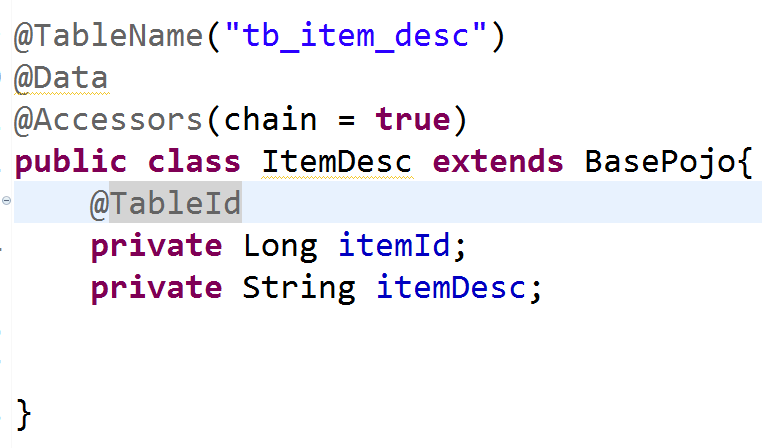
# 京淘项目第五天

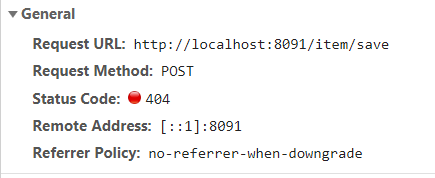
## 商品新增

### 编辑POJO对象



### 新增页面分析

#### 页面url地址



#### 页面ajax

//$.post(url,params,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!");

}

});

### 封装系统的VO对象

说明:由于项目是前后台统一调用.封装一个VO对象,目的是给前台用户一个友好的回执.在jt-common中添加该对象

//系统返回值对象

@Data

@Accessors(chain = **true**)

@NoArgsConstructor

@AllArgsConstructor

**public** **class** SysResult {

**private** Integer status; //200表示成功 201表示失败

**private** String msg; //后台返回值数据提示信息

**private** Object data; //后台返回任意数据

**public** **static** SysResult ok(){

**return** **new** SysResult(200, **null**, **null**);

}

**public** **static** SysResult ok(Object data) {

**return** **new** SysResult(200, **null**, data);

}

**public** **static** SysResult ok(String msg,Object data) {

**return** **new** SysResult(200, msg, data);

}

**public** **static** SysResult fail() {

**return** **new** SysResult(201, **null**, **null**);

}

**public** **static** SysResult fail(String msg) {

**return** **new** SysResult(201, msg, **null**);

}

//假如:用户需要回传一个data data数据类型是String

/\*

\* public static SysResult ok(String msg) {

\*

\* return new SysResult(200, msg,null); }

\*/

}

### 编辑Controller

@RequestMapping("/save")

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

**try** {

//实现数据新增

itemService.saveItem(item);

**return** SysResult.*ok*();

} **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)

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

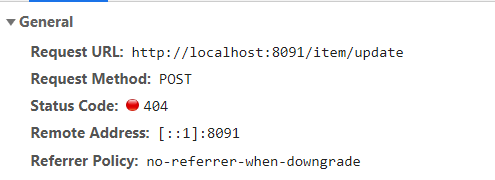
.setUpdated(item.getCreated());

itemMapper.insert(item);

}

## 商品修改

### 页面URL分析



2.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

//修改商品

@RequestMapping("/update")

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

**try** {

itemService.updateItem(item);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

**return** SysResult.*fail*("商品修改失败");

}

}

### 编辑Service

//根据主键更新

@Override

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

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

itemMapper.updateById(item);

}

## 关于事务控制说明

### @Transactional说明

#### 事务策略

Spring中默认的事务控制策略:

1.检查异常/编译异常 不负责事务控制

2.运行时异常 /error 回滚事务

#### Propagation

propagation 事务传播属性

默认值REQUIRED 必须添加事务 事务合并

REQUIRES\_NEW 必须新建一个事务

SUPPORTS 表示事务支持的 查询之前有事务时则合并事务

#### 控制事务是否回滚

按照指定的异常回滚事务.

\* rollbackFor = "异常的类型" 遇到异常回滚

\* noRollbackFor = "异常类型" 遇到异常不回滚.

#### ReadOnly

readOnly = true 不允许修改数据库 只读

## 商品删除

### 消息提示框

/\*定义退出消息框 \*/

$("#btn4").click(**function**(){

$.messager.confirm('1902AAAA','你确定要退出吗',**function**(r){

//判断用户选择按钮

if (r){

alert("确认退出");

}else{

alert("你不是要退出吗??怂了???")

}

});

})

### 删除页面JS

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

**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);

}

});

}

});

### 编辑页面Controller

根据页面url请求地址删除选中数据.

