**品优购电商系统开发**

**第3章**

**规格及模板管理**

传智播客.黑马程序员

# 课程目标

目标1：理解和运用angularJS的service

目标2：理解和运用控制器继承

目标3：掌握代码生成器的使用

目标4：实现规格管理

目标5：实现模板管理

# 1.前端分层开发

## 1.1 需求分析

我们在上次课学习了angularJS并完成的品牌管理的增删改查功能。但是我们看代码，JS和html都放在一起，并不利于我们后期的维护。我们可以在前端代码中也运用MVC的设计模式，将代码进行分离，提高程序的可维护性。

## 1.2 自定义服务

在AngularJS中，服务是一个函数或对象，可在你的AngularJS应用中使用。我们在上次课中使用了内置服务$http .其实我们也可以自己来定义服务，而服务会封装一些操作。我们在不同的控制器中可以调用同一个服务，这样服务的代码将会被重用。

参考代码：



我们现在就修改一下我们的品牌管理代码，使用自定义服务。

|  |
| --- |
| **var** app=angular.module('pinyougou', ['pagination']);//定义模块  //品牌服务层  app.service('brandService',**function**($http){  //读取列表数据绑定到表单中  **this**.findAll=**function**(){  **return** $http.get('../brand/findAll.do');  }  //其它方法省略.......  });  //品牌控制层  app.controller('brandController' ,**function**($scope,brandService){  //读取列表数据绑定到表单中  $scope.findAll=**function**(){  brandService.findAll().success(  **function**(response){  $scope.list=response;  }  );  }  //其它方法省略........  }); |

## 1.3代码分离

我们刚才已经将与后端交互的部分放入自定义服务，目的是不同的控制层都可以重复调用服务层方法。所以我们还需要将代码分离出来，以便调用。

### **1.3.1 前端基础层**

在pinyougou-manager-web工程js下创建base.js:

|  |
| --- |
| **var** app=angular.module('pinyougou',[]); |

创建base\_pagination.js

|  |
| --- |
| **var** app=angular.module('pinyougou',['pagination']); |

一个用于不需要分页功能的页面，一个用于需要分页功能的页面.

### **1.3.2 前端服务层**

在pinyougou-manager-web工程js下创建service文件夹。创建brandService.js

|  |
| --- |
| //品牌服务层  app.service('brandService',**function**($http){  //读取列表数据绑定到表单中  **this**.findAll=**function**(){  **return** $http.get('../brand/findAll.do');  }  //其它方法省略........  }); |

### **1.3.3 前端控制层**

在pinyougou-manager-web的js文件夹下创建brandController.js

|  |
| --- |
| //品牌控制层  app.controller('brandController' ,**function**($scope,brandService){  //读取列表数据绑定到表单中  $scope.findAll=**function**(){  brandService.findAll().success(  **function**(response){  $scope.list=response;  }  );  }  //其它方法省略........  }); |

### **1.3.4 修改页面**

去掉brand.html原来的JS代码，引入刚才我们建立的JS

|  |
| --- |
| <scripttype=*"text/javascript"*src=*"../js/base\_pagination.js"*></script>  <scripttype=*"text/javascript"*src=*"../js/service/brandService.js"*></script>  <scripttype=*"text/javascript"*src=*"../js/controller/brandController.js"*></script> |

# 2.控制器继承

## 2.1需求分析

有些功能是每个页面都有可能用到的，比如分页，复选等等，如果我们再开发另一个功能，还需要重复编写。怎么能让这些通用的功能只写一次呢？我们通过继承的方式来实现。

## 2.2前端代码

### **2.2.1 建立父控制器**

在pinyougou-manager-web的js/controller目录下建立baseController.js

|  |
| --- |
| *//基本控制层* ***app***.**controller**(**'baseController'**, **function** ($scope) {  *//重新加载列表 数据* $scope.reloadList = **function** () {  *//切换页码* $scope.**search**($scope.**paginationConf**.**currentPage**, $scope.**paginationConf**.**itemsPerPage**);  }  *//分页控件配置* $scope.**paginationConf** = {  **currentPage**: 1,  **totalItems**: 10,  **itemsPerPage**: 10,  **perPageOptions**: [10, 20, 30, 40, 50],  onChange: **function** () {  $scope.reloadList();*//重新加载* }  };  $scope.**selectIds** = [];*//选中的ID集合  //更新复选* $scope.updateSelection = **function** ($event, id) {  **if** ($event.**target**.**checked**) {*//如果是被选中,则增加到数组* $scope.**selectIds**.push(id);  } **else** {  **var idx** = $scope.**selectIds**.indexOf(id);  $scope.**selectIds**.splice(idx, 1);*//删除* }  } }); |

