# 商品后台维护

## 商品关联删除

### 业务说明

商品表(tb\_item)与商品详情表(tb\_item\_desc)是一对一关系.当商品删除时,应该同时删除商品详情数据.

### Spring事务控制策略

事务控制策略:

当进行事务控制时.如果遇到运行时异常,则实现事务的回滚.

如果遇到检查异常/编译异常时spring容器不负责回滚事务.

编码策略:将编译异常转化为运行时异常.

### 编辑Service

@Override

@Transactional//添加事务控制

**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);

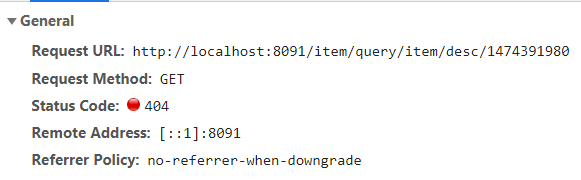
itemMapper.deleteBatchIds(idList);

itemDescMapper.deleteBatchIds(idList);

}

## 实现商品详情回显

### 页面分析



2.JS分析

$.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

/\*\*

\* 根据商品id号,查询商品详情信息

\*/

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

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

//需要将返回值传给页面进行数据展现

ItemDesc itemDesc = itemService.findItemDescById(itemId);

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

}

### 编辑Service

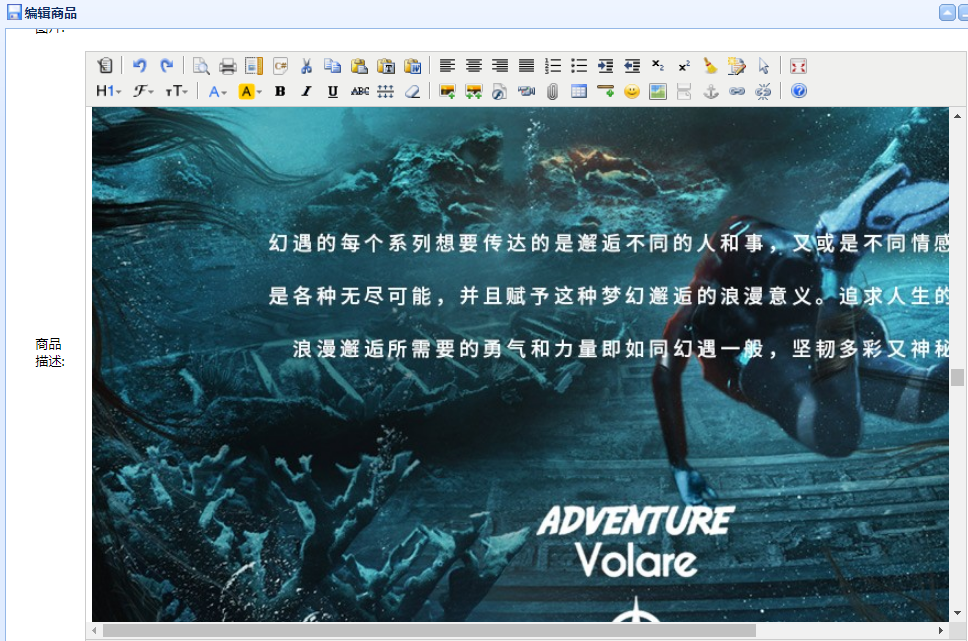
@Override

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

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

}

### 页面效果



## 重构更新

### 编辑Controller

/\*\*

\* 实现商品修改

\*/

@RequestMapping("/update")

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

itemService.updateItem(item,itemDesc);

**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

/\*\*

\* rollbackFor 指定异常类型回滚

\* rollbackFor=Exception.class

\* noRollbackFor=指定异常不回滚

\*/

@Transactional//添加事务控制

@Override

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

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

//sql: xxxx where id = 主键值

itemMapper.updateById(item);

itemDesc.setItemId(item.getId());

itemDesc.setUpdated(item.getUpdated());

itemDescMapper.updateById(itemDesc);

}

## 文件上传

### 入门案例

<body>

<h1>实现文件长传</h1>

<!--enctype="开启多媒体标签" -->

<form action=*"http://localhost:8091/file"* method=*"post"*

enctype=*"multipart/form-data"*>

<input name=*"fileImage"* type=*"file"* />

<input type=*"submit"* value=*"提交"*/>

</form>

</body>

### SpringMVC页面传参说明

页面进行post提交时,提交的参数如下.问题:cid/title由谁赋值????



