# 实现京淘后台管理业务处理

## 通用mapper介绍

### 业务需求

回顾:

JDBC--🡪JPA(思想)以面向对象的方式操作数据(orm)--🡪hibernate---🡪Mybatis(orm,手写sql语句)

问题:

1. 使用mybatis时,无论多么简单的sql语句,都需要自己手写
2. 同时mapper接口和映射文件必须一一对应.
3. 需要编辑映射文件的映射关系.resultType/resultMap

想法:

能否在mybatis的基础上,实现单表的自动增删改查.能够以对象的方式操作数据库.------反射

实现思路:

1. 操作的对象的属性与数据表中的字段一一对应
2. 定义公共的Mapper接口.在其中定义CRUD接口方法,其中参数采用泛型的思想进行定义.
3. 准备java类专门负责Mapper接口方法的实现 sql语句的生成.
4. 对于sql自动生成都是模板性的代码

例如新增:

Insert into 表名(通过反射动态获对象的属性) values(对象中的值…….)

### 通用mapper介绍

说明:通用Mapper是基于Mybatis实现的,利用反射技术.最终实现单表增删改查等功能自动完成的插件.用户操作时只需要传递不为null的对象,即可实现数据操作.(ORM)

### 导入jar包

jar包导入

<!-- 通用Mapper -->

<dependency>

<groupId>com.github.abel533</groupId>

<artifactId>mapper</artifactId>

<version>${mapper.version}</version>

</dependency>

### 导入插件

<plugins>

<!-- 通用Mapper插件 -->

<plugin interceptor=*"com.github.abel533.mapperhelper.MapperInterceptor"*>

<!--主键自增回写方法,默认值MYSQL,详细说明请看文档 -->

<property name=*"IDENTITY"* value=*"MYSQL"* />

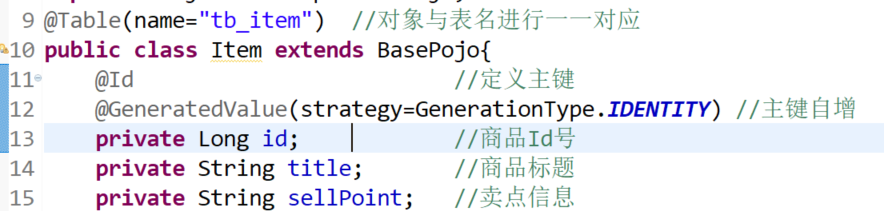
<!--通用Mapper接口，多个通用接口用逗号隔开 -->

<property name=*"mappers"* value=*"com.jt.common.mapper.SysMapper"* />

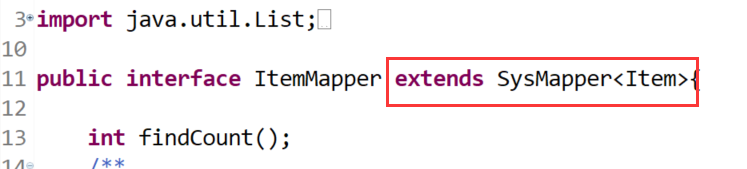
</plugin>

</plugins>

### 定义POJO与表关系

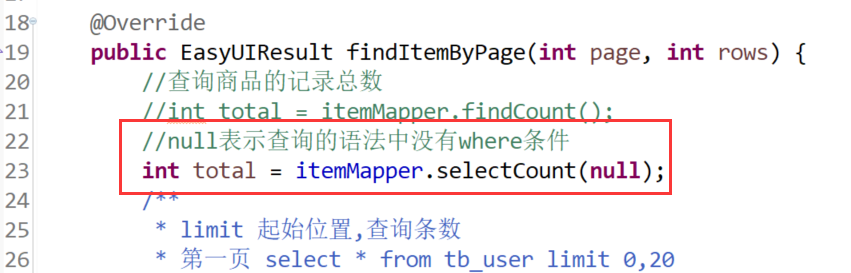


### 扩展接口方法



### 方法测试

说明:基于通用Mapper实现记录总数查询



### 关于通用mapper用法介绍

用法1:入库操作

User中的数据不为null User(id,name,age)

入库操作时插入的数据时user中不为null的属性的值

UserMapper.insert(user)

删除用法:

UserMapper.deletePK(主键Id)

UserMapper.delete(user)

User中不为null的属性充当where条件

查询用法:

UserMapper.selectPK(主键Id)

UserMapper.select(user)

User中不为null的属性充当where条件

修改操作:

UserMapper.update(user)

保证主键不为null.因为where 主键=”XXXXX”

## 商品新增

### 页面校验

由于网站的设计,要求用户输入特定的内容,所以需要校验.

