软件体系结构描述文档

学校： 南京大学

院系： 软件学院

编写小组： 2014届软工2小组33

目录

1 引言………………………..2

1.1编写目的………….2

1.2词汇表……………..2

1.3参考资料………….2

2 产品概述…………………2

3 逻辑模型…………………3

4 组合视角

5 接口视角

6 信息视角

1 引言

1.1编制目的

本报告作为快递物流系统的软件体系结构描述文档，详细完成对该系统的概要设计，使系统分析人员在此基础上进一步提出详细设计说明和完成后续设计和开发工作。

本报告的预期读者为开发人员、测试人员、最终用户以及项目管理人员。

1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统 | 为本次工程的开发产品 |
| presentation | 展示层 | 为用户界面 |
| systemlogic | 系统逻辑层 | 处理业务逻辑 |
| data | 数据层 | 持久化数据 |
| Java RMI | 远程调用方法 | 为实现客户端与服务端之地的信息交互 |
| vo | 值对象 | 用于业务层之间的数据传递 |
| po | 持久对象 | 与数据库中的表对应的Java对象 |

1.3导读

该报告分为六个部分，第一部分是引言，主要是介绍文档的基本信息；第二部分是产品概述，是快递物流系统的概括性描述；第三、第四、第五和第六部分分别是逻辑视角、组合视角、接口视角、信息视角，从多个视角分别进行软件结构描述。

1.4文档约定

对文档的的任何改动都应经过质量保证人员的统一，详细改动由小组成员共同商议。

1.5感谢

本用例文档由小组成员黄迪璇编写，何永俊负责审阅，详细描述由何永俊、许玥琪、邓逸鹏和黄迪璇共同完成，感谢他们的辛勤付出。

1.6参考资料

《软件工程与计算（卷二）——软件开发的技术基础》 机械工程出版社

《快递物流系统软件需求规格说明文档v0.5》

《快递物流系统软件体系描述文档v0.2》

2 产品概述

快递物流系统是为一本地民营物流企业开发的业务系统，开发的目的是帮助该企业处理日常的重点业务，包括收发快件，库存管理，财务操作和人员机构管理。

通过快递物流系统的应用，期望系统上线运行六个月后，保持合理库存，提高用户服务体验，增加业务额，提高财务人员工作效率，为经理的决策做支持。

3 逻辑模型

快递物流系统中，小组讨论决定后采用分层体系结构，将系统分为3层（展示层、系统逻辑层、数据层）能够很好地表现整个高层抽象。展示层包含GUI页面的实现，系统逻辑层包含系统逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示

