袁楚宏 袁阳阳 张鑫龙 周小帆

China Nanjing University

快递物流系统 软件体系结构描述文档

目录

[1. 引言 2](#_Toc439540754)

[1.1 编制目的 2](#_Toc439540755)

[1.2 词汇表 2](#_Toc439540756)

[1.3 参考资料 2](#_Toc439540757)

[2. 产品概述 2](#_Toc439540758)

[3. 逻辑视角 2](#_Toc439540759)

[4. 组合视角 4](#_Toc439540760)

[4.1 开发包图 4](#_Toc439540761)

[4.2 运行时进程 7](#_Toc439540762)

[4.3 物理部署 7](#_Toc439540763)

[5. 接口视角 8](#_Toc439540764)

[5.1 模块的职责 8](#_Toc439540765)

[5.2 用户界面层的分解 10](#_Toc439540766)

[5.2.1用户界面层模块的职责 11](#_Toc439540767)

[5.2.2 用户界面层的接口规范 13](#_Toc439540768)

[5.2.3 用户界面模块设计原理 23](#_Toc439540769)

[5.3 业务逻辑层的分解 23](#_Toc439540770)

[5.3.1 业务逻辑层模块的职责 24](#_Toc439540771)

[5.3.2 业务逻辑层模块的接口规范 24](#_Toc439540772)

[5.4 数据层的分解 39](#_Toc439540773)

[5.4.1 数据层模块的职责 39](#_Toc439540774)

[5.4.2 数据层模块的接口规范 39](#_Toc439540775)

[6. 信息视角 49](#_Toc439540776)

[6.1数据持久化对象 49](#_Toc439540777)

1. 引言
   1. 编制目的

本报告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. 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统 |  |
| RMI | 远程接口调用 | 一般使用JavaRMI包 |

* 1. 参考资料

1. 快递物流系统用例文档
2. 快递物流系统需求规格文档
3. 产品概述

参考快递物流系统用例文档和快递物流系统需求规格说明中对产品的概括描述。

1. 逻辑视角

快递物流系统中，选择了分层体系结构风格，将系统分为三层（界面层，逻辑层和数据层）能够很好地展示整个高层抽象。界面层包含了GUI界面的实现，逻辑层负责业务逻辑的处理，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1 参考体系结构风格的包图表达逻辑视角



图2 软件体系结构逻辑设计方案

1. 组合视角
   1. 开发包图

快递物流系统的最终开发包设计如表1所示。

表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 | userUI,transitInfoUI,orderNewUI,orderApproveUI,accountUI,cityUI,  stockUI,workOrgManUI,checkUI,VO |
| userUI | userLSer,界面类库包,VO |
| userLSer |  |
| userLogic | userLSer,userDSer,PO,utilityLogic,dataFactorySer |
| userDSer | JavaRMI,PO |
| userData | dataBaseUtility,PO,userDSer |
| transitInfoUI | transitInfoLSer,界面类库包,VO |
| transitInfoLSer |  |
| transitInfoLogic | transitLSer,transitDSer,PO,orderNewLogic,utilityLogic,dataFactroySer |
| transitInfoDSer | JavaRMI,PO |
| transitInfoData | dataBaseUtility,PO,transitDSer |
| orderNewUI | orderNewLSer,界面类库包,VO |
| orderNewSer |  |
| orderNewLogic | orderNewLSer,orderNewDSer,cityLogic,checkLogic,PO,utilityLogic,dataFactroySer |
| orderNewDSer | JavaRMI,PO |
| orderNewData | dataBaseUtility,PO, orderNewDSer |
| orderApproveUI | orderApproveLSer,界面类库包,VO |
| orderApproveSer |  |
| orderApproveLogic | orderApproveLSer,orderApproveDSer,PO,utilityLogic,dataFactroySer |
| orderApproveDSer | JavaRMI,PO |
| orderApproveData | dataBaseUtility,PO, orderApproveDSer |
| accountUI | accountLSer,界面类库包,VO |
| accountSer |  |
| accountLogic | accountLSer, accountDSer,PO,utilityLogic,dataFactroySer |
| accountDSer | JavaRMI,PO |
| accountData | dataBaseUtility,PO, accountDSer |
| cityUI | cityLSer,界面类库包,VO |
| citySer |  |
| cityLogic | cityLSer, cityDSer,PO,utilityLogic,dataFactroySer |
| cityDSer | JavaRMI,PO |
| cityData | dataBaseUtility,PO, cityDSer |
| workOrgManUI | workOrgManLSer,界面类库包,VO |
| workOrgManSer |  |
| workOrgManLogic | workOrgManLSer, workOrgManDSer,PO,utilityLogic,dataFactroySer |
| workOrgManDSer | JavaRMI,PO |
| workOrgManData | dataBaseUtility,PO, workOrgManDSer |
| stockUI | stockLSer,界面类库包,VO |
| stockSer |  |
| stockLogic | stockLSer, stockDSer,PO,utilityLogic,dataFactroySer |
| stockDSer | JavaRMI,PO |
| stockData | dataBaseUtility,PO, stockDSer |
| checkUI | checkLSer,界面类库包,VO |
| checkSer |  |
| checkLogic | checkLSer, checkDSer,PO,utilityLogic,dataFactroySer |
| checkDSer | JavaRMI,PO |
| checkData | dataBaseUtility,PO, checkDSer |

快递物流系统客户端开发包图如图3所示，服务器端开发包图如图4所示。



图3 快递物流系统客户端开发包图



图4 快递物流系统服务器端开发包图

4.2 运行时进程

在快递物流管理系统中，会有多个客户端进程和一个服务器端进程，其进程图如图所示。结合部署图，客户端进程是在客户端机器上运行，服务器端进程在服务器端机器上运行。



4.3 物理部署

快递物流管理系统中客户端构建是放在客户端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK7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。部署图如图所示。

1.81

1. 接口视角
   1. 模块的职责

客户端模块和服务器端模块分别如图7和图8所示。客户端各层和服务器端各层的职责分别如表2和表3所示。

|  |
| --- |
| 展示层 |

|  |
| --- |
| 启动模块 |

|  |
| --- |
| 逻辑层 |

|  |
| --- |
| 网络模块 |

图7 客户端模块视图

|  |
| --- |
| 启动模块 |

|  |
| --- |
| 网络模块 |

|  |
| --- |
| 数据层 |

图8 服务端模块视图

**表2 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 | 基于窗口的快递物流系统客户端用户界面 |
| 逻辑层 | 对于用户界面的输入进行响应并进行逻辑处理 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 服务端各层的职责**

|  |  |
| --- | --- |
| 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 | 负责数据持久化及数据访问接口 |
| 服务端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

每一层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4所示

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| 接 口 | 服务调用方 | 服务提供方 |
| userLSer  transitInfoLSer  orderNewLSer  orderApproveLSer  accountLSer  cityLSer  workOrgManLSer  stockLSer  checkLSer | 客户端展示层 | 客户端业务逻辑层 |
| userDSer  transitInfoDSer  orderNewDSer  orderApproveDSer  accountDSer  cityDSer  workOrgManDSer  stockDSer  checkDSer  DatabaseFactory | 客户端逻辑层 | 服务端数据层 |

借用订单信息输入用例来说明层之间的调用，如图9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）。orderNewLSer提供了orderNew界面所需要的所有业务逻辑功能。orderNewDSer提供了对数据库的增、查等操作。这样的实现就大大降低了层与层之间的耦合。



图9 订单信息输入用例层之间调用的接口

* 1. 用户界面层的分解

根据需求，系统存在52个用户界面：登录界面，物流信息查询界面，快递员分界面，营业厅业务员分界面，中转中心业务员分界面，财务人员分界面，总

经理分界面，订单创建界面，订单查询界面，收件输入界面，营业中心装车单创建界面，营业中心接收单创建界面，派件单创建界面，收款单创建界面，车辆信息管理界面，车辆信息列表显示界面，车辆信息新增修改界面，司机信息管理界面，司机信息列表显示界面，司机信息新增修改界面，中转中心装运单创建界面，中转中心接收单创建界面，中转中心装车单创建界面，库存管理界面，库存报警界面，入库单创建界面，出库单创建界面，库存查看界面，库存盘点界面，库存分区调整界面，账户管理界面，账户信息查看界面，账户信息修改界面，收款单查看界面，付款单创建界面，成本收益表创建界面，经营情况表创建界面，期初建账界面，期初信息查询界面，薪水策略管理界面，薪水策略制定界面，薪水策略查看界面，机构管理界面，人员管理界面，城市管理界面，单据审批界面，统计分析查看界面，日志记录查询界面，系统用户管理界面，系统用户信息修改界面，系统用户信息查看界面。界面跳转如图所示。



服务器端无用户界面。用户界面类图如图所示。



图11 用户界面类

5.2.1用户界面层模块的职责

如表所示为用户界面层模块的职责。

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 |
| FunctionFrame | 功能Frame, 负责所有功能panel的显示 |
| HintFrame | 提示信息Frame, 负责所有提示信息的显示 |
| InfoFrame | 信息Frame,负责各种信息Panel（除提示信息外）的显示 |
| AccountManPanel | 账户管理界面，负责账户管理（增、删、改）操作 |
| CountInfoPanel | 期初信息显示界面，负责查看期初信息操作 |
| CountPanel | 账的管理界面，负责账的管理操作 |
| BusiCircumPanel | 经营情况表，负责显示处理经营情况表 |
| CostBenePanel | 成本收益表，负责显示处理成本收益表 |
| DiaryListPanel | 日志记录列表，负责显示处理日志记录列表 |
| IncomeListPanel | 收款记录界面，负责收款记录界面 |
| OrderApprovePanel | 订单审批界面，负责订单审批管理操作 |
| OrderRevisePanel | 未通过单据界面 |
| CityListPanel | 城市列表界面，负责处理城市列表管理操作 |
| ConstantPanel | 常量制定界面，负责常量制定 |
| ArrivePanel | 中转中心到达单生成界面，负责中转中心到达单生成操作 |
| CenterLoadingPanel | 中转中心装车单生成界面，负责中转中心装车单生成操作 |
| ExpressPanel | 快递单生成界面，负责订单信息输入操作 |
| HallLoadingPanel | 营业厅装车单生成界面，负责营业厅装车单生成操作 |
| IncomePanel | 收款单生成界面，负责收款单生成操作 |
| PaymentPanel | 付款单生成界面，负责付款单生成操作 |
| ReceivePanel | 收件单生成界面，负责收件信息输入操作 |
| RecipientPanel | 接收单生成界面，负责接收单生成操作 |
| SendPanel | 派件单生成界面，负责派件单生成操作 |
| StockInPanel | 库存入库单界面，负责处理入库单操作 |
| StockOutPanel | 库存出库单界面，负责处理出库单操作 |
| TransitPanel | 中转单生成界面，负责中转单生成操作 |
| StockCheckNowPanel | 库存查看界面，负责显示库存当前信息 |
| StockCheckPanel | 库存查看界面，负责显示库存信息 |
| StockWarningPanel | 库存警告界面，负责显示库存警告信息界面 |
| StockDividePanel | 库存分区界面，负责处理库存中架信息的管理操作 |
| ExpressInfoInqPanel | 物流信息查询界面 |
| TransitInfoInqPanel | 物流信息查询界面，负责物流信息查询操作 |
| LoginPanel | 登录界面，负责处理登录操作 |
| UserInfoPanel | 用户信息界面，负责显示用户信息 |
| UserListPanel | 用户列表界面，负责显示所有的用户 |
| UserPanel | 用户功能界面，负责显示进入所有功能 |
| DriverInfoPanel | 司机信息管理界面，负责司机信息管理操作 |
| DriverListPanel | 司机信息列表显示界面，负责司机信息查看操作 |
| VanInfoPanel | 车辆信息管理界面，负责车辆信息管理操作 |
| VanListPanel | 车辆信息列表显示界面，负责司机信息查看操作 |
| OrgListPanel | 机构列表及信息管理界面，负责处理机构信息管理操作 |
| WageDraftPanel | 薪水制定界面，负责薪水制定操作 |
| WorkListPanel | 人员列表管理界面，负责人员列表管理操作 |
| WorkWageListPanel | 人员工资管理界面，负责人员工资管理操作 |