### **2.2.2 修改品牌控制器层**

修改brandController.js

|  |
| --- |
| ***app***.**controller**(**'brandController'**, **function** ($scope,$controller,brandService) { //共享$scope  $controller(**'baseController'**,{**$scope**:$scope});   *//读取列表数据绑定到表单中* $scope.findAll = **function** () {  brandService.findAll().success(  **function** (response) {  $scope.**list** = response;  }  );  }  *//分页* $scope.findPage = **function** (page, rows) {  brandService.findPage(page,rows).success(  **function** (response) {  $scope.**list** = response.**rows**;  $scope.**paginationConf**.**totalItems** = response.**total**;*//更新总记录数* }  );  }   $scope.save=**function**(){  **var** serviceObject;*//服务层对象* **if**($scope.**entity**.**id**!=**null**){*//如果有ID* serviceObject=brandService.update( $scope.**entity** ); *//修改* }**else**{  serviceObject=brandService.add( $scope.**entity** );*//增加* }  serviceObject.success(  **function**(response){  **if**(response.**success**){  *//重新查询* $scope.reloadList();*//重新加载* }**else**{  alert(response.**message**);  }  }  );  }  $scope.findOne = **function** (id) {  brandService.findOne(id).success(  **function** (response) {  $scope.**entity** = response;  }  );  }  *//批量删除* $scope.dele = **function** () {  *//获取选中的复选框* brandService.dele($scope.**selectIds**).success(  **function** (response) {  **if** (response.**success**) {  $scope.reloadList();*//刷新列表* }  }  );  }  $scope.**searchEntity** = {};*//定义搜索对象  //条件查询* $scope.search = **function** (page, rows) {  brandService.**search**(page,rows,$scope.**searchEntity**).success(  **function** (response) {  $scope.**paginationConf**.**totalItems** = response.**total**;*//总记录数* $scope.**list** = response.**rows**;*//给列表变量赋值* }  );  } }); |

$controller也是angular提供的一个服务，可以实现伪继承，实际上就是与baseController共享$scope

# 3.代码生成器

由于项目中的代码都是由dao service controller组成，同时使用angular来开发的时候，也是有service controller 所以一般这种代码都是比较通用的。

使用代码生成器来生成来提供效率。

## 3.1代码生成

我们接下来使用《黑马程序员代码生成器2.4》来完成代码的编写。生成后将代码拷贝到工程中。具体步骤如下：

1. 资源中HeimaCodeUtil\_V2.4 就是代码生成器，**将其拷贝到不包含中文和空格的目录下**
2. 运行heima\_code\_util.exe 即可看到数据库连接窗口



1. 选择数据库类型为 MYSQL ,输入用户名和密码后点击“测试连接”按钮,提示连接成功后选择数据库，点击“下一步”。



（4）选择模板为SSM+dubbox+angularJS(服务层+WEB层)



这个模板不会生成数据访问层和实体类，因为我们之前已经用逆向工程完成了数据访问层与实体类的生成。

（5）点击生成代码按钮，提示成功后，到生成路径去找生成的代码，并拷贝到我们的工程中。

## 3.2代码拷贝

将商家商品相关代码拷贝到工程。

1. 拷贝服务接口
2. 拷贝服务实现类
3. 拷贝控制器类
4. 拷贝JS

## 3.3安装到本地仓库

执行maven命令install ,将最新的品优购代码安装到本地仓库

# 4.规格管理

## 4.1需求及表结构分析

### **4.1.1 需求**

实现规格管理功能(CRUD)



### **4.1.2 表结构**

**tb\_specification规格表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 含义 |
| Id | Bigint |  | 主键 |
| Spec\_name | Varchar | 255 | 规格名称 |

**tb\_specification\_option规格选项表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 含义 |
| Id | Bigint |  | 主键 |
| Option\_name | Varchar | 200 | 规格选项名称 |
| Spec\_id | Bigint | 30 | 规格ID |
| Orders | Int | 11 | 排序 |

