# 电子商城--SKU的实现

# 业务分析

SKU：库存量单位

针对电商而言，SKU指得是：  
 1、SKU是指一款商品，每款都有出现一个SKU，便于电商[品牌识别](https://www.baidu.com/s?wd=%E5%93%81%E7%89%8C%E8%AF%86%E5%88%AB&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "/Users/likang\\x/_blank)商品。  
 2、一款商品多色，则是有多个SKU，例：一件衣服，有红色、白色、蓝色，则SKU编码也不相同，如相同则会出现混淆，发错货。

# SKU如何理解：

一部 6S, 它身上有很多的属性和值. 比如 :

毛重: 420.00 g

产地: 中国大陆

容量: 16G, 64G, 128G

颜色: 银, 白, 玫瑰金

你跑进苏宁顺电, 说想要一台 6S, 店员也会再继续问: 你想要什么 6S? 16G 银色? 64G 白色?

每一台 6S 的毛重都是 420.00 g, 产地也都是 中国大陆. 这两个属性就属于 spu 属性.

而容量和颜色, 这种会影响价格和库存的(比如 16G 与 64G 的价格不同, 16G 银色还有货, 金色卖完了)属性就是 sku 属性.

spu 属性(不会影响到库存和价格的属性, 又叫关键属性、也叫基本属性) >>

毛重: 420.00 g

产地: 中国大陆

sku 属性(会影响到库存和价格的属性, 又叫销售属性) >>

容量: 16G, 64G, 128G

颜色: 银, 白, 玫瑰金

sku 在生成时, 会根据 属性生成 相应的 笛卡尔积.

# 流程设计

**表设计：**

1. 首先同一分类下商品的属性基本相同，所有在创建属性的时候建议关联上分类。
2. 商品的属性分为基本属性和销售属性，基本属性不参与销售，只用来展示商品的详细信息，比如手机的重量、长宽高等

而销售属性则需要参与商品的销售，例如商品的内存大小、颜色、网络等。

1. 设计表的时候，需要设计商品的基本属性表和商品的销售属性表
2. 商品会有共同的属性，属性拥有共同的属性值，比如手机存储空间有32g 64g 128g等，基本所有手机的存储空间都是这么大，所有需要把这些共同的属性值也单独存储，因此需要设计属性值表。
3. 属性和属性值表需要根据属性id进行关联。

