

# 《软件开发方法课程设计》总报告

项目名称	基于 Web 的电子商城系统设计与实现
学 号	XXXXXXXX
姓 名	XXXXXX
专 业	计算机科学与技术
任课教师	XXXXXX
同组名单	XXXXXX
日 期	2019. 09. 05

## 一、项目需求分析

### 1. 用户功能分析

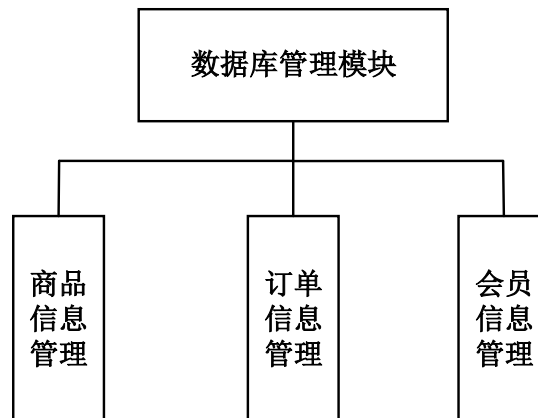
- (1) 在线注册。顾客首先要注册成为电子商城会员。注册事填写登录名、密码、联系方式（电子信箱、电话号码等等）。注册之后可以继续完善个人信息比如填写详细收货地址、修改密码、实名认证、查询及修改订单等。
- (2) 添加商品到购物车。顾客浏览网上商城将自己需要的商品放到购物车中，可连续添加商品。
- (3) 管理购物车。用户在选择完商品之后可在购物车页面进行修改购买商品数量，删除某个商品，清空购物车。
- (4) 提交订单。用户确定购买之后即可提交订单，顾客完善各项收货信息之后，显示让顾客确定，并提示是否记住相关收货信息便于以后使用。顾客还可在电子商城查询订单信息，可以修改、取消订单。
- (5) 付款。顾客在订单被销售方确定之后，选择付款方式支付，收货之后有确认收货功能。

### 2. 管理功能分析

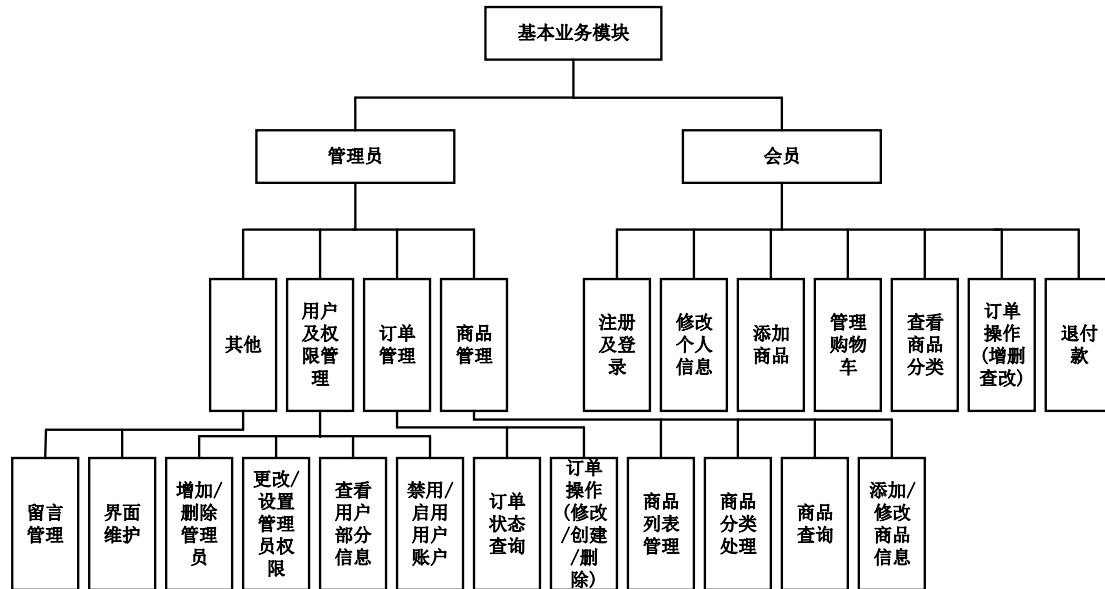
- (1) 用户和权限管理。网站后台管理员可以对注册的会员用户进行账户注销、查看部分信息操作。更改或设置管理员权限，增加/删除管理员。
- (2) 商品管理。商品管理员可以对商城的商品进行登记，比如价格、厂家、名称等录入数据库系统。
- (3) 订单管理。订单管理员应该对每日的订单数量进行统计，以及查阅订单的详细信息，每个订单的处理情况，删除，修改订单。
- (4) 页面维护及留言处理。对商城商品页面的广告，商品的呈现的布局等进行维护。还有对会员的留言进行回复处理。

### 3. 满足以上系统功能的模块可分为如下几个部分：

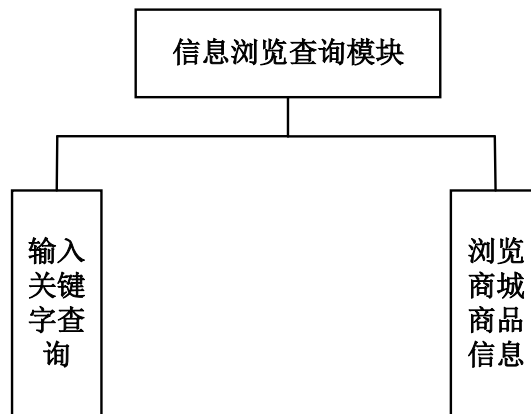
- (1) 数据库管理模块。主要包括注册的会员用户的信息、商品详细信息、订单信息的存储和管理。



- (2) 基本业务模块。主要包括用户方面：注册登录，修改个人信息，添加商品至购物车，管理购物车，查看商品分类，订单操作，退付款；管理员方面：商品管理（商品列表管理，商品分类处理，商品查询，添加/修改商品信息），订单管理（订单状态查询，订单修改/创建/删除），用户及权限管理（查看用户部分信息，启用或禁用用户账户，增加/删除管理员，更改或设置管理员权限），界面维护及留言处理）



(3) 信息浏览查询模块。主要包括用户通过输入关键字查询满足要求的商品，浏览网页商品详情。



## 一、系统分析与设计(UML)

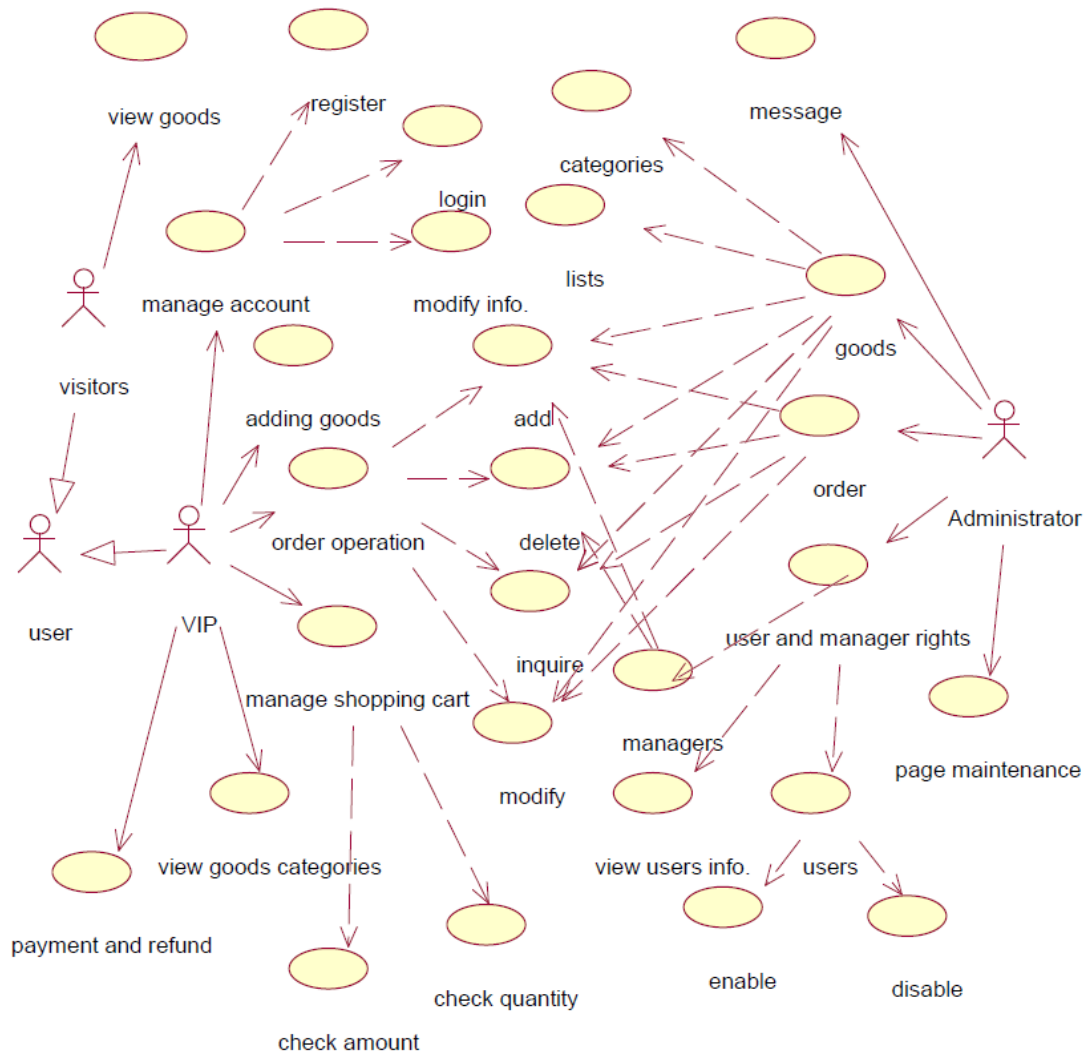
### 1. 用例图

在电子商城系统中，主要的参与者是用户和管理员。

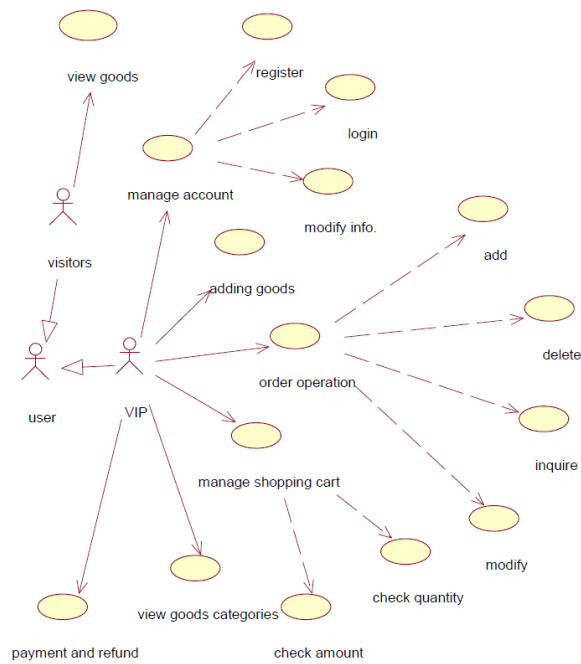
用户分为普通用户（游客）和会员用户，前者指只在本商城商品页面进行浏览，后者指注册了本商城账号的用户，登录账户后即可进行购买商品。

