|  |
| --- |
| 黑化肥挥发不会发灰 小组 |
| 物流管理系统（LCS）软件体系结构描述文档 |
| V2.0 |

|  |
| --- |
| Mouse  2015-10-30 |

物流管理系统（LCS）软件体系结构描述文档

——黑化肥挥发不会发灰 小组

目录

[引言 1](#_Toc439625771)

[编制目的 1](#_Toc439625772)

[词汇表 1](#_Toc439625773)

[参考资料 2](#_Toc439625774)

[产品概述 2](#_Toc439625775)

[逻辑视角 2](#_Toc439625776)

[组合视角 4](#_Toc439625777)

[开发包图 4](#_Toc439625778)

[运行时进程 6](#_Toc439625779)

[物理部署 7](#_Toc439625780)

[接口视角 7](#_Toc439625781)

[模块的职责 7](#_Toc439625782)

[用户界面的分解 9](#_Toc439625783)

[业务逻辑层的分解 10](#_Toc439625784)

[数据层的分解 26](#_Toc439625785)

[信息视角 31](#_Toc439625786)

[数据持久化对象 31](#_Toc439625787)

[文件持久化格式 47](#_Toc439625788)

# 引言

## 编制目的

本报告详细完成对连锁商店系统的概要设计，达到知道详细设计和开发的目的，同时实现和测试人员以及用户的沟通。

本报告面向开发人员、测试人员、以及最终用户而编写，是了解系统的导航。

## 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| LCS | 物流管理系统 | Logicstic Control System |

(详情请见文档中注释)

## 参考资料

# 产品概述

本文档参考物《流管理系统系统用例描述文档》以及《物流管理系统需求规格说明书》中对产品的概括性描述

# 逻辑视角

物流管理系统主要采用分层体系结构设计。系统被分为：展示层(Presentation)、逻辑层(Business Logic)和数据层(Data)。

展示层负责GUI的实现和展示，与用户的交互；逻辑层负责业务逻辑处理的实现；数据层负责持久化数据的储存和访问服务。

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 包图表达的逻辑视角



图2 软件体系结构逻辑设计方案

# 组合视角

## 开发包图

物流管理系统的最终开发包设计如表1所示。

表1物流管理系统的最终开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| MainUI | CompanyUI,LogisticUI,StorageUI,FinancialUI,VO |
| CompanyUI | CompanyService,VO |
| LoginUI | UserBLService |
| LogisticUI | LogisticService,VO |
| StorageUI | StorageService,VO |
| FinancialUI | FinancialService,VO |
| CompanyBLService | VO |
| FransferBLService | VO |
| OrderBLService | VO |
| StorageBLService | VO |
| FinancialBLService | VO |
| UserBL | CompanyData |
| CompanyBL | CompanyDataService,CompanyBLService,PO,  DataFactoryService |
| TransferBL | CompanyDataService,StorageBL,OrderBL,  TranferBLService,PO, DataFactoryService |
| OrderBL | OrderDataService,OrderBLService,PO  , DataFactoryService |
| StorageBL | StorageDataService,OrderBL,StorageBLService,PO,  DataFactoryService |
| FinancialBL | CommpanyBL,FinancialService,FinancialBLService  ,PO, DataFactoryService |
| CompanyDataService | JavaRMI,PO |
| TransferDataService | JavaRMI,PO |
| OrderDataService | JavaRMI,PO |
| StorageDataService | JavaRMI,PO |
| FinancialDataService | JavaRMI,PO |
| DataFactoryService | JavaRMI,PO |
| CompanyData | CompanyDataService,PO |
| TransferData | TransferDataService,PO |
| OrderData | OrderDataService,PO |
| StorageData | StorageDataService,PO |
| FinancialData | FinancialDataService,PO |
| DataFactory | DataFactoryService,PO |
| JavaRMI |  |
| VO |  |
| PO |  |

物流管理系统的客户端开发包图设计如图3所示，服务器端开发包设计如图4所示。



图3 客户端开发包图



图2 服务器端开发包图

## 运行时进程

在物流管理系统中，有多个客户端进程和一个服务器端进程，其进程图如图5所示。结合部署图，客户端进程在客户端机器上运行，服务器端进程在服务器端机器上进行。

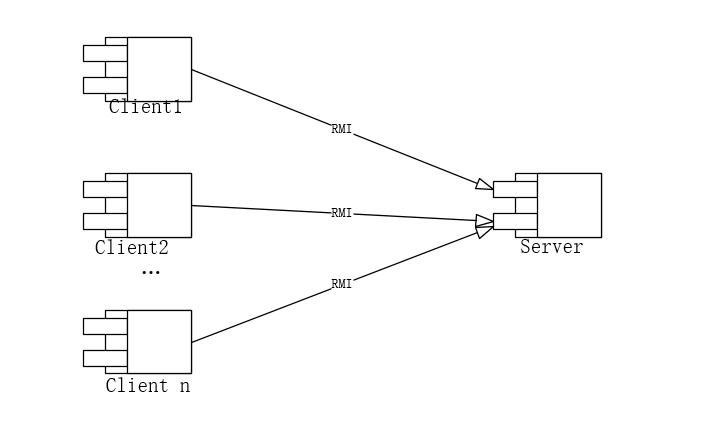


图5 物流管理系统进程图

## 物理部署

物流管理系统中的客户端软件是放在客户端机器上的，服务器端构件是放在服务器端机器上的。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK 6.0及以上版本的一部分，因此在系统JDK环境已经配置好的情况下，不需要独立部署。

部署图如图6所示。

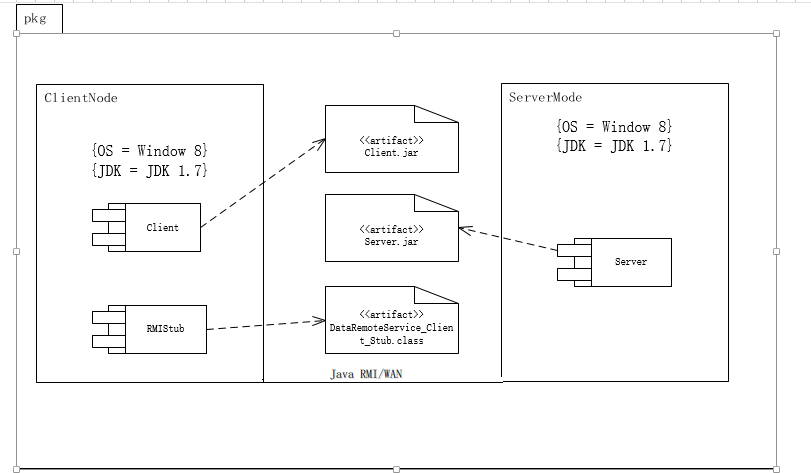


图6 物流管理系统部署图

# 接口视角

## 模块的职责

客户端模块和服务器端模块视图分别如图所示。客户端各层和服务器端各层的职责分别如表所示。



客户端模块视图



服务器端模块视图

客户端模块各层的职责

|  |  |
| --- | --- |
| 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 | 基于窗口的物流公司客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用java RMI机制查找RMI服务 |

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 | 负责数据的持久化以及数据访问接口 |
| 服务器端网络模块 | 利用java RMI机制查找RMI服务 |

层之间调用的接口如表所示。上一层的接口只能保存下一条的接口

层之间调用的接口

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| CompanyService  LogisticService  StorageService  FinancialService | 客户端展示层 | 客户端业务逻辑层 |
| CompanyDataService  TransferDataService  OrderDataService  StorageDataService  CompanyDataService  DataFactoryService | 客户端业务逻辑层 | 服务器端数据层 |