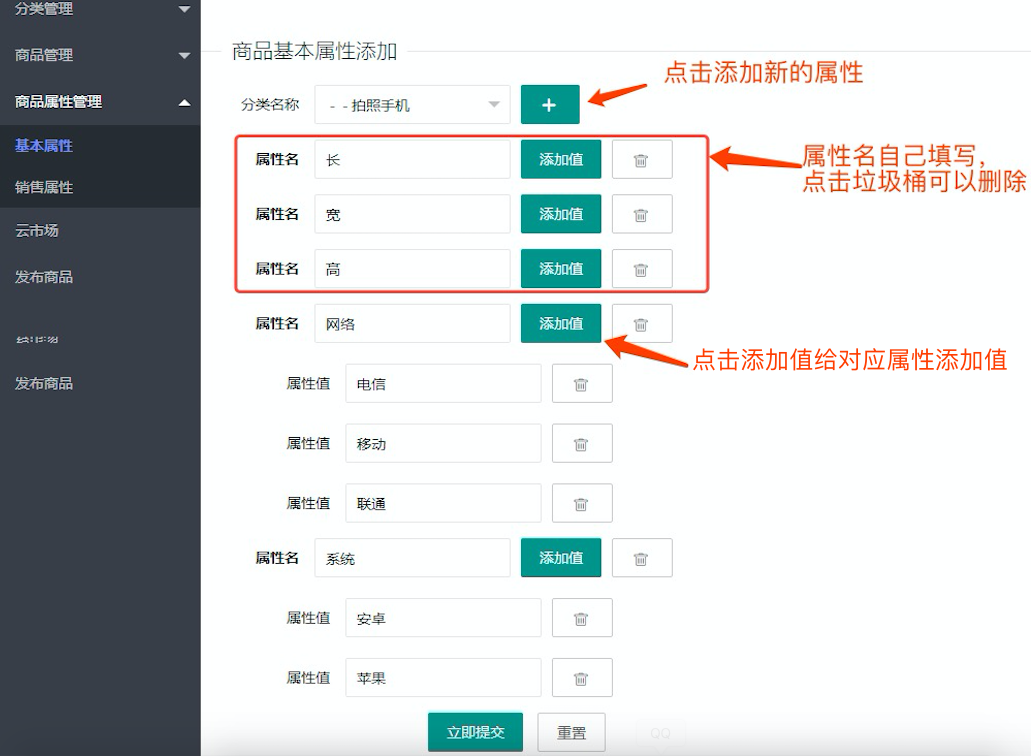
所以：

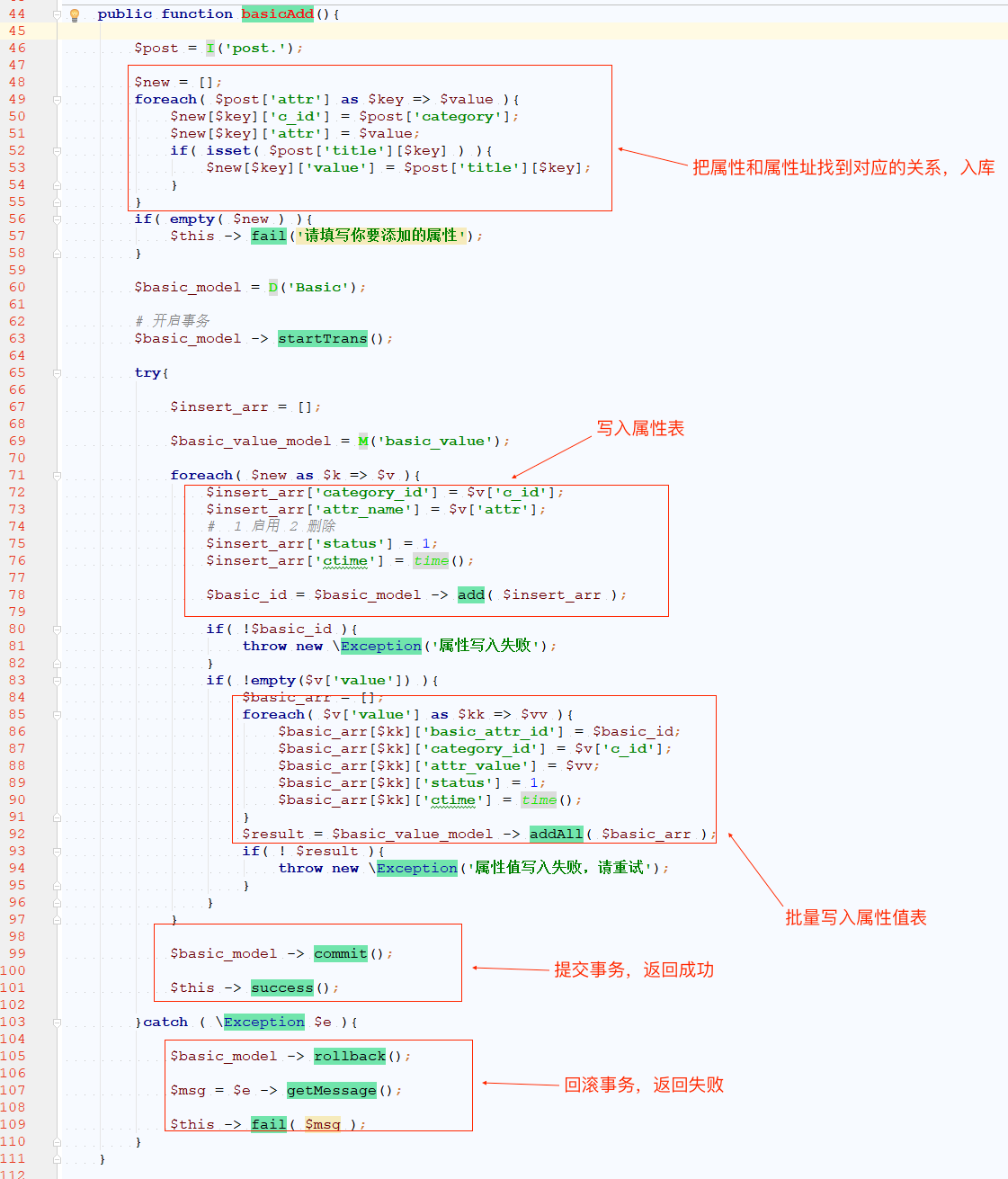
在实现SKU的时候建议设计四张表

1. 基本属性表
2. 基本属性值表
3. 销售属性表
4. 销售属性值表。

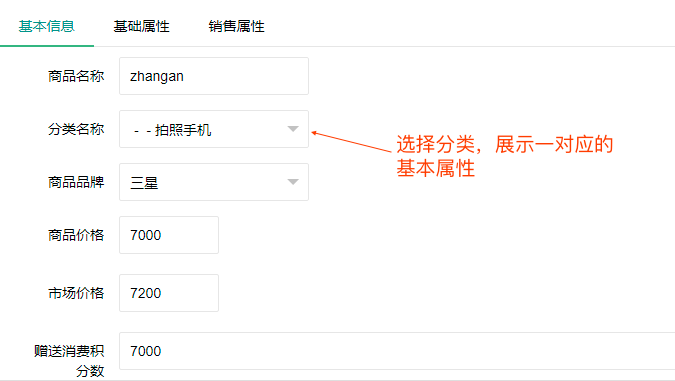
# 代码整理

基本属性添加页面：

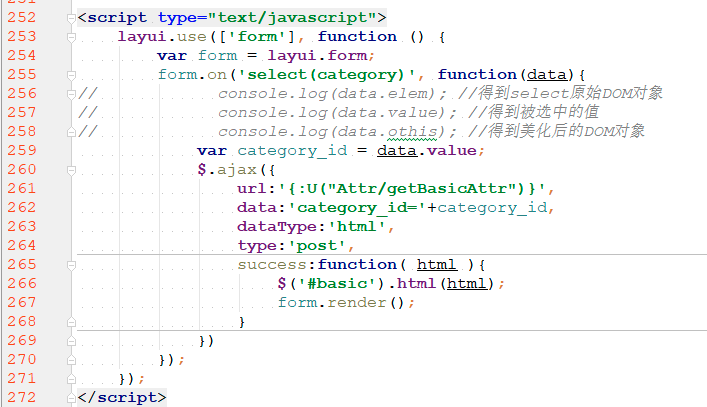




修改添加商品页面，在选择完分类之后展示对应的基本属性数据



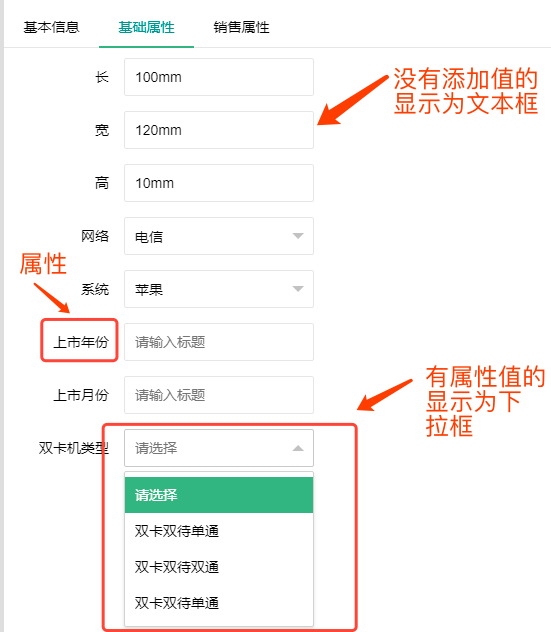
给下拉框加选择事件



ajax获取分类对应的基本属性



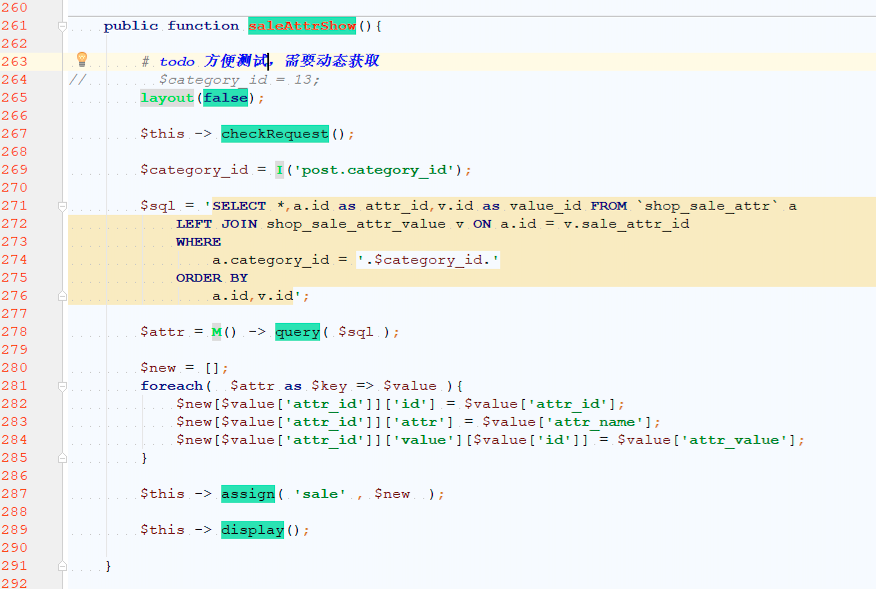
页面展示：



销售属性添加逻辑和基本属性类似。展示页面不同 【销售属性不允许存在属性值为空的属性，必须有内容，建议不要超过三个属性】



