

淘淘商城

第三天

**讲师：传智.入云龙**

# 课程计划

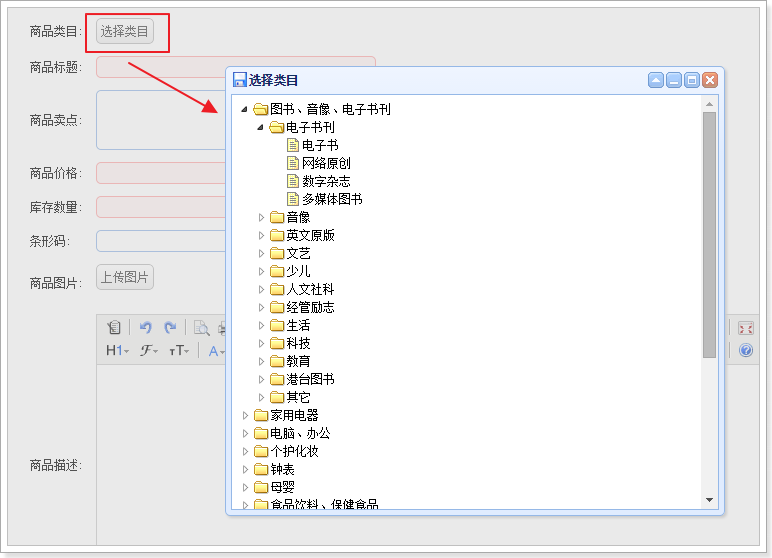
完成商品添加功能

1. 商品类目选择
2. 图片上传
3. 图片服务器搭建
4. kindEditor富文本编辑器的使用
5. 商品添加功能

# 实现商品类目选择功能

## 需求

在商品添加页面，点击“选择类目”显示商品类目列表：



## 实现步骤：

1. 按钮添加点击事件，弹出窗口，加载数据显示tree
2. 将选择类目的组件封装起来，通过TT.iniit()初始化，最终调用initItemCat()方法进行初始化
3. 创建数据库、以及tb \_item\_cat表，初始化数据
4. 编写Controller、Service、Mapper

## EasyUI tree数据结构

**数据结构中必须包含：**

Id：节点id

Text：节点名称

State：如果不是叶子节点就是close，叶子节点就是open。Close的节点点击后会在此发送请求查询子项目。



可以根据parentid查询分类列表。

## Mapper

使用逆向工程生成的mapper文件。

## Service

|  |
| --- |
| @Service  **public** **class** ItemCatServiceImpl **implements** ItemCatService {  @Autowired  **private** TbItemCatMapper itemCatMapper;    @Override  **public** List<TbItemCat> getItemCatList(Long parentId) **throws** Exception {    TbItemCatExample example = **new** TbItemCatExample();  //设置查询条件  Criteria criteria = example.createCriteria();  //根据parentid查询子节点  criteria.andParentIdEqualTo(parentId);  //返回子节点列表  List<TbItemCat> list = itemCatMapper.selectByExample(example);  **return** list;  }  } |

## Controller

|  |
| --- |
| @Controller  @RequestMapping("/item/cat")  **public** **class** ItemCatController {    @Autowired  **private** ItemCatService itemCatService;  @SuppressWarnings({ "rawtypes", "unchecked" })  @RequestMapping("/list")  @ResponseBody  //如果id为null是使用默认值，也就是parentid为0的分类列表  **public** List categoryList(@RequestParam(value="id", defaultValue="0") Long parentId) **throws** Exception {  List catList = **new** ArrayList();  //查询数据库  List<TbItemCat> list = itemCatService.getItemCatList(parentId);  **for** (TbItemCat tbItemCat : list) {  Map node = **new** HashMap<>();  node.put("id", tbItemCat.getId());  node.put("text", tbItemCat.getName());  //如果是父节点的话就设置成关闭状态，如果是叶子节点就是open状态  node.put("state", tbItemCat.getIsParent()?"closed":"open");  catList.add(node);  }  **return** catList;  }    } |

# 图片上传

## 图片服务器

### 传统项目中的图片管理

传统项目中，可以在web项目中添加一个文件夹，来存放上传的图片。例如在工程的根目录WebRoot下创建一个images文件夹。把图片存放在此文件夹中就可以直接使用在工程中引用。