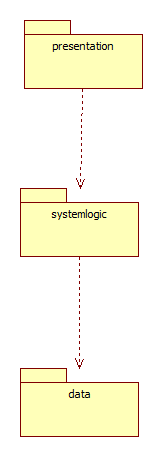


图1 参照体系结构风格的包图表达逻辑视角

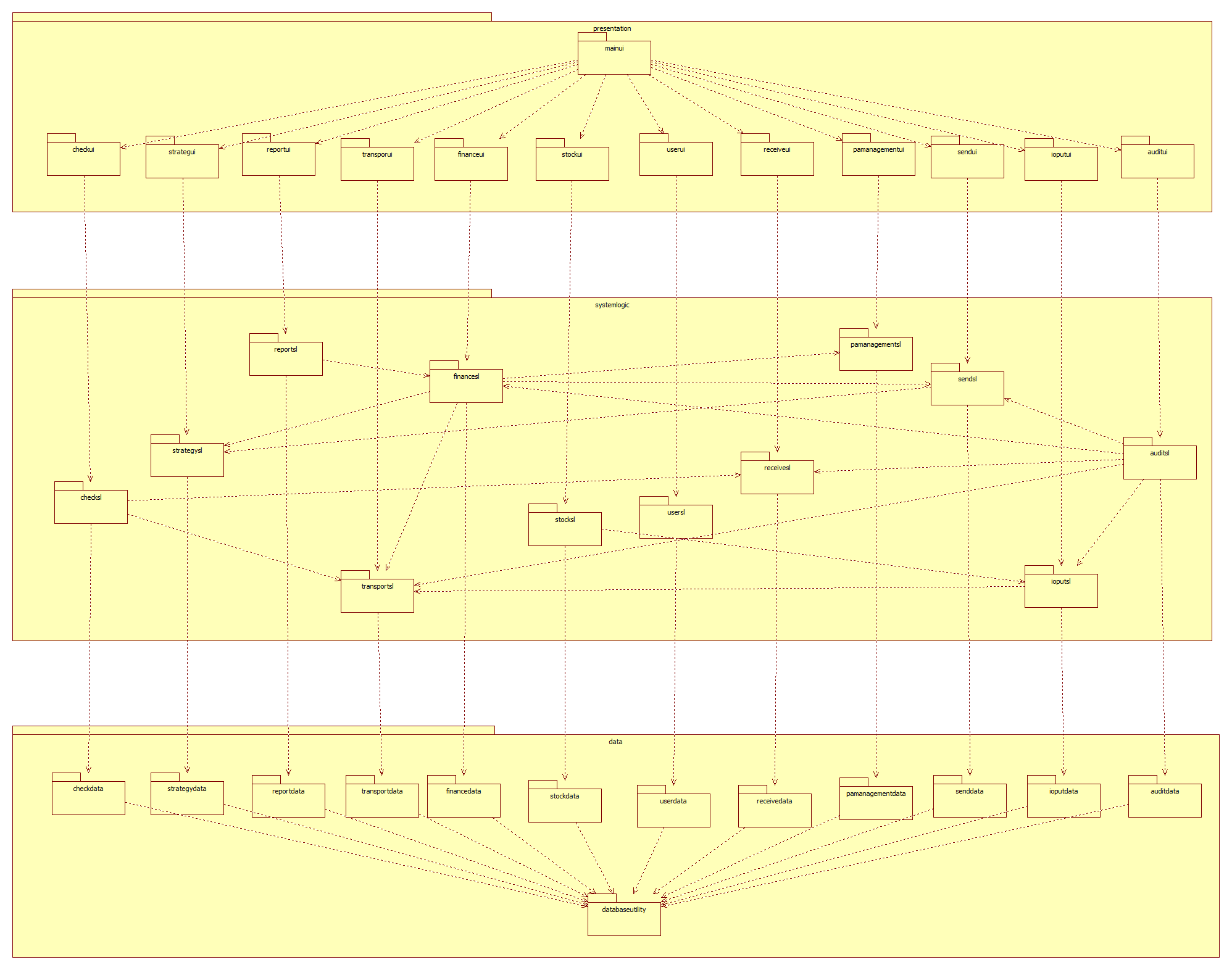


图2 软件体系结构逻辑设计方案

4 组合视角

4.1开发包图

快递物流系统的最终开发包图设计如表1所示

表1 快递物流系统管理系统的最终开发包设计

|  |  |
| --- | --- |
| 开发（物流）包 | 依赖的其他开发包 |
| mainui | checkui, strategyui, reportui, transportui, financeui, stockui, userui, receiveui, pamanagementui, sendui, ioputui, auditui |
| checkui | checkslservice, 界面类库包, vo |
| checkslservice |  |
| checksl | checkslservice, checkdataservice, po, receivesl, transportsl, |
| checkdataservice | Java RMI, po |
| checkdata | databaseutility, po, checkdataservice |
| strategyui | strategyslservice, 界面类库包, vo |
| strategyslservice |  |
| strategysl | strategyslservice, strategydataservice, po |
| strategydataservice | Java RMI, po |
| strategydata | databaseutility, po, strategydataservice |
| reportui | reportslservice, 界面类库包, vo |
| reportslservice |  |
| reportsl | reportslservice, reportdataservice, po, financesl |
| reportdataservice | Java RMI, po |
| reportdata | databaseutility, po, reportdataservice |
| transportui | transportslservice, 界面类库包, vo |
| transportslservice |  |
| transportsl | transportslservice, transportdataservice, po |
| transportdataservice | Java RMI, po |
| transportdata | databaseutility, po, transportdataservice |
| financeui | financeslservice, 界面类库包, vo |
| financeslservice |  |
| financesl | financeslservice, financedataservice, po, strategysl, transportsl, sendsl, pamanagementsl |
| financedataservice | Java RMI, po |
| financedata | databaseutility, po, financedataservice |
| stockui | stockslservice, 界面类库包, vo |
| stockslservice |  |
| stocksl | stockslservice, stockdataservice, po, ioputsl |
| stockdataservice | Java RMI, po |
| stockdata | databaseutility, po, stockdataservice |
| userui | userslservice, 界面类库包, vo |
| userslservice |  |
| usersl | userslservice, uesrdataservice, po |
| uesrdataservice | Java RMI, po |
| userdata | databaseutility, po, uesrdataservice |
| receiveui | receiveslservice, 界面类库包, vo |
| receiveslservice |  |
| receivesl | receiveslservice, receivedataservice, po |
| receivedataservice | Java RMI, po |
| receivedata | databaseutility, po, receivedataservice |
| pamanagementUi | pamanagementslservice, 界面类库包, vo |
| pamanagementslservice |  |
| pamanagementsl | pamanagementslservice pamanagementdataservice, po |
| pamanagementdataservice | Java RMI, po |
| pamanagementdata | databaseutility, po, pamanagementdataservice |
| sendui | sendslservice, 界面类库包, vo |
| sendslservice |  |
| sendsl | sendslservice, senddataservice, po, strategysl, |
| senddataservice | Java RMI, po |
| senddata | databaseutility, po, senddataservice |
| ioputui | ioputslservice, 界面类库包, vo |
| ioputslservice |  |
| ioputsl | ioputslservice, ioputdataservice, po, transportsl |
| ioputdataservice | Java RMI, po |
| ioputdata | databaseutility, po, ioputdataservice |
| auditui | auditslservice, 界面类库包, vo |
| auditslservice |  |
| auditsl | auditslservice, auditdataservice,po, sendsl, financesl, receivesl, transportsl, ioputsl |
| auditdataservice | Java RMI, po |
| auditdata | databaseutility, po, auditdataservice |
| vo |  |
| po |  |
| utilitysl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

快递物流系统客户端开发包图如图3所示，服务端开发端包图如图4所示



图3 快递物流系统客户端开发包图



图4 快递物流系统服务端开发包图

4.2运行时进程

在快递物流系统中，会有多个客户端进程和一个服务端进程，其进程图如图5所示。结合部署图，客户端进程是在客户端机器上运行，服务端进程在服务端机器上运行，我们机构通过Java库的RMI机制实现客户端和服务端之间的联系。

RMI

Client 1

…………

…………

Server

RMI

Client n

…………

…………

图5 进程图

4.3物理部署

快递物流系统中客户端构件是放在客户端机器上，服务端构建是放在服务端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK 8.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要在独立部署。部署图如图6所示。

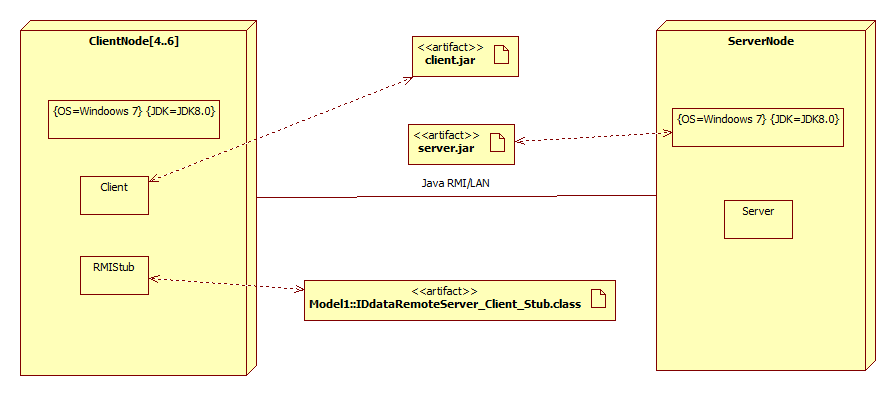


图6 部署图

5 接口视角

5.1通过对组合视角客户端和服务端开发包图的分解，在接口视角中，我们将客户端模块和服务端模块抽象成下面的图7和图8，其职责分别如表2和表3所示。

展示层

启动（初始化）模块

业务逻辑层

网络模块（RMI）

图7 客户端模块视图

启动（初始化）模块

网络模块（RMI）

数据层

图8 服务端模块

表2 客户端各层职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动（初始化）模块 | 负责初始化网络通信机制，同时启动展示层 |
| 展示层 | 基于图形界面（Java.Swing）的快递物流系统的客户端界面 |
| 业务逻辑层 | 对展示层的输入进行响应并进行业务逻辑处理，与服务端中的数据层通过RMI机制进行数据交互 |
| 网络模块 | 通过Java RMI机制寻找RMI服务 |

表3 服务端各层职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动（初始化）模块 | 负责初始化网络通信机制 |
| 数据层 | 基于数据库和序列化文件进行数据的持久化以及数据访问接口 |
| 网络模块 | 通过Java RMI机制开启RMI服务，注册RMI服务 |

每一层只能调用和接触下方的接口，层与层之间调用只能通过接口来进行，从而降低了系统的耦合度。层与层之间的接口调用如表4所示。

表4 层与层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
|  | 客户端暂时层 | 客户端业务逻辑层 |
|  | 客户端业务逻辑层 | 服务端数据层 |

我们通过机构管理用例来说明层与层之间的掉用，如图9所示。每一层之间都是商城依赖一个接口（请求服务），下层实现这个接口（提供服务），层与层之间都依赖同一个结构（依赖倒置原则），大大降低了层与层之间的耦合的。在机构管理用例中，AManagementService提供了AManagement界面所需要的所有业务逻辑功能；AManagement类继承AManagementService接口并实现其业务逻辑功能；AManagementDataService提供对数据库（序列化文件）的增删改查等操作；AManagementData实现AManagementDataService接口内的所有操作。

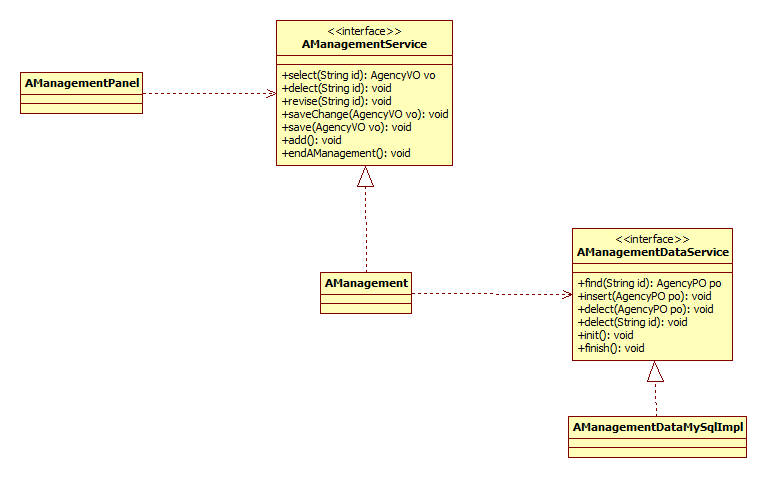


图9 机构管理用例之间调用的接口

5.2用户界面层的分解

根据功能需求的分析，系统存在35个用户界面：登录/查询界面、快递员主界面、营业厅业务员主界面、中转中心业务员主界面、中转中心库存管理人员主界面、总经理主界面、财务人员主界面、管理员主界面、订单输入界面、派件查看界面、收件输入界面、车辆信息管理界面、司机信息管理界面、派件管理界面、收款管理界面、装车管理界面、中转接受界面、装运管理界面、入库界面、出库界面、库存信息初始化界面、库存盘点界面、库区调整界面、库存查看界面、统计报表界面、日志查看界面、人员管理界面、机构管理界面、制定常量界面、审判单据界面、薪水策略界面、账户管理界面、成本管理界面、结算管理界面、帐目初始化界面、用户管理界面

