**品优购电商系统开发**

**第11章**

**搜索解决方案-Solr 【1】**

传智播客.黑马程序员

# 1.Solr安装与配置

## 1.1配置域

域相当于数据库的表字段，用户存放数据，因此用户根据业务需要去定义相关的Field（域），一般来说，每一种对应着一种数据，用户对同一种数据进行相同的操作。

域的常用属性：

* name：指定域的名称
* type：指定域的类型
* indexed：是否索引
* stored：是否存储
* required：是否必须
* multiValued：是否多值

### 1.1.1域(配置)

修改solrhome的schema.xml 文件 设置业务系统 Field

|  |
| --- |
| <field name="item\_goodsid" type="long" indexed="true" stored="true"/>  <field name="item\_title" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_price" type="double" indexed="true" stored="true"/>  <field name="item\_image" type="string" indexed="false" stored="true" />  <field name="item\_category" type="string" indexed="true" stored="true" />  <field name="item\_seller" type="text\_ik" indexed="true" stored="true" />  <field name="item\_brand" type="string" indexed="true" stored="true" />  <field name="item\_updatetime" type="date" indexed="true" stored="true" /> |

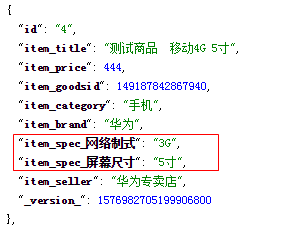
### 1.1.2复制域(配置)

复制域的作用在于将某一个Field中的数据复制到另一个域中

|  |
| --- |
| <field name="item\_keywords" type="text\_ik" indexed="true" stored="false" multiValued="true"/>  <copyField source="item\_title" dest="item\_keywords"/>  <copyField source="item\_category" dest="item\_keywords"/>  <copyField source="item\_seller" dest="item\_keywords"/>  <copyField source="item\_brand" dest="item\_keywords"/> |

### 1.1.3动态域(配置)

当我们需要动态扩充字段时，我们需要使用动态域。对于品优购，规格的值是不确定的，所以我们需要使用动态域来实现。需要实现的效果如下：



配置：

|  |
| --- |
| <dynamicField name="item\_spec\_\*" type="string" indexed="true" stored="true" /> |

# 2.Spring Data Solr入门

## 2.1 Spring Data Solr简介

虽然支持任何编程语言的能力具有很大的市场价值，你可能感兴趣的问题是：我如何将Solr的应用集成到Spring中？可以，Spring Data Solr就是为了方便Solr的开发所研制的一个框架，其底层是对SolrJ（官方API）的封装。

## 2.2 Spring Data Solr入门小Demo

### 2.2.1 搭建工程

1. 创建maven工程spring-demo-solr jar项目，pom.xml中引入依赖

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-solr</artifactId>  <version>1.5.5.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.2.4.RELEASE</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.9</version>  </dependency>  </dependencies>  <build>  <plugins>  <!-- java编译插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.2</version>  <configuration>  <source>1.7</source>  <target>1.7</target>  <encoding>UTF-8</encoding>  </configuration>  </plugin>  </plugins>  </build>  </build> |

（2）在src/main/resources下创建 applicationContext-solr.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:solr=*"http://www.springframework.org/schema/data/solr"*  xsi:schemaLocation=*"http://www.springframework.org/schema/data/solr*  *http://www.springframework.org/schema/data/solr/spring-solr-1.0.xsd*  *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"*>  <!-- solr服务器地址 -->  <solr:solr-server id=*"solrServer"* url=*"http://192.168.200.128:8080/solr"* />  <!-- solr模板，使用solr模板可对索引库进行CRUD的操作 -->  <bean id=*"solrTemplate"* class=*"org.springframework.data.solr.core.SolrTemplate"*>  <constructor-arg ref=*"solrServer"* />  </bean>  </beans> |

### 2.2.2 @Field 注解

创建 cn.itcast.pojo 包，将品优购的Item实体类拷入本工程 ,属性使用@Field注解标识 。 如果属性与配置文件定义的域名称不一致，需要在注解中指定域名称。

|  |
| --- |
| **public** **class** Item **implements** Serializable{  @Field  **private** Long id;  @Field("item\_title")  **private** String title;    @Field("item\_price")  **private** BigDecimal price;  @Field("item\_image")  **private** String image;  @Field("item\_goodsid")  **private** Long goodsId;  @Field("item\_category")  **private** String category;  @Field("item\_brand")  **private** String brand;  @Field("item\_seller")  **private** String seller;  .......  } |

### 2.2.3 增加（修改）

创建测试类TestTemplate.java

|  |
| --- |
| @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations="classpath:applicationContext-solr.xml")  **public** **class** TestTemplate {  @Autowired  **private** SolrTemplate solrTemplate;    @Test  **public** **void** testAdd(){  Item item=**new** Item();  item.setId(1L);  item.setBrand("华为");  item.setCategory("手机");  item.setGoodsId(1L);  item.setSeller("华为2号专卖店");  item.setTitle("华为Mate9");  item.setPrice(**new** BigDecimal(2000));  solrTemplate.saveBean(item);  solrTemplate.commit();  }  } |