管理员主要是对订单（order），商品（goods），用户（user），普通管理员（manager）进行管理。

#### 1.1 总用例图



## 1.2 用户参与者用例图

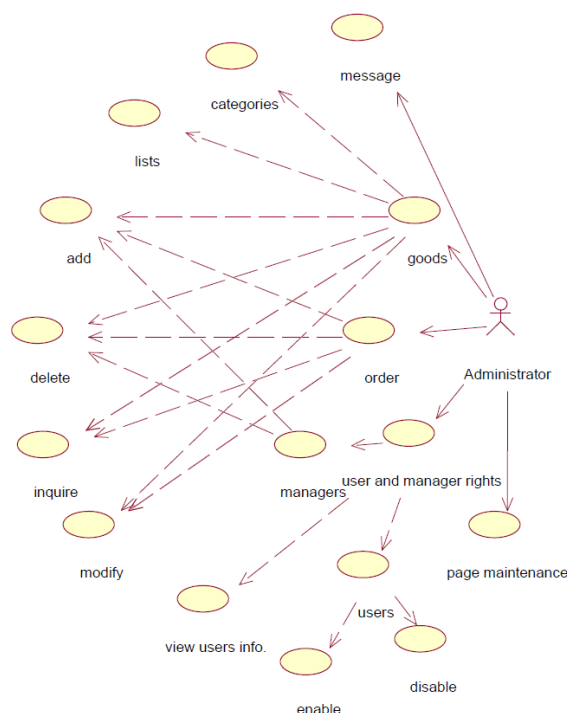


【用例说明】（虚线箭头原型是 include）

VIP 参与者，visitors 参与者：VIP 会员用户和 visitors 都是从 user 泛化而来。

- (1) view goods 用例：游客可以浏览电子商城的商品。
- (2) manage account 用例：用户可以用来管理账户，其中包括注册、登录和修改个人信息。
- (3) register 用例：用户注册商城账户，包括设置账户名、密码、联系方式（邮箱 or 电话号码）。
- (4) login 用例：用户输入账号密码登录账户。
- (5) modify info. 用例：修改个人信息，包括密码、收货地址、联系方式、实名认证等。
- (6) adding goods 用例：向购物车中添加商品。
- (7) order operation 用例：订单操作包括添加订单，删除（取消）订单，查询订单，修改订单操作。
- (8) add 用例：添加订单。
- (9) delete 用例：删除（取消订单）。
- (10) inquire 用例：查询订单状态。
- (11) modify 用例：修改订单信息。包括收货地址，联系方式等信息。
- (12) manage shopping cart 用例：管理购物车，包括修改购买商品数量和查看总购买金额。
- (13) check quantity 用例：核对商品数量，可以增加或者减少购买商品数量。
- (14) check amount 用例：核对金额，查看购买总金额。
- (15) view goods categories 用例：用户可以查看商品分类。
- (16) payment and refund 用例：用户支付和退款功能。用户在只有货款支付成功卖家才发货，也可以在付款之后向商家反馈原因提交退款申请，待商家同意之后退款回用户支付账户。

### 1.3 管理员参与者用例图



【用例说明】(虚线箭头原型是 include)

- (1) Administrator 参与者: 电子商城后台管理人员, 对商品信息、普通管理员权限、用户信息、订单信息进行实时掌控和管理。
- (2) goods 用例: 管理员对商品进行管理, 包括对商品分类, 查看商品列表, 添加商品, 删除商品, 查询商品, 修改商品信息。
- (3) order 用例: 管理员对商品订单进行处理, 包括创建, 删除, 修改, 查询订单操作。
- (4) user and manager rights 用例: 管理员对用户和普通管理员权限进行管理。
- (5) users 用例: 对用户的账户进行控制。
- (6) enable 用例: 启用用户账户。
- (7) disable 用例: 禁用用户账户。
- (8) view user info. 用例: 查看用户的部分详细信息。
- (9) managers 用例: 修改和设置普通管理员权限。包括增加管理员和删除管理员。
- (10) add 用例: 管理员创建订单, 添加管理员, 添加商品信息等。
- (11) delete 用例: 管理员因用户完成订单或取消订单而删除订单记录, 删除管理员, 删除过期商品及信息。
- (12) inquire 用例: 管理员查询已有订单状态, 商品信息。
- (13) modify 用例: 用户修改订单之后管理员同步修改订单信息, 管理员修改商品的部分信息等。

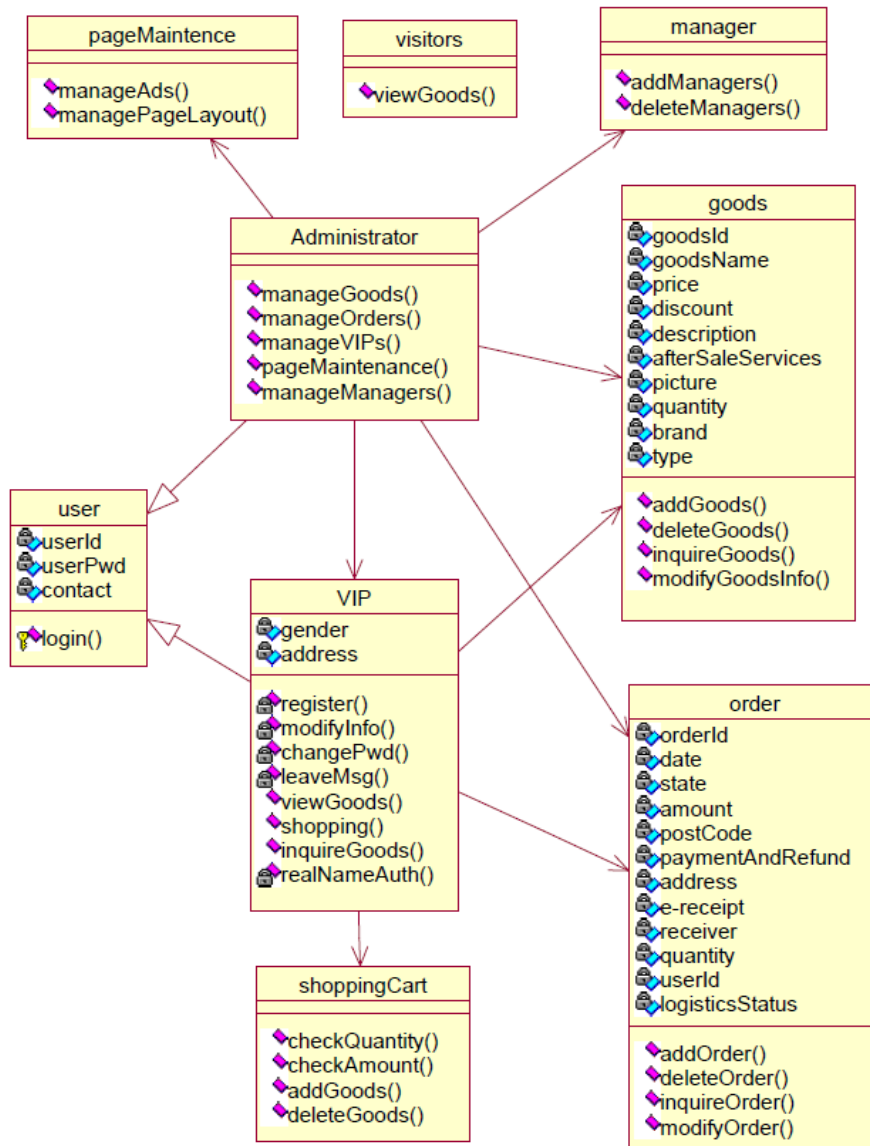
## 2. 类图

### 2.1 系统类分析

整个电子商城系统中, 应该有如下类:

User 类, visitors 类(游客类), VIP 类(会员类), goods 类(商品类), order 类(订单类), shoppingcart (购物车类), Administrator 类(后台管理员类), manager 类(普通管理员类), pageMaintenance (页面维护类)

### 2.2 系统类图



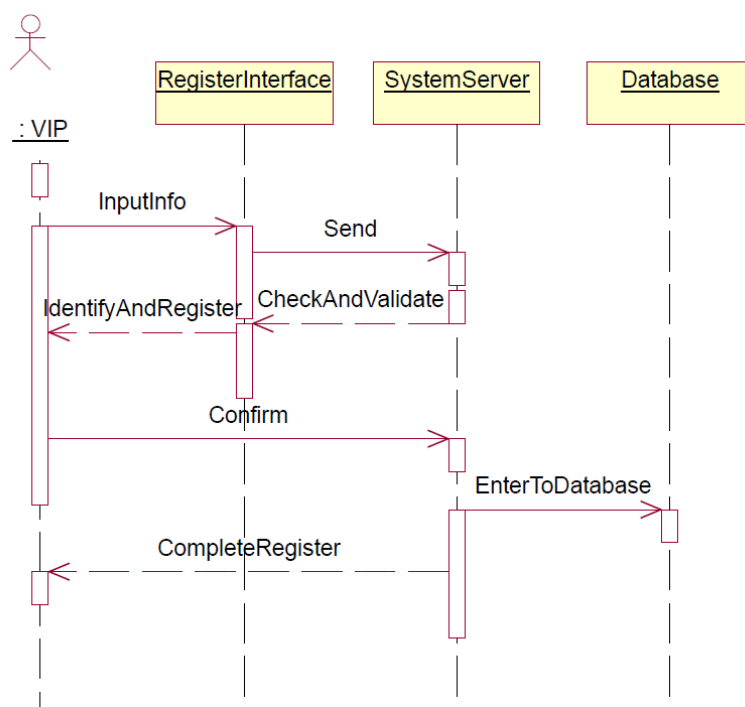
### 【类图说明】

- (1) User 类：所有的类的父类，包括属性 userId（用户登录账户名/账号）、userPwd（用户登录密码）、contact（用户联系方式），方法 login（用户登录操作）。
- (2) visitors 类：游客类，有 viewGoods 方法（浏览商品）。
- (3) VIP(user\_info)类：会员类，除了继承了 user 类的属性和方法之外，还有属性 gender（性别）、address（收货地址），方法 register（注册账户）、modifyInfo（修改资料）、changePwd（修改登录密码）、leaveMsg（给商家留言）、viewGoods（浏览商品）、inquireGoods（查询商品）、realNameAuth（实名认证）。
- (4) Administrator 类：管理员类，除了继承 user 类的属性和方法之外，还有方法 manageGoods（管理商品）、manageOrders（管理订单）、manageVIPs（管理 VIP 会员顾客）、manageManagers（管理普通管理员）、pageMaintenance（页面维护）。
- (5) manager 类：普通管理员类，有方法 addManagers（添加管理员）、deleteManagers（删除管理员）。
- (6) shoppingCart 类：购物车类，有方法 checkQuantity（核对商品数量）、checkAmount

