快递物流管理系统

详细设计文档

V1.0正式版

AXIS工作组

南京大学软件学院

2015-11-16

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 陈自强 | 2015-11-2 | 建立整体框架，完成部分文档 | V0.1 |
| 余旻晨 | 2015-11-5 | Financebl模块好了作参考 | V0.2 |
| 陈自强 | 2015-11-5 | 完成Account、Approval、Personnel、Strategy、Usermes、Statistic六个模块 | V0.3 |
| 余旻晨 | 2015-11-6 | 完成Storebl模块 | V0.4 |
| 邢程 | 2015-11-7 | 完成Orderbl模块 | V0.5 |
| 张瑞 | 2015-11-16 | 完成DTmanage和transport模块，美化文档，统一字号 | V0.6 |
| 邢程 | 2015-11-16 | 完成ui层的模块架构 | V0.7 |
| 陈自强 | 2015-11-16 | 完善目录，修改排版 | V1.0 |
| 余旻晨 | 2015-12-30 | Bl层修改完成 | V1.1 |
| 张瑞 | 2015-12-30 | 根据实际情况修改UI层包划分 | V1.2 |

**目录**

[1 引言 3](#_Toc435478495)

[1.1 编制目的 3](#_Toc435478496)

[1.2 词汇表 3](#_Toc435478497)

[1.3 参考资料 3](#_Toc435478498)

[2 产品概述 3](#_Toc435478499)

[3 体系结构设计概述 4](#_Toc435478500)

[4 结构视角 4](#_Toc435478501)

[4.1 展示层的分解 4](#_Toc435478502)

[4.1.1 UI层设计概要 4](#_Toc435478503)

[4.1.2 UI层内部包的职责与划分 4](#_Toc435478504)

[4.2 业务逻辑层的分解 6](#_Toc435478505)

[4.2.1 account模块 6](#_Toc435478506)

[4.2.2 approval模块 11](#_Toc435478507)

[4.2.3 DTManage模块 16](#_Toc435478508)

[4.2.4 Finance模块 23](#_Toc435478509)

[4.2.5 order模块 28](#_Toc435478510)

[4.2.6 personnel模块 37](#_Toc435478511)

[4.2.7 statistic模块 43](#_Toc435478512)

[4.2.8 storebl模块 48](#_Toc435478513)

[4.2.9 strategy模块 53](#_Toc435478514)

[4.2.10 transport模块 57](#_Toc435478515)

[4.2.11 userMes模块 64](#_Toc435478516)

[4.3 数据层的分解 68](#_Toc435478517)

[4.3.1 DataFactoryService模块 68](#_Toc435478518)

[4.3.2 AccountDataService模块 69](#_Toc435478519)

[4.3.3 DTManageDataService模块 70](#_Toc435478520)

[4.3.4 FinanceDataService模块 71](#_Toc435478521)

[4.3.5 OrderDataService模块 72](#_Toc435478522)

[4.3.6 PersonnelDataService模块 73](#_Toc435478523)

[4.3.7 StatisticDataService模块 74](#_Toc435478524)

[4.3.8 StoreDataService模块 75](#_Toc435478525)

[4.3.9 StrategyDataService模块 76](#_Toc435478526)

[4.3.10 TransportDataService模块 77](#_Toc435478527)

[5 依赖视角 78](#_Toc435478528)

# 引言

## 编制目的

本报告详细完成对快递物流管理系统的详细设计，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** |
| ELMS | 快递物流管理系统 |
| loadDoc | 装车单 |
| arriveZZDoc | 到达单（营业厅和中转中心流转） |
| transferDoc | 中转单 |
| arriveYYDoc | 接收单 |
| sendGoodDoc | 派件单 |
| Bill | 期初账单 |
| order | 订单 |
| inStoreDoc | 入库单 |
| outStoreDoc | 出库单 |
| orderBarCode | 订单的条形码号 |
| DTManage | 司机车辆信息管理 |

## 参考资料

《快递物流管理系统用例文档V1.3》

《快递物流管理系统需求规格说明文档V1.4》

《快递物流管理系统体系结构设计文档V1.2》

# 产品概述

参考快递物流管理系统用例文档及快递物流管理系统规格说明文档对产品的概括描述。

xx物流企业是一本地的民营物流企业，主要希望对城市网点建设，扩大快运服务到达的片区。原只有财务及部分快运信息会使用软件处理，其他都是手工作业，这已经不能适应它的业务发展要求。首先是随着物流企业业务规模的不断扩大，单纯手工记录寄、收件，记录中转中心入出库的吞吐量等会导致物流信息的不通畅，进一步影响快递的物流中转速度，导致流失客源。其次是中转中心、营业厅的快递数量、种类众多，无法精确的掌握库内快递的具体情况，快递积压、损坏的现象明显上升。再次，快递企业众多，竞争比以前更大，希望在降低成本、吸引顾客、增强竞争力的同时，保持盈利水平。

快递物流管理系统就是为满足xx企业业务发展需求而开发的，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才是实时通信的方式完成数据交换。

# 体系结构设计概述

参考快递物流管理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）、快递员、营业厅业务员、财务人员、中转中心业务员、中转中心管理员、总经理、管理员。

Ui层中基本是每一种单据对应两个界面：增加和查询，虽然两种界面风格差不多，但是还有略有区别。其余的基本是一个功能一个类。

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

|  |  |  |
| --- | --- | --- |
| 包 | 包内项目 |  |
| common | FindSimpleOrderInfo | 普通人查询订单  （仅显示物流信息） |
| Courier  快递员 | AddOrder | 增加订单 |
| FindFullOrderInfo | 快递员查询订单  （显示完整订单信息） |
| InputReceiveInfo | 收件信息录入 |
| FinanceMan  财务人员 | BankAccountManage | 银行账户管理 |
| CostManage | 成本管理 |
| DepositeProfitAdd | 增加成本收益表 |
| FApproval | 交易审核 |
| lnitalFinance | 期初建账 |
| PayDocAdd | 增加付款单 |
| PayDocShow | 显示付款单 |
| GeneralManage  总经理 | ConstSet | 制定常量 |
| GMApproval | 审批单据 |
| InstManage | 机构管理 |
| PeopleManage | 人员管理 |
| SalaryStrategySet | 制定薪水策略 |
| Statistic | 统计分析 |
| Saleman  营业厅业务员 | ArriveYYDocAdd | 增加营业厅到达单 |
| ArriveYYDocShow | 显示营业厅到达单 |
| DriverInfoManage | 司机信息管理 |
| LoadDocAdd | 增加装车单 |
| LoadDocShow | 显示装车单 |
| PayDocAdd | 增加收款单 |
| PayDocShow | 显示收款单 |
| SendGoodDocShow | 显示派送单 |
| TruckInfoManage | 车辆信息管理 |
| Storeman  中转中心业务员 | ArriveZZDocAdd | 增加中转中心到达单 |
| ArriveZZDocShow | 显示中转中心到达单 |
| StoreShow | 库存查看 |
| TransferDocAdd | 增加中转中心流转单 |
| TransferDocShow | 显示中转中心流转单 |
| storeManager  中转中心管理员 | AlarmValueRecover | 报警恢复 |
| AlarmValueSet | 设置报警值 |
| InstoreDocAdd | 增加入库单 |
| InstoreDocShow | 显示入库单 |
| OutstoreDocAdd | 增加出库单 |
| outstoreDocShow | 显示出库单 |
| StoreCheck | 库存盘点 |
| User  管理员 | userManage | 管理员账户管理 |
| ／ | mainframe | 负责login，然后根据身份跳转 |
| Tool | MyLabel | 重写SWT自带的原生组件，使得更适合该系统，更美观 |
| MyButtom |
| MyTextField |
| MyTextArea |

## 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。

### account模块

1. 模块概述：

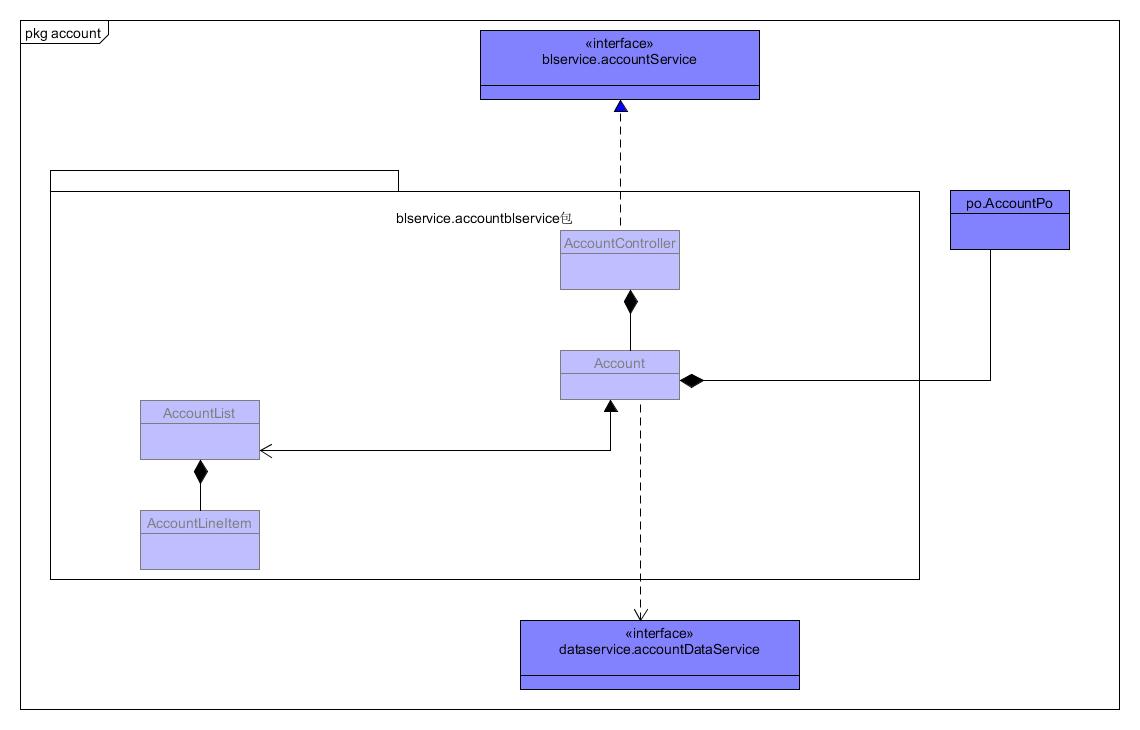
Account模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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 | |

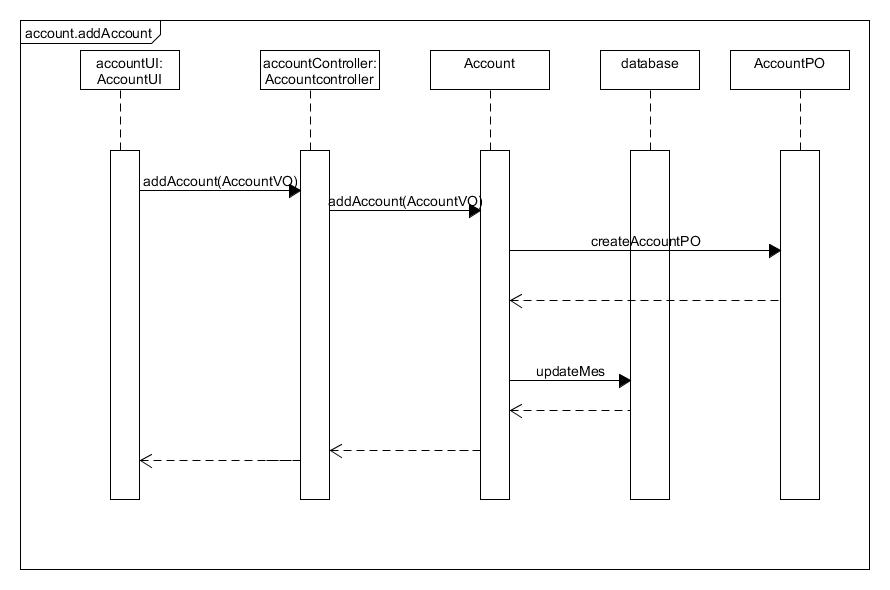
**Account的接口规范**

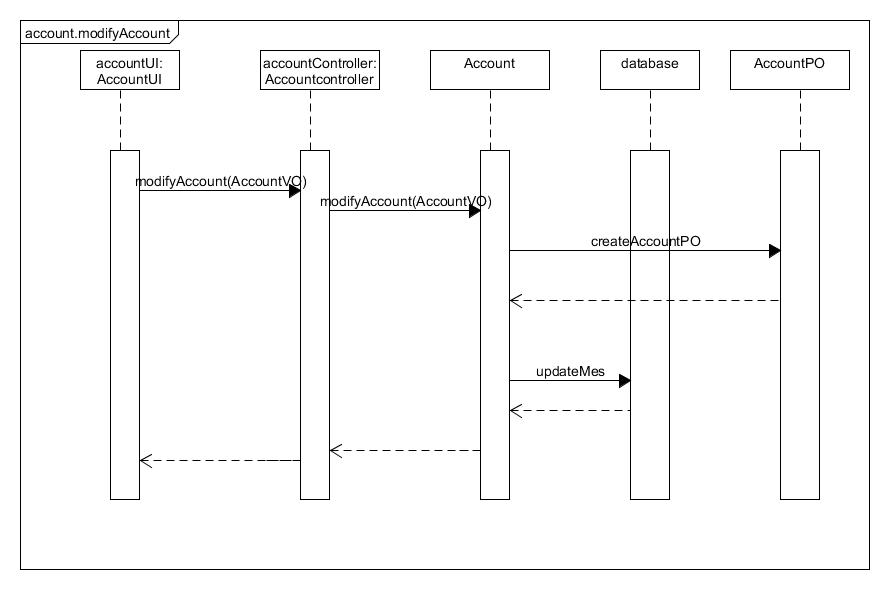
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Account.add | 语法 | Public ResultMessage add(AccountVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| Account.delete | 语法 | Public ResultMessage delete(AccountVO vo) |
| 前置条件 | 无（删除表明该数据必然是存在的） |
| 后置条件 | 系统永久性删除该账户的数据 |
| Account.find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