优点：引用方便，便于管理

缺点：

1. 如果是分布式环境图片引用会出现问题。
2. 图片的下载会给服务器增加额外的压力

WEB工程

图片文件夹

例如：WebRoot/images

传统图片管理方式在分布式环境中的问题：

第一次请求：图片上传

用户请求

Tomcat1

Image（存放图片）

|-image.jpg

第二次请求：

下载图片

负载均衡服务器

Tomcat2

Image（存放图片）

说好的图片呢？？？？？

### 分布式环境的图片管理

Tomcat1

~~Image（存放图片）~~

~~|-image.jpg~~

Tomcat2

~~Image（存放图片）~~

负载均衡服务器

用户请求

http图片上传

图片服务器

Ip或者域名

Image.taotao.com  
nginx实现http服务

/images/2015/07/29/image.jpg

ftp服务接收上传

ftp上传到图片服务器

http请求图片

http图片上传

分布式环境一般都有一个专门的图片服务器存放图片。

我们使用虚拟机搭建一个专门的服务器来存放图片。在此服务器上安装一个nginx来提供http服务，安装一个ftp服务器来提供图片上传服务。

### 搭建图片服务器

第一步：安装vsftpd提供ftp服务

详见：vsftpd安装手册.doc

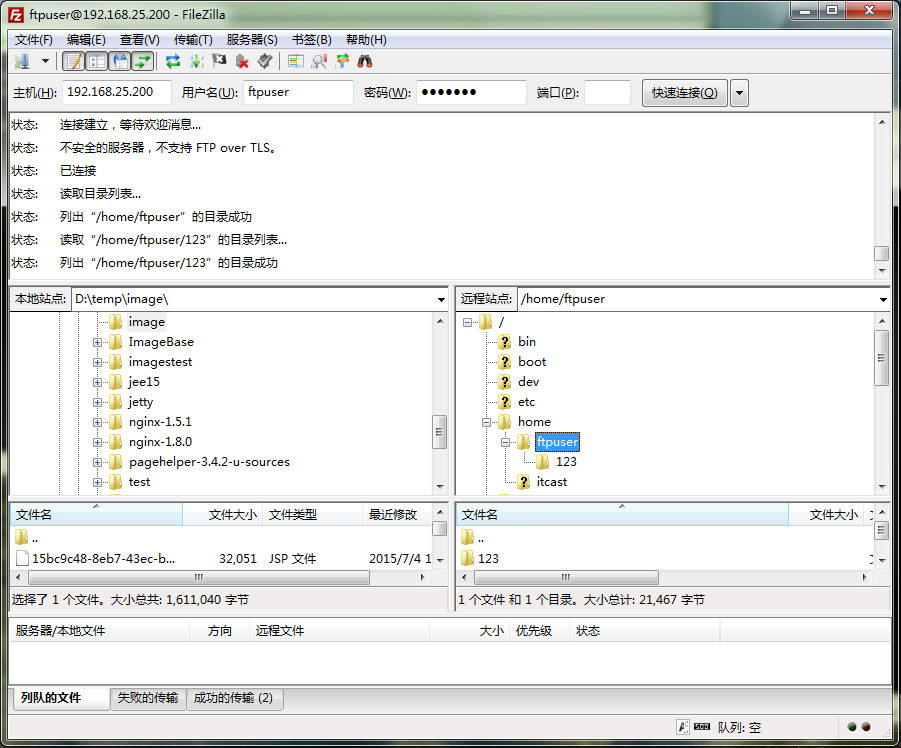
第二步：安装nginx提供http服务

详见：nginx安装手册.doc

### 测试图片服务器

#### ftp服务测试。

1. 使用ftp客户端



1. 使用java程序

ftp可以需要依赖commons-net-3.3.jar包。

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  FTPClient ftpClient = **new** FTPClient();  ftpClient.connect("192.168.25.200");  ftpClient.login("ftpuser", "ftpuser");  FileInputStream inputStream = **new** FileInputStream(**new** File("D:\\Documents\\Pictures\\pics\\21.jpg"));  ftpClient.setFileType(FTP.***BINARY\_FILE\_TYPE***);  ftpClient.storeFile("123.jpg", inputStream);  inputStream.close();    ftpClient.logout();  } |

