# 快递系统详细描述文档

**By** 141250163 徐亚帆

141250151 夏志伟

141250155 熊定波

141250142 王亚杰

# 更新记录

2015/11/03 创建

2015/11/06 第一次整合

2015/11/07 初步设计描述

2015/11/16 加入了度量数据

[快递系统详细描述文档 1](#_Toc435476776)

[更新记录 2](#_Toc435476777)

[1.引言 7](#_Toc435476778)

[1.1编制目的 7](#_Toc435476779)

[1.2词汇表 7](#_Toc435476780)

[1.3参考资料 7](#_Toc435476781)

[2.产品概述 8](#_Toc435476782)

[3.体系结构设计概述 8](#_Toc435476783)

[4.结构视角说明 8](#_Toc435476784)

[1寄件单模块 9](#_Toc435476785)

[1.1寄件单界面的原型 9](#_Toc435476786)

[1.2寄件单模块设计文档 9](#_Toc435476787)

[1.2.1说明 9](#_Toc435476788)

[1.2.2寄件单详细设计描述 11](#_Toc435476789)

[1.2.3寄件单动态模型 15](#_Toc435476790)

[1.2.4寄件单模块设计原理 15](#_Toc435476791)

[1.2.5度量计算 15](#_Toc435476792)

[2收件单模块 16](#_Toc435476793)

[2.1收件单界面的原型 16](#_Toc435476794)

[2.2收件单模块设计文档 16](#_Toc435476795)

[2.2.1说明 16](#_Toc435476796)

[2.2.2收件单详细设计描述 17](#_Toc435476797)

[2.2.3收件单动态模型 19](#_Toc435476798)

[2.2.4收件单模块设计原理 19](#_Toc435476799)

[2.2.5度量计算 20](#_Toc435476800)

[3.付款单模块 20](#_Toc435476801)

[3.1付款单界面原型 20](#_Toc435476802)

[3.2付款单模块设计文档 20](#_Toc435476803)

[3.2.1说明 20](#_Toc435476804)

[3.2.2付款单详细设计说明 21](#_Toc435476805)

[3.2.3付款单动态设计模型 23](#_Toc435476806)

[3.2.4耦合与内聚的度量 23](#_Toc435476807)

[4.账户管理模块 24](#_Toc435476808)

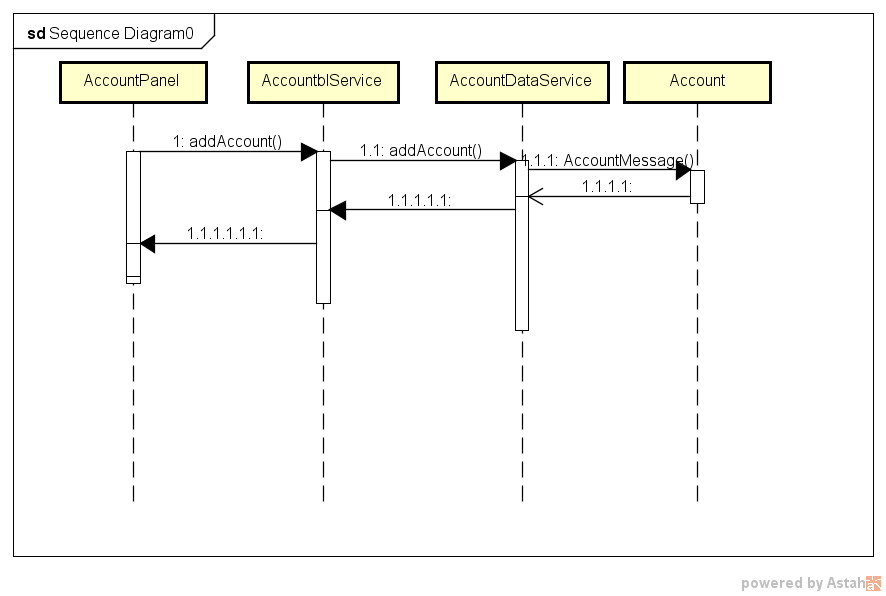
[4.1账户管理页面的模型 24](#_Toc435476809)

[4.2账户管理模块设计文档 26](#_Toc435476810)

[4.2.1说明 26](#_Toc435476811)

[4.2.2详细说明 26](#_Toc435476812)

[4.2.3动态模型 30](#_Toc435476813)

[4.2.4耦合与内聚的度量 30](#_Toc435476814)

[5库存管理模块 31](#_Toc435476815)

[5.1库存管理界面的原型 31](#_Toc435476816)

[5.2库存管理模块设计文档 34](#_Toc435476817)

[5.2.1说明 34](#_Toc435476818)

[5.2.2详细描述 35](#_Toc435476819)

[5.2.3动态模型（以入库示例） 39](#_Toc435476820)

[5.2.4耦合与内聚的度量 39](#_Toc435476821)

[6装车管理模块 40](#_Toc435476822)

[6.1装车管理模块的原型 40](#_Toc435476823)

[6.2装车管理模块设计文档 41](#_Toc435476824)

[6.2.1说明 41](#_Toc435476825)

[6.2.2装车管理详细设计描述 41](#_Toc435476826)

[6.2.3装车管理模块动态模型 44](#_Toc435476827)

[6.2.4耦合与内聚的度量 44](#_Toc435476828)

[7经营查询模块 45](#_Toc435476829)

[7.1经营查询模块的原型 45](#_Toc435476830)

[7.2经营管理管理模块设计文档 45](#_Toc435476831)

[7.2.1说明 45](#_Toc435476832)

[7.2.2详细说明 46](#_Toc435476833)

[7.2.3顺序图 48](#_Toc435476834)

[7.2.4耦合和内聚的度量 48](#_Toc435476835)

[8人员机构管理模块设计文档 49](#_Toc435476836)

[8.1人员机构管理模块原型界面 49](#_Toc435476837)

[8.2.1说明 49](#_Toc435476838)

[8.2.2人员机构管理详细设计描述 50](#_Toc435476839)

[8.2.3人员机构管理动态模型 54](#_Toc435476840)

[8.2.4人员机构管理模块设计原理 55](#_Toc435476841)

[8.2.5度量分析 55](#_Toc435476842)

[9审批单据模块设计文档 56](#_Toc435476843)

[9.1审批单据模块原型界面 56](#_Toc435476844)

[9.2.1说明 56](#_Toc435476845)

[9.2.2审批单据详细设计描述 57](#_Toc435476846)

[9.2.3审批单据动态模型 63](#_Toc435476847)

[9.2.4审批单据模块设计原理 64](#_Toc435476848)

[9.2.5度量分析 64](#_Toc435476849)

[10制定常量模块设计文档 65](#_Toc435476850)

[10.1制定常量原型界面 65](#_Toc435476851)

[10.2.1说明 65](#_Toc435476852)

[10.2.2人员机构管理详细设计描述 66](#_Toc435476853)

[10.2.3人员机构管理动态模型 69](#_Toc435476854)

[10.2.4人员机构管理模块设计原理 70](#_Toc435476855)

[10.2.5度量分析 70](#_Toc435476856)

[11.管理员模块设计文档 70](#_Toc435476857)

[11.1管理员模块原型界面 70](#_Toc435476858)

[11.2.1说明 70](#_Toc435476859)

[11.2.2用户权限管理详细设计描述 71](#_Toc435476860)

[11.2.3 动态模型 73](#_Toc435476861)

[11.2.4模块设计原理 74](#_Toc435476862)

[11.2.5度量计算部分： 74](#_Toc435476863)

[12车辆管理模块 75](#_Toc435476864)

[12.1车辆管理原型界面 75](#_Toc435476865)

[12.2.1说明 75](#_Toc435476866)

[12.2.2车辆管理详细设计描述 76](#_Toc435476867)

[12.2.3车辆管理单动态模型 79](#_Toc435476868)

[12.2.4车辆管理单模块设计原理 80](#_Toc435476869)

[12.2.5度量计算部分： 80](#_Toc435476870)

[13车辆装车管理模块 80](#_Toc435476871)

[13.1车辆装车管理原型界面 80](#_Toc435476872)

[13.2.1说明 80](#_Toc435476873)

[13.2.2装车单详细设计描述 81](#_Toc435476874)

[13.2.3装车单动态模型 85](#_Toc435476875)

[13.2.4装车单模块设计原理 86](#_Toc435476876)

[13.2.5度量计算部分： 86](#_Toc435476877)

[14派件单模块 87](#_Toc435476878)

[14.1派件单原型界面 87](#_Toc435476879)

[14.2.1说明 87](#_Toc435476880)

[14.2.2派件单详细设计描述 88](#_Toc435476881)

[14.2.3派件单动态模型 92](#_Toc435476882)

[14.2.4模块设计原理 92](#_Toc435476883)

[14.2.5度量计算部分： 92](#_Toc435476884)

[15司机信息管理模块 93](#_Toc435476885)

[15.1司机信息管理原型界面 93](#_Toc435476886)

[15.2.1说明 93](#_Toc435476887)

[15.2.2寄件单详细设计描述 94](#_Toc435476888)

[15.2.3司机管理动态模型 97](#_Toc435476889)

[15.2.4司机管理模块设计原理 98](#_Toc435476890)

[15.2.5度量计算部分 98](#_Toc435476891)

[其他原型界面设计 98](#_Toc435476892)

[登录界面 98](#_Toc435476893)

[欢迎界面 99](#_Toc435476894)

# 1.引言

## 1.1编制目的

本文档是关于快递系统的详细设计描述的初步文档。达到指导详细设计和开放的目的，同时实现和测试人员及用户的沟通。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| \*panel | 表示某展示层 |  |
| \_blservice | 表示某逻辑层 |  |
| \_dataservice | 表示某数据层 |  |

## 1.3参考资料

1. IEEE Std 1016-1998 《IEEE Recommended Practice for Software Design Descriptions》

2.丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版社，2012：134—182

# 2.产品概述

参考快递系统软件需求说明文档对该系统的描述。

# 3.体系结构设计概述

参考快递系统体系结构说明文档为该系统体系设计的描述

# 4.结构视角说明

根据界面我们分为以下部分：

|  |  |
| --- | --- |
| 模块 | 界面 |
| 寄件人 | 订单查询界面(OrderQueryPanel) |
| 快递员 | 寄件单界面(postPanel)  收件单界面(RecipientPanel) |
| 营业厅 | 车辆管理界面(BusinessVehiclePanel)  司机管理界面(BusinessDriverPanel)  收款单界面(BusinessColletionPanel)  装车管理界面(BItransportPanel)  派件单界面(BusinessDeliveryPanel) |
| 中转中心 | 入库界面(StockInPanel)  出库界面(StockOutPanel)  库存管理界面(StockManagePanel)  中转中心到达界面(CenterArrivePanel)  装车管理界面 (IItransportPanel)  装车管理界面(IBtransportPanel) |
| 财务中心 | 付款单界面(PaymentPanel)  账户管理界面(AccountPanel)  查询经营情况界面(FinanceInquirePanel)  期初建账界面(FinanceInitializePanel) |
| 总经理 | 人员机构管理界面(UserAndDepartmentManagePanel)  审批单据界面 (DocumentApprovalPanel)  制定价格工资常量的界(MakeConstantPanel)  查询经营情况界面(FinanceInquirePanel) |
| 管理员 | 用户账户权限管理的界面(AdminPanel) |