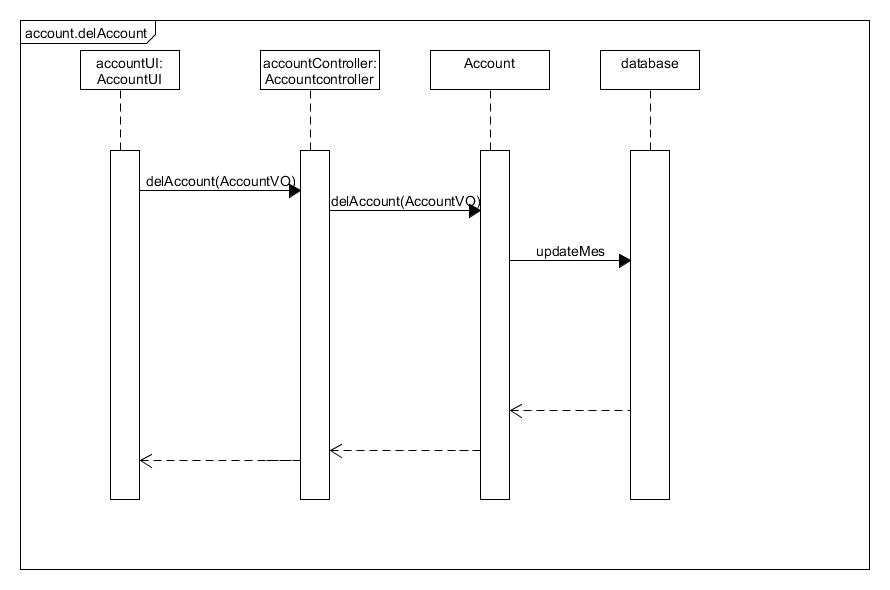
1. 业务逻辑层的动态模型

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

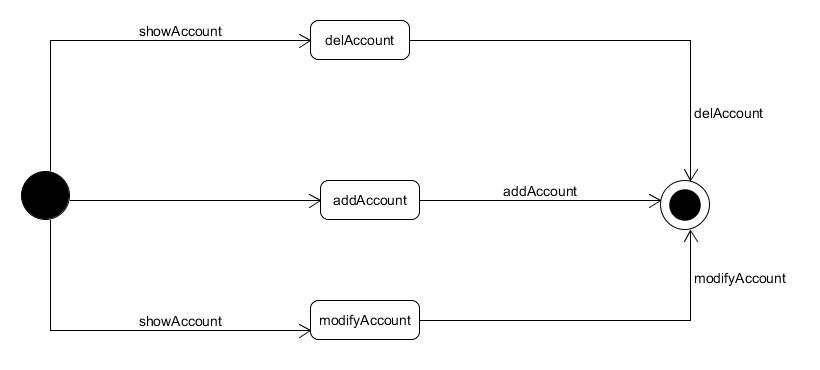
改账户信息的顺序图如下图所示：







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。

### approval模块

1、模块概述：

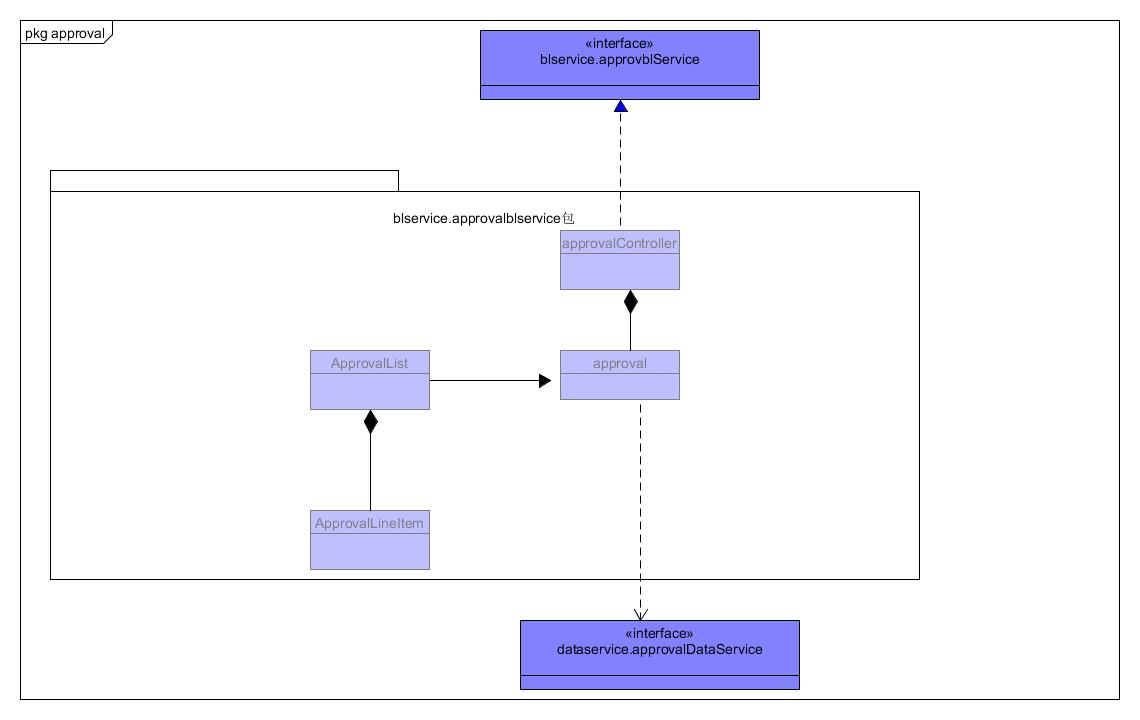
**Approval**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**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 | |

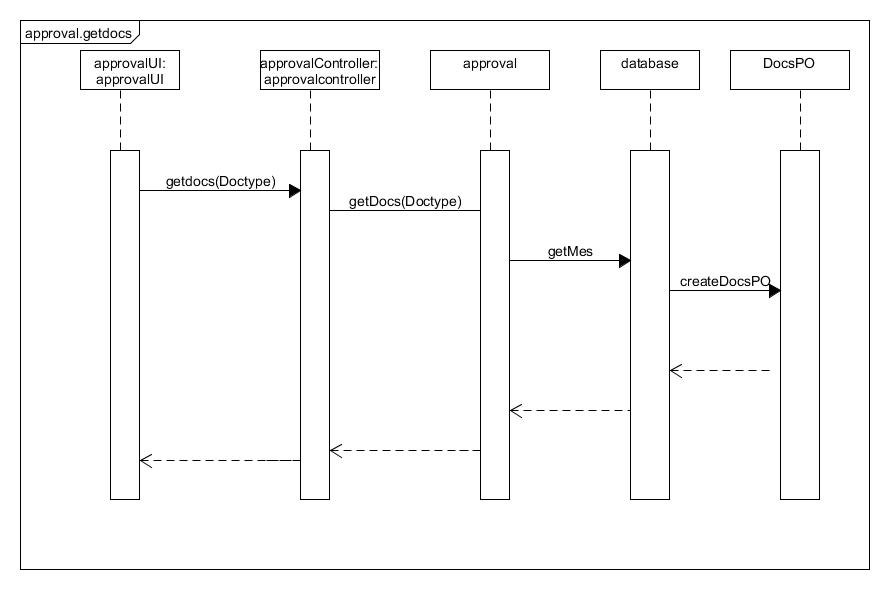
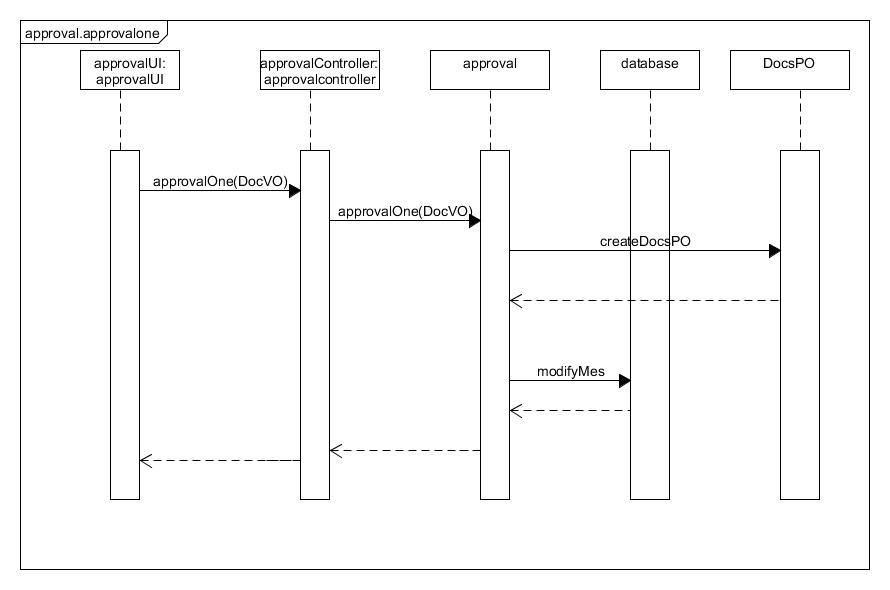
**approvalbl模块的接口规范**

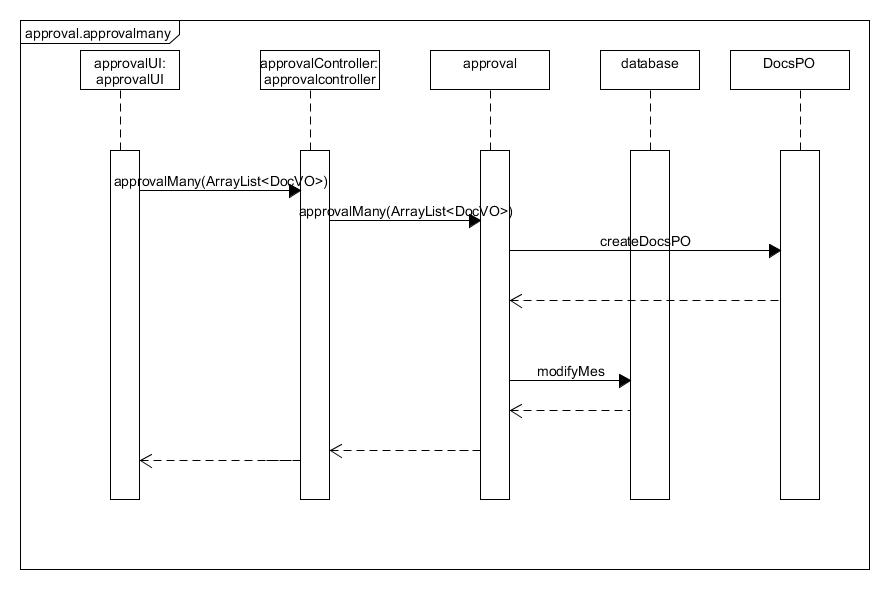
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| approval. getbills | | 语法 | Public ArrayList<DocVO> getbills(DocType type) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| Approval.approveone | | 语法 | Public ResultMessage approveone(DocVO vo) |
| 前置条件 | 该单据未审批过 |
| 后置条件 | 将该单据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

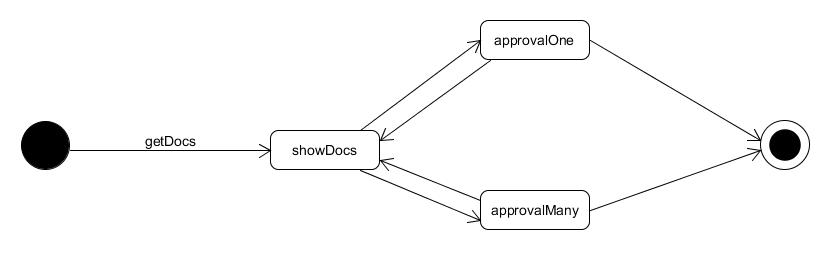
1. 业务逻辑层的动态模型

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

### DTManage模块

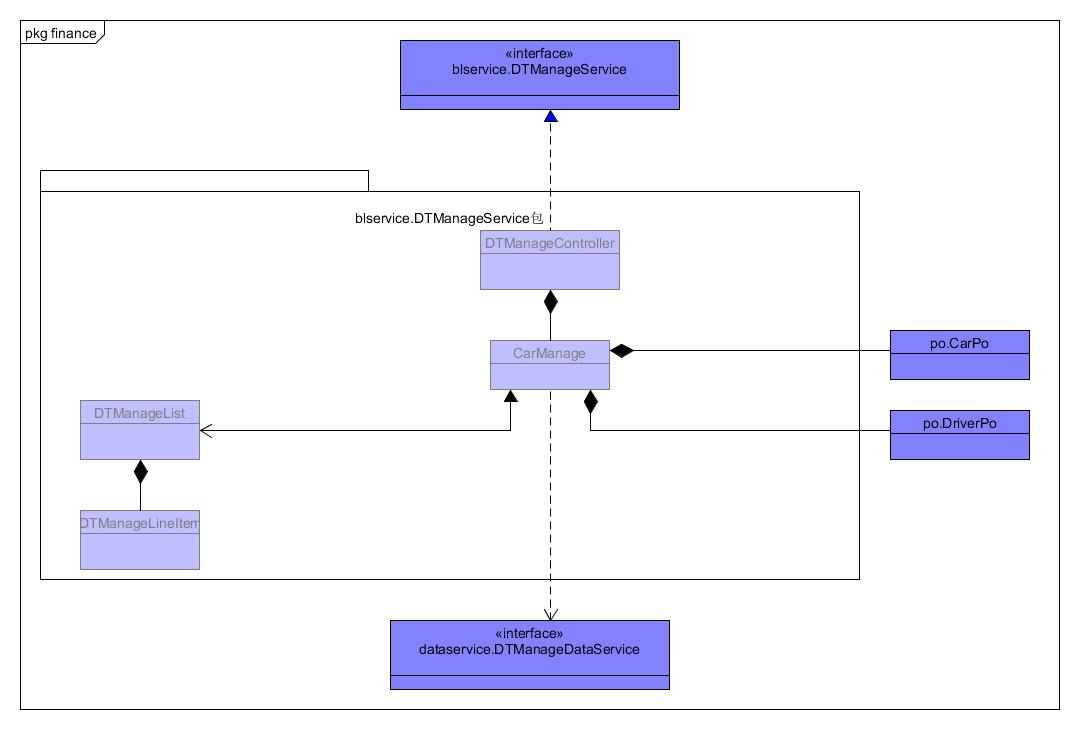
1、模块概述：

**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**

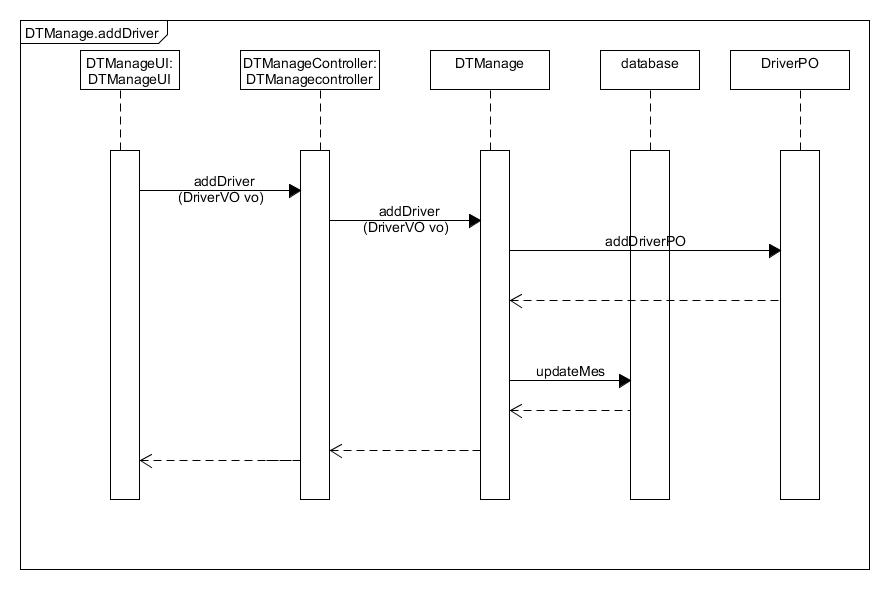
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DTManage.AddDriver | 语法 | Public ResultMessage addDriver  (DriverVO vo) |
| 前置条件 | 输入格式符合输入规范 |
| 后置条件 | 系统自动增加该司机信息并持久化更新涉及的对象的数据 |
| DTManage.CheckDriverbyName | 语法 | Public ArrayList< DriverVO >  CheckDriverByName(String name) |
| 前置条件 | 无 |
| 后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

