酒店预订系统

详细设计文档

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 |
| 郭浩滨 | 2016-10-26 | 建立整体框架，完成部分文档 | V0.1 |

**目录**

[1 引言 2](#_Toc465273046)

[1.1 编制目的 2](#_Toc465273047)

[1.2 词汇表 2](#_Toc465273048)

[1.3 参考资料 3](#_Toc465273049)

[2 产品概述 3](#_Toc465273050)

[3 体系结构设计概述 4](#_Toc465273051)

[4 结构视角 4](#_Toc465273052)

[4.1 展示层的分解 4](#_Toc465273053)

[4.1.1 UI层设计概要 4](#_Toc465273054)

[4.1.2 UI层内部包的职责与划分 4](#_Toc465273055)

[4.2 业务逻辑层的分解 9](#_Toc465273056)

[4.2.1 User模块 9](#_Toc465273057)

[4.2.2 Member模块 14](#_Toc465273058)

[4.2.3 Hotel模块 20](#_Toc465273059)

[4.2.4 Order模块 28](#_Toc465273060)

[4.2.5 Promotion模块 37](#_Toc465273061)

[4.3 数据层的分解 44](#_Toc465273062)

[4.3.1 DataFactoryService模块 44](#_Toc465273063)

[4.3.2 UserDataService模块 46](#_Toc465273064)

[4.3.3 MemberDataService模块 46](#_Toc465273065)

[4.3.4 HotelDataService模块 48](#_Toc465273066)

[4.3.5 OrderDataService模块 50](#_Toc465273067)

[4.3.6 PromotionDataService模块 51](#_Toc465273068)

[5 依赖视角 53](#_Toc465273069)

# 引言

## 编制目的

本报告详细完成对酒店预订系统的详细设计，达到指导后续软件构造的目的，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 | 用户 |
| 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》

《酒店预订系统体系结构设计文档V1.1.2》

# 产品概述

参考酒店预订系统用例文档及酒店预订系统规格说明文档对产品的概括描述。

随着工作和生活节奏的加快，人们的差旅出行需要迅速提升，但是传统上的线下酒店预订模式有着很多不方便之处：

* 因为目的地往往是外地，无法全面了解当地的酒店情况，选择范围大大受限，难以选择到最适合的酒店；
* 提前预订的双方都有风险，酒店担心恶意预订，顾客担心酒店不符合预期；
* 酒店要及时掌握预订顾客的行程变化也会比较麻烦，需要频繁的长途联系；
* 虽然过去的体验能够给未来带来参考，但是时间一长很多信息就会被遗忘掉，过去的参考作用就受到限制。（事实上，过去的出行体验会大大影响人们未来的选择）

酒店预订系统就是为满足xx创业公司创业发展需求开发的，它包括一个数据集中服务器和多个客户端。数据集中服务器将所有的数据储存起来进行维护。用户通过客户端完成日常任务，客户端与数据集中服务器采取实时通信的方式完成数据交换。

# 体系结构设计概述

参考酒店预订系统体系结构设计文档的描述。

主要采用分层的架构方式，采用分布式的部署方式，将数据存储在服务器端，客户端

通过RMI方式调用获取数据。

# 结构视角

## 展示层的分解

### UI层设计概要

ui 层主要以人员的性质来分，包括普通人（未登录）、客户、酒店工作人员、网站营销人员、网站管理人员。

ui层的入口是mainframe，显示的是打开系统的login界面，然后通过不同身份的人的账户登录，跳转到各个职位所对应的ui包内，之后职位所对应的各个功能的界面可以最大程度在包内跳转，更好的实现高内聚、低耦合。

### UI层内部包的职责与划分

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块 | | 包名 | 类名 | | 功能 |
| Common | CommonFrame | | | | 初始化客户查询物流状态的窗口 |
| CommonInitialPanel | | | | 初始化客户查询物流状态的面板 |
| FindSimpleOrderInfoPanel | | | | 查询订单 |
| Config | ButtonOrLabelPicture | | | | 用来构造map的一组图 |
| DataType | | | |  |
| GraphicsUtils | | | | 存放UI层一些可用的静态方法 |
| ParseXML | | | | 对XML文件进行解析（读取修改） |
| SimpleDataFormat | | | | 格式处理包装类 |
| UserfulMethod | | | | 保存一些有用的UI层静态方法 |
| courier | Addorder | | | AddOrderPanel | 查询订单界面 |
| FindFullInfo | | | FindFullIOrderInfoPanel | 查询完整订单信息 |
| showInfoPanel | 订单信息界面 |
| showTable | 订单信息列表 |
| inputInfpReceive | | | InputReceiveInfoPanel | 收件信息录入界面 |
| inputReceiveTablePanel | 收件信息列表 |
| CourierController | | | | 快递员界面跳转控制器 |
| CourierMainPanel | | | | 快递员主界面 |
| financeman | approval | | | FinanceApprovalPanel | 交易审核界面 |
| bankAccount | | | AddBankAccountPanel | 增加银行账户 |
| BankAccountManagePanel | 银行账户管理 |
| BankAccountTable | 银行账户列表 |
| buildBill | | | AddCar | 增加车辆主界面 |
| AddCarTable | 增加车辆信息表格 |
| AddInst | 增加机构主界面 |
| AddPeople | 增加人员主界面 |
| AddPeopleTable | 增加人员信息列表 |
| BuildBillPanel | 期初建账主界面 |
| DetailPanel |  |
| MainTable | 期初建账主界面信息表格 |
| buildPay | | | BuildPayPanel | 新建付款单 |
| PayTable | 付款单表格 |
| buildStateForm | | | BuildStateFormPanel | 经营状况分析界面 |
| costManage | | | CostAddPanel | 成本信息增加界面 |
| CostManagePanel | 成本管理 |
| CostMesTabel | 运费信息表格 |
| FinanceController | | | | 财务管理人员控制器类 |
| FinanceMain | | | | 财务人员主界面 |
| generalmanager | approval | | | ApprovalDetailPanel | 单据审批的详细界面，含有各个单据界面的引用~ |
| ApprovalDocsPanel | 审批单据 |
| DocSimpleInfoTable | 单据简要信息表 |
| constSet | | | ConstSetPanel | 制定常量表格 |
| institution | | | AddInstPanel | 增加机构界面 |
| InstManagePanel | 机构管理 |
| InstTablePanel | 机构信息表格 |
| people | | | AddPeoplePanel | 增加人员界面 |
| PeopleManagePanel | 人员管理，主要进入人员查看界面 |
| PeopleMesPanel | 人员信息表格 |
| salary | | | SalaryStrategyMesTablePanel | 薪水策略信息显示面板 |
| SalaryStrategySetPanel | 制定薪水策略 |
| statistic | | | CostFormPanel | 成本收益表 |
| StateFormDetailPanel | 经营状况表详细信息,包括 所有的收款付款信息 |
| StateFormPanel | 经营状况表,显示开始日期和结束日期以及序号 |
| StatisticPanel | 统计分析，仅有查看功能 |
| GeneralManagerController | | | | 总经理界面跳转控制器 |
| GeneralManagerMain | | | | 总经理主界面 |
| initial | InitialPanel | | | | 主面板界面，不动的界面 |
| mainframe | | | | 主界面 |
| login | Login | | | | 登陆界面 |
| LoginFrame | | | | 登陆界面窗口 |
| LoginPanel | | | | 登陆界面面板 |
| SetIPPanel | | | | 设置IP界面 |
| saleman | ArriveDoc | | | ArriveYYDocAddPanel | 接收单增加界面 |
| ArriveYYDocCheckPanel | 接收单增加界面 |
| ArriveYYDocMesTable | 接收单表格 |
| CarManage | | | CarManageAddPanel | 车辆信息增加界面 |
| CarManageCheckPanel | 车辆信息查看界面 |
| CarMesTable | 车辆信息表格 |
| DriverManage | | | DriverManageAddPanel | 司机信息增加界面 |
| DriverManageCheckPanel | 司机信息查看界面 |
| DriverMesPanel | 司机信息表格 |
| LoadDoc | | | LoadDocAddPanel | 装车单增加界面 |
| LoadDocCheckPanel | 装车单查看界面 |
| LoadDocMesTable | 装车单表格 |
| LoadDocOrders | 装车单里的订单列表 |
| PayDoc | | | PayDocAddPanel | 付款单增加界面 |
| PayDocCheckPanel | 付款单查看界面 |
| PayDocMesTable | 付款单表格 |
| PaySmallTable | 付款单里的订单列表 |
| SendGoodDoc | | | SendGoodDocAddPanel | 派送单增加界面 |
| SendGoodDocCheckPanel | 派送单查看界面 |
| SendGoodMesTable | 派送单表格 |
| SendSmallTable | 派送单里的订单列表 |
| SaleManController | | | | 营业厅业务员界面跳转控制器 |
| SaleManMain | | | | 营业厅业务员主界面 |
| storeman | arrivezz | | | ArriveZZDocAdd | 到达单增加界面 |
| ArriveZZPanel | 到达单查看界面 |
| ArriveZZTablePanel | 到达单表格 |
| storeshow | | | StoreManShowPanel | 库存查看界面 |
| StoreManSingleShowPanel | 库存查看详细界面 |
| transport | | | AddTransportPanel | 中转单增加界面 |
| TransportOrders | 中转单的订单列表 |
| TransportPanel | 中转单主界面 |
| TransportTablePanel | 中转单列表信息 |
| StoreMain | | | | 中转中心业务员主界面 |
| StoreManController | | | | 中转中心业务员界面跳转控制器 |
| storemanager | alarm | | | AlarmRecoverPanel | 警戒值恢复界面 |
| AlarmRecoverSinglePanel | 报警值恢复详细界面 |
| AlarmSetPanel | 报警值设置界面 |
| AlarmSetSinglePanel | 报警值设置详细界面 |
| instore | | | AddInStorePanel | 增加入库单界面 |
| AddLocTablePanel | 增加入库单订单列表 |
| FullInPanel | 一个入库单信息界面 |
| InStorePanel | 入库单主界面 |
| InStoreTablePanel | 入库单列表 |
| ShowLocTablePanel | 一个入库单里的订单列表 |
| outstore | | | AddOutStorePanel | 增加出库单界面 |
| OutStorePanel | 出库单主界面 |
| OutStoreTablePanel | 出库单列表 |
| storecheck | | | StoreCheckInfoPanel | 库存检查详细界面 |
| StoreCheckPanel | 库存检查主界面 |
| storeshow | | | OrderInfoTable | 订单信息列表 |
| StoreShowPanel | 库存查看界面 |
| StoreSingleShowPanel | 库存查看详细界面 |
| StoreManageMain | | | | 中转中心管理员主界面 |
| StoreManagerController | | | | 中转中心管理员界面跳转控制器 |
| table | MyTable | | | | 表格类 |
| MyTableModel | | | | 表格模型类 |
| MyTablePanel | | | | 一个用来封装表格的panel |
| showMessageFrame | | | | 测试用 |
| TableTest | | | | 测试用 |
| tools | AddDocPanel | | | | 增加单据的父类，封装其界面跳转 |
| CheckDocPanel | | | | 所有单据查看的界面  因为所有单据的增加、查看有极大的相似性，基本上分为查看和增加两个界面  这个父类将实现封装界面跳转、预声明部分组件,并完成一些共有组件：搜索框、日历、按钮的初始化、增加工作 |
| MyButton | | | | Button父类 |
| MyCardLayOut | | | | -------------- |
| MyCheckBox | | | | 复写复选框 |
| MyComboBox | | | | 重写下拉框类 |
| MyDatePicker | | | | 日历选择器实现类 包含三个部件：显示日历信息的textfield、按钮button |
| MyFrame | | | | 所有Frame父类 |
| MyJumpListener | | | | 所有跳转监听的父类 |
| MyLabel | | | | 所有Label的父类 |
| MyOptionPane | | | | 未使用 |
| MyPanel | | | | 所有Panel的父类 |
| MyPanelWithScroller | | | | 带有滚动条的面板 |
| MyPasswordField | | | | 密码输入框 |
| MyPictureButton | | | | 带有图片的button按钮 |
| MyPictureLabel | | | | 带有图片的Label父类 |
| MyRadioButton | | | | 单选框 |
| MySearchBox | | | | 搜索框 |
| MySideBarButton | | | | 左边条Button |
| MyTable | | | | 所有表格的实现父类 |
| MyTextArea | | | | 文字输入区域设置透明 |
| MyTextField | | | | 重写输入框 |
| MyWhitePanel | | | | 白色面板的父类 |
| user | AccountMesTablePanel | | | | 账户信息列表 |
| AddAccountPanel | | | | 增加账户界面 |
| AdminstratorController | | | | 管理员界面跳转控制器 |
| ModifyAccountPanel | | | | 修改账户界面 |
| UserManageMain | | | | 用户管理主界面 |
| util | ButtonState | | | | 按钮状态 |
| CancelListener | | | | 各种单据取消时监听按钮 |
| CompomentType | | | | 组件类型枚举 |
| ConfigConstValue | | | | -------------------- |
| ConfirmListener | | | | 各种单据保存时确认按钮 |
| DocPanelForApproval | | | | 审批单据接口，用于单据界面 |
| MyBackListener | | | | 返回按钮 |
| MyButtonListener | | | | 按钮监听 |
| MyMainPanelButtonListener | | | | 主界面按钮监听类 |
| MyPictureButtonListener | | | | 按钮监听抽象父类 |
| MySideBarListener | | | | 左边条按钮监听实现类 |
| PanelController | | | | 界面跳转控制器父类 |
| RefreshPanel | | | | 接口，实现界面刷新 |
| RemoteExceptionHandler | | | | 服务端异常处理机制 |
| TextFieldsManage | | | | 未使用 |
| TipsDialog | | | | 提示框 |

