淘淘商城第四天

# 第三天内容

1. nginx的反向代理、负载均衡
2. fastDFS，分布式文件系统。图片服务器。访问图片使用nginx+fastDFS的插件。
3. 使用FastDFS的客户端上传图片。
4. 图片上传功能实现。
5. Springmvc和spring父子容器。

# 课程计划

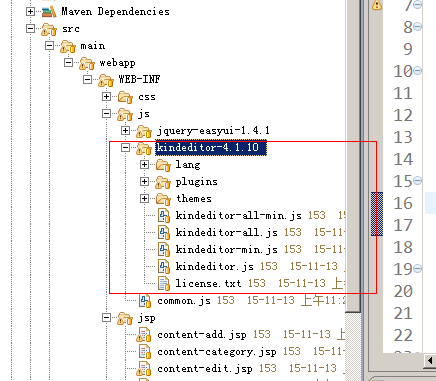
1. 富文本编辑器的使用KindEditor。
2. 商品添加功能的实现。
3. 规格参数的实现

# 富文本编辑器的使用

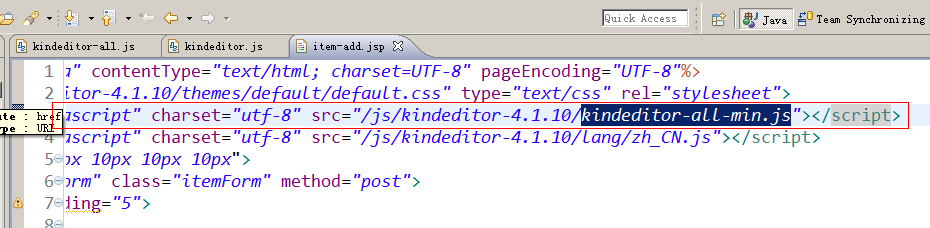
## 使用方法

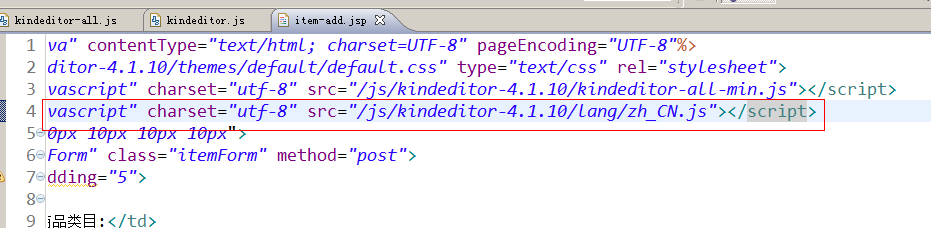
第一步：从KindEditor的官方网站下载源码。http://kindeditor.net/demo.php

