**品优购电商系统开发**

**第8章**

**商品修改与审核**

传智播客.黑马程序员

# 1.商家后台-商品管理【商品列表】

## 1.1需求分析

在商家后台，显示该商家的商品列表信息，如下图：



## 1.2查询商家商品列表

### 1.2.1后端代码

修web-shop工程的GoodsController.java的search方法

|  |
| --- |
| //查询商品 分象对象  @RequestMapping(value = "/search.do")  **public** PageResult search(Integer page,Integer rows,@RequestBody Goods goods){  String name = SecurityContextHolder.*getContext*().getAuthentication().getName();  goods.setSellerId(name);    **return** goodsService.search(page, rows, goods);  } |

修改sellergoods

|  |
| --- |
| //查询分页对象 (商品的)  **public** PageResult search(Integer pageNum,Integer pageSize,Goods goods){  PageHelper.*startPage*(pageNum, pageSize);    GoodsQuery goodsQuery = **new** GoodsQuery();  Criteria createCriteria = goodsQuery.createCriteria();    createCriteria.andSellerIdEqualTo(goods.getSellerId());    **if**(**null** != goods){  **if**(**null** != goods.getAuditStatus()){  createCriteria.andAuditStatusEqualTo(goods.getAuditStatus());  }  **if**(**null** != goods.getGoodsName() && !"".equals(goods.getGoodsName().trim())){  createCriteria.andGoodsNameLike("%"+goods.getGoodsName().trim()+"%");  }  }    Page<Goods> page = (Page<Goods>) goodsDao.selectByExample(goodsQuery);  **return** **new** PageResult(page.getTotal(), page.getResult());  } |

### 1.2.2前端代码

修改goods.html. 引入js

|  |
| --- |
| <script type=*"text/javascript"* src=*"../plugins/angularjs/angular.min.js"*></script>  <!-- 分页组件开始 -->  <script type=*"text/javascript"* src=*"../plugins/angularjs/pagination.js"*></script>  <link rel=*"stylesheet"* href=*"../plugins/angularjs/pagination.css"*>  <!-- 分页组件结束 -->  <script type=*"text/javascript"* src=*"../js/base\_pagination.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/goodsService.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/itemCatService.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/uploadService.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/typeTemplateService.js"*></script>  <script type=*"text/javascript"* src=*"../js/controller/baseController.js"*></script>  <script type=*"text/javascript"* src=*"../js/controller/goodsController.js"*></script> |

添加指令

|  |
| --- |
| <body class=*"hold-transition skin-red sidebar-mini"* ng-app=*"pinyougou"* ng-controller=*"goodsController"*> |

在页面上放置分页控件

|  |
| --- |
| <tm-pagination conf=*"paginationConf"*></tm-pagination> |

循环列表

|  |
| --- |
| <tr ng-repeat=*"entity in list"*>  <td><input type=*"checkbox"*></td>  <td>{{entity.id}}</td>  <td>{{entity.goodsName}}</td>  <td>{{entity.price}}</td>  <td>{{entity.category1Id}}</td>  <td>{{entity.category2Id}}</td>  <td>{{entity.category3Id}}</td>  <td>  {{entity.auditStatus}}  </td>  <td class=*"text-center"*>  <button type=*"button"* class=*"btn bg-olive btn-xs"*>修改</button>  </td>  </tr> |

显示效果如下：



## 1.3显示状态

修改goodsController.js，添加state数组

|  |
| --- |
| $scope.status=['未审核','已审核','审核未通过','关闭'];//商品状态 |

修改列表显示

|  |
| --- |
| {{status[entity.auditStatus]}} |

显示效果如下：



## 1.4显示分类

我们现在的列表中的分类仍然显示ID

如何才能显示分类的名称呢？

方案一：在后端代码写关联查询语句，返回的数据中直接有分类名称。

方案二：在前端代码用ID去查询后端，异步返回商品分类名称。

我们目前采用方案二：

（1）修改goodsController.js

|  |
| --- |
| $scope.itemCatList=[];//商品分类列表  //加载商品分类列表  $scope.findItemCatList=**function**(){  itemCatService.findAll().success(  **function**(response){  **for**(**var** i=0;i<response.length;i++){  $scope.itemCatList[response[i].id]=response[i].name;  }  }  );  } |