## 业务逻辑层的分解

业务逻辑层的开发包图请查看体系结构设计文档。

### User模块

1. 模块概述：

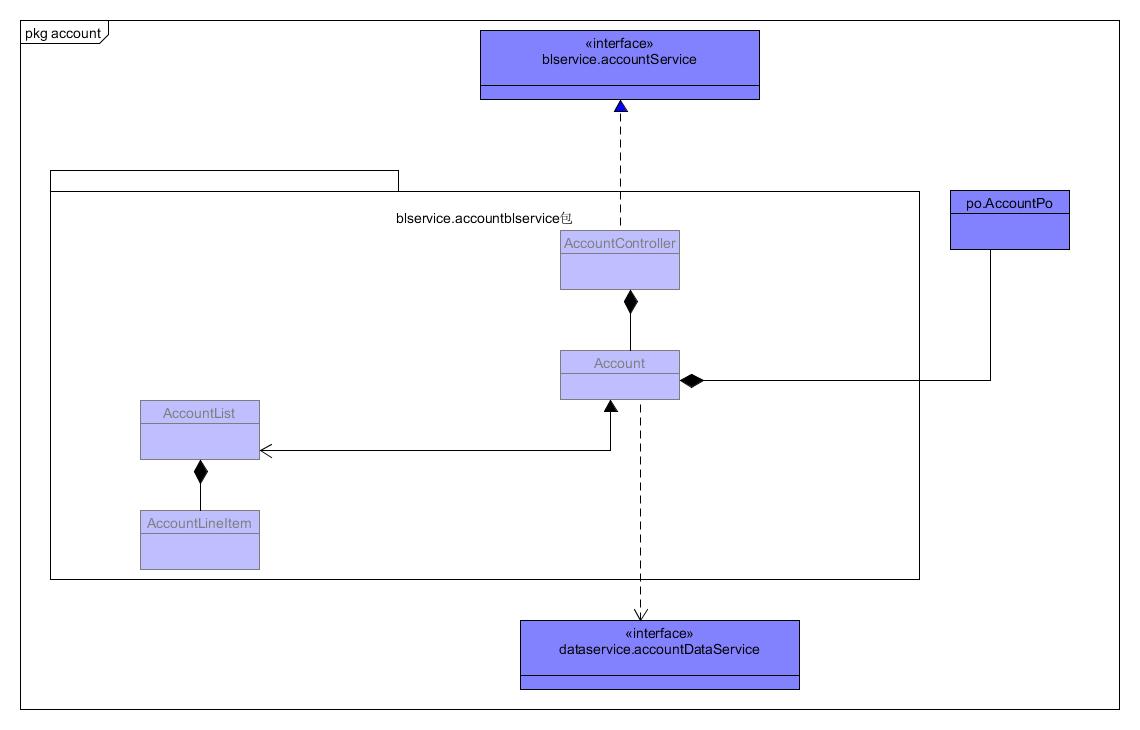
Account模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Account模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

1. 整体结构：

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。在展示层和业务逻辑层之间添加AccountblService接口。在业务逻辑层和数据层之间添加AccountdataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加AccountController，这样AccountController会将对商品管理的业务逻辑处理委托给Account对象。AccountPO是作为账户信息的持久性对象被添加到设计模型中去的。

**Accountbl模块的设计如图：**



**AccountBL模块中各个类的设计**

**Accountbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountController | 负责对应于管理员账户管理所需要的服务 |
| Account | 账户管理的领域模型对象，拥有账户的信息，可以帮助完成账户管理界面所需要的服务 |

1. 模块内部类的接口规范

**AccountController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountController.add | 语法 | Public ResultMessage add(AccountVO vo) |
| 前置条件 | 已经创建一个Account领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Account类，将新的账户添加到数据库中，持久化更新相关数据 |
| AccountController.delete | 语法 | Public ResultMessage delete(AccountVO vo) |
| 前置条件 | 已经创建一个Account领域对象，并且输入符合语法规则） |
| 后置条件 | 调用Account类，系统永久性删除该账户的数据，持久化更新相关数据 |
| AccountController.modify | 语法 | Public AccountVO find(String ID) |
| 前置条件 | 已经创建一个Account领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Account类，系统永久性修改该账户的数据，持久化更新相关数据 |
| AccountController.find | 语法 | Public ResultMessage modify(AccountVO vo) |
| 前置条件 | 已经创建一个Account领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Account类，系统返回查找到的搜索结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Account.add | 增加一个账户 | |
| Account.delete | 删除一个账户 | |
| Account.find | 查找一个账户 | |
| Account.modify | 修改一个账户 | |

