酒店预订系统

体系结构设计文档

V1.0

南京大学软件学院

**Leftovers**工作组

2016-10

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 林志和 | 2016-10-10 | 确定整体模板，目录等 | V1.0 |
| 郭浩滨 | 2016-10-10 | 添加逻辑设计方案及客户端、服务器端开发包图 | V1.0.1 |
| 林志和 | 2016-10-10 | 添加词汇表、开发（物理）包图的表 | V1.0.2 |
| 吴志成 | 2016-10-10 | 添加界面跳转图，部署图，客户端、服务器端模块视图 | V1.0.3 |

目录

[1. 引言 4](#_Toc463824208)

[1.1. 编制目的 4](#_Toc463824209)

[1.2. 词汇表 4](#_Toc463824210)

[1.3. 参考资料 4](#_Toc463824211)

[2. 产品描述 4](#_Toc463824212)

[3. 逻辑视角 5](#_Toc463824213)

[4. 组合视角 6](#_Toc463824214)

[4.1. 开发包图 6](#_Toc463824215)

[4.2. 运行时进程 9](#_Toc463824216)

[4.3. 物理部署 10](#_Toc463824217)

[5. 接口视角 10](#_Toc463824218)

[5.1. 模块的职责 10](#_Toc463824219)

[5.2. 用户界面层的分解 12](#_Toc463824220)

[5.2.1. 用户界面层模块的职责 13](#_Toc463824221)

[5.2.2. 用户界面模块的接口规范 14](#_Toc463824222)

[5.2.3. 用户界面模块设计原理 14](#_Toc463824223)

[5.3. 业务逻辑层的分解 14](#_Toc463824224)

[5.3.1. 业务逻辑层模块的职责 14](#_Toc463824225)

[5.3.2. 业务逻辑模块的接口模范 15](#_Toc463824226)

[5.4. 数据层的分解 33](#_Toc463824227)

[5.4.1. 数据层模块的职责 34](#_Toc463824228)

[5.4.2. 数据层模块的接口规范 34](#_Toc463824229)

[6. 信息视角 48](#_Toc463824230)

[6.1. 数据持久化对象 48](#_Toc463824231)

[6.2. 文件格式 49](#_Toc463824232)

# 引言

## 编制目的

本报告详细完成对酒店预订系统的概要设计，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编写，是了解系统的导航。

## 词汇表

|  |  |
| --- | --- |
| **缩写或单词** | **解释** |
| HOS | 酒店预订系统 |
| member | 会员 |
| user | 客户 |
| personnel | 员工，包括酒店工作人员，网站营销人员，网站管理人员 |
| hotel | 酒店 |
| room | 客房 |
| credit | 信用 |
| evaluation | 评价 |
| comment | 评论 |
| rank | 评分 |
| hotelpromotion | 酒店促销策略 |
| webpromotion | 网站营销策略 |
| order | 订单 |
| unexecuted | 未执行的（订单） |
| executed | 已执行的（订单） |
| abnormal | 异常的（订单） |
| canceled | 已撤销的（订单） |
| appeal | 异常订单申诉 |

## 参考资料

《酒店预订系统用例文档V1.2.11》

《酒店预订系统需求规格说明文档V1.1.5》

# 产品描述

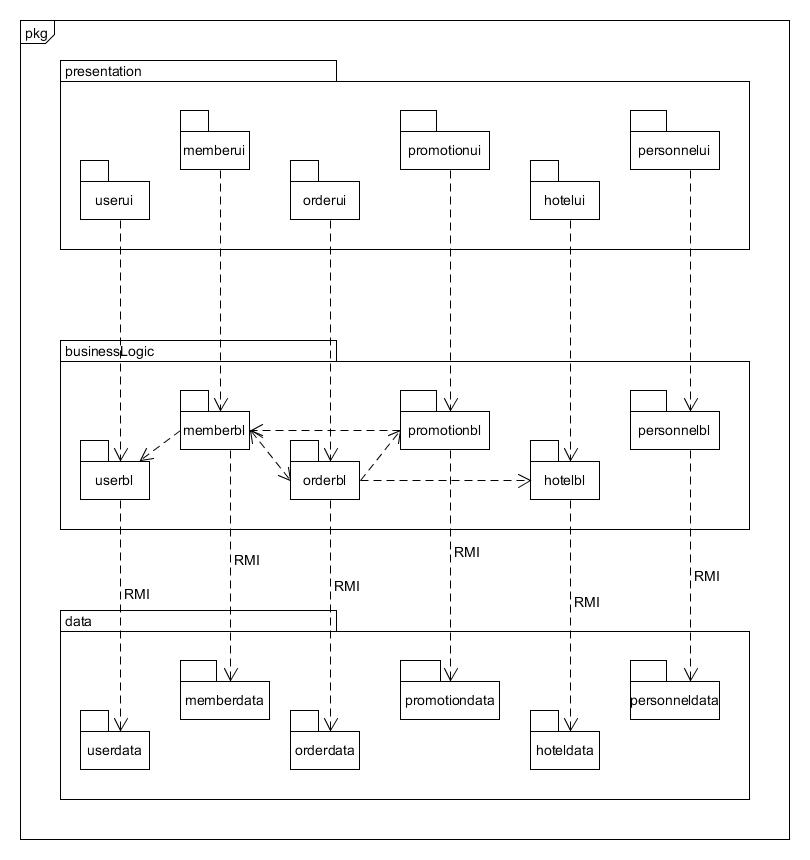
参考《酒店预订系统用例文档》和《酒店预订系统需求规格说明文档》中对产品的概括描述。

# 逻辑视角

酒店预订系统中，选择了分层体系结构的风格，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。展示层包括GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。



