第4章 搜索解决方案-2 过滤查询

学习目标

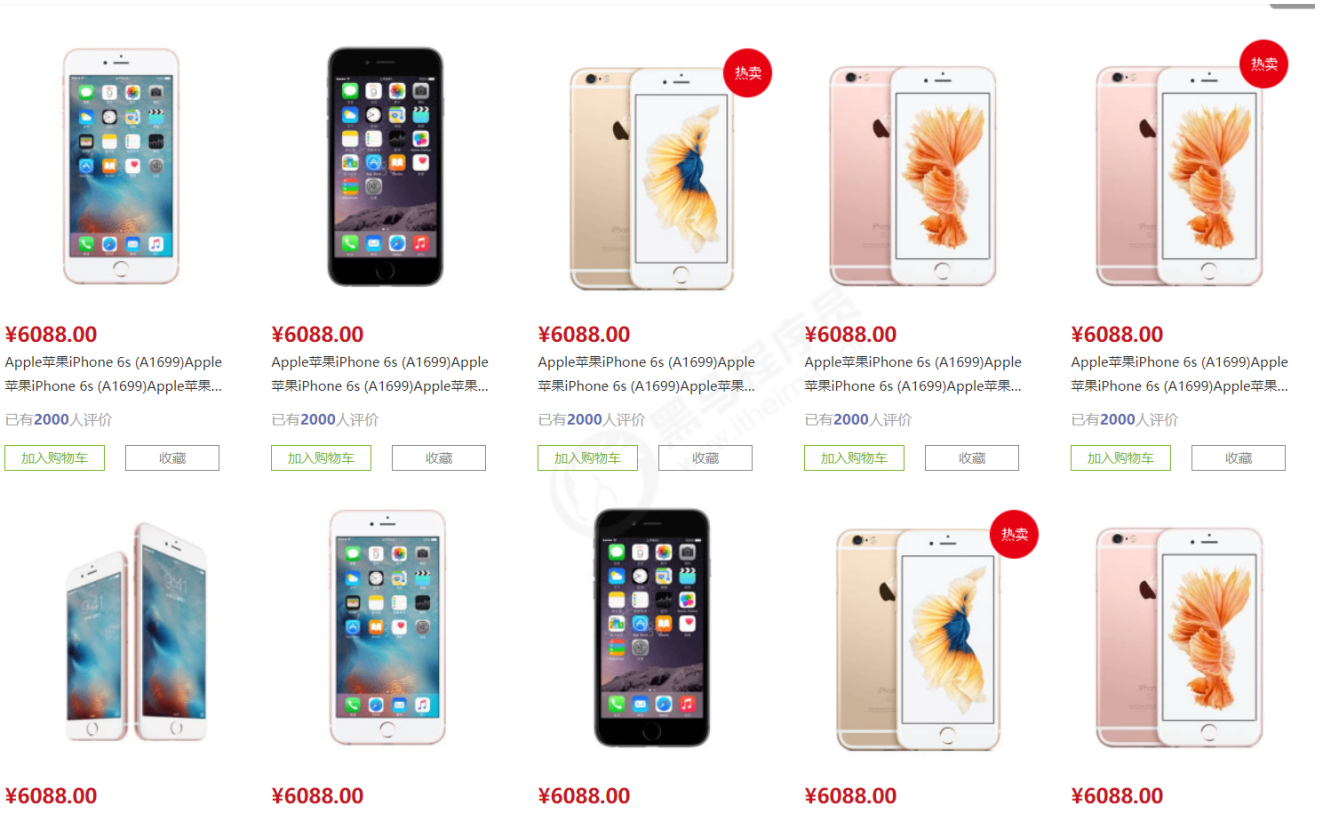
- 完成关键字搜索功能
- 完成商品分类过滤功能
- 完成品牌过滤功能
- 完成规格过滤功能
- 完成价格区间过滤功能

1. 关键字搜索

1.1 需求分析

在首页或搜索页输入关键字，点击搜索

显示列表搜索结果：



1.2 实现思路

（1）后端使用匹配查询和布尔查询

（2）前端使用thymeleaf模板渲染

（3）前端向后端传递map（因为提交的不仅仅是关键字，还有品牌、规格、分类等信息）

（4）后端向前端返回map（因为返回的不仅仅是列表，还有商品分类、品牌和规格列表等数据）

1.3 代码实现

1.3.1 集成elasticsearch高级客户端

（1）qingcheng_service_goods工程pom.xml新增依赖

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>6.5.3</version>
</dependency>
```

（2）qingcheng_service_goods工程新增工厂类