# 1寄件单模块

## 1.1寄件单界面的原型



## 1.2寄件单模块设计文档

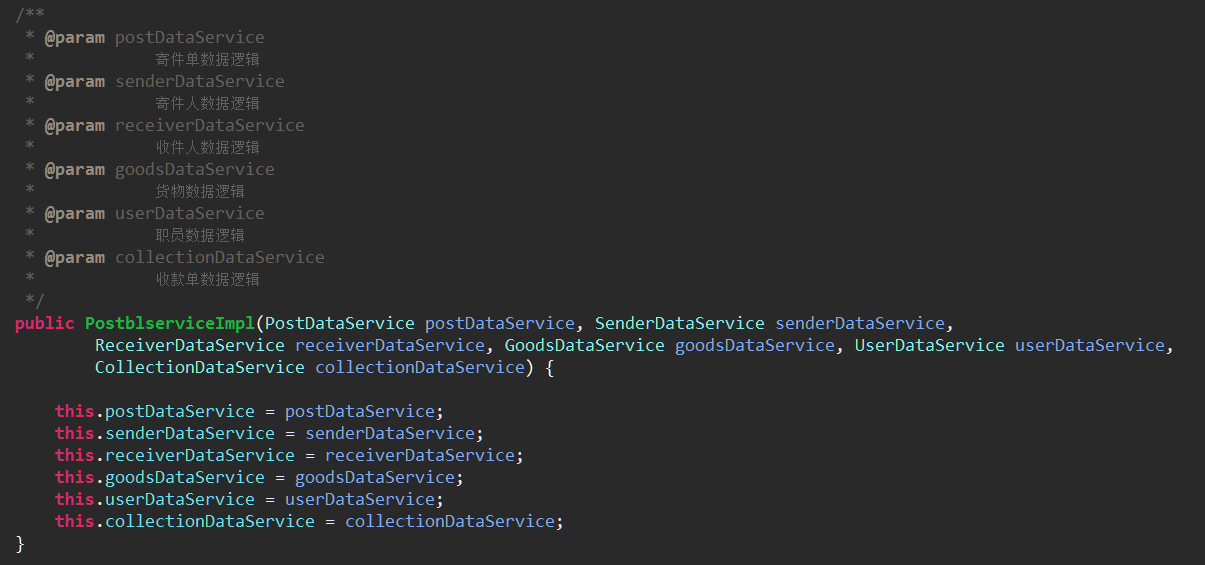
### 1.2.1说明

寄件单界面PostPanel是快递员界面CourierPanel的一部分，快递员界面通过JtabblePanel分为寄件单界面和收件单界面。下面详细说明寄件单界面。

寄件单界面Postpanel需要在构造方法里传入寄件单业务逻辑接口Postblservice

PostblService定义了post寄件单的增删改查，以及获得快递员user的方法。

其具体实现类postblserviceimpl需要在构造方法中传入多个数据接口



在设计的数据库中，一个post有4个外键分别与sender，receiver，goods，collection对应，并且在业务中需要user，所有该业务实现类需要6个数据接口（包括其本身的数据接口）

* postDataService 寄件单数据逻辑
* senderDataService 寄件人数据逻辑
* receiverDataService 收件人数据逻辑
* goodsDataService 货物数据逻辑
* userDataService 职员数据逻辑
* collectionDataService 收款单数据逻辑

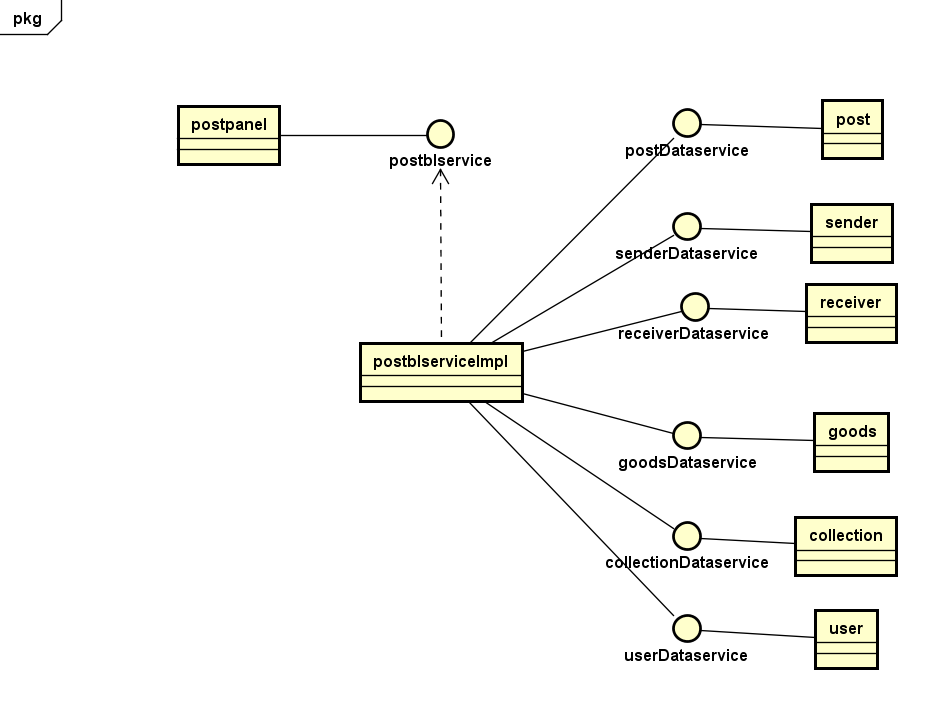
以获得方法为例，从数据库获得一个post对象，需要通过postDataService获得post对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得post里包含的其他对象

若其他对象还包含着其他对象，需要继续封装



### 1.2.2寄件单详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| postPanel | post的用户界面 |
| postblserviceImpl | 负责有关post的业务逻辑实现 |
| postblservice | post业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于post的controller类，而是封装到Panel中

**postPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PostPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| PostPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的post显示在表格中 |
| PostPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的post显示在表格中 |
| 提供的服务（供接口） | | |
| PostPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者post添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| PostPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果postId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| PostPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用postblservice的add方法，然后clear清空 |
| PostPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和post的外键值，调用postblservice的update方法，然后clear清空 |
| PostPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的post的id，调用postblservice的delete方法，然后clear清空 |
| PostPanel.getPost | 语法 | private Post getPost() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装post对象 |
| PostPanel.getSender | 语法 | private Sender getSender() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装sender对象 |
| PostPanel.getReceiver | 语法 | private Receiver getReceiver() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装receiver对象 |
| PostPanel.getGoods | 语法 | private Goods getGoods() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装goods对象 |
| PostPanel.getCollection | 语法 | private Collection getCollection() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装collection对象 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| postblservice | post业务逻辑接口 |
|  |  |

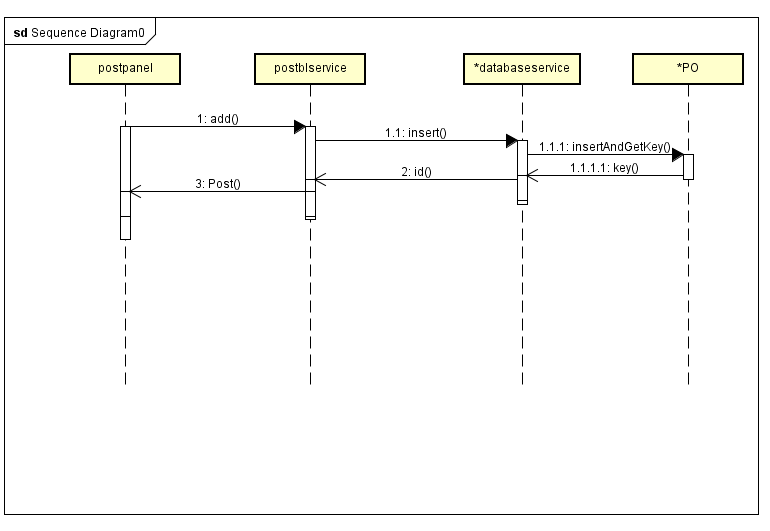
**postblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| postblserviceImpl.getAll | 语法 | public Vector<Post> getAll() |
| 前置条件 | 需要查询所有的post时 |
| 后置条件 | 调用postDataService的getList方法，对获得的每一个post调用senderDataService，receiverDataservice，goodsDataservice，collectionDatas的getById方法，对获得的collection调用userDataservice的getById方法 |
| postblserviceImpl.getById | 语法 | public Post getById(int id) |
| 前置条件 | 需要根据id查询post时 |
| 后置条件 | 调用postDataService的getById方法，对获得的post调用senderDataService，receiverDataservice，goodsDataservice，collectionDatas的getById方法，对获得的collection调用userDataservice的getById方法 |
| postblserviceImpl.add | 语法 | public Post add(Post post) |
| 前置条件 | 需要新增一个post时 |
| 后置条件 | 调用senderDataService，receiverDataservice，goodsDataservice，collectionDatas的insert方法，  获得新增的其他实体类的主键id，设置post对应的外键的属性的值，调用postDataService的insert方法，获得新增的post的主键，根据该主键id查询post并返回这个新增的post |
| postblserviceImpl.update | 语法 | public Post update(Post post) |
| 前置条件 | 需要更新post时 |
| 后置条件 | 调用senderDataService，receiverDataservice，goodsDataservice，collectionDatas的update方法，  调用postDataService的update方法，获得返回的post的主键，根据该主键id查询post并返回这个post |
| postblserviceImpl.getCourier | 语法 | public Vector<User> getCourier() |
| 前置条件 | 需要获得post中collection的快递员user时 |
| 后置条件 | 调用userDataservice的getUserOfJob方法，获得返回的user集合，说明：该方法本应该是collectionblservice的方法，但postblservice也需要这个方法，写在这里可以节省代码 |
| postblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个post时 |
| 后置条件 | 调用postDataservice的delete方法 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| postDataservice.getList() | 获得所有的post持久化对象 |
| postDataservice.getById() | 根据id获得post持久化对象 |
| postDataservice.insert() | 增加一个post持久化对象 |
| postDataservice.update() | 更新一个post持久化对象 |
| postDataservice.delete() | 删除一个post持久化对象 |
|  |  |
| senderDataservice.getList() | 获得所有的sender持久化对象 |
| senderDataservice.getById() | 根据id获得sender持久化对象 |
| senderDataservice.insert() | 增加一个sender持久化对象 |
| senderDataservice.update() | 更新一个sender持久化对象 |
| senderDataservice.delete() | 删除一个sender持久化对象 |
|  |  |
| receiverDataservice.getList() | 获得所有的receiver持久化对象 |
| receiverDataservice.getById() | 根据id获得receiver持久化对象 |
| receiverDataservice.insert() | 增加一个receiver持久化对象 |
| receiverDataservice.update() | 更新一个receiver持久化对象 |
| receiverDataservice.delete() | 删除一个receiver持久化对象 |
|  |  |
| goodsDataservice.getList() | 获得所有的goods持久化对象 |
| goodsDataservice.getById() | 根据id获得goods持久化对象 |
| goodsDataservice.insert() | 增加一个goods持久化对象 |
| goodsDataservice.update() | 更新一个goods持久化对象 |
| goodsDataservice.delete() | 删除一个goods持久化对象 |
|  |  |
| collectionDataservice.getList() | 获得所有的collection持久化对象 |
| collectionDataservice.getById() | 根据id获得collection持久化对象 |
| collectionDataservice.insert() | 增加一个collection持久化对象 |
| collectionDataservice.update() | 更新一个collection持久化对象 |
| collectionDataservice.delete() | 删除一个collection持久化对象 |
|  |  |
| userDataservice.getList() | 获得所有的user持久化对象 |
| userDataservice.getById() | 根据id获得user持久化对象 |
| userDataservice.insert() | 增加一个user持久化对象 |
| userDataservice.update() | 更新一个user持久化对象 |
| userDataservice.delete() | 删除一个user持久化对象 |

### 1.2.3寄件单动态模型

以新增post为例，当用户保存时post主键为空，则为新增操作，调用add方法，调用postblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 1.2.4寄件单模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 1.2.5度量计算

postpanel这个类调用的只有postblservice的6个方法，和一个postblservice对象引用，

所以CBO为6，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1

postblservice有postDataService 寄件单数据逻辑 senderDataService 寄件人数据逻辑

receiverDataService 收件人数据逻辑goodsDataService 货物数据逻辑userDataService 职员数据逻辑collectionDataService 收款单数据逻辑六个对象引用，以及他们的共6\*4=24个方法。

