

谷粒商城

版本：V 1.0

[www.atguigu.com](http://www.atguigu.com)

## 一 业务介绍

### 1 SPU 与 SKU

SPU(Standard Product Unit): 标准化产品单元。是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息的集合，该集合描述了一个产品的特性。

Stock Keeping Unit (库存量单位)。即库存进出计量的基本单元，可以是以件，盒，托盘等为单位。SKU 这是对于大型连锁超市 DC（配送中心）物流管理的一个必要的方法。现在已经被引申为产品统一编号的简称，每种产品均对应有唯一的 SKU 号。

比如，咱们购买一台 iPhoneX 手机，iPhoneX 手机就是一个 SPU，但是你购买的时候，不可能以 iPhoneX 手机为单位买的，商家也不可能以 iPhoneX 为单位记录库存。必须要以什么颜色什么版本的 iPhoneX 为单位。比如，你购买的是一台银色、128G 内存的、支持联通网络的 iPhoneX，商家也会以这个单位来记录库存数。那这个更细致的单位就叫库存单元（SKU）。



### 销售属性与平台属性

销售属性，就是商品详情页右边，可以通过销售属性来定位一组 spu 下的哪款 sku。可以让当前的商品详情页，跳转到自己的“兄弟”商品。

一般每种商品的销售属性不会太多，大约 1-4 种。整个电商的销售属性种类也不会太多，大概 10 种以内。比如：颜色、尺寸、版本、套装等等。不同销售属性的组合也就构成了一个 spu 下多个 sku 的结构。



平台属性，就是之前分类下面，辅助搜索的，类似于条件的属性。

便捷导购：	老人机	长续航手机	高清大屏			
系统：	安卓 (Android)	苹果 (IOS)	微软 (WindowsPhone)	基础功能机系统	其他	
运行内存：	8GB	6GB	4GB	3GB	2GB	2GB以下
						更多选项 ( 热点、屏幕尺寸、机身颜色 等 ) ▾

销售属性与平台属性各自独立。一个 SPU 会决定一个商品都有哪些销售属性，比如 iPhone 会有颜色、版本、内存的销售属性，某个 T 恤衫只有尺寸这个销售属性。

而某个商品有什么平台属性，由他的 3 级分类决定。比如笔记本包括：运行内存、cpu、显卡、硬盘、屏幕尺寸等等。

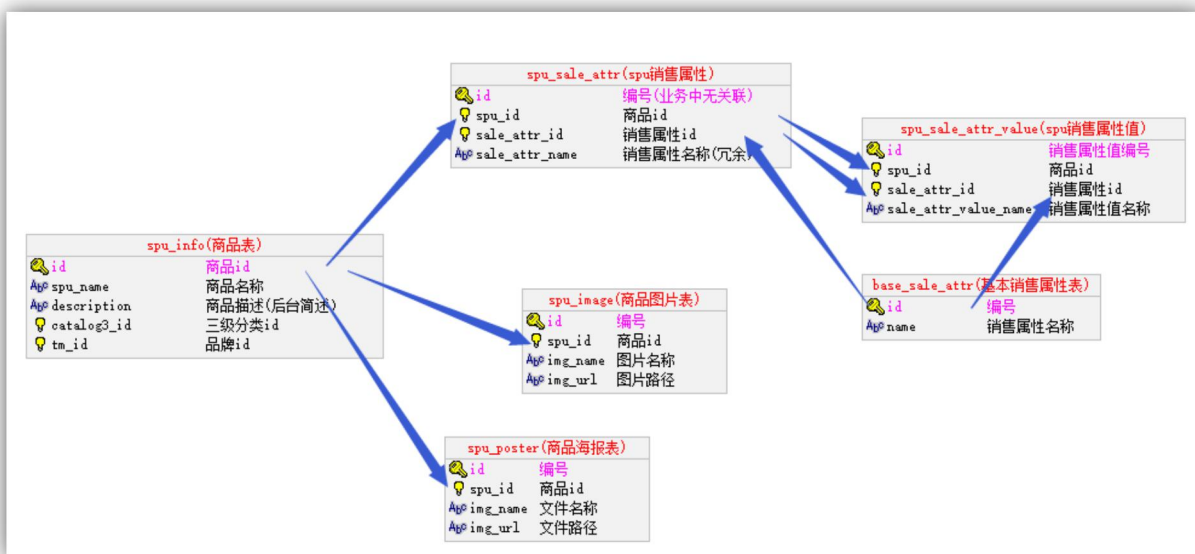
## 2 SKU 与 SPU 的图片资源

另外同一个 SPU 下的 SKU 可以共用一些资源，比如商品图片，海报等等。毕竟同一种商品，大部分图片都是共用的只有因为颜色尺寸等，很少的差别。那么一般来说商品图片都是在新增 SPU 时上传的，在新增 SKU 时从该 SPU 已上传的图片中选择。