代码解释：因为我们需要根据分类ID得到分类名称，所以我们将返回的分页结果以数组形式再次封装。

1. 修改goods.html ,增加初始化调用

|  |
| --- |
| <body class=*"hold-transition skin-red sidebar-mini"* ng-app=*"pinyougou"* ng-controller=*"goodsController"* ng-init=*"findItemCatList()"*> |

1. 修改goods.html , 修改列表

|  |
| --- |
| <td>{{itemCatList[entity.category1Id]}}</td>  <td>{{itemCatList[entity.category2Id]}}</td>  <td>{{itemCatList[entity.category3Id]}}</td> |

## 1.5条件查询

根据状态和商品名称进行查询

修改goods.html

|  |
| --- |
| <div class=*"has-feedback"*>  状态：<select ng-model=*"searchEntity.auditStatus"*>  <option value=*""*>全部</option>  <option value=*"0"*>未审核</option>  <option value=*"1"*>已审核</option>  <option value=*"2"*>审核未通过</option>  <option value=*"3"*>关闭</option>  </select>  商品名称：<input ng-model=*"searchEntity.goodsName"*>  <button class=*"btn btn-default"* ng-click=*"reloadList()"*>查询</button>  </div> |

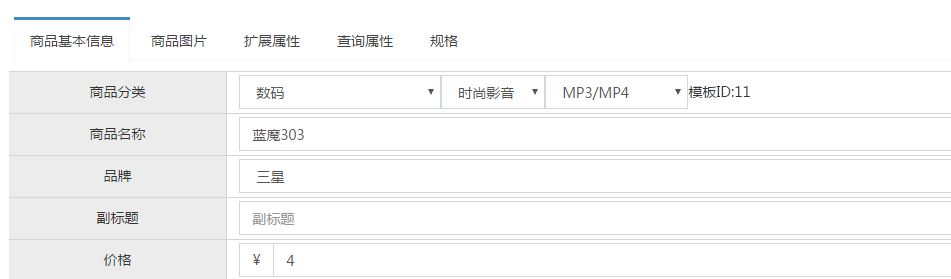
# 2.商家后台-商品管理【商品修改】

## 2.1需求分析

在商品列表页面点击修改，进入商品编辑页面，并传递参数商品ID，商品编辑页面接受该参数后从数据库中读取商品信息，用户修改后保存信息。

## 2.2基本信息读取

我们首选读取商品分类、商品名称、品牌，副标题，价格等信息



### 2.2.1后端代码

1. 修改sellergoods的GoodsServiceImpl.java

|  |
| --- |
| @Override  //查询一个商品 组合 对象  **public** GoodsVo findOne(Long id){  GoodsVo vo = **new** GoodsVo();  //查询商品对象  Goods goods = goodsDao.selectByPrimaryKey(id);  GoodsDesc goodsDesc = goodsDescDao.selectByPrimaryKey(id);  ItemQuery itemQuery = **new** ItemQuery();  itemQuery.createCriteria().andGoodsIdEqualTo(id);  List<Item> items = itemDao.selectByExample(itemQuery);  vo.setGoods(goods);  vo.setGoodsDesc(goodsDesc);  vo.setItemList(items);  **return** vo;  } |

1. 修改shop-web（和manager-web）的GoodsController.java

|  |
| --- |
| /\*\*  \* 获取实体  \* **@param** id  \* **@return**  \*/  //查询一个商品 组合 对象  @RequestMapping(value = "/findOne.do")  **public** GoodsVo findOne(Long id){  **return** goodsService.findOne(id);  } |

### 2.2.2前端代码

（1）在goodsController中引入$location服务

|  |
| --- |
| //商品控制层（商家后台）  app.controller('goodsController',**function**($scope,$controller,**$location**,goodsService,uploadService,item\_catService,type\_templateService){  ...... |