**图1：分层结构**



**图2：逻辑设计方案**

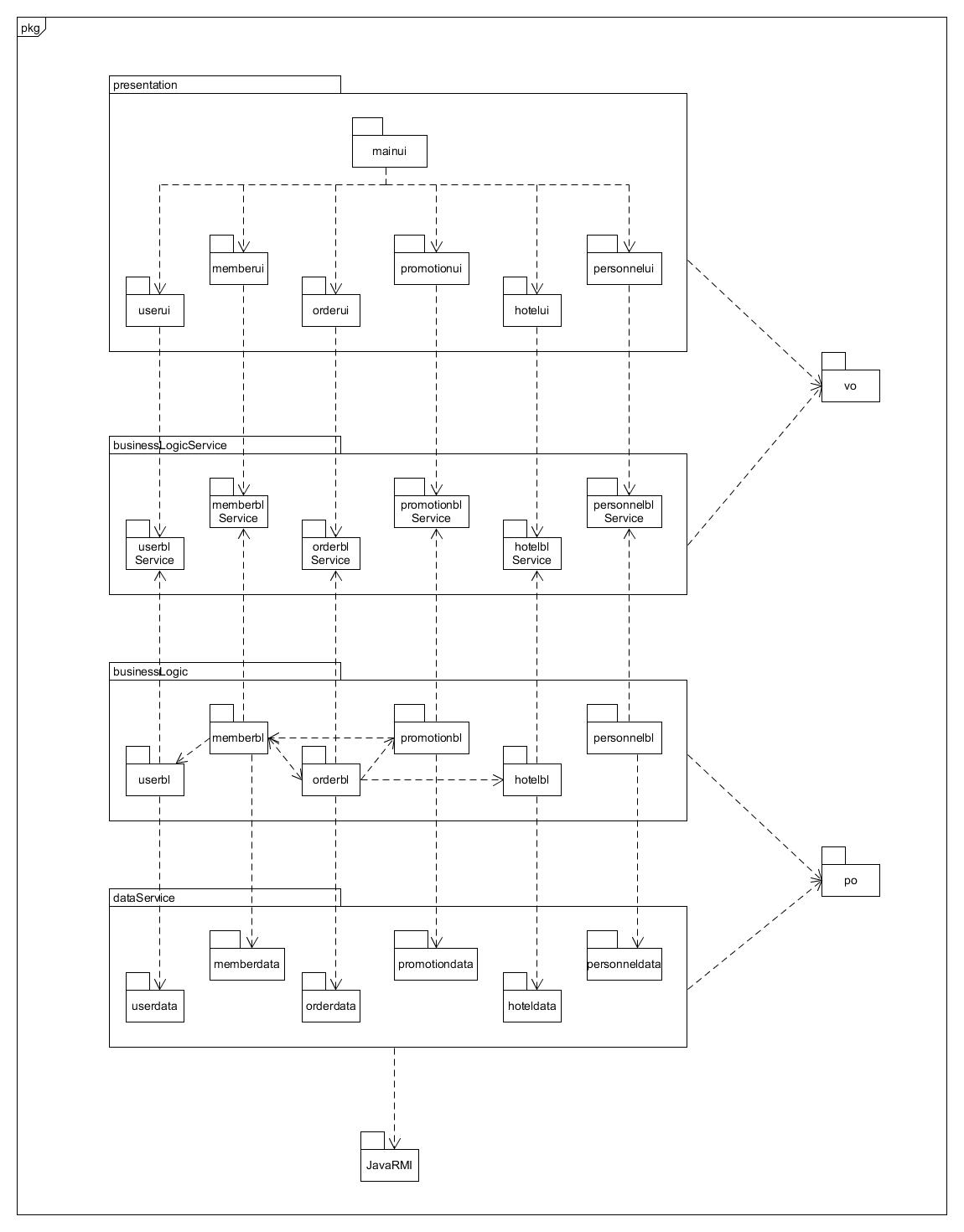
# 组合视角

## 开发包图

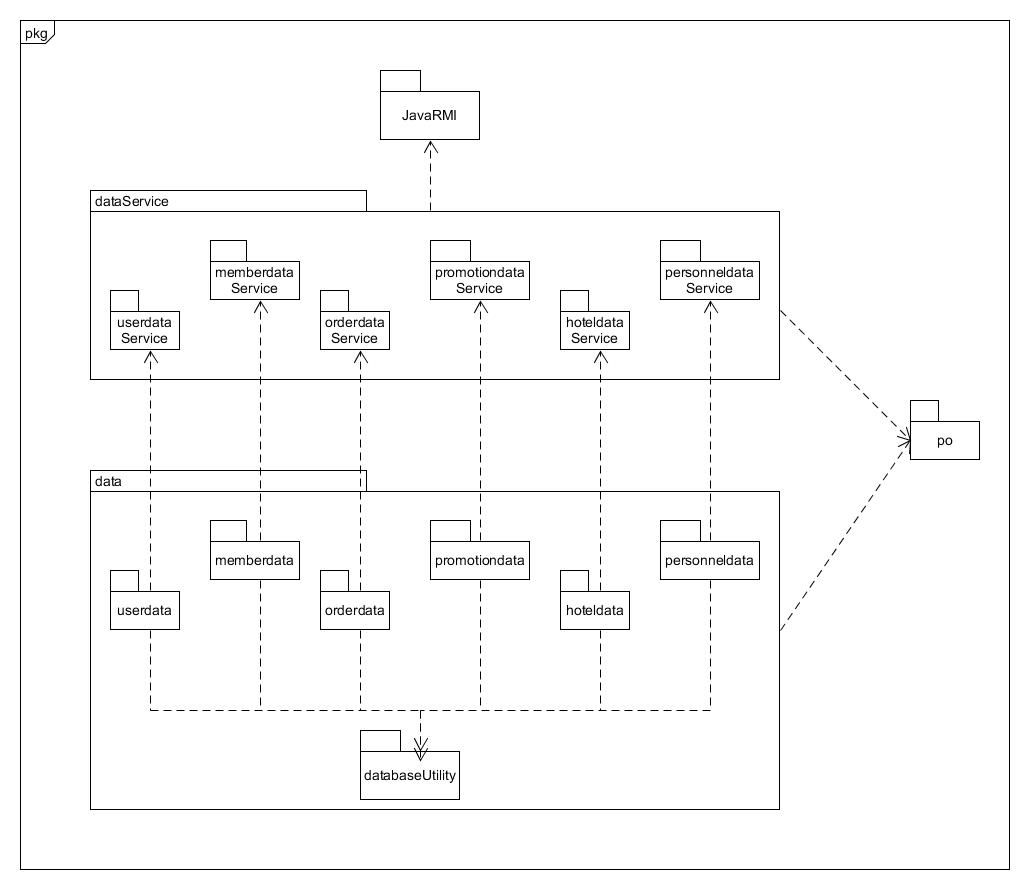
酒店预订系统的最终开发包设计如表1所示：

**表1酒店预订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, memberui, orderui, promotionui, hotelui, personnelui, vo |
| userui | userblservice, 界面类库包, vo |
| userblservice |  |
| userbl | userdataservice, userblservice, po |
| userdataservice | Java RMI, po |
| userdata | userdataservice, databaseseutility, po |
| memberui | memberblservice, 界面类库包, vo |
| memberblservice |  |
| memberbl | memberdataservice, memberblservice, userbl, orderbl, po |
| memberdataservice | Java RMI,po |
| memeberdata | memberdataservice, databaseseutility, po |
| orderui | orderblservice, 界面类库包, vo |
| orderblservice |  |
| orderbl | orderdataservice, orderblservice, memberbl, promotionbl, hotelbl, po |
| orderdataservice | Java RMI, po |
| orderdata | orderdataservice, databaseseutility, po |
| promotionui | promotionblservice, 界面类库包, vo |
| promotionblservice |  |
| promotionbl | promotiondataservice, promotionblservice, memberbl, po |
| promotiondataservice | Java RMI, po |
| promotiondata | promotiondataservice, databaseseutility, po |
| hotelui | hotelblservice, 界面类库包, vo |
| hotelblservice |  |
| hotelbl | hoteldataservice, hotelblservice, po |
| hoteldataservice | Java RMI, po |
| hoteldata | hoteldataservice, databaseseutility, po |
| personnelui | personnelblservice, 界面类库包, vo |
| personnelblservice |  |
| personnelbl | personneldataservice, personnelblservice, po |
| personneldataservice | Java RMI, po |
| personneldata | personneldataservice, databaseseutility, po |
| vo |  |
| po |  |
| 界面类库包 |  |
| databaseseutility | JDBC |
| Java RMI |  |



**图3 客户端开发包图**



**图4 服务器端开发包图**

## 运行时进程

在酒店预订系统中，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，服务器端进程是在服务器端机器上运行。

**图5 进程图**



## 物理部署

酒店预订系统中客户端构件是放在客户端机上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI 构件属于JDK 1.8的一部分。所以，在系统JDK环境已经设置好的情况下，不需要独立部署。部署图如图6所示。



**图 6 部署图**

# 接口视角

## 模块的职责

客户端模块和服务器模块视图分别如图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。

**启动模块**

**业务逻辑层**

**展示层**

**网络模块**

**图7 客户端模块视图**

**网络模块**

**启动模块**

**业务逻辑层**

**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 | 基于窗口的进销存管理系统客户端用户界面 |
| 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 |

**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 |

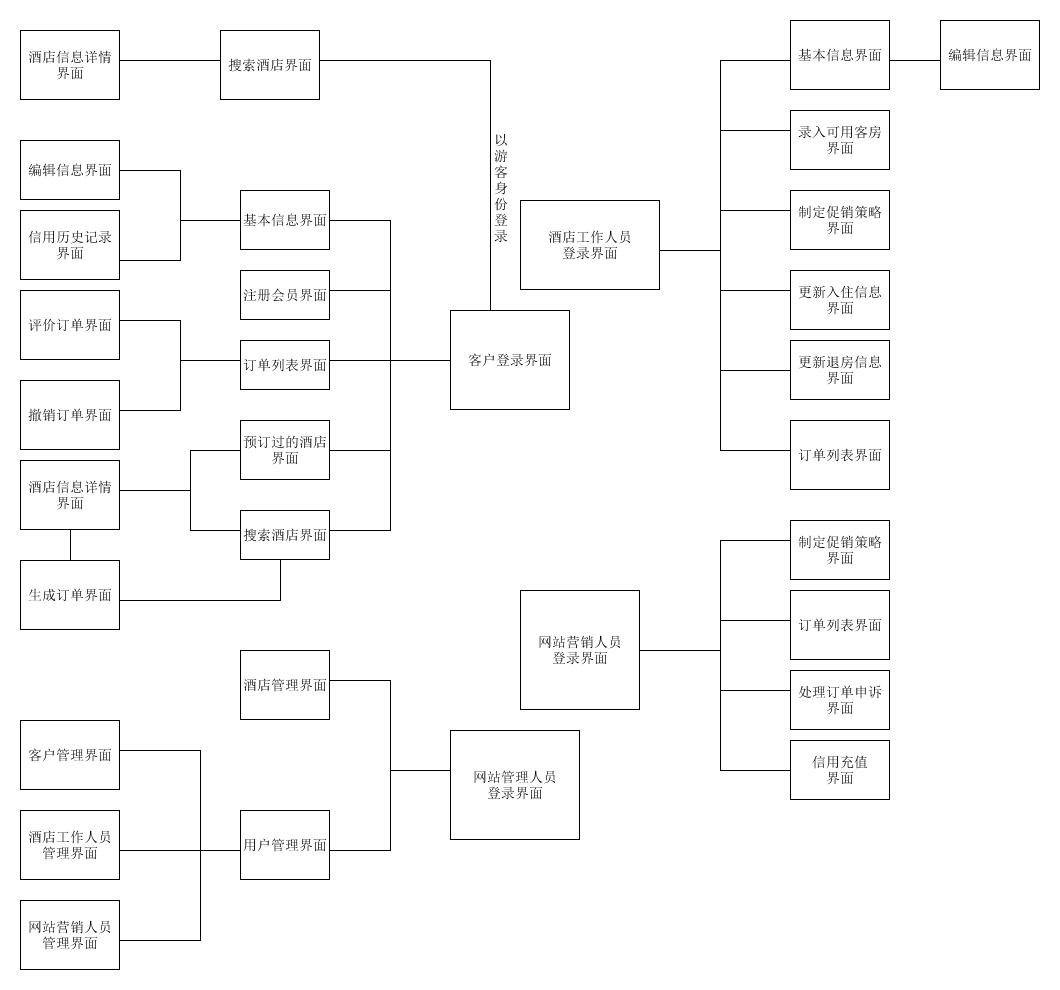
每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| UserMesbl  Accountbl  Personnelbl  Statisticbl  Approvalbl  Stretegybl  Orderbl  Financebl  Storebl  DTmanagebl  transportbl | 客户端展示层 | 客户端业务逻辑层 |
| Accountdataservice  StatisticDataService  PersonnelDataservice  StretegyDataService  financedataservice  storedataservice  DTmanageDataService  TransportDataService | 客户端业务逻辑层 | 服务器端数据层 |

## 用户界面层的分解

根据需求，系统存在33个用户界面（加粗为主界面）：**客户登录界面、**（客户）基本信息界面、（客户）编辑信息界面、信用历史记录界面、注册会员界面、（客户）订单列表界面、评价订单界面、撤销订单界面、预订过的酒店界面、酒店信息详情界面、搜索酒店界面、生成订单界面、**酒店工作人员登录界面、**（酒店）基本信息界面、（酒店）编辑信息界面、录入可用客房界面、（酒店）制定促销策略界面、更新入住信息界面、更新退房信息界面、（酒店）订单列表界面、**网站营销人员登录界面、**（网站）制定促销策略界面、（网站）订单列表界面、处理订单申诉界面、信用充值界面、**网站管理人员登录界面、**酒店管理界面、用户管理界面、客户管理界面、酒店工作人员管理界面、网站营销人员管理界面。