所以CBO为24，DAC为6，postblserviceImpl实现该接口，NOC为1，DIT为0

# 2收件单模块

## 2.1收件单界面的原型



## 2.2收件单模块设计文档

### 2.2.1说明

收件单界面RecipientPanel是快递员界面CourierPanel的一部分，快递员界面通过JtabblePanel分为寄件单界面和收件单界面。下面详细说明收件单界面。

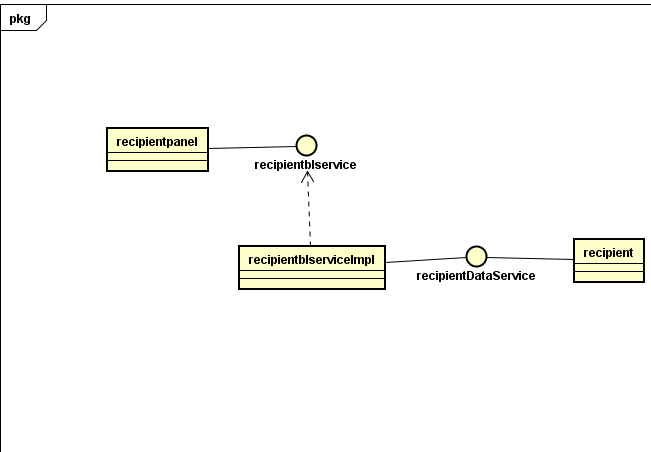
收件单界面RecipientPanel需要在构造方法里传入寄件单业务逻辑接口recipientblservice

recipientblService定义了recipient收件单的确定方法。

其具体实现类recipientblserviceimpl 是一个比较简单的接口实现

在设计的数据库中， recipientblserviceimpl只需要一个数据接口

### 2.2.2收件单详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| recipientPanel | recipient的用户界面 |
| recipientblserviceImpl | 负责有关recipient的业务逻辑实现 |
| recipientblservice | recipient业务逻辑接口，定义需要的业务方法 |
| recipientDateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于recipient的controller类，而是封装到Panel中

recipient**Panel的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| recipientPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| recipientPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的recipient显示在表格中 |
| recipientPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的recipient显示在表格中 |
| 提供的服务（供接口） | | |
| recipientPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者recipient添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| recipientPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果recipientId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| recipientPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用recipientblservice的add方法，然后clear清空 |
| recipientPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和recipient的外键值，调用recipientblservice的update方法，然后clear清空 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| recipientblservice | recipient业务逻辑接口 |
|  |  |

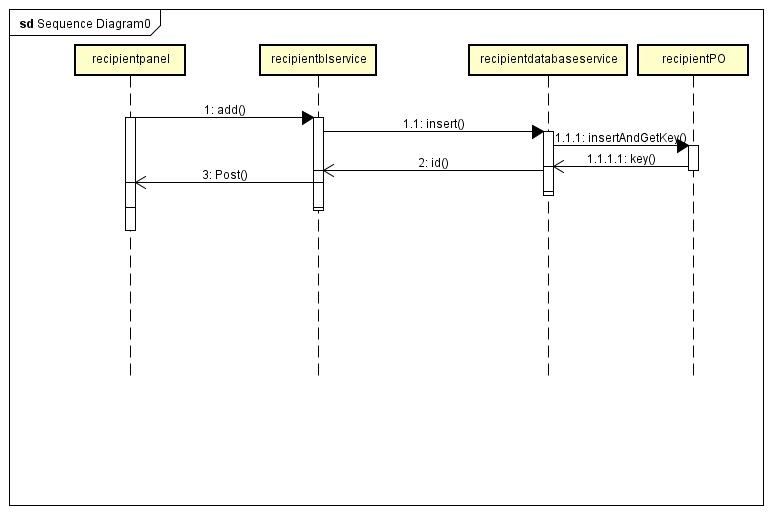
**recipientblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| recipientblserviceImpl.getAll | 语法 | public Vector<Recipient> getAll() |
| 前置条件 | 需要查询所有的recipient时 |
| 后置条件 | 调用recipientDataService的getList方法 |
| recipientblserviceImpl.getById | 语法 | public Recipient getById(int id) |
| 前置条件 | 需要根据id查询recipient时 |
| 后置条件 | 调用recipientDataService的getById方法 |
| recipientblserviceImpl.add | 语法 | public Recipient add(Recipient recipient) |
| 前置条件 | 需要新增一个recipient时 |
| 后置条件 | 调用recipientDataService的insert方法，获得新增的recipient的主键，根据该主键id查询recipient并返回这个新增的recipient |
| recipientblserviceImpl.update | 语法 | public Recipient update(Recipient recipient) |
| 前置条件 | 需要更新recipient时 |
| 后置条件 | 调用recipientDataService的update方法，获得返回的recipient的主键，根据该主键id查询recipient并返回这个recipient |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| recipientDataservice.getList() | 获得所有的recipient持久化对象 |
| recipientDataservice.getById() | 根据id获得recipient持久化对象 |
| recipientDataservice.insert() | 增加一个recipient持久化对象 |
| recipientDataservice.update() | 更新一个recipient持久化对象 |
| recipientDataservice.delete() | 删除一个recipient持久化对象 |
|  |  |

### 2.2.3收件单动态模型

以新增recipient为例，当用户保存时recipient主键为空，则为新增操作，调用add方法，调用recipientblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 2.2.4收件单模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 2.2.5度量计算

recipientpanel这个类调用的只有recipientblservice的3个方法，和一个postblservice对象引用，

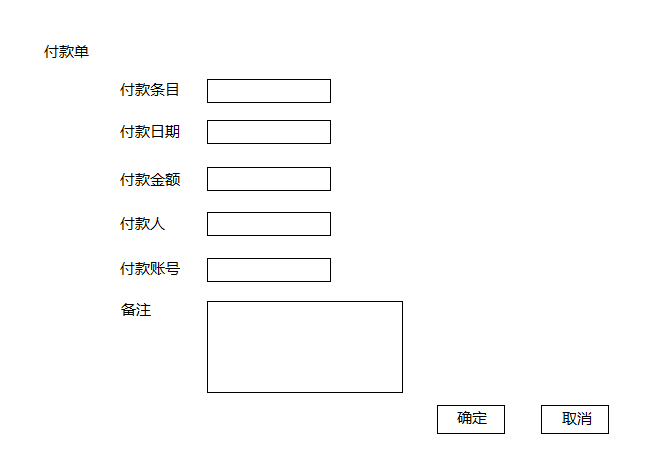
所以CBO为3，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1

recipientblservice有recipientdataservice收件单数据逻辑

所以CBO为4，DAC为1，recipientblserviceImpl实现该接口，NOC为1，DIT为0

# 3.付款单模块

## 3.1付款单界面原型

****

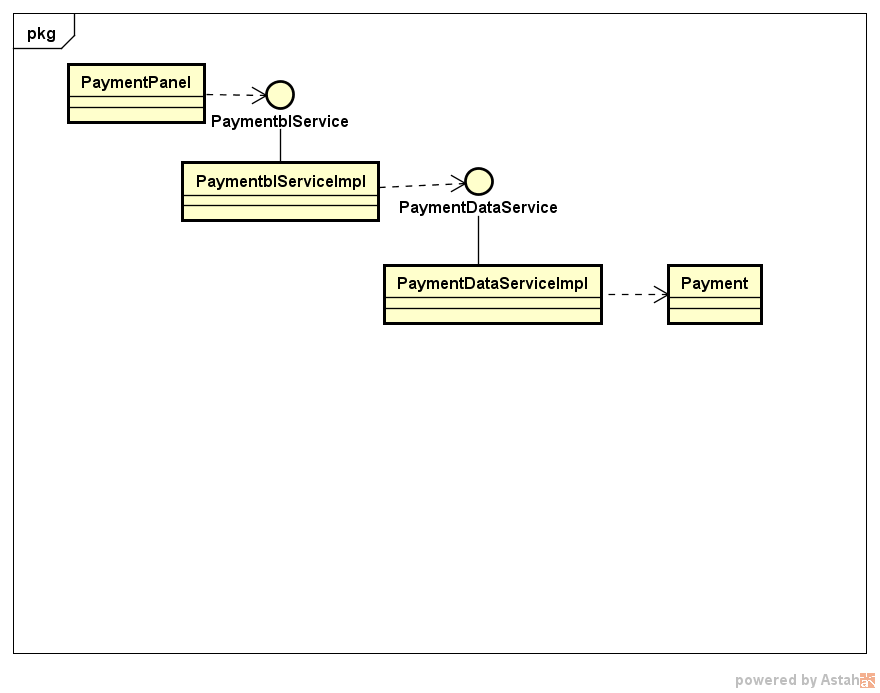
## 3.2付款单模块设计文档

### 3.2.1说明

付款单模块PaymentPanel是财务中心模块的一部分，下面说明该模块。

付款单模块PaymentPanel需要在构造方法中传入付款单逻辑接口PaymentblService，该接口定义了关于付款单的增删改查等方法，其具体实现类PaymentblServiceImpl实现了这些方法，但需要向其中传入Payment这个实体类。

### 3.2.2付款单详细设计说明

****

|  |  |
| --- | --- |
| **模块** | **职责** |
| **PaymentPanel** | Payment的用户界面 |
| **PaymentPanelblservice** | Payment逻辑接口，定义了对付款单Payment的增删改查等反复 |
| **PaymentPanellserviceImpl** | Payment逻辑接口的实现类，实现了上述方法 |
| **PaymentPanelDateservice** | Payment数据接口 |
| **PaymentPanelDateserviceImpl** | Payment数据接口的实现类，负责与数据库之间的交互 |

**PaymentPanel的接口规范**

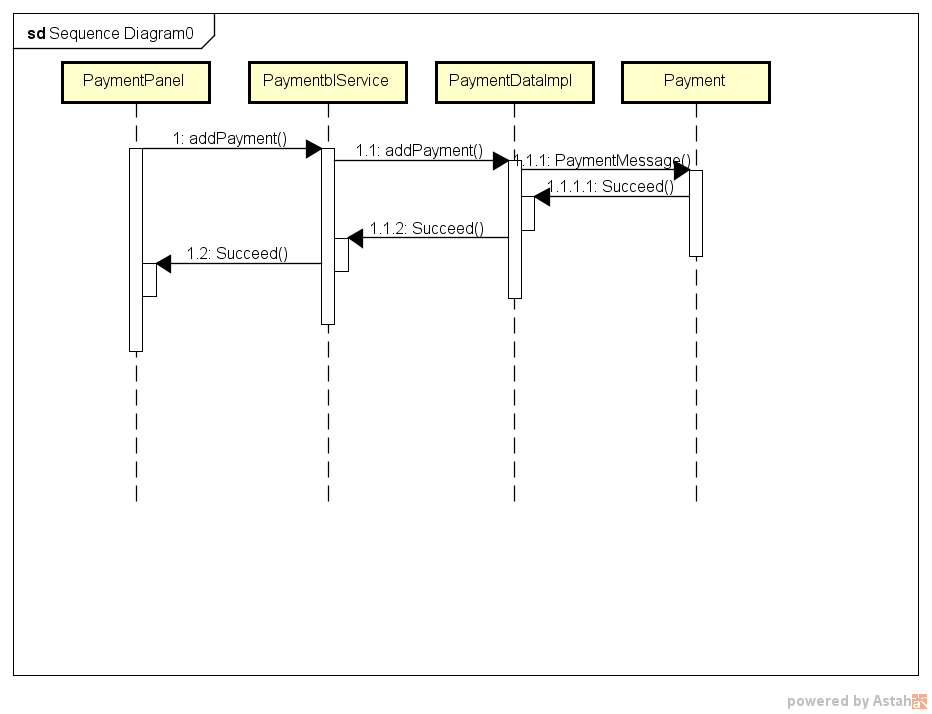
|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| PaymentPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击确定按钮 |
| 后置条件 | 系统保存当前付款单并退出 |
| PaymentPanel.cancel | 语法 | private void cancel() |
| 前置条件 | 用户点击取消按钮 |
| 后置条件 | 系统退出当前页面 |
| PaymentPanel.input | 语法 | private void input() |
| 前置条件 | 用户在JTextArea中输入 |
| 后置条件 | 系统显示用户所输入的内容 |

