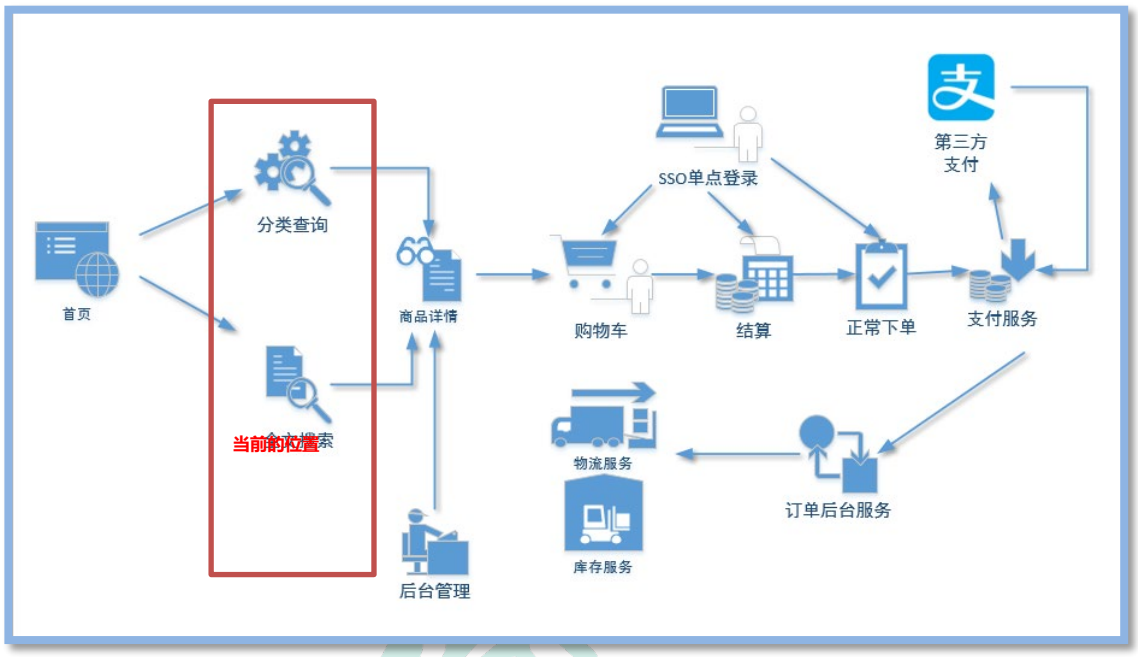
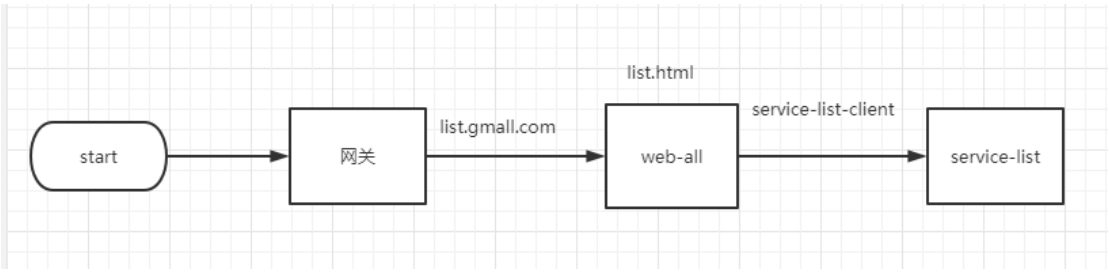
尚品汇商城



一、利用 es 开发电商的搜索列表功能

商品检索流程：



1.1 搜索结果预期数据展示

利用 dsl 语句查询 es 数据

需求分析：

问题：用户进入网站后，可能会根据哪些条件进行查询？

分类 id

平台属性

品牌

商品名称

演示:

情况一：match 匹配查询 title“小米手机”

