**快递物流系统**

**体系结构设计文档**

学 院：南京大学软件学院

成 员：潘凌伟 申彬 谭昕玥 杨雁飞

完成日期：2015 年10月26日

**目录**

[更新历史 3](#_Toc433658202)

[1、引言 4](#_Toc433658203)

[1.1 编制目的 4](#_Toc433658204)

[1.2 词汇表 5](#_Toc433658205)

[1.3 参考资料 6](#_Toc433658206)

[2、产品概述 6](#_Toc433658207)

[3、逻辑视角 7](#_Toc433658208)

[4、组合视角 9](#_Toc433658209)

[4.1 开发包图 10](#_Toc433658210)

[4.2 运行时进程 15](#_Toc433658211)

[4.3 物理部署 16](#_Toc433658212)

[5、接口视角 17](#_Toc433658213)

[5.1 模块的职责 17](#_Toc433658214)

[5.2 用户界面层的分解 20](#_Toc433658215)

[5.2.1 用户界面层模块的职责 22](#_Toc433658216)

[5.2.2用户界面层模块的接口规范 22](#_Toc433658217)

[用户界面层需要的服务接口如表5.2.2-2所示。 25](#_Toc433658218)

[5.2.3 用户界面模块设计原理 25](#_Toc433658219)

[5.3 业务逻辑层的分解 25](#_Toc433658220)

[5.3.1 业务逻辑层模块的职责 26](#_Toc433658221)

[5.3.2 业务逻辑层模块的接口规范 27](#_Toc433658222)

[5.4 数据层的分解 52](#_Toc433658223)

[5.4.1 数据层模块的职责 53](#_Toc433658224)

[5.4.2 数据层模块的接口规范 53](#_Toc433658225)

[6、信息视角 67](#_Toc433658226)

[6.1 数据持久化对象 67](#_Toc433658227)

[6.2 数据库表 72](#_Toc433658228)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 全体人员 | 2015-10-17 | 最初草稿（创建文档模板） | V1.0 |
| 谭昕玥 | 2015-12-30 | 总经理制定城市距离用例修改 | v1.0.1 |

# 1、引言

## 1.1 编制目的

本报表详细完成对互联网酒店预订系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| \_ui | 表示某展示层 |  |
| \_bl | 表示某逻辑层 |  |
| \_data | 表示某数据层 |  |
| RMI | 表示远程方法调用 |  |

## 1.3 参考资料

1. IEEE std 1471-2000
2. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182

# 2、产品概述

参考互联网酒店预订系统用例文档和互联网酒店预订系统软件需求规格说明文档中对产品的概括描述。互联网酒店预订系统主要是应用于客户方便快捷的在网络预订酒店，主要功能见用例图如下。

****

# 3、逻辑视角

互联网酒店预订系统中，选择了分层体系结构风格，将系统分为3层(展示层、业务逻辑层、数据层)能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。



图一 参照体系结构风格的包图表达逻辑视角



图二 软件体系结构逻辑设计方案

# 4、组合视角

[软工2 166]与抽象的逻辑设计相比，实现物理设计要考虑更多的实现细节，这些细节有：

1. presentation层与logic层被置于客户端，data层被置于服务器端，那么logic层的开发包不可能依赖于data层的开发包。使用RMI技术，将data层开发包分解为置于客户端的dataservice接口包和置于服务器端的data层开发包。这样一来，logic层开发包依赖于dataservice包，dataservice包和data层的开发包都依赖于RMI类库包。
2. 所有的data层开发包都需要进行数据持久化（例如读写数据库、读写文件等），所以它们会有一些重复代码，可以将重复代码独立为新的开发包，然后所有的data层开发包都依赖于databasetility。datavaseutility会依赖于JDBC类库包。
3. 所有的presentation层开发包都需要使用图形类型建立界面，都要依赖于图形界面类库包。
4. 此外，presentation层实现时，由mainui包负责整个页面之间的跳转逻辑。其他各包负责各自页面自身的功能。
5. 在分层风格的典型设计中，不希望高层直接依赖于低层，而是为低层建立接口包，实现依赖倒置原则，所以应该调整为：各presentation层开发包（调用）依赖于logic层接口包businesslogicservice包，logic层开发包也依赖于（实现了）logic层接口包businesslogicservice包。
6. 在分层风格的典型设计中，presentation层与logic层之间、logic层与data层之间可能会传递复杂数据对象，那么相邻两层都需要使用数据对象声明，所以需要将数据对象声明独立为开发包（VO包和PO包）。
7. 使用依赖倒置原则消除包的循环依赖现象，将循环依赖变为单向依赖：

* Order和Manage:将部分Manege类抽象接口manegeInfoService置入Order包，这样Manage单向依赖于Order（实现接口+调用）
* Inventory和Manage:将部分Inventory类抽象接口inventoryInfoService置入Manage包，这样Inventory单向依赖于Manage（实现接口+调用）

