

淘淘商城

第四天

**讲师：传智.入云龙**

# 课程计划

1. 实现商品规格参数

# 什么是商品规格参数

规格参数：



## 商品规格参数和类目关系

商品规格参数的模板是和商品类目关联的，不同的类目拥有不同的商品规格参数模板。

商品规格参数是和商品数据关联的，不同的商品拥有不同的商品规格参数数据。

## 实现思路

### 方案一

使用二维表来维护规格数据。

表一：规格组信息

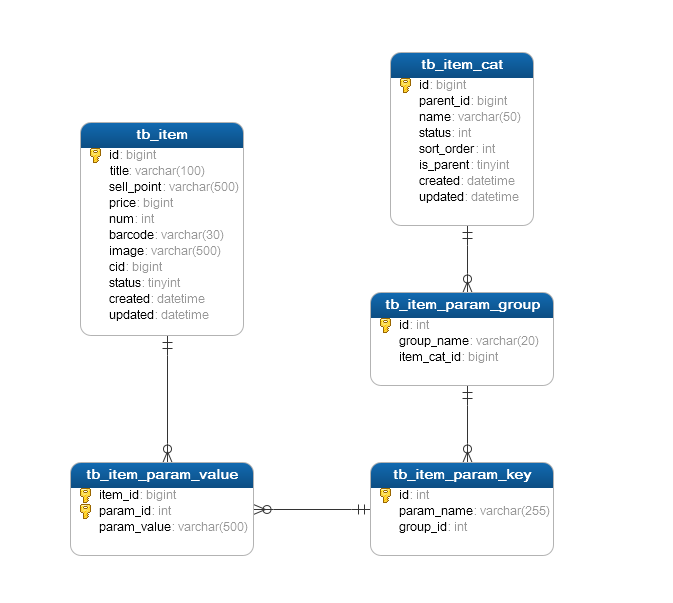
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **可以null** | **键** | **说明** |
| Id | Int |  | 否 | P | 主键（自增长） |
| group\_name | varchar | 20 | 否 |  | 规格分组名称 |
| item\_cat\_id | Int |  | 否 | F | 商品分类id（外键） |

表二：规格项信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **可以null** | **键** | **说明** |
| Id | Int |  | 否 | P | 主键（自增长） |
| param\_name | varchar | 20 | 否 |  | 规格项目名称 |
| group\_id | Int |  | 否 | F | 规格分组id（外键） |

表三：商品规格信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **列名** | **类型** | **长度** | **可以null** | **键** | **说明** |
| item\_id | Int |  | 否 | P | 商品id（联合主键） |
| param\_id | varchar |  | 否 | P | 规格项id（联合主键） |
| param\_value | varchar | 500 | 否 |  | 规格信息 |



问题：

1、需要创建的表比较多，表和表之间的关系复杂，查询时需要大量的关联。查询效率低。

2、如果展示的规格组或者是规格项需要调整实现麻烦，需要添加排序列。

3、维护不方便，如果删除某个规格分组信息，则所有与之相关的商品的规格信息都发生变化。

解决方案：

使用模板的方式来解决。

### 方案二

使用一种动态的方法描述参数模板，每个商品分类对应一个参数模板。添加商品信息时，根据规格参数模板生成规格参数录入项。保存商品时将规格参数生成一个字符串保存到数据库中。展示商品详情时，从数据库中取出规格参数信息，转换成html展示给用户。