需求：只能查询出小米系列的

改进：通过 operator 添加 and 条件

情况二：查询分类使用 filter 中 term 和 terms 演示

情况三：添加品牌条件，在原来的 term 后面，直接追加即可

在没有 filter 的情况，不能直接追加，需要于 must 平齐添加 filter 后，在添加 term 过滤条件。

情况四： 查询，分页，高亮处在同一个阶层。

情况五： 高亮时，必须在查询关键字的基础上

```
GET /goods/_search
{
  "query": {
    "bool": {
      "filter": [
        {
```

```
      "term": {
        "category3Id": "61"
      }
    },
    {
      "term": {
        "tmId": "3"
      }
    },
    {
      "bool": {
        "must": [
          {
            "nested": {
              "path": "attrs",
              "query": {
                "bool": {
                  "must": [
                    {
                      "term": {
                        "attrs.attrValue": {
                          "value": "256G"
                        }
                      }
                    }
                  ]
                }
              }
            }
          }
```

```
          }
        ]
      }
    }
  ],
  "must": [
    {
      "match": {
        "title": "荣耀手机"
      }
    }
  ]
 }
},
"from": 0,
"size": 20,
"sort": [
 {
  "hotScore": {
   "order": "desc"
  }
 }
],
"highlight": {
 "fields": {
  "title": {}
 },
 "post_tags": [
  "</span>"
```

```
    ],
    "pre_tags": [
      "<span style=color:red>"
    ]
  },
  "aggs": {
    "tmIdAgg": {
      "terms": {
        "field": "tmId"
      },
      "aggs": {
        "tmNameAgg": {
          "terms": {
            "field": "tmName",
            "size": 10
          }
        },
        "tmLogoUrlAgg": {
          "terms": {
            "field": "tmLogoUrl",
            "size": 10
          }
        }
      }
    },
    "attrAgg": {
      "nested": {
        "path": "attrs"
      },
```

```
    "aggs": {
      "attrIdAgg": {
        "terms": {
          "field": "attrs.attrId",
          "size": 10
        },
        "aggs": {
          "attrNameAgg": {
            "terms": {
              "field": "attrs.attrName",
              "size": 10
            }
          },
          "attrValueAgg": {
            "terms": {
              "field": "attrs.attrValue",
              "size": 10
            }
          }
        }
      }
    }
  }
}
```

1.2 封装搜索相关实体对象

搜索参数实体：SearchParam

```
package com.atguigu.gmall.model.list;

/**
 * 商品搜索参数
 * 参数说明：
 *      1，商标品牌：trademark=2:华为
 *              2：为品牌id，搜索字段
 *              华为：品牌名称，页面回显属性
 *      2，平台属性：props=23:4G:运行内存
 *              23：平台属性id，搜索字段
 *              运行内存：平台属性名称，页面回显属性
 *              4G：平台属性值，搜索字段与页面回显属性
 * </p>
 *
 */

@Data
public class SearchParam {

    //  ?category3Id=61&trademark=2: 华 为 &props=23:4G: 运 行 内 存
&order=1:desc
    //category3Id=61
    private Long category1Id;;//三级分类id
    private Long category2Id;
    private Long category3Id;

    //trademark=2:华为
    private String trademark;//品牌id

    private String keyword;//检索的关键字

  // order=1:asc  排序规则   0:asc
    private String order = "";// 1：综合排序/热点  2：价格

  //props=23:4G:运行内存
  private String[] props;//页面提交的数组

  private Integer pageNo = 1;//分页信息
  private Integer pageSize = 12;
}
```

搜索结果集实体：SearchResponseVo

```java
package com.atguigu.gmall.model.list;


@Data
public class SearchResponseVo implements Serializable {

    //品牌 此时 vo 对象中的 id 字段保留（不用写） name 就是“品牌” value: [{id:100,name:华为,logo:xxx},{id:101,name:小米,log:yyy}]
  private List<SearchResponseTmVo> trademarkList;
    //所有商品的顶头显示的筛选属性
  private List<SearchResponseAttrVo> attrsList = new ArrayList<>();

    //检索出来的商品信息
  private List<Goods> goodsList = new ArrayList<>();

  private Long total;//总记录数
  private Integer pageSize;//每页显示的内容
  private Integer pageNo;//当前页面
  private Long totalPages;

}
```

结果集品牌实体：SearchResponseTmVo

```java
package com.atguigu.gmall.model.list;


@Data
public class SearchResponseTmVo implements Serializable {

    //当前属性值的所有值
  private Long tmId;
    //属性名称
  private String tmName;//网络制式，分类
}
```

结果集平台属性实体：SearchResponseAttrVo

```java
package com.atguigu.gmall.model.list;


@Data
public class SearchResponseAttrVo implements Serializable {

   private Long attrId;//1
    //当前属性值的所有值
  private List<String> attrValueList = new ArrayList<>();
```

```
    //属性名称
  private String attrName;//网络制式，分类
}
```

1.3 搜索接口封装

SearchService 接口

```
/**
 * 搜索列表
 * @param searchParam
 * @return
 * @throws IOException
 */
SearchResponseVo search(SearchParam searchParam) throws IOException;
```