/\*\*

\* 1.删除商品信息

\* url地址:/item/delete

\* 参数: {ids:ids}

\*/

@RequestMapping("/delete")

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

**try** {

itemService.deleteItem(ids);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

### 编辑Service

@Override

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

//1.手动删除

//itemMapper.deleteItem(ids);

//2.自动删除

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

itemMapper.deleteBatchIds(itemList);

}

### 编辑Mapper接口/映射文件

**public** **interface** ItemMapper **extends** BaseMapper<Item>{

@Select("select \* from tb\_item order by updated desc limit #{start},#{rows}")

List<Item> findItemByPage(@Param("start")Integer start,@Param("rows")Integer rows);

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

}

2.编辑xml映射文件

<mapper namespace="com.jt.mapper.ItemMapper">

<!--1.批量删除数据 Long[] ids

collection="array" 参数是数组

collection="list" 参数是list集合

collection="map中的key" 参数经过map封装

-->

<delete id="deleteItem">

delete from tb\_item where id in (

<foreach collection="array"

item="id"

separator=",">

#{id}

</foreach>

)

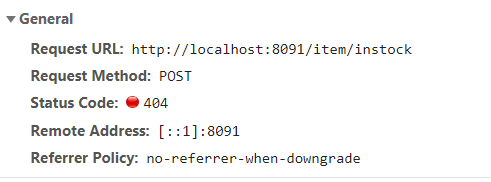
</delete>

</mapper>

## 商品上架/下架

### 页面分析

#### 页面URL地址



#### 页面JS代码

$.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

//实现商品下架

@RequestMapping("/instock")

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

**try** {

**int** status = 2; //下架

itemService.updateStatus(ids,status);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

@RequestMapping("/reshelf")

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

**try** {

**int** status = 1; //上架

itemService.updateStatus(ids,status);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

### 编辑Service

/\*

\* sql: update tb\_item

\* set status=#{status},

\* updated = #{updated}

\* where id in (100,200,300....)

\* \*/

@Override

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

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

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

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

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

updateWrapper.in("id", longIds);

itemMapper.update(item, updateWrapper);

}

## 富文本编辑器

### 富文本入门案例

#### 编辑器引用

<script type="text/javascript">

$(**function**(){

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

KindEditor.create("#editor")

})

})

</script>

#### 页面调用

<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>

## 商品详情新增

### 页面分析

<tr>

<td>商品描述:</td>

<td>

<textarea style="width:*800px*;height:*300px*;visibility:*hidden*;" name=*"itemDesc"*></textarea>

</td>

</tr>

### 编辑Controller封装数据

//itemDesc=<div>页面信息</div>

@RequestMapping("/save")

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

**try** {

//实现数据新增

itemService.saveItem(item,itemDesc);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

### 编辑Service

@Transactional//添加事务控制

@Override

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

item.setStatus(1)

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

.setUpdated(item.getCreated());

itemMapper.insert(item);

//同时入库2张表

itemDesc.setItemId(item.getId())

.setCreated(item.getCreated())

.setUpdated(item.getCreated());

itemDescMapper.insert(itemDesc);

}

## 商品详情回显

### 页面分析

#### 页面URL



#### 页面JS

$.getJSON('/item/query/item/desc/'+data.id,**function**(\_data){

**if**(\_data.status == 200){

//UM.getEditor('itemeEditDescEditor').setContent(\_data.data.itemDesc, false);

itemEditEditor.html(\_data.data.itemDesc);

}

});

### 编辑Controller

说明:利用restFul风格查询数据

//根据itemId查询商品详情信息

@RequestMapping("/query/item/desc/{itemId}")

**public** SysResult findItemDescById(@PathVariable Long itemId) {

**try** {

ItemDesc itemDesc = itemService.findItemDescById(itemId);

**return** SysResult.*ok*(itemDesc);

} **catch** (Exception e) {

e.printStackTrace();

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

}

}

### 编辑Service

@Override

**public** ItemDesc findItemDescById(Long itemId) {

**return** itemDescMapper.selectById(itemId);

}

## 商品多表维护

### 多表更新

说明:因为tb\_item和tb\_item\_desc是一对一关联关系.所以需要同时维护.

#### 编辑Controller

//修改商品

@RequestMapping("/update")

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

**try** {

itemService.updateItem(item,itemDesc);

**return** SysResult.*ok*();

} **catch** (Exception e) {

e.printStackTrace();

**return** SysResult.*fail*("商品修改失败");

}

}

#### 编辑Service

@Transactional

@Override

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

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

itemMapper.updateById(item);

//同时修改2张表数据

itemDesc.setItemId(item.getId())

.setUpdated(item.getUpdated());

itemDescMapper.updateById(itemDesc);

}

### 多表删除

说明:编辑ItemService中的方法

@Transactional

@Override

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

//1.手动删除

//itemMapper.deleteItem(ids);

//2.利用Mybatis-plus自动删除

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

itemMapper.deleteBatchIds(itemList);

//2张表一起删除

itemDescMapper.deleteBatchIds(itemList);

}