**品优购电商系统开发**

**第11章**

**搜索解决方案-Solr 【2】**

传智播客.黑马程序员

# 1.品优购-高亮显示

## 1.1需求分析

将用户输入的关键字在标题中以红色的字体显示出来，就是搜索中常用的高亮显示.



## 1.2后端代码

修改服务层代码ItemSearchServiceImpl.java

创建私有方法，用于返回查询列表的结果（高亮）

|  |
| --- |
| /\*\*  \* 根据关键字搜索列表  \* **@param** keywords  \* **@return**  \*/  **private** Map searchList(Map searchMap){  Map map=**new** HashMap();  HighlightQuery query=**new** SimpleHighlightQuery();  HighlightOptions highlightOptions=**new** HighlightOptions().addField("item\_title");//设置高亮的域  highlightOptions.setSimplePrefix("<em style='color:red'>");//高亮前缀  highlightOptions.setSimplePostfix("</em>");//高亮后缀  query.setHighlightOptions(highlightOptions);//设置高亮选项  //按照关键字查询  Criteria criteria=**new** Criteria("item\_keywords").is(searchMap.get("keywords"));  query.addCriteria(criteria);  HighlightPage<Item> page = solrTemplate.queryForHighlightPage(query, Item.**class**);  **for**(HighlightEntry<Item> h: page.getHighlighted()){//循环高亮入口集合  Item item = h.getEntity();//获取原实体类  **if**(h.getHighlights().size()>0 && h.getHighlights().get(0).getSnipplets().size()>0){  item.setTitle(h.getHighlights().get(0).getSnipplets().get(0));//设置高亮的结果  }  }  map.put("rows",page.getContent());  **return** map;  } |

修改ItemSearchServiceImpl 的search方法，调用刚才编写的私有方法

|  |
| --- |
| @Override  **public** Map<String, Object> search(Map searchMap) {  Map<String,Object> map=**new** HashMap<>();  //1.查询列表  map.putAll(searchList(searchMap));  **return** map;  } |

## 1.3前端代码

我们测试后发现高亮显示的html代码原样输出，这是angularJS为了防止html攻击采取的安全机制。我们如何在页面上显示html的结果呢？我们会用到$sce服务的trustAsHtml方法来实现转换。

因为这个功能具有一定通用性，我们可以通过angularJS的过滤器来简化开发，这样只写一次，调用的时候就非常方便了，看代码：

1. 修改portal项目的base.js

|  |
| --- |
| // 定义模块:  **var** app = angular.module("pinyougou",[]);  /\*$sce服务写成过滤器\*/  app.filter('trustHtml',['$sce',**function**($sce){  **return** **function**(data){  **return** $sce.trustAsHtml(data);  }  }]); |

1. 使用过滤器, web-portal项目中的search.html页面.

ng-bind-html指令用于显示html内容

竖线 |用于调用过滤器

|  |
| --- |
| <div class=*"attr"* ng-bind-html=*"item.title | trustHtml"*>  <em>{{item.title}}</em>  </div> |

*|就是竖线，看起来有点斜是因为字体原因。*

# 2.搜索业务规则分析

## 2.1需求分析

我们今天要完成的目标是在关键字搜索的基础上添加面板搜索功能。

面板上有商品分类、品牌、各种规格和价格区间等条件



业务规则：

1. 当用户输入关键字搜索后，除了显示列表结果外，还应该显示通过这个关键字搜索到的记录都有哪些商品分类。
2. 根据第一个商品分类查询对应的模板，根据模板查询出品牌列表
3. 根据第一个商品分类查询对应的模板，根据模板查询出规格列表
4. 当用户点击搜索面板的商品分类时，显示按照这个关键字查询结果的基础上，筛选此分类的结果。
5. 当用户点击搜索面板的品牌时，显示在以上结果的基础上，筛选此品牌的结果
6. 当用户点击搜索面板的规格时，显示在以上结果的基础上，筛选此规格的结果
7. 当用户点击价格区间时，显示在以上结果的基础上，按价格进行筛选的结果
8. 当用户点击搜索面板的相应条件时，隐藏已点击的条件。

## 2.2实现思路

1. 搜索面板的商品分类需要使用Spring Data Solr的分组查询来实现
2. 为了能够提高查询速度，我们需要把查询面板的品牌、规格数据提前放入redis
3. 查询条件的构建、面板的隐藏需要使用angularJS来实现
4. 后端的分类、品牌、规格、价格区间查询需要使用过滤查询来实现