## 用户界面的分解

根据需求，系统存在35个界面，详情见图7用户界面跳转。

服务器端与客户端的界面接口是一致的。



图7 用户界面跳转

1. 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 初始界面，负责界面之间的显示和跳转控制 |

1. 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | init(args: String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示界面和Login界面 |
| UserUI | 语法 | display() |
| 前置条件 | 无 |
| 后置条件 | 跳转至响应的界面 |
| CompanyUI | 语法 | display() |
| 前置条件 | 以总经理和系统管理员身份正确登录 |
| 后置条件 | 无 |
| StorageUI | 语法 | display() |
| 前置条件 | 以仓库管理员身份正确登陆 |
| 后置条件 | 无 |
| TransferUI | 语法 | display() |
| 前置条件 | 中转中心业务员和营业厅业务员身份正确登陆 |
| 后置条件 | 显示界面和Login界面 |
| FinancialUI | 语法 | display() |
| 前置条件 | 无 |
| 后置条件 | 无 |

1. 用户界面层模块需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| businesslogic.service.UserBLService | 登陆服务 |
| businesslogic.service.\*BLService | 各种业务界面均需要的逻辑接口 |

## 业务逻辑层的分解



1. 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CompanyBL | 负责实现总经理的管理服务，以及系统用户管理等功能 |
| UserBL | 负责实现登录界面相关的服务 |
| TransferBL | 负责运输管理部分的服务 |
| OrderBL | 负责为订单相关的操作提供服务 |
| StorageBL | 负责仓库管理部分所需要的服务 |
| FinacialBL | 负责财务人员财务管理部分所需要的服务 |

1. 业务逻辑层模块的接口规范

* UserBL:

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| User.Login | 语法 | public ResultMessage login(long id, String pswd); | |
| 前置条件 | 用户名和pswd不为空 | |
| 后置条件 | 验证登录信息 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getUserDataService | | | 获得用户数据服务的引用 |
| UserDataService.getUserPO | | | 获得用户数据PO |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| User.Register | 语法 | public ResultMessage Register(UserPO usr); | |
| 前置条件 | 用户名和pswd不为空 | |
| 后置条件 | 在用户数据库查找是否已存在同名用户，并添加 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getUserDataService | | | 获得用户数据服务的引用 |
| UserDataService.getUserPO | | | 获得用户数据PO |
| UserDataService.addUserPO | | | 增加一个用户 |

* OrderBL:

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.addOrder | 语法 | public void addOrder(OrderPO order); | |
| 前置条件 | 无 | |
| 后置条件 | 系统创建新订单 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单数据服务的引用 |
| DataFactoryService.getLogisticInfoDataService | | | 获得物流信息数据服务的引用 |
| LogisticDataService.addRecord | | | 增加一项物流信息 |
| OrderDataService.insertOrder | | | 添加新订单项 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.enquire | 语法 | public String[] enquire(long orderID); | |
| 前置条件 | 用户输入了格式正确的订单号 | |
| 后置条件 | 系统查询订单信息并返回相应接口 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getLogisticInfoDataService | | | 获得物流信息数据服务的引用 |
| LogisticsDataService.enquire | | | 查询指定订单号的订单信息 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.evaluateTime | 语法 | public static int evalutateTime(long orderID); | |
| 前置条件 | 新订单信息已经输入完毕 | |
| 后置条件 | 系统创建新订单 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单信息数据服务的引用 |
| OrderDataService.enquire | | | 查询订单信息 |
| OrderDataService.addLogisticInfo | | | 增加一项物流信息 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.generateFee | 语法 | public static double generateFee(long ID); | |
| 前置条件 | 新订单信息输入完毕 | |
| 后置条件 | 无 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单信息数据服务的引用 |
| OrderDataService.getOrderPO | | | 获取整个订单信息对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.receive | 语法 | public void receive(long ID); | |
| 前置条件 | 一批货物到达了营业厅 | |
| 后置条件 | 系统准备进行派件操作 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单信息数据服务的引用 |
| OrderDataService.enquire | | | 查询订单信息 |
| OrderDataService.getOrderPO | | | 获取订单信息 |
| OrderDataService.setOrderPO | | | 设置订单信息 |
| OrderDataService,addLogisticInfo | | | 增加物流信息 |
| TransferDataService,addSendPO | | | 创建派件单 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.entruck | 语法 | public static double entruck(long[] orderList); | |
| 前置条件 | 系统已经完成了订单信息的输入 | |
| 后置条件 | 系统准备对其进行收货操作 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单信息数据服务的引用 |
| OrderDataService.getOrderPO | | | 获取整个订单信息对象 |
| OrderDataService.addEntruckPO | | | 添加装车单 |

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Order.sign | 语法 | public void sign(long[] ID, String name. long phone); | |
| 前置条件 | 营业厅已经收货，并且生成了派件单 | |
| 后置条件 | 系统准备进行装车操作 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DataFactoryService.getOrderDataService | | | 获得订单信息数据服务的引用 |
| OrderDataService.getOrderPO | | | 获取整个订单信息对象 |
| OrderDataService.setPO | | | 修改订单信息对象 |
| OrderDataService.addLogisticInfo | | | 增加一项物流信息 |
| TransferDataService.findSend | | | 获取派件单信息 |

* FinancialBL:

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Financial.buildStaffExcel | 语法 | public ResultMessage buildStaffExcel(String info) |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 在一个期初建账回合中，系统建立新的人员信息表 |
| Financial. buildOrganisationExcel | 语法 | public ResultMessage buildOrganisationExcel(String info) |
|  | 前置条件 | 启动一个期初建账任务 |
|  | 后置条件 | 在一个期初建账回合中，系统建立新的机构信息表 |
| Financial.buildVehicleInfoExcel | 语法 | public ResultMessage buildVehicleInfoExcel (String info) |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 在一个期初建账回合中，系统建立新的车辆信息表 |
| Financial.buildDriverInfoExcel | 语法 | public ResultMessage buildDriverInfoExcel (String info) |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 在一个期初建账回合中，系统建立新的司机信息表 |
| Financial.buildInventoryExcel | 语法 | public ResultMessage buildInventoryExcel (String info) |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 在一个期初建账回合中，系统建立新的库存信息表 |
| Financial.buildBankAccountExcel | 语法 | public ResultMessage buildBankAccountExcel (String info,String[] name,double[] money) |
| 前置条件 | 启动一个期初建账任务 |
| 后置条件 | 在一个期初建账回合中，系统建立新的银行账户信息表 |
| Financial.selectTime | 语法 | public ResultMessage selectTime(String dateStart,String dateEnd) |
|  | 前置条件 | 启动一个统计报表任务 |
|  | 后置条件 | 在一个统计报表回合中，增加该时间段内所有的付款单和收款单信息 |
| Financial.printManageExcel | 语法 | public ResultMessage printManageExcel (String dateStart,String dateEnd) |
|  | 前置条件 | 启动一个统计报表任务 |
|  | 后置条件 | 在一个统计报表回合中，打印该时间段内所有的付款单和收款单信息 |
| Financial.acIdentity | 语法 | public ResultMessage acIdentity (String name,String password) |
|  | 前置条件 | 启动一个账户管理任务 |
|  | 后置条件 | 在一个账户管理回合中，返回“账号或密码输入错误”或进入登录后界面 |
| Financial.findAccount | 语法 | public ResultMessage findAccount(String name) |
|  | 前置条件 | 启动一个账户管理任务 |
|  | 后置条件 | 在一个账户管理回合中，增加该账户的信息 |
| Financial.addAccount | 语法 | public ResultMessage addAccount(String name,double money) |
|  | 前置条件 | 启动一个账户管理任务 |
|  | 后置条件 | 在一个账户管理回合中，返回更新后的账户信息表 |
| Financial.changeAccount | 语法 | public ResultMessage changeAccount(String name,double money) |
|  | 前置条件 | 启动一个账户管理任务 |
|  | 后置条件 | 在一个账户管理回合中，返回更新后的账户信息表 |
| Financial.deleteAccount | 语法 | public ResultMessage deleteAccount(String name) |
| 前置条件 | 启动一个账户管理任务 |
| 后置条件 | 在一个账户管理回合中，返回更新后的账户信息表 |
| Financial.recordStaffPay | 语法 | public ResultMessage recordStaffPay(String info) |
| 前置条件 | 启动一个成本管理任务 |
| 后置条件 | 在一个成本管理回合中，增加员工工资支出项目信息 |
| Financial.recordPostagePay | 语法 | public ResultMessage recordPostagePay (String info) |
|  | 前置条件 | 启动一个成本管理任务 |
|  | 后置条件 | 在一个成本管理回合中，增加运费支出项目信息 |
| Financial.recordRentPay | 语法 | public ResultMessage recordRentPay (String info) |
|  | 前置条件 | 启动一个成本管理任务 |
|  | 后置条件 | 在一个成本管理回合中，增加租金支出项目信息 |
| Financial.printPayment | 语法 | public ResultMessage printPayment(String info) |
|  | 前置条件 | 启动一个成本管理任务 |
|  | 后置条件 | 在一个成本管理回合中，打印付款单 |
| Financial.endFinancial | 语法 | public void endFinancial() |
|  | 前置条件 | 已结束正在进行的任务 |
|  | 后置条件 | 结束此次财务方面的管理，持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| FinancialDS.find(int id) | | 根据ID进行查找单一持久化对象 |
| FinancialDS.find(String field,int value) | | 根据字段名和值进行查找多个持久化对象 |
| FinancialDS.insert(AccountPO po) | | 插入单一持久化对象 |
| FinancialDS.delete(AccountPO po) | | 删除单一持久化对象 |
| FinancialDS.update(AccountPO po) | | 更新单一持久化对象 |
| FinancialDS.getFinancialDatabase | | 得到Financial数据库的服务的引用 |
| FinancialDS.init() | | 初始化持久化数据库 |
| FinancialDS.finish() | | 结束持久化数据库的使用 |