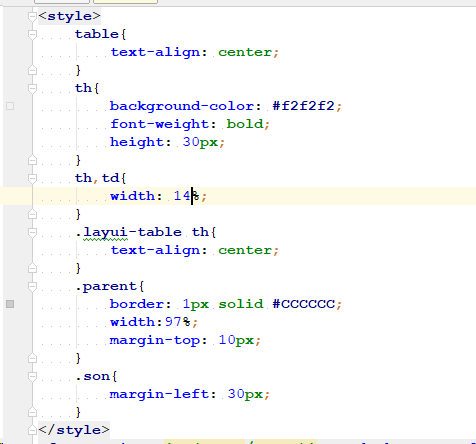
获取销售属性：



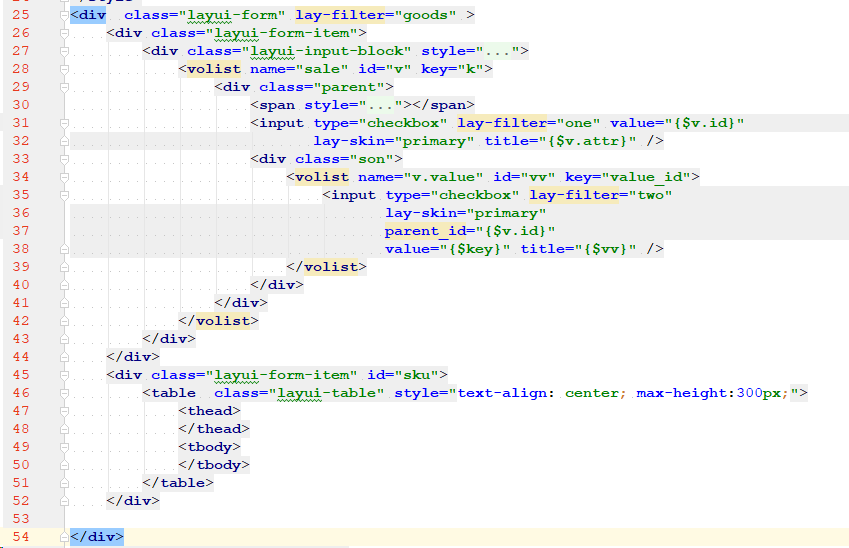
页面展示如下图： 【根据选择的属性和属性址，组合成不同sku】



页面需要的css



页面：



全选、反选事件



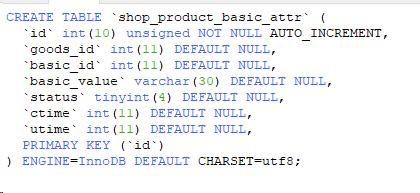
# 商品添加流程改动

上述流程实现了属性和分类关联，后边需要继续完善属性和商品的关联

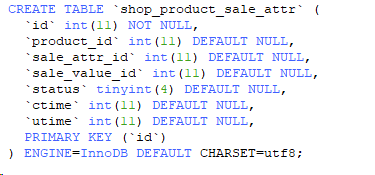
1. 首先 需要需要建立商品和属性的关联表。基本属性关联比较简单，直接关联属性id和属性值即可（没有属性值的直接存储输入的值）
2. 建立商品和销售属性的关联表，销售属性需要参与购买，而且销售属性需要组合起来使用。而且销售属性不存在没有属性值的情况。
3. 需要单独创建表存储对应的属性信息。