如何设计一种动态结构的数据结构？

1. xml
2. Json

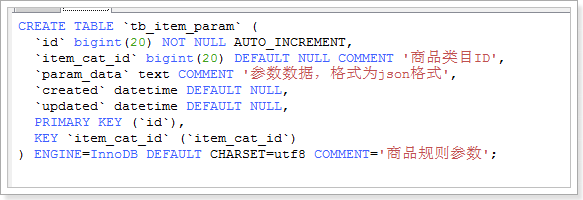
结论：使用json来实现模板以及规格参数数据。

**实现流程：**

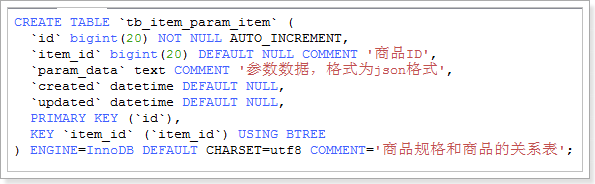


## 规格参数相关表

模板表：



数据表：



## 模板格式

*[*

*{*

*"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"*

*]*

*}*

*]*

## 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 "*

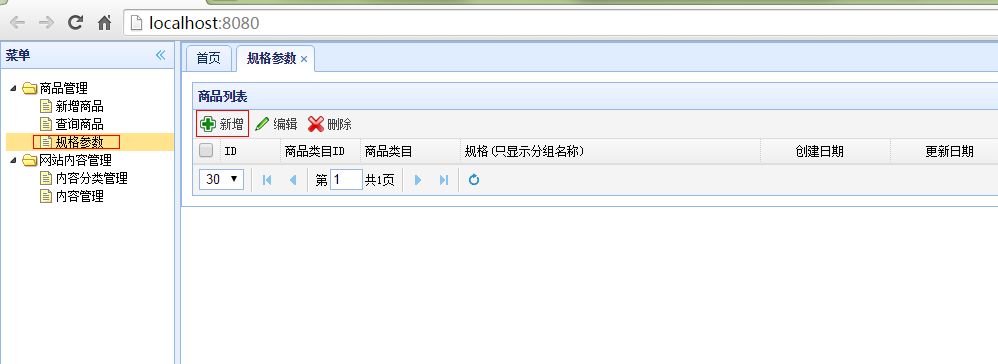
*}*

*]*

*}*

*]*

## 相关页面



## 规格参数模板添加

### 查询商品分类是否有模板

#### 请求url：



#### 返回值：

TaotaoResult

#### Mapper

使用逆向工程生成的mapper文件

#### Service

|  |
| --- |
| @Override  **public** TaotaoResult queryCatalogById(**long** cid) **throws** Exception {    //根据cid查询数据库  TbItemParamExample example = **new** TbItemParamExample();  Criteria criteria = example.createCriteria();  criteria.andItemCatIdEqualTo(cid);  List<TbItemParam> list = itemParamMapper.selectByExampleWithBLOBs(example);  //如果查询中有结果返回查询结果  **if** (list != **null** && !list.isEmpty()) {  **return** TaotaoResult.*ok*(list.get(0));  }  //查询成功但是没有查到数据  **return** TaotaoResult.*ok*();  } |

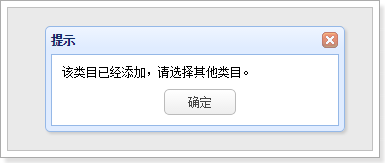
#### Controller

|  |
| --- |
| @RequestMapping("query/itemcatid/{cid}")  **public** TaotaoResult queryCatalogById(@PathVariable **long** cid) **throws** Exception {  //根据分类id查询列表  TaotaoResult result = itemParamService.queryCatalogById(cid);  **return** result;  } |

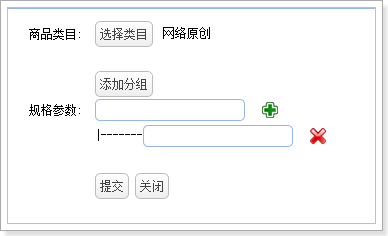
#### 前端JS实现

用户选择商品类目 🡺 根据类目判断模板是否存在 🡺 不存在，让用户新建模板

存在：



不存在：

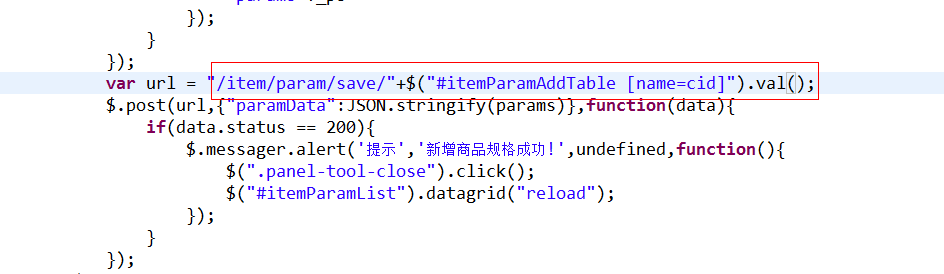


### 提交实现：

#### 前台js



#### 请求url



#### 响应的数据

TaotaoResult

#### Mapper

略

#### Service

|  |
| --- |
| @Override  **public** TaotaoResult saveItemParam(**long** cid, String itemParam) **throws** Exception {  //创建TbItemParam对象  TbItemParam param = **new** TbItemParam();  param.setItemCatId(cid);  param.setParamData(itemParam);  param.setCreated(**new** Date());  param.setUpdated(**new** Date());  //向数据库添加数据  itemParamMapper.insert(param);    **return** TaotaoResult.*ok*();    } |

