# 仿天猫系统报告文档

### --尹捷 2018104161

# 1背景分析和介绍

## 1.1目的和背景

* 天猫系统一个大型网上购物平台
* 在天猫上，用户可以选择和查找自己想要的商品，加入购物车或者直接购买，带给人们一种全新的购物方式

## 1.2项目介绍

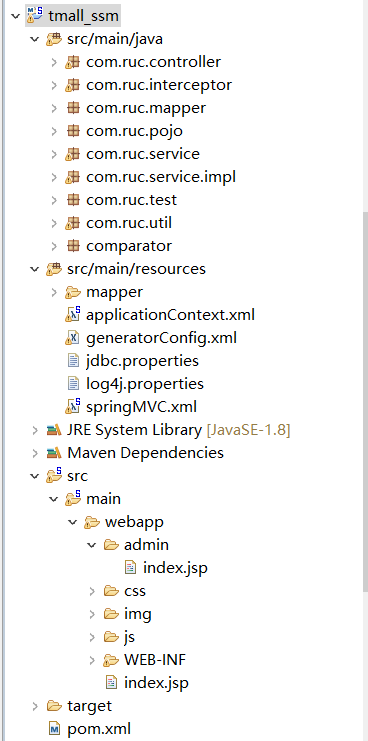
本项目是在原有的天猫系统的基础上进行的模仿和重做，主要工作内容有:

* 前端界面模仿
* 后端架构选型
* 实现大致功能

项目用到的知识如下：

**Java:**  
[Java基础](https://how2j.cn/stage/12.html) 和 [Java中级](https://how2j.cn/stage/25.html) 的大部分内容  
**前端**：  
[HTML](https://how2j.cn/k/html/html-tutorial/175.html), [CSS](https://how2j.cn/k/css2/css2-tutorial/238.html), [JAVASCRIPT](https://how2j.cn/k/javascript/javascript-javascript-tutorial/519.html), [JQUERY](https://how2j.cn/k/jquery/jquery-tutorial/467.html),[AJAX](https://how2j.cn/k/ajax/ajax-tutorial/465.html), [Bootstrap](https://how2j.cn/k/boostrap/boostrap-tutorial/538.html)  
**J2EE**：  
[TOMCAT](https://how2j.cn/k/tomcat/tomcat-tutorial/541.html), [SERVLET](https://how2j.cn/k/servlet/servlet-eclipse/558.html), [JSP](https://how2j.cn/k/jsp/jsp-tutorials/530.html), [Filter](https://how2j.cn/k/filter/filter-tutorial/588.html)  
**框架**：  
[Spring](https://how2j.cn/k/spring/spring-ioc-di/87.html)，[Spring MVC](https://how2j.cn/k/springmvc/springmvc-springmvc/615.html)，[Mybatis](https://how2j.cn/k/mybatis/mybatis-tutorial/1087.html)，[Spring+Mybatis整合](https://how2j.cn/k/spring-mybatis/spring-mybatis-tutorial/1134.html)，[SSM整合](https://how2j.cn/k/ssm/ssm-tutorial/1137.html)  
**数据库**：  
[MySQL](https://how2j.cn/k/mysql/mysql-install/377.html)  
**开发工具**:  
Eclipse,[Maven](https://how2j.cn/k/maven/maven-introduction/1328.html)

项目名称: tmall\_ssm  
java源代码包结构:  
pojo 实体类  
mapper Mapper类  
interceptor 拦截器  
controller 控制层  
service Service层  
test 测试类  
util 工具类  
comparator 比较类  
web目录:  
css css文件  
img 图片资源  
js js文件  
admin 后台管理用到的jsp文件  
fore 前台展示用到的jsp文件  
include 被包含的jsp文件



经过这个项目，完成了如下的一系列典型场景功能  
1. 购物车  
[立即购买](http://how2j.cn/k/tmall_ssm/tmall_ssm-1534/1534.html)，[加入购物车](http://how2j.cn/k/tmall_ssm/tmall_ssm-1535/1535.html)，[查看购物车页面](http://how2j.cn/k/tmall_ssm/tmall_ssm-1541/1541.html)，[购物车页面操作](http://how2j.cn/k/tmall_ssm/tmall_ssm-1536/1536.html)  
  