5.2.2 用户界面层的接口规范

UI模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HintFrame | 语法 | | public HintFrame(Result result,int x,int y,int width,int height) |
| 前置条件 | | 有需要显示的提示信息 |
| 后置条件 | | 显示相应的提示信息 |
| HintFrame | 语法 | | public HintFrame(String error,int x,int y,int width,int height) |
| 前置条件 | | 有需要显示的提示信息 |
| 后置条件 | | 显示相应的提示信息 |
| InfoFrame | 语法 | | public InfoFrame(String title) |
| 前置条件 | | 用户查看各类信息 |
| 后置条件 | | 显示相应信息的Panel |
| FunctionFrame | 语法 | | public FunctionFrame(String title) |
| 前置条件 | | 打开功能界面 |
| 后置条件 | | 显示功能界面及对应的功能Panel |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务 | |
| LoginPanel(MainFrame fr) | | 生成登录界面 | |

UserUI模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| UserListPanel | 语法 | | public UserListPanel (FunctionFrame fr,String name); |
| 前置条件 | | 用户请求查看用户列表 |
| 后置条件 | | 显示用户列表 |
| LoginPanel | 语法 | | public LoginPanel(MainFrame fr) |
| 前置条件 | | 用户打开系统 |
| 后置条件 | | 显示登录界面 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务 | |
| LoginLSer.login(String id,String password) | | 负责处理登录的接口 | |
| UserManLSer. newUser(UserVO user,String name) | | 负责新建用户的接口 | |
| UserManLSer. deleteUser(String id,String name,int index) | | 负责删除用户的接口 | |
| UserManLSer. reviseUser(UserVO user,int index,String name) | | 负责修改用户的接口 | |
| UserManLSer.findUser(String keyword) | | 负责获取用户列表和关键词检索的接口 | |

OrderNewUI模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressPanel | 语法 | public ExpressPanel(JFrame fr,ExpressVO vo) |
| 前置条件 | 用户选择生成快递单 |
| 后置条件 | 显示订单信息输入界面 |
| ExpressPanel | 语法 | public ExpressPanel(JFrame fr,String name,String userId,String orgId) |
| 前置条件 | 用户选择生成快递单 |
| 后置条件 | 显示订单信息输入界面 |
| ReceivePanel | 语法 | public ReceivePanel(JFrame fr,String org,String name) |
| 前置条件 | 用户选择收件 |
| 后置条件 | 显示收件信息输入界面 |
| PaymentPanel | 语法 | public PaymentPanel(JFrame fr,PaymentVO vo) |
| 前置条件 | 用户选择新建付款单 |
| 后置条件 | 显示付款单生成界面 |
| PaymentPanel | 语法 | public PaymentPanel(JFrame fr,String name,String userId) |
| 前置条件 | 用户选择新建付款单 |
| 后置条件 | 显示付款单生成界面 |
| HallLoadingPanel | 语法 | public HallLoadingPanel(JFrame fr,HallLoadingVO vo) |
| 前置条件 | 用户选择生成营业厅装车单 |
| 后置条件 | 显示营业厅装车单生成界面 |
| HallLoadingPanel | 语法 | public HallLoadingPanel(JFrame fr,String userName,String userId,String orgId,String org) |
| 前置条件 | 用户选择生成营业厅装车单 |
| 后置条件 | 显示营业厅装车单生成界面 |
| IncomePanel | 语法 | public IncomePanel(JFrame fr,IncomeVO vo) |
| 前置条件 | 用户选择生成收款单 |
| 后置条件 | 显示收款单生成界面 |
| IncomePanel | 语法 | public IncomePanel(JFrame fr,String orgId,String name,String userId) |
| 前置条件 | 用户选择生成收款单 |
| 后置条件 | 显示收款单生成界面 |
| RecipientPanel | 语法 | public RecipientPanel(JFrame fr,RecipientVO vo) |
| 前置条件 | 用户选择生成接收单 |
| 后置条件 | 显示接收单生成界面 |
| RecipientPanel | 语法 | public RecipientPanel(JFrame fr,String name,String orgId,String userId,String org) |
| 前置条件 | 用户选择生成接收单 |
| 后置条件 | 显示接收单生成界面 |
| SendPanel | 语法 | public SendPanel(JFrame fr,SendVO vo) |
| 前置条件 | 用户选择生成派件单 |
| 后置条件 | 显示派件单生成界面 |
| SendPanel | 语法 | public SendPanel(JFrame fr,String name,String orgId,String userId,LinkedList<String> express) |
| 前置条件 | 用户选择生成派件单 |
| 后置条件 | 显示派件单生成界面 |
| ArrivePanel | 语法 | public ArrivePanel(JFrame fr,ArriveVO arrive) |
| 前置条件 | 用户选择生成到达单 |
| 后置条件 | 显示到达单生成界面 |
| ArrivePanel | 语法 | public ArrivePanel(JFrame fr,String name,String orgId,String userId,String org) |
| 前置条件 | 用户选择生成到达单 |
| 后置条件 | 显示到达单生成界面 |
| CenterLoadingPanel | 语法 | public CenterLoadingPanel(JFrame fr,CenterLoadingVO vo) |
| 前置条件 | 用户选择生成中转中心装车单 |
| 后置条件 | 显示中转中心装车单生成界面 |
| CenterLoadingPanel | 语法 | public CenterLoadingPanel(JFrame fr,String name,String userId,String org,String orgId) |
| 前置条件 | 用户选择生成中转中心装车单 |
| 后置条件 | 显示中转中心装车单生成界面 |
| TransitPanel | 语法 | public TransitPanel(JFrame fr,TransitVO vo) |
| 前置条件 | 用户选择生成中转单 |
| 后置条件 | 显示中转单生成界面 |
| TransitPanel | 语法 | public TransitPanel(JFrame fr,String org,String name,String userId,String orgId) |
| 前置条件 | 用户选择生成中转单 |
| 后置条件 | 显示中转单生成界面 |
| StockInPanel | 语法 | public StockInPanel(JFrame fr,StockInVO vo) |
| 前置条件 | 用户选择新建入库单 |
| 后置条件 | 显示入库单界面 |
| StockInPanel | 语法 | public StockInPanel(JFrame fr,String name,String orgId,String userId) |
| 前置条件 | 用户选择新建入库单 |
| 后置条件 | 显示入库单界面 |
| StockOutPanel | 语法 | public StockOutPanel(JFrame fr,StockOutVO vo) |
| 前置条件 | 用户选择新建出库单 |
| 后置条件 | 显示出库单界面 |
| StockOutPanel | 语法 | public StockOutPanel(JFrame fr,String name,String orgId,String userId,String org) |
| 前置条件 | 用户选择新建出库单 |
| 后置条件 | 显示出库单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| \*OrderLSer.create(\*VO order) | 负责提供新建单据的逻辑接口 | |
| \*OrderLSer. getLocalHallsAndAllCenter(String orgId) | 负责获得当地营业厅和所有中转中心的接口 | |
| ArriveOrderLSer.getArriveOrders(String org) | 负责获得本机构的到达单 | |
| \*OrderLSer.getHalls() | 负责获取营业厅的逻辑接口 | |
| \*OrderLSer. getPrice(String org1,String org2,List<String> expressList) | 负责获得运费的逻辑接口 | |
| \*OrderLSer.getNextId(String orgId) | 负责获得下一个单据编号的逻辑接口 | |
| \*OrderLSer.getExpressList(String orgId) | 负责获得本机构订单的接口 | |
| ExpressOrderLSer. getPrice(String city1,String city2,Express expressKind,Parse pakKind,double weight) | 负责获得订单运费的接口 | |
| ExpressOrderLSer.getTime(String city1,String city2) | 负责获得到达天数的接口 | |
| \*OrderLSer.boolean isLegal(String id) | 负责检测单据编号是否存在的接口 | |
| \*OrderLSer. getVans(String org) | 获得营业厅车辆的接口 | |
| HallLoadingOrderLSer.getUnloadExpresses(String orgId) | 获得未装车订单的接口 | |
| \*OrderLSer.getCouriers(String id) | 获得本机构快递员的接口 | |
| \*OrderLSer.getAccount() | 获得账户的接口 | |
| PaymentOrderLSer.getNextId() | 获得付款单编号的接口 | |
| ReceiveOrderLSer.getExpress(String id) | 获得订单信息的接口 | |
| RecipientOrderLSer.isLoadValid(String id) | 检测装车单是否存在的接口 | |
| RecipientOrderLSer.expressList(String id) | 根据装车单获得订单列表的接口 | |
| RecipientOrderLSer.getComingLoadingOrder(String org) | 获得即将到达的装车单的接口 | |
| SendOrderLSer.getGoingExpress(String orgId) | 获得可派件单据的接口 | |
| \*OrderLSer. getOrgs() | 获得所有机构的接口 | |
| StockInOrderLSer.getShelves(String orgId) | 获得所有架的接口 | |
| \*OrderLSer. isExpressValid(String id) | 检测订单是否存在的接口 | |
| StockInOrderLSer.getComingExpresses(String orgId) | 获得可入库的订单接口 | |
| StockInOrderLSer.getRow(String shelf) | 获得可放置的排的接口 | |
| StockInOrderLSer. getPlace(String shelf,int row) | 获得可放置的位的接口 | |
| \*OrderLSer. isTransitValid(String id) | 检测中转单是否存在的接口 | |
| \*OrderLSer.isConveyValid(String id) | 检测装车单是否存在的接口 | |
| StockOutOrderLSer.getGoingOrders(String org,String orgId) | 获得待出库的中转单和装车单的接口 | |
| StockOutOrderLSer.getGoingExpress(String orderId,String orgId) | 获得可出库的订单接口 | |
| \*OrderLSer.getCenters() | 获得所有中转中心的接口 | |
| TransitOrderLSer. getGoingExpresses(String orgId) | 获得可以中转的订单的接口 | |
| StockCheckWarnLSer.checkWarning(String id) | 负责检测库存是否超出警戒线的接口 | |

TransitInfoUI模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransitInfoInqPanel | 语法 | public TransitInfoInqPanel (MainFrame fr) |
| 前置条件 | 用户选择物流信息查询 |
| 后置条件 | 显示物流信息查询界面 |
| ExpressInfoInqPanel | 语法 | public ExpressInfoInqPanel (JFrame fr) |
| 前置条件 | 用户打开订单查询界面 |
| 后置条件 | 显示订单查询界面 |
| 需要的接口（需接口） | | |

|  |  |
| --- | --- |
| CourierLSer.enterBarcodeCounrier(String  Barcode) | 查询订单信息界面的逻辑接口 |
| CustomerLogicSer.enterBarcodeCustomer(String  Barcode) | 查询物流轨迹和货运状态界面的逻辑接口 |

OrderApproveUI模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| OrderApprovePanel | 语法 | public OrderApprovePanel (FunctionFrame fr,String name) |
| 前置条件 | 用户选择单据审批功能管理 |
| 后置条件 | 显示订单审批界面 |
| OrderRevisePanel | 语法 | public OrderRevisePanel (JFrame fr,String userId) |
| 前置条件 | 用户选择查看未通过单据 |
| 后置条件 | 显示未通过单据界面 |

|  |
| --- |
| 需要的接口（需接口） |

|  |  |
| --- | --- |
| OrderApproveLSer.askExamine() | 提供返回单据信息列表的逻辑接口 |
| OrderApproveLSer. examine (boolean approve, ArrayList <Integer> index,String name) | 更新订单审批状态的业务逻辑接口 |
| OrderApproveLogic.chooseOrder(String id, Order kind) | 系统返回订单详情 |
| OrderResubmitLogic.getOrder(int index) | 提供返回单据信息的逻辑接口 |
| OrderResubmitLSer.getUnpassed(String id) | 提供返回未通过订单数量的逻辑接口 |
| OrderResubmitLSer.getOrderDisplay() | 提供未通过单据列表的逻辑接口 |