* CompanyBL

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Company.modifySalary | 语法 | public void modifySalary(StaffType stafftype,double salary) |
| 前置条件 | 启动员工工资管理的任务 |
| 后置条件 | 修改完成此员工类型的工资(月薪/计次) |
| Company.moveStaff | 语法 | public void moveStaff(Department fromDepartment,Department toDepartment,long id) |
| 前置条件 | 启动员工机构管理调用员工的任务 |
| 后置条件 | 在原机构中删除员工信息添加到新机构中 |
| Company.addStaff | 语法 | public void addStaff(Department department,long id,String sex,int age) |
| 前置条件 | 启动员工机构管理增添员工的任务 |
| 后置条件 | 在指定机构里添加进此员工信息 |
| Company.deleteStaff | 语法 | public void deleteStaff(Department department,long id) |
| 前置条件 | 启动员工机构管理删除员工的任务 |
| 后置条件 | 在指定机构中删除此员工信息 |
| Company.modifyCityDistance | 语法 | public void modifyCityDistance(String fromCity,string toCity,double distance) |
| 前置条件 | 启动城市管理中距离管理的任务 |
| 后置条件 | 修改指定起始城市之间的距离 |
| Company.modifyCityPrice | 语法 | public void modifyCityPrice(String fromCity,String toCity,double price,Transtype transtype) |
| 前置条件 | 启动城市管理中价格管理的任务 |
| 后置条件 | 修改指定运输方式的起始城市之间的运输价格 |
| Company.checkEarningForm | 语法 | public void CheckEarningForm() |
| 前置条件 | 启动查看统计分析中查看成本收益表任务 |
| 后置条件 | 显示成本收益表 |
| Company.checkSituationForm | 语法 | public void CheckSituationForm(String starttime,String endtime) |
| 前置条件 | 启动查看统计分析中查看经营情况表任务 |
| 后置条件 | 显示经营情况表 |

|  |  |  |
| --- | --- | --- |
| Company.showReceipt | 语法 | public void showReceipt() |
| 前置条件 | 启动审批单据的任务 |
| 后置条件 | 将所有需要审批的单据显示出来 |
| Company.modifyReceipt | 语法 | Public void modifyReceipt(long id) |
| 前置条件 | 启动修改单据的任务 |
| 后置条件 | 单据数据被修改 |
| Company.approveReceipt | 语法 | public ReceiptState approveReceipt(long id) |
| 前置条件 | 启动完成单据审批的任务 |
| 后置条件 | 单据状态改为审批后状态 |
| Company.endModifySalary | 语法 | public void endmodifySalary() |
| 前置条件 | 完成修改员工工资 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endMoveStaff | 语法 | public void endmoveStaff() |
| 前置条件 | 完成调用人员任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endAddStaff | 语法 | public void endAddStaff() |
| 前置条件 | 完成添加员工任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endDeleteStaff | 语法 | public void endDeleteStaff() |
| 前置条件 | 完成删除员工任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endModifyCity  Distance | 语法 | public void endModifyCityDistance |
| 前置条件 | 完成修改城市之间距离任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endModifyCityPrice | 语法 | public void endModifyCityPrice() |
| 前置条件 | 完成修改城市之间运输价格任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endCheckEarning  Form | 语法 | public void endCheckEarningForm() |
| 前置条件 | 完成成本收益表的查看 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endCheckSituation  Form | 语法 | public void endCheckSituationForm() |
| 前置条件 | 完成经营情况表的查看 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |

|  |  |  |
| --- | --- | --- |
| Company.showReceipt | 语法 | public void showReceipt() |
| 前置条件 | 启动审批单据的任务 |
| 后置条件 | 将所有需要审批的单据显示出来 |
| Company.modifyReceipt | 语法 | Public void modifyReceipt(long id) |
| 前置条件 | 启动修改单据的任务 |
| 后置条件 | 单据数据被修改 |
| Company.approveReceipt | 语法 | public ReceiptState approveReceipt(long id) |
| 前置条件 | 启动完成单据审批的任务 |
| 后置条件 | 单据状态改为审批后状态 |
| Company.endModifySalary | 语法 | public void endmodifySalary() |
| 前置条件 | 完成修改员工工资 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endMoveStaff | 语法 | public void endmoveStaff() |
| 前置条件 | 完成调用人员任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endAddStaff | 语法 | public void endAddStaff() |
| 前置条件 | 完成添加员工任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endDeleteStaff | 语法 | public void endDeleteStaff() |
| 前置条件 | 完成删除员工任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endModifyCity  Distance | 语法 | public void endModifyCityDistance |
| 前置条件 | 完成修改城市之间距离任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endModifyCityPrice | 语法 | public void endModifyCityPrice() |
| 前置条件 | 完成修改城市之间运输价格任务 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endCheckEarning  Form | 语法 | public void endCheckEarningForm() |
| 前置条件 | 完成成本收益表的查看 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endCheckSituation  Form | 语法 | public void endCheckSituationForm() |
| 前置条件 | 完成经营情况表的查看 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Company.endModifyReceipt | 语法 | public void endModifyReceipt() |
| 前置条件 | 完成对单据的修改 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endApproveReceipt | 语法 | public void endApproveReceipt() |
| 前置条件 | 完成单据审批 |
| 后置条件 | 结束此次任务，持久化更新涉及的领域对象的数据 |
| Company.endCoManage | 语法 | public void endCoManage() |
| 前置条件 | 完成公司管理 |
| 后置条件 | 退出系统 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| CompanyDataService.findSalary(Department department) | | 根据员工机构查找单一持久化对象 |
| CompanyDataService.addSalary(SalaryPO salarypo) | | 添加单一持久化对象 |
| CompanyDataService.updateSalary(SalaryPO salarypo) | | 更新单一持久化对象 |
| CompanyDataService.findStaff(long id) | | 根据id查找单一持久化对象 |
| CompanyDataService.addStaff(StaffPO staffpo) | | 添加单一持久化对象 |
| CompanyDataService.deleteStaff(StaffPO staffpo) | | 删除单一持久化对象 |
| CompanyDataService.updateStaff(StaffPO staffpo) | | 更新单一持久化对象 |
| CompanyDataService.findCityInfo(String fromCity,String toCity ) | | 根据始末城市查找单一持久化对象 |
| CompanyDataService.addCityInfo(CityPO citypo) | | 添加单一持久化对象 |
| CompanyDataService.updateCityInfo(CityPO citypo) | | 新单一持久化对象 |
| CompanyDataService.findEarningFormInfo() | | 查找单一持久化对象 |
| CompanyDataService.findSituationFormInfo(String fromdate String todate) | | 查找多数持久化对象 |
| CompanyDataService.findReceipt(long id) | | 根据id查找单一持久化对象 |
| CompanyDataService.addReceipt(ReceiptPO receiptpo) | | 添加单一持久化对象 |
| CompanyDataService.deleteReceipt(ReceiptPO receiptpo) | | 删除单一持久化对象 |
| CompanyDataService.updateReceipt(ReceiptPO receiptpo) | | 新单一持久化对象 |
| 需要的服务（需接口） | | |
| CompanyDataService.init() | | 初始化持久化数据库 |
| CompanyDataService.finish() | | 结束持久化数据库的使用 |
| CompanyDataService.init() | | 初始化持久化数据库 |
| CompanyDataService.finish() | | 结束持久化数据库的使用 |