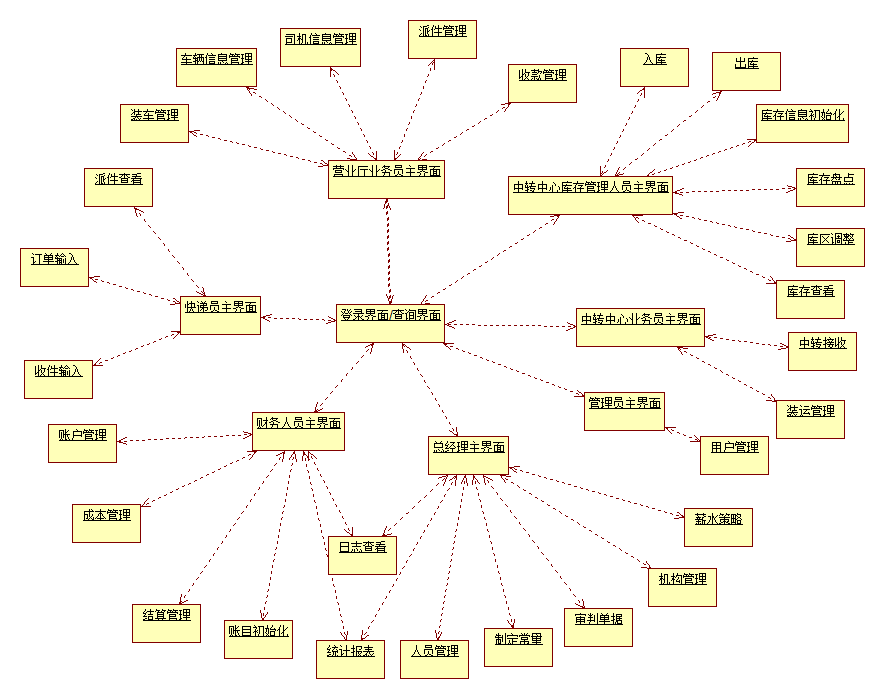


图10 用户界面的跳转

根据我们的设计，服务端不存在用户界面，用户界面类如图11所示。

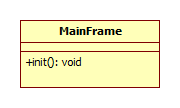


图11 用户界面类

5.2.1用户界面层模块的职责

如表5所示为用户界面层模块的职责

表5 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面的主Frame，负责界面的跳转 |

5.2.2用户界面层模块的接口规范

用户界面层模块的接口规范表6所示

表6 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Void init(); |
| 前置条件 | 无 |
| 后置条件 | 显示登录界面，以及其他界面的挑战 |

用户界面层所需要的服务接口如表7所示。

表7 用户界面层模块所需要的服务接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| systemlogicservice.LoginBLService | 登录界面的业务逻辑接口 |
| Systemlogicservice.\*BLservice | 每个界面都有一个相应的界面跳转 |

5.2.3用户界面模块设计原理

基于需求中要求的图形界面设计，将利用Java的Swing和AWT库来实现用户界面。

5.3业务逻辑层的分解

业务逻辑层包括多个针对界面业务逻辑处理对象，实现systemLogicService中的接口方法。详见5.3.1业务逻辑层模块的职责

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8所示。

表8 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| auditbl | 负责实现审判单据界面所需要的服务 |
| strategybl | 负责实现制定常量界面和薪水策略界面所需要的服务 |
| pamanagement | 负责实现人员管理界面和机构管理界面所需要的服务 |
| checkbl | 负责物流信息查询界面所需要的任务 |
| financebl | 负责实现结算管理、成本管理、账户管理和账目初始化界面所需要的服务 |
| usersl | 负责主页面登录、用户管理、日志管理界面所需要功能 |
| stocksl | 负责库存信息管理界面所需要的服务 |
| ioputsl | 负责出入库管理界面所需要的服务 |
| sendsl | 负责订单输入、收款管理界面所需要的服务 |
| receivesl | 负责收件输入、派件管理界面所需要的任务 |
| transportsl | 负责装车管理、装运管理、中转管理界面所需要的任务 |
| report | 负责统计报表界面所需要的任务 |

业务逻辑层模块所包含的类如表9所示

|  |  |
| --- | --- |
| 模块 | 包含的类 |
| auditsl | Audit |
| strategysl | SalaryStrategy, Constant |
| pamanagement | PManagement, AManagement |
| checksl | Check |
| financesl | Balance,Cost,AccountManagement,AccountInitialize |
| user | User,Log |
| stocksl | StockInfo, StockAdjust |
| ioputsl | InputStock,OutputStock |
| sendsl | OrderReceive,GatheringReceive |

auditsl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Audit.select | 语法 | Public DecumentVO select(String id); |
| 前置条件 | 在审判单据的过程中，该单据被鼠标点击 |
| 后置条件 | 返回该单据的详细内容 |
| Audit.adopt | 语法 | public void adopt(String id); |
| 前置条件 | 在审判单据过程中，通过审核按钮被点击 |
| 后置条件 | 将单据的状态由由提交状态更改为审批后状态，持久化更新单据对象的数据，结束本次审判单据回合 |
| Audit.revise | 语法 | public void revise(String id); |
| 前置条件 | 在审判单据过程中，单据已经被选择，修改按钮被点击 |
| 后置条件 | 调用单据对象的修改方法 |
| Audit.saveChange | 语法 | public void saveChange(String id); |
| 前置条件 | 在修改单据的过程中，保存按钮被点击 |
| 后置条件 | 调用单据对象的保存方法。持久化更新单据对象的数据 |
| Audit.selectNum | 语法 | public ArrayList<DecumentVO> selecNumt(String []ids); |
| 前置条件 | 在审判单据的过程中，单据的多选按钮被点击 |
| 后置条件 | 记录所选择的单据 |
| Audit.getAllDecument | 语法 | Public ArrayList<DecumentVO> getDecumnet(); |
| 前置条件 | 开启新一轮的审判单据 |
| 后置条件 | 返回所有待审核单据的信息 |
| Audit.endAudit | 语法 | Public void endAudit(); |
| 前置条件 | 在审判单据的过程中，结束按钮被点击 |
| 后置条件 | 结束审判单据，持久化更新涉及的领域对象数据 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| AuditDataService.find(String id) | 根据id进行查找单一持久化对象 |
| AuditDataService.finds(String[]ids) | 根据ids查找多个持久化对象 |
| AuditDataService.findAll() | 返回数据库内所有的持久化对象 |
| AuditDataService.insert(DocumentPO po) | 插入单一持久化对象 |
| AuditDataService.delect(DocumentPO po) | 删除单一持久化对象 |
| AuditDataService.delect(String id) | 根据id删除单一持久化对象 |
| AuditDataService.update(ArrayList<Decoument> decuments) | 更新多个持久化对象 |
| AuditDataService.update(DocumentPO po) | 更新单一持久化对象 |

SalaryStrategy的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| SalaryStrategy.getAllSalaryStrategy | 语法 | public ArrayList<SalaryVO> getAllSalaryStrategy(); |
| 前置条件 | 开启新的一轮制定薪水策略 |
| 后置条件 | 返回先前的所制定的薪水策略 |
| SalaryStrategy.newSalary | 语法 | public void newSalaryVO(); |
| 前置条件 | 开启新的一轮制定薪水策略，制定新的薪水策略的按钮被点击 |
| 后置条件 | 返回薪水策略制定表 |
| SalaryStrategy.save | 语法 | public void save(SalaryVO vo); |
| 前置条件 | 在制定薪水策略的过程中，保存按钮被点击 |
| 后置条件 | 在一个制定薪水策略的回合中，持久化更新本次制定的薪水策略。 |
| SalaryStratey.endSalaryStrategy | 语法 | public void endSalaryStrategy(); |
| 前置条件 | 在制定薪水策略的过程中，结束按钮被点击 |
| 后置条件 | 在一个制定薪水策略的回合中，持久化更新本次制定的薪水策略，结束本次制定心塞策略回合 |
| SalaryStrategy.saveChange | 语法 | public void saveChange(SalaryVO vo); |
| 前置条件 | 在制定薪水策略的过程中，保存按钮被点击 |
| 后置条件 | 在一个制定薪水策略的回合中，持久化更新本次制定的薪水策略。 |
| SalaryStrategy.select | 语法 | public SalaryVO select(String id); |
| 前置条件 | 在制定薪水策略的过程中，薪水策略被选择 |
| 后置条件 | 在一个制定薪水策略的回合中，返回被选择的薪水策略详细详细信息 |
| SalaryStrategy.revise | 语法 | public void revise(); |
| 前置条件 | 在制定薪水策略的过程中，薪水策略已被选择 |
| 后置条件 | 在一个制定薪水策略的回合中，返回可修改的被选择的薪水策略详细详细信息 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| SalaryStrategyDataService.findAll() | 返回所有持久化对象 |
| SalaryStrategyDataService.insert(SalaryPO po) | 插入单一持久化对象 |
| SalaryStrategyDataService.delect(SalaryPO po) | 删除单一持久化对象 |
| SalaryStrategyDataService.delect(String id) | 根据id删除单一持久化对象 |
| SalaryStrategyDataService.delect(ArrayList<SalaryPO >salarys) | 删除单一持久化对象 |
| SalaryStrategyDataService.update(SalaryPO po) | 更新单一持久化对象 |
| SalaryStrategyDataService.update(ArrayList<SalaryPO> salarys) | 删除多个持久化对象 |