- (核对金额)、addGoods (添加商品), deleteGoods (删除商品)。
- (7) product\_info 类: 商品类, 有属性 goodsId (商品编号)、goodsName (商品名称)、price (商品价格)、discount (商品折扣)、description (商品文字描述介绍)、afterSaleServices (售后服务)、picture (商品图片描述)、quantity (商品数量库存)、brand (商品品牌)、type (商品类型), 方法 addGoods (商品管理员添加商品), deleteGoods (商品管理员删除商品)、inquireGoods (商品管理员查询商品)、modifyGoodsInfo (商品管理员修改商品信息)。
- (8) order\_info 类: 订单类, 有属性 orderId (订单号)、date (下单日期)、state (订单处理状态)、amount (订单金额)、quantity (商品数量)、userId (购物会员账号)、receiver (收货人)、postCode (邮政编码)、paymentAndRefund (退付款)、address (收货地址)、e-receipt (电子发票)、logisticsStatus (物流状态)。
- (9) pageMaintenance 类: 页面维护类, 有方法 manageAds (管理网站页面广告)、managePageLayout (管理网站页面布局)。

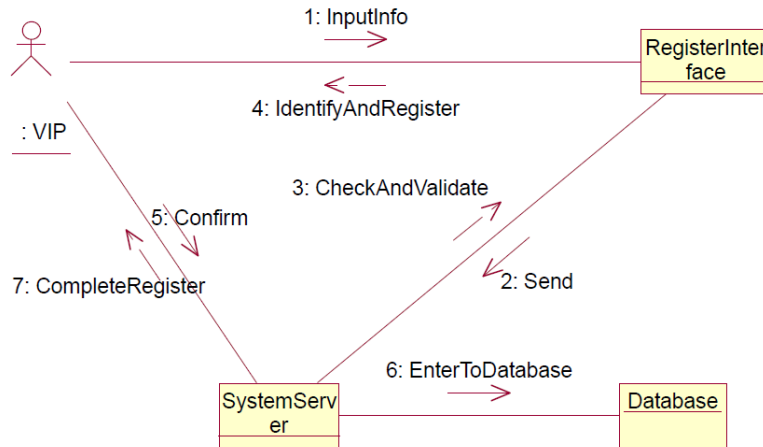
### 3. 顺序图及协作图

#### 3.1.1 用户注册顺序图



#### 3.1.2 用户注册协作图





### 【顺序图及协作图说明】

参与者：VIP 会员

RegisterInterface：注册界面；

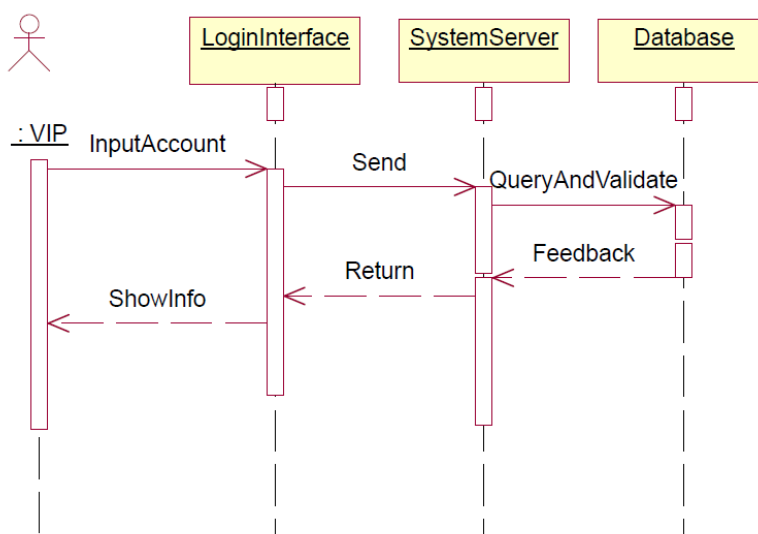
SystemServer：系统服务器；

Administrator：系统管理员；

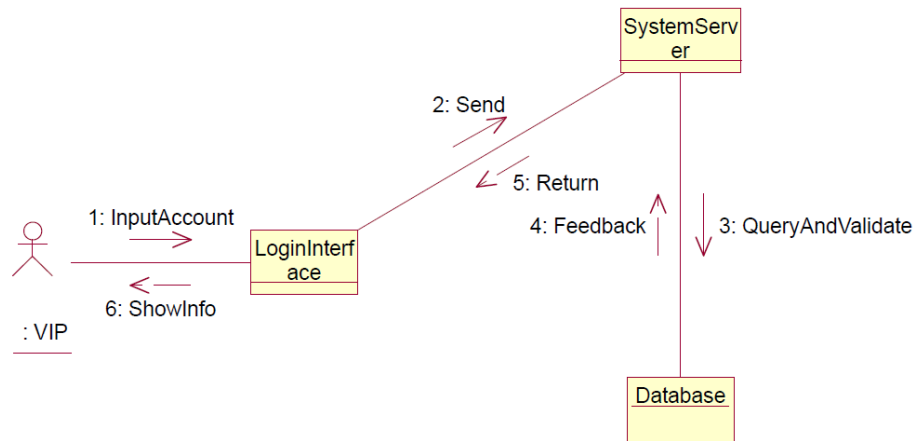
Database：系统数据库；

- ① InputInfo：用户在注册界面输入个人信息；包括账户名，密码，绑定手机号，常用收货地址等重要必备信息。
- ② Send：将填好的信息发送到服务器；
- ③ CheckAndValidate：系统服务器进行验证和审核资料；
- ④ IdentityAndRegister：注册界面提示是否进行确认注册；
- ⑤ Confirm：用户确认注册账户，将信息发送给服务器；
- ⑥ EnterToDatabase：将用户信息录入数据库；
- ⑦ CompleteRegister：系统提示用户完成注册。

### 3. 2. 1 用户登录顺序图



### 3. 2. 2 用户登录协作图



### 【顺序图及协作图说明】

参与者：VIP 会员；

LoginInterface：登录界面；

SystemServer：系统服务器；

Database：系统数据库；

① InputAccount：输入账户及密码；

② Send：将输入的信息发送给服务器；

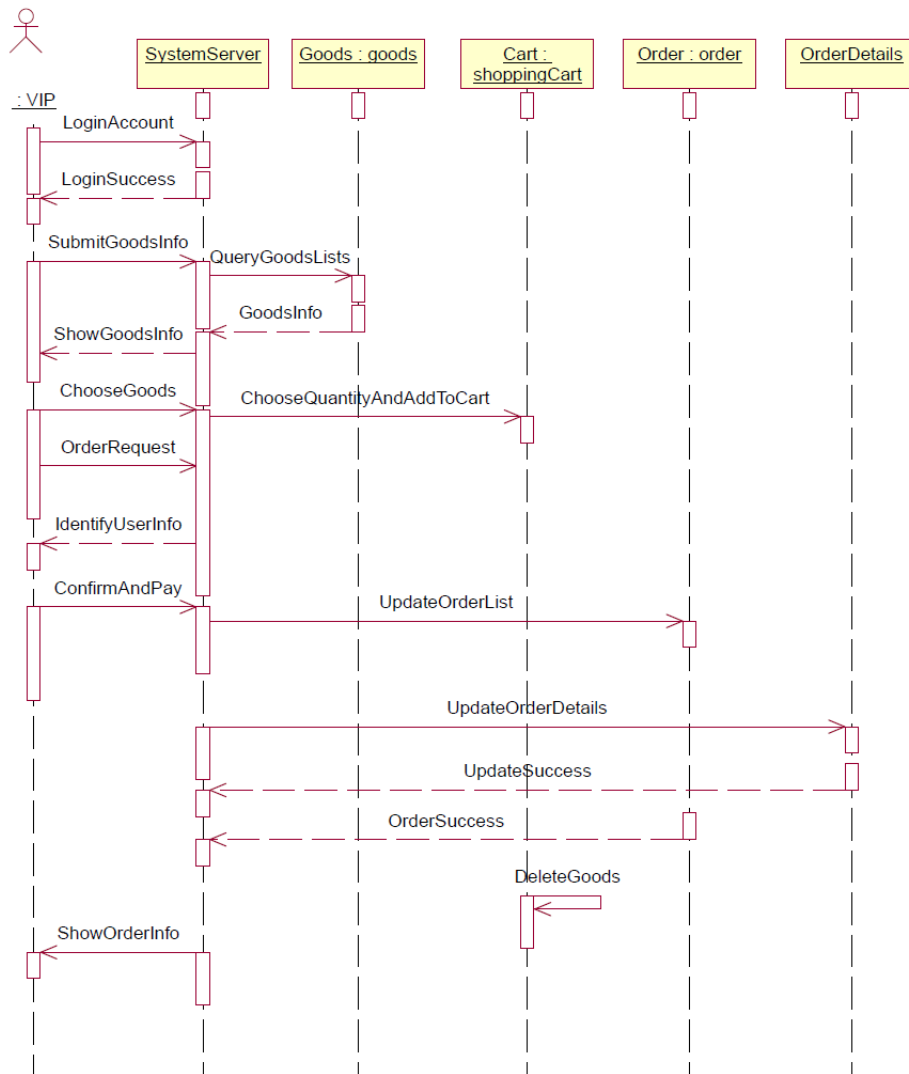
③ QueryAndValidate：在数据库中进行查询及验证账户名是否存在，账户密码是否正确；