2. 订单状态流转  
[生成订单](http://how2j.cn/k/tmall_ssm/tmall_ssm-1537/1537.html)， [确认支付](http://how2j.cn/k/tmall_ssm/tmall_ssm-1537/1537.html#step6377)， [后台发货](http://how2j.cn/k/tmall_ssm/tmall_ssm-1485/1485.html#step6198)， [确认收货](http://how2j.cn/k/tmall_ssm/tmall_ssm-1542/1542.html#step6402)， [评价](http://how2j.cn/k/tmall_ssm/tmall_ssm-1543/1543.html)  
  
3. CRUD   
[后台各种功能](http://how2j.cn/k/tmall_ssm/tmall_ssm-1513/1513.html)  
  
4. 分页  
[后台各种功能](http://how2j.cn/k/tmall_ssm/tmall_ssm-1513/1513.html)  
  
5. 一类产品多属性配置  
[属性管理](http://how2j.cn/k/tmall_ssm/tmall_ssm-1480/1480.html)  
  
6. 一款产品多图片维护  
[产品图片管理](http://how2j.cn/k/tmall_ssm/tmall_ssm-1482/1482.html)  
  
7. 产品展示  
[前台首页](http://how2j.cn/k/tmall_ssm/tmall_ssm-1486/1486.html) ，[前台产品页](http://how2j.cn/k/tmall_ssm/tmall_ssm-1528/1528.html)  
  
8. 搜索查询  
[搜索](http://how2j.cn/k/tmall_ssm/tmall_ssm-1530/1530.html)  
  
9. 登录、注册  
[注册](http://how2j.cn/k/tmall_ssm/tmall_ssm-1527/1527.html) [登录](http://how2j.cn/k/tmall_ssm/tmall_ssm-1526/1526.html) [退出](http://how2j.cn/k/tmall_ssm/tmall_ssm-1531/1531.html)  
  
10. 登录验证  
[登录状态拦截器](http://how2j.cn/k/tmall_ssm/tmall_ssm-1539/1539.html)  
  
11. 事务管理  
[ForeController.对createOrder进行事务管理](http://how2j.cn/k/tmall_ssm/tmall_ssm-1537/1537.html#step6592)  
等等 …

## 3.1 源代码分析列表

### 3.1.1表结构及数据库实现

|  |  |  |
| --- | --- | --- |
| 表名 | 中文含义 | 介绍 |
| Category | 分类表 | 存放分类信息，如女装，平板电视，沙发等 |
| Property | 属性表 | 存放属性信息，如颜色，重量，品牌，厂商，型号等 |
| Product | 产品表 | 存放产品信息，如LED40EC平板电视机，海尔EC6005热水器 |
| PropertyValue | 属性值表 | 存放属性值信息，如重量是900g,颜色是粉红色 |
| ProductImage | 产品图片表 | 存放产品图片信息，如产品页显示的5个图片 |
| Review | 评论表 | 存放评论信息，如买回来的蜡烛很好用，么么哒 |
| User | 用户表 | 存放用户信息，如斩手狗，千手小粉红 |
| Order | 订单表 | 存放订单信息，包括邮寄地址，电话号码等信息 |
| OrderItem | 订单项表 | 存放订单项信息，包括购买产品种类，数量等 |

本项目采用的jdbc连接数据库，数据库名是tmall\_ssm，用户名是root，密码也是root，与数据库的连接定义在jdbc.properties里面：

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/tmall\_ssm?useUnicode=true&characterEncoding=utf8

jdbc.username=root

jdbc.password=root

### 3.1.2主要代码分析

### 分页

新增Page这个类专门为分页提供必要信息  
属性：  
int start; 开始位置  
int count; 每页显示的数量  
int total; 总共有多少条数据  
String param; 参数  
方法：  
getTotalPage 根据 每页显示的数量count以及总共有多少条数据total，计算出总共有多少页  
getLast 计算出最后一页的数值是多少  
isHasPreviouse 判断是否有前一页  
isHasNext 判断是否有后一页

#### 主要代码：

public int getTotalPage(){

        int totalPage;

        // 假设总数是50，是能够被5整除的，那么就有10页

        if (0 == total % count)

            totalPage = total /count;

            // 假设总数是51，不能够被5整除的，那么就有11页

        else

            totalPage = total / count + 1;

        if(0==totalPage)

            totalPage = 1;

        return totalPage;

    }

    public int getLast(){

        int last;

        // 假设总数是50，是能够被5整除的，那么最后一页的开始就是45

        if (0 == total % count)

            last = total - count;