```
public class RestClientFactory {

    public static RestHighLevelClient getRestHighLevelClient(String
hostname,int port){
        HttpHost http=new HttpHost(hostname,port,"http");
        RestClientBuilder builder= RestClient.builder(http);//rest构建器
        return new RestHighLevelClient(builder);//高级客户端对象 （连接）

    }
}
```

（3）qingcheng_service_goods工程applicationContext-service.xml新增配置

```
    <!--es client-->
    <bean id="restHighLevelClient"
class="com.qingcheng.service.impl.RestClientFactory" factory-
method="getRestHighLevelClient">
        <constructor-arg index="0" value="127.0.0.1"></constructor-arg>
        <constructor-arg index="1" value="9200"></constructor-arg>
    </bean>
```

1.3.2 关键字搜索逻辑

（1）qingcheng_interface工程新增接口 SkuSearchService 服务接口新增方法定义

```
public interface SkuSearchService {

    public Map search(Map<String,String> searchMap);

}
```

（2）qingcheng_service_goods工程新增服务实现类SkuSearchServiceImpl 实现此方法，实现关键字查询

```
@Service
public class SkuSearchServiceImpl implements SkuSearchService {

    @Autowired
    private RestHighLevelClient restHighLevelClient;

    public Map search(Map<String,String> searchMap) {

        //1.封装查询请求
        SearchRequest searchRequest=new SearchRequest("sku");
        searchRequest.types("doc"); //设置查询的类型

        SearchSourceBuilder searchSourceBuilder=new
SearchSourceBuilder();

        BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();//
布尔查询构建器

        //1.1 关键字搜索
        MatchQueryBuilder matchQueryBuilder=
QueryBuilders.matchQuery("name",searchMap.get("keywords"));
        boolQueryBuilder.must(matchQueryBuilder);

        searchSourceBuilder.query(boolQueryBuilder);
        searchRequest.source(searchSourceBuilder);


        //2.封装查询结果
        Map resultMap=new HashMap();
        try {
            SearchResponse searchResponse  =
restHighLevelClient.search(searchRequest, RequestOptions.DEFAULT);
            SearchHits searchHits = searchResponse.getHits();
            long totalHits = searchHits.getTotalHits();
            System.out.println("记录数: "+totalHits);
            SearchHit[] hits = searchHits.getHits();

            //2.1 商品列表


            List<Map<String,Object>> resultList=new ArrayList<Map<String,
```

```
Object>>();
            for(SearchHit hit:hits){
                Map<String, Object> skuMap = hit.getSourceAsMap();
                resultList.add(skuMap);
            }
            resultMap.put("rows",resultList);

        } catch (IOException e) {
            e.printStackTrace();
        }
        return resultMap;
    }

}
```

（3）qingcheng_web_portal工程新增类

```
@Controller
public class SearchController {

    @Reference
    private SkuSearchService skuSearchService;

    @GetMapping("/search")
    public String search(Model model, @RequestParam Map<String, String>
searchMap) throws Exception {
        //字符集处理
        searchMap = WebUtil.convertCharsetToUTF8(searchMap);
        //远程调用接口
        Map result = skuSearchService.search(searchMap);
        model.addAttribute("result", result);
        return "search";
    }
}
```

WebUtil类是工具类，用于字符集转码，资源中提供

1.3.3 模板构建

（1）qingcheng_web_portal工程新增模板，将资源\静态原型\网站前台\search.htm拷贝到WEB-INF下，并进行以下修改

```
<html xmlns:th="http://www.thymeleaf.org">
```

遍历查询结果

```
<ul class="yui3-g">
    <li class="yui3-u-1-5" th:each="sku:${result.rows}">
        <div class="list-wrap">
            <div class="p-img">
                <a href="item.html" target="_blank"><img
th:src="${sku.image}" /></a>
            </div>
            <div class="price">
                <strong>
                    <em>¥</em>
                    <i
th:text="${#numbers.formatDecimal(sku.price/100.0,0,2)}"></i>
                </strong>
            </div>
            <div class="attr">
                <em th:text="${sku.name}"></em>
            </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">收藏</a>
            </div>
        </div>
    </li>
</ul>
```

测试 浏览器输入 http://localhost:9102/search.do?keywords=手机

（2）修改index.html的关键字搜索表单

