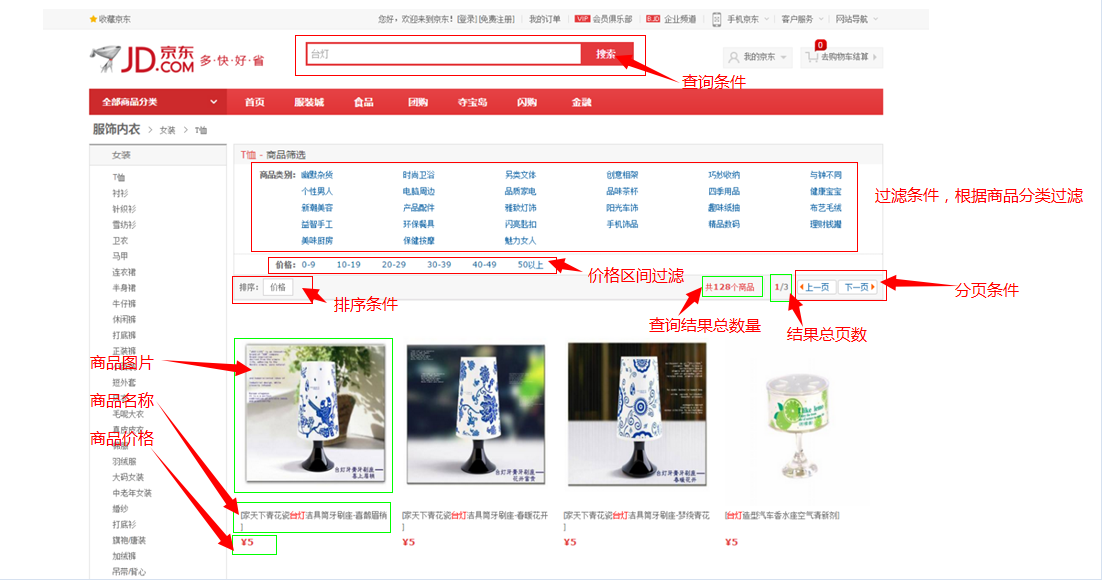
此文章已于 23:03:49 2017/11/20 发布到 HealerJean梦想博客

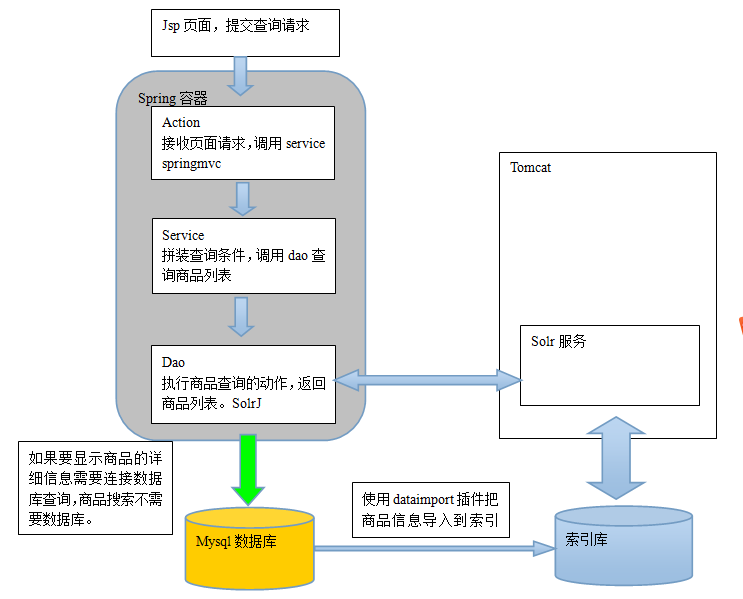
solr案例

类别 solr

# 1、原型分析



# 2、业务流程



# 3、索引库的更新

## 1、实时更新。商品修改后直接更新索引库。

## 2、定时更新。定时使用dataimport插件更新全部商品信息。

# 4、实体类

## 1、商品类对象

|  |
| --- |
| **public** **class** ProductModel {  // 商品编号  **private** String pid;  // 商品名称  **private** String name;  // 商品分类名称  **private** String catalog\_name;  // 价格  **private** **float** price;  // 商品描述  **private** String description;  // 图片名称  **private** String picture;  **public** String getPid() {  **return** pid;  } |