AccountUI模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| AccountManPanel | 语法 | public AccountManPanel(JFrame fr,String name) |
| 前置条件 | 用户选择管理账户信息 |
| 后置条件 | 显示账户管理界面 |
| CountInfoPanel | 语法 | public CountInfoPanel(CountVO count,JFrame fr) |
| 前置条件 | 用户选择查看期初信息 |
| 后置条件 | 显示期初信息界面 |
| CountPanel | 语法 | public CountPanel()FunctionFrame fr,String name |
| 前置条件 | 用户选择期初建账 |
| 后置条件 | 显示期初建账界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CountLSer.newCount(String name) | 新建一套账的界面的逻辑接口 |
| CountLSer.getCount() | 获取账的界面逻辑接口 |
| CountLSer.deleteCount(int index,String name) | 删除一套账的界面的逻辑接口 |
| AccountLSer. addAccount(AccountVO vo,String name) | 新增账户的逻辑接口 |
| AccountLSer.deleteAccount(int index,String name) | 删除账户的逻辑接口 |
| AccountLSer.reviseAccount(AccountVO vo,int index,String opname) | 修改账户信息的逻辑接口 |
| AccountLSer.searchAccount(String keywords) | 检索账户的逻辑接口 |

CityUI模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityListPanel | 语法 | public CityListPanel (JFrame fr,String name) |
| 前置条件 | 用户选择城市管理功能 |
| 后置条件 | 显示城市列表及城市信息管理界面 |
| ConstantPanel | 语法 | public ConstantPanel (JFrame fr,String name) |
| 前置条件 | 用户选择常量制定的功能 |
| 后置条件 | 显示常量制定界面 |

|  |
| --- |
| 需要的接口（需接口） |

|  |  |
| --- | --- |
| 服务名 | 服务 |

|  |  |
| --- | --- |
| CityLSer.addCity(CityVO city,String name) | 新增城市界面的逻辑接口 |
| CityLSer.reviseCity(CityVO city,int index,String name) | 修改城市信息界面的逻辑接口 |
| CityLSer.reviseCities (CitiesVO cities,String name) | 修改城市间距离信息界面的逻辑接口 |
| CityLSer.cityList() | 负责获得城市列表 |
| CityLSer.citiesList() | 负责获得城市间信息列表 |
| ConstantLSer.setConstant(ConstantVO vo,String opname) | 负责设置常量 |
| ConstantLSer.getConstant() | 负责获取常量信息 |

WorkOrgManUI模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| WageDraftPanel | 语法 | public WageDraftPanel (JFrame fr,WorkWageListPanel pa) |
| 前置条件 | 用户选择薪水制定 |
| 后置条件 | 显示薪水制定界面 |
| WorkListPanel | 语法 | public WorkListPanel (FunctionFrame fr,String name) |
| 前置条件 | 用户选择人员管理功能 |
| 后置条件 | 显示=人员列表界面 |
| WorkWageListPanel | 语法 | public WorkWageListPanel (FunctionFrame fr,String name) |
| 前置条件 | 用户选薪水制定功能 |
| 后置条件 | 显示人员薪水情况列表界面 |
| DriverInfoPanel | 语法 | public DriverInfoPanel(JFrame fr,DriverListPanel pa,DriverVO driver,String orgId) |
| 前置条件 | 用户选择管理司机信息 |
| 后置条件 | 显示司机信息界面 |
| DriverListPanel | 语法 | public DriverListPanel(FunctionFrame fr,String orgId,String name) |
| 前置条件 | 用户选择查看司机信息 |
| 后置条件 | 显示司机信息列表界面 |
| VanInfoPanel | 语法 | public VanInfoPanel(InfoFrame fr,VanListPanel father,VanVO vanvo,String orgId) |
| 前置条件 | 用户选择管理车辆信息 |
| 后置条件 | 显示车辆信息界面 |
| VanListPanel | 语法 | public VanListPanel(FunctionFrame fr,String orgId,String name) |
| 前置条件 | 用户选择查看车辆信息 |
| 后置条件 | 显示车辆信息列表界面 |
| OrgListPanel | 语法 | public OrgListPanel (JFrame fr,String name) |
| 前置条件 | 用户选择机构列表管理 |
| 后置条件 | 显示机构信息管理界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| OrgManLSer.addOrg(OrgVO org,String name); | 负责提供新建机构的逻辑接口 |
| OrgManLSer.deleteOrg(String id,String name) | 负责提供删除机构的逻辑接口 |
| OrgManLSer.reviseOrg(OrgVO org，String name) | 负责提供修改机构的逻辑接口 |
| OrgManLSer.searchOrg(String keywords) | 负责提供搜索特定机构的逻辑接口 |
| OrgManLSer.getCities() | 负责返回城市信息的逻辑接口 |
| OrgManLSer.getId (CityVO city,String kind) | 负责获取机构编号的逻辑接口 |
| WorkManLSer.addWork(WorkVO work,String name) | 负责提供新建人员的逻辑接口 |
| WorkManLSer.deleteWork(int index,String name) | 负责提供删除人员的逻辑接口 |
| WorkManLSer.reviseWork(WorkVO work,int index,String name) | 负责提供修改人员的逻辑接口 |
| WorkManLSer.searchWork(String keywords) | 负责提供搜索特定人员的逻辑接口 |
| WorkManLSer. orgList() | 负责获得机构列表的逻辑接口 |
| VanManLSer.addVan(VanVO van,String name） | 负责提供新建车辆的逻辑接口 |
| VanManLSer.deleteVan (String id,String name) | 负责提供删除车辆的逻辑接口 |
| VanManLSer.reviseVan(VanVO van,String name) | 负责提供修改车辆的逻辑接口 |
| VanManLSer.searchVan(String keyword,String hallId) | 负责提供关键词检索车辆的逻辑接口 |
| VanManLSer.getNextId (String id) | 负责获得新建车辆的编号的逻辑接口 |
| DriverManLSer.addDriver(DriverVO driver，String name) | 负责提供新建司机的逻辑接口 |
| DriverManLSer.deleteDriver(String id，String name) | 负责提供删除司机的逻辑接口 |
| DriverManLSer.reviseDriver (DriverVO driver，String name) | 负责提供修改司机的逻辑接口 |
| DriverManLSer.searchDriver(String hallid) | 负责提供根据id在数据库中查找该机构的司机信息并返回的逻辑接口 |
| DriverManLogic.getNextDriver(String hallid) | 负责获得并返回下一个司机编号 |
| WageManLSer.searchWorks(String key) | 负责返回关键字相关人员信息的逻辑接口 |
| WageManLSer.setWage (List<WageVO> wageList,int[]index,String name) | 负责薪水制定的逻辑接口 |

StockUI模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| StockWarningPanel | 语法 | public StockWarningPanel(JFrame fr,String orgId,String name) | |
| 前置条件 | 用户选择库存报警 | |
| 后置条件 | 显示库存报警界面 | |
| StockCheckPanel | 语法 | public StockCheckPanel(JFrame fr,String orgId) | |
| 前置条件 | 用户选择库存查看 | |
| 后置条件 | 显示库存查看界面 | |
| StockCheckNowPanel | 语法 | public StockNowPanel(JFrame fr,String orgId) | |
| 前置条件 | 用户选择库存盘点 | |
| 后置条件 | 显示当前库存情况 | |
| StockDividePanel | 语法 | public StockDividePanel(JFrame fr,String orgId,String name) | |
| 前置条件 | 用户选择库存分区 | |
| 后置条件 | 显示库存分区界面和架列表 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| StockCheckLSer.orderStock(Calendar start,Calendar end,String id) | | | 负责提供时间段内已排序的入库单出库单列表 |
| StockCheckNowLSer.checkNow(String id) | | | 负责提供库存盘点的服务 |
| StockCheckNowLSer.outputExcel(String location, String name) | | | 负责提供导出库存情况的接口 |
| StockDivideLSer.searchShelf(String id,String keyword) | | | 负责提供关键词检索架的接口 |
| StockDivideLSer.addShelf(ShelfVO shelf, String name,String id) | | | 负责提供添加架的接口 |
| StockDivideLSer.deleteShelf(String shelfId, String name,String id) | | | 负责提供删除架的接口 |
| StockDivideLSer.reviseShelf(ShelfVO shelf, String name,String id) | | | 负责提供修改架的接口 |
| StockDivideLSer.getNextId(String orgId) | | | 负责获取下一个架的编号 |
| StockWarningLSer.setWarning(ArrayList<Integer>warnings,,String id,String name) | | | 设置警戒线 |
| StockWarningLSer.getWarning(String id) | | | 获得警戒线 |

CheckUI模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CostBenePanel | 语法 | public CostBenePanel (JFrame fr) |
| 前置条件 | 用户选择查看成本收益表 |
| 后置条件 | 显示成本收益表界面 |
| IncomeListPanel | 语法 | public IncomeListPanel(JFrame fr) |
| 前置条件 | 用户选择查看收款信息列表 |
| 后置条件 | 显示收款信息列表界面 |
| BusiCircumPanel | 语法 | public BusiCircumPanel (JFrame fr) |
| 前置条件 | 用户选择查看经营情况表 |
| 后置条件 | 显示经营情况表界面 |
| DiaryListPanel | 语法 | public DiaryListPanel (JFrame fr) |
| 前置条件 | 用户选择查看日志列表 |
| 后置条件 | 显示经营情况表界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| DiaryLSer. seeDiary (Calendar start, Calendar end) | 负责日志记录查看的逻辑接口 |
| BusiCircumLSer.getBusiCircum(Calendar start, Calendar end) | 负责经营情况表查询的逻辑接口 |
| BusiCircumLSer.outputExcel(String name,String location ) | 负责导出经营情况表的逻辑接口 |
| CostBeneLSer.getCostBene () | 负责获取成本收益的逻辑接口 |
| IncomeListLSer.getHall() | 负责营业厅的编号和名称查询的逻辑接口 |
| IncomeListLSer.total (ArrayList<IncomeVO> income) | 负责合计的收款总额的逻辑接口 |
| IncomeListLSer.seeIncomeList(Calendar date,String id) | 负责查看相应的收款单列表 |

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对象负责处理用户管理和登录界面的业务逻辑，OrderNew对象负责新建单据界面的业务逻辑。业务逻辑层的设计如图12所示。



图12 业务逻辑层的设计

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所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| userLogic | 负责实现用户管理和登录界面所需要的服务 |
| transitInfoLogic | 负责实现物流信息查询界面所需要的服务 |
| orderNewLogic | 负责实现单据生成界面所需要的服务 |
| orderApproveLogic | 负责实现单据审批界面所需要的服务 |
| accountLogic | 负责实现账户管理界面所需要的服务 |
| cityLogic | 负责实现城市管理界面所需要的服务 |
| workOrgManLogic | 负责实现人员机构管理界面所需要的服务 |
| stockLogic | 负责实现库存管理界面所需要的服务 |
| checkLogic | 负责实现统计报表与查看日志界面所需要的服务 |