④ Feedback：数据库反馈给服务器信息，账户密码正确与否；

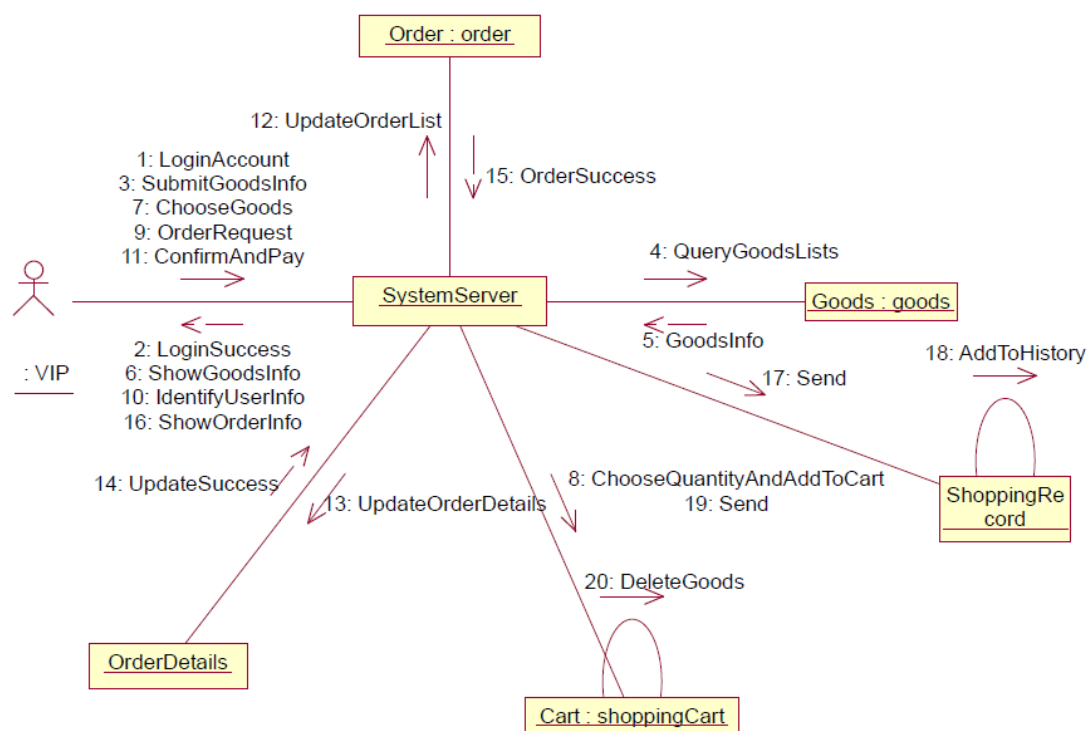
⑤ Return：服务器返回给登录界面，登录界面显示登录成功与否信息。

⑥ ShowInfo：登录界面显示给用户信息；

### 3.3.1 会员下订单顺序图



3.3.2 会员下订单协作图



#### 【顺序图及协作图说明】

执行者：VIP 会员

SystemServer：系统服务器；

Product\_info：商品表；

Order\_info：订单表；

Order\_detail：订单细节表；

Cart：购物车；

ShoppingRecord：购物记录；

① LoginAccount：会员登录账户；

② LoginSuccess：系统服务器返回登录成功信息；

③ SubmitGoodsInfo：提交搜索的商品信息；

④ QueryGoodsList：查询商品列表是否有符合该信息的商品；

⑤ GoodsInfo：若有符合条件的商品，返回商品信息给用户；

⑥ ShowGoodsInfo：显示给用户商品信息；

⑦ ChooseGoods：选择商品；

⑧ ChooseQuantityAndAddToCart：选择商品数量并且添加到购物车；

⑨ OrderRequest：给服务器发送下单请求；

⑩ IdentifyUserInfo：用户审核个人信息；

⑪ ConfirmAndPay：确认下单并且付款；

⑫ UpdateOrderList：服务器根据请求更新订单列表，如订单编号等；

⑬ UpdateOrderDetails：服务器更新订单细节，比如用户个人信息等；

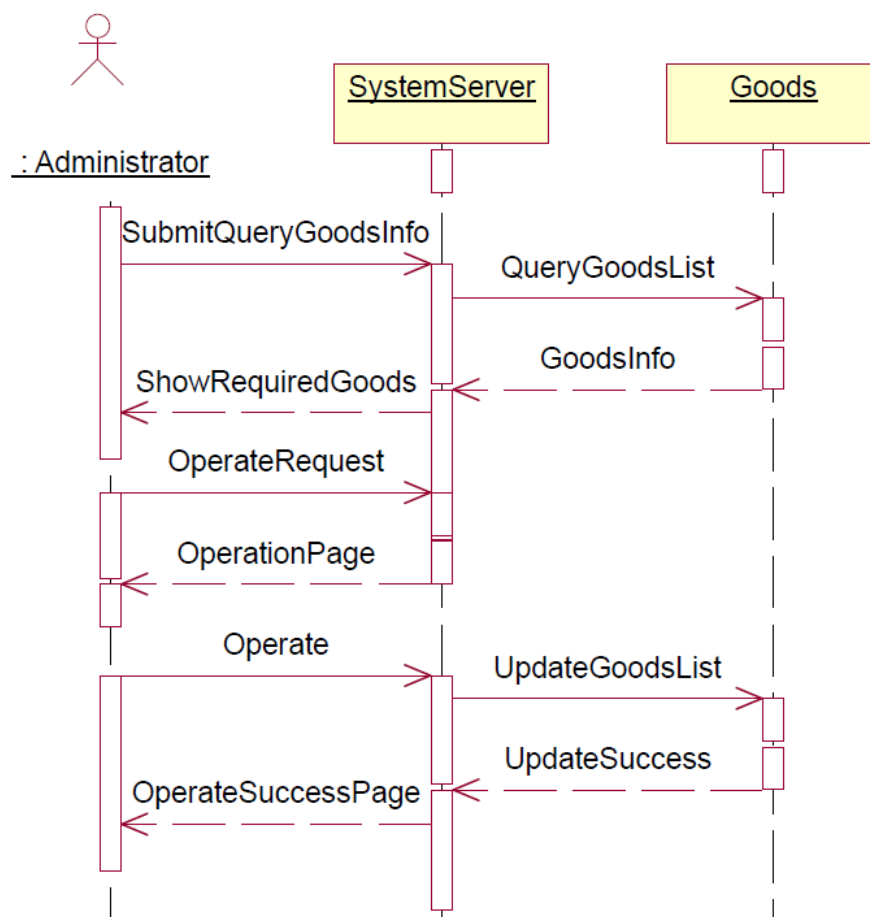
⑭ UpdateSuccess：订单细节更新成功；

⑮ OrderSuccess：下单成功；

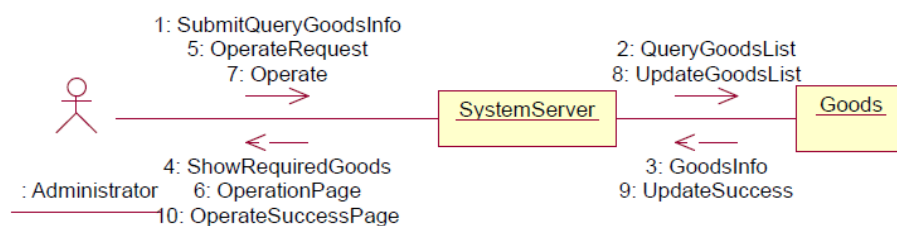
⑯ ShowOrderInfo：显示给用户订单信息；

- ⑰ Send: 服务器发送命令给“购物记录”;
- ⑱ AddToHistory: 将该商品添加到购物历史中;
- ⑲ Send: 服务器发送命令给购物车;
- ⑳ DeleteGoods: 删除购物车中相应的商品;

### 3.5.1 管理员操作商品顺序图（增加，删除，查询，修改等操作）



### 3.5.2 管理员操作商品协作图（增加，删除，查询，修改等操作）



#### 【顺序图及协作图说明】

执行者: Administrator;

SystemServer: 系统服务器;

Goods: 商品列表;

① SubmitQueryGoodsInfo: 管理员提交要查询的商品信息;

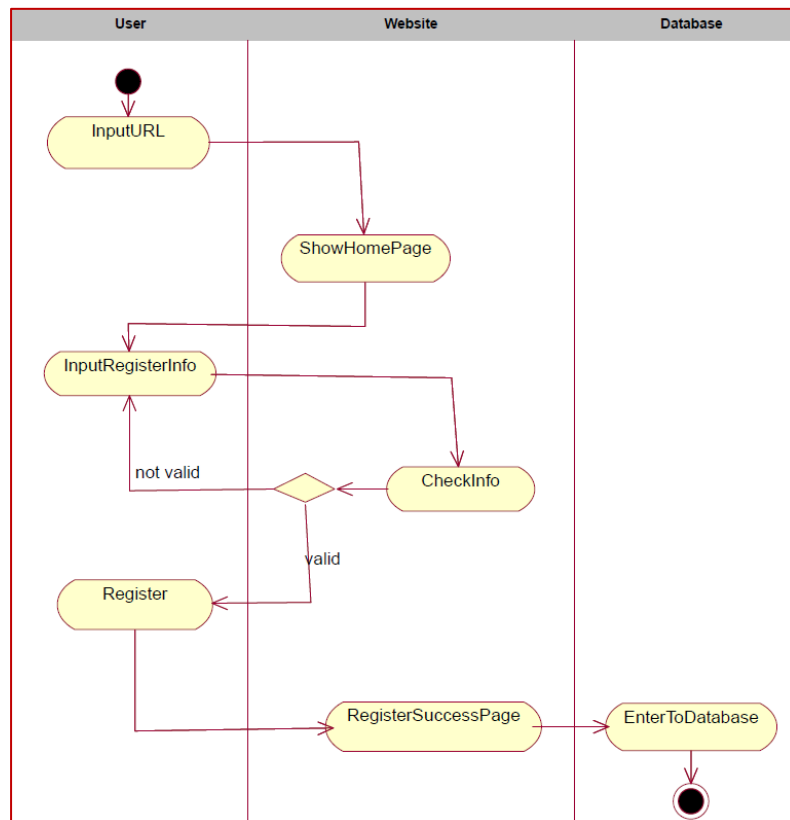
② QueryGoodsList: 查询商品列表;

③ GoodsInfo: 返回符合要求的商品信息;

- ④ ShowRequiredGoods 显示符合管理员查询条件的商品；
- ⑤ OperateRequest: 商品操作请求；
- ⑥ OperationPage: 系统服务器返回商品操作页面；
- ⑦ Operate: 管理员对商品进行操作（增删查改等）；
- ⑧ UpdateGoodsList: 管理员修改商品之后，更新商品列表；
- ⑨ UpdateSuccess: 更新成功提示；
- ⑩ OperateSuccessPage: 服务器返回管理员商品操作成功页面；

## 4. 活动图

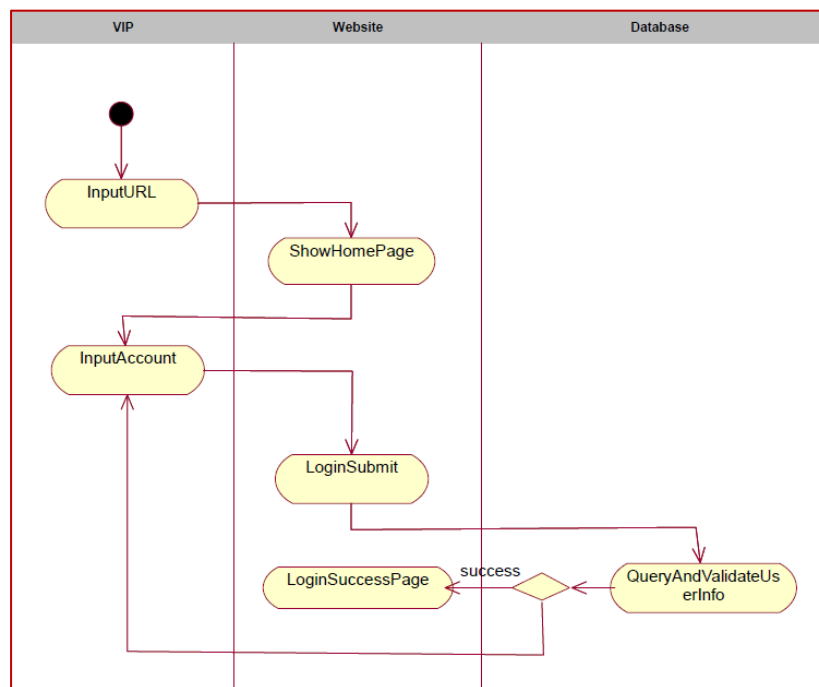
### 4.1 用户注册活动图



#### 【活动图说明】

- (1) InputURL: 用户输入电商网站的链接；
- (2) ShowHomePage: 访问 URL 后显示电商网站的主页；
- (3) InputRegisterInfo: 用户输入注册信息；
- (4) CheckInfo: 网站审核信息的正确有效性；  
若无效，显示错误信息，回到 InputRegisterInfo 继续输入；  
若有效则提交 Register 表单；
- (5) RegisterSuccessPage: 网站显示注册成功页面；
- (6) EnterToDatabase: 将用户的注册信息录入数据库；