## 4.2规格列表

### **4.2.1 引入JS**

修改pinyougou-manager-web工程的specification.html

|  |
| --- |
| <**script type="text/javascript" src="../plugins/angularjs/angular.min.js"**></**script**>  <**script src="../plugins/angularjs/pagination.js"**></**script**> <**link rel="stylesheet" href="../plugins/angularjs/pagination.css"**> <**script type="text/javascript" src="../js/base\_pagination.js"**></**script**>  <**script type="text/javascript" src="../js/service/specificationService.js"**></**script**> <**script type="text/javascript" src="../js/controller/baseController.js"**></**script**>  <**script type="text/javascript" src="../js/controller/specificationController.js"**></**script**> |

### **4.2.2 放置分页组件**

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

### **4.2.3 指令与表达式**

在body元素指定模块名和控制器名

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

循环表格行

|  |
| --- |
| <trng-repeat=*"entity in list"*>  <td><inputtype=*"checkbox"*></td>  <td>{{entity.id}}</td>  <td>{{entity.specName}}</td>  <tdclass=*"text-center"*>  <buttontype=*"button"*class=*"btnbg-olive btn-xs"*data-toggle=*"modal"*data-target=*"#editModal"*>修改</button>  </td>  </tr> |

## 4.3新增规格

### **4.3.1 新增行的实现**

修改specificationController.js 新增以下代码

|  |
| --- |
| //新增选项行  $scope.addTableRow=**function**(){  $scope.entity.specificationOptionList.push({});  } |

specification.html “新建选项”按钮

|  |
| --- |
| <button type=*"button"* class=*"btnbtn-default"* title=*"新建"*ng-click=*"addTableRow()"*><i class=*"fafa-file-o"*></i> 新建</button> |

循环列表行，绑定表格内的编辑框

|  |
| --- |
| <trng-repeat=*"pojo in entity.specificationOptionList"*>  <td><inputtype=*"checkbox"*></td>  <td>  <inputng-model=*"pojo.optionName"*class=*"form-control"*placeholder=*"规格选项"*>  </td>  <td>  <inputng-model=*"pojo.orders"*class=*"form-control"*placeholder=*"排序"*>  </td>  </tr> |

注意：要修改specification.html “新建”按钮，弹出窗口时对entity进行初始化，否则向集合添加数据时会报错！

|  |
| --- |
| <button type=*"button"* class=*"btnbtn-default"* title=*"新建"* data-toggle=*"modal"* data-target=*"#editModal"*ng-click=*"entity={'specificationOptionList':[]}"*><i class=*"fafa-file-o"*></i> 新建</button> |

### **4.3.2 删除行的实现**

实现思路：在每一行将索引值传递给集合，在集合中删除。

修改specificationController.js 新增以下代码

|  |
| --- |
| //批量选项删除  $scope.deleTableRow=**function**(index){  $scope.entity.specificationOptionList.splice(index,1);//删除  } |

修改每行的删除按钮

|  |
| --- |
| <button type=*"button"* class=*"btnbtn-default"* title=*"删除"*ng-click=*"deleTablenRow($index)"*><i class=*"fafa-file-o"*></i> 删除</button> |

$index 用于获取ng-repeat指令循环中的索引。

### **4.3.3 提交保存**

实现思路：我们将规格和规格选项数据合并成一个对象来传递，这时我们需要用一个对象将这两个对象组合起来。在业务逻辑中，得到组合对象中的规格和规格选项列表，插入规格返回规格ID，然后循环插入规格选项。

1. 我们要增加规格选项，必须要知道新增规格的ID, 所以我们在修改pinyougou-dao的TbSpecificationMapper.xml ，在insert节点后添加如下配置 改了DAO需要重新安装下DAO

|  |
| --- |
| <insert id=*"insert"*parameterType=*"com.pinyougou.pojo.TbSpecification"*>  <selectKey resultType=*"java.lang.Long"*order=*"AFTER"*keyProperty=*"id"*>  SELECT LAST\_INSERT\_ID() AS id  </selectKey>  insert into tb\_specification (id, spec\_name)  values (#{id,jdbcType=BIGINT}, #{specName,jdbcType=VARCHAR})  </insert> |

或者：

|  |
| --- |
| <**insert id="insert" parameterType="com.pinyougou.pojo.TbSpecification" useGeneratedKeys="true" keyProperty="id"**>  insert into tb\_specification (id, spec\_name)  values (#{id,jdbcType=BIGINT}, #{specName,jdbcType=VARCHAR}) </**insert**> |