而海报几乎是所有 SPU 下的 SKU 都一样。

## 3 数据结构图

根据以上的需求，以此将 SPU 关联的数据库表结构设计为如下：



## 4 数据示例:

spu_info(商品表)				基本销售属性			
商品编号(spu_id)	spu_name			id	name		
6110	iPhoneX			101	颜色		
				102	尺寸		
				103	版本		
				104	内存容量		

spu销售属性				spu销售属性值			
id	spu_id	sale_attr_id	sale_attr_name	id	spu_id	sale_attr_id	sale_attr_value_name
9901	6110	101	颜色	88801	6110	101	太空银
9902	6110	103	版本	88802	6110	101	土豪金
9903	6110	104	内存容量	88803	6110	103	移动
				88804	6110	103	联通
				88805	6110	104	64G
				88806	6110	104	256G

spu_image 图片表				spu_poster 海报表			
id	spu_id	img_name	img_url	id	spu_id	img_name	img_url
77701	6110	正面图	http://xxxx/xxx/xx.jpg	67701	6110	海报1	http://xxxx/xxx/xx.jpg
77702	6110	侧面图	http://xxxx/xxx/xx.jpg	67702	6110	海报2	http://xxxx/xxx/xx.jpg
77703	6110	反面图	http://xxxx/xxx/xx.jpg	67703	6110	海报3	http://xxxx/xxx/xx.jpg
77704	6110	太空银版图	http://xxxx/xxx/xx.jpg	67704	6110	海报4	http://xxxx/xxx/xx.jpg
77705	6110	土豪金版图	http://xxxx/xxx/xx.jpg	67705	6110	海报5	http://xxxx/xxx/xx.jpg

## 二、列表查询功能开发

### 1 首页菜单 index.html

```
<div title="商品信息管理">
  <ul class="nav">
    <li><a href="javascript:add_tab('商品信息管理','spuListPage')" style="text-decoration:none;">商品
    SPU 管理</a></li>
  </ul>
</div>
```

不要忘记增加 controller 的跳转。

### 2 SpuManageController

```
@Controller
public class SpuManageController {

    @RequestMapping("spuListPage")
    public String getSpuListPage(){
        return "spuListPage";
    }
}
```

### 3 SPU 列表页面

和属性列表大体相同，都是通过三级分类，获得 spu 信息列表。

spuListPage.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>spu 列表</title>
    <script type="text/javascript" src="/easyui/jquery.min.js"></script>
    <script type="text/javascript" src="/easyui/jquery.easyui.min.js"></script>
    <script type="text/javascript" src="/easyui/easyloader.js"></script>

    <link rel="stylesheet" type="text/css" href="/easyui/themes/icon.css">
    <link rel="stylesheet" type="text/css" href="/easyui/themes/default/easyui.css">
```

```

<!-- 引入图片上传工具 webuploader -->
<link rel="stylesheet" type="text/css" href="/webuploader/webuploader.css">
<script type="text/javascript" src="/webuploader/webuploader.js"></script>
</head>
<body>
<div class="easyui-panel" title="" data-options="border:true">
  <!-- 列表 -->
  <table id="spulist_dg" class="easyui-datagrid" title="spu 列表"
    data-options="singleSelect:true,method:'get',toolbar:'#spulist_tb'">
    <thead>
      <tr>
        <th data-options="field:'id' width='10%'">商品 id </th>
        <th data-options="field:'spuName' width='30%'">商品名称</th>
        <th data-options="field:'description' width='60%'">商品描述 </th>
      </tr>
    </thead>
  </table>
  <!-- 列表的工具栏 -->
  <div id="spulist_tb" style="padding:5px;height:auto">
    <div style="margin-bottom:5px">
      <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain="true" onclick="addSpuInfo()">添
    加</a>
      <a href="#" class="easyui-linkbutton" iconCls="icon-edit" plain="true" onclick="editSpuInfo()">编
    辑</a>
      <a href="#" class="easyui-linkbutton" iconCls="icon-remove" plain="true">删除</a>
      <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain="true" onclick="addSkuInfo()">增
    加 sku</a>
      <a href="#" class="easyui-linkbutton" iconCls="icon-search" plain="true"
    onclick="showSkuInfoList()"> sku 列表</a>
    </div>
    <div>
      一级分类:
      <select id="ctg1ForSpuList" class="easyui-combobox" style="width:100px"
    data-options="valueField:'id',textField:'name',url:'getCatalog1',
    onSelect:function(rec){
      $('#ctg2ForSpuList').combobox('clear');
      $('#ctg3ForSpuList').combobox('clear');
      $('#ctg2ForSpuList').combobox('reload','getCatalog2?catalog1Id='+rec.id);
    }" ></select>
      二级分类:
      <select name="ctg2ForSpuList" id="ctg2ForSpuList" class="easyui-combobox"
    data-options="valueField:'id',textField:'name',
    onSelect:function(rec){
      $('#ctg3ForSpuList').combobox('clear');
      $('#ctg3ForSpuList').combobox('reload','getCatalog3?catalog2Id='+rec.id);
    }" style="width:100px" ></select>
      三级分类:
      <select name="ctg3ForSpuList" id="ctg3ForSpuList" class="easyui-combobox"
    data-options="valueField:'id',textField:'name',
    onSelect:function(rec){
      $('#spulist_dg').datagrid({url:'spuList?catalog3Id='+rec.id});
    }
  </div>
    </div>
  </div>

```

```
" style="width:100px" ></select>
    <a href="#" class="easyui-linkbutton" iconCls="icon-search"
onclick="javascript:reloadSpuList()" >刷新列表</a>
  </div>
</div>
</div>
```

特别注意：咱们现在使用的是 **easyui** 的单一页面模式，也就是说其实所有的页面最后都会加载到一个页面上，那么要注意的地方就是中所有打开的标签页的元素 **id**，都不能重复否则会发生冲突。

## 4 后台代码

### 4.1 controller