1.4 接口实现类

api 参考文档:

https://www.elastic.co/guide/en/elasticsearch/client/java-rest/7.8/index.html

https://www.elastic.co/guide/en/elasticsearch/client/java-rest/7.8/index.html

```
@Autowired
private RestHighLevelClient restHighLevelClient;

@Override
public SearchResponseVo search(SearchParam searchParam) throws IOException {
    // 构建 dsl 语句
    SearchRequest searchRequest = this.buildQueryDsl(searchParam);
    SearchResponse response = this.restHighLevelClient.search(searchRequest,
RequestOptions.DEFAULT);
    System.out.println(response);

    SearchResponseVo responseVO = this.parseSearchResult(response);
    responseVO.setPageSize(searchParam.getPageSize());
    responseVO.setPageNo(searchParam.getPageNo());
    long totalPages = (responseVO.getTotal()+searchParam.getPageSize()-
1)/searchParam.getPageSize();
    responseVO.setTotalPages(totalPages);
    return responseVO;
}
```

```
// 制作 dsl 语句
private SearchRequest buildQueryDsl(SearchParam searchParam) {
    // 构建查询器
    SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
    // 构建 boolQueryBuilder
    BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
    // 判断查询条件是否为空 关键字
    if (!StringUtils.isEmpty(searchParam.getKeyword())){
        // 小米手机  小米 and 手机
        //         MatchQueryBuilder          matchQueryBuilder          =          new
MatchQueryBuilder("title",searchParam.getKeyword()).operator(Operator.AND);
        MatchQueryBuilder       title       =       QueryBuilders.matchQuery("title",
searchParam.getKeyword()).operator(Operator.AND);
        boolQueryBuilder.must(title);
    }
    // 构建品牌查询
    String trademark = searchParam.getTrademark();
    if (!StringUtils.isEmpty(trademark)){
        // trademark=2:华为
        String[] split = StringUtils.split(trademark, ":");
        if (split != null && split.length == 2) {
            // 根据品牌 Id 过滤
            boolQueryBuilder.filter(QueryBuilders.termQuery("tmId", split[0]));
        }
    }

    // 构建分类过滤 用户在点击的时候，只能点击一个值，所以此处使用 term
    if(null!=searchParam.getCategory1Id()){

boolQueryBuilder.filter(QueryBuilders.termQuery("category1Id",searchParam.getCategor
y1Id()));
    }
    // 构建分类过滤
    if(null!=searchParam.getCategory2Id()){

boolQueryBuilder.filter(QueryBuilders.termQuery("category2Id",searchParam.getCategor
y2Id()));
    }
    // 构建分类过滤
    if(null!=searchParam.getCategory3Id()){

boolQueryBuilder.filter(QueryBuilders.termQuery("category3Id",searchParam.getCategor
y3Id()));
    }

    // 构建平台属性查询
    // 23:4G:运行内存
    String[] props = searchParam.getProps();
    if (props!=null && props.length>0){
        // 循环遍历
        for (String prop : props) {
            // 23:4G:运行内存
            String[] split = StringUtils.split(prop, ":");
```

```
            if (split!=null && split.length==3){
                // 构建嵌套查询
                BoolQueryBuilder boolQuery = QueryBuilders.boolQuery();
                // 嵌套查询子查询
                BoolQueryBuilder subBoolQuery = QueryBuilders.boolQuery();
                // 构建子查==询中的过滤条件
subBoolQuery.must(QueryBuilders.termQuery("attrs.attrId",split[0]));

subBoolQuery.must(QueryBuilders.termQuery("attrs.attrValue",split[1]));
                // ScoreMode.None ?
                boolQuery.must(QueryBuilders.nestedQuery("attrs",subBoolQuery,
ScoreMode.None));
                // 添加到整个过滤对象中
                boolQueryBuilder.filter(boolQuery);
            }
        }
    }
    // 执行查询方法
    searchSourceBuilder.query(boolQueryBuilder);
    // 构建分页
    int from = (searchParam.getPageNo()-1)*searchParam.getPageSize();
    searchSourceBuilder.from(from);
    searchSourceBuilder.size(searchParam.getPageSize());

    // 排序
    String order = searchParam.getOrder();
    if (!StringUtils.isEmpty(order)){
        // 判断排序规则
        String[] split = StringUtils.split(order, ":");
        if (split!=null && split.length==2){
            // 排序的字段
            String field = null;
            // 数组中的第一个参数
            switch (split[0]){
                case "1":
                    field="hotScore";
                    break;
                case "2":
                    field="price";
                    break;
            }
            searchSourceBuilder.sort(field,"asc".equals(split[1])?
SortOrder.ASC:SortOrder.DESC);
        }else {
            // 没有传值的时候给默认值
            searchSourceBuilder.sort("hotScore",SortOrder.DESC);
        }
    }

    // 构建高亮
        HighlightBuilder highlightBuilder = new HighlightBuilder();
    highlightBuilder.field("title");
    highlightBuilder.postTags("</span>");
```

```
    highlightBuilder.preTags("<span style=color:red>");

    searchSourceBuilder.highlighter(highlightBuilder);

    //  设置品牌聚合
