**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 |

# 1、引言

## 1.1 编制目的

本报表详细完成对快递物流系统的概要设计，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 |  |
| 单据锁 | 上锁表示已经通过审批的单据已被暂时目的地机构拉取 | 上锁是为了解决同一机构对即将到来的单据作出不同处理 |
| 提交单据 | 表示将一种单据存入服务器数据库，等待审批 |  |
| 拉取单据 | 表示营业厅、中转中心、仓库通过轮询方式把数据库中的、以自己为暂时目的地的、通过审批的单据，拉取至本地，并将其上锁 |  |
| 暂时目的地/即将到达地 | 表示在货物转运途中，该货物即将到达的下一个机构，包括营业厅、中转中心、仓库 |  |
| 账单 | 代指收款单和付款单 |  |
| 送货单 | 代指中转单、装车单、到达单、出入库单 |  |

## 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

参考快递物流系统用例文档和快递物流系统软件需求规格说明文档中队产品的概括描述。快递物流系统主要是应用于快递公司内部交流的在线系统，主要功能见用例图如下。



# 3、逻辑视角

快递物流系统中，选择了分层体系结构风格，将系统分为3层(展示层、

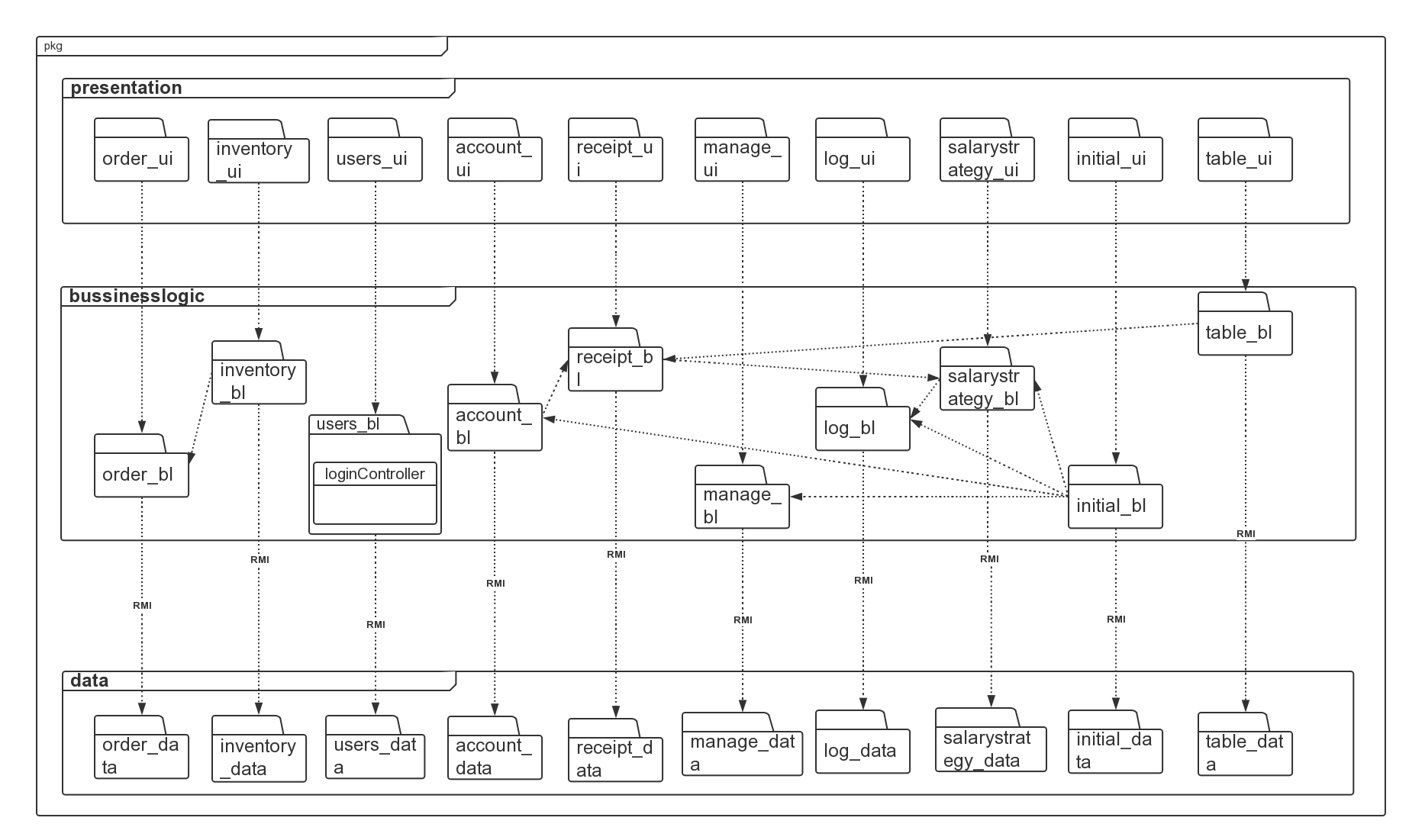
业务逻辑层、数据层)能够很好地示意整个高层抽象。展示层包含GUI页面

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）

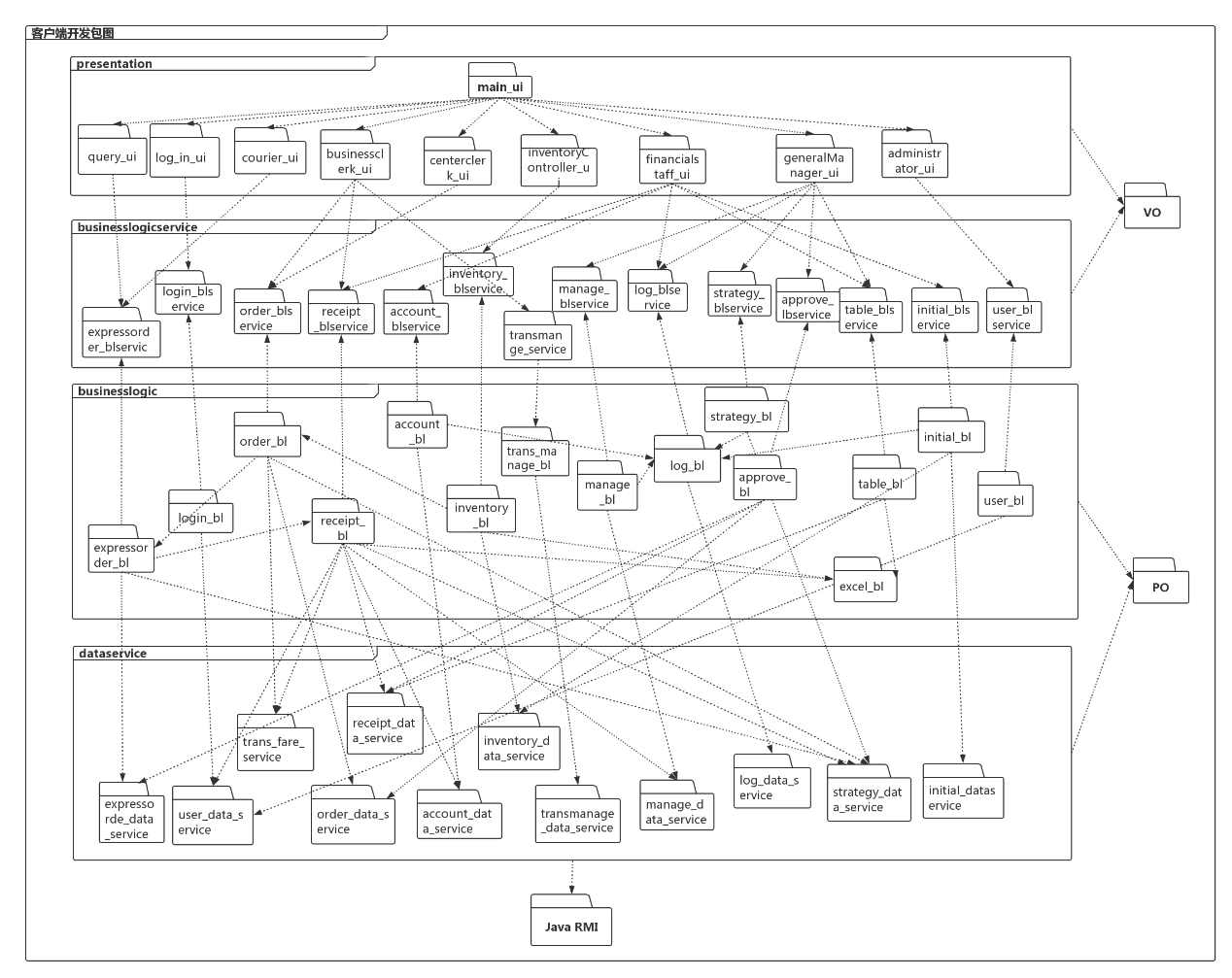
8）在logic层中，初始化和业务逻辑层上下文的工作被分配到utility包中。

经过细节改进，最终建立的快递物流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 |
| main\_ui | query\_ui,login\_ui,courier\_ui,businessclerk\_ui,centerclerk\_ui,inventoryController\_ui,finacialstaff\_ui,generalManager\_ui,administrator\_ui,VO |
| login\_ui | login\_blservice,界面类库包,VO |
| query\_ui | expressorder\_blservice,界面类库包,VO |
| courier\_ui | order\_blservice,界面类库包,VO |
| businessclerk\_ui | order\_blservice,manage\_blservice,界面类库包,VO |
| centerclerk\_ui | order\_blservice,界面类库包,VO |
| inventoryController\_ui | inventory\_blservice,界面类库包,VO |
| finacialstaff\_ui | account\_blservice,table\_blservice,initial\_blservice,界面类库包,VO |
| generalManager\_ui | manage\_blservice,log\_blservice,table\_blservice,界面类库包,VO |
| administrator\_ui | user\_blservice,界面类库包,VO |
| expressorder\_  blservice |  |
| expressorder\_bl | expressorder\_data\_service,PO |
| login\_blservice |  |
| login\_bl | user\_data\_service,PO |
| order\_blservice |  |