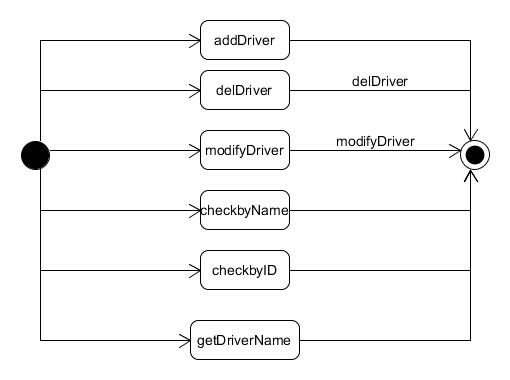
在司机车辆信息管理业务中，分为成本管理，生成收款信息，生成付款信息和利润查看功能。

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**

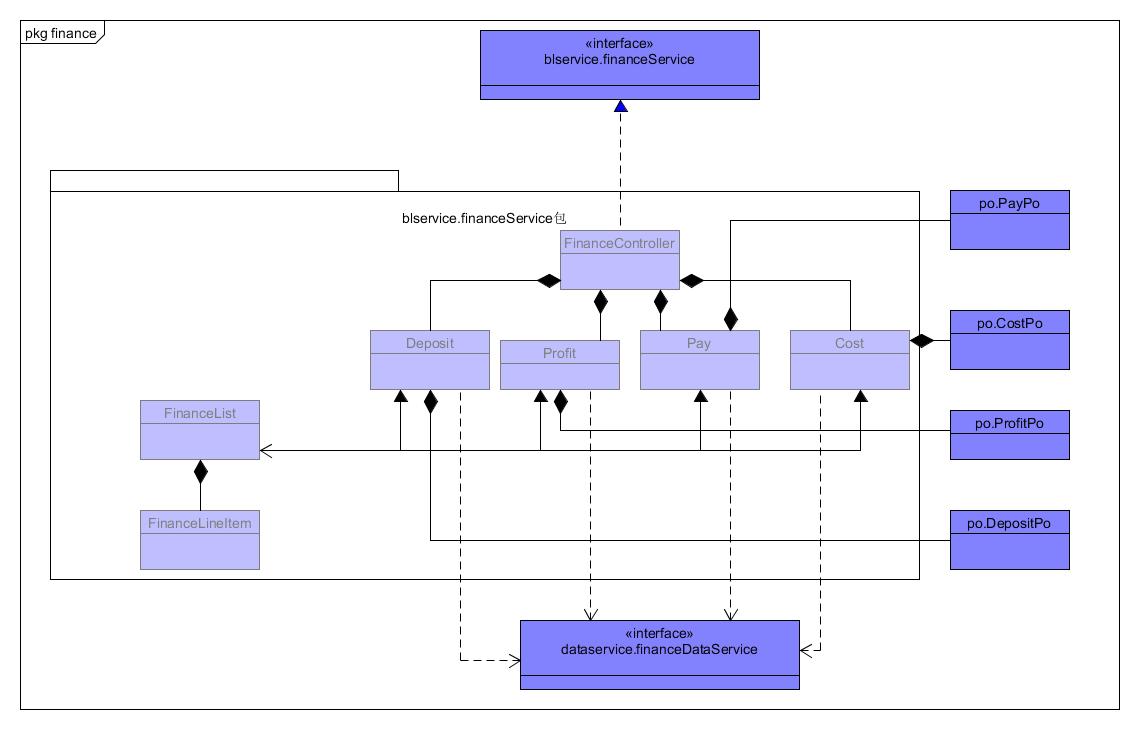
### Finance模块

1、模块概述：

**finance**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。

**finance**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FinanceblService接口。在业务逻辑层和数据层之间添加FinancedataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加FinanceController，这样FinanceController会将对商品管理的业务逻辑处理委托给Finance对象。FinancePO是作为财务信息的持久性对象被添加到设计模型中去的。

**图4-1-4-1 Financebl模块各个类的设计**

Financebl模块各个类的职责如表4-1-1-1所示。

**financebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| FinanceController | 实现财务部分需要的服务 |
| Cost | 实现成本的信息管理 |
| Pay | 实现付款的信息生成 |
| Profit | 实现收益的信息生成 |
| Deposit | 实现收款的信息生成 |

3、模块内部类的接口规范

FinanceController和Finance的接口规范如下所示

**FinanceController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceController.showDeposit | 语法 | Public ArrayLIst<DepositVO> showDeposit () |
| 前置条件 | 无 |
| 后置条件 | 返回DepositVO对象 |
| FinanceController.showCost | 语法 | Public ArrayLIst<CostVO> showDeposit () |
| 前置条件 | 无 |
| 后置条件 | 返回CostVO对象 |
| FinanceController.showPay | 语法 | Public ArrayLIst<PayVO> showDeposit () |
| 前置条件 | 无 |
| 后置条件 | 返回Pay VO对象 |
| FinanceController.createPay | 语法 | Public ResultMessage createPay () |
| 前置条件 | 无 |
| 后置条件 | 系统创建付款单 |
| FinanceController.createDeposit | 语法 | Public ResultMessage createDeposit () |
| 前置条件 | 无 |
| 后置条件 | 系统创建收款单 |
| FinanceController.addCost | 语法 | Public ResultMessage addCost () |
| 前置条件 | 无 |
| 后置条件 | 系统增加成本信息 |
| FinanceController.modifyCost | 语法 | Public ResultMessage modifyCost () |
| 前置条件 | 无 |
| 后置条件 | 系统修改成本信息 |
| FinanceController.delCost | 语法 | Public ResultMessage delCost () |
| 前置条件 | 无 |
| 后置条件 | 系统删除成本信息 |
| FinanceController.getProfit | 语法 | Public ArrayLIst<DepositVO> getProfit () |
| 前置条件 | 无 |
| 后置条件 | 系统返回profit信息 |

|  |  |  |
| --- | --- | --- |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| Finance. createPayVO(PayVO vo) | 生成付款信息 | |
| Finance. showCosts(CostType type) | 显示成本信息 | |
| Finance. addCostVO(CostVO vo) | 增加一条成本信息 | |
| Finance. modifyCostVO(CostVO vo) | 更新成本信息 | |
| Finance. delCostVO(CostVO vo) | 删除成本信息 | |
| Finance. showPays() | 显示付款信息 | |
| Finance. createDeposit(DepositVO vo) | 显示收款信息 | |
| Finance. showDeposit() | 显示收益信息 | |
| Finance. getAccounts() | 显示银行账户信息 | |
| Finance. deleteAccount(String ID) | 删除银行账户信息 | |
| Finance. addAccount(BankAccountVO vo) | 增加银行账户信息 | |
| Finance. checkAccount(String ID, int money) | 检查银行账户信息 | |
| Finance. modifyAccount(BankAccountVO vo) | 修改银行账户信息 | |
| Finance.getMessage() | 提供利润相关信息 | |

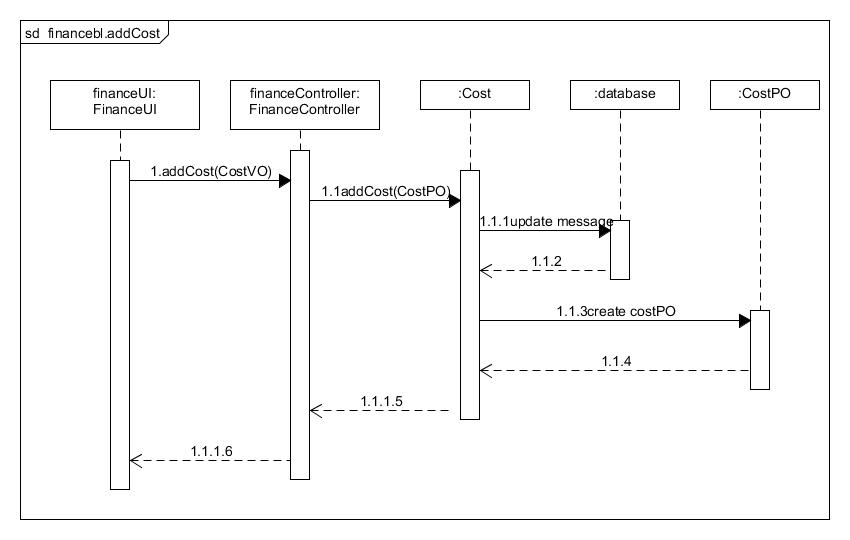
**financebl模块的接口规范**

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

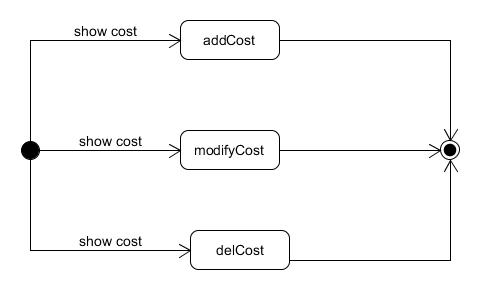
4、业务逻辑层的动态模型

在财务业务中，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4-1-4-2表明了对成本信息进行管理的相关对象之间的协作。



**图4-1-4-2成本信息进行管理**

下图所示的状态图描述了cost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。

**cost对象状态图**

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。

### order模块

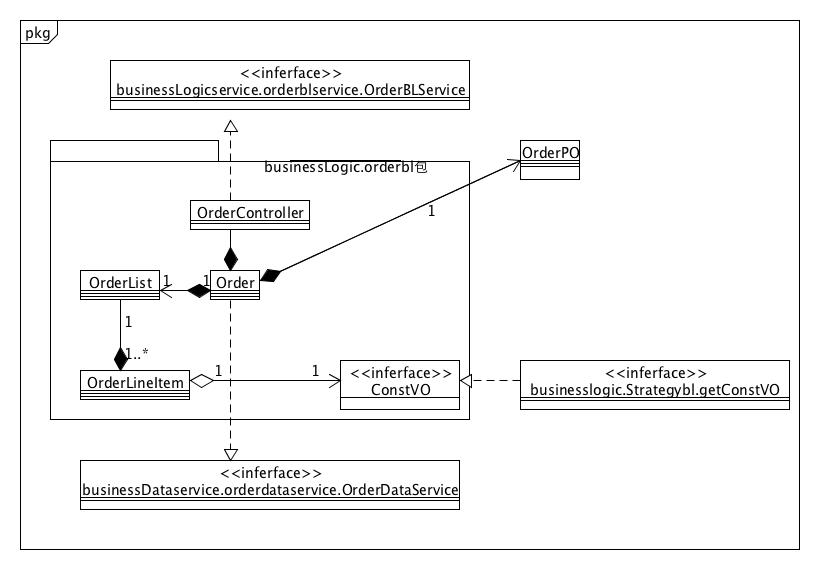
1、模块概述：

**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**

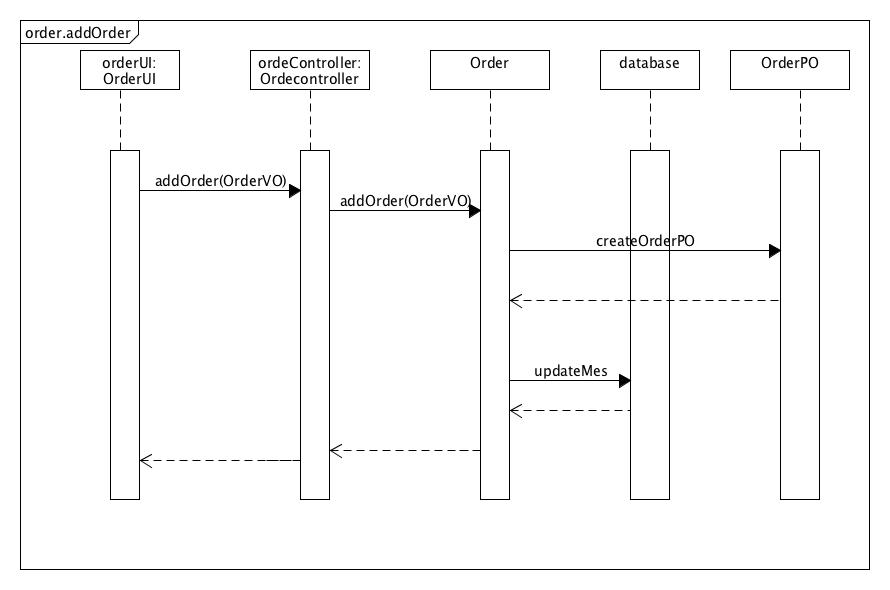
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderController.add | 语法 | Public ResultMessage add(orderVO orderVO) |
| 前置条件 | 已经创建一个order领域对象，并且输入符合语法规则 |
| 后置条件 | 调用Order类add方法 |
| OrderController.checkBarCode | 语法 | Public ResultMessage checkBarCode(String  orderBarCode) |
| 前置条件 | 在presentation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 | |