    TermsAggregationBuilder termsAggregationBuilder = AggregationBuilders.terms("tmIdAgg").field("tmId")
        .subAggregation(AggregationBuilders.terms("tmNameAgg").field("tmName"))
        .subAggregation(AggregationBuilders.terms("tmLogoUrlAgg").field("tmLogoUrl"));

    searchSourceBuilder.aggregation(termsAggregationBuilder);

    //  设置平台属性聚合
    searchSourceBuilder.aggregation(AggregationBuilders.nested("attrAgg", "attrs")
                    .subAggregation(AggregationBuilders.terms("attrIdAgg").field("attrs.attrId")
                    .subAggregation(AggregationBuilders.terms("attrNameAgg").field("attrs.attrName"))
                    .subAggregation(AggregationBuilders.terms("attrValueAgg").field("attrs.attrValue"))));


    // 结果集过滤
    searchSourceBuilder.fetchSource(new String[]{"id","defaultImg","title","price"},null);

    SearchRequest searchRequest = new SearchRequest("goods");
    //searchRequest.types("_doc");
    searchRequest.source(searchSourceBuilder);
    System.out.println("dsl:"+searchSourceBuilder.toString());
    return searchRequest;
}
```

```
// 制作返回结果集
private SearchResponseVo parseSearchResult(SearchResponse response) {
    SearchHits hits = response.getHits();
    //声明对象
    SearchResponseVo searchResponseVo = new SearchResponseVo();
    //获取品牌的集合
    Map<String, Aggregation> aggregationMap = response.getAggregations().asMap();
    //ParsedLongTerms ?
    ParsedLongTerms tmIdAgg = (ParsedLongTerms) aggregationMap.get("tmIdAgg");
    List<SearchResponseTmVo> trademarkList = tmIdAgg.getBuckets().stream().map(bucket -> {
        SearchResponseTmVo trademark = new SearchResponseTmVo();
        //获取品牌 Id
         trademark.setTmId((Long.parseLong(((Terms.Bucket) bucket).getKeyAsString())));
        //trademark.setTmId(Long.parseLong(bucket.getKeyAsString()));
        //获取品牌名称
        Map<String, Aggregation> tmIdSubMap = ((Terms.Bucket) bucket).getAggregations().asMap();
```

```
        ParsedStringTerms tmNameAgg = (ParsedStringTerms) tmIdSubMap.get("tmNameAgg");
        String tmName = tmNameAgg.getBuckets().get(0).getKeyAsString();

        trademark.setTmName(tmName);

ParsedStringTerms tmLogoUrlAgg = (ParsedStringTerms) tmIdSubMap.get("tmLogoUrlAgg");
String tmLogoUrl = tmLogoUrlAgg.getBuckets().get(0).getKeyAsString();
trademark.setTmLogoUrl(tmLogoUrl);

        return trademark;
    }).collect(Collectors.toList());
    searchResponseVo.setTrademarkList(trademarkList);

    //赋值商品列表
    SearchHit[] subHits = hits.getHits();
    List<Goods> goodsList = new ArrayList<>();
    if (subHits!=null && subHits.length>0){
        //循环遍历
        for (SearchHit subHit : subHits) {
            // 将 subHit 转换为对象
            Goods goods = JSONObject.parseObject(subHit.getSourceAsString(), Goods.class);

            //获取高亮
            if (subHit.getHighlightFields().get("title")!=null){
                Text title = subHit.getHighlightFields().get("title").getFragments()[0];
                goods.setTitle(title.toString());
            }
            goodsList.add(goods);
        }
    }
    searchResponseVo.setGoodsList(goodsList);

    //获取平台属性数据
    ParsedNested attrAgg = (ParsedNested) aggregationMap.get("attrAgg");
    ParsedLongTerms attrIdAgg = attrAgg.getAggregations().get("attrIdAgg");
    List<? extends Terms.Bucket> buckets = attrIdAgg.getBuckets();
    if (!CollectionUtils.isEmpty(buckets)){
        List<SearchResponseAttrVo> searchResponseAttrVOS = buckets.stream().map(bucket -> {
            //声明平台属性对象
            SearchResponseAttrVo responseAttrVO = new SearchResponseAttrVo();
            //设置平台属性值 Id
            responseAttrVO.setAttrId(((Terms.Bucket) bucket).getKeyAsNumber().longValue());
            ParsedStringTerms attrNameAgg = bucket.getAggregations().get("attrNameAgg");
            List<? extends Terms.Bucket> nameBuckets = attrNameAgg.getBuckets();
            responseAttrVO.setAttrName(nameBuckets.get(0).getKeyAsString());
            //设置规格参数列表
            ParsedStringTerms attrValueAgg = ((Terms.Bucket)
```

```
bucket).getAggregations().get("attrValueAgg");
            List<? extends Terms.Bucket> valueBuckets = attrValueAgg.getBuckets();


            List<String>                                values                                =
valueBuckets.stream().map(Terms.Bucket::getKeyAsString).collect(Collectors.toList())
;
            responseAttrVO.setAttrValueList(values);


            return responseAttrVO;


        }).collect(Collectors.toList());
        searchResponseVo.setAttrsList(searchResponseAttrVOS);
    }
    // 获取总记录数
    searchResponseVo.setTotal(hits.getTotalHits().value);

    return searchResponseVo;
}
```

