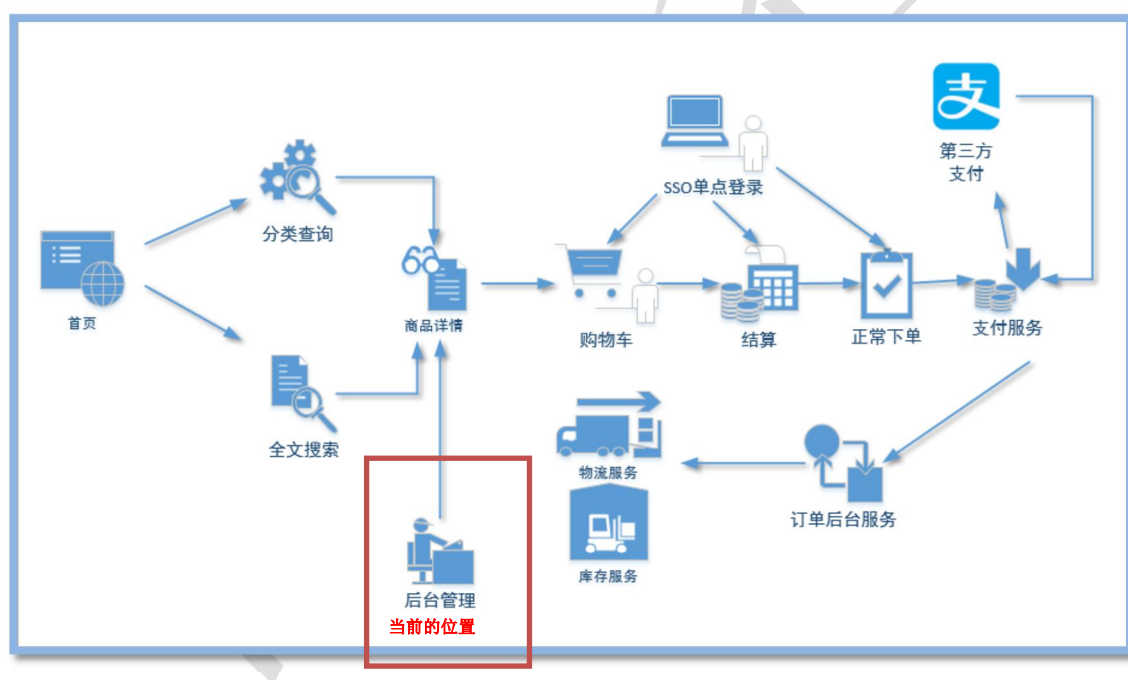


# 谷粒商城

版本: V 1.0  
www.atguigu.com

## 第一章 电商的业务简介

### 1 整体业务简介



首页	静态页面，包含了商品分类，搜索栏，商品广告位。
全文搜索	通过搜索栏填入的关键字进行搜索，并列表展示
分类查询	根据首页的商品类目进行查询

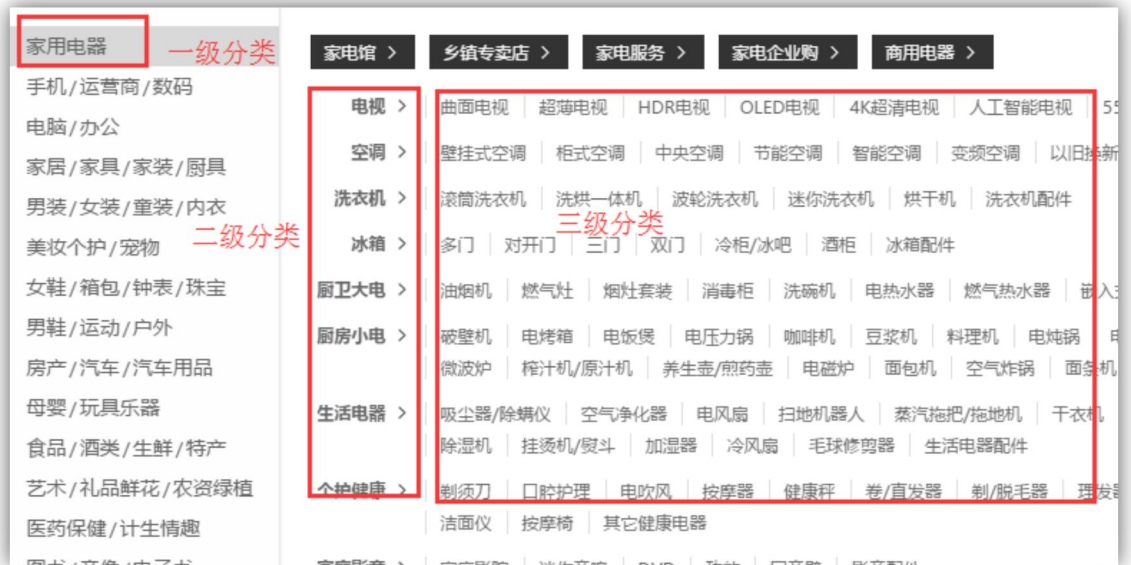
商品详情	商品的详细信息展示
购物车	将有购买意向的商品临时存放的地方
单点登录	用户统一登录的管理
结算	将购物车中勾选的商品初始化成要填写的订单
下单	填好的订单提交
支付服务	下单后，用户点击支付，负责对接第三方支付系统。
订单服务	负责确认订单是否付款成功，并对接仓储物流系统。
仓储物流	独立的管理系统，负责商品的库存。（只对接接口不单独讲解）
后台管理	主要维护类目、商品、库存单元、广告位等信息。