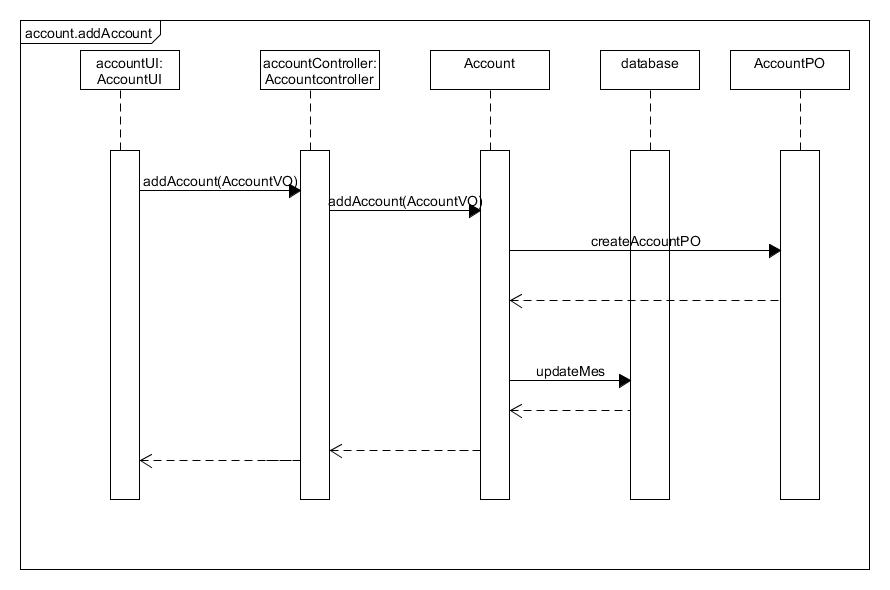
**Account的接口规范**

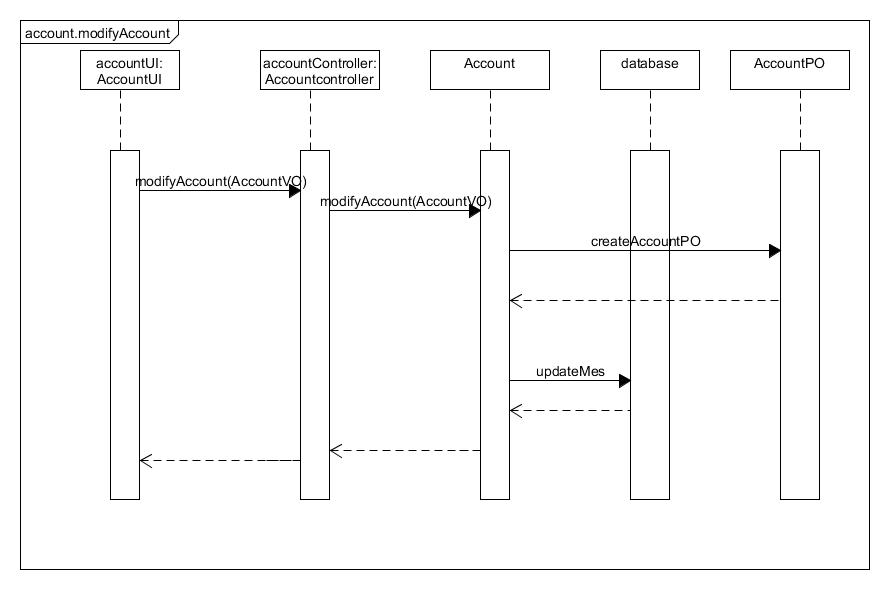
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Account.add | 语法 | Public ResultMessage add(AccountVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| Account.delete | 语法 | Public ResultMessage delete(AccountVO vo) |
| 前置条件 | 无（删除表明该数据必然是存在的） |
| 后置条件 | 系统永久性删除该账户的数据 |
| Account.find | 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 |
| 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 | |

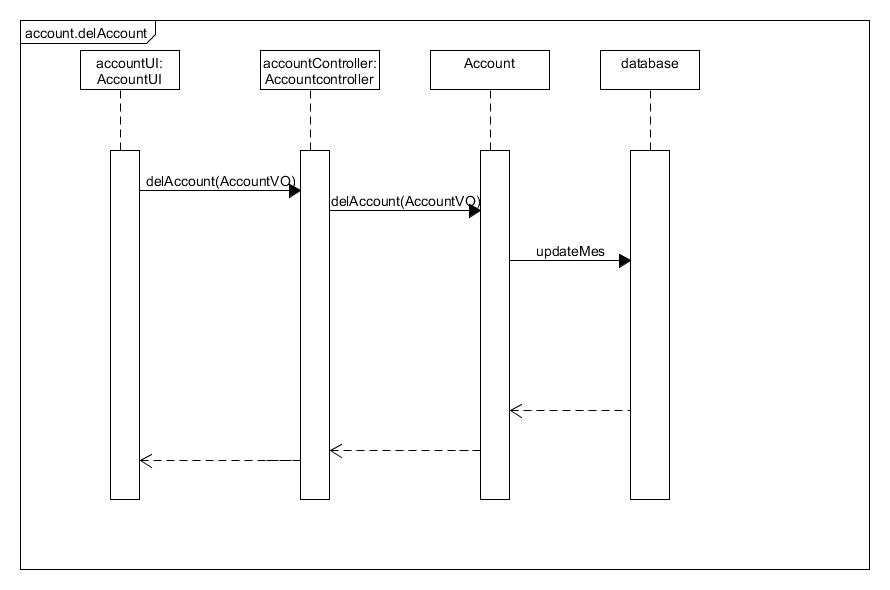
1. 业务逻辑层的动态模型

在系统账户管理中，业务分为增加、删除、修改、查找账户信息，其中增加、删除、修

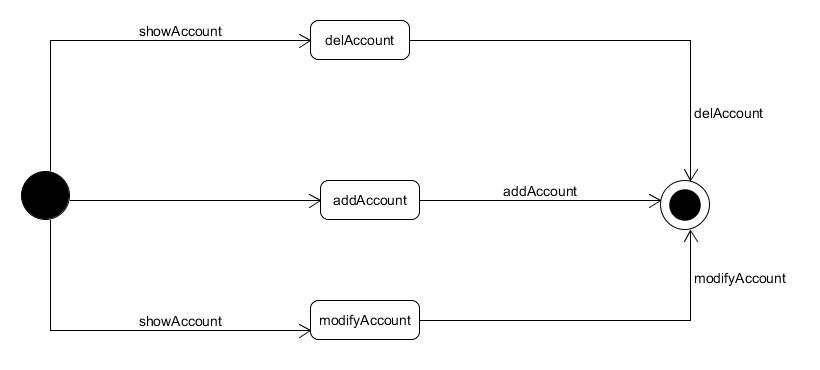
改账户信息的顺序图如下图所示：







下图描述了Account对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



1. 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### Member模块

1、模块概述：

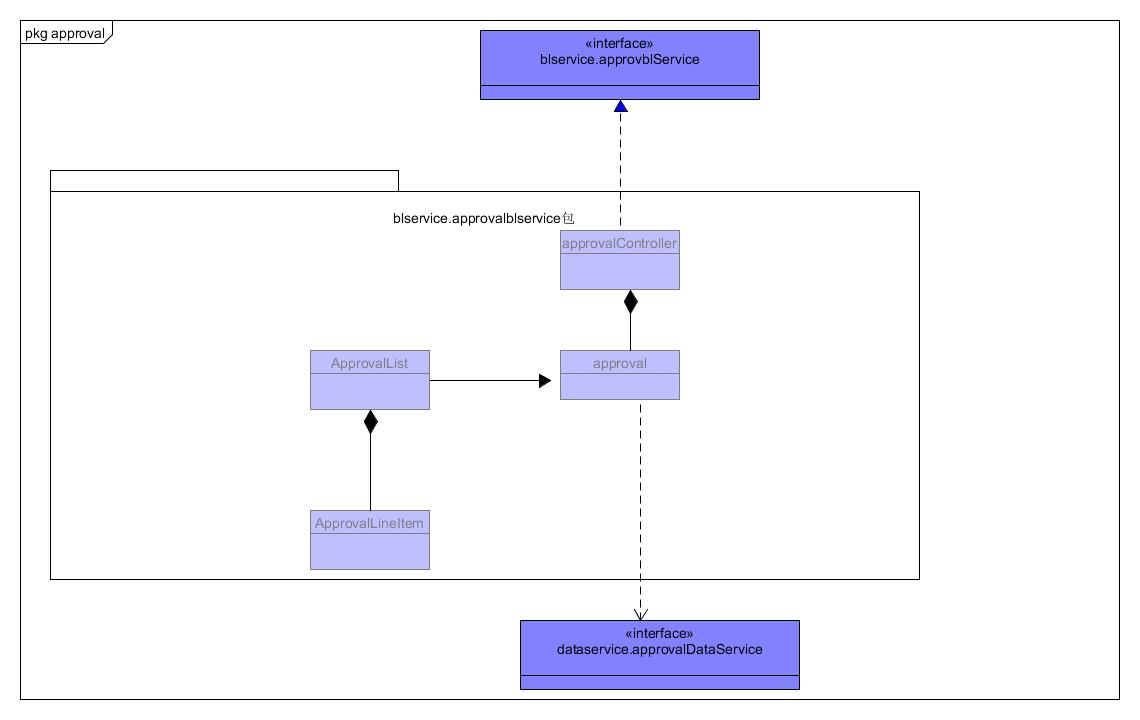
**Approval**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**Approval**模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