5.3.2 业务逻辑层模块的接口规范

各逻辑层模块接口规范如后表所示。

注：各模块基本都用到了DiaryUpdateLSer.addDiary (String op,String name)方法，不单独列出，服务是新增一个日志记录。

userLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginLogic.login | 语法 | public ResultMessage login(String id,String password) |
| 前置条件 | id和password均输入,选择登陆 |
| 后置条件 | 查找是否存在相应的用户，根据输入的id和password返回结果和用户信息 |
| UserManLogic.newUser | 语法 | public Result newUser(UserVO user,String name) |
| 前置条件 | 输入所有用户信息后，确认新建 |
| 后置条件 | 将新用户信息加入数据库，返回加入结果 |
| UserManLogic.deleteUser | 语法 | public Result deleteUser(String id,String name ,int index) |
| 前置条件 | 选择了一个用户，选择删除 |
| 后置条件 | 在数据库中除去相应用户的信息，返回删除结果 |
| UserManLogic.reviseUser | 语法 | public Result reviseUser(UserVO user,intinde,String name) |
| 前置条件 | 修改了用户的信息，确认 |
| 后置条件 | 在数据库中修改相应用户的信息，返回修改结果 |
| UserManLogic.findUser | 语法 | public ResultMessage findUser(String keyword) |
| 前置条件 | 用户选择查看用户列表或关键词检索 |
| 后置条件 | 返回数据库中所有用户的列表或含关键词的用户列表，包含用户的账号、姓名和所属机构 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| LoginDFacSer .getLogin() | 得到登录数据层服务 | |
| UserManDFacSer .getUserMan() | 得到用户管理数据层服务 | |
| UserManDSer.findUser(String keyword) | 根据关键词获得UserPO | |
| UserManDSer.addUser(UserPO po) | 在数据库中添加UserPO对象 | |
| UserManDSer.deleteUser(String id) | 在数据库中删除id所指向的对象 | |
| UserManDSer.reviseUser(UserPO po) | 在数据库中修改UserPO中id指向的对象的信息 | |
| UserManDSer.checkIsUsed(String id) throws RemoteException; | 检测用户名是否被使用 | |
| LoginDSer.findUser(String id,String password) | 在数据库中查找返回对应id的PO信息 | |

transitInfo模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CustomersLogic.enterBarcodeCustom | 语法 | Public ResultMessage enterBarcodeCustom(String barcode); |
| 前置条件 | barcode符合输入规则 |
| 后置条件 | 查找是否存在相应的订单，据输入的barcode返回相应订单的物流信息 |
| CourierLogic.enterBarcodeCourier | 语法 | Public ResultMessage enterBarcodeCourier(String barcode); |
| 前置条件 | barcode符合输入规则 |
| 后置条件 | 查找是否存在相应的订单，根据输入的barcode返回该订单所有信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CourierDSer.find(String  barcode) | 根据barcode进行查找订单对象 | |
| CustomerDSer.findTransit(String  barcode) | 根据barcode查找相应订单的物流信息 | |

orderNewLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ArriveOrderLogic.Create | 语法 | public Result Create(ArriveVO order) |
| 前置条件 | 到达单的各项信息已输入并确认 |
| 后置条件 | 将该新增到达单信息上传至数据库，返回上传是否成功 |
| ArriveOrderLogic.geLocalHallsAndAllCenter(String orgId) | 语法 | public ResultMessage geLocalHallsAndAllCenter(String orgId) |
| 前置条件 | 该机构id在数据中存在，用户打开到达单界面 |
| 后置条件 | 在数据库中查找本地营业厅及所有中转中心 |
| ArriveOrderLogic.getArriveOrders(String org) | 语法 | public ResultMessage getArriveOrders(String org) |
| 前置条件 | 该机构id在数据中存在，用户打开到达单界面 |
| 后置条件 | 在数据库中查找所有到达本机构的单据，并返回到达单列表 |
| CenterLoadingLogic.Create | 语法 | public Result Create(CenterLoadingVO order) |
| 前置条件 | 中转中心装车单的各项信息已输入并确认 |
| 后置条件 | 将该新增中转中心装车单信息上传至数据库，返回上传是否成功 |
| CenterLoadingLogic.getHalls | 语法 | public ResultMessage getHalls() |
| 前置条件 | 用户输入中转中心装车单界面 |
| 后置条件 | 在数据库中查找所有营业厅，并返回营业厅名称列表 |
| CenterLoadingLogic.getPrice | 语法 | public String getPrice(String org1,String org2,List<String >expressList) |
| 前置条件 | 用户打开中转中心装车单界面 |
| 后置条件 | 获取计算常量并计算运费 |
| CenterLoadingLogic.getNextId | 语法 | public ResultMessage getNextId(String id) |
| 前置条件 | 用户打开中转中心装车单界面 |
| 后置条件 | 获得新生成的装车单的编号 |
| CenterLoadingLogic.getExpressList | 语法 | public ResultMessage getExpressList(String id) |
| 前置条件 | 用户打开中转中心装车单 |
| 后置条件 | 在数据库中获取快递单列表 |
| ExpressOrderLogic.Create | 语法 | public Result Create(ExpressVO order) |
| 前置条件 | 寄件单的各项信息已输入并确认 |
| 后置条件 | 将该寄件单信息上传至数据库，返回上传是否成功 |
| ExpressOrderLogic.getPrice | 语法 | public String getPrice (String city1,String city2,Express expressKind,Parse pakKind,double weight) |
| 前置条件 | 用户已经输入出发地到达地 |
| 后置条件 | 自动计算运费并返回 |
| ExpressOrderLogic.getTime | 语法 | public int getTime (String city1,String city2) |
| 前置条件 | 快递单已经填写城市 |
| 后置条件 | 自动计算到达时间 |
| HallLoadingOrderLogic.Create | 语法 | public Result Create(HallLoadingVO order) |
| 前置条件 | 营业厅装车单的各项信息已输入并确认 |
| 后置条件 | 将该新增营业厅装车单上传至数据库，返回上传是否成功 |
| HallLoadingOrderLogic.getLocalHallAndAllCenter | 语法 | public ResultMessage getLocalHallAndAllCenter (String orgId) |
| 前置条件 | 用户打开营业厅装车单界面 |
| 后置条件 | 获得所有本地营业厅及所有中转中心 |
| HallLoadingOrderLogic.getVans | 语法 | public ResultMessage getVan(String org) |
| 前置条件 | 用户打开营业厅装车单界面 |
| 后置条件 | 查找数据库中所有车辆，返回车辆代号列表 |
| HallLoadingOrderLogic.getPrice | 语法 | public String getPrice () |
| 前置条件 | 用户输入出发地到达地 |
| 后置条件 | 计算运费并返回 |
| HallLoadingOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String orgId) |
| 前置条件 | 用户打开营业厅装车单 |
| 后置条件 | 处理数据计算新装车单编号 |
| HallLoadingOrderLogic.getUnloadExpress | 语法 | public ResultMessage getUnloadExpress (String org) |
| 前置条件 | 用户打开营业厅装车单界面 |
| 后置条件 | 得到所有未装载订单 |
| HallLoadingOrderLogic.isLegal | 语法 | public Boolean isLegal(String id) |
| 前置条件 | 用户输入装车单 |
| 后置条件 | 检查订单的非法性 |
| IncomeOrderLogic.Create | 语法 | public Result Create(IncomeVO order) |
| 前置条件 | 收款单的各项信息已输入并确认 |
| 后置条件 | 将该新增收款单信息上传至数据库，返回上传是否成功 |
| IncomeOrderLogic.getCouriers | 语法 | public ResultMessage getCouriers(String id) |
| 前置条件 | 该id对应的营业厅在数据库中存在，打开收款单界面 |
| 后置条件 | 根据id在数据库中查找对应营业厅并返回该营业厅快递员姓名列表 |
| IncomeOrderLogic.getAccount | 语法 | public ResultMessage getAccount () |
| 前置条件 | 打开收款单输入界面 |
| 后置条件 | 获取并返回账户列表 |
| IncomeOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String orgid) |
| 前置条件 | 该机构id在数据库中存在，打开收款单界面 |
| 后置条件 | 获取并返回新收款单编号 |
| IncomeOrderLogic.expressAvailable | 语法 | public ResultMessage expressAvailable (String orgid) |
| 前置条件 | 已填写快递单号 |
| 后置条件 | 快递单有效性检测 |
| PaymentLogic.Create | 语法 | public Result Create(PaymentVO order) |
| 前置条件 | 付款单的各项信息已输入并确认 |
| 后置条件 | 将该新增付款单信息上传至数据库，返回上传是否成功 |
| PaymentLogic.getAccount | 语法 | public ResultMessage getAccount () |
| 前置条件 | 开启付款单填写 |
| 后置条件 | 在数据库中查找并返回账户列表 |
| PaymentLogic.getNextId | 语法 | public ResultMessage getNextId () |
| 前置条件 | 开启一个新的付款单填写界面 |
| 后置条件 | 获得新付款单的编号 |
| ReceiveOrderLogic.Create | 语法 | public Result Create(ReceiveVO order,String org,String name) |
| 前置条件 | 收件单的各项信息已输入并确认 |
| 后置条件 | 将该新增收件信息上传至数据库，返回上传是否成功 |
| ReceiveOrderLogic.getExpress | 语法 | public ResultMessage getExpress(String id) |
| 前置条件 | 该id对应的订单信息在数据库中存在 |
| 后置条件 | 根据id在数据库中查找对应的订单信息并返回 |
| RecipientOrderLogic.Create | 语法 | public Result Create(RecipientVO order) |
| 前置条件 | 接收单的各项信息已输入并确认 |
| 后置条件 | 将该新增接收单信息上传至数据库，返回上传是否成功 |
| RecipientOrderLogic.getLocalHallAndAllCenter | 语法 | public ResultMessage getLocalHallAndAllCenter (String id) |
| 前置条件 | 打开营业厅接收单 |
| 后置条件 | 从数据层获取所有的本地营业厅和中转中心 |
| RecipientOrderLogic.isLoadValid | 语法 | public boolean isLoadValid (String id) |
| 前置条件 | 输入装车单编号 |
| 后置条件 | 检测装车单编号是否存在并返回结果 |
| RecipientOrderLogic.expressList | 语法 | public List<String> expressList (String id) |
| 前置条件 | 打开营业厅接收单 |
| 后置条件 | 从数据库中获取单据中的订单号并返回 |
| RecipientOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String orgId) |
| 前置条件 | 打开营业厅接收单 |
| 后置条件 | 获得新增的接收单的编号 |
| RecipientOrderLogic.getComingLoadingOrder | 语法 | public ResultMessage getComingLoadingOrder (String org) |
| 前置条件 | 打开接收单创建界面 |
| 后置条件 | 从数据层获取信息并返回正在到来的装车单 |
| SendOrderLogic.Create | 语法 | public Result Create(SendOrderVO order) |
| 前置条件 | 派件单的各项信息已输入并确认 |
| 后置条件 | 将该新增派件信息上传至数据库，返回上传是否成功 |
| SendOrderLogic.getCouriers | 语法 | public ResultMessage getCouriers (String id) |
| 前置条件 | 打开派件单界面 |
| 后置条件 | 获取该机构的快递员并返回 |
| SendOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String orgId) |
| 前置条件 | 打开派件单界面 |
| 后置条件 | 获取派件单编号并返回 |
| SendOrderLogic.getGoingExpress | 语法 | public ResultMessage getGoingExpress (String orgId) |
| 前置条件 | 打开派件单界面 |
| 后置条件 | 获取所有正在到来的中转单列表 |
| StockInOrderLogic.Create | 语法 | public Result Create(StockInOrderVO order) |
| 前置条件 | 入库单的各项信息已输入并确认 |
| 后置条件 | 将该新增入库信息上传至数据库，返回上传是否成功 |
| StockInOrderLogic.getOrgs | 语法 | public ResultMessage getOrgs () |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 获取机构列表并返回 |
| StockInOrderLogic.getShelves | 语法 | public ResultMessage getShelves (String orgId) |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 获取空余的架列表并返回 |
| StockInOrderLogic.isExpressValid | 语法 | public boolean isExpressValid (String id) |
| 前置条件 | 输入中转单编号 |
| 后置条件 | 检测并返回中转单编号是否合法 |
| StockInOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String id) |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 获取并返回新入库单的编号 |
| StockInOrderLogic.getComingExpress | 语法 | public ResultMessage getComingExpress (String id) |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 从数据库中获取正在到来的中转单 |
| StockInOrderLogic.getRow | 语法 | public Integer[] getRow (String shelf) |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 计算可以入库的排并返回 |
| StockInOrderLogic.getPlace | 语法 | public Integer[] getPlace (String id,int row) |
| 前置条件 | 打开入库单界面 |
| 后置条件 | 计算可以入库的位并返回 |
| StockOutOrderLogic.Create | 语法 | public Result Create(StockoutOrderVO order) |
| 前置条件 | 出库单信息已填写并确认 |
| 后置条件 | 将该新增出库信息上传至数据库，返回上传是否成功 |
| StockOutOrderLogic.getOrgs | 语法 | public Result getOrgs () |
| 前置条件 | 打开出库单界面 |
| 后置条件 | 获取所有机构列表 |
| StockOutOrderLogic.isExpressValid | 语法 | public Result isExpressValid (String id) |
| 前置条件 | 填写快递单 |
| 后置条件 | 检查快递单是否有效并返回 |
| StockOutOrderLogic.isTransitValid | 语法 | public Result isTransitValid (String id) |
| 前置条件 | 填写中转单 |
| 后置条件 | 检查中转单是否有效并返回 |
| StockOutOrderLogic.getNextId | 语法 | public ResultMessage getNextId (String orgId) |
| 前置条件 | 打开出库单界面 |
| 后置条件 | 从数据层获取并返回订单编号 |
| StockOutOrderLogic.isConveyValid | 语法 | public Result isConveyValid (String id) |
| 前置条件 | 填写装车单 |
| 后置条件 | 检查并返回装车单的有效性 |
| StockOutOrderLogic.getGoingOrders | 语法 | public Result getGoingOrders (String org,String orgId) |
| 前置条件 | 打开出库单界面 |
| 后置条件 | 从数据层获取并返回正在到来的订单 |
| StockOutOrderLogic.getGoingExpress | 语法 | public Result getGoingExpress (String orderId,String orgId) |
| 前置条件 | 从数据层获取并返回正在到来的中转单 |
| 后置条件 | 获得待出库订单列表并返回 |
| TransitOrderLogic.Create | 语法 | public Result Create(TransitOrderVO order) |
| 前置条件 | 中转单的各项信息已输入并确认 |
| 后置条件 | 将该新增中转单信息上传至数据库，返回上传是否成功 |
| TransitOrderLogic.getCenter | 语法 | public ResultMessage getCenters () |
| 前置条件 | 打开中转单 |
| 后置条件 | 获取中转中心列表并返回 |
| TransitOrderLogic.isExpressValid | 语法 | public int isExpressValid (List<String >expressList) |
| 前置条件 | 输入快递单编号 |
| 后置条件 | 检查并返回快递单是否有效 |
| TransitOrderLogic.getPrice | 语法 | public ResultMessage getPrice(String org1,String org2,int tran,List<String>expressList) |
| 前置条件 | 已经输入出发地到达地 |
| 后置条件 | 计算并返回运费 |
| TransitOrderLogic.getNextId | 语法 | public ResultMessage getNextId (TransitOrderVO order) |
| 前置条件 | 打开中转单界面 |
| 后置条件 | 得到并返回新中转单的编号 |
| TransitOrderLogic.getGoingExpress | 语法 | public ResultMessage getGoingExpress (TransitOrderVO order) |
| 前置条件 | 打开中转单界面 |
| 后置条件 | 得到并返回可以中转的快递单列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| \*OrderDFacSer.get\*Order() | 得到数据层服务的引用 | |
| \*OrderDSer.create(\*PO order ) | 在数据库中添加对应订单\*PO对象 | |
| \*OrderDSer. getOderSize(int partrId) | 根据单据中机构编号和日期部分在数据库中获取并返回当天单据数量 | |
| ReceiveOrderDSer. getExpress(String id) | 在数据库中获取并返回快递单PO | |
| SendOrderDSer.getCouriers() | 在数据库中获取并返回快递员信息 | |
| StockInOrderDSer.getShelves() | 在数据库中获取并返回架信息列表 | |
| StockInOrderDSer.addGood(GoodPO good) | 在数据库中添加货物PO的储存信息 | |
| StockOutOrderDSer.removeGood(String id) | 在数据库中删除货物PO并返回结果 | |
| UpdateTransitOrderDSer.addInfo(TransitInfoPO info) | 创建快递的物流信息并返回结果 | |
| UpdateTransitOrderDSer.update(TransitInfoPO info) | 在该快递物流信息中添加新的物流轨迹 | |
| UpdateTranStateDSer. updateTranState(UpdateTranState order) | 更新单据的物流状态并返回结果 | |
| UpdateTranStateDSer.update (LocationFind order) | 根据当前机构ID从数据库中获得所有订单列表并返回 | |

