# 商品后台管理

## 商品分类管理

### 数据分析

说明:商品分类信息分为三级目录,通过parent\_id进行关联.

查询一级目录:

select \* from tb\_item\_cat where parent\_id = 0;

查询二级目录:

select \* from tb\_item\_cat where parent\_id =一级分类Id;

查询三级目录:

select \* from tb\_item\_cat where parent\_id =二级分类Id;

### EasyUITree树形结构介绍

#### 树形结构JS

$("#tree").tree({

**url:"tree.json",** //加载远程JSON数据

method:"get", //请求方式 POST

animate:**true**, //表示显示折叠端口动画效果

checkbox:**true**, //表述复选框

lines:**true**, //表示显示连接线

dnd:**true**, //是否拖拽

onClick:**function**(node){ //添加点击事件

//控制台

console.info(node);

}

});

#### 树形结构返回值

结构如下:

[{id:1,text:"分类名称",state:"open/**closed**"}]

#### 封装VO对象

//VO:是服务器数据与页面交互的对象

//一般都需要转化为JSON

@Data //安装lombok插件

@Accessors(chain = **true**)

@NoArgsConstructor

@AllArgsConstructor

**public** **class** EasyUI\_Tree {

**private** Long id; //分类Id号

**private** String text;//分类名称

**private** String state; //open 节点打开 closed 节点关闭

}

### 页面分析

1. url分析



2.js分析

$("ul",\_win).tree({

url:'/item/cat/list',

animate:**true**,

onClick : **function**(node){

**if**($(**this**).tree("isLeaf",node.target)){

// 填写到cid中

\_ele.parent().find("[name=cid]").val(node.id);

\_ele.next().text(node.text).attr("cid",node.id);

$(\_win).window('close');

**if**(data && data.fun){

data.fun.call(**this**,node);

}

}

}

});

### 商品分类异步树加载

树控件读取URL。子节点的加载依赖于父节点的状态。当展开一个封闭的节点，如果节点没有加载子节点，**它将会把节点id的值作为http请求参数并命名为'id'，通过URL发送到服务器上面检索子节点。**

**当鼠标节点父级节点时,会将节点Id当做参数查询子节点信息.**

### 编辑Controller

/\*\*

\* 实现商品分类树形结构查询

\*/

@RequestMapping("/list")

**public** List<EasyUI\_Tree> findItemCatByParentId

(@RequestParam(name="id",defaultValue="0")Long parentId){

**return** itemCatService.findItemCatByParentId(parentId);

}

### 编辑Service

/\*\*

\* List<EasyUI\_Tree> 返回的是VO对象集合

\* EasyUI\_Tree:id/text/state

\*

\* List<ItemCat> 返回ItemCat集合对象

\* ItemCat: id/name/判断是否为父级

\*/

@Override

**public** List<EasyUI\_Tree> findItemCatByParentId(Long parentId) {

List<EasyUI\_Tree> treeList = **new** ArrayList<>();

//1.获取数据库数据

List<ItemCat> itemCatList = findItemCatList(parentId);

**for** (ItemCat itemCat : itemCatList) {

Long id = itemCat.getId();

String text = itemCat.getName();

//一级二级菜单 closed 三级菜单 open

String state =

itemCat.getIsParent()?"closed":"open";

EasyUI\_Tree tree =

**new** EasyUI\_Tree(id, text, state);

treeList.add(tree);

}

**return** treeList;

}

**public** List<ItemCat> findItemCatList(Long parentId) {

QueryWrapper<ItemCat> queryWrapper

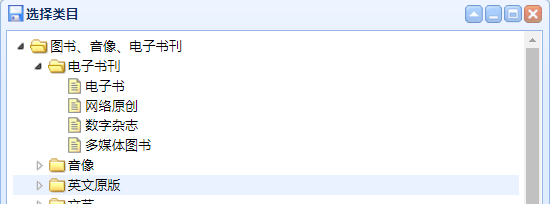
= **new** QueryWrapper<ItemCat>();

queryWrapper.eq("parent\_id", parentId);

**return** itemCatMapper.selectList(queryWrapper);

}

### 页面效果



## EasyUI数据校验

### 非空校验

required:true:表示数据为必填项.

data-options="required:true"

### 数值范围设定

data-options="min:1,max:99999999,precision:2,required:true"

## 完成商品入库

### 商品入库业务流程

1. 用户根据页面提示输入数据后点击提交按钮
2. 获取用户输入内容,发起ajax post请求实现入库操作.
3. 服务端利用对象获取数据,之后完成入库操作.
4. 无论操作是否成功 都应该返回页面**回执(VO对象)**. 页面通过回执判断数据操作是否正确.

### 编辑SysResult对象

该对象中有3个属性

1.Integer status 200 操作成功 201 操作失败

2.String msg 服务器返回页面提示信息

3.Object data 服务器返回业务数据.之后页面解析.