（2）修改goodsController 添加代码：

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  **var** id= $location.search()['id'];//获取参数值  **if**(id==**null**){  **return** ;  }  goodsService.findOne(id).success(  **function**(response){  $scope.entity= response;  }  );  } |

在goods\_edit.html页面上添加指令

|  |
| --- |
| <body class=*"hold-transition skin-red sidebar-mini"* ng-app=*"pinyougou"* ng-controller=*"goodsController"* ng-init=*"selectItemCat1List();findOne()"*> |

测试：

地址栏输入

<http://localhost:9102/admin/goods_edit.html#?id=149187842867969>

注意： ?前要加# ，则是angularJS的地址路由的书写形式



## 2.3读取商品介绍（富文本编辑器）

修改前端代码 goodsController

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  .................  goodsService.findOne(id).success(  **function**(response){  $scope.entity= response;  //向富文本编辑器添加商品介绍  editor.html($scope.entity.goodsDesc.introduction);  }  );  } |

## 2.4显示商品图片列表

修改goodsController.js ，在dataLogic方法添加代码，将图片列表由字符串转换为json集合对象

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  ..............  //如果有ID,则查询实体  goodsService.findOne(id).success(  **function**(response){  $scope.entity= response;  //向富文本编辑器添加商品介绍  editor.html($scope.entity.goodsDesc.introduction);  //显示图片列表  $scope.entity.goodsDesc.itemImages=  JSON.parse($scope.entity.goodsDesc.itemImages);  }  );  } |

## 2.5读取商品扩展属性

修改goodsController.js

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  .........  goodsService.findOne(id).success(  **function**(response){  .......................  **//显示扩展属性**  **$scope.entity.goodsDesc.customAttributeItems= JSON.parse($scope.entity.goodsDesc.customAttributeItems);**  }  );  } |

经过测试，我们发现扩展属性值并没有读取出来，这是因为与下列代码发生冲突

|  |
| --- |
| $scope.$watch('entity.goods.typeTemplateId',**function**(newValue,oldValue){  ......  $scope.entity.goodsDesc.customAttributeItems = JSON.parse($scope.typeTemplate.customAttributeItems);//扩展属性  } |

我们读取出来的值被覆盖了，我们需要改写代码, 添加判断，当用户没有传递id参数时再执行此逻辑

|  |
| --- |
| //模板ID选择后 更新品牌列表  $scope.$watch('entity.goods.typeTemplateId', **function**(newValue, oldValue) {  // 根据模板ID查询模板的数据  typeTemplateService.findOne(newValue).success(  **function**(response){  $scope.typeTemplate = response;  // 将品牌的字符串数据转成JSON  $scope.typeTemplate.brandIds = JSON.parse( $scope.typeTemplate.brandIds );  // 将扩展属性的字符串转成JSON  **if**($location.search()['id'] == **null**){  $scope.entity.goodsDesc.customAttributeItems = JSON.parse( $scope.typeTemplate.customAttributeItems );  }  }  );  …………………等等 |

## 2.6读取商品规格属性

修改goodsController

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  ......  goodsService.findOne(id).success(  **function**(response){  $scope.entity= response;  editor.html($scope.entity.goodsDesc.introduction);//商品介绍  $scope.entity.goodsDesc.itemImages= JSON.parse($scope.entity.goodsDesc.itemImages);//图片列表  //扩展属性列表  $scope.entity.goodsDesc.customAttributeItems =JSON.parse($scope.entity.goodsDesc.customAttributeItems);  **//规格 $scope.entity.goodsDesc.specificationItems=JSON.parse($scope.entity.goodsDesc.specificationItems);**  }  );  } |

|  |
| --- |
| //根据规格名称和选项名称返回是否被勾选  $scope.checkAttributeValue=**function**(specName,optionName){  **var** items= $scope.entity.goodsDesc.specificationItems;  **var** object= $scope.searchObjectByKey(items,'attributeName',specName);  **if**(object==**null**){  **return** **false**;  }**else**{  **if**(object.attributeValue.indexOf(optionName)>=0){  **return** **true**;  }**else**{  **return** **false**;  }  }  } |