```
@RequestMapping("spuList")
@ResponseBody
public List<SpuInfo> getSpuInfoList(@RequestParam Map<String,String> map){
    String catalog3Id = map.get("catalog3Id");
    SpuInfo spuInfo =new SpuInfo();
    spuInfo.setCatalog3Id(catalog3Id);
    List<SpuInfo> spuInfoList = manageService.getSpuInfoList(spuInfo);
    return spuInfoList;
}
```

### 4.2 bean

```
public class SpuInfo implements Serializable {
    private String id;
    private String spuName;
    private String description;
    private String catalog3Id;
}
```

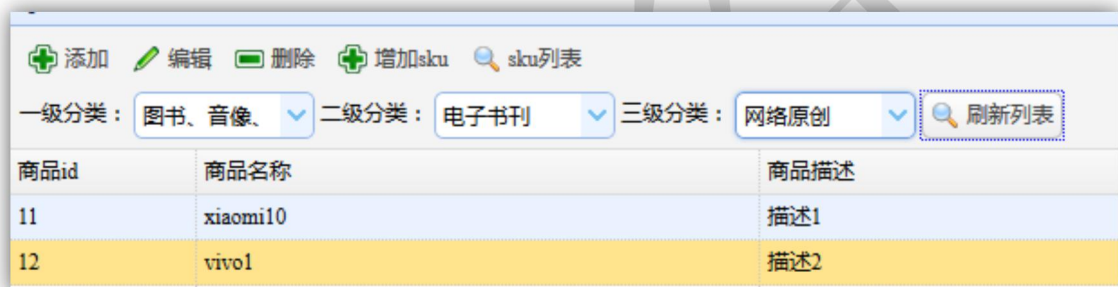
在 `gmall-manage-service` 工程中增加

### 4.3 mapper

```
public interface SpuInfoMapper extends Mapper<SpuInfo> {  
}
```

实现类 `manageServiceImpl` 增加方法

```
public List<SpuInfo> getSpuInfoList(SpuInfo spuInfo){  
    List<SpuInfo> spuInfoList = spuInfoMapper.select(spuInfo);  
    return spuInfoList;  
}
```

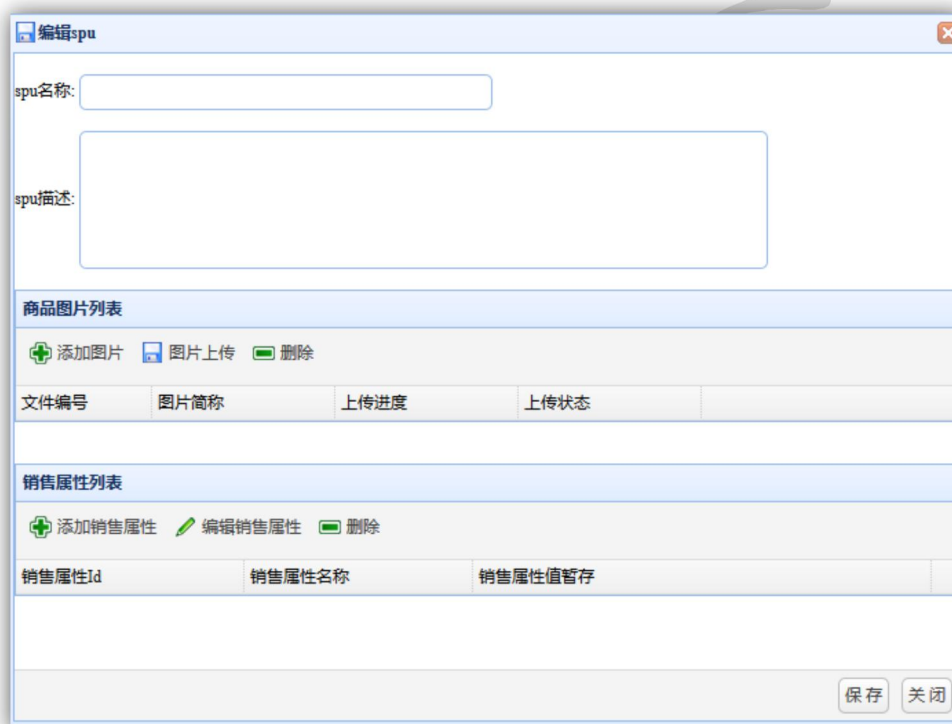


商品id	商品名称	商品描述
11	xiaomi10	描述1
12	vivo1	描述2



### 三、 spu 的保存功能

#### 1 点击新增弹出录入框



首先由列表页点击【增加】出现弹出框

```
function addSpulInfo(){  
    initSpulInfoDlg(); //在 spuInfoPage.html 中  
}
```

由于这次弹出框是新的窗体，虽然整个后台系统是一个单一页面，但是如果把所有代码都集中到一个 html 的话，维护起来非常不方便，可读性也差。所以咱们利用页面的渲染工具 themeleaf 的一个标签，把一段代码提取到另外一个 html 文件中。