8）在logic层中，初始化和业务逻辑层上下文的工作被分配到utility包中。

经过细节改进，最终建立的互联网酒店预订系统开发包设计如表4.1-1，其局部包图如图4.1-1和4.1-2所示：

## 4.1 开发包图

表4.1-1 快递物流系统的最终开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| Mainui | OrderHanui,OrderViewui,Userui,Promotionui,Searchui,Hotelui |
| Orderui | OrderHanblservice,界面类库包,vo |
| Orderbl | OrderHanblservice,OderHandataservice,po,Userbl,Promotionbl  Hotelbl |
| Orderblservice |  |
| Orderdata | databaseutility,po, OrderHandataservice |
| Orderdataservice | Java RMI,po |
| Userui | Userblservice,界面类库包,vo |
| Userbl | Userblservice,Userdataservice,po, |
| Userblservice |  |
| Userdata | databaseutility,po, Userdataservice |
| Userdataservice | Java RMI,po |
| Promotionui | Promotionblservice,界面类库包,vo |
| Promotionbl | Promotionblservice,Promotiondataservice,po, |
| Promotionblservice |  |
| Promotiondata | databaseutility,po, Promotiondataservice |
| Promotiondataservice | Java RMI,po |
| Searchui | Searchblservice,界面类库包,vo |
| Searchbl | Searchblservice,Searchdataservice,po,Userbl,Hotelbl |
| Searchblservice |  |
| Searchdata | databaseutility,po, Searchdataservice |
| Searchdataservice | Java RMI,po |
| Hotelui | Hotelblservice,界面类库包,vo |
| Hotelbl | Hotelblservice,Hoteldataservice,po, |
| Hotelblservice |  |
| Hoteldata | databaseutility,po, Hoteldataservice |
| Hoteldataservice | Java RMI,po |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| Databaseutility | JDBC |

图4.1-1 互联网酒店预订系统客户端开发包图



图4.1-2 互联网酒店预订系统服务器端开发包图

## 4.2 运行时进程

在互联网酒店预订系统中，会有多个客户端进程和一个服务器端进程，其进程图如图4.2所示。结合部署图，客户端进程实在客户端机器上运行，服务器端进程是在服务器端机器上运行。

图4.2 进程图

## 4.3 物理部署

互联网酒店预订系统中客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于JavaRMI构件属于JDK6.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图4.3所示。



4.3 部署图

# 5、接口视角

## 5.1 模块的职责

客户端模块和服务器端模块视图分别如图5.1-1和图5.1-2所示。客户端各层和服务器端各层的职责分别如表5.1-1和表5.1-2所示。



图5.1-1 客户端模块视图



图5.1-2 服务器端模块视图

表5.1-1 客户端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面。 |
| 用户界面层 | 基于窗口的进销存系统客户端用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

表5.1-2 服务器端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面。 |
| 数据层 | 负责数据的持久化及数据访问接口。 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表5.1-3所示。

表5.1-3 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| order\_blservice  user\_blservice  promotion\_blservice  search\_blservice  hotel\_blservice | 客户端展示层 | 客户端业务逻辑层 |
| order\_data\_service  user\_data\_service  promotion\_data\_service  search\_data\_service  hotel\_data\_service | 客户端业务逻辑层 | 服务器端数据层 |

借用用户管理用例来说明层之间的调用，如图5.1-3所示。每一层之间都是由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。UserBLService提供了User界面所需要的所有业务逻辑功能UserDataService提供了对数据库的增、删、改、查等操作。这样的实现就大大降低了层与层之间的耦合。



图5.1-3 用户管理用例层之间调用的接口

## 5.2 用户界面层的分解

根据需求，系统存在19个用户界面：登录界面、客户主界面、酒店工作人员主界面、网站营销人员主界面、网站管理人员主界面、个人信息界面、酒店详情界面、酒店搜索界面、酒店搜索记录界面、酒店评价界面、会员登记界面、酒店信息维护界面、客房信息维护界面、酒店优惠政策制定界面、信用值修改界面、网站优惠政策制定界面、调整用户界面、订单浏览界面、订单处理界面。

界面跳转如图5.2所示。



图5.2 用户界面跳转

服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如图5.2-2所示。



图5.2-2 用户界面类

### 5.2.1 用户界面层模块的职责

如表5.2.1-1所示为用户界面层模块的职责。

表5.2.1-1 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和界面的跳转。 |

### 5.2.2用户界面层模块的接口规范

用户界面层模块的接口规范如表5.2.2-1所示。

表5.2.2-1 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| Main\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| Order\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示OrderPanel |
| User\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示UserPanel |
| Promotion\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示PromotionPanel |
| Search\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示SearchPanel |
| Hotel\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示HotelPanel |

用户界面层需要的服务接口如表5.2.2-2所示。

表5.2.2-2 用户界面层模块需要的服务接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| businesslogicservice.order\_blservice | 处理与订单查看、订单处理相关的接口 |
| businesslogicservice.user \_blservice | 处理用户信息的接口 |
| businesslogicservice.promotion\_blservice | 处理酒店、网站促销策略制定的接口 |
| businesslogicservice.search\_blservice | 负责用户信息查询、酒店搜索的接口 |
| businesslogicservice.hotel\_blservice | 处理酒店信息的接口 |

