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。

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

```
<script type="text/javascript" src="../js/base_pagination.js"></script>

<script type="text/javascript" src="../js/service/brandService.js"></script>

<script type="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{
            varidx = $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

```
//品牌控制层
app.controller('brandController' ,function($scope,$controller,brandService){
    $controller('baseController',{$scope:$scope});//继承
//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.....
});
```

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

3.代码生成器

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

（3）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)

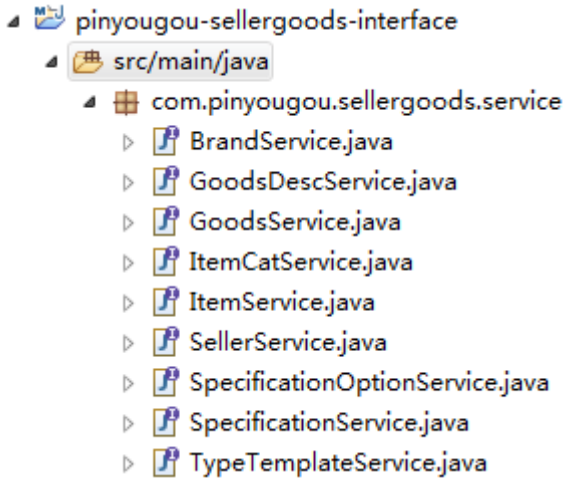
这个模板不会生成数据访问层和实体类，因为我们之前已经用逆向工程完成了数据访问层与实体类的生成。

（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。

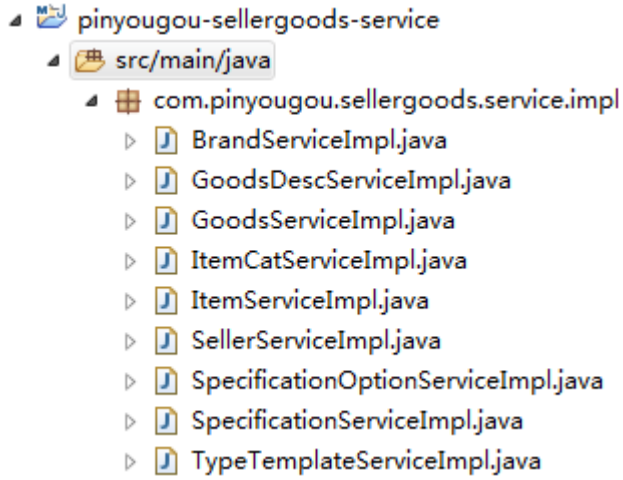
3.2 代码拷贝

将商家商品相关代码拷贝到工程。

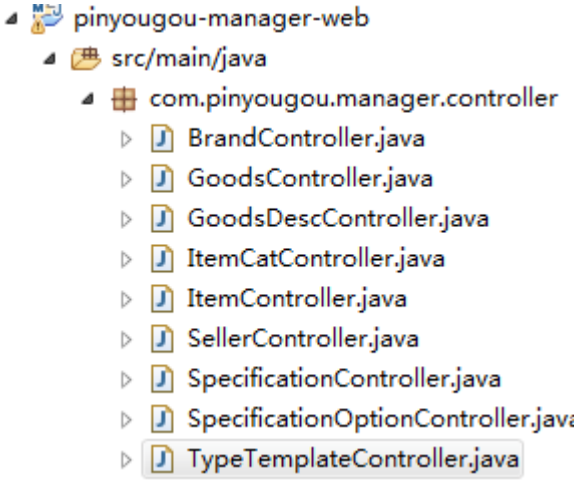
（1）拷贝服务接口



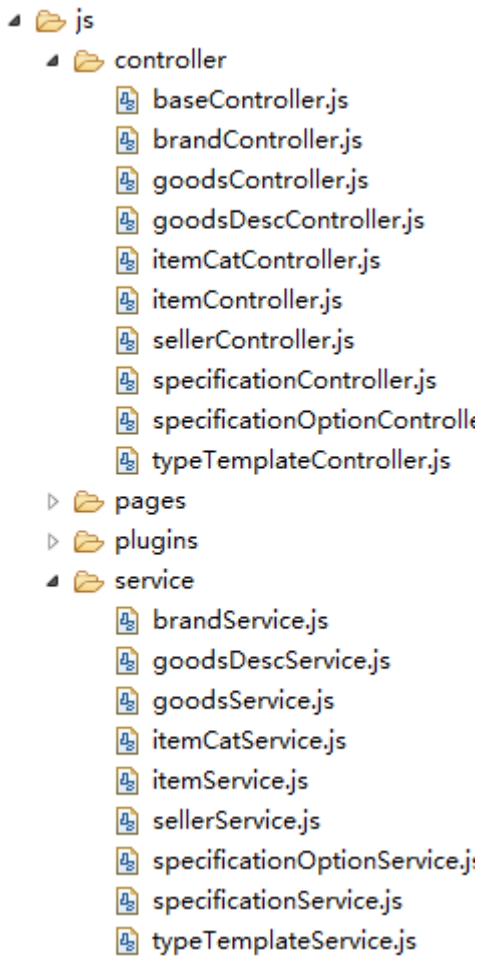
（2）拷贝服务实现类



（3）拷贝控制器



（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

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

