谷粒商城

版本：V 1.0



**当前的位置**

# 一、利用elasticSearch开发电商的搜索列表功能

## 1 检索的页面

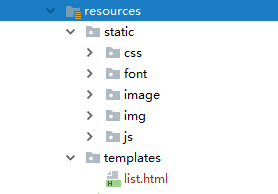
检索功能



### 1.1 为gmall-list-web模块添加静态页面

#### 1.1.1 静态网页及资源文件

拷贝静态文件到resources目录下，手工建立static和templates目录



### 1.2 sku列表功能



首先是根据关键字、属性值、分类Id、页码查询sku列表。

#### 1.2.1 ListController

|  |
| --- |
| @RequestMapping(**"list.html"**) **public** String getList( SkuLsParams skuLsParams, Model model){   *//根据参数返回sku列表* SkuLsResult skuLsResult = **listService**.search(skuLsParams);  model.addAttribute(**"skuLsInfoList"**,skuLsResult.getSkuLsInfoList());  **return "list"**;  } |

#### 1.2.2 页面html渲染

|  |
| --- |
| <**div style="width**:215**px" th:each="skuLsInfo:${skuLsInfoList}"** >  <**p class="da"**>  <**a href="#" th:onclick="'javascript:item('+${skuLsInfo.id}+')'"**>  <**img th:src="${skuLsInfo.skuDefaultImg}" src="img/57d0d400Nfd249af4.jpg" class="dim"**>  </**a**>  </**p**>   <**p class="tab\_R"**>  <**span th:text="'￥'+${#numbers.formatDecimal(skuLsInfo.price,1,2)}"**>¥5199.00</**span**>  </**p**>   <**a href="#" title="" th:onclick="'javascript:item('+${skuLsInfo.id}+')'" class="tab\_JE" th:utext="${skuLsInfo.skuName}"** >  Apple iPhone 7 Plus (A1661) 32G 黑色 移动联通电信4G手机  </**a**>  </**div**> |

要注意的是其中skuName中因为关键字标签所以必须要用utext否则标签会被转义。

#### 1.2.3 搜索栏相关html

|  |
| --- |
| *<!--搜索导航-->* <**div class="header\_sous"**>  <**div class="logo"**>  <**a href="#"**><**img src="/image/jdlogo-201708-@1x.png" alt=""**></**a**>  </**div**>  <**div class="header\_form"**>  <**input id="keyword" name="keyword" type="text" placeholder="手机"** />  <**a href="#" onclick="***searchList*()**"**>搜索</**a**>  </**div**>  <**div class="header\_ico"**>  <**div class="header\_gw"**>  <**span**><**a href="#"**>我的购物车</**a**></**span**>  <**img src="/image/settleup-@1x.png"** />  <**span**>0</**span**>  </**div**>  <**div class="header\_ko"**>  <**p**>购物车中还没有商品，赶紧选购吧！</**p**>  </**div**>  </**div**> |

#### 1.2.4 js代码

|  |
| --- |
| **function** *searchList*(){  **var** keyword = **$**(**"#keyword"**).val();  **window**.**location**.**href**=**"/list.html?keyword="**+keyword; }  **function** *item*(skuid) {  **window**.**location**.**href**=**"http://localhost:8084/"**+skuid+**".html"**; } |

这时可以看到列表效果了。

### 1.3 页面功能—提供可供选择的属性列表

#### 1.3.1 思路：

这个列表有两种情况

1. 如果是通过首页的3级分类点击进入的，要按照分类Id查询对应的属性和属性值列表。
2. 如果是直接用搜索栏输入文字进入的，要根据sku的查询结果涉及的属性值(好在我们已经通过es的聚合取出来了)，再去查询数据库把文字列表显示出来。

#### 1.3.2 ListController

数据来源于：base\_attr\_info ,base\_attr\_value

|  |
| --- |
| *//根据查询的结果返回属性和属性值列表* @Reference ListService **listService**; @Reference **private** ManageService **manageService**;  @RequestMapping(**"list.html"**) **public** String getList(SkuLsParams skuLsParams, Model model){  SkuLsResult skuLsResult = **listService**.search(skuLsParams);   *// 获取sku属性值列表* List<SkuLsInfo> skuLsInfoList = skuLsResult.getSkuLsInfoList();  model.addAttribute(**"skuLsInfoList"**,skuLsInfoList);  *// 从结果中取出平台属性值列表* List<String> attrValueIdList = skuLsResult.getAttrValueIdList();  List<BaseAttrInfo> attrList = **manageService**.getAttrList(attrValueIdList);  model.addAttribute(**"attrList"**,attrList);  *//return JSON.toJSONString(search);* **return "list"**; } |