#### http服务测试

* 1. 浏览器测试

### SpringMVC中实现图片上传

上传思路：

第一步：

导入common-fileupload的依赖

|  |
| --- |
| <!-- 文件上传组件 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  </dependency> |

第二步：

在SpringMVC配置文件中添加文件上传解析器

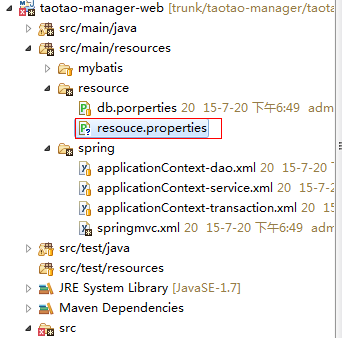
|  |
| --- |
| <!-- 定义文件上传解析器 -->  <bean id=*"multipartResolver"*  class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>  <!-- 设定默认编码 -->  <property name=*"defaultEncoding"* value=*"UTF-8"*></property>  <!-- 设定文件上传的最大值5MB，5\*1024\*1024 -->  <property name=*"maxUploadSize"* value=*"5242880"*></property>  </bean> |

### Service实现

#### 获取资源配置文件的内容

第一步：

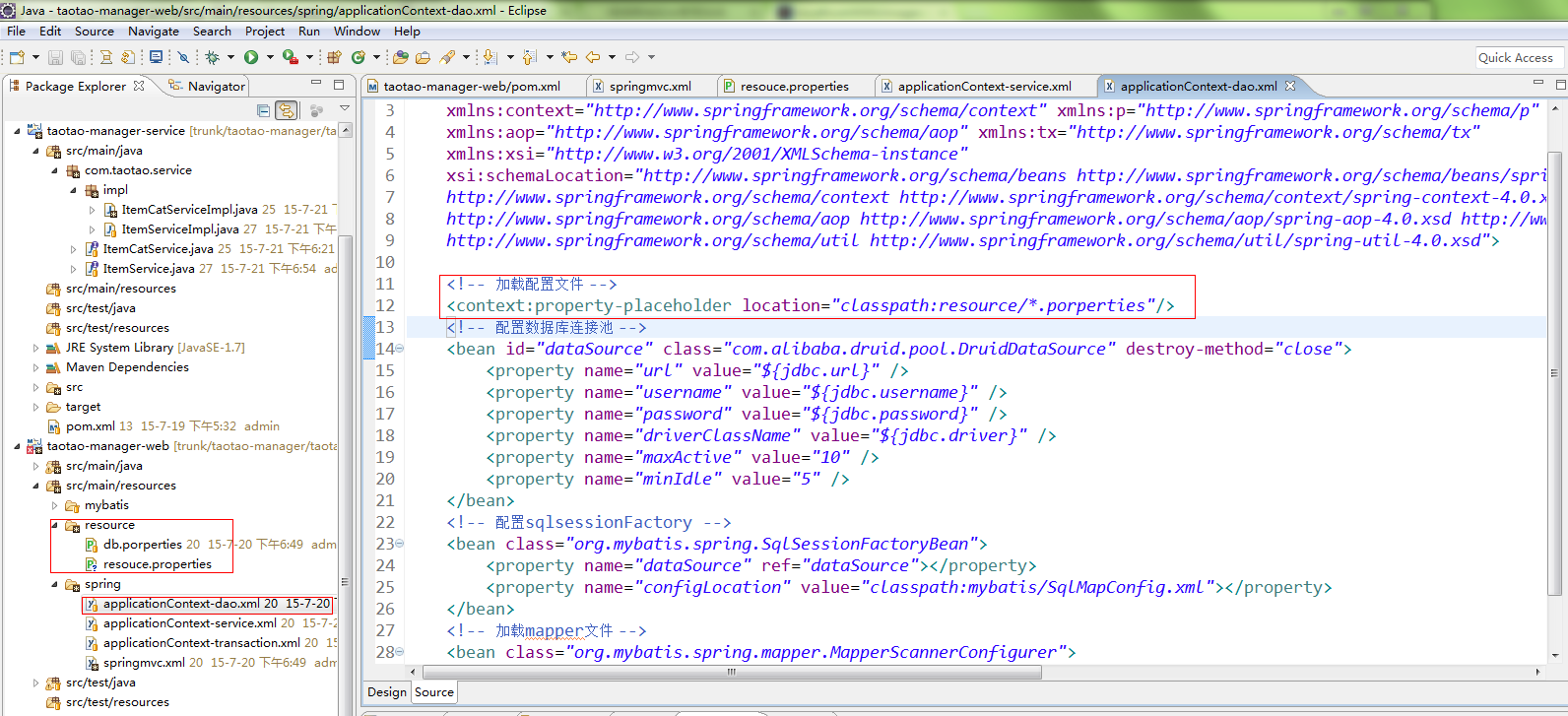
创建资源配置文件



|  |
| --- |
| FILI\_UPLOAD\_PATH=D:/temp/imagestest/webapps/images  IMAGE\_BASE\_URL=http://localhost:9000/images |

第二步：

在Spring（taotao-manage-servlet.xml）容器中加载资源文件



第二步：

在Service中获取资源配置：

|  |
| --- |
| @Value("${FILI\_UPLOAD\_PATH}")  **private** String FILI\_UPLOAD\_PATH;  @Value("${IMAGE\_BASE\_URL}")  **private** String IMAGE\_BASE\_URL; |

#### 图片名生成策略

时间+随机数：

|  |
| --- |
| /\*\*  \* 图片名生成  \*/  **public** **static** String genImageName() {  //取当前时间的长整形值包含毫秒  **long** millis = System.*currentTimeMillis*();  //long millis = System.nanoTime();  //加上三位随机数  Random random = **new** Random();  **int** end3 = random.nextInt(999);  //如果不足三位前面补0  String str = millis + String.*format*("%03d", end3);    **return** str;  } |

使用UUID：

|  |
| --- |
| UUID.randomUUID(); |

#### Service实现

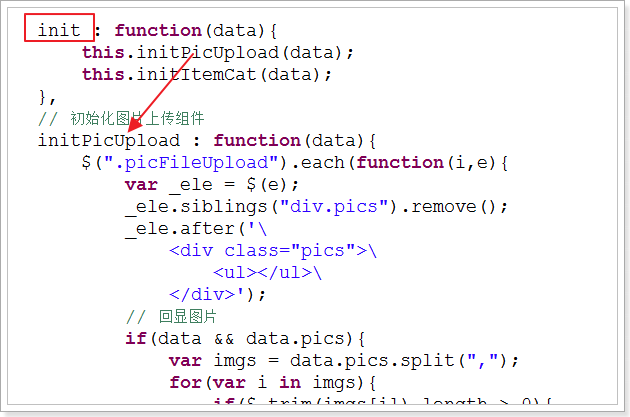
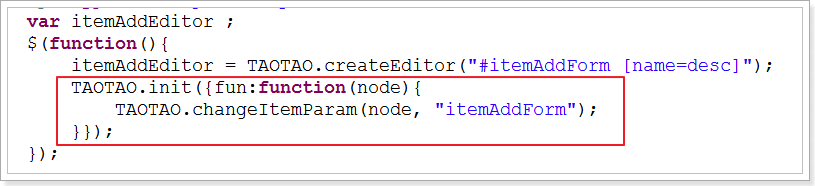
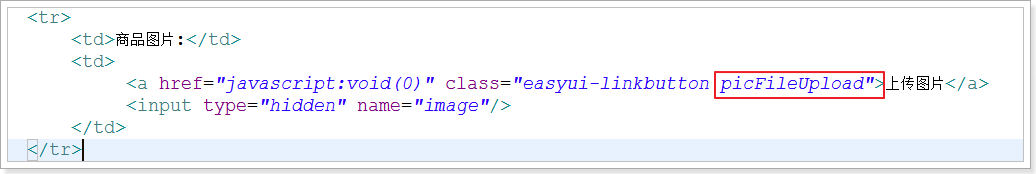
|  |
| --- |