## 2 商品管理

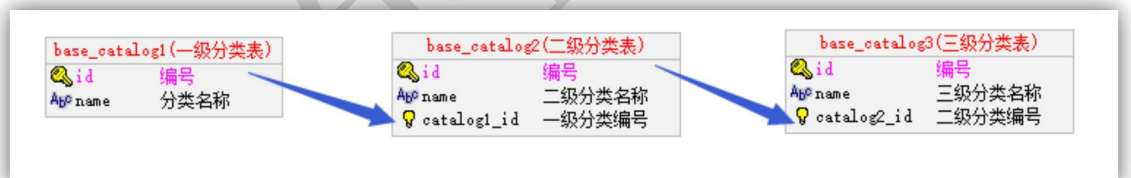
### 2.1 基本信息—分类

一般情况可以分为两级或者三级。咱们的项目一共分为三级，即一级分类、二级分类、三级分类。

比如：家用电器是一级分类，电视是二级分类，那么超薄电视就是三级分类。



## 数据库结构



## 2.2 基本信息—平台属性

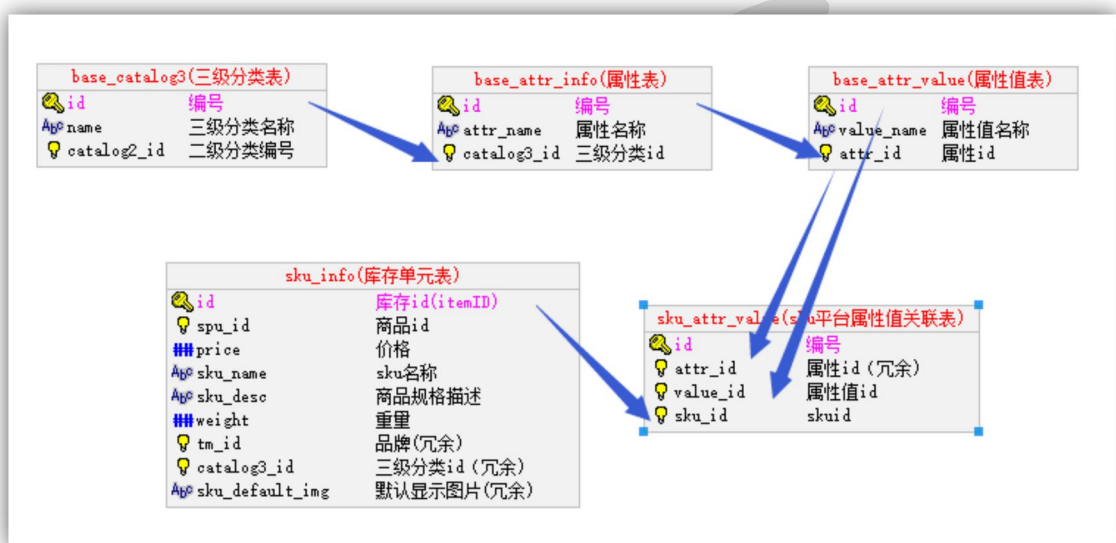
平台属性和平台属性值

屏幕尺寸:	70英寸及以上	65英寸	58-60英寸	55英寸	49-50英寸	45-48英寸	42-43英寸	39-40英寸	32英寸及以下
分辨率:	超高清	全高清	高清						
观看距离:	3.5米以上	3-3.5米	2.5-3米	2-2.5米	2米以内				

平台属性和平台属性值主要用于商品的检索，每个三级分类对应的属性都不同。



而每个商品对应的每种属性都有对应的属性值。



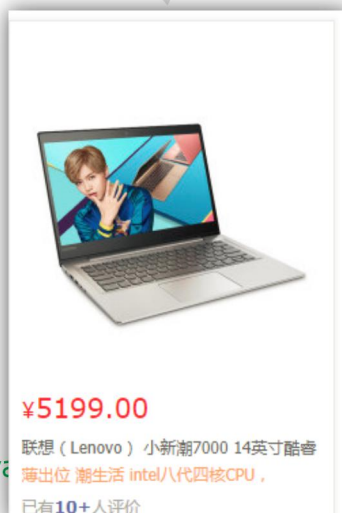
比如

电脑整机的一级分类下，有笔记本、游戏本、台式机、一体机的二级分类。

笔记本这个二级分类又包含了处理器、屏幕尺寸、内存容量、硬盘容量、显卡类别这些属性。

那么针对联想某个型号的笔记本，它作为笔

记本这种分类，每个分类属性都有对应的值，cpu(属性)是 i7(属性值)的，内存(属性)是 8G(属性值)的，屏幕尺寸(属性)是 14 寸(属性值)的。



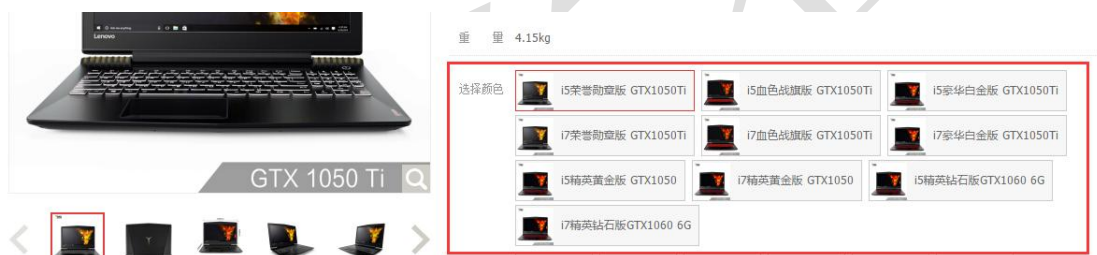
## 2.3 基本信息—spu 与 sku

SKU=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**。

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。

首先通过检索搜索出来的商品列表中，每个商品都是一个 sku。每个 sku 都有自己独立的库存数。也就是说每一个商品详情展示都是一个 sku。

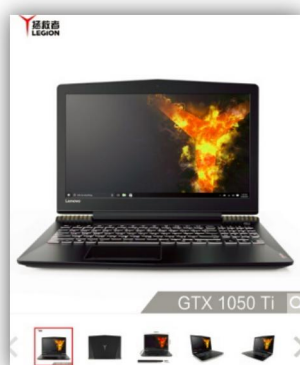
那 spu 又是干什么的呢？



如上图，一般的电商系统你点击进去以后，都能看到这个商品关联了其他好几个类似的商品，而且这些商品很多的信息都是共用的，比如商品图片，海报、销售属性等。

那么系统是靠什么把这些 sku 识别为一组呢，那是这些 sku 都有一个公用的 spu 信息。而它们公共的信息，都放在 spu 信息下。

所以，sku 与 spu 的结构如下：



图中有两个图片信息表，其中 spu\_image 表示整个 spu 相关下的所有图片信息，而 sku\_image 表示这个 spu 下的某个 sku 使用的图片。sku\_image 中的图片是从 spu\_image 中选取的。

但是由于一个 spu 下的所有 sku 的海报都是一样，所以只存一份 spu\_poster 就可以了。

## 2.4 商品信息—销售属性与平台属性

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。



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。

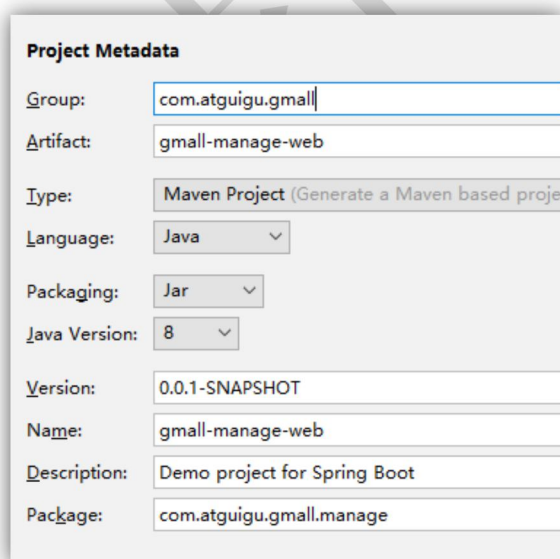
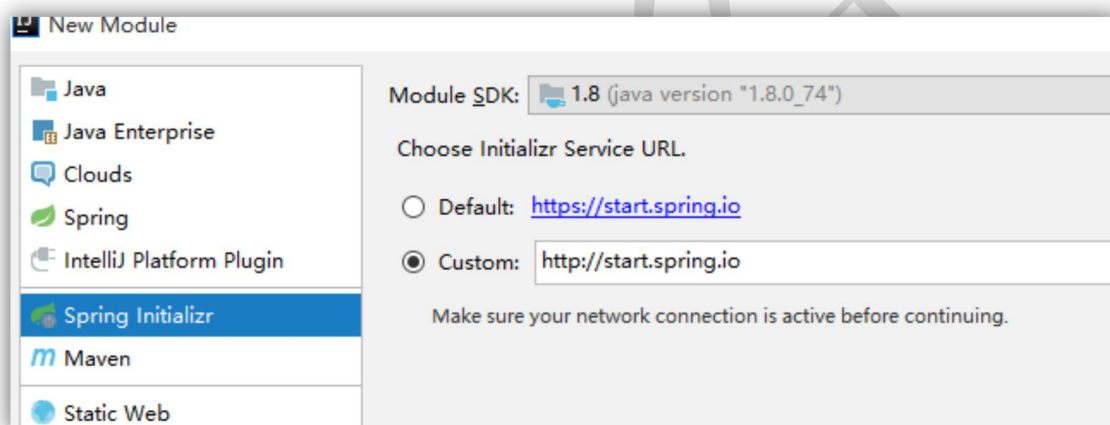
一般每种商品的销售属性不会太多，大约 1-4 种。整个平台的属性种类也不会太多，大概 10 种以内。比如：颜色、尺寸、版本、套装等等。



