通用电子商务平台项目

GWAP

详细设计说明书

Version 3.1

General WEB application platform

(GWAP 3.1)

Design elucidation in detail

文档编号：GWAP-03-02

NO.[GWAP-03-02]

达内 IT 培训集团

版本	标题	内容	作者	时间
1.0.0	创建	创建	马东锋	08/18/2009
1.0.1	修订	1、 详细设计文档中 4.3,技术体系“数据库”变更为“MySQL”。 2、 增加异常实现逻辑 6.3.2（向管理员发 email）。 3、 6.3.1 UML 图 Factory 成员隐去。	马东锋	08/24/2009

目　录

1．编写目的

明确业务背景、业务范围、基本业务逻辑和业务框架，期望读者包括：项目发起人、最终用户、项目投资方、项目管理团队、项目执行团队，以及其他项目干系人。

2．参考文档

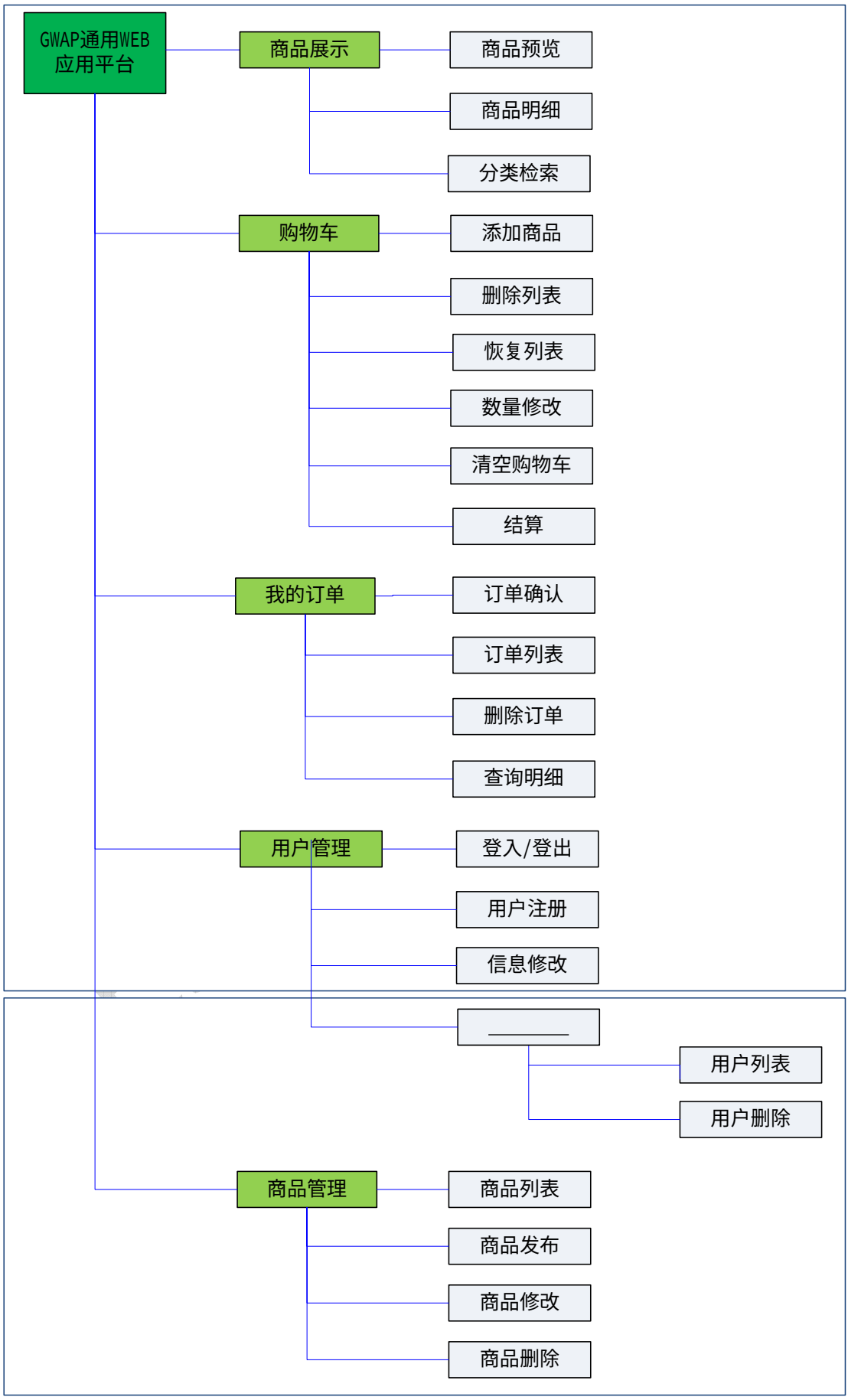
- “GWAP1.x 需求分析说明书.doc”：需求分析说明书模板。
- “GWAP 需求变更意见.doc”：需求变更说明。
- “TTS 会议纪要(2009-07-31).doc”：项目启动会议纪要。
- “系统需求分析（Ver 1.0）.doc”：前一版本的需求分析。
- “GWAP3.1 需求分析说明书.doc”：系统需求分析说明书。

3．名称解释

- GWAP：通用 WEB 应用平台（General WEB application platform)。
- B/S：Browser/Server（浏览器/服务器)。
- C/S：Client/Server（客户端/服务器)。
- B2B：Business to Business 商家对商家，电子商务的一种业务形式。
- B2C：Business to Consumer 商家对顾客，电子商务的一种业务形式。
- C2C：Consumer to Consumer 顾客对顾客，电子商务的一种业务形式。
- CSS：是 Cascading Style Sheets(层叠样式表)的简称，CSS 语言是一种标记语言,它不需要编译可以直接由浏览器执行。
- HTML：Hypertext Markup Language 超文字标记语言。
- JAVASCRIPT：一种由 Sun Microsystems 所开发的程序脚本语言(它是一种严密的物件导向的语言,适合在网际网络上发展主从架构的应用程序)，常常简称 JS。

4．系统架构

4.1 功能结构


GWAP通用WEB应用平台
商品展示
商品预览
商品明细
分类检索
购物车
添加商品
删除列表
恢复列表
数量修改
清空购物车
结算
我的订单
订单确认
订单列表
删除订单
查询明细
用户管理
登入/登出
用户注册
信息修改
用户列表
用户删除
商品管理
商品列表
商品发布
商品修改
商品删除

通用 WEB 应用系统，包括交易系统和管理系统两个独立的子系统，主要包括：商品展示、购物车管理、我的订单、用户管理、商品管理五个功能模块。

- **交易系统功能描述**
  - ✓ 商品展示：实现商品预览、商品明细、分类检索功能。
  - ✓ 购物车：实现添加商品、删除列表、恢复列表、数量修改、清空购物车、结算功能。
  - ✓ 我的订单：实现订单确认、订单列表、删除订单、查询明细功能。
  - ✓ 用户管理：实现登入/登出、用户注册、信息修改功能。
- **管理系统管理描述**
  - ✓ 用户管理：对于系统管理员提供用户删除、用户查询功能。
  - ✓ 商品管理：实现商品列表、商品发布、商品删除、商品修改功能。

4.2 系统架构

交易系统和**管理系统**在系统架构方面完全一致，采用 4 层结构，主要包括：表现层、控制层、业务层、持久层。

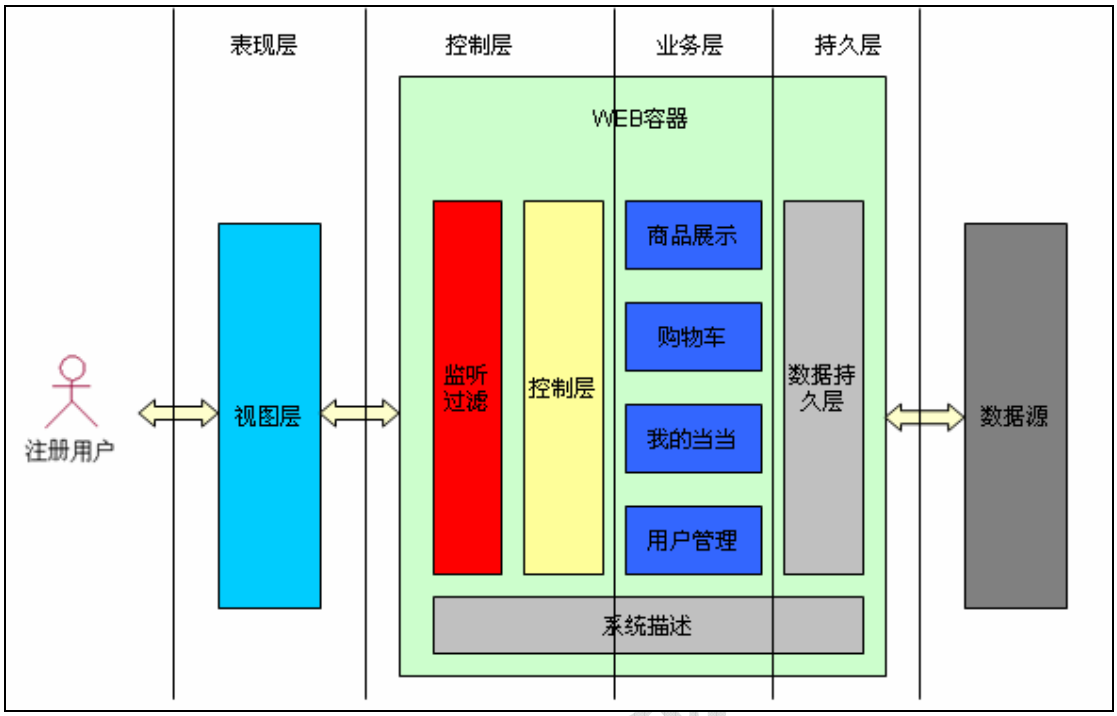
表现层：主要负责用户交互和结果显示，交易系统和管理系统有不同的 URL 入口界面，详细设计参考“ GWAP DEMO ”。

控制层：主要负责系统的访问控制、数据加载和注销，是系统的核心控制单元。控制层通过系统描述来组织工作。

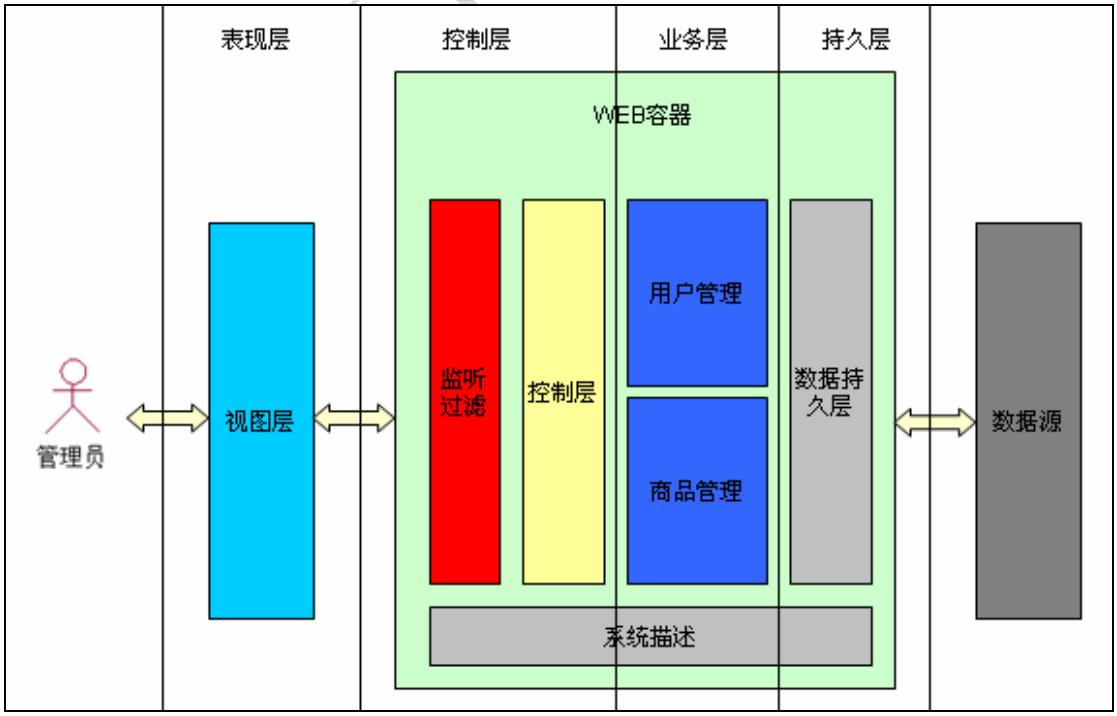
业务层：实现了交易系统和管理系统的主要业务逻辑，是系统主要的运算单元。

持久层：实现了静态数据和数据库数据的持久化管理，为业务层提供数据服务，是系统进行数据操作的统一界面。

4.2.1 交易系统架构

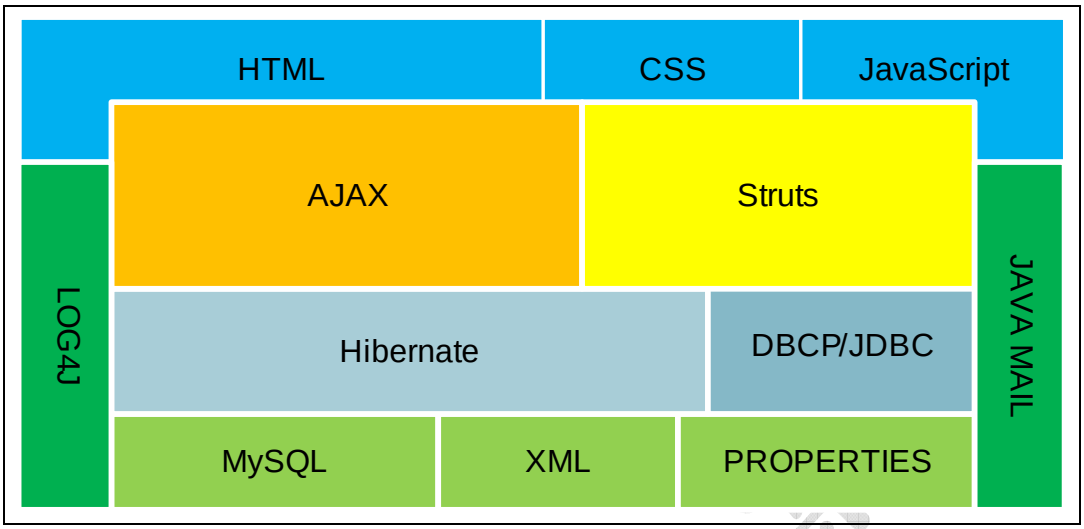


4.2.2 管理系统架构



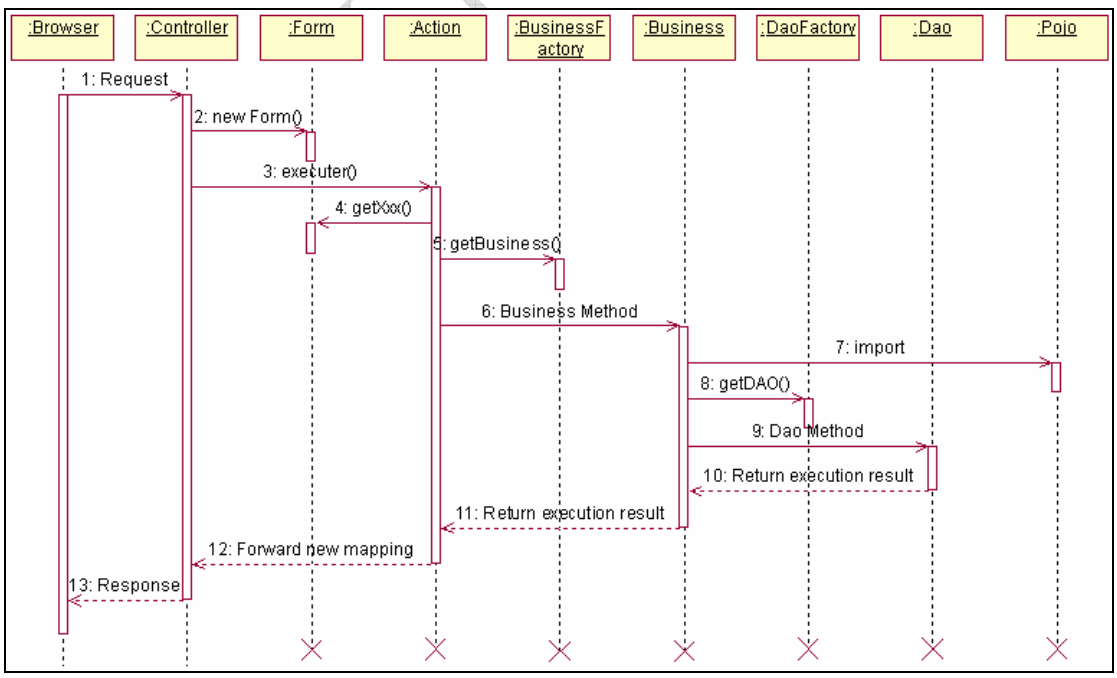
4.3 技术体系

下图描述了本系统将要使用到的具体的技术和工具。



5．访问控制设计

5.1 访问控制流程图



5.2 访问控制流程说明

结合上图，针对每一个消息请求，按照消息请求/回复顺序，作如下说明。

编号	消息名称	消息说明
1	HttpServletRequest	浏览器发来的 HttpServletRequest 请求
2	ActionForm	通过 ActionForm 获取 Struts 加载的页面表单数据
3	Exectule	根据 struts-config.xml 配置调用指定的 Action 指定的方法处理请求
4	ActionForm.getXxx()	通过 ActionForm 获取页面表单数据
5	BusinessFactory .getBusiness	获得 Business 对象
6	Business Method	调用业务逻辑
7	Import	使用 Pojo 进行数据操作
8	DAOFactory.getDAO	获得 Dao 对象
9	Dao Method	调用 Dao 方法，执行数据库操作
10	Return execution result	返回 Dao 执行结果，给 Business
11	Return execution result	返回 Business 执行结果，给 Action
12	ActionForward.findForward	Action 返回执行结果给新的画面(Struts-Config.xml 配置)
13	HttpServletResponse	MappingDispatchAction 返回处理结果给浏览器

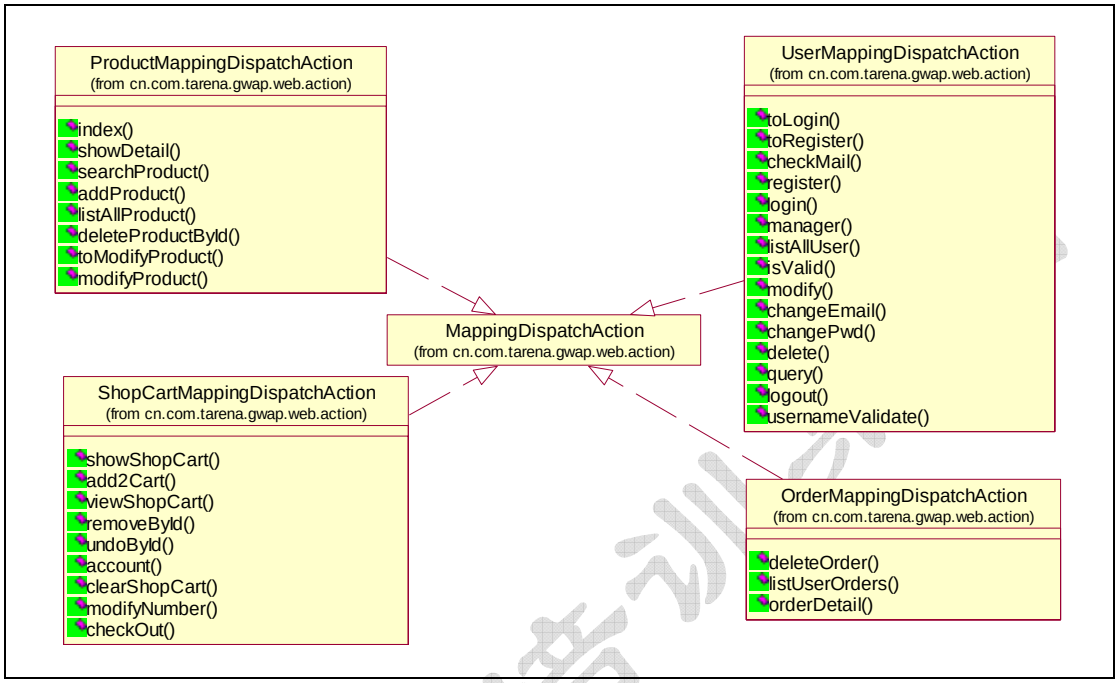
6. 域模型设计

6.1 视图层设计

- 视图层设计参考“ GWAP3.1 需求分析说明书.doc ”
- 实现效果参考“ GWAP(VER1.0)-20090811.zip ”

6.2 控制层设计

6.2.1 类图



6.2.2 类描述

- **OrderMappingDispatchAction 订单管理核心控制类**

方法名	功能描述	参数和返回值	异常定义

deleteOrder	删除用户订单.	**参数**
		`ActionMapping,`
		`ActionForm,`
		`HttpServletRequest,`
		`HttpServletResponse`
		返回值
		ActionForward
			Exception

listUserOrders	显示用户订单列表	同上	Exception
orderDetail	显示订单明细	同上	Exception

- **ProductMappingDispatchAction 商品管理核心控制类**

方法名	功能描述	参数和返回值	异常定义
addProduct	添加发布商品（管理员）	**参数** ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse **返回值** ActionForward	Exception
index	显示我的 GWAP 首页面	同上	Exception
showDetail	显示商品明细	同上	Exception
searchProduct	按照类别检索商品	同上	Exception
listAllProduct	得到所有的商品	同上	Exception
deleteProductById	删除和下架商品	同上	Exception
toModifyProduct	打起修改商品信息页面	同上	Exception
modifyProduct	修改商品信息	同上	Exception

- **ShopCartMappingDispatchAction 购物车管理核心控制类**

方法名	功能描述	参数和返回值	异常定义
account	统计商品，进入订单确认页面	**参数** ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse **返回值** ActionForward	Exception

showShopCart	打起购物车	同上	Exception
add2Cart	添加商品到购物车	同上	Exception
viewShopCart	暂时没有使用	同上	Exception
removeById	根据商品 ID，从购物车中移出商品	同上	Exception
undoById	恢复移出商品到购物车	同上	Exception
clearShopCart	清空购物车	同上	Exception
modifyNumber	修改商品数量	同上	Exception
checkOut	订单检查和入库	同上	Exception