**图9 酒店预订系统界面跳转图**

### 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 |

### 用户界面模块的接口规范

**表6 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **MainFrame** | 语法 | init(args:String[]) |
| 前置条件 | 用户触发相应需要显示界面的条件 |
| 后置条件 | 显示Frame并且加载LoginPanel |

**表7 用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| **服务名** | **服务** |
| bl.accountservice | 登录界面的业务逻辑接口 |
| bl.\*blservice | 每个ui模块会有对应的数据逻辑接口 |

### 用户界面模块设计原理

用户界面拟用java的Swing和AWT库来实现。

## 业务逻辑层的分解

业务逻辑层包括多个针对界面及业务逻辑处理对象。例如User对象负责处理登录界面

的业务逻辑。业务逻辑层的设计如下图所示：

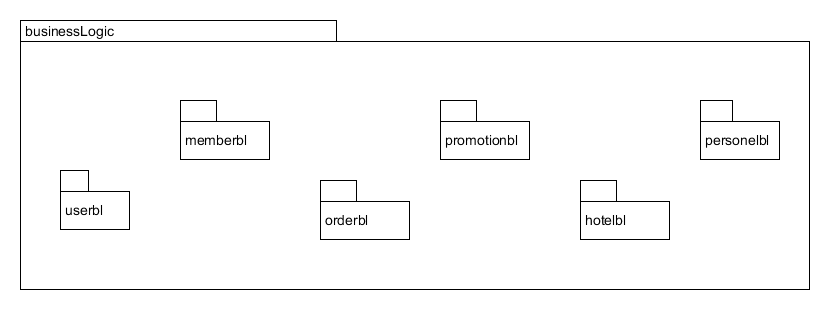


图10 业务逻辑层的设计

### 业务逻辑层模块的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| Userbl | 实现登录与个人账户信息修改的服务 |
| Accountbl | 实现与管理员有关的对账户增删改查操作的服务 |
| Personnelbl | 实现与人员机构管理相关的服务 |
| Statisticbl | 实现与查看相关报表、期初建账的服务 |
| Approvalbl | 实现与审批单据相关的服务 |
| Strategybl | 实现与制定常量、薪水策略的操作相关的服务，为其他包提供查询的接口 |
| Orderbl | 实现增、删、查订单的信息，收件信息录入 |
| Financebl | 实现成本与收益信息管理 |
| Storebl | 负责实现出入库和库存管理，库存报警、库存盘点（快照） |
| DTmanagebl | 负责实现车辆信息管理和司机信息管理所需要的增删改查 |
| transportbl | 负责实现与流转中产生的接收单、派件单、中转单、到达单、装车单相关的服务 |

### 业务逻辑模块的接口模范

#### orderbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| **Order.add** | 语法 | Public ResultMessage add(orderVO orderVO) |
| 前置条件 | 输入的数据格式正确 |
| 后置条件 | 系统自动生成新的订单，并持久化添加数据 |
| **Order.checkBarCode** | 语法 | Public ResultMessage checkBarCode(String  orderBarCode) |
| 前置条件 | 在presentation层已经确认barCode（条形码）为10位 |
| 后置条件 | 无 |
| **Order.getOrderVO** | 语法 | public ArrayList< OrderVO >  getOrderVO(Date date) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.del | 语法 | Public ResultMessage del(String  orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化删除所在订单的所有信息 |
| **Order.getSimpleInfo** | 语法 | Public ArrayList<OrderSimpleInfoVO> getSimpleInfo(String orderBarCode); |
| 前置条件 | 在presentation层已经对输入的条形码进行判断，数据库内存在订单 |
| 后置条件 | 无 |
| **Order.getFullInfo** | 语法 | public OrderVO getFullInfo(String orderBarCode); |
| 前置条件 | 在presentation层已经对输入的条形码进行判断，数据库内存在订单 |
| 后置条件 | 无 |
| **Order.receiveInfo** | 语法 | Public ResultMessage  receiveInfo(ArrayList<ReceiveVO>) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新所在订单的收件信息 |
| **Order.addDocList** | 语法 | Public ResultMessage addDocList(DocVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化更新所在订单的流转信息 |
| **Order.getPreReceive** | 语法 | public ArrayList<PreReceiveVO> getPreReceive() ; |
| 前置条件 | 在presentation层已经对输入的条形码进行判断，数据库内存在订单，需要读取收件信息录入表格的基础信息 |
| 后置条件 | 无 |
| **Order.getOrderVO** | 语法 | public ArrayList<OrderVO> getOrderVO(MyDate date); |
| 前置条件 | 在presentation层已经对输入的条形码进行判断，数据库内存在订单 |
| 后置条件 | 无 |
| **Order.getEstiDate** | 语法 | public double getEstiDate(City one,City two); |
| 前置条件 | 选择好寄件人和收件人的城市 |
| 后置条件 | 无 |
| **Order.setEstiDate** | 语法 | public ResultMessage setEstiDate(double day,City one,City two); |
| 前置条件 | 当订单的收件时间成功录入 |
| 后置条件 | 系统持久化更新城市间流转时间的估计量 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **Strategy.getConstVO()** | 获得所有常量 | |
| **Strategy.setConst(ConstVO vo)** | 根据已有的数据，修正流转中预计时间的常量 | |
|  |  | |

#### accountbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| **Account.add** | 语法 | Public ResultMessage add(AccountVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| **Account.delete** | 语法 | Public ResultMessage delete(AccountVO vo) |
| 前置条件 | 无（删除表明该数据必然是存在的） |
| 后置条件 | 系统永久性删除该账户的数据 |
| **Account.find** | 语法 | Public AccountVO find(String ID) |
| 前置条件 | 账户ID符合输入语法要求 |
| 后置条件 | 系统返回查找到的搜索结果 |
| **Account.modify** | 语法 | Public ResultMessage modify(AccountVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统修改该账户并持久化保存该账户数据 |
| **Account. show** | 语法 | public ArrayList<AccountVO> show() |
| 前置条件 | 无 |
| 后置条件 | 返回所有账户信息 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **AccountDataService.**  **add(AccountPO po)** | 根据提供的po增加持久化对象 | |
| **AccountDataService.**  **find(String ID)** | 根据ID查找持久化对象 | |
| **AccountDataService.**  **delete(AccountPO po)** | 删除单一持久化对象 | |
| **AccountDataService.init()** | 清空所有持久化对象 | |
| **AccountDataService.modify**  **(AccountPO po)** | 修改单一持久化对象 | |
| **AccountDataService.show** | 显示生成的账户信息 | |

#### userbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| **User.login** | 语法 | Public ResultMessage login(AccountVO vo) |
| 前置条件 | 账号、密码符合语法规则 |
| 后置条件 | 系统确认账号密码是否正确，若正确，进入主界面，否则提示错误信息 |
| **User.modify** | 语法 | Public ResultMessage modify(AccountVO vo) |
| 前置条件 | 原密码正确，新密码符合规范 |
| 后置条件 | 系统永久性更新个人账户信息 |
| **User.getMes** | 语法 | Public AccountVO getMes() |
| 前置条件 | 已登录 |
| 后置条件 | 返回系统目前账户信息 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **AccountDataService.**  **check(AccountPO po)** | 根据提供的Vo判断是否允许登录 | |
| **AccountDataService.modify**  **(AccountPO po)** | 根据提供的Vo修改账户信息 | |
| **AccountDataService.**  **getMes(AccountPOpo)** | 根据提供的Vo返回该账户信息 | |

#### personnelbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| **Personnel.getpeopleByInstID** | 语法 | Public ArrayList<PersonVO>  getpeopleByInst(String ID) |
| 前置条件 | 机构ID输入正确 |
| 后置条件 | 系统返回该机构下所有人员信息 |
| **Personnel.getpeopleByID** | 语法 | Public PersonVO getpeopleByID(String ID) |
| 前置条件 | 人员ID输入合法 |
| 后置条件 | 返回搜索结果 |
| **Personnel.getpeopleByName** | 语法 | Public ArrayList<PersonVO>  getpeopleByName(String name) |
| 前置条件 | 无 |
| 后置条件 | 返回有重合关键字的人员信息 |
| **Personnel.addpeople** | 语法 | Public ResultMessage addpeople(PersonVO vo) |
| 前置条件 | 人员姓名、机构ID、手机号、种类合法 |
| 后置条件 | 系统增加该持久化对象 |
| **Personnel.delpeople** | 语法 | Public ResultMessage delpeople(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统删除该持久化对象 |
| **Personnel.addInst** | 语法 | Public ResultMessage addInst(InstVO vo) |
| 前置条件 | 机构信息输入合法 |
| 后置条件 | 系统增加该机构信息 |
| **Personnel.getInst** | 语法 | Public ArrayList<InstVO> getInst(); |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有机构信息 |
| **Personnel.delnst** | 语法 | Public ResultMessage deleteInst(String ID) |
| 前置条件 | 无（是在查找的基础上删除，故一定存在） |
| 后置条件 | 系统删除该机构信息 |
| **Personnel.getPersons** | 语法 | public ArrayList<PersonVO> getPersons() |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的人员信息 |
| **Personnel.modifyInst** | 语法 | public ResultMessage modifyInst(InstVO vo) |
| 前置条件 | 机构信息输入合法 |
| 后置条件 | 系统修改该机构信息 |
| **Personnel.modifyPerson** | 语法 | public ResultMessage modifyPerson(PersonVO vo) |
| 前置条件 | 人员信息输入合法 |
| 后置条件 | 系统修改该人员信息 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **Personnelds.getpeobyInst(String ID)** | 根据机构ID返回该机构下属人员信息 | |
| **Personnelds.getpeobyID(String ID)** | 根据ID返回人员信息 | |
| **Personnelds.getpeobyName(String name)** | 根据姓名返回人员信息 | |
| **Personnelds.addpeople(PersonPo po)** | 增加人员的持久化对象 | |
| **Personnelds.delpeople(String ID)** | 删除人员的持久化对象 | |
| **Personnelds.addInst(InstPo po)** | 增加机构的持久化对象 | |
| **Personnelds.delInst(InstPo po)** | 删除机构的持久化对象 | |
| **Personnelds. getPersons ()** | 得到所有人员信息的持久化对象 | |
| **Personnelds.modifyInst (InstPo po)** | 修改机构的持久化对象 | |
| **Personnelds.modifyPerson (PersontPo po)** | 修改人员的持久化对象 | |

#### statisticbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Statisticbl.getstateForm** | 语法 | Public ArrayList<StateFormVO>  getstateForm () |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的经营状况表 |
| **Statisticbl. getIncomeForm** | 语法 | Public ArrayList<CostIncomeVO>  getIncomeForm() |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的成本收益表 |
| **Statisticbl.bulidStateForm** | 语法 | Public ResultMessage  bulidStateForm(StateFormVO vo) |
| 前置条件 | 经营状况表关键信息齐全且合法 |
| 后置条件 | 系统生成并储存该表 |
| **Statisticbl.bulidCostIncomeForm** | 语法 | Public ResultMessage bulidCostIncomeForm  (CostIncomeVO vo) |
| 前置条件 | 成本收益表关键信息齐全且合法 |
| 后置条件 | 系统生成并存储该表 |
| **Statisticbl.bulidbill** | 语法 | Public ResultMessage bulidBill(BillVO vo) |
| 前置条件 | 账单具有完整的信息 |
| 后置条件 | 系统永久储存 |
| **Statisticbl.getbills** | 语法 | Public ArrayList< BillVO > getIBills() |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的账单 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **statsiticds.getstateForm ()** | 返回所有经营状况表 | |
| **statsiticds.getCostIncomeFForm ()** | 返回所有的成本收益表 | |
| **Statisticds.bulidStateForm(StateFormVO vo)** | 永久储存经营状况表 | |
| **Statisticds.bulidCostIncomeForm(CostIncomeFormPO po)** | 永久储存成本收益表 | |
| **Statisticds.bulidbill** | 永久储存账单 | |
| **Statisticds.getbill** | 获得所有账单 | |

#### approvalbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **approval. getbills** | | 语法 | Public ArrayList<DocVO> getbills(DocType type) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| **Approval.approveone** | | 语法 | Public ResultMessage approveone(DocVO vo) |
| 前置条件 | 该单据未审批过 |
| 后置条件 | 将该单据状态改为审批通过 |
| **Approval.approvemany** | | 语法 | Public ResultMessage approvemany(Arraylist< DocVO > vo) |
| 前置条件 | 选中单据均未审批过 |
| 后置条件 | 将该单据状态改为审批通过 |
| **需要的接口（需接口）** | | | |
| **服务名** | 服务内容 | | |
| **Transport.getDocLists (DocType type)** | 从货物流转包中获取各种单据 | | |
| **Order. getDocLists ()** | 获取订单 | | |
| **Store.getDocLists(DocType type)** | 获取库存相关的单据 | | |
| **Transport. changeDocsState (DocType type)** | 改变流转过程中单据的状态 | | |
| **Order. changeDocsState(DocType type)** | 改变订单的状态 | | |
| **Store.changeDocsState (DocType type)** | 改变库存单据的状态 | | |
| **Transport. changeOneDocState (DocType type)** | 改变流转过程中单个单据的状态 | | |
| **Order. changeOneDocState ()** | 改变单个订单的状态 | | |
| **Store. changeOneDocState (DocType type)** | 改变库存单个单据的状态 | | |

#### Strategy模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Strategy.getConst** | 语法 | Public ConstVO getConst(String const) |
| 前置条件 | 无 |
| 后置条件 | 返回所有常量 |
| **Strategy.setConst** | 语法 | Public ResultMessage setConst(ConstVO vo) |
| 前置条件 | 常量符合语法规则 |
| 后置条件 | 系统记录新常量 |
| **Strategy.getsalary** | 语法 | Public ArrayList<SalaryWayVO> getsalary() |
| 前置条件 | 无 |
| 后置条件 | 返回所有薪水策略 |
| **Strategy.getOnesalary** | 语法 | Public SalaryWayVO getOnesalary(StaffType  type) |
| 前置条件 | 无 |
| 后置条件 | 返回该职员的薪水策略 |
| **Strategy.setsalary** | 语法 | Public ResultMessage setSalary(SalaryWayVO  way) |
| 前置条件 | 薪水策略中的值符合语法规则 |
| 后置条件 | 永久修改薪水策略 |
| **Strategy. getEstiDateVO** | 语法 | public EstiDateVO getEstiDateVO() |
| 前置条件 | 无 |
| 后置条件 | 返回货物预计到达的时间 |
| **Strategy. setEstiDateVO** | 语法 | public ResultMessage setEstiDateVO(EstiDateVO vo) |
| 前置条件 | 预计时间的值符合规范 |
| 后置条件 | 修改预计到达的时间 |
| **需要的接口（需接口）** | | |
| **服务名** | 服务内容 | |
| **Strategyds. getConst** | 从数据库中读取常量 | |
| **Strategyds.setConst** | 设置常量 | |
| **Strategyds.getsalary** | 从数据库中获得所有职员的薪资策略 | |
| **Strategyds.getOnesalary** | 从数据库中获得某一职员的薪资策略 | |
| **Strategyds.setsalary** | 设置某一职员的薪资策略 | |
| **Strategyds. getEstiDateVO** | 得到货物的预计到达时间 | |
| **Strategyds. setEstiDateVO** | 设置货物的预计到达时间 | |

#### financebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Finance. createPayVO** | 语法 | public ResultMessage create(PayVO vo) |
| 前置条件 | 付款信息符合规范 |
| 后置条件 | 系统生成付款信息 |
| **Finance. showCosts** | 语法 | public ArrayList<? extends CostVO> showCosts(CostType type) |
| 前置条件 | 无 |
| 后置条件 | 系统显示成本信息 |
| **Finance. addCostVO** | 语法 | public ResultMessage add(CostVO vo) |
| 前置条件 | 成本信息符合编写规范 |
| 后置条件 | 系统增加一条成本信息 |
| **Finance. modifyCostVO** | 语法 | public ResultMessage modify(CostVO vo) |
| 前置条件 | 成本信息符合编写规范 |
| 后置条件 | 系统更新成本信息 |
| **Finance. delCostVO** | 语法 | public ResultMessage del(CostVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统删除成本信息 |
| **Finance. showPays** | 语法 | public ArrayList<PayVO> showPays() |
| 前置条件 | 无 |
| 后置条件 | 系统显示付款信息 |
| **Finance. createDeposit** | 语法 | public ResultMessage createDeposit(DepositVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统显示收款信息 |
| **Finance. showDeposit** | 语法 | public ArrayList<DepositVO> showDeposit() |
| 前置条件 | 无 |
| 后置条件 | 系统显示收益信息 |
| **Finance. getAccounts** | 语法 | public ArrayList<BankAccountVO> getAccounts() |
| 前置条件 | 无 |
| 后置条件 | 显示银行账户信息 |
| **Finance. deleteAccount** | 语法 | public ResultMessage deleteAccount(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统删除银行账户信息 |
| **Finance. addAccount** | 语法 | Public  ResultMessage addAccount(BankAccountVO vo) |
| 前置条件 | 银行账户信息符合规范 |
| 后置条件 | 系统增加银行账户信息 |
| **Finance. checkAccount** | 语法 | public ResultMessage checkAccount(String ID, int money) |
| 前置条件 | 无 |
| 后置条件 | 检查银行账户信息 |
| **Finance. modifyAccount** | 语法 | public ResultMessage modifyAccount(BankAccountVO vo) |
| 前置条件 | 银行账户信息符合规范 |
| 后置条件 | 系统修改银行账户信息 |
| **Finance.getMessage** | 语法 | Public ProfitVO getMessage() |
| 前置条件 | 无 |
| 后置条件 | 提供利润相关信息 |
| **需要的服务（需接口）** | | |
| **服务名** | 服务 | |
| **FinanceDataService.showCost()** | 得到cost对象的持续引用 | |
| **FinanceDataService. showPays ()** | 得到PayPO对象的持续引用 | |
| **FinanceDataService. createDeposit (DepositPO po)** | 增加DepositPO持久化对象 | |
| **FinanceDataService. modifyCost(CostPO po)** | 修改CostPO的持久化对象 | |
| **FinanceDataService. addCostPO(CostPO po)** | 增加CostPO的持久化对象 | |
| **FinanceDataService. delCostPO (String ID)** | 删除CostPO持久化对象 | |
| **FinanceDataService .createPay(PayPO po)** | 增加PayPO的持久化对象 | |
| **FinanceDataService.showDeposit ()** | 得到DepositPO对象的持续引用 | |
| **FinanceDataService.getAccounts ()** | 得到BankAccountPO对象的持续引用 | |
| **FinanceDataService.deleteAccount (String ID)** | 删除BankAccountPO持久化对象 | |
| **FinanceDataService.checkAccount (String ID, int money)** | 查找BankAccountPO持久化对象 | |
| **FinanceDataService.modifyAccount (BankAccounPO po)** | 修改BankAccountPO持久化对象 | |
| **Strategy.getSalary()** | 获得所有的薪水策略 | |
| **Strategy.getOneSalary()** | 获得某一职员的薪水策略 | |
| **Strategy.setSalary()** | 设置某一职员的薪水策略 | |
| **Transport.getExpense()** | 得到运费成本信息 | |
| **Order.getOrderVO()** | 得到订单信息 | |
| **Personnel.getPerMes()** | 得到人员信息 | |

#### storebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Store.showInStoreDoc** | 语法 | Public Arraylist<InStoreDocVO> show() |
| 前置条件 | 无 |
| 后置条件 | 系统返回入库单信息 |
| **Store. generateInStoreDoc** | 语法 | Public ResultMessage generate  (InStoreDocVO vo) |
| 前置条件 | 信息符合输入规则 |
| 后置条件 | 系统生成入库单 |
| **Store.showOutStoreDoc** | 语法 | Public Arraylist<OutStoreDocVO> show() |
| 前置条件 | 无 |
| 后置条件 | 系统返回出库单信息 |
| **Store. generateOutStoreDoc** | 语法 | Public ResultMessage generate  (OutStoreDocVO vo) |
| 前置条件 | 信息符合输入规则 |
| 后置条件 | 系统生成出库单 |
| **Store.showMessage** | 语法 | Public Arraylist<StoreMessageVO> show() |
| 前置条件 | 无 |
| 后置条件 | 系统返回库存信息 |
| **Store.showCheck** | 语法 | Public Arraylist<StoreCheckVO> showCheck() |
| 前置条件 | 无 |
| 后置条件 | 系统返回库存快照 |
| **Store.exportExcel** | 语法 | public ResultMessage exportExcel  (String path) |
| 前置条件 | 地址输入符合规定 |
| 后置条件 | 系统导出当天库存快照的Excel表 |
| **Store.update** | 语法 | public ResultMessage update  (StoreMessageVO vo) |
| 前置条件 | 库存信息发生更改 |
| 后置条件 | 系统更新库存信息 |
| **Store.setAlarmValue** | 语法 | public Boolean setAlarmValue  (String value) |
| 前置条件 | 无 |
| 后置条件 | 系统更新库存报警信息 |
| **Store.getAlarmValue** | 语法 | public Boolean getAlarmValue  (City city) |
| 前置条件 | 无 |
| 后置条件 | 得到该城市的库存警戒值 |
| **Store. getDocLists** | 语法 | public ArrayList<? extends DocVO> getDocLists(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到对应单据的单据列表 |
| **Store. changeDocsState** | 语法 | Public ResultMessage changeDocsState(ArrayList<String> docsID, DocType type, DocState state) |
| 前置条件 | 总经理审批单据符合规范 |
| 后置条件 | 系统改变相应单据的状态 |
| **Store. changeOneDocState** | 语法 | public ResultMessage changeOneDocState(String docID, DocType type, DocState state) |
| 前置条件 | 总经理审批单据符合规范 |
| 后置条件 | 系统改变相应单据的状态 |
| **Store. getDayDocCount** | 语法 | public int getDayDocCount(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到相应单据当天生成的数量 |
| **Store. getByID** | 语法 | public DocVO getByID(String ID , DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到单据的详细信息 |
| 需要的服务（需接口） | | |
| **服务名** | 服务 | |
| **StoreDataService.getStoreMessage()** | 得到storeMessage对象的引用 | |
| **StoreDataService.addIn(InStoreDocPO po)** | 增加InStoreDocPO持久化对象 | |
| **StoreDataService.addOut(OutStoreDocPO po)** | 增加OutStoreDocPO持久化对象 | |
| **StoreDataService.getIn()** | 得到InStoreDocPO对象的引用 | |
| **StoreDataService.getOut()** | 得到OutStoreDocPO对象的引用 | |
| **StoreDataService.update(StoreMessengePO po)** | 更新库存信息 | |
| **StoreDataService.setAlarmValue(String value)** | 更新库存警戒值 | |
| **StoreDataService.storeCheck(StoreCheckPO po)** | 储存库存快照PO | |
| **StoreDataService.getCheck()** | 得到库存库存快照PO | |
| **Order.addDocToList(DocVO vo)** | 更新订单的信息 | |
| **StoreDataService.getAlarmValue(City city)** | 得到该城市的库存警戒值 | |
| **StoreDataService.changeDocsState(ArrayList<String> docsID,DocType type, DocState state)** | 改变单据的审批状态 | |
| **StoreDataService.changeOneDocState(String docID, DocType type,DocState state)** | 改变单据的审批状态 | |
| **StoreDataService.getDayDocCount (DocType type)** | 得到相应单据当天生成的数量 | |
| **StoreDataService.getByID (String ID , DocType type)** | 得到单据的详细信息 | |
| **StoreDataService. getDocLists (DocType type)** | 得到单据的详细信息 | |

#### DTmanagebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **DTManage.AddDriver** | 语法 | Public ResultMessage addDriver  (DriverVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加该司机信息并持久化更新涉及的对象的数据 |
| **DTManage.CheckDriverbyName** | 语法 | Public ArrayList< DriverVO >  CheckDriverByName(String name) |
| 前置条件 | 无 |
| 后置条件 | 系统返回有重合关键字的司机信息 |
| **DTManage.CheckDriverbyID** | 语法 | Public DriverVO CheckDriverByID  (String ID) |
| 前置条件 | 输入的ID符合规范 |
| 后置条件 | 系统返回相应ID对应的司机信息 |
| **DTManage.CheckDriverbyInst** | 语法 | Public ArrayList< DriverVO > CheckDriverByInst  (String InstID) |
| 前置条件 | 输入的机构ID符合规范 |
| 后置条件 | 系统返回对应机构名下的司机信息 |
| **DTManage.ModifyDriver** | 语法 | Public ResultMessage modifyDriver  (DriverVO vo) |
| 前置条件 | 输入的信息格式符合规范并且存在于现有数据中 |
| 后置条件 | 系统持久化修改并更新涉及的对象的数据 |
| **DTManage.delDriver** | 语法 | Public ResultMessage delDriver (DriverVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化删除该对象的数据 |
| **DTManage.getDriverName** | 语法 | Public ArrayList<DriverList> getDriverName (String InstID) |
| 前置条件 | 生成一个装车单 |
| 后置条件 | 无 |
| **DTManage.AddCar** | 语法 | Public ResultMessage addCar  (CarVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加该车辆信息并持久化更新涉及的对象的数据 |
| **DTManage.CheckCarbyID** | 语法 | Public CarVO CheckByID(String ID) |
| 前置条件 | 输入的ID符合规范 |
| 后置条件 | 系统返回对应ID下的车辆信息 |
| **DTManage.CheckbyPlateNum** | 语法 | Public CarVO CheckByPlateNum  (String PlateNum) |
| 前置条件 | 输入的车牌号符合规范 |
| 后置条件 | 系统返回车牌号对应的车辆信息 |
| **DTManage.ModifyCar** | 语法 | Public ResultMessage modifyCar  (CarVO vo) |
| 前置条件 | 输入的信息格式符合规范并且存在于现有数据中 |
| 后置条件 | 系统持久化更新涉及的对象的数据 |
| **DTManage.DelCar** | 语法 | Public ResultMessage DelCar (CarVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化删除该车辆信息 |
| **DTManage.getPlateNum** | 语法 | Public ArrayList<PlateNumber>  getPlateNumber(String InstID) |
| 前置条件 | 生成一个装车单 |
| 后置条件 | 无 |
| **DTManage.getAllCars** | 语法 | public ArrayList<CarVO> getAllCars() |
| 前置条件 | 无 |
| 后置条件 | 得到所有的车辆信息 |
| **DTManage. getAllDrivers** | 语法 | public ArrayList<DriverVO> getAllDrivers() |
| 前置条件 | 无 |
| 后置条件 | 得到所有的司机信息 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **服务名** | 服务 |
| **DTmanageDataService.**  **getDriverMesPO()** | 得到DriverMes对象的持续引用 |
| **DTmanageDataService.**  **getCarMesPO()** | 得到CarMes对象的持续引用 |
| **DTmanageDataService.**  **addDriver (DriverPO po)** | 添加Driver单一化持久对象 |
| **DTmanageDataService.**  **addCar (CarPO po)** | 添加Car单一化持久对象 |
| **DTmanageDataService.**  **modifyDriver (DriverPO po)** | 更改Driver单一化持久对象 |
| **DTmanageDataService.**  **modifyCar (CarPO po)** | 更改Car单一化持久对象 |
| **DTmanageDataService.**  **delDriver (DriverPO po)** | 删除Driver单一化持久对象 |
| **DTmanageDataService.**  **delCar (CarPO po)** | 删除Car单一化持久对象 |
| **DTmanageDataService.**  **getAllCars ()** | 得到所有的车辆信息 |
| **DTmanageDataService.**  **getAllDrivers ()** | 得到所有的司机信息 |