第二步：解压缩，把js源码添加到工程中。

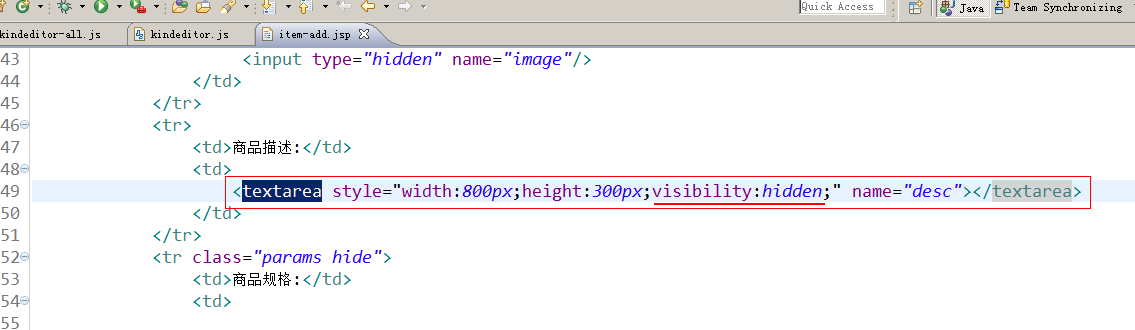


第三步：把*kindeditor-all-min.js*引入到jsp中

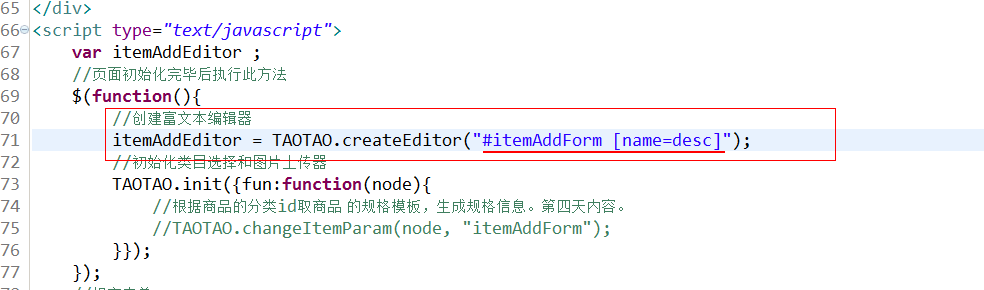
  
第四步：把kindEditor的语言包引入到jsp

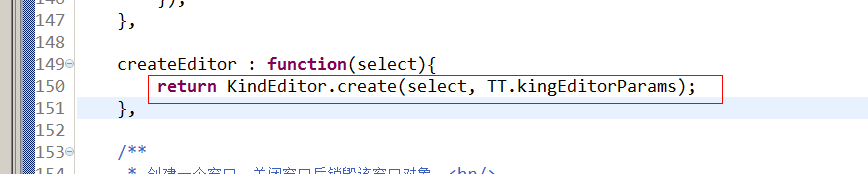


第五步：创建一个textArea控件，作为富文本编辑器的数据源。



第六步：编写js代码初始化KindEditor控件。需要指定textArea控件。





第七步：在提交表单之前，先把富文本编辑器中的内容同步到textArea控件中。

Sync（）方法实现。



## 流行的编辑器

1. KindEditor
2. <http://ueditor.baidu.com/website/>
3. Ckeditor <http://ckeditor.com/>

# 商品添加的实现

## 分析

### 数据库

相关的表：tb\_item、tb\_item\_desc

执行insert操作，可以使用逆向工程生成的代码。

商品id应该是一串数字。

可以使用毫秒+两位随机数生成一个id。

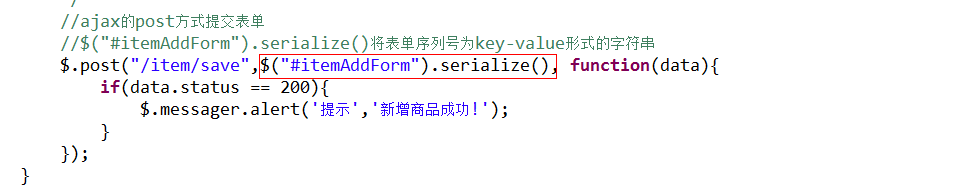
使用工具类：



### 页面

提交请求的url：/item/save

请求的参数：表单中的内容。



可以生成如下格式的字符串：

Name=zhangsan&password=123&id=szfda2313as

返回值：

Json数据，应该包含一个status属性，代码状态码200正常。

TaotaoResult

## Dao层

可以使用逆向工程生成的代码。

## Service层

1、接收TbItem对象，String desc 商品描述两个参数。

2、在TbItem对象中补全属性，包括id、status、create、update。

3、调用Mapper的insert方法插入数据。

4、创建一个TbItemDesc对象，补全属性。

5、调用Mapper方法插入到商品描述表。

6、返回TaotaoResult对象。

参数：TbItem、String

返回值：TaotaoResult

|  |
| --- |
| @Override  **public** TaotaoResult createItem(TbItem item, String desc) {  // 生成商品id  **long**itemId = IDUtils.*genItemId*();  // 补全TbItem属性  item.setId(itemId);  // '商品状态，1-正常，2-下架，3-删除'  item.setStatus((**byte**) 1);  // 创建时间和更新时间  Date date = **new** Date();  item.setCreated(date);  item.setUpdated(date);  // 插入商品表  itemMapper.insert(item);  // 商品描述  TbItemDesc itemDesc = **new** TbItemDesc();  itemDesc.setItemId(itemId);  itemDesc.setItemDesc(desc);  itemDesc.setCreated(date);  itemDesc.setUpdated(date);  // 插入商品描述数据  itemDescMapper.insert(itemDesc);  **return** TaotaoResult.*ok*();  } |