其中**manageService**.getAttrList(String catalog3Id)这个方法我们已经有现成的了。

但是**manageService**.getAttrList(attrValueIdList) 这个方法我们要新添加。

#### 1.3.3 在ManageServiceImpl中增加方法

|  |
| --- |
| @Override **public** List<BaseAttrInfo> getAttrList(List<String> attrValueIdList) {  String attrValueIds = StringUtils.*join*(attrValueIdList.toArray(), **","**);  List<BaseAttrInfo> baseAttrInfoList = **baseAttrInfoMapper**.selectAttrInfoListByIds(attrValueIds);  **return** baseAttrInfoList;  } |

#### 1.3.4 BaseAttrInfoMapper.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>* <mapper namespace="com.atguigu.gmall.manage.mapper.BaseAttrInfoMapper">  <**select id="selectAttrInfoListByIds" resultMap="baseAttrInfoMap"**>  SELECT ai.id,ai.attr\_name,ai.catalog3\_id, av.id attr\_value\_id ,av.attr\_id ,av.value\_name  FROM base\_attr\_info ai INNER JOIN base\_attr\_value av ON ai.id=av.attr\_id  WHERE av.id IN (${valueIds}) </**select**>  </mapper> |

注意这里面没有用#{}是因为attrValueIds 是两个数字用逗号分开的，所以不能整体套上单引，所以使用${}。

#### 1.3.5 BaseAttrInfoMapper.class

|  |
| --- |
| **public interface** BaseAttrInfoMapper **extends** Mapper<BaseAttrInfo> {  List<BaseAttrInfo> selectAttrInfoListByIds(@Param(**"valueIds"**) String valueIds); } |

此处必须要用@Param注解否则${ }无法识别。

#### 1.3.6 点击属性值的链接

getAttrList(List<String> attrValueIdList)方法实现后，还有一个问题就是，点击属性时，要把上次查询的内容也带上，即带上历史参数。

|  |
| --- |
| 在controller中添加方法拼接条件方法  **public** String makeUrlParam(SkuLsParams skuLsParam){  String urlParam=**""**;  **if**(skuLsParam.getKeyword()!=**null**){  urlParam+=**"keyword="**+skuLsParam.getKeyword();  }  **if** (skuLsParam.getCatalog3Id()!=**null**){  **if** (urlParam.length()>0){  urlParam+=**"&"**;  }  urlParam+=**"catalog3Id="**+skuLsParam.getCatalog3Id();  }  *// 构造属性参数* **if** (skuLsParam.getValueId()!=**null** && skuLsParam.getValueId().**length**>0){  **for** (**int** i=0;i<skuLsParam.getValueId().**length**;i++){  String valueId = skuLsParam.getValueId()[i];  **if** (urlParam.length()>0){  urlParam+=**"&"**;  }  urlParam+=**"valueId="**+valueId;  }  }  **return** urlParam; } |
| @RequestMapping(**"list.html"**) **public** String getList(SkuLsParams skuLsParams, Model model){  SkuLsResult skuLsResult = **listService**.search(skuLsParams);  *// 从结果中取出平台属性值列表* List<String> attrValueIdList = skuLsResult.getAttrValueIdList();  List<BaseAttrInfo> attrList = **manageService**.getAttrList(attrValueIdList);   *// 已选的属性值列表\* String urlParam = makeUrlParam(skuLsParams);  *// itco* **for** (Iterator<BaseAttrInfo> iterator = attrList.iterator(); iterator.hasNext(); ) {  BaseAttrInfo baseAttrInfo = iterator.next();  List<BaseAttrValue> attrValueList = baseAttrInfo.getAttrValueList();  **for** (BaseAttrValue baseAttrValue : attrValueList) {  **if**(skuLsParams.getValueId()!=**null**&&skuLsParams.getValueId().**length**>0){  **for** (String valueId : skuLsParams.getValueId()) {  *//选中的属性值 和 查询结果的属性值* **if**(valueId.equals(baseAttrValue.getId())){  iterator.remove();  }  }  }  }  }  model.addAttribute(**"urlParam"**,urlParam);  model.addAttribute(**"attrList"**,attrList);  *// 获取sku属性值列表* List<SkuLsInfo> skuLsInfoList = skuLsResult.getSkuLsInfoList();  model.addAttribute(**"skuLsInfoList"**,skuLsInfoList);  *//return JSON.toJSONString(search);* **return "list"**; } |

#### 1.3.7 生成属性列表的html部分

|  |
| --- |
| <div class="GM\_selector">  *<!--手机商品筛选-->* <div class="title">  <h3><em>商品筛选</em></h3>   </div>  <div class="GM\_nav\_logo">   <div class="GM\_pre" th:each="attrInfo:${attrList}">  <div class="sl\_key">  <span th:text="${attrInfo.attrName}+':'">属性：</span>  </div>  <div class="sl\_value">  <ul>  <li th:each="attrValue:${attrInfo.attrValueList}"><a th:href="'/list.html?'+${urlParam}+'&valueId='+${attrValue.id}" th:text="${attrValue.valueName}">属性值</a></li>  </ul>  </div>  </div>  </div> </div> |