| @Service  **public** **class** PictureServiceImpl **implements** PictureService {  @Value("${IMAGE\_BASE\_URL}")  **private** String IMAGE\_BASE\_URL;  @Value("${FILI\_UPLOAD\_PATH}")  **private** String FILI\_UPLOAD\_PATH;  @Value("${FTP\_SERVER\_IP}")  **private** String FTP\_SERVER\_IP;  @Value("${FTP\_SERVER\_PORT}")  **private** Integer FTP\_SERVER\_PORT;  @Value("${FTP\_SERVER\_USERNAME}")  **private** String FTP\_SERVER\_USERNAME;  @Value("${FTP\_SERVER\_PASSWORD}")  **private** String FTP\_SERVER\_PASSWORD;  @Override  **public** PictureResult uploadFile(MultipartFile uploadFile) **throws** Exception {  // 上传文件功能实现  String path = savePicture(uploadFile);  // 回显  PictureResult result = **new** PictureResult(0, IMAGE\_BASE\_URL + path);  **return** result;  }  **private** String savePicture(MultipartFile uploadFile) {  String result = **null**;  **try** {  // 上传文件功能实现  // 判断文件是否为空  **if** (uploadFile.isEmpty())  **return** **null**;  // 上传文件以日期为单位分开存放，可以提高图片的查询速度  String filePath = "/" + **new** SimpleDateFormat("yyyy").format(**new** Date()) + "/"  + **new** SimpleDateFormat("MM").format(**new** Date()) + "/"  + **new** SimpleDateFormat("dd").format(**new** Date());  // 取原始文件名  String originalFilename = uploadFile.getOriginalFilename();  // 新文件名  String newFileName = IDUtils.*genImageName*() + originalFilename.substring(originalFilename.lastIndexOf("."));  // 转存文件，上传到ftp服务器  FtpUtil.*uploadFile*(FTP\_SERVER\_IP, FTP\_SERVER\_PORT, FTP\_SERVER\_USERNAME, FTP\_SERVER\_PASSWORD,  FILI\_UPLOAD\_PATH, filePath, newFileName, uploadFile.getInputStream());  result = filePath + "/" + newFileName;  } **catch** (Exception e) {  e.printStackTrace();  }  **return** result;  }  } |