插件:

1. jQueryValidate

http://www.runoob.com/jquery/jquery-plugin-validate.html

1. JS中API
2. bootStarp中js校验
3. EasyUI中js校验

### 检验的案例

1. 非空校验

data-options="required:true"

1. 有效性校验

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

### EasyUI中树形结构

$("#tree").tree({

url:"tree.json", //加载远程JSON数据

method:"POST", //请求方式 POST

animate:**true**, //表示显示折叠端口

checkbox:**true**, //表述复选框

lines:**false**, //表示显示连接线

dnd:**true**, //是否拖拽

onClick:**function**(node){ //添加点击事件

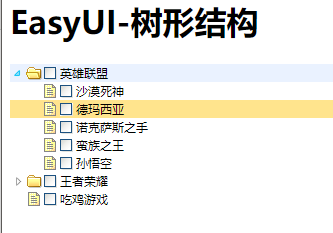
//控制台

console.info(node);

}

});

页面效果:



## 商品分类目录展现

### 页面分析



2.返回值结构分析

[

{

"id":"1",

"text":"英雄联盟",

"children":[

{

"id":"4",

"text":"沙漠死神"

}

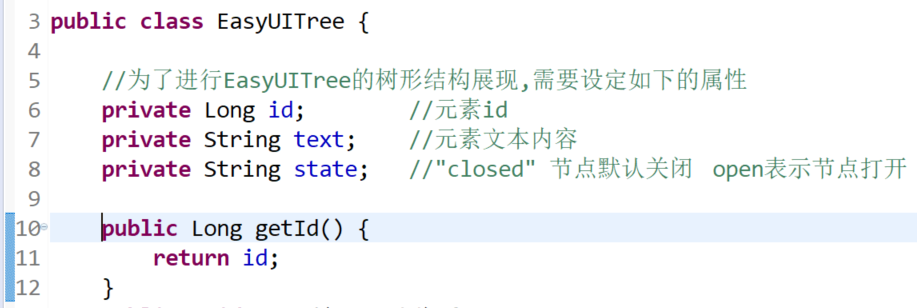
]

"state":"closed"

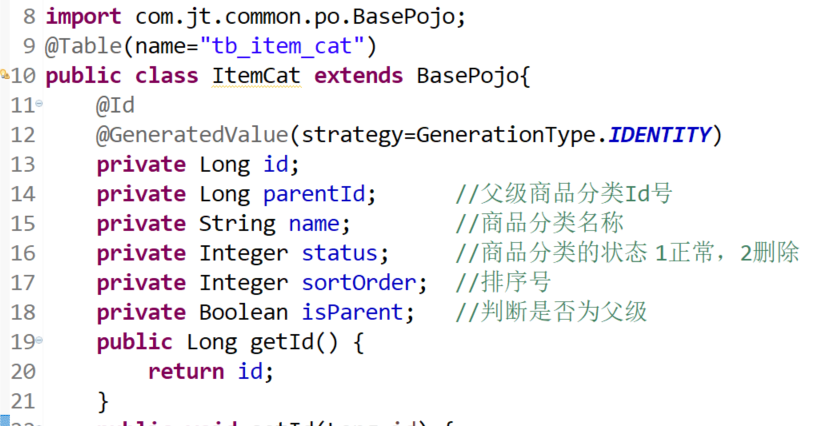
}

]

### 编辑VO对象



### 编辑POJO实体对象



### 节点加载的策略

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。

### 编辑Controller

@Controller

@RequestMapping("/item/cat")

**public** **class** ItemCatController {

@Autowired

**private** ItemCatService itemCatService;

//实现商品分类目录展现

@RequestMapping("/list")

@ResponseBody

**public** List<EasyUITree> findItemCatList

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

//1.查询一级商品分类信息

//Long parentId = 0L;

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

}

}

### 编辑Service

@Service

**public** **class** ItemCatServiceImpl **implements** ItemCatService {

@Autowired

**private** ItemCatMapper itemCatMapper;

/\*\*

\* 实现思路:

\* 1.利用通用Mapper查询商品分类信息

\* 2.将ItemCatList转化为EasyUITreeList

\*/

@Override

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

ItemCat itemCat = **new** ItemCat();

itemCat.setParentId(parentId);

//select \* from tb\_item\_cat where is\_parent = 0

List<ItemCat> itemCatList = itemCatMapper.select(itemCat);