### 5.2.3 用户界面模块设计原理

用户界面利用Java的Swing和AWT库来实现。

## 5.3 业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，User对象负责处理登陆界面的业务逻辑；User对象负责处理管理用户的业务逻辑。业务逻辑层的设计如图5.3-1所示。



图5.3-1 业务逻辑层

### 5.3.1 业务逻辑层模块的职责

业务逻辑层模块的职责如表5.3.1-1所示。

表5.3.1-1 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| orderHanbl | 负责与订单处理相关的服务 |
| oderViewbl | 负责处理与订单查看相关的服务 |
| Userbl | 负责处理与用户信息管理相关的服务 |
| Promotionbl | 负责与促销策略相关的服务 |
| Searchbl | 负责处理与搜索酒店和查询用户信息的服务 |
| Hotelbl | 负责处与酒店信息相关的服务 |

### 5.3.2 业务逻辑层模块的接口规范

表5.3.2-1 orderHanbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| orderHan.handleAbnormalOrder | 语法 | public ResultMessage handleAbnormalOrder(); | |
| 前置条件 | 处理异常订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已执行并且增加用户被扣除的信用值 | |
| orderHan.reverseOrder | 语法 | public ResultMessage reverseOrder(); | |
| 前置条件 | 撤销订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已撤销，根据撤销时间减少用户信用值 | |
| orderHan.finishOrder | 语法 | public ResultMessage finishOrder(); | |
| 前置条件 | 执行订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已执行并且增加客户信用值 | |
| orderHan.createOrder | 语法 | public ExpressOrderVO  createOrder(OrderInfoVO order\_info); | |
| 前置条件 | 生成订单，已知完整订单信息 | |
| 后置条件 | 提交订单，持久化保存订单信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Expressorder\_data.add(ExpressOrderPO order); | | | 新增单一持久化序列 |
| Expressorder\_ data.find(String order\_id); | | | 查找单一持久化序列 |
| Expressorder\_ data.modify(ExpressOrderPO order) | | | 改变单一持久化序列 |
| Expressorder\_data.findArriving(String institution\_id) | | | 查找列表持久化序列（个人订单） |
| Expressorder\_data.findCurrent(String institution\_id) | | | 查找列表持久化序列(酒店订单) |
|  | | | 查找列表持久化序列(网站订单) |

表5.3.2-2 orderViewbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| orderView.personOrders | 语法 | public List<OrderViewVO> personOrders(String username); | |
| 前置条件 | 查看个人订单，已知用户名和密码 | |
| 后置条件 | 根据用户ID返回个人订单列表 | |
| orderView.hotelOrders | 语法 | public List<OrderViewVO> hotelOrders(String hotelname); | |
| 前置条件 | 浏览酒店订单，已知酒店名称 | |
| 后置条件 | 根据酒店名称返回酒店订单列表 | |
|  |  |  | |
|  |  | |
|  |  | |
| orderView.netOrders | 语法 | public List<OrderViewVO> personOrders(String hotelname); | |
| 前置条件 | 浏览网站订单，已经登录验证 | |
| 后置条件 | 返回网站未执行和异常订单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| User\_data.find(String username); | | | 查找单一持久化序列 |

表5.3.2-3 userbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| user.  register | 语法 | public ResultMessage register(UserVO user\_info) | |
| 前置条件 | 客户注册，已知注册信息 | |
| 后置条件 | 根据注册信息更新网站用户信息并返回注册结果 | |
| Order.personInfoView | 语法 | public UserVO personInfoView(String username); | |
| 前置条件 | 显示个人信息，已知客户ID,用户类型为客户 | |
| 后置条件 | 根据客户ID返回客户个信息 | |
| Order.personInfoSave | 语法 | public ResultMessage personInfoSave(UserVO user\_info); | |
| 前置条件 | 保存更改的个人信息，已知更改后的个人信息 | |
| 后置条件 | 更新个人信息并返回保存结果 | |
| Order.userView | 语法 | public UserVO createStockInOrder(String username,); | |
| 前置条件 | 显示用户信息，已知用户类型和ID | |
| 后置条件 | 根据用户类型和ID返回用户信息 | |
| Order.userSave | 语法 | public ResultMessage userSave(UserVO user\_info); | |
| 前置条件 | 保存更改的用户信息，已知更改后用户信息 | |
| 后置条件 | 更新用户信息并返回保存结果 | |
| Order.addCredit | 语法 | public ResultMessage addCredit (UserVO user\_info); | |
| 前置条件 | 已知增加的信用值，需要更新个人信息内的信用值 | |
| 后置条件 | 更新个人信息并返回结果 | |
| Order.registerMember | 语法 | public ResultMessage registerMember(UserVO user\_info); | |
| 前置条件 | 已知会员信息，需要向个人信息中加入会员信息 | |
| 后置条件 | 更新个人信息并返回结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order\_data.add(ArrivedOrderPO order); | | | 新增单一持久化序列（客户个人信息） |
| Order\_data.add(LoadOrderPO order); | | | 返回单一持久化序列（客户个人信息） |
| Order\_data.add(DeliveryOrderPO order); | | | 新增单一持久化序列（酒店工作人员信息） |
| Order\_data.add(StockInOrderPO order); | | | 返回单一持久化序列（酒店工作人员信息） |
| Order\_data.add(StockOutOrderPO order); | | | 新增单一持久化序列（网站营销人员信息） |
| Order\_data.findLoad(String id); | | | 返回单一持久化序列（网站营销人员信息） |