1.5 控制器 ListApiController

```
/**
 * 搜索商品
 * @param searchParam
 * @return
 * @throws IOException
 */
@PostMapping
public Result list(@RequestBody SearchParam searchParam) throws
IOException {
    SearchResponseVo response = searchService.search(searchParam);
    return Result.ok(response);
}
```

说明：http://localhost:8203/swagger-ui.html 测试接口

1.6 在 service-list 模块中配置 logstash

首先在 service 模块中添加依赖

```
<dependency>
        <groupId>net.logstash.logback</groupId>
```

```
        <artifactId>logstash-logback-encoder</artifactId>
        <version>5.1</version>
    </dependency>
```

其次，将日志配置文件放入到 resources 目录下！

二、 在 service-list-client 模块添加接口

```
package com.atguigu.gmall.list.client;

@FeignClient(value = "service-list", fallback = ListDegradeFeignClient.class)
public interface ListFeignClient {

    /**
     * 搜索商品
     * @param listParam
     * @return
     */
    @PostMapping("/api/list")
    Result list(@RequestBody SearchParam listParam);

    /**
     * 上架商品
   * @param skuId
     * @return
     */
    @GetMapping("/api/list/inner/upperGoods/{skuId}")
    Result upperGoods(@PathVariable("skuId") Long skuId);

    /**
     * 下架商品
   * @param skuId
     * @return
     */
    @GetMapping("/api/list/inner/lowerGoods/{skuId}")
    Result lowerGoods(@PathVariable("skuId") Long skuId);

}
```

```
package com.atguigu.gmall.list.client.impl;


@Component
```

```
public class ListDegradeFeignClient implements ListFeignClient {

    @Override
    public Result list(SearchParam searchParam) {
        return Result.fail();
    }

    @Override
    public Result upperGoods(Long skuId) {
        return null;
    }

    @Override
    public Result lowerGoods(Long skuId) {
        return null;
    }
}
```

三、修改 web-all 模块

3.1 修改 pom.xml 文件

```
<dependencies>
    <dependency>
        <groupId>com.atguigu.gmall</groupId>
        <artifactId>service-item-client</artifactId>
        <version>1.0</version>
    </dependency>
    <dependency>
        <groupId>com.atguigu.gmall</groupId>
        <artifactId>service-product-client</artifactId>
        <version>1.0</version>
    </dependency>
    <dependency>
        <groupId>com.atguigu.gmall</groupId>
        <artifactId>service-list-client</artifactId>
        <version>1.0</version>
    </dependency>
</dependencies>
```

3.2 在 ListController 控制器调用接口

```
package com.atguigu.gmall.all.controller;
/**
 * <p>
 * 产品列表接口
 * </p>
 *
 */
@Controller
public class ListController {

    @Autowired
    private ListFeignClient listFeignClient;

    /**
     * 列表搜索
   * @param searchParam
     * @return
     */
    @GetMapping("list.html")
    public String search(SearchParam searchParam, Model model) {
        Result<Map> result = listFeignClient.list(searchParam);
        model.addAllAttributes(result.getData());

        return "list/index";
    }

}
```

3.3 配置网关

```
#==================web 前端==========================
- id: web-list
  uri: lb://web-all
  predicates:
  - Host=list.gmall.com
```

3.4 页面渲染

1，列表显示

```
<li class="yui3-u-1-5" th:each="goods: ${goodsList}">
    <div class="list-wrap">
```

```
        <div class="p-img">
            <a
th:href="@{http://item.gmall.com/{id}.html(id=${goods.id})}"
target="_blank"><img th:src="${goods.defaultImg}"/></a>
        </div>
        <div class="price">
            <strong>
                <em>¥</em>
                <i th:text="${goods.price}">6088.00</i>
            </strong>
        </div>
        <div class="attr">
            <a
th:href="@{http://item.gmall.com/{id}.html(id=${goods.id})}"
target="_blank" th:utext="${goods.title}">Apple 苹果 iPhone 6s
(A1699)Apple 苹果 iPhone 6s (A1699)Apple 苹果 iPhone 6s (A1699)Apple 苹
果 iPhone 6s (A1699)</a>
        </div>
        <div class="commit">
            <i class="command">已有<span>2000</span>人评价</i>
        </div>
        <div class="operate">
            <a href="javascript:void(0);" class="sui-btn btn-
bordered btn-danger">自营</a>
            <a href="javascript:void(0);" class="sui-btn btn-
bordered">收藏</a>
        </div>
    </div>
</li>
```