List<EasyUITree> treeList = **new** ArrayList<>();

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

EasyUITree easyUITree = **new** EasyUITree();

easyUITree.setId(itemCatTemp.getId());

easyUITree.setText(itemCatTemp.getName());

//如果是父级 应该设置为closed,否则open即可

String state = itemCatTemp.getIsParent() ? "closed" : "open";

easyUITree.setState(state);

treeList.add(easyUITree);

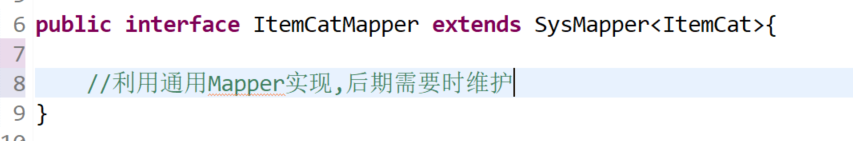
}

**return** treeList;

}

}

### 编辑Mapper接口



## 商品新增

### 页面分析

//id=1&name=tom&age=18

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

alert($("#itemAddForm").serialize());

**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

@RequestMapping("/save")

@ResponseBody

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

**try** {

itemService.saveItem(item);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201, "商品新增失败");

}

### 编辑Service

@Override

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

//封装数据

item.setStatus(1); //1表示正常

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

item.setUpdated(item.getCreated());

itemMapper.insert(item); //利用mybatis实现入库

}

## 商品修改

### 页面分析

1. 页面弹出框

<div id="itemEditWindow" class="easyui-window" title="编辑商品" data-options="modal:true,closed:true,iconCls:'icon-save',**href:'/page/item-edit**'" style="width:80%;height:80%;padding:10px;">

</div>

1. 页面数据提交

$.post("/item/update",$("#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

//实现商品的更新操作

@RequestMapping("/update")

@ResponseBody

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

**try** {

itemService.updateItem(item);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"修改商品数据失败");

}

### 编辑Service

@Override

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

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

//动态更新操作 将对象中不为null的数据充当set条件 主键除外

itemMapper.updateByPrimaryKeySelective(item);

}

## 商品删除

### 页面分析

$.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

@RequestMapping("/delete")

@ResponseBody

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

**try** {

itemService.deleteItems(ids);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"商品删除失败");

}

### 编辑Service

@Override

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

itemMapper.deleteByIDS(ids);

}

## 商品上架和下架

### 页面js分析

1. 页面url

<http://localhost:8091/item/instock>

$.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

//下架

@RequestMapping("/instock")

@ResponseBody

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

**try** {

**int** status = 2; //商品下架

itemService.updateStatus(ids,status);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201, "商品下架失败");

}

//商品上架

@RequestMapping("/reshelf")

@ResponseBody

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

**try** {

**int** status = 1; //商品下架

itemService.updateStatus(ids,status);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201, "商品下架失败");

}

### 编辑Service

//实现商品的上下架处理

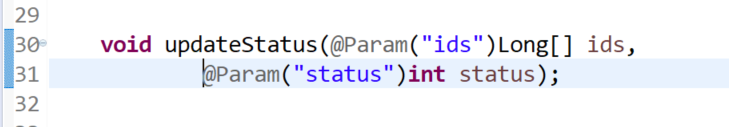
@Override

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

itemMapper.updateStatus(ids,status);

}

### 编辑Mapper接口



1. 编辑映射文件

<!--更新商品状态

collection中参数分析

collection="ids"

1.如果参数是数组 array 作业:自己完成批量删除

2.如果参数是List集合 list

3.如果参数是Map map中的key

-->

<update id=*"updateStatus"*>

update tb\_item set status = #{status}, updated = now()

where id in (

<foreach collection=*"ids"* item=*"id"* separator=*","*>

#{id}

</foreach>

)

</update>

## Log4j

### 日志说明

1. 添加log4j.jar包.

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

1. 添加log4j.配置文件

log4j.rootLogger=DEBUG,A1

log4j.logger.java.sql.PreparedStatement=DEBUG

log4j.appender.A1=org.apache.log4j.ConsoleAppender

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c]-[%p] %m%n

一般规定:日志的文件名称都叫**log4j.properties**/xml

文件目录:一般默认条件下将日志文件放到项目根目录下即可.