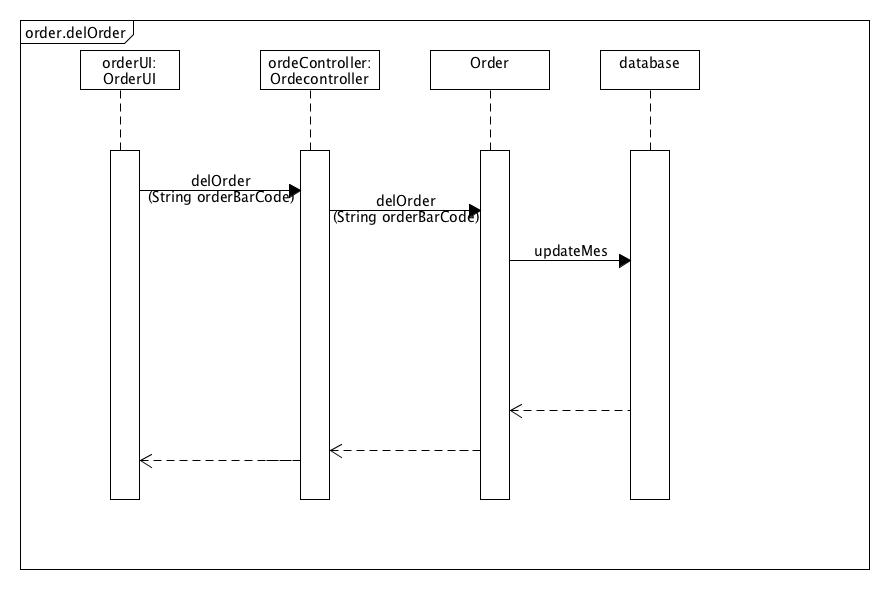
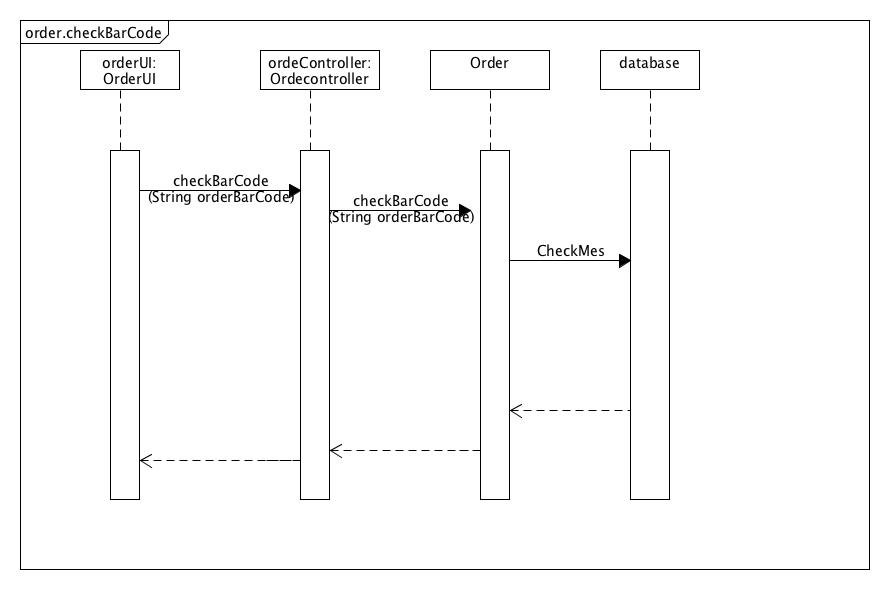
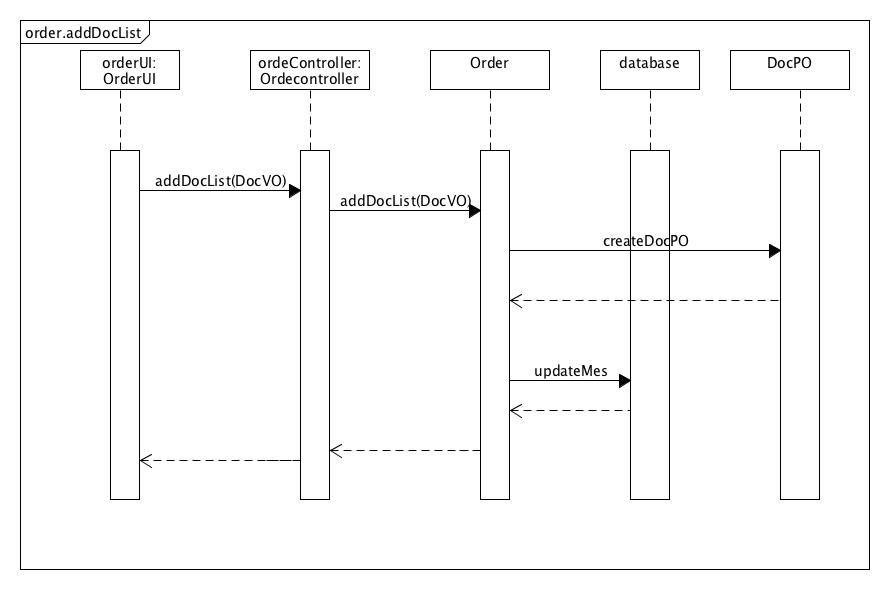
**orderbl模块的接口规范**

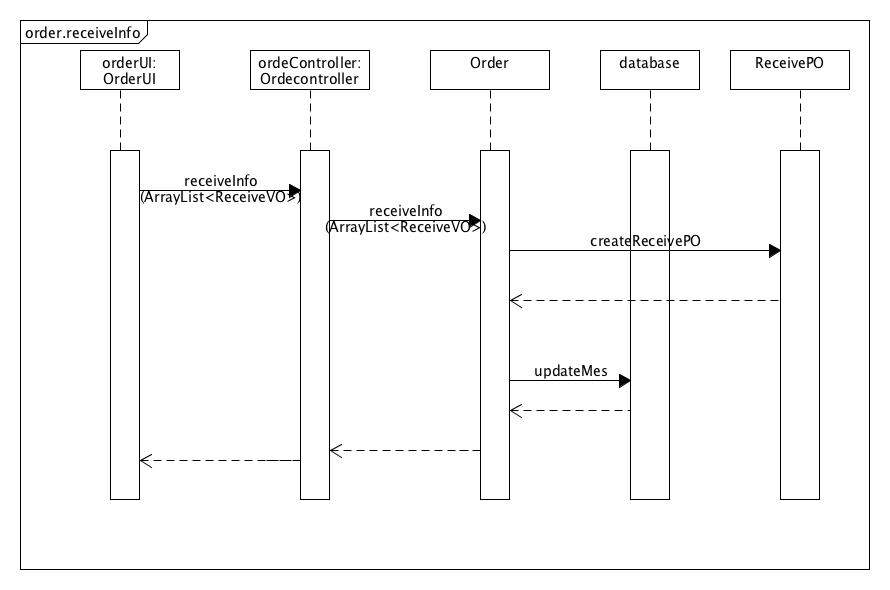
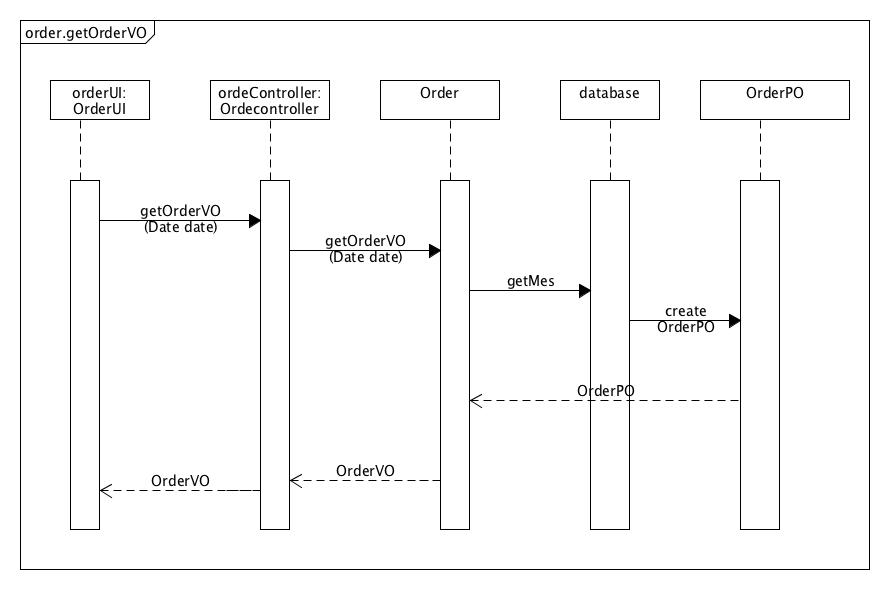
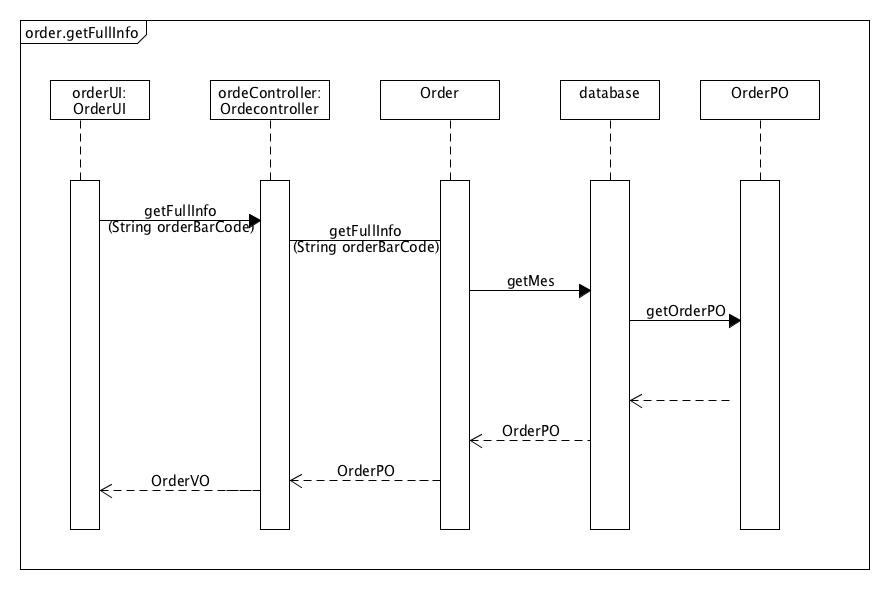
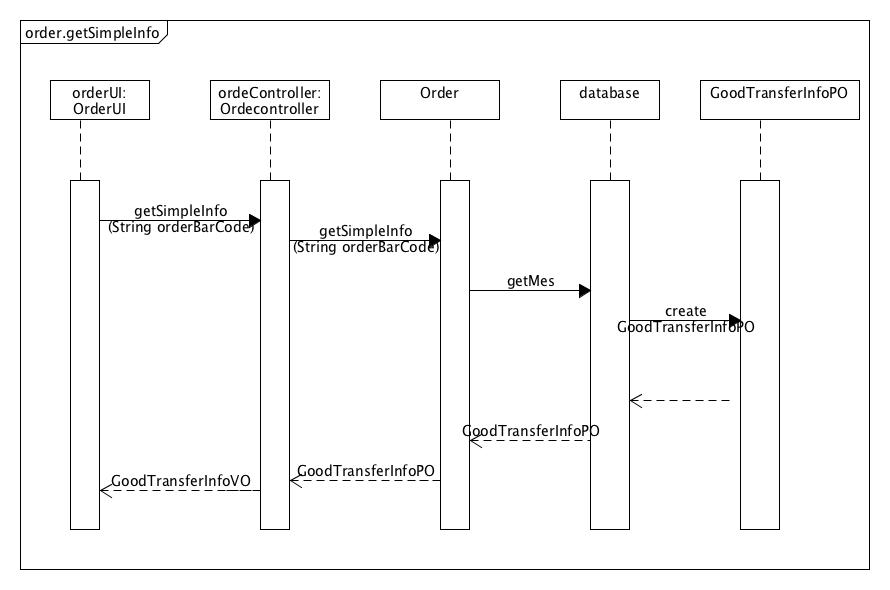
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.add | 语法 | Public ResultMessage add(orderVO orderVO) |
| 前置条件 | 输入的数据格式正确 |
| 后置条件 | 系统自动生成新的订单，并持久化添加数据 |
| Order.checkBarCode | 语法 | Public ResultMessage checkBarCode(String  orderBarCode) |
| 前置条件 | 在presentation层已经确认barCode（条形码）为10位 |
| 后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

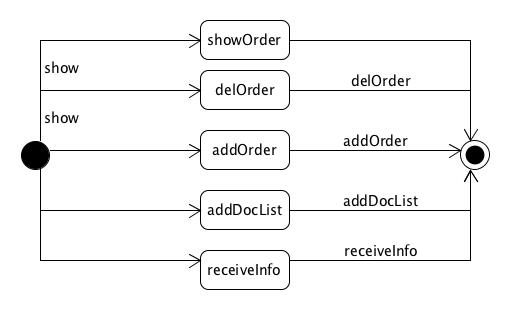
在订单管理中，业务分为增加订单、增加流转信息、检查条形码号、删除订单、获得完整订单信息、获得简略订单信息、获得orderpo，收件信息录入，它们的顺序图如下图所示：