Constant的接口规范

|  |  |  |
| --- | --- | --- |
| Constant.getConstant | 语法 | public ConstantVO getConstant(); |
| 前置条件 | 开启新的一轮制定常量 |
| 后置条件 | 返回先前的所制定的常量表 |
| Constant.newConstant | 语法 | public void newConstant(); |
| 前置条件 | 开启新的一轮制订常量，制定新的常量表的按钮被点击 |
| 后置条件 | 返回常量制定表 |
| Constant.save | 语法 | public void save(ConstantVO vo); |
| 前置条件 | 在制定常量的过程中，保存按钮被点击 |
| 后置条件 | 在一个制定常量的回合中，持久化更新本次制定的常量。 |
| Constant.getFare | 语法 | public double getFare(Position departure,Position destination,Traffic trafficType); |
| 前置条件 | 运输单据计算运费时被调用 |
| 后置条件 | 返回相应的运费 |
| Constant.endConstant | 语法 | public void endConstant(); |
| 前置条件 | 在制定常量的过程中，结束按钮被点击 |
| 后置条件 | 在一个制定常量的回合中，持久化更新本次制定的常量，结束本次制定常量回合 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| ConstantDataService.find() | 返回最顶端单一持久化对象 |
| ConstantDataService.insert(ConstantPO po) | 插入单一持久化对象 |
| ConstantDataService.delect(ConstantPO po) | 删除单一持久化对象 |
| ConstantDataService.delect(String id) | 根据id删除单一持久化对象 |
| ConstantDataService.update(ConstantPO po) | 更新单一持久化对象 |

pamannagementsl模块的接口规范

|  |  |  |
| --- | --- | --- |
| PManagement.select | 语法 | public StaffVO select(String id); |
| 前置条件 | 开启新的一轮人员管理 |
| 后置条件 | 返回该人员的详细信息 |
| PManagement.delete | 语法 | public ResultMessage delete(String id); |
| 前置条件 | 一开启新一轮的人员信息管理，已选择好需要删除的人员信息单，删除按钮被点击 |
| 后置条件 | 在一个人员管理回合中，删除该人员信息单，持久化更新本次删除的信息单，结束人员信息删除 |
| PManagement.revise | 语法 | public StaffVO revise(String id); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启人员信息单对象的修改方法 |
| PManagement.saveChange | 语法 | public ResultMessage saveChange(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次人员管理的回合中，持久化更新人员信息单对象的数据，结束本次人员信息修改 |
| PManagement.save | 语法 | public ResultMessage save(StaffVO vo); |
| 前置条件 | 在人员管理的过程中，人员信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次人员管理的回合中，持久化更新人员信息单对象的数据，结束本次人员信息添加 |
| PManagement.add | 语法 | public ResultMessage add(String name,Work work,String workNumber,String workPlaceNumber,String birthDate,String idNumber,String phoneNumber,String address,Sex sex,double page); |
| 前置条件 | 在人员管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的人员信息单 |
| PManagement.getAllStaff | 语法 | public ArrayList<StaffVO> getAllStaff(); |
| 前置条件 | 开启新的一轮人员管理 |
| 后置条件 | 返回所有的人员信息单 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getPManagementDatabase | 得到PManagement数据库服务的引用 |
| PManagementDataService.find(String id) | 返回最顶端单一持久化对象 |
| PManagementDataService.insert(StaffPO po) | 插入单一持久化对象 |
| PManagementDataService.delete(StaffPO po) | 删除单一持久化对象 |
| PManagementDataService.delete(String id) | 删除单一持久化对象 |
| PManagementDataService.update(StaffPO po) | 更新单一持久化对象 |
| PManagementDataService.findAll() | 返回所有持久化对象 |
| PManagementDataService.findMember() | 返回符合条件的持久化对象 |

|  |  |  |
| --- | --- | --- |
| AManagement.select | 语法 | public AgencyPO select(String id); |
| 前置条件 | 开启新的一轮机构管理 |
| 后置条件 | 返回该机构的详细信息 |
| AManagement.delete | 语法 | public ResultMessage delete(String id); |
| 前置条件 | 一开启新一轮的机构信息管理，已选择好需要删除的机构信息单，删除按钮被点击 |
| 后置条件 | 在一个机构管理回合中，删除该机构信息单，持久化更新本次删除的信息单，结束机构信息删除 |
| AManagement.revise | 语法 | public AgencyVO revise(String id); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启机构信息单对象的修改方法 |
| AManagement.saveChange | 语法 | public ResultMessage saveChange(AgencyVO vo); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次机构管理的回合中，持久化更新机构信息单对象的数据，结束本次机构信息修改 |
| AManagement.save | 语法 | public void save(); |
| 前置条件 | 在机构管理的过程中，机构信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次机构管理的回合中，持久化更新机构信息单对象的数据，结束本次机构信息添加 |
| AManagement.add | 语法 | public ResultMessage add(String name,String idNumber,ArrayList<String> staff,String phonenumber,String address,String leader); |
| 前置条件 | 在机构管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的机构信息单 |
| AManagement.getAllAgency | 语法 | public ArrayList<AgencyVO> getAllAgency(); |
| 前置条件 | 存在机构持久化对象 |
| 后置条件 | 返回所有机构信息 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getAManagementDatabase | 得到AManagement数据库服务的引用 |
| AManagementDataService.find(String id) | 返回最顶端单一持久化对象 |
| AManagementDataService.insert(StaffPO po) | 插入单一持久化对象 |
| AManagementDataService.delete(StaffPO po) | 删除单一持久化对象 |
| AManagementDataService.delete(String id) | 删除单一持久化对象 |
| AManagementDataService.update(StaffPO po) | 更新单一持久化对象 |
| AManagementDataService.findAll() | 返回所有持久化对象 |

DManagement的接口规范

|  |  |  |
| --- | --- | --- |
| DManagement.select | 语法 | public DriverVO select(String id); |
| 前置条件 | 开启新的一轮司机管理 |
| 后置条件 | 返回该司机的详细信息 |
| DManagement.delete | 语法 | public ResultMessage delete(String id); |
| 前置条件 | 已开启新一轮的司机信息管理，已选择好需要删除的司机信息单，删除按钮被点击 |
| 后置条件 | 在一个司机管理回合中，删除该司机信息单，持久化更新本次删除的信息单，结束司机信息删除 |
| DManagement.revise | 语法 | public DriverVO revise(String id); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启司机信息单对象的修改方法 |
| DManagement.saveChange | 语法 | public ResultMessage saveChange(DriverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次司机管理的回合中，持久化更新司机信息单对象的数据，结束本次机构信息修改 |
| DManagement.save | 语法 | public ResultMessage save(DriverVO vo); |
| 前置条件 | 在司机管理的过程中，司机信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次司机管理的回合中，持久化更新司机信息单对象的数据，结束本次机构信息添加 |
| DManagement.add | 语法 | public ResultMessage add(String name,Work work,String workNumber,String workPlaceNumber,String birthDate,String idNumber,String phoneNumber,String address,Sex sex,int driverYear,int page); |
| 前置条件 | 在司机管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的机构信息单 |
| DManagement.getAllDriver | 语法 | public ArrayList<DriverVO> getAllDriver(); |
| 前置条件 | 开启新的一轮司机管理 |
| 后置条件 | 返回所有的司机信息单 |
| DManagement.endDManagement | 语法 | public void endDManagement(); |
| 前置条件 | 司机管理已开始 |
| 后置条件 | 结束本次司机管理，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 | 需要的服务 |
| DManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| DManagementDataService.insert(DriverPO po) | 插入单一持久化对象 |
| DManagementDataService.delete(DriverPO po) | 根据对象删除单一持久化对象 |
| DManagementDataService.delete(String id) | 根据id删除单一持久化对象 |
| DManagementDataService.update(DriverPO po) | 更新单一持久化对象 |
| DManagementDataService.findAll(); | 返回所有的持久化对象 |