            // 假设总数是51，不能够被5整除的，那么最后一页的开始就是50

        else

            last = total - total % count;

        last = last<0?0:last;

        return last;

    }

@Controller

@RequestMapping("")

public class CategoryController {

    @Autowired

    CategoryService categoryService;

    @RequestMapping("admin\_category\_list")

    public String list(Model model,Page page){

        List<Category> cs= categoryService.list(page);

        int total = categoryService.total();

        page.setTotal(total);

        model.addAttribute("cs", cs);

        model.addAttribute("page", page);

        return "admin/listCategory";

    }

}

### 注册与登录

#### 注册

注册页面的主体功能，用于提交账号密码。 在提交之前会进行为空验证，以及密码是否一致验证。  
  
这段代码用于当账号提交到服务端，服务端判断当前账号已经存在的情况下，显示返回的错误提示**"用户名已经被使用,不能使用"**

<c:if test="${!empty msg}">

$("span.errorMessage").html("${msg}");

$("div.registerErrorMessageDiv").css("visibility","visible");

</c:if>

|  |  |
| --- | --- |
|  | <%@ page language="java" contentType="text/html; charset=UTF-8"      pageEncoding="UTF-8" isELIgnored="false"%>    <script>  $(function(){        <c:if test="${!empty msg}">      $("span.errorMessage").html("${msg}");      $("div.registerErrorMessageDiv").css("visibility","visible");      </c:if>        $(".registerForm").submit(function(){          if(0==$("#name").val().length){              $("span.errorMessage").html("请输入用户名");              $("div.registerErrorMessageDiv").css("visibility","visible");              return false;          }          if(0==$("#password").val().length){              $("span.errorMessage").html("请输入密码");              $("div.registerErrorMessageDiv").css("visibility","visible");              return false;          }          if(0==$("#repeatpassword").val().length){              $("span.errorMessage").html("请输入重复密码");              $("div.registerErrorMessageDiv").css("visibility","visible");              return false;          }          if($("#password").val() !=$("#repeatpassword").val()){              $("span.errorMessage").html("重复密码不一致");              $("div.registerErrorMessageDiv").css("visibility","visible");              return false;          }            return true;      });  })  </script> |

[registerPage.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1527/1527.html#step6320) 的form提交数据到路径 foreregister,导致ForeController.register()方法被调用  
  
1. 通过参数User获取浏览器提交的账号密码  
2. 通过HtmlUtils.htmlEscape(name);把账号里的特殊符号进行转义  
3. 判断用户名是否存在  
3.1 如果已经存在，就服务端跳转到reigster.jsp，并且带上错误提示信息  
3.2 如果不存在，则加入到数据库中，并服务端跳转到registerSuccess.jsp页面  
  
**注**为什么要用 HtmlUtils.htmlEscape？ 因为有些人在恶意注册的时候，会使用诸如 <script>alert('papapa')</script> 这样的名称，会导致网页打开就弹出一个对话框。 那么在转义之后，就没有这个问题了。  
**注** model.addAttribute("user", null); 这句话的用处是当用户存在，服务端跳转到register.jsp的时候不带上参数user, 否则当注册失败的时候，会在原本是“请登录”的超链位置显示刚才注册的名称。

@RequestMapping("foreregister")

 public String register(Model model,User user) {

     String name =  user.getName();

     name = HtmlUtils.htmlEscape(name);

     user.setName(name);

     boolean exist = userService.isExist(name);

     if(exist){

         String m ="用户名已经被使用,不能使用";

         model.addAttribute("msg", m);

         model.addAttribute("user", null);

         return "fore/register";

     }

     userService.add(user);

     return "redirect:registerSuccessPage";

 }

#### 登录

登陆业务页面，用于向服务器提交账号和密码  
  
1. 与[registerPage.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1527/1527.html#step6320)类似的，用于显示登录密码错误

<c:if test="${!empty msg}">

$("span.errorMessage").html("${msg}");

$("div.loginErrorMessageDiv").show();

</c:if>

2. 其他js函数是用于为空验证

[loginPage.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1526/1526.html#step6314)的form提交数据到路径 forelogin,导致ForeController.login()方法被调用  
  
1. 获取账号密码  
2. 把账号通过HtmlUtils.htmlEscape进行转义  
3. 根据账号和密码获取User对象  
3.1 如果对象为空，则服务端跳转回login.jsp，也带上错误信息，并且使用 [loginPage.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1526/1526.html#step6314) 中的办法显示错误信息  
3.2 如果对象存在，则把对象保存在session中，并客户端跳转到首页"forehome"