修改goods\_edit.html页面上规格面板的复选框,运用 ng-checked指令控制复选框的勾选状态

|  |
| --- |
| <input type=*"checkbox"* ng-click=*"updateSpecAttribute($event,pojo.text,p.optionName);createSKUTable()"* ng-checked=*"checkAttributeValue(pojo.text,p.optionName)"*>{{p.optionName}} |

## 2.7读取SKU数据

显示SKU商品列表，并自动读取价格、库存等数据加载到列表中



### 2.7.1后端代码

在GoodsServiceImpl的findOne方法中加载SKU商品数据

|  |
| --- |
| //查询一个商品 组合 对象  **public** GoodsVo findOne(Long id){  GoodsVo vo = **new** GoodsVo();  //查询商品对象  Goods goods = goodsDao.selectByPrimaryKey(id);  GoodsDesc goodsDesc = goodsDescDao.selectByPrimaryKey(id);  ItemQuery itemQuery = **new** ItemQuery();  itemQuery.createCriteria().andGoodsIdEqualTo(id);  List<Item> items = itemDao.selectByExample(itemQuery);  vo.setGoods(goods);  vo.setGoodsDesc(goodsDesc);  vo.setItemList(items);  **return** vo;  } |

### 2.7.2前端代码

在goodsController.js修改findOne方法的代码

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  ........  goodsService.findOne(id).success(  **function**(response){  $scope.entity= response;  .........  //SKU列表规格列转换  **for**( **var** i=0;i<$scope.entity.itemList.length;i++ ){  $scope.entity.itemList[i].spec =  JSON.parse( $scope.entity.itemList[i].spec);  }  }  );  } |

## 2.8保存数据

### 2.8.1后端代码

修改 interface 的 GoodsService.java

|  |
| --- |
| **public** **void** update(Goods goods); |

修改sellergoods的GoodsServiceImpl ,将SKU列表插入的代码提取出来，封装到私有方法中

|  |
| --- |
| //插入sku列表数据  //修改商品  **public** **void** update(GoodsVo vo){    goodsDao.updateByPrimaryKeySelective(vo.getGoods());  goodsDescDao.updateByPrimaryKeySelective(vo.getGoodsDesc());  //先删除 再添加  ItemQuery itemQuery = **new** ItemQuery();  itemQuery.createCriteria().andGoodsIdEqualTo(vo.getGoods().getId());  itemDao.deleteByExample(itemQuery);  //保存SKu表 库存  //判断是否启用规格  **if**("1".equals(vo.getGoods().getIsEnableSpec())){  //启用  List<Item> itemList = vo.getItemList();  **for** (Item item : itemList) {  Map<String, Object> map = JSON.*parseObject*(item.getSpec());  //标题  String title = vo.getGoods().getGoodsName();  Set<Entry<String, Object>> entrySet = map.entrySet();  **for** (Entry<String, Object> entry : entrySet) {  title += " " + entry.getValue();  }  item.setTitle(title);  //创建Item对象  setItemValues(vo,item);  //保存item表  itemDao.insertSelective(item);  }  }**else**{  //不启用  Item item = **new** Item();  //标题  item.setTitle(vo.getGoods().getGoodsName());  //价格  item.setPrice(vo.getGoods().getPrice());  //库存  item.setNum(99999);  //规格  item.setSpec("{}");  //状态 启用  item.setStatus("1");  //默认  item.setIsDefault("1");  //设置Item的属性  setItemValues(vo, item);  //保存item表  itemDao.insertSelective(item);    }  }  //创建Item对象  **public** **void** setItemValues(GoodsVo vo,Item item){  //图片保存第一张  List<Map> list = JSON.*parseArray*(vo.getGoodsDesc().getItemImages(), Map.**class**);  item.setImage((String) list.get(0).get("url"));  //商品分类Id 保存第三级商品分类ID  item.setCategoryid(vo.getGoods().getCategory3Id());  //商品分类名称  ItemCat itemCat = itemCatDao.selectByPrimaryKey(vo.getGoods().getCategory3Id());  item.setCategory(itemCat.getName());  //添加时间  item.setCreateTime(**new** Date());  //修改时间  item.setUpdateTime(**new** Date());  //商品ID  item.setGoodsId(vo.getGoods().getId());  //当前登陆的用户ID  item.setSellerId(vo.getGoods().getSellerId());  //当前登陆的用户名  Seller seller = sellerDao.selectByPrimaryKey(vo.getGoods().getSellerId());  item.setSeller(seller.getName());  //品牌名称  Brand brand = brandDao.selectByPrimaryKey(vo.getGoods().getBrandId());  item.setBrand(brand.getName());  } |