2、整体结构：

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。在展示层和业务逻辑层之间添加ApprovalblService接口。在业务逻辑层和数据层之间添加ApprovaldataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加ApprovalController，这样ApprovalController会将对商品管理的业务逻辑处理委托给Approval对象。ApprovalPO是作为账户信息的持久性对象被添加到设计模型中去的。

**Approval模块中各个类的设计：**



**Approval模块中各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ApprovalController | 负责对应于审批单据所需要的服务 |
| Approval | 审批单据的领域模型对象，拥有单据的信息，可以帮助完成审批单据所需要的服务 |

1. 模块内部类的接口规范

**Approvalcontroller模块的接口规范**

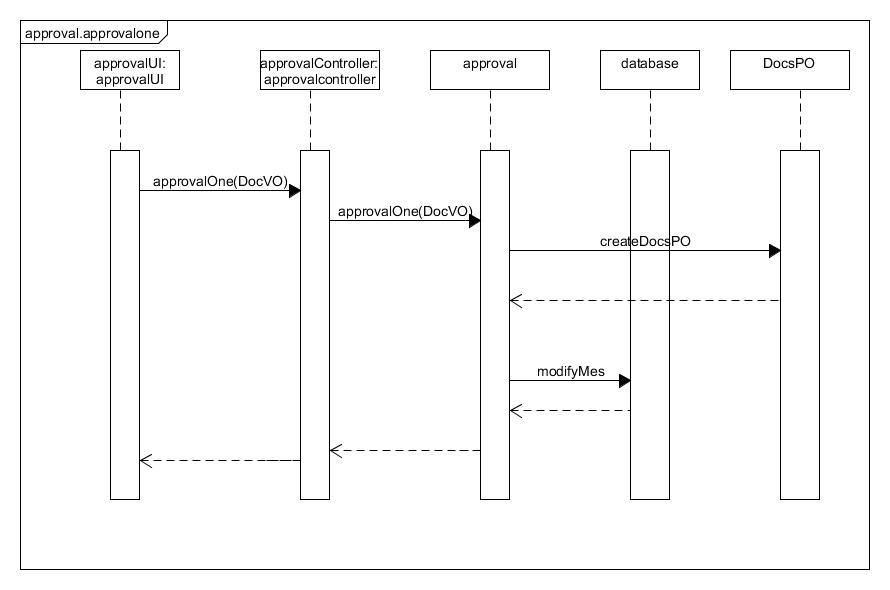
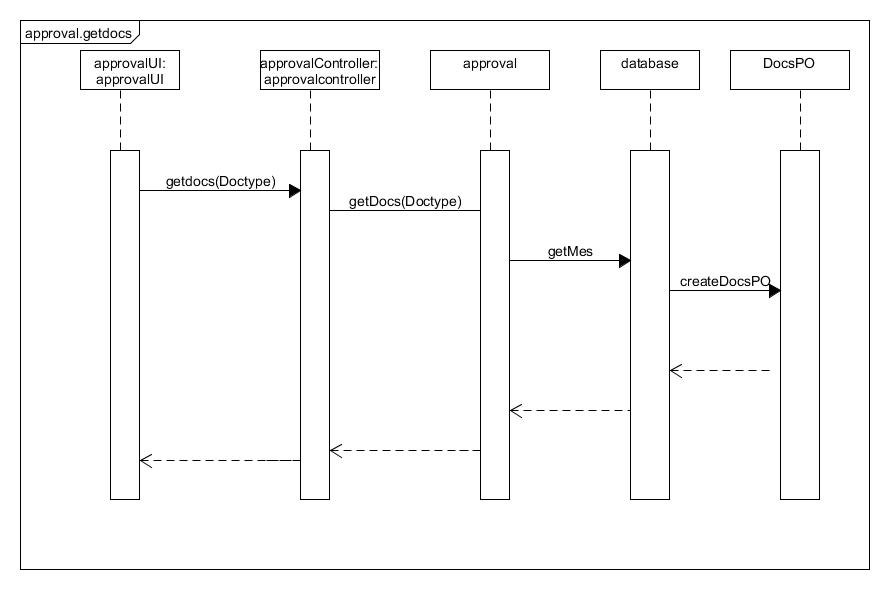
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Approvalcontroller.getDocs | 语法 | Public ArrayList<DocVO> getbills(DocType type) |
| 前置条件 | 已经创建一个Approval领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Approval类的getDocs方法 |
| Approvalcontroller.approveone | 语法 | Public ResultMessage approveone(DocVO vo) |
| 前置条件 | 已经创建一个Approval领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Approval类的approveone方法 |
| Approvalcontroller.approvemany | 语法 | PublicResultMessage approvemany(Arraylist<  DocVO > vo) |
| 前置条件 | 已经创建一个Approval领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Approval类的approvemany方法 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Transport.getDoc(DocType type) | 从货物流转包中获取各种单据 | |
| Order.getorders() | 获取订单 | |
| Store.getDoc(DocType type) | 获取库存相关的单据 | |
| Approval.getbills | 获得所有单据 | |
| Approval.approveone | 审批单个单据 | |
| Approval.approvemany | 审批多个单据 | |

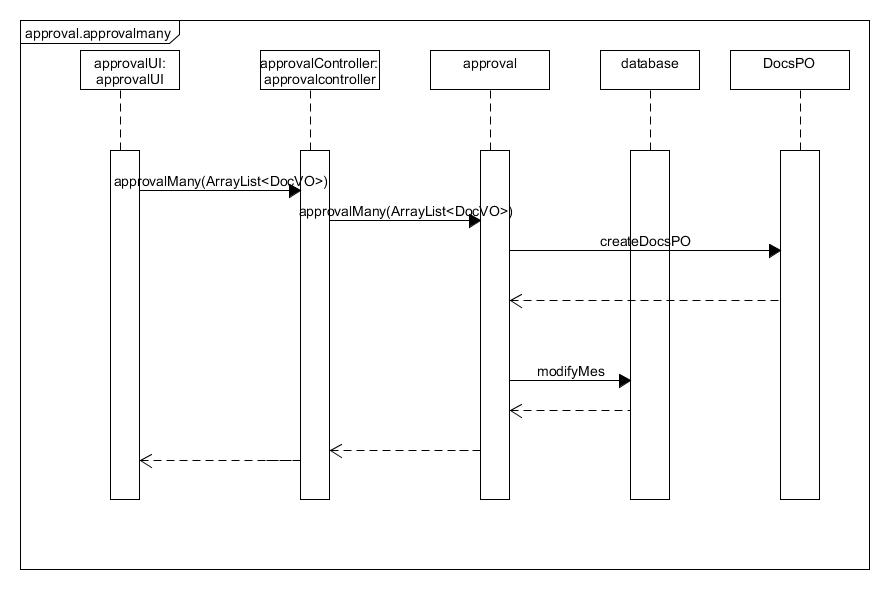
**approvalbl模块的接口规范**

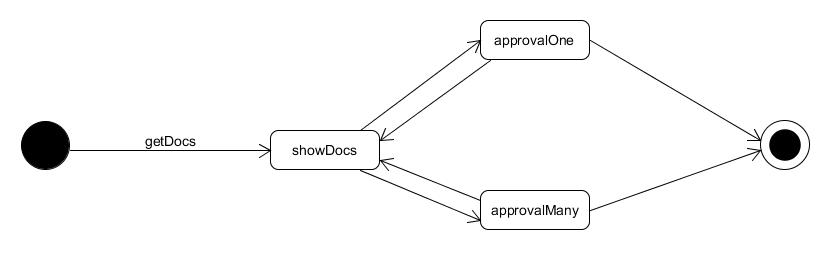
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| approval. getbills | 语法 | Public ArrayList<DocVO> getbills(DocType type) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| Approval.approveone | 语法 | Public ResultMessage approveone(DocVO vo) |
| 前置条件 | 该单据未审批过 |
| 后置条件 | 将该单据状态改为审批通过 |
| Approval.approvemany | 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 | |