#### Transport模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Transport.AddLoadDoc** | 语法 | Public ResultMessage.add  (LoadDocVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加装车单并持久化更新涉及的对象的数据 |
| **Transport. getDayLoadDocs** | 语法 | Public ArrayList< LoadDocVO >  getDayLoadDocs(Date date) |
| 前置条件 | 日期符合输入规范 |
| 后置条件 | 系统返回当日所有装车单 |
| **Transport.AddSendGoodDoc** | 语法 | Public ResultMessage add  (SendGoodDocVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加派件单并持久化更新涉及的对象的数据 |
| **Transport.getDay SendGoodDocs** | 语法 | Public ArrayList< GoodDocVO >  getDaySendDocs(Date date) |
| 前置条件 | 日期符合输入规范 |
| 后置条件 | 系统返回当日所有派件单信息 |
| **Transport.AddArriveYYDoc** | 语法 | Public ResultMessage.add  (ArriveYYDocVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加接收单并持久化更新涉及的对象的数据 |
| **Transport.getDayArriveYYDocs** | 语法 | Public ArrayList<ArriveYYDocVO>  getDayArriveYYDocs(Date date); |
| 前置条件 | 日期符合输入规范 |
| 后置条件 | 系统返回当日所有接收单信息 |
| **Transport.AddArriveZZDoc** | 语法 | Public ArriveZZDoc.add  (ArriveZZDocVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加到达单并持久化更新涉及的对象的数据 |
| **Transport.getDayArriveZZDocs** | 语法 | Public ArrayList<ArriveZZDocVO>  getDayArriveZZDocs(Date date); |
| 前置条件 | 日期符合输入规范 |
| 后置条件 | 系统返回当日所有到达单信息 |
| **Transport.AddTransferDoc** | 语法 | Public TransferDoc.add  (TransferDocVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加中转单并持久化更新涉及的对象的数据 |
| **Transport.getDayTransferDocs** | 语法 | Public ArrayList<TransferDocVO>  getDayTransferDocs(Date date); |
| 前置条件 | 日期符合输入规范 |
| 后置条件 | 系统返回当日所有中转单信息 |
| **Transport.getDoc** | 语法 | Public ArrayList<DocVO >  getDoc(DocType type) |
| 前置条件 | 该类型单据已经生成 |
| 后置条件 | 系统从货物流转包中获得该单据 |
| **Transport.getExpense** | 语法 | Public double getExpense(DocType type) |
| 前置条件 | 该单据ID已经生成 |
| 后置条件 | 系统自动得到该单据的运费成本信息 |
| **Transport. getDocLists** | 语法 | public ArrayList<? extends DocVO> getDocLists(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到对应单据的单据列表 |
| **Transport. changeDocsState** | 语法 | Public ResultMessage changeDocsState(ArrayList<String> docsID, DocType type, DocState state) |
| 前置条件 | 总经理审批单据符合规范 |
| 后置条件 | 系统改变相应单据的状态 |
| **Transport.changeOneDocState** | 语法 | public ResultMessage changeOneDocState(String docID, DocType type, DocState state) |
| 前置条件 | 总经理审批单据符合规范 |
| 后置条件 | 系统改变相应单据的状态 |
| **Transport. getDayDocCount** | 语法 | public int getDayDocCount(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到相应单据当天生成的数量 |
| **Transport. getByID** | 语法 | public DocVO getByID(String ID , DocType type) |
| 前置条件 | 无 |
| 后置条件 | 得到单据的详细信息 |
| **Transport. addOnePay** | 语法 | public ResultMessage addOnePay(PayDocVO vo) |
| 前置条件 | 生成的的付款单信息符合规范 |
| 后置条件 | 增加一个付款单 |
| **Transport. getPays** | 语法 | public ArrayList<PayDocVO> getPays() |
| 前置条件 | 无 |
| 后置条件 | 获得当日所有付款单 |
| 需要的服务（需接口） | | |
| **服务名** | 服务 | |
| **TransportService.**  **add(LoadDocPO po)** | 添加装车单单一化持久对象 | |
| **TransportService.**  **add(SendGoodDocPO po)** | 添加派送单化持久对象 | |
| **TransportService.**  **add(ArriveYYDocPO po)** | 添加营业厅到达单单一化持久对象 | |
| **TransportService.**  **add(ArriveZZDocPO po)** | 添加中转中心到达单单一化持久对象 | |
| **TransportService.**  **add(TransferDocPO po)** | 添加中转单单一化持久对象 | |
| **TransportService.**  **getLoadDocPO(String id)** | 得到装车单单一化持久对象 | |
| **TransportService.**  **getSendGoodDocPO (String id)** | 得到派送单化持久对象 | |
| **TransportService.**  **getArriveYYDocPO (String id)** | 得到营业厅到达单单一化持久对象 | |
| **TransportService.**  **geArriveZZDocPO (String id)** | 得到中转中心到达单单一化持久对象 | |
| **TransportService.**  **getTransferDocPO (String id)** | 得到中转单单一化持久对象 | |
| **Order.addDocToList(DocVO vo)** | 更新订单的信息 | |
| **Strategy.getConst()** | 获得生成订单所需要的常量 | |
| **TransportService.changeDocsState(ArrayList<String> docsID,DocType type, DocState state)** | 改变单据的审批状态 | |
| **TransportService.changeOneDocState(String docID, DocType type,DocState state)** | 改变单据的审批状态 | |
| **TransportService.getDayDocCount (DocType type)** | 得到相应单据当天生成的数量 | |
| **TransportService.getByID (String ID , DocType type)** | 得到单据的详细信息 | |
| **TransportService. getDocLists (DocType type)** | 得到单据的详细信息 | |

## 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。例如，AccountBusinessService的服务主要有AccountDataService提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。以AccountDataService为例，数据层模块的具体描述下图所示：

**图11 数据层模块的描述**

### 数据层模块的职责

数据层模块职责如下表所示

**数据层模块职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| Accountdataservice | 负责保存用户账户的信息，进行增删改查的操作 |
| StatisticDataService | 负责保存经营情况表、成本收益表，供读取修改写入 |
| PersonnelDataservice | 负责保存人员机构的信息供读取和修改 |
| StretegyDataService | 负责保存常量、薪水策略，供读取修改写入 |
| financedataservice | 负责成本和收益信息 |
| storedataservice | 负责库存和出入库信息 |
| DTmanageDataService | 负责保存司机和车辆信息供读取、修改和删除 |
| TransportDataService | 负责保存流转中的单据信息供读取 |

### 数据层模块的接口规范

数据层模块的接口规范如下表

#### DataFactory

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **DataFactory.getAccountDataService** | 语法 | Public AccountDataService getAccountDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回AccountDataService对象 |
| **DataFactory.getDTManageDataService** | 语法 | public DTManagedataservice getDTMangeDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回DTManageDataService对象 |
| **DataFactory.getFinanceDataService** | 语法 | public FinanceDataService getFinanceDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回FinanceDataService对象 |
| **DataFactory.getOrderDataService** | 语法 | public OrderDataService  getOrderDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回OrderDataService对象 |
| **DataFactory.getPersonnelDataService** | 语法 | public PersonnelDataService getPersonnelDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回OrderDataService对象 |
| **DataFactory.getStatisticDataService** | 语法 | public StatisticDataService getStatisticDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回StatisticDataService对象 |
| **DataFactory.getStoreDataService** | 语法 | public StoreDataService  getStoreDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回StoreDataService对象 |
| **DataFactory.getStrategyDataService** | 语法 | public StrategyDataService getStrategyDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回StrategyDataService对象 |
| **DataFactory.getTransportDataService** | 语法 | public Transportdataservice getTransportDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回TransportDataService对象 |

#### personneldataService

|  |  |  |
| --- | --- | --- |
| **提供的服务** | | |
| **PersonnelDataService.init** | 语法 | Public void initial() |
| **PersonnelDataService.getpeobyInst** | 语法 | public ArrayList<PersonPO>  getPeoByInst(String ID) |
| 前置条件 | 系统中有该机构 |
| 后置条件 | 系统返回该机构下所有人员信息 |
| **PersonnelDataService.getpeobyID** | 语法 | public PersonPO getPersonByID(String ID) |
| 前置条件 | 无 |
| 后置条件 | 返回该ID查找结果 |
| **PersonnelDataService.getpeobyName** | 语法 | public ArrayList<PersonPO>  getPeoByName(String name) |
| 前置条件 | 无 |
| 后置条件 | 返回该姓名查找结果 |
| **PersonnelDataService.addpeople** | 语法 | public ResultMessage addPerson(PersonPO po) |
| 前置条件 | 人员信息合法 |
| 后置条件 | 系统增加该人员信息 |
| **PersonnelDataService.deletepeople** | 语法 | public ResultMessage delPerson(String ID) |
| 前置条件 | 该账户存在在数据库中 |
| 后置条件 | 修改账户数据 |
| **PersonnelDataService.addInst** | 语法 | public ResultMessage addInst(InstPO po) |
| 前置条件 | 机构信息合法 |
| 后置条件 | 系统增加该机构 |
| **PersonnelDataService.delInst** | 语法 | public ResultMessage delInst(String ID) |
| 前置条件 | 该账户存在在数据库中 |
| 后置条件 | 系统删除该账户信息 |
| **PersonnelDataService.getInst** | 语法 | public ArrayList<InstPO> getInst() |
| 前置条件 | 无 |
| 后置条件 | 返回所有机构信息 |
| **PersonnelDataService.modifyInst** | 语法 | public ResultMessage modifyInst(InstPO po) |
| 前置条件 | 无 |
| 后置条件 | 修改所有机构信息 |
| **PersonnelDataService.getPersons** | 语法 | public ArrayList<PersonPO> getPersons() |
| 前置条件 | 无 |
| 后置条件 | 返回所有人员信息 |
| **PersonnelDataService.modifyPerson** | 语法 | public ResultMessage  modifyPerson(PersonPO po) |
| 前置条件 | 无 |
| 后置条件 | 修改所有机人员信息 |

#### AccountdataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| **AccountDataService.**  **add(AccountPO po)** | 语法 | public ResultMessage  add(AccountPO po) |
| 前置条件 | 同样的账户在数据库中不存在 |
| 后置条件 | 系统永久增加该账户数据 |
| **AccountDataService.init()** | 语法 | public void initial() |
| 前置条件 | 无 |
| 后置条件 | 清空所有账户数据 |
| **AccountDataService.**  **find(String ID)** | 语法 | public AccountPO  find(String ID) |
| 前置条件 | ID符合语法规则 |
| 后置条件 | 返回该账户信息，若找不到返回null |
| **AccountDataService.**  **delete(AccountPO po)** | 语法 | public ResultMessage  delete(String ID) |
| 前置条件 | 该账户存在在数据库中 |
| 后置条件 | 系统删除该账户信息 |
| **AccountDataService.modify**  **(AccountPO po)** | 语法 | public ResultMessage  modify(AccountPO po) |
| 前置条件 | 该账户存在在数据库中 |
| 后置条件 | 修改账户数据 |
| **AccountDataService.**  **check(AccountPO po)** | 语法 | public AccountPO check(String ID, String password) |
| 前置条件 | 账号、密码合法 |
| 后置条件 | 系统检查是否存在账户账户密码是否对应 |
| **AccountDataService.**  **getMes(String ID)** | 语法 | public AccountPO  getMes(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回该账户信息 |