@RequestMapping("forelogin")

    public String login(@RequestParam("name") String name, @RequestParam("password") String password, Model model, HttpSession session) {

        name = HtmlUtils.htmlEscape(name);

        User user = userService.get(name,password);

        if(null==user){

            model.addAttribute("msg", "账号密码错误");

            return "fore/login";

        }

        session.setAttribute("user", user);

        return "redirect:forehome";

    }

}

### 3.1.3 模态登录

立即购买和加入购物车这两个按钮的监听是放在[imgAndInfo.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1528/1528.html#step6329)页面中  
  
这两个按钮都会通过[JQuery的.get方法](http://how2j.cn/k/jquery/jquery-ajax/474.html)，用异步ajax的方式访问forecheckLogin，获取当前是否登录状态  
如果返回的不是"success" 即表明是未登录状态，那么就会打开登录的模态窗口

$("#loginModal").modal('show');

@RequestMapping("forecheckLogin")

    @ResponseBody

    public String checkLogin( HttpSession session) {

        User user =(User)  session.getAttribute("user");

        if(null!=user)

            return "success";

        return "fail";

    }

}

在上一步[modal.jsp](http://how2j.cn/k/tmall_ssm/tmall_ssm-1532/1532.html#step6355)中，点击了登录按钮之后，访问路径/foreloginAjax,导致ForeController.loginAjax()方法被调用  
1. 获取账号密码  
2. 通过账号密码获取User对象  
2.1 如果User对象为空，那么就返回"fail"字符串。  
2.2 如果User对象不为空，那么就把User对象放在session中，并返回"success" 字符串

@RequestMapping("foreloginAjax")

    @ResponseBody

    public String loginAjax(@RequestParam("name") String name, @RequestParam("password") String password,HttpSession session) {

        name = HtmlUtils.htmlEscape(name);

        User user = userService.get(name,password);

        if(null==user){

            return "fail";

        }

        session.setAttribute("user", user);

        return "success";

    }

}

### 3.1.4 商品分类排序

分类这个页面有排序功能，所以在讲解ForeController之前，先准备5个Comparator比较器  
  
1. ProductAllComparator 综合比较器  
把 销量x评价 高的放前面  
2. ProductReviewComparator 人气比较器  
把 评价数量多的放前面  
3. ProductDateComparator 新品比较器  
把 创建日期晚的放前面  
4. ProductSaleCountComparator 销量比较器  
把 销量高的放前面  
5. ProductPriceComparator 价格比较器  
把 价格低的放前面

1. 获取参数cid  
   2. 根据cid获取分类Category对象 c  
   3. 为c填充产品  
   4. 为产品填充销量和评价数据  
   5. 获取参数sort  
   5.1 如果sort==null，即不排序  
   5.2 如果sort!=null，则根据sort的值，从[5个Comparator比较器](http://how2j.cn/k/tmall_ssm/tmall_ssm-1529/1529.html#step6334)中选择一个对应的排序器进行排序  
   6. 把c放在model中  
   7. 服务端跳转到 category.jsp

@RequestMapping("forecategory")

1. public String category(int cid,String sort, Model model) {
2. Category c = categoryService.get(cid);
3. productService.fill(c);
4. productService.setSaleAndReviewNumber(c.getProducts());
6. if(null!=sort){
7. switch(sort){
8. case "review":
9. Collections.sort(c.getProducts(),new ProductReviewComparator());
10. break;
11. case "date" :
12. Collections.sort(c.getProducts(),new ProductDateComparator());
13. break;
15. case "saleCount" :
16. Collections.sort(c.getProducts(),new ProductSaleCountComparator());
17. break;
19. case "price":
20. Collections.sort(c.getProducts(),new ProductPriceComparator());
21. break;
23. case "all":
24. Collections.sort(c.getProducts(),new ProductAllComparator());
25. break;
26. }
27. }
29. model.addAttribute("c", c);
30. return "fore/category";
31. }
32. }

### 3.1.5 购物流程

1. 登录  
2. 访问产品页  
3. 立即购买  
4. 进入结算页面  
5. 加入购物车  
6. 查看购物车  
7. 选中购物车中的商品  
8. 又到了第4步的结算页面  
9. 在结算页面生成订单  
10. 付款  
11. 确认收货  
12. 评价

