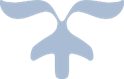


**快递物流系统**

**软件体系结构描述文档**



**V1.0正式版**

**2015/10/26**

**噔海马小王子Team**

**文档修改历史：**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 修改原因 | 版本号 |
| 全体小组成员 | 2015/10/26 | 软件体系结构文档初次建立 | V1.0 正式版 |

**小组成员：**

|  |  |
| --- | --- |
| **姓名** | **学号** |
| 曹姝玥 | 141250005 |
| 雷婷 | 141250061 |
| 李莹 | 141250065 |
| 徐海溦 | 141250158 |

# 目 录

[**1 引言 4**](#_Toc30793)

[1.1编制目的 4](#_Toc3626)

[1.2词汇表 4](#_Toc2621)

[1.3参考资料 4](#_Toc23570)

[**2 产品概述 4**](#_Toc22234)

[**3 逻辑视角 4**](#_Toc3677)

[**4 组合视角 5**](#_Toc17965)

[4.1开发包图 5](#_Toc16864)

[4.2运行时进程 8](#_Toc27343)

[4.3物理部署 8](#_Toc19354)

[**5 接口视角 9**](#_Toc18273)

[5.1模块的职责 9](#_Toc9920)

[5.2用户界面层的分解 15](#_Toc18683)

[5.2.1用户界面层模块的职责 18](#_Toc14329)

[5.2.2用户界面层模块的接口规范 18](#_Toc26084)

[5.2.3用户界面层模块设计原理 18](#_Toc14242)

[5.3业务逻辑层的分解 19](#_Toc2903)

[5.3.1业务逻辑层模块的职责 19](#_Toc25369)

[5.3.2业务逻辑层模块的接口规范 19](#_Toc6688)

[5.4数据层的分解 27](#_Toc21845)

[5.4.1数据层模块的职责 27](#_Toc29002)

[5.4.2数据层模块的接口规范 28](#_Toc10588)

[**6 信息视角 33**](#_Toc16197)

[6.1数据持久化对象 33](#_Toc23896)

# 1 引言

## 1.1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。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ES | 快递物流系统 | 无 |
| bl | 业务逻辑 | 无 |