```
<scripttype="text/javascript"src="../plugins/angularjs/angular.min.js"></script>
<scriptsrc="../plugins/angularjs/pagination.js"></script>
<linkrel="stylesheet"href="../plugins/angularjs/pagination.css">
<scripttype="text/javascript"src="../js/base_pagination.js"></script>
<scripttype="text/javascript"src="../js/service/specificationService.js"></script>
<scripttype="text/javascript"src="../js/controller/baseController.js"></script>
<scripttype="text/javascript"src="../js/controller/specificationController.js"></sc
ript>
```

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

```
<tr ng-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

```
<tr ng-repeat="pojo in entity.specificationOptionList">
<td><input type="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

```
<insert id="insert"parameterType="com.pinyougou.pojo.TbSpecification">
    <selectKeyresultType="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>
```

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

```
packagecom.pinyougou.pojogroup;

importjava.io.Serializable;

importjava.util.List;

importcom.pinyougou.pojo.TbSpecification;

importcom.pinyougou.pojo.TbSpecificationOption;

/**

 * 规格组合实体类
```

```
 * @author Administrator
 *
 */
publicclassSpecificationimplementsSerializable {
    privateTbSpecificationspecification;
    private List<TbSpecificationOption>specificationOptionList;

    publicTbSpecificationgetSpecification() {
        returnspecification;
    }
    publicvoidsetSpecification(TbSpecificationspecification) {
        this.specification = specification;
    }
    public List<TbSpecificationOption>getSpecificationOptionList() {
        returnspecificationOptionList;
    }
    publicvoidsetSpecificationOptionList(List<TbSpecificationOption>specificationOp
tionList) {
        this.specificationOptionList = specificationOptionList;
    }
}
```

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

```
    /**
```

```
     * 增加
    */
    publicvoid add(Specification specification);
```

（4）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.getSpecificationOpti
onList()){
    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);
            returnnew Result(true, "增加成功");
        } catch (Exception e) {
            e.printStackTrace();
            returnnew Result(false, "增加失败");
        }
    }
```

（6）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
 */
publicSpecificationfindOne(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
    TbSpecificationtbSpecification = specificationMapper.selectByPrimaryKey(id);
```

```
        //查询规格选项列表
        TbSpecificationOptionExampleexample=newTbSpecificationOptionExample();
        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);
        returnspec;
    }
```

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

```
    @RequestMapping("/findOne")
    public Specificationfind One(Long id){
        return specificationService.findOne(id);
    }
```

（4）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
     */
    publicvoid 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=newTbSpecificationOptionExample();
        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.getSpecificationOpti
onList()){

    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(@RequestBodySpecification 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
    $scope.save=function(){
        varserviceObject;//服务层对象
        if($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
            TbSpecificationOptionExampleexample=newTbSpecificationOptionExample();
            com.pinyougou.pojo.TbSpecificationOptionExample.Criteriacriteria =
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