- **UserMappingDispatchAction 用户管理核心控制类**

方法名	功能描述	参数和返回值	异常定义

changeEmail	修改 Mail 信息.	**参数**
		`ActionMapping,`
		`ActionForm,`
		`HttpServletRequest,`
		`HttpServletResponse`
		返回值
		ActionForward
			Exception

toLogin	调转到登陆画面	同上	Exception
toRegister	打起用户注册页面	同上	Exception
checkMail	邮箱验证	同上	Exception
register	用户注册	同上	Exception
login	用户登陆	同上	Exception
manager	管理员用户登陆	同上	Exception

listAllUser	查找所有用户得到用户列表	同上	Exception
isValid	用户信息有效性验证	同上	Exception
modify	编辑个人信息	同上	Exception
changePwd	修改个人密码	同上	Exception
delete	删除用户	同上	Exception
query	查询用户.	同上	Exception
logout	用户退出	同上	Exception
usernameValidate	AJAX 检查用户名是否存在（用户注册时）	**参数** ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse **返回值** Void	Exception

6.2.3 控制文件描述

- **web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <!-- 添加监听器 -->
  <listener>
    <listener-class>
        <!-- 启动服务器时加载商品信息到内存 -->
        cn.com.tarena.gwap.web.listener.ProductContextListener
    </listener-class>
```

```
</listener>
<listener>
  <listener-class>
      <!-- 启动服务器时加载购物车信息到内存 -->
      cn.com.tarena.gwap.web.listener.CartSessionListener
  </listener-class>
</listener>

<!-- 添加过滤器 -->
<filter>
  <!-- 处理字符编码的过滤器 -->
  <filter-name>encodingFilter</filter-name>
  <filter-class>
      cn.com.tarena.gwap.web.filter.EncodingFilter
  </filter-class>
  <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>*.do</url-pattern>
</filter-mapping>
  <filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>

<filter>
  <!-- 判断用户是否登陆的过滤器 -->
  <filter-name>loginFilter</filter-name>
  <filter-class>
      cn.com.tarena.gwap.web.filter.LoginFilter
  </filter-class>
</filter>
<!--商品管理-->
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/product/list.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/product/productAdd.do</url-pattern>
```

```
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/product/productModify.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/product/productModified.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/product/deleteProduct.do</url-pattern>
</filter-mapping>
<!--用户管理-->
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/modify.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/changeemail.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/changepwd.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/query.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/delete.do</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/listAllUser.do</url-pattern>
</filter-mapping>
<!--购物车管理-->
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/user/add2cart.do</url-pattern>
</filter-mapping>
<filter-mapping>
```

```
    <filter-name>loginFilter</filter-name>
    <url-pattern>/shopcart/showShopCart.do</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/shopcart/removeProduct.do</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/shopcart/updateProductNum.do</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/shopcart/removeAllProduct.do</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/user/saveOrder.do</url-pattern>
  </filter-mapping>
  <!--订单管理-->
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/order/list.do</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>loginFilter</filter-name>
    <url-pattern>/order/delete.do</url-pattern>
  </filter-mapping>

  <!-- 添加控制器 -->
  <servlet>
    <servlet-name>action</servlet-name>

<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/classes/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>3</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
```

```
      <param-value>3</param-value>
    </init-param>
    <load-on-startup>0</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
```

- **struts-config.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="registerActionForm"
type="cn.com.tarena.gwap.web.form.RegisterActionForm"/>
        <form-bean name="productActionForm"
type="cn.com.tarena.gwap.web.form.ProductForm"/>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="fail" path="/error.jsp"/>
        <forward name="login" path="/user/signin.jsp"/>
    </global-forwards>
    <action-mappings>
        <!--首页面-->
        <action path="/product/index"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="index">
            <forward name="manager" path="/user/logout.do" ></forward>
           <forward name="success"
path="/product/productIndex.jsp"></forward>
        </action>
```

```
        <action path="/product/search"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="searchProduct">
            <forward name="success"
path="/product/productSearch.jsp"></forward>
        </action>
        <!--商品管理-->
        <action path="/product/detail"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="showDetail">
            <forward name="success"
path="/product/productDetail.jsp"></forward>
        </action>
        <action path="/product/list"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="listAllProduct">
            <forward name="success"
path="/product/productList.jsp"></forward>
        </action>
        <action path="/product/productAdd"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="addProduct" name="productActionForm">
            <forward name="success"
path="/product/addProduct.jsp"></forward>
        </action>
        <action path="/product/productModify"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="toModifyProduct">
            <forward name="success"
path="/product/productModify.jsp"></forward>
        </action>
        <action path="/product/productModified"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="modifyProduct" name="productActionForm">
            <forward name="success" path="/product/list.do"></forward>
        </action>
        <action path="/product/deleteProduct"
type="cn.com.tarena.gwap.web.action.ProductMappingDispatchAction"
parameter="deleteProductById">
            <forward name="success" path="/product/list.do"></forward>
        </action>
        <!--用户管理-->
        <action path="/user/tologin"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
```

```
parameter="toLogin">
          <forward name="success" path="/user/signin.jsp" ></forward>
       </action>
       <action path="/user/toRegister"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="toRegister">
          <forward name="success" path="/user/register.jsp"
></forward>
       </action>
       <action name="registerActionForm" validate="false"
         path="/user/checkmail"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="checkMail">
          <forward name="checkmail" path="/user/checkmail.jsp"
></forward>
       </action>
       <action path="/user/register"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="register">
          <forward name="success" path="/user/registed.jsp"
></forward>
          <forward name="refresh" path="/user/register.jsp"
></forward>
       </action>
       <action path="/user/login"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="login">
          <forward name="fail" path="/user/signin.jsp" ></forward>
          <forward name="success" path="/index.jsp" ></forward>
       </action>
       <action path="/user/manager"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="manager">
          <forward name="fail" path="/manager.jsp" ></forward>
          <forward name="admin" path="/user/listAllUser.do"></forward>
       </action>
       <action path="/user/modify"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="modify">
          <forward name="success"
path="/user/myarchives.jsp"></forward>
       </action>
       <action path="/user/changeemail"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
```

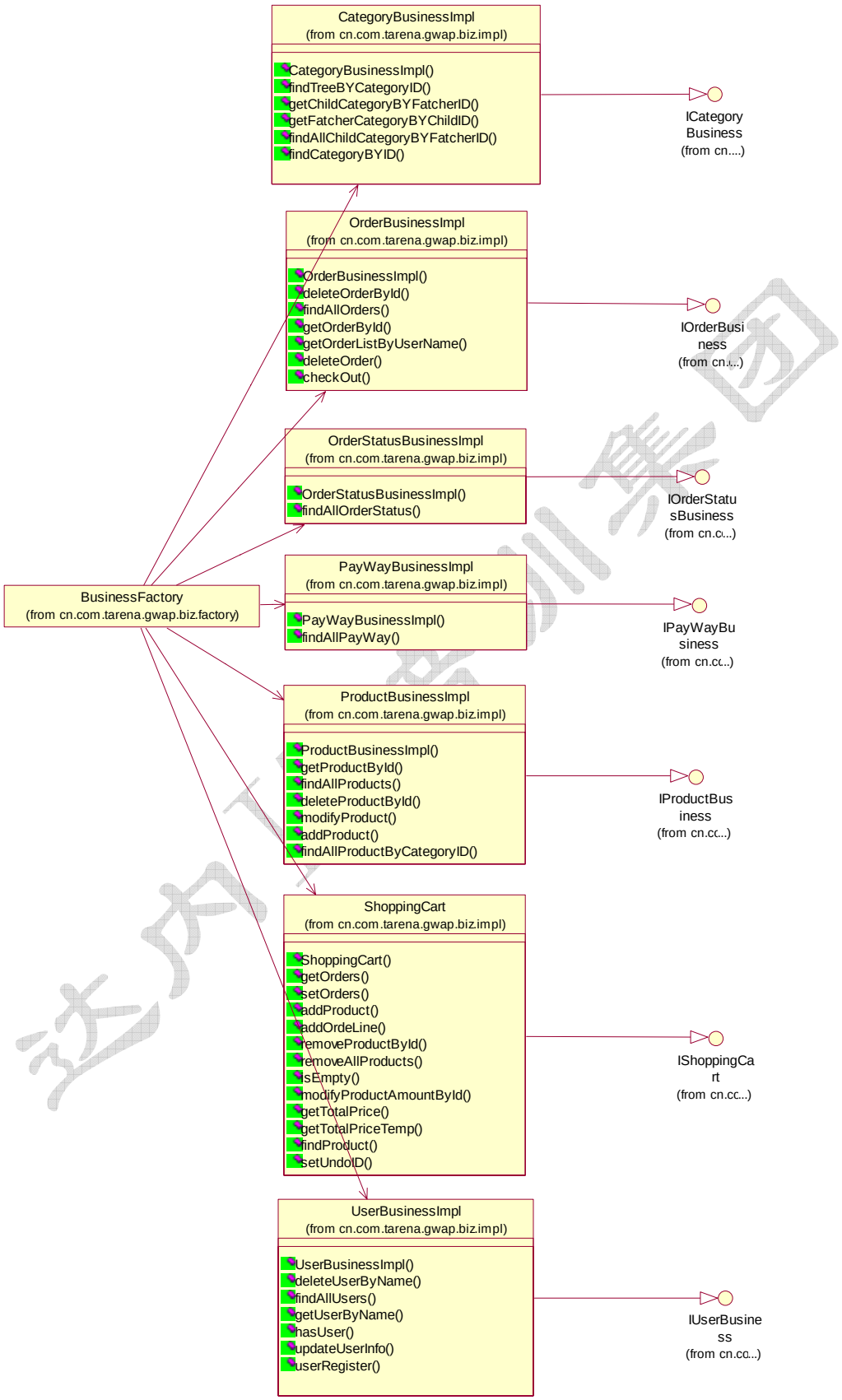
```
parameter="changeEmail">
          <forward name="success" path="/user/myemail.jsp"></forward>
       </action>
       <action path="/user/changepwd"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="changePwd">
          <forward name="success"
path="/user/passwordchange.jsp"></forward>
       </action>
       <action path="/user/query"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="query">
          <forward name="success"
path="/user/userQuery.jsp"></forward>
       </action>
       <action name="/user/delete"
         path="/user/delete"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="delete">
          <forward name="success" path="/user/listAllUser.do"
redirect="true"></forward>
       </action>
       <action path="/user/logout"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="logout">
        <forward name="manager" path="/manager.jsp" ></forward>
          <forward name="index" path="/index.jsp"></forward>
       </action>
       <action path="/user/listAllUser"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="listAllUser">
          <forward name="success" path="/user/userList.jsp"
></forward>
       </action>
       <!--购物车管理-->
       <action path="/user/add2cart"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="add2Cart">
          <forward name="tocart" path="/shopcart/shoppingcart.jsp"
redirect="true"></forward>
       </action>
       <action path="/shopcart/showShopCart"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="showShopCart">
```

```
            <forward name="success" path="/shopcart/shoppingcart.jsp"
redirect="true"></forward>
        </action>
        <action path="/shopcart/removeProduct"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="removeById">
            <forward name="success" path="/shopcart/shoppingcart.jsp"
redirect="true"></forward>
        </action>
        <action path="/shopcart/undoProduct"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="undoById">
            <forward name="success" path="/shopcart/shoppingcart.jsp"
redirect="true"></forward>
        </action>
        <action path="/shopcart/accountCart"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="account">
            <forward name="success" path="/shopcart/checkout.jsp"
redirect="true"></forward>
            <forward name="index" path="/index.jsp"
redirect="true"></forward>
        </action>
        <action path="/shopcart/updateProductNum"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="modifyNumber">
            <forward name="success" path="/shopcart/shoppingcart.jsp"
redirect="true"></forward>
        </action>
        <action path="/shopcart/removeAllProduct"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="clearShopCart">
            <forward name="index" path="/index.jsp"
redirect="true"></forward>
        </action>
        <action path="/user/saveOrder"
type="cn.com.tarena.gwap.web.action.ShopCartMappingDispatchAction"
parameter="checkOut">
            <forward name="success" path="/order/list.do"
redirect="true"></forward>
        </action>
        <!--订单管理-->
        <action path="/order/list"
type="cn.com.tarena.gwap.web.action.OrderMappingDispatchAction"
```

```
parameter="listUserOrders">
          <forward name="success"
path="/order/myorders.jsp"></forward>
       </action>
       <action path="/order/delete"
type="cn.com.tarena.gwap.web.action.OrderMappingDispatchAction"
parameter="deleteOrder">
          <forward name="success" path="/order/list.do"></forward>
       </action>
       <!-- Ajax的验证 -->
       <action validate="false"
         path="/user/username"
type="cn.com.tarena.gwap.web.action.UserMappingDispatchAction"
parameter="usernameValidate">
          <forward name="success" path="/user/register.jsp"
redirect="true"></forward>
       </action>
   </action-mappings>
</struts-config>
```

6.3 业务层设计

6.3.1 类图


CategoryBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
CategoryBusinessImpl()
findTreeBYCategoryID()
getChildCategoryBYFatcherID()
getFatcherCategoryBYChildID()
findAllChildCategoryBYFatcherID()
findCategoryBYID()
ICategory Business
(from cn....)
OrderBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
OrderBusinessImpl()
deleteOrderById()
findAllOrders()
getOrderById()
getOrderListByUserName()
deleteOrder()
checkOut()
IOrderBusi ness
(from cn...)
OrderStatusBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
OrderStatusBusinessImpl()
findAllOrderStatus()
IOrderStatu sBusiness
(from cn.c...)
PayWayBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
PayWayBusinessImpl()
findAllPayWay()
IPayWayBu siness
(from cn.cc...)
BusinessFactory
(from cn.com.tarena.gwap.biz.factory)
ProductBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
ProductBusinessImpl()
getProductById()
findAllProducts()
deleteProductById()
modifyProduct()
addProduct()
findAllProductByCategoryID()
IProductBus iness
(from cn.cc...)
ShoppingCart
(from cn.com.tarena.gwap.biz.impl)
ShoppingCart()
getOrders()
setOrders()
addProduct()
addOrdeLine()
removeProductById()
removeAllProducts()
isEmpty()
modifyProductAmountById()
getTotalPrice()
getTotalPriceTemp()
findProduct()
setUndoID()
IShoppingCa rt
(from cn.cc...)
UserBusinessImpl
(from cn.com.tarena.gwap.biz.impl)
UserBusinessImpl()
deleteUserByName()
findAllUsers()
getUserByName()
hasUser()
updateUserInfo()
userRegister()
IUserBusine ss
(from cn.cc...)

6.3.2 类描述

- **CategoryBusinessImpl 商品类别管理实现类**

方法名	功能描述	参数和返回值	异常定义
findAllChildCategoryBYFatcherID	根据父类别 ID 得到其下没有层次的所有类别的集合（LIST）	**参数：** Long **返回：** List:the list< category>	GWAPException
findTreeBYCategoryID	得到从特定类别 ID 开始，封装好的类别树（MAP）.	**参数：** Long **返回：** Map: 所有类别的 Map	GWAPException
getChildCategoryBYFatcherID	根据父类别得到紧下一级子类别集合（MAP）.	**参数：** Long **返回：** Map: 商品类别 Map	GWAPException
getFatcherCategoryBYChildID	根据当前类别 ID 得到父类别	**参数：** Long **返回：** Category: 商品类别 Pojo	GWAPException
findCategoryBYID	根据类别 ID 得到类别信息	**参数：** Long **返回：** Category	GWAPException

- **OrderBusinessImpl 订单管理实现类**

方法名	功能描述	参数和返回值	异常定义

checkOut	根据订单 Pojo，下此订单.	**参数：** Orders：订单 Pojo **返回：** void	GWAPException
deleteOrderById	根据订单 ID 删除相应的订单	**参数：** Long **返回：** void	GWAPException
findAllOrders	取得全部的订单	**参数：** / **返回：** List：全部订单的 List	GWAPException
getOrderById	根据订单 ID 获得相应订单	**参数：** Long **返回：** Orders：订单 ID 的订单	GWAPException
getOrderListByUserName	根据用户 ID 取得相应的订单 List	**参数：** String **返回：** List：用户 ID 的订单 List	GWAPException
deleteOrder	根据订单 Pojo 删除相应的订单	**参数：** Orders **返回：** void	GWAPException

- **OrderStatusBusinessImpl 订单状态管理实现类**

方法名	功能描述	参数和返回值	异常定义
findAllOrderStatus	取得全部的订单状态.	**参数：**	GWAPException

		/ **返回：** List: 全部订单状态的 List	

- **PayWayBusinessImpl 付款方式实现类**

方法名	功能描述	参数和返回值	异常定义
findAllPayWay	取得全部的支付方法式.	**参数：** Long **返回：** List: 全部支付方法的 List	GWAPException

- **ProductBusinessImpl 商品管理实现类**

方法名	功能描述	参数和返回值	异常定义
addProduct	添加商品.	**参数：** Product **返回：** void	GWAPException
getProductById	根据商品 ID 取得相应的商品	**参数：** Long **返回：** Product: 商品 Pojo	GWAPException
findAllProducts	取得全部的商品	**参数：** / **返回：** List：所有商品的 List	GWAPException
deleteProductById	删除商品根据指定的商品 ID	**参数：** Long **返回：**	GWAPException

		void	
modifyProduct	修改商品根据指定的商品 ID	**参数：** Product **返回：** void	GWAPException
findAllProductByCategoryID	根据指定的类别 ID 得到其下所有商品	**参数：** Long **返回：** List: 所有商品的 List	GWAPException

- **ShoppingCart 购物车管理实现类**

方法名	功能描述	参数和返回值	异常定义
addOrdeLine	Adds the orde line.	**参数：** OrderLine **返回：** void	GWAPException
getOrders	Gets the orders	**参数：** / **返回：** Orders	GWAPException
setOrders	Sets the orders	**参数：** Orders **返回：** void	GWAPException
addProduct	向购物车中增加一个商品	**参数：** Product int **返回：** void	GWAPException
removeProductById	根据商品 ID 在购物车	**参数：**	GWAPException

	中删除一个商品	OrderLine **返回：** void	
removeAllProducts	删除购物车中所有的商品	**参数：** Long **返回：** void	GWAPException
isEmpty	Checks if is empty	**参数：** / **返回：** bolean	GWAPException
modifyProductAmountById	根据商品 ID，修改购物车中此商品的数量	**参数：** Long int **返回：** void	GWAPException
getTotalPrice	获得购物车中商品金额的合计	**参数：** / **返回：** double: 商品价格合计	GWAPException
getTotalPriceTemp	/	**参数：** / **返回：** double	GWAPException
findProduct	从购物车中获取订单列表	**参数：** Long **返回：** OrderLine: 订单列表	GWAPException
setUndoID	/	**参数：** String **返回：**	GWAPException

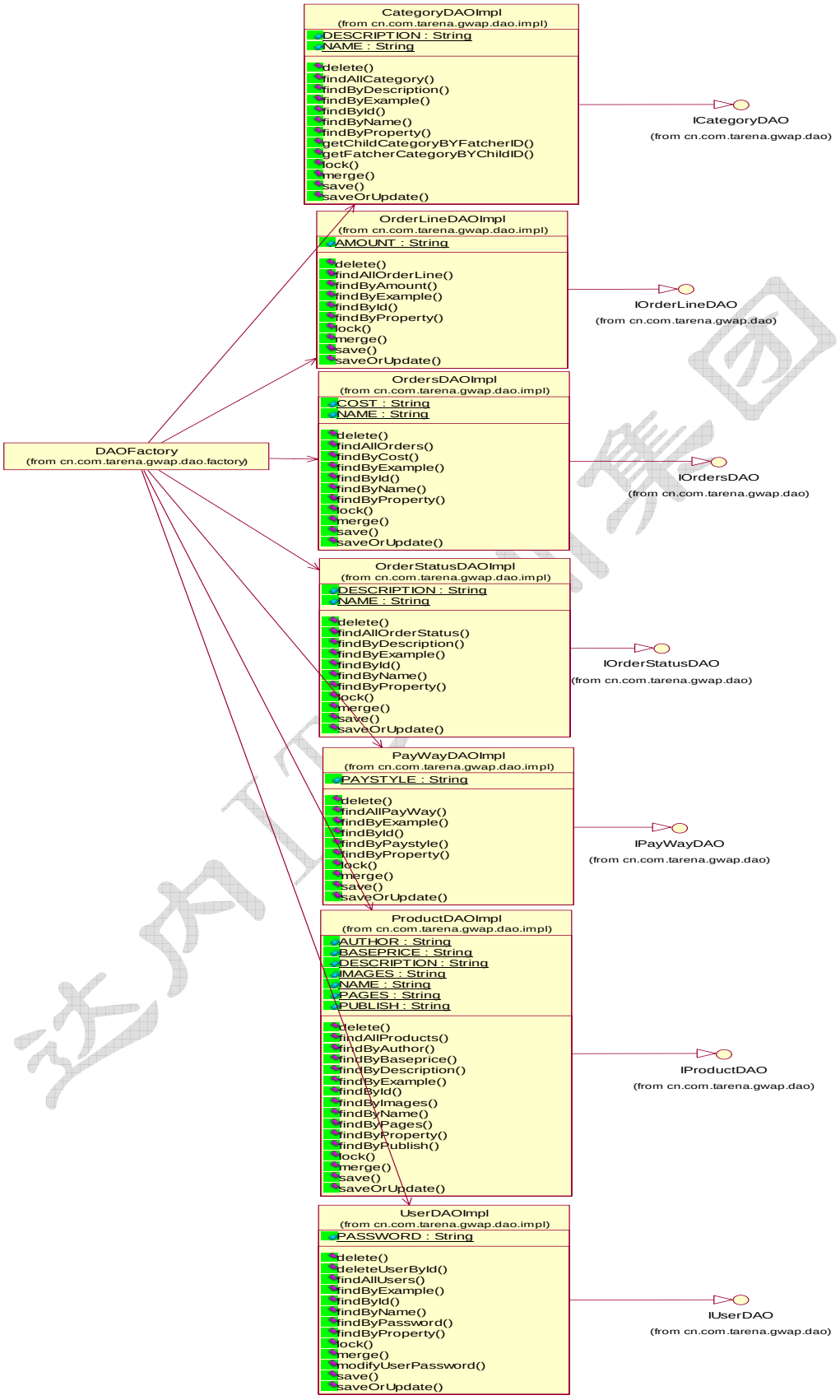
		void	

- **UserBusinessImpl 用户管理实现类**

方法名	功能描述	参数和返回值	异常定义
deleteUserByName	根据用户 Pojo，删除相应用户.	**参数：** String **返回：** boolean	GWAPException
findAllUsers	取得全部的用户	**参数：** / **返回：** List: 全部用的 List	GWAPException
getUserByName	根据用户 ID 取得相应的用户	**参数：** String **返回：** User: 用户 Pojo	GWAPException
hasUser	根据用户 username，检查相应用户在数据库中是否已经存在	**参数：** String **返回：** boolean: true 存在相应用户，false 不存在相应用户	GWAPException
updateUserInfo	更新用户信息.	**参数：** User **返回：** void	GWAPException
userRegister	根据用户 Pojo 和联系信息 Pojo，注册一个新用户	**参数：** User **返回：** void	GWAPException