## 1.3参考资料

1. IEEE标准；
2. 快递物流系统用例文档V3.3；
3. 快递物流系统软件需求规格说明文档V1.6。

# 2 产品概述

参考快递物流系统用例文档和快递物流系统软件需求规格说明中对产品的概括描述。

# 3 逻辑视角

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

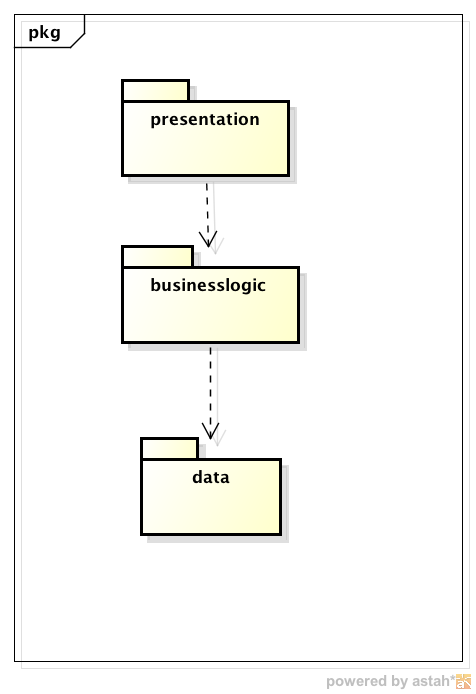


图1 参照体系结构风格的包图表达逻辑视角

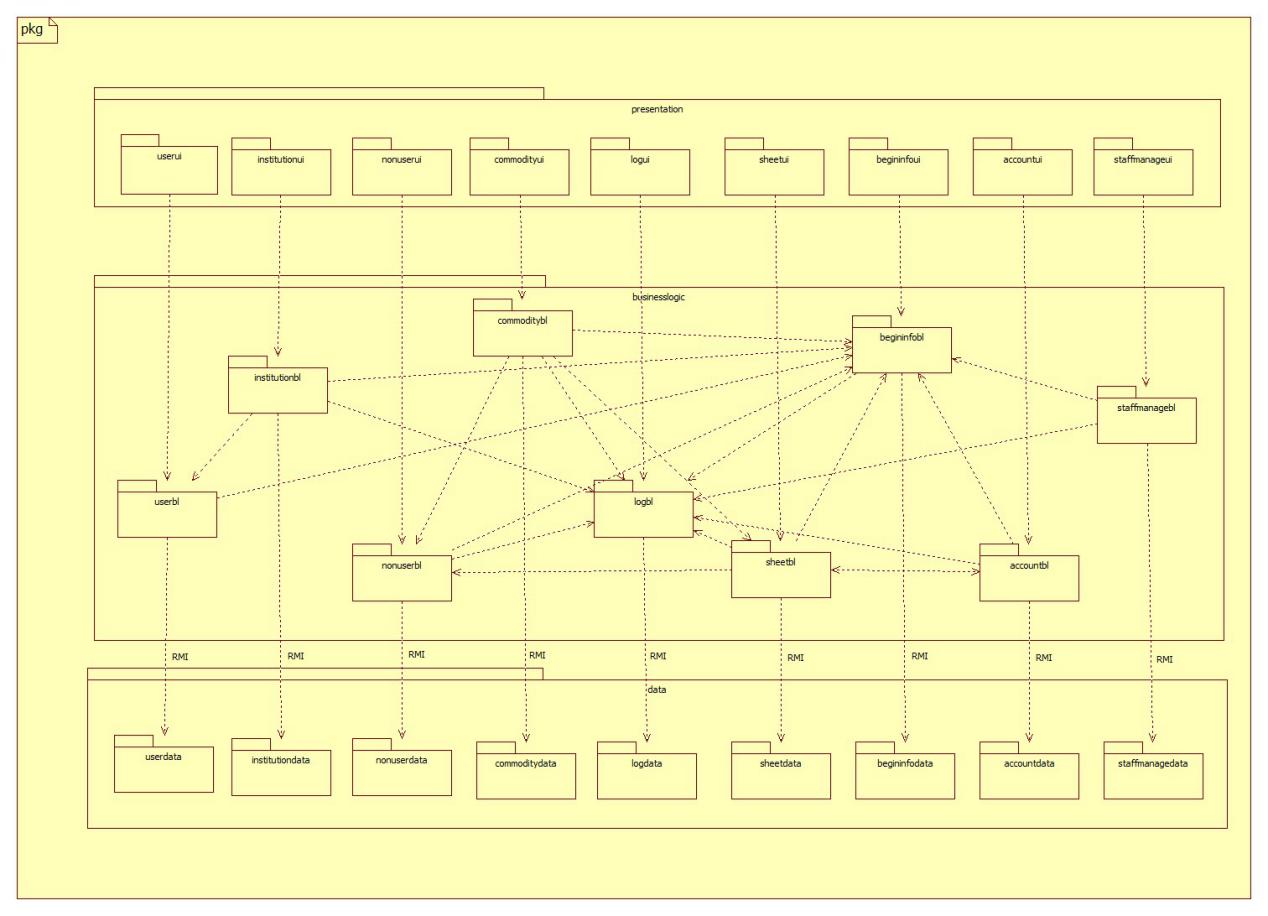


图2 软件体系结构逻辑设计方案

# 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, institutionui, nonuserui, commodityui, logui, sheetui, begininfoui, accountui, staffmanageui, vo |
| userui | userblservice, 界面类库包, vo |
| userblservice |  |
| userbl | begininfobl, userdataservice, userblservice,po |
| userdataservice | Java RMI, po |
| userdata | userdataservice , po |
| nonuserui | nonuserblservice, 界面类库包, vo |
| nonuserblservice |  |
| nonuserbl | nonuserblservice, nonuserdataservice, po, logbl |
| nonuserdataservice | Java RMI, po |
| nonuserdata | nonuserdataservice, po |
| institutionui | institutionblservice, 界面类库包, vo |
| institutionblservice |  |
| institutionbl | institutionblservice, institutiondataservice, po, begininfobl,logbl |
| institutiondataservice | Java RMI, po |
| institutiondata | institutiondataservice, po |
| commodityui | commodityblservice, 界面类库包, vo |
| commodityblservice |  |
| commoditybl | commodityblservice, commoditydataservice, po,nonuserbl,begininfobl,  logbl,sheetbl |
| commoditydataservice | Java RMI, po |
| commoditydata | commoditydataservice, po |
| logui | logblservice, 界面类库包, vo |
| logblservice |  |
| logbl | logblservice, logdataservice, po |
| logdataservice | Java RMI, po |
| logdata | logdataservice, po |
| sheetui | sheetblservice, 界面类库包, vo |
| sheetblservice |  |
| sheetbl | sheetblservice, sheetdataservice, po,nonuserbl,acountbl,begininfobl,  logbl |
| sheetdataservice | Java RMI, po |
| sheetdata | sheetdataservice, po |
| begininfoui | begininfoblservice, 界面类库包, vo |
| begininfoblservice |  |
| begininfobl | begininfoblservice, begininfodataservice, po |
| begininfodataservice | Java RMI, po |
| begininfodata | begininfodataservice, po |
| accountui | accountblservice, 界面类库包, vo |
| accountblservice |  |
| accountbl | accountblservice, accountdataservice, po,begininfobl,logbl,sheetbl |
| accountdataservice | Java RMI, po |
| accountdata | accountdataservice, po |
| staffmanageui | staffmanageblservice, 界面类库包, vo |
| staffmanageblservice |  |
| staffmanagebl | staffmanageblservice, staffmanagedataservice, po, begininfobl,logbl |
| staffmanagedataservice | Java RMI, po |
| staffmanagedata | staffmanagedataservice, po |
| vo |  |
| po |  |
| 界面类库包 |  |
| Java RMI |  |