### 2.2.4 按主键查询

|  |
| --- |
| @Test  **public** **void** testFindOne(){  Item item = solrTemplate.getById(1, Item.**class**);  System.***out***.println(item.getTitle());  } |

### 2.2.5 按主键删除

|  |
| --- |
| @Test  **public** **void** testDelete(){  solrTemplate.deleteById("1");  solrTemplate.commit();  } |

### 2.2.6 分页查询

首先循环插入100条测试数据

|  |
| --- |
| @Test  **public** **void** testAddList(){  List<Item> list=**new** ArrayList<Item>();  **for**(**int** i=0;i<100;i++){  Item item=**new** Item();  item.setId(i+1L);  item.setBrand("华为");  item.setCategory("手机");  item.setGoodsId(1L);  item.setSeller("华为2号专卖店");  item.setTitle("华为Mate"+i);  item.setPrice(**new** BigDecimal(2000+i));  list.add(item);  }  solrTemplate.saveBeans(list);  solrTemplate.commit();  } |

编写分页查询测试代码：

|  |
| --- |
| @Test  **public** **void** testPageQuery(){  Query query=**new** SimpleQuery("\*:\*");  query.setOffset(20);//开始索引（默认0）  query.setRows(20);//每页记录数(默认10)  ScoredPage<Item> page = solrTemplate.queryForPage(query, Item.**class**);  System.***out***.println("总记录数："+page.getTotalElements());  List<Item> list = page.getContent();  showList(list);  }  //显示记录数据  **private** **void** showList(List<Item> list){  **for**(Item item:list){  System.***out***.println(item.getTitle() +item.getPrice());  }  } |

### 2.2.7 条件查询

Criteria 用于对条件的封装：

|  |
| --- |
| @Test  **public** **void** testPageQueryMutil(){  Query query = **new** SimpleQuery("\*:\*");  Criteria criteria = **new** Criteria();  criteria = criteria.or("item\_title").contains("2");  criteria = criteria.or("item\_title").contains("5");  query.addCriteria(criteria);  ScoredPage<Item> page = solrTemplate.queryForPage(query, Item.**class**);  System.***out***.println("总记录数："+page.getTotalElements());  List<Item> list = page.getContent();  showList(list);  } |

### 2.2.8 删除全部数据

|  |
| --- |
| @Test  **public** **void** testDeleteAll(){  Query query=**new** SimpleQuery("\*:\*");  solrTemplate.delete(query);  solrTemplate.commit();  } |

# 3.品优购-批量数据导入

## 3.1需求分析

编写专门的导入程序，将商品数据导入到Solr系统中

## 3.2查询商品数据列表

### 3.2.1 工程搭建

（1）创建spring-demo-solrutil（jar） ,引入依赖dao项目

在spring-demo-solrutil项目src/main/resource/spring文件夹下创建applicationContext-solr.xml文件

|  |
| --- |
| <?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:solr=*"http://www.springframework.org/schema/data/solr"*  xsi:schemaLocation=*"http://www.springframework.org/schema/data/solr*  *http://www.springframework.org/schema/data/solr/spring-solr-1.0.xsd*  *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"*>    <!-- solr服务器地址 -->  <solr:solr-server id=*"solrServer"* url=*"http://192.168.200.128:8080/solr"* />    <!-- solr模板，使用solr模板可对索引库进行CRUD的操作 -->  <bean id=*"solrTemplate"* class=*"org.springframework.data.solr.core.SolrTemplate"*>  <constructor-arg ref=*"solrServer"* />  </bean>  </beans> |

### 3.2.2 代码编写

创建类SolrUtil ,实现商品数据的查询(开启商品)

|  |
| --- |
| /\*\*  \* 导入数据  \*  \* **@author** lx  \*  \*/  @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration(locations = { "classpath:applicationContext-solr.xml","classpath\*:spring/applicationContext\*.xml" })  **public** **class** SolrUtils {  @Autowired  **private** SolrTemplate solrTemplate;  @Autowired  **private** ItemDao itemDao;  // 导入所有数据  @Test  **public** **void** importSolr() **throws** Exception {  ItemQuery itemQuery = **new** ItemQuery();  itemQuery.createCriteria().andStatusEqualTo("1");  List<Item> items = itemDao.selectByExample(itemQuery);  **if** (**null** != items && items.size() > 0) {  **for** (Item item : items) {  Map<String, String> specMap = JSON.*parseObject*(item.getSpec(), Map.**class**);  item.setSpecMap(specMap);  }  solrTemplate.saveBeans(items);  solrTemplate.commit();  }  }  } |