| order\_bl | expressorder\_bl,order\_data\_service,PO |
| receipt\_blservice |  |
| receipt\_bl | receipt\_data\_service,PO |
| account\_blservice |  |
| account\_bl | account\_data\_service,PO |
| manage\_blservice |  |
| manage\_bl | manage\_data\_service,strategy\_data\_service,PO |
| transmanage\_blservice |  |
| transmanage\_bl | transmanage\_data\_service |
| log\_blservice |  |
| log\_bl | log\_data\_service,PO |
| strategy\_blservice |  |
| strategy\_bl | strategy\_data\_service,PO |
| table\_blservice |  |
| table\_bl | receipt\_data\_service,PO |
| approve\_blservice |  |
| approve\_bl | expressorder\_data\_service,receipt\_data\_service,order\_data\_service,strategy\_data\_service |
| initial\_blservice |  |
| initial\_bl | strategy\_bl,manage\_bl,log\_bl,strategy\_data\_servidce,PO |
| user\_blservice |  |
| user\_bl | user\_data\_service,PO |
| expressorde\_data\_  service | JAVA RMI,PO |
| expressorde\_data | databaseutility,PO |
| user\_data\_service | JAVA RMI,PO |
| user\_data | databaseutility,PO |
| receipt\_data\_  service | JAVA RMI,PO |
| receipt\_data | databaseutility,PO |
| account\_data\_  service | JAVA RMI,PO |
| account\_data | databaseutility,PO |
| inventory\_data\_  service | JAVA RMI,PO |
| inventory\_data | databaseutility,PO |
| manage\_data\_  service | JAVA RMI,PO |
| manage\_data | databaseutility,PO |
| transmanage\_data\_service | JAVA RMI,PO |
| transmanage\_data | databaseutility,PO |
| log\_data\_service | JAVA RMI,PO |
| log\_data | databaseutility,PO |
| strategy\_data\_  service | JAVA RMI,PO |
| strategy\_data | databaseutility,PO |
| initial\_data\_service | JAVA RMI,PO |
| initial\_data | databaseutility,PO |
| VO |  |
| PO |  |
| excel\_bl |  |
| databaseutility | java.JDBC |
| 界面类库包 |  |
| JAVA RMI | java.swing,java.awt,java.2D,java.3D |