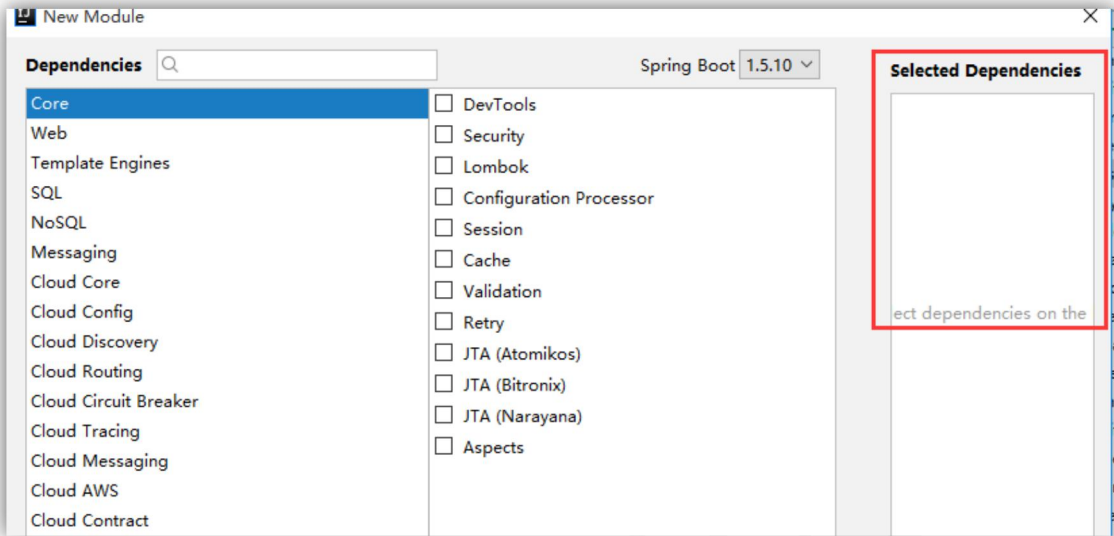
## 第二章 后台管理模块开发

### 1 后台的 Web 模块搭建

#### 1.1 建 module







## pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.atguigu.gmall</groupId>
  <artifactId>gmall-manage-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>gmall-manage-web</name>
  <description>Demo project for Spring Boot</description>

  <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>
</project>
```



```

</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

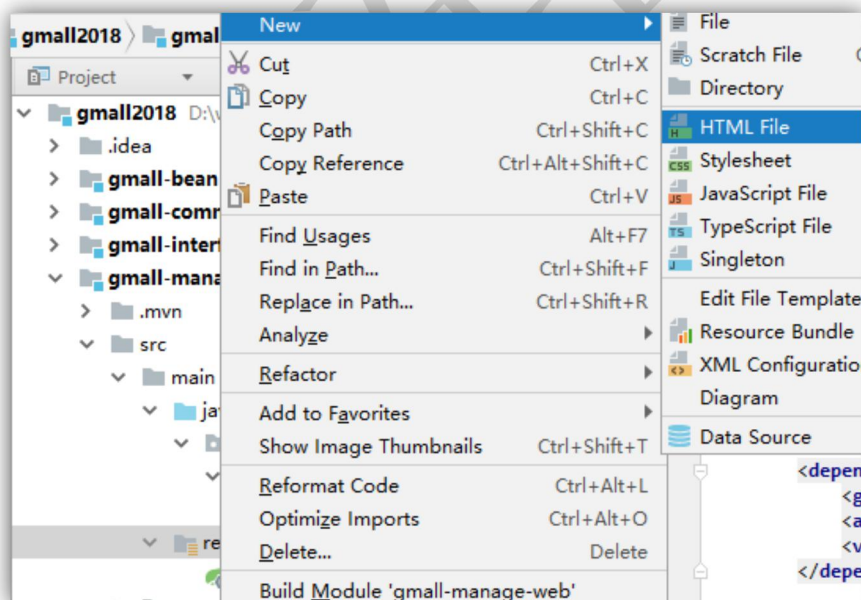
</project>

```

## 1.2 页面开发

如何在 web 模块中加入网页：

在 resources 中的 templates 文件夹，加入一个 indexhtml 的页面



index.html 代码

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
hello world !
</body>
</html>
```

这时如果你启动服务，然后用浏览器访问其实是访问不到的。index.html 放到 templates 只是一个模板，必须要经过 controller 层渲染，才能被访问。

增加 ManageController

```
@Controller
public class ManageController {
    @RequestMapping(value = "index")
    public String index(){
        return "index";
    }
}
```

同时 在 web-util 的 pom.xml 文件中增加

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

如果后台出现

```
threw exception [Request processing failed; nested exception is org.thymeleaf.exception
3] with root cause

org.xml.sax.SAXParseException: 元素类型 "meta" 必须由匹配的结束标记 "</meta>" 终止。
    at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.createSAXParseException
    at com.sun.org.apache.xerces.internal.util.ErrorHandlerWrapper.fatalError(ErrorHandl
    at com.sun.org.apache.xerces.internal.impl.XMLErrorReporter.reportError(XMLErrorRepo
```

这是因为 springboot 默认使用 thymeleaf 渲染，而这种模板语言对 html 的标签约束非常严格，所有标签必须有开有闭，比如 <br><br/> 或者<br/> 是可以的，而<br> 是会报错的。

解决页面松校验的方式

application.properties 中增加

```
spring.thymeleaf.mode=LEGACYHTML5
```

在 gmall-parent 模块中的 pom.xml 增加依赖管理

```
<nekohtml.version>1.9.20</nekohtml.version>
<xml-apis.version>1.4.01</xml-apis.version>
<batik-ext.version>1.9.1</batik-ext.version>

<dependency>
  <groupId>net.sourceforge.nekohtml</groupId>
  <artifactId>nekohtml</artifactId>
  <version>${nekohtml.version}</version>
</dependency>

<dependency>
  <groupId>xml-apis</groupId>
  <artifactId>xml-apis</artifactId>
  <version>${xml-apis.version}</version>
</dependency>

<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-ext</artifactId>
  <version>${batik-ext.version}</version>
</dependency>
```