1. 业务逻辑层的动态模型

在审批单据中，业务分为获得单据信息、单个审批单据、批量审批单据，它们的顺序图

如下图所示：



下图描述了Approval对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。

1. 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象

### Hotel模块

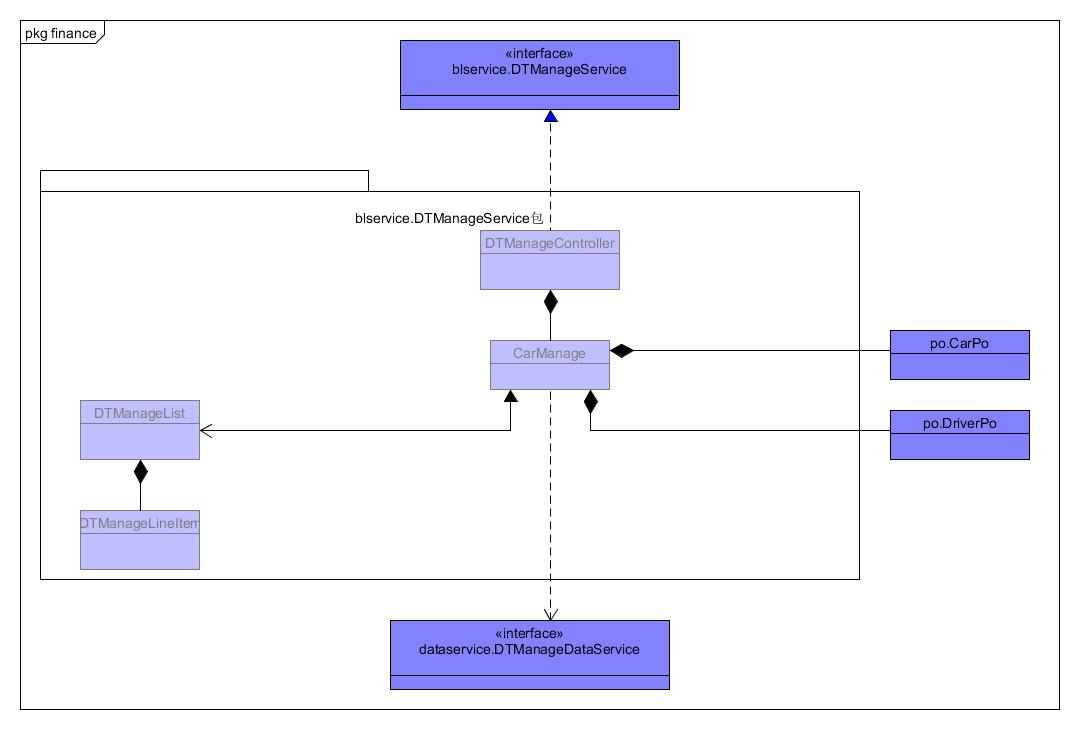
1、模块概述：

**DTManage**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**DTManage**模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

2、整体结构：

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。在展示层和业务逻辑层之间添加DTManageblService接口。在业务逻辑层和数据层之间添加DTManagedataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加DTManageController，这样DTManageController会将对司机和车辆信息管理的业务逻辑处理委托给DTManage对象。DTManagePO是作为账户信息的持久性对象被添加到设计模型中去的。



**图4-1-3-1 DTManagebl模块各个类的设计**

DTManagebl模块各个类的职责如表4-1-3-2所示。

**表4-1-3-2 DTManagebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| DTManageController | **实现司机车辆信息管理部分需要的服务** |
| DTManage | **实现司机车辆的信息管理，包括增删改查4项功能** |

3、模块内部类的接口规范

DTManageController和DTManage的接口规范如表4-1-3-3和表4-1-3-4所示

**表4-1-3-3 DTManageController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DTManageController.AddDriver | 语法 | Public ResultMessage addDriver  (DriverVO vo) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，将新的账户添加到数据库中，持久化更新相关数据 |
| DTManageController.CheckDriverbyName | 语法 | Public ArrayList< DriverVO >  CheckDriverByName(String name) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的CheckDriverbyName方法，系统返回有重合关键字的司机信息 |
| DTManageController.CheckDriverbyID | 语法 | Public DriverVO CheckDriverByID  (String ID) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的CheckDriverbyID方法，系统返回相应ID对应的司机信息 |
| DTManageController.CheckDriverbyInst | 语法 | Public ArrayList< DriverVO >  CheckDriverByInst(String InstID) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的CheckDriverbyInst方法，系统返回对应机构名下的司机信息 |
| DTManageController.ModifyDriver | 语法 | Public ResultMessage modifyDriver  (DriverVO vo) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，系统永久性修改该账户的数据，持久化更新相关数据 |
| DTManageController.DelDriver | 语法 | Public ResultMessage DelDriver(DriverVO vo) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，系统永久性删除该账户的数据，持久化更新相关数据 |
| DTManageController.getDriverName | 语法 | Public ArrayList<DriverList> getDriverName (String InstID) |
| 前置条件 | 已经创建一个Driver领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的getDriverName方法，获得该机构名下的司机列表 |
| DTManageController.AddCar | 语法 | Public ResultMessage addCar  (CarVO vo) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，将新的账户添加到数据库中，持久化更新相关数据 |
| DTManageController.CheckCarbyID | 语法 | Public CarVO CheckCarByID(String ID) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的CheckCarbyID方法，系统返回对应ID下的车辆信息 |
| DTManageController.CheckCarbyPlateNum | 语法 | Public CarVO CheckCarByPlateNum  (String PlateNum) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的CheckCarbyPlateNUm方法，系统返回车牌号对应的车辆信息 |
| DTManageController.ModifyCar | 语法 | Public ResultMessage modifyCar  (CarVO vo) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，系统永久性修改该账户的数据，持久化更新相关数据 |
| DTManageController.DelCar | 语法 | Public ResultMessage DelCar (CarVO vo) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类，系统永久性删除该账户的数据，持久化更新相关数据 |
| DTManageController.getPlateNum | 语法 | Public ArrayList<PlateNumber>  getPlateNumber(String InstID) |
| 前置条件 | 已经创建一个Car领域对象，并且输入符合语法规则 |
| 后置条件 | 调用DTManage类的getPlateNum方法，得到该机构下的车辆车牌号列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DTManage.AddDriver | 增加司机信息 | |
| DTManage.CheckDriverbyName | 通过姓名查找司机 | |
| DTManage.CheckDriverbyID | 通过ID查找司机 | |
| DTManage.CheckDriverbyInst | 通过机构名查找司机 | |
| DTManage.ModifyDriver | 修改司机信息 | |
| DTManage.DelDriver | 删除司机信息 | |
| DTManage.getDriverName | 获得该机构名下的司机列表 | |
| DTManage.AddCar | 增加车辆信息 | |
| DTManage.CheckCarByID | 通过ID查找车辆 | |
| DTManage.CheckCarByPlateNum | 通过车牌号查找车辆 | |
| DTManage.modifyCar | 修改车辆信息 | |
| DTManage.delCar | 删除车辆信息 | |
| DTManage.getAllCars | 得到所有的车辆信息 | |
| DTManage. getAllDrivers | 得到所有的司机信息 | |
| DTManage.getPlateNum | 得到该机构下的车辆车牌号列表 | |

**DTmanagebl模块的接口规范**

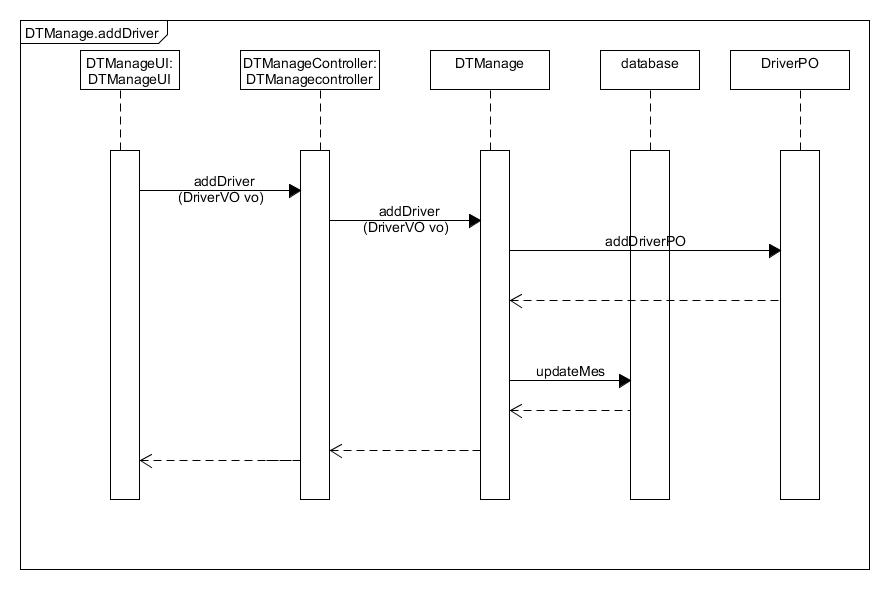
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DTManage.AddDriver | 语法 | Public ResultMessage addDriver  (DriverVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加该司机信息并持久化更新涉及的对象的数据 |
| DTManage.CheckDriverbyName | 语法 | Public ArrayList< DriverVO >  CheckDriverByName(String name) |
| 前置条件 | 无 |
| 后置条件 | 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 |

4、业务逻辑层的动态模型

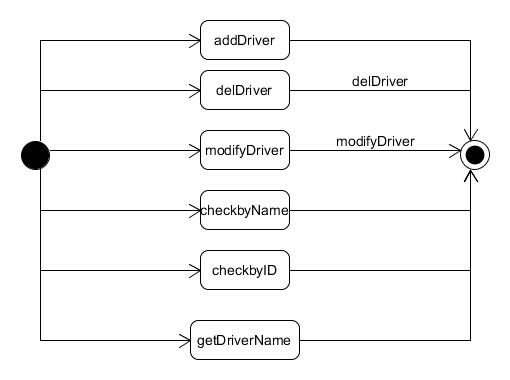
在司机车辆信息管理业务中，分为成本管理，生成收款信息，生成付款信息和利润查看功能。