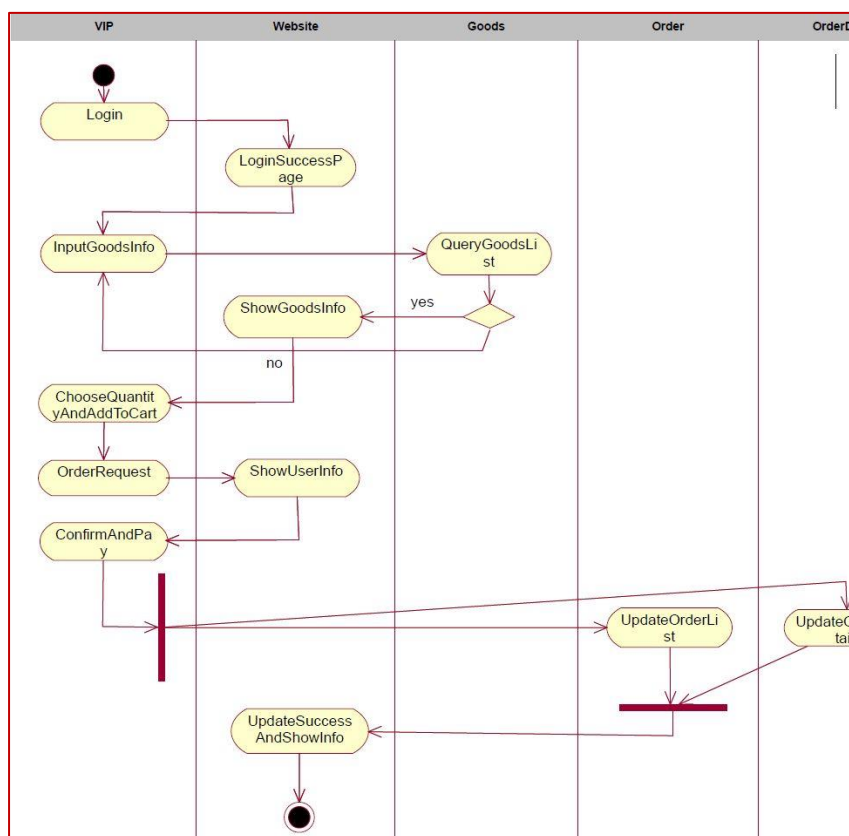
### 4.2 用户登录活动图



#### 【活动图说明】

- (1) InputURL: 用户输入电商网站的链接;
- (2) ShowHomePage: 访问 URL 后显示电商网站的主页;
- (3) InputAccount: 输入账户名和密码;
- (4) LoginSubmit: 提交登录表单;
- (5) QueryAndValidateUserInfo: 在数据库查询并验证用户输入的账户是否存及账户存在时密码是否正确;
- (6) LoginSuccessPage: 账户名及密码有效且正确, 显示登录成功页面; 否则, 回到 InputAccount 继续输入。

#### 4.3 会员下订单活动图



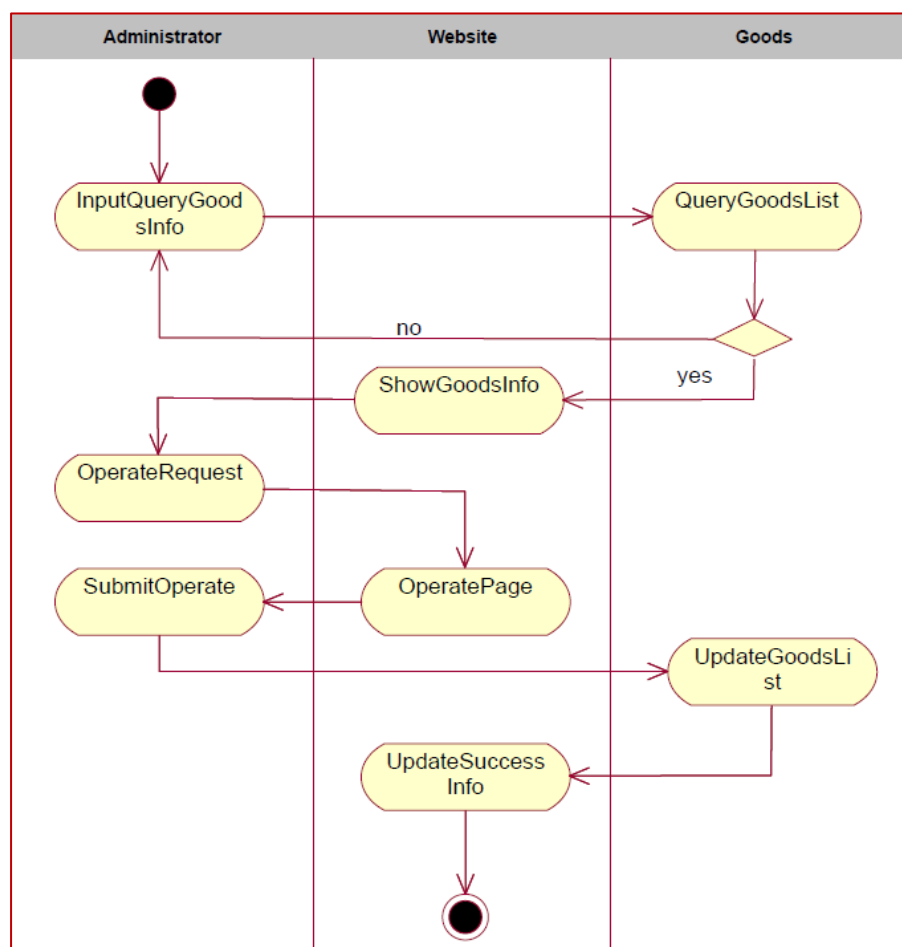
注：图中右边缺少部分为 UpdateOrderDetails 和 OrderDetails。

#### 【活动图说明】

- (1) Login: 登录电商账户;
- (2) LoginSuccessPage: 登录成功页面; (网站主页)
- (3) InputGoodsInfo: 用户输入查询商品信息;
- (4) QueryGoodsList: 查询商品列表若有该商品信息, 则在网站页面显示 (ShowGoodsInfo), 若没有, 则返回 InputGoodsInfo 继续输入;
- (5) ChooseQuantityAndAddToCart: 选择符合条件的商品, 选择数量并且添加到购物车。
- (6) OrderRequest: 下单请求;
- (7) ShowUserInfo: 网站显示下单用户的具体信息 (收货地址, 联系方式等);
- (8) ConfirmAndPay: 用户确认个人信息之后进行付款;  
之后同时进行更新订单列表的订单编号等管理员掌握的信息以及订单的细节 (收货地址, 收货人, 联系方式等) 信息;
- (9) UpdateSuccessAndShowInfo: 更新成功提示及向用户显示下单成功的订单状态。

#### 4.4 管理员操作商品活动图 (增删查改)





#### 【活动图说明】

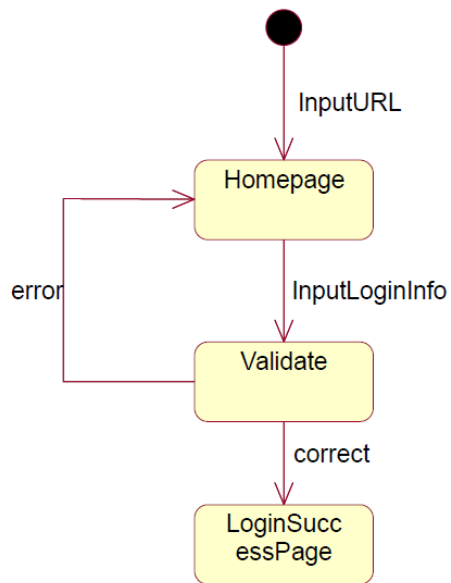
- (1) InputQueryGoodsInfo: 输入要查询的商品信息;
- (2) QueryGoodsList: 查询商品列表;  
如果符合条件的物品存在, 则在网站显示商品信息;  
如果没有符合条件的物品, 则继续 InputQueryGoodsInfo;
- (3) OperateRequest: 商品操作请求;
- (4) OperatePage: 网站显示商品操作页面;
- (5) SubmitOperate: 提交商品操作表单;
- (6) UpdateGoodsList: 更新操作后的商品列表;
- (7) UpdateSuccessInfo: 更新成功消息提示;

## 5. 状态图

### 5.1 系统状态分析

首先登录该网的过程会有状态的转变, 其次, 商品本身会有一定的状态转变, 管理添加商品有创建商品状态, 在数据库中, 在购物车中, 已发货, 在运输途中, 已删除等状态。订单也会有和商品同样的状态, 当用户下单后, 其变化和商品一样, 同步变化。

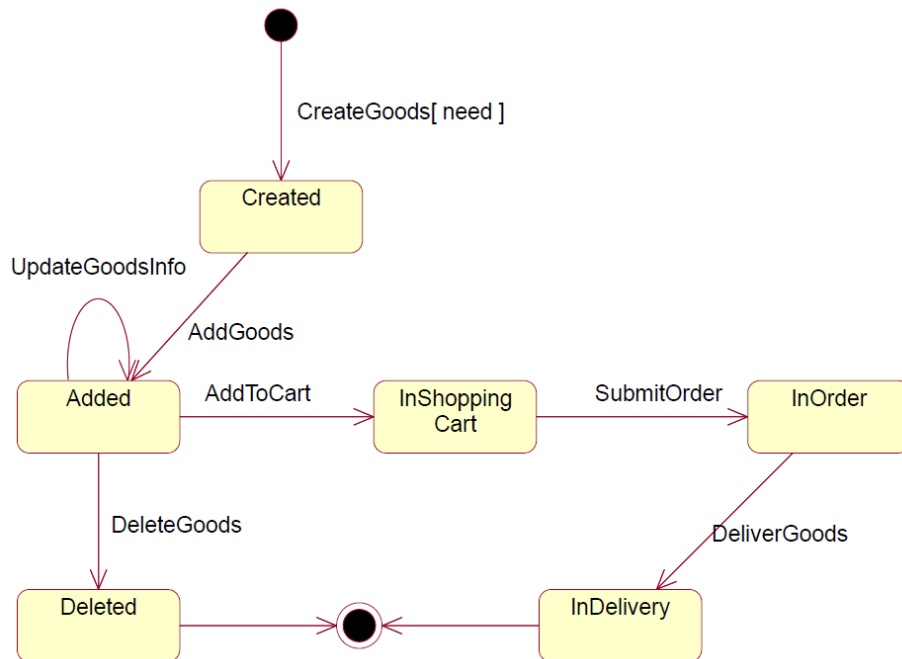
### 5.2 用户登录状态图



#### 【状态图说明】

- (1) Homepage: 网站主页;
- (2) Validate: 验证账户及密码;
- (3) LoginSuccessPage: 用户登录成功页面状态;