6.4 持久层设计

6.4.1 类图


CategoryDAOImpl
(from cn.com.tarena.gwap.dao.impl)
DESCRIPTION : String
NAME : String
delete()
findAllCategory()
findByDescription()
findByExample()
findById()
findByName()
findByProperty()
getChildCategoryBYFatcherID()
getFatcherCategoryBYChildID()
lock()
merge()
save()
saveOrUpdate()
ICategoryDAO
(from cn.com.tarena.gwap.dao)
OrderLineDAOImpl
(from cn.com.tarena.gwap.dao.impl)
AMOUNT : String
delete()
findAllOrderLine()
findByAmount()
findByExample()
findById()
findByProperty()
lock()
merge()
save()
saveOrUpdate()
IOrderLineDAO
(from cn.com.tarena.gwap.dao)
OrdersDAOImpl
(from cn.com.tarena.gwap.dao.impl)
COST : String
NAME : String
delete()
findAllOrders()
findByCost()
findByExample()
findById()
findByName()
findByProperty()
lock()
merge()
save()
saveOrUpdate()
IOrdersDAO
(from cn.com.tarena.gwap.dao)
DAOFactory
(from cn.com.tarena.gwap.dao.factory)
OrderStatusDAOImpl
(from cn.com.tarena.gwap.dao.impl)
DESCRIPTION : String
NAME : String
delete()
findAllOrderStatus()
findByDescription()
findByExample()
findById()
findByName()
findByProperty()
lock()
merge()
save()
saveOrUpdate()
IOrderStatusDAO
(from cn.com.tarena.gwap.dao)
PayWayDAOImpl
(from cn.com.tarena.gwap.dao.impl)
PAYSTYLE : String
delete()
findAllPayWay()
findByExample()
findById()
findByPaystyle()
findByProperty()
lock()
merge()
save()
saveOrUpdate()
IPayWayDAO
(from cn.com.tarena.gwap.dao)
ProductDAOImpl
(from cn.com.tarena.gwap.dao.impl)
AUTHOR : String
BASEPRICE : String
DESCRIPTION : String
IMAGES : String
NAME : String
PAGES : String
PUBLISH : String
delete()
findAllProducts()
findByAuthor()
findByBaseprice()
findByDescription()
findByExample()
findById()
findByImages()
findByName()
findByPages()
findByProperty()
findByPublish()
lock()
merge()
save()
saveOrUpdate()
IProductDAO
(from cn.com.tarena.gwap.dao)
UserDAOImpl
(from cn.com.tarena.gwap.dao.impl)
PASSWORD : String
delete()
deleteUserById()
findAllUsers()
findByExample()
findById()
findByName()
findByPassword()
findByProperty()
lock()
merge()
modifyUserPassword()
save()
saveOrUpdate()
IUserDAO
(from cn.com.tarena.gwap.dao)

6.4.2 类描述

- **CategoryDAOImpl 提供产品分类相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** Category **返回：** void	GWAPException
findAllCategory	取得全部的商品分类.	**参数：** / **返回：** List: 全部的产品分类的 List	GWAPException
findByDescription	Find by description.	**参数：** Object **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** Category **返回：** List: Pojo 的 List	GWAPException
findById	根据 Pojo 的主键取得数据库中的相应数据.	**参数：** Long **返回：** Category	GWAPException
findByName	Find by name.	**参数：** Object **返回：** List	GWAPException

findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List: Pojo 的 List	GWAPException
getChildCategoryBYFatcherID	取得子的 Category 对象集合，根据当前 category.id。	**参数：** Long **返回：** List	GWAPException
getFatcherCategoryBYChildID	取得父亲 Category 对象，根据当前 category.id。	**参数：** Long **返回：** Category	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo.	**参数：** Category **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo.	**参数：** Category **返回：** Category	GWAPException
save	把尚未持久化的 Pojo 登录至数据库.	**参数：** Category **返回：**	GWAPException

		void	
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作.	**参数：** Category **返回：** void	GWAPException

- **OrderLineDAOImpl 提供订单明细相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** OrderLine **返回：** void	GWAPException
findAllOrderLine	取得全部的订单明细.	**参数：** / **返回：** List	GWAPException
findByAmount	Find by amount.	**参数：** Object **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** OrderLine **返回：** List: Pojo 的 List	GWAPException
findById	根据 Pojo 的主键取得数据库中的相应数据.	**参数：** Long **返回：** OrderLine	GWAPException

findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo.	**参数：** OrderLine **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo.	**参数：** OrderLine **返回：** OrderLine	GWAPException
save	把尚未持久化的 Pojo 登录至数据库.	**参数：** OrderLine **返回：** void	GWAPException
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作.	**参数：** OrderLine **返回：** void	GWAPException

- **OrdersDAOImpl 提供订单相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** Orders **返回：** void	GWAPException
findAllOrders	Method	**参数：** /	GWAPException

	In IOrdersDAO.	**返回：** List	
findByCost	Find by cost.	**参数：** Object **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** Orders **返回：** List	GWAPException
findById	根据 Pojo 的主键取得数据库中的相应数据.	**参数：** Long **返回：** Orders	GWAPException
findByName	Find by name.	**参数：** Object **返回：** List	GWAPException
findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo.	**参数：** Orders **返回：** void	GWAPException
save	把尚未持久化的 Pojo 登录至数据库.	**参数：** Orders **返回：**	GWAPException

		void	
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo.	**参数：** Orders **返回：** Orders	GWAPException
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作.	**参数：** Orders **返回：** void	GWAPException

- **OrderStatusDAOImpl 提供订单状态相关数据库**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** OrderStatus **返回：** void	GWAPException
findAllOrderStatus	Method In IOrdersDAO.	**参数：** / **返回：** List	GWAPException
findByDescription	Find by description.	**参数：** Object **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** OrderStatus **返回：** List	GWAPException
findById	根据 Pojo 的主键取得	**参数：**	GWAPException

	数据库中的相应数据.	Long **返回：** OrderStatus	
findByName	Find by name.	**参数：** Object **返回：** List	GWAPException
findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo.	**参数：** OrderStatus **返回：** void	GWAPException
save	把尚未持久化的 Pojo 登录至数据库..	**参数：** OrderStatus **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo..	**参数：** OrderStatus **返回：** OrderStatus	GWAPException
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作.	**参数：** OrderStatus **返回：** void	GWAPException

- **PayWayDAOImpl 提供支付方式相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删	**参数：**	GWAPException

	除数据库中相应数据.	PayWay **返回：** void	
findAllPayWay	Method in IPayWayDAO.	**参数：** / **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** PayWay **返回：** List：Pojo 的 List	GWAPException
findById	根据 Pojo 的主键取得数据库中的相应数据.	**参数：** Long **返回：** PayWay	GWAPException
findByPaystyle	Find by paystyle.	**参数：** Object **返回：** List	GWAPException
findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo.	**参数：** PayWay **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持	**参数：** PayWay	GWAPException

	久化的 Pojo	**返回：** PayWay	
save	把尚未持久化的 Pojo 登录至数据库.	**参数：** PayWay **返回：** void	GWAPException
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作	同上	GWAPException

- **ProductDAOImpl 提供商品相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** Product **返回：** void	GWAPException
findAllProducts	Method in IProductDAO.	**参数：** / **返回：** List	GWAPException
findByAuthor	Find by author.	**参数：** Object **返回：** List	GWAPException
findByBaseprice	Find by baseprice.	同上	GWAPException
findByDescription	Find by description.	同上	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据.	**参数：** Product **返回：**	GWAPException

		List	
findById	根据 Pojo 的主键取得数据库中的相应数据.	**参数：** Long **返回：** Product	GWAPException
findByImages	Find by images.	**参数：** Object **返回：** List	GWAPException
findByName	Find by name.	同上	GWAPException
findByPages	Find by pages.	同上	GWAPException
findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据.	**参数：** String Object **返回：** List	GWAPException
findByPublish	Find by publish.	**参数：** Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo	**参数：** Product **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo	**参数：** Product **返回：** Product	GWAPException
save	把尚未持久化的 Pojo	**参数：**	GWAPException

	登录至数据库	Product **返回：** void	
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作	同上	GWAPException

- **UserDAOImpl 提供用户相关数据库操作**

方法名	功能描述	参数和返回值	异常定义
delete	根据持久化的 Pojo 删除数据库中相应数据.	**参数：** User **返回：** void	GWAPException
deleteUserById	Delete user by id	**参数：** String **返回：** void	GWAPException
findAllUsers	IUserDAO	**参数：** / **返回：** List	GWAPException
findByExample	根据范例 Pojo 取得数据库中的相应数据	**参数：** User **返回：** List: Pojo 的 List	GWAPException
findById	根据 Pojo 的主键取得数据库中的相应数据	**参数：** Long **返回：** User	GWAPException
findByName	Find by name	**参数：** String **返回：**	GWAPException

		User	
findByPassword	Find by password	**参数：** Object **返回：** List	GWAPException
findByProperty	根据 Pojo 的属性名和 Pojo 属性的值在数据库中取得相应数据	**参数：** String Object **返回：** List	GWAPException
lock	把游离的 Pojo 再次变成持久化的 Pojo	**参数：** User **返回：** void	GWAPException
merge	合并游离的 Pojo，使游离的 Pojo 再次变成持久化的 Pojo	**参数：** User **返回：** User	GWAPException
modifyUserPassword	根据用户 Pojo，修改用户的密码	**参数：** User **返回：** void	GWAPException
save	把尚未持久化的 Pojo 登录至数据库	同上	GWAPException
saveOrUpdate	根据 Pojo 的状态，进行登录或者更新至数据库的操作	同上	GWAPException

6.4.3 映射文件描述

- **Category.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="Category" table="category">
        <id name="categoryid" type="java.lang.Long">
            <column name="category_id" precision="10" scale="0" />
            <generator class="increment">
            </generator>
        </id>
        <property name="fathercategoryid" type="java.lang.Long">
            <column name="fathercategoryid" precision="6" scale="0" />
        </property>
        <property name="name" type="java.lang.String">
            <column name="name" length="128" not-null="true" />
        </property>
        <property name="description" type="java.lang.String">
            <column name="description" length="512" />
        </property>
        <set name="products" inverse="true">
            <key>
                <column name="category_id" precision="10" scale="0" />
            </key>
            <one-to-many class="Product"/>
        </set>
    </class>
</hibernate-mapping>
```

- **OrderLine.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
```

```
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="OrderLine" table="orderline">
        <id name="lineid" type="java.lang.Long" unsaved-value="null">
            <column name="orderline_id" precision="20" scale="0"
not-null="true"/>
            <generator class="increment">
            </generator>
        </id>
        <many-to-one name="product" class="Product" fetch="join"
lazy="false">
            <column name="product_id" precision="16" scale="0" />
        </many-to-one>
        <many-to-one name="orders" class="Orders" fetch="join"
lazy="false">
            <column name="orders_id" precision="20" scale="0" />
        </many-to-one>
        <property name="amount" type="java.lang.Integer">
            <column name="amount" precision="20" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

- **Orders.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="Orders" table="orders">
        <id name="orderid" type="java.lang.Long" unsaved-value="null">
            <column name="orders_id" precision="20" scale="0"
not-null="true"/>
            <generator class="increment">
            </generator>
        </id>
        <many-to-one name="payway" class="PayWay" fetch="join"
lazy="false">
            <column name="payway_id" precision="3" scale="0" />
        </many-to-one>
```

```
        <many-to-one name="users" class="User" fetch="join"
lazy="false">
            <column name="username" length="16" />
        </many-to-one>
        <many-to-one name="orderstatus" class="OrderStatus" fetch="join"
lazy="false">
            <column name="orderstatus_id" precision="3" scale="0" />
        </many-to-one>

        <property name="name" type="java.lang.String">
            <column name="name" length="32" />
        </property>
        <property name="cost" type="java.lang.Double">
            <column name="cost" precision="15" scale="3" not-null="true"
/>
        </property>
        <set name="orderlines" fetch="join" inverse="true" cascade="all"
lazy="false">
            <key>
                <column name="orders_id" precision="20" scale="0" />
            </key>
            <one-to-many class="OrderLine" />
        </set>
    </class>
</hibernate-mapping>
```

- **OrderStatus.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="OrderStatus" table="orderstatus">
        <id name="statusid" type="java.lang.Long">
            <column name="orderstatus_id" precision="3" scale="0" />
            <generator class="increment">
            </generator>
        </id>
        <property name="name" type="java.lang.String">
            <column name="name" length="32" not-null="true" />
```

```
        </property>
        <property name="description" type="java.lang.String">
            <column name="description" length="64" />
        </property>
        <set name="orderses" inverse="true">
            <key>
                <column name="orderstatus_id" precision="3" scale="0" />
            </key>
            <one-to-many class="Orders" />
        </set>
    </class>
</hibernate-mapping>
```

- **PayWay.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="PayWay" table="payway">
        <id name="paywayid" type="java.lang.Long">
            <column name="payway_id" precision="3" scale="0" />
            <generator class="increment">
            </generator>
        </id>
        <property name="paystyle" type="java.lang.String">
            <column name="paystyle" length="64" not-null="true" />
        </property>
        <set name="orderses" inverse="true">
            <key>
                <column name="payway_id" precision="3" scale="0" />
            </key>
            <one-to-many class="Orders" />
        </set>
    </class>
</hibernate-mapping>
```

- **Product.hbm.xml**

```
<?xml version="1.0"?>
```

```
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="Product" table="product">
        <id name="productid" type="java.lang.Long">
            <column name="product_id" precision="16" scale="0" />
            <generator class="increment">
            </generator>
        </id>
        <many-to-one name="category" class="Category" fetch="select"
lazy="false">
            <column name="category_id" precision="10" scale="0" />
        </many-to-one>

        <property name="name" type="java.lang.String">
            <column name="name" length="64" />
        </property>
        <property name="description" type="java.lang.String">
            <column name="description" length="4000" />
        </property>
        <property name="baseprice" type="java.lang.Double">
            <column name="baseprice" precision="12" />
        </property>
        <property name="author" type="java.lang.String">
            <column name="author" length="128"  />
        </property>
        <property name="publish" type="java.lang.String">
            <column name="publish" length="256" />
        </property>
        <property name="pages" type="java.lang.Long">
            <column name="pages" precision="6" scale="0" />
        </property>
        <property name="images" type="java.lang.String">
            <column name="images" length="128" />
        </property>
        <set name="orderlines" inverse="false">
            <key>
                <column name="product_id" precision="16" scale="0" />
            </key>
            <one-to-many class="OrderLine" />
```

```
        </set>
    </class>
</hibernate-mapping>
```

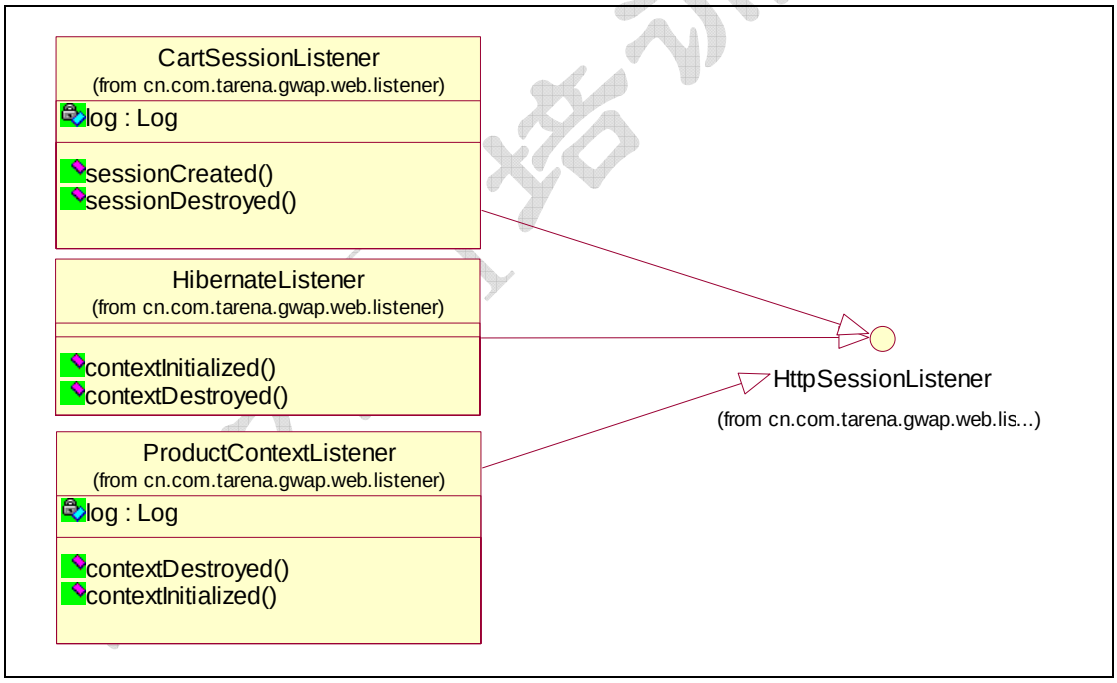
- **User.hbm.xml**

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse - Hibernate Tools
-->
<hibernate-mapping package="cn.com.tarena.gwap.pojo">
    <class name="User" table="users">
        <id name="username" type="java.lang.String">
            <column name="username" length="20" not-null="true"/>
        </id>
        <property name="password" type="java.lang.String">
            <column name="password" length="12" not-null="true" />
        </property>
        <property name="email" type="java.lang.String">
            <column name="email" length="32"/>
        </property>
        <property name="telephone" type="java.lang.String">
            <column name="telephone" length="16"/>
        </property>
        <property name="mobtelephone" type="java.lang.String">
            <column name="mobtelephone" length="16"/>
        </property>
        <property name="province" type="java.lang.String">
            <column name="province" length="32"/>
        </property>
        <property name="city" type="java.lang.String">
            <column name="city" length="32"/>
        </property>
        <property name="address" type="java.lang.String">
            <column name="address" length="200"/>
        </property>
        <property name="state"  type="java.lang.Long">
            <column name="state" precision="16" not-null="true" />
        </property>
        <property name="confirmcode" type="java.lang.String">
            <column name="confirmcode" length="20" not-null="true" />
        </property>
```