## 2、返回实体类，根据上面的原型而来

|  |
| --- |
| **public** **class** ResultModel {  // 商品列表  **private** List<ProductModel> productList;  // 商品总数  **private** Long recordCount;  // 总页数  **private** **int** pageCount;  // 当前页  **private** **int** curPage; |

# 5、控制层Controller

## 1、参数配置（具体参数，请看jsp）

|  |
| --- |
| 1. **查询条件：String queryString （商品描述）** 2. **过滤条件，商品分类名称：String catalog\_name** 3. **价格区间的过滤条件，价格区间格式：0-10，11-20，21-30，30-\*：String price** 4. **排序条件，根据价格排序，可以只传递排序方式0：升序 1：降序：String sort** 5. **分页条件，只需要得到页码即可，每页显示商品的数量定义常量：Integer page** |

## 2、代码如下

|  |
| --- |
| @Controller  **public** **class** ProductController {    @Autowired  **private** ProductService service;  @RequestMapping("/list")  **public** String queryProduct(String queryString, String catalog\_name, String price,  String sort, Integer page, Model model) {  //执行查询  ResultModel resultModel = **null**;  **try** {  resultModel = service.queryProduct(queryString, catalog\_name, price, sort, page);  } **catch** (Exception e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  //把查询结果传递到页面  model.addAttribute("result", resultModel);  //参数回显  model.addAttribute("queryString", queryString);  model.addAttribute("catalog\_name", catalog\_name);  model.addAttribute("price", price);  model.addAttribute("sort", sort);  model.addAttribute("page", page);  //返回jsp的名称  ///WEB-INF/jsp/ + product\_list + .jsp  **return** "product\_list";  }  } |

# 6、service层

|  |
| --- |
| @Service  **public** **class** ProductServiceImpl **implements** ProductService {    @Autowired  **private** ProductDao productDao;  @Override  **public** ResultModel queryProduct(String queryString, String catalog\_name,  String price, String sort, Integer page) **throws** Exception {  //拼装查询条件  SolrQuery query = **new** SolrQuery();  //查询条件  **if** (**null** != queryString && !"".equals(queryString)) {  query.setQuery(queryString);  } **else** {  query.setQuery("\*:\*");  }  //商品类别过滤  **if** (**null** != catalog\_name && !"".equals(catalog\_name)) {  query.addFilterQuery("product\_catalog\_name:" + catalog\_name);  }  //价格过滤  **if** (**null** != price && !"".equals(price)) {  String[] strings = price.split("-");  query.addFilterQuery("product\_price:["+strings[0]+" TO "+strings[1]+"]");  }  //排序条件  **if** ("1".equals(sort)) {  query.setSort("product\_price", ORDER.*desc*);  } **else** {  query.setSort("product\_price", ORDER.*asc*);  }  //分页处理  **if** (**null** == page) page = 1;  **int** start = (page - 1) \* Global.*PAGE\_SIZE*;  query.setStart(start);  query.setRows(Global.*PAGE\_SIZE*);  //设置默认搜索域  query.set("df", "product\_keywords");  //高亮设置  query.setHighlight(**true**);  query.addHighlightField("product\_name");  query.setHighlightSimplePre("<span style=\"color:red\">");  query.setHighlightSimplePost("</span>");    //执行查询  ResultModel resultModel = productDao.queryProduct(query);  //计算总页数  Long recordCount = resultModel.getRecordCount();  **int** pages = (**int**) (recordCount / Global.*PAGE\_SIZE*);  **if** (recordCount % Global.*PAGE\_SIZE* > 0) {  pages++;  }  resultModel.setPageCount(pages);  resultModel.setCurPage(page);    **return** resultModel;  }  } |