CManagement的接口规范

|  |  |  |
| --- | --- | --- |
| CManagement.select | 语法 | public CarVO select(String id); |
| 前置条件 | 开启新的一轮车辆管理 |
| 后置条件 | 返回该车辆的详细信息 |
| CManagement.delete | 语法 | public ResultMessage delete(String id); |
| 前置条件 | 已开启新一轮的车辆信息管理，已选择好需要删除的车辆信息单，删除按钮被点击 |
| 后置条件 | 在一个车辆管理回合中，删除该车辆信息单，持久化更新本次删除的信息单，结束车辆信息删除 |
| CManagement.revise | 语法 | public CarVO revise(String id); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被选择，修改按钮被点击 |
| 后置条件 | 开启车辆信息单对象的修改方法 |
| CManagement.saveChange | 语法 | public ResultMessage saveChange(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被修改，保存按钮被点击 |
| 后置条件 | 在一次车辆管理的回合中，持久化更新车辆信息单对象的数据，结束本次车辆信息修改 |
| CManagement.save | 语法 | public ResultMessage save(CarVO vo); |
| 前置条件 | 在车辆管理的过程中，车辆信息单已经被添加，保存按钮被点击 |
| 后置条件 | 在一次车辆管理的回合中，持久化更新车辆信息单对象的数据，结束本次车辆信息添加 |
| CManagement.add | 语法 | public ResultMessage add(String idNumber, String workPlaceNumber, String licenseNumber, int workYear); |
| 前置条件 | 在车辆管理的过程中，添加按钮被点击 |
| 后置条件 | 返回新的车辆信息单 |
| CManagement.getAllCar | 语法 | public ArrayList<CarVO> getAllCar(); |
| 前置条件 | 开启新的一轮车辆管理 |
| 后置条件 | 返回所有的车辆信息单 |
| CManagement.endCManagement | 语法 | public void endCManagement(); |
| 前置条件 | 车辆管理已开始 |
| 后置条件 | 结束本次车辆管理，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 | 需要的服务 |
| CManagementDataService.find(String id) | 根据id返回的单一持久化对象 |
| CManagementDataService.insert(CarPO po) | 插入单一持久化对象 |
| CManagementDataService.delete(CarPO po) | 根据对象删除单一持久化对象 |
| CManagementDataService.delete(String id) | 根据id删除单一持久化对象 |
| CManagementDataService.update(CarPO po) | 更新单一持久化对象 |
| CManagementDataService.findAll(); | 返回所有的持久化对象 |

checksl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Check.orderNumberCheck | 语法 | public LogisticsVO  orderNumberCheck(String ordernumber); |
| 前置条件 | ordernumber符合输入规则 |
| 后置条件 | 查找是否存在响应的OrderMessage,根据输入的ordernumber返回所对应的物流信息 |
| Check.add | 语法 | public ResultMessage  add(LogisticsVO vo); |
| 前置条件 | vo符合输入规则 |
| 后置条件 | 将vo转换为po插入数据库，返回插入结果信息 |
| Check.update | 语法 | public ResultMessage  update(LogisticsVO vo); |
| 前置条件 | vo符合输入规则 |
| 后置条件 | 将vo转换为po更新数据库，返回更新结果信息 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getCheckDatabase | 得到Check数据库服务的引用 |
| CheckDataService.find(String ordernumber) | 根据ordernumber进行查找单一持久化对象 |
| CheckDataService.add(LogisticsPO po) | 插入单一持久化对象 |
| CheckDataService.update(LogisticsPO po) | 更新单一持久化对象 |

financesl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Balance.getBalanceMessage | 语法 | public ArrayList<ReceiptsVO> getBalanceMessage(String office,String starttime,String endtime); |
| 前置条件 | 选择的营业厅，输入的时间段符合输入规则 |
| 后置条件 | 返回该条件下的结算详细内容 |
| Balance.getBalanceMessage | 语法 | public ArrayList<ReceiptsVO> getBalanceMessage(String date,String office); |
| 前置条件 | 选择的营业厅，输入的时间符合输入规则 |
| 后置条件 | 返回该条件下的结算详细内容 |
| Balance.getTotalMoney | 语法 | Public double getTotalMoney(ArrayList<ReceiptsVO list> |
| 前置条件 | 输入付款单符合条件 |
| 后置条件 | 返回所选择付款单的总金额 |
| Cost.setPayment | 语法 | public PaymentVO setPayment(AccountVO account,String paymentType,StaffVO receiver); |
| 前置条件 | 输入的账户、款项类型、收款人符合输入规则; |
| 后置条件 | 返回一个填写了初步信息的Payment |
| Cost.computePayment | 语法 | public PaymentVO computePayment(PaymentVO payment); |
| 前置条件 | 已经通过setPayment设置了payment的初步信息；输入的款项类型为工资 |
| 后置条件 | 返回一个拥有应支付款项信息的Payment |
| Cost.payPayment | 语法 | public void payPayment(Payment payment); |
| 前置条件 | 已经通过computePayment计算了所需支付款项; |
| 后置条件 | 根据所选账户支付所需支付款项 |
| Cost.computePayment | 语法 | public PaymentVO computePayment(Payment payment,double money); |
| 前置条件 | 已经通过setPayment设置了payment的初步信息；输入的款项类型为奖金或租金 |
| 后置条件 | 返回一个拥有应支付款项信息的Payment |
| Cost.computeAuditNumber | 语法 | Public String computeAuditNumber(); |
| 前置条件 | 无 |
| 后置条件 | 返回当前付款单编号 |
| AccountManagement.addAccount | 语法 | public Account addAccount(String accountname,double accountmoney); |
| 前置条件 | 输入的账户名和账户余额符合输入规则 |
| 后置条件 | 返回一个新的Account对象 |
| AccountManagement.selectAccount | 语法 | public Account selectAccount(String accountname); |
| 前置条件 | 在账户管理的过程中，某个账户被选中 |
| 后置条件 | 返回一个所选中的账户的信息 |
| AccountManagement.deleteAccount | 语法 | public ResultMessage deleteAccount(Account account); |
| 前置条件 | 在选中账户后，进行了账户删除操作 |
| 后置条件 | 在数据库中删除相对应的账户信息 |
| AccountManagement.fixAccount | 语法 | public Account fixAccount(Account account,String accountname); |
| 前置条件 | 在选中账户后，进行了账户修改操作 |
| 后置条件 | 在数据库中更新相对应的账户信息 |
| AccountManagement.seekAccount | 语法 | public ArrayList<AccountPO> seekAccount(String accountname); |
| 前置条件 | 在账户管理页面，进行了账户查找信息 |
| 后置条件 | 返回一个符合查询信息的Account列表 |
| AccountManagement.findAccount | 语法 | public AccountVO findAccount(String accountname); |
| 前置条件 | 数据库已存在此账户信息 |
| 后置条件 | 返回符合查询信息的账户 |
| AccountInitialize.  initialize | 语法 | public CompanyAccount initialize(CompanyAccount oldaccount,String customer,String agency,String people,String stock,String bankaccount); |
| 前置条件 | 在账目初始化页面，进行了新建账目操作 |
| 后置条件 | 返回一个新建的账目 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getFinanceDatabase | 得到finance数据库服务的引用 |
| FinanceDataService.insertPaymentPO(PaymentPO paymennt) | 插入paymentPO |
| FinanceDataService.findsPaymentPO(String workplacenumber) | 根据营业厅获取paymentPO |
| FinanceDataService.findAudit() | 获取未被审判的单据 |
| FinanceDataService.updatePaymentPO(PaymentPO payment) | 更新paymentPO |
| FinanceDataService.findLastest() | 获取最新的paymentPO |
| FinanceDataService.findAccountPO(String id) | 根据id获取accountPO |
| FinanceDataService.seekAccount(String name) | 根据name获取accountPO列表 |
| FinanceDataService.insertAccountPO(AccountPO account) | 插入accountPO |
| FinanceDataService.deleteAccountPO(AccountPO account) | 删除accountPO |
| FinanceDataService.updateAccountPO(AccountPO account) | 更新accountPO |
| FinanceDataService.findCompanyAccountPO(String id) | 获取companyAccountPO |
| FinanceDataService.insertCompanyAccountPO(CompanyAccountPO companyaccount) | 插入companyAccountPO |
| FinanceDataService.findPWithdContion(String nameOfWriter, DocumentCondition dCondition) | 模糊查找paymentPO |