**PaymentblService的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Payment.creat | 语法 | PaymentblServiceImpl.creatPayment(String datetime,double amount,String name,String type) |
| 前置条件 | 用户输入用于新建付款单的日期、类型、金额、名称 |
| 后置条件 | 系统记录并新建付款单 |
| Payment.check | 语法 | PaymentblServiceImpl. checkPayment(int id) |
| 前置条件 | 用户输入付款单id |
| 后置条件 | 系统返回对应的付款单数据 |
| Payment.statistic | 语法 | PaymentblServiceImpl. statistics(String starttime,String endtime) |
| 前置条件 | 用户输入起止时间 |
| 后置条件 | 系统根据起止条件返回该时段内的付款单总金额 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| PaymentDataService.insert(Payment payment) | 插入单一持久化对象 |
| PaymentDataService.find(int id) | 根据id查找单一持久化对象 |
| PaymentDataService.ststistic(String starttime,String endtime) | 根据时间统计付款单金额总数并返回 |

### 3.2.3付款单动态设计模型

****

### 3.2.4耦合与内聚的度量

付款单页面PaymentPanel这个类调用的是PaymentblService的3个方法，和一个PaymentblService对象引用，所以CBO为3，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1。

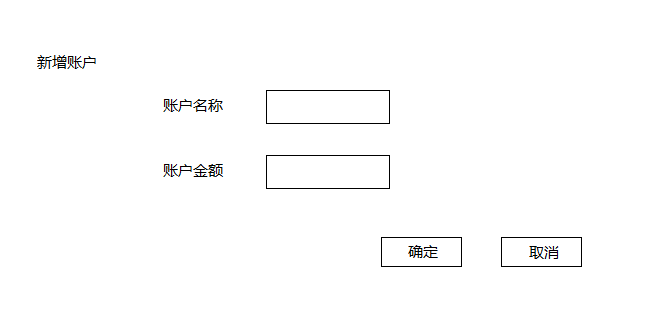
付款单逻辑接口Paymentblservice这个类调用的是PaymentDataservice的3个方法，和一个PaymentDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

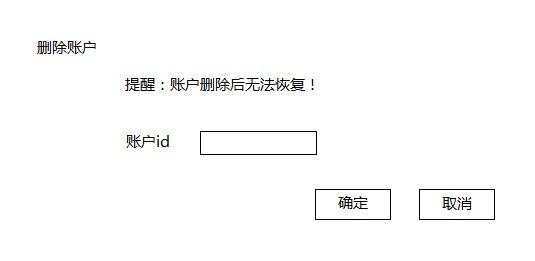
付款单逻辑接口实现类PaymentblserviceImpl这个类调用的是Paymentblservice的3个方法，和一个Paymentblservice对象引用，所以CBO为3，DAC为1，继承一个Paymentblservice，无子类，NOC为0，DIT为1。

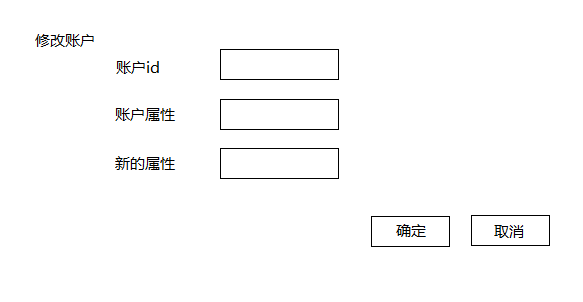
# 4.账户管理模块

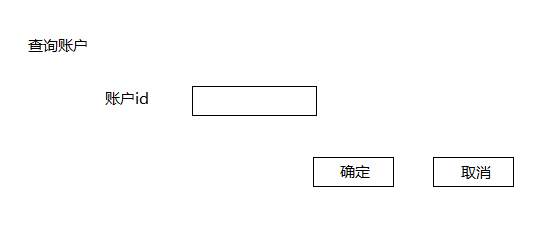
## 4.1账户管理页面的模型

****

****

****

****

****

## 4.2账户管理模块设计文档

### 4.2.1说明

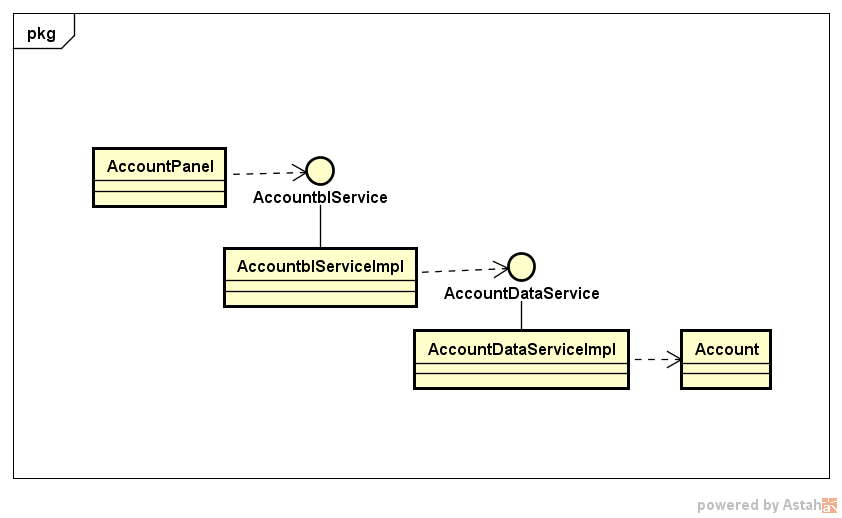
账户管理界面AccountPanel是财务中心页面的一部分，财务中心界面通过JtabblePanel分为主界面、新增账户界面、删除账户界面、修改账户界面、查询账户界面。下面详细说明新增账户界面（其他界面类似于该界面）。

账户管理界面AccountPanel需要在构造方法里传入账户管理业务逻辑接口Accountblservice

Accountblservice定义了Account账户的增删改查。

其具体实现类Accountblserviceimpl需要在构造方法中传入Account这个数据实体类。

### 4.2.2详细说明

****

账户管理模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| LoginController | 负责账户管理人员的登录 |
| Account | 账户的领域模型对象，拥有快递编号、账户位置、出库入库人日期等信息 |
| AccountblServiceImpl | 账户管理的逻辑层实现 |
| AccountDataServiceImpl | 账户管理的数据层实现 |
| AccountPanel | 账户管理的页面 |
| addAccount | 新增账户 |
| deleteAccount | 删除账户 |
| updateAccount | 更新账户 |
| checkAccount | 查询账户 |

**LoginController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LoginController.login | 语法 | private void login(int id,String password) |
| 前置条件 | 用户输入账号密码 |
| 后置条件 | 系统检验账号与密码是否符合，符合进行登录，否则报错 |

**AccountblServiceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| AccountblServiceImpl.check | 语法 | AccountblServiceImpl.check(String startTime,String endTime) |
| 前置条件 | 输入起止时间 |
| 后置条件 | 显示对应时间内的单据简单统计 |
| AccountblServiceImpl. inventory | 语法 | AccountblServiceImpl. Inventory（） |
| 前置条件 | 选择盘点库存 |
| 后置条件 | 统计当日库存单据，生成excel文件 |
|  |  |  |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService .insert(Account Account) | 插入单一持久化对象 |
| AccountDataService Account search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

**AccountPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| AccountPanel.addAccount | 语法 | AccountPanel.addAccount() |
| 前置条件 | 用户点击新增账户 |
| 后置条件 | 系统跳转至新增账户界面 |
| AccountPanel. deleteAccount | 语法 | AccountPanel. deleteAccount() |
| 前置条件 | 用户点击删除账户 |
| 后置条件 | 系统跳转至删除账户界面 |
| AccountPanel. updateAccount | 语法 | AccountPanel. updateAccount() |
| 前置条件 | 用户点击更新账户 |
| 后置条件 | 系统跳转至更新账户界面 |
| AccountPanel. checkAccount | 语法 | AccountPanel. checkAccount() |
| 前置条件 | 用户点击查看账户 |
| 后置条件 | 系统跳转至查看账户界面 |

addAccount**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| addAccount.save | 语法 | addAccount.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| addAccount. cancel | 语法 | addAccount. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |
| addAccount.input | 语法 | addAccount.input() |
| 前置条件 | 用户进行输入 |
| 后置条件 | 系统显示用户所输入内容 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService .insert(Account Account) | 插入单一持久化对象 |
| AccountDataService Account search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

deleteAccount**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| deleteAccount.save | 语法 | deleteAccount.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| deleteAccount. cancel | 语法 | deleteAccount. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |
| deleteAccount.input | 语法 | deleteAccount.input() |
| 前置条件 | 用户进行输入 |
| 后置条件 | 系统显示用户所输入内容 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService .insert(Account Account) | 插入单一持久化对象 |
| AccountDataService Account search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

updateAccount**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| updateAccount.save | 语法 | updateAccount.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| updateAccount. cancel | 语法 | updateAccount. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |
| updateAccount.input | 语法 | updateAccount.input() |
| 前置条件 | 用户进行输入 |
| 后置条件 | 系统显示用户所输入内容 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService .insert(Account Account) | 插入单一持久化对象 |
| AccountDataService Account search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

checkAccount**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| checkAccount.save | 语法 | checkAccount.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| checkAccount. cancel | 语法 | checkAccount. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |
| checkAccount.input | 语法 | checkAccount.input() |
| 前置条件 | 用户进行输入 |
| 后置条件 | 系统显示用户所输入内容 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService .insert(Account Account) | 插入单一持久化对象 |
| AccountDataService Account search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

### 4.2.3动态模型

### 顺序图4.2.4耦合与内聚的度量

登录控制LoginController这个类未调用其他类的方法，获得了User这个对象，没有继承类，所以DAC为1，CBO、NOC、DIT都为0。

账户Account这个类没有调用其它类的方法，也没有获得其他对象，没有继承类，所以DAC、CBO、NOC、DIT都为0。

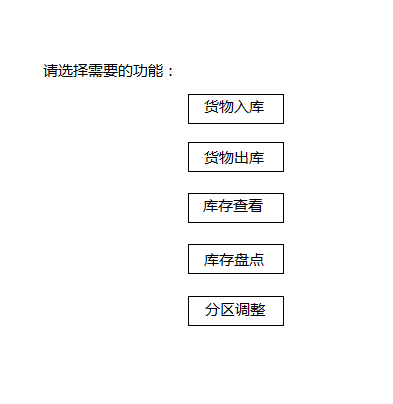
账户管理逻辑接口AccountblServiceImpl这个类调用的是AccountDataservice的3个方法，和一个AccountDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