### Controller实现

|  |
| --- |
| @Controller  @RequestMapping("/pic")  **public** **class** PictureController {    @Autowired  **private** PictureService pictureService;  @RequestMapping("/upload")  @ResponseBody  **public** PictureResult uploda(MultipartFile uploadFile) **throws** Exception {  //调用service上传图片  PictureResult pictureResult = pictureService.uploadFile(uploadFile);  //返回上传结果  **return** pictureResult;    }  } |

### 前端JS实现图片上传

#### Js实现逻辑



KindEditor 4.x 文档

<http://kindeditor.net/doc.php>

上传图片使用kindeditor的上传组件实现。



#### 上传图片请求url：



#### 返回值

参考文档：

<http://kindeditor.net/docs/upload.html>

返回格式(JSON)

|  |
| --- |
| //成功时  {  "error" : 0,  "url" : "http://www.example.com/path/to/file.ext"  }  //失败时  {  "error" : 1,  "message" : "错误信息"  } |

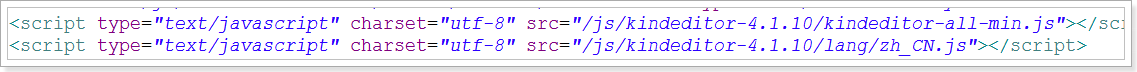
返回值数据类型：

|  |
| --- |
| **public** **class** PictureResult {  /\*\*  \* 上传图片返回值，成功：0 失败：1  \*/  **private** Integer error;  /\*\*  \* 回显图片使用的url  \*/  **private** String url;  /\*\*  \* 错误时的错误消息  \*/  } |