图4.1-1 快递物流系统客户端开发包图

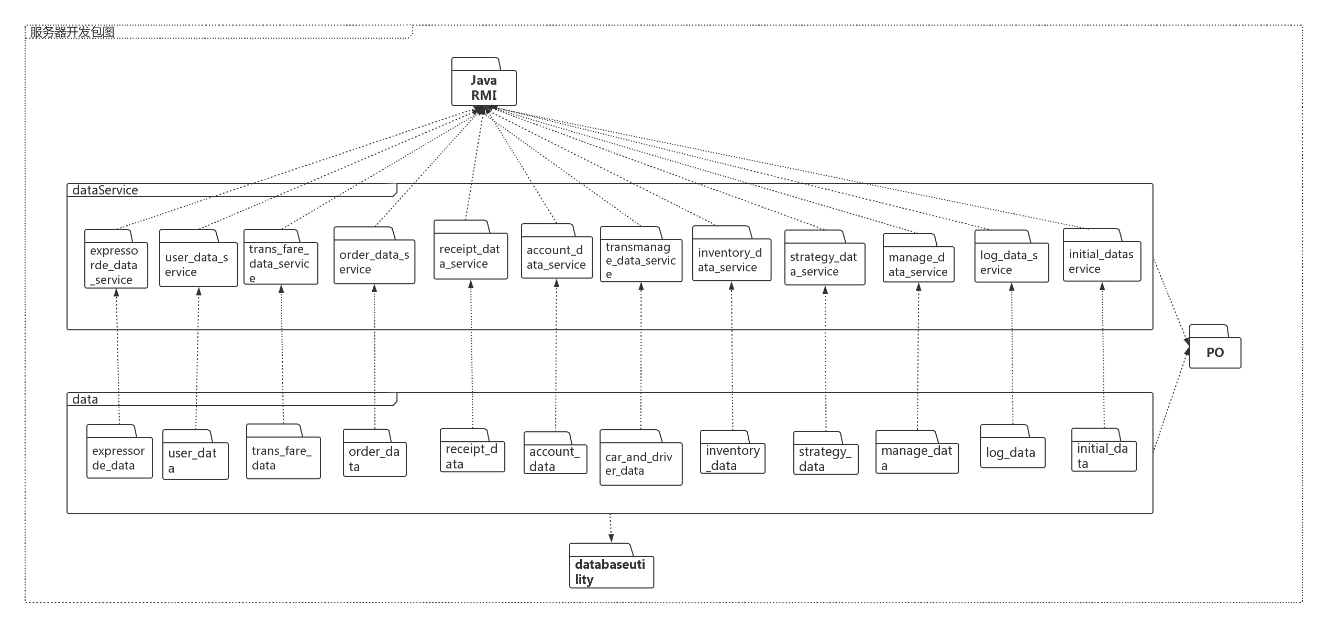


图4.1-2 快递物流系统服务器端开发包图

## 4.2 运行时进程

在快递物流系统中，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

快递物流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 |
| expressorder\_blservice  login\_blservice  order\_blservice  receipt\_blservice  account\_blservice  inventory\_blservice  transmanage\_blservice  manage\_blservice  strategy\_blservice  table\_blservice  initial\_blservice  approvel\_blservice  user\_blservice | 客户端展示层 | 客户端业务逻辑层 |
| expressorder\_data\_service  user\_data\_service  order\_data\_service  receipt\_data\_service  account\_data\_service  inventory\_data\_service  transmanage\_data\_service  manage\_data\_service  log\_data\_service  strategy\_data\_service  initial\_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29个用户界面：登录界面，营业厅业务员界面，制定装车单界面，制定到达单界面，制定派件单界面，制定收款单界面，订单查询界面，货运状态显示界面，快递员界面，库存盘点界面，库区调整界面，库存信息初始化界面，快递入库界面，快递出库界面，查看库存信息界面，制定中转单或装车单界面，中转接收界面，用户账号管理界面，制定薪水策略界面，管理机构界面，查询账户状态界面，制定常量界面，审批单据界面，管理银行账户界面，期初建账界面，制定单据界面，查看日志界面，查看经验情况表界面，查看成本收益表界面。

界面跳转如图5.2所示。

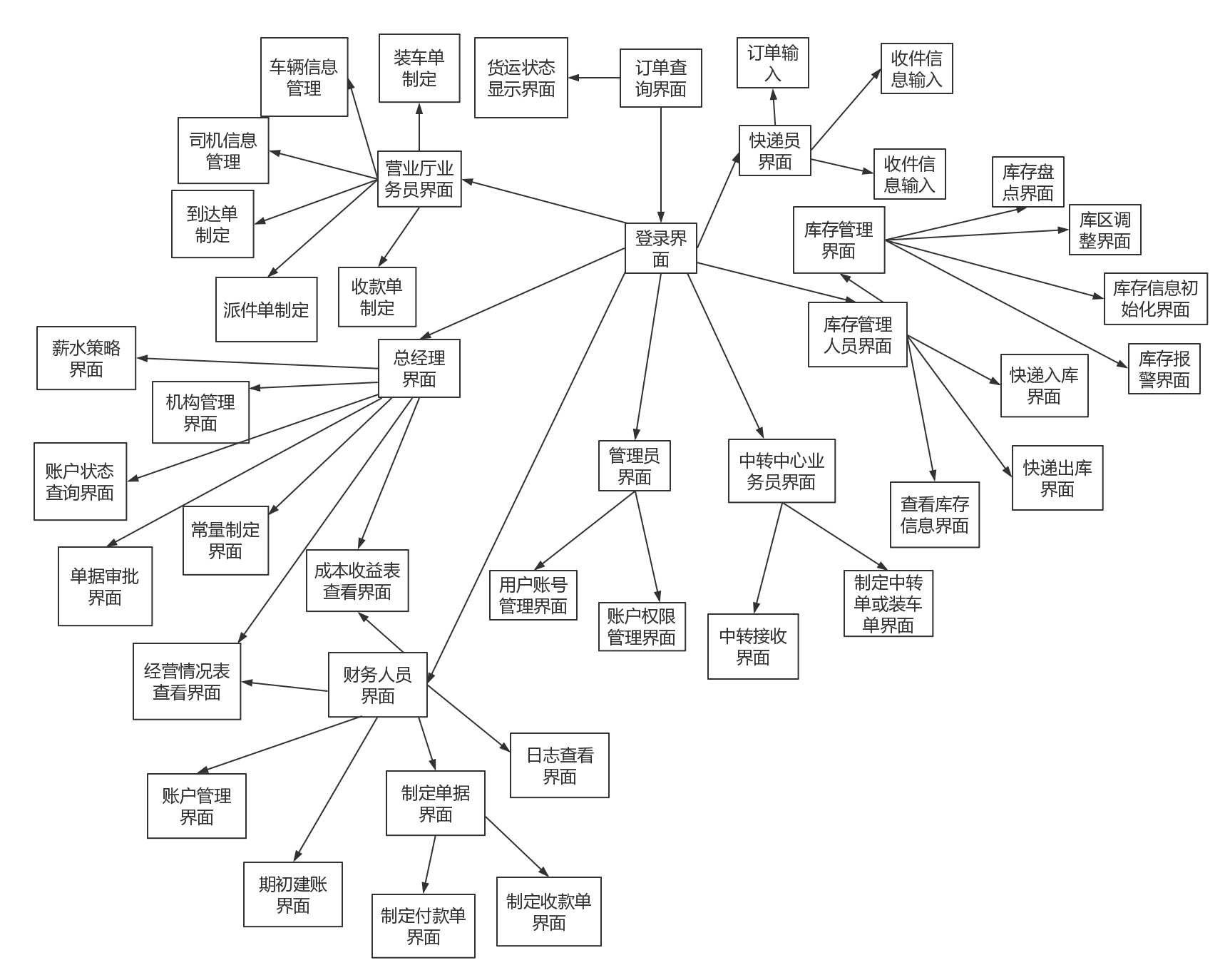


图5.2 用户界面跳转

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以及QueryPanel |
| Query\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示物流状态显示界面 |
| Login\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示LoginPanel |
| Courier\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示CourierPanel |
| Bussinessclerk\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示BussinessclerkPanel |
| Centerclerk\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示CenterClerkPanel |
| InventoryController\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示InventoryControllerPanel |
| Financialstaff\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示FinancialstaffPanel |
| GeneralManager\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示GeneralManagerPanel |
| Administrator\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示AdministratorPanel |