## Controller层

1、接收页面提交的内容，可以使用TbItem接收表单中的内容。还需要使用一个字符串类型的参数接收商品描述。

2、调用Service插入商品信息。

3、返回TaotaoResult。（json格式）

|  |
| --- |
| @RequestMapping(value="/item/save", method=RequestMethod.***POST***)  @ResponseBody  **public** TaotaoResult createItem(TbItem item, String desc) {  TaotaoResult result = itemService.createItem(item, desc);  **return**result;  } |

# 商品的规格参数

## 分析

规格参数：

1. 同一类商品的规格分类相同。
2. 同一类商品的规格项一致。
3. 不同商品的规格参数不同。

思路：

第一种方式：可以把数据保存到表中。

n

1

1

1

规格参数-商品规格参数

n

1

商品表

规格参数-规格项表

n

1

n

1

规格参数-分类表

商品分类表

查询的sql语句比较复杂，需要关联的表很多，会给数据库造成很大压力。不推荐使用此方法。

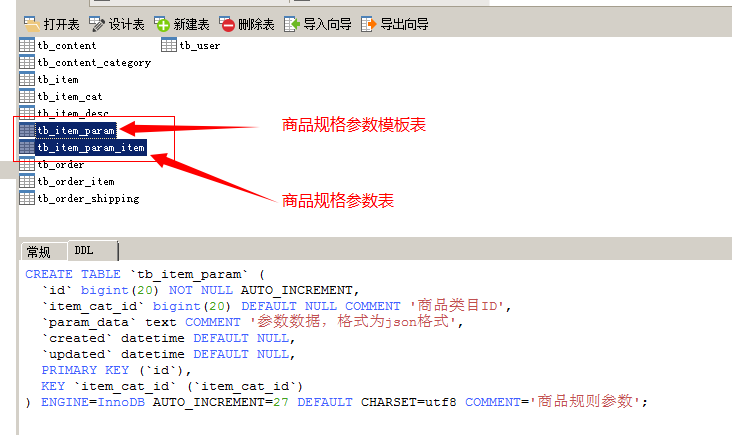
第二种方法：

1. 可以把规格分组和规格项使用json数据表示，保存到一个字段中。规格参数模板。
2. 添加商品时读取规格参数的模板，根据模板生成表单，供客户输入规格参数。
3. 把动态表单中的内容读取出来生成json数据保存到数据库中。
4. 展示规格参数时，根据商品id读取规格参数json数据生成html展示到jsp页面。



