**谷粒商城**

版本：V 1.0

**商品详情页**

# 一 业务介绍

商品详情页，简单说就是以购物者的角度展现一个sku的详情信息。

这个页面不同于传统的crud的详情页，使用者并不是管理员，需要对信息进行查删改查，取而代之的是点击购买、放入购物车、切换颜色等等。

另外一个特点就是该页面的高访问量，虽然只是一个查询操作，但是由于频繁的访问所以我们必须对其性能进行最大程度的优化。

# 二 难点分析

1 光从功能角度上来说，并没有太多难点，唯一实现起来麻烦的就是用户对于不同销售属性的切换操作。



2 从性能角度来看，需要最大程度的提升页面的访问速度。

|  |
| --- |
| 难点：  1.skuInfo 表中一些基本信息。  2.销售属性展示：应该从spu\_sale\_attr,spu\_sale\_attr\_value  3.通过选择销售属性，来切换商品，实际上：通过选择销售属性展示不同的商品。 |

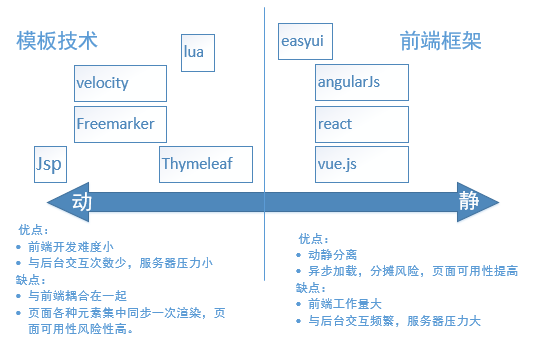
# 三 Thymeleaf

## 1 模板技术

把页面中的静态数据替换成从后台数据库中的数据。这种操作用jsp就可以实现。但是Spring boot 的架构不推荐使用Jsp，而且支持也不好，所以如果你是用springboot的话，一般使用Freemarker或者Thymeleaf。

而官方也是推荐使用**Thymeleaf**。

关于与前端有关的技术的比较



## 2 Thymeleaf 简介

Thymeleaf的主要目标是提供一个优雅和高度可维护的创建模板的方式。为了实现这一点，它建立在自然模板的概念上，将其逻辑注入到模板文件中，不会影响模板被用作设计原型。这改善了设计的沟通，弥合了设计和开发团队之间的差距。

比Jsp和Freemarker的优势，一般的模板技术都会在页面加各种表达式、标签甚至是java代码，而这些都必须要经过后台服务器的渲染才能打开。

但如果前端开发人员做页面调整，双击打开某个jsp或者ftl来查看效果，基本上是打不开的。

那么Thymeleaf的优势就出来了，因为Thymeleaf没有使用自定义的标签或语法，所有的模板语言都是扩展了标准H5标签的属性

比如 给div 添加一个值<div> hello </div>

|  |
| --- |
| <**div th:text="${item.skuName} "**></div> **th:text ：表示div显示的文本**  **${item.skuName} 表示在后台会有一个作用域将数据存储起来** |

它的效果和Jsp中的

|  |
| --- |
| <div>**${item.skuName}**</div> |

渲染后效果一样，但是如果你直接用浏览器打开页面文件，H5会把th:text这种不认识的属性忽略掉。效果就和<div></div> 没有区别，所以对于前端调页面影响更小。以上只是举了一个例子，如果是循环、分支的判断效果更明显。

## 3 快速入门：

项目创建，依赖模块web，Thymeleaf.模板。

### 3.1 所有头文件

就想Jsp的<%@Page %>一样 ，Thymeleaf的也要引入标签规范。不加这个虽然不影响程序运行，但是你的idea会认不出标签，不方便开发。

|  |
| --- |
| <**html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"**> |

关闭页面松校验

|  |
| --- |
| 页面需要进行松校验  在web-util 中添加  <**dependency**>  <**groupId**>net.sourceforge.nekohtml</**groupId**>  <**artifactId**>nekohtml</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>xml-apis</**groupId**>  <**artifactId**>xml-apis</**artifactId**> </**dependency**> <**dependency**>  <**groupId**>org.apache.xmlgraphics</**groupId**>  <**artifactId**>batik-ext</**artifactId**> </**dependency**> |
| 在application.properties中做热部署  **spring.thymeleaf.cache**=**false spring.thymeleaf.mode**=**LEGACYHTML5** |