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

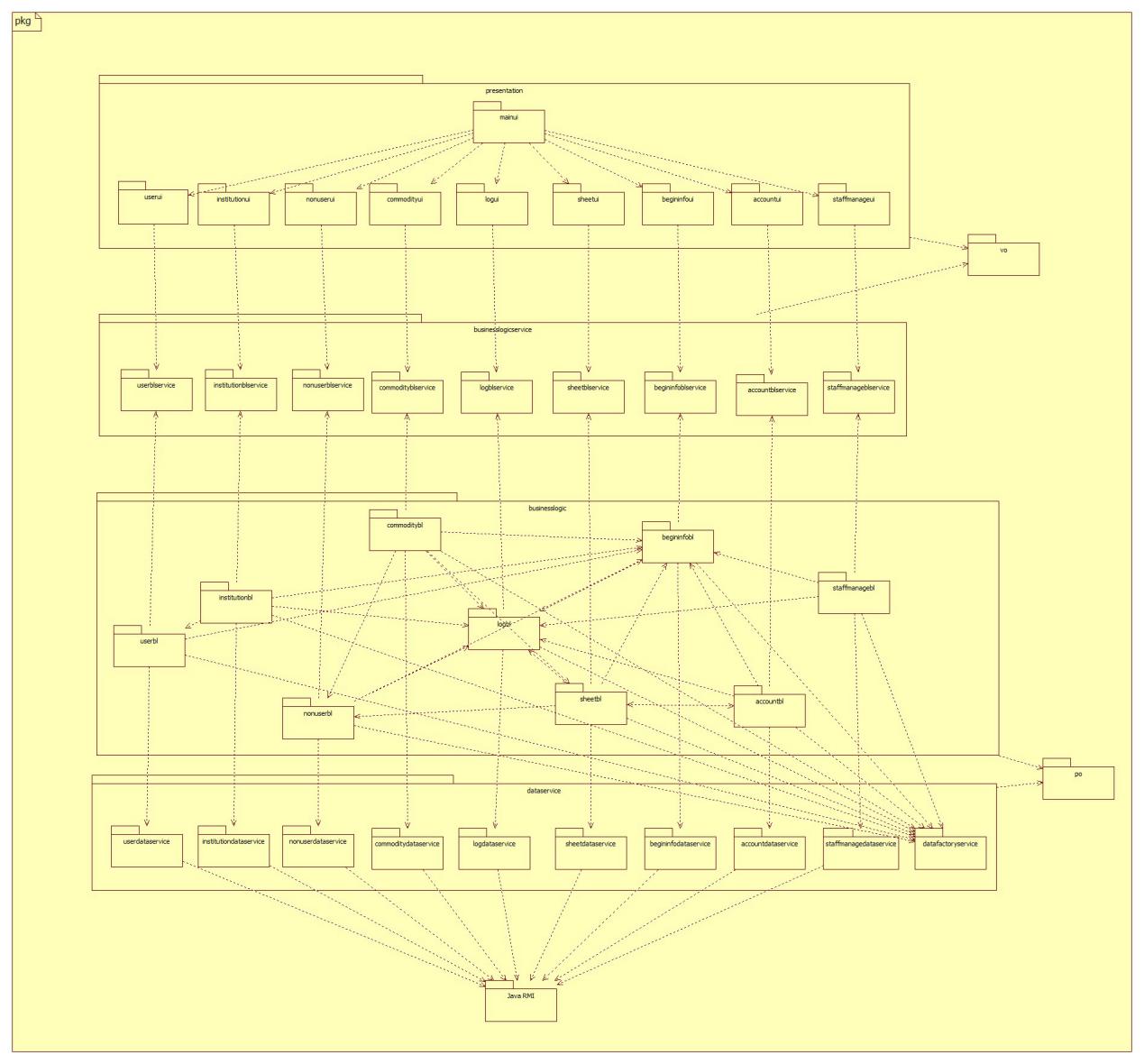


图3 快递物流系统客户端开发包图

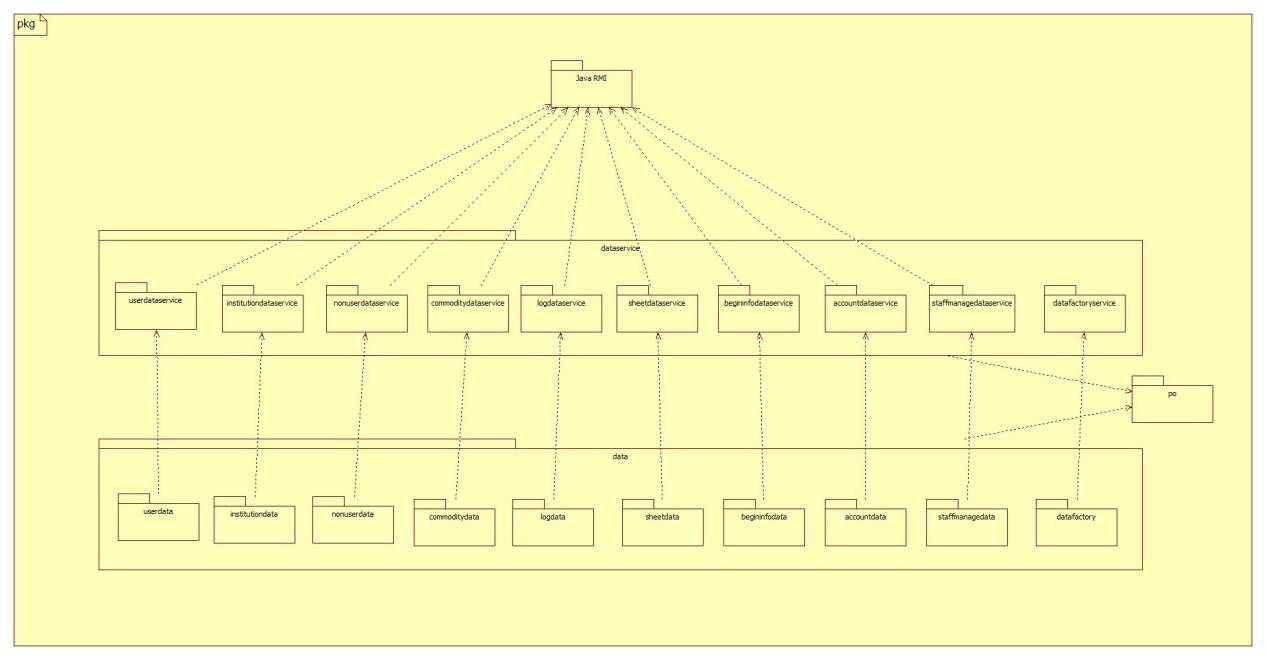


图4 快递物流系统服务器端开发包图

## 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，服务器端进程在服务器端机器上运行。



图5 进程图

## 4.3物理部署

快递物流系统中客户端构建是放在客户端机器上，服务器端构建是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI 构件属于JDK6.0的一部分，所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图6所示。

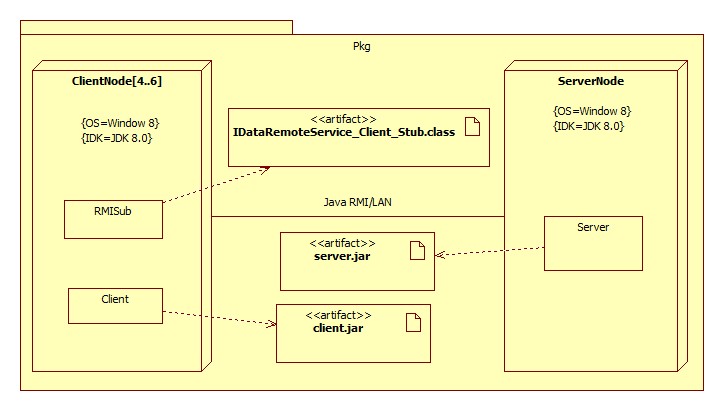


图6 部署图

# 5 接口视角

## 5.1模块的职责

客户端模块和服务器端模块视图分别如图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。



图7 客户端模块视图



图8 服务器端模块视图

**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客户端用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 服务器端各层的职责**

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

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表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** |
| CommodityBLService  UserBLService  LogBLService  AccountBLService  InstitutionBLServive  BeginInfoBLService  StaffManageBLService  SheetBLService  NonUserBLService | 客户端展示层 | 客户端业务逻辑层 |
| DatabaseFactory  CommodityDataService  UserDataService  LogDataService  AccountDataService  InstitutionDataService  BeginInfoDataService  StaffManageDataService  SheetDataService  NonUserDataService | 客户端业务逻辑层 | 服务器端数据层 |

借用单据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）。SheetBLService提供了Sheet界面所需要的所有业务逻辑功能。SheetDataService提供了对逻辑文件的增、改、查等操作。这样的实现就大大降低了层与层之间的耦合。