表5.3.2-4 promotion\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Promotion\_bl.priceCut | 语法 | public double priceCut(hotelPO price ,orderhanVO number,orderhanVO type，userPO info); | |
| 前置条件 | 已知用户所需房间数量与类型，已知该酒店房间价格 | |
| 后置条件 | 返回考虑优惠政策后的价格 | |
| Promotion\_bl.changeProm | 语法 | public changeProm(promotionVO prom); | |
| 前置条件 | 酒店工作人员或网站营销人员需要改变优惠政策 | |
| 后置条件 | 改变存储的优惠政策 | |
| Promotion\_bl.getProm | 语法 | public promotionPO(hotelPO hotelname); | |
| 前置条件 | 需要获取当前的优惠政策内容 | |
| 后置条件 | 返回当前优惠政策PO类 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Promotion\_data.add(PromotionPO promotion); | | | 新增单一持久化序列 |
| Promotion\_data.find(hotelVO hotelname) | | | 查找单一持久化序列 |
| Promotion\_data.modify(PromotionPO promotion) | | | 改变单一持久化序列 |

表5.3.2-5 search\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Search\_bl.findhotel | 语法 | public List<hotelPO>getState(searchVO); | |
| 前置条件 | 已知客户的筛选条件,需要查找符合条件的酒店 | |
| 后置条件 | 返回符合条件的酒店PO | |
| Search\_bl.savehistory | 语法 | Public saveOrder (String hotelID); | |
| 前置条件 | 保存搜索酒店记录 | |
| 后置条件 | 持久化保存搜索酒店的记录 | |
| Search\_bl.deletehistory | 语法 | public receiveOrder(String hotelID) | |
| 前置条件 | 已知客户需要删除的酒店编号 | |
| 后置条件 | 数据层删除选定的搜索记录 | |
| Seach\_bl.searchinfo | 语法 | Public userPO/hotelPO(userVO/hotelVO) | |
| 前置条件 | 已知需要查询的用户类型和索引，需要返回查找结果 | |
| 后置条件 | 返回查找的对象的PO类 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Search\_data.add(String hotelID); | | | 增加搜索记录 |
| Hotel\_ data.findWithReq(SearchPO condition); | | | 依据条件查找单一持久化序列 |
| Search\_ datadelete(String hotelID) | | | 删除搜索记录 |
| user\_data.find(String userID, usertype) | | | 依据条件查找用户信息 |

表5.3.2-6 Hotel\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Hotel\_bl.showHotelinfo | 语法 | public HotelPO getHotel(String HotelID); | |
| 前置条件 | 已知酒店编号，需要返回酒店的信息 | |
| 后置条件 | 返回酒店的PO类 | |
| Hotel\_bl.modifyHotelinfo | 语法 | public modifyHotelinfo(HotelVO hotelinfo); | |
| 前置条件 | 酒店工作人员维护酒店的信息 | |
| 后置条件 | 数据层修改酒店的信息 | |
| Hotel\_bl.addcomment | 语法 | public addcomment(HotelVO comment，String userID); | |
| 前置条件 | 用户需要评论酒店 | |
| 后置条件 | 数据层增加酒店评论 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Hotel\_data.getHotelinfo(String HotelID); | | | 根据酒店编号返回酒店信息 |
| Hotel\_ data.modify(HotelPO hotelinfo) | | | 改变单一持久化序列 |
| Hotel\_data.addcomment (comment) | | | 增加评论 |

**以下还没有修改**

## 5.4 数据层的分解

数据层主要给业务逻辑层提供数据防伪服务，包括对于持久化数据的增、删、改、查。User业务逻辑需要的服务由UserDataService接口提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的具体描述如图5.4所示。



图5.4 数据层模块的描述

### 5.4.1 数据层模块的职责

数据层模块的职责如表5.4.1所示。

表5.4.1 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DataService | 持久化数据库的接口，提供集体载入，集体保存、增、删、改、查服务。 |
| DataServiceTxtImpl | 基于TXT文件的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| DataServiceDatabaseImpl | 基于数据库的持久化数据的接口，提供集体载入、集体保存、增、删、改、查服务 |