* StorageBL:

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| storageIn | | | |
| StorageBusinessBL.getArriveListCode | 语法 | | public ArriveListVO inputArriveListCode(long arriveListCode) |
| 前置条件 | | 启动一个入库活动 |
| 后置条件 | | 系统显示到达单信息 |
| StorageBusinessBL.sort  （分拣） | 语法 | | public PositionInfoVO sort() |
| 前置条件 | | 已经获得到达单信息 ArriveListPO |
| 后置条件 | | 返回找到相应数量的位置信息（PositionInfoVO）,系统显示库存位置信息 Area-shelf-row-point-code，位置数量不足，返回null |
| StorageBussinessBL.sortSure  （确认分拣） | 语法 | | public EntryListVO sortSure (PositionInfoVO info) |
| 前置条件 | | 已经生成了库存信息位置 |
| 后置条件 | | 调用库存信息表,生成入库单方法 |
| StorageBusinessBL.createEntryList  (生成入库单) |  | | Public EntryListVO createEntryList(PositionInfoVO info) |
| 前置条件 | | sure() 返回true |
| 后置条件 | | 生成EntryListVO |
| StorageBusinessBL.EntryListInput  （入库单输入） |  | | Public ArrayList<string> EntryListInput() |
| 前置条件 | | 用户输入入库单剩余信息 |
| 后置条件 | | 返回ArrayList<string> lastInfo |
| StorageBusinessBL.completeEntryList  （完成入库单） |  | | Public ResultMessage completeEntryList(ArrayList<string> lastInfo,EntryListVO evo) |
| 前置条件 | | 已经返回了lastInfo |
| 后置条件 | | 返回一个EntryListPO保存完整入库单, 更新快件状态信息, |
|  | | | |
| storageOut | | | |
| StorageBusinessBL.getTransferList（输入中转单号） | 语法 | | Public TransferListVO getTransferList(long TransferListCode) |
|  | 前置条件 | | 生成了一个中转单，发起了一个出库活动 |
|  | 后置条件 | | 返回中转单信息,若格式错误，返回null |
| StorageBusinessBL.createStorageOutList  （生成出库单） | 语法 | | Public DeliveryListVO createStorageOutList() |
|  | 前置条件 | | 找到中转单 |
|  | 后置条件 | | 根据中转单信息生成默认出库单 |
| StorageBusinessBL.modifyStorageOutList  (修改出库单) | 语法 | | Public ResultMessage modifyStorageOutList() |
|  | 前置条件 | | 生成了默认出库单，发起修改出库单活动 |
|  | 后置条件 | | 完成最终出库单，更新库存信息表，更新快件状态信息 |
|  | | | |
| 需要的服务（需接口） | | | |
|  | | | |
| 服务名 | | 服务 | |
| TransferInfoService.getArriveList(long ArriveListCode) | | 根据到达单号查找到达单 | |
| StorageOperateService.checkSpace() | | 查询仓库剩余空间，返回剩余空间信息 | |
| StorageOperateService. getPosition(Char area) | | 查找一个空位置，参数为区域字符，返回找到的位置信息 char-shelf-row-point-code，以便下一次查找空位置 | |
| StorageOperateService. adjustStorage(Area source,Area target,int amout) | | 调整库存 | |
| OrderService. updateOrder(StorageListPO list) | | 更新订单信息，参数为入库单/出库单信息 | |
| StorageDataService. updateStorage(StorageListPO e) | | 更新库存信息，参数为入库单/出库单信息 | |

* StorageOperator:

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageAdjust | | |
| StorageOperate.showSpace | 语法 | Public String showSpace() |
|  | 前置条件 | 发起了一个库存调整活动 |
|  | 后置条件 | 系统返回各区域剩余空间信息 xxx-xxx-xxx-xxx，分别对应航运区，铁运区，汽运区，机动区 |
| StorageOperate.inputChoice | 语法 | Public ResultMessage inputChoice(StorageArea source,StorageArea target,int amount) |
|  | 前置条件 | 用户输入了选中的源区域和目标区域，输入了调整的空间大小，输入合法。 |
|  | 后置条件 | 设置源区域  设置目标区域  设置调整的位置数目 |
| StorageShow | | |
| StorageOperate.inputTime | 语法 | Public ResultMessage inputTime(String startTime,String endTime) |
|  | 前置条件 | 用户输入了起始时间和终止时间  yyyy-MM-dd |
|  | 后置条件 | 调用getList |
| StorageOperate.getList | 语法 | Public ArrayList<StorageListVO> getList(string startTime,String endTime) |
|  | 前置条件 | 输入了合法的开始时间和终止时间 |
|  | 后置条件 | 返回该时间段内所有出库单和出库单 |
| StorageCheck(库存盘点) | | |
| StorageOperate.storageCheck | 语法 | Public StorageInfoPO storageCheck(String date) |
|  | 前置条件 | 用户发起了一次库存盘点活动 |
|  | 后置条件 | 并生成一个库存快照StorageInfoPO |
| StorageOperate.saveStorageInfo | 语法 | Public ResultMessage saveStorageInfo(StorageInfoPO info) |
|  | 前置条件 | 生成了一份库存快照，用户要求保存 |
|  | 后置条件 | 将快照保存在数据文件中 |
| StorageOperate.showStorageInfo | 语法 | Public showStorageInfoVO storageInfo(StorageInfoPO info) |
|  | 前置条件 | 生成了一份库存快照 |
|  | 后置条件 | 系统显示这份库存快照 |
| StorageOperate.storageInfoOutput | 语法 | Public void storageCheckOutput(storageInfoPO info) |
|  | 前置条件 | 成功得到当天库存信息ArrayList<String>  storageInfo |
|  | 后置条件 | 将storageInfo导出为excel (未解决) |
| StorageInit | | |
| StorageOperate.checkPassword | 语法 | Public ResultMessage checkPassword(String password) |
|  | 前置条件 | 用户输入了初始化密码 |
|  | 后置条件 | 返回检查结果 |
|  |  |  |
| StorageOperate.inputStorageInitInfo | 语法 | Public ResultMessage inputStorageSeperateInfo(int plane,int train,int track,int flexible) |
|  | 前置条件 | 用户在界面上输入了四个区域的位置大小 |
|  | 后置条件 | 将仓库各区大小按照输入数据进行设置 |
|  |  |  |