在add方法中调用 此方法，修改如下：

|  |
| --- |
| @Override  **public** **void** add(GoodsVo vo) {  // 保存商品表 主键  // 未审核状态  vo.getGoods().setAuditStatus("0");  // 返回商品ID (自增长的方式生成的  goodsDao.insertSelective(vo.getGoods());  // 商品详情表  vo.getGoodsDesc().setGoodsId(vo.getGoods().getId());  goodsDescDao.insertSelective(vo.getGoodsDesc());  // 是否启用规格  **if** ("1".equals(vo.getGoods().getIsEnableSpec())) {  // 启用了  // 库存表  // **TODO** 库存 一部iphone7手机 黑色 白色 ...  List<Item> itemList = vo.getItemList();  **for** (Item item : itemList) {  // ID  // SKU名称 = SPU名称 + 规格  // 规格 : {"机身内存":"16G","网络":"联通3G"}  String spec = item.getSpec();  Map<String, String> specMap = JSON.*parseObject*(spec, Map.**class**);  Set<Entry<String, String>> entrySet = specMap.entrySet();  String title = vo.getGoods().getGoodsName();  **for** (Entry<String, String> entry : entrySet) {  title += " " + entry.getValue();  }  item.setTitle(title);  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 保存  itemDao.insertSelective(item);  }  } **else** {  // 不启用  // ID  Item item = **new** Item();// 创建库存表对象  // 规格 : {"机身内存":"16G","网络":"联通3G"}  item.setSpec("{}");  // SKU名称 = SPU名称  item.setTitle(vo.getGoods().getGoodsName());  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 价格  item.setPrice(**new** BigDecimal(0));  // 库存数量  item.setNum(9999);  // 是否启用  item.setStatus("1");  // 默认  item.setIsDefault("1");  // 保存  itemDao.insertSelective(item);  }  }  @Override  **public** **void** add(GoodsVo vo) {  // 保存商品表 主键 一部iphone7手机  // 未审核状态  vo.getGoods().setAuditStatus("0");  // 返回商品ID (自增长的方式生成的 返回回来)  goodsDao.insertSelective(vo.getGoods());  // 商品详情表  vo.getGoodsDesc().setGoodsId(vo.getGoods().getId());  goodsDescDao.insertSelective(vo.getGoodsDesc());  // 是否启用规格  **if** ("1".equals(vo.getGoods().getIsEnableSpec())) {  // 启用了  // 库存表  // **TODO** 库存 一部iphone7手机 黑色 白色 ...  List<Item> itemList = vo.getItemList();  **for** (Item item : itemList) {  // ID  // SKU名称 = SPU名称 + 规格  // 规格 : {"机身内存":"16G","网络":"联通3G"}  String spec = item.getSpec();  Map<String, String> specMap = JSON.*parseObject*(spec, Map.**class**);  Set<Entry<String, String>> entrySet = specMap.entrySet();  String title = vo.getGoods().getGoodsName();  **for** (Entry<String, String> entry : entrySet) {  title += " " + entry.getValue();  }  item.setTitle(title);  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 保存  itemDao.insertSelective(item);  }  } **else** {  // 不启用  // ID  Item item = **new** Item();// 创建库存表对象  // 规格 : {"机身内存":"16G","网络":"联通3G"}  item.setSpec("{}");  // SKU名称 = SPU名称  item.setTitle(vo.getGoods().getGoodsName());  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 价格  item.setPrice(**new** BigDecimal(0));  // 库存数量  item.setNum(9999);  // 是否启用  item.setStatus("1");  // 默认  item.setIsDefault("1");  // 保存  itemDao.insertSelective(item);  }  } |