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.login\_blservice | 登陆界面的业务逻辑接口 |
| businesslogicservice.user \_blservice | 处理用户信息的接口 |
| businesslogicservice.expressorder\_blservice | 处理与订单管理相关的接口 |
| businesslogicservice.order\_blservice | 处理各种单据（装车单、到达单）信息的接口 |
| businesslogicservice.account\_blservice | 提供财务人员银行账户管理的接口 |
| businesslogicservice.receipt\_blservice | 根据寄件单记录厅收款单，生成付款单，更改银行账户收入支出信息的接口。 |
| businesslogicservice.inventory\_blservice | 负责处理与库存管理相关的服务 |
| businesslogicservice.manage\_blservice | 负责处理与机构管理相关的接口 |
| businesslogicservice. transmanage\_blservice | 负责处理与运输（车辆、司机）管理相关的服务 |
| businesslogicservice.log\_blservice | 记录系统重要操作的接口 |
| businesslogicservice.strategy\_blservice | 负责处理与薪水策略、常量制定相关的接口 |
| businesslogicservice.table\_blservice | 查看成本收益表、经营状况表的接口 |
| businesslogicservice.initial\_blservice | 实现期初建账的接口 |

### 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.LoginBLService | 登陆界面的业务逻辑接口 |
| businesslogicservice.\*BLService | 每个界面都有一个相应的业务逻辑接口 |

### 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。

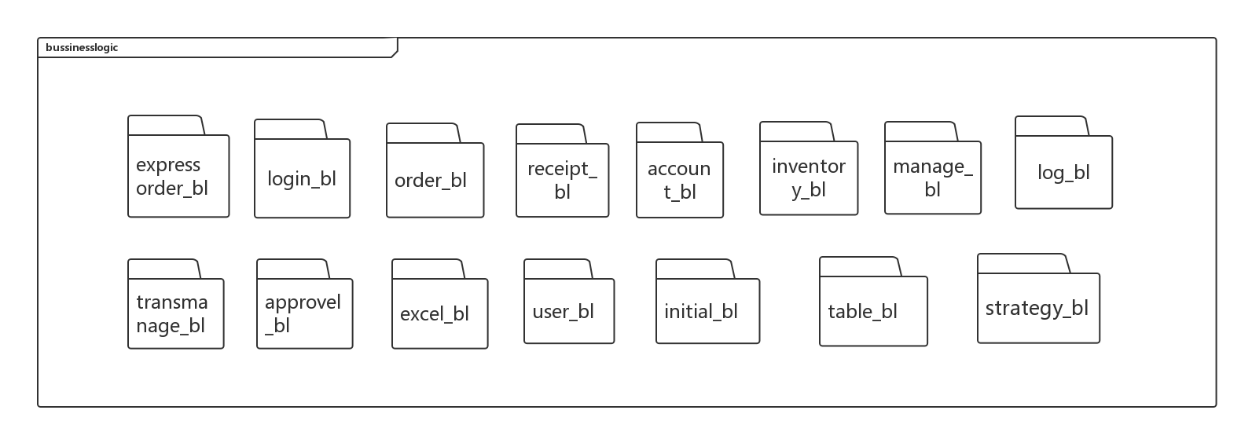


图5.3-1 业务逻辑层

### 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 |
| userbl | 提供管理员的账号管理功能的服务 |
| inventorybl | 负责处理与库存管理相关的服务 |
| expressorderbl | 负责处理与订单管理相关的服务 |
| loginbl | 负责提供验证用户登录的服务 |
| orderbl | 负责处理与货物单据相关的服务 |
| receiptbl | 根据寄件单记录厅收款单，生成付款单，更改银行账户收入支出信息。 |
| accountbl | 提供财务人员银行账户管理的服务。 |
| transmanagebl | 负责处理与运输（车辆、司机）管理相关的服务 |
| managebl | 负责处理与机构管理相关的服务 |
| logbl | 记录系统重要操作，重要操作见需求规格说明中的log |
| arrrovelbl | 负责处理总经理审批单据的服务 |
| Strategybl | 负责处理与薪水策略、常量制定相关的服务 |
| tablebl | 负责查看成本收益表、经营状况表的服务。 |
| excelbl | 负责处理导出excel表格的服务 |
| initialbl | 负责实现期初建账 |

### 5.3.2 业务逻辑层模块的接口规范

表5.3.2-1 expressorder\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Expressorder.getState | 语法 | public StateVO getState(String order\_id); | |
| 前置条件 | 已知订单号，需要获得订单货运状态 | |
| 后置条件 | 返回订单的货运状态 | |
| Expressorder.getArrivingOrders | 语法 | public List<ExpressOrderVO> getArrivingOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得即将送达该机构的订单 | |
| 后置条件 | 返回即将送达该机构的订单 | |
| Expressorder.getCurrentOrders | 语法 | public List< ExpressOrderVO > getCurrentOrders(String institution\_id); | |
| 前置条件 | 已知机构id，需要获得当前机构未送出的订单 | |
| 后置条件 | 返回当前机构未送出的订单 | |
| Expresssorder.setArrivedOrders | 语法 | public ResultMessage setArrivedOrder(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定到达单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态 | |
| Expressorder.setOffOrders | 语法 | public Boolean setOffOrders(List< ExpressOrderVO> orders,String institution\_id); | |
| 前置条件 | 制定装车单，已知机构id和订单列表 | |
| 后置条件 | 系统更改对应订单的货运状态，添加发往地址 | |
| Expressorder.createOrder | 语法 | public ExpressOrderVO createOrder(ExpressorderVO order\_info); | |
| 前置条件 | 生成订单，已知完整订单信息 | |
| 后置条件 | 根据订单信息给出报价和时间信息，返回订单PO类 | |
| Expressorder.saveOrder | 语法 | Public ResultMessage saveOrder (ExpressOrderVO order); | |
| 前置条件 | 已知订单，需要将订单持久化保存 | |
| 后置条件 | 持久化保存订单信息 | |
| ExpressOrder.receiveOrder | 语法 | public ResultMessage receiveOrder(String order\_id,ReceiveInfoVO receiveInfo); | |
| 前置条件 | 订单确认收货，已知订单号和收件信息 | |
| 后置条件 | 更改订单货运状态，订单添加收件信息 | |
| 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 |
| Expressorder\_data.findCurrent(String institution\_id) | | | 查找列表持久化序列 |