原因:和页面中的标签的name属性有关. 并且name属性是必填项

<input type=*"hidden"* **name=*"cid"*** style="width: *280px*;" ></input>

重点知识:

由于现在通过SpringMVC进行业务数据获取,要求参数接收时必须与标签中的name属性一致.否则利用REquest获取参数时一定为null.

### 编辑Controller

@Controller //跳转页面时使用

**public** **class** FileController {

/\*\*

\* 1.确定url请求路径

\* http://localhost:8091/file

\* 2.获取用户提交参数

\*

\* 3.响应合适的页面

\*

\* 业务要求:

\* 将文件上传到D:\1-JT\images\goodgoodstudy.jpg

\* **@return**

\* **@throws** IOException

\* **@throws** IllegalStateException

\*/

@RequestMapping("/file")

**public** String file(MultipartFile fileImage) **throws** IllegalStateException, IOException {

File file = **new** File("D:/1-JT/images/goodgoodstudy.jpg");

//1.实现文件上传API

fileImage.transferTo(file);

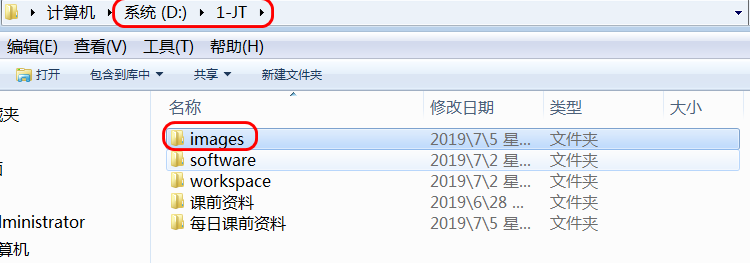
//2要求:重定向到file.jsp

**return** "redirect:/file.jsp";

}

}

### 准备文件夹



### 文件上传优化

@RequestMapping("/file")

**public** String file(MultipartFile fileImage) **throws** IllegalStateException, IOException {

//获取图片名称 xxxxx.xxx

String fileName = fileImage.getOriginalFilename();

//判断文件夹是否存在

File fileDir = **new** File("D:/1-JT/images");

**if**(!fileDir.exists()) {

//表示文件不存在

fileDir.mkdirs();

}

File file =

**new** File("D:/1-JT/images/"+fileName);

//1.实现文件上传API

fileImage.transferTo(file);

//2要求:重定向到file.jsp

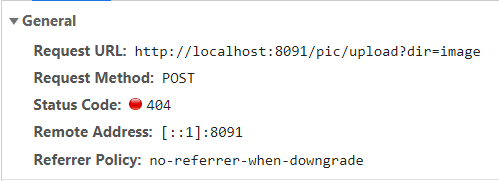
**return** "redirect:/file.jsp";

}

## 实现商品文件上传

### 页面分析

1. url分析



1. 用户传参

// 编辑器参数

kingEditorParams : {

**filePostName : "uploadFile",**

**uploadJson : '/pic/upload',**

dir : "image"

},

3.返回值JSON要求

{"error":0,"url":"图片的保存路径","width":图片的宽度,"height":图片的高度}

参数说明： 0代表是一张图片，如果是0，前台才可以解析并显示。1代表不是图片，

不显示如果不设置宽度和高度，则默认用图片原来的大小，所以不用设置

### 封装对象

@Data

@Accessors(chain = **true**)

@NoArgsConstructor

@AllArgsConstructor

**public** **class** EasyUI\_Image {

**private** Integer error=0;

//表示用户上传文件是是否有错

**private** String url; //图片的虚拟路径

**private** Integer width; //宽度

**private** Integer height; //高度

/\*\*

\* 多系统之间对象直接传递时必须序列化

\* manage.jt.com 后台系统 EasyUI\_Image~~序列化~字节数组

\* www.jt.com 前台系统 EasyUI\_Image~~反序列~~字节数组

\*

\* manage.jt.com JSON~~String

\* www.jt.com JSON~~String

\*/

}

### 正则表达式语法

**正则表达式**，又称规则表达式**。**（英语：Regular Expression，在代码中常简写为regex、regexp或RE），计算机科学的一个概念。正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本。

