**商品列表页制作说明**

本说明同时适用于b2c和b2b2c

# 概念解释

## 搜索选择器

列表页是由 搜索选择器区块 和 商品列表 块来组成的：

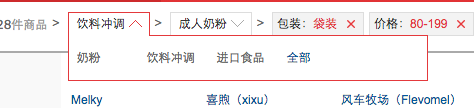


其中搜索选择区 每一项 我们称之为 “搜索选择器”（SearchSelector），主要是分类选择器、品牌选择器、价格选择器、属性选择器、排序选择器:

选择器分为两大部分：已经选择的和未选择的：



已经选择的通过点击可以取消选择，而已经选中的分类是可以再次选择同级分类的：



### 品牌选择器：



是可以显示图片的

### 价格选择器



可以用于过滤某个价格区间的商品

### 属性选择器



在商品类型中定义的各种商品属性（渐进式搜索类型的）会在这里显示

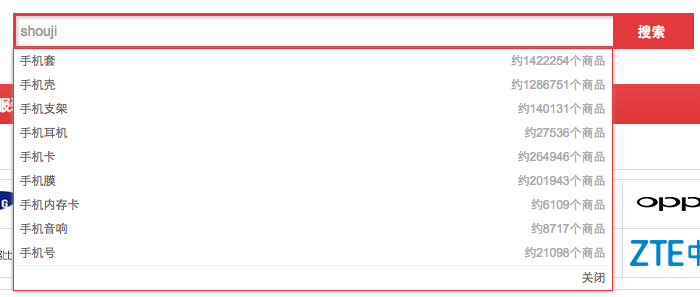
### 排序选择器：



排序选择器分为，选中和未选择状态，选择中状态又分类正序和倒序

### 搜索猜测

效果：



在搜索框中输入部分或全部词，会提示相关的词，同时提示有多少个商品，支持拼音输入。

# 制作说明

## 选择器

### 选择器的Java对象：

com.enation.app.shop.core.plugin.search.SearchSelector

他的通用属性：

name:用于显示选择器的文字

url:用于选择器的点击连接

一般地通过下面的代码来显示一个选择器：

<a href=*"${selector.url!''}"* >${selector.name}</a>

他的一些其它属性：

isSelected：是否选中，对排序选择器有效

value:对排序选择器和已经选中的属性选择器有效

otherOptions:其它的选择项，对选中分类有效

### 标签

首先我们要用到搜索选择器标签，id为 searchSelectorTag

返回是一个Map，这里包含了各种搜索选择器。

举例：

我们先得到搜索选择器

<#-- 创建选择器标签 -->

<#assign searchSelectorTag= newTag("searchSelectorTag") >

<#-- 获取选择器 -->

<#assign goodsSelector = searchSelectorTag() >

然后我们获取各种选择器：

<#-- 各种选中的选择器 -->

<#assign selected\_prop= goodsSelector.selected\_prop >

<#assign selected\_cat= goodsSelector.selected\_cat >

<#assign selected\_brand= goodsSelector.selected\_brand >

<#assign selected\_price= goodsSelector.selected\_price![] >

<#-- 各种未选中的选择器 -->

<#assign propMap= goodsSelector.prop >

<#assign catList= goodsSelector.cat >

<#assign brandList= goodsSelector.brand >

<#assign sortList= goodsSelector.sort![] >

<#assign priceList= goodsSelector.price![] >

从上面的示例可以看出，我是通过goodsSelector.某选择器id 来获取到某个选择器的，具体的id和选择器对照表如下：

|  |  |
| --- | --- |
| id | 选择器 |
| selected\_prop | 已选中的属性选择器 |
| selected\_cat | 已选中的分类选择器 |
| selected\_brand | 已选中的品牌选择器 |
| selected\_price | 已选中的价格选择器 |
| prop | 属性选择器 |
| cat | 分类选择器 |
| brand | 品牌选择器 |
| price | 价格选择器 |