### 3.2 取出请求域中的值，即取得request.attribute中的值，Model同样可以使用

|  |
| --- |
| <p th:text="${hello}" th:value="${hello}">打底值</p> |

### 3.3 循环

|  |
| --- |
| <table border="1">  <tr th:each="skuImage:${skuInfo.skuImageList}">  <td th:text="${skuImage.id}">  </td>  <td th:text="${skuImage.imgName}">  </td>  </tr> </table> |
| *<!--测试循环-->* <**table**>  *<!-- <c:foreach items="${arrayList}" var = "arry">-->* <**tr th:each="arry,stat: ${arrayList}"**>  <**td th:text="${arry}"**></**td**>  <**td th:text="${stat.index}"**></**td**>  </**tr**> </**table**> |

### 3.4 判断

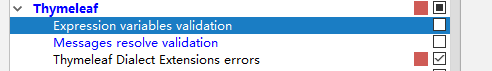
**Thymeleaf：判断的时候，数字类型，字符串类型，单引号可以加，也可以不加。**

|  |
| --- |
| 三元运算：  <td th:text="(${skuImage.id}=='10')?'是10':'不是10'"> </td>  If 判断 <td th:if="${skuImage.id}=='10'" >123456</td> |

如果idea编辑器质疑你页面中的元素是否存在



通常不准确，可以去掉验证，在settings->editor->Inspections中关掉验证。



### 3.5 th:utext :解析样式

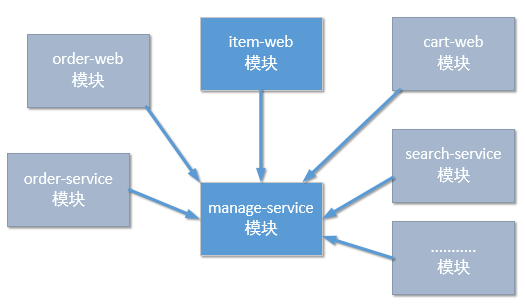
Controller:

|  |
| --- |
| request.setAttribute(**"gname"**,**"<span style=\"color:green\">宝强</span>"**); |

Th:utext:识别html中的标签

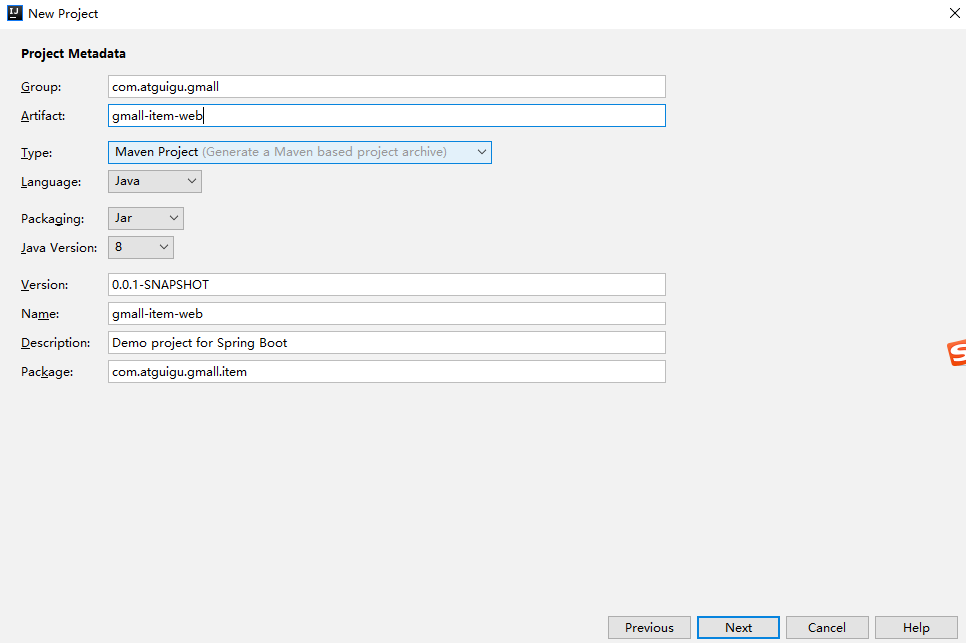
|  |
| --- |
| <**p th:utext="${gname}"**>color</**p**> |