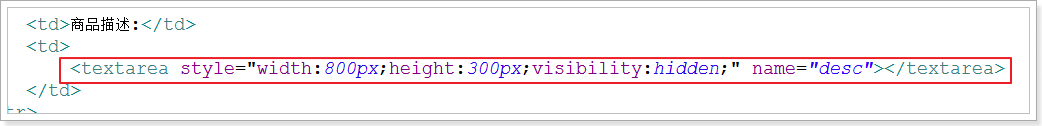
# kindeditor（富文本编辑器）的使用

## kindeditor的使用过程：

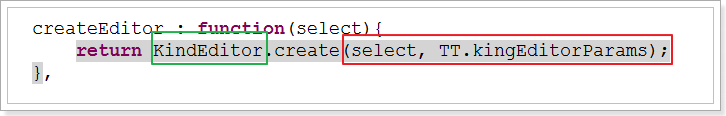
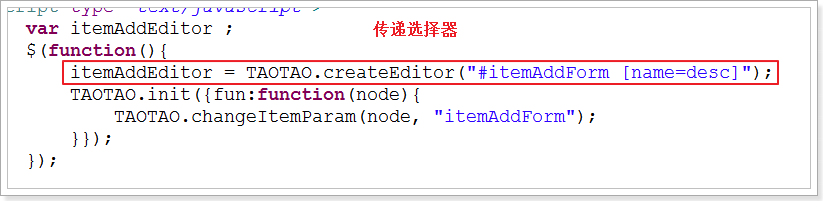
1、导入js：



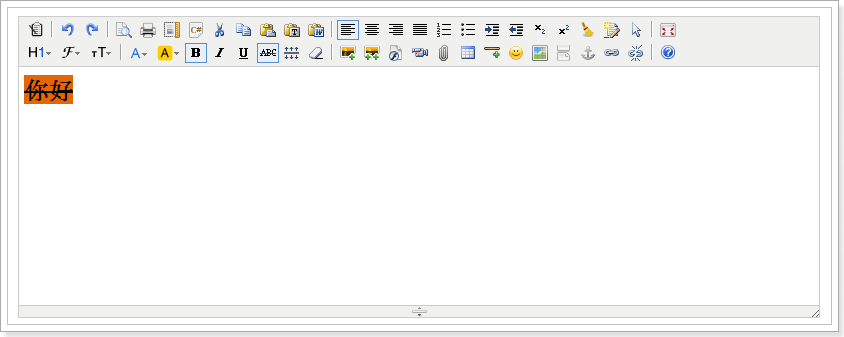
**2、定义多行文本（不可见、给定name）**



3、调用TT.createEditor



4、效果



## 取文本编辑器中的内容

将编辑器的内容设置到原来的textarea控件里。

editor**.**sync**();**

# 新增商品实现

## js编写逻辑

|  |
| --- |
| //提交表单  **function** submitForm(){  //有效性验证  **if**(!$('#itemAddForm').form('validate')){  $.messager.alert('提示','表单还未填写完成!');  **return** ;  }  //取商品价格，单位为“分”  $("#itemAddForm [name=price]").val(eval($("#itemAddForm [name=priceView]").val()) \* 100);  //同步文本框中的商品描述  itemAddEditor.sync();  //取商品的规格  /\*  var paramJson = [];  $("#itemAddForm .params li").each(function(i,e){  var trs = $(e).find("tr");  var group = trs.eq(0).text();  var ps = [];  for(var i = 1;i<trs.length;i++){  var tr = trs.eq(i);  ps.push({  "k" : $.trim(tr.find("td").eq(0).find("span").text()),  "v" : $.trim(tr.find("input").val())  });  }  paramJson.push({  "group" : group,  "params": ps  });  });  //把json对象转换成字符串  paramJson = JSON.stringify(paramJson);  $("#itemAddForm [name=itemParams]").val(paramJson);  \*/  //ajax的post方式提交表单  //$("#itemAddForm").serialize()将表单序列号为key-value形式的字符串  $.post("/item/save",$("#itemAddForm").serialize(), **function**(data){  **if**(data.status == 200){  $.messager.alert('提示','新增商品成功!');  }  });  } |