usersl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务 | | | |
| User.login | 语法 | Public ResultMessage login(String[] accountandcode) | |
| 前置条件 | password符合输入规则 | |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回登录验证结果 | |
| Log.logmessage | 语法 | public ArrayList<LogVO> logmessage(String office,String time); | |
| 前置条件 | 输入的营业厅、时间符合输入规则; | |
| 后置条件 | 返回符合选择条件的log列表 | |
| Log.addMessage | 语法 | public ResultMessage addMessage(String userId,String logmessage) | |
| 前置条件 | 输入的用户ID和日志信息符合输入规则； | |
| 后置条件 | 返回插入日志信息结果 | |
| User.select | 语法 | public UserVO select(String id); | |
| 前置条件 | 开启新的一轮用户管理 | |
| 后置条件 | 返回该人员的详细信息 | |
| User.delete | 语法 | public ResultMessage delete(String id); | |
| 前置条件 | 一开启新一轮的用户信息管理，已选择好需要删除的人员信息单，删除按钮被点击 | |
| 后置条件 | 在一个用户管理回合中，删除该用户信息单，持久化更新本次删除的信息单，结束用户信息删除 | |
| User.revise | 语法 | public void revise(String id); | |
| 前置条件 | 在用户管理的过程中，用户信息单已经被选择，修改按钮被点击 | |
| 后置条件 | 开启用户信息单对象的修改方法 | |
| User.saveChange | 语法 | public ResultMessage saveChange(UserVO vo); | |
| 前置条件 | 在用户管理的过程中，用户信息单已经被修改，保存按钮被点击 | |
| 后置条件 | 在一次用户管理的回合中，持久化更新用户信息单对象的数据，结束本次用户信息修改 | |
| User.save | 语法 | public ResultMessage save(UserVO vo); | |
| 前置条件 | 在用户管理的过程中，用户信息单已经被添加，保存按钮被点击 | |
| 后置条件 | 在一次用户管理的回合中，持久化更新用户信息单对象的数据，结束本次用户信息添加 | |
| User.add | 语法 | public UserVO add(String name,String accountnumber,String code,String privileges); | |
| 前置条件 | 在用户管理的过程中，添加按钮被点击 | |
| 后置条件 | 返回新的用户信息单 | |
| User.getAllUser | 语法 | public ArrayList<UserVO> getAllUser(); | |
| 前置条件 | 开启新的一轮用户管理 | |
| 后置条件 | 返回所有的用户信息单 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getUserDatabase | 得到user数据库服务的引用 |
| UserDataService.findLogPO(String office,String time) | 根据office和time获取logPO |
| UserDataService.insertLogPO(LogPO po) | 插入单一LogPO |
| UserDataService.findLogPO(String office,String time) | 查找多个LogPO |
| UserDataService.insertUserPO(UserPO PO) | 插入单一持久化对象 |
| UserDataService.delectUserPO(UserPO PO) | 删除单一持久化对象 |
| UserDataService.updateUserPO(UserPO PO) | 更新单一持久化对象 |
| UserDataService.findUserPO(String id) | 根据id进行查找单一持久化对象 |
| UserDataService.findsUserPO(String[]ids) | 根据ids查找多个持久化对象 |
| UserDataServicr.getAllUserPO() | 获取全部UserPO |

Receive模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Receive.addExpress | 语法 | public OrderVO addExpress(String name,String time, String orderNumber) |
| 前置条件 | 启动一个派件回合 |
| 后置条件 | 在一个派件回合内，增加快件号，收件人，收件时间（系统自动记录） |
| Receive.endReceive | 语法 | public void endReceive() |
| 前置条件 | 派件信息已提交 |
| 后置条件 | 结束此次派件回合，持久化更新涉及的领域对象的数据 |
| Receive.saveExpress | 语法 | public ResultMessage saveExpress(OrderVO ordervo) |
| 前置条件 | 已添加收件信息 |
| 后置条件 | 系统保存收件信息 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ReceiveDataService.findO(String orderNumber) | 根据orderNumber进行查找单一持久化对象 |
| ReceiveDataService.findsO(String field, String orderNumber) | 根据字段名和值进行查找多个持久化对象 |
| ReceiveDataService.insertO(OrderPO po) | 插入单一持久化对象 |
| ReceiveDataService.deleteO(OrderPO po) | 删除单一持久化对象 |
| ReceiveDataService.updateO(OrderPO po) | 更新单一持久化对象 |
| DatabaseFactory.getReceiveDatabase | 得到Receive数据库的服务的引用 |

Deliver模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Deliver.newDeliver | 语法 | public DeliverVO newDeliver(String id，String nameOfWriter,String time,ArrayList<String> member,ArrayList<String> order,DeliverVO delivervo) |
| 前置条件 | 启动一个派件回合，输入信息符合规则 |
| 后置条件 | 返回新的派件单信息 |
| Deliver.findDWithCondition | 语法 | public ArrayList<DeliverVO> findDWithCondition(String nameOfWriter,DocumentCondition dCondition) |
| 前置条件 | 输入信息符合规则 |
| 后置条件 | 返回指定条件的派件单信息 |
| Deliver.printDeliver | 语法 | public void printDeliver(String deliverNumber) |
| 前置条件 | 已生成新的派件单 |
| 后置条件 | 打印派件单 |
| Deliver.endReceive | 语法 | public void endReceive() |
| 前置条件 | 派件信息已提交 |
| 后置条件 | 结束此次派件回合，持久化更新涉及的领域对象的数据 |
| Deliver.getDeliver | 语法 | public DeliverVO getDeliver(String deliverNumber) |
| 前置条件 | 启动一个派件回合，deliverNumber符合输入规则 |
| 后置条件 | 返回派件单信息 |
| Deliver.saveDeliver | 语法 | public ResultMessage saveDeliver(DeliverVO delivervo) |
| 前置条件 | 已生成派件单 |
| 后置条件 | 系统保存派件单 |
| Deliver.updateDeliver | 语法 | public ResultMessage updaeDeliver(DeliverVO delivervo) |
| 前置条件 | 已生成派件单 |
| 后置条件 | 系统更新派件单 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ReceiveDataService.findD(String deliverNumber) | 根据deliverNumber进行查找单一持久化对象 |
| ReceiveDataService.findsD(String field, String deliverNumber) | 根据字段名和值进行查找多个持久化对象 |
| ReceiveDataService.findDWithCondition(String nameOfWriter,DocumentCondition dCondition) | 根据字段名和值进行查找多个持久化对象 |
| ReceiveDataService.insertD(DeliverPO po) | 插入单一持久化对象 |
| ReceiveDataService.deleteD(DeliverPO po) | 删除单一持久化对象 |
| ReceiveDataService.updateD(DeliverPO po) | 更新单一持久化对象 |
| DatabaseFactory.getReceiveDatabase | 得到Receive数据库的服务的引用 |