```
<div th:include="spuInfoPage"></div>
```

这个有点类似于 Jsp 的 `<jsp:include page="">` 标签。

这样我们跟这个新增弹出框窗体有关系的都可以放到 `spuInfoPage.html` 这个文件中了。

## 1.1 创建 `spuInfoPage.html`（商品 `spu` 的详情页）

代码见思维导图

## 2 难点分析

两个难点：

- 1、涉及上传图片附件，而且要上传多个图片附件，每个图片都要可以管理分配，因为每张图片最后都要分配给不同的 `sku`。同时上传成功后用户可以预览。
- 2、一个 `spu` 对应多个销售属性，但是每个销售属性又对应了多个属性值。

## 3 图片上传预览解决方案：

- ※ WebUploader (客户端图片上传控件)
- ※ FastDFS(服务器端文件管理软件)
- ※ datagridview (easyui 预览控件)

### 3.1 首先先说 WebUploader

官方网站：‘

WebUploader 是由 Baidu WebFE(FEX)团队开发的一个简单的以 HTML5 为主，FLASH 为辅的现代文件上传组件。

如何使用，可以参考官网上给的例子。

简单归纳：要安装约定完成控件需要的几个动作。

### 3.2 webuploader 的实现步骤

控件的创建	<pre> var spulmgUploader = WebUploader.create({     auto:false,     // swf 文件路径     swf: '/webuploader/Uploader.swf',     // 文件接收路径     server: '/fileUpload',     // 选择文件的按钮。     pick: '#spulmgAdd',     // 不压缩 image, 默认如果是 jpeg, 文件上传前会压缩一把再上传!     resize: false,     // 设定文件大小上限 2M     fileSingleSizeLimit:2*1024*1024,     // 可接受的文件类型     accept: {         title: 'Images',         extensions: 'gif,jpg,jpeg,bmp,png',         mimeTypes: 'image/*'     } }); </pre>
文件被选择后	<pre> spulmgUploader.on('fileQueued',function (file) {     console.log("用户增加文件:"+file.id+' '+file.name); }); </pre>
上传过程中，会反复触发的事件，每次 percentage 会变化。	<pre> spulmgUploader.on( 'uploadProgress', function( file, percentage ) {     console.log("用户增加文件:"+file.id+' '+file.name+' '+ percentage); }); </pre>
上传成功后触发的事件	<pre> spulmgUploader.on( 'uploadSuccess', function( file ,response) {     console.log("上传完成: "+file.id+" "+response._raw); }); </pre>
点击上传后，要执行的动作	<pre> \$('#spulmgUploadBtn').click(function(){     console.log("开始上传");     if(spulmgUploader.getFiles().length&lt;=0){         \$.messager().alert('警告','没有需要上传的文件','warning');         return;     }     spulmgUploader.upload(); }); </pre>

后台代码（临时测试）


```
@RestController
public class FileUploadController {

    @RequestMapping(value = "fileUpload",method = RequestMethod.POST)
    public String fileUpload(@RequestParam("file") MultipartFile[] files){
        if(files.length!=0){
            System.out.println("multipartFile = " + files[0].getName()+"|"+files[0].getSize());
        }
        return
        "https://m.360buyimg.com/babel/jfs/t5137/20/1794970752/352145/d56e4e94/591417dcN4fe5ef33.jpg";
    }
}
```

实际业务代码见脑图

### 3.3 解决上传后的预览，通过伸缩行实现

利用 datagridview 。这是 easyui 的一个扩展空间。

DataGrid - DetailView						
	Item ID	Product ID	List Price	Unit Cost	Attribute	Status
+	EST-1	FI-SW-01	16.5	10	Large	P
+	EST-2	K9-DL-01	18.5	12	Spotted Adult Female	P
+	EST-3	RP-SN-01	18.5	12	Venomless	P
-	EST-4	RP-SN-01	18.5	12	Rattleless	P
<div>  Attribute: Rattleless Status: P </div>						
+	EST-5	RP-LI-02	18.5	12	Green Adult	P
+	EST-6	FI-DSH-01	58.5	12	Tailless	P

使用方式，把 datagridview.js 拷贝到，并且在表格中加入如下标红部分

```
$(function(){
    $('#tt').datagrid({
        title:'DataGrid - DetailView',
        width:500,
        height:250,
        remoteSort:false,
        singleSelect:true,
        nowrap:false,
```