workOrgManLogic模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrgManLogic.addOrg | 语法 | | public Result addOrg(OrgVO org,String name); |
| 前置条件 | | 用户输入了新机构的信息 |
| 后置条件 | | 将新增的机构信息加入数据库，返回加入结果 |
| OrgManLogic.deleteOrg | 语法 | | public Result deleteOrg(String id,String name) |
| 前置条件 | | 用户选择一个机构进行删除 |
| 后置条件 | | 根据机构名在数据库中删除该账户信息，返回删除结果 |
| OrgManLogic.reviseOrg | 语法 | | public Result reviseOrg(OrgVO org，String name) |
| 前置条件 | | 修改了机构信息并确认 |
| 后置条件 | | 更新数据库中该机构的信息，返回更新结果 |
| OrgManLogic.searchOrg | 语法 | | public ResultMessage searchOrg(String keyword) |
| 前置条件 | | 用户打开机构管理或检索机构并输入关键字 |
| 后置条件 | | 返回所有机构或根据该关键字查找符合条件的机构并返回 |
| OrgManLogic.getCities | 语法 | | public ResultMessage getCities () |
| 前置条件 | | 用户打开机构管理 |
| 后置条件 | | 返回城市间信息 |
| OrgManLogic.getId | 语法 | | public String getId (CityVO city,String kind) |
| 前置条件 | | 用户新增一个机构 |
| 后置条件 | | 从数据库中获取机构编号 |
| WorkManLogic.addWork | 语法 | | public Result addWork(WorkVO work,String name) |
| 前置条件 | | 用户输入了新人员信息 |
| 后置条件 | | 将新增的人员信息加入数据库，返回加入结果 |
| WorkManLogic.deleteWork | 语法 | | public Result deleteWork(int index,String name) |
| 前置条件 | | 用户选择一个人员进行删除 |
| 后置条件 | | 根据人员名在数据库中删除该人员信息，返回删除结果 |
| WorkManLogic.reviseWork | 语法 | | public Result reviseWork(WorkVO work, int index,String name) |
| 前置条件 | | 修改了人员信息并确认 |
| 后置条件 | | 更新数据库中该人员的信息，返回更新结果 |
| WorkManLogic.searchWork | 语法 | | public ResultMessage searchWork(String keyword) |
| 前置条件 | | 用户打开人员管理或输入关键字检索 |
| 后置条件 | | 返回全部人员列表或根据该关键字查找符合条件的人员并返回 |
| WorkManLogic. orgList | 语法 | | public ResultMessage orgList() |
| 前置条件 | | 用户打开人员管理或输入关键字检索 |
| 后置条件 | | 返回机构列表 |
| DriverManLogic.addDriver | 语法 | | public Result addDriver(DriverVO driver，String name) |
| 前置条件 | | 用户选择添加司机信息 |
| 后置条件 | | 在数据库中增加该司机信息，并返回添加结果 |
| DriverManLogic.deleteDriver | 语法 | | public Result deleteDriver(String id,String name) |
| 前置条件 | | 该id对应的司机信息在数据库中存在 |
| 后置条件 | | 根据id在数据库中删除该司机信息，并返回删除结果 |
| DriverManLogic.reviseDriver | 语法 | | public Result reviseDriver(DriverVO driver,String name) |
| 前置条件 | | 该司机在数据库中存在 |
| 后置条件 | | 在数据库中更新该司机信息，并返回更新结果 |
| DriverManLogic.searchDriver | 语法 | | public ResultMessage SearchDriver (String hallid) |
| 前置条件 | | 该ID对应的营业厅存在 |
| 后置条件 | | 根据id在数据库中查找机构所属司机信息并返回 |
| DriverManLogic.getNextDriver | 语法 | | public ResultMessage getNextDriver (String hallid) |
| 前置条件 | | 新建司机信息 |
| 后置条件 | | 在数据库中获得并返回下一个司机编号 |
| VanManLogic.addVan | 语法 | | public Result addVan (VanVO van,String name) |
| 前置条件 | | 用户选择添加车辆信息 |
| 后置条件 | | 在数据库中增加该车辆信息，并返回添加结果 |
| VanManLogic.deleteVan | 语法 | | public Result deleteVan (String id,String name) |
| 前置条件 | | 该id对应的车辆信息在数据库中存在 |
| 后置条件 | | 根据id在数据库中删除该车辆信息，并返回删除结果 |
| VanManLogic.reviseVan | 语法 | | public Result reviseVan (VanVO van,String name) |
| 前置条件 | | 该车辆在数据库中存在 |
| 后置条件 | | 在数据库中更新该车辆信息，并返回更新结果 |
| VanManLogic.getNextId | 语法 | | public ResultMessage getNextId (String hallId) |
| 前置条件 | | 新增车辆信息 |
| 后置条件 | | 从数据库中获取并返回车辆id |
| VanManLogic.searchVan | 语法 | | public ResultMessage searchVan(String keyword,String hallId) |
| 前置条件 | | 用户打开车辆管理界面或输入关键字检索 |
| 后置条件 | | 返回全部车辆列表或根据该关键字查找符合条件的车辆并返回车辆信息列表 |
| WageManLogic.setWage | 语法 | | public Result setWage (List<WageVO> wageList,int[]index,String name) |
| 前置条件 | | 用户选择设置薪水 |
| 后置条件 | | 在数据库中更新该薪水信息，并返回更新结果 |
| WageManLogic. searchWorks | 语法 | | public ResultMessage searchWorks (String keyword) |
| 前置条件 | | 逻辑层需要搜索相关人员信息 |
| 后置条件 | | 在数据库找到并根据关键字返回人员信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| WageManDSer.searchWork(String keyword) | | 从数据库中找到含有该关键字的人员，返回数据 | |
| WageManDFacSer.getWageMan() | | 得到薪水管理数据层服务的引用 | |
| WorkManDSer.addWork(WorkPO po) | | 在数据库中新增一个人员的数据 | |
| WorkManDSer.deleteWork(String id) | | 在数据库中删除一个人员的数据 | |
| WorkManDSer.reviseWork(WorkPO po) | | 在数据库中修改一个人员的数据 | |
| WorkManDSer.findWork(String keyword) | | 从数据库中找到含有该关键字的人员，返回数据 | |
| WagekManDSer.updateWork(WorkPOwork) | | 在数据库中更新人员的薪资信息 | |
| WorkManDFacSer. getWorkMan() | | 得到人员管理数据层的服务的引用 | |
| OrgManDSer.findOrg(String keyword) | | 在数据库中找到含有该关键字的机构，返回数据 | |
| OrgManDSer.addOrg(OrgPO po) | | 在数据库中新增一个机构的信息 | |
| OrgManDSer.deleteOrg(String id) | | 在数据库中删除一个机构的信息 | |
| OrgManDSer.reviseOrg(OrgPO po) | | 在数据库中修改一个机构的信息 | |
| OrgManDFacSer.getOrgMan() | | 得到机构管理数据层服务的引用 | |
| DriverManDSer.addDriver(DriverPO driver) | | 在数据库中新增一个司机的数据 | |
| DriverManDSer.deleteDriver (String id) | | 在数据库中删除一个司机的数据 | |
| DriverManDSer.reviseDriver(DriverPO driver) | | 在数据库中修改一个司机的数据 | |
| DriverManDSer.findDriver (String keyword) | | 在数据库中找到含有该关键字的司机，返回数据 | |
| DriverManDFacSer.getDriverMan() | | 得到司机管理数据层的服务的引用 | |
| VanManDSer.findVan(String keywords) | | 从数据库中找到含有该关键字的车辆，返回数据 | |
| VanManDSer.addVan (VanPO van) | | 在数据库中新增一个车辆的数据 | |
| VanManDSer.deleteVan (String id) | | 在数据库中删除一个车辆的数据 | |
| VanManDSer.reviseVan(VanPO van) | | 在数据库中修改一个车辆的数据 | |
| VanManDFacSer.getVanMan() | | 得到汽车管理数据层的服务的引用 | |
| UtilityLSer.getOrgs() | | 获得机构名称列表 | |
| UtilityLSer.orgs() | | 获得机构PO列表 | |

cityLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityLogic.addCity | 语法 | public Result addCity(CityVO city,String name); |
| 前置条件 | 增加一个城市 |
| 后置条件 | 将该城市信息加入数据库，返回添加结果 |
| CityLogic.reviseCity | 语法 | public Result reviseCity(CityVO city,int index,String name) |
| 前置条件 | 修改了一个城市信息 |
| 后置条件 | 更新数据库已有信息，返回修改结果 |
| CityLogic.reviseCities | 语法 | public Result reviseCities(CitiesVO cities,String name) |
| 前置条件 | 总经理已经选定了两个城市 |
| 后置条件 | 将两个城市间的距离和价格信息加入数据库或更新数据库中已有的距离与价格信息 |
| CityLogic.deleteCitiy | 语法 | public Result deleteCity(String cityName,String name) |
| 前置条件 | 总经理已经选定了城市 |
| 后置条件 | 从数据库中删除城市信息及与城市有关的信息 |
| CityLogic.citiesList | 语法 | public ResultMessage citiesList() |
| 前置条件 | 打开城市管理界面 |
| 后置条件 | 从数据库中获取并返回城市间信息列表 |
| City.cityList | 语法 | public ResultMessage cityList(); |
| 前置条件 | 需要获取城市对象 |
| 后置条件 | 查找数据库中所有城市并返回城市对象列表 |
| ConstantLogic.setConstant | 语法 | public Result setConstant (ConstantVO city,String opname); |
| 前置条件 | 选择确定设置常量 |
| 后置条件 | 更新序列化文件中已有常量信息 |
| ConstantLogic.getConstant | 语法 | public ResultMessage getConstant (); |
| 前置条件 | 选择确定设置常量 |
| 后置条件 | 从序列化文件中获取常量信息并返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CityDSer.addCity(CityPO po) | 新增城市对象 | |
| CityDSer.reviseCity(CityPO city) | 更新城市对象信息 | |
| CityDSer.deleteCity(String cityname) | 在数据库中删除该city | |
| CityDSer.addCities(CitiesPO cities) | 新增城市间信息对象 | |
| CityDSer.deleteCities(CitiesPO cities) | 在数据库中删除该citiesPO信息 | |
| CityDSer.reviseCities(CitiesPO cities) | 更新城市之间距离和价格 | |
| CityDSer.getCitiesInfo () | 返回城市间信息列表 | |
| CityDSer.isCityUsed(String name,String id) | 检查城市名和区号是否被使用过 | |
| CityDFacSer.getCity() | 获得城市管理数据层服务的引用 | |
| ConstantDSer.getConstant() | 获取并返回常量信息 | |
| ConstantDSer.setConstant(ConstantPO constant) | 更新常量信息 | |

accountLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CountLogic.newCount | 语法 | public Result newCount(String name) |
| 前置条件 | 用户新建一套账 |
| 后置条件 | 将新增的期初信息序列化存储，返回加入结果 |
| CountLogic.deleteCount | 语法 | public Result deleteCount(int index,String name) |
| 前置条件 | 用户选择删除期初信息 |
| 后置条件 | 删除期初信息对应的序列化文件并返回结果 |
| CountLogic.getCount | 语法 | public ResultMessage getCount() |
| 前置条件 | 用户打开期初信息查看界面 |
| 后置条件 | 返回期初信息 |
| AccountLogic.addAccount | 语法 | public Result addAccount(AccountVO account，String opname); |
| 前置条件 | 用户输入了新账户信息 |
| 后置条件 | 将新增的账户信息加入数据库，返回加入结果 |
| AccountLogic.deleteAccount | 语法 | public Result deleteAccount(int index,String opname) |
| 前置条件 | 用户选择一个账户进行删除 |
| 后置条件 | 根据账户名在数据库中删除该账户信息，返回删除结果 |
| AccountLogic.reviseAccount | 语法 | public Result reviseAccount(AccountVO account,int index,String opname) |
| 前置条件 | 修改了账户信息并确认 |
| 后置条件 | 更新数据库中该账户的信息，返回更新结果 |
| AccountLogic.searchAccount | 语法 | public ResultMessage searchAccount(String keywords) |
| 前置条件 | 用户打开账户管理界面或输入关键字检索 |
| 后置条件 | 返回全部账户列表并根据该关键字查找符合条件的账户并返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AccountDSer.findAccount(String keyword) | 在数据库中查找含有关键词的账户，返回数据 | |
| AccountDSer.addAccount(AccountPO  po) | 在数据库中新增一个账户 | |
| AccountDSer.deleteAccount(int id) | 在数据库中删除一个账户 | |
| AccountDSer.reviseAccount(AccountPO po) | 在数据库中修改一个账户的信息 | |
| AccountDFacSer.getAccountData() | 得到账户管理数据层的服务的引用 | |
| CountDSer.addCount(CountPO po) | 新增一个期初信息的序列化存储 | |
| CountDSer.getCount() | 获得所有的期初信息 | |
| CountDSer.deleteCount(int id) | 删除id对应的期初信息序列化文件 | |
| CountDFacSer. getCountData() | 得到期初建账数据层服务的引用 | |
| UtilityLSer. orgs() | 得到所有机构信息 | |
| UtilityLSer.works(String orgId) | 得到所有人员信息 | |
| UtilityLSer.stocks() | 得到所有库存信息 | |
| UtilityLSer.vans(String orgId) | 得到所有车辆信息 | |
| UtilityLSer.accounts() | 得到所有账户信息 | |

stockLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockWarningLogic.setWarning | 语法 | Public Result setWarning (ArrayList <Integer> warnings, String id ,String name) |
| 前置条件 | 用户输入警戒线比例并提交 |
| 后置条件 | 修改序列化文件中的警戒线比例，并返回修改是否成功 |
| StockWarningLogic.getWarning | 语法 | public ResultMessage getWarning(String id) |
| 前置条件 | 用户选择了库存报警 |
| 后置条件 | 在序列化文件中获得各区警戒线比例，返回值 |
| StockCheckWarnLogic.checkWarning | 语法 | public ResultMessage checkWarning(String id) |
| 前置条件 | 用户新建了入库单 |
| 后置条件 | 对比数据库中库存数/库存容量和序列化的警戒线比例，返回结果 |
| StockCheckNowLogic.checkNow | 语法 | public ResultMessage checkNow(String id) |
| 前置条件 | 用户选择了库存盘点 |
| 后置条件 | 从数据库中获取当前库存情况，返回结果 |
| StockCheckNowLogic.outputExcel | 语法 | public Result outputExcel(String location,String name) |
| 前置条件 | 用户选择导出库存情况和导出位置 |
| 后置条件 | 将生成的Excel导出到相应位置，返回导出是否成功 |
| StockDivideLogic.shelfList | 语法 | public ResultMessage shelfList(String id) |
| 前置条件 | 用户选择了库存分区 |
| 后置条件 | 读取数据库中的架信息，返回架信息 |
| StockDivideLogic.addShelf | 语法 | public Result addShelf(ShelfVO shelf，String name, String orgId) |
| 前置条件 | 用户输入了新建的架的所有信息，确认新建 |
| 后置条件 | 在数据库中添加新建的架的信息，返回是否新建成功 |
| StockDivideLogic.deleteShelf | 语法 | public Result deleteShelf(String id,String name) |
| 前置条件 | 用户选择了一个架，选择删除 |
| 后置条件 | 在数据库中去除相应架的信息，返回是否删除成功 |
| StockDivideLogic.reviseShelf | 语法 | public Result reviseShelf(ShelfVO shelf,String name,String orgId) |
| 前置条件 | 用户修改了架信息，并确认 |
| 后置条件 | 在数据库中修改相应架的信息，返回是否修改成功 |
| StockDivideLogic.searchShelf | 语法 | public ResultMessage searchShelf(String id，String keyword ) |
| 前置条件 | 用户输入了关键词，选择搜索 |
| 后置条件 | 在数据库中找到含有关键词的架的信息，返回信息 |
| StockDivideLogic.getNextId | 语法 | public String getNextId (String orgId) |
| 前置条件 | 用户新增架 |
| 后置条件 | 计算并返回新增的架的编号 |
| StockCheckLogic.orderStock | 语法 | public ResultMessage checkStockIn(Calendar start, Calendar end,String id) |
| 前置条件 | 用户输入了开始和结束时间 |
| 后置条件 | 在数据库中按照开始时间和结束时间，查找所有出入库信息并按时间排序后返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| StockWarningDSer.getWarning (String id) | 从序列化文件中获得ID相应仓库的警戒线比例 | |
| StockWarningDSer.setWarning(List<Integer> warnings,String id) | 在序列化文件中设置ID相应仓库和相应区的警戒线比例 | |
| StockWarningDFacSer. .getStockWarning() | 获得库存报警数据层服务的引用 | |
| StockCheckDSer.getIn(Calendar start, Calendar end,String id) | 在数据库中获得ID相应仓库开始时间和结束时间之间的入库单 | |
| StockCheckDSer.getOut (Calendar start, Calendar end, String id) | 在数据库中获得ID相应仓库开始时间和结束时间之间的出库单 | |
| StockCheckDFacSer.getStockCheck() | 获得库存查看数据层服务的引用 | |
| StockCheckNowDSer.getStock(String id) | 在数据库中获得ID相应仓库当前库存的情况 | |
| StockCheckNowDFacSer.getStockCheckNow() | 获得库存盘点数据层服务的引用 | |
| UtiltiyLSer.outputExcel(String[][] data, String name, String location) | 导出Excel表格 | |
| StockDivideDSer.getShelves(String stockId,String keyword) | 在数据库中获得ID相应仓库含有关键词的当前架的情况 | |
| StockDivideDSer.addShelf(ShelfPO shelf) | 在数据库中ID相应仓库添加一个架的信息 | |
| StockDivideDSer.deleteShelf(String id) | 在数据库中ID相应仓库删除一个架的信息 | |
| StockDivideDSer.reviseShelf(ShelfPO po) | 在数据库中ID相应仓库修改一个架的信息 | |
| StockDivideDFacSer.getStockDivide() | 获得库存分区数据层服务的引用 | |
| StockCheckWarnDFacSer.getStockCheckWarn() | 获得检测库存报警数据层服务的引用 | |
| StockCheckWarnDSer.getWarning(String id) | 获得仓库警戒线 | |
| StockCheckWarnDSer getStock(String stockId) | 获得当前库存情况 | |
| StockCheckWarnDSer getShelves(String id) | 获得仓库架的情况 | |

checkLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DiaryLogic.seeDiary | 语法 | public ResultMessage seeDiary(Calendar start, Calendar end) |
| 前置条件 | 输入的日期符合输入规则 |
| 后置条件 | 根据输入的开始和结束日期，返回相应的日志记录 |
| BusiCircumLogic.getBusiCircum | 语法 | public ResultMessage getBusiCircum (Calendar start, Calendar end) |
| 前置条件 | 输入的日期符合输入规则 |
| 后置条件 | 根据输入的开始和结束日期，返回由入款单和收款单信息组成的经营情况表 |
| BusiCircumLogic.outputExcel | 语法 | public boolean outputExcel(String name, String location) |
| 前置条件 | 用户选择地址后确认导出经营情况表 |
| 后置条件 | 导出经营情况表到相应地址 |
| CostBeneLogic.getCostBene | 语法 | public ResultMessage getCostBene() |
| 前置条件 | 用户开启成本收益表查看 |
| 后置条件 | 计算总收入、总支出、总利润并返回 |
| IncomeListLogic.seeIncomeList | 语法 | public ResultMessage seeIncomeList(Calendar date, String id) |
| 前置条件 | 输入了日期，选择了营业厅确认获取收款记录 |
| 后置条件 | 根据输入的日期，返回当日的收款单列表 |
| IncomeListLogic.total | 语法 | public ResultMessage total(ArrayList<IncomeVO> income) |
| 前置条件 | 用户选择合计收款 |
| 后置条件 | 返回合计的收款总额 |
| IncomeListLogic.getHall | 语法 | public ResultMessage getHall() |
| 前置条件 | 用户选择查看经营情况表 |
| 后置条件 | 返回所有营业厅的名称 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DiaryDSer.seeDiary(Calendar start, Calendar end) | 根据开始和结束日期从数据库中获得日志记录，返回数据 | |
| DiaryDFacSer .getDiary() | 获得日志查看数据层服务的引用 | |
| IncomeListDSer.getCollectRec (Calendar date, String id) | 根据给的日期和营业厅编号从数据库中获得当日该营业厅收款记录，返回数据 | |
| IncomeListDFacSer .getIncomeList() | 获得查看收款记录数据层服务的引用 | |
| UtiltityLSer.getHall() | 获得所有营业厅列表 | |
| BusiCircumDSer.findIncome(Calendar start，Calendar end) | 根据开始结束时段从数据库中获取并返回所有的收款记录 | |
| BusiCircumDSer.findPayment(Calendar start，Calendar end) | 根据开始结束时段从数据库中获取并返回所有的付款记录 | |
| BusiCircumDFacSer .getBusiCircum() | 获得经营情况表数据层服务的引用 | |
| UtiltityLSer.outputExcel(String[][] data, String name, String location) | 导出Excel表格 | |
| CostBeneDSer.getIncomes() | 从数据库中获取并返回所有的收款记录 | |
| CostBeneDSer.getPayments() | 从数据库中获取并返回所有的付款记录 | |
| CostBeneDFacSer. .getCostBene() | 获得成本收益表数据层服务的引用 | |