图4-1-3-2表明了对司机车辆信息进行管理的相关对象之间的协作。



**图4-1-3-2司机信息进行管理**

图4-1-3-3 所示的状态图描述了Driver对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



**图4-1-3-3 Driver对象状态图**

### Order模块

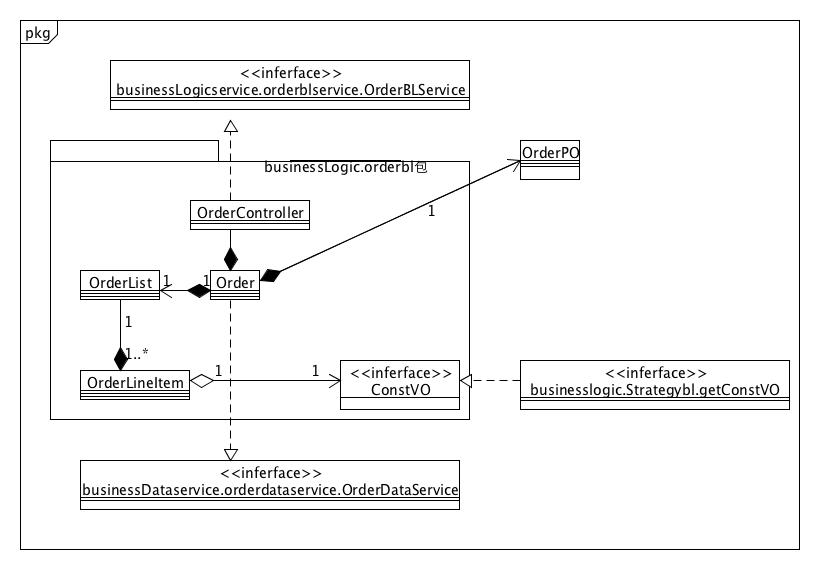
1、模块概述：

**Order**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**Order**模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

2、整体结构：

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。在展示层和业务逻辑层之间添加OrderblService接口。在业务逻辑层和数据层之间添加OrderdataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加OrderController，这样OrderController会将对商品管理的业务逻辑处理委托给Finance对象。Order PO是作为订单信息的持久性对象被添加到设计模型中去的。



**orderbl模块各个类的设计**

orderbl模块各个类的职责如表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| orderController | 负责对应于订单所需要的服务 |
| order | 订单的领域模型对象，拥有订单的基本信息、货物信息、流转信息、收件信息，可以帮助完成订单所需要的服务 |

3、模块内部类的接口规范

**orderController模块的接口规范**

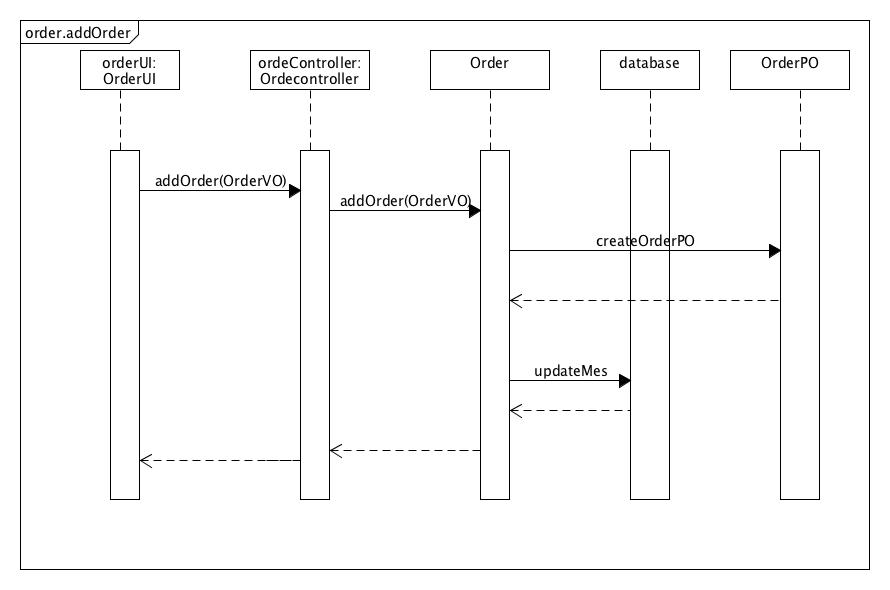
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderController.add | 语法 | Public ResultMessage add(orderVO orderVO) |
| 前置条件 | 已经创建一个order领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Order类add方法 |
| OrderController.checkBarCode | 语法 | Public ResultMessage checkBarCode(String  orderBarCode) |
| 前置条件 | 在presentation层已经确认barCode（条形码）为10位 |
| 后置条件 | 调用Order类checkBarCode方法 |
| OrderController.getOrderVO | 语法 | public ArrayList< OrderVO >  getOrderVO(Date date) |
| 前置条件 | 已经创建一个order领域对象，并且输入符合语法规则 |
| 后置条件 | 调用order类getOrderVO方法 |
| OrderController.delete | 语法 | Public ResultMessage delete(String  orderBarCode) |
| 前置条件 | 已经创建一个order领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Order类delete方法 |
| OrderController.getSimpleInfo | 语法 | Public ResultMessage getSimpleInfo(String  orderBarCode) |
| 前置条件 | 已经创建一个ordor领域对象，并且输入符合语法规则 |
| 后置条件 | 调用order类，将指定的订单的完整信息展示出来 |
| OrderController.getFullInfo | 语法 | Public ResultMessage getFullInfo(String  orderBarCode) |
| 前置条件 | 已经创建一个ordor领域对象，并且输入符合语法规则 |
| 后置条件 | 调用order类，将指定的订单的完整信息展示出来 |
| OrderController.receiveInfo | 语法 | Public ResultMessage  receiveInfo(ArrayList<ReceiveVO>) |
| 前置条件 | 已经创建一个ordor领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Order类，将新的订单信息添加到数据库中，持久化更新相关数据 |
| OrderController.addDocList | 语法 | Public ResultMessage addDocList(DocVO vo) |
| 前置条件 | 已经创建一个order领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Order类，将新的订单信息添加到数据库中，持久化更新相关数据 |
| OrderController.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 |
| OrderController.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 |
| OrderController.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 |
| OrderController.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 | |

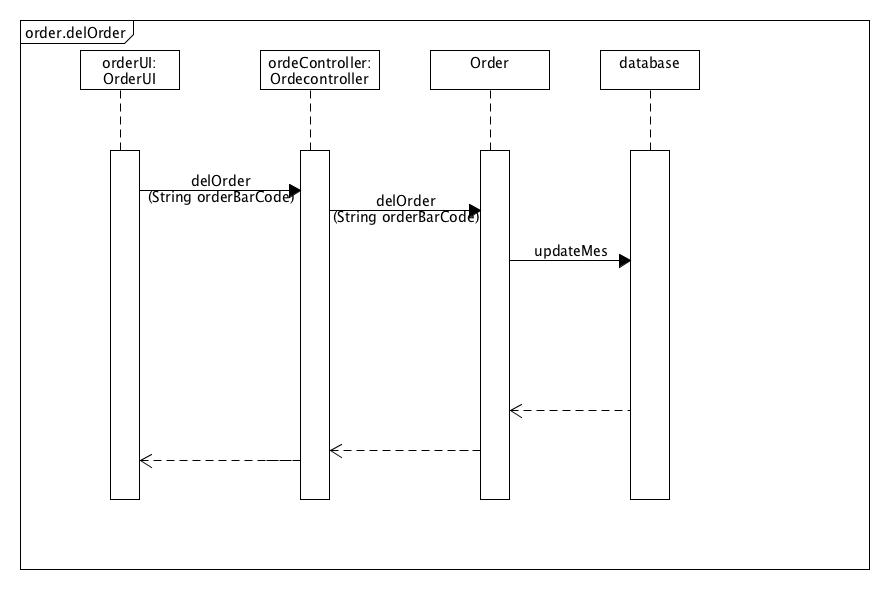
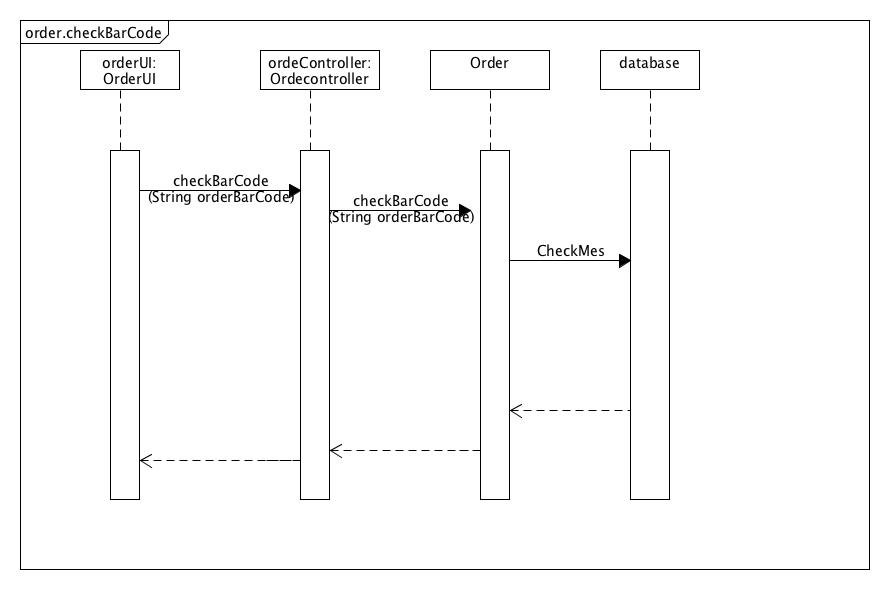
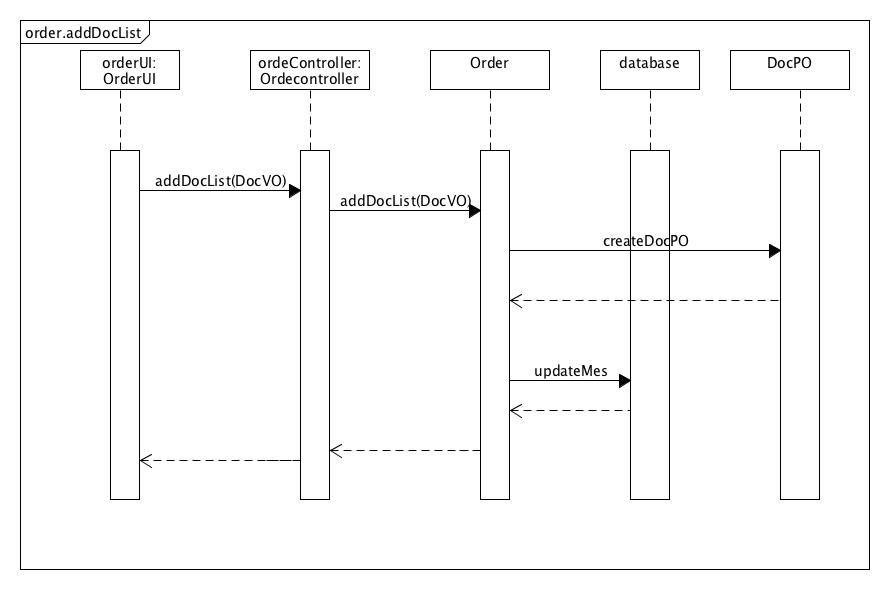
**orderbl模块的接口规范**