* StorageOther:

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageOtherBL.getStorageInfo | 语法 | Public StorageInfoPO getStorageInfo(TransferCenter center) |
|  | 前置条件 | 获得了StorageDataService |
|  | 后置条件 | 返回一个StorageInfoPO仓库信息对象 |
|  |  |  |
| 需要的服务（需接口） | | |
| StorageDataService.search（POType type , Institution ins） | 根据目的地提供库存信息 | |

* TransferCenter:

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 装运管理（中转中心） | | |
| TransferCenterBL.showCenterInfo | 语法 | Public CenterShowVO showCenterInfo(TransferCenter thisCenter) |
|  | 前置条件 | 用户发起一次装运管理活动 |
|  | 后置条件 | 系统显示发往各中转中心的仓库货物数量 |
| TransferCenterBL.chooseDestination | 语法 | Public ArrayList<BriefOrderVO> chooseDestination(TransferCenter center) |
|  | 前置条件 | 用户选择了一个中转中心,或营业厅 |
|  | 后置条件 | 系统显示待发往该中转中心或营业厅的所有简要订单信息 |
| TransferCenterBL.addOrder | 语法 | Public ResultMessage addOrder(ArrayList<BriefOrderVO> source,ArrayList<BriefOrderVO> target ,long code) |
|  | 前置条件 | 用户选择一个订单 |
|  | 后置条件 | 系统将该订单号装进待发订单号列表 |
| TransferCenterBL.checkWeight() | 语法 | Public ResultMessage checkWeight(StorageArea vehicle ,ArrayList<BriefOrder> target) |
|  | 前置条件 | 用户确认订单选择完成 |
|  | 后置条件 | 系统检查选中订单货物重量和是否满足装运重量要求 |
| TransferCenterBL.deleteOrder | 语法 | Public ResultMessage deleteOrder(ArrayList<BriefOrderVO> source,ArrayList<BriefOrderVO> target ,long code) |
|  | 前置条件 | 用户取消选择一个订单 |
|  | 后置条件 | 系统将该订单号从待发订单号列表删除 |
| TransferCenterBL.createTransferList | 语法 | Public TransferListVO createTransferList(TransferCenter center , ArrayList<BriefOrderVO> order) |
|  | 前置条件 | CheckWeight返回true |
|  | 后置条件 | 生成默认中转单或装车单 |
| TransferCenterBL.ModifyTransferList | 语法 | Public TransferListPO modifyTransferList(ArrayList<String> input, TransferListVO list) |
|  | 前置条件 | 用户确认修改完成 |
|  | 后置条件 | 信息格式无误且全部填写，则返回PO，信息有误，返回null |
| TransferCenterBL.saveTransferList | 语法 | Public ResultMessage saveTransferList(TransferList PO list) |
|  | 前置条件 | 用户确认保存中转单或装车单 |
|  | 后置条件 | 系统保存该单据 |
|  | | |
| 中转接收 | | |
| TransferCenterBL.getTransferList | 语法 | Public TransferListVO getTransferList(long code) |
|  | 前置条件 | 得到了一个格式正确的单号 |
|  | 后置条件 | 若可以查到单据，返回单据，系统显示单据；查不到单据，返回null，系统提示输入错误 |
| TransferCenterBL.createArriveList | 语法 | Public ArriveListVO createArriveList(TransferListVO list) |
|  | 前置条件 | 用户要求生成默认到达单 |
|  | 后置条件 | 系统显示默认到达单 |
| TransferCenterBL.modifyArriveList | 语法 | Public ArriveListVO modifyArriveList(ArriveListVO list,ArrayList<String> input) |
|  | 前置条件 | 用户确认修改 |
|  | 后置条件 | 若信息格式正确，且信息完整，返回到达单PO；否则，返回null |
| TransferCenterBL.saveArriveList | 语法 | Public ResultMessage saveArriveList(ArriveListPO list) |
|  | 前置条件 | 用户确认保存到达单 |
|  | 后置条件 | 系统保存到达单 |
|  |  |  |
|  |  |  |
| 需要的服务（需接口） | | |
| TransferData.add (POType type,DataPO data ) | 保存中转单或装车单,到达单 | |
| TransferData.search(POType type ,DataPO data) | 根据单号查找中转单或装车单 | |
| StorageOtherService.getStorageInfo(TransferCenter center) | 对外提供获取库存信息方法 | |

* TransferHall:

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| 收货 | | | |
| TransferHallBL.getTransferList | 语法 | | Public TransferListVO getTransferList(long code) |
| 前置条件 | | 用户输入的汽运编号格式正确 |
| 后置条件 | | 单号可查到，返回一个装车单信息；单号查不到，返回null |
| TransferHallBL.createOrderArrival | 语法 | | Public ArrayList<OrderArrivalVO> createOrderArrival(TransferListPO list) |
| 前置条件 | | 得到了一个装车单 |
| 后置条件 | | 为每个订单生成一个默认订单到达单据，并显示 |
| TransferHallBL.modifyOrderArrival | 语法 | | Public ArrayList<OrderArriveList> modifyOrderArrival(ArrayList<OrderArrival> list,ArrayList<String> input) |
| 前置条件 | | 生成了默认到达单,用户修改了到达单信息 |
| 后置条件 | | 修改后的到达单信息保存在本地 |
| TransferHallBL.saveOrderArriveList | 语法 | | Public ResultMessage saveOrderArriveList() |
| 前置条件 | | 用户确认保存订单到达单 OrderArrivalPO |
| 后置条件 | | 系统保存订单到达单到数据文件 |
| TransferHallBL.createSend | 语法 | | Public ArrayList<SendListVO> createSend(ArrayList<OrderArriveListPO>) |
| 前置条件 | | 系统已经保存订单到达单 |
| 后置条件 | | 系统生成派件单，并显示 |
| TransferHallBL.saveSend | 语法 | | Public ResultMessage saveSend() |
| 前置条件 | | 用户要求保存派件单 |
| 后置条件 | | 系统保存派件单SendPO |
|  |  | |  |
| 分拣和装车 | | | |
| TransferHallBl.showNewOrder | 语法 | | public ArrayList<OrderVO> showNewOrder(long hallNum); |
|  | 前置条件 | | 业务员正常登陆，可以获取所在营业厅编号 |
|  | 后置条件 | | 系统显示所有未分拣订单信息 |
| TransferHallBL.createEntruckList |  | | **public** TransferListVO createEntruckList(); |
|  | 前置条件 | | 已经确认保存一张装车单 |
|  | 后置条件 | | 生成一张发往相同目的地的装车单，保存在本地文件中 |
| TransferHallBL.modifyEntruckList | 语法 | | public ResultMessage modifyEntruckList(EntruckListVO list,ArrayList<String> input); |
|  | 前置条件 | | 用户输入了装车单信息 |
|  | 后置条件 | | 系统修改本地装车单信息 |
| TransferHallBL.sort(OrderVO order) | 语法 | | public ResultMessage sort(OrderVO order); |
|  | 前置条件 | | 用户输入了一个新订单 |
|  | 后置条件 | | 系统根据该订单目的地将该订单信息加入对应本地装车单 |
| TransferHallBL.deleteOrder | 语法 | | public ResultMessage deleteOrder(OrderVO order); |
|  | 前置条件 | | 用户选择删除本地装车单上一条订单 |
|  | 后置条件 | | 系统将该订单从本地装车单上删除 |
| TransferHallBL.completeEntruckList | 语法 | | public EntruckListVO completeEntruckList(EntruckListPO list); |
|  | 前置条件 | | 一个本地装车单已满 |
|  | 后置条件 | | 系统将该装车单状态设置为已满 |
| TransferHallBL.getEntruckList | 语法 | | public EntruckListVO getEntruckList(long code); |
|  | 前置条件 | | 用户选择一张本地装车单 |
|  | 后置条件 | | 系统显示选中装车单 |
|  | | | |
| 需要的服务（需接口） | | | |
| TransferData.search(POType type,long code) | | 根据汽运编号查找装车单 | |
| TransferData.add (POType type,DataPO po) | | 系统保存装车单到数据文件 | |
| OrderService.modify(long orderCode,StockStatuse state) | | 根据单号，修改订单状态信息 | |