### 5.4.2 数据层模块的接口规范

表5.4.2-1数据层orderdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Orderdata.addArrived | 语法 | public ResultMessage add(ArrivedOrderPO order); |
| 前置条件 | 与order的id相同的order在数据库中不存在 |
| 后置条件 | 数据库中添加一条ArrivedOrder记录 |
| Orderdata.addLoad | 语法 | public ResultMessage add(LoadOrderPO order); |
| 前置条件 | 与order的id相同的order在数据库中不存在 |
| 后置条件 | 数据库中添加一条LoadOrder记录 |
| Orderdata.addDelivery | 语法 | public ResultMessage add (DeliveryOrderPO order); |
| 前置条件 | 与order的id相同的order在数据库中不存在 |
| 后置条件 | 数据库中添加一条DeliveryOrder记录 |
| Orderdata.findArriving | 语法 | public ArrivingOrderPO findArriving(String order\_id); |
| 前置条件 | 无 |
| 后置条件 | 返回与对应id的ArrivingOrderPO,如果order\_id不存在，返回Null |
| Orderdata.findLoad | 语法 | public LoadOrderPO findLoad(String order\_id); |
| 前置条件 | 无 |
| 后置条件 | 返回与对应id的LoadOrderPO,如果order\_id不存在，返回Null |
| Orderdata.findDelivery | 语法 | public DeliveryOrderPO findDelivery(String order\_id); |
| 前置条件 | 无 |
| 后置条件 | 返回与对应id的DeliveryOrderPO,如果order\_id不存在，返回Null |

表5.4.2-2数据层userdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService. add | 语法 | public ResultMessaage add(UserPO po) |
| 前置条件 | 同样id的po在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个po记录 |
| UserDataService. find | 语法 | public UserPO find(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的UserPO结果 |
| UserDataService. modify | 语法 | public ResultMessage  modify(UserPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |
| UserDataService. delete | 语法 | public ResultMessage  delete(String user\_id) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 删除一个po记录 |
| User\_data.getPayeeInfo | 语法 | public List<PayeePO>  getPayeeInfo() |
| 前置条件 | 逻辑层请求得到数据库中营业员、快递员的ID、名字、职务和最后收款时间。假如是快递员，就加一项揽件量 |
| 后置条件 | 无 |
| User\_data.setPayeeInfo | 语法 | User\_data.setPayeeInfo(List<PayeePO>) |
| 前置条件 | 逻辑层设置用户的最后付款时间，清零本月揽件量 |
| 后置条件 | 数据库修改数据 |

表5.4.2-3数据层accountdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Account\_data.add | 语法 | Public boolean add(AccountPO) |
| 前置条件 | 逻辑层请求增加一条账户 |
| 后置条件 | 数据库增加账户，收入和支出为0 |
| Account\_data.delete | 语法 | Public boolean delete(AccountPO) |
| 前置条件 | 逻辑层请求删除一条账户 |
| 后置条件 | 数据库删除账户 |
| Account\_data. modify | 语法 | public boolean modify(AccountPO) |
| 前置条件 | 逻辑层请求修改一条账户 |
| 后置条件 | 数据库修改账户 |
| Account\_data.trade | 语法 | Public Boolean trade(treadePO) |
| 前置条件 | 逻辑层请求增加或减少某一账户余额 |
| 后置条件 | 增加或减少余额，修改账户的收入或支出记录 |
| Account\_data. getCost\_Profit | 语法 | Public List<Cost\_profitPO> getCost\_Profit () |
| 前置条件 | 逻辑层请求获得成本收益表 |
| 后置条件 | 无 |
| Account\_data. getAccountList | 语法 | Public List<AccountPO> getAccountList() |
| 前置条件 | 逻辑层请求获得银行账户列表 |
| 后置条件 | 无 |

表5.4.2-4数据层inventorydata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InventoryDataService.add | 语法 | Public ResultMessage add(GoodsPO expressorder) |
| 前置条件 | 输入入库快递单号 |
| 后置条件 | 系统将该快递信息加入仓库中 |
| InventoryDataService.delete | 语法 | Public ResultMessage delete(String expressorder\_id) |
| 前置条件 | 输入删除的快递单号 |
| 后置条件 | 系统删去该快递在仓库中的信息 |
| InventoryDataService.modify | 语法 | Public ResultMessage modify (GoodsPO expressorder) |
| 前置条件 | 输入更改的快递信息 |
| 后置条件 | 系统更改快递信息 |
| InventoryDataService.find | 语法 | Public GoodsPO find(String expressorder\_id) |
| 前置条件 | 输入查询的快递单号 |
| 后置条件 | 系统显示查询的快递的信息 |
| InventoryDataService.setAlarm | 语法 | Public ResultMessage setAlarm (double alarmValue) |
| 前置条件 | 输入设置的警戒值 |
| 后置条件 | 系统返回警戒值设置状况 |
| InventoryDataService.getAlarm | 语法 | Public double getAlarm() |
| 前置条件 |  |
| 后置条件 | 系统返回当前警戒值 |