# 3.查询分类列表

## 3.1需求分析

根据搜索关键字查询商品分类名称列表



## 3.2后端代码

修改SearchItemServiceImpl.java创建方法

|  |
| --- |
| /\*\*  \* 查询分类列表  \* **@param** searchMap  \* **@return**  \*/  **private** List searchCategoryList(Map searchMap){  List<String> list=**new** ArrayList();  Query query=**new** SimpleQuery();  //按照关键字查询  Criteria criteria=**new** Criteria("item\_keywords").is(searchMap.get("keywords"));  query.addCriteria(criteria);  //设置分组选项  GroupOptions groupOptions=**new** GroupOptions().addGroupByField("item\_category");  query.setGroupOptions(groupOptions);  //得到分组页  GroupPage<Item> page = solrTemplate.queryForGroupPage(query, Item.**class**);  //根据列得到分组结果集  GroupResult<Item> groupResult = page.getGroupResult("item\_category");  //得到分组结果入口页  Page<GroupEntry<em>> groupEntries = groupResult.getGroupEntries();  //得到分组入口集合  List<GroupEntry<Item>> content = groupEntries.getContent();  **for**(GroupEntry<Item> entry:content){  list.add(entry.getGroupValue());//将分组结果的名称封装到返回值中  }  **return** list;  } |

search方法调用

|  |
| --- |
| @Override  **public** Map<String, Object> search(Map searchMap) {  Map<String,Object> map=**new** HashMap<>();  //1.按关键字查询（高亮显示）  ......  //2.根据关键字查询商品分类  List categoryList = searchCategoryList(searchMap);  map.put("categoryList",categoryList);  **return** map;  } |

## 3.3前端代码

修改search.html

|  |
| --- |
| <div class=*"type-wrap"* ng-if=*"resultMap.categoryList!=null"*>  <div class=*"fl key"*>商品分类</div>  <div class=*"fl value"*>  <span ng-repeat=*"category in resultMap.categoryList"*>  <a href=*"#"*>{{category}}</a>  </span>  </div>  <div class=*"fl ext"*></div>  </div> |

# 4.缓存品牌和规格数据

## 4.1需求分析

将商品分类数据、品牌数据、和规格数据都放入Redis存储。

1. 当用户进入运营商后台的商品分类页面时，将商品分类数据放入缓存（Hash）。以分类名称作为key ,以模板ID作为值
2. 当用户进入运营商后台的模板管理页面时，分别将品牌数据和规格数据放入缓存（Hash）。以模板ID作为key,以品牌列表和规格列表作为值。

## 4.2缓存商品分类数据

将商品分类表存入缓存 sellergoods工程需要引入pinyougou-common工程依赖。

修改sellergoods的ItemCatServiceImpl.java，添加redisTemplate

|  |
| --- |
| @Autowired  **private** RedisTemplate redisTemplate;    /\*\*  \* 根据上级ID查询列表  \*/  @Override  **public** List<ItemCat> findByParentId(Long parentId) {  TbItemCatExample example1=**new** TbItemCatExample();  Criteria criteria1 = example1.createCriteria();  criteria1.andParentIdEqualTo(parentId);  //每次执行查询的时候，一次性读取缓存进行存储 (因为每次增删改都要执行此方法)  List<ItemCat> list = findAll();  **for**(ItemCat itemCat:list){  redisTemplate.boundHashOps("itemCat").put(itemCat.getName(), itemCat.getTypeId());  }  System.***out***.println("更新缓存:商品分类表");  **return** itemCatMapper.selectByExample(example1);  } |

## 4.3缓存品牌和规格列表数据

（1）修改sellergoods的TypeTemplateServiceImpl.java

|  |
| --- |
| @Autowired  **private** RedisTemplate redisTemplate;    /\*\*  \* 将数据存入缓存  \*/  **private** **void** saveToRedis(){  //获取模板数据  List<TypeTemplate> typeTemplateList = findAll();  //循环模板  **for**(TbTypeTemplate typeTemplate :typeTemplateList){  //存储品牌列表  List<Map> brandList = JSON.*parseArray*(typeTemplate.getBrandIds(), Map.**class**);  redisTemplate.boundHashOps("brandList").put(typeTemplate.getId(), brandList);  //存储规格列表  List<Map> specList = findSpecList(typeTemplate.getId());//根据模板ID查询规格列表  redisTemplate.boundHashOps("specList").put(typeTemplate.getId(), specList);  }  } |