```
fitColumns:true,
url: './datagrid_data.json',
columns:[
    {field:'itemid',title:'Item ID',width:80},
    {field:'productid',title:'Product ID',width:100,sortable:true},
    {field:'listprice',title:'List Price',width:80,align:'right',sortable:true},
    {field:'unitcost',title:'Unit Cost',width:80,align:'right',sortable:true},
    {field:'attr1',title:'Attribute',width:150,sortable:true},
    {field:'status',title:'Status',width:60,align:'center'}
],
view: detailview,
detailFormatter: function(rowIndex, rowData){
    return '<table><tr>' +
        '<td rowspan=2 style="border:0"></td>' +
        '<td style="border:0">' +
        '<p>Attribute: ' + rowData.attr1 + '</p>' +
        '<p>Status: ' + rowData.status + '</p>' +
        '</td>' +
        '</tr></table>';
    }
});
```

## 4、文件服务器

现在咱们实现了文件从客户端提交，并展示的功能。服务器端要做的就是接收文件流，保存起来，并且返回给客户端文件的访问地址。

传统的用 io 流保存到 web 服务器本地的方式，可以直接用当前 web 服务的路径+图片名称来访问。

但是类似于商品图片这种海量级文件，光靠 web 服务器的硬盘是无法满足的。

另外如果，web 服务器是集群的那么 A 服务器是没法访问 B 服务器的本地文件的。

所以需要把文件服务单独管理起来，成为文件服务器。

实现方式就是 nginx+FastDFS

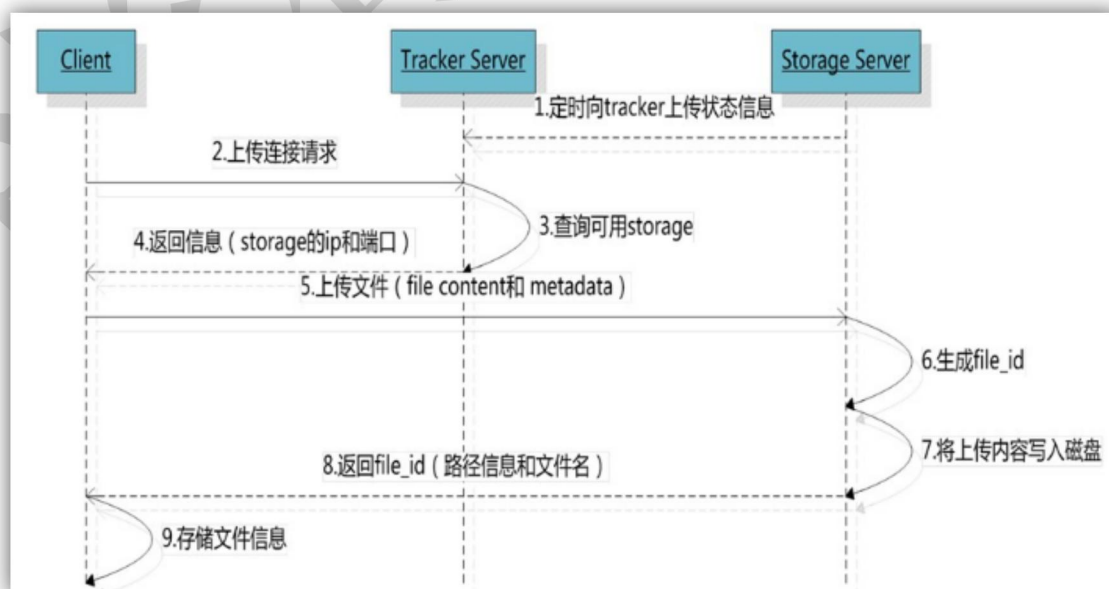
## 4.1 FastDFS 介绍

FastDFS 是一个由 C 语言实现的开源轻量级分布式文件系统，作者余庆 (happyfish100)，支持 Linux、FreeBSD、AIX 等 Unix 系统，解决了大数据存储和读写负载均衡等问题，适合存储 4KB~500MB 之间的小文件，如图片网站、短视频网站、文档、app 下载站等，UC、京东、支付宝、迅雷、酷狗等都有使用。



该软件作者是阿里巴巴大牛、chinaUnix 版主**余庆**个人独立开发的。

## 4.2 FastDFS 上传下载的流程



### 4.3 安装步骤参见《FastDFS 安装说明》

### 4.4 利用 Java 客户端调用 FastDFS

服务器安装完毕后，咱们通过 Java 调用 fastdfs

加载 Maven 依赖

fastdfs 没有在中心仓库中提供获取的依赖坐标。

只能自己通过源码方式编译，打好 jar 包，安装到本地仓库。

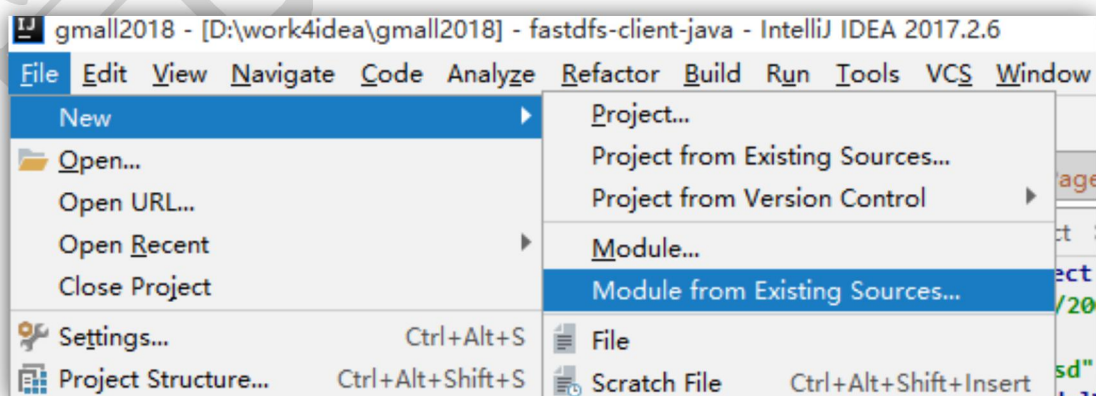
官方仓库地址：

<https://github.com/happyfish100/fastdfs-client-java>

```
zc@DESKTOP-S7LNAI5 MINGW64 /d/git_project
$ git clone https://github.com/happyfish100/fastdfs-client-java.git
Cloning into 'fastdfs-client-java'...
remote: Counting objects: 258, done.
remote: Total 258 (divineta 0g ob), reused 0 (dejects: 91t8a % (0253), p/a
c258), 92k-reused 258
Receiving objects: 100% (258/258), 125.79 KiB | 43.00 KiB/s, done.
Resolving deltas: 100% (109/109), done.
Checking connectivity... done.

