**品优购电商系统开发**

**第4章**

**模版管理&商品分类&FastDFS**

传智播客.黑马程序员

# 课程目标

目标1：完成模版管理功能

目标2：完成商品分类功能

目标3：掌握FastDFS的基本理论

# 1.模板管理

## 1.1 需求及表结构分析

### **1.1.1 需求分析**

实现模板管理CRUD功能

首选我们需要理解模板的作用。

作用：当添加商品时，需要选择商品分类，商品分类关联了模版，这样就知道该商品所属品牌及规格、扩展属性。

模板主要用于(模板表实际是 品牌与规格的中间表，用于组装品牌与规格)：

1. 用于关联品牌

2、用于关联规格

3、定义扩展属性



### **1.1.2 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（json格式） |

## 1.2 模板列表

### **1.2.1 引入JS**

在pinyougou-manager-web/src/main/webapp/admin/type\_template.html，引入JS：

|  |
| --- |
| <**script src="/plugins/angularjs/angular.min.js"**></**script**> <**script src="/plugins/angularjs/pagination.js"**></**script**> <**link rel="stylesheet" href="/plugins/angularjs/pagination.css"**/> <**script src="/js/base-pagination.js"**></**script**> <**script src="/js/service/baseService.js"**></**script**> <**script src="/js/controller/baseController.js"**></**script**> <**script src="/js/controller/typeTemplateController.js"**></**script**> |

### **1.2.2 放置分页指令**

type\_template.html页面<table>结束标签后

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

### **1.2.3 指令与表达式**

type\_template.html页面<body>标签中

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

用ng-repeat迭代数据：

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

pinyougou-manager-web/src/main/webapp/js/controller/typeTemplateController.js：

|  |
| --- |
| */\*\* 定义搜索对象 \*/* $scope.**searchEntity** = {}; */\*\* 分页查询类型模版信息 \*/* $scope.search = **function**(page, rows){  */\*\* 调用服务层分页查询类型模版数据 \*/* baseService.findByPage(**"/typeTemplate/findByPage"**,page,  rows,$scope.**searchEntity**)  .then(**function**(response){  $scope.**dataList** = response.data.rows;  */\*\* 更新总记录数 \*/* $scope.**paginationConf**.**totalItems** = response.data.total;  }); }; |

type\_template.html页面搜索条件：

|  |
| --- |
| <**div class="has-feedback"**>  <**input ng-model="searchEntity.name"  placeholder="分类模板名称"**/>  <**button class="btn btn-default"  ng-click="reload()"**>查询</**button**> </**div**> |

### **1.2.4 后台代码**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller包下创建TypeTemplateController.java：

|  |
| --- |
| @RestController @RequestMapping(**"/typeTemplate"**) **public class** TypeTemplateController {  @Reference(timeout = 10000)  **private** TypeTemplateService **typeTemplateService**;  */\*\* 分页查询类型模版 \*/* @GetMapping(**"/findByPage"**)  **public** PageResult findByPage(TypeTemplate typeTemplate,  Integer page, Integer rows){  */\*\* GET请求中文转码 \*/* **if** (typeTemplate != **null** &&  StringUtils.*isNoneBlank*(typeTemplate.getName())){  **try** {  typeTemplate.setName(**new** String(typeTemplate.getName()  .getBytes(**"ISO8859-1"**),**"UTF-8"**));  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return typeTemplateService**.findByPage(typeTemplate, page, rows);  } } |

pinyougou-interface/src/main/java/com.pinyougou.service/TypeTemplateService.java：

|  |
| --- |
| */\*\* 分页查询类型模版 \*/* PageResult findByPage(TypeTemplate typeTemplate,**int** page,**int** rows); |

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl包下创建TypeTemplateServiceImpl.java：

|  |
| --- |
| @Service(interfaceName = **"com.pinyougou.service.TypeTemplateService"**) @Transactional **public class** TypeTemplateServiceImpl **implements** TypeTemplateService {  */\*\* 注入数据访问接口代理对象 \*/* @Autowired  **private** TypeTemplateMapper **typeTemplateMapper**;  @Override  **public void** save(TypeTemplate typeTemplate) {  }  @Override  **public void** update(TypeTemplate typeTemplate) {  }  @Override  **public void** delete(Serializable id) {  }  @Override  **public void** deleteAll(Serializable[] ids) {  }  @Override  **public** TypeTemplate findOne(Serializable id) {  **return null**;  }  @Override  **public** List<TypeTemplate> findAll() {  **return null**;  }  */\*\* 分页查询类型模版 \*/* @Override  **public** PageResult findByPage(TypeTemplate typeTemplate, **int** page, **int** rows) {  **try**{  */\*\* 开始分页 \*/* PageInfo<TypeTemplate> pageInfo = PageHelper.*startPage*(page, rows)  .doSelectPageInfo( **new** ISelect() {  @Override  **public void** doSelect() {  **typeTemplateMapper**.findAll(typeTemplate);  }  });  **return new** PageResult(pageInfo.getTotal(), pageInfo.getList());  }**catch**(Exception ex){  **throw new** RuntimeException(ex);  }  } } |

pinyougou-mapper/src/main/java/com.pinyougou.mapper/TypeTemplateMapper.java：

|  |
| --- |
| */\*\* 多条件查询类型模版 \*/* List<TypeTemplate> findAll(TypeTemplate typeTemplate); |

pinyougou-mapper/src/main/resources/mappers/TypeTemplateMapper.xml：

|  |
| --- |
| *<!-- 多条件查询类型模版 -->* <**select id="findAll" resultType="TypeTemplate"**>  select \* from tb\_type\_template  <**where**>  <**if test="name != null and name != ''"**>  name like CONCAT('%',#{name},'%')  </**if**>  </**where**>  order by id asc </**select**> |

