**品优购电商系统开发**

**第11章**

**搜索解决方案-ElasticSearch**

传智播客.黑马程序员

# 课程目标

目标1：能够实现高亮显示效果

目标2：完成条件过滤构建功能

目标3：完成过滤查询的功能

目标4：完成分页动态页码展示

目标5：实现搜索结果排序功能

目标6：实现搜索页与首页对接功能

# 1.搜索结果【高亮】

## 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：



**Spring-data-elasticsearch常用API说明：**

* **ElasticsearchTemplate** 操作ES索引库核心类

-- elasticsearchTemplate.queryForPage()查询分页方法

* **NativeSearchQueryBuilder** 原生搜索条件构建

-- withQuery() 添加搜索条件

-- withHighlightFields() 添加高亮字段

* **HighlightBuilder.Field** 高亮字段

-- preTags() 高亮前缀

-- postTags() 高亮后缀

-- fragmentSize() 文本截断

* **AggregatePage** 合计分页

-- getTotalElements() 总记录数

-- getContent() 分页数据

## 1.2后端代码

pinyougou-search-service/src/main/java/com/pinyougou/search/service.impl/ItemSearchServiceImpl.java

|  |
| --- |
| */\*\* 搜索方法 \*/* @Override **public** Map<String, Object> search(Map<String,Object> params) {  */\*\* 获取查询关键字 \*/* String keywords = (String)params.get(**"keywords"**);   *// 创建原生的搜索条件构建对象* NativeSearchQueryBuilder builder = **new** NativeSearchQueryBuilder();  *// 设置默认搜索全部* builder.withQuery(QueryBuilders.*matchAllQuery*());   */\*\* 判断检索关键字是否为空 \*/* **if** (StringUtils.*isNoneBlank*(keywords)) {  *// 设置根据多条件匹配查询:* builder.withQuery(QueryBuilders.*multiMatchQuery*(keywords,  "title", "category", "brand", "seller"));  *// 创建高亮字段对象* HighlightBuilder.Field field = new HighlightBuilder  .Field("title") *// 设置需要高亮的字段* .preTags("<font color='red'>") *// 设置高亮前缀* .postTags("</font>") *// 设置高亮后缀* .fragmentSize(50); *// 设置文本截断  // 设置高亮字段对象* builder.withHighlightFields(field);  }   *// 构建搜索查询对象* SearchQuery query = builder.build();   *// 分页查询，得到合计分页对象* AggregatedPage<EsItem> page = **esTemplate**.queryForPage(query, EsItem.**class**,  **new** SearchResultMapper() { *// 搜索结果转化* @Override  **public** <T> AggregatedPage<T> mapResults(SearchResponse sr,  Class<T> aClass, Pageable pageable) {  *// 定义List集合封装分页结果* List<T> content = **new** ArrayList<>();  *// 循环封装分页结果* **for** (SearchHit hit : sr.getHits()) {  *// 获取搜索的文档json字符串，转化成EsItem对象* EsItem esItem = JSON.*parseObject*(hit.getSourceAsString(), EsItem.**class**);  *// 获取标题高亮对象* HighlightField highlightTitle = hit.getHighlightFields().get(**"title"**);  *// 判断是否有标题高亮对象* **if** (highlightTitle != **null**){  *// 获取标题高亮内容字符串* String title = highlightTitle.getFragments()[0].toString();  *// 设置标题高亮内容* esItem.setTitle(title);  }  *// 添加到新的集合* content.add((T)esItem);  }  **return new** AggregatedPageImpl<T>(content,  pageable, sr.getHits().getTotalHits());  }  });  *// 定义Map集合封装返回数据* Map<String,Object> data = **new** HashMap<>();  *// 设置总记录数* data.put(**"total"**, page.getTotalElements());  *// 设置分页结果* data.put(**"rows"**, page.getContent());  **return** data; } |

## 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VueJS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？我们要用到v-html。

v-html指令用于显示html内容

修改pinyougou-search-web\src\main\webapp\index.html页面(299行)

|  |
| --- |
| <**div class="attr" style="line-height**: 20**px"**>  <**em v-html="item.title"**></**em**> </**div**> |

# 2.搜索业务规则分析

## 2.1需求分析

面板上有商品分类、品牌、各种规格、价格区间等过滤条件



业务规则：

1. 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。
2. 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
3. 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
4. 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
5. 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. 查询条件的构建、面板的隐藏需要使用VueJS来实现
2. 后端的商品分类、品牌、规格选项、价格区间查询需要使用过滤查询来实现

# 3.前端封装过滤条件

## 3.1需求分析

点击搜索面板上的分类、品牌、规格、价格区间，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、规格、价格区间时，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。撤销后显示搜索面板相应的区域。