下面我们介绍每种选择器的输出：

### 已选中分类选择器

类型:List，里面的每个元素是一个SelectSelector

特殊属性：otherOptions，他的类型也是一个List，每个元素还是SelectSelector

举例：

<#list selected\_cat as selector >

<span class=*"cat"*>

&gt;<a href=*"${selector.url!''}"* >${selector.name}</a>

<div class=*"other"*>

<#list selector.otherOptions as other >

<a href=*"${other.url!''}"* >${other.name}</a>|

</#list>

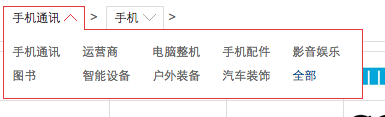
</div>

</span>

</#list>

</li>

通过otherOptions可以获取此分类的同级其它分类，以便形成如下的效果：



### 分类选择器：

通过标准的输出：

<#list catList as selector >

<li><a href=*"${selector.url!''}"* >${selector.name}</a></li>

</#list>

### 品牌选择器

品牌选择器是支持图片输出的，他的图片路径存在SearchSelector的value属性中：

<h1 >品牌</h1>

<#list brandList as selector >

<li><a href=*"${selector.url!''}"* >${selector.name}<@image src="${selector.value!''}" /></a></li>

</#list>

注意已选中品牌value属性是空的

### 属性选择器：

属性选择器是一个map，而不是一个list，map的key是属性的名字，使用示例：

<h1>属性</h1>

<#list propMap?keys as propname>

<li>

<h2>${propname}：</h2>

<ul>

<#list propMap[propname] as selector >

<li><a href=*"${selector.url!''}"* >${selector.name}</a></li>

</#list>

</ul>

</li>

</#list>

效果：



### 已选中属性选择器

已经选中的属性是一个List，输出时可以用name和value来输出选中的属性名称和值：

<h1>已经选择属性：</h1>

<#list selected\_prop as selector >

<li><a href=*"${selector.url!''}"* >${selector.name}:${selector.value}</a></li>

</#list>

效果：



### 价格选择器

已选中和未选中的都通过标准的方式来输出，需要说明的是，价格的范围是根据当前的分类来对应的，如果没有分类的搜索输出默认范围，这种对应关系是在

themes目录中的 price\_filter.xml 来定义的：

<cat id=*"0"*>

<price max=*"5"*>5元以下</price>

<price min=*"5"* max=*"10"*>5-10元</price>

<price min=*"10"* max=*"20"*>10-20元</price>

<price min=*"20"* max=*"50"*>20-50元</price>

<price min=*"50"* max=*"100"*>50-100元</price>

<price min=*"100"* max=*"200"*>100-200元</price>

<price min=*"200"* max=*"500"*>200-500元</price>

<price min=*"500"* max=*"1000"*>500-1000元</price>

<price min=*"1000"*>1000元以上</price>

</cat>

<cat id=*"9"*>

<price max=*"5"*>5元以下</price>

<price min=*"5"* max=*"10"*>5-10元</price>

<price min=*"10"* max=*"20"*>10-20元</price>

<price min=*"20"* max=*"50"*>20-50元</price>

<price min=*"50"* max=*"100"*>50-100元</price>

<price min=*"100"* max=*"200"*>100-200元</price>

<price min=*"200"* max=*"500"*>200-500元</price>

<price min=*"500"* max=*"1000"*>500-1000元</price>

<price min=*"1000"*>1000元以上</price>

</cat>

其中catid 为0的是默认的价格范围。

### 排序选择器

排序选择器有两个特殊的属性：isSelected表示是否选中的，value代表排序方式，是desc还是asc,举例：

<h1>排序：</h1>

<#list sortList as selector >

<li><a href=*"${selector.url!''}"* <#if selector.isSelected>class="selected ${selector.value}"</#if> >${selector.name}</a></li>

</#list>

## 搜索猜测

首先要了解的是搜索猜测必须依托于商品索引组件（lucene），要安装此组件，并在系统设置中开启lucene全文索引组件：



并重新生成一次索引才可以使用。

### 自动猜测jquery插件

用法示例：

**var** url\_ = "${ctx}/api/shop/goods-words.do";

$("#namekeyword").bigAutocomplete(

{url:url\_,

callback:**function**(data){

location.href="${ctx}/search.html?keyword="+data.title;

}});

其中$("#namekeyword")是搜索框，这样搜索框就实现了猜测的效果。

我们在制作模板时要注意通过标签获取一下用户是否开启了lucene索引，完整的代码如下：

<#assign luceneCheckTag = newTag("luceneCheckTag") />

<#assign lucene =luceneCheckTag() />

<#**if** lucene=='ON' >

**var** url\_ = "${ctx}/api/shop/goods-words.do";

$("#namekeyword").bigAutocomplete({url:url\_, callback:**function**(data){

location.href="${ctx}/search.html?keyword="+data.title;

}});

</#**if**>