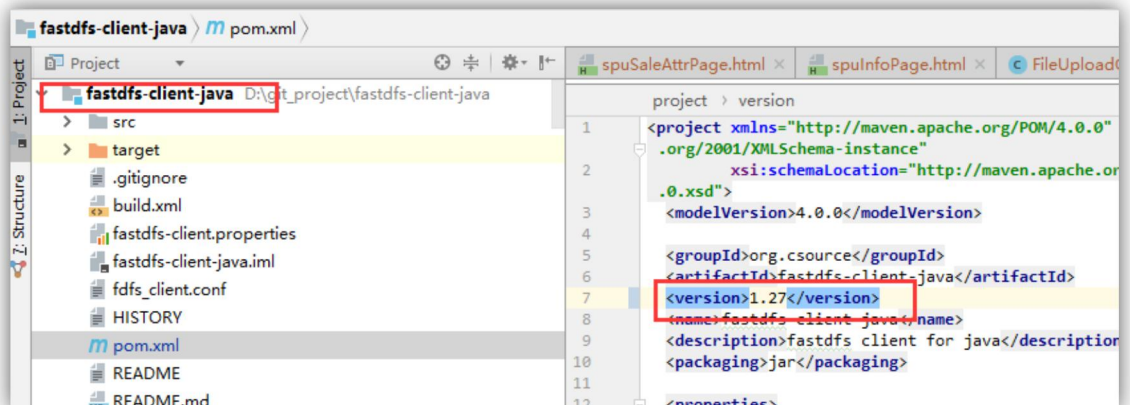
zc@DESKTOP-S7LNAI5 MINGW64 /d/git_project
$ |
```

直接用 idea 直接把这个源码作为模块导入工程

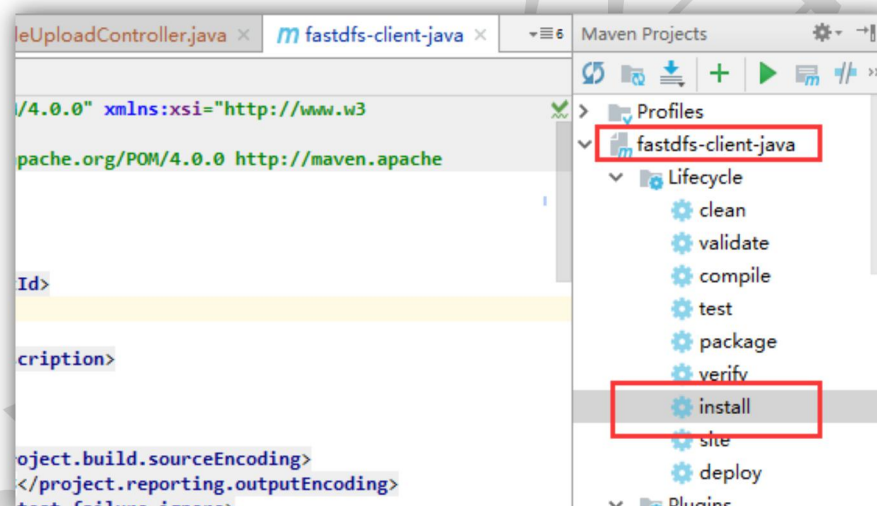


别的不用改，只把 pom.xml 中的版本改成 1.27。





然后右边 执行 install 就好了



安装好了，别的模块就可以直接使用这个坐标了。

```
<groupId>org.csource</groupId>
<artifactId>fastdfs-client-java</artifactId>
<version>1.27</version>
```

而这个 fastdfs-client-java 模块可以从 idea 中删除。

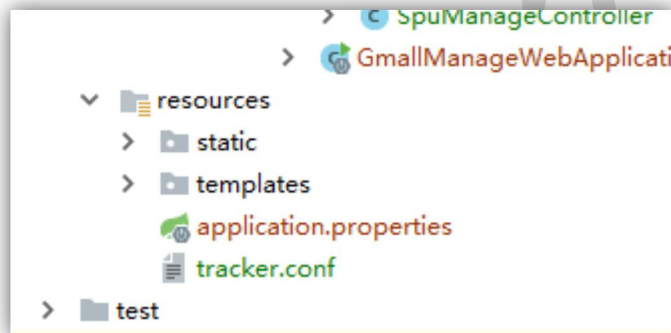
然后可以进行一下上传的测试

```
@Test
public void textFileUpload() throws IOException, MyException {
    String file = this.getClass().getResource("/tracker.conf").getFile();
    ClientGlobal.init(file);
    TrackerClient trackerClient=new TrackerClient();
```



```
TrackerServer trackerServer=trackerClient.getConnection();
StorageClient storageClient=new StorageClient(trackerServer,null);
String originalFilename="e://victor.jpg";
String[] upload_file = storageClient.upload_file(originalFilename, "jpg", null);
for (int i = 0; i < upload_file.length; i++) {
    String s = upload_file[i];
    System.out.println("s = " + s);
}
}
```

加入 tracker.conf 文件



```
tracker_server=file.server.com:22122

# 连接超时时间，针对 socket 套接字函数 connect，默认为 30 秒
connect_timeout=30000

# 网络通讯超时时间，默认是 60 秒
network_timeout=60000
```

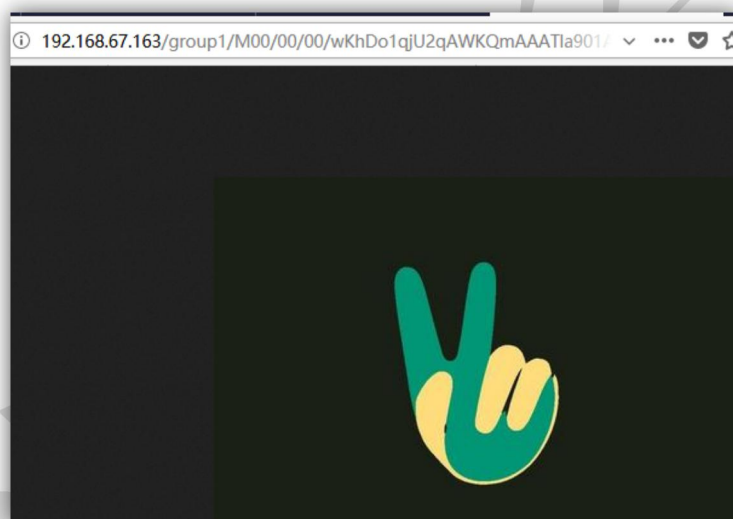
打印结果

```
"C:\Program Files\Java\jdk1.8.0_74\bin\java" ...  
s = group1  
s = M00/00/00/wKhDo1qjU2qAWKQmAAATla901AQ534.jpg  
  