# 7、Dao层，查询结果

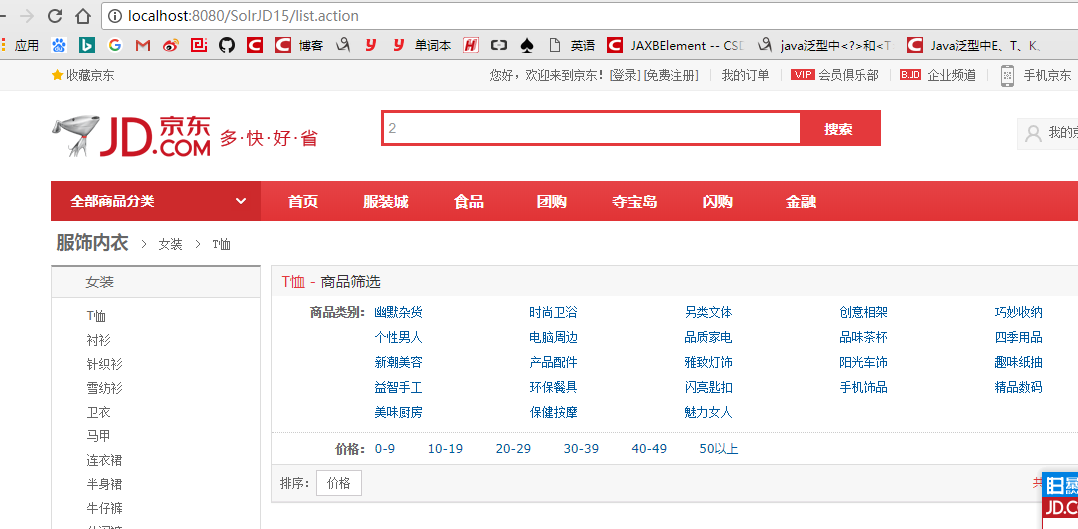
|  |
| --- |
| @Repository  **public** **class** ProductDaoImpl **implements** ProductDao {  @Autowired  **private** SolrServer solrServer;    @Override  **public** ResultModel queryProduct(SolrQuery query) **throws** Exception {  //执行查询  QueryResponse queryResponse = solrServer.query(query);  //取商品列表  SolrDocumentList solrDocumentList = queryResponse.getResults();  //商品列表  List<ProductModel> productList = **new** ArrayList<>();  //遍历商品列表  **for** (SolrDocument solrDocument : solrDocumentList) {  ProductModel productModel = **new** ProductModel();  productModel.setPid((String) solrDocument.get("id"));  //取高亮显示  Map<String, Map<String, List<String>>> highlighting = queryResponse.getHighlighting();  List<String> list = highlighting.get(solrDocument.get("id")).get("product\_name");  String productName = "";  **if** (**null** != list && list.size() > 0) {  productName = list.get(0);  } **else** {  productName = (String) solrDocument.get("product\_name");  }  productModel.setName(productName);  productModel.setCatalog\_name((String) solrDocument.get("product\_catalog\_name"));  productModel.setPrice((**float**) solrDocument.get("product\_price"));  productModel.setPicture((String) solrDocument.get("product\_picture"));  //添加到商品列表  productList.add(productModel);  }  //返回值对象  ResultModel resultModel = **new** ResultModel();  resultModel.setProductList(productList);  resultModel.setRecordCount(solrDocumentList.getNumFound());  **return** resultModel;  } |

# 8、jsp不介绍

# 9、springmvc.xml，需要配置solr，*HttpSolrServer*

|  |
| --- |
| <?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:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.1.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.1.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.1.xsd "*>  <!-- 配置扫描包 -->  <context:component-scan base-package=*"com.itheima.jd"*/>  <!-- 配置注解驱动 -->  <mvc:annotation-driven/>  <!-- jsp视图解析器 -->  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"* >  <!-- 前缀 -->  <property name=*"prefix"* value=*"/WEB-INF/jsp/"*></property>  <!-- 后缀 -->  <property name=*"suffix"* value=*".jsp"*></property>  </bean>  <bean class=*"org.apache.solr.client.solrj.impl.HttpSolrServer"*>  <constructor-arg value=*"http://localhost:8080/solr/"*></constructor-arg>  </bean>  </beans> |

# 10、可以访问，结果未实现，但是代码很简单，我也看懂了。相信大家也都看懂了



# 11、代码位置