表5.4.2-5数据层receiptdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Receipt\_data.submitPayment | 语法 | Public boolean submitPayment(PaymentListPO) |
| 前置条件 |  |
| 后置条件 |  |
| Receipt\_data.getApprovedPayment | 语法 | Public PaymentListPO getApprovedPayment(String departmentID) |
| 前置条件 | 逻辑层请求获得已提交的审批通过的单据 |
| 后置条件 | 返回已通过的单据列表 |
| Receipt\_data. setLocked | 语法 | setLocked (String departmentID) |
| 前置条件 | 逻辑层请求获得已拉取的单据上锁 |
| 后置条件 | 将已拉取的单据上锁 |
| Receipt\_data.getReceipt | 语法 | Public getReceipt(Date begin,Date end) |
| 前置条件 | 逻辑层请求获得时间段内所有收付款单 |
| 后置条件 | 返回收付款单 |
| Receipt\_data.addBill | 语法 | Public Boolean addBill(BillPO) |
| 前置条件 | 逻辑层请求增加一项收款项进入该营业厅今天的收款单草稿 |
| 后置条件 | 增加该收款项 |

表5.4.2-6数据层managedata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService. addIns | 语法 | public ResultMessaage  addIns(InstitutionPO po) |
| 前置条件 | 同样id的po在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个po记录 |
| ManageDataService. findIns | 语法 | public InstitutionPO  findIns(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的InstitutionPO结果 |
| ManageDataService.  modifyIns | 语法 | public ResultMessage  modifyIns(InstitutionPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |
| ManageDataService. deleteIns | 语法 | public ResultMessage  deleteIns(String ins\_id) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 删除一个po记录 |

表5.4.2-7数据层logdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Log\_data.getLogList | 语法 | Public logPO getLogList(); |
| 前置条件 | 逻辑层请求获得log记录 |
| 后置条件 | 返回log记录 |
| Log\_data.addLog | 语法 | Public Boolean addLog(LogPO) |
| 前置条件 | 逻辑层请求增加一条log记录 |
| 后置条件 | 在数据库中增加该log记录 |

表5.4.2-8数据层strategydata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyDataService. addSalary | 语法 | public ResultMessaage addSalary(SalaryPO po) |
| 前置条件 | 同样id的po在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个po记录 |
| StrategyDataService. findSalary | 语法 | public SalaryPO  findSalary(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的InstitutionPO结果 |
| StrategyDataService. modifySalary | 语法 | public ResultMessage  modifySalary(SalaryPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |
| StrategyDataService. deleteSalary | 语法 | public ResultMessage  deleteSalary(SalaryPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 删除一个po记录 |
| StrategyDataService. findPrice | 语法 | public PricePO find(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的PricePO结果 |
| StrategyDataService. modifyPrice | 语法 | public ResultMessage  modifyPrice(PricePO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |

表5.4.2-9数据层expressorderdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Expressorderdata.add | 语法 | public ResultMessage add(ExpressOrderPO order) |
| 前置条件 | 与order的id相同的order在数据库中不存在 |
| 后置条件 | 在数据库中添加一个order记录 |
| Expressorderdata.  find | 语法 | public ExpressOrderPO find(String order\_id); |
| 前置条件 | 无 |
| 后置条件 | 返回对应id的ExpressOrderPO,如果id不存在，返回null值 |
| Expressorderdata.modify | 语法 | public ResultMessage modify (ExpressOrderPO order); |
| 前置条件 | 数据库中存在与order的id相同的order |
| 后置条件 | 更新数据库中对应的order的信息 |
| Expressorderdata.findArrivingOrders | 语法 | public List<ExpressOrderPO> findArrivingOrders(String institution\_id); |
| 前置条件 | 对应id的机构在数据库中 |
| 后置条件 | 返回正在发往对应id机构的订单列表 |
| Expressorderdata.findCurrentOrders | 语法 | public List<ExpressOrderPO> findCurrentOrders(String institution\_id); |
| 前置条件 | 对应id的机构在数据库中 |
| 后置条件 | 返回正处于对应id机构的订单列表 |

表5.4.2-10数据层initialdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Initial\_data.addInitial | 语法 | Initial\_data.addInitial(InitialPO) |
| 前置条件 | 逻辑层增加一条期初信息 |
| 后置条件 | 数据库中增加该期初信息 |
| Initial\_data.getHistory | 语法 | Public List<initialPO> getHistory(); |
| 前置条件 | 逻辑层请求获得期初信息历史 |
| 后置条件 | 返回期初信息历史 |