```
<form action="/search.do" class="sui-form form-inline">
    <!--searchAutoComplete-->
    <div class="input-append">
        <input type="text" name="keywords" id="autocomplete"
class="input-error input-xxlarge" />
        <button class="sui-btn btn-xlarge btn-danger" >搜索</button>
    </div>
</form>
```

修改search.html的关键字搜索表单，内容同上

2. 商品分类过滤

2.1 需求分析

以关键字作为查询条件，查询结果中包含的商品分类，在页面中显示出来

点击商品分类，按商品分类对结果进行过滤查询，并且在查询条件列表中添加已经选择的商品分类标签，隐藏搜索面板中的商品分类一行。

点击条件标签的“×”，取消该过滤条件

2.2 实现思路

（1）商品分类列表的显示使用聚合查询

（2）使用过滤查询

2.3 代码实现

2.3.1 商品分类列表

（1）修改SkuSearchServiceImpl的search方法，在第一段代码（封装查询请求）的末尾处添加以下代码：

```
//聚合查询（商品分类）
TermsAggregationBuilder termsAggregationBuilder =
AggregationBuilders.terms("sku_category").field("categoryName");
searchSourceBuilder.aggregation(termsAggregationBuilder);
```

在第二段代码中添加以下代码：

```
//2.2 商品分类列表
Aggregations aggregations = searchResponse.getAggregations();
Map<String, Aggregation> aggregationMap = aggregations.getAsMap();
Terms terms = (Terms) aggregationMap.get("sku_category");

List<? extends Terms.Bucket> buckets =  terms.getBuckets();
List<String> categoryList=new ArrayList();
for( Terms.Bucket bucket:buckets ){
    categoryList.add(bucket.getKeyAsString());
}
resultMap.put("categoryList",categoryList);
```

（2）修改qingcheng_web_portal的search.html 商品分类列表

```
<div class="type-wrap">
    <div class="fl key">商品分类</div>
    <div class="fl value" th:each="category:${result.categoryList}">
        <span>
              <a th:text="${category}" ></a>
         </span>
    </div>
    <div class="fl ext"></div>
</div>
```

2.3.2 分类过滤查询

（1）修改SkuSearchServiceImpl类的search方法，在第一段代码（封装查询请求）中添加如下代码：

```
//1.2 商品分类过滤
if(searchMap.get("category")!=null){
    TermQueryBuilder termQueryBuilder =
QueryBuilders.termQuery("categoryName", searchMap.get("category"));
    boolQueryBuilder.filter(termQueryBuilder);
}
```

（2）修改SearchController的search方法，添加以下代码

```
//url处理
StringBuffer url=new StringBuffer("/search.do?");
for(String key:searchMap.keySet()){
    url.append( "&"+key+"="+ searchMap.get(key)  );
}
model.addAttribute("url",url);
```

（3）修改changgou_web_search的search.html 商品分类列表

```
<a th:href="${url+'&category='+category}" th:text="${category}"></a>
```

2.3.3 取消分类过滤

（1）修改SearchController的search方法，添加以下代码

```
model.addAttribute("searchMap",searchMap);
```

（2）修改search.html的条件列表部分

```
<div class="type-wrap"
th:if="!${#maps.containsKey(searchMap,'category')}">
    <div class="fl key">商品分类</div>
    <div class="fl value" th:each="category:${result.categoryList}">
        <span>
            <a th:href="${url+'&category='+category}"
th:text="${category}"></a>
        </span>
    </div>
    <div class="fl ext"></div>
</div>
```

为商品分类的div添加条件，判断当查询条件中不存在category时显示商品分类

```
<div class="type-wrap" th:if="!${#maps.containsKey(searchMap,
'category')}">
```

3. 品牌过滤

3.1 需求分析

根据关键字搜索得到的商品分类列表，按照第一个分类查询该分类下的品牌列表

如果用户选择了商品分类，按照选择的商品分类查询该分类下的品牌，并添加条件标签，隐藏搜索面板中的品牌一行。

点击品牌，按品牌筛选查询结果。

点击条件标签的“×”，取消该过滤条件

3.2 实现思路

（1）数据访问层添加方法，根据商品分类名称得到品牌列表

（2）在搜索商品的方法中，添加按品牌过滤的逻辑。

3.3 代码实现

3.3.1 品牌列表

（1）BrandMapper接口新增方法定义

```
    /**
     * 根据分类名称查询品牌列表
     * @param categoryName
     * @return
     */
    @Select("SELECT name,image FROM tb_brand WHERE id  IN (SELECT
brand_id FROM tb_category_brand WHERE  category_id IN (SELECT id FROM
tb_category WHERE NAME=#{name}) )order by seq")
    public List<Map> findListByCategoryName(@Param("name") String
categoryName);
```

（2）修改 SkuSearchServiceImpl ，注入BrandMapper