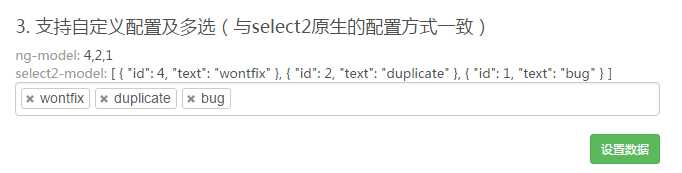
## 1.3 品牌下拉列表

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。



### **1.3.1 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



### **1.3.2 显示品牌下拉列表（静态）**

修改pinyougou-manager-web/src/main/webapp/admin/type\_template.html引入JS(放在angularjs后面)

|  |
| --- |
| <**link rel="stylesheet" href="/plugins/select2/select2.css"**/> <**link rel="stylesheet" href="/plugins/select2/select2-bootstrap.css"**/> <**script src="/plugins/select2/select2.min.js"**></**script**> <**script src="/plugins/select2/angular-select2.js"**></**script**> |

修改pinyougou-manager-web/src/main/webapp/js/controller的

typeTemplateController.js，定义品牌列表数据

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

在pinyougou-manager-web/src/main/webapp/admin/type\_template.html 用select2组件实现多选下拉框

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

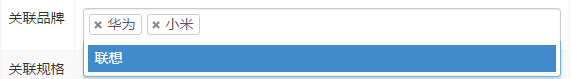
multiple：表示可多选

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

select2：控制样式

最终实现效果如下：



### **1.3.3前端代码**

pinyougou-manager-web/src/main/webapp/js/controller/typeTemplateController.js

|  |
| --- |
| */\*\* 品牌列表 \*/* $scope.findBrandList = **function**(){  baseService.sendGet(**"/brand/findBrandList"**)  .then(**function**(response){  $scope.**brandList** = {**data**:response.data};  }); }; |

pinyougou-manager-web/src/main/webapp/admin/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()"**> |

### **1.3.4后端代码**

我们现在让这个下拉列表的数据从数据库中提取，修改

pinyougou-manager-web/src/main/java/com.pinyougou.manager.

controller的BrandController.java，增加方法：

|  |
| --- |
| */\*\* 查询所有的品牌 \*/* @GetMapping(**"/findBrandList"**) **public** List<Map<String, Object>> findBrandList(){  **return brandService**.findAllByIdAndName(); } |

pinyougou-interface/src/main/java/com.pinyougou.service/BrandService.java，增加方法

|  |
| --- |
| */\*\* 查询所有的品牌(id与name) \*/* List<Map<String, Object>> findAllByIdAndName(); |

修改pinyougou-sellergoods-service/src/main/java/com.pinyougou.

sellergoods.service.impl的BrandServiceImpl.java，增加方法

|  |
| --- |
| */\*\* 查询所有的品牌(id与name) \*/* **public** List<Map<String, Object>> findAllByIdAndName(){  **try**{  **return brandMapper**.findAllByIdAndName();  }**catch**(Exception ex){  **throw new** RuntimeException(ex);  } } |

在pinyougou-mapper/src/main/java/com.pinyougou.mapper

的BrandMapper.java中添加方法：

|  |
| --- |
| */\*\* 查询所有的品牌(id与name) \*/* @Select(**"select id, name as text from tb\_brand order by id asc"**) List<Map<String,Object>> findAllByIdAndName(); |

## 1.4 规格下拉列表【参考品牌下拉列表】



### **1.4.1前端代码**

|  |
| --- |
| <**input select2 select2-model="entity.specIds"  config="specList" multiple placeholder="选择规格（可多选）"  class="form-control" type="text"**/>  <**body class="hold-transition skin-red sidebar-mini"  ng-app="pinyougou" ng-controller="typeTemplateController"  ng-init="findBrandList();findSpecList();"**>  */\*\* 规格列表 \*/* $scope.findSpecList = **function**(){  baseService.sendGet(**"/specification/findSpecList"**)  .then(**function**(response){  $scope.**specList** = {**data**:response.data};  }); }; |