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。

### personnel模块

1、模块概述：

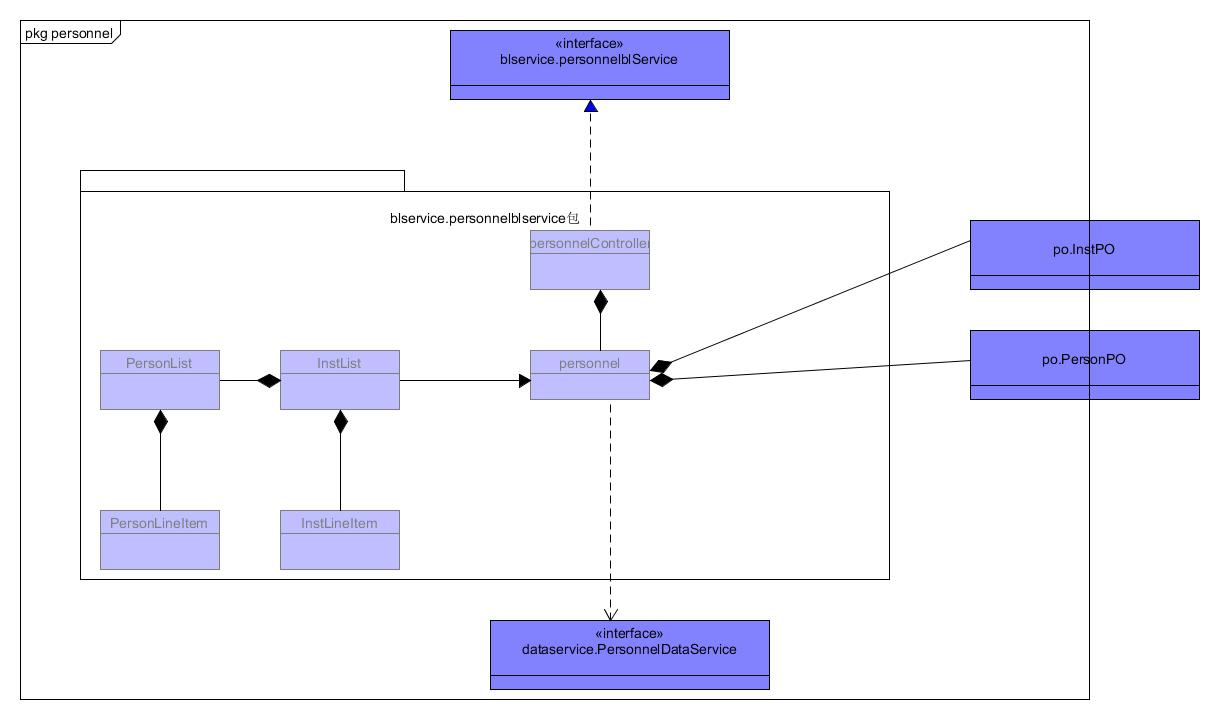
**personnel**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**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

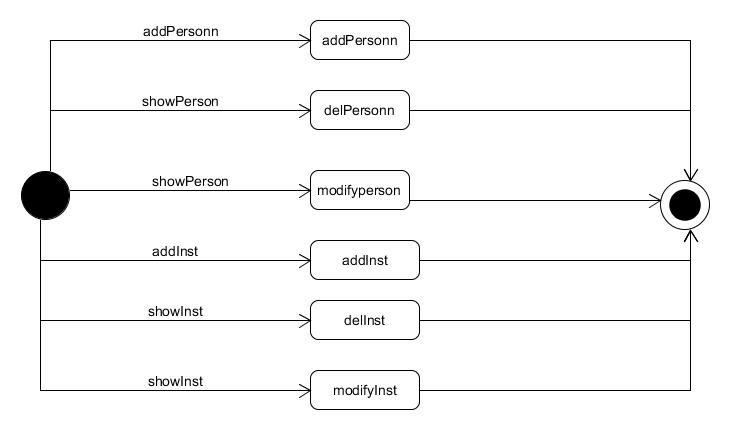
在人员机构管理管理中，业务分为增加、删除、修改、查找人员及机构信息，其中增加、删除、修改人员信息的顺序图如下图所示，机构的管理与此类似：







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。

### statistic模块

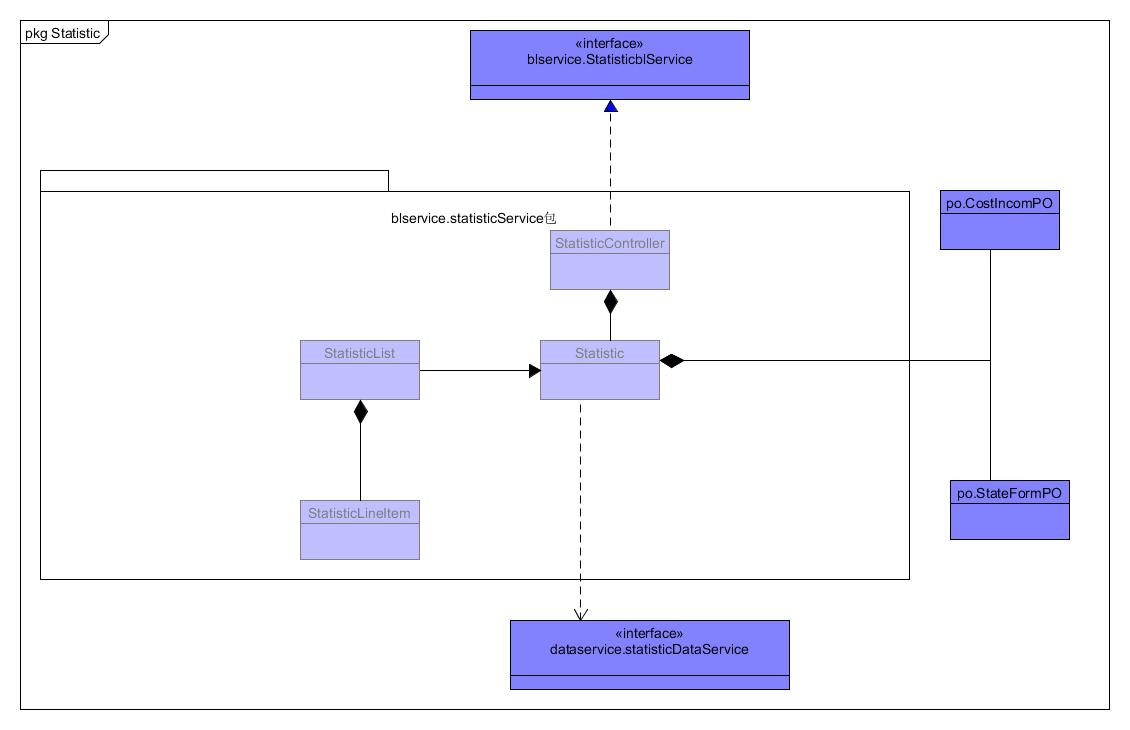
1、模块概述：

**statistic**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。

**statistic**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StatisticblService接口。在业务逻辑层和数据层之间添加StatisticdataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加StatisticController，这样StatisticController会将对统计分析的业务逻辑处理委托给Statistic对象。CostInComePO、StateFormPO是作为经营状况表、成本收益表的持久性对象被添加到设计模型中去的。

Statistic模块内部类的设计：

Statistic模块类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| StatisticController | 负责对应于统计分析所需要的服务 |
| Statistic | 统计分析的领域模型对象，拥有成本收益表、经营状况表的信息，可以帮助完成审批单据所需要的服务 |
| StatisticList | 负责保存统计分析表的信息 |
| StatisticLineItem | 统计分析表的条目信息 |

3、模块内部类的接口规范

**StatisticController类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StatisticController.getstateForm | 语法 | Public ArrayList<StateFormVO>  getstateForm () |
| 前置条件 | 已经创建一个Statistic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Statistic类的getstateForm方法，系统返回所有的经营状况表 |
| StatisticController.getCostIncomeForm | 语法 | Public ArrayList<CostIncomeFormVO>  getIncomeForm() |
| 前置条件 | 已经创建一个Statistic领域对象 |
| 后置条件 | 调用Statistic类的getCostIncomeForm方法，系统返回所有的成本收益表 |
| StatisticController.bulidStateForm | 语法 | Public ResultMessage  bulidStateForm(StateFormVO vo) |
| 前置条件 | 已经创建一个Statistic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Statistic类的bulidStateForm方法，系统生成并储存该表 |
| StatisticController.bulidCostIncomeForm | 语法 | Public ResultMessage bulidCostIncomeForm  (CostIncomeFormVO vo) |
| 前置条件 | 已经创建一个Statistic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Statistic类的bulidCostIncomeForm方法，系统生成并存储该表 |
| StatisticController.bulidbill | 语法 | Public ResultMessage bulidBill(BillVO vo) |
| 前置条件 | 已经创建一个Statistic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Statistic类的bulidbill方法，系统永久储存 |
| StatisticController.getbill | 语法 | Public ArrayList< BillVO > getIBills() |
| 前置条件 | 已经创建一个Statistic领域对象 |
| 后置条件 | 调用Statistic类的getbill方法，系统返回所有的账单 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| statsitic.getstateForm () | 返回所有经营状况表 | |
| statsitic.getCostIncomeFForm () | 返回所有的成本收益表 | |
| statsitic.bulidStateForm(StateFormVO vo) | 永久储存经营状况表 | |
| statsitic.bulidCostIncomeForm(CostIncomeFormPO po) | 永久储存成本收益表 | |
| statsitic.bulidbill | 永久储存账单 | |
| statsitic.getbill | 获得所有账单 | |
| Financebl.getPays | 获得付款单 | |
| Financebl.getProfit | 获得利润 | |
| Financebl.getdepositPO | 获得存款信息 | |

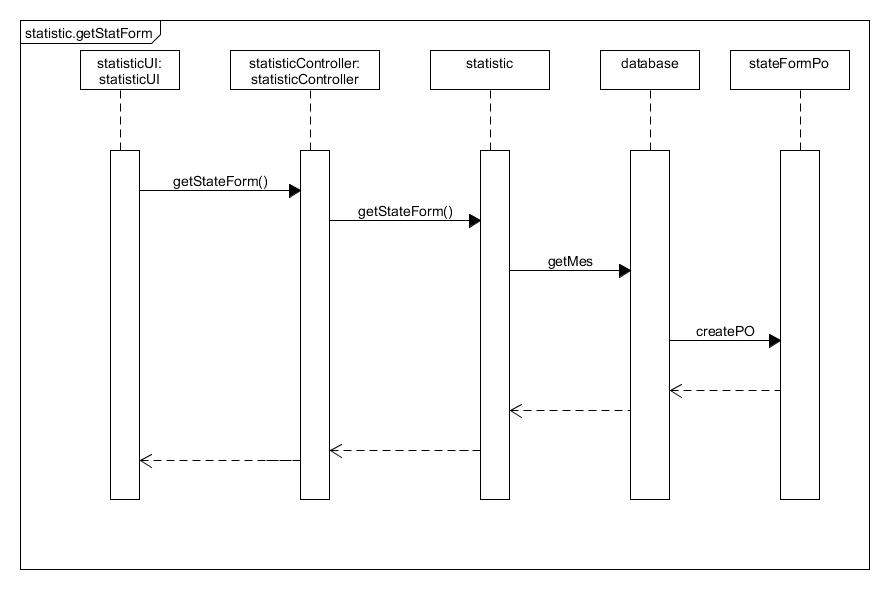
**statisticbl模块的接口规范**

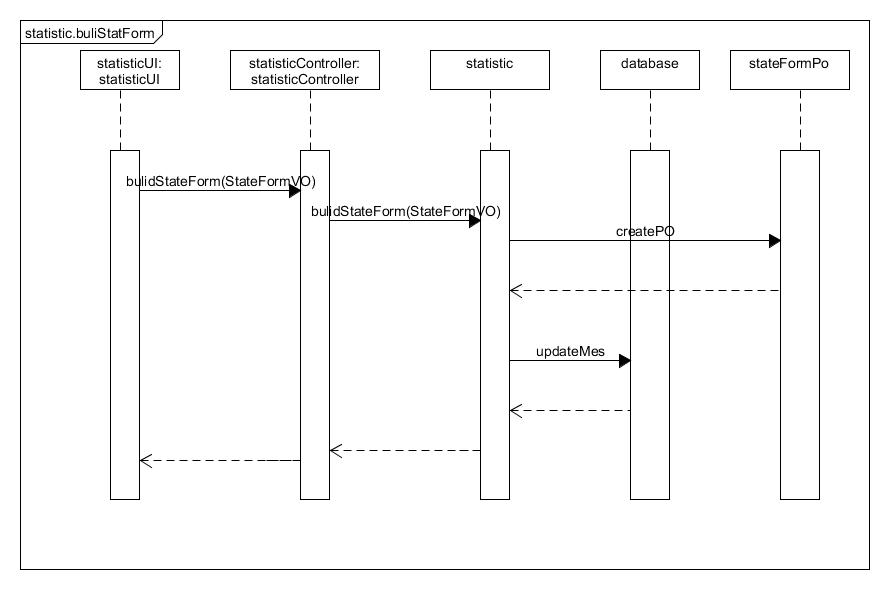
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Statisticbl.getstateForm | 语法 | Public ArrayList<StateFormVO>  getstateForm () |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的经营状况表 |
| Statisticbl. getIncomeForm | 语法 | Public ArrayList<CostIncomeVO>  getIncomeForm() |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有的成本收益表 |
| Statisticbl.bulidStateForm | 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 | |

