog\_name:幽默杂货

solrQuery.setFilterQueries("product\_price:{4 TO 18.9]", "product\_catalog\_name:幽默杂货");

// 设置排序sort:product\_price asc

solrQuery.setSort("product\_price", ORDER.***asc***);

// 设置分页start, rows,

solrQuery.setStart(10);

solrQuery.setRows(10);

// 设置fl，显示的字段id,product\_name,product\_catalog\_name,product\_price

solrQuery.setFields("id,product\_name,product\_catalog\_name,product\_price");

// 设置默认查询域df,product\_name

solrQuery.set("df", "product\_name");

// 设置wt，返回的数据格式

solrQuery.set("wt", "json");

// 设置高亮

// 开启高亮

solrQuery.setHighlight(**true**);

// 设置高亮字段

solrQuery.addHighlightField("product\_name");

// 设置高亮前缀

solrQuery.setHighlightSimplePre("<font color='red'>");

// 设置高亮后缀

solrQuery.setHighlightSimplePost("</font>");

// 执行查询

QueryResponse response = **this**.httpSolrServer.query(solrQuery);

// 获取高亮数据

// {

// "1": {"product\_name": ["花儿朵朵彩色金属门"]},

// "2": {"product\_name": ["幸福一家人彩色金"]}

// }

Map<String, Map<String, List<String>>> map = response.getHighlighting();

// 获取结果集

SolrDocumentList results = response.getResults();

// 打印查询到的数据总条数

System.***out***.println("查询的数据总条数是：" + results.getNumFound());

// 遍历结果集

**for** (SolrDocument solrDocument : results) {

// 解析高亮数据

List<String> hlist = map.get(solrDocument.get("id").toString()).get("product\_name");

System.***out***.println("------------------------");

// 打印结果集

// 商品id

System.***out***.println(solrDocument.get("id").toString());

// 商品product\_name

System.***out***.println(solrDocument.get("product\_name").toString());

// 打印高亮数据

**if** (hlist != **null** && hlist.size() > 0) {

System.***out***.println("高亮数据是：" + hlist.get(0));

}

// 商品product\_catalog\_name

System.***out***.println(solrDocument.get("product\_catalog\_name").toString());

// 商品product\_price

System.***out***.println(solrDocument.get("product\_price").toString());

}

}

# 关卡2

## 训练案例1

### 训练描述

1. 创建工程并加入SpringMVC、Spring、solrj的相关jar包
2. 添加相关配置文件整合ss和solrj
3. 分析需求并完成页面跳转和pojo编写
4. 编写Controller
5. 编写Service

### 操作步骤描述答案

1. 创建工程并加入SpringMVC、Spring、solrj的相关jar包

使用课堂资料完成搭建

1. 添加相关配置文件整合ss和solrj

编写Spring+SpringMVC配置文件

applicationContext-service.xml

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

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-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>

<!-- 配置service扫描 -->

<context:component-scan base-package=*"cn.itcast.jd.service"* />

</beans>

springmvc.xml

<?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:p=*"http://www.springframework.org/schema/p"*

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-4.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd"*>

<!-- 配置controller扫描 -->

<context:component-scan base-package=*"cn.itcast.jd.controller"* />

<!-- 配置注解驱动 -->

<mvc:annotation-driven />

<!-- 配置视图解析器 -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!-- 配置前缀 -->

<property name=*"prefix"* value=*"/WEB-INF/jsp/"* />

<!-- 配置后缀 -->

<property name=*"suffix"* value=*".jsp"* />

</bean>

</beans>

编写Spring管理solrj的配置文件

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

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-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>

<!-- 配置HttpSolrServer连接对象 -->

<bean class=*"org.apache.solr.client.solrj.impl.HttpSolrServer"*>

<!-- 配置构造方法 -->

<constructor-arg name=*"baseURL"* value=*"http://127.0.0.1:8081/solr/"* />

</bean>

</beans>

1. 分析需求并完成页面跳转和pojo编写

编写pojo

**public** **class** Product {

**private** String pid;

**private** String name;

**private** String price;

**private** String picture;

}

**public** **class** Result {

**private** Integer curPage;// 当前页

**private** Integer pageCount;// 总页数

**private** Long recordCount;// 数据总条数

**private** List<Product> productList;// 商品结果集

}

1. 编写Controller

@Controller

@RequestMapping("search")

**public** **class** SearchController {

@Autowired

**private** SearchService searchService;

@RequestMapping(value = "list")

**public** String search(Model model, String queryString, String catalog\_name, String price,

@RequestParam(value = "page", defaultValue = "1") Integer page, String sort) {

// 调用service方法，根据条件查询商品

Result result = **this**.searchService.search(queryString, catalog\_name, price, page, sort);