## 表设计：

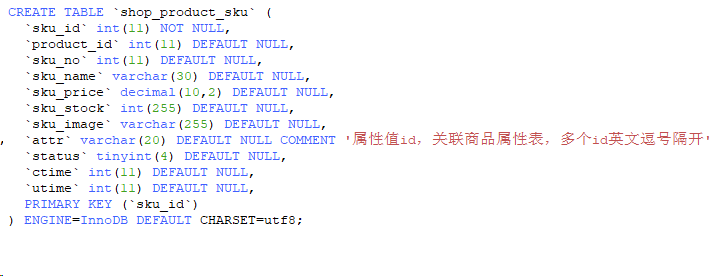
商品和基本属性的关联表：



商品和销售属性的关联表



货品表（商品和销售属性组合而来）



## 流程设计：

1. 在添加商品的时候，填写商品的基本信息，插入商品表数据
2. 根据商品分类筛选出对应的基本属性，根据属性值展示不同内容，插入商品的基本属性表中。
3. 根据分类筛选出对应的销售属性，组合成为货品，把数据写入商品的销售属性表。
4. 根据选择的销售属性，组合成sku，把商品的销售属性的id和商品id，以及库存、价格等写入货品表。
5. 前台商品列表页面修改为货品展示。
6. 前台商品详情页进入之后可以查询商品的销售属性表查询出对应的属性，根据用户选择的属性组合为货品。