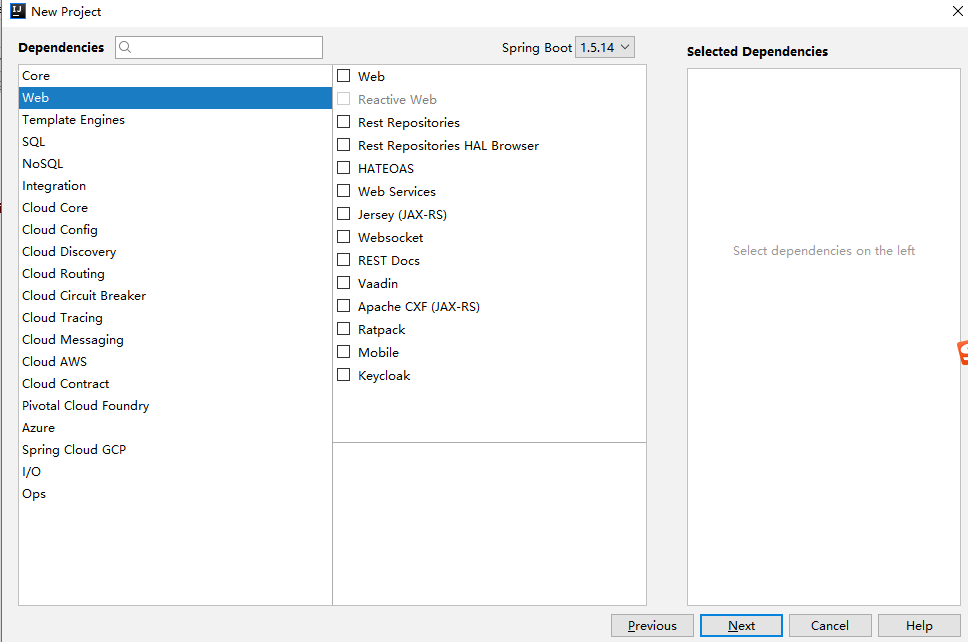
# 四 功能开发



# 五 开发详情页功能

## 项目搭建





一路下一步完成模块搭建

修改pom.xml

|  |
| --- |
| <**parent**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-parent</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**> </**parent**> <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-interface</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>com.atguigu.gmall</**groupId**>  <**artifactId**>gmall-web-util</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**> </**dependencies**> |

application.properties配置文件

|  |
| --- |
| **spring.thymeleaf.mode**=**LEGACYHTML5 server.port**=**8084 spring.thymeleaf.cache**=**false  spring.dubbo.application.name**=**item-web spring.dubbo.registry.protocol**=**zookeeper spring.dubbo.registry.address**=**192.168.67.200:2181 spring.dubbo.base-package**=**com.atguigu.gmall spring.dubbo.protocol.name**=**dubbo spring.dubbo.consumer.timeout**=**10000 spring.dubbo.consumer.check**=**false** |

## 页面资源导入

启动类上添加

|  |
| --- |
| @ComponentScan(basePackages = **"com.atguigu.gmall"**) redis 用！ |

## 测试项目是否正常

|  |
| --- |
| @Controller **public class** ItemController {  @RequestMapping(**"{skuId}.html"**)  **public** String skuInfoPage(@PathVariable(value = **"skuId"**) String skuId){  **return "item"**;  } } |

导入页面信息；在资料01中

## 简单信息查询显示

根据商品id查找商品信息，图片信息并在页面显示

|  |
| --- |
| 接口  SkuInfo getSkuInfo(String skuId); |
| 实现类  **public** SkuInfo getSkuInfo(String skuId){  SkuInfo skuInfo = **skuInfoMapper**.selectByPrimaryKey(skuId);  SkuImage skuImage = **new** SkuImage();  skuImage.setSkuId(skuId);  List<SkuImage> skuImageList = **skuImageMapper**.select(skuImage);  skuInfo.setSkuImageList(skuImageList);  **return** skuInfo; } |
| 控制器  @Controller **public class** ItemController {  @Reference  **private** ManageService **manageService**;   @RequestMapping(**"{skuId}.html"**)  **public** String skuInfoPage(@PathVariable(value = **"skuId"**) String skuId, Model model){  SkuInfo skuInfo = **manageService**.getSkuInfo(skuId);  model.addAttribute(**"skuInfo"**,skuInfo);  **return "item"**;  } } |