原因:

Log4j日志jar包中logManager中有静态代码块加载配置文件

static public final String DEFAULT\_CONFIGURATION\_FILE = "log4j.properties";

Static {

**if**(url == **null**) {

url = Loader.*getResource*(~~DEFAULT\_CONFIGURATION\_FILE~~);

}

}

### 日志应用

private static final Logger logger = Logger.getLogger(ItemController.class);

private static final org.slf4j.Logger logger = org.slf4j.LoggerFactory.getLogger(ItemController.class);

//实现商品的更新操作

@RequestMapping("/update")

@ResponseBody

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

**try** {

itemService.updateItem(item);

//System.out.println("sadfasdfasdfasdafsdfdsa");//执行性能太低

***logger***.info("~~~~~~~~~~~~~~~~~~~商品更新成功}");

***logger***.debug("~~~~~~~~~~~~~~~~~~~商品更新成功}");

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

***logger***.error(e.getMessage());

}

**return** SysResult.*build*(201,"修改商品数据失败");

}

## 富文本编辑器

### 富文本编辑器介绍

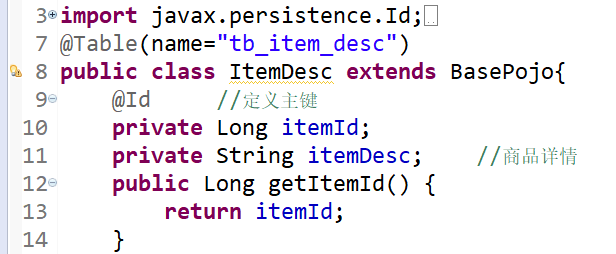
富文本编辑器，Rich Text Editor, 简称 RTE, 它提供类似于 Microsoft Word 的编辑功能，容易被不会编写 HTML 的用户并需要设置各种文本格式的用户所喜爱。它的应用也越来越广泛。最先只有 IE 浏览器支持，其它浏览器相继跟进，在功能的丰富性来说，还是 IE 强些。虽然没有一个统一的标准，但对于最基本的功能，各浏览器提供的 API 基本一致，从而使编写一个跨浏览器的富文本编辑器成为可能。

功能:

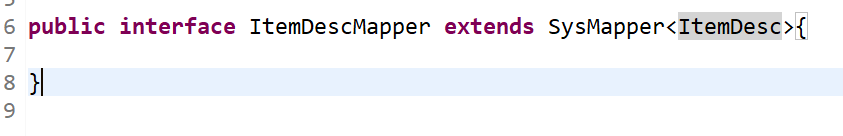


## 商品详情入库

### 编辑POJO



### 编辑Mapper接口



### 编辑Service

//当用户添加商品时,需要同时入库2张表数据

/\*\*

\* 1,2,3,4,6,8,9,10 主键自增时 即使数据回滚那么该id则不会被其他线程使用

\*/

@Override

**public** **void** saveItem(Item item,String desc) {

//封装数据

item.setStatus(1); //1表示正常

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

item.setUpdated(item.getCreated());

itemMapper.insert(item); //利用mybatis实现入库

//没有item的主键如何获取数据????

//Item itemDB = itemMapper.select(item);????

/\*\*

\* 入库itemDesc表

\* Preparing: INSERT INTO tb\_item (SELL\_POINT,PRICE,NUM,TITLE,CID,STATUS,ID,IMAGE,CREATED,BARCODE,UPDATED) VALUES ( ?,?,?,?,?,?,?,?,?,?,? )

Parameters: 第四届牛奶节盛大开启！囤奶日就今天了！精选牛奶低至三件七折，更有蜂蜜麦片超值赠品等你来拿！点此抢199-(String), 5900(Long), 100(Integer), 京东超市 澳大利亚 进口奶粉 德运 （Devondale）调制乳粉（全脂）成人奶粉 1kg 袋装(String), 298(Long), 1(Integer), null, (String), 2018-09-12 17:41:09.627(Timestamp), 10086(String), 2018-09-12 17:41:09.627(Timestamp)

Updates: 1

Executing: SELECT LAST\_INSERT\_ID()

Total: 1

\*/

ItemDesc itemDesc = **new** ItemDesc();

itemDesc.setItemId(item.getId());

itemDesc.setCreated(item.getCreated());

itemDesc.setUpdated(item.getCreated());

itemDesc.setItemDesc(desc);

itemDescMapper.insert(itemDesc);

}

### 编辑Controller

//实现商品新增

@RequestMapping("/save")

@ResponseBody

**public** SysResult saveItem(Item item,String desc){

**try** {

itemService.saveItem(item,desc);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201, "商品新增失败");

}