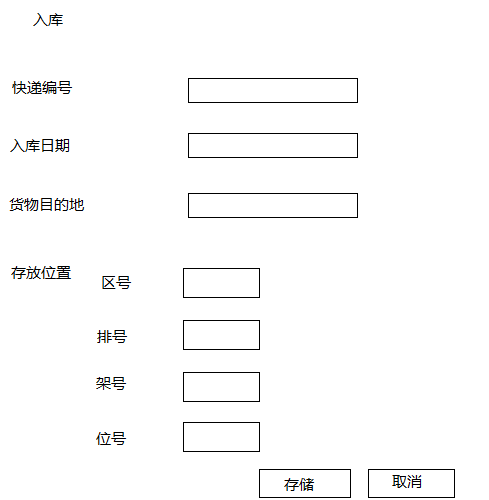
账户管理页面AccountPanel这个类未调用其它类的方法，调用了addAccount、deleteAccount、updateAccount、checkAccount这4个对象，未继承其它类，所有DAC为4，CBO、NOC和DIT为0。

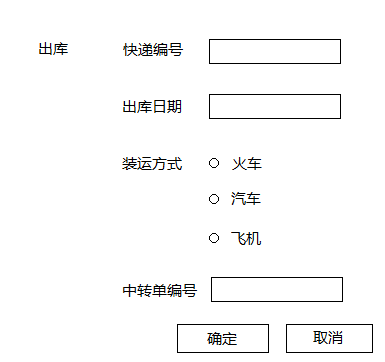
新增账户addAccount（deleteAccount、updateAccount、checkAccount与addAccount相同）这个类调用的是AccountDataservice的3个方法，和一个AccountDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

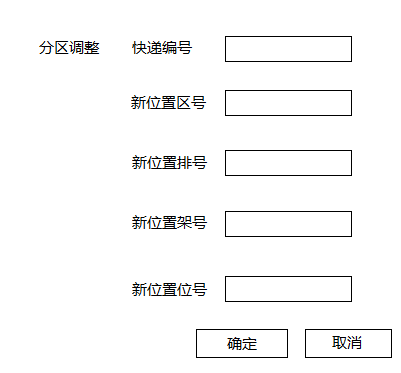
# 5库存管理模块

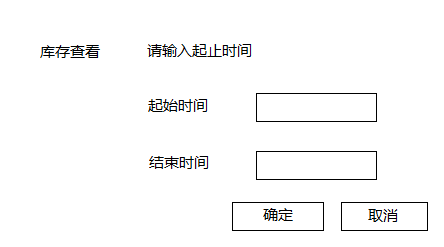
## 5.1库存管理界面的原型

****

****

****

****

****

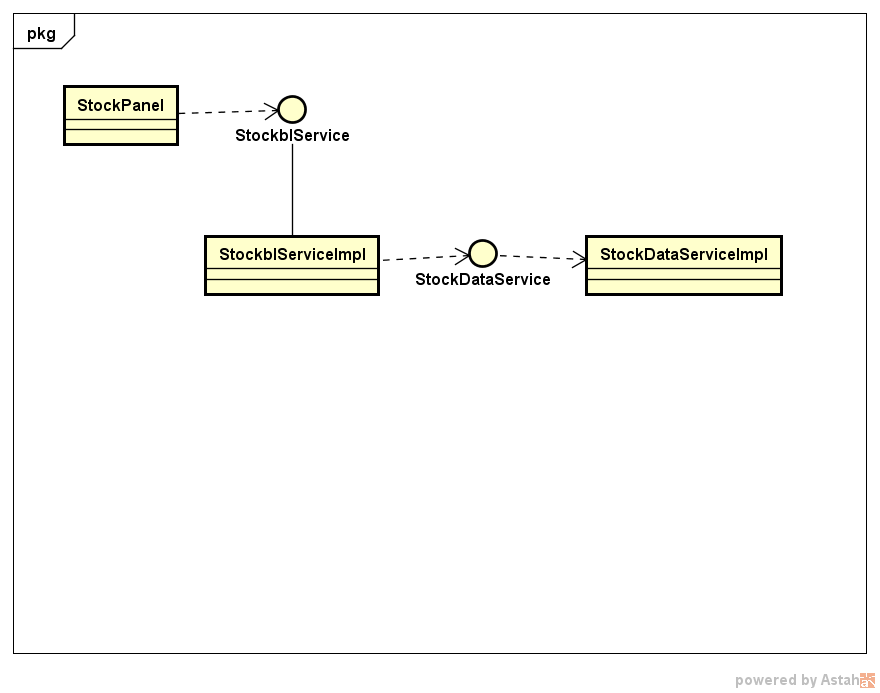
## 5.2库存管理模块设计文档

### 5.2.1说明

库存管理模块InventoryMangement是中转中心界面的一部分，库存管理界面分为入库、出库、库存查看和分区调整页面。下面说明库存管理页面。

库存管理界面StockPanel需要在构造方法中传入库存逻辑接口StockblService。StockblService接口中定义了入库单、出库单、查看以及分区调整等方法。

### 5.2.2详细描述

****

库存管理模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| LoginController | 负责库存管理人员的登录 |
| Stock | 库存的领域模型对象，拥有快递编号、库存位置、出库入库人日期等信息 |
| StockblServiceImpl | 库存管理的逻辑层实现 |
| StockDataServiceImpl | 库存管理的数据层实现 |
| StockPanel | 库存管理的页面 |
| StockIn | 入库管理 |
| StockOut | 出库管理 |
| StockUpdate | 分区调整 |
| StockCheck | 库存查看 |

**LoginController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LoginController.login | 语法 | private void login(int id,String password) |
| 前置条件 | 用户输入账号密码 |
| 后置条件 | 系统检验账号与密码是否符合，符合进行登录，否则报错 |

**StockblServiceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockblServiceImpl.check | 语法 | StockblServiceImpl.check(String startTime,String endTime) |
| 前置条件 | 输入起止时间 |
| 后置条件 | 显示对应时间内的单据简单统计 |
| StockblServiceImpl. inventory | 语法 | StockblServiceImpl. Inventory（） |
| 前置条件 | 选择盘点库存 |
| 后置条件 | 统计当日库存单据，生成excel文件 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| StockDataService .insert(Stock stock) | 插入单一持久化对象 |
| StockDataService Stock search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

**StockPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockPanel.StockIn | 语法 | StockPanel.StockIn() |
| 前置条件 | 用户点击入库 |
| 后置条件 | 系统跳转至入库界面 |
| StockPanel. StockOut | 语法 | StockPanel. StockOut() |
| 前置条件 | 用户点击出库 |
| 后置条件 | 系统跳转至出库界面 |
| StockPanel. StockUpdate | 语法 | StockPanel. StockUpdate() |
| 前置条件 | 用户点击分区调整 |
| 后置条件 | 系统跳转至分区调整界面 |
| StockPanel. StockCheck | 语法 | StockPanel. StockCheck() |
| 前置条件 | 用户点击库存查看 |
| 后置条件 | 系统跳转至库存查看界面 |

**StockIn的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockIn.save | 语法 | StockIn.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| StockIn. cancel | 语法 | StockIn. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| StockDataService .insert(Stock stock) | 插入单一持久化对象 |
| StockDataService Stock search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

**StockOut的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockOut.save | 语法 | StockOut.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| StockOut. cancel | 语法 | StockOut. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| StockDataService .insert(Stock stock) | 插入单一持久化对象 |
| StockDataService Stock search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

**StockUpdate的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockUpdate.save | 语法 | StockUpdate.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| StockUpdate. cancel | 语法 | StockUpdate. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |

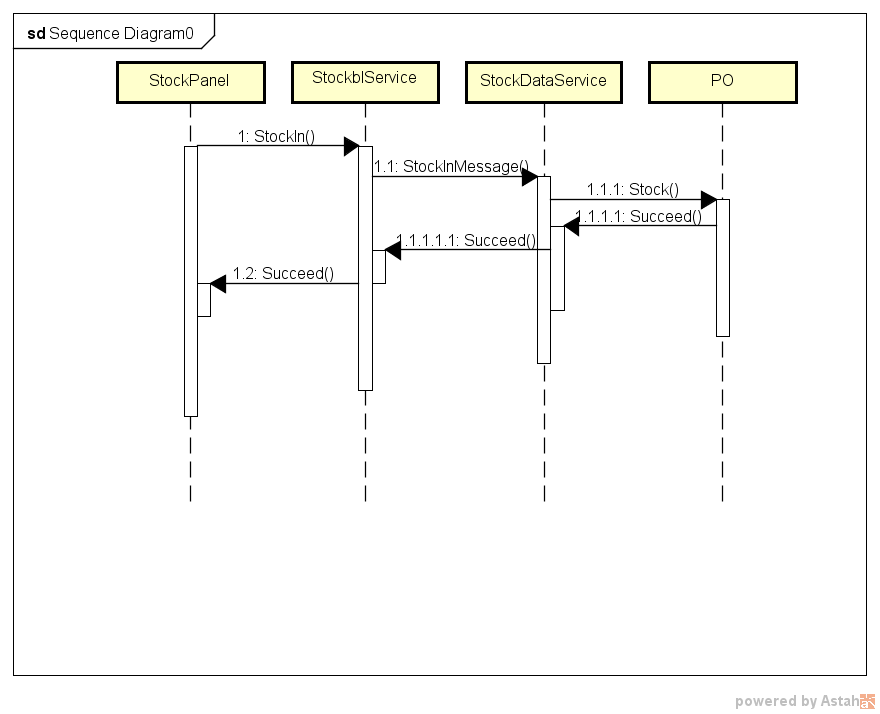
|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| StockDataService .insert(Stock stock) | 插入单一持久化对象 |
| StockDataService Stock search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

**StockCheck的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockCheck.save | 语法 | StockCheck.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| StockCheck. cancel | 语法 | StockCheck. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| StockDataService .insert(Stock stock) | 插入单一持久化对象 |
| StockDataService Stock search(String time1,String time2) | 根据起止时间，查询多个持久化对象 |

### 5.2.3动态模型（以入库示例）

****

### 5.2.4耦合与内聚的度量

登录控制LoginController这个类未调用其他类的方法，获得了User这个对象，没有继承类，所以DAC为1，CBO、NOC、DIT都为0。

库存Stock这个类没有调用其它类的方法，也没有获得其他对象，没有继承类，所以DAC、CBO、NOC、DIT都为0。

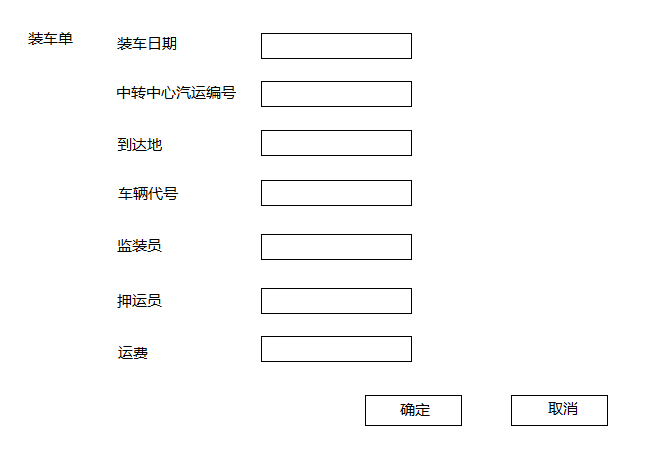
库存管理逻辑接口StockblServiceImpl这个类调用的是StockDataservice的3个方法，和一个StockDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

库存管理逻辑接口实现类StockblserviceImpl这个类调用的是Stockblservice的3个方法，和一个Stockblservice对象引用，所以CBO为3，DAC为1，继承一个Stockblservice，无子类，NOC为0，DIT为1。