在 web-util 模块中的 pom.xml 增加

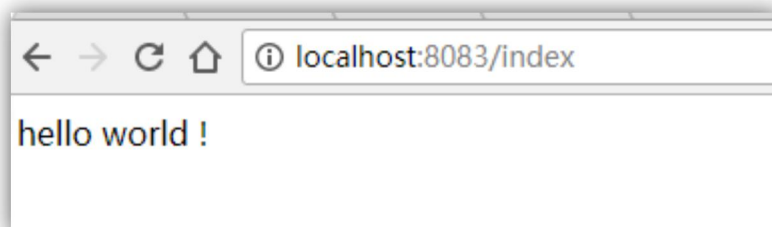
```
<dependency>
  <groupId>net.sourceforge.nekohtml</groupId>
  <artifactId>nekohtml</artifactId>
</dependency>
<dependency>
  <groupId>xml-apis</groupId>
  <artifactId>xml-apis</artifactId>
</dependency>
<dependency>
  <groupId>org.apache.xmlgraphics</groupId>
  <artifactId>batik-ext</artifactId>
</dependency>
```

请在 application.properties 中增加

```
spring.thymeleaf.cache=false
```

这个是关闭了 springboot 的页面缓存，如果不关闭会影响开发调试过程中的热部署，如果系统上线可以再把这个缓存打开。

然后测试页面



## 2 搭建后台页面

### 2.1 前置学习：EasyUI

easyui 是一个前端框架，基于 jquery。它提供一些相应的 js，css 文件。为 web 开发提供了方便，并且有丰富的页面(美观)。通常 easyUI 用户后台的管理系统的开发模板。通常 easyui 都用来跟 bootstrap 作比较。

共同点：

都有自己的一套完整的 ui 组件，都对各种组件进行了包装美好。但是 bootstrap 更加精致美观，动画特效，阴影，质感更好。

easyui 虽然不及 bootstrap 美观。但是组件与数据的对接更加方便，比如 easyui 中可以直接把从后台获得的一个 json 转化成一个表格直接展示给用户。而 bootstrap 的表格控件就是炫，对数据没有什么支持。

所以 easyui 对于管理系统中大量的表单、表格、插删改查的页面，开发起来更加快捷。而这些页面往往是后台的专业人员（客服、商家、运营）来使用，并不太追求界面多炫。

而 bootstrap 更适合给网上的前台用户使用，增加页面的使用手感，增强用户的体验

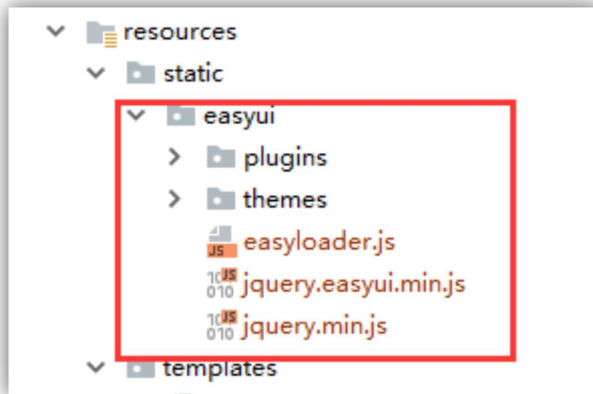
### 导入资源包

版本:easyui 1.5.4.1

12

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可访问百度：尚硅谷官网

把资源包解压后，要把如下图的这个几个文件包和文件考入到 idea 中的 small-manage-web 模块。



在页面中引用资源

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>主界面</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">
</head>
```

页面布局

```
<body class="easyui-layout">
  <div data-options="region:'north',title:'头部',split:true" style="height:100px;">
    <h1> 顶部 </h1>
  </div>
  <div data-options="region:'south',title:'底部',split:true" style="height:100px;">
    <h1>底部 </h1>
  </div>
  <div data-options="region:'west',title:'左部',split:true" style="width:200px;">
    <h1>左部 </h1>
  </div>
  <div data-options="region:'center',title:'中部'" >
    <h1>中部 </h1>
  </div>
</body>
```

## 2.2 easyUI 的布局

main.html

主界面分为三部分：top, left, right

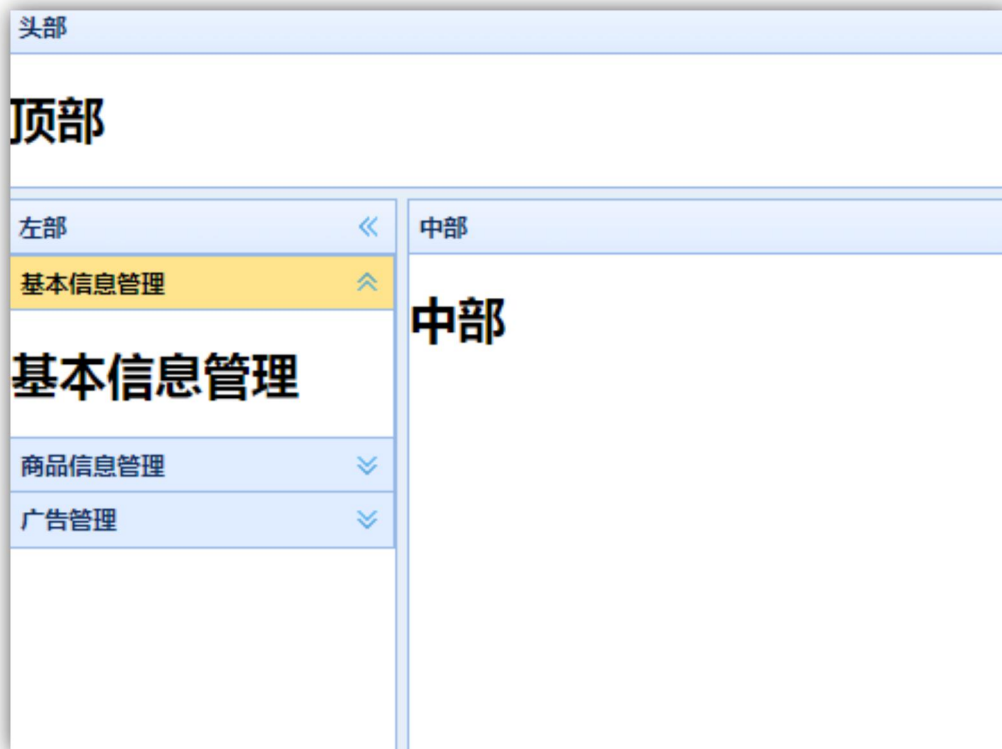
Left 部分有个树形菜单

right 部分显示左侧菜单点击后的详细页面。

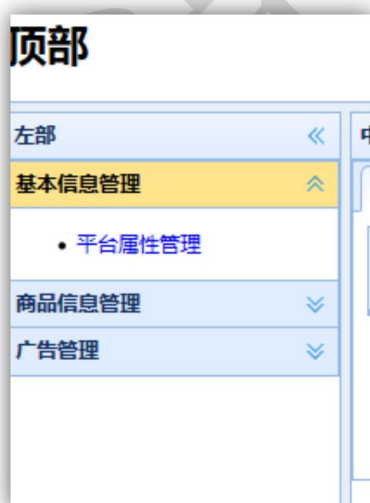


增加手风琴菜单

```
<div data-options="region:'west',title:'左部',split:true" style="width:200px;">
  <div class="easyui-accordion" style="overflow:auto;">
    <div title="基本信息管理">
      <h1>基本信息管理</h1>
    </div>
    <div title="商品信息管理">
      <h1>商品信息管理</h1>
    </div>
    <div title="广告管理">
      <h1>广告管理</h1>
    </div>
  </div>
</div>
```



左侧增加菜单



```
<div title="基本信息管理">  
  <ul class="nav">
```



```
        <li><a href="#" style="text-decoration:none;">平台属性管理</a></li>
    </ul>
</div>
```

中部增加 tab 组件