```
<scripttype="text/javascript"src="../plugins/angularjs/angular.min.js"></script>
<scriptsrc="../plugins/angularjs/pagination.js"></script>
<linkrel="stylesheet"href="../plugins/angularjs/pagination.css">

<scripttype="text/javascript"src="../js/base_pagination.js"></script>
<scripttype="text/javascript"src="../js/service/typeTemplateService.js"></script>
<scripttype="text/javascript"src="../js/controller/baseController.js"></script>
<scripttype="text/javascript"src="../js/controller/typeTemplateController.js"></scr
ipt>
```

5.2.2 放置分页组件

```
<tm-paginationconf="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">
```

```
<trng-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>
```

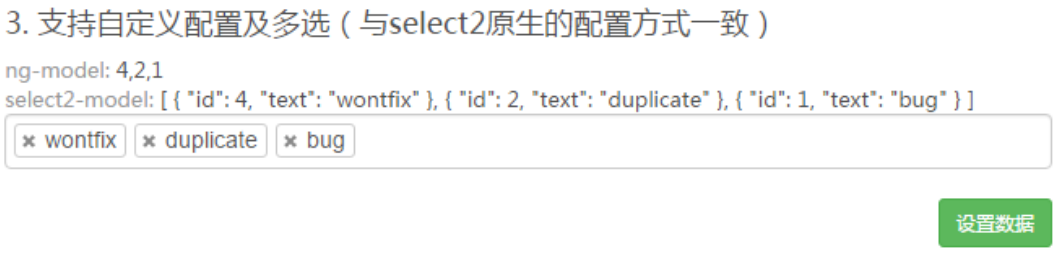
```
        <tdclass="text-center">

    <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

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" />

<scriptsrc="../plugins/select2/select2.min.js" type="text/javascript"></script>

<scripttype="text/javascript"src="../js/angular-select2.js"></script>
```

（2）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
```

（3）在 type_template.html 用 select2 组件实现多选下拉框

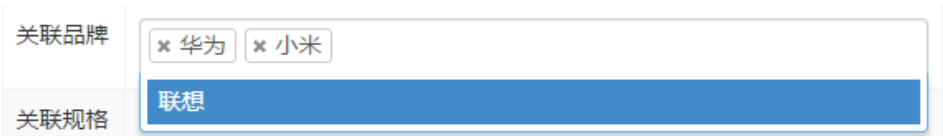
```
<input select2 select2-model="entity.brandIds"config="brandList"multipleplaceholder="选择品牌（可多选）"class="form-control"type="text"/>
```

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

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

```
<selectid="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() {
        returnbrandMapper.selectOptionList();
    }
```

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

```
    @RequestMapping("/selectOptionList")
    public List<Map>selectOptionList(){
        returnbrandService.selectOptionList();
    }
```

（6）修改 pinyougou-manager-web 的 brandService.js

```
    //下拉列表数据
    this.selectOptionList=function(){
```

```
        return $http.get('../brand/selectOptionList.do');
    }
```

（7）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

（代码略，参照品牌下拉列表的实现步骤）

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

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。

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。

	模板ID	分类模板名称	关联品牌	关联规格	扩展属性	操作
	35	手机	[{"id":1,"text":"联想"},{"id":3,"text":"三星"},{"id":2,"text":"华为"},{"id":5,"text":"OPPO"},{"id":4,"text":"小米"},{"id":9,"text":"苹果"}]	[{"id":27,"text":"网络制式"},{"id":28,"text":"屏幕尺寸"}]	[{"text":"内存大小"},{"text":"颜色"}]	修改

我们需要将信息以更友好的方式展现出来,如下图形式

	模板ID	分类模板名称	关联品牌	关联规格	扩展属性	操作
	35	手机	联想,三星,华为,OPPO,小米,苹果	网络制式,屏幕尺寸	内存大小,颜色	修改

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){
    varjson=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

```
<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>
```