（2）在pinyougou-pojo建立com.pinyougou.pojogroup包，包下建立Specification类

|  |
| --- |
| **package** com.pinyougou.pojogroup;  **import**java.io.Serializable;  **import**java.util.List;  **import**com.pinyougou.pojo.TbSpecification;  **import**com.pinyougou.pojo.TbSpecificationOption;  /\*\*  \* 规格组合实体类  \* **@author** Administrator  \*  \*/  **Publicclass** Specification **implements** Serializable {  **private** TbSpecification specification;  **private** List<TbSpecificationOption> specificationOptionList;    **public**TbSpecificationgetSpecification() {  **return** specification;  }  **public void** setSpecification(TbSpecification specification) {  **this**.specification = specification;  }  **public** List<TbSpecificationOption> getSpecificationOptionList() {  **return** specificationOptionList;  }  **publicvoid**setSpecificationOptionList(List<TbSpecificationOption>specificationOptionList) {  **this**.specificationOptionList = specificationOptionList;  }  } |

1. 修改pinyougou-sellergoods-interface的SpecificationService.java

|  |
| --- |
| /\*\*  \* 增加  \*/  **public void** add(Specification specification); |

1. 修改pinyougou-sellergoods-service的SpecificationServiceImpl.java

|  |
| --- |
| /\*\*  \* 增加  \*/  @Override  **publicvoid** add(Specification specification) {  specificationMapper.insert(specification.getSpecification());//插入规格  //循环插入规格选项  **for**(TbSpecificationOptionspecificationOption:specification.getSpecificationOptionList()){ specificationOption.setSpecId(specification.getSpecification().getId());//设置规格ID specificationOptionMapper.insert(specificationOption);  }  } |

（5）修改pinyougou-manager-web的SpecificationController.java

|  |
| --- |
| /\*\*  \* 增加  \* **@param specification**  \* **@return**  \*/  @RequestMapping("/add")  **public** Result add(@RequestBody Specification specification){  **try** {  specificationService.add(specification);  **return new** Result(**true**, "增加成功");  } **catch** (Exception e) {  e.printStackTrace();  **returnnew** Result(**false**, "增加失败");  }  } |

1. 修改页面specification.html

绑定规格名称

|  |
| --- |
| <table class=*"table table-bordered table-striped"*width=*"800px"*>  <tr>  <td>规格名称</td>  <td>  <input ng-model=*"entity.specification.specName"* class=*"form-control"* placeholder=*"规格名称"*>  </td>  </tr>  </table> |

绑定保存按钮事件

|  |
| --- |
| <button class=*"btnbtn-success"* data-dismiss=*"modal"* aria-hidden=*"true"*ng-click=*"save()"*>保存</button> |

## 4.4修改规格

### **4.4.1 获取规格数据**

实现思路：通过规格ID，到后端查询规格和规格选项列表，然后通过组合实体类返回结果

（1）修改pinyougou-sellergoods-interface的SpecificationService.java

|  |
| --- |
| /\*\*  \* 根据ID获取实体  \* **@param** id  \* **@return**  \*/  **public Specification** findOne(Long id); |

（2）修改pinyougou-sellergoods-service的SpecificationServiceImpl.java

|  |
| --- |
| /\*\*  \* 根据ID获取实体  \* **@param** id  \* **@return**  \*/  @Override  **public** Specification findOne(Long id){  //查询规格  TbSpecification tbSpecification = specificationMapper.selectByPrimaryKey(id);  //查询规格选项列表  TbSpecificationOptionExampleexample=**new**TbSpecificationOptionExample();  Criteria criteria = example.createCriteria();  criteria.andSpecIdEqualTo(id);//根据规格ID查询  List<TbSpecificationOption>optionList = specificationOptionMapper.selectByExample(example);  //构建组合实体类返回结果  Specification spec=**new** Specification();  spec.setSpecification(tbSpecification);  spec.setSpecificationOptionList(optionList);  **return**spec;  } |

（3）修改pinyougou-manager-web的SpecificationController.java

|  |
| --- |
| @RequestMapping("/findOne")  public Specification findOne(Long id){  return specificationService.findOne(id);  } |

1. 修改页面specification.html 中列表的修改按钮

|  |
| --- |
| <button type=*"button"* class=*"btnbg-olive btn-xs"* data-toggle=*"modal"* data-target=*"#editModal"*ng-click=*"findOne(entity.id)"*>修改</button> |

### **4.4.2 保存修改结果**

（1）修改pinyougou-sellergoods-interface的SpecificationService.java

|  |
| --- |
| /\*\*  \* 修改  \*/  **public void** update(Specification specification); |