表5.4.2-11数据层transmanagedata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService. addCar | 语法 | public ResultMessaage addCar(CarPO po) |
| 前置条件 | 同样id的po在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个po记录 |
| ManageDataService. findCar | 语法 | public CarPO findCar(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的CarPO结果 |
| ManageDataService. modifyCar | 语法 | public ResultMessage  modifyCar(CarPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |
| ManageDataService. deleteCar | 语法 | public ResultMessage  deleteCar(String car\_id) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 删除一个po记录 |
| ManageDataService. addDriver | 语法 | public ResultMessaage addDriver(DriverPO po) |
| 前置条件 | 同样id的po在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个po记录 |
| ManageDataService. findDriver | 语法 | public DriverPO find(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的DriverPO结果 |
| ManageDataService. modifyDriver | 语法 | public ResultMessage  modifyDriver(DriverPO po) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 更新一个po |
| ManageDataService. deleteDriver | 语法 | public ResultMessage  deleteDriver(String driver\_id) |
| 前置条件 | 相同id的po在数据文件中已存在 |
| 后置条件 | 删除一个po记录 |

表5.4.2-12数据层trans\_fare\_data模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Trans\_fare\_data. getTransFare | 语法 | public List<TransFarePO> getTransFare() |
| 前置条件 | 逻辑层请求获得运费列表 |
| 后置条件 | 返回运费列表 |
| Trans\_fare\_data.deleteTransFare | 语法 | Trans\_fare\_data.deleteTransFare(List<TransFarePO>) |
| 前置条件 | 逻辑层请求删除数据库表中的运费项 |
| 后置条件 | 删除相应的运费项 |
| Trans\_Fare\_data.addTransFare | 语法 | public boolean  addTransFare(TransFarePO po) |
| 前置条件 | 逻辑层请求增加一条运费项 |
| 后置条件 | 增加运费项 |

# 6、信息视角

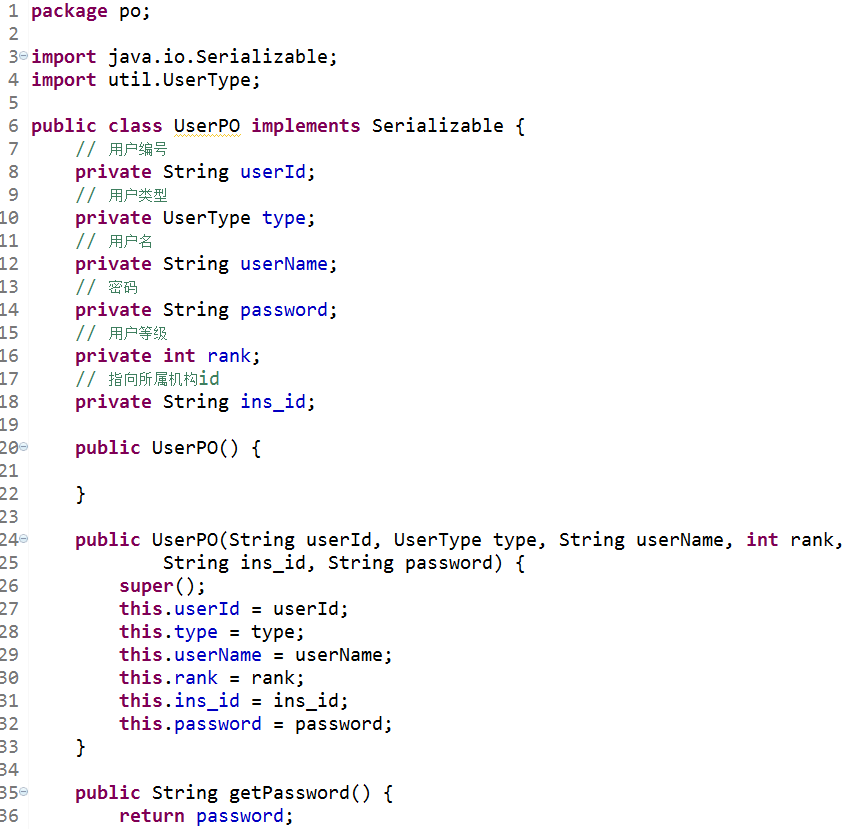
## 6.1 数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍，如表6.1所示：