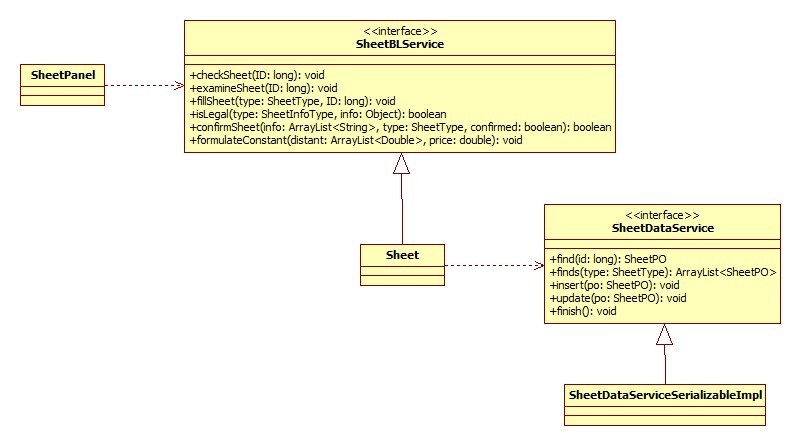


图9 单据用例层之间调用的接口

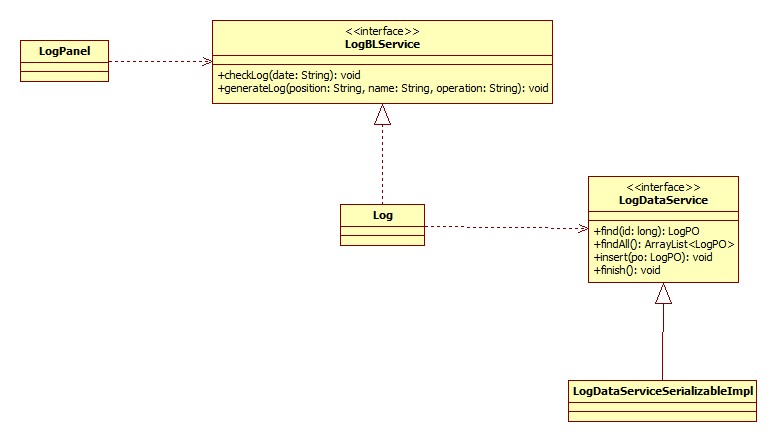


图10 日志用例层之间调用的接口

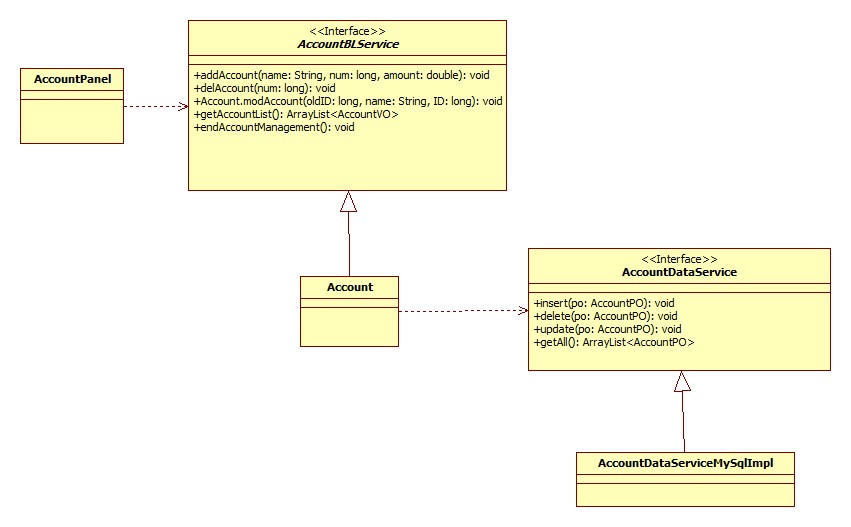


图11 账户用例层之间调用的接口

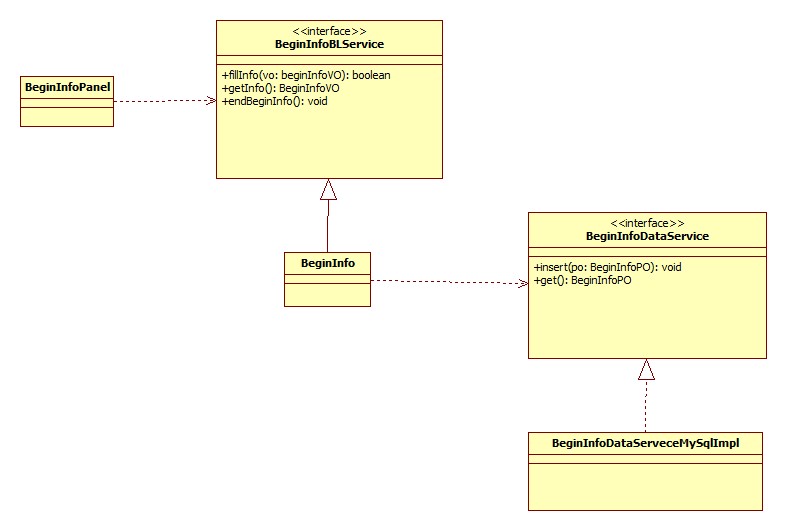


图12 期初建账用例层之间调用的接口



图13 用户管理用例层之间调用的接口



图14 人员机构管理用例层之间调用的接口



图15 个人帐号管理用例层之间调用的接口

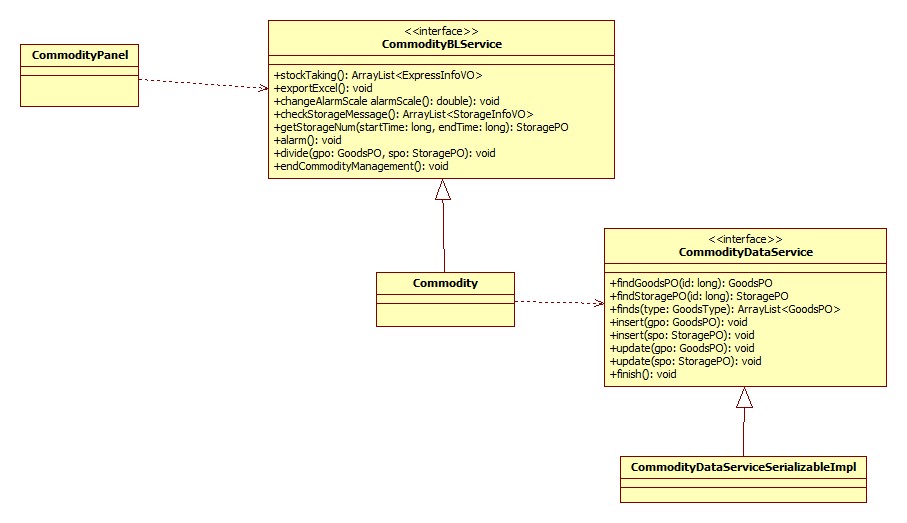


图16 库存用例层之间调用的接口

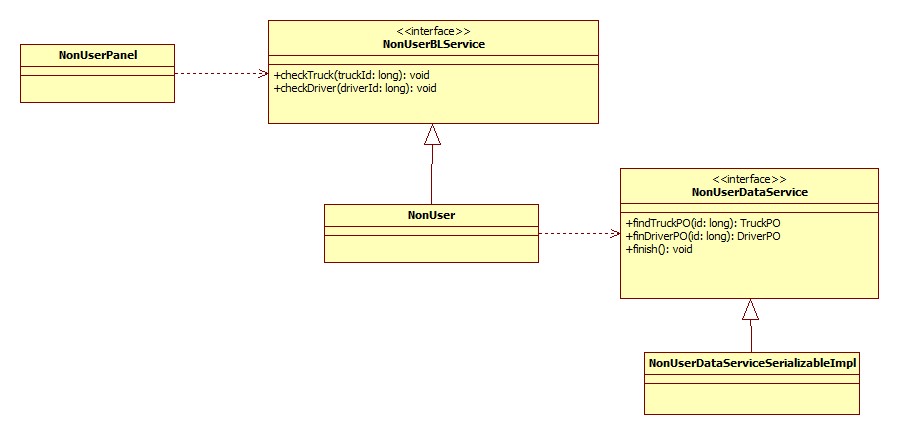


图17 非用户层之间调用的接口

## 5.2用户界面层的分解

根据需求，系统存在33个用户界面：库存管理人员界面、财务人员界面、总经理界面、管理员界面、快递员界面、营业厅业务员界面、中转中心业务员界面、入库单填写界面、出库单填写界面、库存分区界面、库存报警界面、库存盘点界面、人员机构管理界面、常量制定界面、成本收益表界面、单据审批界面、经营情况表界面、日志查询界面、银行账户管理界面、期初建账界面、付款单界面、调整用户界面、订单信息记录界面、收件信息记录界面、中转中心发货单界面、中转货物接收单界面、装车单界面、司机信息界面、收款单界面、派件单填写界面、接收单填写界面、车辆信息界面、车辆装车管理界面。界面跳转如图18所示。