表5.3.2-2 login\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Login.login | 语法 | public LogVO login(String username,String password); | |
| 前置条件 | 启动登录服务，已知用户名和密码 | |
| 后置条件 | 验证用户名和密码，返回登录信息（登录是否成功、登录权限） | |
| 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 order\_bl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.  createLoadOrder | 语法 | Public LoadOrderVO createLoadOrders(List<ExpressOrderVO> orders,LoadOrderVO loadOrderInfo) | |
| 前置条件 | 装车单制定，已知订单列表和装车信息 | |
| 后置条件 | 根据订单和装车信息生成装车单，返回装车单 | |
| Order.createArrivedOrder | 语法 | Public ArrivedOrderVO createArrivedOrder(List<ExpressOrderVO> orders,ArrivedOrderVO arrivedOrderInfo); | |
| 前置条件 | 到达单制定，已知订单列表和到达信息 | |
| 后置条件 | 根据订单和到达信息生成到达单，返回到达单 | |
| Order.createDeliveryOrder | 语法 | public DeliveryOrderVO createDeliveryOrder(List<ExpressOrderVO> orders,DeliveryOrderVO deliveryOrderInfo); | |
| 前置条件 | 派件单制定，已知订单列表和派件信息 | |
| 后置条件 | 根据订单和派件信息生成派件单，返回派件单 | |
| Order.createStockInOrder | 语法 | public StockInOrderVO createStockInOrder(List<ExpressOrderVO> orders,StockInOrderVO stockInOrderInfo); | |
| 前置条件 | 入库单制定，已知订单列表和入库信息 | |
| 后置条件 | 根据订单和入库信息生成入库单，返回入库单 | |
| Order.createStockOutOrder | 语法 | public StockOutOrderVO createStockOutOrder(List<ExpressOrderVO> orders,StockOutOrderVO stockOutOrderInfo); | |
| 前置条件 | 出库单制定，已知订单列表和出库信息 | |
| 后置条件 | 根据订单和出库信息生成出库单，返回出库单 | |
| Order.saveStockOutOrder | 语法 | public ResultMessage saveStockOutOrder(StockOutOrderVO order); | |
| 前置条件 | 已知出库单，需要将出库单持久化保存 | |
| 后置条件 | 持久化保存出库单 | |
| Order.saveStockInOrder | 语法 | public ResultMessage saveStockInOrder(StockInOrderVO order); | |
| 前置条件 | 已知入库单，需要将入库单持久化保存 | |
| 后置条件 | 持久化保存入库单 | |
| Order.saveDeliveryOrder | 语法 | public ResultMessage saveDeliveryOrder(DeliveryOrderVO order); | |
| 前置条件 | 已知派件单，需要将派件单持久化保存 | |
| 后置条件 | 持久化保存派件单 | |
| Order.saveLoadOrder | 语法 | public ResultMessage saveLoadOrder(LoadOrderVO order); | |
| 前置条件 | 已知装车单，需要将装车单持久化保存 | |
| 后置条件 | 持久化保存装车单 | |
| Order.saveArrivedOrder | 语法 | public ResultMessage saveArrivedOrders (ArrivedOrderVO order); | |
| 前置条件 | 已知到达单，需要将到达单持久化保存 | |
| 后置条件 | 持久化保存到达单 | |
| Order.findLoadOrder | 语法 | public LoadOrderVO findLoadOrder(String id); | |
| 前置条件 | 已知装车单单号 | |
| 后置条件 | 返回装车单 | |
| Order.findArrivedOrder | 语法 | public ArrivedOrderVO findArrivedOrder(String id); | |
| 前置条件 | 已知到达单单号 | |
| 后置条件 | 返回到达单 | |
| Order.findDeliveryOrder | 语法 | public DeliveryOrderVO findDeliveryOrder(String id); | |
| 前置条件 | 已知派件单单号 | |
| 后置条件 | 返回派件单 | |
| Order.findStockInOrder | 语法 | public StockInOrderVO findStockInOrder(String id); | |
| 前置条件 | 已知入库单单号 | |
| 后置条件 | 返回入库单 | |
| Order.findStockOutOrder | 语法 | public StockOutOrderVO findStockOutOrder(String id); | |
| 前置条件 | 已知出库单单号 | |
| 后置条件 | 返回出库单 | |
| 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 |
| Order\_data.add(LoadOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(DeliveryOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(StockInOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.add(StockOutOrderPO order); | | | 新增单一持久化序列 |
| Order\_data.findLoad(String id); | | | 查找单一持久化序列 |
| Order\_data.findArrived(String id); | | | 查找单一持久化序列 |
| Order\_data.findDelivery(String id); | | | 查找单一持久化序列 |
| Order\_data.findStockIn(String id); | | | 查找单一持久化序列 |
| Order\_data.findStockOut(String id); | | | 查找单一持久化序列 |
| Priceservice.getPrice(String city) | | | 返回距离的计算公式 |
| DistanceService.getDistance(String start,String end) | | | 返回距离 |
| Trans\_Fare\_data.addTransFare(TransFarePO po) | | | 在运费表中增加一条运费持久化对象 |

表5.3.2-4 receiptbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Receipt.getPaymentList | 语法 | | Public paymentListVO getPaymentList(); |
| 前置条件 | | 启动制定付款单服务 |
| 后置条件 | | 返回收款人的列表 |
| Receipt.submitPaymentList | 语法 | | Public ResultMessage submitPaymentList(); |
| 前置条件 | | 提交付款单 |
| 后置条件 | | 系统等待总经理审批，审批通过后，记录付款单信息，更新日志，修改收款人的最后收款时间，修改银行账户数据，修改运费数据条目，清零快递员的本月揽件量 |
| Receipt.paymetnExcel | 语法 | | Public ResultMessage paymetnExcel (paymentListVO); |
| 前置条件 | | 逻辑层请求导出Excel表 |
| 后置条件 | | 导出excel |
| Recept.getBillOrder | 语法 | | Public billOrderVO getBillOrder |
| 前置条件 | | 制定收款单的服务启动 |
| 后置条件 | | 逻辑层返回收款单PO |
| Recept.submitBillOrder | 语法 | | Public ResultMessage submitBillOrder() |
| 前置条件 | | 提交收款单的服务启动 |
| 后置条件 | | 系统提交收款单，等待总经理审批 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| User\_data.getPayeeInfo | | 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 | |
| User\_data.setPayeeInfo(List<UserPayeePO>) | | 修改营业员、快递员的最后付款时间，清零快递员的本月揽件量 | |
| Trans\_fare\_data.getTransFare | | 得到还未付的运费项 | |
| Trans\_fare\_data.deleteTransFare(List<TransFarePO>) | | 将已经支付的运费项删除 | |
| Receipt\_data.submitPayment(PaymentListPO) | | 将付款单具体信息提交至数据库，状态为未审批。 | |
| Receirpt\_data.getApprovedPayment(String departmentID) | | 从数据库获得本部门提交的，已经通过审批的单据 | |
| Receirpt\_data.setLocked(String departmentID) | | 将本部门所有已经通过审批的单据上锁（在上边的方法使用之后使用） | |
| Strategy\_data.getSalary | | 从数据库获得快递员提成比、营业员工资 | |
| Strategy\_data.getRent | | 从数据库获得房租的金额和付款日期 | |
| Account.trade(tradePO) | | 从银行账户扣款或收款 | |