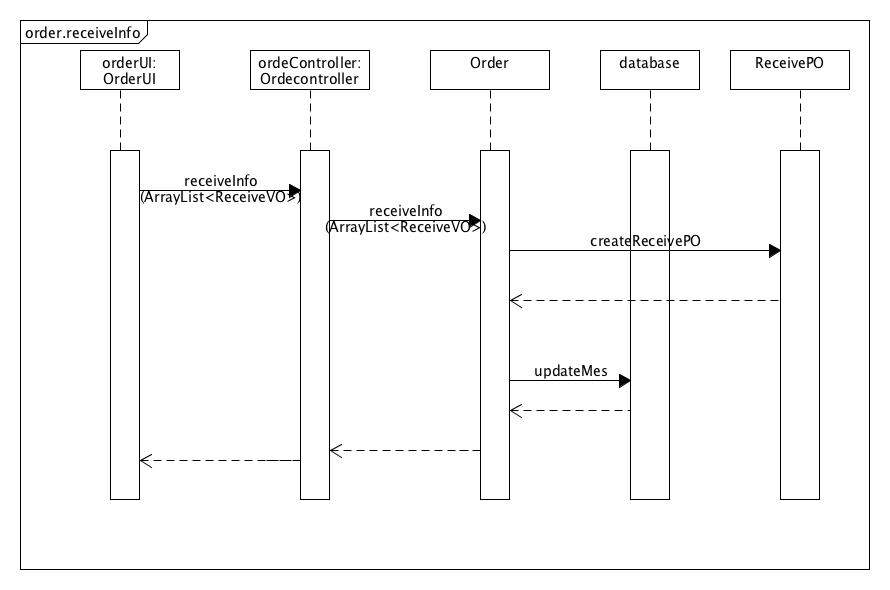
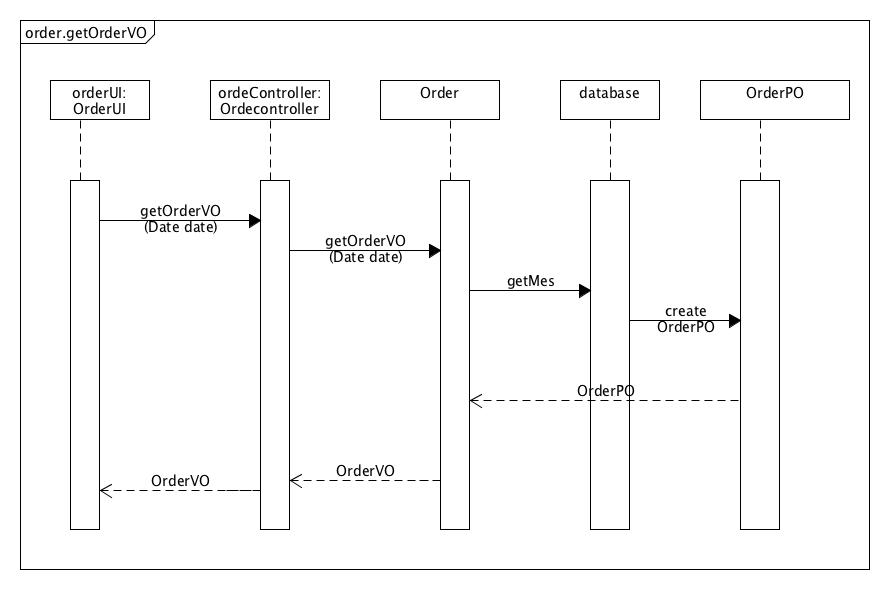
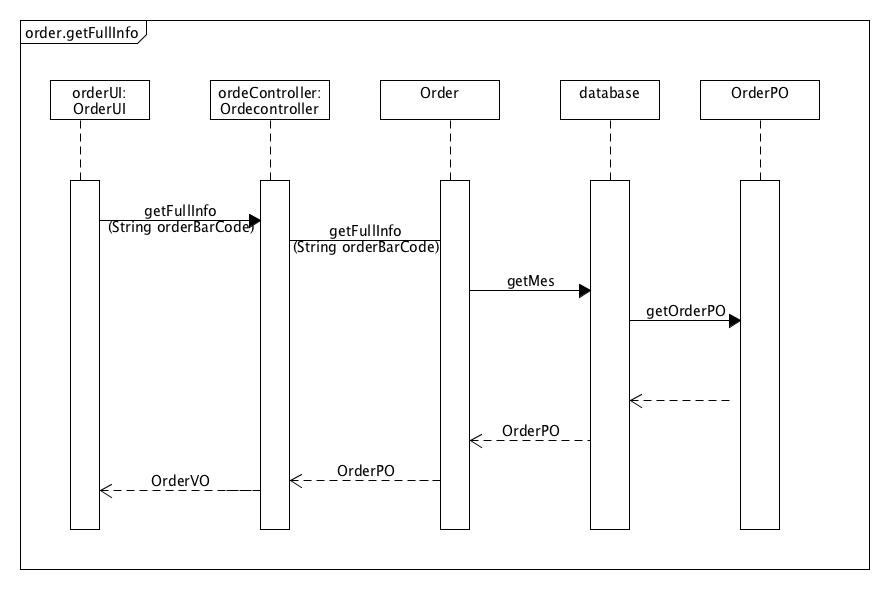
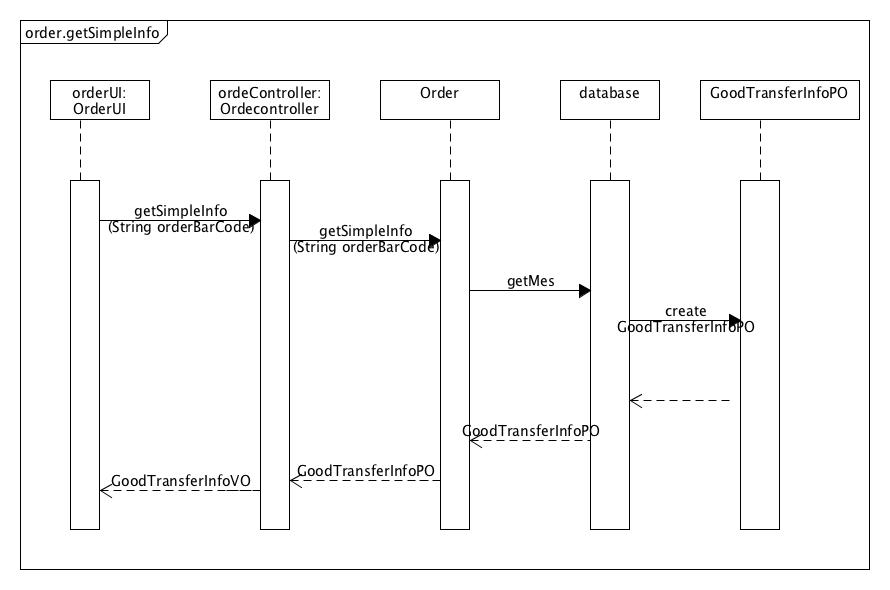
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.add | 语法 | Public ResultMessage add(orderVO orderVO) |
| 前置条件 | 输入的数据格式正确 |
| 后置条件 | 系统自动生成新的订单，并持久化添加数据 |
| Order.checkBarCode | 语法 | Public ResultMessage checkBarCode(String  orderBarCode) |
| 前置条件 | 在presentation层已经确认barCode（条形码）为10位 |
| 后置条件 | 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 | |
|  |  | |