|  |  |
| --- | --- |
| 元字符 | 描述 |
| \ | 将下一个字符标记符、或一个向后引用、或一个八进制转义符。例如，“\\n”匹配\n。“\n”匹配换行符。序列“\\”匹配“\”而“\(”则匹配“(”。即相当于多种编程语言中都有的“转义字符”的概念。 |
| ^ | 匹配输入字行首。如果设置了RegExp对象的Multiline属性，^也匹配“\n”或“\r”之后的位置。 |
| $ | 匹配输入行尾。如果设置了RegExp对象的Multiline属性，$也匹配“\n”或“\r”之前的位置。 |
| \* | 匹配前面的子表达式任意次。例如，zo\*能匹配“z”，也能匹配“zo”以及“zoo”。\*等价于{0,}。 |
| + | 匹配前面的子表达式一次或多次(大于等于1次）。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。 |
| ? | 匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”。?等价于{0,1}。 |
| {*n*} | *n*是一个非负整数。匹配确定的*n*次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个o。 |
| {*n*,} | *n*是一个非负整数。至少匹配*n*次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o\*”。 |
| {*n*,*m*} | *m*和*n*均为非负整数，其中*n*<=*m*。最少匹配*n*次且最多匹配*m*次。例如，“o{1,3}”将匹配“fooooood”中的前三个o为一组，后三个o为一组。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格。 |
| ? | 当该字符紧跟在任何一个其他限制符（\*,+,?，{*n*}，{*n*,}，{*n*,*m*}）后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+”将尽可能多地匹配“o”，得到结果[“oooo”]，而“o+?”将尽可能少地匹配“o”，得到结果 ['o', 'o', 'o', 'o'] |
| .点 | 匹配除“\n”和"\r"之外的任何单个字符。要匹配包括“\n”和"\r"在内的任何字符，请使用像“[\s\S]”的模式。 |
| (pattern) | 匹配pattern并获取这一匹配。所获取的匹配可以从产生的Matches集合得到，在VBScript中使用SubMatches集合，在JScript中则使用$0…$9属性。要匹配圆括号字符，请使用“\(”或“\)”。 |
| (?:pattern) | 非获取匹配，匹配pattern但不获取匹配结果，不进行存储供以后使用。这在使用或字符“(|)”来组合一个模式的各个部分时很有用。例如“industr(?:y|ies)”就是一个比“industry|industries”更简略的表达式。 |
| (?=pattern) | 非获取匹配，正向肯定预查，在任何匹配pattern的字符串开始处匹配查找字符串，该匹配不需要获取供以后使用。例如，“Windows(?=95|98|NT|2000)”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。 |
| (?!pattern) | 非获取匹配，正向否定预查，在任何不匹配pattern的字符串开始处匹配查找字符串，该匹配不需要获取供以后使用。例如“Windows(?!95|98|NT|2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。 |
| (?<=pattern) | 非获取匹配，反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“(?<=95|98|NT|2000)Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”。  \*python的正则表达式没有完全按照正则表达式规范实现，所以一些高级特性建议使用其他语言如java、scala等 |
| (?<!patte\_n) | 非获取匹配，反向否定预查，与正向否定预查类似，只是方向相反。例如“(?<!95|98|NT|2000)Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”。  \*python的正则表达式没有完全按照正则表达式规范实现，所以一些高级特性建议使用其他语言如java、scala等 |
| x|y | 匹配x或y。例如，“z|food”能匹配“z”或“food”(此处请谨慎)。“[z|f]ood”则匹配“zood”或“food”。 |
| [xyz] | 字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”。 |
| [^xyz] | 负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“plin”任一字符。 |
| [a-z] | 字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。  注意:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围; 如果出字符组的开头,则只能表示连字符本身. |
| [^a-z] | 负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。 |
| \b | 匹配一个单词的边界，也就是指单词和空格间的位置（即正则表达式的“匹配”有两种概念，一种是匹配字符，一种是匹配位置，这里的\b就是匹配位置的）。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”；“\b1\_”可以匹配“1\_23”中的“1\_”，但不能匹配“21\_3”中的“1\_”。 |
| \B | 匹配非单词边界。“er\B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”。 |
| \cx | 匹配由x指明的控制字符。例如，\cM匹配一个Control-M或回车符。x的值必须为A-Z或a-z之一。否则，将c视为一个原义的“c”字符。 |
| \d | 匹配一个数字字符。等价于[0-9]。grep 要加上-P，perl正则支持 |
| \D | 匹配一个非数字字符。等价于[^0-9]。grep要加上-P，perl正则支持 |
| \f | 匹配一个换页符。等价于\x0c和\cL。 |
| \n | 匹配一个换行符。等价于\x0a和\cJ。 |
| \r | 匹配一个回车符。等价于\x0d和\cM。 |
| \s | 匹配任何不可见字符，包括空格、制表符、换页符等等。等价于[ \f\n\r\t\v]。 |
| \S | 匹配任何可见字符。等价于[^ \f\n\r\t\v]。 |
| \t | 匹配一个制表符。等价于\x09和\cI。 |
| \v | 匹配一个垂直制表符。等价于\x0b和\cK。 |
| \w | 匹配包括下划线的任何单词字符。类似但不等价于“[A-Za-z0-9\_]”，这里的"单词"字符使用Unicode字符集。 |
| \W | 匹配任何非单词字符。等价于“[^A-Za-z0-9\_]”。 |
| \x*n* | 匹配*n*，其中*n*为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，“\x41”匹配“A”。“\x041”则等价于“\x04&1”。正则表达式中可以使用ASCII编码。 |
| \*num* | 匹配*num*，其中*num*是一个正整数。对所获取的匹配的引用。例如，“(.)\1”匹配两个连续的相同字符。 |
| \*n* | 标识一个八进制转义值或一个向后引用。如果\*n*之前至少*n*个获取的子表达式，则*n*为向后引用。否则，如果*n*为八进制数字（0-7），则*n*为一个八进制转义值。 |
| \*nm* | 标识一个八进制转义值或一个向后引用。如果\*nm*之前至少有*nm*个获得子表达式，则*nm*为向后引用。如果\*nm*之前至少有*n*个获取，则*n*为一个后跟文字*m*的向后引用。如果前面的条件都不满足，若*n*和*m*均为八进制数字（0-7），则\*nm*将匹配八进制转义值*nm*。 |
| \*nml* | 如果*n*为八进制数字（0-7），且*m*和*l*均为八进制数字（0-7），则匹配八进制转义值*nml*。 |
| \u*n* | 匹配*n*，其中*n*是一个用四个十六进制数字表示的Unicode字符。例如，\u00A9匹配版权符号（&copy;）。 |
| \p{P} | 小写 p 是 property 的意思，表示 Unicode 属性，用于 Unicode 正表达式的前缀。中括号内的“P”表示Unicode 字符集七个字符属性之一：标点字符。  其他六个属性：  L：字母；  M：标记符号（一般不会单独出现）；  Z：分隔符（比如空格、换行等）；  S：符号（比如数学符号、货币符号等）；  N：数字（比如阿拉伯数字、罗马数字等）；  C：其他字符。  *\*注：此语法部分语言不支持，例：javascript。* |
| \<  \> | 匹配词（word）的开始（\<）和结束（\>）。例如正则表达式\<the\>能够匹配字符串"for the wise"中的"the"，但是不能匹配字符串"otherwise"中的"the"。注意：这个元字符不是所有的软件都支持的。 |
| ( ) | 将( 和 ) 之间的表达式定义为“组”（group），并且将匹配这个表达式的字符保存到一个临时区域（一个正则表达式中最多可以保存9个），它们可以用 \1 到\9 的符号来引用。 |
| | | 将两个匹配条件进行逻辑“或”（Or）运算。例如正则表达式(him|her) 匹配"it belongs to him"和"it belongs to her"，但是不能匹配"it belongs to them."。注意：这个元字符不是所有的软件都支持的。 |