怎么样，是不是比原来更加清爽了呢？

接下来，我们修改update方法，实现修改

|  |
| --- |
| // 修改  **public** **void** update(GoodsVo vo) {  // 修改商品表  goodsDao.updateByPrimaryKeySelective(vo.getGoods());  // 修改商品详情表  goodsDescDao.updateByPrimaryKeySelective(vo.getGoodsDesc());  // 修改库存表 SKU  // 先删除  ItemQuery itemQuery = **new** ItemQuery();  itemQuery.createCriteria().andGoodsIdEqualTo(vo.getGoods().getId());  itemDao.deleteByExample(itemQuery);  // 再添加  // 是否启用规格  **if** ("1".equals(vo.getGoods().getIsEnableSpec())) {  // 启用了  // 库存表  // **TODO** 库存 一部iphone7手机 黑色 白色 ...  List<Item> itemList = vo.getItemList();  **for** (Item item : itemList) {  // ID  // SKU名称 = SPU名称 + 规格  // 规格 : {"机身内存":"16G","网络":"联通3G"}  String spec = item.getSpec();  Map<String, String> specMap = JSON.*parseObject*(spec, Map.**class**);  Set<Entry<String, String>> entrySet = specMap.entrySet();  String title = vo.getGoods().getGoodsName();  **for** (Entry<String, String> entry : entrySet) {  title += " " + entry.getValue();  }  item.setTitle(title);  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 保存  itemDao.insertSelective(item);  }  } **else** {  // 不启用  // ID  Item item = **new** Item();// 创建库存表对象  // 规格 : {"机身内存":"16G","网络":"联通3G"}  item.setSpec("{}");  // SKU名称 = SPU名称  item.setTitle(vo.getGoods().getGoodsName());  // 设置库存表对 象的属性  setItemAttribute(item, vo);  // 价格  item.setPrice(**new** BigDecimal(0));  // 库存数量  item.setNum(9999);  // 是否启用  item.setStatus("1");  // 默认  item.setIsDefault("1");  // 保存  itemDao.insertSelective(item);  }  } |

修改manager-web工程的GoodsController.java

|  |
| --- |
| @RequestMapping("/update")  **public** Result update(@RequestBody Goods goods){  ......  } |

修改shop-web工程的GoodsController.java

|  |
| --- |
| //修改  @RequestMapping("/update")  **public** Result update(@RequestBody GoodsVo vo){  **try** {  //添加  //商家ID  String sellerId = SecurityContextHolder.*getContext*().getAuthentication().getName();  vo.getGoods().setSellerId(sellerId);  //修改  goodsService.update(vo);  **return** **new** Result(**true**,"提交成功");  } **catch** (Exception e) {  // **TODO**: handle exception  **return** **new** Result(**false**,"提交失败");  }  } |

代码解释：出于安全考虑，在商户后台执行的商品修改，必须要校验提交的商品属于该商户

### 2.8.2前端代码

（1）修改goodsController.js ,新增保存的方法

|  |
| --- |
| //保存  $scope.save=**function**(){  //提取文本编辑器的值  $scope.entity.goodsDesc.introduction=editor.html();  **var** serviceObject;//服务层对象  **if**($scope.entity.goods.id!=**null**){//如果有ID  serviceObject=goodsService.update( $scope.entity ); //修改  }**else**{  serviceObject=goodsService.add( $scope.entity );//增加  }  serviceObject.success(  **function**(response){  **if**(response.success){  alert('保存成功');  $scope.entity={};  editor.html("");  }**else**{  alert(response.message);  }  }  );  } |

（2）修改goods\_edit.html 调用

|  |
| --- |
| <div class=*"btn-toolbar list-toolbar"*>  <button class=*"btn btn-primary"* ng-click=*"save()"*><i class=*"fa fa-save"*></i>保存</button>  <button class=*"btn btn-default"* >返回列表</button>  </div> |