### **1.4.2后端代码**

|  |
| --- |
| SpecificationController.java：  */\*\* 查询所有的规格(id与specName) \*/* @GetMapping(**"/findSpecList"**) **public** List<Map<String, Object>> findSpecList(){  **return specificationService**.findAllByIdAndName(); }  SpecificationService.java：  */\*\* 查询所有的规格(id与specName) \*/* List<Map<String, Object>> findAllByIdAndName();  SpecificationServiceImpl.java：  */\*\* 查询所有的规格(id与specName) \*/* **public** List<Map<String, Object>> findAllByIdAndName(){  **try**{  **return specificationMapper**.findAllByIdAndName();  }**catch**(Exception ex){  **throw new** RuntimeException(ex);  } }  SpecificationMapper.java：  */\*\* 查询所有的规格(id与specName) \*/* @Select(**"select id, spec\_name as text from tb\_specification order by id asc"**) List<Map<String, Object>> findAllByIdAndName(); |

## 1.5 扩展属性

### **1.5.1 增加行**

修改type\_template.html页面“新增扩展属性按钮”，绑定点击事件

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

在pinyougou-manager-web/src/main/webapp/js/controller的

typeTemplateController.js中新增代码

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

在pinyougou-manager-web/src/main/webapp/admin/type\_template.html中的“新建”按钮，执行实体的初始化操作(48行)

|  |
| --- |
| <**button type="button" class="btn btn-default" title="新建"  data-toggle="modal" data-target="#editModal"  ng-click="entity.customAttributeItems=[];"**>  <**i class="fa fa-file-o"**></**i**> 新建 </**button**> |

用ng-repeat循环行

|  |
| --- |
| <**tr ng-repeat="c in entity.customAttributeItems"**>  <**td**><**input class="form-control" ng-model="c.text" placeholder="属性名称"**></**td**>  <**td**><**button type="button" class="btn btn-default" title="删除"**>  <**i class="fa fa-trash-o"**></**i**> 删除</**button**>  </**td**> </**tr**> |

### **1.5.2 删除行**

修改每行的删除按钮，绑定点击事件

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

说明：$index 用于获取ng-repeat指令循环中的索引。

修改pinyougou-manager-web/src/main/webapp/js/controller的

typeTemplateController.js新增以下代码

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

## 1.6 新增模板

### **1.6.1前端代码**

pinyougou-manager-web/src/main/webapp/admin/type\_template.html，绑定文本框

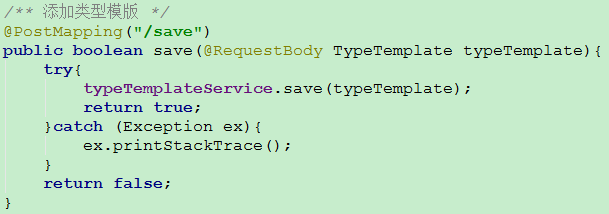
|  |
| --- |
| <**td**><**input class="form-control" ng-model="entity.name" placeholder="模版名称"**></**td**> |

保存按钮，绑定点击事件

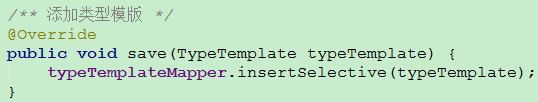
|  |
| --- |
| <**button class="btn btn-success" data-dismiss="modal"  ng-click="saveOrUpdate()" aria-hidden="true"**>保存</**button**> |

### **1.6.2后端代码**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller的TypeTemplateController.java，增加方法：



pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl的TypeTemplateServiceImpl.java



## 1.7 修改模板

### **1.7.1前端代码**

修改pinyougou-manager-web/src/main/webapp/admin/type\_template.html

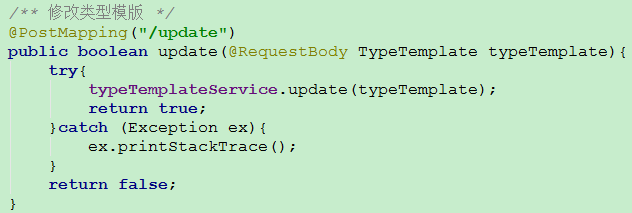
|  |
| --- |
| <**button type="button" class="btn bg-olive btn-xs"  data-toggle="modal" data-target="#editModal"  ng-click="show(entity);"**>修改</**button**> |

pinyougou-manager-web/src/main/webapp/js/controller/typeTemplateController.js

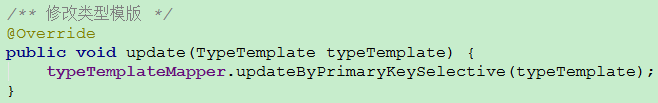
|  |
| --- |
| */\*\* 显示修改 \*/* $scope.show = **function**(entity){  $scope.**entity** = ***JSON***.parse(***JSON***.stringify(entity));  */\*\* 转换品牌列表 \*/* $scope.**entity**.**brandIds** = ***JSON***.parse(entity.**brandIds**);  */\*\* 转换规格列表 \*/* $scope.**entity**.**specIds** = ***JSON***.parse(entity.**specIds**);  */\*\* 转换扩展属性 \*/* $scope.**entity**.**customAttributeItems** = ***JSON*** .parse(entity.**customAttributeItems**); }; |

### **1.7.2后台代码**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller/TypeTemplateController.java，增加方法：



pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl/TypeTemplateServiceImpl.java