```
<div data-options="region:'center',title:'中部'" >
    <div id="tt" class="easyui-tabs" style="width:100%;height:100%;" >

    </div>
</div>
```

点击菜单增加右边的标签

```
<div title="基本信息管理">
    <ul class="nav">
        <li><a href="javascript:add_tab('平台属性管理','attrListPage')"
style="text-decoration:none;">平台属性管理</a></li>
    </ul>
</div>
```

```
<script type="text/javascript">
    function add_tab(title,url){
        if($("#tt").tabs('exists',title)){
            $("#tt").tabs('select',title);
        }else{
            $("#tt").tabs('add',{
                title:title,
                href:url,
                closable:true
            });
        }
    }
</script>
```

左侧的属性列表页面

增加 AttrManageController

```
@Controller
public class AttrManageController {
    @RequestMapping("attrListPage")
    public String getAttrListPage(){
        return "attrListPage";
    }
}
```

```

    }
}

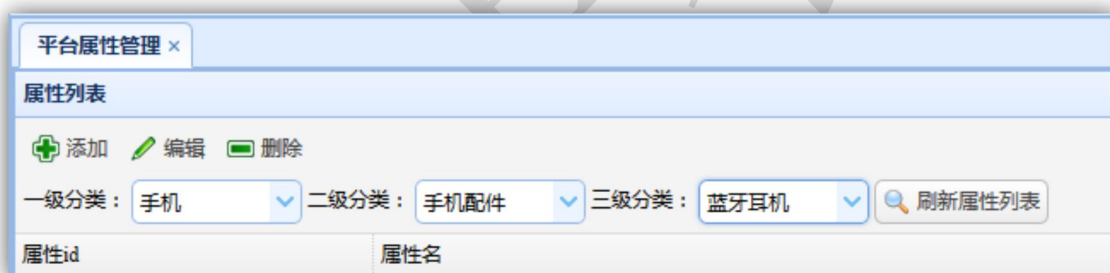
```

## 3 属性管理功能

### 3.1 分类信息及属性的查询

#### 3.1.1 页面

要达到的效果



包含：两个下拉菜单(分类选择)

一个属性列表

一组按钮

属性列表：(属性列要与 bean 的属性名称相同)

```

<table id="dg" class="easyui-datagrid" title="属性列表"
    data-options="singleSelect:true ,method:'get',toolbar:'#tb'">
    <thead>
    <tr>
        <th data-options="field:'id' width='20%'">属性 id </th>
        <th data-options="field:'attrName' width='80%'">属性名</th>
    </tr>
    </thead>
</table>

```

其中 toolbar: '#tb' 是引用了另一个工具栏，这个工具栏要额外定义如下：

工具栏

```
<div id="tb" style="padding:5px;height:auto">
  <div style="margin-bottom:5px">
    <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain="true">添加</a>
    <a href="#" class="easyui-linkbutton" iconCls="icon-edit" plain="true">编辑</a>
    <a href="#" class="easyui-linkbutton" iconCls="icon-remove" plain="true">删除</a>
  </div>
  <div>
    一级分类:
    <select id="ctg1ForAttrList" class="easyui-combobox" style="width:100px" ></select>
    二级分类:
    <select name="ctg2ForAttrList" id="ctg2ForAttrList" class="easyui-combobox"
style="width:100px" ></select>
    三级分类:
    <select name="ctg3ForAttrList" id="ctg3ForAttrList" class="easyui-combobox"
style="width:100px" ></select>
  </div>
</div>
```

有了页面结构，还要加上交互：

- ※ 一级菜单的变化，会动态改变二级菜单的内容。
- ※ 二级菜单的变化，会动态改变三级菜单的内容。
- ※ 三级菜单的变化，会动态改变列表显示的属性。

还有一个页面初始化的时候，就要执行的方法：

- ※ 一级菜单内容的加载

这种动态效果，一般情况下要写 js 实现。但是利用 easyui 可以直接把这种动态变化绑定到元素上。

给下拉菜单绑定一个 data-options

```
data-options="valueField:'id',textField:'name',url:'/getCatalog1',
  onSelect:function(rec){
    $('#ctg2ForAttrList').combobox('clear');
    $('#ctg2ForAttrList').combobox('reload','getCatalog2?catalog1Id='+rec.id);
  }"
```

data-option 中可以设的内容

18

更多 Java - 大数据 - 前端 - python 人工智能资料下载，可访问百度：尚硅谷官网

url	通过此链接利用 ajax 获得 json 数据
valueField	下拉选项的值要绑定 json 中的元素，要跟传入数据的字段对应
textField	下拉选项的显示内容要绑定 json 中的元素
onSelect	选中某个下拉选项会触发的方法。
.combobox('clear')	清空下拉菜单
.combobox('reload','xxx');	让某个下拉菜单按照路径加载数据。

加载 Datagrid 表格的值

```
$('#dg').datagrid({url:'getAttrList?catalog3Id='+rec.id})
```

最终代码如下：

```
<div>
    一级分类:
    <select id="ctg1ForAttrList" class="easyui-combobox" style="width:100px"
data-options="valueField:'id',textField:'name',url:'getCatalog1',
onSelect:function(rec){
    $('#ctg2ForAttrList').combobox('clear');
    $('#ctg3ForAttrList').combobox('clear');
    $('#ctg2ForAttrList').combobox('reload','getCatalog2?catalog1Id='+rec.id);
    }" ></select>
    二级分类:
    <select name="ctg2ForAttrList" id="ctg2ForAttrList" class="easyui-combobox"
data-options="valueField:'id',textField:'name',
onSelect:function(rec){
    $('#ctg3ForAttrList').combobox('clear');
    $('#ctg3ForAttrList').combobox('reload','getCatalog3?catalog2Id='+rec.id);
    }" style="width:100px" ></select>
    三级分类:
    <select name="ctg3ForAttrList" id="ctg3ForAttrList" class="easyui-combobox"
data-options="valueField:'id',textField:'name'" style="width:100px" ></select>
    <a href="#" class="easyui-linkbutton" iconCls="icon-search"
onclick="javascript:reloadAttrList()">刷新属性列表</a>
</div>
```

属性列表的刷新按钮增加一个 js 方法

```
<script language="javascript">
/**/
function reloadAttrList(){
    var ctg3val=$('#ctg3ForAttrList').combobox('getValue');</pre>
</div>
<div data-bbox="146 893 169 907" data-label="Page-Footer">
19
</div>
<div data-bbox="144 912 792 930" data-label="Page-Footer">
<p>更多 Java - 大数据 - 前端 - python 人工智能资料下载，可访问百度：尚硅谷官网</p>
</div>
```

```
        $('#dg').datagrid({url:'getAttrList?catalog3Id='+ctg3val});  
    }  
    /*]]>*/  
</script>
```

其中红色部分，是向后台调用的 get 请求路径。

那么相应的后台要准备四个请求

getCatalog1	获得一级分类
getCatalog2	获得二级分类
getCatalog3	获得三级分类
getAttrList	获得属性列表

### 3.1.2 后台代码

首先是 bean, 以下代码由于节省篇幅没有生产 getter, setter 方法，请自行用 idea 生成。

BaseCatalog1

```
public class BaseCatalog1 implements Serializable {  
    @Id  
    @Column  
    private String id;  
    @Column  
    private String name;  
}
```

BaseCatalog2