4、业务逻辑层的动态模型

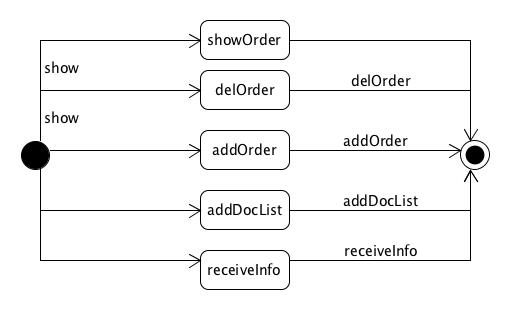
在订单管理中，业务分为增加订单、增加流转信息、检查条形码号、删除订单、获得完整订单信息、获得简略订单信息、获得orderpo，收件信息录入，它们的顺序图如下图所示：







下图描述了order对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



5、业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### Promotion模块

1、模块概述：

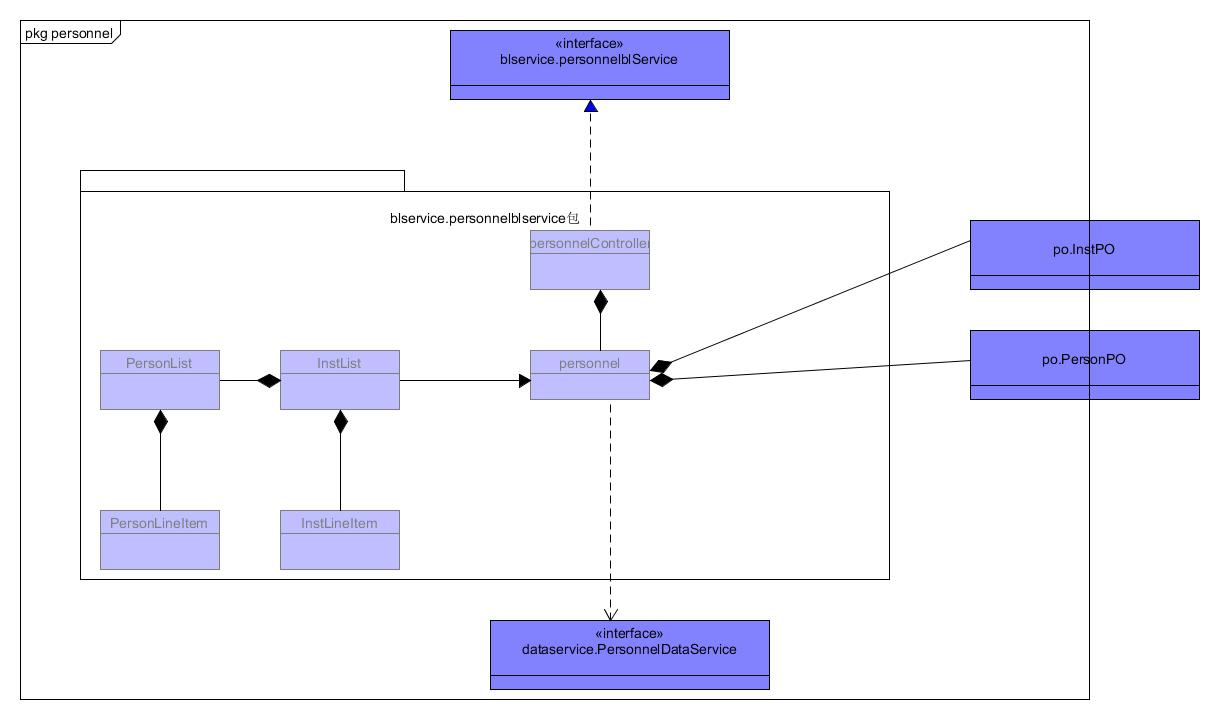
**personnel**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**personnel**模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

2、整体结构：

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。在展示层和业务逻辑层之间添加PersonnelblService接口。在业务逻辑层和数据层之间添加PersonneldataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加PersonnelController，这样PersonnelController会将对商品管理的业务逻辑处理委托给Personnel对象。InstPO、PersonPO是作为人员、机构信息的持久性对象被添加到设计模型中去的。

Personnel模块中各个类的设计：



Personnelbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| PersonnelController | 负责实现商品分类管理界面所需要的服务 |
| Personnel | 商品分类的领域模型对象，拥有商品分类的信息，可以帮助完成商品分类界面所需要的服务 |
| InstList | 保存机构的数据 |
| InstLineItem | 机构的条目信息 |
| PersonList | 保存人员的数据 |
| PersonLineItem | 人员的条目信息 |

3、模块内部类的接口规范

**PersonnelController类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PersonnelController.getpeopleByInstID | 语法 | Public ArrayList<PersonVO>  getpeopleByInst(String ID) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的getpeopleByInstID方法，系统返回该机构下所有人员信息 |
| PersonnelController.getpeopleByID | 语法 | Public PersonVO getpeopleByID(String ID) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的getpeopleByID方法，返回搜索结果 |
| PersonnelController.getpeopleByName | 语法 | Public ArrayList<PersonVO>  getpeopleByName(String name) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的getpeopleByName方法，返回有重合关键字的人员信息 |
| PersonnelController.addpeople | 语法 | Public ResultMessage addpeople(PersonVO vo) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的addpeople方法，系统增加该持久化对象 |
| PersonnelController.deletepeople | 语法 | Public ResultMessage deletepeople(String ID) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的deletepeople方法，系统删除该持久化对象 |
| PersonnelController.addInst | 语法 | Public ResultMessage addInst(InstVO vo) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的addInst方法，系统增加该机构信息 |
| PersonnelController.getInst | 语法 | Public ArrayList<InstVO> getInst(); |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的getInst方法，系统返回所有机构信息 |
| PersonnelController.delnst | 语法 | Public ResultMessage deleteInst(String ID) |
| 前置条件 | 已经创建一个Personnel领域对象，并且输入符合语法规则信息符合规则且齐全 |
| 后置条件 | 调用Personnel的delnst方法，系统删除该机构信息 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Personnel.getpeobyInst(String ID) | 根据机构ID返回该机构下属人员信息 | |
| Personnel.getpeobyID(String ID) | 根据ID返回人员信息 | |
| Personnel.getpeobyName(String name) | 根据姓名返回人员信息 | |
| Personnel.addpeople(PersonPo po) | 增加人员的持久化对象 | |
| Personnel.delpeople(String ID) | 删除人员的持久化对象 | |
| Personnel.addInst (InstPo po) | 增加机构的持久化对象 | |
| Personnel.delInst (InstPo po) | 删除机构的持久化对象 | |
| Personnel.getInst() | 得到返回所有机构信息 | |
| Personnel.getPersons() | 得到所有的人员信息 | |
| Personnel.modifyInst(InstVO vo) | 修改该机构信息 | |
| Personnel.modifyPerson(PersonVO vo) | 修改该人员信息 | |

**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 | |

4、业务逻辑层的动态模型

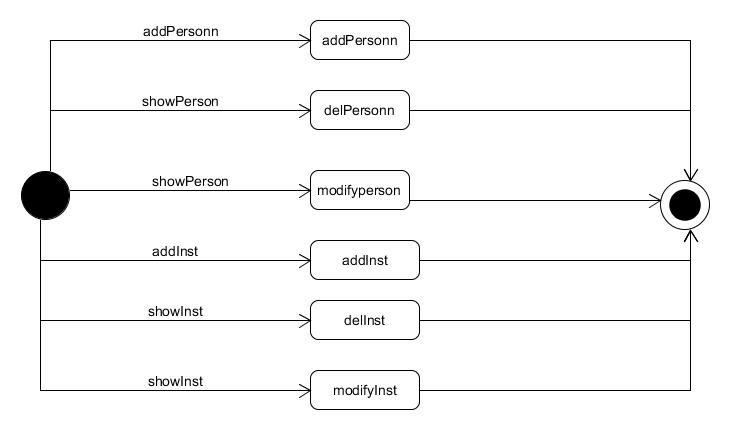
在人员机构管理管理中，业务分为增加、删除、修改、查找人员及机构信息，其中增加、删除、修改人员信息的顺序图如下图所示，机构的管理与此类似：







下图所示的状态图描述了personnel对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



5、业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

## 数据层的分解

数据层模块的分解详见需求规格说明文档中的相关描述

### DataFactoryService模块

1. 模块概述

DataFactoryService模块是一个工厂类，用来创建其他的DataService实例，并将其统一管理起来。

1. 模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 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 |

### UserDataService模块

（1）模块概述

AccountDataService模块负责保存用户账户的信息，进行增删改查的操作。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 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 |

### MemberDataService模块

（1）模块概述

DTManageDataService模块负责保存司机和车辆信息供读取、修改和删除。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 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 |

### HotelDataService模块

（1）模块概述

FinanceDataService模块将负责成本、收益的相关信息的读取、保存、修改。

（2）模块内部接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| 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 |

### OrderDataService模块

（1）模块概述

OrderDataService模块负责保存订单的相关信息，供读取、保存、修改。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 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 |

### PromotionDataService模块

（1）模块概述

PersonnelDataService模块用来保存人员和机构的信息，供读取、修改、写入。

（2）模块内部接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| 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 |

# 依赖视角

下图分别为客户端和服务器端各自的包之间的联系：