## 1.8 删除模板

### **1.8.1前端代码**

修改pinyougou-manager-web/src/main/webapp/admin/type\_template.html

表格中的复选框，绑定点击事件

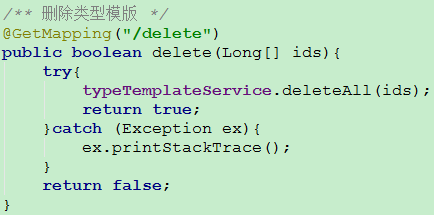
|  |
| --- |
| <**td**><**input type="checkbox"**  **ng-click="updateSelection($event,entity.id)"**></**td**> |

type\_template.html删除按钮，绑定点击事件

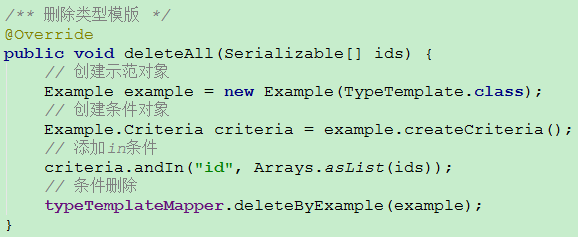
|  |
| --- |
| <**button type="button" class="btn btn-default" title="删除"  ng-click="delete()"**>  <**i class="fa fa-trash-o"**></**i**> 删除 </**button**> |

### **1.8.2后台代码**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller/TypeTemplateController.java，增加方法：

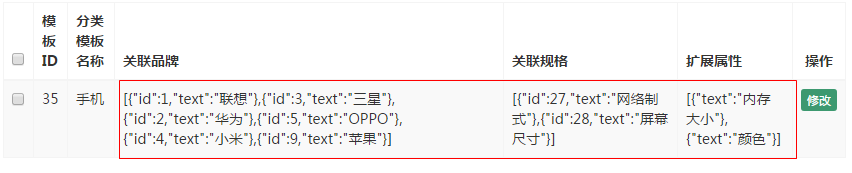


pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl/TypeTemplateServiceImpl.java



## 1.9 优化模板列表的显示（了解）

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pinyougou-manager-web

/src/main/webapp/js/controller/baseController.js

|  |
| --- |
| */\*\* 提取数组中json某个属性，返回拼接的字符串(逗号分隔) \*/* $scope.jsonArr2Str = **function**(jsonArrStr, key){  *// 把jsonArrStr转化成JSON数组对象* **var** jsonArr = ***JSON***.parse(jsonArrStr);  *// 定义新数组* **var** resArr = [];  *// 迭代json数组* **for** (**var** i = 0; i < jsonArr.**length**; i++){  *// 取数组中的一个元素* **var** json = jsonArr[i];  *// 把json对象的值添加到新数组* resArr.push(json[key]);  }  *// 返回数组中的元素用逗号分隔的字符串* **return** resArr.join(**","**); }; |

type\_template.html页面上使用该函数进行转换

|  |
| --- |
| <**tr ng-repeat="entity in dataList"**>  <**td**><**input type="checkbox" ng-click="updateSelection($event,entity.id)"**></**td**>  <**td**>{{**entity**.**id**}}</**td**>  <**td**>{{**entity**.**name**}}</**td**>  <**td**>{{jsonArr2Str(**entity**.**brandIds**, 'text')}}</**td**>  <**td**>{{jsonArr2Str(**entity**.**specIds**,'text')}}</**td**>  <**td**>{{jsonArr2Str(**entity**.**customAttributeItems**,'text')}}</**td**>  ...... </**tr**> |

# 2.商品分类管理

## 2.1**需求及表结构分析**

### 2.1.1**需求分析**

实现三级商品分类列表查询功能

进入页面首先显示所有的一级分类，效果如下：



点击列表行的查询下级按钮，进入下级分类列表，同时更新面包屑导航



再次点击表行的查询下级按钮，进入三级分类列表，因为三级分类属于最后一级，所以在列表中不显示查询下级按钮，同时更新面包屑导航



点击面包屑导航，可以进行返回操作。

### 2.1.2**表结构分析**

**tb\_item\_cat 商品分类表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 含义 |
| id | bigint |  | 主键 |
| parent\_id | bigint |  | 上级ID |
| name | varchar |  | 分类名称 |
| type\_id | bigint |  | 类型模板ID |