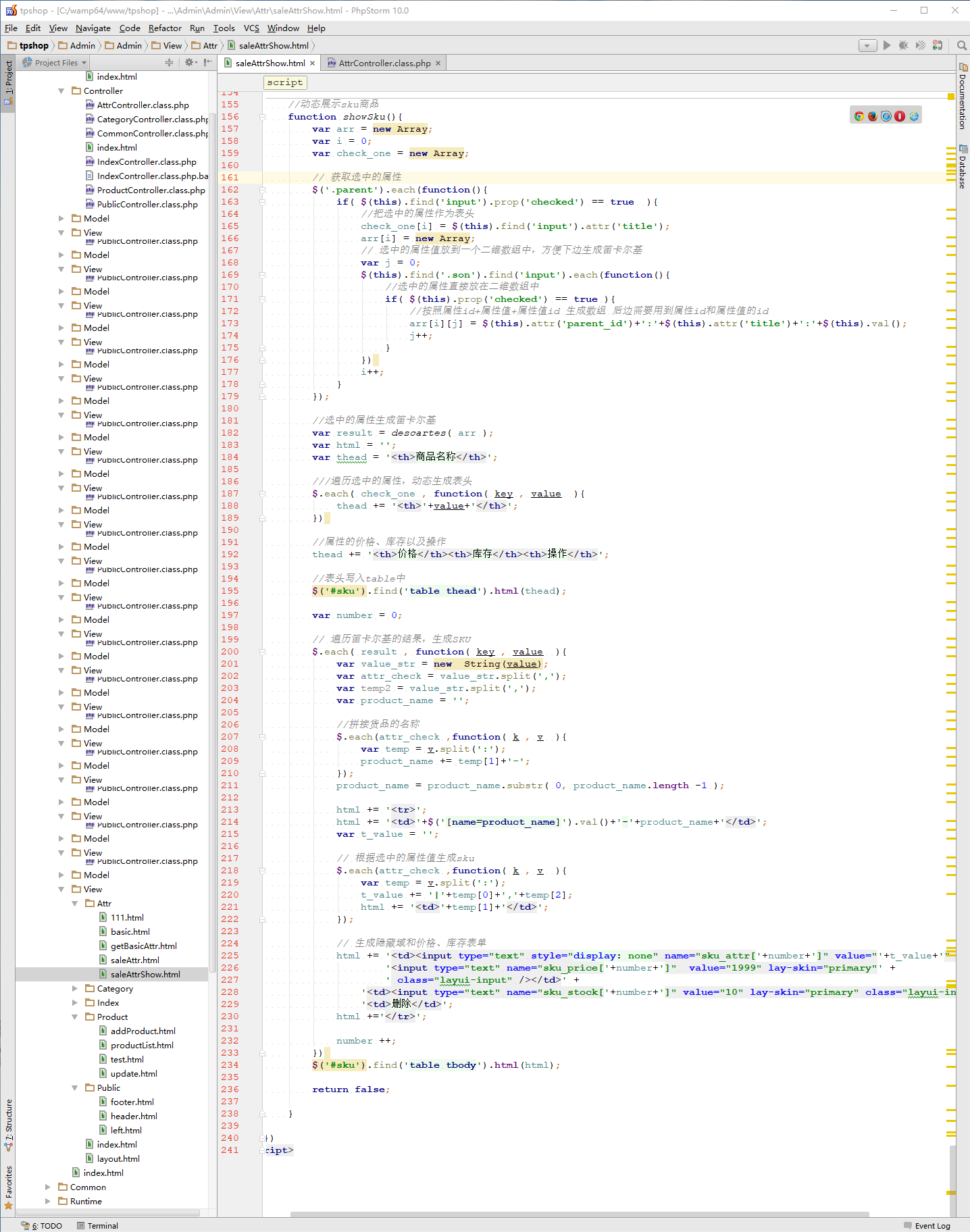
商品入库，下边是商品添加提交到后台的所有数据



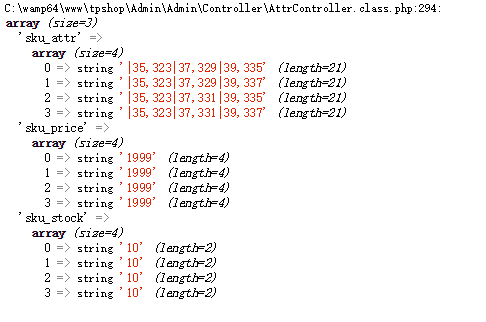
笛卡尔积算法见代码参考

动态生成表格js代码

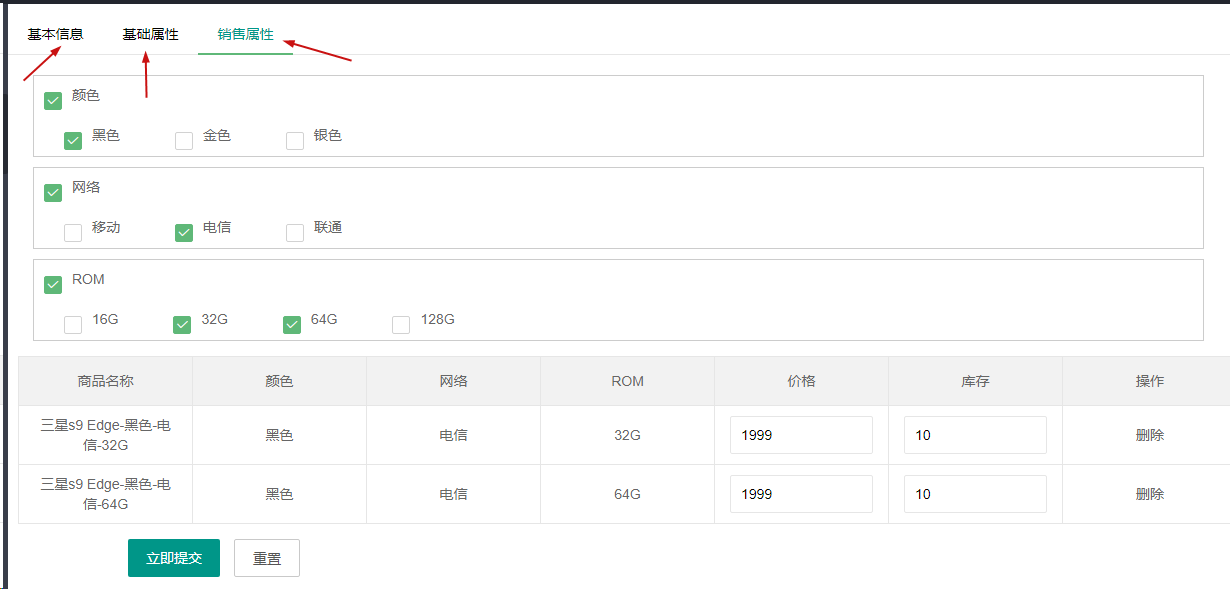
获取销售属性：



把表单提交数据格式为：组合起来之后变成一个完整的商品，下边为0的所有组合起来为属性id和属性值id，价格为1999，库存为10



修改商品添加页面，添加table选项卡功能，点击对应的选项卡可以晚上对应的数据。修改三个选项卡共用一个提交按钮。

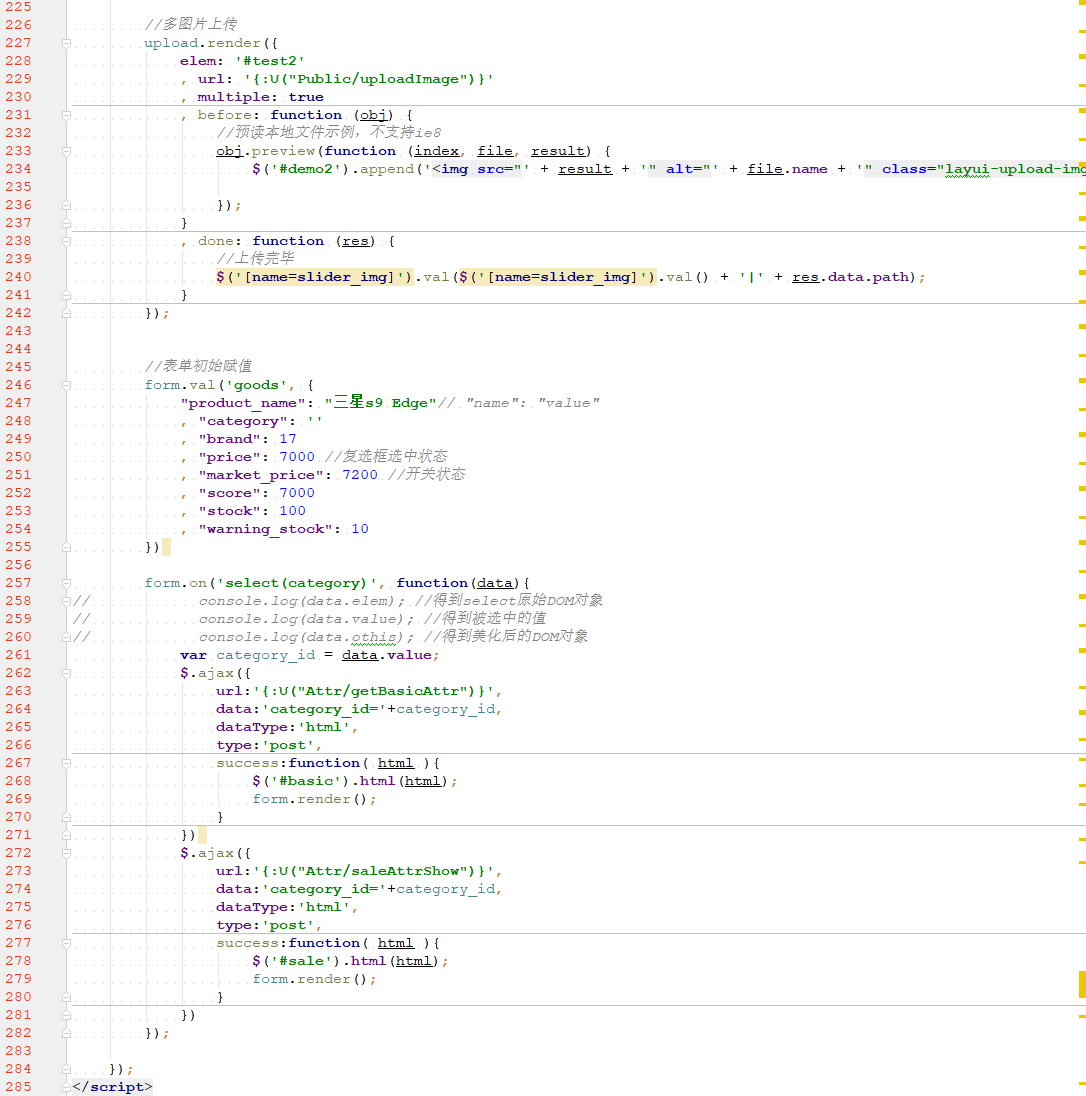




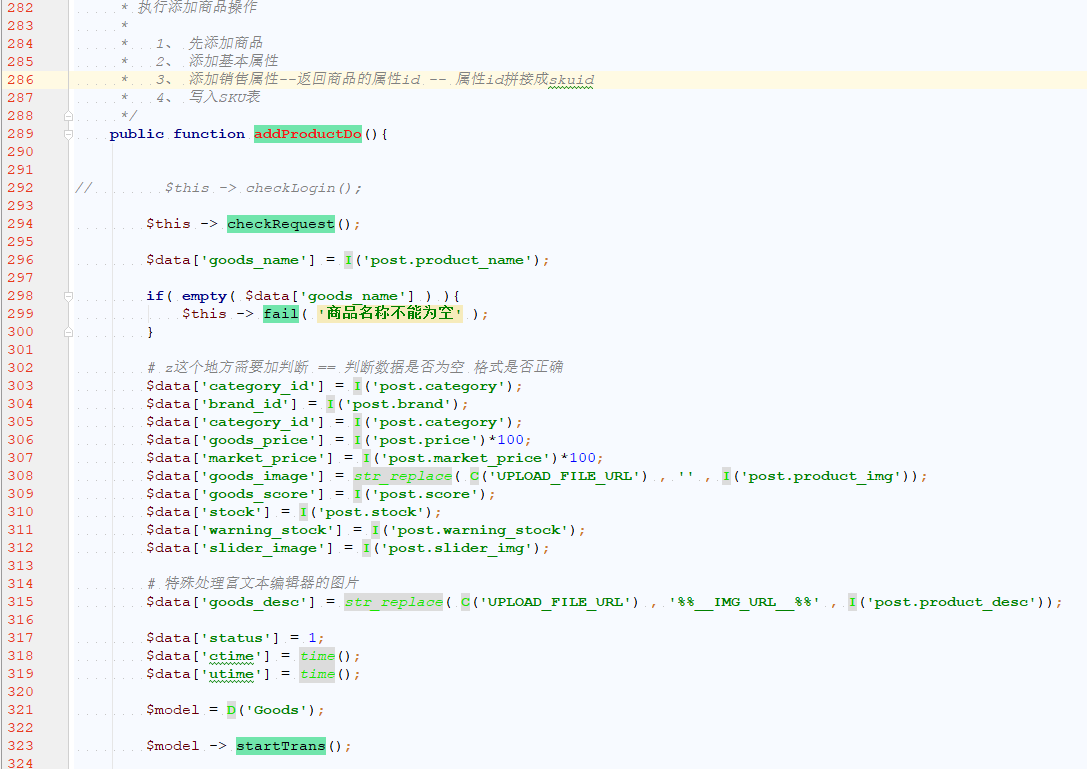




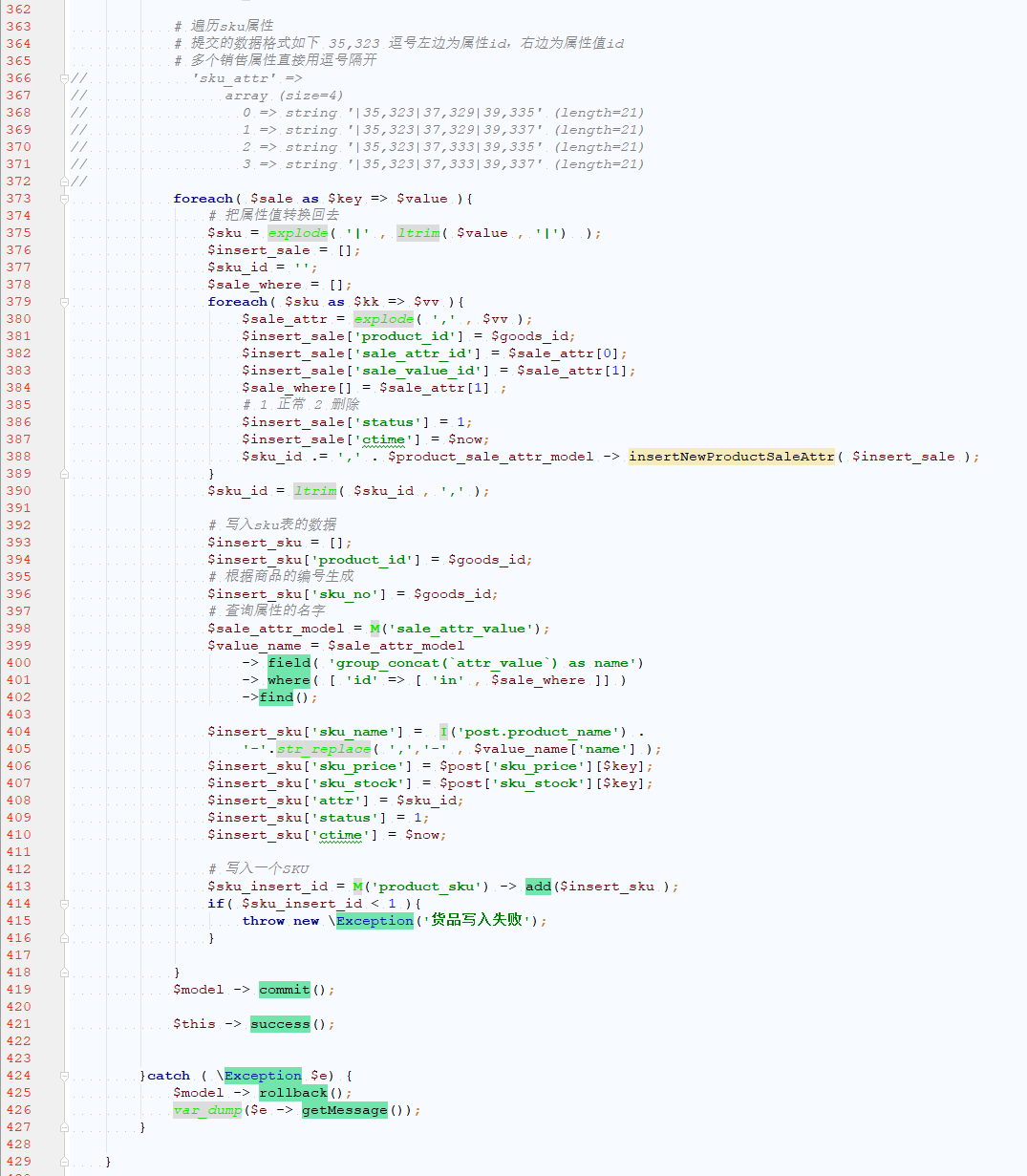