1. 业务逻辑层的动态模型

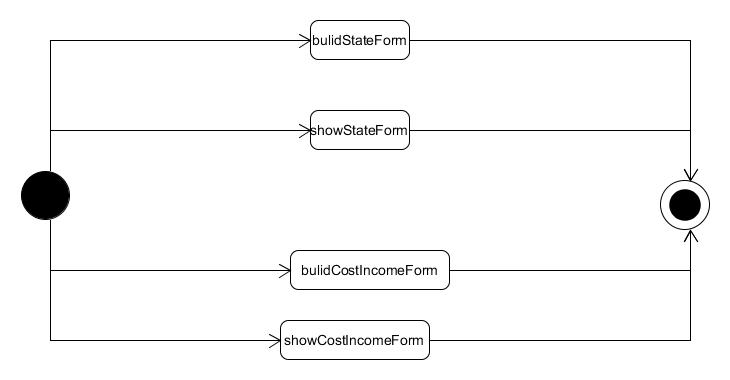
在统计分析中，业务分为对经营状况表、成本收益表的生成和查看，其中成本收益表的

查看、生成的顺序图如下图所示：





下图所示的状态图描述了Statistic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。

### storebl模块

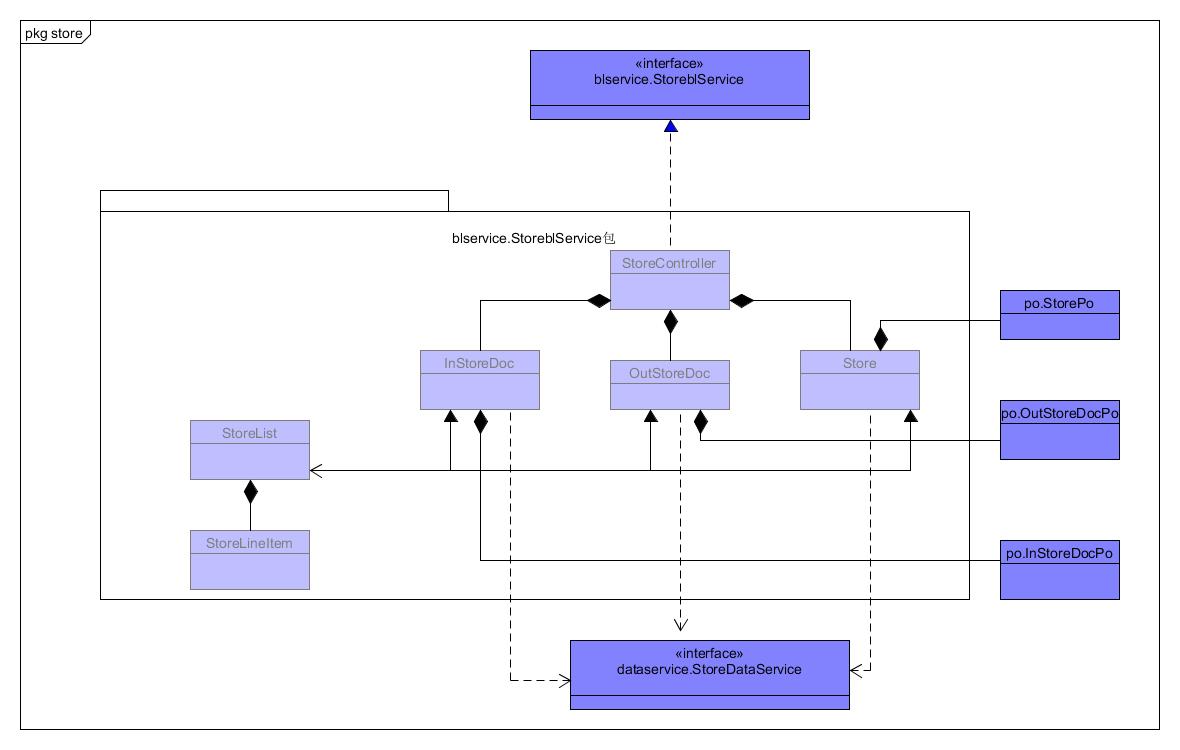
1、模块概述：

**storebl**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。

**storebl**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StoreblService接口。在业务逻辑层和数据层之间添加StoreblService接口。为了隔离业务逻辑职责和逻辑控制职责，添加StoreblController，这样StoreblController会将对商品管理的业务逻辑处理委托给StoreService对象。StorePO是作为财务信息的持久性对象被添加到设计模型中去的。



**图4-1-8-1 Storebl模块各个类的设计**

Storebl模块各个类的职责如表4-1-8-1所示。

**Storebl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| StoreController | 实现库存部分需要的服务 |
| Store | 实现库存的信息管理 |
| InStoreDoc | 实现货物入库 |
| OutStoreDoc | 实现货物出库 |

3、模块内部类的接口规范

StoreController和Store的接口规范如下表所示

**StoreController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StoreController.showInStoreDoc | 语法 | Public ArrayLIst<InStoreDocVO>  showInStoreDoc () |
| 前置条件 | 无 |
| 后置条件 | 返回InStoreDocVO对象 |
| StoreController.showOutStoreDoc | 语法 | Public ArrayLIst<OutStoreDocVO>  showOutStoreDoc () |
| 前置条件 | 无 |
| 后置条件 | 返回OutStoreDocVO对象 |
| StoreController.generateInStoreDoc | 语法 | Public ResultMessage  generateInStoreDoc (InStoreDocVO) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |
| StoreController.generateOutStoreDoc | 语法 | Public ResultMessage  generateOutStoreDoc (OutStoreDocVO) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |
| StoreController.showStoreMessage | 语法 | Public ArrayLIst<StoreMessageVO>  showStoreMessage () |
| 前置条件 | 无 |
| 后置条件 | 返回StoreMessageVO对象 |
| StoreController.update | 语法 | Public ResultMessage  update (StoreMessageVO) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |
| StoreController. exportExcel | 语法 | Public ResultMessage  exportExcel (String path) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |
| StoreController.setAlarmValue | 语法 | Public ResultMessage  setAlarmValue (String value) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| InStoreDoc.generate | 创建入库单对象信息 |
| OutStoreDoc.generate | 创建出库单对象信息 |
| InStoreDoc.show | 显示InStoreDoc对象信息 |
| OutStoreDoc.show | 显示OutStoreDoc对象信息 |
| Store.show | 显示库存信息 |
| Store.update | 更新库存信息 |
| Store.setAlarmValue | 设置成本警戒值 |
| Store.getAlarmValue | 得到成本警戒值 |
| Store.exportExcel | 导出excel图表 |
| Store. getDocLists(DocType type) | 得到对应单据的单据列表 |
| Store.changeDocsState(ArrayList<String> docsID, DocType type, DocState state) | 改变相应单据的状态 |
| Store. changeOneDocState(String docID, DocType type, DocState state) | 改变相应单据的状态 |
| Store. getDayDocCount(DocType type) | 得到相应单据当天生成的数量 |
| Store. getByID(String ID , DocType type) | 得到单据的详细信息 |

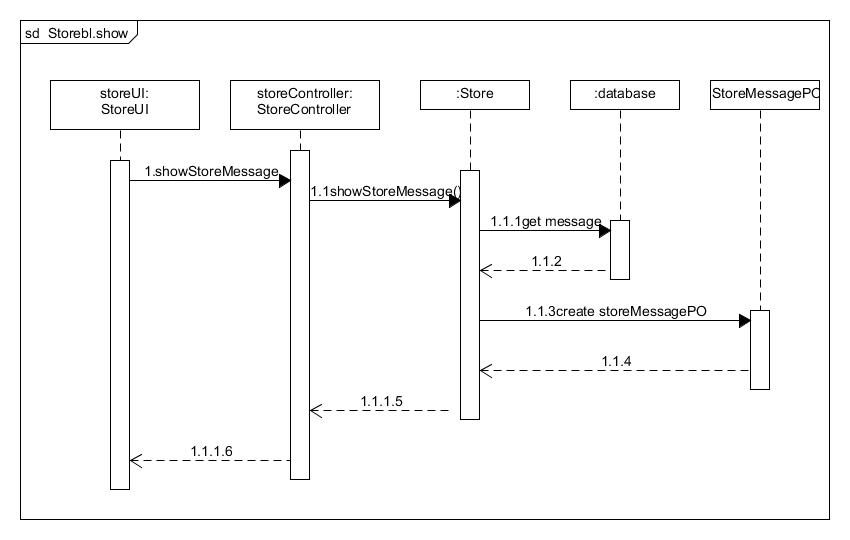
**storebl模块的接口规范**

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

4、业务逻辑层的动态模型

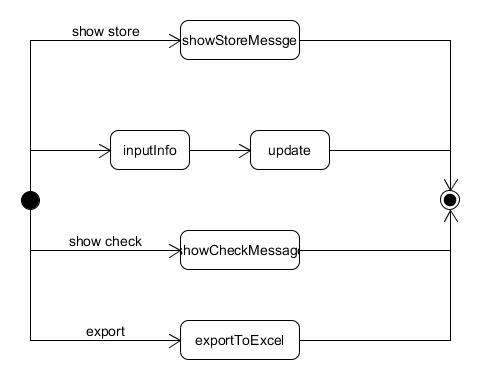
在库存业务中，分为库存管理，生成入库单，生成出库单和利润查看库存报警功能。

图4-1-8-2表明了对库存信息进行管理的相关对象之间的协作。



**图4-1-8-2库存信息进行管理**

图4-1-8-3 所示的状态图描述了store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4-1-8-3 store对象状态图**

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。

### strategy模块

1、模块概述：

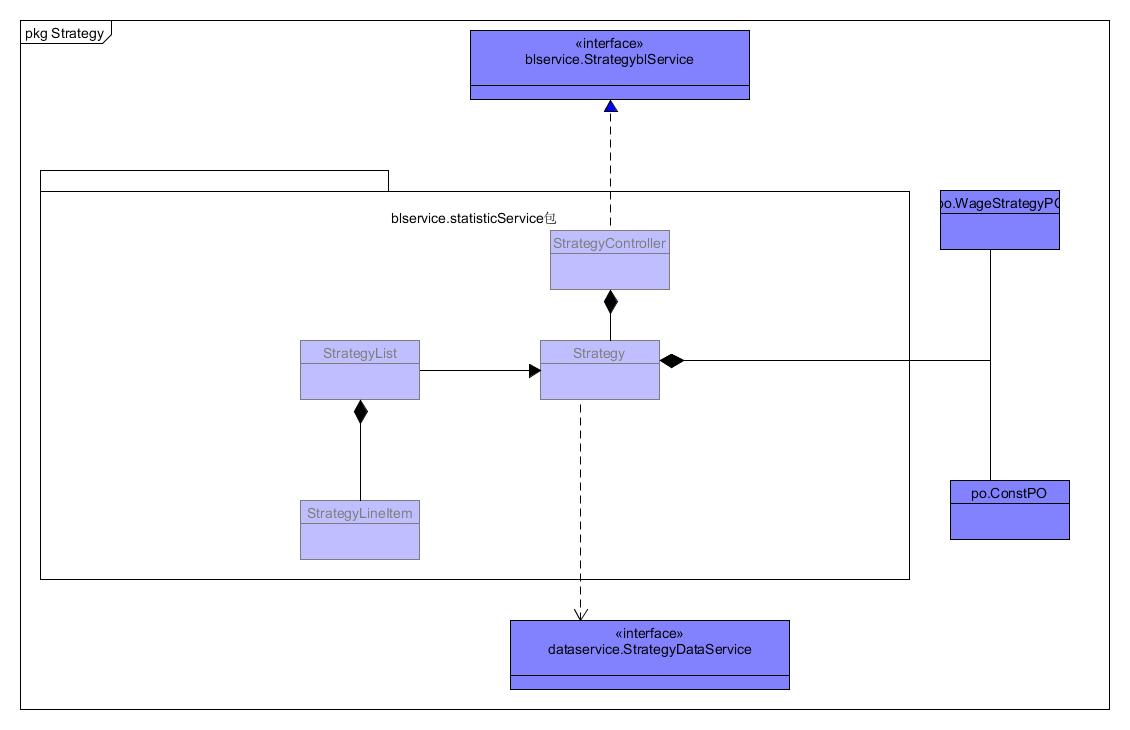
**strategy**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**strategy**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StrategyblService接口。在业务逻辑层和数据层之间添加StrategydataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加StrategyController，这样StrategyController会将对薪水策略、制定常量的业务逻辑处理委托给Strategy对象。ConstPO、WageStrategyPO是作为账户信息的持久性对象被添加到设计模型中去的。

Strategy 内部类的设计：



|  |  |
| --- | --- |
| 模块 | 职责 |
| StrategyController | 负责对应于制定策略、常量所需要的服务 |
| Strategy | 策略制定的领域模型对象，拥有常量、薪水的信息，可以帮助完成制定常量、策略所需要的服务 |
| StrategyList | 负责保存常量、薪水策略的相关信息 |
| StrategyLineItem | 常量、薪水策略的相关条目信息 |

3、模块内部类的接口规范

**StrategyController类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyController.getConst | 语法 | Public ConstVO getConst(String const) |
| 前置条件 | 已经创建一个Strategy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Strategy类的getConst方法，返回所有常量 |
| StrategyController.setConst | 语法 | Public ResultMessage setConst(ConstVO vo) |
| 前置条件 | 已经创建一个Strategy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Strategy类的setConst方法，系统记录新常量 |
| StrategyController.getsalary | 语法 | Public ArrayList<SalaryWayVO> getsalary() |
| 前置条件 | 已经创建一个Strategy领域对象 |
| 后置条件 | 调用Strategy类的getsalary方法，返回所有薪水策略 |
| StrategyController.getOnesalary | 语法 | Public SalaryWayVO getOnesalary(StaffType  type) |
| 前置条件 | 已经创建一个Strategy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Strategy类的getOnesalary方法，返回该职员的薪水策略 |
| StrategyController.setsalary | 语法 | Public ResultMessage setSalary(SalaryWayVO  way) |
| 前置条件 | 已经创建一个Strategy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Strategy类的setsalary方法，永久修改薪水策略 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Strategy. getConst | 从数据库中读取常量 | |
| Strategy.setConst | 设置常量 | |
| Strategy.getsalary | 从数据库中获得所有职员的薪资策略 | |
| Strategy.getOnesalary | 从数据库中获得某一职员的薪资策略 | |
| Strategy.setsalary | 设置某一职员的薪资策略 | |
| Strategy. getEstiDateVO | 得到货物预计到达的时间 | |
| Strategy. setEstiDateVO(EstiDateVO vo) | 修改预计到达的时间 | |