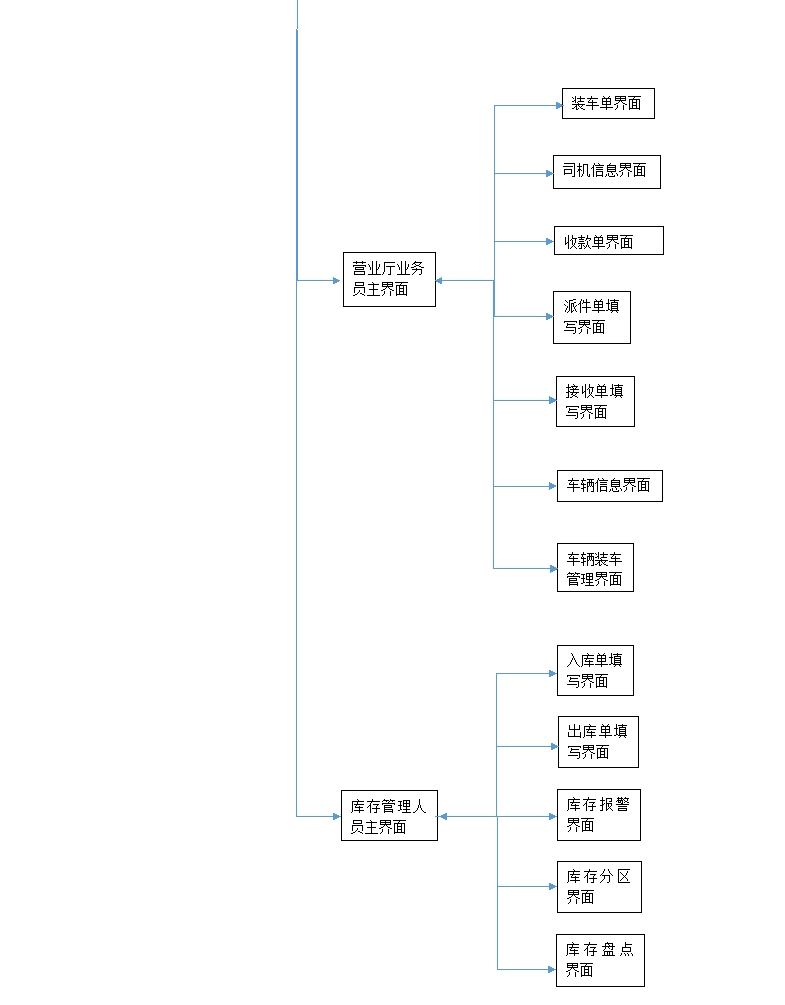
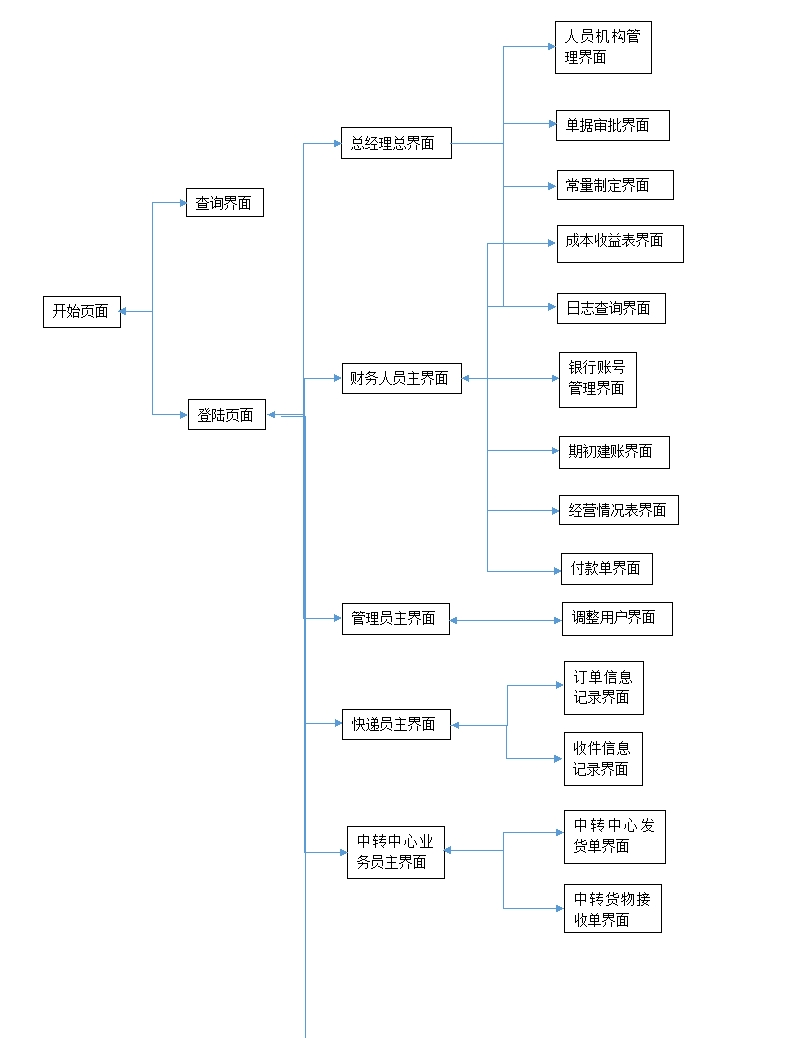


图18 用户界面跳转

服务器端和客户端的用户设计接口是一致的，只是具体的页面不一样，用户界面类如图19所示。

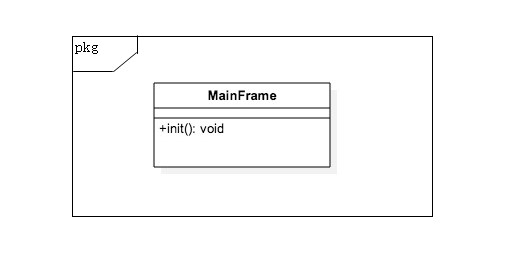


图19 用户界面类

### 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Frame,负责界面的显示和界面的跳转 |

### 5.2.2用户界面层模块的接口规范

用户界面层模块的接口规范如表6所示。

**表6 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Frame及LoginPanel |

用户界面层模块需要的服务接口如表7所示。

**表7 用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| presentation.MainFrame.init() | 初始化窗口，显示登陆界面。 |
| java.rmi.Naming.lookup () | 查找需要的RMI服务 |
| businesslogicservice.LoginBLService | 登陆界面的业务逻辑接口 |
| Businesslogicservice.\*BLService | 每个登陆界面都有一个相应的业务逻辑接口 |

### 5.2.3用户界面层模块设计原理

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，Sheet对象负责处理单据界面的业务逻辑；Log对象负责日志界面的业务逻辑。业务逻辑层的设计如图20所示。

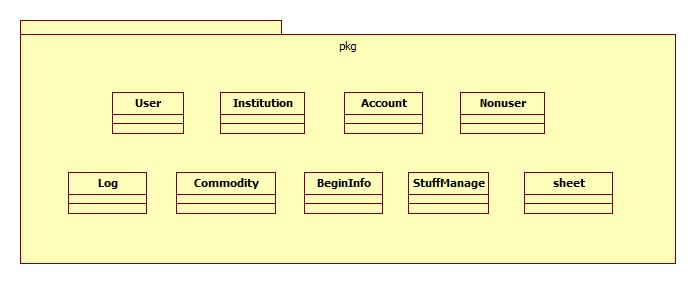


图20 业务逻辑层的设计

### 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8所示。

**表8 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模 块** | **职 责** |
| sheetbl | 负责实现与单据有关的界面所需要的服务 |
| logbl | 负责实现与操作日志有关的界面所需要的服务 |
| institutionbl | 负责实现与机构人员有关的界面所需要的服务 |
| userbl | 负责实现与用户个人及登陆有关的界面所需要的服务 |
| staffmanagebl | 负责实现与员工账号管理有关的界面所需要的服务 |
| accountbl | 负责实现与银行账户管理有关的界面所需要的服务 |
| begininfobl | 负责实现与期初建账有关界面所需要的服务 |
| nonuserbl | 负责实现与非用户信息有关界面所需要的服务 |
| commoditybl | 负责实现与库存有关界面所需要的服务 |