## 数据层的分解

* UserDataService:

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| getUserPO | 语法 | public UserPO getUserPO(long ID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找并返回用户数据PO |
| setUserPO | 语法 | public boolean setUserPO(UserPO user) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找用户PO并更改 |

* OrderDataService:

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| getOrderPO | 语法 | public OrderPO getOrderPO (long ID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找并返回用户数据PO |
| setOrderPO | 语法 | public boolean setOrderrPO(OrderPO order) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找用户PO并更改,若无则添加 |
| addOrderPO | 语法 | public boolean addOrderrPO(OrderPO order) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 查找用户PO并更改,若有则更改 |

* FinancialDataService:

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| FinancialDS.Search | 语法 | public AccountPO find(long id) throws RemoteException; | |
| 前置条件 | 无 | |
| 后置条件 | 按ID进行查找返回相应的FinancialPO结果 | |
| FinancialDS.insert | 语法 | public void insert(AccountPO po) throws RemoteException; | |
| 前置条件 | 同一个ID的po在Mapper中不存在 | |
| 后置条件 | 在数据库中增加一个po记录 | |
| FinancialDS.delete | 语法 | public void delete(AccountPO po) throws RemoteException; | |
| 前置条件 | 在数据库中存在同样ID的po | |
| 后置条件 | 更新一个po | |
| FinancialDS.update | 语法 | public void update(AccountPO po) throws RemoteException; | |
| 前置条件 | 在数据库中存在同样ID的po | |
| 后置条件 | 更新一个po | |
| FinancialDS.init() | 语法 | public void init() throws RemoteException; | |
|  | 前置条件 | 无 | |
|  | 后置条件 | 初始化持续化数据库 | |
| FinancialDS.finish() | 语法 | public void finish() throws RemoteException; | |
|  | 前置条件 | 无 | |
|  | 后置条件 | 结束持续化数据库的使用 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| FinancialDS.find(int id) | | | 根据ID进行查找单一持久化对象 |
| FinancialDS.find(String field,int value) | | | 根据字段名和值进行查找多个持久化对象 |
| FinancialDS.insert(FinancialPO po) | | | 插入单一持久化对象 |
| FinancialDS.delete(FinancialPO po) | | | 删除单一持久化对象 |
| FinancialDS.update(FinancialPO po) | | | 更新单一持久化对象 |
| FinancialDS.getFinancialDatabase | | | 得到Financial数据库的服务的引用 |
| FinancialDS.init() | | | 初始化持久化数据库 |
| FinancialDS.finish() | | | 结束持久化数据库的使用 |

* CompanyDataService:

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CompanyDataService.find  Salary | 语法 | public SalaryPo findSalary(Department department) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按照员工机构进行查找返回的SalaryPo结果 |
| CompanyDataService.add  Salary | 语法 | public void addSalary(SalaryPO salarypo) throws RemoteException |
| 前置条件 | 同样的stafftype在数据库中不存在 |
| 后置条件 | 在数据库中增加一个salarypo |
| CompanyDataService.update  Salary | 语法 | public void updateSalary(SalaryPO salarypo) throws RemoteException |
| 前置条件 | 在数据库中存在同样stafftype的salarypo |
| 后置条件 | 更新一个salarypo |
| CompanyDataService.find  Staff | 语法 | public StaffPo findStaff(long id)  throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按照员工id进行查找返回的StaffPo结果 |
| CompanyDataService.add  Staff | 语法 | public void insertStaff(StaffPO staffpo)  throws RemoteException |
| 前置条件 | 同样的id在数据库中不存在 |
| 后置条件 | 在数据库中增加一个staffpo记录 |
| CompanyDataService.delete  Staff | 语法 | public void deleteStaff(StaffPO staffpo)  throws RemoteException |
| 前置条件 | 在数据库中存在同样id的staffpo |
| 后置条件 | 删除这个staffpo |
| CompanyDataService.update  Staff | 语法 | public void updateStaff(StaffPO staffpo)  throws RemoteException |
| 前置条件 | 在数据中存在同样id的staffpo |
| 后置条件 | 更新这个staffpo |
| CompanyDataService.findCityInfo | 语法 | public CityPo findCityInfo(String fromCity,String toCity ) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据始末城市查找返回相应的CityInfoPo结果 |
| CompanyDataService.addCityInfo | 语法 | public void addCityInfo(CityPO citypo) |
| 前置条件 | 同样始末城市的的citypo在数据库中不存在 |
| 后置条件 | 在数据库中增加一个citypo记录 |
| CompanyDataService.update  CityInfo | 语法 | public void updateCityInfo(CityPO citypo) |
| 前置条件 | 在数据库中存在同样始末城市的citypo |
| 后置条件 | 更新这个citypo |
| CompanyDataService.find  SituationFormInfo | 语法 | public ArrayList<MoneyDataPO> findSituationFormInfo() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回ArrayList<MoneyDataPO>结果 |
| CompanyDataService.find  EarningFormInfo | 语法 | public EarningPo findSituationForm(String date) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按照日期进行查找返回EarningPO结果 |
| CompanyDataService.find  Data | 语法 | public DataPO findData(long id)  throws RemoteExce |
| 前置条件 | 无 |
| 后置条件 | 按单据id进行查找返回相应的DataPO结果 |
| CompanyDataService.add  Data | 语法 | public void addData(DataPO Datapo)  throws RemoteException |
| 前置条件 | 同样id的Datapo在数据库中不存在 |
| 后置条件 | 在数据库中增加Datapo这个记录 |
| CompanyDataService.delete  Data | 语法 | public void deleteData(DataPO Datapo)  throws RemoteException |
| 前置条件 | 在数据库中存在同样id的Datapo |
| 后置条件 | 删除一个Datapo记录 |
| CompanyDataService.update  Data | 语法 | public void updateData(DataPO Datapo)  throws RemoteException |
| 前置条件 | 在数据库中存在同样id的staffpo |
| 后置条件 | 更新这个staffpo |
| CompanyDataService.init | 语法 | public void init() |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| CompanyDataService.finish | 语法 | public void finish() |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

* StorageData:

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StorageData. adjustStorage | 语法 | Public boolean adjustStorage(String source,String target,int amout) |
| 前置条件 | 用户发起了一次仓库调整活动 |
| 后置条件 | 根据传入的参数调整仓库 |
| StorageData.getCurrentStorageInfo  StorageData.search(POtype type,DataPO data)  （获取仓库信息） | 语法 | Public PositionInfoPO getCurrentStorageInfo() |
| 前置条件 | 用户发起一次库存盘点活动，调用本方法 |
| 后置条件 | 返回给调用者当前库存全部信息 |
| StorageData.addStorageInfoInfo | 语法 | Public boolean addPositionInfo(PositionInfoPO info) |
| 前置条件 | 用户确认保存库存盘点信息 |
| 后置条件 | 系统保存该信息到数据文件 |
| StorageData.getStorageList  (获取出库单/入库单) | 语法 | Public StorageListPO getStorageList(long code) |
| 前置条件 | 单号合法 |
| 后置条件 | 返回给调用者该对应单据的PO，若未找到，返回NULL |
| StorageData.getStorageList  (获取出库单/入库单) | 语法 | Public StorageListPO getStorageList(String startTime,String endTime) |
|  | 前置条件 | Endtime 晚于 startTime |
| 后置条件 | 返回给调用者该对应单据的PO列表，若未找到，返回NULL |
| StorageData.checkSpace | 语法 | Public int[] checkSpace() |
| 前置条件 | 用户发起仓库分拣活动 |
| 后置条件 | 返回各区域剩余空间（位置个数） |
| StorageData.getPosition | 语法 | Public String getPosition(char area) |
| 前置条件 | 获得需要的位置所在区域area，且位置空间充足 |
| 后置条件 | 返回一个空位置信息 area-shelf-row-point |
| StorageData.addStoregeList | 语法 | Public boolean addStorageList(StorageListPO storagepo) |
| 前置条件 | 有一个出库单或入库单生成，需要保存到数据文件中 |
| 后置条件 | 将该PO数据保存到数据文件中 |
| StorageData.updateStorage | 语法 | Public boolean updateStorage(StorageListPO e) |
| 前置条件 | 生成了入库单或出库单 |
| 后置条件 | 系统根据出入库单修改库存信息 |
| StorageData.initStorage | 语法 | public boolean initStorage(int plane,int train,int truck,int flexible); |
| 前置条件 | 用户输入正确初始化密码 |
| 后置条件 | 将仓库空间重新分配 |
| StorageData.getStorageInfo | 语法 | public StorageInfoPO getStorageInfo(String destination); |
| 前置条件 | 提供了货物目的地 |
| 后置条件 | 系统返回响应货物信息StorageInfoPO |

* TransferData

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| TransferData.getDelivryList | 语法 | Public DeliveryListPO getShortDeliveryList(long code) |
| 前置条件 | 提供一个装车单或中转单号 |
| 后置条件 | 系统查找并返回对应单据 |
| TransferData.addDeliverryList | 语法 | Public boolean addDeliveryList(DeliveryList PO list) |
| 前置条件 | 提供一个DeliveryListPO |
| 后置条件 | 系统将单据信息保存到数据文件 |
| TransferData.getArriveList | 语法 | Public ArriveListPO getArriveList(long code) |
| 前置条件 | 提供一个到达单号 |
| 后置条件 | 系统查找并返回一个到达单 |
| TransferData.addArriveList | 语法 | Public boolean addArriveList(ArriveListPO list) |
| 前置条件 | 提供一个ArriveListPO |
| 后置条件 | 系统将单据信息保存到数据文件中 |

# 信息视角

## 数据持久化对象

银行账户信息AccountPO：

|  |
| --- |
| package data.po;  import java.io.Serializable;  public class AccountPO implements Serializable{    /\*\*  \*  \*/  private static final long serialVersionUID = -5344511556034467117L;    String name;  double money;    public AccountPO(String name, double money) {  this.money = money;  this.name = name;  }  public String getName() {  return name;  }    public double getMoney() {  return money;  }  public void setName(String name) {  this.name = name;  }    public void setMoney(double money) {  this.money = money;  }  } |

到达单信息ArrivalPO：

|  |
| --- |
| package data.po;  import data.enums.BillType;  public class ArrivalPO extends DataPO {  private static final long serialVersionUID = -7918393384250479651L;  long officeIndex; // 中转中心/营业厅编号，根据位数判断类型（营业厅6位编号，中转中心4位编号）  // Format: yyyy/mm/dd  String arrivalDate;  long ID;    public ArrivalPO(long officeIndex, String arrDate, long ID) {  super(BillType.ARRIVAL);  this.officeIndex = officeIndex;  this.arrivalDate = arrDate;  this.ID = ID;  }    public long getOfficeIndex() {  return officeIndex;  }    public String getArrivalDate() {  return arrivalDate;  }    public long getID() {  return ID;  }    public void setOfficeIndex(long offIndex) {  this.officeIndex = offIndex;  }    public void setArrivalDate(String date) {  this.arrivalDate = date;  }    public void setID(long ID) {  this.ID = ID;  }  } |

成本收益表CostBenefitPO：

|  |
| --- |
| package data.po;  import java.io.Serializable;  public class CostBenefitPO implements Serializable{    double allIncome;  double allPay;  double allProfit;  public CostBenefitPO(double allIncome, double allPay, double allProfit) {  this.allIncome = allIncome;  this.allPay = allPay;  this.allProfit = allProfit;  }  public double getAllIncome() {  return allIncome;  }  public double getAllPay() {  return allPay;  }  public double getAllProfit() {  return allProfit;  }  public void setAllIncome(double allIncome) {  this.allIncome = allIncome;  }  public void setAllPay(double allPay) {  this.allPay = allPay;  }  public void setAllProfit(double allProfit) {  this.allPay = allProfit;  }  } |

司机信息DriverInfoPO：

|  |
| --- |
| package data.po;  import data.enums.BillType;  import utils.IDGenerator;  public class DriverInfoPO extends DataPO {  private static final long serialVersionUID = 7713110376436575616L;  long IDCard;  String name;  boolean engaged;  boolean gender /\* true = female, false = male \*/;  String phoneNum;  String bornDate /\* yyyy/mm/dd \*/;  String timeLimit /\* yyyy/mm/dd \*/;    public DriverInfoPO(long ID, String name, boolean gender, String phoneNum, String bornDate, String timeLimit) {  this.serialNum = IDGenerator.*getNextID*(BillType.ORDER);  this.IDCard = ID;  this.phoneNum = phoneNum;  this.bornDate = bornDate;  this.timeLimit = timeLimit;  this.gender = gender;  engaged = false;  }    public long getSerialNum() { return serialNum; }  public long getIDCard() { return IDCard; }  public boolean isEngaged() { return engaged; }  public boolean getGender() { return gender; }  public String getPhoneNum() { return phoneNum; }  public String getBornDate() { return bornDate; }  public String getTimeLimit() { return timeLimit; }  public void setIDCard(long IDCard) { this.IDCard = IDCard; }  public void setEngaged(boolean engaged) { this.engaged = engaged; }  public void setPhoneNum(String phoneNum) { this.phoneNum = phoneNum; }  public boolean setBornDate(String bornDate) {  if (bornDate.matches("[0-9]{4}[/][0-9]{2}[/][0-9]{2}")) {  this.bornDate = bornDate;  return true;  }  return false;  }  public boolean setTimeLimit(String timeLimit) {  if (timeLimit.matches("[0-9]{4}[/][0-9]{2}[/][0-9]{2}")) {  this.timeLimit = timeLimit;  return true;  }  return false;  }  } |

装车单信息EntruckPO：

|  |
| --- |
| package data.po;  /\*\*  \* 装车单  \* **@author** Mouse  \*/  import java.io.Serializable;  public class EntruckPO extends DataPO {    public static final int TRANS = 1, DESTINATION = 2, MONITOR = 3, ESCORT = 4;    String loadingDate;  long transID, destID, vehicleID, monitorID, escortID;  int order\_amount;  long orderList[];    public EntruckPO(int amount) {  this.order\_amount = amount;  orderList = new long[amount];  }    public long getID(int type) {  switch (type) {  case TRANS:  return transID;  case DESTINATION:  return destID;  case MONITOR:  return monitorID;  case ESCORT:  return escortID;  default:  return 0;  }  }    public void setID(int type, long ID) {  switch (type) {  case TRANS:  transID = ID;  case DESTINATION:  destID = ID;  case MONITOR:  monitorID = ID;  case ESCORT:  escortID = ID;  default:  }  }  public long getTransID() {  return transID;  }  public long getDestID() {  return destID;  }  public long getVehicleID() {  return vehicleID;  }  public long getMonitorID() {  return monitorID;  }  public long getEscortID() {  return escortID;  }  public void setTransID(long transID) {  this.transID = transID;  }  public void setDestID(long destID) {  this.destID = destID;  }  public void setVehicleID(long vehicleID) {  this.vehicleID = vehicleID;  }  public void setMonitorID(long monitorID) {  this.monitorID = monitorID;  }  public void setEscortID(long escortID) {  this.escortID = escortID;  }  public void setAmount(int amount) {  this.order\_amount = amount;  orderList = new long[amount];  }    public long[] getOrderList() {  return this.orderList;  }    public void setOrderList(long[] list) {  this.orderList = list;  }  } |