## 2.2**列表实现**

### 2.2.1**前端代码**

pinyougou-manager-web/src/main/webapp/admin/item\_cat.html引入js

|  |
| --- |
| <**script src="/plugins/angularjs/angular.min.js"**></**script**> <**script src="/js/base.js"**></**script**> <**script src="/js/service/baseService.js"**></**script**> <**script src="/js/controller/baseController.js"**></**script**> <**script src="/js/controller/itemCatController.js"**></**script**> |

item\_cat.html指令定义

|  |
| --- |
| <**body class="hold-transition skin-red sidebar-mini"  ng-app="pinyougou"  ng-controller="itemCatController"  ng-init="findItemCatByParentId(0)"**> |

pinyougou-manager-web/src/main/webapp/js/controller/itemCatController.js

|  |
| --- |
| */\*\* 根据上级ID显示下级列表 \*/* $scope.findItemCatByParentId = **function**(parentId){  baseService.sendGet(**"/itemCat/findItemCatByParentId"**,  **"parentId="** + parentId).then(  **function**(response){  $scope.**dataList** = response.data;  }  ); }; |

item\_cat.html页面用ng-repeat循环列表

|  |
| --- |
| <**tr ng-repeat="entity in dataList"**>  <**td**><**input type="checkbox"**></**td**>  <**td**>{{**entity**.**id**}}</**td**>  <**td**>{{**entity**.**name**}}</**td**>  <**td**>{{**entity**.**typeId**}}</**td**>  <**td class="text-center"**>  <**button type="button" class="btn bg-olive btn-xs"  ng-click="findItemCatByParentId(entity.id);"**>查询下级  </**button**>  <**button type="button" class="btn bg-olive btn-xs"  data-toggle="modal" data-target="#editModal"**>修改  </**button**>  </**td**> </**tr**> |

### 2.2.2**后端代码**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller包下创建ItemCatController.java

|  |
| --- |
| @RestController @RequestMapping(**"/itemCat"**) **public class** ItemCatController {  @Reference(timeout = 10000)  **private** ItemCatService **itemCatService**;  */\*\* 根据父级id查询商品分类 \*/* @GetMapping(**"/findItemCatByParentId"**)  **public** List<ItemCat> findItemCatByParentId(Long parentId){  **return itemCatService**.findItemCatByParentId(parentId);  } } |

pinyougou-interface/src/main/java/com.pinyougou.service/ItemCatService.java，增加方法

|  |
| --- |
| */\*\*  \* 根据父级id查询商品分类  \** ***@param parentId*** *父级id  \** ***@return*** *商品分类集合  \*/* List<ItemCat> findItemCatByParentId(Long parentId); |

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl包下创建ItemCatServiceImpl.java

|  |
| --- |
| @Service(interfaceName = **"com.pinyougou.service.ItemCatService"**) @Transactional **public class** ItemCatServiceImpl **implements** ItemCatService {  @Autowired  **private** ItemCatMapper **itemCatMapper**;  @Override  **public void** save(ItemCat itemCat) {  }  @Override  **public void** update(ItemCat itemCat) {  }  @Override  **public void** delete(Serializable id) {  }  @Override  **public void** deleteAll(Serializable[] ids) {  }  @Override  **public** ItemCat findOne(Serializable id) {  **return null**;  }  @Override  **public** List<ItemCat> findAll() {  **return null**;  }  @Override  **public** List<ItemCat> findByPage(ItemCat itemCat, **int** page, **int** rows) {  **return null**;  }  */\*\* 根据父级id查询商品分类 \*/* @Override  **public** List<ItemCat> findItemCatByParentId(Long parentId){  **try**{  */\*\* 创建ItemCat封装查询条件 \*/* ItemCat itemCat = **new** ItemCat();  itemCat.setParentId(parentId);  **return itemCatMapper**.select(itemCat);  }**catch**(Exception ex){  **throw new** RuntimeException(ex);  }  } } |

## 2.3**面包屑导航**

我们需要返回上级列表，需要通过点击面包屑来实现。

修改列表的查询下级按钮点击事件，设定级别值变量：grade

|  |
| --- |
| <**button type="button" class="btn bg-olive btn-xs"  ng-if="grade!=3"  ng-click="selectList(entity, grade + 1);"**>查询下级</**button**> |

这里使用了ng-if指令，用于条件判断，当级别不等于3的时候才显示“查询下级”按钮。

pinyougou-manager-web/src/main/webapp/js/controller/itemCatController.js

|  |
| --- |
| */\*\* 默认为1级 \*/* $scope.**grade** = 1; */\*\* 查询下级 \*/* $scope.selectList = **function**(entity, grade){  $scope.**grade** = grade;  **if**(grade == 1){ *//如果为1级* $scope.**itemCat\_1** = **null**;  $scope.**itemCat\_2** = **null**;  }  **if**(grade == 2){ *//如果为2级* $scope.**itemCat\_1** = entity;  $scope.**itemCat\_2** = **null**;  }  **if**(grade == 3){ *//如果为3级* $scope.**itemCat\_2** = entity;  }  */\*\* 查询此级下级列表 \*/* $scope.findItemCatByParentId(entity.**id**); }; |