```
    @Autowired
    private BrandMapper brandMapper;
```

修改search方法，新增代码

```
//2.3 品牌列表
String categoryName="";//商品分类名称
if(searchMap.get("category")==null){ // 如果没有分类条件
    if(categoryList.size()>0){
        categoryName=categoryList.get(0);//提取分类列表的第一个分类
    }
}else{
    categoryName=searchMap.get("category");//取出参数中的分类
}
List<Map> brandList = brandMapper.findListByCategoryName(categoryName);//
查询品牌列表
resultMap.put("brandList",brandList);
```

（3）修改search.html ，展现品牌列表

```
<div class="type-wrap logo">
    <div class="fl key brand">品牌</div>
    <div class="value logos">
        <ul class="logo-list">
            <li th:each="brand:${result.brandList}">
                <a>
                    <img th:if="${brand.image}!=''"
th:src="${brand.image}" />
                    <span th:if="${brand.image}==''"
th:text="${brand.name}"></span>
                </a>
            </li>
        </ul>
    </div>
    <div class="ext">
    </div>
</div>
```

3.3.2 品牌过滤查询

（1）修改SkuSearchServiceImpl类的search方法，添加以下代码

```
//1.3 品牌过滤
if(searchMap.get("brand")!=null){
    TermQueryBuilder termQueryBuilder =
QueryBuilders.termQuery("brandName", searchMap.get("brand"));
    boolQueryBuilder.filter(termQueryBuilder);
}
```

（2）修改模板

```
<a th:href="${url+'&brand='+brand.name}">
    <img th:if="${brand.image}!=''" th:src="${brand.image}" />
    <span th:if="${brand.image}==''" th:text="${brand.name}"></span>
</a>
```

3.3.3 取消品牌过滤

修改search.html的条件列表部分

```
<!--品牌-->
<li class="with-x" th:if="${#maps.containsKey(searchMap, 'brand')}">
    <span th:text="|品牌：${searchMap.brand}|"></span>
    <i><a th:href="${#strings.replace(url,'&brand='+searchMap.brand
,'')}">×</a></i>
</li>
```

在品牌列表的div上添加条件

```
<div class="type-wrap logo" th:if="${!#maps.containsKey(searchMap,
'brand') }">
```

代码优化：修改SkuSearchServiceImpl类的search方法，当查询条件中包含品牌信息，则不查询品牌列表

```
//2.3 品牌列表
if(searchMap.get("brand")==null) {
    //...品牌列表查询代码.....
}
```

4. 规格过滤

4.1 需求分析

根据关键字搜索得到的商品分类列表，按照第一个分类查询该分类下的规格和规格选项列表

如果用户选择了商品分类，按照选择的商品分类查询该分类下的规格和规格选项列表

点击某规格下的规格选项，按照该规格和规格选项对结果进行过滤。

点击条件标签的“×”，取消该过滤条件

4.2 实现思路

（1）数据访问层添加方法，根据商品分类名称得到规格列表

（2）在搜索商品的方法中，添加按规格过滤的逻辑。

4.3 代码实现

4.3.1 规格列表

（1）SpecMapper新增方法定义

```
 @Select("SELECT name,options FROM tb_spec WHERE template_id IN ( SELECT
template_id FROM tb_category WHERE NAME=#{categoryName}) order by seq")
    public List<Map> findListByCategoryName(@Param("categoryName") String
categoryName);
```

（2）修改SkuSearchServiceImpl ， 引入specMapper

```
@Autowired
private SpecMapper specMapper;
```

修改search方法，新增代码

```
//2.4 规格列表
List<Map> specList = specMapper.findListByCategoryName(categoryName);//规
格列表
for(Map spec:specList){
    String[] options = ((String) spec.get("options")).split(",");//规格选
项列表
    spec.put("options",options);
}
resultMap.put("specList",specList);
```

（3）修改search.html

```
<div class="type-wrap" th:each="spec:${result.specList}">
    <div class="fl key" th:text="${spec.name}"></div>
    <div class="fl value">
        <ul class="type-list">
            <li th:each="option:${spec.options}">
                <a th:text="${option}"
th:href="|${url}&spec.${spec.name}=${option}|"></a>
            </li>
        </ul>
    </div>
    <div class="fl ext"></div>
</div>
```

4.3.2 规格过滤查询

对于对象类型的查询语法：

前后端约定：所有spec.开头的参数都是规格

修改SkuSearchServiceImpl类的search方法，添加以下代码

```
//1.4 规格过滤
for(String key: searchMap.keySet() ){
    if( key.startsWith("spec.") ){//如果是规格参数
        TermQueryBuilder termQueryBuilder =
QueryBuilders.termQuery(key+".keyword", searchMap.get(key));
        boolQueryBuilder.filter(termQueryBuilder);
    }
}
```

修改模板