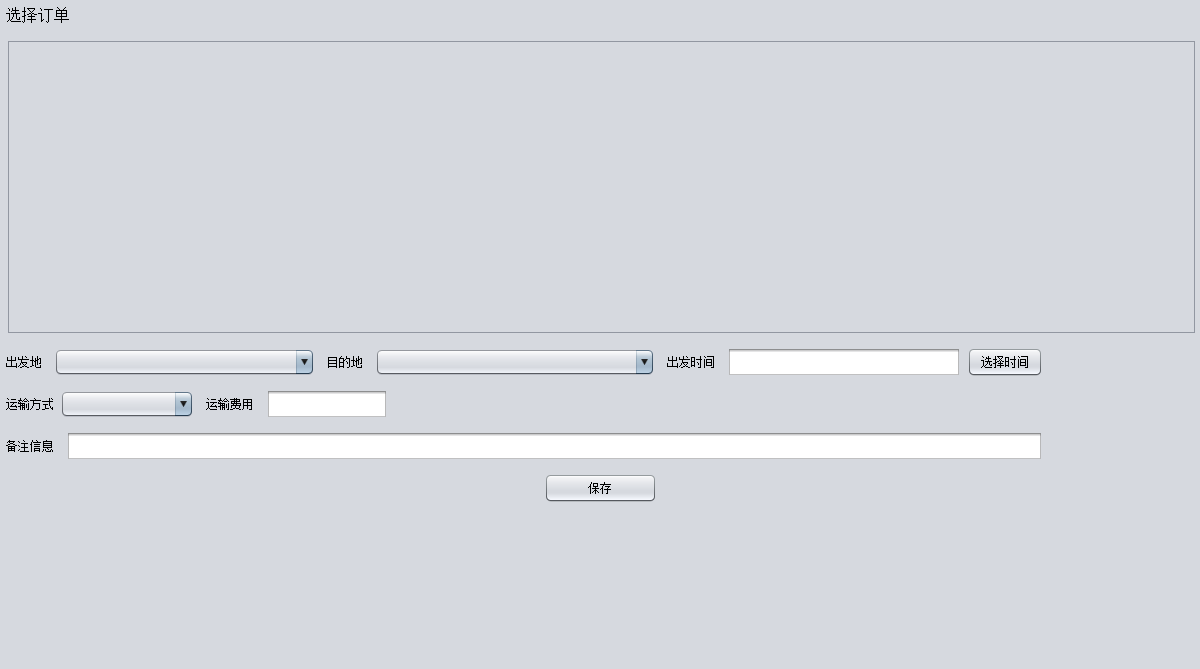
入库管理StockIn（StockOut、StockUpdate、StockCheck与StockIn相同）这个类调用的是StockDataservice的3个方法，和一个StockDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

# 6装车管理模块

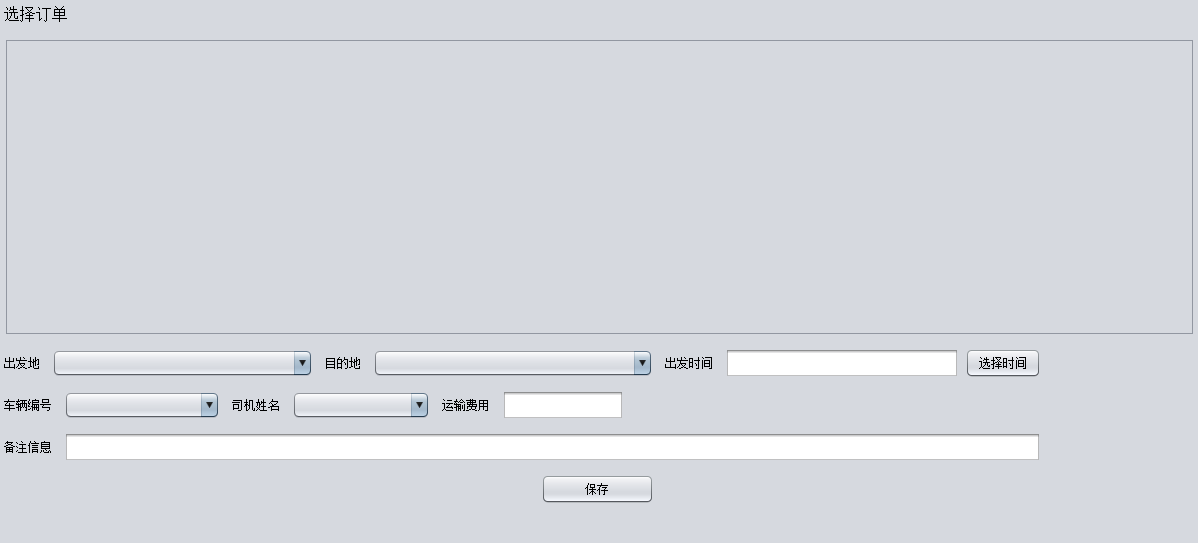
## 6.1装车管理模块的原型

****

发往其他中转中心的

****

发往营业厅的

****

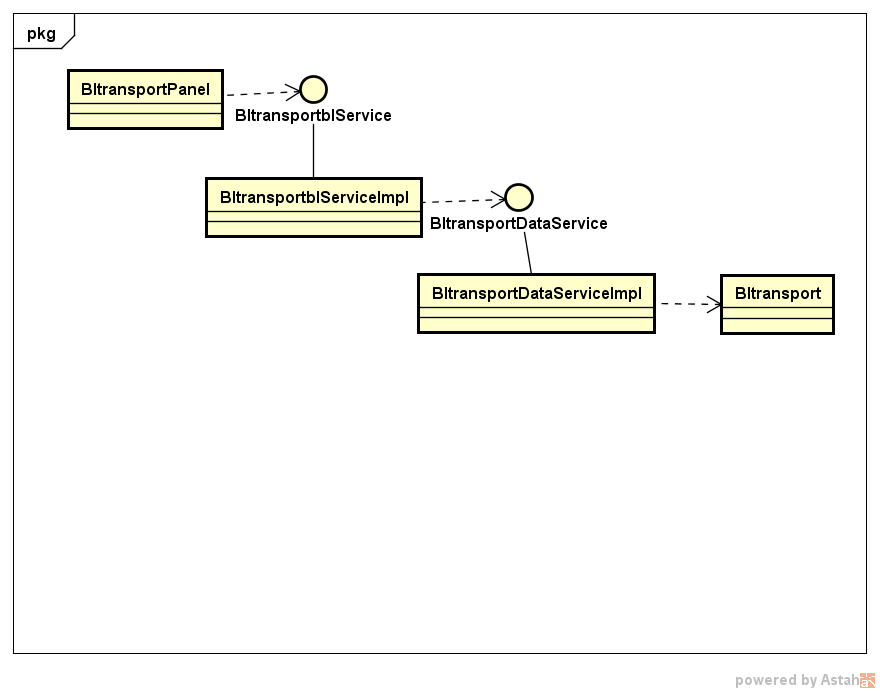
## 6.2装车管理模块设计文档

### 6.2.1说明

装车管理界面BItransportPanel是中转中心页面的一部分，下面说明该模块。

装车管理界面BItransportPanel需要向构造方法传入装车管理逻辑接口BItransportblService,该接口定义了对于装车单的增删改查等方法，其具体实现类BItransportblServiceImpl在实现这些方法时，需要向其中传入实体类BItransport。

### 6.2.2装车管理详细设计描述

****

|  |  |
| --- | --- |
| **模块** | **职责** |
| **BItransportPanel** | BItransport的页面 |
| **BItransportblServiceImpl** | BItransport的逻辑接口，当中包括对BItransport的增删改查 |
| **BItransportblService** | BItransport的逻辑接口的实现类，实现了增删改查等方法 |
| **BItransportDataService** | BItransport的数据接口 |
| **BItransportDataServiceImpl** | BItransport的数据接口的实现类，进行与数据库之间的交互 |

**BItransportPanel的接口规范**

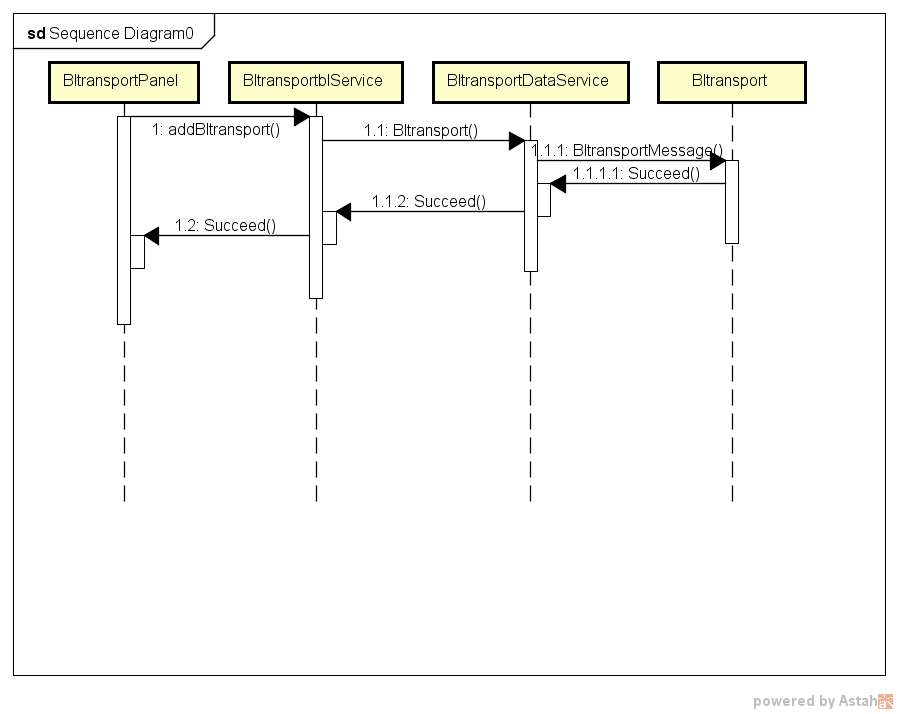
|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BItransportPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击确定按钮 |
| 后置条件 | 系统保存装车单并退出当前页面 |
| BItransportPanel.cancel | 语法 | private void cancel() |
| 前置条件 | 用户点击取消按钮 |
| 后置条件 | 系统直接退出当前页面 |
| BItransportPanel.input | 语法 | public void input() |
| 前置条件 | 用户在JtextArea中输入 |
| 后置条件 | 系统显示客户输入 |

**BItransportblServiceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BItransportblService.addBItransport | 语法 | public void addBItransport(BItransport bitransport) |
| 前置条件 | 用户要求新增装车单 |
| 后置条件 | 新增一个装车单数据 |
| BItransportblService.deleteBItransport | 语法 | public void deleteBItransport(int id) |
| 前置条件 | 用户要求删除装车单 |
| 后置条件 | 删除对应装车单 |
| BItransportblService.updateBItransport | 语法 | public void updateBItransport(int id,BItransport bitransport) |
| 前置条件 | 用户要求更改装车单信息 |
| 后置条件 | 系统更新装车单信息 |
| BItransportblService.getBItransport | 语法 | public void getBItransport(int id) |
| 前置条件 | 用户要求得到装车单信息 |
| 后置条件 | 系统返回装车单信息 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BItransportDataService.addBItransport | 添加新装车单 |
| BItransportDataService.deleteBItransport | 删除指定装车单 |
| BItransportDataService.updateBItransport | 更新装车单信息 |
| BItransportDataService.getBItransport | 获取装车单信息 |

### 6.2.3装车管理模块动态模型

****

### 6.2.4耦合与内聚的度量

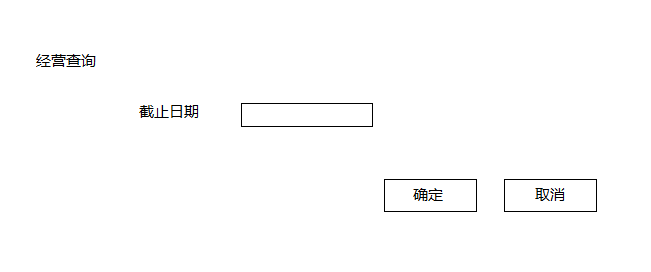
装车管理查询页面BItransportPanel这个类调用的是BItransportDataservice的3个方法，和一个BItransportDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