Report模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Report.caculate | 语法 | public void caculate（ReportVO reportvo） | |
| 前置条件 | 已经生成经营情况表 | |
| 后置条件 | 根据经营情况表，生成成本收益表 | |
| Report.addMessage | 语法 | Public ReportVO addMessage(String id,String beginTime,String endTime) | |
| 前置条件 | 已请求查询统计报表 | |
| 后置条件 | 在一个统计报表回合中，增加起始时间、终止时间和被查询营业厅，生成经营情况表 | |
| Report.exportReport | 语法 | public void exportReport(ReportVO reportvo) | |
| 前置条件 | 已生成统计报表 | |
| 后置条件 | 导出统计报表 | |
| Report.printReport | 语法 | public void printReport(ReportVO reportvo) | |
| 前置条件 | 已生成统计报表 | |
| 后置条件 | 打印统计报表 | |
| Report.endReport | 语法 | public void endReport() | |
| 前置条件 | 已生产统计报表 | |
| 后置条件 | 结束此次统计报表回合 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| ReportDataService.find(String id) | | | 根据id进行查找单一持久化对象 |
| ReportDataService.finds(String field, String id) | | | 根据字段名和值进行查找多个持久化对象 |
| ReportDataService.insert(ReportPO po) | | | 插入单一持久化对象 |
| ReportDataService.delete(ReportPO po) | | | 删除单一持久化对象 |
| ReportDataService.update(ReportPO po) | | | 更新单一持久化对象 |
| DatabaseFactory.getReportDatabase | | | 得到Report数据库的服务的引用 |

Transport模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Transportsl.newTransport | 语法 | public TransportVO newTransport(TransportType sign, String id,  String departure, String destination, String transportID,  String time, String trafficID, Traffic trafficType, double fare,  ArrayList<String> member, ArrayList<String> order,  ArrayList<Condition> condition,  DocumentCondition documentCondition, String nameOfWriter) |
| 前置条件 | 启动一个运输回合 |
| 后置条件 | 在一个运输回合中，生成运输单并增加运输单信息 |
| public ArrayList<TransportVO> getTransportList | 语法 | public ArrayList<TransportVO> getTransportList(String nameOfWriter,DocumentCondition dCondition) |
| 前置条件 | 启动一个运输回合 |
| 后置条件 | 在一个运输回合中，获得指定营业厅营业员的所有未审批运输单 |
| Transportsl.addFare | 语法 | public TransportVO addFare(TransportVO transportvo) |
| 前置条件 | 已生成运输单 |
| 后置条件 | 生成运费 |
| Transportsl.getTransport | 语法 | public TransportVO getTransport(String transportid) |
| 前置条件 | 启动一个运输回合，运输单编号符合输入规则 |
| 后置条件 | 在一个运输回合中，返回运输单信息 |
| Transportsl.endTransport | 语法 | public void endTransport() |
| 前置条件 | 无 |
| 后置条件 | 结束此次运输回合，持久化更新涉及的领域对象的数据 |
| Transportsl.printTransport | 语法 | public void printTransport(String id) |
| 前置条件 | 启动一个运输回合 |
| 后置条件 | 在一个运输回合中，打印运输单 |
| Transportsl.saveTransport | 语法 | public ResultMessage saveTransport(TransportVO transportvo) |
| 前置条件 | 启动一个运输回合，已生成运输单 |
| 后置条件 | 在一个运输回合中，保存运输单信息 |
| Transportsl.updateTransport | 语法 | public ResultMessage updateTransport(TransportVO transportvo) |
| 前置条件 | 启动一个运输回合 |
| 后置条件 | 在一个运输回合中，更新运输单信息 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| TransportDataService.find(String id) | 根据id进行查找单一持久化对象 |
| TransportDataService.finds(String field, String id) | 根据字段名和值进行查找多个持久化对象 |
| public ArrayList<TransportPO> findWithdCondition(String nameOfWriter, DocumentCondition dCondition) | 根据字段名和值进行查找多个持久化对象 |
| TransportDataService.insert(TransportPO po) | 插入单一持久化对象 |
| TransportDataService.delete(TransportPO po) | 删除单一持久化对象 |
| TransportDataService.update(TransportPO po) | 更新单一持久化对象 |
| DatabaseFactory.getTransportDatabase | 得到Transport数据库的服务的引用 |

stocksl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| StockInfo.show | 语法 | public stockinfoVO show (String[] time); |
| 前置条件 | 已输入时间段，确认按钮被点击 |
| 后置条件 | 返回该时间段内的库存信息 |
| StockInfo.initialize | 语法 | public ResultMessage initialize(StockInitializeVO vo); |
| 前置条件 | 已输入初始信息，库存信息初始化按钮被点击 |
| 后置条件 | 结束库存信息初始化，持久化更新库存的数据 |
| StockInfo.check | 语法 | public ResultMessage check(); |
| 前置条件 | 库存盘点按钮被点击 |
| 后置条件 | 结束库存盘点 |
| StockAdjust.getArea | 语法 | public AreaVO[] getAreas(); |
| 前置条件 | 库区范围调整开始按钮被点击 |
| 后置条件 | 显示库区信息 |
| StockAdjust.selectArea | 语法 | public AreaVO selectArea(String name); |
| 前置条件 | 选中库区 |
| 后置条件 | 显示库区详细信息 |
| StockAdjust.range | 语法 | public WarningMessage range(String adjustrange); |
| 前置条件 | 库区范围调整过程中，已输入调整范围，调整按钮被点击 |
| 后置条件 | 结束库区调整，检查库存是否在安全范围内，持久化更新库存的数据 |
|  |  |  |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getStockDatabase | 得到Stock数据库服务的引用 |
| StockDataService.findposition(int row,int shelf,int seat) | 根据位置进行查找单一持久化对象 |
| StockDataService.findpositions(ArrayList<int[][]> positions ) | 根据位置查找多个持久化对象 |
| StockDataService.insert(StockPO PO) | 插入单一持久化对象 |
| StockDataService.delete(StockPO PO) | 删除单一持久化对象 |
| StockDataService.update(StockPO PO) | 更新单一持久化对象 |
| StockDataService.getAll(ArrayList<StockPO>) | 获取全部库存信息 |
| StockDataService.clean(ResultMessage) | 清空库存信息 |
| StockDataService.findWithdCondition（ArrayList<IoputPO>） | 根据单据创建者和单据状态寻找多个持久化对象 |
| StockDataService.findAudit(ArrayList<IoputPO>) | 寻找未被审批的单据持久化对象 |

ioputsl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| InputStock.showInfo | 语法 | public InMessageVO showInfo (String id); |
| 前置条件 | 在入库管理过程中，已输入条形码号，确定按钮被点击 |
| 后置条件 | 显示相应的快递信息 |
| InputStock.position | 语法 | public WarningMessage position (String position,InMessageVO); |
| 前置条件 | 在入库处理过程中，已输入快递位置，确认按钮被点击 |
| 后置条件 | 结束入库管理，已检查库区库存数量是否在安全范围内，持久化更新出入库的数据 |
| OutputStock.getOutputList | 语法 | ArrayList<String[]> getOutputList(); |
| 前置条件 | 出库管理界面被打开 |
| 后置条件 | 返回待出库快递信息列表 |
| OutputStock.showInfo | 语法 | public OutMessageVO showOutputInfo(String id,String userId); |
| 前置条件 | 在出库管理过程中，已输入条形码号，确认按钮被点击 |
| 后置条件 | 显示相应的快递信息 |
| OutputStock.report | 语法 | public ResultMessage report(String id,userId); |
| 前置条件 | 在出库管理中，已输入条形码号，报错按钮被点击 |
| 后置条件 | 系统记录快递信息，持久化更新出入库的数据 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getIoputDatabase | 得到Ioput数据库服务的引用 |
| IoputDataService.find(String id) | 根据id进行查找单一持久化对象 |
| IoputDataService.finds(String[]ids) | 根据ids查找多个持久化对象 |
| IoputDataService.findTimes(String[] time) | 根据时间段寻找多个持久化对象 |
| IoputDataService.insert(IoputPO PO) | 插入单一持久化对象 |
| IoputDataService.delete(IoputPO PO) | 删除单一持久化对象 |
| IoputDataService.update(IoputPO PO) | 更新单一持久化对象 |