### 编辑VO对象

@Data

@Accessors(chain = **true**)

@NoArgsConstructor

@AllArgsConstructor

**public** **class** EasyUI\_Image {

**private** Integer error=0;

//表示用户上传文件是是否有错

**private** String url; //图片的虚拟路径

**private** Integer width; //宽度

**private** Integer height; //高度

/\*\*

\* 多系统之间对象直接传递时必须序列化

\* manage.jt.com 后台系统 EasyUI\_Image~~序列化~字节数组

\* www.jt.com 前台系统 EasyUI\_Image~~反序列~~字节数组

\*

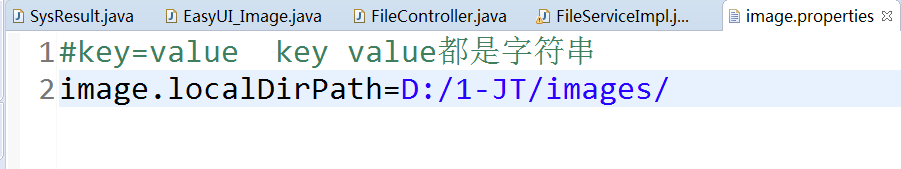
\* manage.jt.com JSON~~String

\* www.jt.com JSON~~String

\*/

}

### 编辑Pro文件



### 编辑Controller

/\*\*

\* 实现用户文件上传

\*/

@RequestMapping("/pic/upload")