## 提交请求的数据格式

$("#itemAddForm").serialize()将表单序列号为key-value形式的字符串

以post 的形式将表单的内容提交。

**请求的url：**

/item/save

**返回的结果：**

淘淘自定义返回结果：

1. 状态码
2. 响应的消息
3. 响应的数据

|  |
| --- |
| /\*\*  \* 淘淘商城自定义响应结构  \*/  **public** **class** TaotaoResult {  // 定义jackson对象  **private** **static** **final** ObjectMapper ***MAPPER*** = **new** ObjectMapper();  // 响应业务状态  **private** Integer status;  // 响应消息  **private** String msg;  // 响应中的数据  **private** Object data;  **public** **static** TaotaoResult build(Integer status, String msg, Object data) {  **return** **new** TaotaoResult(status, msg, data);  }  **public** **static** TaotaoResult ok(Object data) {  **return** **new** TaotaoResult(data);  }  **public** **static** TaotaoResult ok() {  **return** **new** TaotaoResult(**null**);  }  **public** TaotaoResult() {  }  **public** **static** TaotaoResult build(Integer status, String msg) {  **return** **new** TaotaoResult(status, msg, **null**);  }  **public** TaotaoResult(Integer status, String msg, Object data) {  **this**.status = status;  **this**.msg = msg;  **this**.data = data;  }  **public** TaotaoResult(Object data) {  **this**.status = 200;  **this**.msg = "OK";  **this**.data = data;  }  // public Boolean isOK() {  // return this.status == 200;  // }  **public** Integer getStatus() {  **return** status;  }  **public** **void** setStatus(Integer status) {  **this**.status = status;  }  **public** String getMsg() {  **return** msg;  }  **public** **void** setMsg(String msg) {  **this**.msg = msg;  }  **public** Object getData() {  **return** data;  }  **public** **void** setData(Object data) {  **this**.data = data;  }  /\*\*  \* 将json结果集转化为TaotaoResult对象  \*  \* **@param** jsonData json数据  \* **@param** clazz TaotaoResult中的object类型  \* **@return**  \*/  **public** **static** TaotaoResult formatToPojo(String jsonData, Class<?> clazz) {  **try** {  **if** (clazz == **null**) {  **return** ***MAPPER***.readValue(jsonData, TaotaoResult.**class**);  }  JsonNode jsonNode = ***MAPPER***.readTree(jsonData);  JsonNode data = jsonNode.get("data");  Object obj = **null**;  **if** (clazz != **null**) {  **if** (data.isObject()) {  obj = ***MAPPER***.readValue(data.traverse(), clazz);  } **else** **if** (data.isTextual()) {  obj = ***MAPPER***.readValue(data.asText(), clazz);  }  }  **return** *build*(jsonNode.get("status").intValue(), jsonNode.get("msg").asText(), obj);  } **catch** (Exception e) {  **return** **null**;  }  }  /\*\*  \* 没有object对象的转化  \*  \* **@param** json  \* **@return**  \*/  **public** **static** TaotaoResult format(String json) {  **try** {  **return** ***MAPPER***.readValue(json, TaotaoResult.**class**);  } **catch** (Exception e) {  e.printStackTrace();  }  **return** **null**;  }  /\*\*  \* Object是集合转化  \*  \* **@param** jsonData json数据  \* **@param** clazz 集合中的类型  \* **@return**  \*/  **public** **static** TaotaoResult formatToList(String jsonData, Class<?> clazz) {  **try** {  JsonNode jsonNode = ***MAPPER***.readTree(jsonData);  JsonNode data = jsonNode.get("data");  Object obj = **null**;  **if** (data.isArray() && data.size() > 0) {  obj = ***MAPPER***.readValue(data.traverse(),  ***MAPPER***.getTypeFactory().constructCollectionType(List.**class**, clazz));  }  **return** *build*(jsonNode.get("status").intValue(), jsonNode.get("msg").asText(), obj);  } **catch** (Exception e) {  **return** **null**;  }  }  } |

## 获得商品id

临时主键生成策略：

|  |
| --- |
| /\*\*  \* 商品id生成  \*/  **public** **static** **long** genItemId() {  //取当前时间的长整形值包含毫秒  **long** millis = System.*currentTimeMillis*();  //long millis = System.nanoTime();  //加上两位随机数  Random random = **new** Random();  **int** end2 = random.nextInt(99);  //如果不足两位前面补0  String str = millis + String.*format*("%02d", end2);  **long** id = **new** Long(str);  **return** id;  } |

## ItemServiceImpl

调用mapper的insert方法添加商品信息

|  |
| --- |
| @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);    } |

## Controller实现

|  |
| --- |
| @RequestMapping("/save")  @ResponseBody  **public** TaotaoResult saveItem(TbItem item, String desc) **throws** Exception {  //添加商品信息  itemService.saveItem(item, desc, **null**);  **return** TaotaoResult.*ok*();  } |

# 课后作业

完成商品修改功能。