### 3.3.3实体类

pojo项目中Item.java加入下面这些属性, 并创建对应的get和set方法

|  |
| --- |
| **public** **class** Item **implements** Serializable{  @Field  **private** Long id;  @Field("item\_title")  **private** String title;  @Field("item\_price")  **private** BigDecimal price;  @Field("item\_image")  **private** String image;  @Field("item\_goodsid")  **private** Long goodsId;  @Field("item\_category")  **private** String category;  @Field("item\_brand")  **private** String brand;  @Field("item\_seller")  **private** String seller;  @Field("item\_updatetime")  **private** Date updateTime;    @Dynamic  @Field("item\_spec\_\*")  **private** Map<String,String> specMap;    **public** Map<String, String> getSpecMap() {  **return** specMap;  }  **public** **void** setSpecMap(Map<String, String> specMap) {  **this**.specMap = specMap;  }  } |

# 4.品优购-关键字搜索

## 4.1需求分析

打开搜索页面，在搜索框输入要搜索的关键字，点击搜索按钮即可进行搜索，展示搜索结果



## 4.2后端代码

### 4.2.1服务接口层

（1）interface项目，创建业务接口

|  |
| --- |
| **public** **interface** ItemSearchService {  /\*\*  \* 搜索  \* **@param** keywords  \* **@return**  \*/  **public** Map<String,Object> search(Map searchMap);  } |

### 4.2.2服务实现层

（1）创建war工程search-service ，依赖于dao和interface项目

（2）拷贝资源中的“03配置文件\第一天搭建\search-service”中的配置文件到项目中。

（3）编写服务实现类ItemSearchServiceImpl.java

|  |
| --- |
| @Service  **public** **class** ItemSearchServiceImpl **implements** ItemSearchService{  @Autowired  **private** SolrTemplate solrTemplate;  @Override  **public** Map<String, Object> search(Map searchMap) {  Map<String,Object> map=**new** HashMap<>();  Query query=**new** SimpleQuery();  //添加查询条件  Criteria criteria=**new** Criteria("item\_keywords").is(searchMap.get("keywords"));  query.addCriteria(criteria);  ScoredPage<Item> page = solrTemplate.queryForPage(query, Item.**class**);  map.put("rows", page.getContent());  **return** map;  }  } |

### 4.2.3控制层

（1）在portal-web 工程中创建ItemSearchController.java控制器类

|  |
| --- |
| @RestController  @RequestMapping("/itemsearch")  **public** **class** ItemSearchController {  @Reference  **private** ItemSearchService itemSearchService;    @RequestMapping("/search")  **public** Map<String, Object> search(@RequestBody Map searchMap ){  **return** itemSearchService.search(searchMap);  }  } |

## 4.3前端代码

### 4.3.1拷贝资源

将下列资源拷贝至web-portal



将angularJS拷贝到插件文件夹

拷贝base.js到js文件夹

### 4.3.2服务层

portal-web工程创建searchService.js

|  |
| --- |
| //搜索服务层  app.service("searchService",**function**($http){  **this**.search=**function**(searchMap){  **return** $http.post('itemsearch/search.do',searchMap);  }  }); |

### 4.3.3控制层

portal-web工程searchController.js

|  |
| --- |
| app.controller('searchController',**function**($scope,searchService){  //搜索  $scope.search=**function**(){  searchService.search( $scope.searchMap ).success(  **function**(response){  $scope.resultMap=response;//搜索返回的结果  }  );  }  }); |

### 4.3.4页面

web-portal工程search.html

引入js

|  |
| --- |
| <script type=*"text/javascript"* src=*"plugins/angularjs/angular.min.js"*> </script>  <script type=*"text/javascript"* src=*"js/base.js"*> </script>  <script type=*"text/javascript"* src=*"js/service/searchService.js"*> </script>  <script type=*"text/javascript"* src=*"js/controller/searchController.js"*> </script> |

指定控制器

|  |
| --- |
| <body ng-app=*"pinyougou"* ng-controller=*"searchController"*> |

绑定搜索框

|  |
| --- |
| <div class=*"input-append"*>  <input type=*"text"* id=*"autocomplete"* ng-model=*"searchMap.keywords"* type=*"text"* class=*"input-error input-xxlarge"* />  <button class=*"sui-btn btn-xlarge btn-danger"* ng-click=*"search()"* type=*"button"*>搜索</button>  </div> |

循环显示数据

|  |
| --- |
| <li class=*"yui3-u-1-5"* ng-repeat=*"item in resultMap.rows"*>  <div class=*"list-wrap"*>  <div class=*"p-img"*>  <a href=*"item.html"* target=*"\_blank"*><img src=*"{{item.image}}"* /></a>  </div>  <div class=*"price"*>  <strong>  <em>¥</em>  <i>{{item.price}}</i>  </strong>  </div>  <div class=*"attr"*>  <em>{{item.title}}</em>  </div>  <div class=*"cu"*>  <em><span>促</span>满一件可参加超值换购</em>  </div>  <div class=*"commit"*>  <i class=*"command"*>已有2000人评价</i>  </div>  <div class=*"operate"*>  <a href=*"success-cart.html"* target=*"\_blank"* class=*"sui-btn btn-bordered btn-danger"*>加入购物车</a>  <a href=*"javascript:void(0);"* class=*"sui-btn btn-bordered"*>对比</a>  <a href=*"javascript:void(0);"* class=*"sui-btn btn-bordered"*>关注</a>  </div>  </div>  </li> |