### 页面分析

1. 页面URL分析



1. 页面JS

$.post(url地址,提交的参数,回调函数);

$.post("/item/save",$("#itemAddForm").serialize(), **function**(data){

**if**(data.status == 200){

$.messager.alert('提示','新增商品成功!');

}**else**{

$.messager.alert("提示","新增商品失败!");

}

});

### 编辑Controller

/\*\*

\* 实现商品新增

\*

\* 准备AOP 统一异常处理机制.

\*/

@RequestMapping("/save")

**public** SysResult saveItem(Item item) {

**try** {

itemService.saveItem(item);

**return** SysResult.*success*();

} **catch** (Exception e) {

e.printStackTrace();

**return** SysResult.*fail*();

}

}

### 编辑Service

@Transactional//添加事务控制

@Override

**public** **void** saveItem(Item item) {

item.setStatus(1) //1.表示状态正常

.setCreated(**new** Date())

.setUpdated(item.getCreated());

itemMapper.insert(item);

}

## 统一异常处理

### 编辑切面

@RestControllerAdvice

@Slf4j

**public** **class** SysResultAspect {

/\*\*

\* 如果程序报错,则统一返回系统异常信息

\* SysResult.fail()

\*/

@ExceptionHandler({RuntimeException.**class**}) //如果遇到指定的异常类型执行下列方法

//@ResponseBody

**public** SysResult sysResultFail(Exception e) {

e.printStackTrace();

***log***.error("服务器异常信息:"+e.getMessage());

**return** SysResult.*fail*();

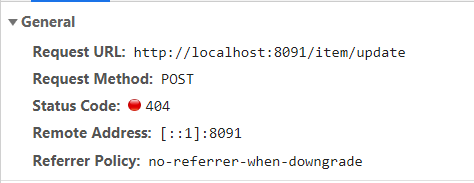
}

}

## 商品修改

### 页面分析

1. url分析

\

1. 页面JS分析

$.post("/item/update",$("#itemeEditForm").serialize(), **function**(data){

**if**(data.status == 200){

$.messager.alert('提示','修改商品成功!','info',**function**(){

$("#itemEditWindow").window('close');

$("#itemList").datagrid("reload");

});

}**else**{

$.message.alert("提示",data.msg);

}

});

### 编辑Controller

/\*\*

\* 实现商品修改

\*/

@RequestMapping("/update")

**public** SysResult updateItem(Item item) {

itemService.updateItem(item);

**return** SysResult.*success*();

//由于已经编辑统一异常处理.

//所以不需要考虑异常

}

### 编辑Service

@Override

**public** **void** updateItem(Item item) {

item.setUpdated(**new** Date());

//sql: xxxx where id = 主键值

itemMapper.updateById(item);

}

## 商品删除

### 页面分析

1. url分析



1. JS分析

{

text:'删除',

iconCls:'icon-cancel',

handler:**function**(){

//用户选中Id的字符串

**var** ids = getSelectionsIds();

**if**(ids.length == 0){

$.messager.alert('提示','未选中商品!');

**return** ;

}

$.messager.confirm('确认','确定删除ID为 '+ids+' 的商品吗？',**function**( ){

**if** (r){

**var** params = {"ids":ids};

$.post("/item/delete",params, **function**(data){

**if**(data.status == 200){

$.messager.alert('提示','删除商品成功!',undefined,**function**(){

$("#itemList").datagrid("reload");

});

}**else**{

$.messager.alert("提示",data.msg);

}

});

}

});

}

### 编辑Controller

/\*\*

\* 删除商品信息

\* var params = {"ids":1,2,3,4,5};

\* 如果参数是通过,号分割.

\* 则接收时可以使用数组

\*/

@RequestMapping("/delete")

**public** SysResult deleteItems(Long[] ids) {

itemService.deleteItems(ids);

**return** SysResult.*success*();

}

### 编辑Service

/\*\*

\* 1.自己手写sql

\* delete from tb\_item where id in (1,2,3,4)

\* 2.使用mybatisplus

\*/

@Override

**public** **void** deleteItems(Long[] ids) {

//1.手写sql 复习mybatis用法

//itemMapper.deleteItems(ids);

//2.利用mybatisMapper操作

//将数组转化为集合

List<Long> idList = Arrays.*asList*(ids);

itemMapper.deleteBatchIds(idList);

}

### 编辑Mapper接口映射文件

/\*\*

\* 关于Mybatis传参问题

\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*

\* 规则:mybaits中默认是不允许多值传参,

\* 需要用户将多值转化为单值

\* 1.本身数据就是一个参数

\* 2.将数据使用对象进行封装

\* 3.使用集合进行封装 list array

\* 4.无论参数是什么都可以封装为Map集合 **@Param**

\* **@param** ids

\*/

**void** deleteItems(Long[] ids);

2.编辑映射文件

<!--1.批量删除

delete from tb\_item where id in (1,2,3,4)

collection:

如果参数为array或者为list集合则直接写类型名称

如果参数为Map集合.则collection写key

-->

<delete id=*"deleteItems"*>

delete from tb\_item where id in (

<foreach collection=*"array"* item=*"id"*

separator=*","*>

#{id}

</foreach>

)

</delete>

## 商品上架/下架

### 业务说明

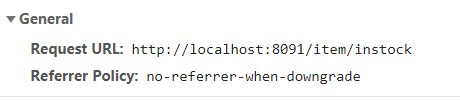
商品新增时status=1表示正常的.如果该商品需要下架,则将status信息改为2即可.