名称：

|  |
| --- |
| <**div class="box-name" th:text="${skuInfo.skuName}"**>  华为 HUAWEI Mate 10 6GB+128GB 亮黑色 移动联通电信4G手机 双卡双待 </**div**> |

价格：

|  |
| --- |
| <**span th:text="${#numbers.formatDecimal(skuInfo.price,1,2)}"**>4499.00</**span**> |

重量：

|  |
| --- |
| <**li th:text="${#numbers.formatDecimal(skuInfo.weight,1,2)}+' kg'"**></**li**> |

图片

|  |
| --- |
| <**div class="box-lh-one"**>  <**ul**>  <**li th:each="skuImage:${skuInfo.skuImageList}"**>  <**img th:src="${skuImage.imgUrl}"**/>  </**li**>  </**ul**> </**div**> |

**numbers.formatDecimal(<值>,<小数点左边的占位>,<小数点右边的保留位>)**

# 六 销售属性的处理

## 1 思路：

1、查出该商品的spu的所有销售属性和属性值

2、标识出本商品对应的销售属性

3、点击其他销售属性值的组合，跳转到另外的sku页面

## 2 查询出sku对应spu的销售属性

第1、2条通过此sql实现

|  |
| --- |
| SELECT sa.id ,sa.spu\_id, sa.sale\_attr\_name,sa.sale\_attr\_id,  sv.id sale\_attr\_value\_id,  sv.sale\_attr\_value\_name,  skv.sku\_id,  IF(skv.sku\_id IS NOT NULL,1,0) is\_checked[  FROM spu\_sale\_attr sa  INNER JOIN spu\_sale\_attr\_value sv ON sa.spu\_id=sv.spu\_id AND sa.sale\_attr\_id=sv.sale\_attr\_id  LEFT JOIN sku\_sale\_attr\_value skv ON skv.sale\_attr\_id= sa.sale\_attr\_id AND skv.sale\_attr\_value\_id=sv.id AND skv.sku\_id=18  WHERE sa.spu\_id=45  ORDER BY sv.sale\_attr\_id,sv.id |

此sql列出所有该spu的销售属性和属性值，并关联某skuid如果能关联上is\_check设为1，否则设为0。

在对应的实体类中【SpuSaleAttrValue】添加属性字段

|  |
| --- |
| @Transient String **isChecked** |
| 接口 SpuSaleAttrMapper  List<SpuSaleAttr> selectSpuSaleAttrListCheckBySku(String skuId, String spuId); |
| SpuSaleAttrMapper.xml  <**select id ="selectSpuSaleAttrListCheckBySku" resultMap="spuSaleAttrMap"**>  SELECT sa.id ,sa.spu\_id, sa.sale\_attr\_name,sa.sale\_attr\_id,  sv.id sale\_attr\_value\_id,  sv.sale\_attr\_value\_name,  skv.sku\_id,  IF(skv.sku\_id IS NULL,0,1) is\_checked  FROM spu\_sale\_attr sa  INNER JOIN spu\_sale\_attr\_value sv ON sa.spu\_id=sv.spu\_id AND sa.sale\_attr\_id=sv.sale\_attr\_id  LEFT JOIN sku\_sale\_attr\_value skv ON skv.sale\_attr\_id= sa.sale\_attr\_id AND skv.sale\_attr\_value\_id=sv.id AND skv.sku\_id=#{arg0}  WHERE sa.spu\_id=#{arg1}  ORDER BY sv.sale\_attr\_id,sv.id </**select**> |
| ManageService 接口  List<SpuSaleAttr> getSpuSaleAttrListCheckBySku(SkuInfo skuInfo); |
| **实现类**  **public** List<SpuSaleAttr> getSpuSaleAttrListCheckBySku (SkuInfo skuInfo){  return spuSaleAttrMapper.selectSpuSaleAttrListCheckBySku(skuInfo.getId(),skuInfo.getSpuId()); } |
| **控制器**  @RequestMapping(**"/{skuId}.html"**) **public** String getSkuInfo(@PathVariable(**"skuId"**) String skuId, Model model){  *// 存储基本的skuInfo信息* SkuInfo skuInfo = **manageService**.getSkuInfo(skuId);  model.addAttribute(**"skuInfo"**,skuInfo);  *// 存储 spu，sku数据* List<SpuSaleAttr> saleAttrList = **manageService**.getSpuSaleAttrListCheckBySku(skuInfo);  model.addAttribute(**"saleAttrList"**,saleAttrList);  **return "item"**; } |