## 数据库

1. 规格参数模板表（商品分类id）
2. 规格参数表（商品id）



## 规格参数模板管理

功能列表：

1. 规格参数模板列表
2. 添加规格参数模板
3. 修改规格参数模板
4. 删除模板

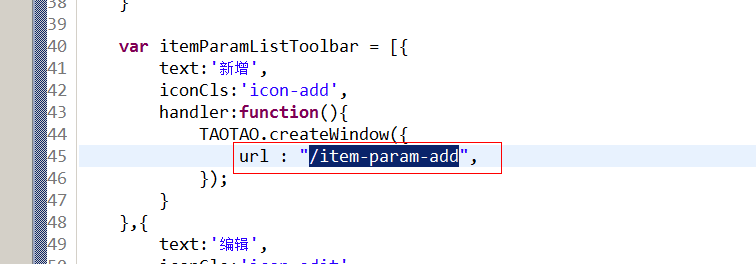
### 规格参数列表

从tb\_item\_param中查询数据展示到jsp，单表查询，实现分页。

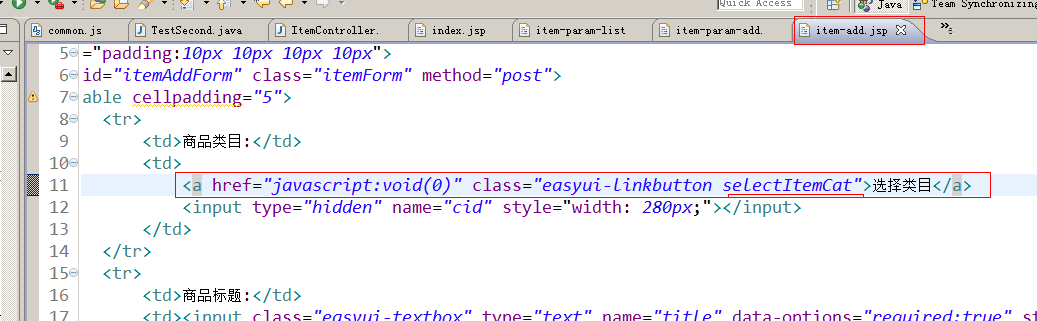
作业！！

### 新增规格参数模板

#### 展示规格参数添加页面



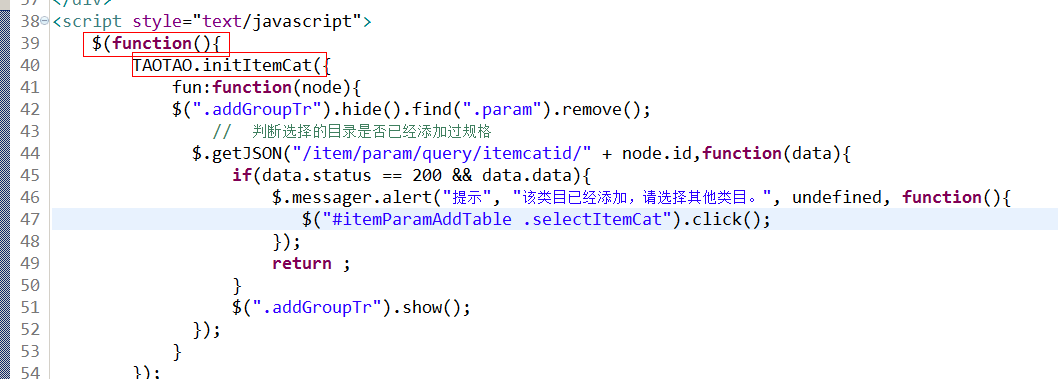
类目选择功能实现：



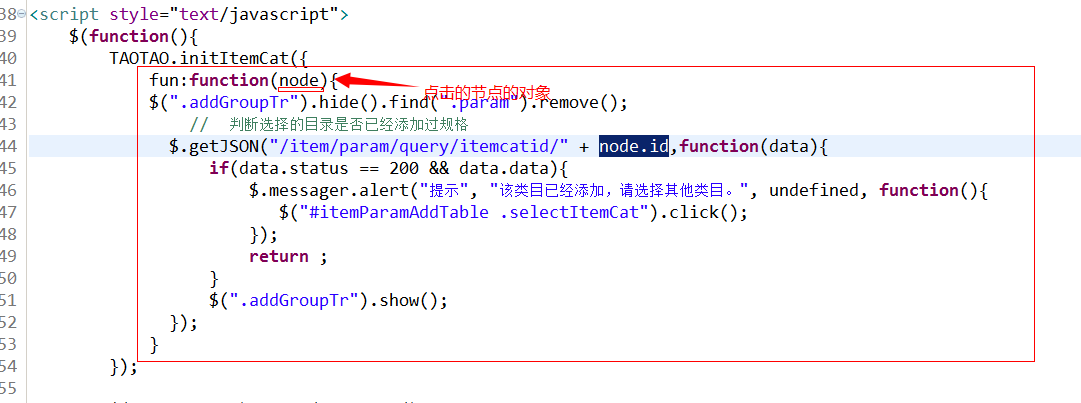
#### 判断模板是否存在

##### 分析

初始化类目选择组件：



回调函数：



根据商品的分类id查询tb\_item\_param表，如果查询到结果说明此模板已经添加。

请求的url：

/item/param/query/itemcatid/{cid}

参数：cid

返回结果：TaotaoResult

##### Dao层

可以使用逆向工程

##### Service层

接收参数cid，根据cid调用mapper查询模板是否存在，如果存在应该返回TaotaoResult，并且把查询的结果放到TaotaoResult的data属性。如果不存在，直接返回TaotaoResult。