3.5 搜索条件处理

3.5.1 根据搜索对象 SearchParam 拼接 url

ListController

```
@Autowired
private ListFeignClient listFeignClient;
@GetMapping("list.html")
public String search(SearchParam searchParam, Model model){
    Result<Map> list = listFeignClient.list(searchParam);
    model.addAllAttributes(list.getData());

    // 记录拼接url;
   String urlParam = makeUrlParam(searchParam);
    model.addAttribute("searchParam",searchParam);
    model.addAttribute("urlParam",urlParam);
```

```
    return "list/index";
}
```

```
// 制作返回的url
private String makeUrlParam(SearchParam searchParam) {
    StringBuilder urlParam = new StringBuilder();
    // 判断关键字
  if (searchParam.getKeyword()!=null){
        urlParam.append("keyword=").append(searchParam.getKeyword());
    }
    // 判断一级分类
  if (searchParam.getCategory1Id()!=null){

urlParam.append("category1Id=").append(searchParam.getCategory1Id());
    }
    // 判断二级分类
  if (searchParam.getCategory2Id()!=null){

urlParam.append("category2Id=").append(searchParam.getCategory2Id());
    }
    // 判断三级分类
  if (searchParam.getCategory3Id()!=null){

urlParam.append("category3Id=").append(searchParam.getCategory3Id());
    }
    // 处理品牌
  if(searchParam.getTrademark()!=null){
        if (urlParam.length()>0){

urlParam.append("&trademark=").append(searchParam.getTrademark());
        }
    }
    // 判断平台属性值
  if (null != searchParam.getProps()){
        for (String prop : searchParam.getProps()) {
            if (urlParam.length() > 0){
                urlParam.append("&props=").append(prop);
            }
        }
    }
    return "list.html?" + urlParam.toString();
}
```

1. 平台属性处理

```
<div class="type-wrap" th:each="baseAttrInfo:${attrsList}"
th:unless="${#strings.contains(urlParam,
'props='+baseAttrInfo.attrId)}">
    <div class="fl key" th:text="${baseAttrInfo.attrName}">网络制式
</div>
    <div class="fl value">
        <ul class="type-list">
            <li th:each="attrValue:${baseAttrInfo.attrValueList}">
                <a href="" th:text="${attrValue}" >属性值 111</a>
```

```
            </li>
        </ul>
    </div>
    <div class="fl ext"></div>
</div>
```

说明：

1，这样平台属性就拼接到 url 中，并且能保持参数了

2，点击平台属性，改平台属性就不在列表中显示了，控制如下：

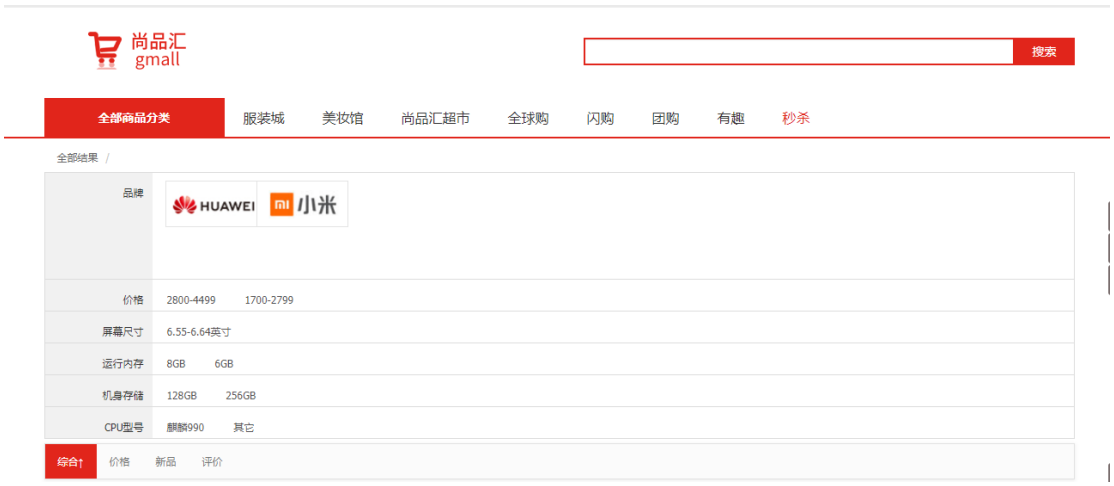
```
th:unless="${#strings.contains(urlParam,
'props='+baseAttrInfo.attrId)}"
```