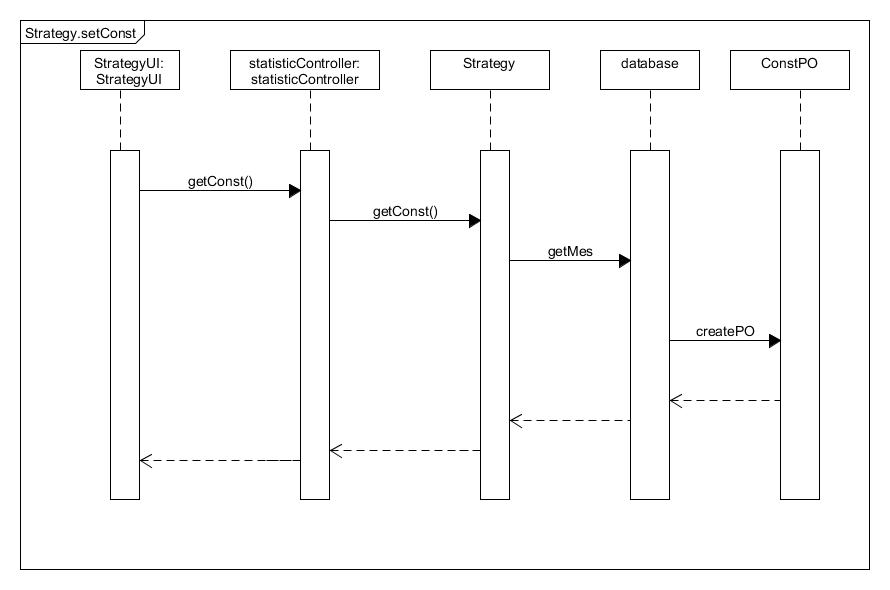
**Strategy模块的接口规范**

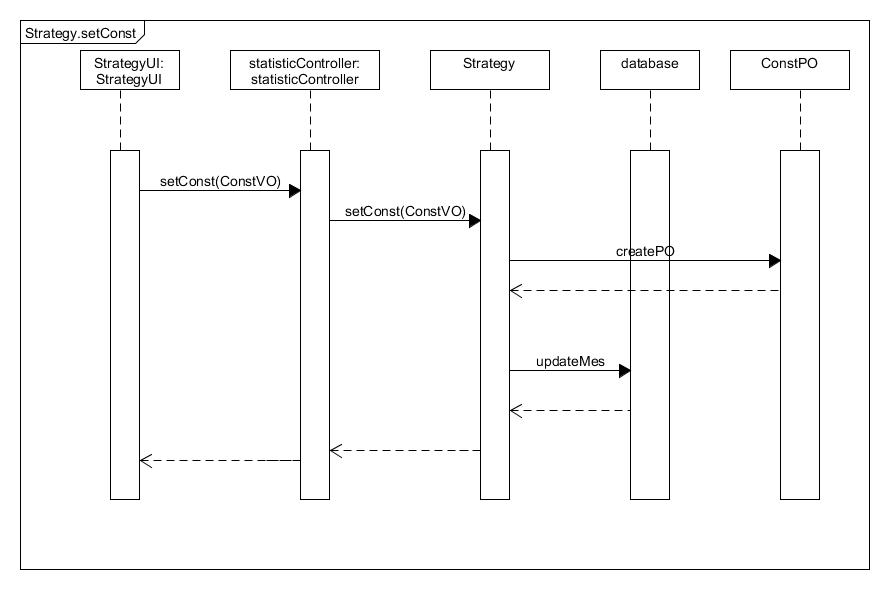
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Strategy.getConst | 语法 | Public ConstVO getConst(String const) |
| 前置条件 | 无 |
| 后置条件 | 返回所有常量 |
| Strategy.setConst | 语法 | Public ResultMessage setConst(ConstVO vo) |
| 前置条件 | 常量符合语法规则 |
| 后置条件 | 系统记录新常量 |
| Strategy.getsalary | 语法 | Public ArrayList<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 | |

4、业务逻辑层的动态模型

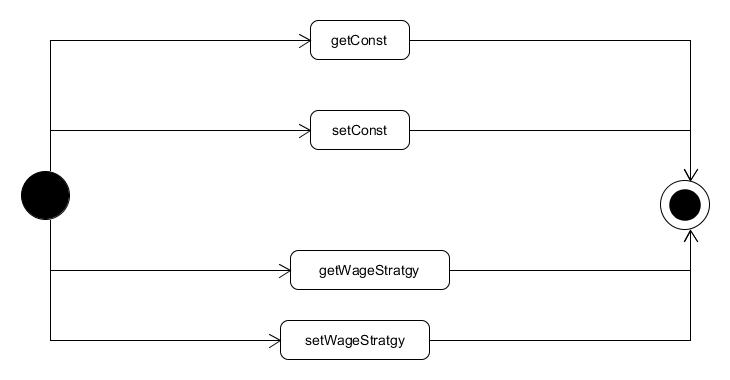
在制定常量策略中，业务分为制定修改常量，制定修改薪水策略，其中增加、修

改系统常量的顺序图如下图所示：





下图所示的状态图描述了Strategy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。

### transport模块

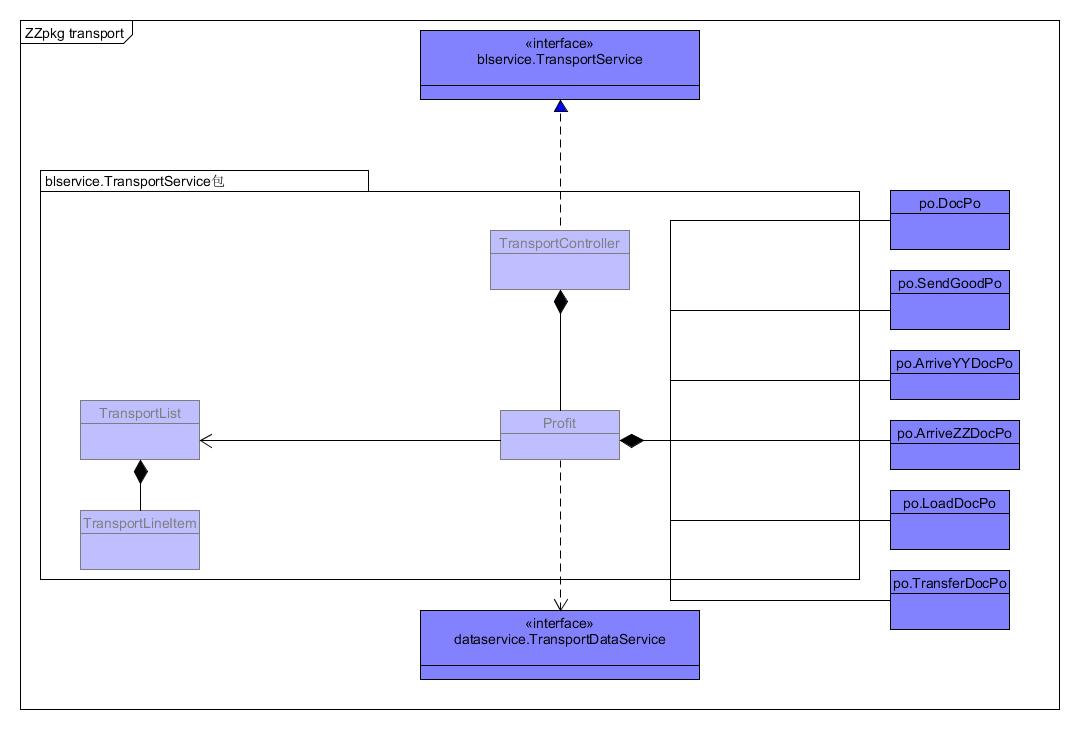
1、模块概述：

**transportbl**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。

**transportbl**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TransportblService接口。在业务逻辑层和数据层之间添加TransportblService接口。为了隔离业务逻辑职责和逻辑控制职责，添加TransportblController，这样TransportblController会将对商品管理的业务逻辑处理委托给TransportService对象。DocPO是作为流转信息的持久性对象被添加到设计模型中去的。



**图4-1-10-1Transportbl模块各个类的设计**

Transportbl模块各个类的职责如表4-1-10-1所示。

**表4-1-10-1** **Transportbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| TransportController | 实现流转部分需要的服务 |
| Transport | 实现流转过程中产生的单据的管理 |

3.模块内部类的接口规范

TransportController和Transport的接口规范如下表所示

**TransportController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportController.AddLoadDoc | 语法 | Public ResultMessage.addLoadDoc  (LoadDocVO vo) |
| 前置条件 | 已经创建一个transport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transport类，将新的账户添加到数据库中，持久化更新相关数据 |
| TransportController. getDayLoadDocs | 语法 | Public ArrayList< LoadDocVO >  getDayLoadDocs(Date date) |
| 前置条件 | 已经创建一个transport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transport类的getDayLoadDocs方法，返回当日所有装车单信息 |
| TransportController.AddSendGoodDoc | 语法 | Public ResultMessage addsendGoodDoc  (SendGoodDocVO vo) |
| 前置条件 | 已经创建一个transport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transport类，将新的账户添加到数据库中，持久化更新相关数据 |
| TransportController.getDay SendGoodDocs | 语法 | Public ArrayList< GoodDocVO >  getDaySendDocs(Date date) |
| 前置条件 | 已经创建一个transport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transport类的getDayLoadDocs方法，返回当日所有装车单信息 |
| TransportController.AddArriveYYDoc | 语法 | Public ResultMessage.addArriveYYDoc  (ArriveYYDocVO vo) |
| 前置条件 | 已经创建一个transport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transport类，将新的账户添加到数据库中，持久化更新相关数据 |
| TransportController.getDayArriveYYDocs | 语法 | Public ArrayList<ArriveYYDocVO>  getDayArriveYYDocs(Date date); |
| 前置条件 | 已经创建一个transport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transport类的getDayArriveYYDocs方法，返回当日所有装车单信息 |
| TransportController.AddArriveZZDoc | 语法 | Public ArriveZZDoc.addArriveZZDoc  (ArriveZZDocVO vo) |
| 前置条件 | 已经创建一个transport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transport类，将新的账户添加到数据库中，持久化更新相关数据 |
| TransportController.getDayArriveZZDocs | 语法 | Public ArrayList<ArriveZZDocVO>  getDayArriveZZDocs(Date date); |
| 前置条件 | 已经创建一个transport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transport类的getDayArriveZZDocs方法，返回当日所有装车单信息 |
| TransportController.Add TransferDoc | 语法 | Public TransferDoc.addTransferDoc  (TransferDocVO vo) |
| 前置条件 | 已经创建一个transport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transport类，将新的账户添加到数据库中，持久化更新相关数据 |
| TransportController.getDayTransferDocs | 语法 | Public ArrayList<TransferDocVO>  getDayTransferDocs(Date date); |
| 前置条件 | 已经创建一个transport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transport类的getDayTransferDocs方法，返回当日所有装车单信息 |
| TransportController.getDoc | 语法 | Public ArrayList<DocVO >  getDoc(DocType type) |
| 前置条件 | 已经创建一个transport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transport类的getDoc方法，从货物流转包中获得单据 |
| TransportController.getExpense | 语法 | Public double getExpense(DocType type) |
| 前置条件 | 已经创建一个transport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transport类的getDoc方法，获得该单据的运费成本信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Transport.AddLoadDoc | 增加一个装车单 | |
| Transport. getDayLoadDocs | 获得当天的装车单信息 | |
| Transport.AddSendGoodDoc | 增加一个派送单 | |
| Transport.getDay SendGoodDocs | 获得当天的派送单信息 | |
| Transport.AddArriveYYDoc | 增加一个接收单 | |
| Transport.getDayArriveYYDocs | 获得当天的接收单信息 | |
| Transport.AddArriveZZDoc | 增加一个到达单 | |
| Transport.getDayArriveZZDocs | 获得当天的到达单信息 | |
| Transport.Add TransferDoc | 增加一个中转单 | |
| Transport.getDayTransferDocs | 获得当天的中转单信息 | |
| Transport.getDoc | 从货物流转包中获得单据 | |
| Transport.getExpense | 获得该单据的运费成本信息 | |
| Transport.getDocLists(DocType type) | 得到对应单据的单据列表 | |
| Transport.changeDocsState(ArrayList<String> docsID, DocType type, DocState state) | 改变相应单据的状态 | |
| Transport.changeOneDocState(String docID, DocType type, DocState state) | 改变相应单据的状态 | |
| Transport.getDayDocCount(DocType type) | 得到相应单据当天生成的数量 | |
| Transport. getByID(String ID , DocType type) | 得到单据的详细信息 | |
| Transport. addOnePay(PayDocVO vo) | 增加一个付款单 | |
| Transport. getPays() | 获得当日所有付款单 | |

**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 | |

4、业务逻辑层的动态模型

在流转业务中，分为生成单据和查看单据两种功能

图4-1-10-2表明了对装车单信息进行管理的相关对象之间的协作。

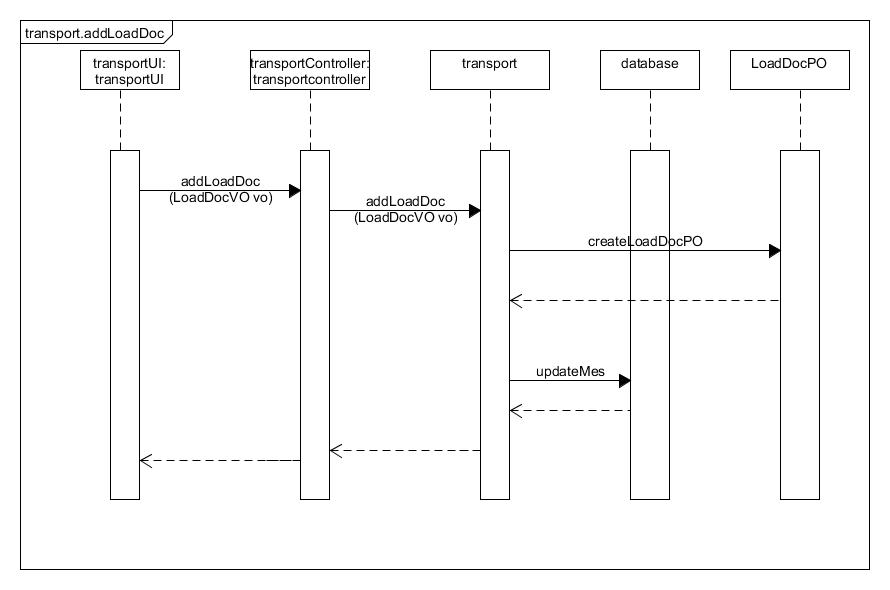
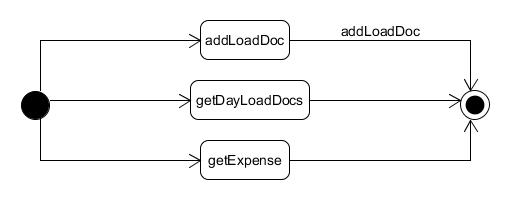


图4-1-10-2装车单信息进行管理

图4-1-10-3 所示的状态图描述了transport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4-1-10-3 transport对象状态图**