表5.3.2-5 accountbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Account.getAccountManageList | 语法 | | Public AccountListVO getAccountManageList() |
| 前置条件 | | 启动账户管理服务 |
| 后置条件 | | 逻辑层返回银行账户的名称和余额列表 |
| Account.setAccountManageList | 语法 | | Public ResultMessage setAccountManageList(AccountListVO) |
| 前置条件 | | 财务人员确认账户修改完毕 |
| 后置条件 | | 逻辑层将调用数据库更新银行数据。 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Account\_data.add(AccountManagePO) | | 增加一条账户信息 | |
| Account\_data.delete(String name) | | 删除一条账户信息 | |
| Account\_data.Modify(AccountManagePO) | | 修改一条账户信息 | |
| Account\_data. getAccountList | | 获得银行账户列表 | |

表5.3.2-6 inventorybl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Inventory.observeStock | 语法 | | Public StockObVO observeStock(String begin,String end); |
| 前置条件 | | 请求查看库存且输入正确的起始时间 |
| 后置条件 | | 显示仓库在该时间段内出入库快递数量 |
| Inventory. countStock | 语法 | | Public StockCheckVO countStock(); |
| 前置条件 | | 请求库存盘点 |
| 后置条件 | | 根据盘点时间记录截至点，批次批号，显示库存快照 |
| Inventory. setAlarm | 语法 | | Public ResultMessage setAlarm (double alarmValue); |
| 前置条件 | | 请求设置库存警戒值并输入设置的警戒库存占用比例 |
| 后置条件 | | 更新警戒值,系统自动检测 |
| Inventory.distributeSector | 语法 | | Public ResultMessage distributeSector(String beginColumn,String endColumn,String toSector) |
| 前置条件 | | 请求调整库存分区并输入分配的区域和分配给的分区 |
| 后置条件 | | 更新机动区货架状态 |
| Inventory.initialadd | 语法 | | Public ResultMessage initialadd (GoodsVO expressorder) |
| 前置条件 | | 初始化信息时请求增加信息并输入快递信息和存放位置 |
| 后置条件 | | 保存新增信息，更新库存 |
| Inventory.initialdelete | 语法 | | Public ResultMessage initialdelete(String expressorder\_id) |
| 前置条件 | | 初始化信息时请求删除信息并输入快递编号 |
| 后置条件 | | 删除该快递信息，更新库存 |
| Inventory.initialmodify | 语法 | | Public ResultMessage initialmodify （GoosVO expressorder) |
| 前置条件 | | 初始化信息时请求更改信息并输入已有的快递编号和新的存放位置 |
| 后置条件 | | 修改该快递信息，更新库存 |
| Inventory.getArrivingOrders | 语法 | | Public List<ExpressOrderVO> getArrivingOrders(String institution\_id); |
| 前置条件 | | 已知机构id，需要获得即到达该中转中心的订单 |
| 后置条件 | | 返回到达该机构的订单 |
| Inventory.stockIn | 语法 | | Public GoodsVO stockIn(ExpressOrderVO expressorder) |
| 前置条件 | | 请求快递入库并传入入库快递信息 |
| 后置条件 | | 系统为其分配存储位置，设置入库时间，并返回这些信息，更新库存信息 |
| Inventory.createStockinOrder | 语法 | | Public StockinOrderVO createStockinOrder (List<String> expressorders\_id) |
| 前置条件 | | 入库单制定，已知入库快递列表 |
| 后置条件 | | 生成入库单，返回入库单 |
| Inventory.stockOut | 语法 | | Public void stockOut(String destination,String transportation,String expressorder\_id) |
| 前置条件 | | 请求快递出库，并输入出库快递的目的地，货运方式，和快递编号 |
| 后置条件 | | 更新库存信息 |
| Inventory.createStockoutOrder | 语法 | | Public StockoutOrderVO createStockoutOrder (List<ExpressOrderVO> expressorders) |
| 前置条件 | | 出库单制定，已知出库快递列表 |
| 后置条件 | | 生成出库单，返回出库单 |
| Inventory.getSingleOrder | 语法 | | Public ExpressOrderVO getSingleExpressOrder(String expressorder\_id) |
| 前置条件 | | 请求查看单个快递信息并输入快递编号 |
| 后置条件 | | 返回该快递存储位置和快递本身大小，目的地等信息 |
| Inventory.Alarm | 语法 | | Public String Alarm() |
| 前置条件 | | 库存占用比超出警戒值 |
| 后置条件 | | 系统发出警报信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Inventory\_data.add(ExpressOrderPO expressorder) | | 新增单一持久化对象 | |
| Inventory\_data.delete(String expressorder\_id) | | 新增单一持久化对象 | |
| Inventory\_data.modify(ExpressOrderPO expressorder) | | 修改单一持久化对象 | |
| Inventory\_data.find(String expressorder\_id) | | 根据ID查找单一持久化对象 | |
| Inventory\_data.setAlarm(int alarmValue) | | 设置单一持久化信息 | |
| Inventory\_data.getAlarm() | | 获得单一持久化信息 | |
| Order.createStockInOrder | | 生成单一持久化对象 | |
| Order.createStockOutOrder | | 生成单一持久化对象 | |

表5.3.2-7 transmanagebl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manage. getAllCar | 语法 | public CarVO[] getAllCar(String id) | |
| 前置条件 | id符合营业厅编号规范 | |
| 后置条件 | 返回该营业厅所有的车辆 | |
| Manage. getSingleCar | 语法 | public CarVO getSingleCar(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该id对应的车辆的信息 | |
| Manage. modifyCar | 语法 | public ResultMessage  modifyCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addCar | 语法 | public ResultMessage  addCar(CarVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteCar | 语法 | public ResultMessage  deleteCar(String id) | |
| 前置条件 | 对应id的车辆在数据库中 | |
| 后置条件 | 删除对应id的车辆 | |
| Manage. getAllDriver | 语法 | public DriverVO[] getAllDriver(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该机构所有的司机 | |
| Manage. getSingleDriver | 语法 | public DriverVO getSingleDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 返回该id对应的司机的信息 | |
| Manage. modifyDriver | 语法 | public ResultMessage  modifyDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addDriver | 语法 | public ResultMessage  addDriver(DriverVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteDriver | 语法 | public ResultMessage  deleteDriver(String id) | |
| 前置条件 | id>=0&&id<DriverVO[].length | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| manage. findSingleCar(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyCar(CarPO po) | | | 修改单一持久化对象 |
| manage.addCar(CarPO po) | | | 增加单一持久化对象 |
| manage.deleteCar(String id) | | | 删除单一持久化对象 |
| manage. findSingleDriver(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyDriver(DriverPO po) | | | 修改单一持久化对象 |
| manage.addDriver(DriverPO po) | | | 增加单一持久化对象 |
| manage.deleteDriver(String id) | | | 删除单一持久化对象 |