```
        <set name="orderses" inverse="true" cascade="delete-orphan"
lazy="false">
            <key>
                <column name="username" length="16" />
            </key>
            <one-to-many class="Orders" />
        </set>
    </class>
</hibernate-mapping>
```

6.5 监听器设计

6.5.1 类图



6.5.2 类描述

- **CartSessionListener 购物车监听器**

方法名	功能描述	参数和返回值	异常定义

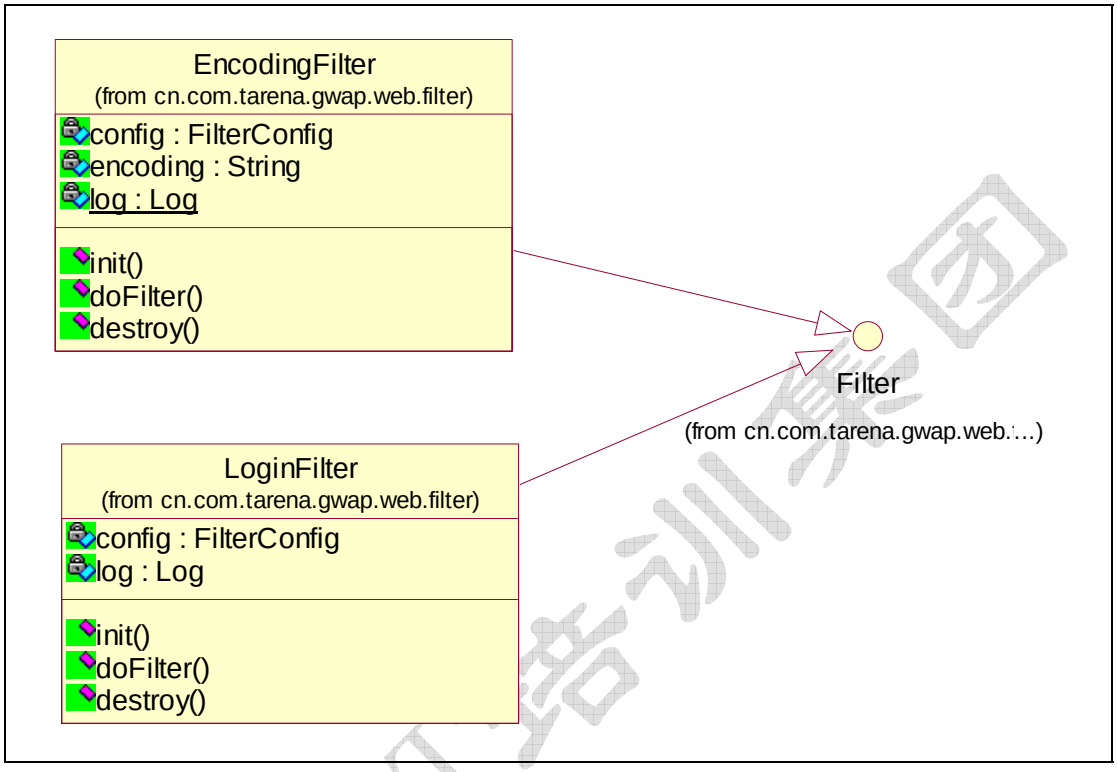
sessionCreated	初始化购物车对象到 session 中	**参数：** HttpSessionEvent **返回：** void	/
sessionDestroyed	销毁 session 中购物车对象	同上	/

- **ProductContextListener 商品内容监听器**

方法名	功能描述	参数和返回值	异常定义
contextInitialized	初始化（系统启动），加载 IProductBusiness，IPayWayBusiness，IOrderStatusBusiness，ICategoryBusiness 到内存。	**参数：** ServletContextEvent **返回：** void	/
contextDestroyed	注销监听器	同上	/

6.6 过滤器设计

6.6.1 类图



6.6.2 类描述

- **EncodingFilter 编码过滤器**

方法名	功能描述	参数和返回值	异常定义
init	初始化 web.xml 配置的过滤规则.	**参数：** FilterConfig **返回：** void	ServletException
doFilter	编码过滤	**参数：** ServletRequest ServletResponse FilterChain	IOException, ServletException

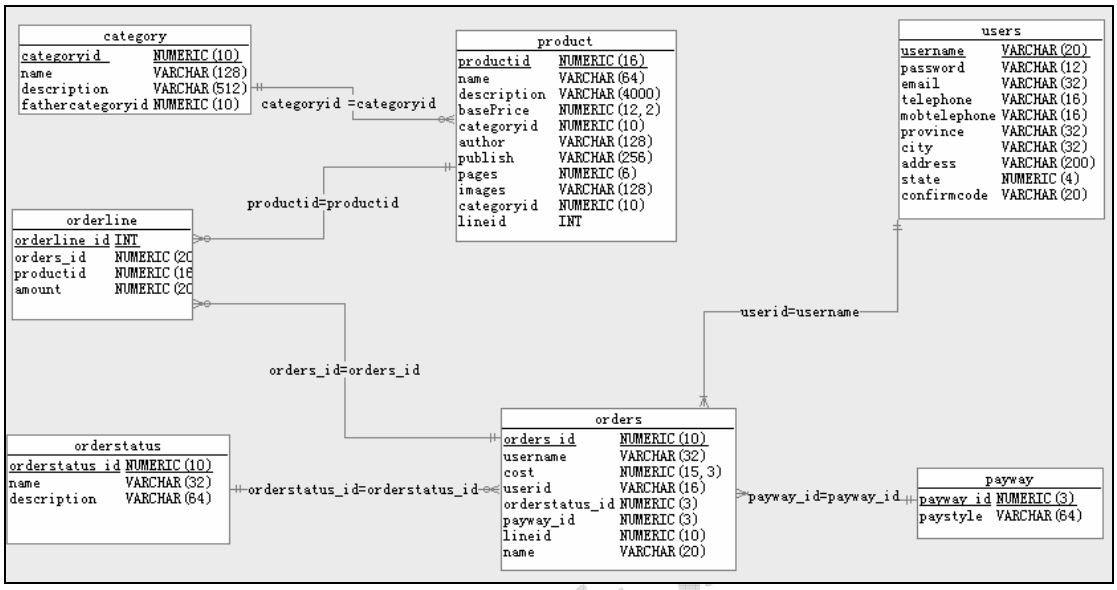
		返回： void	
destroy	注销过滤器调用	/	/

- **LoginFilter 登陆过滤器**

方法名	**功能描述**	**参数和返回值**	**异常定义**
init	初始化过滤规则.	**参数：** FilterConfig **返回：** void	ServletException
doFilter	登陆过滤功能（限制非法用户）	**参数：** ServletRequest ServletResponse FilterChain **返回：** void	IOException, ServletException
destroy	注销过滤器调用	/	/

7．数据库设计

7.1 E-R 图



7.2 数据实体描述

- **类别表（category）**

字段名称	类型	约束	描述
category_id	integer	pk	类别 id
fathercategoryid	integer	not null	父类别 id
name	varchar(128)	not null	类别名
description	varchar(512)	/	类别说明

- **商品表（product）**

字段名称	类型	约束	描述
product _id	integer	auto_increment/ pk	商品 id
name	varchar(40)	not null	商品名
basePrice	numeric(12,2)	not null	单价
category_id	integer	/	类别 id
author	varchar(128)	not nul	作者
publish	varchar(256)	not nul	出版社
pages	int	/	页数

images	varchar(128)	/	图片
description	varchar(512)	/	描述

- **用户表（users）**

字段名称	类型	约束	描述
username	varchar(20)	pk	用户名
password	varchar(12)	not null	密码
email	varchar(32)	not null	Email
telephone	varchar(16)	/	固定电话
mobtelephone	varchar(16)	/	移动电话
province	varchar(32)	/	省份
city	varchar(32)	/	城市
address	varchar(200)	/	地址
state	integer	not null	用户状态（0-已验证、1-未验证）
confirmcode	varchar(20)	not null	验证码

- **订单状态表（orderstatus）**

字段名称	类型	约束	描述
orderstatus_id	integer	pk	状态 id
name	varchar(20)	not null	状态名称
description	varchar(64)	/	状态描述

- **付款方式表（payway）**

字段名称	类型	约束	描述
payway_id	integer	pk	付款方式 id
paystyle	varchar(64)	not null	付款方式名称

- **订单表（orders）**

字段名称	类型	约束	描述
orders_id	integer	auto_increment / pk	订单 id
username	varchar(20)	not null	用户名称
payway_id	integer	/	付款方式 id
orderstatus_id	integer	/	订单状态 id
name	varchar(20)	/	订单名称（用户名-日期）
cost	numeric(15,3)	not null	订单金额

- **订单项表（orderline）**

字段名称	类型	约束	描述
orderline_id	integer	auto_increment / pk	订单项 id
orders_id	integer	not null	订单 id
product_id	integer	not null	商品 id
amount	numeric(20,2)	not null	商品数量

7.3 实体关系描述

```
alter table product
add constraint product_category_id_fk foreign key(category_id) references category(category_id);

alter table orders
add constraint orders_username_fk foreign key(username) references users(username);
alter table orders
add constraint orders_payway_fk foreign key(payway_id) references payway(payway_id);
alter table orders
add constraint orders_orderstatus_fk foreign key(orderstatus_id) references orderstatus(orderstatus_id);

alter table orderline
add constraint orderline_orders_id_fk foreign key(orders_id) references orders(orders_id);
alter table orderline
add constraint orderline_product_id_fk foreign key(product_id) references product(product_id);
```

7.4 实体数据初始化

```
--初始化商品类别表
insert into category values(1,0,'小说','顶级分类');
insert into category values(2,0,'文学','顶级分类');
insert into category values(3,0,'商业','顶级分类');
insert into category values(4,0,'艺术','顶级分类');
insert into category values(5,1,'当代小说','当代小说');
insert into category values(6,1,'近代小说','近代小说');
--初始化商品表
INSERT INTO product VALUES(1,'杜拉拉 2 华年似水(《杜拉拉升职记》第二部：超越职场的华年似水)','希望这本虚构的小说，能够对人们的生活有一些超越职场规则的现实意义，使我能回报市场和读者的知遇于万一<br>\r\n。——李可<br>\r\n2005－2006 年，中国一线城市。<br>\r\n 对大部分人而言，挣钱的速度明显跟不上房价的涨幅。而沉寂四年的 A 股正走出漫漫熊途，开始了从 998 向 6140 的辉煌进发。<br>\r\n 淡泊从容越来越成为奢侈，时代在湍
```

急中奔流。
\r\n 作为对各种难题胸有成竹的 PROBLEM SOLVER，杜拉拉日显强大。当女性的缺点和可爱，日益为专业的成熟及规则所取代，
\r\n 让人不禁疑惑“ 成熟 ”是褒义还是贬义？而职场版的说文解字中，下属无性别，上级的性别则只关乎授权的程度，WHY 比 WHAT 更重要。
\r\n 每个人都有难处：姚杨年过而立竞聘经理失败，怀孕日程遥遥无期；李坤掏心掏肺栽培小苏，反遭小苏翻脸无情；林如成被下属的发财气得发疯，
\r\n 但无从干掉业绩不错的股神杨瑞；TONY 林认为培训生制度弱智，却被迫协助好大喜功的 HR 成就功名。
\r\n 对八十后沙当当而言，爱人不是问题，问题是房子的产权。
\r\n 对七十后杜拉拉而言，失恋不是问题，问题是没有更好的恋情。
\r\n 宽带薪酬制中吃了亏的拉拉决定跳槽，怀着恶劣的心情，在 2006 炎热的盛夏，她踏上了漫漫的求职之路……作为著名的“ 倔驴 ”，她一直以为：你我会情长意久。
\r\n 中产之梦多半就是在这号人手中实现的。 ',28.00,5,'李可','陕西师范大学出版',283,'images/product/zb_20232215_m.jpg');

--初始化用户信息表
INSERT INTO users VALUES ('admin', 'admin', 'admin@tarena.com.cn', '021-61209549', '13900000000', '上海', '上海', '北京东路 668 弄', 0, '12345678');
INSERT INTO users VALUES ('tarena', 'tarena', 'tarena@tarena.com.cn', '021-61209549', '13900000000', '上海', '上海', '北京东路 668 弄', 0, '12345678');
--初始化订单状态表
INSERT INTO orderstatus VALUES (1,'付款...','等待买家付款...');
INSERT INTO orderstatus VALUES (2,'收获...','等待买家收获确认...');
INSERT INTO orderstatus VALUES (3,'结束','订单已经关闭。');
--初始化付款方式表
INSERT INTO payway VALUES (1,'货到付款');
INSERT INTO payway VALUES (2,'邮局汇款');
INSERT INTO payway VALUES (3,'银行转帐');

8．程序结构设计

必须使用相同的工程名、目录结构和目录名规范，具体要求如下：

项目	要求	说明
项目名称	GWAP(VER3.0)-0x	每个小组使用一个仓库，0x 代码自己项目组的编号（每组项目名称相同）
项目目录环境	GWAP(VER3.0)-0x — CVS |— doc |— lib |— refer |— sql |— src |— test |— web |— *release*	CVS：连接 cvs 后自动产生的桩文件 doc：项目 java 代码 api 文件 lib：项目需要使用的第三方包 refer：主要包括 design、task、test 三个文件夹 sql：存放“ 数据库创建、数据初始化 ”脚本文件 src：系统 java 代码，目录结构参考“ 工程代码 ”

		test：junit 测试代码，和 src 结构对应 Web：web 工程文件目录，如：jsp、js、css 等 *release：临时文件加，ant 自动维护*
JAVA 代码结构	src—cn—com—tarena—gwap—biz |—common |—dao |—exception |—pojo |—web—action |—filter |—forms |—listener |—messages	biz：业务层代码，包括 impl 和接口类 common：工具包，hib 的 dao 工具和 log dao：数据持久层，包括 impl 和接口类 exception：自定义异常类 pojo：数据库表映射类 action：4 个主要控制器 filter：过滤器 forms：struts 表单类 listener：监听器 messages：struts 资源文件 xxx.properties
WEB 代码结构	web—css |—images |—js |—order |—product |—shopcart |—user |—WEB-INF |—xxx.jsp	css：存放 tarene.css 文件，和其它 css 文件 images：图片 js：tarena.Js 文件，和其它 JavaScripter 文件 order：订单管理相关 JSP 文件 product：商品管理相关 JSP 文件 shopcart：购物车相关 JSP 文件 user：用户管理相关 JSP 文件 WEB-INF：默认目录，编译后的文件和 xml xxx.jsp：顶级 JSP 文件
IDE 配置	Web Context-root : /gwap3 Source : src 和 test JRE : jdk1.6.0 Compiler compliance level : 6.0 Text file encoding : UTF-8	工程名-右键-Properties

9．系统环境设计

9.1 开发环境设计

- 操作系统：Red Had Linux9 或 WINDOWS XP
- 虚拟机：jdk-6-linux-i586 或 jdk-6-win-i586
- 浏览器：FireFox2.0 或 IE6
- Web 服务器：Apache Tomcat 6.0

- 数据库：MySql 5.5
- 设计工具：IBM Rose 2003
- IDE 工具：MyEclipse6.0（集成 Eclipse3.3）
  - ✓ 数据库客户端插件：Quantum DB3.0
  - ✓ E-R 插件：clay1.4
  - ✓ 代码风格插件：checkstyle4.3

9.2 发布环境设计

- 操作系统：Red Had Linux9 或 WINDOWS XP
- 虚拟机：jdk-6-linux-i586 或 jdk-6-win-i586
- 浏览器：FireFox2.0 或 IE6
- Web 服务器：Apache Tomcat 6.0
- 数据库：MySql 5.5

9.3 编译和发布工具

9.3.1 ANT 介绍

- ANT 的概念：ANT 是著名 Java 开源组织 Apache 的一个项目，是一个基于 java 的 build 工具。它可以使你通过 ant 脚本语言，自动你的项目拷贝到某个目录，发布项目，或者生成一些代码，执行 SQL 语言。总之它可以帮助你完成项目开发中除了开发代码以外的大部分辅助性工作。
- ANT 的作用：当一个代码项目大了以后，每次重新编译，打包，测试等都会变得非常复杂而且重复。JAVA 使用 ANT（一种流程脚本引擎），用于自动化调用程序完成项目的编译，打包，测试等工作。每个 ANT 脚本（缺省叫 build.xml）中设置了一系列任务(target)。

9.3.2 ANT 在 GWAP 中的使用

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="GWAP" default="release" basedir=".">
    <!-- 变量 -->
    <property name="ant.dir" value="~temp" />
    <property name="release.dir" value="release" />
    <property name="src.dir" value="${ant.dir}/src" />
```

```
    <property name="test.dir" value="${ant.dir}/test" />
    <property name="lib.dir" value="${ant.dir}/lib" />

    <property name="doc.dir" value="${ant.dir}/doc" />
    <property name="sql.dir" value="${ant.dir}/sql" />
    <property name="web.dir" value="${ant.dir}/Web" />
    <property name="classes.dir"
value="${ant.dir}/Web/WEB-INF/classes" />
    <!-- 如果使用了CVS管理请定义 -->
    <property name="cvsroot"
value=":pserver:cvsroot@192.168.0.188:/cvs/cvsroot" />
    <property name="cvspass" value="" />
    <property name="cvs.passfile" value="${ant.dir}/ant.cvspass" />
    <property name="project.dir" value="group/gwap/GWAP(VER3.1)" />

    <property name="reports.dir" value="${ant.dir}/reports" />

    <!—- 定义了发布的war包名称 -->
    <property name="web.war" value="gwap3_1.war" />
    <!—- 必须修改使用自己的tomcat.home -->
    <property name="tomcat.home" value="D:/Program Files/Apache Software
Foundation/Tomcat 6.0" />

    <!-- ClassPath -->
    <path id="master-classpath">
        <fileset file="${lib.dir}/*.jar" />
        <fileset file="${classes.dir}" />
    </path>

    <!-- 初始化 -->
    <target name="init" description="init env">
        <tstamp>
            <format property="today" pattern="yyyy-MM-dd hh:mm:ss" />
        </tstamp>
        <echo message="${today}" />

        <delete dir="${release.dir}" />
    </target>

    <!-- CVS -->
    <target name="checkout" depends="init" description="check out cvs
project">
        <mkdir dir="${ant.dir}" />
        <copy todir="${ant.dir}/doc">
```

```
            <fileset dir="${basedir}/doc" includes="**/*" />
        </copy>
        <copy todir="${ant.dir}/lib">
            <fileset dir="${basedir}/lib" includes="**/*" />
        </copy>
        <copy todir="${ant.dir}/sql">
            <fileset dir="${basedir}/sql" includes="**/*" />
        </copy>
        <copy todir="${ant.dir}/src">
            <fileset dir="${basedir}/src" includes="**/*" />
        </copy>
        <copy todir="${ant.dir}/test">
            <fileset dir="${basedir}/test" includes="**/*" />
        </copy>
        <copy todir="${ant.dir}/Web">
            <fileset dir="${basedir}/Web" includes="**/*" />
        </copy>
        <!--
        <cvspass cvsroot="${cvsroot}"  password="${cvspass}"
passfile="${cvs.passfile}"/>
       <cvs cvsRoot="${cvsroot}" command="checkout -d ${ant.dir}"
package="${project.dir}" dest="${basedir}"
passfile="${cvs.passfile}"/>
       -->
    </target>

    <!-- 编译 -->
    <target name="javac" depends="checkout" description="compile the
source files">
        <mkdir dir="${classes.dir}" />
        <javac srcdir="${src.dir}" encoding="utf8"
destdir="${classes.dir}" target="1.6">
            <classpath refid="master-classpath" />
        </javac>
        <javac srcdir="${test.dir}" encoding="utf8"
destdir="${classes.dir}" target="1.6">
            <classpath refid="master-classpath" />
        </javac>
        <copy todir="${classes.dir}">
            <fileset dir="${src.dir}" includes="log4j.properties" />
            <fileset dir="${src.dir}" includes="**/*.xml" />
        </copy>
    </target>
```

```
    <!-- 测试 -->
    <target name="junit" depends="javac" description="create junit test
report">
        <mkdir dir="${release.dir}/report/java" />
        <mkdir dir="${release.dir}/report/html" />
        <junit>
            <formatter type="xml" />
            <batchtest haltonfailure="no">
                <fileset dir="${src.dir}">
                    <include name="**/*Test.java" />
                    <include name="**/AllTests.java" />
                </fileset>
            </batchtest>
        </junit>
        <junitreport todir="${release.dir}/report/java">
            <fileset dir="${release.dir}/report/java">
                <include name="TEST-*.xml" />
            </fileset>
            <report format="noframes"
todir="${release.dir}/report/html" />
        </junitreport>
    </target>

    <!-- 生成API -->
    <target name="doc" depends="javac" description="create api doc">
        <mkdir dir="${doc.dir}" />
        <javadoc destdir="${doc.dir}" encoding="utf8" author="true"
version="true" use="true" windowtitle="GWAP V3.0 API">
            <packageset dir="${src.dir}" defaultexcludes="yes">
                <include name="**/**" />
            </packageset>
            <classpath refid="master-classpath" />
            <doctitle>
                <![CDATA[<h1>Welcome GWAP</h1>]]>
            </doctitle>
        <bottom>
            <![CDATA[<i>Tarena All Rights Reserved.</i>]]>
            </bottom>
    <tag name="todo" scope="all" description="To do:" />
</javadoc>
</target>

<!-- 生成WAR -->
<target name="war" depends="doc" description="make .war file">
```

```
<mkdir dir="${release.dir}" />
<mkdir dir="${release.dir}/pro" />
<war destfile="${release.dir}/pro/${web.war}"
webxml="${web.dir}/WEB-INF/web.xml">
    <fileset dir="${web.dir}" excludes="**/web.xml" />
    <lib dir="${lib.dir}" />
    <classes dir="${classes.dir}" />
</war>
</target>

<!-- Building -->
<target name="building" depends="war" description="copy * file">
<mkdir dir="${release.dir}/api" />
<mkdir dir="${release.dir}/sql" />
<copy todir="${release.dir}/api">
    <fileset dir="${doc.dir}" />
</copy>
<copy todir="${release.dir}/sql">
    <fileset dir="${sql.dir}" />
</copy>
<copy file="${release.dir}/pro/${web.war}"
todir="${tomcat.home}/webapps" />
<delete dir="${ant.dir}" />
</target>

<!-- Release -->
<target name="release" description="release gwap">
<echo message="+=========================================+" />
<echo message="|     Start Building GWAP project.      |" />
<echo message="+=========================================+" />

<antcall target="building" />

<echo message="+=========================================+" />
<echo message="|     End Building GWAP project.        |" />
<echo message="+=========================================+" />
</target>

</project>
```

10．用例实现

10.1 交易系统

10.1.1 显示首页面

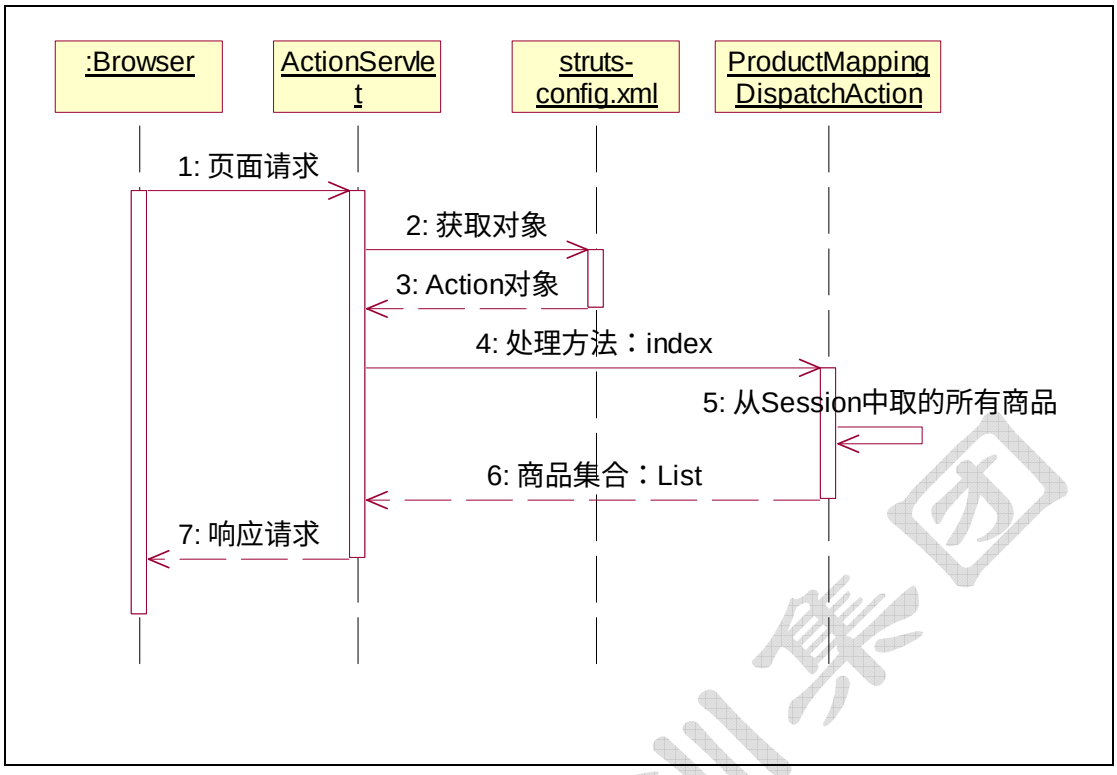
- **功能概述**

交易系统门户、首页面，交易功能入口界面。

- **组件定义**

View	/index.jsp	
	/product/productIndex.jsp	
Action	ProductMappingDispatchAction	对应 index 方法
Service		
Dao		