（2）修改pinyougou-sellergoods-service的SpecificationServiceImpl.java

|  |
| --- |
| /\*\*  \* 修改  \*/  @Override  **publicvoid** update(Specification specification){  //保存修改的规格  specificationMapper.updateByPrimaryKey(specification.getSpecification());//保存规格  //删除原有的规格选项  TbSpecificationOptionExampleexample=**new**TbSpecificationOptionExample();  com.pinyougou.pojo.TbSpecificationOptionExample.Criteriacriteria = example.createCriteria();  criteria.andSpecIdEqualTo(specification.getSpecification().getId());//指定规格ID为条件  specificationOptionMapper.deleteByExample(example);//删除  //循环插入规格选项  **for**(TbSpecificationOptionspecificationOption:specification.getSpecificationOptionList()){  specificationOption.setSpecId(specification.getSpecification().getId());  specificationOptionMapper.insert(specificationOption);  }  } |

（3）修改pinyougou-manager-web的SpecificationController.java

|  |
| --- |
| /\*\*  \* 修改  \* **@param** specification  \* **@return**  \*/  @RequestMapping("/update")  **public** Result update(@RequestBody **Specification** specification){  **try** {  specificationService.update(specification);  **returnnew** Result(**true**, "修改成功");  } **catch** (Exception e) {  e.printStackTrace();  **returnnew** Result(**false**, "修改失败");  }  } |

（4）修改specification.js的save方法

|  |
| --- |
| //保存  $scope.save=**function**(){  **var**serviceObject;//服务层对象  **if**($scope.entity.specification.id!=**null**){//如果有ID  serviceObject=specificationService.update( $scope.entity ); //修改  }**else**{  serviceObject=specificationService.add( $scope.entity );//增加  }  serviceObject.success(  **function**(response){  **if**(response.success){  //重新查询  $scope.reloadList();//重新加载  }**else**{  alert(response.message);  }  }  );  } |

## 4.5删除规格

实现思路：我们要删除规格的同时，还要记得将关联的规格选项删除掉。

### **4.5.1 后端代码**

修改pinyougou-sellergoods-service的SpecificationServiceImpl.java

|  |
| --- |
| /\*\*  \* 批量删除  \*/  @Override  **publicvoid** delete(Long[] ids) {  **for**(Long id:ids){  specificationMapper.deleteByPrimaryKey(id);  //删除原有的规格选项  TbSpecificationOptionExample example=**new**TbSpecificationOptionExample();  com.pinyougou.pojo.TbSpecificationOptionExample.Criteria criteria = example.createCriteria();  criteria.andSpecIdEqualTo(id);//指定规格ID为条件  specificationOptionMapper.deleteByExample(example);//删除  }  } |

### **4.5.2 前端代码**

修改pinyougou-manager-web的specification.html

列表的复选框

|  |
| --- |
| <input type=*"checkbox"***ng-click=*"updateSelection($event,entity.id)"***> |

删除按钮

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

# 5.模板管理

## 5.1 需求及表结构分析

### **5.1.1 需求分析**

首选我们需要理解模板的作用。模板主要有两个：

1是用于关联品牌与规格

2定义扩充属性

### **5.1.2 表结构分析**

**tb\_type\_template模板表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 含义 |
| Id | Bigint |  | 主键 |
| Name | Varchar | 80 | 模板名称 |
| Spec\_ids | Varchar | 1000 | 关联规格（json格式） |
| brand\_ids | Varchar | 1000 | 关联品牌（json格式） |
| custom\_attribute\_items | Varchar | 2000 | 扩展属性 |

## 5.2 模板列表

### **5.2.1 引入JS**

修改type\_template.html ，引入JS

|  |
| --- |
| <**script type="text/javascript" src="../plugins/angularjs/angular.min.js"**></**script**>  <**script src="../plugins/angularjs/pagination.js"**></**script**> <**link rel="stylesheet" href="../plugins/angularjs/pagination.css"**>  <**script type="text/javascript" src="../js/base\_pagination.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/typeTemplateController.js"**></**script**> |