item\_cat.html绑定面包屑：

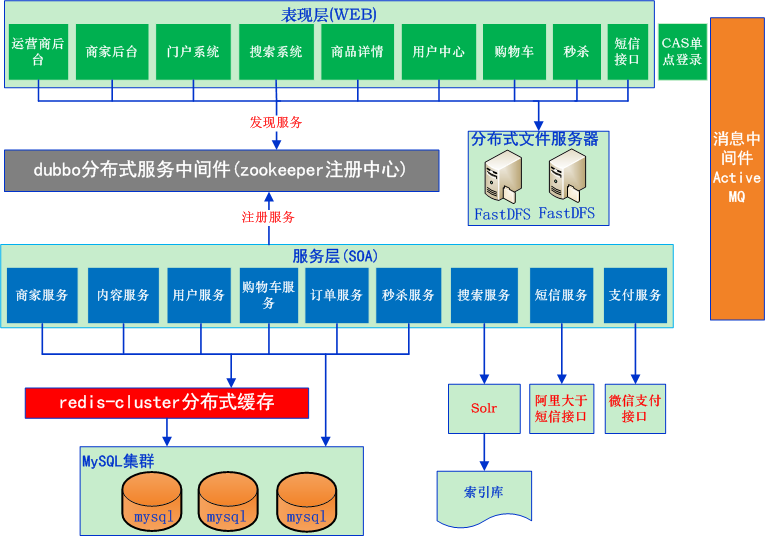
|  |
| --- |
| <**ol class="breadcrumb"**>  <**li**>  <**a href="#" ng-click="selectList({id:0}, 1)"**>顶级分类列表</**a**>  </**li**>  <**li ng-if="grade > 1"**>  <**a href="#"  ng-click="selectList(itemCat\_1, 2)"**>{{itemCat\_1.name}}</**a**>  </**li**>  <**li ng-if="grade > 2"**>  <**a href="#"  ng-click="selectList(itemCat\_2, 3)"**>{{itemCat\_2.name}}</**a**>  </**li**> </**ol**> |

## 2.4添加商品分类（学员挑战）

## 2.5添加商品分类（学员挑战）

## 2.6删除商品分类（学员挑战）

# 3.FastDFS【分布式文件系统】



## **3.1图片服务器介绍**

要实现图片上传功能，需要有一个图片服务器。图片服务器的特点：

1. 存储空间可扩展
2. 提供统一的访问方式
3. 访问效率高

企业中的图片服务器有两个方案：

1. 使用FastDFS分布式文件系统。
2. 使用Nginx服务器。

## **3.2什么是FastDFS**

FastDFS(Fast Distributed file system)是用c语言编写的一款开源的分布式文件系统。FastDFS为互联网量身定制，充分考虑了冗余备份、负载均衡、横向扩展等机制，并注重高可用、高性能等指标，使用FastDFS很容易搭建**一套高性能的文件服务器集群提供文件上传、下载等服务**。

## **3.3 FastDFS架构**

FastDFS架构包括 Tracker server和Storage server。客户端请求Tracker server进行文件上传、下载，通过Tracker server调度最终由Storage server完成文件上传和下载。

Tracker server作用是负载均衡和调度，通过Tracker server在文件上传时可以根据一些策略找到Storage server提供文件上传服务。可以将tracker称为追踪服务器或调度服务器。

Storage server作用是文件存储，客户端上传的文件最终存储在Storage服务器上，Storage server**没有实现自己的文件系统而是利用操作系统的文件系统来管理文件**。可以将storage称为存储服务器。

如下图：



Tracker 管理集群，收集信息，处理信息，为了保证高可用，可以搭建集群。

Storage 保存文件，分为很多组，组和组之间的数据不一样，组内成员数据是一样的，保证数据的高可用，可以增加组，达到扩容的效果。

## **3.4文件上传流程**

文件上传流程如下图（时序图）：



客户端上传文件后存储服务器将文件ID返回给客户端，此文件ID用于以后访问该文件的索引信息。文件索引信息包括：组名，虚拟磁盘路径，数据两级目录，文件名。



* 组名：文件上传后所在的storage组名称，在文件上传成功后有storage服务器返回，需要客户端自行保存。
* 虚拟磁盘路径：storage配置的虚拟路径，与磁盘选项store\_path\*对应。如果配置了store\_path0则是M00，如果配置了store\_path1则是M01，以此类推。
* 数据两级目录：storage服务器在每个虚拟磁盘路径下创建的两级目录，用于存储数据文件。两级目录的范围都是 00~FF
* 文件名：与文件上传时不同。是由存储服务器根据特定信息生成，文件名包含：源存储服务器IP地址、文件创建时间戳、文件大小、随机数和文件拓展名等信息。

## **3.5文件下载流程**

文件下载流程如下图：



## **3.6架构图**



## **3.7安装FastDFS【运维】**

在导入的虚拟机中已经安装；

若需自行安装详见“【资料\FastDFS\文档\ **Linux安装FastDFS.docx**】”

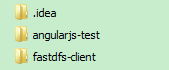
# 4.FastDFS客户端【上传&下载】

## **4.1添加依赖**

学习使用FastDFS需要有对应的Java客户端来支持；而遗憾的是FastDFS的作者没有把FastDFS的依赖jar放置到maven中央仓库提供下载；但可以自行导入项目并安装，然后再引用它。