Process finished with exit code 0
```

这个打印结果实际上就是我们访问的路径，加上服务器地址我们可以拼接成一个字符串

<http://192.168.67.163/group1/M00/00/00/wKhDo1qjU2qAWKQmAAATla901AQ534.jpg>

直接放到浏览器去访问



上传成功！

对接到业务模块中

在修改 FileUploadController 的方法

```
@Value("${fileServer.url}")  
String fileUrl;  
  
@RequestMapping(value = "fileUpload", method = RequestMethod.POST)  
public String fileUpload(@RequestParam("file") MultipartFile file) throws IOException, MyException {  
    String imgUrl=fileUrl;  
    if(file!=null){
```

```

System.out.println("multipartFile = " + file.getName()+"|"+file.getSize());

String configFile = this.getClass().getResource("/tracker.conf").getFile();
ClientGlobal.init(configFile);
TrackerClient trackerClient=new TrackerClient();
TrackerServer trackerServer=trackerClient.getConnection();
StorageClient storageClient=new StorageClient(trackerServer,null);
String filename= file.getOriginalFilename();
String extName = StringUtils.substringAfterLast(filename, ".");

String[] upload_file = storageClient.upload_file(file.getBytes(), extName, null);
imgUrl=fileUrl ;
for (int i = 0; i < upload_file.length; i++) {
    String path = upload_file[i];
    imgUrl+="/" + path;
}

}

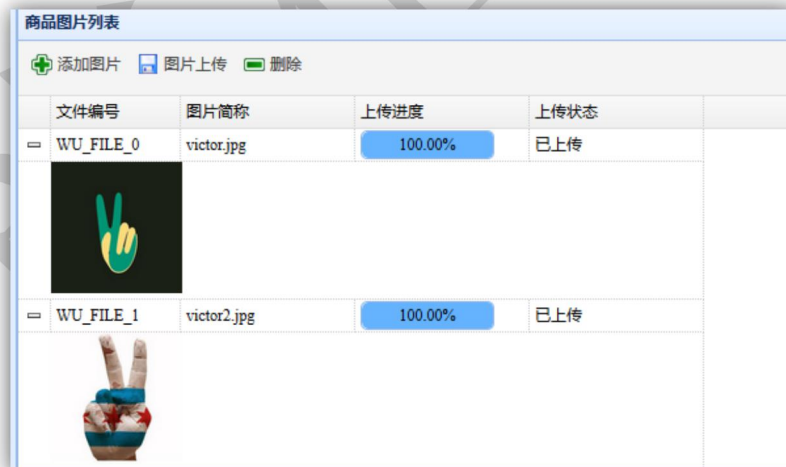
return imgUrl;
}

```

利用@Value 标签可以引用 application.properties 中的值

```
fileServer.url=http://file.server.com
```

测试结果:



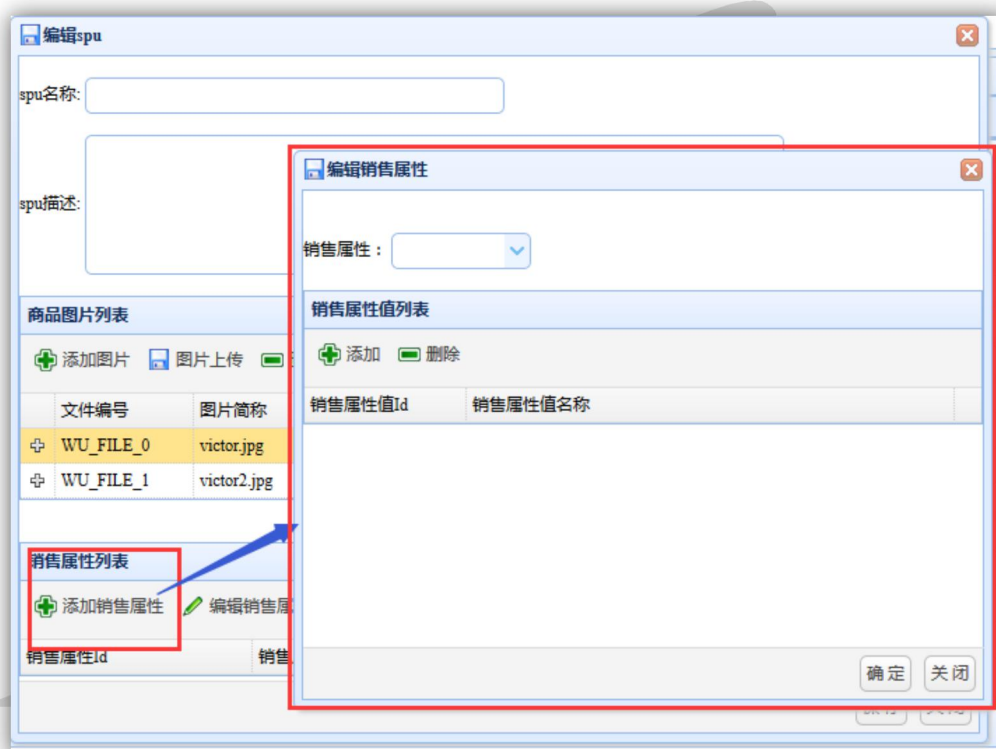
至此我们解决了文件上传的功能。

## 5 销售属性 ----两层多对多的关系

一个 spu 对应多个销售属性，但是每个销售属性又对应了多个属性值。

在 spu 的详情页面销售属性已经通过表格方式展现了一对多的关系，但是每一行代表的属性又有多个属性值。

只能针对每一行再弹出一个表格来增加多个属性。



但是跟之前弹出框保存有所不同的是，弹出框打开，软后录入数值，点击右下角确定的时候，不能够将数据直接保存到数据库中而是要把这个子弹出框的数据保存到销售属性表格控件中。只有当整个 spu 都录入完成，点击保存时才真的去写入数据库。

录入属性值信息确认后保存