### 5.3 商品状态图



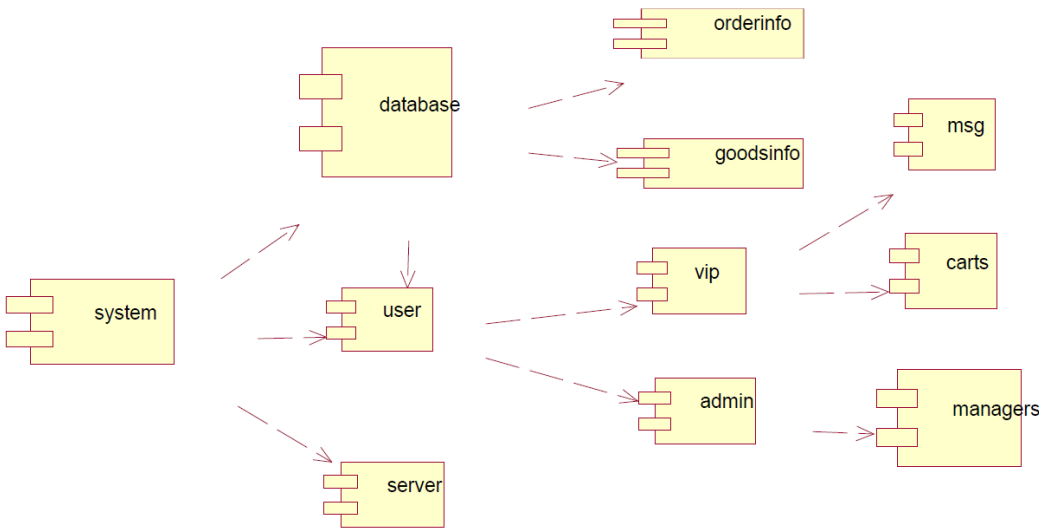
#### 【状态图说明】

- (1) Created: 通过 CreateGoods（创建商品）事件添加，处于创建状态;
- (2) Added: 通过 AddGoods（添加商品）事件添加，处于加入状态;
- (3) InShoppingCart: 通过 AddToCart（添加到购物车）事件添加，处于购物车中状态;
- (4) InOrder: 通过 SubmitOrder（提交订单）事件添加，处于下单状态;

- (5) InDelivery: 通过 DeliverGoods（运送商品）事件添加，处于运送状态；
- (6) Deleted: 通过 DeleteGoods（删除商品）事件添加，处于删除状态；

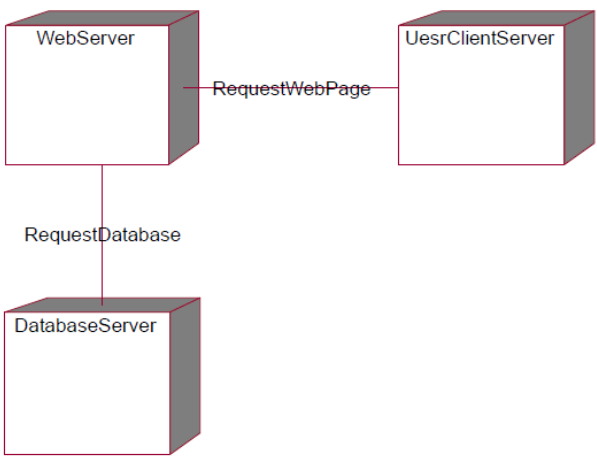
6. 构件图

该系统的构建包括数据库、用户、系统服务器以及数据库里面的订单信息、商品信息、用户信息三个组件，vip 用户里面的购物车、留言等组件，admin 后台管理员里面的普通管理员组件。



7. 部署图

部署图主要是用来说明如何配置系统的软件和硬件。整个电商网站服务器应该是总服务器，数据库服务器负责保存整个电商网站的数据记录，用户客户端服务器为客户访问该电商网站提供服务，应该是一个终端的集合。部署图如下：




二、 数据库设计

在数据库 digital 中主要使用的表有：

user\_info(用户信息表)、  
 type(商品类型表)、  
 product\_info(商品信息表)、  
 order\_info(订单信息表)、  
 order\_detail(订单明细表)、  
 admin\_detail(管理员详细信息表)、  
 admin\_info(管理员信息表)、  
 functions(系统功能信息表)、  
 powers(用户权限表)。

具体表的设计如下：

user\_info:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	用户编号
userName	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>		用户名
password	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>		用户密码
realName	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		真实姓名
sex	varchar	2	0	<input type="checkbox"/>	<input type="checkbox"/>		性别
address	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		家庭地址
email	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		电子邮件
regDate	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		注册日期

type:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	类型编号
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		类型名称

Product\_info:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
code	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>		商品编号
name	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		商品名称
tid	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		商品类别
brand	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		商品品牌
picture	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		商品图片
inventory	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		商品库存
price	int	10	0	<input type="checkbox"/>	<input type="checkbox"/>		商品单价
introduce	longtext	0	0	<input type="checkbox"/>	<input type="checkbox"/>		商品简介
status	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		商品状态
credit	int	5	0	<input type="checkbox"/>	<input type="checkbox"/>		商品积分

Order\_info:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	订单信息id
uid	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		用户id
status	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>		订单处理状态
ordertime	datetime	0	0	<input type="checkbox"/>	<input type="checkbox"/>		下单时间
orderprice	decimal	8	2	<input type="checkbox"/>	<input type="checkbox"/>		订单总价

Order\_detail:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	订单明细id
oid	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		订单id
pid	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		产品id
num	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		购买数量


Admin\_detail 表:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	管理员编号
address	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>		管理员地址
realname	varchar	8	0	<input type="checkbox"/>	<input type="checkbox"/>		管理员真实姓名



Admin\_info 表:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	编号
name	varchar	16	0	<input type="checkbox"/>	<input type="checkbox"/>		姓名
pwd	varchar	50	0	<input type="checkbox"/>	<input type="checkbox"/>		密码
role	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		角色

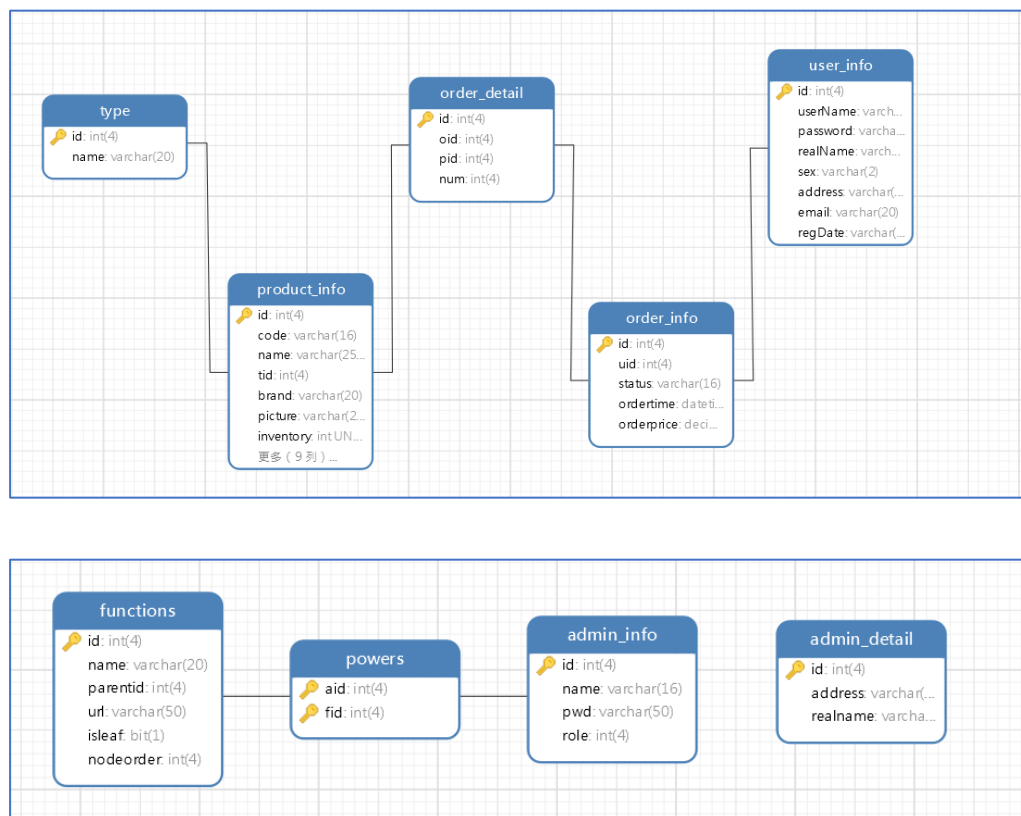
Functions 表:

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	编号
name	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>		功能菜单
parentid	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		父id
url	varchar	50	0	<input type="checkbox"/>	<input type="checkbox"/>		链接
isleaf	bit	1	0	<input type="checkbox"/>	<input type="checkbox"/>		订单是否离开
nodeorder	int	4	0	<input type="checkbox"/>	<input type="checkbox"/>		订单顺序

Powers 表:

名	类型	长度	小数点	不是 null	虚拟	键	注释
aid	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	管理员编号
fid	int	4	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 2	功能编号

各图表之间的 ER 关系图如下:



### 三、系统功能与界面实现

该部分主要是——Struts2 进行表示层开发。其中涉及到任务的主要就是运用 Struts2 框架进行注册和登录，以及利用 Struts2 的特性进行一些列程序的完善和国际化，具体任务包括：

#### 1. 认识 Struts 2 框架

Struts2 框架是 Struts1 和 WebWork 两大框架的整合，建立的一个兼容 WebWork 和 Struts1 的 MVC 框架。

Struts2 的工作原理：通过拦截器来处理用户的请求从而允许用户的业务逻辑控制器和 servlet 分离，在处理用户的请求过程中以用户的业务逻辑控制器为目标，创建一个控制器代理，控制器代理回调业务控制器中的 execute 方法来处理用户的请求，该方法的返回值决定了 struts2 以怎样的视图资源呈现给用户。

#### 2. 使用 JSP+Struts 2+JDBC 实现用户登录

关键代码：

Action 类 login() 函数如下：

```

public String login() throws Exception {
    Connection conn=null;
    PreparedStatement preparedStatement=null;
    ResultSet resultSet=null;
    String back=null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/digital?"
            + "useUnicode=true&characterEncoding=utf8&serverTimezone=GMT",
            "root","zdz199804103033");
        String sql="select * from user_info where userName='"+userName+"' "
            + " and password='"+password+"'";
        preparedStatement=conn.prepareStatement(sql);
        resultSet=preparedStatement.executeQuery(sql);
        if (resultSet.next()) {
            Map<String, Object> session=null;
            ActionContext ac=ActionContext.getContext();
            session=ac.getSession();
            session.put("CURRENT_USER", userName);

            long currentTime=System.currentTimeMillis();
            Long startTime=(Long)session.get("startTime");
            if(startTime==null){
                startTime=currentTime;
                session.put("startTime", startTime);
            }

            long usedTime=(currentTime-startTime)/1000/60;
            if(usedTime>=60){
                setMessage("您已经访问喵购网"+usedTime+"分钟，起来活动一下呦！");
            }else if(usedTime==0){
                setMessage("您已经开始访问喵购网，祝您愉快！");
            }else{
                setMessage("您已经访问喵购网"+usedTime+"分钟！");
            }
            back="success";
        }
    }
}

```

```

    }else{
        setMessage("登录失败，请检查用户名和密码是否正确！");
        back="input";
    }
} catch (Exception e) {
    e.printStackTrace();
}finally{
    try {
        if (null!=resultSet) {
            resultSet.close();
        }
        if (null!=preparedStatement) {
            preparedStatement.close();
        }
        if (null!=conn) {
            conn.close();
        }
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
return back;
}
}

```

Struts2 配置如下：

```

<!--登录action -->
<action name="Login" class="com.digital.action.LoginAction" method="login">
    <result name="success">/index.jsp</result>
    <result name="input">/login.jsp</result>
</action>


```

功能界面实现截图：


## 登录界面

# 登录

用户名

 请输入用户名




密码

 请输入密码

[忘记密码?](#)

登录

第三方登录


[没有账户? 立即注册](#)

## 登录验证:


# 登录

登录失败, 请检查用户名和密码是否正确!