## 2.9页面跳转

（1）由商品列表页跳转到商品编辑页

修改goods.html表格行的修改按钮

|  |
| --- |
| <a href=*"goods\_edit.html#?id={{entity.id}}"* class=*"btn bg-olive btn-xs"*>修改</a> |

1. 由商品编辑页跳转到商品列表

修改goods\_edit.html 的返回列表按钮

|  |
| --- |
| <a href=*"goods.html"* class=*"btn btn-default"*>返回列表</a> |

（3）保存成功后返回列表页面

|  |
| --- |
| //保存  $scope.save=**function**(){  .....  serviceObject.success(  **function**(response){  if(response.success){  alert('保存成功');  $scope.entity={};  editor.html("");  location.href="goods.html";//跳转到商品列表页  }**else**{  alert(response.message);  }  }  );  } |

# 3.运营商后台-商品管理【商品审核】

## 3.1待审核商品列表

需求：参照商家后台商品列表。代码：

1. 修改manager的goodsController.js，注入itemCatService，添加代码

|  |
| --- |
| $scope.status=['未审核','已审核','审核未通过','关闭'];//商品状态  $scope.itemCatList=[];//商品分类列表  //查询商品分类  $scope.findItemCatList=**function**(){  itemCatService.findAll().success(  **function**(response){  **for**(**var** i=0;i<response.length;i++){  $scope.itemCatList[response[i].id ]=response[i].name;  }  }  );  } |

1. 修改goods.html ,引入js

|  |
| --- |
| <script type=*"text/javascript"* src=*"../plugins/angularjs/angular.min.js"*></script>  <!-- 分页组件开始 -->  <script type=*"text/javascript"* src=*"../plugins/angularjs/pagination.js"*></script>  <link rel=*"stylesheet"* href=*"../plugins/angularjs/pagination.css"*>  <!-- 分页组件结束 -->  <script type=*"text/javascript"* src=*"../js/base\_pagination.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/goodsService.js"*></script>  <script type=*"text/javascript"* src=*"../js/service/itemCatService.js"*></script>  <script type=*"text/javascript"* src=*"../js/controller/baseController.js"*></script>  <script type=*"text/javascript"* src=*"../js/controller/goodsController.js"*></script> |

1. 指令，完成初始调用

|  |
| --- |
| <body class=*"hold-transition skin-red sidebar-mini"* ng-app=*"pinyougou"* ng-controller=*"goodsController"* ng-init=*"searchEntity={auditStatus:'0'};findItemCatList()"*> |

1. 循环列表

|  |
| --- |
| <tr ng-repeat=*"entity in list"*>  <td><input type=*"checkbox"*></td>  <td>{{entity.id}}</td>  <td>{{entity.goodsName}}</td>  <td>{{entity.price}}</td>  <td>{{itemCatList[entity.category1Id]}}</td>  <td>{{itemCatList[entity.category2Id]}}</td>  <td>{{itemCatList[entity.category3Id]}}</td>  <td>{{status[entity.auditStatus]}}</td>  <td class=*"text-center"*> </td>  </tr> |

1. 分页控件

|  |
| --- |
| <tm-pagination conf=*"paginationConf"*></tm-pagination> |

## 3.2商品审核与驳回

需求：商品审核通过的状态值为1，驳回的状态值为2 。用户在列表中选中ID后，点击审核或驳回，修改商品状态，并刷新列表。

### 3.2.1后端代码

1. 在interface的GoodsService.java新增方法定义

|  |
| --- |
| /\*\*  \* 批量修改状态  \* **@param** ids  \* **@param** status  \*/  **public** **void** updateStatus(Long []ids,String status); |

1. 在sellergoods的GoodsServiceImpl.java实现该方法

|  |
| --- |
| @Override  //更新状态 审核通过 或驳回  **public** **void** updateStatus(Long[] ids,String status){  Goods goods = **new** Goods();  goods.setAuditStatus(status);  **for** (Long id : ids) {  goods.setId(id);  goodsDao.updateByPrimaryKeySelective(goods);  }    } |