```
<a th:text="${option}" th:href="|${url}&spec.${spec.name}=${option}|">
</a>
```

4.3.3 取消规格过滤

修改search.html 添加条件标签

```
<!--规格-->
<li class="with-x" th:each="item:${searchMap}"
th:if="${#strings.startsWith(item.key,'spec.')}" >
    <span th:text="|${#strings.substring(item.key,5)}:${item.value}|">
</span>
    <i><a th:href="${#strings.replace(url,'&'+item.key +'='+item.value
,'')}">×</a></i>
</li>
```

在规格列表的div上添加条件

```
<div th:if="!${#maps.containsKey(searchMap, 'spec.'+spec.name)}">
```

5. 价格过滤

5.1 需求分析

点击价格区间表现，按价格区间搜索

点击条件标签的“×”，取消该过滤条件

5.2 实现思路

（1）价格区间可以在模板中写成固定的。

（2）使用范围查询。

需求：查询价格大于等于10000并且小于等于20000的记录

范围查询的语法如下：

```
#范围查询
GET sku/_search
{
  "query": {
     "range": {
            "price": {
              "gte": 10000,
              "lte": 20000
            }
          }
     }
}
```

rest高级客户端：

```
RangeQueryBuilder rangeQueryBuilder =
QueryBuilders.rangeQuery("price").gte(10000).lte(20000);
```

比较神奇的是，我们可以传递字符串

```
RangeQueryBuilder rangeQueryBuilder =
QueryBuilders.rangeQuery("price").gte("10000").lte("20000");
```

也是可以查询到结果，它会自动转换类型。

5.3 代码实现

5.3.1 价格条件列表

```
<div class="fl key">价格</div>
<div class="fl value">
    <ul class="type-list">
        <li>
            <a th:href="${url+'&price=0-500'}">0-500元</a>
        </li>
        <li>
            <a th:href="${url+'&price=500-1000'}">500-1000元</a>
        </li>
        <li>
            <a th:href="${url+'&price=1000-1500'}">1000-1500元</a>
        </li>
        <li>
            <a th:href="${url+'&price=1500-2000'}">1500-2000元</a>
        </li>
        <li>
            <a th:href="${url+'&price=2000-3000'}">2000-3000元 </a>
        </li>
        <li>
            <a th:href="${url+'&price=3000-*'}">3000元以上</a>
        </li>
    </ul>
</div>
```

5.3.2 价格过滤

修改SkuSearchServiceImpl类的buildBasicQuery方法，添加代码

```
//1.5 价格过滤
if(searchMap.get("price")!=null ){
    String[] price = searchMap.get("price").split("-");
    if(!price[0].equals("0")){ //最低价格不等于0
        RangeQueryBuilder rangeQueryBuilder =
QueryBuilders.rangeQuery("price").gte(price[0] + "00");
        boolQueryBuilder.filter(rangeQueryBuilder);
    }
    if(!price[1].equals("*")){ //如果价格由上限
        RangeQueryBuilder rangeQueryBuilder =
QueryBuilders.rangeQuery("price").lte(price[1] + "00");
        boolQueryBuilder.filter(rangeQueryBuilder);
    }
}
```

5.3.3 取消价格过滤

修改search.html 添加条件标签

```
<!--价格条件-->
<li class="with-x" th:if="${#maps.containsKey(searchMap,'price')}">
    <span th:text="${'价格:'+searchMap.price}"></span>
    <i>
        <a
th:href="${#strings.replace(url,'&price='+searchMap.price,'')}">×</a>
    </i>
</li>
```

在搜索面板的价格一行添加条件

```
<div class="type-wrap" th:if="!${#maps.containsKey(searchMap,'price')}">
```

6. 品牌与规格列表缓存（作业）

6.1 需求分析

现在我们每次查询时都需要根据商品分类名称读取品牌和规格列表，这样对于数据库造成比较大的访问压力，所以我们需要将品牌和规格列表放入缓存。

6.2 实现思路

方式一：定时预热

写一个定时任务，每天执行一次，查询所有的商品分类的品牌和规格列表，放入缓存

以hash形式存储，以分类名称作为key，以品牌列表和规格列表作为值。

启动时检测缓存中是否存在数据，如果不存在数据则立即执行缓存预热。

方式二：随机过期

获取品牌和规格列表查询缓存中是否存在数据，如果缓存中有数据则返回缓存中的数据，如果没有则查询数据库并放入缓存并设置过期时间，为了避免缓存雪崩，我们将过期时间设置为一定范围内的随机数。