完成后



### 1.4 页面功能--面包屑



面包屑导航是为了能够让用户清楚的知道当前页面的所在位置和筛选条件的功能。但是这个小的人性化功能却有点麻烦。

功能点： 1、点击某个属性值的时候对应的那行属性要消失掉不能再次选择。

2、列在上面的属性面包屑，要可以取消掉，恢复到没选择之前。

#### 1.4.1 思路：

1. 把本应显示的列表与用户已选择的属性值列表用循环交叉判断，如果匹配把本应显示的那个属性去掉。
2. 已选择的属性值列表，要携带点击跳转的路径，这个路径参数就是咱们上边讲的那个“历史参数”，但是要把自己本身的属性值去掉。
3. 重构BaseAttrValue实体类

|  |
| --- |
| @Transient  private String urlParam; |

#### 1.4.2 重构makeUrlParam方法

将多余的条件去除！

|  |
| --- |
| **public** String makeUrlParam(SkuLsParams skuLsParam,**String... excludeValueIds**){  String urlParam=**""**;  List<String> paramList = **new** ArrayList<>();  **if**(skuLsParam.getKeyword()!=**null**){  urlParam+=**"keyword="**+skuLsParam.getKeyword();  }  **if** (skuLsParam.getCatalog3Id()!=**null**){  **if** (urlParam.length()>0){  urlParam+=**"&"**;  }  urlParam+=**"catalog3Id="**+skuLsParam.getCatalog3Id();  }  *// 构造属性参数* **if** (skuLsParam.getValueId()!=**null** && skuLsParam.getValueId().**length**>0){  **for** (**int** i=0;i<skuLsParam.getValueId().**length**;i++){  String valueId = skuLsParam.getValueId()[i];  if (excludeValueIds!=null && excludeValueIds.length>0){  **String excludeValueId = excludeValueIds[0];**  if (excludeValueId.equals(valueId)){  *// 跳出代码，后面的参数则不会继续追加【后续代码不会执行】*  *// 不能写break；如果写了break；其他条件则无法拼接！* continue;  }  }  **if** (urlParam.length()>0){  urlParam+=**"&"**;  }  urlParam+=**"valueId="**+valueId;  }  }  **return** urlParam; }  } |

#### 1.4.3 controller中的getList方法增加

|  |
| --- |
| 代码如下用红色标识部分！@RequestMapping(**"list.html"**) **public** String getList(SkuLsParams skuLsParams, Model model){  SkuLsResult skuLsResult = **listService**.search(skuLsParams);  *// 从结果中取出平台属性值列表* List<String> attrValueIdList = skuLsResult.getAttrValueIdList();  List<BaseAttrInfo> attrList = **manageService**.getAttrList(attrValueIdList);  *// 已选的属性值列表* **List<BaseAttrValue> baseAttrValuesList = new ArrayList<>();**  String urlParam = makeUrlParam(skuLsParams);  *// itco* **for** (Iterator<BaseAttrInfo> iterator = attrList.iterator(); iterator.hasNext(); ) {  BaseAttrInfo baseAttrInfo = iterator.next();  List<BaseAttrValue> attrValueList = baseAttrInfo.getAttrValueList();  **for** (BaseAttrValue baseAttrValue : attrValueList) {  baseAttrValue.setUrlParam(urlParam);  **if**(skuLsParams.getValueId()!=**null**&&skuLsParams.getValueId().**length**>0){  **for** (String valueId : skuLsParams.getValueId()) {  *//选中的属性值 和 查询结果的属性值* **if**(valueId.equals(baseAttrValue.getId())){  iterator.remove();   ***// 构造面包屑列表* BaseAttrValue baseAttrValueSelected = new BaseAttrValue();  baseAttrValueSelected.setValueName(baseAttrInfo.getAttrName()+":"+baseAttrValue.getValueName());  *// 去除重复数据* String makeUrlParam = makeUrlParam(skuLsParams, valueId);  baseAttrValueSelected.setUrlParam(makeUrlParam);  baseAttrValuesList.add(baseAttrValueSelected);**  }  }  }  }  }  *// 保存面包屑清单* **model.addAttribute("baseAttrValuesList",baseAttrValuesList);  model.addAttribute("keyword", skuLsParams.getKeyword());  model.addAttribute("urlParam",urlParam);**  model.addAttribute(**"attrList"**,attrList);  *// 获取sku属性值列表* List<SkuLsInfo> skuLsInfoList = skuLsResult.getSkuLsInfoList();  model.addAttribute(**"skuLsInfoList"**,skuLsInfoList);  *//return JSON.toJSONString(search);* **return "list"**; } |