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。

### userMes模块

1、模块概述：

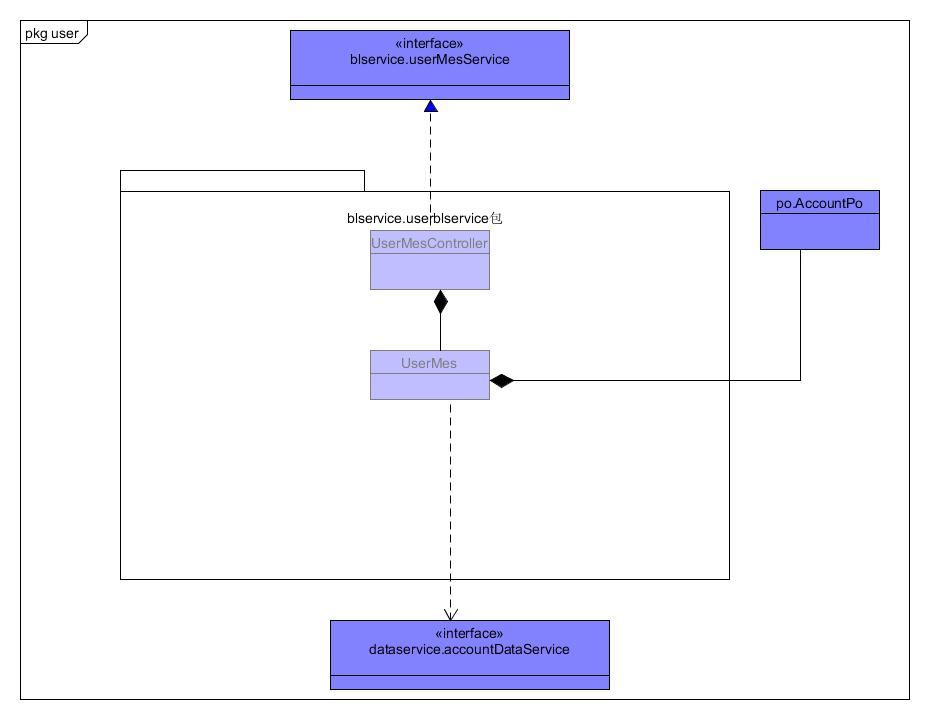
**userMes**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**userMes**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UserMesblService接口。在业务逻辑层和数据层之间添加UserMesdataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加UserMesController，这样UserMesController会将对个人账户信息管理的业务逻辑处理委托给UserMes对象。

UserMesbl模块中各个类的设计：



UserMes模块类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserMesController | 负责对应于用户个人信息管理所需要的服务 |
| UserMes | 用户信息管理的领域模型对象，拥有用户账户的信息，可以帮助完成用户个人信息管理所需要的服务 |

3、模块内部类的接口规范

**UserMesController类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserMesController.login | 语法 | Public ResultMessage login(AccountVO vo) |
| 前置条件 | 已经创建一个UserMes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UserMes的login方法，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 |
| UserMesController.modify | 语法 | Public ResultMessage modify(AccountVO vo) |
| 前置条件 | 已经创建一个UserMes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UserMes的modify方法，系统永久性更新个人账户信息 |
| UserMesController.getMes | 语法 | Public AccountVO getMes() |
| 前置条件 | 已经创建一个UserMes领域对象 |
| 后置条件 | 调用UserMes的getMes方法，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 | |

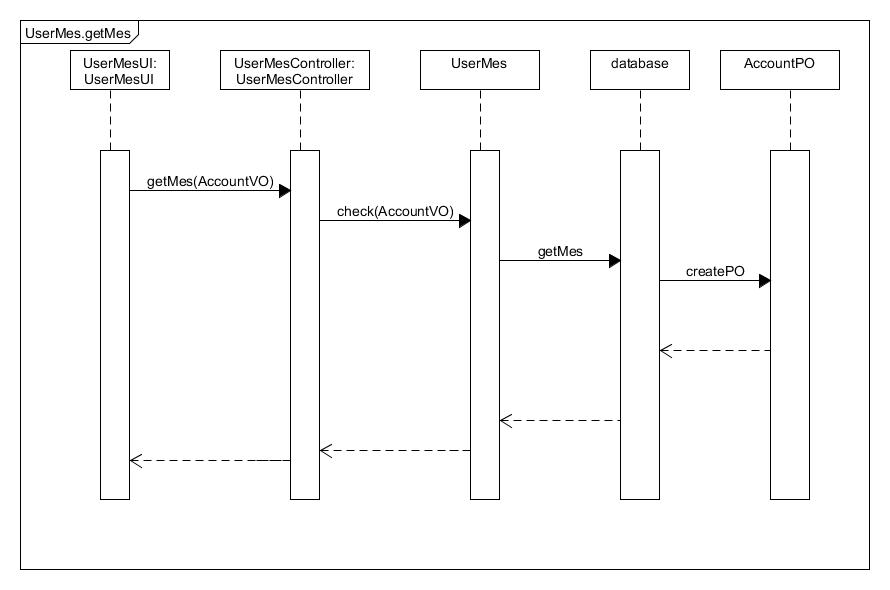
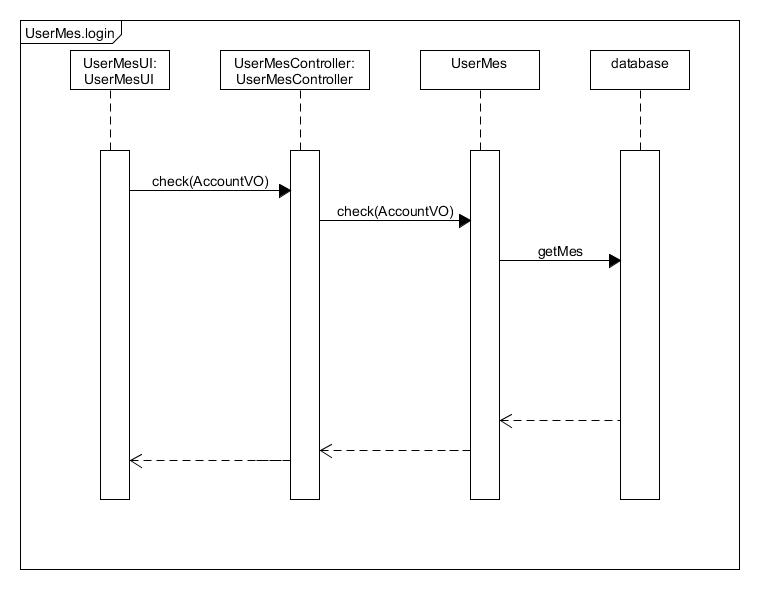
**userMes类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserMes.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 |
| UserMes.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 |
| UserMes.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 | |

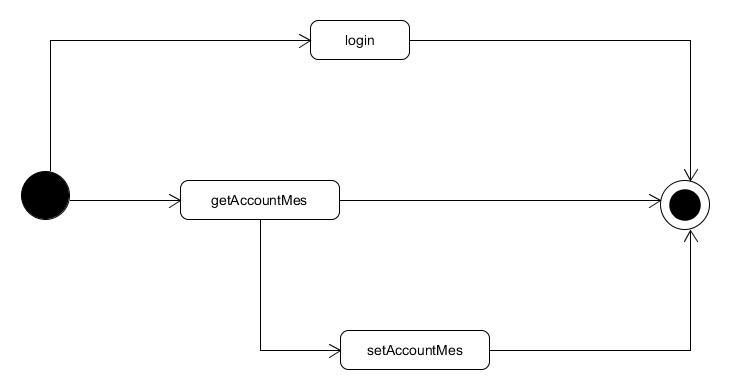
4、业务逻辑层的动态模型

在个人账户信息管理中，业务分为登录、获得、修改账户信息，其中登录、获得账户信

息的顺序图如下图所示：



下图所示的状态图描述了cost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  getAccountDataService() |
| 前置条件 | 无 |
| 后置条件 | 返回AccountDataService对象 |
| DataFactory.getDTManageDataService | 语法 | Public DTManageDataService  getDTManageDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回DTManageDataService对象 |
| DataFactory.getFinanceDataService | 语法 | Public FinanceDataService  getFinanceDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回FinanceDataService对象 |
| DataFactory.getOrderDataService | 语法 | Public OrderDataService  getOrderDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回OrderDataService对象 |
| DataFactory.getPersonnelDataService | 语法 | Public PersonnelDataService  getPersonnelDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回OrderDataService对象 |
| DataFactory.getStatisticDataService | 语法 | Public StatisticDataService  PersonnelDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回StatisticDataService对象 |
| DataFactory.getStoreDataService | 语法 | Public StoreDataService  getStoreDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回StoreDataService对象 |
| DataFactory.getStrategyDataService | 语法 | Public StrategyDataService  getStrategyDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回StrategyDataService对象 |
| DataFactory.getTransportDataService | 语法 | Public TransportDataService  getTransportDataService () |
| 前置条件 | 无 |
| 后置条件 | 返回TransportDataService对象 |

### AccountDataService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 bool 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 bool init() |
| 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 bool delete(AccountPO po) |
| 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 bool 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 bool 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 |

### DTManageDataService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） | | |
| DTManageDataSercvice.getDriverMes | 语法 | Public DriverMesPO getDriverMes(string id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 系统返回司机信息PO |
| DTManageDataSercvice.getCarMes | 语法 | Public CarMesPO getCarMes()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 系统返回车辆信息PO |
| DTManageDataSercvice.addDriverPO | 语法 | Public DriverPO addDriverPO(DriverPO 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Driverpo记录 |
| DTManageDataSercvice.addCarPO | 语法 | Public CarPO addCarPO(CarPO 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Carpo记录 |
| DTManageDataSercvice.deleteDriverPO | 语法 | Public DriverPO deleteDriverPo(DriverPO po)Throws RemoteException |
| 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 CarPO deleteCarPo(CarPO po)Throws RemoteException |
| 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 DriverPO updateDriverPo(DriverPO po)Throws RemoteException |
| 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 CarPO updateCarPo(CarPO po)Throws RemoteException |
| 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 |

### FinanceDataService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.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(DepositPO po) | 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(CostPO po) | 语法 | Public ResultMessage  addCostPO(CostPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统增加一条成本信息 |
| FinanceDataService. modifyCostPO (CostPO po) | 语法 | Public ResultMessage  modifyCostPO(CostPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统修改一条成本信息 |
| FinanceDataService. delCostPO (CostPO po) | 语法 | Public ResultMessage  delCostPO(CostPO po) |
| 前置条件 | 无 |
| 后置条件 | 系统删除一条成本信息 |
| FinanceDataService.addPay(PayPO po) | 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 |

### 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.add | 语法 | Public ResultMessage add(orderPOpo) Throws RemoteException |
| 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(orderPOpo)  Throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中删除原有的订单信息 |
| OrderDateService.addDocToList | 语法 | Public ResultMessage addDocToList (DocPO po)  Throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加订单的流转信息 |
| OrderDateService.getDayOrderPO | 语法 | Public ArrayList<OrderPO> getDayOrderPO (Date date)  Throws RemoteException |
| 前置条件 | 选择好查看的日期 |
| 后置条件 | 无 |
| OrderDateService.getSingleOrderPO | 语法 | Public OrderPO getSingleOrderPO(String orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDateService.getSingleOrderDocs | 语法 | Public ArrayList<DocPO >  getSingleOrderDocs(String orderBarCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDateService.receiveInfo | 语法 | public ResultMessage  receiveInfo(ReceivePO PO) |
| 前置条件 | 无 |
| 后置条件 | 在数据库中更新订单的收件信息 |

### PersonnelDataService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） | | |
| Personnelds.getpeobyInst(String ID) | 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 |
| Personnelds.getpeobyID(String ID) | 语法 | Public PersonVO getpeobyID(String ID) |
| 前置条件 | 无 |
| 后置条件 | 返回该ID查找结果 |
| Personnelds.getpeobyName(String name) | 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 |
| Personnelds.addpeople(PersonPo po) | 语法 | Public bool addPeople(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 |
| Personnelds.deletepeople(PersonPo po) | 语法 | Public bool delPeople(PersonPO 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 |
| Personnelds.addInst(InstPo po) | 语法 | Public bool addInst(InstPO po) |
| 前置条件 | 机构信息合法 |
| 后置条件 | 系统增加该机构 |
| Personnelds.delInst(InstPo po) | 语法 | Public bool delInst(InstPO po) |
| 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 |
| Personnelds.getInst() | 语法 | Public ArrayList<InstPO> getInst() |
| 前置条件 | 无 |
| 后置条件 | 返回所有机构信息 |

### StatisticDataService模块

（1）模块概述

StatisticDataService模块负责保存经营状况表、成本收益表、供读取修改和写入。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StatisticDataService.getStateForm | 语法 | Public ArrayList< StateFormPO >  getStateForm(); |
| 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(StateFormPO) |
| 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(CostIncomeFormPO) |
| 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 |

### StoreDataService模块

（1）模块概述

StoreDataService模块负责保存库存的相关信息，提供保存、修改、读取的功能。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 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(InStoreDocPO po) | 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(OutStoreDocPO po) | 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(StoreMessengePO po) | 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.alarmValue(String value) | 语法 | Public ResultMessage alarmValue(String value) |
| 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.storeCheck(StoreCheckPO po) | 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 |

### StrategyDataService模块

（1）模块概述

StrategyDataService模块负责对常量、薪水策略这些数据的保存、读取、修改。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StretegydataService.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 |
| StretegydataService.setConst | 语法 | Public bool setConst(ConstPO) |
| 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 |
| StretegydataService.getSalary | 语法 | Public ArrayList<SalaryWayPO> getSalary() |
| 前置条件 | 无 |
| 后置条件 | 返回所有工资策略 |
| StretegydataService.setSalary | 语法 | Public bool setSalaryWay(SalaryWayPO po) |
| 前置条件 | 工资策略合法 |
| 后置条件 | 设置新的策略 |
| StretegydataService.getoneSalary | 语法 | Public bool getoneSalary(StaffType type) |
| 前置条件 | 无 |
| 后置条件 | 返回该职员的常量数据 |
| StretegydataService.bulidBill | 语法 | Public bool bulidBill(BillPO po) |
| 前置条件 | 账单数据合法、完整 |
| 后置条件 | 系统储存该账单 |
| StretegydataService.getbills | 语法 | Public ArrayList<BillPO> getBills() |
| 前置条件 | 无 |
| 后置条件 | 返回所有账单 |

### TransportDataService模块

（1）模块概述

TransportDataService模块负责保存流转单（如装车单、到达单等等）中的单据的保存、读取、修改等功能。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportDataSercvice.getLoadDocPO | 语法 | Public LoadDocPO getLoadDocPO(int id) throws RemoteException |
| 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) 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LoadDocpo记录 |
| TransportDataSercvice.getSendGoodDocPO | 语法 | Public SendGoodDocPO  getSendGoodDocPO(int id) throws  RemoteException |
| 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) 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SendGoodDocpo记录 |
| TransportDataSercvice.getArriveZZDocPO | 语法 | Public ArriveZZDocPO getArriveZZDocPO(int id) throws RemoteException |
| 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) 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ArriveZZDocpo记录 |
| TransportDataSercvice.getArriveYYDocPO | 语法 | Public ArriveYYDocPO getArriveYYDocPO(int id) throws RemoteException |
| 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) 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ArriveYYDocpo记录 |
| TransportDataSercvice.getTransferDocPO | 语法 | Public TransferDocPO getTransferDocPO(int id) throws RemoteException |
| 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 |

# 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：