参数：Long cid

返回值：TaotaoResult

|  |
| --- |
| @Service  **publicclass** ItemParamServiceImpl **implements** ItemParamService {  @Autowired  **private** TbItemParamMapper itemParamMapper;    @Override  **public** TaotaoResult getItemParamByCid(Long cid) {  //根据cid查询规格参数模板  TbItemParamExample example = **new** TbItemParamExample();  Criteria criteria = example.createCriteria();  criteria.andItemCatIdEqualTo(cid);  //执行查询  List<TbItemParam>list = itemParamMapper.selectByExampleWithBLOBs(example);  //判断是否查询到结果  **if** (list != **null**&&list.size() > 0) {  TbItemParam itemParam = list.get(0);  **return** TaotaoResult.*ok*(itemParam);  }  **return** TaotaoResult.*ok*();  }  } |

##### Controller

请求的url：

/item/param/query/itemcatid/{cid}

从url中取cid，调用Service查询规格参数模板。得到TaoTaoResult对象，返回TaotaoResult（json数据）。

|  |
| --- |
| @Controller  @RequestMapping("/item/param")  **publicclass** ItemParamController {  @Autowired  **private** ItemParamService itemParamService;    @RequestMapping("/query/itemcatid/{cid}")  @ResponseBody  **public**TaotaoResult getItemCatByCid(@PathVariable Long cid) {  TaotaoResult result = itemParamService.getItemParamByCid(cid);  **return**result;  }  } |

#### 提交规格参数模板

##### 分析

请求的url：/item/param/save/{cid}

请求的参数：Long cid、String paramData

返回值：TaotaoResult

##### Dao层

向tb\_item\_param表中插入记录。逆向工程代码。

##### Service层

接收参数：cid、paramData两个参数，创建一个tb\_item\_param表对应的pojo对象。设置好对象中的属性。调用insert方法，添加一条记录。返回TaotaoResult。

|  |
| --- |
| @Override  **public** TaotaoResult insertItemParam(Long cid, String paramData) {  //创建一个pojo  TbItemParam itemParam = **new** TbItemParam();  itemParam.setItemCatId(cid);  itemParam.setParamData(paramData);  itemParam.setCreated(**new** Date());  itemParam.setUpdated(**new** Date());  //插入记录  itemParamMapper.insert(itemParam);  **return** TaotaoResult.*ok*();  } |

##### Controller层

接收两个参数，从url中取cid、从参数中取 paramData。调用Service插入记录，返回TaotaoResult（json），需要使用@ResourceBody

|  |
| --- |
| @RequestMapping("/save/{cid}")  @ResponseBody  **public**TaotaoResult insertItemParam(@PathVariable Long cid, String paramData) {  TaotaoResult result = itemParamService.insertItemParam(cid, paramData);  **return**result;  } |