#### StrategydataService

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **StrategyDataService.Init** | 语法 | Public void initial() |
| **StrategyDataService.getConst** | 语法 | Public ConstPO getConst() |
| 前置条件 | 无 |
| 后置条件 | 系统永久增加该账户数据 |
| **StrategyDataService.setConst** | 语法 | public ResultMessage  setConst(ConstPO po) |
| 前置条件 | ConstPO数据合法且完整 |
| 后置条件 | 系统永久保存新的常量数据 |
| **StrategyDataService.getSalary** | 语法 | Public ArrayList<SalaryWayPO>  getSalary() |
| 前置条件 | 无 |
| 后置条件 | 返回所有工资策略 |
| **StrategyDataService.setSalary** | 语法 | public ResultMessage  setSalaryWay(SalaryWayPO po) |
| 前置条件 | 工资策略合法 |
| 后置条件 | 设置新的策略 |
| **StrategyDataService.getoneSalary** | 语法 | public SalaryWayPO  getOneSalary(StaffType type) |
| 前置条件 | 无 |
| 后置条件 | 返回该职员的常量数据 |
| **StrategyDataService. getEstiDatePO** | 语法 | public EstiDatePO getEstiDatePO() |
| 前置条件 | 账单数据合法、完整 |
| 后置条件 | 获得城市估计时间 |
| **StrategyDataService. setEstiDatePO** | 语法 | public ResultMessage  setEstiDatePO(EstiDatePO po) |
| 前置条件 | 无 |
| 后置条件 | 设置城市估计时间 |

#### StatisticDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **StatisticDataService.Init** | 语法 | Public void initial() |
| **StatisticDataService.getStateForm** | 语法 | Public ArrayList< StateFormPO >  getStateForm()f |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有经营状况表 |
| **StatisticDataService**  **getCostIncomeForm** | 语法 | Public ArrayList<CostIncomePO>  getCostIncomeForm() |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有成本收益表 |
| **StatisticDataService.**  **bulidStateForm** | 语法 | Public ResultMessage  bulidStateForm(StateForm po) |
| 前置条件 | 经营状况表信息合法齐全 |
| 后置条件 | 系统保存新的经营状况表 |
| **StatisticDataService.bulidCostIncomeForm** | 语法 | Public ResultMessage  CostIncomeForm(CostIncomeForm po) |
| 前置条件 | 成本收益表信息合法齐全 |
| 后置条件 | 系统保存新的成本收益表 |
| **StatisticDataService.bulidBill** | 语法 | public ResultMessage  bulidBill(BillPO po); |
| 前置条件 | 账单信息齐全规范 |
| 后置条件 | 系统永久建立该账单 |
| **StatisticDataService.getBills** | 语法 | public ArrayList<BillPO> getBills(); |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有期初账单 |

#### FinanceDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务** | | |
| **FinanceDataService.init** | 语法 | public void initial() |
| **FinanceDataService.getDepositPO** | 语法 | Public ArrayList<DepositPO>  getDepositPO() |
| 前置条件 | 无 |
| 后置条件 | 系统返回收款信息 |
| **FinanceDataService.getPayPO** | 语法 | Public ArrayList<PayPO> getPayPO() |
| 前置条件 | 无 |
| 后置条件 | 系统返回付款信息 |
| **FinanceDataService.addDeposit** | 语法 | Public ResultMessage  addDeposit(DepositPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统增加一条收款信息 |
| **FinanceDataService. getCostPO** | 语法 | Public ArrayList<CostPO>  getCostPO() |
| 前置条件 | 无 |
| 后置条件 | 系统返回成本信息 |
| **FinanceDataService. addCostPO** | 语法 | public ResultMessage  add(CostPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统增加一条成本信息 |
| **FinanceDataService. modifyCostPO** | 语法 | public ResultMessage  modify(CostPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统修改一条成本信息 |
| **FinanceDataService. delCostPO** | 语法 | public ResultMessage  del(String ID , CostType type) |
| 前置条件 | 无 |
| 后置条件 | 系统删除一条成本信息 |
| **FinanceDataService.addPay** | 语法 | Public ResultMessage addPay(PayPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统增加一条付款信息 |
| **FinanceDataService.getAccounts** | 语法 | public ResultMessage  modifyAccount(BankAccountPO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统获得一条账户信息 |
| **FinanceDataService.deleteAccount** | 语法 | public ResultMessage  deleteAccount(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统删除一条账户信息 |
| **FinanceDataService.addAccount** | 语法 | public ResultMessage  addAccount(BankAccountPO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统增加一条账户信息 |
| **FinanceDataService.checkAccount** | 语法 | public ResultMessage  checkAccount(String iD, int money) |
| 前置条件 | 无 |
| 后置条件 | 系统返回一条账户信息 |
| **FinanceDataService.getCost** | 语法 | public ArrayList<? extends CostPO>  show(CostType type) |
| 前置条件 | 无 |
| 后置条件 | 获得成本信息 |

#### StoreDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **StoreDataService.init** | 语法 | Public void initial() |
| **StoreDataService.getStoreMessage** | 语法 | Public StoreMessagePO getStoreMessage() |
| 前置条件 | 无 |
| 后置条件 | 系统返回库存信息PO |
| **StoreDataService.addIn** | 语法 | Public ResultMessage addIn(InStoreDocPO po) |
| 前置条件 | 增加入库单信息 |
| 后置条件 | 系统更新数据 |
| **StoreDataService.addOut** | 语法 | Public ResultMessage  addOut(OutStoreDocPO po) |
| 前置条件 | 增加出库单信息 |
| 后置条件 | 系统更新数据 |
| **StoreDataService.getIn** | 语法 | Public ArrayList<InStoreDocPO>  getIn() |
| 前置条件 | 无 |
| 后置条件 | 系统返回入库单PO |
| **StoreDataService.getOut** | 语法 | Public ArrayList<InStoreDocPO>  getOut() |
| 前置条件 | 无 |
| 后置条件 | 系统返回出库单PO |
| **StoreDataService.update** | 语法 | Public ResultMessage  update(StoreMessagePO po) |
| 前置条件 | 库存信息发生变动 |
| 后置条件 | 系统更新库存信息 |
| **StoreDataService.setalarmValue** | 语法 | public ResultMessage  setAlarmValue(String value, City city) |
| 前置条件 | 库存警戒值发生变动 |
| 后置条件 | 系统更新库存警戒信息 |
| **StoreDataService.getalarmValue** | 语法 | public String getAlarmValue(City city) |
| 前置条件 | 无 |
| 后置条件 | 获得该城市库存警戒值 |
| **StoreDataService.storeCheck** | 语法 | Public ResultMessage  storeCheck(StoreCheckPO po) |
| 前置条件 | 增加库存快照信息 |
| 后置条件 | 系统更新库存快照信息 |
| **StoreDataService.getCheck** | 语法 | Public ArrayList<StoreCheckPO> getCheck() |
| 前置条件 | 无 |
| 后置条件 | 系统返回快照信息 |
| **StoreDataService.getDocLists** | 语法 | public ArrayList<? extends DocPO>  getDocLists(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 返回单据列表 |
| **StoreDataService.changeDocsState** | 语法 | public ResultMessage  changeDocsState(ArrayList<String> docsID,  DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变所有单据状态 |
| **StoreDataService.changeOneDocState** | 语法 | public ResultMessage  changeOneDocState (String docID,DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变一个单据状态 |
| **StoreDataService.getDayDocCount** | 语法 | public int getDayDocCount(DocType type, MyDate date) |
| 前置条件 | 无 |
| 后置条件 | 获得当日该类型单据数量 |
| **StoreDataService.getStoreMessage** | 语法 | public ArrayList<StoreMessagePO>  getStoreMessages() |
| 前置条件 | 无 |
| 后置条件 | 获得库存信息 |
| **StoreDataService.getInstoreDocs** | 语法 | private ArrayList<InStoreDocPO>  getInstoreDocs(String[] instores) |
| 前置条件 | 无 |
| 后置条件 | 获得入库单 |
| **StoreDataService.getOutStoreDocs** | 语法 | private ArrayList<OutStoreDocPO>  getOutstoreDocs(String[] outstores) |
| 前置条件 | 无 |
| 后置条件 | 获得出库单 |
| **StoreDataService.getOneInstoreDoc** | 语法 | public InStoreDocPO  getOneInstoreDoc(String ID) |
| 前置条件 | 无 |
| 后置条件 | 获得一个入库单 |
| **StoreDataService.getOneOutstoreDoc** | 语法 | public OutStoreDocPO  getOneOutStoreDoc(String ID) |
| 前置条件 | 无 |
| 后置条件 | 获得一个出库单 |

#### DTManageDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **DTManageDataService.init()** | 语法 | public void initial() |
| **DTManageDataSercvice.getDriverMes** | 语法 | public DriverPO getDriverMes(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回司机信息PO |
| **DTManageDataSercvice.getCarMes** | 语法 | public CarPO getCarMes(String ID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回车辆信息PO |
| **DTManageDataSercvice.addDriverPO** | 语法 | public ResultMessage  addDriverPO(DriverPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个Driverpo记录 |
| **DTManageDataSercvice.addCarPO** | 语法 | public ResultMessage  addCarPO(CarPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个Carpo记录 |
| **DTManageDataSercvice.deleteDriverPO** | 语法 | public ResultMessage  delDriverPO(String ID) |
| 前置条件 | 数据库中存在一个相同ID的PO |
| 后置条件 | 删除一个DriverPO |
| **DTManageDataSercvice.deleteCarPO** | 语法 | public ResultMessage  delCarPO(String ID) |
| 前置条件 | 数据库中存在一个相同ID的PO |
| 后置条件 | 删除一个CarPO |
| **DTManageDataSercvice.updateDriverPO** | 语法 | public ResultMessage  updateDriverPo(DriverPO po) |
| 前置条件 | 数据库中存在一个相同ID的PO |
| 后置条件 | 更新一个DriverPO |
| **DTManageDataSercvice.updateCarPO** | 语法 | public ResultMessage updateCarPo(CarPO po) |
| 前置条件 | 数据库中存在一个相同ID的PO |
| 后置条件 | 更新一个CarPO |
| **DTManageDataSercvice.getDriverByName** | 语法 | public ArrayList<DriverPO>  getDriverByName(String name) |
| 前置条件 | 无 |
| 后置条件 | 返回一个DriverPO |
| **DTManageDataSercvice.getDriverByInst** | 语法 | public ArrayList<DriverPO>  getDriverByInst(String inst) |
| 前置条件 | 无 |
| 后置条件 | 返回一个DriverPO |
| **DTManageDataSercvice.getAllCars** | 语法 | public ArrayList<CarPO>  getAllCars() |
| 前置条件 | 无 |
| 后置条件 | 返回所有CarPO |
| **DTManageDataSercvice.getAllDrivers** | 语法 | public ArrayList<DriverPO>  getAllDrivers() |
| 前置条件 | 无 |
| 后置条件 | 返回所有DriverPO |
| **DTManageDataSercvice. checkByPlateNum** | 语法 | public CarPO checkByPlateNum  (String plateNum) |
| 前置条件 | 无 |
| 后置条件 | 返回一个CarPO |
| **DTManageDataSercvice. getPlateNums** | 语法 | public ArrayList<String> getPlateNums() |
| 前置条件 | 无 |
| 后置条件 | 返回车牌号 |

#### TransportDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务** | | |
| **TransportDataSercvice.init** | **语法** | **Public void initial()** |
| **TransportDataSercvice.getLoadDocPO** | 语法 | Public LoadDocPO  getLoadDocPO(int id) |
| 前置条件 | 无 |
| 后置条件 | 系统返回相应的单据信息LoadDocPO |
| **TransportDataSercvice.addLoadDocPO** | 语法 | public ResultMessage  addLoadDocPO(LoadDocPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个LoadDocpo记录 |
| **TransportDataSercvice.getSendGoodDocPO** | 语法 | Public SendGoodDocPO  getSendGoodDocPO(int id) |
| 前置条件 | 无 |
| 后置条件 | 系统返回相应的单据信息SendGoodDocPO |
| **TransportDataSercvice.addSendGoodDocPO** | 语法 | public ResultMessage  addSendGoodDocPO(SendGoodDocPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个SendGoodDocpo记录 |
| **TransportDataSercvice.getArriveZZDocPO** | 语法 | Public ArriveZZDocPO  getArriveZZDocPO(int id) |
| 前置条件 | 无 |
| 后置条件 | 系统返回相应的单据信息ArriveZZDocPO |
| **TransportDataSercvice.addArriveZZDocPO** | 语法 | public ResultMessage  addArriveZZDocPO(ArriveZZDocPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个ArriveZZDocpo记录 |
| **TransportDataSercvice.getArriveYYDocPO** | 语法 | Public ArriveYYDocPO getArriveYYDocPO(int id) |
| 前置条件 | 无 |
| 后置条件 | 系统返回相应的单据信息ArriveYYDocPO |
| **TransportDataSercvice.addArriveYYDocPO** | 语法 | public ResultMessage  addArriveYYDocPO(ArriveYYDocPO po) |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个ArriveYYDocpo记录 |
| **TransportDataSercvice.getTransferDocPO** | 语法 | Public TransferDocPO getTransferDocPO(int id) |
| 前置条件 | 无 |
| 后置条件 | 系统返回相应的单据信息TransferDocPO |
| **TransportDataSercvice.addTransferDocPO** | 语法 | public ResultMessage  addTransferDocPO(TransferDocPO po) throws  RemoteException |
| 前置条件 | 同样条件的PO在数据库里不存在 |
| 后置条件 | 在数据库中增加一个TransferDocpo记录 |

|  |  |  |
| --- | --- | --- |
| **TransportDataService.getDocLists** | 语法 | public ArrayList<? extends DocPO>  getDocLists(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 返回单据列表 |
| **TransportDataService.changeDocsState** | 语法 | public ResultMessage  changeDocsState(ArrayList<String> docsID,  DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变所有单据状态 |
| **TransportDataService.changeOneDocState** | 语法 | public ResultMessage  changeOneDocState (String docID,DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变一个单据状态 |
| **TransportDataService.getDayDocCount** | 语法 | public int getDayDocCount(DocType type, MyDate date) |
| 前置条件 | 无 |
| 后置条件 | 获得当日该类型单据数量 |
| **TransportDataService.addPayDoc** | 语法 | public ResultMessage addPayDoc(PayDocPO po) |
| 前置条件 | 无 |
| 后置条件 | 增加一个付款单 |
| **TransportDataService.getPays** | 语法 | public ArrayList<PayDocPO> getPays() |
| 前置条件 | 无 |
| 后置条件 | 获得当日付款单数量 |

#### OrderDataService

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| OrderDataService.init | 语法 | Public void initial() |
| **OrderDataService.add** | 语法 | Public ResultMessage  add(orderPO po) |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加新增订单信息 |
| **OrderDataService.del** | 语法 | public ResultMessage  del(String orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 在数据库中删除原有的订单信息 |
| **OrderDataService.addDocToList** | 语法 | public ResultMessage addDocToList(DocPO po, ArrayList<String> orderBarCodes) |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加订单的流转信息 |
| **OrderDataService.checkType** | 语法 | private String checkType(String temp, DocType docType) |
| 前置条件 | 无 |
| 后置条件 | 返回一个单据类型 |
| **OrderDataService.getSingleOrderPO** | 语法 | public OrderPO getSingleOrderPO(String orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| **OrderDataService.getSingleOrderDocs** | 语法 | public ArrayList<String>  getSingleOrderDocs(String orderBarCode ) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| **OrderDataService.receiveInfo** | 语法 | public ResultMessage receiveInfo(ReceivePO PO ,String orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新订单的收件信息 |
| **OrderDataService.getDocLists** | 语法 | public ArrayList<? extends DocPO>  getDocLists(DocType type) |
| 前置条件 | 无 |
| 后置条件 | 返回单据列表 |
| **OrderDataService.changeDocsState** | 语法 | public ResultMessage  changeDocsState(ArrayList<String> docsID,  DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变所有单据状态 |
| **OrderDataService.changeOneDocState** | 语法 | public ResultMessage  changeOneDocState (String docID,DocType type, DocState state) |
| 前置条件 | 无 |
| 后置条件 | 改变一个单据状态 |
| **OrderDataService.getDayDocCount** | 语法 | public int getDayDocCount(DocType type, MyDate date) |
| 前置条件 | 无 |
| 后置条件 | 获得当日该类型单据数量 |

# 信息视角

## 数据持久化对象

系统的PO类就是对应的相关的实体类，如下所示。本系统共包含23个PO类，在此

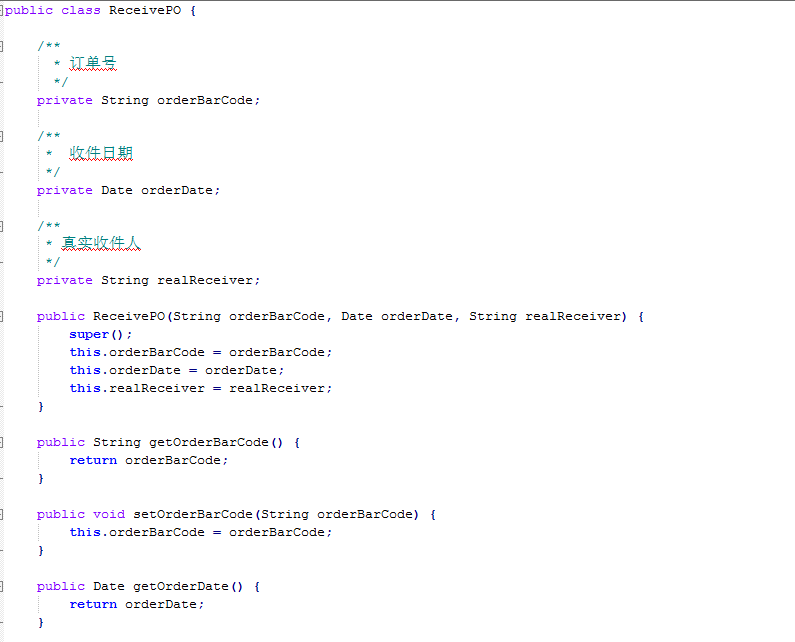
不一一赘述，只做部分介绍：

AccountPO类为账户信息，包括ID、密码、账户类型、个人其他信息。

ConstPO类为系统常量，包括城市间距离，交通方式运费，包装费用等等。

DocPO类为所有单据父类，储存了单据ID、生成时间、单据类型、单据状态。

持久化对象ReceivePO类的定义如图所示：



## 文件格式

本系统依据不同情况分别采用数据库、序列化、txt文件持久化保存三种方式保存文件