### **5.2.2 放置分页组件**

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

### **5.2.3 指令与表达式**

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

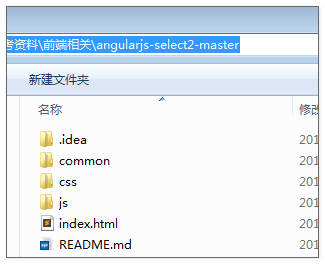
|  |
| --- |
| <tr ng-repeat=*"entity in list"*>  <td><inputtype=*"checkbox"*></td>  <td>{{entity.id}}</td>  <td>{{entity.name}}</td>  <td>{{entity.brandIds}}</td>  <td>{{entity.specIds}}</td>  <td>{{entity.customAttributeItems}}</td>  <tdclass=*"text-center"*>  <buttontype=*"button"*class=*"btnbg-olive btn-xs"*data-toggle=*"modal"*data-target=*"#editModal"*>修改</button>  </td>  </tr> |

## 5.3 品牌下拉列表

在弹出窗口中有个品牌下拉列表，要求品牌是可以选择多个，这与我们之前的单选的下拉列表是不同的。我们要想实现这个功能，需要使用**select2** 组件来完成。

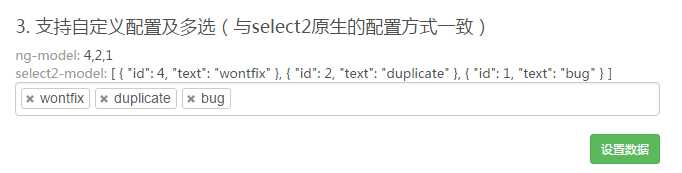


### **5.2.1 认识select2**



https://github.com/think2011/angularjs-select2

我们来看例子：我们需要的就是这样可以多选的下拉框



### **5.2.2 显示品牌下拉列表（静态）**

（1）修改 type\_template.html 引入JS

|  |
| --- |
| <linkrel=*"stylesheet"*href=*"../plugins/select2/select2.css"*/>  <linkrel=*"stylesheet"*href=*"../plugins/select2/select2-bootstrap.css"* />  <script src=*"../plugins/select2/select2.min.js"* type=*"text/javascript"*></script>  <**script type="text/javascript" src="../plugins/angular-select2.js"**></**script**> |

1. 修改typeTemplateController.js ，定义品牌列表数据

|  |
| --- |
| $scope.brandList={data:[{id:1,text:'联想'},{id:2,text:'华为'},{id:3,text:'小米'}]};//品牌列表 |

1. 在type\_template.html 用select2组件实现多选下拉框

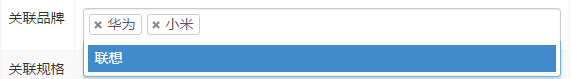
|  |
| --- |
| <**input select2 config="brandList" select2-model="entity.brandIds" multiple placeholder="选择品牌（可多选）" class="form-control" type="text"**/> |

multiple 表示可多选

Config用于配置数据来源

select2-model用于指定用户选择后提交的变量

最终实现效果如下：



### **5.2.3 后端数据支撑**

我们现在让这个下拉列表的数据从数据库中提取，修改后端代码 修改了dao需要重新安装。

（1）pinyougou-dao工程，在TbBrandMapper.xml中添加SQL语句配置

|  |
| --- |
| <select id=*"selectOptionList"* resultType=*"java.util.Map"*>  select id,name as text from tb\_brand  </select> |

（2）在pinyougou-dao的TbBrandMapper中添加方法定义

|  |
| --- |
| List<Map>selectOptionList(); |

（3）修改pinyougou-sellergoods-interface 的BrandService.java，增加方法定义

|  |
| --- |
| /\*\*  \* 品牌下拉框数据  \*/  List<Map>selectOptionList(); |

（4）修改pinyougou-sellergoods-service的BrandServiceImpl.java，增加方法

|  |
| --- |
| /\*\*  \* 列表数据  \*/  **public** List<Map>selectOptionList() {  **return** brandMapper.selectOptionList();  } |

（5）修改pinyougou-manager-web的BrandController.java

|  |
| --- |
| @RequestMapping("/selectOptionList")  **public** List<Map> selectOptionList(){  **return**brandService.selectOptionList();  } |

1. 修改pinyougou-manager-web的brandService.js

|  |
| --- |
| //下拉列表数据  **this**.selectOptionList=**function**(){  **return** $http.get('../brand/selectOptionList.do');  } |

1. 修改pinyougou-manager-web的typeTemplateController.js

因为我们在模板控制层中需要使用品牌服务层的方法，所以需要添加**依赖注入**

|  |
| --- |
| //控制层  app.controller('typeTemplateController' ,**function**($scope,$controller ,typeTemplateService,brandService){ |