## 3.2添加搜索选项

### 3.2.1绑定点击事件

修改pinyougou-search-web/src/main/webapp/index.html页面，为搜索面板添加点击事件

点击商品分类标签(165多行)：

|  |
| --- |
| <**div class="fl value"**>  <**a href="javascript:;" style="margin-right**: 10**px"  @click="addSearchItem('category','手机')"**>手机</**a**> </**div**> |

点击品牌标签(172多行)：

|  |
| --- |
| <**ul class="logo-list"**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','苹果')"**>  <**img src="img/\_/phone01.jpg"**/></**li**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','三星')"**>  <**img src="img/\_/phone02.jpg"**/></**li**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','华为')"**>  <**img src="img/\_/phone03.jpg"**/></**li**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','小米')"**>  <**img src="img/\_/phone04.jpg"**/></**li**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','OPPO')"**>  <**img src="img/\_/phone05.jpg"**/></**li**>  <**li style="cursor**: **pointer"  @click="addSearchItem('brand','VIVO')"**>  <**img src="img/\_/phone06.jpg"**/></**li**> </**ul**> |

点击网络标签(201多行)：

|  |
| --- |
| <**li**>  <**a @click="addSearchItem('网络','移动3G')"**>移动3G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('网络','联通3G')"**>联通3G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('网络','电信3G')"**>电信3G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('网络','移动4G')"**>移动4G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('网络','联通4G')"**>联通4G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('网络','电信4G')"**>电信4G</**a**> </**li**> |

点击机身内存标签(226多行)：

|  |
| --- |
| <**li**>  <**a @click="addSearchItem('机身内存','16G')"**>16G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('机身内存','32G')"**>32G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('机身内存','64G')"**>64G</**a**> </**li**> <**li**>  <**a @click="addSearchItem('机身内存','128G')"**>128G</**a**> </**li**> |

点击价格区间标签(246行)：

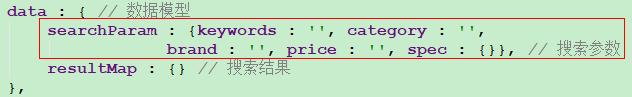


|  |
| --- |
| <**li**>  <**a @click="addSearchItem('price','0-500')"**>0-500元</**a**> </**li**> <**li**>  <**a @click="addSearchItem('price','500-1000')"**>500-1000元</**a**> </**li**> <**li**>  <**a @click="addSearchItem('price','1000-1500')"**>1000-1500元</**a**> </**li**> <**li**>  <**a @click="addSearchItem('price','1500-2000')"**>1500-2000元</**a**> </**li**> <**li**>  <**a @click="addSearchItem('price','2000-3000')"**>2000-3000元 </**a**> </**li**> <**li**>  <**a @click="addSearchItem('price','3000-\*')"**>3000元以上</**a**> </**li**> |

### 3.2.2添加搜索选项方法

pinyougou-search-web/src/main/webapp/js/controller/searchController.js：

* 在data数据模型中定义变量



* 在methods中定义方法

|  |
| --- |
| *// 添加搜索选项* **addSearchItem** : **function** (key, value) {  */\*\* 判断是商品分类、品牌、价格 \*/* **if**(key == **'category'** || key == **'brand'** || key == **'price'**){  **this**.**searchParam**[key] = value;  }**else**{  */\*\* 规格选项 \*/* **this**.**searchParam**.**spec**[key] = value;  }  *// 执行搜索* **this**.search(); } |

### 3.2.3显示面包屑

修改pinyougou-search-web/src/main/webapp/index.html页面，用面包屑形式显示搜索条件(147行)

|  |
| --- |
| *<!--bread-->* <**div class="bread"**>  <**ul class="fl sui-breadcrumb"**>  <**li**>搜索条件</**li**>  </**ul**>  <**ul class="tags-choose"**>  <**li class="tag" v-if="searchParam.category != ''"**>  商品分类：{{**searchParam**.**category**}}  <**i class="sui-icon icon-tb-close"**></**i**>  </**li**>  <**li class="tag" v-if="searchParam.brand != ''"**>  品牌：{{**searchParam**.**brand**}}  <**i class="sui-icon icon-tb-close"**></**i**>  </**li**>  <**li class="tag" v-for="(value,key) in searchParam.spec"**>  {{***key***}} : {{***value***}}  <**i class="sui-icon icon-tb-close"**></**i**>  </**li**>  <**li class="tag" v-if="searchParam.price != ''"**>  价格：{{**searchParam**.**price**}}  <**i class="sui-icon icon-tb-close"**></**i**>  </**li**>  </**ul**> </**div**> *<!--selector-->* |