orderApprove模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveLogic.askExamine | 语法 | public ResultMessage askExamine() |
| 前置条件 | 用户请求审批单据 |
| 后置条件 | 返回待审批单据列表 |
| OrderApproveLogic.examine | 语法 | public Result examine(boolean approve, ArrayList <Integer> indexs,String name) |
| 前置条件 | 用户决定单据是否通过审批 |
| 后置条件 | 返回审批结果，更新相应状态 |
| OrderApproveLogic.chooseOrder | 语法 | public ResultMessage chooseOrder(String id, Order kind) |
| 前置条件 | 用户选择一个订单，查看详情 |
| 后置条件 | 系统返回订单详情 |
| OrderResubmitLogic.getUnpassed | 语法 | public ResultMessage getUnpassed(String id) |
| 前置条件 | 用户登录后，一段时间轮询一次 |
| 后置条件 | 返回未通过单据的数量 |
| OrderResubmitLogic.getOrdersDisplay | 语法 | public ResultMessage getOrdersDisplay () |
| 前置条件 | 用户选择查看未通过单据 |
| 后置条件 | 返回未通过单据列表 |
| OrderResubmitLogic.getOrder | 语法 | public ResultMessage getOrder (int index) |
| 前置条件 | 用户选择一个订单，查看详情 |
| 后置条件 | 通过传入参数得到列表中单据在表中位置，返回单据的vo |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderApproveDSer.update(Boolean isPassed, ArrayList<String>id, Order kind) | 更新订单审批状态 | |
| OrderApproveDSer.getOrder(String id,Order kind) | 从数据库中获取相应单据的信息，返回。 | |
| OrderApproveDSer.getUser(String userId) | 根据用户id从数据库获得并返回相应的用户信息 | |
| OrderApproveDSer.getExamine(String id) | 从数据库中获取待审批的单据信息，返回。 | |
| OrderApproveDFacSer .getOrderApprove() | 获得单据审批数据层服务的引用 | |
| OrderResubmitDSer.getOrders(String userId) | 根据用户ID从数据库获得并返回相应的未通过单据 | |
| OrderResubmitDFacSer. getOrderResubmit() | 获得查看未通过单据数据层服务的引用 | |

5.4 数据层的分解

数据层主要给逻辑层提供数据访问服务，包括对持久化数据的增、删、改、查。User逻辑需要的服务由UserDSer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，所以抽象了数据服务。数据层模块的描述具体如图13所示。



图13 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所示。

数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserDSer | 持久化数据库的接口，提供用户数据增删改查等服务 |
| TransitInfoDSer | 持久化数据库的接口，提供更新物流信息等服务 |
| OrderNewDSer | 持久化数据库的接口，提供在数据库新建单据等服务 |
| OrderApproveDSer | 持久化数据库的接口，提供导入单据信息和保持单据审批状态等服务 |
| AccountDSer | 持久化数据库接口，提供账户数据的增删改查和期初建账服务 |
| CityDSer | 持久化数据库接口，提供城市数据的增删改查服务 |
| WorkOrgManDSer | 持久化数据库接口，提供人员机构数据的增删改查服务 |
| StockDSer | 持久化数据库接口，提供库存数据的增删改查服务 |
| CheckDSer | 持久化数据库接口，提供统计报表和日志记录的载入服务 |

5.4.2 数据层模块的接口规范

数据层模块的接口如表所示。

数据层模块的接口规范

UserDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserManData.addUser | 语法 | public Result addUser(UserPO user) throws RemoteException |
| 前置条件 | 同样ID的po在数据库中不存在 |
| 后置条件 | 在数据库中增加一个PO记录 |
| UserManData.deleteUser | 语法 | public Result deleteUser(String id) throws RemoteException |
| 前置条件 | 数据库中存在同样ID的用户 |
| 后置条件 | 在数据库中删除一个PO记录 |
| UserManData.reviseUser | 语法 | public Result reviseUser(UserPO user) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的用户 |
| 后置条件 | 在数据库中修改该用户的数据 |
| UserManData.findUser | 语法 | public ArrayList<UserPO> findUser(String keyword) throws RemoteException |
| 前置条件 | 数据库中存在含关键字的用户（关键字为空不检索） |
| 后置条件 | 返回数据库中含关键字用户数据的列表（关键字为空返回全部） |
| UserManData.getUserMan | 语法 | public static UserManData getUserMan() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回用户管理数据层服务实例 |
| UserManData.checkIsUsed | 语法 | public Result checkIsUsed(String id) throws RemoteException |
| 前置条件 | 新增或修改一个用户时 |
| 后置条件 | 返回用户名是否重复 |
| LoginData.findUser | 语法 | public UserPO findUser(String id,String password) throws RemoteException |
| 前置条件 | 用户输入用户名密码登录时 |
| 后置条件 | 返回数据库中含关键字用户PO信息 |
| LoginData.getLogin | 语法 | public static LoginData getLogin() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回登录数据层服务的实例 |

TransitInfoDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CourierData.find | 语法 | public ExpressPO find(String barcode) throws RemoteException |
| 前置条件 | 需要根据barcode获取快递单信息 |
| 后置条件 | 按barcode进行查找返回相应的PO结果 |
| CourierData. getCourierData | 语法 | public static CourierData getCourierData() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回订单查询数据层服务实例 |
| CustomerData.findTransit | 语法 | public ArrayList<String> findTransit(String barcode) throws RemoteException |
| 前置条件 | 需要通过barcode获取物流轨迹列表 |
| 后置条件 | 按barcode进行查找返回相应订单的物流轨迹 |
| CustomerData. getCustomerData | 语法 | public static CustomerData getCustomerData() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回物流信息查询数据层服务的引用 |

CityDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityData.addCity | 语法 | public Result addCity (CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中不存在 |
| 后置条件 | 在数据库中增加一个city记录 |
| CityData.reviseCity | 语法 | public Result reviseCity(CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中已存在 |
| 后置条件 | 在数据库中更新该city |
| CityData.deleteCity | 语法 | public Result deleteCity(String cityname) throws RemoteException |
| 前置条件 | Cityname在数据库中存在 |
| 后置条件 | 在数据库中删除该city |
| CityData.addCities | 语法 | public Result addCities(CitiesPO cities) throws RemoteException |
| 前置条件 | 该cities在数据库中不存在 |
| 后置条件 | 在数据库中添加该Cities |
| CityData.deleteCities | 语法 | public Result deleteCities(CitiesPO cities) throws RemoteException |
| 前置条件 | 该CitiesPO在数据库中已存在 |
| 后置条件 | 在数据库中删除该citiesPO信息 |
| CityData.reviseCities | 语法 | public Result reviseCities(CityPO cities) throws RemoteException |
| 前置条件 | 该citiesPO在数据库中已存在 |
| 后置条件 | 改变城市之间距离和价格 |
| CityData.getCitiesInfo | 语法 | public ArrayList<CityPO> getCitiesInfo () throws RemoteException |
| 前置条件 | 需要获取城市间信息 |
| 后置条件 | 返回城市间信息列表 |
| CityData.isCityUsed | 语法 | public Result isCityUsed(String name,String id) throws Remote Exception |
| 前置条件 | 创建或修改城市信息后 |
| 后置条件 | 检查城市名和区号是否被使用并返回结果 |
| CityData. getCity | 语法 | public static CityData getCity() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回城市管理数据层服务的实例 |
| ConstantData.getConstant | 语法 | public ConstantPO getConstant() throws Remote Exception |
| 前置条件 | 需要获取常量时 |
| 后置条件 | 从序列化文件中获取并返回常量信息 |
| ConstantData.setConstant | 语法 | public Result setConstant (ConstantPO constant) throws Remote Exception |
| 前置条件 | 设置常量后 |
| 后置条件 | 在序列化文件中更新常量信息 |
| ConstantData. getConstantData | 语法 | public static ConstantData getConstantData() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 获得期初建账数据层服务的引用 |

AccountDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountData.findAccount | 语法 | public ArrayList<AccountPO> findAccount(String keyword) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按关键字查找返回符合条件的账户列表 |
| AccountData.addAccount | 语法 | public Result addAccount(AccountPO account) throws RemoteException |
| 前置条件 | 同样账户名的账户在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| AccountData.deleteAccount | 语法 | public Result deleteAccount(int id) throws RemoteException |
| 前置条件 | 该id对应的账户在数据库中存在 |
| 后置条件 | 在数据库中删除一个po记录 |
| AccountData. getAccountData | 语法 | public static AccountData getAccountData() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回账户管理数据层服务的实例 |
| AccountData.reviseAccount | 语法 | public Result reviseAccount(AccountPO po) throws RemoteException |
| 前置条件 | 该账户在数据库中存在 |
| 后置条件 | 在数据库中更新该账户信息 |
| CountData.addCount | 语法 | public Result addCount(CountPO count) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 增加一个po记录的序列化 |
| CountData.getCount | 语法 | public ArrayList<CountPO> getCount() throws RemoteException |
| 前置条件 | 需要获取期初信息 |
| 后置条件 | 从序列化文件中获得期初信息 |
| CountData.deleteCount | 语法 | public Result deleteCount(int id) throws RemoteException |
| 前置条件 | id对应的期初信息存在 |
| 后置条件 | 删除期初信息序列化文件 |
| CountData. getCountData | 语法 | public static CountData getCountData() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 获得期初建账数据层服务的引用 |

OrderNewData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| \*OrderData.create | 语法 | public Result create (\*PO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个\*单记录 |
| \*OrderData.getOrderSize | 语法 | public int getOrderSize(String partId)throws RemoteException |
| 前置条件 | 需要获得当天生成单据数量 |
| 后置条件 | 在数据库中获取并返回当天单据数量 |
| \*OrderData.get\* | 语法 | public static \*OrderData get\*() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回单据生成数据层服务的实例 |
| StockInOrderData.getShelves | 语法 | public ArrayList<ShelfPO> getShelves() throws RemoteException |
| 前置条件 | 单据新建逻辑请求架信息 |
| 后置条件 | 在数据库中获取并返回架信息 |

workOrgManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DriverManData.findDriver | 语法 | public ArrayList<DriverPO> findDriver(String hallId) throws RemoteException |
| 前置条件 | 需要获得所有的Driver信息 |
| 后置条件 | 按营业厅Id查找并返回相应的DriverPO列表 |
| DriverManData.addDriver | 语法 | public Result addDriver (DriverPO po) throws RemoteException |
| 前置条件 | 新人员信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| DriverManData.deleteDriver | 语法 | public Result deleteDriver (String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的司机 |
| 后置条件 | 在数据库中删除相应的司机 |
| DriverManData.reviseDriver | 语法 | public Result reviseDriver(DriverPO) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的司机 |
| 后置条件 | 在数据库中更新相应ID司机的信息 |
| OrgManData.findOrg | 语法 | public ArrayList<OrgPO> findOrg(String keywords) throws RemoteException |
| 前置条件 | 逻辑层需要搜索获取机构PO信息 |
| 后置条件 | 按关键词查找并返回相应的OrgPO结果 |
| OrgManData.addOrg | 语法 | public Result addOrg(OrgPO po) throws RemoteException |
| 前置条件 | 新机构信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| OrgManData.deleteOrg | 语法 | public Result deleteOrg(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的机构 |
| 后置条件 | 在数据库中删除相应机构的信息 |
| OrgManData.reviseOrg | 语法 | public Result reviseOrg(OrgPO) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的机构 |
| 后置条件 | 在数据库中更新相应机构的信息 |
| VanManData.findVan | 语法 | public ArrayList<VanPO> findVan(String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要搜索获取相应的车辆信息 |
| 后置条件 | 按关键词查找并返回相应的VanPO结果 |
| VanManData.addVan | 语法 | public Result addVan(VanPO Van) throws RemoteException |
| 前置条件 | 新货车信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| VanManData.deleteVan | 语法 | public Result deleteVan(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的货车 |
| 后置条件 | 在数据库中删除相应货车的信息 |
| VanManData.reviseVan | 语法 | public Result reviseOrg(VanPO Van) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的货车 |
| 后置条件 | 在数据库中更新相应机构的信息 |
| WageManData.searchWork | 语法 | public List<WorkPO> searchWork (String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要获取关键字的人员信息 |
| 后置条件 | 数据库中获取并返回关键字有关的人员信息 |
| WageManData.updateWork | 语法 | public Result updateWork (WorkPO work) throws RemoteException |
| 前置条件 | 逻辑层更新了人员薪水制定信息 |
| 后置条件 | 在数据库中更新人员的薪水信息 |
| WorkManData.findWork | 语法 | public ArrayList<WorkPO> findWork(String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要获取关键字相关的人员信息 |
| 后置条件 | 按关键词查找并返回相应的WorkPO列表 |
| WorkManData.addWork | 语法 | public Result addWork(WorkPO work) throws RemoteException |
| 前置条件 | 新工人信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| WorkManData.deleteWork | 语法 | public Result deleteWork(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的工人 |
| 后置条件 | 在数据库中删除相应工人的信息 |
| WorkManData.reviseWork | 语法 | public Result reviseWork(WorkPO work) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的工人 |
| 后置条件 | 在数据库中更新相应工人的信息 |