机构信息InstitutionPO：

|  |
| --- |
| package data.po;  /\*\*  \*  \* Created by Mouse on 2015/10/24 0024.  \*/  public class InstitutionPO extends DataPO {  String name;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  } |

物流信息表LogisticInfoPO：

|  |
| --- |
| package data.po;  import java.io.Serializable;  import java.util.\*;  public class LogisticInfoPO extends DataPO implements Serializable{  private static final long serialVersionUID = 4149146341000434330L;  List<LogisticInfo> infoList;    public LogisticInfoPO(long serialNum) {  this.serialNum = serialNum;  infoList = new ArrayList<LogisticInfo>();  }    public void addInfo(String time, String location) {  infoList.add(new LogisticInfo(time, location));  }    @Override  public String toString() {  String result = "";  Iterator<LogisticInfo> tmp = infoList.iterator();  while (tmp.hasNext()) {  result += tmp.next().toString() + "\n\n";  }  return result;  }  public long getSerialNum() {  return serialNum;  }    // ������Ϣ�����ڴ˴�  public class LogisticInfo {  /\*\*  \* FORMAT = "yyyy/mm/dd hh:mm:ss"  \*/  String timestamp;  String location;    public LogisticInfo(String timestamp, String location) {  this.timestamp = timestamp;  this.location = location;  }    @Override  public String toString() {  String result = timestamp + " 货物到达 " + location;  return result;  }    public void setTimestamp(String timestamp) {  this.timestamp = timestamp;  }    public void setLocation(String location) {  this.location = location;  }  }  } |

订单信息OrderPO：

|  |
| --- |
| package data.po;  import data.enums.BillType;  import data.vo.OrderVO;  import utils.IDGenerator;  public class OrderPO extends DataPO {  // 收/寄件人信息，s开头表示寄件人（sender），r开头表示收件人（receiver）  String sname, saddress, scompany, sphone,  rname, raddress, rcompany, rphone;  long destID;  // 货物数量  int stockNum;  // 重量/kg、体积/cm^3  double weight, volume;  // 字符串保存每一件货品的类型（名称）  String stockType[];  // 计算出的运费  double fee;  // 计算出的预计送达时间  int evaluatedTime;  public void setDestID(long destID) {  this.destID = destID;  }  public long getDestID() {  return destID;  }  public OrderPO() {  serialNum = IDGenerator.*getNextID*(BillType.ORDER);  }    public OrderPO(int stockNum) {  serialNum = IDGenerator.*getNextID*(BillType.ORDER);  stockType = new String[stockNum];  }    public long getSerialNum() {  return serialNum;  }    public String getSInfo(int infoType) {  switch (infoType) {  case 0:  return sname;  case 1:  return saddress;  case 2:  return scompany;  case 3:  return sphone;  case 4:  return rname;  case 5:  return raddress;  case 6:  return rcompany;  case 7:  return rphone;  default:  return "";  }  }    public void setInfo(int infoType, String info) {  switch (infoType) {  case 0:  sname = info;  case 1:  saddress = info;  case 2:  scompany = info;  case 3:  sphone = info;  case 4:  rname = info;  case 5:  raddress = info;  case 6:  rcompany = info;  case 7:  rphone = info;  default:  }  }  public void setInfo(OrderVO order) {  this.sname = order.sname;  this.saddress = order.saddress;  this.scompany = order.scompany;  this.sphone = order.sphone;  this.rname = order.rname;  this.raddress = order.raddress;  this.rcompany = order.rcompany;  this.rphone = order.rphone;  }  /\*\*  \*  \* **@param** num number of the stocks  \* **@param** type a String array that stores the name of the stocks. be adviced that the length of <code>type</code> must equal <code>IDCard</code>  \*/  public void setStockNum(int num, String type[]) {  this.stockNum = num;  stockType = new String[num];  }    public double getWeight() {  return weight;  }    public void setWeight(double weight) {  this.weight = weight;  }    public double getVolume() {  return volume;  }    public void setVolume(double volume) {  this.volume = volume;  }  public double getFee() {  return fee;  }  public void setFee(double fee) {  this.fee = fee;  }    public void generateFee() {  // **TODO**:  }  public int getEvaluatedTime() {  return evaluatedTime;  }  public void setEvaluatedTime(int time) {  evaluatedTime = time;  }  } |

人员信息StuffPO：

|  |
| --- |
| package data.po;  /\*\*  \*  \* Created by Mouse on 2015/10/24 0024.  \*/  public class StuffPO {  public long institution;  public String name;  public boolean gender /\* 0 - male , 1 - female \*/;  public long phoneNum;  public long getInstitution() {  return institution;  }  public long getPhoneNum() {  return phoneNum;  }  public String getName() {  return name;  }  public boolean getGender() {  return gender;  }  public void setInstitution(long institution) {  this.institution = institution;  }  public void setPhoneNum(long phoneNum) {  this.phoneNum = phoneNum;  }  public void setName(String name) {  this.name = name;  }  public void setGender(boolean gender) {  this.gender = gender;  }  } |

用户信息UserPO：

|  |
| --- |
| package data.po;  import data.enums.POType;  import data.enums.UserRole;  import utils.IDGenerator;  import utils.PasswordHelper;  public class UserPO extends DataPO {  private static final long serialVersionUID = 2205631303533979093L;  String name;  /\*\*  \* store password's MD5 checksum  \*/  String password\_signature;  UserRole role;  public UserPO(String name, String pswd, UserRole role) {  this.serialNum = IDGenerator.*getNextID*(POType.USER);  this.name = name;  this.password\_signature = PasswordHelper.*generateMD5Checksum*(pswd);  this.role = role;  }  public long getSerialNum() {  return serialNum;  }  public String getName() {  return name;  }  public UserRole getRole() {  return role;  }  public String getPassword() {  return password\_signature;  }  public void setRole(UserRole role) {  this.role = role;  }  //  public void setPassword(String pswd) {  this.password\_signature = PasswordHelper.*generateMD5Checksum*(pswd);  }  } |

VehicleInfoPO：

|  |
| --- |
| package data.po;  public class VehicleInfoPO extends DataPO{  private static final long serialVersionUID = -8365240623519578242L;  long ID;  String license;  // Format: yyyy/mm/dd  String dutyDate;    public VehicleInfoPO(long ID, String license, String dutyDate) {  this.ID = ID;  this.license = license;  this.dutyDate = dutyDate;  }    public long getID() { return ID; }  public String getLicense() { return license; }  public String getDutyDate() { return dutyDate; }    public void setID(long ID) {  this.ID = ID;  }  public void setLicense(String license) {  this.license = license;  }  public boolean setDutyDate(String dutyDate) {  if (dutyDate.matches("[0-9]{4}[/][0-9]{2}[/][0-9]{2}")) {  this.dutyDate = dutyDate;  return true;  }  return false;  }    } |

## 文件持久化格式

物流管理系统的所有po的属性值以字段的方式保存在文件中。

每一种表单数据，保存在一个ArrayList中，用序列化的方式保存在\*.dat文件中。文件名为PO名。

服务器每隔十分钟将会把内存中的数据保存在磁盘上。每次启动时从内存中读取数据。