用户名

 tom




密码

 ●●●●●●

[忘记密码?](#)

登录

第三方登录



### 3. 使用 Struts 2 表单标签实现用户注册

Action 类 Register() 函数:

```
public String register() {  
  
    UserDAO userDAO=new UserDAOImpl();  
    int result=userDAO.addUser(user);  
    String back;  
    if(result!=0){  
        back="success";  
    }else{  
        back="input";  
    }  
    return back;  
}
```

DAO 层接口:

```
public interface UserDAO { //接口  
    //实现增加一个用户  
    public int addUser(User user);  
}
```

DAO 的实现类:

```
public class UserDAOImpl extends BaseDAO implements UserDAO{  
  
    Connection connection=null;  
    PreparedStatement preparedStatement=null;  
    ResultSet resultSet=null;  
  
    public int addUser(User user) {  
        int result=0;  
        String sql="INSERT INTO user_info(userName,password,"  
            + "realName,address,email,regDate) values (?, ?, ?, ?, ?, ?)";  
        try{  
            connection=this.getConnection();  
            preparedStatement=connection.prepareStatement(sql);  
            preparedStatement.setString(1, user.getRealName());  
            preparedStatement.setString(2, user.getPassword());  
            preparedStatement.setString(3, user.getRealName());  
            preparedStatement.setString(4, user.getAddress());  
            preparedStatement.setString(5, user.getEmail());  
            preparedStatement.setString(6, user.getRegDate());  
            result=preparedStatement.executeUpdate();  
        }catch(Exception e){  
            e.printStackTrace();  
        }finally{  
            closeAll(connection, preparedStatement, resultSet);  
        }  
        return result;  
    }  
}
```

Struts2 配置:

```
<!--注册action 注意: 在struts框架中的action name要和对应 action的jsp文件中表单  
的action名称一致, 其实是将对应jsp文件中表单的action名称传给struts框架进行处理  
method 值为对应action Java文件中的方法名称或者直接不写该属性  
用重写execute方法就会默认执行该方法 -->  
<action name="register" class="com.digital.action.RegisterAction" method="register" >  
    <result name="success" /> /userinfo.jsp</result>  
    <result name="input" /> /register.jsp</result>  
</action>
```

Register.jsp 页面 struts2 标签实现注册: (部分)

```


<!-- 使用Struts 2标签完成注册功能 -->
<s:form cssClass="login100-form validate-form" action="register" method="post" name="regForm">
  <span class="login100-form-title p-b-49">
    <s:text name="register.title"/>
  </span>
  <div class="wrap-input100">
    <div class="label-input100">
      <s:text name="userName"/>
      <font color="red">${fieldErrors["user.userName"][0]}</font>
    </div>
    <s:textfield cssClass="input100" name="user.userName" placeholder="%{getText('userName_tip')}/>
    <span class="focus-input100" data-symbol="&#xf207;"></span>
  </div>
  <br>
  <div class="wrap-input100">
    <div class="label-input100">
      <s:text name="password"/>
      <font color="red">${fieldErrors["user.password"][0]}</font>
    </div>
    <s:textfield cssClass="input100" name="user.password" type="password" placeholder="%{getText('password_tip')}/>
    <span class="focus-input100" data-symbol="&#xf191;"></span>
  </div>
  <br>
  <div class="wrap-input100">
    <div class="label-input100">
      <s:text name="repassword"/>
      <font color="red">${fieldErrors["repassword"][0]}</font>
    </div>
    <s:textfield cssClass="input100" name="repassword" type="password" placeholder="%{getText('repassword_tip')}/>
    <span class="focus-input100" data-symbol="&#xf191;"></span>
  </div>
</s:form>

```


功能界面实现:

注册


用户名

 zdz


密码

 .....


确认密码

 .....


真实姓名

 zhaodeze

家庭住址

 甘肃省武威市

电子邮箱

 1360536767@qq.com

注册

[已有账户? 立即登录](#)

#### 4. 使用 Struts 2 标签实现商品类别显示

Action 类 list() 函数:

```
public class ProductInfoAction extends ActionSupport implements RequestAware {
    Map<String, Object> request;
    public String list() {
        TypeDAO typeDAO = new TypeDAOImpl();
        List<Type> typeList = typeDAO.getAll();
        if (typeList.size() > 0) {
            request.put("typeList", typeList);
        }
        return "index";
    }

    @Override
    public void setRequest(Map<String, Object> request) {
        this.request = request;
    }
}
```

DAO 层接口:

```
public interface TypeDAO {
    //得到type表的所有类型
    public List<Type> getAll();
}
```

DAO 的实现类:

```
public class TypeDAOImpl extends BaseDAO implements TypeDAO {
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    @Override
    public List<Type> getAll() {
        List<Type> typelist = null;
        typelist = new ArrayList<Type>();
        String sql = "select * from type";
        try {
            connection = this.getConnection();
            preparedStatement = connection.prepareStatement(sql);
            resultSet = preparedStatement.executeQuery(sql);
            Type type = null;
            while (resultSet.next()) {
                type = new Type();
                type.setId(resultSet.getInt("id"));
                type.setName(resultSet.getString(2));
                typelist.add(type);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeAll(connection, preparedStatement, resultSet);
        }
        return typelist;
    }
}
```

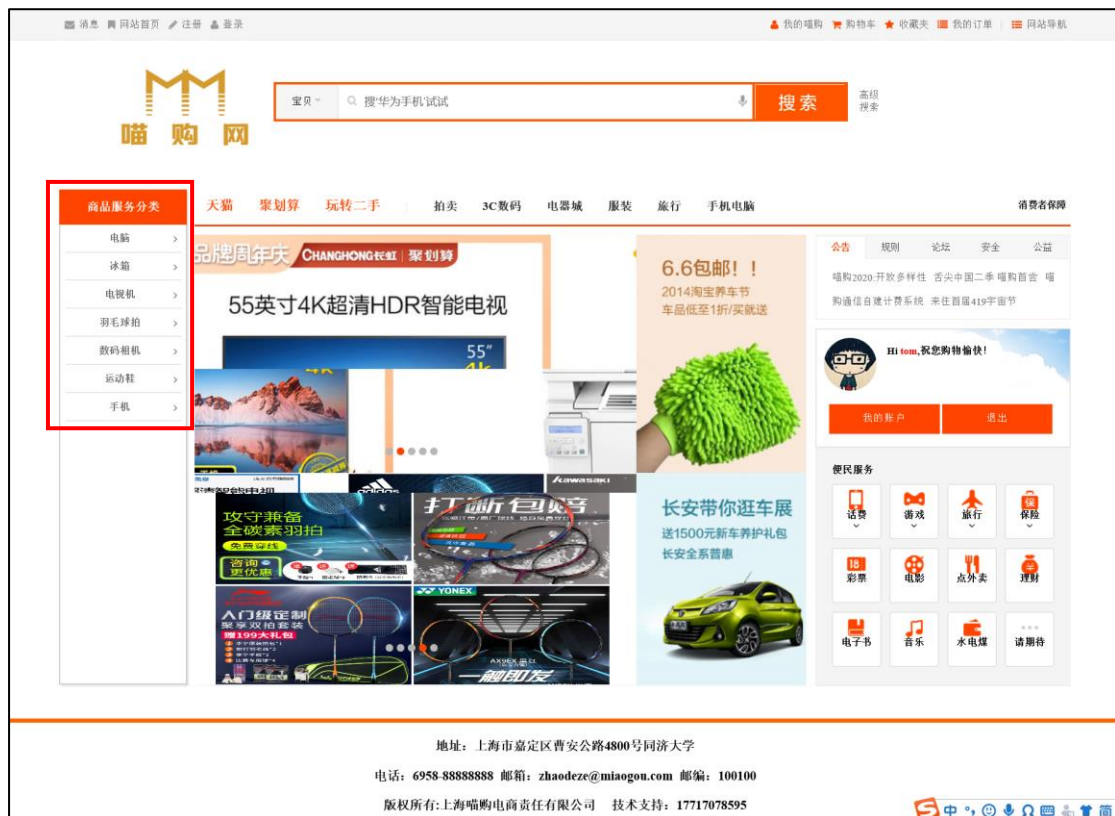
Struts2 配置:

```
<!--商品信息分类action -->
<action name="list" class="com.digital.action.ProductInfoAction" method="list">
    <result name="index">/index.jsp</result>
    <result name="input">/login.jsp</result>
</action>
```

JSP 页面代码:

```
<!--sidebar-->
<div class="sidebar">
  <h3>商品服务分类</h3>
  <!--sidebar-info-->
  <div class="sidebar-info">
    <ul class="side-li">
      <s:iterator id="typeItem" value="#request.typeList">
        <li><h3>${typeItem.name}</h3><span class="fa fa-angle-right fa-loc"></span></h3>
        <ul>
          <li>商品种类</li><li>商品种类</li><li>商品种类</li>
        </ul>
      </s:iterator>
    </ul>
  </div>
</div>
</div>
```

功能界面实现: (需放大查看)



## 5. 使用 method 属性及通配符修改登录注册

该子任务在上面的 struts2 配置中即可看到 method 属性下有相应的登录注册 Action 类的方法名。

通配符:

在 struts 配置中修改登录和注册 action:

```
<!--登录action -->
<action name="*Action" class="com.digital.action.LoginAction" method="{1}" >
  <!--
  <result name="success">/index.jsp</result> -->
  <result name="success" type="redirectAction">list</result>
  <result name="input">/{1}.jsp</result>
</action>
```

```

<!--注册action 注意: 在struts框架中的action name要和对应 action的jsp文件中表单
的action名称一致, 其实是对应jsp文件中表单的action名称传给struts框架进行处理
method 值为对应action Java文件中的方法名称或者直接不写该属性
用重写execute方法就会默认执行该方法 -->
<action name="*" class="com.digital.action.{1}Action" method="{2}" >
    <result name="success" /> /userinfo.jsp</result>
    <result name="input" /> {2}.jsp</result>
</action>

```

同时对应地, 还要修改 login.jsp 登录 form 表单中的 action 属性内容为 loginAction, 修改 register.jsp 注册 form 表单中的 action 属性内容为 Register\_register, (在 Action 中 method 名分别为 login 和 register 的前提下)。

## 6. 使用自定义拦截器完善登录功能

拦截器类:

```

public class AuthorizationInterceptor extends AbstractInterceptor {

    @Override
    public String intercept(ActionInvocation invocation) throws Exception {
        Map session=invocation.getInvocationContext().getSession() ;
        String user=(String)session.get("CURRENT_USER");
        if(user==null){
            return Action.INPUT;
        }else{
            return invocation.invoke();
        }
    }
}