（2）在查询分页方法(findPage) 时调用此方法

|  |
| --- |
| **public** PageResult findPage(TypeTemplate typeTemplate, **int** pageNum, **int** pageSize) {  ......  saveToRedis();//存入数据到缓存  **return** **new** PageResult(page.getTotal(), page.getResult());  } |

这样在增删改后会自动调用该方法.

## 4.4加载缓存数据

启动redis ,运行运营商管理后台，打开商品分类和模板管理页，即可将数据放入缓存中。

# 5.显示品牌和规格数据

## 5.1需求分析

在搜索面板区域显示第一个分类的品牌和规格列表



## 5.2后端代码

修改ItemSearchServiceImpl.java ，增加方法

|  |
| --- |
| @Autowired  **private** RedisTemplate redisTemplate;    /\*\*  \* 查询品牌和规格列表  \* **@param** category 分类名称  \* **@return**  \*/  **private** Map searchBrandAndSpecList(String category){  Map map=**new** HashMap();  Long typeId = (Long) redisTemplate.boundHashOps("itemCat").get(category);//获取模板ID  **if**(typeId!=**null**){  //根据模板ID查询品牌列表  List brandList = (List) redisTemplate.boundHashOps("brandList").get(typeId);  map.put("brandList", brandList);//返回值添加品牌列表  //根据模板ID查询规格列表  List specList = (List) redisTemplate.boundHashOps("specList").get(typeId);  map.put("specList", specList);  }  **return** map;  } |

Search方法调用此方法

|  |
| --- |
| @Override  **public** Map<String, Object> search(Map searchMap) {  Map<String,Object> map=**new** HashMap<>();  //1.按关键字查询（高亮显示）  //2.根据关键字查询商品分类  //3.查询品牌和规格列表  **if**(categoryList.size()>0){  map.putAll(searchBrandAndSpecList(String.*valueOf*(categoryList.get(0))));  }  **return** map;  } |

## 5.3前端代码

### 5.3.1获取品牌列表

修改页面search.html，实现品牌列表

|  |
| --- |
| <div class=*"type-wrap logo"* ng-if=*"resultMap.brandList!=null"*>  <div class=*"fl key brand"*>品牌</div>  <div class=*"value logos"*>  <ul class=*"logo-list"*>  <li ng-repeat=*"brand in resultMap.brandList"*>  {{brand.text}}  </li>  </ul>  </div>  <div class=*"ext"*>  <a href=*"javascript:void(0);"* class=*"sui-btn"*>多选</a>  <a href=*"javascript:void(0);"*>更多</a>  </div>  </div> |

### 5.3.2获取规格列表

修改页面search.html，实现规格列表

|  |
| --- |
| <div class=*"type-wrap"* ng-repeat=*"spec in resultMap.specList"*>  <div class=*"fl key"*>{{spec.text}}</div>  <div class=*"fl value"*>  <ul class=*"type-list"*>  <li ng-repeat=*"option in spec.options"*>  <a>{{option.optionName}}</a>  </li>  </ul>  </div>  <div class=*"fl ext"*></div>  </div> |

# 6.过滤条件构建

## 6.1需求分析

点击搜索面板上的分类、品牌和规格，实现查询条件的构建。查询条件以面包屑的形式显示。

当面包屑显示分类、品牌和规格时，要同时隐藏搜索面板对应的区域。

用户可以点击面包屑上的X 撤销查询条件。撤销后显示搜索面包相应的区域。

## 6.2添加搜索项

### 6.2.1添加搜索项方法

修改portal-web的searchController.js

|  |
| --- |
| $scope.searchMap={'keywords':'','category':'','brand':'','spec':{}};//搜索对象  //添加搜索项  $scope.addSearchItem=**function**(key,value){  **if**(key=='category' || key=='brand'){//如果点击的是分类或者是品牌  $scope.searchMap[key]=value;  }**else**{  $scope.searchMap.spec[key]=value;  }  } |