## 3.3删除搜索选项

### 3.3.1绑定点击事件

修改pinyougou-search-web/src/main/webapp/index.html页面

|  |
| --- |
| <**ul class="tags-choose"**>  <**li class="tag" v-if="searchParam.category != ''"**>  商品分类：{{**searchParam**.**category**}}  <**i class="sui-icon icon-tb-close"  @click="removeSearchItem('category')"**></**i**>  </**li**>  <**li class="tag" v-if="searchParam.brand != ''"**>  品牌：{{**searchParam**.**brand**}}  <**i class="sui-icon icon-tb-close"  @click="removeSearchItem('brand')"**></**i**>  </**li**>  <**li class="tag" v-for="(value,key) in searchParam.spec"**>  {{***key***}} : {{***value***}}  <**i class="sui-icon icon-tb-close"  @click="removeSearchItem(key)"**></**i**>  </**li**>  <**li class="tag" v-if="searchParam.price != ''"**>  价格：{{**searchParam**.**price**}}  <**i class="sui-icon icon-tb-close"  @click="removeSearchItem('price')"**></**i**>  </**li**> </**ul**> |

### 3.3.2删除搜索选项方法

pinyougou-search-web/src/main/webapp/js/controller/searchController.js，在methods中定义方法：

|  |
| --- |
| *// 删除搜索选项* **removeSearchItem** : **function** (key) {  */\*\* 判断是商品分类、品牌、价格 \*/* **if**(**key** == **'category'** || **key** == **'brand'** || **key** == **'price'**){  **this**.**searchParam**[**key**] = **""**;  }**else**{  */\*\* 规格选项 \*/* **delete this**.**searchParam**.**spec**[**key**];  }  *// 执行搜索* **this**.search(); } |

## 3.4隐藏查询面板

### 3.4.1隐藏商品分类面板

修改pinyougou-search-web/src/main/webapp/index.html(174行)

|  |
| --- |
| <**div class="type-wrap" v-if="searchParam.category == ''"**>  <**div class="fl key"**>商品分类</**div**>  ......  <**div class="fl ext"**></**div**> </**div**> |

### 3.4.2隐藏品牌面板

修改pinyougou-search-web/src/main/webapp/index.html(182行)

|  |
| --- |
| <**div class="type-wrap logo" v-if="searchParam.brand == ''"**>  <**div class="fl key brand"**>品牌</**div**>  <**div class="value logos"**>  ......  </**div**> </**div**> |

### 3.4.3隐藏网络面板

修改pinyougou-search-web/src/main/webapp/index.html(208行)

|  |
| --- |
| <**div class="type-wrap" v-if="searchParam.spec['网络'] == null"**>  <**div class="fl key"**>网络</**div**>  ......  <**div class="fl ext"**></**div**> </**div**> |

### 3.4.4隐藏机身内存面板

修改pinyougou-search-web/src/main/webapp/index.html(235行)

|  |
| --- |
| <**div class="type-wrap" v-if="searchParam.spec['机身内存'] == null"**>  <**div class="fl key"**>机身内存</**div**>  ......  <**div class="fl ext"**></**div**> </**div**> |

### 3.4.5隐藏价格区间面板

修改pinyougou-search-web/src/main/webapp/index.html(255行)

|  |
| --- |
| <**div class="type-wrap" v-if="searchParam.price == ''"**>  <**div class="fl key"**>价格</**div**>  ......  <**div class="fl ext"**></**div**> </**div**> |

# 4.搜索结果【过滤】

## 4.1需求分析

根据上一步构建的搜索条件，按分类、品牌、规格、价格区间实现过滤查询

## 4.2实现步骤

### 4.2.1创建组合查询构建对象

创建BoolQueryBuilder组合查询构建对象，组合多个过滤条件

|  |
| --- |
| */\*\* ############# 过滤查询 ########### \*/ // 1. 创建组合查询构建对象，封装多个过滤条件* BoolQueryBuilder boolBuilder = QueryBuilders.*boolQuery*(); |

### 4.2.2组合多个过滤条件

组合(分类、品牌、规格、价格)过滤条件，采用must方式组合

#### 4.2.2.1分类过滤

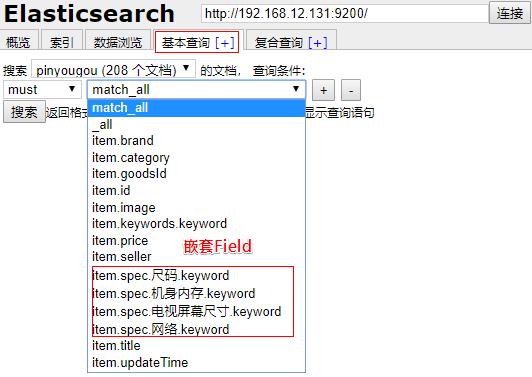
|  |
| --- |
| *// 2.1 按商品分类过滤* String category = (String)params.get(**"category"**); **if** (StringUtils.*isNoneBlank*(category)){  *// 词条查询* boolBuilder.must(QueryBuilders.*termQuery*(**"category"**, category)); } |