```
public class BaseCatalog2 implements Serializable {  
    @Id  
    @Column  
    private String id;  
    @Column  
    private String name;  
    @Column  
    private String catalog1Id;  
}
```

BaseCatalog3

```
public class BaseCatalog3 implements Serializable {  
    @Id  
    @Column  
    private String id;  
    @Column  
    private String name;  
    @Column  
    private String catalog2Id;  
}
```

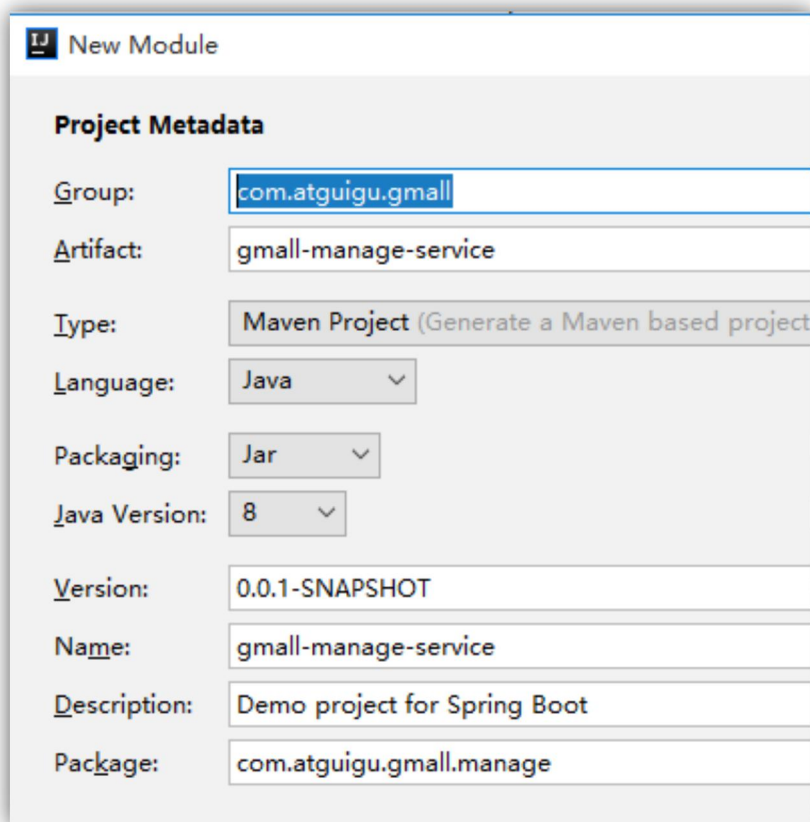
BaseAttrValue

```
public class BaseAttrValue implements Serializable {  
    @Id  
    @Column  
    private String id;  
    @Column  
    private String valueName;  
    @Column  
    private String attrId;  
    @Column  
    private String isEnabled;  
}
```

BaseAttrInfo

```
public class BaseAttrInfo implements Serializable {  
    @Id  
    @Column  
    private String id;  
    @Column  
    private String attrName;  
    @Column  
    private String catalog3Id;  
    @Column  
    private String isEnabled;  
}
```

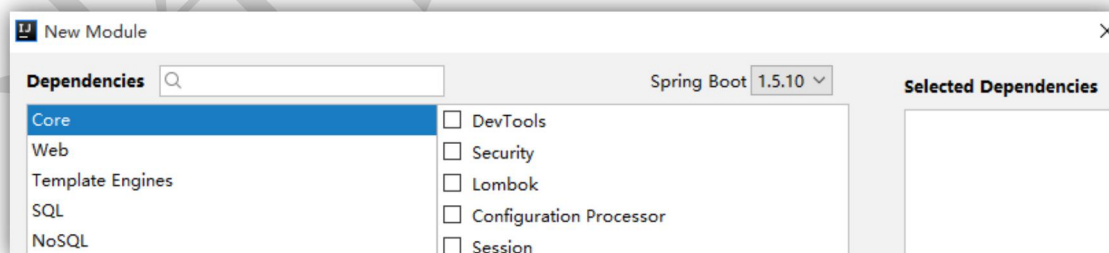
创建 manage-service 模块



The 'New Module' dialog box in IntelliJ IDEA. The 'Project Metadata' section contains the following fields:

- Group: com.atguigu.gmall
- Artifact: gmall-manage-service
- Type: Maven Project (Generate a Maven based project)
- Language: Java
- Packaging: Jar
- Java Version: 8
- Version: 0.0.1-SNAPSHOT
- Name: gmall-manage-service
- Description: Demo project for Spring Boot
- Package: com.atguigu.gmall.manage

不用添加任何依赖



The 'New Module' dialog box, Dependencies tab. The 'Spring Boot' version is set to 1.5.10. The 'Selected Dependencies' list is empty.

Dependencies	Selected Dependencies
<input checked="" type="checkbox"/> Core	
<input type="checkbox"/> Web	
<input type="checkbox"/> Template Engines	
<input type="checkbox"/> SQL	
<input type="checkbox"/> NoSQL	
<input type="checkbox"/> DevTools	
<input type="checkbox"/> Security	
<input type="checkbox"/> Lombok	
<input type="checkbox"/> Configuration Processor	
<input type="checkbox"/> Session	

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.atguigu.gmall</groupId>
  <artifactId>gmall-manage-service</artifactId>
```



```
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>gmall-manage-service</name>

<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-service-util</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

在 manage-service 中 创建 Mapper

BaseCatalog1Mapper

```
public interface BaseCatalog1Mapper extends Mapper<BaseCatalog1> {
}
```

BaseCatalog2Mapper

```
public interface BaseCatalog2Mapper extends Mapper<BaseCatalog2> {
}
```

BaseCatalog3Mapper

```
public interface BaseCatalog3Mapper extends Mapper<BaseCatalog3> {
}
```

BaseAttrInfoMapper

```
public interface BaseAttrInfoMapper extends Mapper<BaseAttrInfo> {  
}
```

BaseAttrValueMapper

```
public interface BaseAttrValueMapper extends Mapper<BaseAttrValue> {  
}
```

在 interface 中 增加 service 接口

```
public interface ManageService {  
  
    public List<BaseCatalog1> getCatalog1();  
  
    public List<BaseCatalog2> getCatalog2(String catalog1Id);  
  
    public List<BaseCatalog3> getCatalog3(String catalog2Id);  
  
    public List<BaseAttrInfo> getAttrList(String catalog3Id);  
  
}
```

增加实现类

```
@com.alibaba.dubbo.config.annotation.Service  
public class ManageServiceImpl implements ManageService {  
  
    @Autowired  
    BaseAttrInfoMapper baseAttrInfoMapper;  
  
    @Autowired  
    BaseAttrValueMapper baseAttrValueMapper;  
  
    @Autowired  
    BaseCatalog1Mapper baseCatalog1Mapper;  
  
    @Autowired  
    BaseCatalog2Mapper baseCatalog2Mapper;  
  
    @Autowired  
    BaseCatalog3Mapper baseCatalog3Mapper;  
  
    @Override  
    public List<BaseCatalog1> getCatalog1() {  
        List<BaseCatalog1> baseCatalog1List = baseCatalog1Mapper.selectAll();  
        return baseCatalog1List;  
    }  
}
```

```
@Override
public List<BaseCatalog2> getCatalog2(String catalog1Id) {
    BaseCatalog2 baseCatalog2=new BaseCatalog2();
    baseCatalog2.setCatalog1Id(catalog1Id);