### 6.2.2点击搜索项

修改portal-web 的search.html ，为搜索面板添加点击事件

点击商品分类标签

|  |
| --- |
| <div class=*"fl key"*>商品分类</div>  <div class=*"fl value"*>  <span ng-repeat=*"category in resultMap.categoryList"* >  <a href=*"#"* ng-click=*"addSearchItem('category',category)"*>{{category}}</a>  </span>  </div>  …………….. |

点击品牌标签

|  |
| --- |
| <div class=*"fl key brand"*>品牌</div>  <div class=*"value logos"*>  <ul class=*"logo-list"*>  <li ng-repeat=*"brand in resultMap.brandList"*>  <a href=*"#"* ng-click=*"addSearchItem('brand',brand.text)"*>{{brand.text}}</a>  </li>  </ul>  </div>  ………….. |

点击规格标签

|  |
| --- |
| <div class=*"type-wrap"* ng-repeat=*"spec in resultMap.specList"*>  <div class=*"fl key"*>{{spec.text}}</div>  <div class=*"fl value"*>  <ul class=*"type-list"*>  <li ng-repeat=*"option in spec.options"*>  <a href=*"#"* ng-click=*"addSearchItem(spec.text,option.optionName)"*>{{option.optionName}}</a>  </li>  </ul>  </div>  <div class=*"fl ext"*></div>  </div> |

### 6.2.3显示面包屑

修改web-portal 的search.html，用面包屑形式显示搜索条件

|  |
| --- |
| <div class=*"bread"*>  <ul class=*"fl sui-breadcrumb"*>  <li>  <a href=*"#"*>搜索条件：</a>  </li>  </ul>  <ul class=*"tags-choose"*>  <li class=*"tag"* ng-if=*"searchMap.category!=''"* >商品分类：{{searchMap.category}}<i class=*"sui-icon icon-tb-close"*></i></li>  <li class=*"tag"* ng-if=*"searchMap.brand!=''"* >品牌：{{searchMap.brand}}<i class=*"sui-icon icon-tb-close"*></i></li>  <li class=*"tag"* ng-repeat=*"(key,value) in searchMap.spec"* >{{key}}：{{value}}<i class=*"sui-icon icon-tb-close"*></i></li>  </ul>  <div class=*"clearfix"*></div>  </div> |

## 6.3撤销搜索项

### 6.3.1撤销搜索项的方法

修改portal-web工程searchController.js

|  |
| --- |
| //移除复合搜索条件  $scope.removeSearchItem=**function**(key){  **if**(key=="category" || key=="brand"){//如果是分类或品牌  $scope.searchMap[key]="";  }**else**{//否则是规格  **delete** $scope.searchMap.spec[key];//移除此属性  }  } |

### 6.3.2页面调用方法

portal-web工程的search.html

|  |
| --- |
| <div class=*"bread"*>  <ul class=*"fl sui-breadcrumb"*>  <li>  <a href=*"#"*>搜索条件：</a>  </li>  </ul>  <ul class=*"tags-choose"*>  <li class=*"tag"* ng-if=*"searchMap.category!=''"* ng-click=*"removeSearchItem('category')"*>商品分类：{{searchMap.category}}<i class=*"sui-icon icon-tb-close"*></i></li>  <li class=*"tag"* ng-if=*"searchMap.brand!=''"* ng-click=*"removeSearchItem('brand')"*>品牌：{{searchMap.brand}}<i class=*"sui-icon icon-tb-close"*></i></li>  <li class=*"tag"* ng-repeat=*"(key,value) in searchMap.spec"* ng-click=*"removeSearchItem(key)"*>{{key}}：{{value}}<i class=*"sui-icon icon-tb-close"*></i></li>    </ul>  <div class=*"clearfix"*></div>  </div> |

## 6.4隐藏查询面板

### 6.4.1隐藏分类面板

修改search.html

|  |
| --- |
| <div class=*"type-wrap"* ng-if=*"resultMap.categoryList!=null && searchMap.category==''"*>  <div class=*"fl key"*>商品分类</div>  ......  </div> |

### 6.4.2隐藏品牌面板

修改search.html

|  |
| --- |
| <div class=*"type-wrap logo"* ng-if=*"resultMap.brandList!=null && searchMap.brand==''"*>  <div class=*"fl key brand"*>品牌</div>  .......  </div> |