### 规格参数的编辑、删除

作业！！！

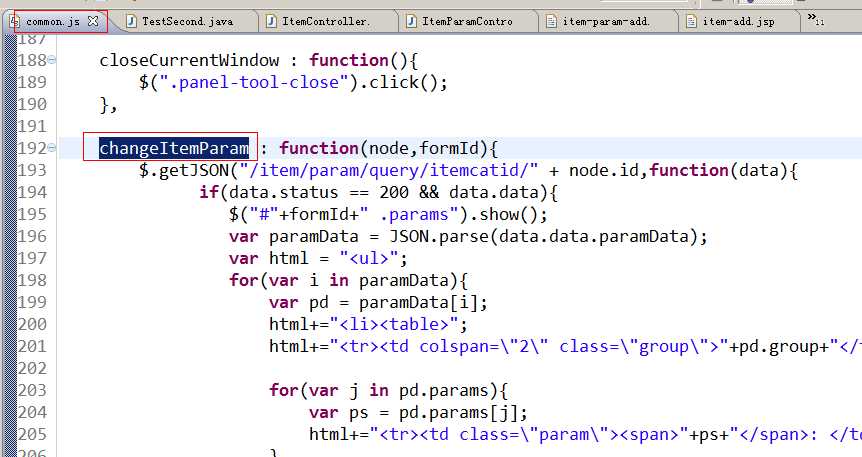
# 规格参数模板的使用

## 展示规格参数模板

应该在商品添加或者商品修改时，根据商品的分类id查询此商品分类对应的规格参数模板。根据规格参数模板，生成一个表单供用户使用。



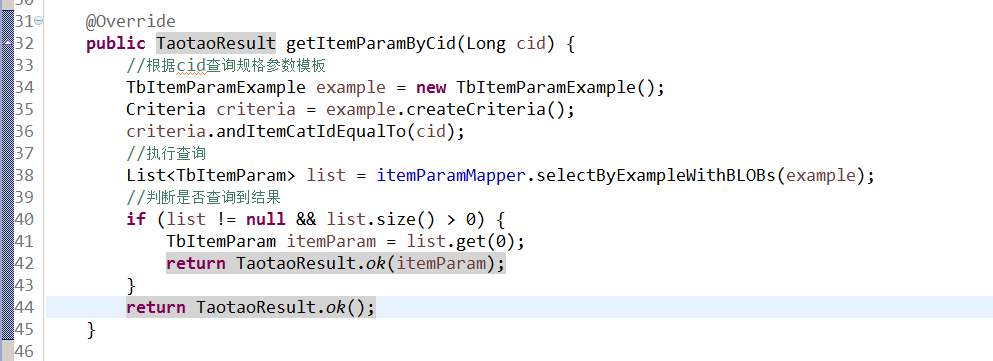
当商品类目选择完成后，调用此方法：

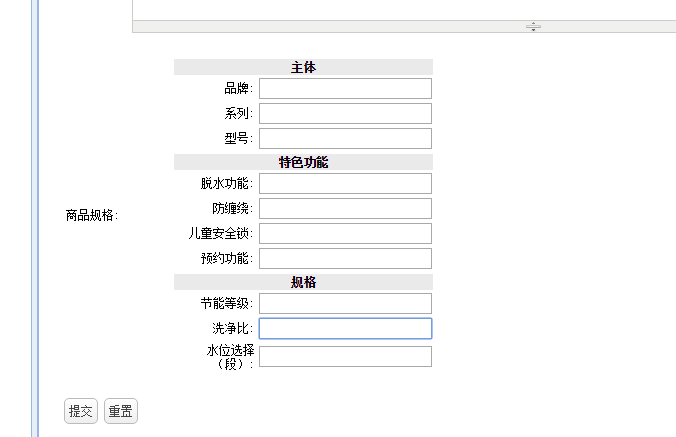


展示规格参数表单：

请求的url：/item/param/query/itemcatid/{cid}

可以和其他功能公用同一个方法。





## 规格参数的提交

### 分析

在商品表单提交之前，先把规格参数的信息，转换成json数据。把json提交到后台插入到表中即可。

Json数据的格式：

1、生成的规格模板数据格式

*[*

*{*

*"group": "主体", //组名称*

*"params": [ // 记录规格成员*

*"品牌",*

*"型号",*

*"颜色",*

*"上市年份",*

*"上市月份"*

*]*

*}，*

*{*

*"group": "网络", //组名称*

*"params": [ // 记录规格成员*

*"4G",*

*"3G,*

*"2G"*

*]*

*}*

*]*

2、生成的规格数据格式

*[*

*{*

*"group": "主体",*

*"params": [*

*{*

*"k": "品牌",*

*"v": "苹果（Apple）"*

*},*

*{*

*"k": "型号",*

*"v": "iPhone 6 A1589"*

*},*

*{*

*"k": "智能机",*

*"v": "是"*

*}*

*]*

*}*

*]*

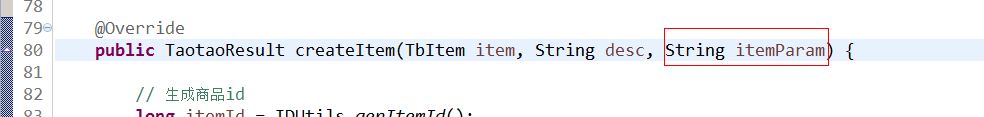
生成规格参数json字符串的处理：

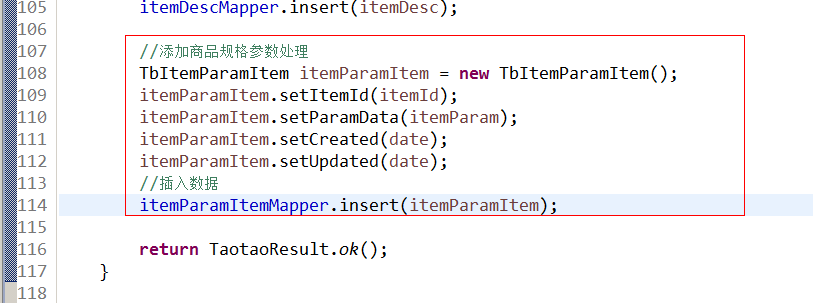


只需要在Controller中添加一个参数，接收商品的规格参数即可。

Service中也需要添加 一个参数，增加插入规格参数表的处理。

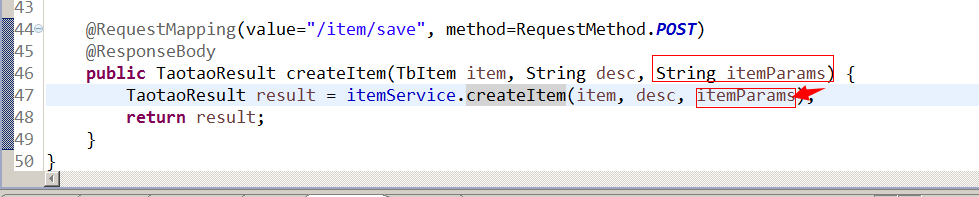
### Service





### Controller

添加一个接收商品规格参数的形参。



# 展示规格参数

## 分析

从表中把规格参数json数据取处理，可以在java代码中解析json数据生成html展示到jsp页面。

根据商品id查询规格参数。

请求的url：/item/{itemId}

返回结果：String（逻辑视图）

## Dao层

单表查询，可以使用逆向工程。

## Service

接收商品id，根据商品id取规格参数。可以把json转换成java对象。遍历java对象，生成html，返回。

参数：商品id

返回值：字符串（html片段）

|  |
| --- |