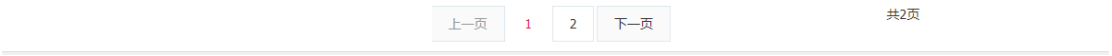
2，品牌显示

```
<div class="type-wrap logo" th:if="${searchParam.trademark ==
null}">
    <div class="fl key brand">品牌</div>
    <div class="value logos">
        <ul class="logo-list">
            <li th:each="trademark:${trademarkList}">
                <a href="" th:text="${trademark.tmName}">属性值</a>
            </li>
        </ul>
    </div>
</div>
```

说明：`th:if="${searchParam.trademark == null}" 控制品牌是否显示`

说明：目前页面已经渲染，但是搜索条件我们怎么处理，搜索条件值如何保持等问题还没解决，如图：





说明：所有的搜索条件都拼接到了一个 url 上面，除分页参数与排序

3、分页处理

```
<div class="sui-pagination pagination-large">
    <ul>
        <li class="prev" th:if="${pageNo != 1}">
            <a th:href="${urlParam}+'&pageNo='+${pageNo - 1}">
上一页</a>
        </li>
        <li class="prev disabled" th:if="${pageNo == 1}">
            <a href="javascript:">上一页</a>
        </li>

        <li th:each="i : ${#numbers.sequence(1,totalPages)}"
th:class="${i == pageNo} ? 'active' : ''">
            <a th:href="${urlParam}+'&pageNo='+${i}"><span
th:text="${i}"></span></a>
        </li>

        <li class="next" th:if="${pageNo < totalPages}">
            <a th:href="${urlParam}+'&pageNo='+${pageNo + 1}">下一页
</a>
        </li>
        <li class="next disabled" th:if="${pageNo == totalPages}">
            <a href="javascript:">下一页</a>
        </li>
    </ul>
    <div><span>共<span th:text="${totalPages }"></span>页
 </span><span></div>
</div>
```

3.5.2 面包屑处理

品牌：平台属性

1，品牌与平台属性

```
ListController

/**
 * 处理品牌条件回显
 * @param trademark
```

```
 * @return
 */
private String makeTrademark(String trademark) {
    if (!StringUtils.isEmpty(trademark)) {
        String[] split = StringUtils.split(trademark, ":");
        if (split != null && split.length == 2) {
            return "品牌：" + split[1];
        }
    }
    return "";
}
```

前台页面数据展示：

```
{${#strings.replace(urlParam+'&order='+searchParam.order,'props='+prop.attrId+':'+prop.attrValue+':'+prop.attrName,'')}}">
```

```
/**
 * 处理平台属性条件回显
 * @param props
 * @return
 */
// 处理平台属性
private List<Map<String, String>> makeProps(String[] props) {
    List<Map<String, String>> list = new ArrayList<>();
    // 2:v:n
    if (props!=null && props.length!=0){
        for (String prop : props) {
            String[] split = StringUtils.split(prop, ":");
            if (split!=null && split.length==3){
                // 声明一个map
                HashMap<String, String> map = new HashMap<String,String>();
                    map.put("attrId",split[0]);
                    map.put("attrValue",split[1]);
                    map.put("attrName",split[2]);
                    list.add(map);
            }
        }
    }
    return list;
}
```

```
@GetMapping("list.html")
public String search(SearchParam searchParam, Model model) {
    Result<Map> result = listFeignClient.list(searchParam);
    model.addAllAttributes(result.getData());

    //拼接url
    String urlParam = makeUrlParam(searchParam);
    //处理品牌条件回显
  String                    trademarkParam                    =
this.makeTrademark(searchParam.getTrademark());
    //处理平台属性条件回显
```

```
  List<Map<String, String>> propsParamList = this.makeProps(searchParam.getProps());

    model.addAttribute("searchParam",searchParam);
    model.addAttribute("urlParam",urlParam);
    model.addAttribute("trademarkParam",trademarkParam);
    model.addAttribute("propsParamList",propsParamList);
    return "list/index";
}
```

页面处理

1，关键字

```
<ul class="fl sui-breadcrumb">
    <li>
        <a href="#">全部结果</a>
    </li>
    <li  class="active">
        <span th:text="${searchParam.keyword}"></span>
    </li>
</ul>
```

2，品牌处理

```
<ul class="fl sui-tag">
    <li th:if="${searchParam.trademark != null}" class="with-x">
        <span th:text="${trademarkParam}"></span>
        <a
th:href="@{${#strings.replace(urlParam,'trademark='+searchParam.trad
emark,'')}}">×</a>
    </li>
    </ul>
```

说明：urlParam 里面已经包含品牌参数，该链接必须去除该参数，所以我们可以使用 thymeleaf 字符串替换函数，把品牌参数替换了就可以了，

```
${#strings.replace(urlParam,'trademark='+searchParam.trademark,'')}
```