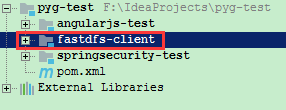
### **4.1.1 fastdfs-client模块**

切换到pyg-test项目，将“资料\FastDFS\java客户端\fastdfs-client”复制到pyg-test项目下。

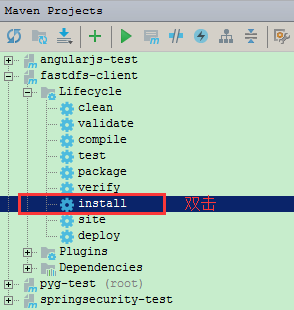


配置pyg-test的pom.xml：

|  |
| --- |
| <**module**>fastdfs-client</**module**> |

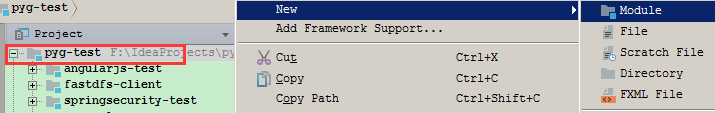


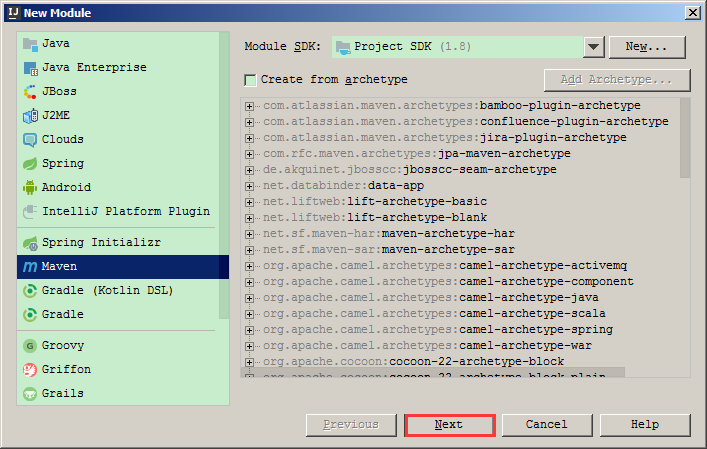
### **4.1.2安装fastdfs-client**

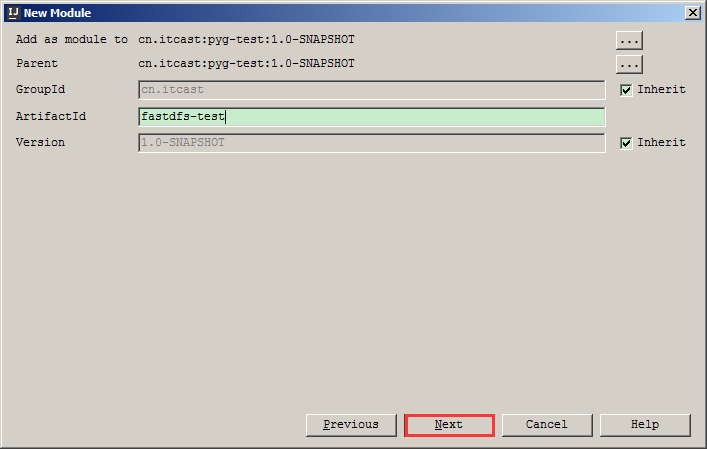


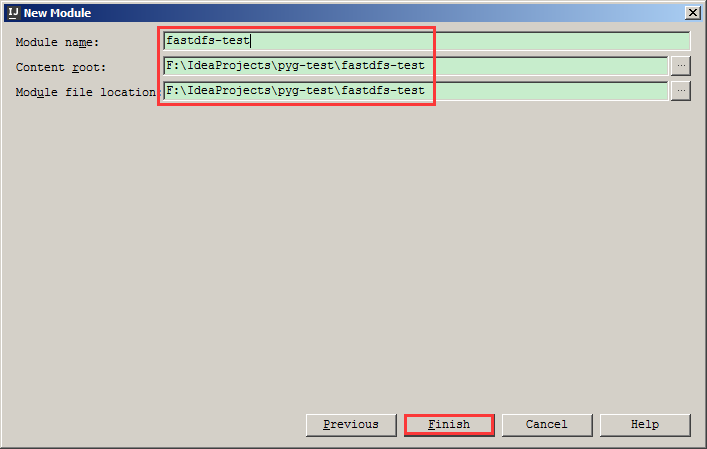
## **4.2 fastdfs-test测试模块**

创建fastdfs-test模块(jar)