### 5.3.2业务逻辑层模块的接口规范

**表9 sheetbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Sheet.checkSheet | 语法 | public void checkSheet(long ID) | |
| 前置条件 | ID符合输入规则 | |
| 后置条件 | 返回此表格的完整信息 | |
| Sheet.examineSheet | 语法 | public void examineSheet(long ID) | |
| 前置条件 | ID符合输入规则 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Sheet.fillSheet | 语法 | public void fillSheet(int type,long ID) | |
| 前置条件 | type符合输入规则 | |
| 后置条件 | 根据type和ID启动一个填写表格回合 | |
| Sheet.isLegal | 语法 | public boolean isLegal(SheetInfoType type, Object info) | |
| 前置条件 | 启动一个检测输入信息回合 | |
| 后置条件 | 结束此次检测输入信息回合，并根据输入的info和type返回信息输入是否合法的结果 | |
| Sheet.confirmSheet | 语法 | public boolean confirmSheet(ArrayList<String> info, int type, boolean confirmed) | |
| 前置条件 | 启动一个填写表格回合 | |
| 后置条件 | 结束此次填写表格回合，持久化更新涉及的领域对象的数据 | |
| Sheet.formulateConstant | 语法 | public void formulateConstant(double distant, double price) | |
| 前置条件 | 启动一个制定常量回合 | |
| 后置条件 | 结束此次制定常量回合，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 |
| DatabaseFactory.getSheetDatabase | | | 得到Sheet数据库的服务的引用 |
| SheetDataService.insert(SheetPO po) | | | 插入单一持久化对象 |
| SheetDataService.update(SheetPO po) | | | 更新单一持久化对象 |
| SheetDataService.find(long ID) | | | 根据ID进行查找单一持久化对象 |
| SheetDataService.finds(int type) | | | 根据type进行查找多个持久化对象 |

**表10 logbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Log.checkLog | 语法 | public void checkLog(String date) | |
| 前置条件 | 启动一个查看操作日志的回合 | |
| 后置条件 | 如果date为空，查看所有操作日志，否则返回所选日期的操作日志，之后结束此次查看操作日志回合 | |
| Log.generateLog | 语法 | public void generateLog(String position, String name, String operation) | |
| 前置条件 | 操作已完成 | |
| 后置条件 | 生成操作日志，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 |
| DatabaseFactory.getLogDatabase | | | 得到Log数据库的服务的引用 |
| LogDataService.insert(LogPO po) | | | 插入单一持久化对象 |
| LogDataService.find(String date) | | | 根据日期进行查找多个持久化对象 |
| LogDataService.findAll() | | | 进行查找所有持久化对象 |

**表11 accountbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| Account.addAccount | 语法 | public boolean addAccount(String name,long num,double amount ); | |
| 前置条件 | 开启一次银行账号管理任务 | |
| 后置条件 | 增加一个银行账户 | |
| Account.delAccount | 语法 | public void delAccount (long num); | |
| 前置条件 | 启动一次银行账号管理任务 | |
| 后置条件 | 删除一个银行账户 | |
| Account.modAccount | 语法 | public void modAccount(long oldID,String name,long ID); | |
| 前置条件 | 启动一次银行账号管理任务 | |
| 后置条件 | 保存更新后的银行信息 | |
| Account.getAccountList | 语法 | public ArrayList<AccountVO> getAccountList(); | |
| 前置条件 | 启动一次银行账号管理任务 | |
| 后置条件 | 获取账号列表 | |
| Account.endAccountManagement | 语法 | public void endAccountManagement(); | |
| 前置条件 | 启动一次银行账号管理任务 | |
| 后置条件 | 结束银行账号管理任务 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| DatabaseFactory.getAccountDatabase | | | 得到Account数据库的服务的引用 |
| AccountDataService.insert(AccountPO po) | | | 插入单一持久化账户对象 |
| AccountDataService.delete(AccountPO po) | | | 删除单一持久化账户对象 |
| AccountDataService.getAll () | | | 获得所有Account持久化账户对象 |
| AccountDataService.getAll () | | | 获得单一持久化账户对象 |
| AccountDataService.update(AccountPo po) | | | 更新单一持久化账户对象 |

**表12 begininfobl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| beginInfo.fillInfo | 语法 | public boolean fillInfo(beginInfoVO vo); |
| 前置条件 | 启动系统 |
| 后置条件 | 系统保存期初信息并更新到持久化数据 |
| beginInfo.getInfo | 语法 | public BeginInfoVO getInfo(); |
| 前置条件 | 启动一次期初建账查看任务 |
| 后置条件 | 获取期初信息 |
| beginInfo.endBeginInfo | 语法 | public void endBeginInfo(); |
| 前置条件 | 启动一次期初建账查看任务 |
| 后置条件 | 关闭期初建账查看 |
| **需要的服务（需接口**） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getBeginInfoDatabase | | 得到Account数据库的服务的引用 |
| BeginInfoDataService.insert(BeginInfoPO po) | | 插入单一持久化期初信息对象 |
| BeginInfoDataService.get() | | 获得单一持久化期初信息对象 |

**表13 institutionbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Institution.getInstitutionInfo | 语法 | Public InstitutionVO getInstitutionInfo (long InstitutionId) |
| 前置条件 | 总经理启动人员机构管理 |
| 后置条件 | 在一次人员机构管理中获得机构信息 |
| Institution.addInstitution | 语法 | Public void addInstitution (InstitutionVO Institution) |
| 前置条件 | 总经理启动人员机构管理 |
| 后置条件 | 在一次人员机构管理中增加机构 |
| Institution. changeStaffPosition | 语法 | Public void changeStaffPosition (String position,long StaffId) |
| 前置条件 | 总经理启动人员机构管理 |
| 后置条件 | 在一次人员机构管理中更改员工所在部门 |
| Institution. deleteInstitution | 语法 | Public void deleteInstitution(long InstitutionId) |
| 前置条件 | 总经理启动人员机构管理 |
| 后置条件 | 在一次人员机构管理中删除机构 |
| Institution. changeSalary | 语法 | Public void changeSalary (long StaffId;int salary) |
| 前置条件 | 总经理启动一个人员工资管理回合 |
| 后置条件 | 一个人员工资管理回合中，更改员工的工资信息 |
| Institution.endManagement | 语法 | Public void endManagement() |
| 前置条件 | 已确认所有管理 |
| 后置条件 | 保存操作日志，保存所有已更新数据 |
| Institution.getAllInstitution | 语法 | Public ArrayList<InstitutionVO> getAllInstitution() |
| 前置条件 | 总经理启动人员机构管理 |
| 后置条件 | 在一次人员机构管理中获得所有机构信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| InstitutionDataService. findInstitution(long InstitutionId) | 根据ID进行查找单一持久化对象 | |
| InstitutionDataService. insertInstitution(InstitutionPO po) | 插入单一持久化对象 | |
| InstitutionDataService. deleteInstitution(InstitutionPO po) | 删除单一持久化对象 | |
| InstitutionDataService. updateInstitution(InstitutionPO po) | 更新单一持久化对象 | |
| InstitutionDataService. findAllInstitution() | 查找所有持久化对象 | |

**表13 userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| User.logIn | 语法 | Public ResultMessage logIn(long UserId,String password) |
| 前置条件 | 个人帐号拥有者启动账号信息管理 |
| 后置条件 | 登陆成功则显示相应界面，失败则显示登录失败 |
| User. getPersonalInfo | 语法 | Public StaffInfoVO getPersonalInfo(long UserId) |
| 前置条件 | 个人帐号拥有者启动账号信息管理 |
| 后置条件 | 在一次个人账号信息管理中获得个人帐号信息 |
| User. changePassword | 语法 | Public void changePassword(String password) |
| 前置条件 | 个人帐号拥有者启动账号信息管理 |
| 后置条件 | 员工在一次帐号信息管理中更改密码 |
| User. endManagement | 语法 | Public void endManagement() |
| 前置条件 | 已确认所有个人信息的管理 |
| 后置条件 | 保存操作日志，保存所有已更新数据 |
| User. initial | 语法 | Public void initial(ArrayList<InitUserVO> User) |
| 前置条件 | 期初建账 |
| 后置条件 | 在期初建账中初始化信息 |
| User.getCity | 语法 | Public String getCity(long UserId) |
| 前置条件 | 中转中心仓库管理员启动仓库管理回合 |
| 后置条件 | 获得仓库管理对应的城市 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| UserDataService.findPerson(long UserId) | 根据ID进行查找单一持久化对象 | |
| UserDataService. updatePassword(String password) | 更新单一持久化对象 | |