使用品牌服务方法实现查询，结果赋给变量

|  |
| --- |
| $scope.brandList={data:[]};//品牌列表  //读取品牌列表  $scope.findBrandList=**function**(){  brandService.selectOptionList().success(  **function**(response){  $scope.brandList={data:response};  }  );  } |

（8）修改type\_template.html ，添加JS引入

|  |
| --- |
| <script type=*"text/javascript"*src=*"../js/base\_pagination.js"*></script>  <script type=*"text/javascript"*src=*"../js/service/typeTemplateService.js"*></script>  **<script type=*"text/javascript"*src=*"../js/service/brandService.js"*></script>**  <script type=*"text/javascript"*src=*"../js/controller/baseController.js"*></script>  <script type=*"text/javascript"*src=*"../js/controller/typeTemplateController.js"*></script> |

特别注意一下，JS引入的位置，要在typeTemplateController.js之前，因为该控制器要使用到它

（9）修改type\_template.html ，添加初始化

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

## 5.4 规格下拉列表



参考品牌下拉。

html:,注意引入specificationService.js

|  |
| --- |
| <**td**>关联规格</**td**> <**td**>  <**input select2 select2-model="entity.specIds" config="specificationList" multiple placeholder="选择规格（可多选）" class="form-control" type="text"**/> </**td**> |

typeTemplateController.JS:先注入specificationService.js

|  |
| --- |
| *//规格列表* $scope.**specificationList**={**data**:[]};   $scope.findSpecificationList=**function**(){  specificationService.selectOptionList().success(  **function** (response) {  $scope.**specificationList**={**data**:response};  }  ) } |

specificationService.js:

|  |
| --- |
| **this**.selectOptionList=**function** () {  **return** $http.get(**'../specification/selectOptionList.do'**);  } |

specificationController.java:

|  |
| --- |
| @RequestMapping(**"/selectOptionList"**) **public** List<Map> selectOptionList(){  **return specificationService**.selectOptionList(); } |

specificationServiceImpl.java: 注意接口也需要改

|  |
| --- |
| @Override **public** List<Map> selectOptionList() {  **return specificationMapper**.selectOptionList(); } |

specificationMapper.java:

接口：

List<Map> selectOptionList();

xml配置文件：

<**select id="selectOptionList" resultType="java.util.Map"**>  
 select id,spec\_name as text from tb\_specification  
</**select**>

## 5.5 扩展属性

### **5.5.1 增加行**

在typeTemplateController.js中新增代码

|  |
| --- |
| //新增扩展属性行  $scope.addTableRow=**function**(){  $scope.entity.customAttributeItems.push({});  } |

在type\_template.html中的“新建”按钮，执行实体的初始化操作

|  |
| --- |
| <button type=*"button"* class=*"btnbtn-default"* title=*"新建"* data-toggle=*"modal"* data-target=*"#editModal"***ng-click=*"entity={customAttributeItems:[]}"***><i class=*"fafa-file-o"*></i> 新建</button> |

修改“新增扩展属性按钮”

|  |
| --- |
| <button type=*"button"* class=*"btnbtn-default"* title=*"新增扩展属性"***ng-click=*"addTableRow()"***><i class=*"fafa-file-o"*></i> 新增扩展属性</button> |

循环表格

|  |
| --- |
| <trng-repeat=*"pojo in entity.customAttributeItems"*>  <td><inputclass=*"form-control"*ng-model=*"pojo.text"*placeholder=*"属性名称"*></td>  <td><buttontype=*"button"*class=*"btnbtn-default"*title=*"删除"*><iclass=*"fafa-trash-o"*></i> 删除</button></td>  </tr> |

### **5.5.2 删除行**

实现思路：在每一行将索引值传递给集合，在集合中删除。

修改typeTemplateController.js新增以下代码

|  |
| --- |
| //删除扩展属性行  $scope.deleTableRow=**function**(index){  $scope.entity.customAttributeItems.splice(index,1);//删除  } |

修改每行的删除按钮

|  |
| --- |
| <button type=*"button"*ng-click=*"deleTableRow($index)"* class=*"btnbtn-default"* title=*"删除"*><i class=*"fafa-trash-o"*></i> 删除</button> |

$index 用于获取ng-repeat指令循环中的索引。

## 5.6 新增模板

修改type\_template.html ，绑定文本框

|  |
| --- |
| <tr>  <td>模板名称</td>  <td><inputng-model=*"entity.name"*class=*"form-control"*placeholder=*"模板名称"*></td>  </tr> |