#### Controller

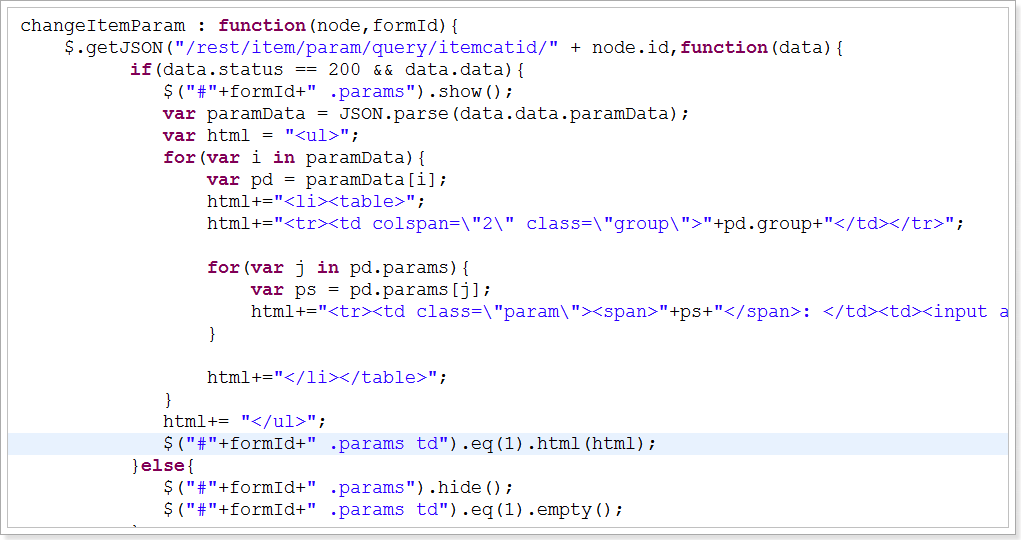
|  |
| --- |
| @RequestMapping("/save/{cid}")  @ResponseBody  **public** TaotaoResult saveItemParam(@PathVariable Long cid, String paramData) **throws** Exception {  TaotaoResult result = itemParamService.saveItemParam(cid, paramData);  **return** result;  } |