**表14 staffmanagebl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StaffManage.changeAuthority | 语法 | Public void changeAuthority (ArrayList<Integer> authority,String position) |
| 前置条件 | 管理员启动员工帐号管理 |
| 后置条件 | 在一次员工帐号管理中修改相应职位的权限 |
| StaffManage. getStaffInfo | 语法 | Public StaffInfoVO getStaffInfo (long StaffId) |
| 前置条件 | 管理员启动员工帐号管理 |
| 后置条件 | 在一次员工帐号管理中获得员工信息 |
| StaffManage. addStaffInfo | 语法 | Public void addStaffInfo (StaffInfoVO StaffInfo) |
| 前置条件 | 管理员启动员工帐号管理 |
| 后置条件 | 在一次员工帐号管理中增加员工帐号信息 |
| StaffManage. deleteStaff | 语法 | Public void deleteStaff (long StaffId) |
| 前置条件 | 管理员启动员工帐号管理 |
| 后置条件 | 在一次员工帐号管理中删除员工帐号信息 |
| StaffManage.endManagement | 语法 | Public void endManagement() |
| 前置条件 | 已确认所有管理 |
| 后置条件 | 保存操作日志，保存所有已更新数据 |
| StaffManage. getAllStaff | 语法 | Public ArrayList<StaffInfoVO> getAllStaff () |
| 前置条件 | 管理员启动员工帐号管理 |
| 后置条件 | 在一次员工帐号管理中获得所有员工帐号信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StaffManageDataService. insertStaff(StaffInfoPO po) | 插入单一持久化对象 | |
| StaffManageDataService. deleteStaff(StaffInfoPO po) | 删除单一持久化对象 | |
| StaffManageDataService. updateStaff(StaffInfoPO po) | 更新单一持久化对象 | |
| StaffManageDataService. findStaff(long StaffId) | 根据ID进行查找单一持久化对象 | |
| StaffManageDataService. findAllStaff() | 查找所有持久化对象 | |

**表15 commoditybl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Commodity.stockTaking | 语法 | public ArrayList<ExpressInfoVO> stockTaking() |
| 前置条件 | 启动一个库存盘点的回合 |
| 后置条件 | 在一个库存盘点回合中，显示货物信息，完成对相应库存信息的盘点 |
| Commodity.exportExcel | 语法 | public void exportExcel () |
| 前置条件 | 启动一个导出Excel的回合 |
| 后置条件 | 导出库存信息的Excel表格 |
| Commodity.changeAlarmScale | 语法 | public void changeAlarmScale (double alarmScale) |
| 前置条件 | 启动一个修改警戒比例的回合 |
| 后置条件 | 在一个警戒比例修改回合中完成对警戒比例的修改 |
| Commodity.checkStorageMessage | 语法 | Public ArrayList<StorageInfoVO> checkStorageMessage () |
| 前置条件 | 启动一个查看库存信息的回合 |
| 后置条件 | 在一个查看库存信息回合中显示一段时间内出入库的货物的存储位置 |
| Commodity. getStorageNum | 语法 | public StoragePO getStorageNum(long startTime,long endTime) |
| 前置条件 | 启动一个查看库存信息的回合 |
| 后置条件 | 在一个查看库存信息回合中显示一段时间内的出/入库数量/金额，合计的库存数量 |
| Commodity.alarm | 语法 | public void alarm () |
| 前置条件 | 货物数量发生变化 |
| 后置条件 | 若货物数量超出报警值则报警 |
| Commodity.divide | 语法 | public void divide (GoodsPO gpo，StoragePO spo) |
| 前置条件 | 启动一个库存分区的回合 |
| 后置条件 | 在一个库存分区回合中完成对货物的库存分区 |
| Commodity.endCommodityManagement | 语法 | public void endCommodityManagement() |
| 前置条件 | 已完成库存操作 |
| 后置条件 | 结束此次库存操作，持久化更新涉及的领域的对象的数据 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| CommodityDataService.find(long id) | 根据ID进行查找单一持久化对象 | |
| CommodityDataService.finds(GoodsType type) | 根据字段名和值进行查找多个持久化对象 | |
| CommodityDataService.insert(GoodsPO gpo) | 插入单一持久化对象 | |
| CommodityDataService.update(GoodsPO gpo) | 更新单一持久化对象 | |

**表16 nonuserbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| nonUserbl.checkTruck | 语法 | public void checkTruck(long truckId) |
| 前置条件 | 启动一个查询车辆信息的回合 |
| 后置条件 | 在一个车辆信息查询回合中，显示车辆信息，包括车辆代号、服役时间、车牌号 |
| nonUserbl.checkDriver | 语法 | public void checkDriver (long driverId) |
| 前置条件 | 启动一个查询司机信息的回合 |
| 后置条件 | 在一个司机信息查询回合中，显示司机信息，包括司机编号、姓名、出生日期、身份证号、手机 、性别、行驶证期限 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| nonUserblDataService.find(int id) | 根据ID进行查找单一持久化对象 | |
| nonUserblDataService.finds(String field,int value) | 根据字段名和值进行查找多个持久化对象 | |

## 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。这些服务由DataService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。具体如图20所示。

图20 数据层模块的描述

### 5.4.1数据层模块的职责

**表18 Sheet数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| SheetDataService | 持久化数据库的接口，提供集体载入、集体保存、增、改、查服务 |
| SheetDataServiceSerializableFileImpl | 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、改、查服务 |

**表19 Log数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| LogDataService | 持久化数据库的接口，提供集体载入、集体保存、增、查服务 |
| LogDataServiceSerializableFileImpl | 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、增查服务 |

**表20 Account数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountDataService | 持久化的接口，提供增、删、改、查服务 |
| AccountDataServiceSerializableFileImpl | 基于序列化文件的持久化数据库的接口，提供增、删、改、查服务 |

**表22 Account数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| BeginInfoDataService | 持久化的接口，提供增、查服务 |
| BeginInfoDataServiceSerializableFileImpl | 基于序列化文件的持久化数据库的接口，提供增、查服务 |

**表23 Institution数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| InstitutionDataService | 持久化的接口，提供增、删、改、查服务 |
| InstitutionDataServiceSerializableFileImpl | 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 |

**表24 User数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserDataService | 持久化的接口，提供改、查服务 |
| UserDataServiceSerializableFileImpl | 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、改、查服务 |

**表25 StaffManage数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| StaffManageDataService | 持久化的接口，提供增、删、改、查服务 |
| StaffManageDataServiceSerializableFileImpl | 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 |

**表26 Commodity数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| CommodityDataService | 持久化的接口，提供增、删、改、查服务 |
| CommodityDataServiceSerializableFileImpl | 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 |

**表27 NonUser数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| NonUserDataService | 持久化的接口，提供增、删、改、查服务 |
| NonUserDataServiceSerializableFileImpl | 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