装车管理逻辑接口BItransportblServiceImpl这个类调用的是BItransportDataservice的3个方法，和一个BItransportDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

装车管理逻辑接口实现类BItransportblserviceImpl这个类调用的是BItransportblservice的3个方法，和一个BItransportblservice对象引用，所以CBO为3，DAC为1，继承一个BItransportblservice，无子类，NOC为0，DIT为1。

# 7经营查询模块

## 7.1经营查询模块的原型

****

## 7.2经营管理管理模块设计文档

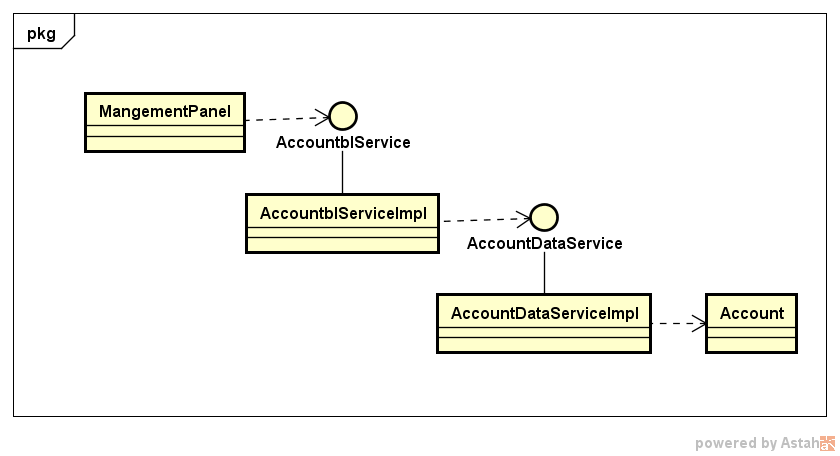
### 7.2.1说明

经营管理管理界面MangementPanel是财务中心页面的一部分，财务中心界面通过JtabblePanel分为主界面、新增经营管理界面、删除经营管理界面、修改经营管理界面、查询经营管理界面。下面详细说明新增经营管理界面（其他界面类似于该界面）。

经营管理管理界面MangementPanel需要在构造方法里传入经营管理管理业务逻辑接口Accountblservice，Accountblservice定义了Account经营管理的查询。

其具体实现类Accountblserviceimpl需要在构造方法中传入Account这个数据实体类。

### 7.2.2详细说明

****

经营查询管理模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| LoginController | 负责账户管理人员的登录 |
| Account | 账户的领域模型对象，拥有快递编号、账户位置、出库入库人日期等信息 |
| AccountblServiceImpl | 账户管理的逻辑层实现 |
| AccountDataServiceImpl | 账户管理的数据层实现 |
| MangementPanel | 经营查询管理的页面 |

**LoginController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LoginController.login | 语法 | private void login(int id,String password) |
| 前置条件 | 用户输入账号密码 |
| 后置条件 | 系统检验账号与密码是否符合，符合进行登录，否则报错 |

**AccountblServiceImpl的接口规范**

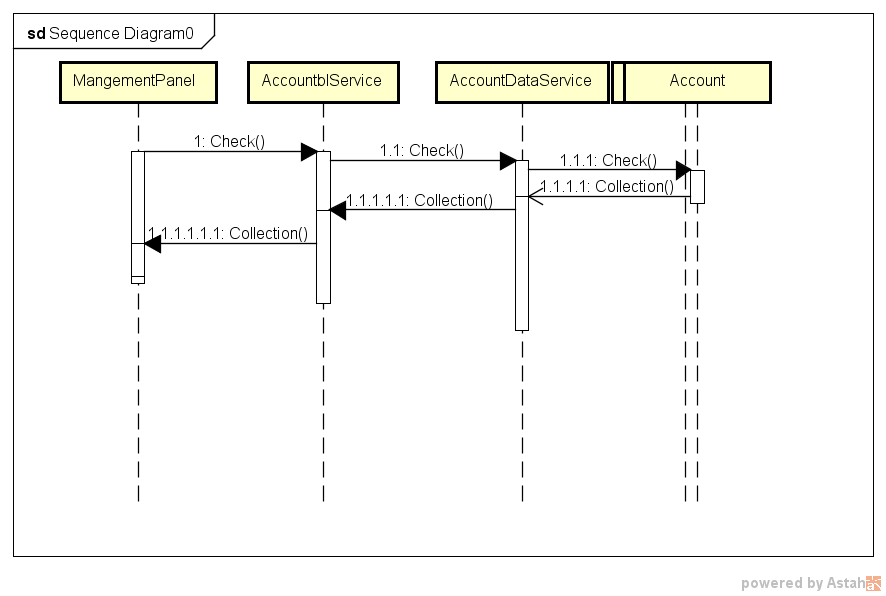
|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| AccountblService.checkCollection | 语法 | public List<Collection> checkCollection(int id, String account\_name, String account\_startdate,  String account\_enddate) |
| 前置条件 | 用户要求查看所有收款单 |
| 后置条件 | 系统返回到当前时间的所有收款单 |
| AccountblService.checkPayment | 语法 | public List<Payment> checkPayment(int id, String account\_name, String account\_startdate, String account\_enddate) |
| 前置条件 | 用户要求查看所有付款单 |
| 后置条件 | 系统返回到当前时间的所有付款单 |
| AccountblService.checkamount | 语法 | public double checkamount(int id, String account\_name, String account\_startdate, String account\_enddate) |
| 前置条件 | 用户要求查看到当前时间的总收益 |
| 后置条件 | 系统更新账号信息 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| AccountDataService.checkCollection | 输出到当前时间的所有收款单 |
| AccountDataService.checkPayment | 输出到当前时间的所有付款单 |
| AccountDataService.checkamount | 输出到当前时间的总收益 |

**MangementPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| MangementPanel.save | 语法 | MangementPanel.save() |
| 前置条件 | 用户点击确定 |
| 后置条件 | 系统保存并退出当前页面 |
| MangementPanel. cancel | 语法 | MangementPanel. Cancel() |
| 前置条件 | 用户点击取消 |
| 后置条件 | 系统直接退出当前页面 |
| MangementPanel.input | 语法 | MangementPanel.input() |
| 前置条件 | 用户进行输入 |
| 后置条件 | 系统显示用户所输入内容 |

### 7.2.3顺序图

****

### 7.2.4耦合和内聚的度量

登录控制LoginController这个类未调用其他类的方法，获得了User这个对象，没有继承类，所以DAC为1，CBO、NOC、DIT都为0。

账户Account这个类没有调用其它类的方法，也没有获得其他对象，没有继承类，所以DAC、CBO、NOC、DIT都为0。

账户管理逻辑接口AccountblServiceImpl这个类调用的是AccountDataservice的3个方法，和一个AccountDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

账户管理逻辑接口实现类AccountblserviceImpl这个类调用的是Accountblservice的3个方法，和一个Accountblservice对象引用，所以CBO为3，DAC为1，继承一个Accountblservice，无子类，NOC为0，DIT为1。

经营管理查询页面MangementPanel这个类调用的是AccountDataservice的3个方法，和一个AccountDataservice对象引用，所以CBO为3，DAC为1，未继承其他类，无子类，所以NOC和DIT都为0。

# 8人员机构管理模块设计文档

## 8.1人员机构管理模块原型界面

### 8.2.1说明

人员机构管理界面UserAndDepartmentManagePanel是总经理界面ManagerPanel的一部分，总经理界面通过JtabblePanel分为人员机构管理界面、审批单据界面、制定价格工资常量的界面。下面详细说明人员机构管理界面。

人员机构管理界面UserAndDepartmentManagePanel需要在构造方法里传入用户管理业务逻辑接口UserAndDepartmentManageblservice

UserblService定义了User人员的增删改查的方法。

其具体实现类UserAndDepartmentManageblserviceimpl需要在构造方法中传入多个数据接口

在设计的数据库中，一个User有两个外键分别于job、department对应。

业务实现需要3个数据接口

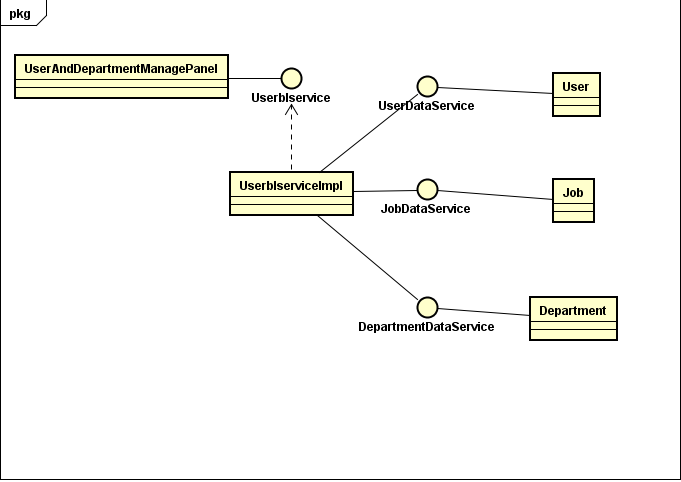
* UserDataService 用户数据逻辑
* JobDataService 工作数据逻辑
* DepartmentDataService 部门数据逻辑

以获得方法为例，从数据库获得一个User对象，需要通过UserDataService获得User对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得User里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 8.2.2人员机构管理详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| UserAndDepartmentManagePanel | UserAndDepartmentManage的用户界面 |
| UserblserviceImpl | 负责有关User的业务逻辑实现 |
| Userblservice | User业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于UserAndDepartment的controller类，而是封装到Panel中

**UserAndDepartmentManagelPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserAndDepartmentManagelPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| UserAndDepartmentManagelPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的User显示在表格中 |
| UserAndDepartmentManagelPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将结果的User显示在表格中 |
| UserAndDepartmentManagelPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击取消按钮，或者User添加，修改，删除，查询操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| UserAndDepartmentManagelPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果UserId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| UserAndDepartmentManagelPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用Userblservice的add方法，然后clear清空 |
| UserAndDepartmentManagelPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和post的外键值，调用postblservice的update方法，然后clear清空 |
| UserAndDepartmentManagelPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的User的id，调用Userblservice的delete方法，然后clear清空 |
| UserAndDepartmentManagelPanel.getUser | 语法 | private User getUser() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装User对象 |
| UserAndDepartmentManagelPanel.getJob | 语法 | private Job getJob() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装Job对象 |
| UserAndDepartmentManagelPanel.getDepartment | 语法 | private Department getDepartment() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装Department对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| Userblservice | User业务逻辑接口 |
|  |  |