#### 4.2.2.2品牌过滤

|  |
| --- |
| *// 2.2 按商品品牌过滤* String brand = (String)params.get(**"brand"**); **if** (StringUtils.*isNoneBlank*(brand)){  *// 词条查询* boolBuilder.must(QueryBuilders.*termQuery*(**"brand"**, brand)); } |

#### 4.2.2.3规格过滤

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



|  |
| --- |
| *// 2.3 按商品规格过滤* Map<String,String> specMap = (Map<String, String>)params.get(**"spec"**); **if** (specMap != **null** && specMap.size() > 0){  *// 迭代Map集合* **for** (String key : specMap.keySet()) {  *// 嵌套Field的名称* String field = **"spec."** + key + **".keyword"**;  *// 嵌套查询* boolBuilder.must(QueryBuilders.***nestedQuery***(**"spec"**,  QueryBuilders.*termQuery*(field, specMap.get(key)),  ScoreMode.***Max***));  } } |

#### 4.2.2.4价格过滤

|  |
| --- |
| *// 2.4 按商品价格区间过滤* String price = (String)params.get(**"price"**); **if** (StringUtils.*isNoneBlank*(price)) {  *// 得到价格区间数组* String[] priceArr = price.split(**"-"**);  *// 范围查询构建对象* RangeQueryBuilder rqBuilder = QueryBuilders.*rangeQuery*(**"price"**);  *// 如果价格结束为星号* **if** (**"\*"**.equals(priceArr[1])){  *// 大于起始价格 3000-\** rqBuilder.gte(priceArr[0]);  }**else**{  *// 从起始价格 到 结束价格* rqBuilder.from(priceArr[0]).to(priceArr[1]);  }  *// 组合范围查询* boolBuilder.must(rqBuilder); } |

### 4.2.3原生搜索查询添加过滤查询

|  |
| --- |
| *// 3. 添加过滤查询(组合查询对象)* builder.withFilter(boolBuilder); |

## 4.3完整代码

pinyougou-search-service\src\main\java\com\pinyougou\search\service\impl\ItemSearchServiceImpl.java，修改search()方法：

|  |
| --- |
| */\*\* 搜索方法 \*/* @Override **public** Map<String, Object> search(Map<String,Object> params) {  */\*\* 获取查询关键字 \*/* String keywords = (String)params.get(**"keywords"**);  ......  */\*\* 判断检索关键字是否为空 \*/* **if** (StringUtils.*isNoneBlank*(keywords)) {  *// 设置根据多条件匹配查询*  *// 创建高亮字段对象*  ......}  */\*\* ############# 过滤查询 ########### \*/* ***// 1. 创建组合查询构建对象，封装多个过滤条件*** BoolQueryBuilder boolBuilder = QueryBuilders.*boolQuery*();   ***// 2. 组合多个过滤条件(必须的方式组合)*** ***// 2.1 按商品分类过滤*** String category = (String)params.get(**"category"**);  **if** (StringUtils.*isNoneBlank*(category)){  *// 词条查询* boolBuilder.must(QueryBuilders.*termQuery*(**"category"**, category));  }  ***// 2.2 按商品品牌过滤*** String brand = (String)params.get(**"brand"**);  **if** (StringUtils.*isNoneBlank*(brand)){  *// 词条查询* boolBuilder.must(QueryBuilders.*termQuery*(**"brand"**, brand));  }  ***// 2.3 按商品规格过滤*** Map<String,String> specMap = (Map<String, String>)params.get(**"spec"**);  **if** (specMap != **null** && specMap.size() > 0){  *// 迭代Map集合* **for** (String key : specMap.keySet()) {  *// 嵌套Field的名称* String field = **"spec."** + key + **".keyword"**;  *// 嵌套查询* boolBuilder.must(QueryBuilders.*nestedQuery*(**"spec"**,  QueryBuilders.*termQuery*(field, specMap.get(key)),  ScoreMode.***Max***));  }  }  ***// 2.4 按商品价格区间过滤*** String price = (String)params.get(**"price"**);  **if** (StringUtils.*isNoneBlank*(price)) {  *// 得到价格区间数组* String[] priceArr = price.split(**"-"**);  *// 范围查询构建对象* RangeQueryBuilder rqBuilder = QueryBuilders.*rangeQuery*(**"price"**);  *// 如果价格结束为星号* **if** (**"\*"**.equals(priceArr[1])){  *// 大于起始价格 3000-\** rqBuilder.gt(priceArr[0]);  }**else**{  *// 从起始价格 到 结束价格* rqBuilder.from(priceArr[0]).to(priceArr[1]);  }  *// 组合范围查询* boolBuilder.must(rqBuilder);  }   ***// 3. 原生搜索查询对象添加过滤查询*** **builder.withFilter(boolBuilder);**   *// 构建搜索查询对象* SearchQuery query = builder.build();  *// 分页查询，得到合计分页对象*  ......**return** data; } |