- **序列图**



10.1.2 分类检索

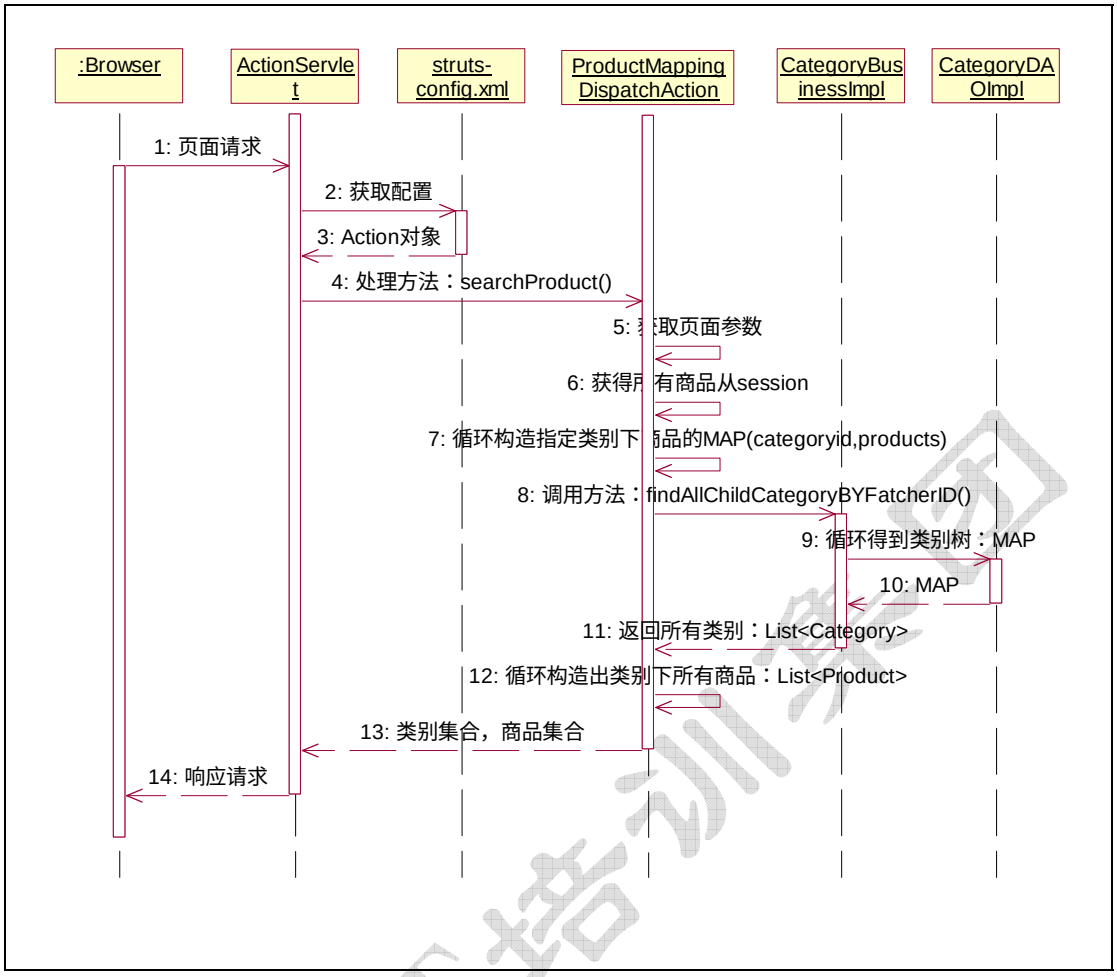
- **功能概述**

通过系统首页面进入，进行商品按类别分类减速。

- **组件定义**

View	/product/productIndex.jsp	
	/product/productSearch.jsp	
Action	ProductMappingDispatchAction	searchProduct 方法
Service	ProductBusinessImpl	商品管理业务处理
	CategoryBusinessImpl	商品类别管理业务处理
Dao	ProductDAOImpl	商品管理数据持久层操作
	CategoryBusinessImpl	商品类别数据持久层操作

- **序列图**



10.1.3 显示商品明细

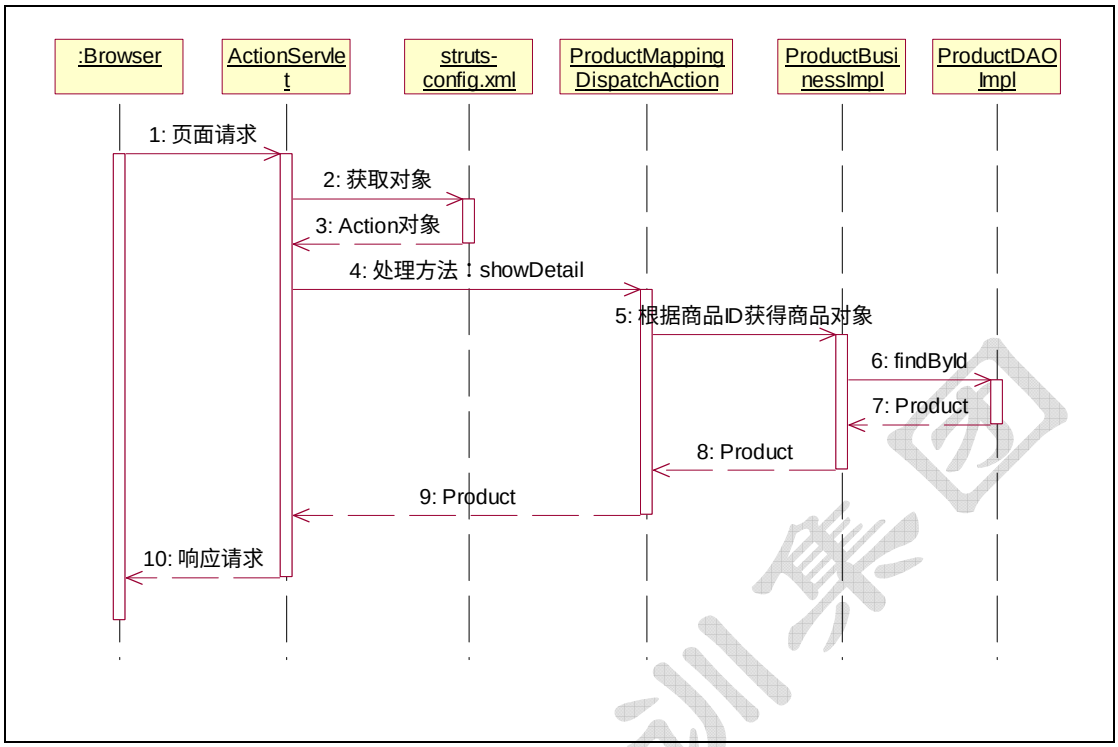
- **功能概述**

通过系统首页面、分类检索页面进入，查看商品详细信息。

- **组件定义**

View	/product/productIndex.jsp	
	/product/productSearch.jsp	
	/product/productDetail.jsp	
Action	ProductMappingDispatchAction	showDetail 方法
Service	ProductBusinessImpl	商品管理业务处理
Dao	ProductDAOImpl	商品管理数据持久层操作

➢ **序列图**



10.1.4 订单列表

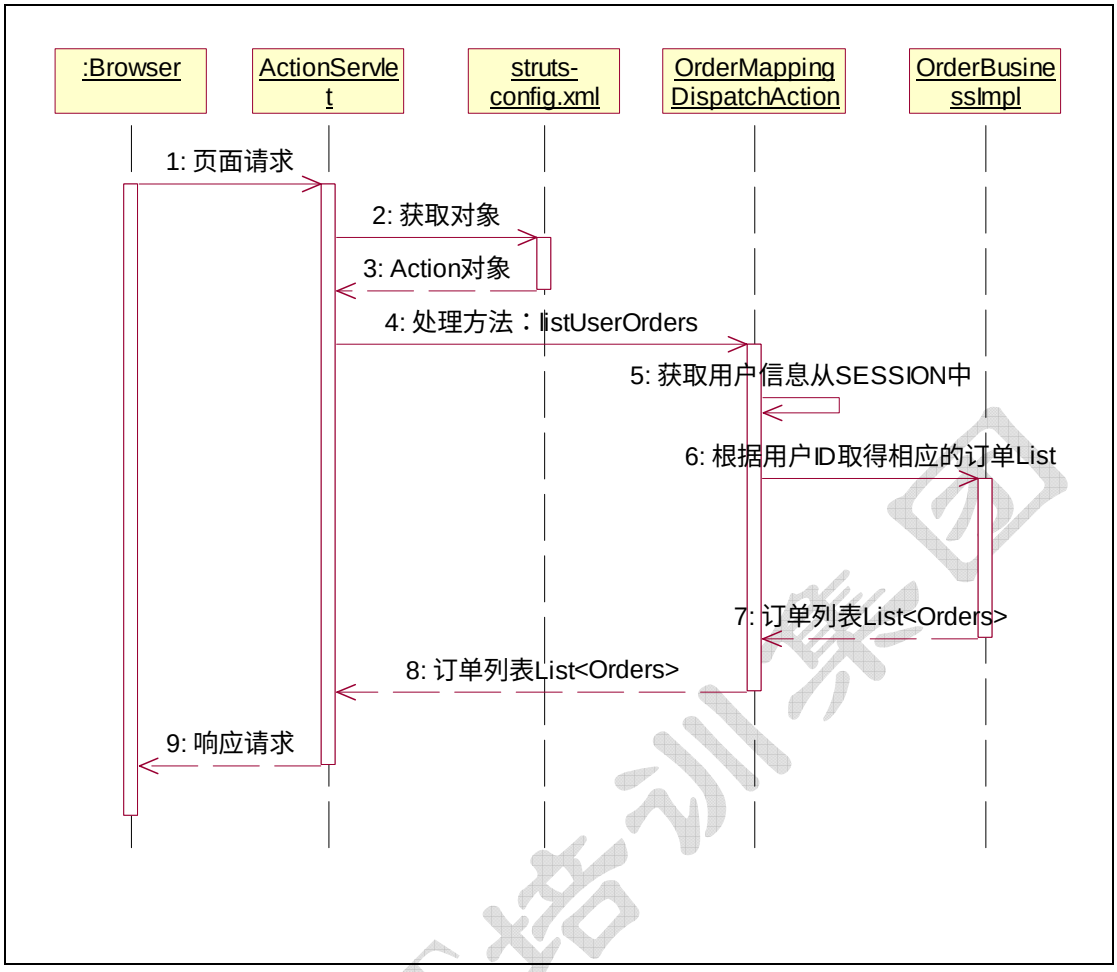
➢ **功能概述**

登陆用户，通过任意页面“我的当当”进入，默认显示当前交易用户的订单列表。

➢ **组件定义**

View	/product/productIndex.jsp	
	/inc/header.jsp	系统同一的也头
	/order/myorders.jsp	订单列表信息
Action	OrderMappingDispatchAction	listUserOrders 方法
Service	OrderBusinessImpl	订单管理业务类
Dao	OrdersDAOImpl	订单管理数据持久层操作

➢ **序列图**



10.1.5 删除订单

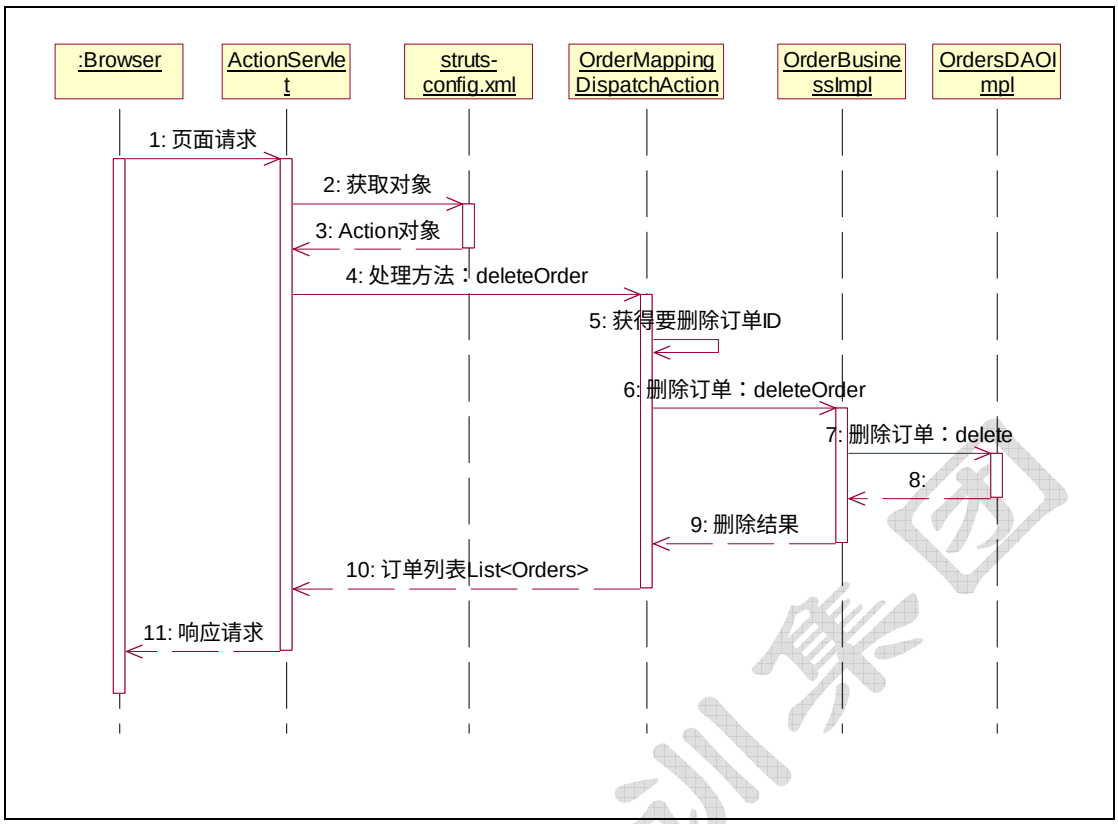
- **功能概述**

登陆用户，在“订单列表”页面删除指定的一个订单。

- **组件定义**

View	/order/myorders.jsp	
Action	OrderMappingDispatchAction	deleteOrder 方法
Service	OrderBusiness	订单管理业务类
Dao	OrdersDAOImpl	订单管理数据持久层操作

- **序列图**



10.1.6 添加商品到购物车

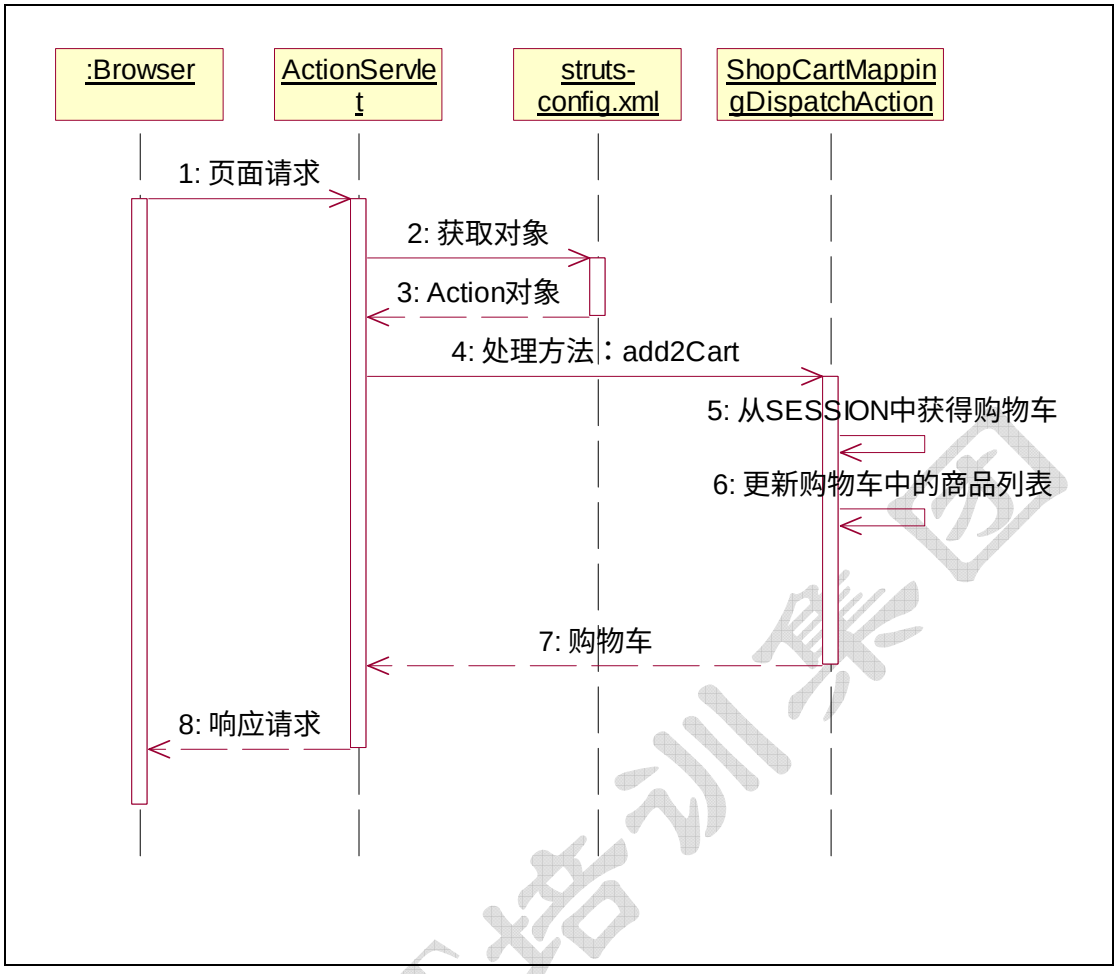
- **功能概述**

登陆用户，从商品明细画面，添加一个商品到购物车。

- **组件定义**

View	/product/productDetail.jsp	
	/shopcart/shoppingcart.jsp	
Action	ShopCartMappingDispatchAction	add2Cart 方法（数据保存在 Session 中）
Service		
Dao		

- **序列图**



10.1.7 在购物车中删除一个订单列表

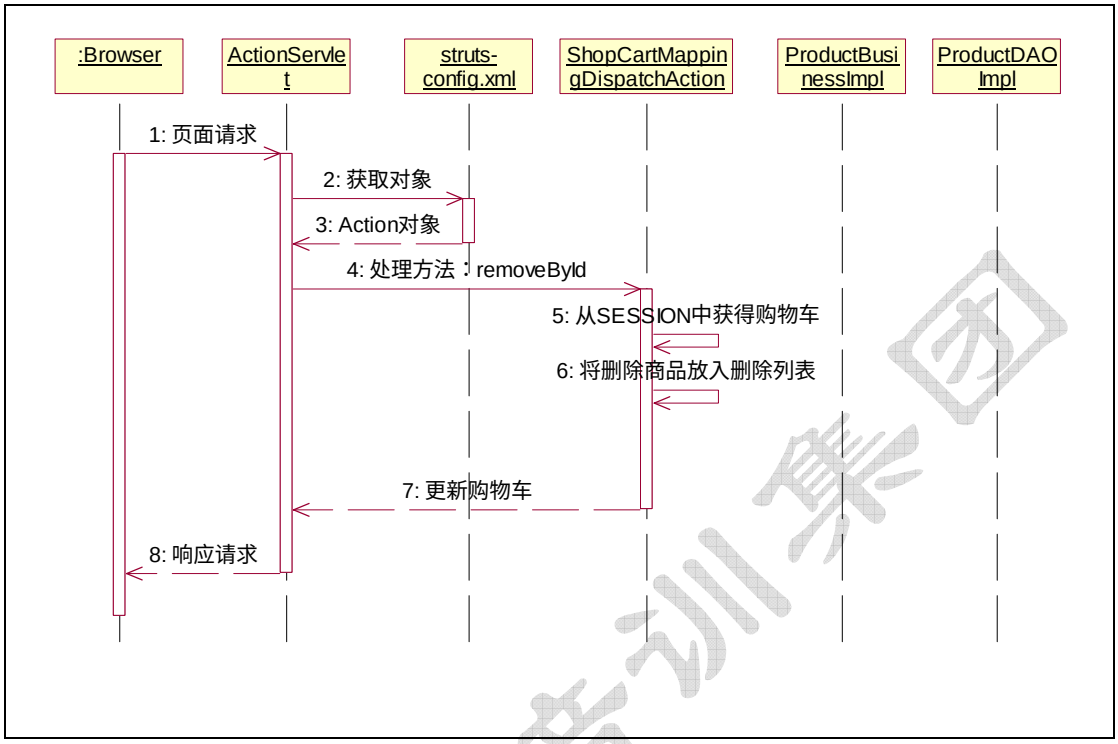
- **功能概述**

登陆用户，在购物车页面临时取消一个已经选购的商品。

- **组件定义**

View	/shopcart/shoppingcart.jsp	
Action	ShopCartMappingDispatchAction	removeById 方法
Service		
Dao		