| @Override  **public**String getItemParamHtml(Long itemId) {  // 根据商品id查询规格参数  TbItemParamItemExample example = **new** TbItemParamItemExample();  com.taotao.pojo.TbItemParamItemExample.Criteria criteria = example.createCriteria();  criteria.andItemIdEqualTo(itemId);  //执行查询  List<TbItemParamItem>list = itemParamItemMapper.selectByExample(example);  **if** (list == **null** || list.isEmpty()) {  **return**"";  }  //取规格参数  TbItemParamItem itemParamItem = list.get(0);  //取json数据  String paramData = itemParamItem.getParamData();  //转换成java对象  List<Map>mapList = JsonUtils.*jsonToList*(paramData, Map.**class**);  //遍历list生成html  StringBuffer sb = **new** StringBuffer();    sb.append("<table cellpadding=\"0\" cellspacing=\"1\" width=\"100%\" border=\"0\" class=\"Ptable\">\n");  sb.append(" <tbody>\n");  **for** (Mapmap : mapList) {  sb.append(" <tr>\n");  sb.append(" <th class=\"tdTitle\" colspan=\"2\">"+map.get("group")+"</th>\n");  sb.append(" </tr>\n");  //取规格项  List<Map>mapList2 = (List<Map>) map.get("params");  **for** (Mapmap2 : mapList2) {  sb.append(" <tr>\n");  sb.append(" <td class=\"tdTitle\">"+map2.get("k")+"</td>\n");  sb.append(" <td>"+map2.get("v")+"</td>\n");  sb.append(" </tr>\n");  }  }  sb.append(" </tbody>\n");  sb.append("</table>");    **return**sb.toString();  } |

## Controller层

接收商品id，调用Service查询规格参数，返回html片段。把html片段传递给jsp。

参数：

1、商品id

2、Model

|  |
| --- |
| @RequestMapping("/item/{itemId}")  **public**String showItemParam(@PathVariable Long itemId, Model model) {  String html = itemService.getItemParamHtml(itemId);  model.addAttribute("html", html);  **return**"itemparam";  } |