    List<BaseCatalog2> baseCatalog2List = baseCatalog2Mapper.select(baseCatalog2);
    return baseCatalog2List;
}

@Override
public List<BaseCatalog3> getCatalog3(String catalog2Id) {
    BaseCatalog3 baseCatalog3=new BaseCatalog3();
    baseCatalog3.setCatalog2Id(catalog2Id);

    List<BaseCatalog3> baseCatalog3List = baseCatalog3Mapper.select(baseCatalog3);
    return baseCatalog3List;
}

@Override
public List<BaseAttrInfo> getAttrList(String catalog3_id) {
    BaseAttrInfo baseAttrInfo = new BaseAttrInfo();
    baseAttrInfo.setCatalog3Id(catalog3_id);

    List<BaseAttrInfo> baseAttrInfoList = baseAttrInfoMapper.select(baseAttrInfo);
    return baseAttrInfoList;
}
}
```

manage-web 的 controller 中 AttrManageController 中增加方法

```
@Controller
public class AttrManageController {

    @Reference
    ManageService manageService;

    @RequestMapping("attrListPage")
    public String getAttrListPage(){
        return "attrListPage";
    }

    /**
     * 获得一级分类
     * @return
     */
    @RequestMapping("getCatalog1")
    @ResponseBody
    public List<BaseCatalog1> getCatalog1(){
```

```
List<BaseCatalog1> catalog1List = manageService.getCatalog1();
return catalog1List;
}

/**
 * 获得二级分类
 * @param map
 * @return
 */
@RequestMapping("getCatalog2")
@ResponseBody
public List<BaseCatalog2> getCatalog2(@RequestParam Map<String,String> map){
    String catalog1Id = map.get("catalog1Id");
    List<BaseCatalog2> catalog2List = manageService.getCatalog2(catalog1Id);
    return catalog2List;
}

/**
 * 获得三级分类
 * @param map
 * @return
 */
@RequestMapping("getCatalog3")
@ResponseBody
public List<BaseCatalog3> getCatalog3(@RequestParam Map<String,String> map){
    String catalog2Id = map.get("catalog2Id");
    List<BaseCatalog3> catalog3List = manageService.getCatalog3(catalog2Id);
    return catalog3List;
}

/**
 * 获得属性列表
 * @param map
 * @return
 */
@RequestMapping("getAttrList")
@ResponseBody
public List<BaseAttrInfo> getAttrList(@RequestParam Map<String,String> map){
    String catalog3Id = map.get("catalog3Id");
    List<BaseAttrInfo> attrList = manageService.getAttrList(catalog3Id);
    return attrList;
}
}
```

配置 manage-web 的 application.properties

```
server.port=8083

spring.thymeleaf.cache=false

spring.thymeleaf.mode=LEGACYHTML5
```

```
spring.dubbo.application.name=manage-web
spring.dubbo.registry.protocol=zookeeper
spring.dubbo.registry.address=192.168.67.159:2181
spring.dubbo.base-package=com.atguigu.gmall
spring.dubbo.protocol.name=dubbo
spring.dubbo.consumer.timeout=10000
spring.dubbo.consumer.check=false
```

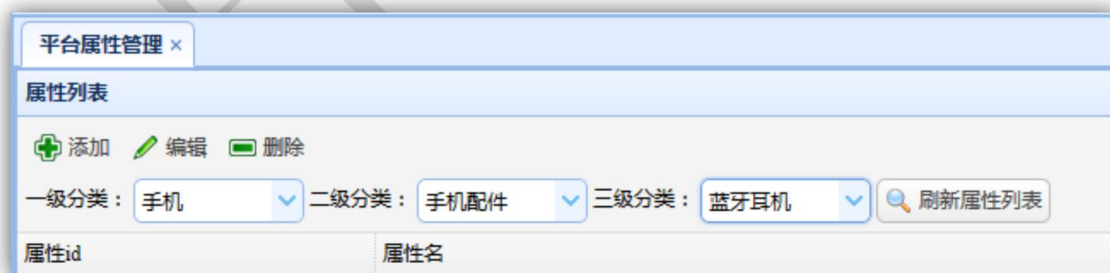
配置 manage-service 的 application.properties

```
server.port=8073

logging.level.root=debug

spring.dubbo.application.name=manage-service
spring.dubbo.registry.protocol=zookeeper
spring.dubbo.registry.address=192.168.67.159:2181
spring.dubbo.base-package=com.atguigu.gmall
spring.dubbo.protocol.name=dubbo
spring.datasource.url=jdbc:mysql://59.110.141.236:3306/gmall?characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=123123
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

启动服务后



## 3.2 属性的添加、编辑



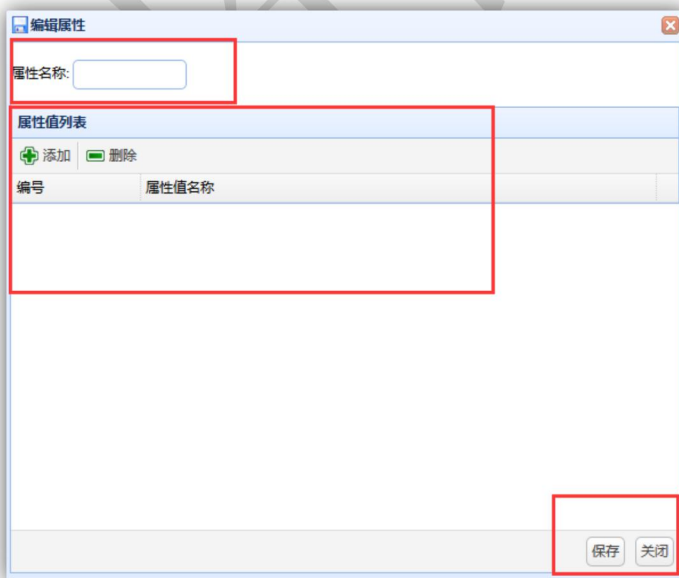
### 3.2.1 页面

点击增加后弹出编辑框

```
<div style="margin-bottom:5px">
  <a href="#" class="easyui-linkbutton" iconCls="icon-add" plain="true"
onclick="addAttrInfo()">添加</a>
  <a href="#" class="easyui-linkbutton" iconCls="icon-edit" plain="true"
onclick="editAttrInfo()">编辑</a>
  <a href="#" class="easyui-linkbutton" iconCls="icon-remove" plain="true">删除</a>
</div>
```

制作弹出框

弹出框三部分组成：属性名称的文本域，属性值列表，保存按钮。



弹出框 html

```
<div id="dlg" class="easyui-dialog" title="编辑属性" style="width:600px;height:500px;"
```

```
data-options="iconCls:'icon-save',resizable:true,modal:true" buttons="#bb" >
<form id="attrForm">
  <br/>
  <label>属性名称:</label>
  <input id="attrName" name="attrName" class="easyui-textbox" data-options=""
style="width:100px"/>
  <input id="attrId" name="attrId" type="hidden" />
  <br/><br/>
  <table id="dg_av" class="easyui-datagrid" title="属性值列表"></table>
</form>
</div>

<div id="bb">
  <a href="#" class="easyui-linkbutton" onclick="saveAttr()">保存</a>
  <a href="#" class="easyui-linkbutton">关闭</a>
</div>
```