➢ **序列图**



10.1.8 在购物车中恢复删除的订单列表

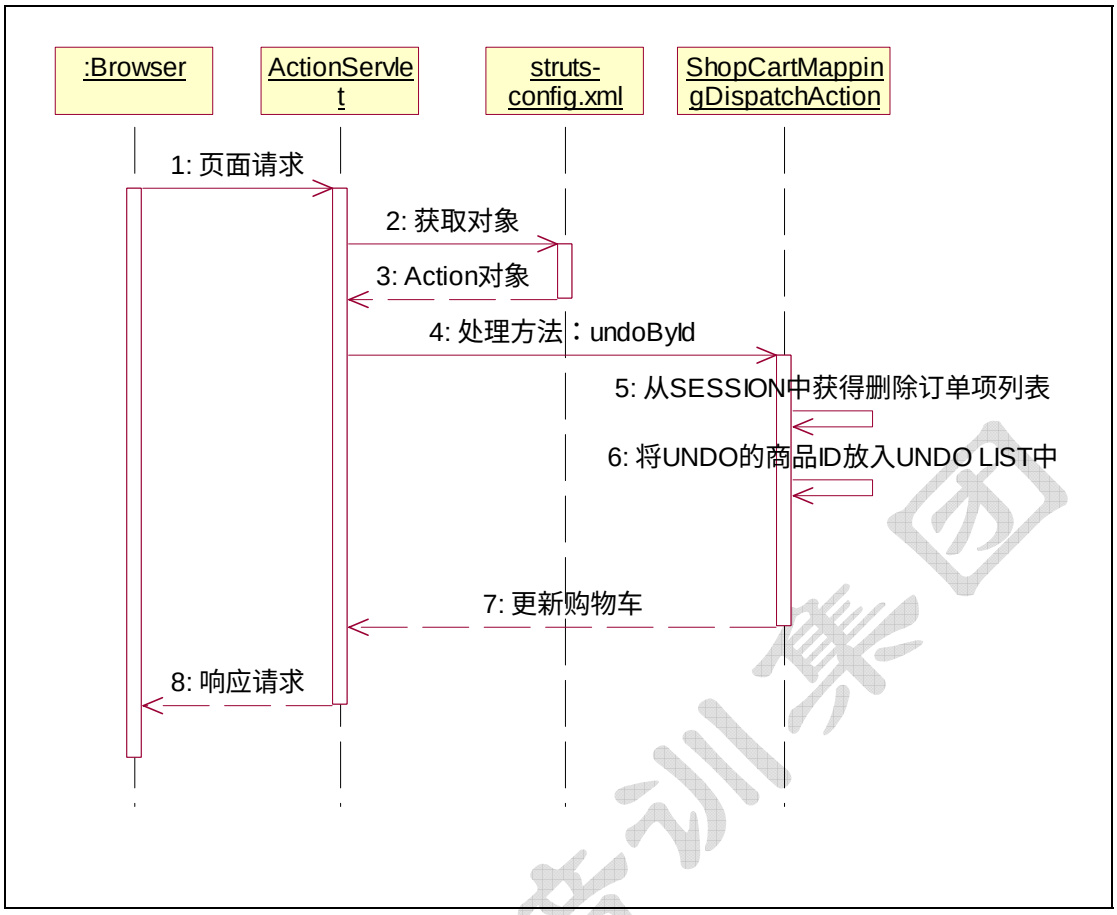
➢ **功能概述**

登陆用户，恢复购物车临时删除的商品。

➢ **组件定义**

View	/shopcart/shoppingcart.jsp	
Action	ShopCartMappingDispatchAction	undoById 方法
Service		
Dao		

➢ **序列图**



10.1.9 修改订单列表中商品数量

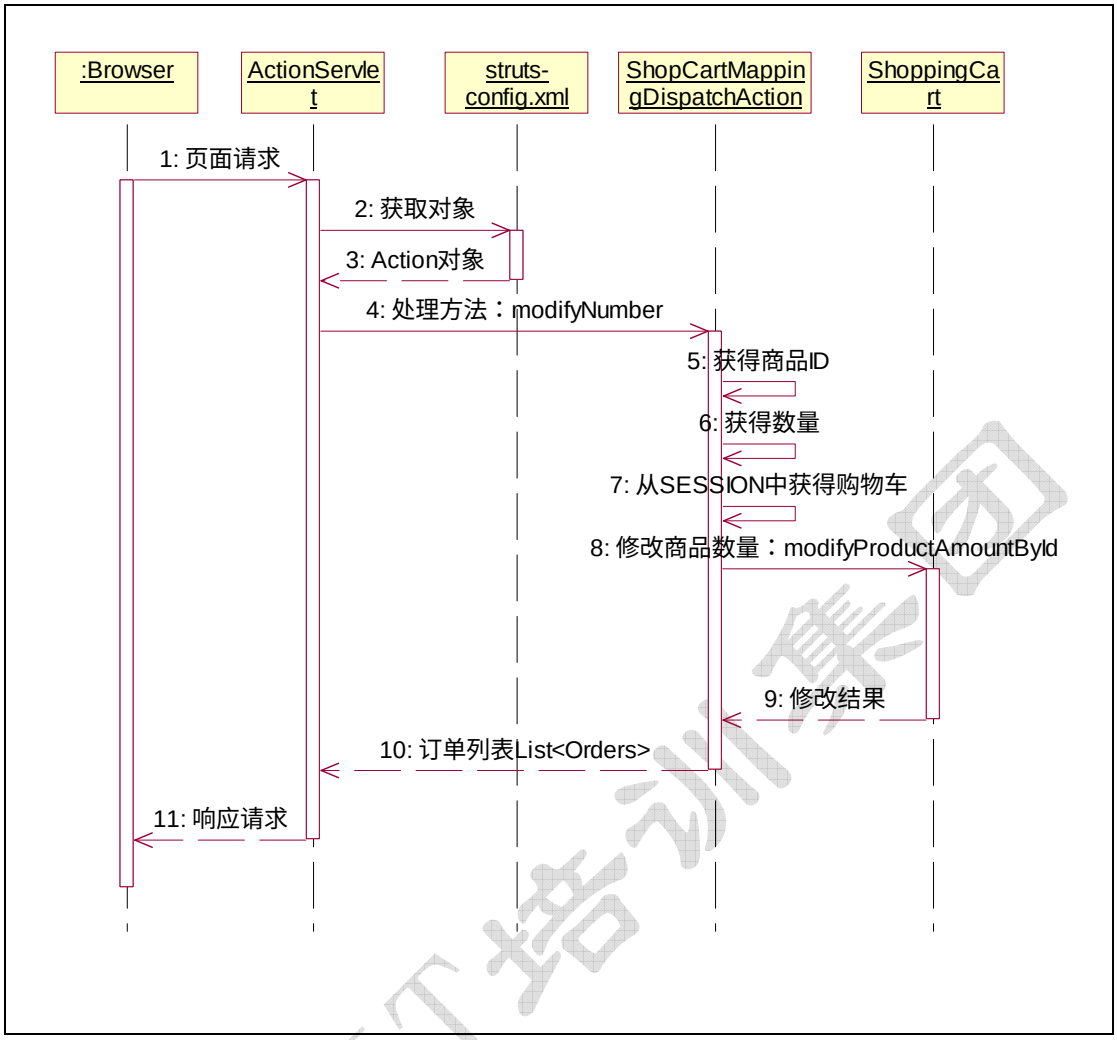
- **功能概述**

登陆用户，修改购物清单中，选购商品数量。

- **组件定义**

View	/shopcart/shoppingcart.jsp	
Action	ShopCartMappingDispatchAction	modifyNumber 方法
Service	ShoppingCart	购物车
Dao		

- **序列图**



10.1.10 购物车结算

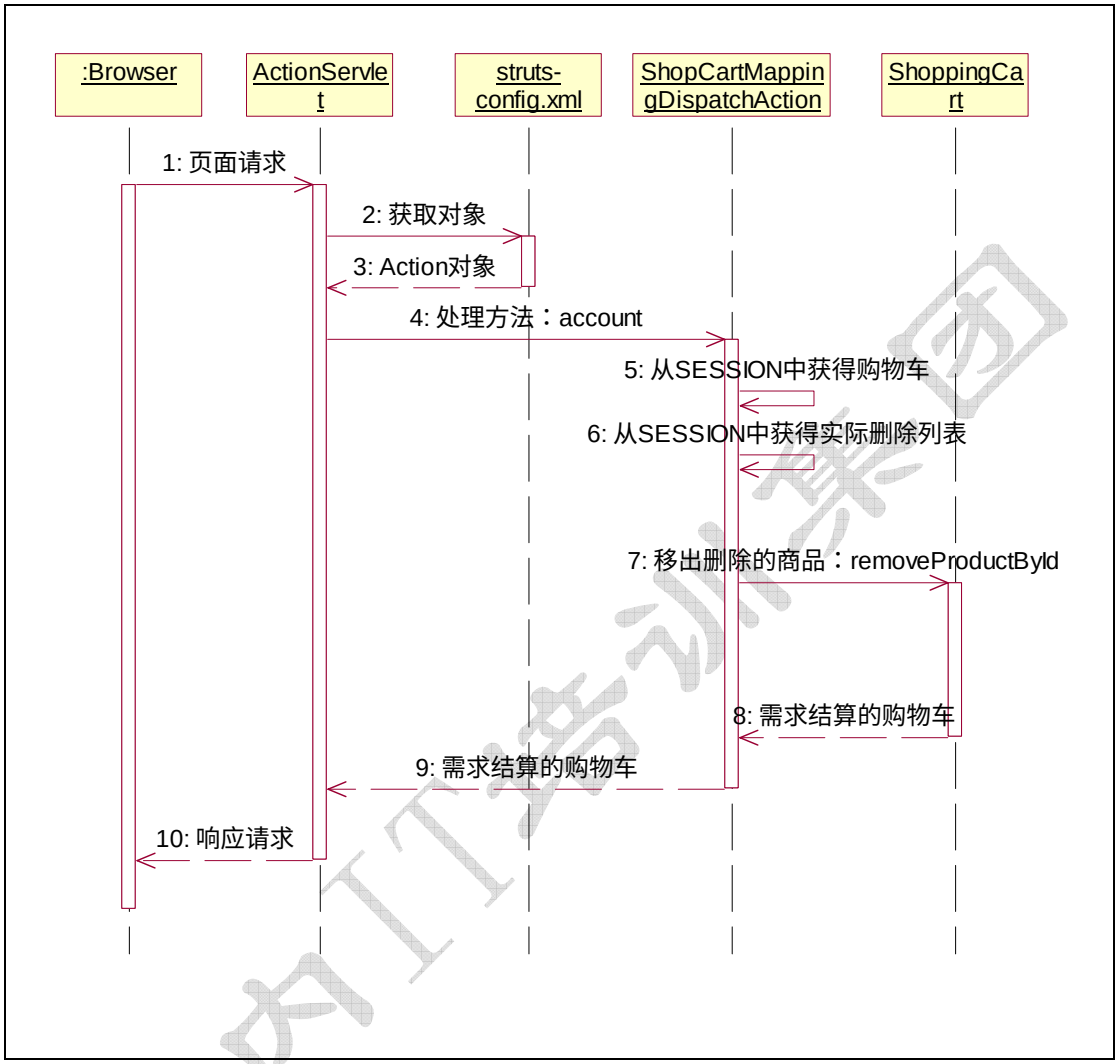
- **功能概述**

登陆用户，商品选购完成，结算购物车中商品，进入订单确认页面。

- **组件定义**

View	/shopcart/shoppingcart.jsp	
	/shopcart/checkout.jsp	
Action	ShopCartMappingDispatchAction	Account 方法
Service	ShoppingCart	购物车管理类
Dao		

➢ **序列图**



10.1.11 订单确认

➢ **功能概述**

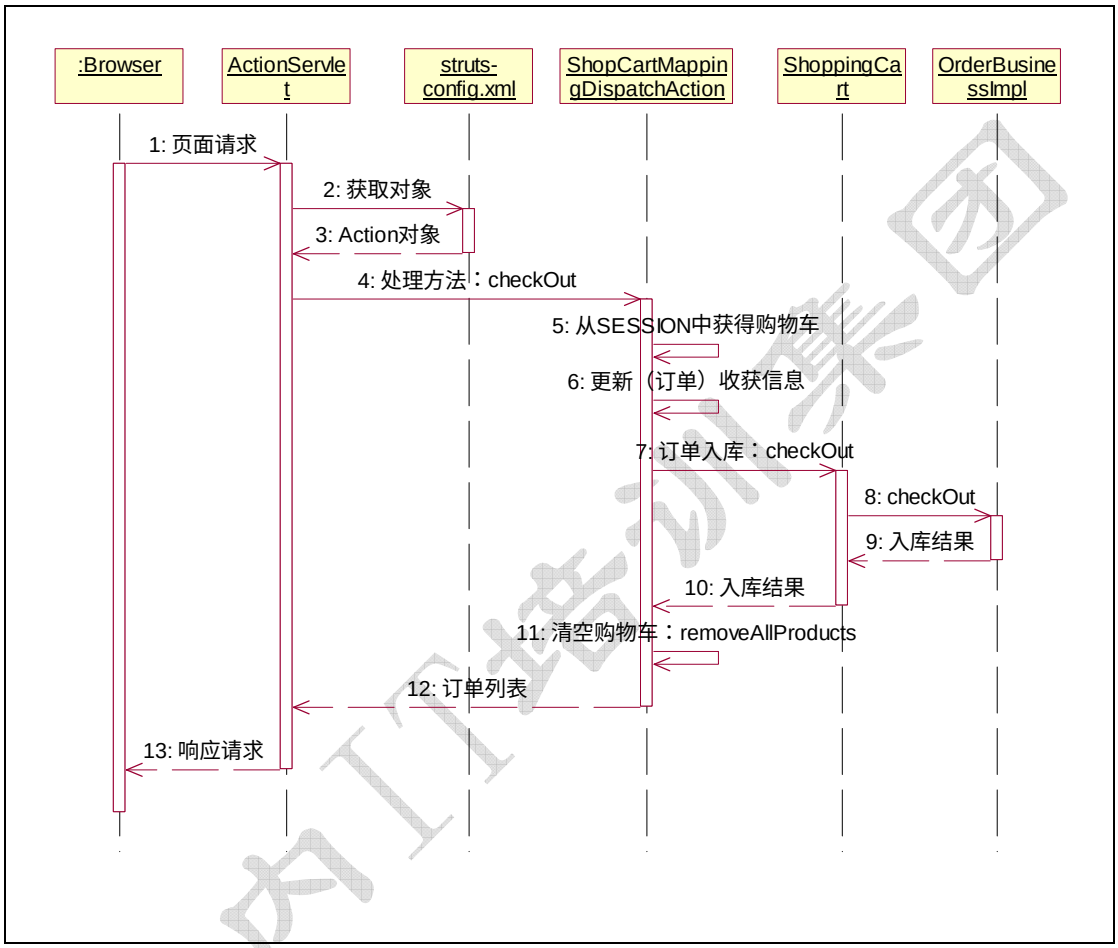
登陆用户，订单确认页面（可以修改收获信息）。

➢ **组件定义**

View	/shopcart/checkout.jsp	
	/order/myorders.jsp	
Action	ShopCartMappingDispatchAction	checkOut 方法

Service	OrderBusinessImpl	订单管理实现类
	ShoppingCart	购物车管理类
Dao	OrdersDAOImpl	提供订单相关数据库操作

- **序列图**



10.1.12 清空购物车

- **功能概述**

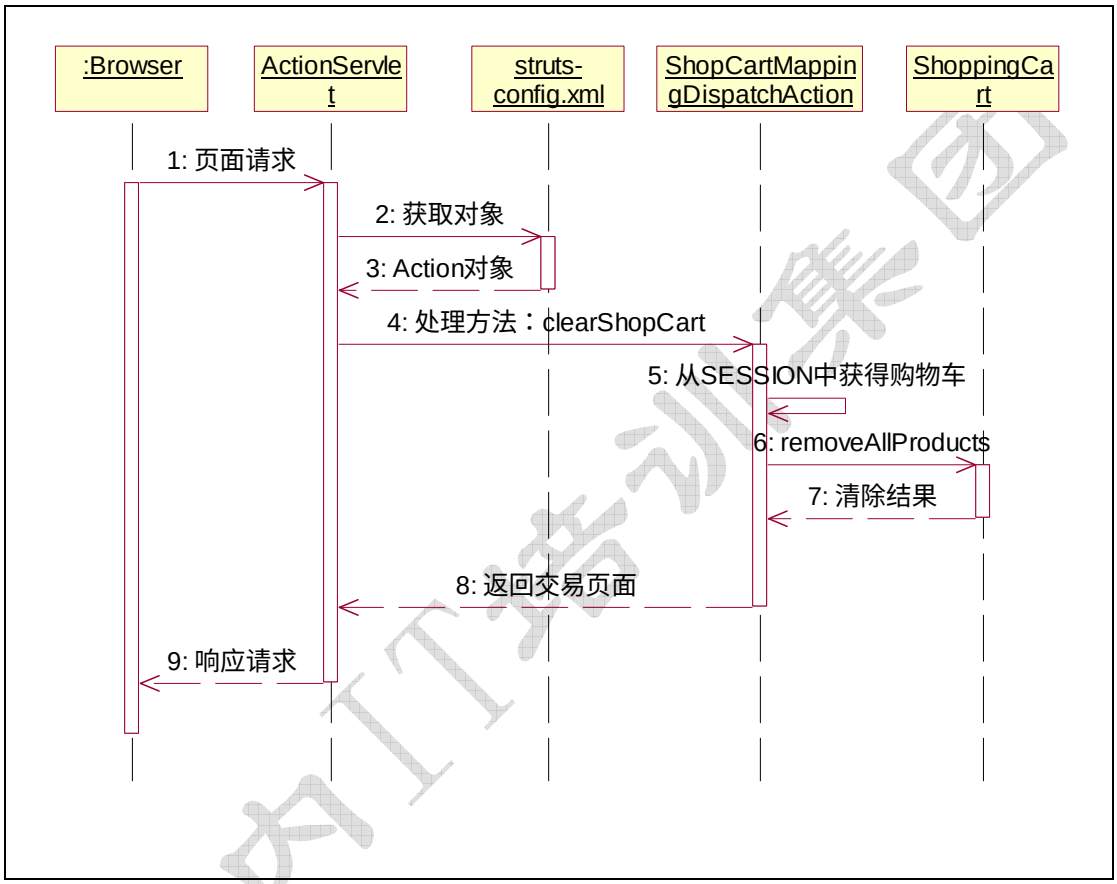
登陆用户，清空购物车中选购的所有商品。

- **组件定义**

View	/shopcart/shoppingcart.jsp	
	/product/productIndex.jsp	
Action	ShopCartMappingDispatchAction	clearShopCart 方法