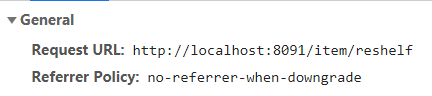
下架sql:update tb\_item set status=2,updated=now() where id in (1,2,3,4,5);

上架sql:update tb\_item set status=1,updated=now() where id in (1,2,3,4,5);

### 页面分析

1. url分析





1. js分析

$.post("/item/instock",params, **function**(data){

**if**(data.status == 200){

$.messager.alert('提示','下架商品成功!',undefined,**function**(){

$("#itemList").datagrid("reload");

});

}

});

### 编辑Controller

/\*\*

\* 商品下架

\*/

@RequestMapping("/instock")

**public** SysResult itemInstock(Long[] ids) {

**int** status = 2; //表示下架

itemService.updateStatus(ids,status);

**return** SysResult.*success*();

}

/\*\*

\* 商品上架

\*/

@RequestMapping("/reshelf")

**public** SysResult itemReshelf(Long[] ids) {

**int** status = 1; //表示上架

itemService.updateStatus(ids,status);

**return** SysResult.*success*();

}

### 编辑Service

@Override

**public** **void** updateStatus(Long[] ids, **int** status) {

Item item = **new** Item();

item.setStatus(status)

.setUpdated(**new** Date());

List<Long> longList = Arrays.*asList*(ids);

UpdateWrapper<Item> upteUpdateWrapper =

**new** UpdateWrapper<Item>();

upteUpdateWrapper.in("id", longList);

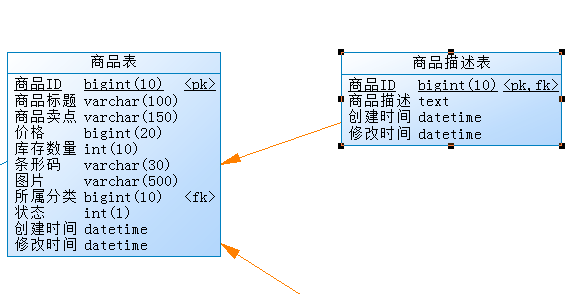
itemMapper.update(item, upteUpdateWrapper);

}

## 富文本编辑器

### 表设计

说明:根据表设计定义.商品表Id与商品详情表id一致的.并且是一对一关系.



### 编辑POJO

@Data

@Accessors(chain = **true**)

@TableName("tb\_item\_desc")

**public** **class** ItemDesc **extends** BasePojo{

//item表id与itemDesc表Id一致的

@TableId

**private** Long itemId;

**private** String itemDesc;

}

### 富文本编辑器入门

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<link href="/js/kindeditor-4.1.10/themes/default/default.css" type="text/css" rel="stylesheet">

<script type="text/javascript" charset="utf-8" src="/js/kindeditor-4.1.10/kindeditor-all-min.js"></script>

<script type="text/javascript" charset="utf-8" src="/js/kindeditor-4.1.10/lang/zh\_CN.js"></script>

<script type="text/javascript" charset="utf-8" src="/js/jquery-easyui-1.4.1/jquery.min.js"></script>

<script type="text/javascript">

$(**function**(){

KindEditor.ready(**function**(){

KindEditor.create("#editor")

})

})

</script>

</head>

<body>

<h1>富文本编辑器</h1>

<textarea style="width:700px;height:350px" id="editor"></textarea>

</body>

### 编辑Controller

@RequestMapping("/save")

**public** SysResult saveItem(Item item,ItemDesc itemDesc) {

/\*

\* try { itemService.saveItem(item); return SysResult.success(); } catch

\* (Exception e) { e.printStackTrace(); return SysResult.fail(); }

\*/

itemService.saveItem(item,itemDesc);

**return** SysResult.*success*();

}

### 编辑Service

/\*\*

\* 由于主键自增,每次入库后返回id值

INSERT INTO USER VALUES (NULL,"xxxx",19,"其他");

查询最后一个入库id信息 不会有线程安全性问题

SELECT LAST\_INSERT\_ID();

\*/

@Transactional//添加事务控制

@Override

**public** **void** saveItem(Item item,ItemDesc itemDesc) {

item.setStatus(1) //1.表示状态正常

.setCreated(**new** Date())

.setUpdated(item.getCreated());

itemMapper.insert(item);

//因为商品详情与商品是id一致.但是item数据是主键

//自增只有入库之后才能获取主键信息.

//答案:入库时返回Id值

itemDesc.setItemId(item.getId())

.setCreated(item.getCreated())

.setUpdated(item.getCreated());

itemDescMapper.insert(itemDesc);

}