```
function saveSpuSaleAttr() {

    var spuSaleAttrValueJson= $('#spuSaleAttrValueDg').datagrid('getData');
    var saleAttrId=$('#saleAttrSelect').combobox("getValue");
    var saleAttrName=$('#saleAttrSelect').combobox("getText");

    //
    var rowIndex = $("#spuSaleAttrDg").datagrid("getRowIndex",saleAttrId);
```

```
console.log("delete rowIndex:"+rowIndex);
if(rowIndex>=0){
    $("#spuSaleAttrDg").datagrid("deleteRow",rowIndex);
}

$("#spuSaleAttrDg").datagrid("appendRow",{saleAttrId:saleAttrId,saleAttrName:saleAttrName,spuSaleAttrValueJson:spuSaleAttrValueJson});

$("#spuSaleAttr_dlg").dialog("close");
}
```

其中 datagrid('getData') 是把整个 datagrid 的数据提取成一个 Json。

```
var spuSaleAttrValueJson= $('#spuSaleAttrValueDg').datagrid('getData');
```

这个 Json 的格式如下

```
{ total:2,
  row:[ {id:xxx,name:xxxx}, {id:xxx,name:xxxxx},..... ]
}
```

但是这是一个 Json 对象，不是一个 Json 字符串。

那么如果保存到了控件的一个暂存列中。

```
function initSpuSaleAttrListDatagrid(spuInfo){
    console.log("初始化销售属性表格:"+spuInfo);
    var spuSaleAttrDg=$('#spuSaleAttrDg').datagrid({url:""});
    spuSaleAttrDg=$('#spuSaleAttrDg').datagrid('loadData', { total: 0, rows: [] });

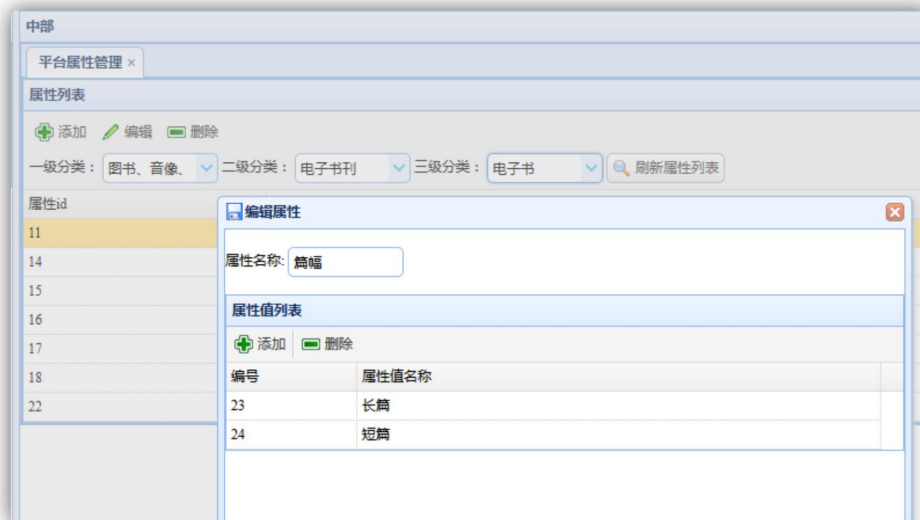
    spuSaleAttrDg.datagrid({
        idField: 'saleAttrId',
        columns:[
            { field:'id',title:'id',hidden:true },
            { field:'saleAttrId',title:'销售属性Id',width:'35%' },
            { field:'saleAttrName',title:'销售属性名称',width:'45%' },
            { field:'spuSaleAttrValueJson',title:'销售属性值暂存',hidden:true, width:'20%' }
        ]
    });

    //有初始数据说明是编辑状态，要加载数据
    if(spuInfo){
        spuSaleAttrDg.datagrid({url:"spuSaleAttrList?spuId="+spuInfo.id});
    }
}
</script>
```

那么当以后，想要编辑该属性时候，利用 loadData 把原数据加载回来就可以了。

```
console.log("spuSaleAttr:"+JSON.stringify(spuSaleAttr.saleAttrValue));
if(spuSaleAttr.&&spuSaleAttr.spuSaleAttrValueJson&&spuSaleAttr.spuSaleAttrValueJson!=""){
    console.log("加载暂存");
}
```

```
spuSaleAttrValueDg.datagrid("loadData",spuSaleAttr.spuSaleAttrValueJson);  
}
```



## 6 页面的整体保存

详见脑图中的代码