Service	ShoppingCart	购物车管理类
Dao		

- **序列图**



10.1.13 用户登入

- **功能概述**

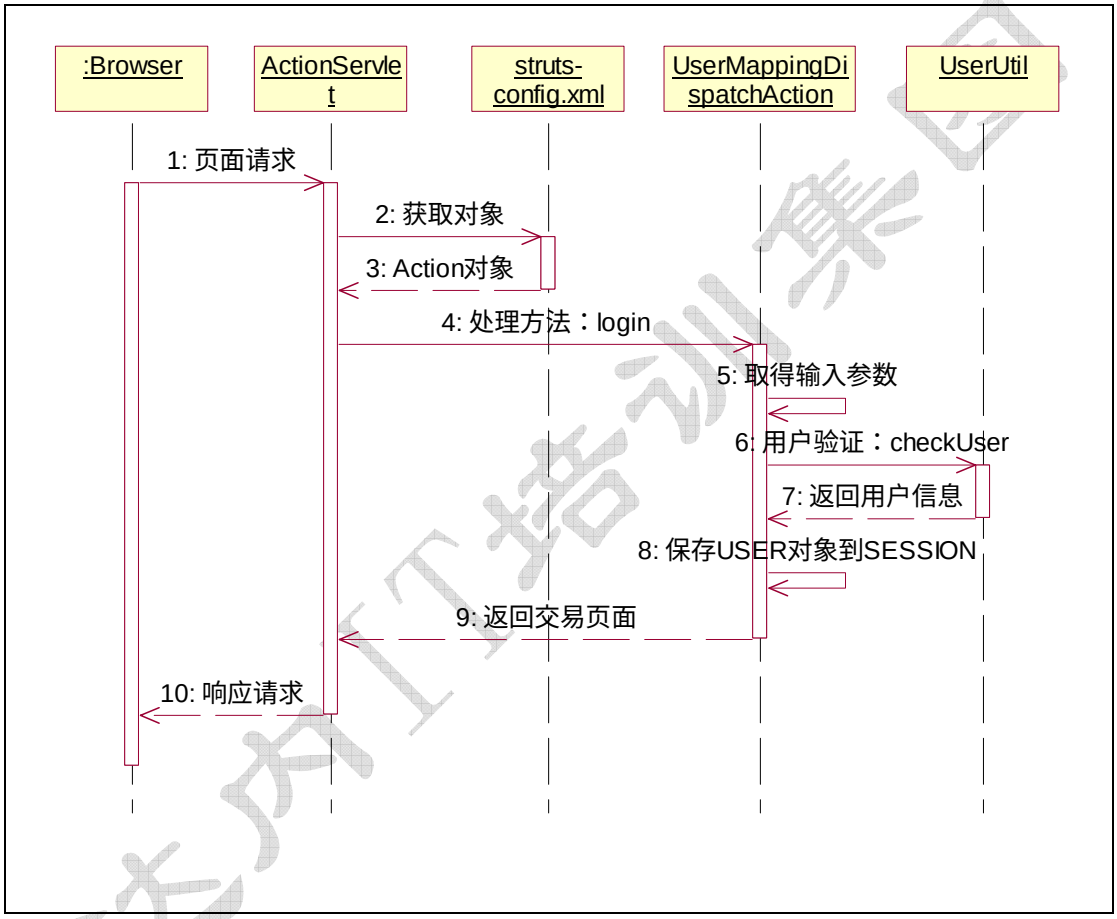
注册用户登陆。

- **组件定义**

View	/product/productIndex.jsp	
	/user/signin.jsp	

Action	UserMappingDispatchAction	Login 方法
Service	UserUtil	checkUser 用户登陆工具类
Dao		

➢ **序列图**



10.1.14 用户登出

➢ **功能概述**

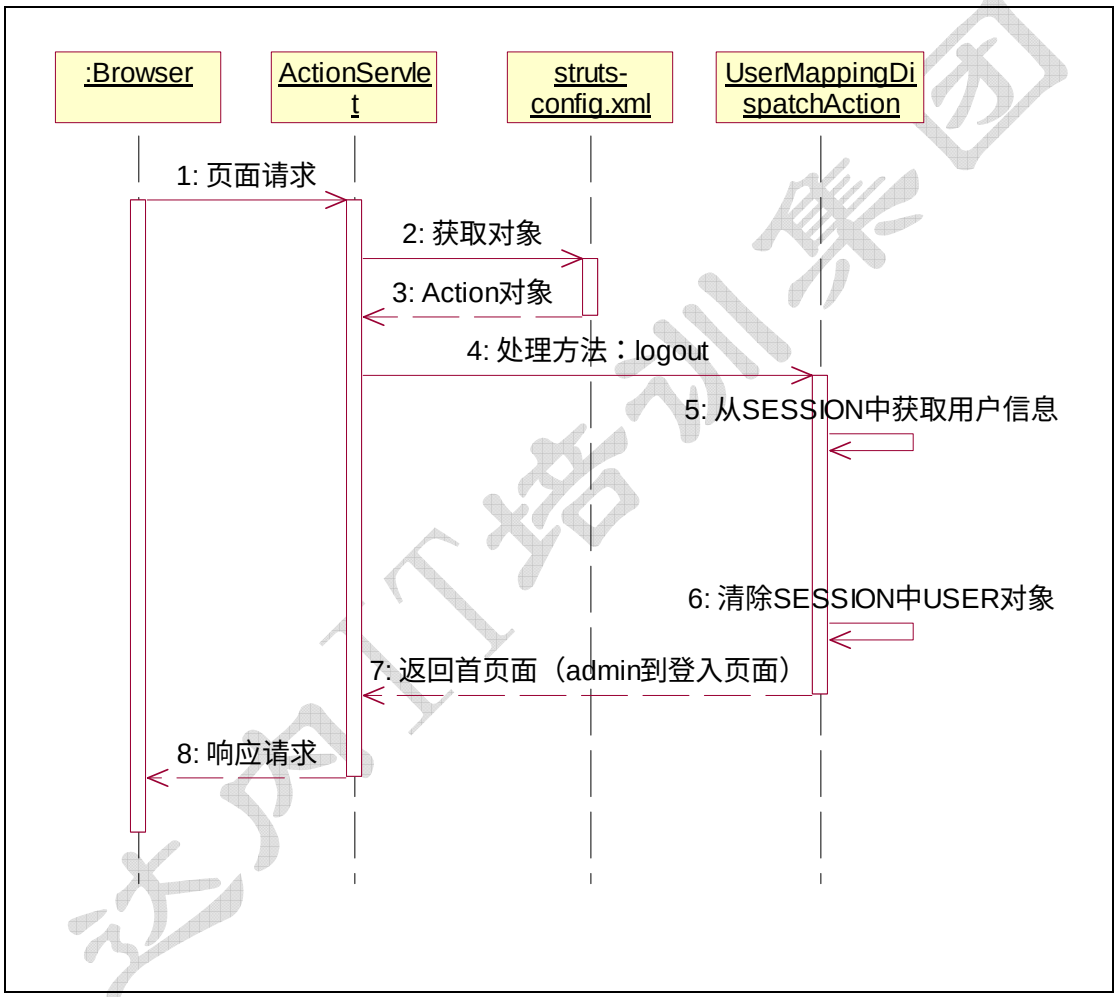
已经登入用户登出系统。

➢ **组件定义**

View	/product/productIndex.jsp	
	/inc/header.jsp	

Action	UserMappingDispatchAction	Logout 方法
Service		
Dao		

- **序列图**



10.1.15 用户注册

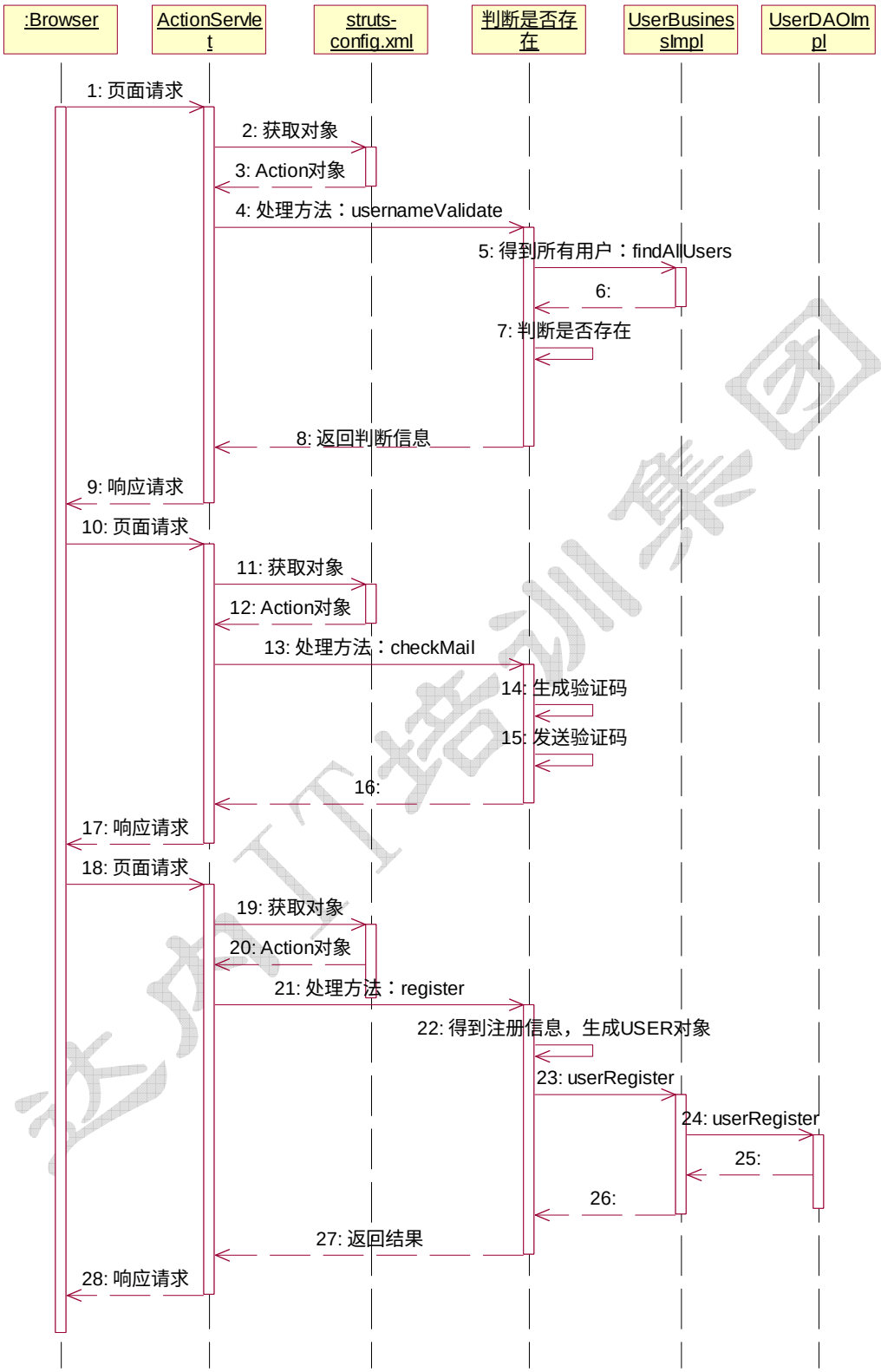
- **功能概述**

未注册用户，注册成为系统交易用户。

- **组件定义**

View	/product/productIndex.jsp	
	/inc/header.jsp	
	/user/register.jsp	
	/user/registed.jsp	
Action	UserMappingDispatchAction	toRegister 方法
	UserMappingDispatchAction	Register 方法
	UserMappingDispatchAction	usernameValidate 方法验证（用户是否存在）
Service	UserBusinessImpl	用户管理实现类
Dao	UserDAOImpl	用户管理的数据操作

- **序列图**


:Browser
ActionServlet
struts-config.xml
判断是否存在
UserBusinesslmpl
UserDAOlmpl
1: 页面请求
2: 获取对象
3: Action对象
4: 处理方法：usernameValidate
5: 得到所有用户：findAllUsers
6:
7: 判断是否存在
8: 返回判断信息
9: 响应请求
10: 页面请求
11: 获取对象
12: Action对象
13: 处理方法：checkMail
14: 生成验证码
15: 发送验证码
16:
17: 响应请求
18: 页面请求
19: 获取对象
20: Action对象
21: 处理方法：register
22: 得到注册信息，生成USER对象
23: userRegister
24: userRegister
25:
26:
27: 返回结果
28: 响应请求

10.1.16 编辑个人基本信息

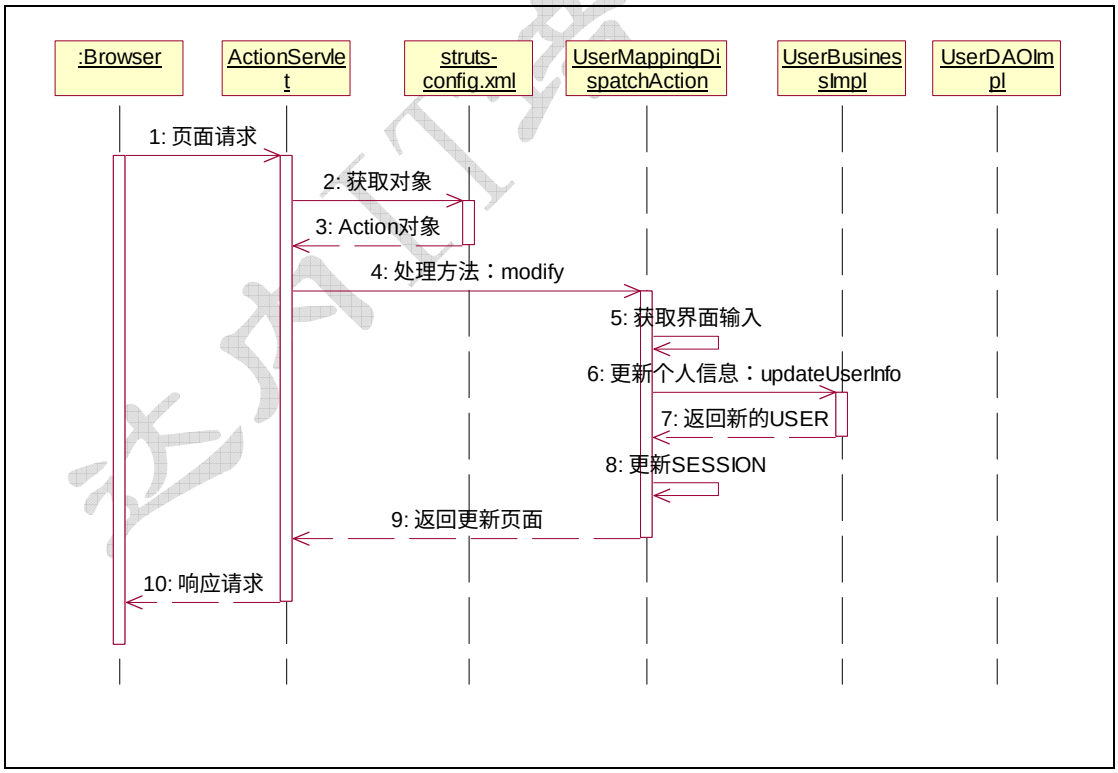
➢ **功能概述**

登陆用户，修改个人基本信息。

➢ **组件定义**

View	/user/myarchives.jsp	
Action	UserMappingDispatchAction	Modify 方法
Service	UserBusinessImpl	用户管理实现类
Dao	UserDAOImpl	用户管理的数据操作

➢ **序列图**



10.1.17 修改 Email

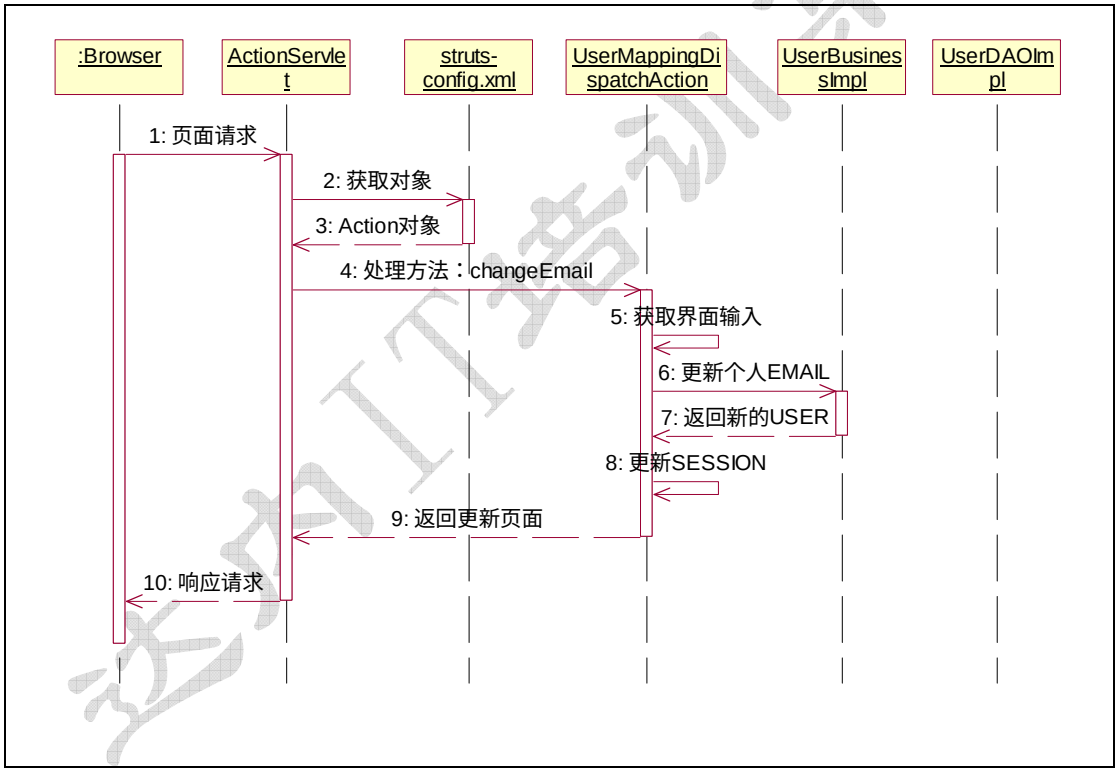
➢ **功能概述**

登陆用户，修改用户 Email 信息。

- **组件定义**

View	/user/myemail.jsp	
Action		
Service	UserMappingDispatchAction	changeEmail 方法
Dao	UserBusinessImpl	用户管理实现类

- **序列图**



10.1.18 修改密码

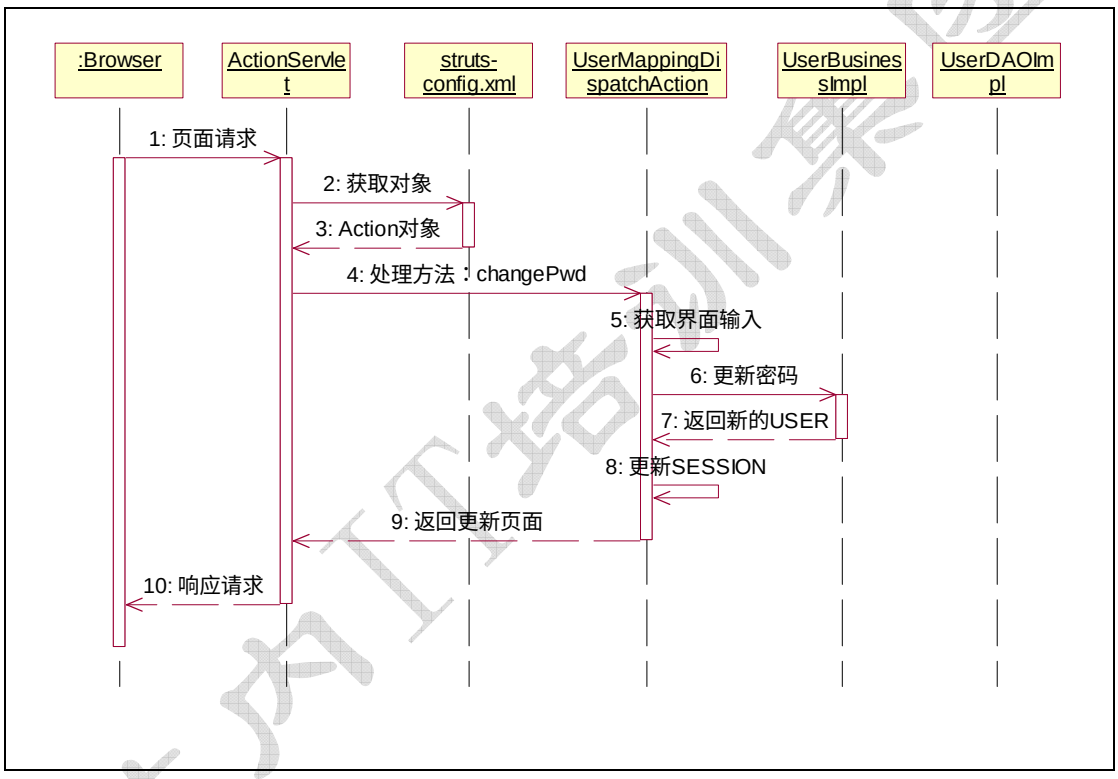
- **功能概述**

登陆用户，修改个人登陆密码。

- **组件定义**

View	/user/passwordchange.jsp	
Action		
Service	UserMappingDispatchAction	changePwd 方法
Dao	UserBusinessImpl	用户管理实现类

➢ **序列图**



10.2 管理系统

10.2.1 商品列表

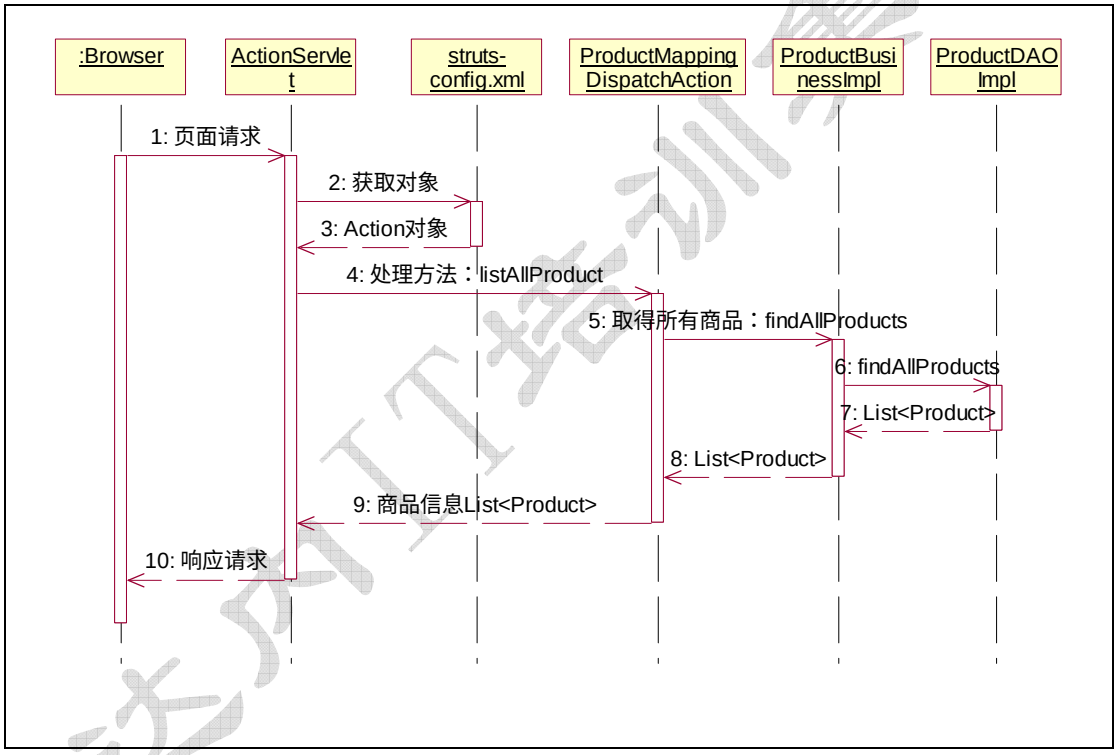
➢ **功能概述**

管理员登入后，进入商品列表，删除和修改商品。

➢ **组件定义**

View	/product/productList.jsp	
Action	ProductMappingDispatchAction	listAllProduct 方法
Service	ProductBusinessImpl	商品管理类
Dao	ProductDAOImpl	商品管理的数据操作

➢ **序列图**



10.2.2 添加商品

➢ **功能概述**

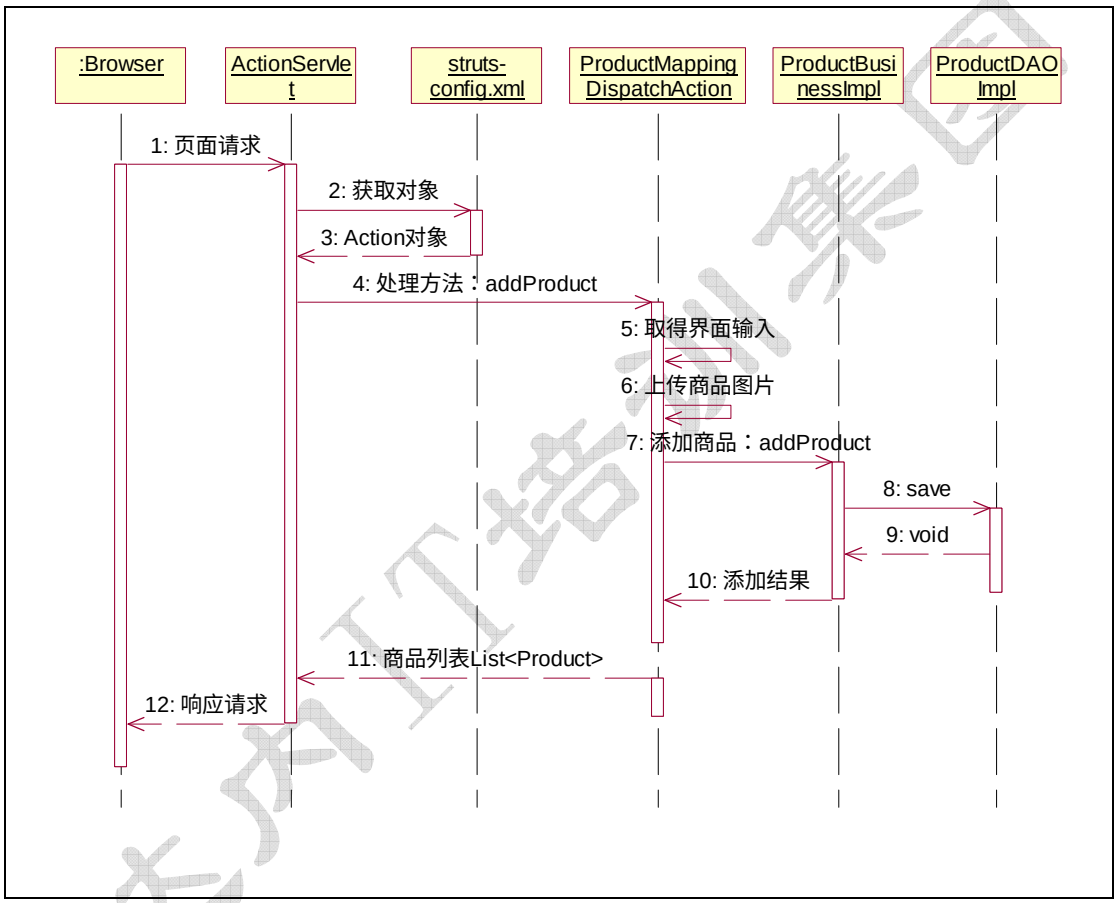
管理员登入后，添加发布商品。

➢ **组件定义**

View	/product/addProduct.jsp	

Action	ProductMappingDispatchAction	addProduct 方法
Service	ProductBusinessImpl	商品管理类
	CategoryBusinessImpl	商品类别管理类
Dao	ProductDAOImpl	商品管理的数据操作
	CategoryDAOImpl	商品类别管理的数据操作