```

Struts2 配置:

```

<interceptors>
    <interceptor name="myAuthorization" class="com.digital.interceptor.AuthorizationInterceptor"></interceptor>
    <!-- 定义拦截器栈 -->
    <interceptor-stack name="myStack">
        <interceptor-ref name="defaultStack"/>
        <interceptor-ref name="myAuthorization"/>
    </interceptor-stack>
</interceptors>

```

```

<!--登录action -->
<action name="Login" class="com.digital.action.LoginAction" method="login">
    <result name="success" /> /index.jsp</result>
    <result name="success" type="redirectAction">list</result>
</action>

```

```

<!--商品信息分类action -->
<action name="list" class="com.digital.action.ProductInfoAction" method="list">
    <result name="index" /> /index.jsp</result>
    <interceptor-ref name="myStack"></interceptor-ref>
    <result name="input" /> /login.jsp</result><!-- 有了global results 此句可省略 -->
</action>

```

此时在浏览器地址栏输入 localhost:8080/digital-4/list 即可跳转到登录页面 (login.jsp), 从而实现要想获得商品分类列表, 就必须得登录。

## 7. 使用 Struts 2 的验证框架完善程序

## 8. 使用 Struts 2 的国际化完善程序

由于登录部分的验证是在 java 后台 message 实现的, 在此对注册部分的进行 struts2 框架验证架进行验证。再者, 我将国际化也一并用到了登录和注册中, 两个任务一并进行。

register-validation.xml:

```
<validators>
  <field name="user.userName">
    <field-validator type="requiredstring">
      <param name="trim">true</param>
      <message key="userName.null"></message>
    </field-validator>
    <field-validator type="stringlength">
      <param name="maxLength">10</param>
      <param name="minLength">2</param>
      <message key="userName.length"></message>
    </field-validator>
  </field>

  <field name="user.password">
    <field-validator type="requiredstring">
      <param name="trim">true</param>
      <message key="password.null"></message>
    </field-validator>
    <field-validator type="stringlength">
      <param name="minLength">8</param>
      <param name="maxLength">16</param>
      <message key="password.length"></message>
    </field-validator>
  </field>

  <field name="repassword">
    <field-validator type="requiredstring">
      <param name="trim">true</param>
      <message key="repassword.null"></message>
    </field-validator>
    <field-validator type="fieldexpression">
      <param name="expression">user.password==repassword</param>
      <message key="repassword.same"></message>
    </field-validator>
  </field>
</validators>
```

```

<field name="user.realName">
  <field-validator type="requiredstring">
    <param name="trim">true</param>
    <message key="realName.null"></message>
  </field-validator>
  <field-validator type="stringlength">
    <param name="maxLength">10</param>
    <param name="minLength">2</param>
    <message key="realName.length"></message>
  </field-validator>
</field>

<field name="user.address">
  <field-validator type="requiredstring">
    <param name="trim">true</param>
    <message key="address.null"></message>
  </field-validator>
  <field-validator type="stringlength">
    <param name="maxLength">50</param>
    <message key="address.length"></message>
  </field-validator>
</field>

<field name="user.email">
  <field-validator type="requiredstring">
    <param name="trim">true</param>
    <message key="email.null"></message>
  </field-validator>
  <field-validator type="email">
    <param name="trim">true</param>
    <message key="email.format"></message>
  </field-validator>
</field>
</validators>

```

Resource.properties(简体中文)

name	value
register.page	用户注册
register.title	注册
userName	用户名
password	密码
repassword	确认密码
realName	真实姓名
address	家庭住址
email	电子邮箱
submit	立即注册
userName.null	不能为空!
userName.length	长度必须在 \${minLength} 和 \${maxLength}之间!
password.length	长度必须在 \${minLength} 和 \${maxLength}之间!
repassword.null	不能为空!
realName.length	长度必须在 \${minLength} 和 \${maxLength}之间!
address.length	长度不能超过 \${maxLength}!
repassword.same	必须和密码一致!
realName.null	不能为空!
address.null	不能为空!
email.null	不能为空!
password.null	不能为空!
email.format	格式无效!
userName_tip	请输入用户名
realName_tip	请输入真实姓名
address_tip	请输入家庭住址
password_tip	请输入密码
repassword_tip	请确认密码
email_tip	请输入电子邮箱
has_account	已有账户? 立即登录
no_account	没有账户? 立即注册
login.title	登录
forget	忘记密码?
third_login	第三方登录

## Resource\_zh\_HK.properties(繁体中文)

name	value
register.page	用戶註冊
register.title	註冊
userName	用戶名
password	密碼
repassword	確認密碼
realName	真實姓名
address	家庭住址
email	電子郵件
submit	註冊
userName.null	不能為空!
userName.length	長度必須在 \${minLength} 和 \${maxLength}之間!
password.length	長度必須在 \${minLength} 和 \${maxLength}之間!
repassword.null	不能為空!
realName.length	長度必須在 \${minLength} 和 \${maxLength}之間!
address.length	長度不能超過 \${maxLength}!
repassword.same	必須和密碼一致!
realName.null	不能為空!
address.null	不能為空!
email.null	不能為空!
password.null	不能為空!
email.format	格式無效!
userName_tip	請輸入用戶名
realName_tip	請輸入真實姓名
address_tip	請輸入家庭住址
password_tip	請輸入密碼
repassword_tip	請確認密碼
email_tip	請輸入電子郵件
forget	忘記密碼?
login.title	登錄
third_login	第三方登錄
has_account	已有賬戶? 立即登錄
no_account	沒有賬戶? 立即註冊



## Resource\_en\_US.properties (英文)

name	value
register.page	Sign Up
register.title	Sign Up
userName	User Name
password	Password
repassword	Confirm Password
realName	Real Name
address	Address
email	Email
submit	sign up
userName.null	cannot be null!
userName.length	length must between \${minLength} and \${maxLength}!
password.length	length must between \${minLength} and \${maxLength}!
repassword.null	cannot be null!
realName.length	length must between \${minLength} and \${maxLength}!
address.length	length cannot exceed \${maxLength}!
repassword.same	must be the same as password!
realName.null	cannot be null!
address.null	cannot be null!
email.null	cannot be null!
password.null	cannot be null!
email.format	format is not valid!
userName_tip	enter user name
realName_tip	enter your real name
address_tip	enter your address
password_tip	enter password
repassword_tip	confirm password
email_tip	enter your email address
has_account	already has an account ? Sign In
no_account	Don't have an Account ? Sign Up
forget	Forgot Password?
login.title	Sign In
third_login	Sign In With Third Part

功能界面实现：

## 注册

用户名 不能为空!

 请输入用户名

密码 不能为空!

 请输入密码

确认密码 不能为空!

 请确认密码

真实姓名 不能为空!

 请输入真实姓名

家庭住址 不能为空!

 请输入家庭住址

电子邮箱 不能为空!


 请输入电子邮箱

注册

[已有账户? 立即登录](#)

## 注册

用户名 长度必须在 2 和 10之间!

 zxxwqeeweqeq

密码 长度必须在8 和 16之间!

 .....

确认密码 必须和密码一致!

 .....

真实姓名 长度必须在 2 和 10之间!

 赵

家庭住址 长度不能超过50!

 甘肃省武威市民勤县大滩镇大西村七社七号甘肃

电子邮箱 格式无效!

 213141241.qq.com

注册

[已有账户? 立即登录](#)

## 註冊

用戶名 長度必須在 2 和 10之間！

zzzwqeeweqeq

密碼 長度必須在 8 和 16之間！

●●●●●●●●

確認密碼 必須和密碼一致！

●●●●●●●●

真實姓名 長度必須在2 和 10之間！

赵

家庭住址 長度不能超過50！

甘肅省武威市民勤縣大灣鎮大西村七社七號甘肅

電子郵箱 格式無效！

213141241.qq.com

註冊

[已有賬戶？立即登錄](#)

## 註冊

用戶名 不能為空！

請輸入用戶名

密碼 不能為空！

請輸入密碼

確認密碼 不能為空！

請確認密碼

真實姓名 不能為空！

請輸入真實姓名

家庭住址 不能為空！

請輸入家庭住址

電子郵箱 不能為空！

請輸入電子郵箱

註冊

[已有賬戶？立即登錄](#)

Sign Up

User Name length must between 2 and 10!

zxzwqeeweqeq

Password length must between 8 and 16!

.....

Confirm Password must be the same as password!

.....

Real Name length must between 2 and 10!

赵

Address length cannot exceed 50!

甘肃省武威市民勤县大滩镇大西村七社七号甘肃

Email format is not valid!

213141241.qq.com

SIGN UP

Already Has An Account ? Sign In

Sign Up

User Name cannot be null!

enter user name

Password cannot be null!

enter password

Confirm Password cannot be null!

confirm password

Real Name cannot be null!

enter your real name

Address cannot be null!

enter your address

Email cannot be null!

enter your email address

SIGN UP

Already Has An Account ? Sign In

## 9. 使用 Struts 2 的 Ajax 标签显示提示信息

首先导入头标签: `<%@ taglib prefix="sx" uri="/struts-dojo-tags" %>`

接着在 index.jsp 页面添加代码:

```

<!--user-info-->
<div class="user-info">
  <s:if test="#session.CURRENT_USER!=null">
    <div class="gold-top">
      
      <div class="inner-user">
        <h3>Hi <font color="red">${sessionScope.CURRENT_USER}</font>, 祝您购物愉快! </h3>
        <sx:div id="tsdiv" updateFreq="8000" href="loginAction">
          </sx:div>
        <h3><font color="red">${message}</font></h3>
      </div>
    </div>
    <div class="login">
      <a class="login-btn" href="userInfo.jsp">我的账户</a>
      <a class="login-btn free" href="logout">退出</a>
    </div>
  </s:if>

  <s:else>
    <div class="gold-top">
      
      <div class="inner-user">
        <h3>亲! 欢迎光临, 登录享精彩哦! </h3>
      </div>
    </div>
  </s:else>
</div>

<!--login-->
<div class="login">
  <a class="login-btn" href="login.jsp"><i class="fa fa-user fa-user-loc"></i>登录</a>
  <a class="login-btn free" href="register.jsp">注册</a>
</div>
</div>

```

功能界面实现：（对应的是 `login()` 红框中的代码）



## 四、系统配置

### (1) 开发平台

计算机型号: Lenovo 小新潮7000

计算机内存: 8GB

计算机CPU: Intel Core i7-8550U @1.80GHz @2.00GHz

计算机操作系统类型: 64位操作系统, 基于x64的处理器

计算机Windows版本: Windows 10 家庭中文版

开发软件: MyEclipse Enterprise Workbench Version: 2017 Stable 2.0

Build id: 15.0.1-20171220

开发框架: Struts2框架: struts-2.3.37

MySQL版本: MySQL 8.0.16 connect/J: 8.0.16

MySQL Workbench: Navicat Premium 12.1.18 64-bit

Tomcat版本: Apache Tomcat 8.0 、 Apache Tomcat 9.0

浏览器: Google Chrome 、 microsoft-edge、 IE

### (2) 运行环境

在以上集成环境下进行项目配置即可在浏览器调试运行。