围绕购物流程最重要的两个表是OrderItem 和 Order表  
  
关于OrderItem的业务行为  
1. 立即购买 —— 新增 OrderItem  
2. 加入购物车 —— 新增 OrderItem  
3. 查看购物车 —— 显示未和Order关联的OrderItem  
4. 选中购物车中的商品 —— 选中OrderItem  
5. 结算页面 —— 显示选中的OrderItem  
  
6. 生成订单 —— 新增Order  
7 .付款 —— 修改Order状态  
8. 我的订单 —— 显示Order  
9. 确认收货 —— 修改Order状态

### 3.1.6 登录状态拦截器

在查看购物车之前，应该进行登录操作，但是又不能确保用户一定会记得登录，那么怎么办呢？   
准备一个过滤器，当访问那些需要登录才能做的页面的时候，进行是否登录的判断，如果不通过，那么就跳转到login.jsp页面去，提示用户登录。  
  
哪些页面需要登录？哪些页面不需要呢？  
a. 不需要登录也可以访问的  
如：注册，登录，产品，首页，分类，查询等等  
b. 需要登录才能够访问的  
如：购买行为，加入购物车行为，查看购物车，查看我的订单等等

新建一个过滤器LoginInterceptor   
哪些页面需要登录？哪些页面不需要呢？  
a. 不需要登录也可以访问的  
如：注册，登录，产品，首页，分类，查询等等  
b. 需要登录才能够访问的  
如：购买行为，加入购物车行为，查看购物车，查看我的订单等等  
不需要登录也可以访问的已经确定了，但是需要登录才能够访问，截止目前为止还不能确定，所以这个过滤器就判断如果不是注册，登录，产品这些，就进行登录校验  
1. 准备字符串数组 noNeedAuthPage，存放哪些不需要登录也能访问的路径  
2. 获取uri  
3. 去掉前缀/tmall\_ssm  
4. 如果访问的地址是/fore开头  
4.1 取出fore后面的字符串，比如是forecart,那么就取出cart  
4.2 判断cart是否是在noNeedAuthPage   
4.2 如果不在，那么就需要进行是否登录验证  
4.3 从session中取出"user"对象  
4.4 如果对象不存在，就客户端跳转到login.jsp  
4.5 否则就正常执行

 /\*\*

     \* 在业务处理器处理请求之前被调用

     \* 如果返回false

     \*     从当前的拦截器往回执行所有拦截器的afterCompletion(),再退出拦截器链

     \* 如果返回true

     \*    执行下一个拦截器,直到所有的拦截器都执行完毕

     \*    再执行被拦截的Controller

     \*    然后进入拦截器链,

     \*    从最后一个拦截器往回执行所有的postHandle()

     \*    接着再从最后一个拦截器往回执行所有的afterCompletion()

     \*/

    public boolean preHandle(HttpServletRequest request,

                             HttpServletResponse response, Object handler) throws Exception {

        HttpSession session = request.getSession();

        String contextPath=session.getServletContext().getContextPath();

        String[] noNeedAuthPage = new String[]{

                "home",

                "checkLogin",

                "register",

                "loginAjax",

                "login",

                "product",

                "category",

                "search"};

        String uri = request.getRequestURI();

        uri = StringUtils.remove(uri, contextPath);

//        System.out.println(uri);

        if(uri.startsWith("/fore")){

            String method = StringUtils.substringAfterLast(uri,"/fore" );

            if(!Arrays.asList(noNeedAuthPage).contains(method)){

                User user =(User) session.getAttribute("user");

                if(null==user){

                    response.sendRedirect("loginPage");

                    return false;

                }

            }

        }

        return true;

    }

    /\*\*

     \* 在业务处理器处理请求执行完成后,生成视图之前执行的动作

     \* 可在modelAndView中加入数据，比如当前时间

     \*/

    public void postHandle(HttpServletRequest request,

            HttpServletResponse response, Object handler,

            ModelAndView modelAndView) throws Exception {

    }

    /\*\*

     \* 在DispatcherServlet完全处理完请求后被调用,可用于清理资源等

     \*

     \* 当有拦截器抛出异常时,会从当前拦截器往回执行所有的拦截器的afterCompletion()

     \*/

    public void afterCompletion(HttpServletRequest request,

            HttpServletResponse response, Object handler, Exception ex)

    throws Exception {

    }

}

3.3 应用结果