# 5.搜索结果【分页】

## 5.1需求分析

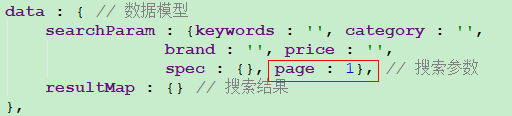
在上述功能基础上实现分页查询(index.html页面)：

IMG_256

## 5.2分页参数

需求：页面需要定义当前页码。

pinyougou-search-web/src/main/webapp/js/controller/searchController.js，在data数据模型的searchParam对象中，增加分页变量：



## 5.3后端代码

pinyougou-search-service/src/main/java/com/pinyougou/search/service/impl/ItemSearchServiceImpl.java，搜索方法中增加分页代码：

|  |
| --- |
| */\*\* 搜索方法 \*/* @Override **public** Map<String, Object> search(Map<String,Object> params) {  */\*\* 获取查询关键字 \*/* String keywords = (String)params.get(**"keywords"**);  ......  */\*\* 判断检索关键字是否为空 \*/* **if** (StringUtils.*isNoneBlank*(keywords)) {  ......  }  */\*\* ############# 过滤查询 ########### \*/  // 1. 创建组合查询构建对象，封装多个过滤条件*  *// 2. 组合多个过滤条件(必须的方式组合)  // 2.1 按商品分类过滤  // 2.2 按商品品牌过滤  // 2.3 按商品规格过滤  // 2.4 按商品价格区间过滤  // 3. 原生搜索查询对象添加过滤查询* builder.withFilter(boolBuilder);  *// 构建搜索查询对象* SearchQuery query = builder.build();  ***/\*\* ############# 分页查询 ########### \*/  // 1. 获取当前页面* Integer curPage = (Integer) params.get("page");  if (curPage == null){  curPage = 1;  }  *// 2. 设置分页对象 (注意，页码从0开始)* query.setPageable(PageRequest.*of*(curPage - 1, 20));**   *// 分页查询，得到合计分页对象* AggregatedPage<EsItem> page = **esTemplate**.queryForPage(query,  EsItem.**class**,**new** SearchResultMapper() {......});  *// 定义Map集合封装返回数据* Map<String,Object> data = **new** HashMap<>();  *// 设置总记录数* data.put(**"total"**, page.getTotalElements());  *// 设置分页结果* data.put(**"rows"**, page.getContent());   ***// 设置总页数* data.put("totalPages", page.getTotalPages());**  **return** data; } |

## 5.4前端代码

### 5.4.1显示页码

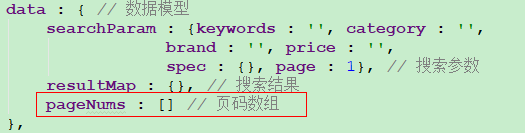
pinyougou-search-web/src/main/webapp/index.html页面，循环产生页码(352行)：

|  |
| --- |
| <**div class="sui-pagination pagination-large"**>  <**ul**>  <**li class="prev disabled"**>  <**a href="#"**>«上一页</**a**>  </**li**>  **<li :class="searchParam.page == p ? 'active' : ''"**  **v-for="p in pageNums">**  <**a href="#"**>{{***p***}}</**a**>  </**li**>  <**li class="next"**>  <**a href="#"**>下一页»</**a**>  </**li**>  </**ul**>  <**div**>  <**span**>共{{**resultMap**.totalPages}}页**&nbsp;**</**span**>  <**span**>到第<**input type="text" class="page-num"**>页  <**button class="page-confirm" onclick="***alert*(1)**"**>确定</**button**>  </**span**>  </**div**> </**div**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js，搜索后调用初始化页码方法：

|  |
| --- |
| *// 商品搜索* search : **function** () {  *// 发送异步请求* **axios**.post(**"/search"**, **this**.**searchParam**)  .then(**function**(response){  *// 获取搜索结果: {}* vue.**resultMap** = response.**data**;  ***/\*\* 调用初始化页码方法 \*/* vue.initPageNum();**  }); }, |