// 把查询结果放到Model中，传递给页面

model.addAttribute("result", result);

// 回显查询参数

model.addAttribute("queryString", queryString);

model.addAttribute("catalog\_name", catalog\_name);

model.addAttribute("price", price);

model.addAttribute("page", page);

model.addAttribute("sort", sort);

**return** "product\_list";

}

}

1. 编写Service

@Service

**public** **class** SearchServiceImpl **implements** SearchService {

@Autowired

**private** HttpSolrServer httpSolrServer;

@Override

**public** Result search(String queryString, String catalog\_name, String price, Integer page, String sort) {

// 创建查询对象

SolrQuery solrQuery = **new** SolrQuery();

// 设置查询语句

**if** (StringUtils.*isNotBlank*(queryString)) {

solrQuery.setQuery(queryString);

} **else** {

solrQuery.setQuery("\*:\*");

}

// 设置默认搜索域

solrQuery.set("df", "product\_name");

// 设置商品类别和商品价格，使用过滤查询

// 拼接商品类目名称过滤语句product\_catalog\_name:幽默杂货

**if** (StringUtils.*isNotBlank*(catalog\_name)) {

// 不为空，设置过滤语句

catalog\_name = "product\_catalog\_name:" + catalog\_name;

}

// 拼接商品价格过滤语句,product\_price:[4 TO 18.9]

// 50-\*

**if** (StringUtils.*isNotBlank*(price)) {

String[] str = StringUtils.*split*(price, "-");

price = "product\_price:[" + str[0] + " TO " + str[1] + "]";

}

// 设置过滤查询到查询对象中

solrQuery.setFilterQueries(catalog\_name, price);

// 设置价格排序

**if** ("1".equals(sort)) {

solrQuery.setSort("product\_price", ORDER.***asc***);

} **else** {

solrQuery.setSort("product\_price", ORDER.***desc***);

}

// 设置分页

// 设置每页显示的数据条数

**int** rows = 16;

**if** (page == **null**) {

page = 1;

}

solrQuery.setStart((page - 1) \* rows);

solrQuery.setRows(rows);

// 设置高亮

// 开启高亮

solrQuery.setHighlight(**true**);

// 设置高亮字段

solrQuery.addHighlightField("product\_name");

// 设置前缀

solrQuery.setHighlightSimplePre("<font color='red'>");

// 设置后缀

solrQuery.setHighlightSimplePost("</font>");

**try** {

// 执行查询

QueryResponse response = **this**.httpSolrServer.query(solrQuery);

// 获取高亮数据

Map<String, Map<String, List<String>>> map = response.getHighlighting();

// 获取结果集

SolrDocumentList results = response.getResults();

// 遍历结果集

// 声明存放Product的list容器

List<Product> list = **new** ArrayList<Product>();

**for** (SolrDocument solrDocument : results) {

// 封装成Product对象

// 声明Product

Product product = **new** Product();

// 解析高亮数据

List<String> hlist = map.get(solrDocument.get("id").toString()).get("product\_name");

// 把查询结果的Document里的值封装到Product

// 商品id

product.setPid(solrDocument.get("id").toString());

// 商品名称product\_name

**if** (hlist != **null** && hlist.size() > 0) {

product.setName(hlist.get(0));

} **else** {

product.setName(solrDocument.get("product\_name").toString());

}

// 商品价格product\_price

product.setPrice(solrDocument.get("product\_price").toString());

// 商品图片product\_picture

product.setPicture(solrDocument.get("product\_picture").toString());

// 把封装好的数据放到list容器中

list.add(product);

}

// 封装返回对象Result

Result result = **new** Result();

// 设置当前页

result.setCurPage(page);

// 设置总页数 total%rows==0?total/rows:(total/rows)+1

**long** total = results.getNumFound();

result.setPageCount((**int**) (total % rows == 0 ? total / rows : (total / rows) + 1));

// 设置结果集

result.setProductList(list);

// 设置查询的数据总条数

result.setRecordCount(total);

// 返回封装的Result

**return** result;

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

// 如果查询出现异常，返回null

**return** **null**;

}

}

# 关卡3

## 训练案例1

### 训练描述

使用lucene&solr第一天的关卡3的数据，完成以下要求

要求：

1. 配置solr的schema.xml文件，加入pic和desc两个Field域
2. 添加lucene&solr第一天的关卡3准备的所有数据到solr中
3. 实现复杂查询数据

查询条件包括：查询语句，过滤条件，排序，分页，设置返回哪些数据，设置默认搜索字段，设置输出格式，设置高亮