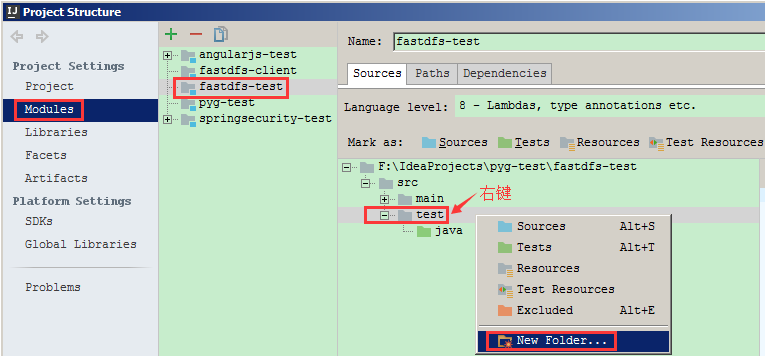


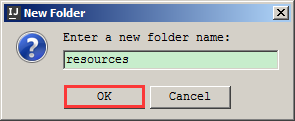


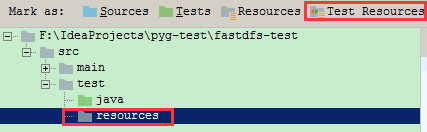


新增fastdfs-test/src/main/test/resources测试资源文件夹









fastdfs-test/pom.xml依赖信息如下：

|  |
| --- |
| <**dependencies**>  *<!-- fastdfs-client -->* <**dependency**>  <**groupId**>org.csource</**groupId**>  <**artifactId**>fastdfs-client</**artifactId**>  <**version**>1.25-RELEASE</**version**>  </**dependency**>  </**dependencies**> |

## **4.3上传文件**

### **4.3.1配置tracker server地址**

新增fastdfs-test/src/test/resources/fastdfs\_client.conf文件，内容为：

|  |
| --- |
| # 配置追踪服务器连接地址 tracker\_server=192.168.12.131:22122 |

### **4.3.2编写上传代码**

fastdfs-test/src/test/java/cn.itcast.fastdfs.FastdfsTest测试类

|  |
| --- |
| **public class** FastdfsTest {  */\*\* 上传文件 \*/* @Test  **public void** upload\_file() **throws** Exception{  */\*\* 加载配置文件，产生该文件的绝对路径 \*/* String conf\_filename = **this**.getClass()  .getResource(**"/fastdfs\_client.conf"**).getPath();  */\*\* 初始化客户端全局的对象 \*/* ClientGlobal.*init*(conf\_filename);  */\*\* 创建存储客户端对象 \*/* StorageClient storageClient = **new** StorageClient();  */\*\* 上传文件 \*/* String[] arr = storageClient.upload\_file(**"C:/Users/Public/Pictures/Sample Pictures/15.jpg"**, **"jpg"**, **null**);  */\*\*  \* 访问路径：http://192.168.12.131/group1/M00/00/00/wKgMg1o53fOAL1CRAABonuLw4M4127.jpg  \* [group1, M00/00/00/wKgMgFlIkk2AHfnLAABonuLw4M4075.jpg]  \* 数组中的第一个元素：组的名称  \* 数组中的第二个元素：远程文件名称  \*/* System.***out***.println(Arrays.*toString*(arr));  }  } |

**访问地址：**

**http://192.168.12.131/M00/00/00/KgMgFlHqumASKsWAABonuLw4M4076.jpg**

### **4.3.3上传步骤小结**

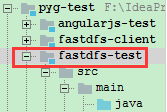
1. 加载tracker server的配置文件地址，配置文件为tracker server的地址；在这里设置了tracker\_server=192.168.12.131:22122
2. 初始化客户端全局信息ClientGlobal.init(conf)
3. 创建存储客户端StorageClient
4. 调用storageClient的upload\_file()方法上传图片
5. 解析并获取上传结果，返回组名和文件的相对路径

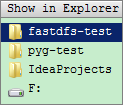
## **4.4下载文件**

修改fastdfs-test/src/test/java/cn.itcast.fastdfs.FastdfsTest类，新增download\_file方法

|  |
| --- |
| */\*\* 文件下载 \*/* @Test **public void** download\_file() **throws** Exception{  */\*\* 加载配置文件，产生绝对路径 \*/* String conf\_filename = **this**.getClass()  .getResource(**"/fastdfs\_client.conf"**).getPath();  */\*\* 初始化客户端全局的对象 \*/* ClientGlobal.*init*(conf\_filename);  */\*\* 创建存储客户端对象 \*/* StorageClient storageClient = **new** StorageClient();  */\*\* 下载文件 \*/* **byte**[] data = storageClient.download\_file(**"group1"**, **"M00/00/00/wKgMgFlIkk2AHfnLAABonuLw4M4075.jpg"**);  System.***out***.println(data.**length**);  FileOutputStream fos = **new** FileOutputStream(**new** File(**"F:/workspace/java/fastdfs-test/src/test/resources/1.jpg"**));  fos.write(data);  fos.flush();  fos.close(); } |

IDEA中查看项目磁盘路径：





## **4.5删除文件**

修改fastdfs-test/src/test/java/cn.itcast.fastdfs.FastdfsTest类，新增delete\_file方法

|  |
| --- |
| */\*\* 删除文件 \*/* @Test **public void** delete\_file() **throws** Exception{  */\*\* 加载配置文件，产生绝对路径 \*/* String conf\_filename = **this**.getClass()  .getResource(**"/fastdfs\_client.conf"**).getPath();  */\*\* 初始化客户端全局的对象 \*/* ClientGlobal.*init*(conf\_filename);  */\*\* 创建存储客户端对象 \*/* StorageClient storageClient = **new** StorageClient();  */\*\* 删除文件 \*/* **int** res = storageClient.delete\_file(**"group1"**,  **"M00/00/00/wKgMgFlIkk2AHfnLAABonuLw4M4075.jpg"**);  System.***out***.println(res); } |