1. 在manager的GoodsController.java新增方法

|  |
| --- |
| //更新状态 审核通过 或驳回  @RequestMapping(value = "/updateStatus.do")  **public** Result updateStatus(Long[] ids,String status){  **try** {  goodsService.updateStatus(ids, status);  **return** **new** Result(**true**,"成功");  } **catch** (Exception e) {  // **TODO**: handle exception  **return** **new** Result(**true**,"失败");  }  } |

### 3.2.2前端代码

（1）修改manager-web的goodsService.js ，增加方法

|  |
| --- |
| //更改状态  **this**.updateStatus=**function**(ids,status){  **return** $http.get('../goods/updateStatus.do?ids='+ids+"&status="+status);  } |

（2）修改manager-web的goodsController.js ，增加方法

|  |
| --- |
| //更改状态  $scope.updateStatus=**function**(status){  goodsService.updateStatus($scope.selectIds,status).success(  **function**(response){  **if**(response.success){//成功  $scope.reloadList();//刷新列表  $scope.selectIds=[];//清空ID集合  }**else**{  alert(response.message);  }  }  );  } |

（3）修改manager-web的goods.html 页面，为复选框绑定事件指令

|  |
| --- |
| <tbody>  <tr ng-repeat=*"entity in list"*>  <td><input type=*"checkbox"* ng-click=*"updateSelection($event,entity.id)"*></td> |

1. 修改页面上的审核通过和驳回按钮

|  |
| --- |
| <button type=*"button"* class=*"btn btn-default"* title=*"审核通过"* ng-click=*"updateStatus('1')"*><i class=*"fa fa-check"*></i> 审核通过</button>  <button type=*"button"* class=*"btn btn-default"* title=*"驳回"* ng-click=*"updateStatus('2')"* ><i class=*"fa fa-ban"*></i> 驳回</button> |

# 4.运营商后台-商品管理【商品删除】

## 4.1需求分析

我们为商品管理提供商品删除功能，用户选中部分商品，点击删除按钮即可实现商品删除。注意，这里的删除并非是物理删除，而是修改tb\_goods表的is\_delete字段为1 ，我们可以称之为“逻辑删除”

## 4.2逻辑删除的实现

### 4.2.1后端代码

修改sellergoods-service工程的GoodsServiceImpl.java的delete方法

|  |
| --- |
| /\*\*  \* 批量删除  \*/  //删除 逻辑删除  **public** **void** delete(Long[] ids){  Goods goods = **new** Goods();  goods.setIsDelete("1");  **for** (Long id : ids) {  goods.setId(id);  goodsDao.updateByPrimaryKeySelective(goods);  }  } |

### 4.2.2前端代码

修改manager-web的goods.html上的删除按钮

|  |
| --- |
| <button type=*"button"* class=*"btn btn-default"* title=*"删除"* ng-click=*"dele()"*><i class=*"fa fa-trash-o"*></i> 删除</button> |

## 4.3排除已删除记录

修改pinyougou-sellergoods工程GoodsServiceImpl.java的findPage方法，添加以下代码：

|  |
| --- |
| //查询分页对象 (商品的)  **public** PageResult search(Integer pageNum,Integer pageSize,Goods goods){  PageHelper.*startPage*(pageNum, pageSize);    GoodsQuery goodsQuery = **new** GoodsQuery();  Criteria createCriteria = goodsQuery.createCriteria();  criteria.andIsDeleteIsNull();//指定条件为未逻辑删除记录 |

# 5.商家后台-【商品上下架】（学员实现）

## 5.1需求分析

什么是商品上下架？其实上下架也是商品的一个状态，但是不同于审核状态。审核状态的控制权在运营商手中，而上下架的控制权在商户手中。商户可以随时将一个已审核状态的商品上架或下架。上架表示正常销售，而下架则表示暂停销售。

## 5.2实现思路提示

其实商品的上下架就是对上下架状态的修改。字段为tb\_goods表的is\_marketable字段。1表示上架、0表示下架。