3，平台属性处理

```
<ul class="fl sui-tag">
        <li th:if="${searchParam.props != null}" th:each="prop :
${propsParamList}" class="with-x">
        <span th:text="${prop.attrName}+' :
'+${prop.attrValue}"></span>
        <a
th:href="@{${#strings.replace(urlParam+'&order='+searchParam.order,'
props='+prop.attrId+':'+prop.attrValue+':'+prop.attrName,'')}}">×</a
>
```

```
    </li>
</ul>
```

说明：与品牌一样，替换掉对应的平台属性值即可

```
${#strings.replace(urlParam,'props='+prop.attrId+':'+prop.attrValue+
':'+prop.attrName,'')}
```

3.5.3 排序处理

排序

```
ListController

/**
 * 处理排序
 * @param order
 * @return
 */
private Map<String, Object> dealOrder(String order) {
    Map<String,Object> orderMap = new HashMap<>();
    if(!StringUtils.isEmpty(order)) {
        String[] split = StringUtils.split(order, ":");
        if (split != null && split.length == 2) {
            // 传递的哪个字段
     orderMap.put("type", split[0]);
            // 升序降序
     orderMap.put("sort", split[1]);
        }
    }else {
        orderMap.put("type", "1");
        orderMap.put("sort", "asc");
    }
    return orderMap;
}
```

```
@GetMapping("list.html")
public String search(SearchParam searchParam, Model model) {
    Result<Map> result = listFeignClient.list(searchParam);
    model.addAllAttributes(result.getData());

    //拼接url
    String urlParam = makeUrlParam(searchParam);
    //处理品牌条件回显
  String                          trademarkParam                          =
this.makeTrademark(searchParam.getTrademark());
    //处理平台属性条件回显
  List<Map<String,          String>>          propsParamList          =
this.makeProps(searchParam.getProps());
    //处理排序
  Map<String,Object>                        orderMap                    =
this.dealOrder(searchParam.getOrder());
```

```
    model.addAttribute("searchParam",searchParam);
    model.addAttribute("urlParam",urlParam);
    model.addAttribute("trademarkParam",trademarkParam);
    model.addAttribute("propsParamList",propsParamList);
    model.addAttribute("orderMap",orderMap);
    return "list/index";
}
```

页面

```
<ul class="sui-nav">
    <li th:class="${orderMap.type == '1' ? 'active': ''}">
        <a th:href="${urlParam}+'&order=1:'+${orderMap.sort ==
'asc' ? 'desc' : 'asc'}">
            综合<span th:if="${orderMap.type == '1'}"
th:text="${orderMap.sort == 'asc' ? '↑' : '↓'}"></span>
        </a>
    </li>
    <li th:class="${orderMap.type == '2' ? 'active': ''}">
        <a th:href="${urlParam}+'&order=2:'+${orderMap.sort ==
'asc' ? 'desc' : 'asc'}">
            价格<span th:if="${orderMap.type == '2'}"
th:text="${orderMap.sort == 'asc' ? '↑' : '↓'}"></span>
        </a>
    </li>
    <li>
        <a href="#">新品</a>
    </li>
    <li>
        <a href="#">评价</a>
    </li>
</ul>
```

说明：

1，排序没有拼接到 urlParam 中，原因：如果拼接会重复出现

2，为了保持排序条件，所以其他所有链接都需加上排序参数

改造其他连接

分页

```
        <a
th:href="${urlParam}+'&pageNo='+${i}+'&order='+${searchParam.order}"
><span th:text="${i}"></span></a>
```

1，平台属性

```
<a
th:href="${urlParam}+'&props='+${baseAttrInfo.attrId}+':'+${attrValu
e}+':'+${baseAttrInfo.attrName}+'&order='+${searchParam.order}"
th:text="${attrValue}" >属性值 111</a>
</li>
```

2，品牌

```
<a
th:href="${urlParam}+'&trademark='+${trademark.tmId}+':'+${trademark
.tmName}+'&order='+${searchParam.order}"
th:text="${trademark.tmName}">属性值 111</a>
</li>
```

3，面包屑

```
<ul class="fl sui-tag">
    <li th:if="${searchParam.trademark != null}" class="with-x">
        <span th:text="${trademarkParam}"></span>
        <a
th:href="@{${#strings.replace(urlParam+'&order='+searchParam.order,'
trademark='+searchParam.trademark,'')}}">×</a>
    </li>
    <li th:if="${searchParam.props != null}" th:each="prop :
${propsParamList}" class="with-x">
        <span th:text="${prop.attrName}+' :
'+${prop.attrValue}"></span>
        <a
th:href="@{${#strings.replace(urlParam+'&order='+searchParam.order,'
props='+prop.attrId+':'+prop.attrValue+':'+prop.attrName,'')}}">×</a
>
    </li>
</ul>
```