表5.3.2-8 managebl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manage. getAllInstitution | 语法 | public InstitutionVO[] getAllInstitution(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回该城市所有的机构 | |
| Manage. getSingleIns | 语法 | public InstitutionVO  getSingleIns(String id) | |
| 前置条件 | id>=0&&id<InstitutionVO[].length | |
| 后置条件 | 返回该id对应的机构的信息 | |
| Manage. modifyIns | 语法 | public ResultMessage modifyIns(InstitutionVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. addIns | 语法 | public ResultMessage  addIns(InstitutionVO vo) | |
| 前置条件 | vo存在且合法 | |
| 后置条件 | 持久化该操作 | |
| Manage. deleteIns | 语法 | public ResultMessage  deleteIns(String id) | |
| 前置条件 | id>=0&&id<InstitutionVO[].length | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory.getManage | | | 取得Manage数据库的引用 |
| manage. findSingleIns(String id) | | | 根据ID查找单一持久化对象 |
| manage. modifyIns(InstitutionPO po) | | | 修改单一持久化对象 |
| manage.addIns(InstitutionPO po) | | | 增加单一持久化对象 |
| manage.deleteIns(String id) | | | 删除单一持久化对象 |

表5.3.2-9 logbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Log.getLog | 语法 | | Public LogListVO getLog(); |
| 前置条件 | | 查看日志功能启动 |
| 后置条件 | | 无 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Log\_data.getLogList | | 得到log日志的所有记录PO | |
| Log\_data.addLog(LogPO) | | 增加一条LogPO | |

表5.3.2-10 strategybl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Strategy. getSalary | 语法 | public SalaryVO getSalary() | |
| 前置条件 | 无 | |
| 后置条件 | 返回薪水策略 | |
| Strategy. ModifySalary | 语法 | public ResultMessage modifySalary(SalaryVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| Strategy. GetAllSalary | 语法 | public SalaryVO[] getAllSalary(String id) | |
| 前置条件 |  | |
| 后置条件 | 返回所有薪水策略 | |
| Strategy. getDistance | 语法 | public DistanceVO getDistance() | |
| 前置条件 |  | |
| 后置条件 | 返回城市距离表 | |
| Strategy.  getSingleDistance | 语法 | public DistanceVO getSingleDistance(String ins\_id1,String ins\_id2); | |
| 前置条件 |  | |
| 后置条件 | 返回id1与id2对应的机构所在城市距离 | |
| Strategy. modifyDistance | 语法 | public ResultMessage modifyDistance(DistanceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| Strategy. addDistance | 语法 | public ResultMessage addDistance(DistanceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| Strategy. GetAllDistance | 语法 | public DistanceVO[] getAllDistance() | |
| 前置条件 |  | |
| 后置条件 | 返回所有城市距离 | |
| Strategy. getPriceStrategy | 语法 | public PriceVO getPriceStrategy() | |
| 前置条件 |  | |
| 后置条件 | 返回运费价格策略 | |
| Strategy. modifyPriceStrategy | 语法 | public ResultMessage modifyPriceStrategy(PriceVO vo) | |
| 前置条件 | vo合法 | |
| 后置条件 | 持久化该操作 | |
| Strategy.getRent | 语法 | public double getRent(); | |
| 前置条件 |  | |
| 后置条件 | 返回房租价格 | |
| Strategy.modifyRent | 语法 | public ResultMessage  modifyRent(double rent) | |
| 前置条件 |  | |
| 后置条件 | 持久化该操作 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory. getStrategy | | | 取得对Strategy数据库的引用 |
| StrategyDataService. findSalaryStrategy() | | | 获得单一持久化对象 |
| StrategyDataService.modifySalaryStrategy  (SalaryPO po) | | | 修改单一持久化对象 |
| StrategyDataService. findDistance() | | | 获得单一持久化对象 |
| StrategyDataService. modifyDistance  (DistancePO po) | | | 修改单一持久化对象 |
| StrategyDataService. findPriceStrategy() | | | 获得单一持久化对象 |
| StrategyDataService. modifyPrice  (PricePO po) | | | 修改单一持久化对象 |
| StrategyDataService. findRent() | | | 获得单一持久化对象 |
| StrategyDataService.  modifyRent(double rent) | | | 修改单一持久化对象 |

表5.3.2-11 approvalbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Approval. getSuperOrder | 语法 | | public SuperOrderPO getSuperOrder() |
| 前置条件 | |  |
| 后置条件 | | 返回单剧 |
| Approval. modifySuperOrderState | 语法 | | public ResultMessage  setSuperOrderState(Boolean pass) |
| 前置条件 | |  |
| 后置条件 | | 持久化该操作 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| ApprovalDataService. findOrderPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyOrderPO  (OrderPO po) | | 修改单一持久化对象 | |
| Approval. findExpressOrderPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyExpressOrderPO  (ExpressOrderPO po) | | 修改单一持久化对象 | |
| ApprovalDataService. findReceiptPO() | | 获得单一持久化对象 | |
| ApprovalDataService. modifyReceiptPO  (ReceiptPO po) | | 修改单一持久化对象 | |

表5.3.2-13 tablebl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Table.getCost\_ProfitList | 语法 | | Public Cost\_ProfitListVO getCost\_ProfitList() |
| 前置条件 | | 查看成本收益表的服务启动 |
| 后置条件 | | 无 |
| Table.getOperatingConditionList | 语法 | | Public OperatingConditionListVO getOperatingConditionList(Date begin,Date end); |
| 前置条件 | | 查看经营状况表的服务启动 |
| 后置条件 | | 无 |
| Table. Cost\_ProfitExcel | 语法 | | Public ResultMessage Cost\_ProfitExcel(Cost\_ProfitListVO) |
| 前置条件 | | 导出成本收益表的服务启动 |
| 后置条件 | | 导出表格 |
| Table. OperatingConditionExcel | 语法 | | Public ResultMessage OperatingConditionExcel(OperatingConditionListVO) |
| 前置条件 | | 导出经营状况表的服务启动 |
| 后置条件 | | 导出表格 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Receipt\_data.getReceipt(Date begin,Date end) | | 获得时间段内所有收付款单的PO表 | |
| Account\_data.getCost\_Profit() | | 获得银行成本收益的PO，包括账户名称、支出、收入 | |

表5.3.2-14 initialbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Initial.getInitialList | 语法 | | Public InitialVO getInitialList(); |
| 前置条件 | | 期初建账功能启动 |
| 后置条件 | | UI层进入期初建账界面 |
| Initial.submitInitialList | 语法 | | Public ResultMessage submitInitialList(InitialVO) |
| 前置条件 | | 完成期初建账 |
| 后置条件 | | 记录期初信息，记录日志，更新数据库的账户信息 |
| Initial.getInitialHistory | 语法 | | Public InitialHistoryVO getInitialHistory(); |
| 前置条件 | | 查看期初信息功能启动 |
| 后置条件 | | UI进入查看期初信息界面 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Inventory.getInventory | | 获得所有仓库的库存信息PO | |
| Manage.getInstitutionPO | | 获得机构、人员、车辆信息 | |
| Initial\_data.addInitial(InitialPO) | | 增加一条期初信息 | |
| Initial\_data.getHistory | | 获得期初信息历史 | |