保存按钮

|  |
| --- |
| <buttonclass=*"btnbtn-success"*data-dismiss=*"modal"*aria-hidden=*"true"*ng-click=*"save()"*>保存</button> |

## 5.7 修改模板

测试是，无法获取数据，因为从后台获取到的是字符串 而不是JSON对象，需要转成JSON.修改typeTemplateController.js的findOne方法

|  |
| --- |
| //查询实体  $scope.findOne=**function**(id){  typeTemplateService.findOne(id).success(  **function**(response){  $scope.entity= response;  $scope.entity.brandIds= JSON.parse($scope.entity.brandIds);//转换品牌列表  $scope.entity.specIds= JSON.parse($scope.entity.specIds);//转换规格列表  $scope.entity.customAttributeItems= JSON.parse($scope.entity.customAttributeItems);//转换扩展属性  }  );  } |

从数据库中查询出来的是字符串，我们必须将其转换为json对象才能实现信息的回显。

或者也可以这么转换：

$scope.**entity**.**brandIds**=**angular**.*fromJson*($scope.**entity**.**brandIds**);  
$scope.**entity**.**specIds**=**angular**.*fromJson*($scope.**entity**.**specIds**);  
$scope.**entity**.**customAttributeItems**=**angular**.*fromJson*($scope.**entity**.**customAttributeItems**);

## 5.8 删除模板

修改type\_template.html

表格中的复选框

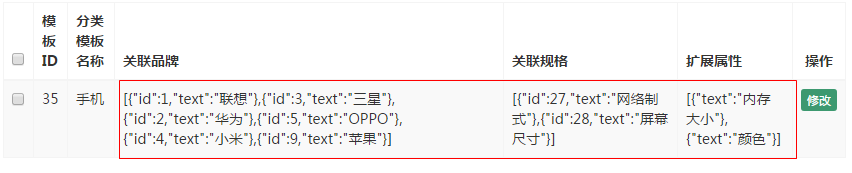
|  |
| --- |
| <input type=*"checkbox"*ng-click=*"updateSelection($event,entity.id)"*> |

删除按钮

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

## 5.9 优化模板列表的显示

我们现在完成的列表中都是以JSON格式显示的，不利于用户的查询。



我们需要将信息以更友好的方式展现出来,如下图形式



我们需要将一个json字符串中某个属性的值提取出来，用逗号拼接成一个新的字符串。这样的功能比较常用，所以我们将方法写到baseController.js

|  |
| --- |
| //提取json字符串数据中某个属性，返回拼接字符串 逗号分隔  $scope.jsonToString=**function**(jsonString,key){  **var** json=JSON.parse(jsonString);//将json字符串转换为json对象  **var** value="";  **for**(**var** i=0;i<json.length;i++){  **if**(i>0){  value+=","  }  value+=json[i][key];  }  **return** value;  } |

页面上使用该函数进行转换 或者：

|  |
| --- |
| *//从给定的json对象中（数组）查询出给定的属性的值 拼接逗号 返回  /\*\*  \*  \** ***@param*** *list [{"id":27,"text":"网络"},{"id":32,"text":"机身内存"}]  \** ***@param*** *key text 就是list中要找的属性  \*/* $scope.jsonToString=**function** (list,key) {  *//将字符串 转成JSON对象（数组）* **var** fromJson = **angular**.*fromJson*(list);*// [{"id":27,"text":"网络"},{"id":32,"text":"机身内存"}]* **var** str=**""**;  **for**(**var** i=0;i<fromJson.**length**;i++){  **var** obj = fromJson[i];*//{"id":27,"text":"网络"}* str+=obj[key]+**","**;  *//逻辑就是转换并拼接* }  **if**(str.**length**>=1){  str = str.substring(0,str.**length**-1);  }    **return** str;   } |

|  |
| --- |
| <trng-repeat=*"entity in list"*>  <td><input type=*"checkbox"*ng-click=*"updateSelection($event,entity.id)"*></td>  <td>{{entity.id}}</td>  <td>{{entity.name}}</td>  <td>{{jsonToString(entity.brandIds,'text')}}</td>  <td>{{jsonToString(entity.specIds,'text')}}</td> <td>{{jsonToString(entity.customAttributeItems,'text')}}</td>  <td class=*"text-center"*>  <button type=*"button"* class=*"btnbg-olive btn-xs"* data-toggle=*"modal"*data-target=*"#editModal"*ng-click=*"findOne(entity.id)"*>修改</button>  </td>  </tr> |