## 新增商品套用模板生成数据

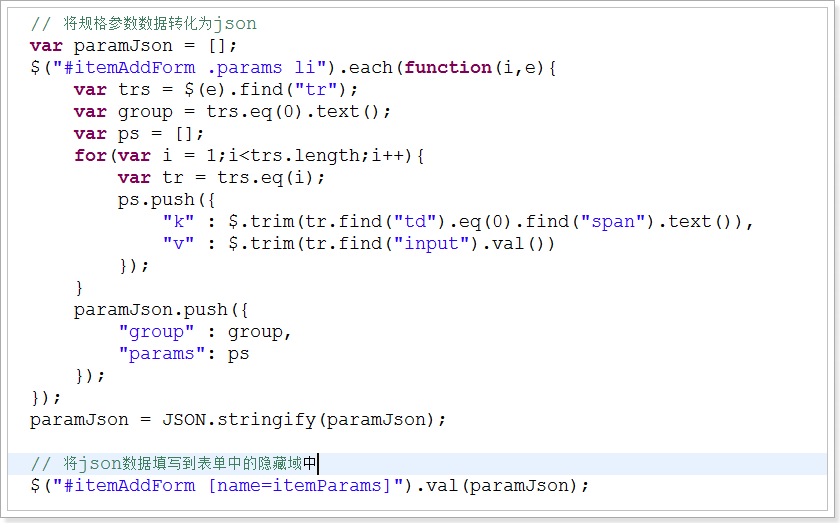
新增商品 🡺 选择类目 🡺 查找类目所对应的模板 🡺 生成表单



### 选择类目执行逻辑

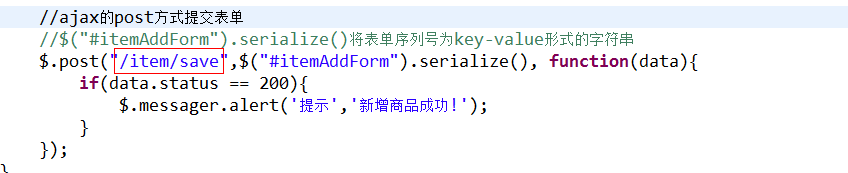


### 点击提交，将表单转化json数据



## 提交商品规格数据

### 请求的url

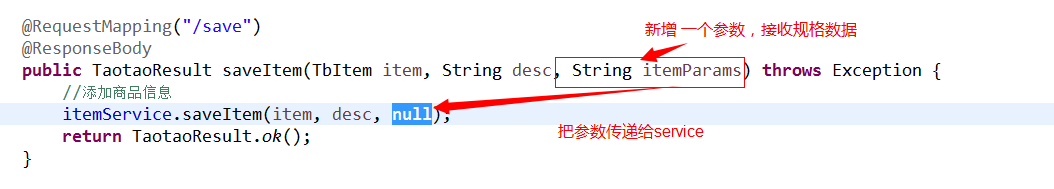


### 响应的数据格式

TaotaoResult

### Controller

由于此操作是在商品添加时同时提交所以，只需要修改商品添加controller即可。



### Service

在ItemServiceImpl的saveItem方法中添加插入规格数据的逻辑：

|  |
| --- |
| @Override  **public** **void** saveItem(TbItem item, String desc, String itemParams) **throws** Exception {  Date date = **new** Date();  //获得商品id  **long** id = IDUtils.*genItemId*();  //添加商品信息  item.setId(id);  //商品状态，1-正常，2-下架，3-删除  item.setStatus((**byte**) 1);  item.setCreated(date);  item.setUpdated(date);  itemMapper.insert(item);  //添加商品描述  //创建TbItemDesc对象  TbItemDesc itemDesc = **new** TbItemDesc();  //获得一个商品id  itemDesc.setItemId(id);  itemDesc.setItemDesc(desc);  itemDesc.setCreated(date);  itemDesc.setUpdated(date);  //插入数据  itemDescMapper.insert(itemDesc);  //添加商品规格  TbItemParamItem itemParamItem = **new** TbItemParamItem();  itemParamItem.setItemId(id);  itemParamItem.setParamData(itemParams);  itemParamItem.setCreated(date);  itemParamItem.setUpdated(date);    itemParamItemMapper.insert(itemParamItem);  } |

## 展示规格参数

### Service

|  |
| --- |
| @Service  **public** **class** ItemParamItemServiceImpl **implements** ItemParamItemService {    @Autowired  **private** TbItemParamItemMapper itemParamItemMapper;  @Override  **public** String geParamItemByItemId(**long** itemId) {    TbItemParamItemExample example = **new** TbItemParamItemExample();  Criteria criteria = example.createCriteria();  criteria.andItemIdEqualTo(itemId);  List<TbItemParamItem> list = itemParamItemMapper.selectByExampleWithBLOBs(example);  **if** (**null** == list || list.isEmpty()) {  **return** "";  }  //取出参数信息  TbItemParamItem itemParamItem = list.get(0);  String paramData = itemParamItem.getParamData();  //把json数据转换成java对象  List<Map> paramList = JsonUtils.*jsonToList*(paramData, Map.**class**);  //将参数信息转换成html  StringBuffer sb = **new** StringBuffer();  //sb.append("<div>");  sb.append("<table cellpadding=\"0\" cellspacing=\"1\" width=\"100%\" border=\"1\" class=\"Ptable\">\n");  sb.append(" <tbody>\n");  **for** (Map map : paramList) {  sb.append(" <tr>\n");  sb.append(" <th class=\"tdTitle\" colspan=\"2\">"+map.get("group")+"</th>\n");  sb.append(" </tr>\n");  List<Map> params = (List<Map>) map.get("params");  **for** (Map map2 : params) {  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>");  //sb.append("</div>");  **return** sb.toString();      }  } |

### Controller

|  |
| --- |
| @Controller  **public** **class** ItemParamItemController {  @Autowired  **private** ItemParamItemService itemParamItemService;    @RequestMapping("/items/param/item/{itemId}")  **public** String getItemParamItemById(@PathVariable Long itemId, Model model) {  String itemParamItem = itemParamItemService.geParamItemByItemId(itemId);  model.addAttribute("param1", itemParamItem);  **return** "item-param-show";  }  } |

## 规格参数列表（课后作业）

### 请求的url、参数

请求的url：

/item/param/list

参数：page、rows

http://localhost:8080/item/param/list?page=1&rows=30

### 返回值

使用EasyUIResult

### Mapper

使用逆向工程生成的mapper文件。

### Service

|  |
| --- |
| @Service  **public** **class** ItemParamServiceImpl **implements** ItemParamService {  @Autowired  **private** TbItemParamMapper itemParamMapper;    @Override  **public** EasyUIResult getItemParamList(Integer page, Integer rows) **throws** Exception {  //分页处理  PageHelper.*startPage*(page, rows);  //查询规格列表  List<TbItemParam> list = itemParamMapper.selectByExampleWithBLOBs(**new** TbItemParamExample());  //取分页信息  PageInfo<TbItemParam> pageInfo = **new** PageInfo<>(list);  //返回结果  EasyUIResult result = **new** EasyUIResult(pageInfo.getTotal(), list);    **return** result;  }  } |

### Controller

|  |
| --- |
| @Controller  @RequestMapping("/item/param")  **public** **class** ItemParamController {  @Autowired  **private** ItemParamService itemParamService;  @RequestMapping("/list")  @ResponseBody  **public** EasyUIResult getItemParamList(@RequestParam(defaultValue = "1") Integer page,  @RequestParam(defaultValue = "30") Integer rows) **throws** Exception {  //查询列表  EasyUIResult result = itemParamService.getItemParamList(page, rows);  **return** result;  }  } |

### 前端js实现