在data数据模型中定义页码数组变量



在methods中，定义初始化页码方法：

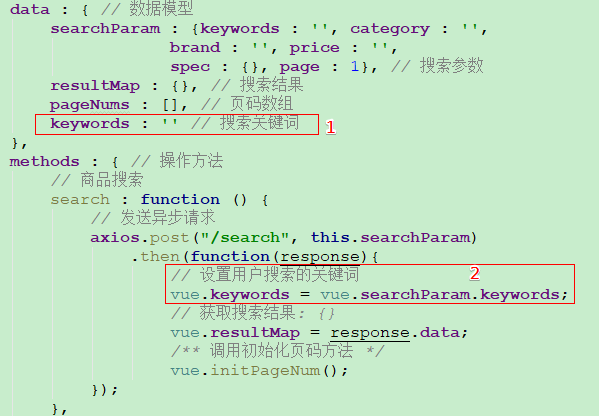
|  |
| --- |
| *// 初始化页码* initPageNum : **function** () {  */\*\* 定义页码数组 \*/* **this**.**pageNums** = [];  */\*\* 获取总页数 \*/* **var** totalPages = **this**.**resultMap**.totalPages;  */\*\* 开始页码 \*/* **var** firstPage = 1;  */\*\* 结束页码 \*/* **var** lastPage = totalPages;  */\*\* 如果总页数大于5，显示部分页码 \*/* **if** (totalPages > 5){  *// 如果当前页码处于前面位置* **if** (**this**.**searchParam**.**page** <= 3){  lastPage = 5; *// 生成前5页页码* }  *// 如果当前页码处于后面位置* **else if** (**this**.**searchParam**.**page** >= totalPages - 3){  firstPage = totalPages - 4; *// 生成后5页页码* }**else**{ *// 当前页码处于中间位置* firstPage = **this**.**searchParam**.**page** - 2;  lastPage = **this**.**searchParam**.**page** + 2;  }  }  */\*\* 循环产生页码 \*/* **for** (**var** i = firstPage; i <= lastPage; i++){  **this**.**pageNums**.push(i);  } } |

### 5.4.2显示搜索条件和总记录

index.html页面145行：

|  |
| --- |
| <**ul class="fl sui-breadcrumb"**>  <**li**>  按"<**span style="color**: **red**;**"**>{{**keywords**}}</**span**>"关键字，  搜索到 <**b**>{{**resultMap**.total}}</**b**>条记录.  </**li**> </**ul**> <**ul class="tags-choose"**>  ......  **</ul>** |

pinyougou-search-web\src\main\webapp\js\controller\searchController.js，修改data与search方法



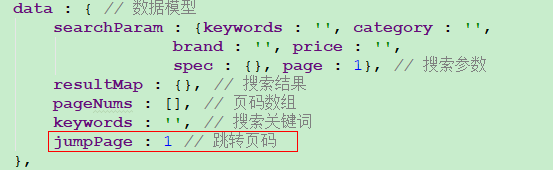
### 5.4.3根据页码查询

修改pinyougou-search-web/src/main/webapp/index.html页面，点击页码调用方法(350行)：

|  |
| --- |
| <**div class="sui-pagination pagination-large"**>  <**ul**>  <**li class="prev disabled"**>  <**a href="javascript:;"  @click="pageSearch(searchParam.page - 1);"**>«上一页</**a**>  </**li**>  <**li :class="searchParam.page == p ? 'active' : ''"  v-for="p in pageNums"**>  <**a href="javascript:;"  @click="pageSearch(p);"**>{{***p***}}</**a**>  </**li**>  <**li class="dotted"**><**span**>...</**span**></**li**>  <**li class="next"**>  <**a href="javascript:;"  @click="pageSearch(searchParam.page + 1);"**>下一页»</**a**>  </**li**>  </**ul**>  <**div**>  <**span**>共<**b**>{{**resultMap**.totalPages}}</**b**>页**&nbsp;**</**span**>  <**span**>到第<**input type="text" style="width**: 25**px**;**"  v-model="jumpPage"**>页  <**button class="page-confirm"  @click="pageSearch(jumpPage);"**>确定</**button**>  </**span**>  </**div**> </**div**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js：

* 在data数据模型中定义变量



* 在methods中定义分码搜索方法

|  |
| --- |
| *// 根据页码搜索* pageSearch: **function** (page) {  page = *parseInt*(page);  */\*\* 页码验证 \*/* **if** (page >= 1 && page <= **this**.**resultMap**.totalPages  && page != **this**.**searchParam**.**page**){  **this**.**searchParam**.**page** = page;  **this**.**jumpPage** = page;  **this**.search();  } } |

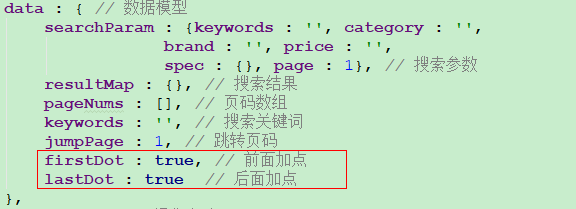
### 5.4.4显示省略号

修改index.html页面，页码前后加省略号(356行)：

|  |
| --- |
| <**li class="dotted" v-if="firstDot"**><**span**>...</**span**></**li**> <**li class="{{searchParam.page == *p* ? 'active' : ''}}"  ng-repeat="p in pageNums"**>  <**a href="javascript:;" ng-click="pageSearch(p);"**>{{***p***}}</**a**> </**li**> <**li class="dotted" v-if="lastDot"**><**span**>...</**span**></**li**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js:

* 在data数据模型中定义变量



* 修改methods中的initPageNum方法

|  |
| --- |
| *// 初始化页码* initPageNum : **function** () {  */\*\* 定义页码数组 \*/* **this**.**pageNums** = [];  */\*\* 获取总页数 \*/* **var** totalPages = **this**.**resultMap**.totalPages;  */\*\* 开始页码 \*/* **var** firstPage = 1;  */\*\* 结束页码 \*/* **var** lastPage = totalPages;  **this**.**firstDot** = **true**; *// 前面有点* **this**.**lastDot** = **true**; *// 后面有点  /\*\* 如果总页数大于5，显示部分页码 \*/* **if** (totalPages > 5){  *// 如果当前页码处于前面位置* **if** (**this**.**searchParam**.**page** <= 3){  lastPage = 5; *// 生成前5页页码* **this**.**firstDot** = **false**; *// 前面没点* }  *// 如果当前页码处于后面位置* **else if** (**this**.**searchParam**.**page** >= totalPages - 3){  firstPage = totalPages - 4; *// 生成后5页页码* **this**.**lastDot** = **false**; *// 后面没点* }**else**{ *// 当前页码处于中间位置* firstPage = **this**.**searchParam**.**page** - 2;  lastPage = **this**.**searchParam**.**page** + 2;  }  }**else** {  **this**.**firstDot** = **false**; *// 前面没点* **this**.**lastDot** = **false**; *// 后面没点* }  */\*\* 循环产生页码 \*/* **for** (**var** i = firstPage; i <= lastPage; i++){  **this**.**pageNums**.push(i);  } }, |

### 5.4.5页码不可用样式

修改index.html页面，页码不可用样式(351行)：

|  |
| --- |
| <**ul**>  <**li :class="searchParam.page == 1 ? 'prev disabled' : 'prev'"**>  <**a href="javascript:;"  @click="pageSearch(searchParam.page - 1);"**>«上一页</**a**>  </**li**>  <**li class="dotted" v-if="firstDot"**><**span**>...</**span**></**li**>  <**li :class="searchParam.page == p ? 'active' : ''"  v-for="p in pageNums"**>  <**a href="javascript:;"  @click="pageSearch(p);"**>{{***p***}}</**a**>  </**li**>  <**li class="dotted" v-if="lastDot"**><**span**>...</**span**></**li**>  <**li :class="searchParam.page == resultMap.totalPages ? 'next disabled' : 'next'"**>  <**a href="javascript:;"  @click="pageSearch(searchParam.page + 1);"**>下一页»</**a**>  </**li**> </**ul**> |

### 5.4.6搜索起始页码处理

测试：如果先按照“手机”关键字进行搜索，得出的页数是10页，然后点击第10页进行查询，然后再根据“小米”关键字搜索，会发现没有结果显示。是因为当前页仍然为10，而小米的结果只有5页，所以无法显示。我们需要在每次点击查询时将页码设置为1。

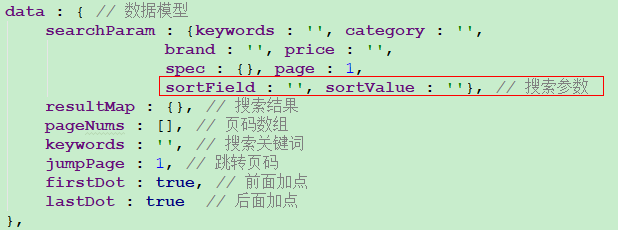
修改index.html页面，搜索按钮(80行)，调用搜索前将起始页码设置为1

|  |
| --- |
| <**button class="sui-btn btn-xlarge btn-danger"  @click="searchParam.page=1;search();"  type="button"**>搜索</**button**> |

# 6.搜索结果【排序】

## 6.1排序参数

pinyougou-search-web/src/main/webapp/js/controller/searchController.js，在data中的searchParam中增加排序参数:



## 6.2后端代码

pinyougou-search-service/src/main/java/com/pinyougou/search/service.impl/ItemSearchServiceImpl.java，增加排序代码：

|  |
| --- |
| */\*\* 搜索方法 \*/* @Override **public** Map<String, Object> search(Map<String,Object> params) {   */\*\* 获取查询关键字 \*/* String keywords = (String)params.get(**"keywords"**);  */\*\* ############# 过滤查询 ########### \*/  /\*\* ############# 分页查询 ########### \*/*  ...... */\*\* ############# 排序查询 ########### \*/  // 1. 获取排序参数* String sortValue = (String) params.get(**"sortValue"**); *// ASC DESC* String sortField = (String) params.get(**"sortField"**); *// 排序字段* **if**(StringUtils.*isNoneBlank*(sortValue)  && StringUtils.*isNoneBlank*(sortField)){  *// 2. 创建排序对象* Sort sort = **new** Sort(**"ASC"**.equalsIgnoreCase(sortValue) ?  Sort.Direction.***ASC*** : Sort.Direction.***DESC***, sortField);  *// 3. 设置添加排序* query.addSort(sort);  }  *// 分页查询，得到合计分页对象*  ......**return** data; } |

## 6.3按价格排序

实现价格的排序（升降序可切换index.html）



修改pinyougou-search-web/src/main/webapp/index.html页面(293行)：

|  |
| --- |
| <**div class="navbar-inner filter"**>  <**ul class="sui-nav"**>  <**li :class="searchParam.sortField == '' ? 'active' : ''"**>  <**a href="javascript:;"  @click="sortSearch('','');"**>综合</**a**>  </**li**>  <**li**>  <**a href="javascript:;"**>销量</**a**>  </**li**>  <**li**>  <**a href="javascript:;"**>新品</**a**>  </**li**>  <**li**>  <**a href="javascript:;"**>评价</**a**>  </**li**>  <**li :class="searchParam.sortField == 'price' && searchParam.sortValue == 'ASC' ? 'active' : ''"**>  <**a href="javascript:;"  @click="sortSearch('price','ASC');"**>价格↑</**a**>  </**li**>  <**li :class="searchParam.sortField == 'price' && searchParam.sortValue == 'DESC' ? 'active' : ''"**>  <**a href="javascript:;"  @click="sortSearch('price','DESC');"**>价格↓</**a**>  </**li**>  </**ul**> </**div**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js，在methods中，定义排序搜索方法：

|  |
| --- |
| *// 排序搜索* sortSearch : **function** (sortField, sortValue) {  **this**.**searchParam**.**sortField** = sortField;  **this**.**searchParam**.**sortValue** = sortValue;  **this**.search(); } |

## 6.4按上架时间排序



修改pinyougou-search-web/src/main/webapp/index.html页面(302行)：

|  |
| --- |
| <**li :class="searchParam.sortField == 'updateTime' ? 'active' : ''"**>  <**a href="javascript:;"  @click="sortSearch('updateTime','DESC');"**>新品</**a**> </**li**> |

## 6.5按销量排序（实现思路）

1. 增加域item\_salecount 用于存储每个SKU的销量数据。
2. 编写定时器程序，用于更新每个SKU的销量数据（查询近1个月的销量数据，不是累计数据）。
3. 定时器每天只需执行一次，可以设定为凌晨开始执行。

## 6.6按评价排序（实现思路）

与按销量排序思路基本相同，有一个细节需要注意：

评论分为好评、中评、差评，我们不能简单地将评论数相加，而是应该根据每种评论加权进行统计。比如好评的权重是3 ，中评的权重是1，而差评的权重是 -3，这样得出的是评价的综合得分。

# 7.首页与搜索页对接

## 7.1需求分析

用户在首页的搜索框输入关键字，点击搜索后自动跳转到搜索页查询

## 7.2前端代码

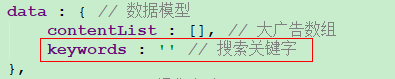
### 7.2.1首页传入关键字

修改pinyougou-portal-web/src/main/webapp/index.html(71行)：

|  |
| --- |
| *<!--searchAutoComplete-->* <**div class="input-append"**>  <**input type="text" id="autocomplete"  v-model="keywords"  class="input-error input-xxlarge"**/>  <**button class="sui-btn btn-xlarge btn-danger"  @click="search();" type="button"**>搜索</**button**> </**div**> |

pinyougou-portal-web/src/main/webapp/js/controller/indexController.js：

* 在data数据模型中定义变量



* 在methods中定义事件方法

|  |
| --- |
| *// 搜索按钮事件* search : **function** () {  ***location***.href = **"http://search.pinyougou.com?keywords="** + **this**.**keywords**; } |

### 7.2.2搜索页接收关键字

pinyougou-search-web/src/main/webapp/index.html页面，引入js

|  |
| --- |
| <**script src="js/location.js"**></**script**> |

pinyougou-search-web/src/main/webapp/js/controller/searchController.js:

* 在methods中定义方法

|  |
| --- |
| *// 初始化搜索* initSearch : **function** () {  *// 获取请求URL后面的参数* **var** keywords = **this**.getUrlParam(**"keywords"**);  **this**.**searchParam**.**keywords** = keywords ? keywords : **''**;  **this**.search(); }, |

* 修改created创建生命周期