### 6.4.3隐藏规格面板

修改search.html

|  |
| --- |
| <div class=*"type-wrap"* ng-repeat=*"spec in resultMap.specList"* ng-if=*"searchMap.spec[spec.text]==null"*>  <div class=*"fl key"*>{{spec.text}}</div>  ......  </div> |

## 6.5提交查询

修改searchController.js 在添加和删除筛选条件时自动调用搜索方法

|  |
| --- |
| //添加复合搜索条件  $scope.addSearchItem=**function**(key,value){  **if**(key=="category" || key=="brand"){//如果是分类或品牌  $scope.searchMap[key]=value;  }**else**{//否则是规格  $scope.searchMap.spec[key]=value;  }  $scope.search();//执行搜索  }  //移除复合搜索条件  $scope.removeSearchItem=**function**(key){  **if**(key=="category" || key=="brand"){//如果是分类或品牌  $scope.searchMap[key]="";  }**else**{//否则是规格  **delete** $scope.searchMap.spec[key];//移除此属性  }  $scope.search();//执行搜索  } |

# 7.过滤查询

## 7.1需求分析

根据上一步构建的查询条件，实现分类、品牌和规格的过滤查询



## 7.2代码实现

这里过滤分为, 分类过滤, 规格过滤, 品牌过滤, 根据分类查询品牌规格列表

规格过滤:

实现思路：规格有多项，需要循环过滤。循环规格查询条件，根据key得到域名城，根据value设置过滤条件。

修改search-service工程的SearchItemServiceImpl.java

|  |
| --- |
| **private** Map searchList(Map searchMap){  Map map=**new** HashMap();  //高亮选项初始化  HighlightQuery query=**new** SimpleHighlightQuery();  HighlightOptions highlightOptions=**new** HighlightOptions().addField("item\_title");//高亮域  highlightOptions.setSimplePrefix("<em style='color:red'>");//前缀  highlightOptions.setSimplePostfix("</em>");  query.setHighlightOptions(highlightOptions);//为查询对象设置高亮选项    //1.1 关键字查询  Criteria criteria=**new** Criteria("item\_keywords").is(searchMap.get("keywords"));  query.addCriteria(criteria);    //1.2 按商品分类过滤  **if**(!"".equals(searchMap.get("category")) ) {//如果用户选择了分类  FilterQuery filterQuery=**new** SimpleFilterQuery();  Criteria filterCriteria=**new** Criteria("item\_category").is(searchMap.get("category"));  filterQuery.addCriteria(filterCriteria);  query.addFilterQuery(filterQuery);  }    //1.3 按品牌过滤  **if**(!"".equals(searchMap.get("brand")) ) {//如果用户选择了品牌  FilterQuery filterQuery=**new** SimpleFilterQuery();  Criteria filterCriteria=**new** Criteria("item\_brand").is(searchMap.get("brand"));  filterQuery.addCriteria(filterCriteria);  query.addFilterQuery(filterQuery);  }  //1.4 按规格过滤  **if**(searchMap.get("spec")!=**null**){  Map<String,String> specMap= (Map<String, String>) searchMap.get("spec");  **for**(String key :specMap.keySet()){    FilterQuery filterQuery=**new** SimpleFilterQuery();  Criteria filterCriteria=**new** Criteria("item\_spec\_"+key).is( specMap.get(key) );  filterQuery.addCriteria(filterCriteria);  query.addFilterQuery(filterQuery);    }    }        //\*\*\*\*\*\*\*\*\*\*\* 获取高亮结果集 \*\*\*\*\*\*\*\*\*\*\*  //高亮页对象  HighlightPage<Item> page = solrTemplate.queryForHighlightPage(query, Item.**class**);  //高亮入口集合(每条记录的高亮入口)  List<HighlightEntry<Item>> entryList = page.getHighlighted();  **for**(HighlightEntry<Item> entry:entryList ){  //获取高亮列表(高亮域的个数)  List<Highlight> highlightList = entry.getHighlights();  /\*  for(Highlight h:highlightList){  List<String> sns = h.getSnipplets();//每个域有可能存储多值  System.out.println(sns);  }\*/  **if**(highlightList.size()>0 && highlightList.get(0).getSnipplets().size()>0 ){  Item item = entry.getEntity();  item.setTitle(highlightList.get(0).getSnipplets().get(0));  }  }  map.put("rows", page.getContent());  **return** map;  } |