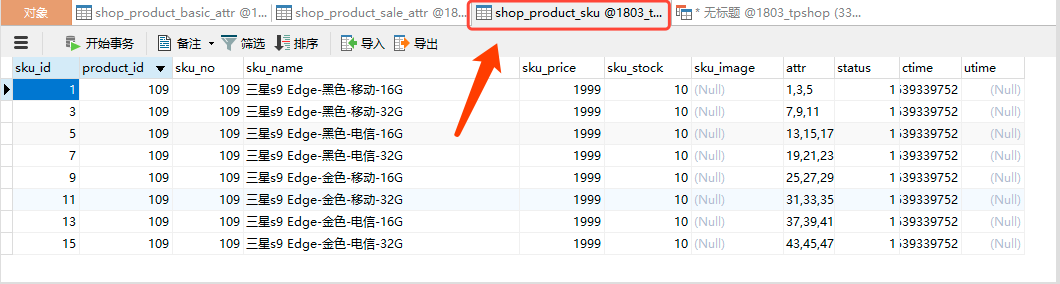
插入SKU相关表：



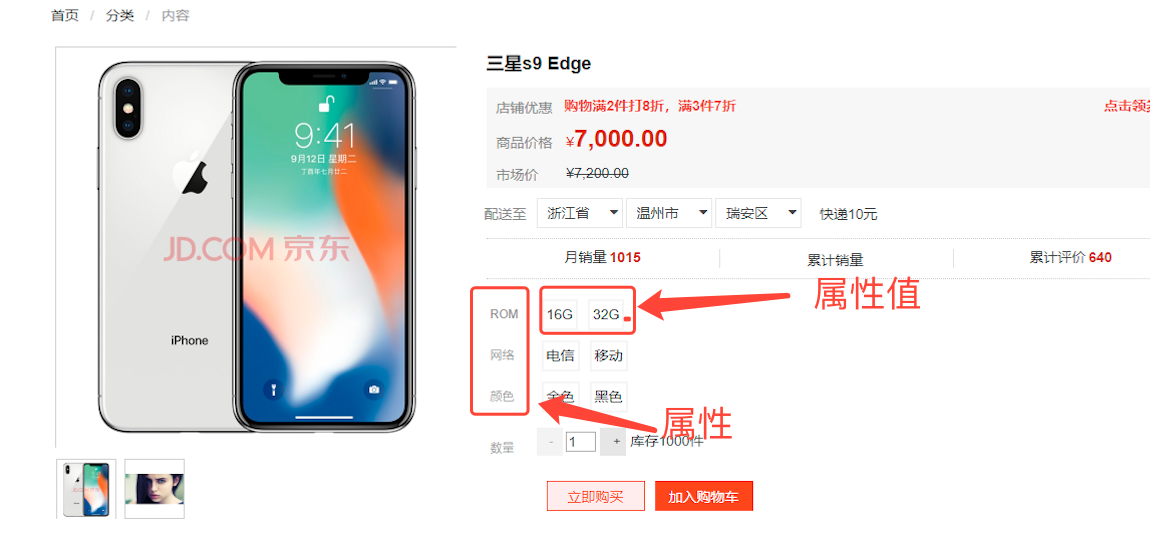




入库之后SKU如下：



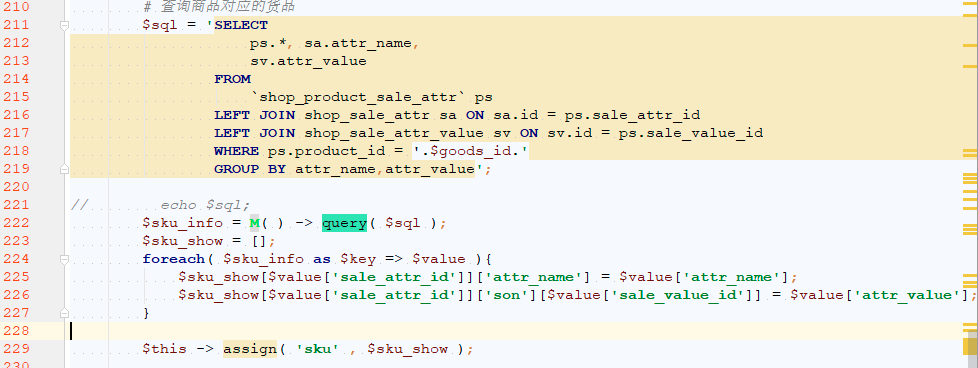
修改前台商品详情页：



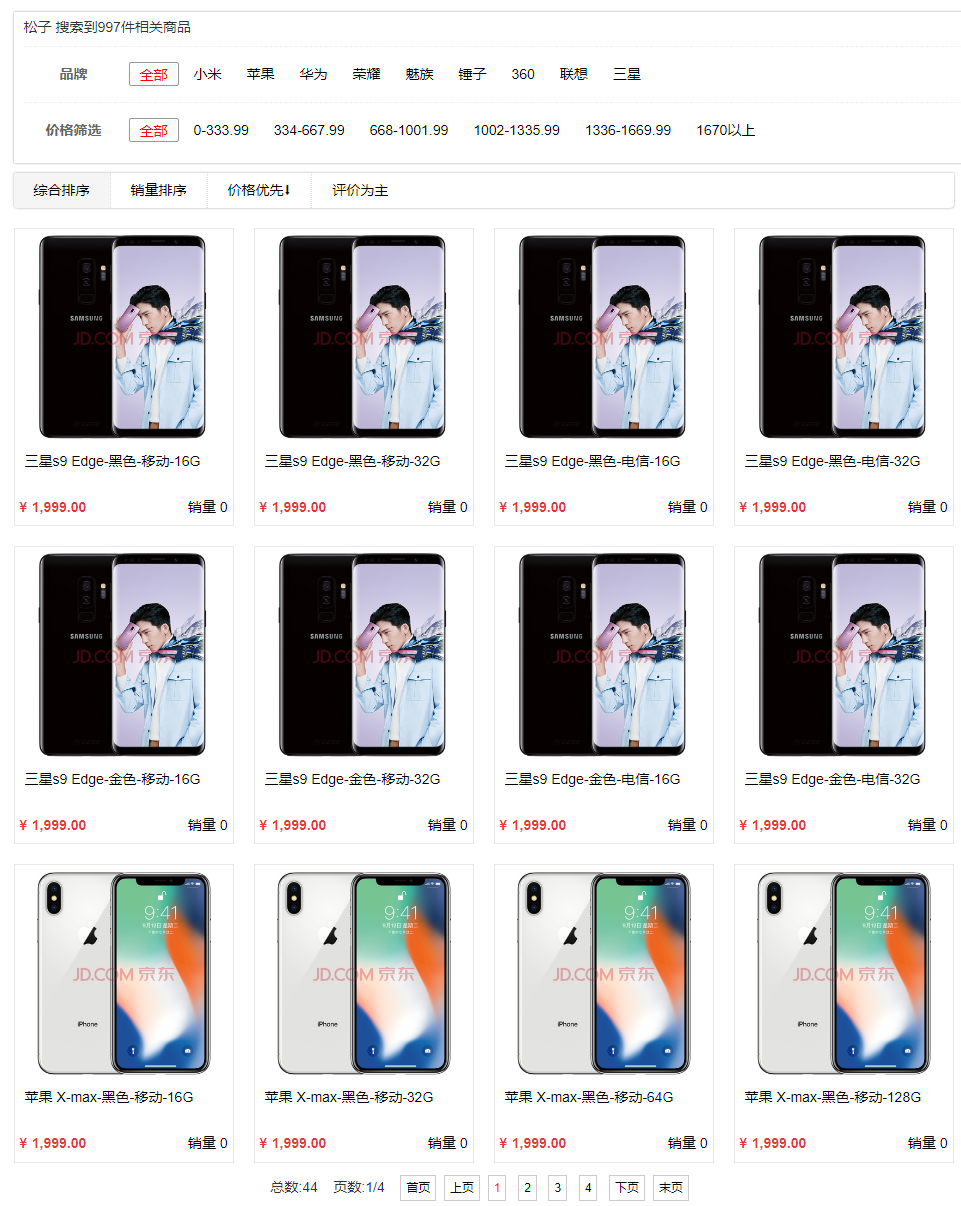
页面展示代码：



根据商品取出销售属性信息：



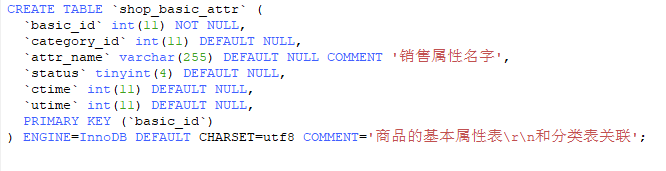
修改商品列表页，展示的为货品：



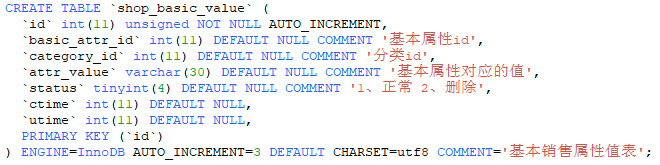


# 数据库设计

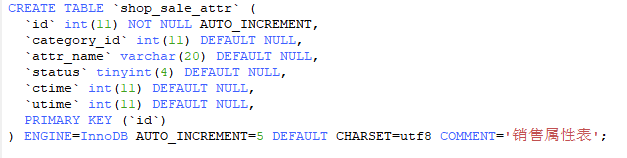
**销售属性表：**



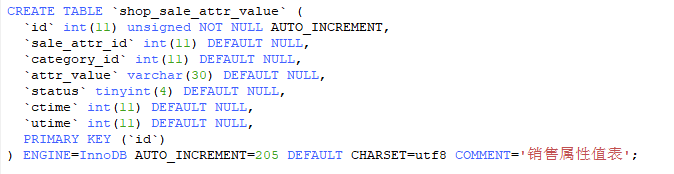
**销售属性值表：**



**销售属性表：**



**销售属性址表：**



代码参考：

// 笛卡尔积算法

function descartes(list)

{

//parent上一级索引;count指针计数

var point = {};

var result = [];

var pIndex = null;

var tempCount = 0;

var temp = [];

//根据参数列生成指针对象

for(var index in list)

{

if(typeof list[index] == 'object')

{

point[index] = {'parent':pIndex,'count':0}

pIndex = index;

}

}

//单维度数据结构直接返回

if(pIndex == null)

{

return list;

}

//动态生成笛卡尔积

while(true)

{

for(var index in list)

{

tempCount = point[index]['count'];

temp.push(list[index][tempCount]);

}

//压入结果数组

result.push(temp);

temp = [];

//检查指针最大值问题

while(true)

{

if(point[index]['count']+1 >= list[index].length)

{

point[index]['count'] = 0;

pIndex = point[index]['parent'];

if(pIndex == null)

{

return result;

}

//赋值parent进行再次检查

index = pIndex;

}

else

{

point[index]['count']++;

break;

}

}

}

}