@ResponseBody //将数据转化为json

**public** EasyUI\_Image fileUpload(MultipartFile uploadFile) {

**return** fileService.fileUpload(uploadFile);

}

### 编辑Service

@Service

//加载配置文件,将数据交给Spring容器管理

@PropertySource("classpath:/properties/image.properties")

**public** **class** FileServiceImpl **implements** FileService {

/\*\*

\* 由于成员变量将路径写死,扩展不易.最好方式

\* 应该写到配置文件中动态获取

\* YML properties

\*/

@Value("${image.localDirPath}")

**private** String localDirPath; //= "D:/1-JT/images/"; //定义本地磁盘路径

/\*\*

问题1: 校验文件类型是否为图片 如何校验?

利用后缀校验

问题2: 如何防止恶意文件上传?

将文件交给工具API校验从中获取宽高

问题3: 众多图片如何保存?

分文件存储: 按照yyyy/MM/dd

问题4: 文件如果重名如何处理?

自定义UUID为文件名称

\* 文件上传思路:

\* 1.获取用户文件名称用户校验

\* 2.校验文件名称是否为图片

\* 3.利用工具API校验图片的宽度和高度

\* 4.以时间格式创建文件夹保存数据 yyyy/MM/dd

\* 5.判断文件夹是否存在 不存在 新建文件目录

\* 6.采用UUID为文件名称,防止文件重名 32位16进制数=2^

\* 1位16进制数 2^4 0-9-A-F

\* 32位共有 2^128种可能性

\* 2^4 \* 2^4 \*2^4...

\* (2^4)^32=2^128次

\*/

@Override

**public** EasyUI\_Image fileUpload(MultipartFile uploadFile) {

EasyUI\_Image ui\_Image = **new** EasyUI\_Image();

//1.获取文件名称 a.jpg

String fileName = uploadFile.getOriginalFilename();

//2.校验文件名称 正则表达式

fileName = fileName.toLowerCase();//将字符转化为小写

**if**(!fileName.matches("^.+\\.(jpg|png|gif)$")) {

ui\_Image.setError(1); //表示文件上传有误

**return** ui\_Image;

}

//3.利用API读取用户提交数据

**try** {

BufferedImage bufferedImage =

ImageIO.*read*(uploadFile.getInputStream());

**int** height = bufferedImage.getHeight();

**int** width = bufferedImage.getWidth();

//如果有一项为0 表示不是图片

**if**(height==0 || width==0) {

ui\_Image.setError(1);

**return** ui\_Image;

}

//封装图片数据

ui\_Image.setHeight(height)

.setWidth(width);

//4.以时间格式创建文件夹 D:/1-JT/images/yyyy/MM/dd

String datePathDir =

**new** SimpleDateFormat("yyyy/MM/dd")

.format(**new** Date());

// D:/1-JT/images/yyyy/MM/dd

String realDirPath = localDirPath + datePathDir;

File dirFile = **new** File(realDirPath);

**if**(!dirFile.exists()) {

dirFile.mkdirs();

}

//6.采用UUID命名文件名称 替换中间的-

String uuid = UUID.*randomUUID*()

.toString()

.replace("-","");

//截串 含头不含尾 abc.jpg

String fileType =

fileName.substring(fileName.lastIndexOf("."));

String realName = uuid + fileType;

//7.实现文件上传 D:/1-JT/images/yyyy/MM/dd/uuid.jpg

String realFilePath = realDirPath +"/"+ realName;

uploadFile.transferTo(**new** File(realFilePath));

//8.编辑虚拟路径数据返回

ui\_Image.setUrl("https://img14.360buyimg.com/n0/jfs/t1/35/35/4260/115519/5b98a404E67e837b4/c75e10edb4fb7986.jpg");

} **catch** (Exception e) {

e.printStackTrace();

ui\_Image.setError(1);//对象转化时异常

**return** ui\_Image;

}

**return** ui\_Image;

}

}