表6.1 PO类简单介绍

|  |  |
| --- | --- |
| 类名 | 包含的属性 |
| ExpressOrderPO | 寄件单信息：寄件人姓名、住址、单位、电话、手机；收件人姓名、住址、单位、电话、手机；货物原件数、实际重量、体积、内件品名、尺寸；包装类型；本次装箱订单条形码号；快递类型； |
| LoadOrderPO | 装车单或中转单信息：装车日期、机构编号、汽运编号/航班号/火车班次、到达地、车辆代号、监装员、押运员、本次装箱所有订单条形码号 |
| ArrivedOrderPO | 到达单信息：到达日期、中转单编号、出发地、货物到达状态,到达订单 |
| DeliveryOrderPO | 派件单信息：到达日期、托运订单条形码号、派送员，派件订单 |
| StockInOrderPO | 入库单信息：快递编号、入库日期、目的地、区号、排号、架号、位号 |
| StockOutOrderPO | 出库单信息：快递编号、出库日期、目的地、装运形式（火车、飞机、汽车）、中转单编号或者汽运编号 |
| UserPO | 身份ID、密码 |
| BillOrderPO | 包括一个BillPO的List和机构的名称，表格简历的时间 |
| BillPO | 收款单中的一行，包括收款日期、收款金额、收款快递员、对应的所有快递订单条形码号 |
| CarPO | 车辆信息：车辆代号、营业厅编号、车牌号，服役时间 |
| DriverPO | 司机信息：司机编号、营业厅编号、姓名、出生日期、身份证号、手机、性别、行驶证期限 |
| InstitutionPO | 机构编号，人员，所在城市，机构类型 |
| SalaryStrategyPO | 付薪水的策略(计算公式) |
| DistancePO | 城市名、经纬度 |
| PriceStrategyPO | 运费计算(计算公式) |
| SuperOrderPO | 单据审批，查看 |
| PayeePO | 营业员、快递员的ID、名字、职务和最后收款时间。假如是快递员，还有一项揽件量。在该PO用于付款完毕，修改数据库时，付款日期改为，新的最后付款日期，若是快递员，其揽件量为0 |
| TransFarePO | 运费所在的送货单的单号、车辆编号、运费数额。 |
| PaymentListPO | 表示整个付款单PO，包括多条付款项，每条付款项包括：付款日期、付款金额、付款人、付款账号、条目（租金、运费、人员工资、奖励）、备注（租金年份、运单号、标注工资月份） |
| RentPO | 房租的金额，最后交租日期。 |
| AccountPO | 银行账户的名称、余额 |
| LogPO | 操作发起人的职务、名称、操作发起时间、具体操作 |
| Cost\_ProfitPO | 银行账户名称，收入，支出 |
| InventoryPO | 仓库的容量、已用空间以及具体货物编号、位置 |
| InitialPO | 期初信息，包括机构PO、人员PO、车辆PO、库存PO、银行账户PO |

持久化用户对象如UserPO的定义如图6.1，更多定义见原型代码



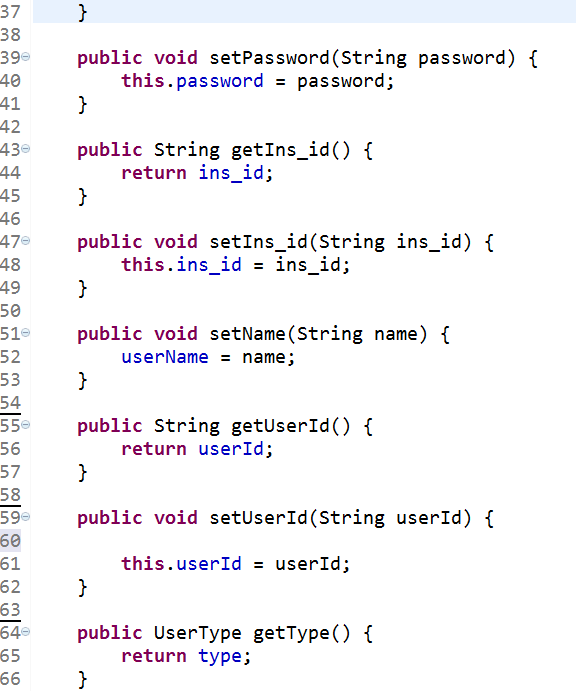
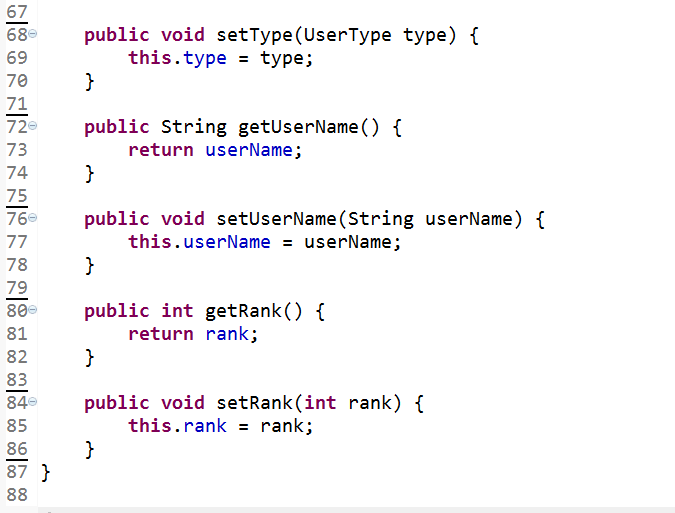
 

图6.1 持久化用户对象UserPO的定义

## 6.2 数据库表

|  |  |
| --- | --- |
| 表名 | 内容 |
| User | 使用系统的用户信息 |
| SalaryStrategy | 薪水策略 |
| CityDistance | 城市距离 |
| Account | 银行账户 |
| ExpressOrder | 寄件单 |
| ArrivedOrder | 到达单 |
| BillOrder | 收款单 |
| Car | 车辆 |
| Driver | 司机 |
| Goods | 库存的货物 |
| LoadOrder | 装车单 |
| DeliveryOrder | 派件单 |
| Institution | 机构 |
| AlarmValue | 警戒值 |
| PaymentList | 付款单 |