页面开发部分

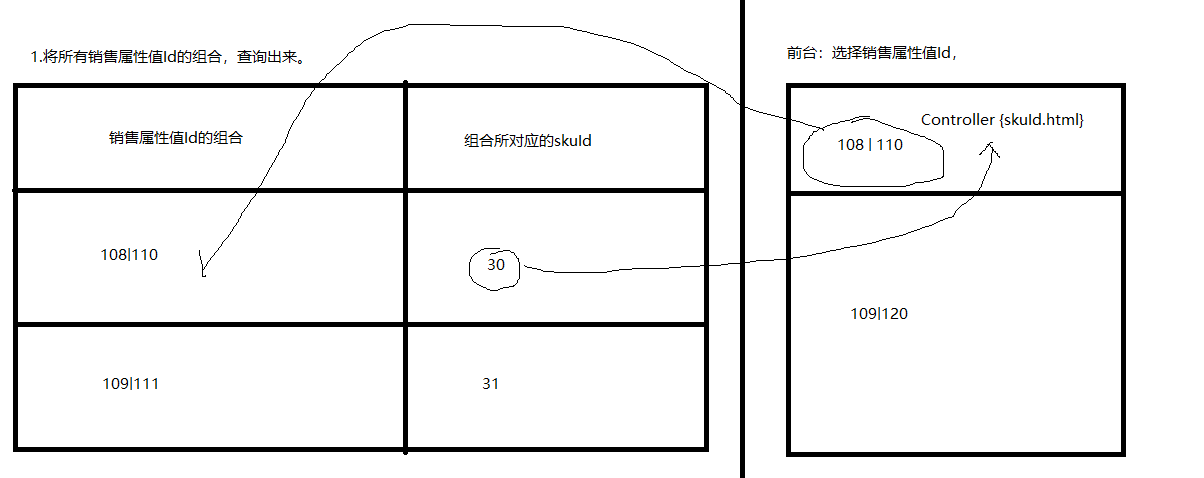
|  |
| --- |
| <**div class="box-attr-2 clear" th:each="saleAttrInfo: ${saleAttrList}"** >  <**dl**>  <**dt th:text="${saleAttrInfo.saleAttrName}"** >选择颜色</**dt**>  <**dd th:each="saleAttrValue: ${saleAttrInfo.spuSaleAttrValueList}" th:class="(${saleAttrValue.isChecked}=='1')?'redborder':''" class="redborder"** >  <**div th:value="${saleAttrValue.id}" th:text="${saleAttrValue.saleAttrValueName}"** >  摩卡金  </**div**>  </**dd**>  </**dl**>  <**br**/> </**div**> |

其中th:class的设置，redborder是一个自定义的样式类，主要是边框设红。如果isCheck=1标识当前这个sku的所拥有的属性值，所以锁定为红边框。



## 3 点击其他销售属性值的组合，跳转到另外的sku页面

实现思路：



1 、从页面中获得得所有选中的销售属性进行组合比如：

“属性值1|属性值2|属性值3” 用这个字符串匹配一个对照表，来获得skuId。并进行跳转，或者告知无货。

2、后台要生成一个“属性值1|属性值2|属性值3：skuId”的一个json串以提供页面进行匹配。如

3、需要从后台数据库查询出该spu下的所有skuId和属性值关联关系。然后加工成如上的Json串，用该json串，跟前台匹配。

实现：

SkuSaleAttrValueMapper.xml中sql

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>*** <**mapper namespace="com.atguigu.gmall0401.manage.mapper.SkuSaleAttrValueMapper"**>  <**select id="getSaleAttrValuesBySpu" resultMap="spuValueIdsMap"**>  SELECT sku\_id , GROUP\_CONCAT(sale\_attr\_value\_id ORDER BY sale\_attr\_value\_id ASC SEPARATOR '|') value\_ids  FROM `sku\_sale\_attr\_value` sv JOIN `sku\_info` si ON sv.sku\_id=si.id  WHERE si.spu\_id=#{spuId}  GROUP BY sku\_id  </**select**>  <**resultMap id="spuValueIdsMap" type="java.util.Map" autoMapping="true"**>  </**resultMap**> </**mapper**> |