其中有一个 hidden 的隐藏域用来，存放当前属性的 id。

这里的 datagrid 没有初始化按钮和列头，所以首先要编写 datagrid 的初始化方法。

```
function initAttrValueDatagrid(){
  $('#dg_av').datagrid('loadData',{ total: 0, rows: [] });
  datagrid = $('#dg_av').datagrid({
    columns:[[
      { field:'id',title:'编号',width:'20%' },
      { field:'valueName',title:'属性值名称',width:'80%',
        editor:{
          type:'validatebox', options:{ required: true } //必填项
        }
      ]
    ]],
    toolbar:[{text:'添加',iconCls:'icon-add',
      handler:function () {
        datagrid.datagrid('appendRow',{id:'',valueName:''});
      }
    },{
      text:'删除',iconCls:'icon-remove',
      handler:function () {
        var row = datagrid.datagrid('getSelected');
        if(row){
          var rowIndex = datagrid.datagrid('getRowIndex', row);
          datagrid.datagrid('deleteRow', rowIndex);
        }
      }
    }
  ]],
    onDoubleClickRow: function (rowIndex, rowData) {
      //双击开启编辑行
      datagrid.datagrid("beginEdit", rowIndex);
      //设定当失去焦点时,退出编辑状态
      var valueName = rowData.valueName;
      $('#input.datagrid-editable-input').val(valueName).bind("blur",function(evt){
        datagrid.datagrid('endEdit',rowIndex);
      });
    }
  });
}
```



```
});  
}  
});  
}
```

然后点击增加后要弹出对话框

```
function addAttrInfo(){  
    if(!checkBeforeDialog()){  
        return ;  
    }  
    //进系统前先清空  
    $("#attrId").val("");  
    $("#attrName").textbox('clear');  
    $("#dg_av").datagrid({url:""});  
    // 初始化 datagrid  
    initAttrValueDatagrid();  
  
    //弹出框  
    $("#dlg").dialog("open");  
}  
  
function editAttrInfo(){  
    if(!checkBeforeDialog()){  
        return ;  
    }  
    // 初始化 datagrid  
    initAttrValueDatagrid();  
    //进页面前先加载数据  
    var attrInfoRow=$("#dg").datagrid('getSelected');  
    $("#dg_av").datagrid({url:'getAttrValueList?attrId='+attrInfoRow.id});  
    $("#attrId").val(attrInfoRow.id);  
    $("#attrName").textbox('setValue',attrInfoRow.attrName);  
  
    //弹出框  
    $("#dlg").dialog("open");  
}  
  
function checkBeforeDialog(){  
    var ctg3val = $("#ctg3ForAttrList").combobox('getValue');  
    if(ctg3val==""){  
        $.messager.alert('警告','请先选择三级分类','warning');  
        return false;  
    }  
    return true;  
}
```

### 3.2.2 后台代码

点击编辑时调用后台的请求

<code>getAttrValueList</code>	获得属性值列表信息的请求
-------------------------------	--------------

由于属性和属性值是一对多的关系，所以属性的 bean 中增加一个列表元素

```
public class BaseAttrInfo implements Serializable {  
  
    @Id  
    @Column  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private String id;  
    @Column  
    private String attrName;  
    @Column  
    private String catalog3Id;  
  
    @Transient  
    private List<BaseAttrValue> attrValueList;  
}
```

其中@Transient 表示该 Bean 类对应的数据库表中不包含的字段，这个注解必须要加上否则报错。

AttrManageController

```
@RequestMapping(value = "getAttrValueList",method = RequestMethod.POST)  
@ResponseBody  
public List<BaseAttrValue> getAttrValueList(@RequestParam Map<String,String>  
map){  
    String attrId= map.get("attrId");  
    BaseAttrInfo attrInfo = manageService.getAttrInfo(attrId);  
    return attrInfo.getAttrValueList();  
}
```

gmall-manage-service 中的实现 ManageServiceImpl

```
@Override  
public BaseAttrInfo getAttrInfo(String id) {
```

```
// 查询属性基本信息
BaseAttrInfo baseAttrInfo = baseAttrInfoMapper.selectByPrimaryKey(id);

// 查询属性对应的属性值
BaseAttrValue baseAttrValue4Query = new BaseAttrValue();
baseAttrValue4Query.setAttrId(baseAttrInfo.getId());
List<BaseAttrValue> baseAttrValueList =
baseAttrValueMapper.select(baseAttrValue4Query);

baseAttrInfo.setAttrValueList(baseAttrValueList);
return baseAttrInfo;
}
```

### 3.3 属性和属性值的保存

#### 3.3.1 页面 js

```
function saveAttr(){
    var attrJson = {};
    //把表格中的数据循环组合成 json
    var attrValueRows = $("#dg_av").datagrid('getRows');
    for (var i = 0; i < attrValueRows.length; i++) {
        //技巧：与 bean 中的属性同名可以借助 springmvc 直接注入到实体 bean 中，即使是 list 也可以。
        attrJson["attrValueList["+i+"].id"] = attrValueRows[i].id;
        attrJson["attrValueList["+i+"].valueName"] = attrValueRows[i].valueName;
    }

    attrJson["attrName"] = $("#attrName").val();
    attrJson["id"] = $("#attrId").val();
    attrJson["catalog3Id"] = $("#ctg3ForAttrList").combobox('getValue');

    //ajax 保存到后台
    $.post("saveAttrInfo", attrJson, function(data){
        $("#dlg").dialog("close");
        $("#dg").datagrid("reload");
    })
}
```

### 3.3.2 后台代码

保存时调用后台的请求：

<b>saveAttrInfo</b>	保存属性和属性值的请求
---------------------	-------------

AttrManageController

```
@RequestMapping(value = "saveAttrInfo",method = RequestMethod.POST)
@ResponseBody
public String saveAttrInfo(BaseAttrInfo baseAttrInfo){
    manageService.saveAttrInfo(baseAttrInfo);
    return "success";
}
```

gmall-manage-service 中的实现 ManageServiceImpl

```
@Override
public void saveAttrInfo(BaseAttrInfo baseAttrInfo) {

    //如果有主键就进行更新，如果没有就插入
    if(baseAttrInfo.getId()!=null&&baseAttrInfo.getId().length()>0){
        baseAttrInfoMapper.updateByPrimaryKey(baseAttrInfo);
    }else{
        //防止主键被赋上一个空字符串
        if(baseAttrInfo.getId().length()==0){
            baseAttrInfo.setId(null);
        }
        baseAttrInfoMapper.insertSelective(baseAttrInfo);
    }

    //把原属性值全部清空
    BaseAttrValue baseAttrValue4Del = new BaseAttrValue();
    baseAttrValue4Del.setAttrId(baseAttrInfo.getId());
    baseAttrValueMapper.delete(baseAttrValue4Del);

    //重新插入属性
    if(baseAttrInfo.getAttrValueList()!=null&&baseAttrInfo.getAttrValueList().size()>0) {
        for (BaseAttrValue attrValue : baseAttrInfo.getAttrValueList()) {
            //防止主键被赋上一个空字符串
            if(attrValue.getId()!=null&&attrValue.getId().length()==0){
                attrValue.setId(null);
            }
            attrValue.setAttrId(baseAttrInfo.getId());
            baseAttrValueMapper.insertSelective(attrValue);
        }
    }
}
```

节省篇幅，接口类中的代码不再赋上，请自行添加。

### 3.4 最终效果