StockData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckData.getIn | 语法 | public ArrayList<StockInPO> getIn(Calendar start, Calendar end,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取入库单列表 |
| 后置条件 | 返回数据库中开始时间和结束时间之间的入库单列表 |
| StockCheckData.getOut | 语法 | Public ArrayList<StockOutPO> getOut (Calendar te start, Calendar end,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取出库单列表 |
| 后置条件 | 返回数据库中开始时间和结束时间之间的出库单列表 |
| StockCheckNowData.getStock | 语法 | public StockPO getStock(String id)throws RemoteException |
| 前置条件 | 逻辑层需要获取库存情况 |
| 后置条件 | 返回数据库中库存的情况，包括各个架上的货物 |
| StockDivideData.getShelves | 语法 | public ArrayList<ShelfPO> getShelves(String id,String keyword)throws RemoteException |
| 前置条件 | 逻辑层需要获取架的信息 |
| 后置条件 | 返回数据库中所有架的信息 |
| StockDivideData.addShelf | 语法 | public Result addShelf(ShelfPO shelf) throws RemoteException |
| 前置条件 | 数据库中不含有同样ID的架 |
| 后置条件 | 在数据库中增加一个架的信息 |
| StockDivideData.deleteShelf | 语法 | public Result deleteShelf(String id) throws RemoteException |
| 前置条件 | 数据库中含有同样ID的架 |
| 后置条件 | 在数据库中删除相应的架 |
| StockDivideData.reviseShelf | 语法 | public Result reviseShelf(ShelfPO shelf) throws RemoteException |
| 前置条件 | 数据库中含有同样ID的架 |
| 后置条件 | 在数据库中修改相应的架 |
| StockWarningData.getWarning | 语法 | public ArrayList<Integer> getWarning(String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取当前警戒线比例 |
| 后置条件 | 根据仓库ID返回数据库中航运，铁运，汽运和机动的警戒线比例 |
| StockWarningData.setWarning | 语法 | public Result setWarning(int w,Part part,String id) throws RemoteException |
| 前置条件 | 逻辑层设置改变警戒线比例后 |
| 后置条件 | 根据仓库ID更新数据库中航运，铁运，汽运和机动的警戒线比例，返回结果 |
| StockCheckWarnData.getWarning | 语法 | public ArrayList<Integer> getWarning(String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取当前警戒线比例 |
| 后置条件 | 根据仓库ID返回数据库中航运，铁运，汽运和机动的警戒线比例 |
| StockCheckWarnData.getShelves | 语法 | public ArrayList<ShelfPO> getShelves(String id,String keyword)throws RemoteException |
| 前置条件 | 逻辑层需要获取架的信息 |
| 后置条件 | 返回数据库中所有架的信息 |
| StockCheckWarnData.getStock | 语法 | public StockPO getStock(String id)throws RemoteException |
| 前置条件 | 逻辑层需要获取库存情况 |
| 后置条件 | 返回数据库中库存的情况，包括各个架上的货物 |

OrderApproveData模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveData.getExamine | 语法 | public ArrayList<OrderPO> getExamine () throws RemoteException |
| 前置条件 | 逻辑层开启单据审批 |
| 后置条件 | 从数据库获得并返回需要审批单据列表 |
| OrderApproveData.update | 语法 | public Result update(Boolean isPassed, ArrayList<String>id, Order kind) throws RemoteException |
| 前置条件 | id存在于数据库中 |
| 后置条件 | 在数据库中更新订单审批状态 |
| OrderApproveData.getOrder | 语法 | public PO getOrder(String id,Order kind) throws RemoteException |
| 前置条件 | 逻辑层开启单据审批 |
| 后置条件 | 根据订单种类从数据库获得并返回相应的单据信息 |
| OrderApproveData.getUser | 语法 | public UserPO getUser (String userId) throws RemoteException |
| 前置条件 | 逻辑层需要用户信息 |
| 后置条件 | 根据用户id从数据库获得并返回相应的用户信息 |
| OrderResubmitData.getOrders | 语法 | public ArrayList<KindGetter> getOrders(String userId) throws RemoteException |
| 前置条件 | 逻辑层需要获得相应的单据列表 |
| 后置条件 | 根据用户ID从数据库获得并返回相应的单据列表 |

CheckData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DiaryData.seeDiary | 语法 | public ArrayList<DiaryPO> seeDiary(Calendar start, Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要日志列表 |
| 后置条件 | 根据输入的开始和结束日期从数据库中获得日志记录，返回相应的数据 |
| IncomeListData.getCollectRec | 语法 | public ArrayList<IncomePO> getCollectRec (Calendar date,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录 |
| 后置条件 | 根据日期和营业厅名，从数据库中获取并返回该营业厅的所有收款单记录 |
| BusiCircumData.findIncome | 语法 | public ArrayList<IncomePO> findIncome (Calendar start，Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录列表 |
| 后置条件 | 根据开始结束时段从数据库中获取并返回所有的收款记录 |
| BusiCircumData.findPayment | 语法 | public ArrayList<PaymentPO> findPayment (Calendar start，Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要获得付款记录列表 |
| 后置条件 | 根据开始结束时段从数据库中获取并返回所有的付款记录 |
| CostBeneData.getIncomes | 语法 | public ArrayList<IncomePO> findPayment () throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录列表 |
| 后置条件 | 从数据库中获取并返回所有的收款记录 |
| CostBeneData.getPayments | 语法 | public ArrayList<PaymentPO> getPayments () throws RemoteException |
| 前置条件 | 逻辑层需要获得付款记录列表 |
| 后置条件 | 从数据库中获取并返回所有的付款记录 |

1. 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。

· UserPO类包含用户的用户名，密码，权限等级属性

· AccountPO类包含账户名称与余额属性

· CitiesPO类包含城市间距离和价格属性

· CityPO类包含区号，名称属性。

· ConstantPO类包含最大装载量，价格，快递种类等属性。

· CountPO类包含编号，机构，人员，车辆，库存，账户属性。

· DiaryPO类包含操作日志的时间和操作方式属性

· DriverPO类包含司机的编号，姓名，出生日期，身份证号，手机，车辆单位，性别，行驶证期限属性。

· GoodPO类包含货物的id，库存id,日期，到达地，区，架，行，位属性

· OrgPO类包含机构的种类，地址，名称属性

· ShelfPO类包含快递编号，入库日期，目的地，区号，排号，架号，位号属性。

· StockPO类包含编号，库存名，货物属性。

· VanPO类包含车辆代号，发动机号，车辆号，底盘号，购买时间，服役时间，车辆图片属性

· WorkPO类包含工作人员的姓名，性别，职位，所属机构，薪水发放方式，发放金额，提成份额属性。

· TransitInfoPO类包含快递编号，物流信息属性。

· ArrivePO类包含到达日期，中转单编号，出发地，货物到达状态，订单处理人姓名，编号，处理日期属性。

· CenterLoadingPO类包含装车日期，本中转中心中转单编号，出发地，到达地，货柜号，监装员，本次装箱所有托运单号，运费属性，到达日期订单，处理人姓名，编号，处理日期。

· SendPO类包含到达日期，托运订单条形码号，派送员，订单处理人姓名，编号，处理日期属性。

· HallLoadingPO类包含装车日期、本中转中心汽运编号、到达地（营业厅）、车辆代号、监装员、押运员、本次装箱所有订单条形码号、运费，订单处理人姓名，编号，处理日期属性。

· IncomePO类包含收款日期，收款金额，收款快递员，对应的所有快递订单条形码号，订单处理人姓名，编号，处理日期。

· PaymentPO类包含付款日期，金额，付款人，付款账号，条目，备注属性，订单处理人姓名，编号，处理日期。

· ReceivePO类包含收件编号，收件人，收件日期，订单处理人姓名，编号，处理日期属性。

· RecipientPO类包含中转中心编号，到达日期，中转单编号，出发地，货物到达状态，订单处理人姓名，编号，处理日期属性。

· ExpressPO类包含寄件人姓名、住址、单位、电话、手机，收件人姓名、住址、单位、电话、手机，托运货物信息，包装费，总费用，订单条形码号，种类，订单处理人姓名，编号，处理日期属性。

· StockInPO类包含快递编号、入库日期、目的地、区号、排号、架号、位号，订单处理人姓名，编号，处理日期属性

· StockOutPO类包含快递编号、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，订单处理人姓名，编号，处理日期属性。

· TransitPO类包含运输工具种类，装车日期，本中转中心中转单编号，航班号，出发地，到达地，货柜号，监装员，本次装箱所有托运单号，运费，订单处理人姓名，编号，处理日期属性。

· PO类包含单据信息和是否审批属性

UserPO定义如下

public class UserPO extends PO implements Serializable{

private String id;

private String password;

private Jurisdiction juri;

private String name;

private String mobile;

private String org;

private int key;

private String orgId;

public UserPO(String id, String password, Jurisdiction juri, String name,

String mobile, String org, int key, String orgId) {

super();

this.id = id;

this.password = password;

this.juri = juri;

this.name = name;

this.mobile = mobile;

this.org = org;

this.key = key;

this.orgId = orgId;

}

public String getId() {

return id;

}

public String getPassword() {

return password;

}

public Jurisdiction getJuri() {

return juri;

}

public String getName() {

return name;

}

public String getMobile() {

return mobile;

}

public String getOrg() {

return org;

}

public int getKey(){

return key;

}

public String getOrgId() {

return orgId;

}

public String getCommand(MysqlOperation op) {

String command=null;

switch(op){

case *INSERT*:command="insert into user values"+"("+"'"+id+"','"+password+"','"+name+"','"+mobile+"',"+juri.ordinal()+","+key+",null,null,0,0,0)";break;

case *DELETE*:command="delete from user where id='"+id+"'";break;

case *FIND*:

if(name != null){

if(juri == null){

command="select \* from user where id like '%"+id+"%' or name like '%"+name+"%' or org like '%"+org+"%'";

} else {

command="select \* from user where juri="+juri.ordinal();

}

} else {

command = "select \* from user where id = '"+id+"'";

}

break;

case *UPDATE*:command="update user set id='"+id+"',password='"+password+"',name='"+name+"',mobile='"+mobile+"',juri="+juri.ordinal()+" where keyid="+key;break;

}

return command;

}

}

6.2数据库表

数据库中包含Account表，Cities表，City表，Count表，Diary表，Driver表，Org表，Shelf表，Stock表，User表，Van表，Wage表，Work表，Arrive表，CenterLoading表，Express表，HallLoading表，Income表，Order表，Payment表，Receive表，Recipient表，Send表，StockIn表，StockOut表，TransitPO表。