表5.3.2-15 userbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User. getAll | 语法 | | Public List<UserVO> getAll() |
| 前置条件 | | 无 |
| 后置条件 | | 返回所有的用户信息 |
| User. getSingle | 语法 | | Public UserVO getSingle(int id) |
| 前置条件 | | id>=0&&id<userVO.length |
| 后置条件 | | 返回该ID的user信息 |
| User. add | 语法 | | Public ResultMessage add(UserVO newUserVO) |
| 前置条件 | | 输入的ID不存在，password符合规范 |
| 后置条件 | | 新增该持久化对象 |
| User. modify | 语法 | | Public ResultMessage modify(UserVO userVO) |
| 前置条件 | |  |
| 后置条件 | | 持久化该操作 |
| User. delete | 语法 | | Public ResultMessage delete(int id) |
| 前置条件 | | Id>=0&&id<userVO.length |
| 后置条件 | | 持久化该操作 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| user\_data\_service.  add(UserPO po) | | 新增单一持久化对象 | |
| user\_data\_service. modify(UserPO po) | | 修改单一持久化对象 | |
| user\_data\_service.  delete(int id) | | 删除单一持久化对象 | |
| user\_data\_service.  find(int id) | | 根据id查找单一持久化对象 | |

## 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. addDistance | 语法 | public ResultMessaage addDistance(DistancePO 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. findDistance | 语法 | public DistancePO  findDistance(String id) |
| 前置条件 |  |
| 后置条件 | 返回相应的CarPO结果 |
| StrategyDataService. modifyDistance | 语法 | public ResultMessage  modifyDistance(DistancePO 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. deleteDistance | 语法 | public ResultMessage  deleteDistance(DistancePO 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

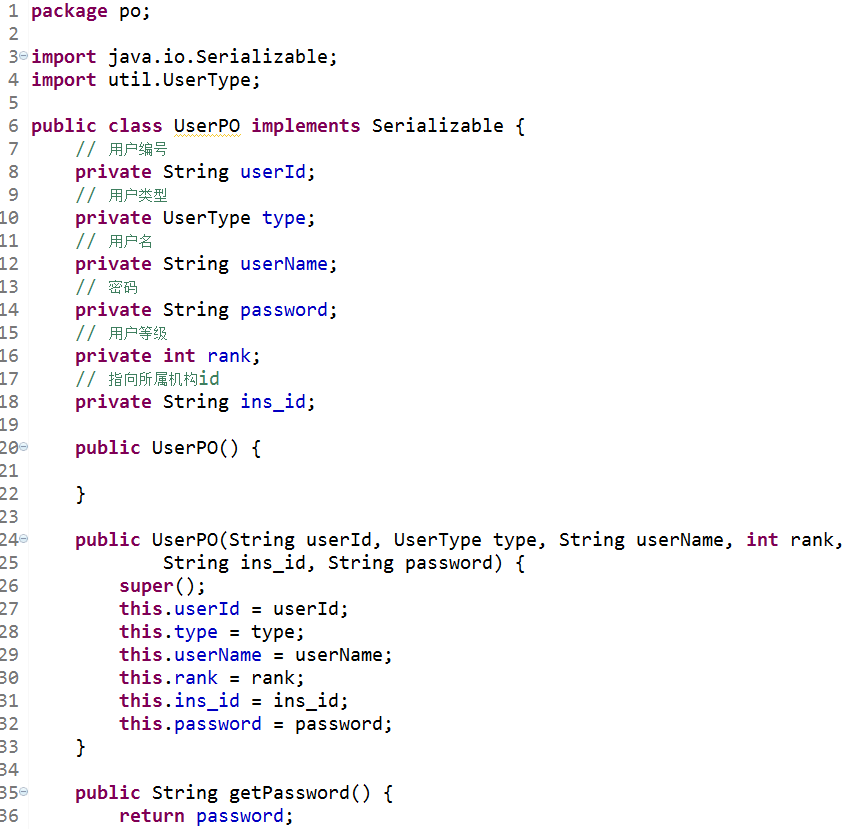
## 6.1 数据持久化对象

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



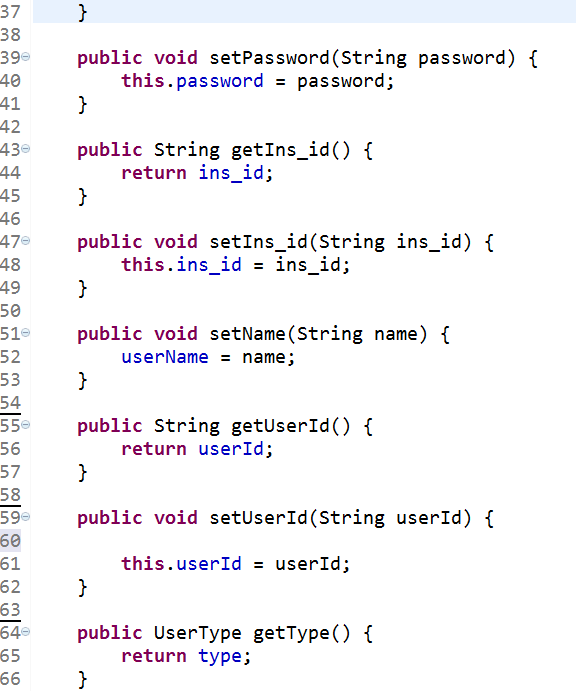
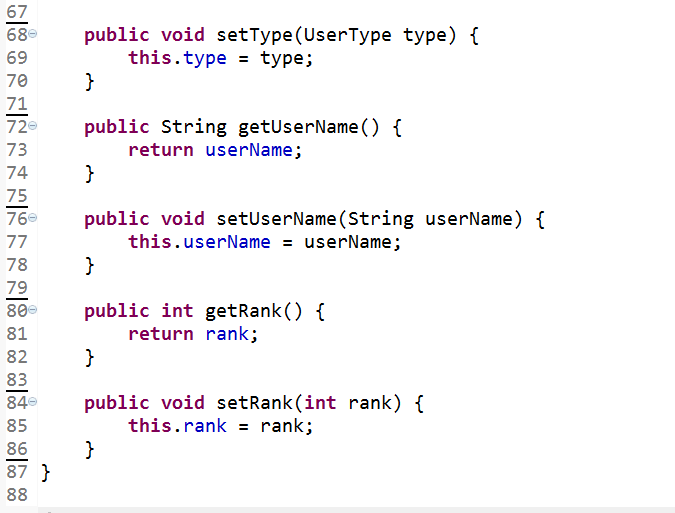
 

图6.1 持久化用户对象UserPO的定义

## 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 |