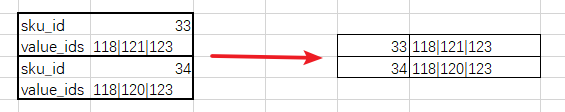
要注意排序，方便后面整理。

把同组的某个字段进行字符串拼接

GROUP\_CONCAT（拼接字段 排序字段 拼接分隔符）

实现类

目的：把List<Map>转换成为 Map



|  |
| --- |
| **gmall-interface接口**  **public** Map getSkuValueIdsMap(String spuId) |
| @Override **public** Map getSkuValueIdsMap(String spuId) {  List<Map> mapList = **skuSaleAttrValueMapper**.getSaleAttrValuesBySpu(spuId);  Map skuValueIdsMap =**new** HashMap();   **for** (Map map : mapList) {  String skuId =(Long ) map.get(**"sku\_id"**) +**""**;  String valueIds =(String ) map.get(**"value\_ids"**);  skuValueIdsMap.put(valueIds,skuId);   }  **return** skuValueIdsMap; } |

|  |
| --- |
| @GetMapping(**"{skuId}.html"**) **public** String item(@PathVariable(**"skuId"**) String skuId, HttpServletRequest request){  SkuInfo skuInfo = **manageService**.getSkuInfo(skuId);  List<SpuSaleAttr> spuSaleAttrList = **manageService**.getSpuSaleAttrListCheckSku(skuId, skuInfo.getSpuId());  request.setAttribute(**"skuInfo"**,skuInfo);  request.setAttribute(**"spuSaleAttrList"**,spuSaleAttrList);  *//得到属性组合与skuid的映射关系 ，用于页面根据属性组合进行跳转* Map skuValueIdsMap = **manageService**.getSkuValueIdsMap(skuInfo.getSpuId());  String valuesSkuJson = JSON.*toJSONString*(skuValueIdsMap);  request.setAttribute(**"valuesSkuJson"**,valuesSkuJson);  **return "item"**; } |

在item.html中增加隐藏域

|  |
| --- |
| <**input id="valuesSku" type="hidden" th:value= " ${valuesSkuJson}"** /> <**input id="skuId" type="hidden" th:value= " ${skuInfo.id}"** /> |

valuesSku存储属性值与skuid的对照json

skuId存放当前商品的skuId.

以上两个id 属性值不能变！

因为js.js 中需要获取id 的数据为固定的！

修改页面的js

|  |
| --- |
| *//红边框* **$**(**".box-attr-2 dd"**).click(**function**() {  **$**(**this**).addClass(**"redborder"**).siblings(**"dd"**).removeClass(**"redborder"**);  *switchSkuId*(); }) |

增加点击销售属性时触发切换sku方法

|  |
| --- |
| **function** *switchSkuId*() {  **var** checkDivs = **$**(**".redborder div"**);  **var** valueIds=**""**;  **for** (**var** i = 0; i < checkDivs.**length**; i++) {  **var** saleAttrValueDiv = checkDivs.eq(i);  **if**(i>0){  valueIds= valueIds+**"|"**;  }  valueIds=valueIds+saleAttrValueDiv.attr(**"value"**);  }  **var** valuesSku = **$**(**"#valuesSku"**).attr(**"value"**);  **console**.log(**"valuesSku:"**+valuesSku);  **var** valuesSkuJson=***JSON***.parse(valuesSku);  **var** skuId= valuesSkuJson[valueIds]; **var** skuIdSelf=**$**(**"#skuId"**).val(); **if**(skuId){  **if**(skuId==skuIdSelf){  **$**(**"#cartBtn"**).attr(**"class"**,**"box-btns-two"**);  }**else**{  **window**.**location**.**href**=**"/"**+skuId+**".html"**;  } }**else**{  **$**(**"#cartBtn"**).attr(**"class"**,**"box-btns-two-off"**); }  } |

最后测试：