- **序列图**



10.2.3 修改商品

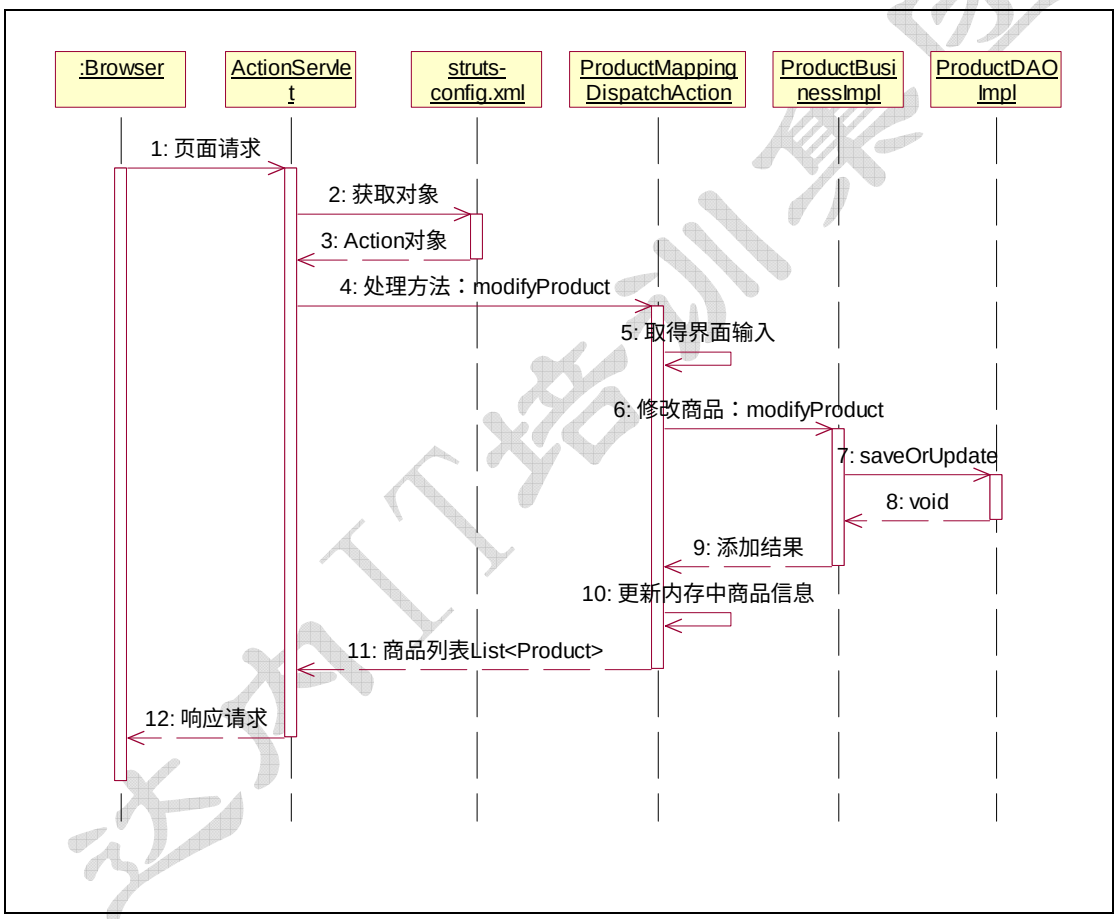
- **功能概述**

管理员登入后，修改商品信息。

- **组件定义**

View	/product/productList.jsp	
	/product/productModify.jsp	
Action	ProductMappingDispatchAction	toModifyProduct 方法
	ProductMappingDispatchAction	modifyProduct 方法
Service	ProductBusinessImpl	商品管理实现类
Dao	ProductDAOImpl	商品管理的 DAO 操作

- **序列图**



10.2.4 删除商品

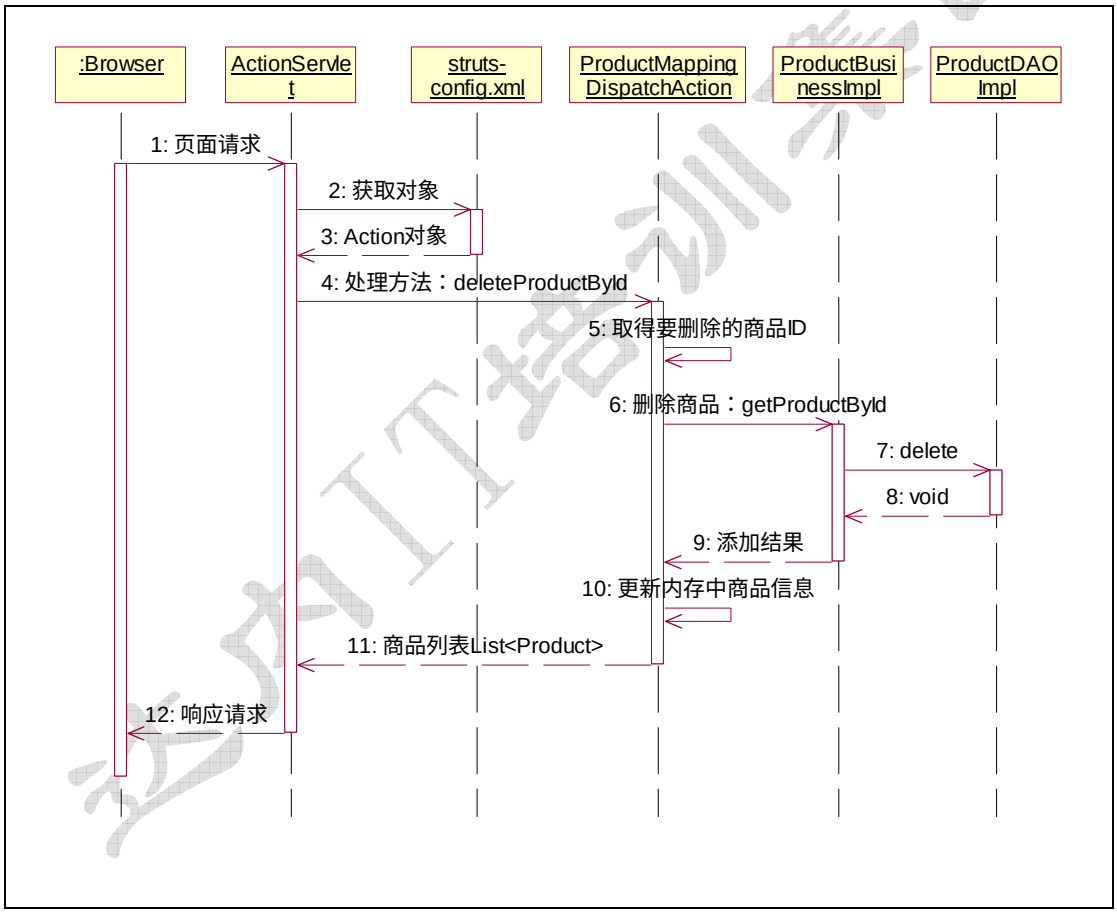
- **功能概述**

管理员登入后，删除指定商品。

➢ **组件定义**

View	/product/productList.jsp	
Action	ProductMappingDispatchAction	deleteProductById 方法
Service	ProductBusinessImpl	商品管理实现类
Dao	ProductDAOImpl	商品管理的 DAO 操作

➢ **序列图**



10.2.5 管理员登入

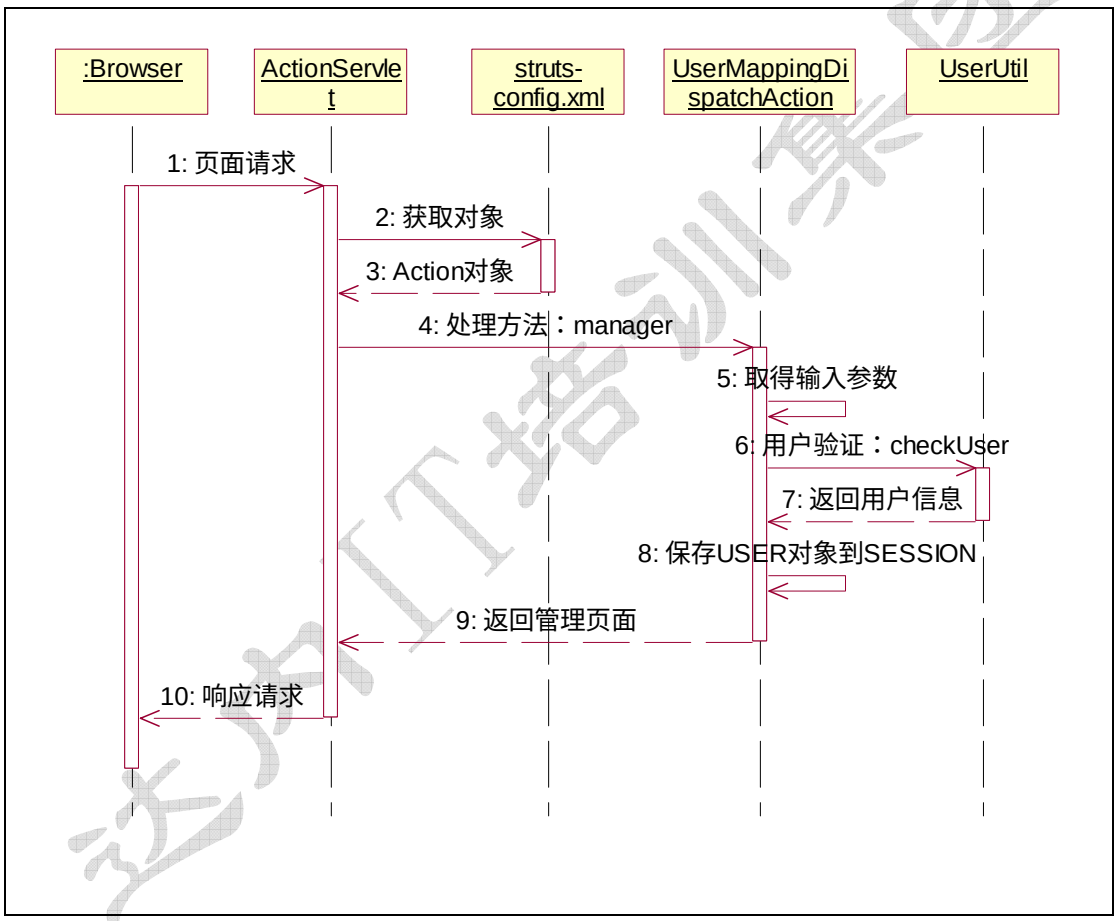
➢ **功能概述**

管理用户，登入功能。

➢ **组件定义**

View	/manager.jsp	
	/user/userList.jsp	
Action	UserMappingDispatchAction	Manager 方法
Service		
Dao		

➢ **序列图**



10.2.6 管理员登出

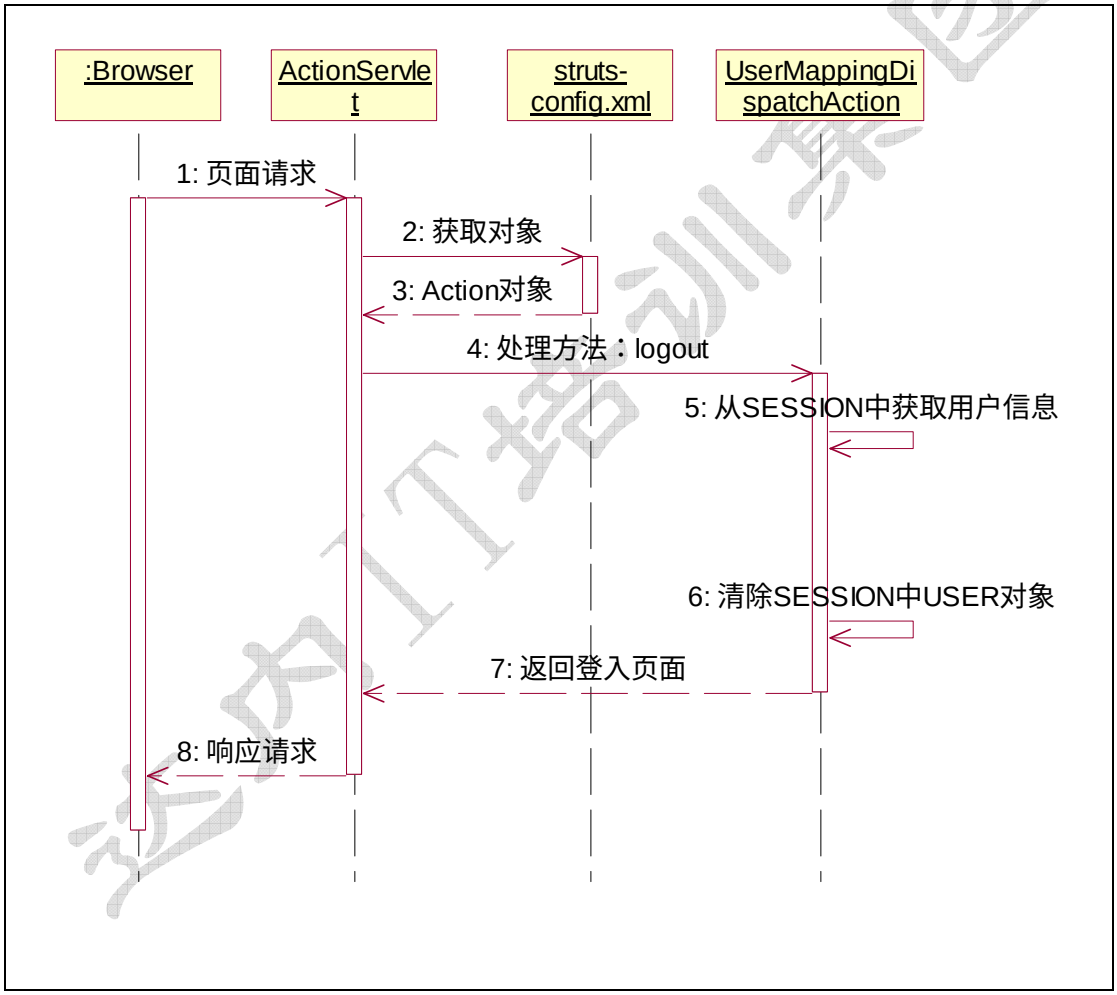
➢ **功能概述**

管理员登出。

➢ **组件定义**

View	/manager.jsp	
Action	UserMappingDispatchAction	logout 方法
Service		
Dao		

➢ **序列图**



10.2.7 用户列表

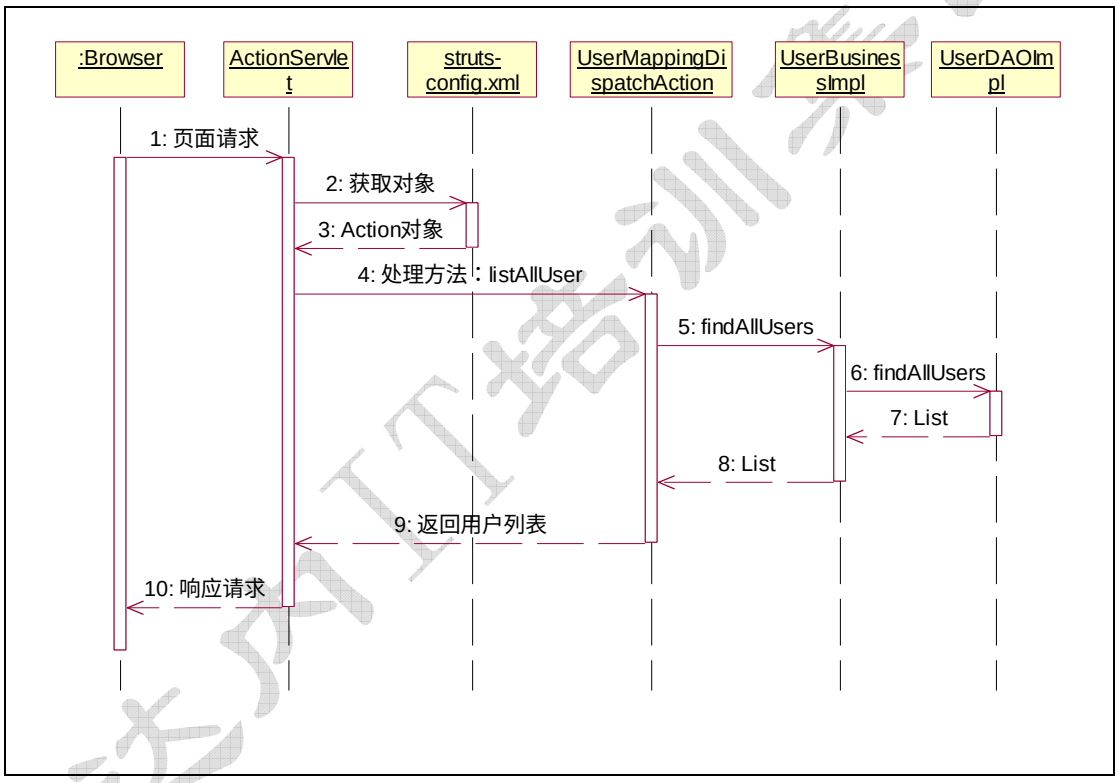
➢ **功能概述**

管理员用户登入后，显示用户列表功能。

➢ **组件定义**

View	/manager.jsp	
	/user/userList.jsp	
Action	UserMappingDispatchAction	listAllUser 方法
Service	UserBusinessImpl	用户管理业务实现类
Dao	UserDAOImpl	用户管理数据操作类

➢ **序列图**



10.2.8 删除用户

➢ **功能概述**

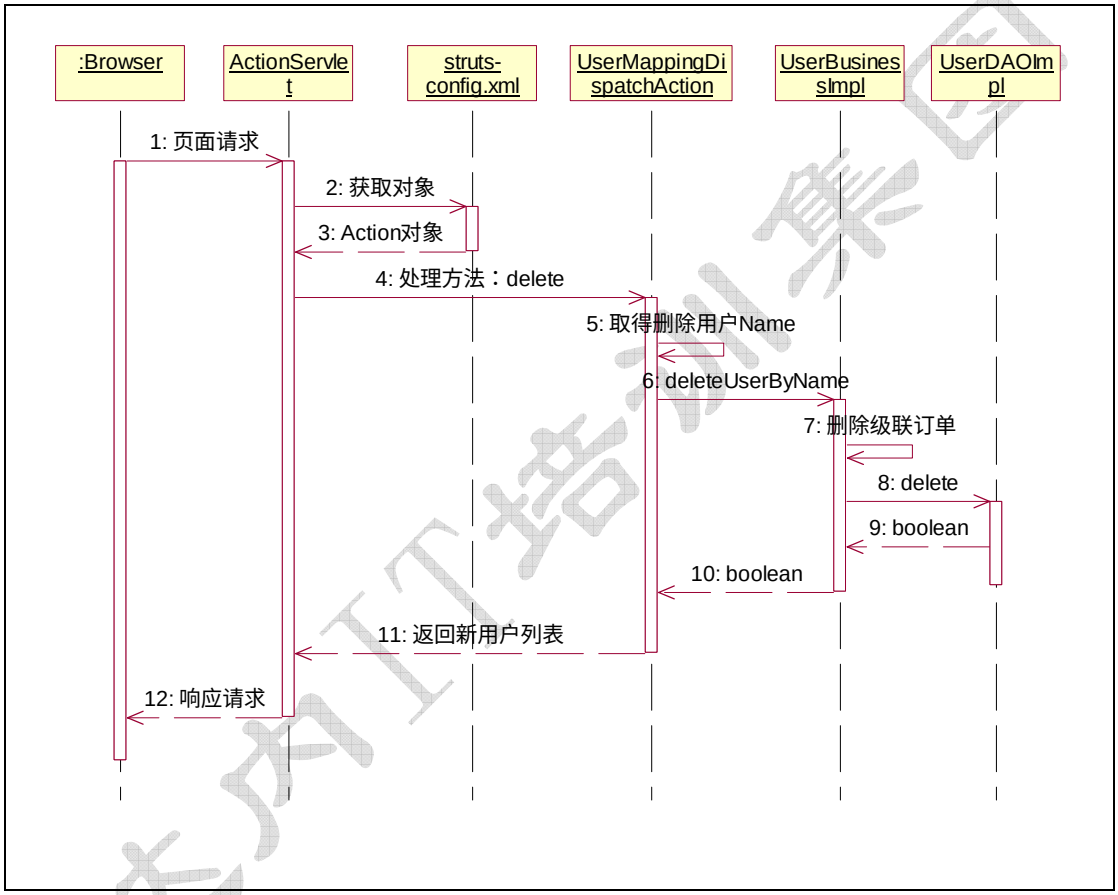
管理员用户登入后，删除列表中用户（admin 不可以删除）。

➢ **组件定义**

View	/user/userList.jsp	

Action	UserMappingDispatchAction	Delete 方法
Service	UserBusinessImpl	用户管理业务实现类
Dao	UserDAOImpl	用户管理数据操作类

- **序列图**



----------完----------