这块代码看似多层循环嵌套性能隐患，其实因为单次循环基本不会超过五次，循环中没有网络或者io访问，完全在虚拟机中运行，所以即使多层循环嵌套压力也不会太大。

#### 1.4.4 页面代码

|  |
| --- |
| <**div class="GM\_ipone"**>  <**div class="GM\_ipone\_bar"**>  <**div class="GM\_ipone\_one a"**>  筛选条件  </**div**>  <**i**><**img src="/image/right-@1x.png" alt=""**></**i**>  <**span th:if="${keyword}!=null" th:text="'&quot;'+${keyword}+'&quot;'"**>**&quot;**小米**&quot;**</**span**>  <**a class="select-attr" th:each="baseAttrValue: ${baseAttrValuesList}" th:utext="${baseAttrValue.valueName}+'<b> ✖ </b>'" th:href="'/list.html?'+${baseAttrValue.urlParam}" href="#"** > 2G<**b**> ✖ </**b**>  </**a**>  *<!--<a class="select-attr" href="#" > 屏幕尺寸:5.1-5.5英寸<b> ✖ </b>-->  <!--</a>-->* </**div**> </**div**> |

### 1.5 分页：在ListController中添加如下代码

|  |
| --- |
| *// 设置每页显示的条数* skuLsParams.setPageSize(2);  model.addAttribute(**"totalPages"**, skuLsResult.getTotalPages()); model.addAttribute(**"pageNo"**,skuLsParams.getPageNo()); |

## 2 排序

页面结构完成了，考虑一下如何排序，es查询的dsl语句中我们是用了hotScore来进行排序的。

但是hotScore从何而来，根据业务去定义，也可以扩展更多类型的评分，让用户去选择如何排序。

这里的hotScore我们假定以点击量来决定热度。

那么我们每次用户点击，将这个评分+1,不就可以了么。

### 2.1 问题：

1、 es大量的写操作会影响es 性能，因为es需要更新索引，而且es不是内存数据库，会做相应的io操作。

2、而且修改某一个值，在高并发情况下会有冲突，造成更新丢失，需要加锁，而es的乐观锁会恶化性能问题。

从业务角度出发，其实我们为商品进行排序所需要的热度评分，并不需要非常精确，大致能比出个高下就可以了。

利用这个特点我们可以稀释掉大量写操作。

### **2.2 解决思路：**

用redis做精确计数器，redis是内存数据库读写性能都非常快，利用redis的原子性的自增可以解决并发写操作。

redis每计10次数（可以被10整除）我们就更新一次es ，这样写操作就被稀释了1倍，这个倍数可以根据业务情况灵活设定。

代码 在listServiceImpl中增加更新操作

### 在gmall-list-service更新redis计数器

gmall-list-service在配置文件中添加配置

|  |
| --- |
| **spring.redis.host**=**192.168.67.202 spring.redis.port**=**6379 spring.redis.database**=**0** |

注意：在启动类上添加扫描注解：

|  |
| --- |
| @ComponentScan(basePackages = **"com.atguigu.gmall1128"**) |

|  |
| --- |
| 接口：  **public void** incrHotScore(String skuId);  在ListServiceImpl实现类中添加方法  *//更新热度评分* @Override **public void** incrHotScore(String skuId){  Jedis jedis = **redisUtil**.getJedis();  **int** timesToEs=10;  Double hotScore = jedis.zincrby(**"hotScore"**, 1, **"skuId:"** + skuId);  **if**(hotScore%timesToEs==0){  updateHotScore(skuId, Math.*round*(hotScore));  }  } |

### 2.4 代码：更新es

|  |
| --- |
| **private void** updateHotScore(String skuId,Long hotScore){  String updateJson=**"{\n"** +  **" \"doc\":{\n"** +  **" \"hotScore\":"**+hotScore+**"\n"** +  **" }\n"** +  **"}"**;   Update update = **new** Update.Builder(updateJson).index(**"gmall\_sku\_info"**).type(**"\_doc"**).id(skuId).build();  **try** {  **jestClient**.execute(update);  } **catch** (IOException e) {  e.printStackTrace();  } } |

### 2.5 详情页调用

这个incrHotScore 方法可以在进入详情页面的时候调用。

|  |
| --- |
| @Controller **public class** ItemController {   @Reference  ListService **listService**;   @Reference  ManageService **manageService**;   @RequestMapping(**"/{skuId}.html"**)  **public** String getSkuInfo(@PathVariable(**"skuId"**) String skuId, Model model){  ……  **listService**.incrHotScore(skuId); *//最终应该由异步方式调用*  } |