Sendsl模块的接口规范

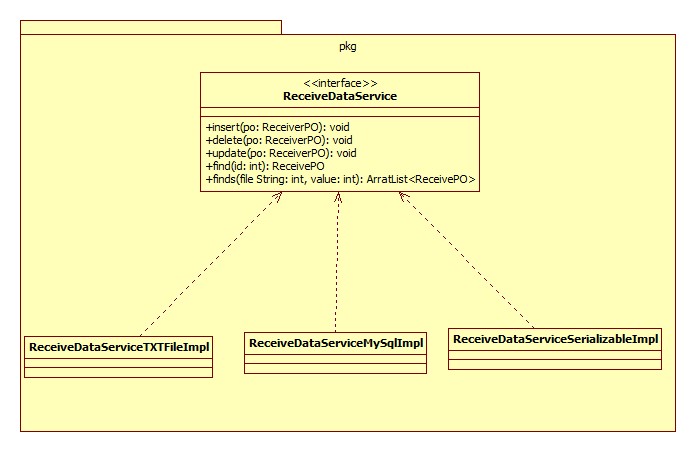
|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| OrderSend.calculate | 语法 | public OrderVO calculate(OrderVO info); |
| 前置条件 | 已输入快递件信息，确认按钮被点击 |
| 后置条件 | 系统显示快递信息，计算的价格和预期到达的日期 |
| OrderSend.setBill | 语法 | public OrderVO setBill(OrderVO order); |
| 前置条件 | orderVO已存入bill信息 |
| 后置条件 | 存入bill信息 |
| OrderSend.getChange | 语法 | public BillVO getChange(double cash,BillVO bill); |
| 前置条件 | 已计算价格 |
| 后置条件 | 根据支付的金额，计算找零的金额 |
| OrderSend.orderend | 语法 | public void orderend(BillVO bill,OrderVO order); |
| 前置条件 | 已完成支付 |
| 后置条件 | 结束订单输入，持久化更新订单的数据 |
| OrderSend.computeOrdernumber | 语法 | public String computeOrdernumber(); |
| 前置条件 | 订单信息输入完成 |
| 后置条件 | 生成订单号码 |
| OrderSend.computedue | 语法 | public String computedue(OrderVO order); |
| 前置条件 | 订单信息输入完成 |
| 后置条件 | 生成预期到达日期 |
| GatheringSend. receiptsNew | 语法 | public ArrayList<ReceiptsVO> newReceipts(String Date,String workplace) |
| 前置条件 | 生成收款单按钮被点击 |
| 后置条件 | 系统计算并显示收款单信息，记录收款单数据 |
| GatheringSend.findReceipt | 语法 | public ReceiptsVO findReceipt(String Date,String courier) |
| 前置条件 | 日期、快递员编号输入符合输入规则 |
| 后置条件 | 返回符合条件的收款单信息 |
| GatheringSend.findReceipts | 语法 | public ArrayList<ReceiptsVO> findReceipts(String Date,String office) |
| 前置条件 | 日期、营业厅编号输入符合输入规则 |
| 后置条件 | 返回符合条件的收款单信息 |
| GatheringSend.updateReceipts | 语法 | public ResultMessage updateReceipts(ReceiptsVO receiptsvo); |
| 前置条件 | 输入收款单在数据库已存在 |
| 后置条件 | 收款单更新 |
| GatheringSend.deleteReceipts | 语法 | public ResultMessage deleteReceipts(String Date,String courier); |
| 前置条件 | 输入收款单在数据库已存在 |
| 后置条件 | 删除符合条件的收款单 |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 需要的服务 |
| DatabaseFactory.getSendDatabase | 得到Send数据库服务的引用 |
| SendDataService.insert(OrderPO PO) | 插入单一OrderPO持久化对象 |
| SendDataService.updateOrderPO(OrderPO po) | 更新单一OrderPO持久化对象 |
| SendDataService.findForGathering | 查询订单列表 |
| SendDataService.findLatest() | 查询最新的OrderPO持久化对象 |
| SendDataService.findReceived() | 查询最近三个OrderPO持久化对象 |
| SendDataService.findWithdCondition(String nameOfCourier,  DocumentCondition dCondition) | 查询未被审判的OrderPO持久化对象 |
| SendDataService.insertReceiptsPO(ReceiptsPO receipts) | 插入单一ReceiptsPO持久化对象 |
| SendDataService.updateReceiptsPO(ReceiptsPO receipts) | 更新单一ReceiptsPO持久化对象 |
| SendDataService.deleteReceiptsPO(String date, String courier) | 删除单一ReceiptsPO持久化对象 |
| SendDataService.findReceipt(String date, String courier) | 查询单一ReceiptsPO持久化对象 |
| SendDataService.findReceipts(String date, String office) | 查询复数ReceiptsPO持久化对象 |
| SendDataService.findAudit() | 查询未被审判的OrderPO持久化对象 |

5.4数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的增、删、改、查。下面以Receive业务逻辑作为例子说明。Receive业务逻辑需要的服务由ReceiveDataService接口提供。采用数据库对持久化数据进行保存。数据层模块的描述具体如图所示。



5.4.1数据层模块的职责

数据层模块的职责如表11所示。

表11 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ReceiveDataService | 持久化数据库的接口，提供集体载入，集体保存，增删改查服务 |
| ReceiveDataServiceMySqlImpl | 基于MySql数据库的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ReceiveDataServiceTXTFileImpl | 基于Txt文件的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ReceiveDataServiceMySqlImpl | 基于序列化文件的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |

5.4.2数据层模块的接口规范

数据层模块的接口规范如表12所示。

表12 数据库模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceiveDataService.findO | 语法 | public OrderPO findO(String id) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的OrderPO结果 |
| ReceiveDataService.findD | 语法 | public DeliverPO findD(String id) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的DeliverPO结果 |
| ReceiveDataService.findsO | 语法 | public ArrayList<OrderPO> findsO(String field, String id) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 按字段名和ID进行查找返回相应的多个OrderPO结果 |
| ReceiveDataService.findsD | 语法 | public ArrayList<DeliverPO> findsD(String field, String id) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 按字段名和ID进行查找返回相应的多个DeliverPO结果 |
| ReceiveDataService.insertO | 语法 | public void insertO(OrderPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| ReceiveDataService.insertD | 语法 | public void insertO(DeliverPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| ReceiveDataService.deleteO | 语法 | public void deleteO(OrderPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中删除一个po |
| ReceiveDataService.deleteD | 语法 | public void deleteD(DeliverPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中删除一个po |
| ReceiveDataService.UpdateO | 语法 | public void updateO(OrderPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新一个po |
| ReceiveDataService.UpdateD | 语法 | public void updateD(DeliverPO po) throws Exception |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新一个po |
| ReceiveDataService.init() | 语法 | public void init() throws Exception |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| ReceiveDataService.finish() | 语法 | public void finish() throws Exception |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

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，在此只做简单的介绍。

LogisticsPO类包含快递件的物流信息，包括订单号和历史物流信息属性

AccountPO类包含账户的名称、余额属性

PaymentPO类包含付款单的收款人、款项类型和款项数目属性

OrderPO类包含订单的寄件人、收件人、账单、货物信息、寄件时间、预期到达时间、订单号、快递员名字、收件时间属性

BilPO类包含账单的所收金额、应收金额和找零金额属性

GoodsPO类包含货物的数目、重量、体积、尺寸、名称和快递类型属性

UserPO类包含用户的用户名、密码、账户、日志列表和权限属性

LogPO类包含日志的时间、营业厅、用户ID和日志信息属性

持久化用户对象UserPO的定义如下所示

**public** **class** UserPO {

String name;

String accountnumber;

String code;

String privileges;

ArrayList<LogPO> loglist=**new** ArrayList<LogPO>();

**public** UserPO(String name,String accountnumber,String code,String privileges){

**this**.name=name;

**this**.accountnumber=accountnumber;

**this**.code=code;

**this**.privileges=privileges;

}

**public** **void** addMessage(LogPO newlogpo){

loglist.add(newlogpo);

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAccountnumber() {

**return** accountnumber;

}

**public** **void** setAccountnumber(String accountnumber) {

**this**.accountnumber = accountnumber;

}

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getPrivileges() {

**return** privileges;

}

**public** **void** setPrivileges(String privileges) {

**this**.privileges = privileges;

}

}

6.2数据库表

数据库中包含Logistics表、Account表、Payment表、Bill表、Goods表、Order表、Log表、User表、Deliver表、Report表、Receipt表、IOput表、Stock表。