**表28 Sheet数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SheetDataService.find | 语法 | public SheetPO 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SheetPO结果 |
| SheetDataService.finds | 语法 | public ArrayList<SheetPO> finds(SheetType type) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据type进行查找返回多个相应的SheetPO结果 |
| SheetDataService.insert | 语法 | public void insert(SheetPO po) throws RemoteException; |
| 前置条件 | 同样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 |
| SheetDataService.update | 语法 | public void update(SheetPO 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 |
| SheetDataService.finish | 语法 | public void finish() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

**表29 Log数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LogDataService.find | 语法 | public LogPO 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LogPO结果 |
| LogDataService.findAll | 语法 | public ArrayList<LogPO> findAll() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回所有的LogPO结果 |
| LogDataService.insert | 语法 | public void insert(LogPO po) throws RemoteException; |
| 前置条件 | 同样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 |
| LogDataService.finish | 语法 | public void finish() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

**表30 Account数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| AccountDataService.insert | 语法 | public void insert(AccountPO po) throws RemoteExceptions; |
| 前置条件 | 同样的ID的po中不存在 |
| 后置条件 | 增加一个银行账户 |
| AccountDataService.delete | 语法 | public void delete(AccountPO po)throws RemoteExceptions; |
| 前置条件 | 存在同样ID的po |
| 后置条件 | 删除一个po |
| AccountDataService.getAll () | 语法 | public ArrayList<AccountPO> getAll () throws RemoteExceptions; |
| 前置条件 | 无 |
| 后置条件 | 返回所有的AccountPO |
| AccountDataService.update | 语法 | public void update(AccountPo po)throws RemoteException; |
| 前置条件 | 存在同样ID的po |
| 后置条件 | 更新一个po |

**表31 BeginInfo数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BeginInfoDataService.insert | 语法 | public void insert(BeginInfoPO po) throws RemoteExceptions; |
| 前置条件 | 无 |
| 后置条件 | 增加一个银行账户 |
| BeginInfoDataService.get | 语法 | public void get()throws RemoteExceptions; |
| 前置条件 | 存在期初信息 |
| 后置条件 | 获得一个期初信息po |

**表32 User数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserDataService.findPerson | 语法 | public StaffInfoPO findPerson (long UserId) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的PO结果 |
| UserDataService.updatePassword | 语法 | public void updatePassword (String password) throws RemoteException; |
| 前置条件 | 已登录个人帐号信息更改界面 |
| 后置条件 | 更新密码 |

**表33 StaffManage数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StaffManageDataService.findStaff | 语法 | public StaffInfoPO findStaffInfo (long StaffId) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的PO结果 |
| StaffManageDataService.insertStaff | 语法 | public void insertStaff (StaffInfoPO po) throws RemoteException; |
| 前置条件 | 同样ID的po不存在 |
| 后置条件 | 增加一个po记录 |
| StaffManageDataService.deleteStaff | 语法 | public void deleteStaff (StaffInfoPO po) throws RemoteException; |
| 前置条件 | 存在同样ID 的po |
| 后置条件 | 删除一个po |
| StaffManageDataService.updateStaff | 语法 | public void updateStaff (StaffInfoPO po) throws RemoteException; |
| 前置条件 | 存在同样ID 的po |
| 后置条件 | 更新一个po |
| StaffManageDataService.findAllStaff | 语法 | public ArrayList<StaffInfoPO> findAllStaff() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回所有PO结果 |

**表34 Institution数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InstitutionDataService.findInstitution | 语法 | public InstitutionPO findInstitution (long InstitutionId) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的PO结果 |
| InstitutionDataService.insertInstitution | 语法 | public void insertInstitution (InstitutionPO po) throws RemoteException; |
| 前置条件 | 同样ID的po不存在 |
| 后置条件 | 增加一个po记录 |
| InstitutionDataService.deleteInstitution | 语法 | public void deleteInstitution (InstitutionPO po) throws RemoteException; |
| 前置条件 | 存在同样ID 的po |
| 后置条件 | 删除一个po |
| InstitutionDataService.updateInstitution | 语法 | public void updateInstitution (InstitutionPO po) throws RemoteException; |
| 前置条件 | 存在同样ID 的po |
| 后置条件 | 更新一个po |
| InstitutionDataService.findAllInstitution | 语法 | public ArrayList<InstitutionPO> findAllInstitution() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回所有PO结果 |

**表35 Commodity数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CommodityDataService.findGoodsPO | 语法 | public GoodsPO findGoodsPO(long id) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的GoodsPO结果 |
| CommodityDataService.findStoragePO | 语法 | public StoragePO findStoragePO(long id) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的StoragePO结果 |
| CommodityDataService. finds | 语法 | public ArrayList<GoodsPO> finds(GoodsType type) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的一组GoodsPO 结果 |
| CommodityDataService.insert | 语法 | public void insert(GoodsPO gpo) throws RemoteException; |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在Mapper中增加一个GoodsPO记录 |
| CommodityDataService.insert | 语法 | public void insert(StoragePO spo) throws RemoteException; |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在Mapper中增加一个StoragePO记录 |
| CommodityDataService.update | 语法 | public void update(GoodsPO gpo) throws RemoteException; |
| 前置条件 | 在Mapper中存在同样ID 的po |
| 后置条件 | 更新一个GoodsPO |
| CommodityDataService.update | 语法 | public void update(StoragePO spo) throws RemoteException; |
| 前置条件 | 在Mapper中存在同样ID 的po |
| 后置条件 | 更新一个StoragePO spo |
| CommodityDataService.finish | 语法 | public void finish()throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用。 |

**表36 Commodity数据层模块的接口**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| NonUserDataService. findTruckPO | 语法 | public TruckPO findTruckPO(long id) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的TruckPO结果 |
| NonUserDataService. findDriverPO | 语法 | public DriverPO finDriverPO(long id) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的DriverPO结果 |
| NonUserDataService.finish | 语法 | public void finish()throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用。 |

# 6 信息视角

## 6.1数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

* + ConstantPO类是保存系统常量的类，包括城市距离常量和价格常量。
  + LogPO类包含操作日志信息的职位、姓名、操作名称和时间。
  + OrderSheetPO类包含订单信息的快递物流编号、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、收件状态和收件时间。
  + PaymentSheetPO类包含付款单信息的付款日期、付款金额、付款人、付款账号、条目和备注部分。
  + StorageInSheetPO类包含快递编号，入库日期，目的地，区号，排号，架号，位号几个属性。
  + StorageOutSheetPO类是包含快递编号，入库日期，目的地，装运形式，中转单编号，汽运编号几个属性。
  + ReceiveSheetPO类包含到达日期、中转中心编号、出发地、货物到达状态和接收的业务员。
  + SendSheetPO类包括出发日期、到达日期、出发地、到达地、托运订单条形码号和派送员。
  + CashPO类包括了收款日期、收款金额、收款快递员、对应的所有快递订单条形码号
  + GoodsPO类包含物流状态，原件数量，货物的长宽高，实际重量，体积，内件品名，快递方式，包装方式，快递编号，入库日期，目的地，区号，排号，架号，位号，装运形式，中转单编号，汽运编号几个属性。
  + StoragePO类包含商品库存数量，警戒比例几个属性。
  + AccountPO类包含账户名、银行账号和金额。
  + BeginInfoPO类包含常量信息、库存信息、人员机构信息、银行账户信息和司机车辆信息。
  + UserPO类包含用户的工号、账号、密码、姓名、职位、权限和薪水。
  + InstitutionPO类包含机构的编号、名字、职能、员工信息。

持久化对象定义详情见代码。