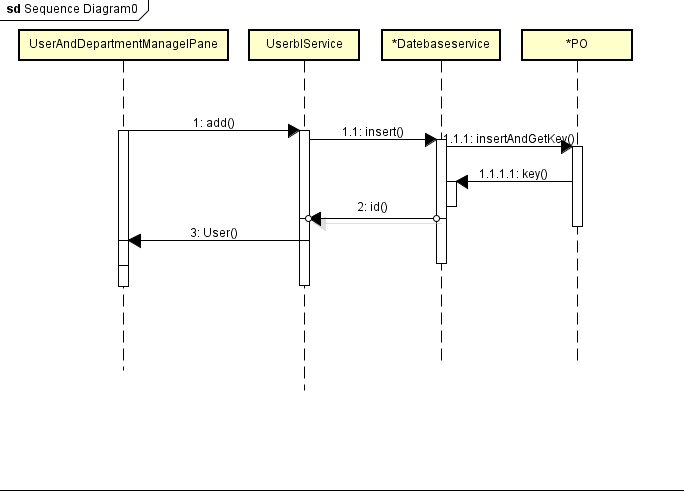
**UserblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserblserviceImpl.getById | 语法 | public User getById(int id) |
| 前置条件 | 需要根据id查询post时 |
| 后置条件 | 调用User DataService的getById方法 |
| UserblserviceImpl.add | 语法 | public Post add(Post post) |
| 前置条件 | 需要新增一个User时 |
| 后置条件 | 调用User DataService的insert方法，  获得新增的其他实体类的主键id，设置User对应的外键的属性的值，调用User DataService的insert方法，获得新增User的主键，根据该主键id查询User并返回这个新增的User |
| UserblserviceImpl.update | 语法 | public User update(User user) |
| 前置条件 | 需要更新User时 |
| 后置条件 | 调用User DataService的update方法，  调用User DataService的update方法，获得返回的User的主键，根据该主键id查询User并返回这个User |
| UserblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个User时 |
| 后置条件 | 调用UserDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| departmentDataservice.getList() | 获得所有的department持久化对象 |
| departmentDataservice.getById() | 根据id获得department持久化对象 |
| departmentDataservice.insert() | 增加一个department持久化对象 |
| departmentDataservice.update() | 更新一个department持久化对象 |
| departmentDataservice.delete() | 删除一个department持久化对象 |
|  |  |
| jobDataservice.getList() | 获得所有的job持久化对象 |
| jobDataservice.getById() | 根据id获得job持久化对象 |
| jobDataservice.insert() | 增加一个job持久化对象 |
| jobDataservice.update() | 更新一个job持久化对象 |
| jobDataservice.delete() | 删除一个job持久化对象 |
|  |  |
| userDataservice.getList() | 获得所有的user持久化对象 |
| userDataservice.getById() | 根据id获得user持久化对象 |
| userDataservice.insert() | 增加一个user持久化对象 |
| userDataservice.update() | 更新一个user持久化对象 |
| userDataservice.delete() | 删除一个user持久化对象 |

### 8.2.3人员机构管理动态模型

以新增User为例，当用户保存时User主键为空，则为新增操作，调用add方法，调用Userblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 8.2.4人员机构管理模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 8.2.5度量分析

UserAndDepartmentManagePanel这个类调用的只有Userblservice的4个方法，和一个Userblservice对象引用，

所以CBO为4，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1

Userblservice有UserDataService 用户数据逻辑 JobDataService 工作数据逻辑DepartmentDataService 部门数据逻辑3个对象引用，以及他们的共5\*3=15个方法。

所以CBO为15，DAC为3，UserblserviceImpl实现该接口，NOC为1，DIT为0

# 9审批单据模块设计文档

## 9.1审批单据模块原型界面

### 9.2.1说明

审批单据界面 DocumentApprovalPanel是总经理界面ManagerPanel的一部分，总经理界面通过JtabblePanel分为人员机构管理界面、审批单据界面、制定价格工资常量的界面。下面详细说明人员机构管理界面。

审批单据界面 DocumentApprovalPanel需要在构造方法里传入审批单据业务逻辑接口DocumentApprovalblservice

DocumentApprovalblService定义了查询和审批单据的方法。

其具体实现类DocumentApprovalblserviceimpl需要在构造方法中传入多个数据接口

业务实现需要8个数据接口

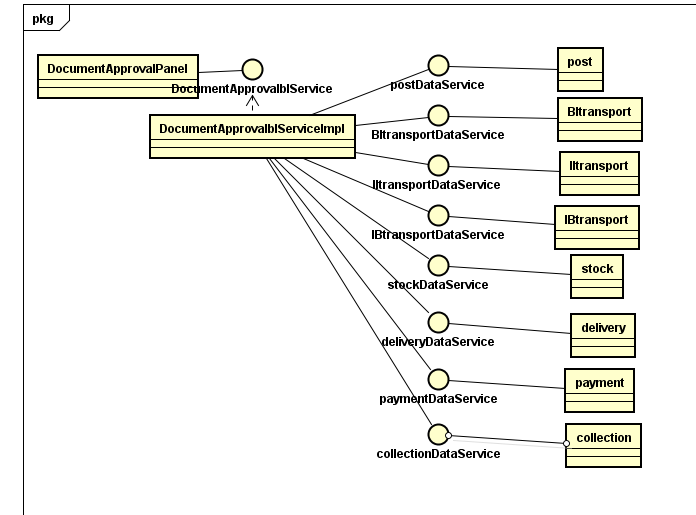
* postDataService 寄件单数据逻辑
* BItransportDataService BI运输表数据逻辑
* IItransportDataService II运输表数据逻辑
* IBtransportDataService IB运输表数据逻辑
* stockDataService 库存单数据逻辑
* deliveryDataService 派件单数据逻辑
* paymentDataService 付款单数据逻辑
* collectionDataService 收款单数据逻辑

以获得方法为例，从数据库获得一个post对象，需要通过postDataService获得post 对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得post里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 9.2.2审批单据详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| DocumentApprovalPanel | DocumentApproval的用户界面 |
| DocumentApprovalblserviceImpl | 负责有关DocumentApproval的业务逻辑实现 |
| DocumentApprovalblservice | DocumentApproval业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于DocumentApproval的controller类，而是封装到Panel中

DocumentApprovalPane**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| DocumentApprovalPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| DocumentApprovalPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的表单显示在表格中 |
| DocumentApprovalPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将结果的表单显示在表格中 |
| DocumentApprovalPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击取消按钮，或者修改、审批操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| DocumentApprovalPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果id  的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| DocumentApprovalPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用各个数据接口的add方法，然后clear清空 |
| DocumentApprovalPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和post的外键值，调用各个数据接口肚饿的update方法，然后clear清空 |
| DocumentApprovalPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的User的id，调用各个数据接口的delete方法，然后clear清空 |
| DocumentApprovalPanel.getPost | 语法 | Private Post getPost () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装post对象 |
| DocumentApprovalPanel.getBItransport | 语法 | private BItransport getBItransport () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装BItransport对象 |
| DocumentApprovalPanel.getIItransport | 语法 | private IItransport getIItransport () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装IItransport对象 |
| DocumentApprovalPanel.geIBtransport | 语法 | private IBtransport getIBtransport () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装IBtransport对象 |
| DocumentApprovalPanel.getstock | 语法 | private stock getstock () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装stock对象 |
| DocumentApprovalPanel.getdelivery | 语法 | private delivery getdelivery () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装delivery对象 |
| DocumentApprovalPanel.getpayment | 语法 | private payment getpayment () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装payment对象 |
| DocumentApprovalPanel.getcollection | 语法 | private collection getcollection () |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装collection对象 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DocumentApprovalblservice | User业务逻辑接口 |
|  |  |

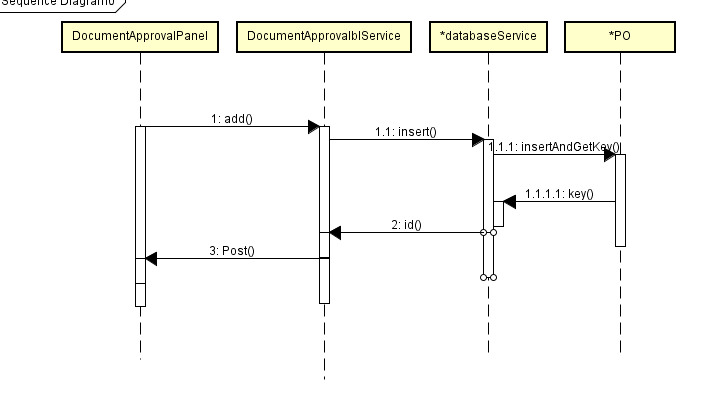
DocumentApprovalServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DocumentApprovalServiceImpl.getById | 语法 | public E getById(int id) |
| 前置条件 | 需要根据id查询表单时 |
| 后置条件 | 调用各个数据接口的的getById方法， |
| DocumentApprovalServiceImpl.add | 语法 | public E add(E e) |
| 前置条件 | 需要新增一个表单时 |
| 后置条件 | 调用各个数据接口的insert方法，  获得新增的其他实体类的主键id，设置表单对应的外键的属性的值，调用各个数据接口的insert方法，获得新增的表单的主键，根据该主键id查询表单并返回这个新增的表单 |
| DocumentApprovalServiceImpl.update | 语法 | public Eupdate(E e) |
| 前置条件 | 需要更新表单时 |
| 后置条件 | 调用各个数据皆苦的update方法，  调用DocumentApprovalService的update方法，获得返回的比导弹的主键，根据该主键id查询表单 并返回这个表单 |
| DocumentApprovalServiceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个表单时 |
| 后置条件 | 调用DocumentApprovalService的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| postDataservice.getList() | 获得所有的post持久化对象 |
| postDataservice.getById() | 根据id获得post持久化对象 |
| postDataservice.insert() | 增加一个post持久化对象 |
| postDataservice.update() | 更新一个 post持久化对象 |
| postDataservice.delete() | 删除一个post持久化对象 |
|  |  |
| BItransportDataservice.getList() | 获得所有的BItransport持久化对象 |
| BItransportDataservice.getById() | 根据id获得BItransport持久化对象 |
| BItransportDataservice.insert() | 增加一个BItransport持久化对象 |
| BItransportDataservice.update() | 更新一个BItransport持久化对象 |
| BItransportDataservice.delete() | 删除一个BItransport持久化对象 |
|  |  |
| IItransportDataservice.getList() | 获得所有的IItransport持久化对象 |
| IItransportDataservice.getById() | 根据id获得IItransport持久化对象 |
| IItransportDataservice.insert() | 增加一个IItransport持久化对象 |
| IItransportDataservice.update() | 更新一个IItransport持久化对象 |
| IItransportDataservice.delete() | 删除一个IItransport持久化对象 |
|  |  |
| IBtransportDataservice.getList() | 获得所有的IBtransport持久化对象 |
| IBtransportDataservice.getById() | 根据id获得IBtransport持久化对象 |
| IBtransportDataservice.insert() | 增加一个IBtransport持久化对象 |
| IBtransportDataservice.update() | 更新一个IBtransport持久化对象 |
| IBtransportDataservice.delete() | 删除一个IBtransport持久化对象 |
|  |  |
| stockDataservice.getList() | 获得所有的stock持久化对象 |
| stockDataservice.getById() | 根据id获得stock持久化对象 |
| stockDataservice.insert() | 增加一个stock持久化对象 |
| stockDataservice.update() | 更新一个stock持久化对象 |
| stockDataservice.delete() | 删除一个stock持久化对象 |
|  |  |
| deliveryDataservice.getList() | 获得所有的delivery持久化对象 |
| deliveryDataservice.getById() | 根据id获得delivery持久化对象 |
| deliveryDataservice.insert() | 增加一个delivery持久化对象 |
| deliveryDataservice.update() | 更新一个delivery持久化对象 |
| deliveryDataservice.delete() | 删除一个delivery持久化对象 |
|  |  |
| paymentDataservice.getList() | 获得所有的payment持久化对象 |
| paymentDataservice.getById() | 根据id获得payment持久化对象 |
| paymentDataservice.insert() | 增加一个payment持久化对象 |
| paymentDataservice.update() | 更新一个payment持久化对象 |
| paymentDataservice.delete() | 删除一个payment持久化对象 |
|  |  |
| collectionDataservice.getList() | 获得所有的collection持久化对象 |
| collectionDataservice.getById() | 根据id获得collection持久化对象 |
| collectionDataservice.insert() | 增加一个collection持久化对象 |
| collectionDataservice.update() | 更新一个collection持久化对象 |
| collectionDataservice.delete() | 删除一个collection持久化对象 |

### 9.2.3审批单据动态模型

以新增post为例，当用户保存时post主键为空，则为新增操作，调用add方法，调用postblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 9.2.4审批单据模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 9.2.5度量分析

DocumentApprovalPanel这个类调用的只有DocumentApprovalblservice的4个方法，和一个DocumentApprovalblservice对象引用，

所以CBO为4，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1

Userblservice有postDataService 寄件单数据逻辑 BItransportDataService BI运输表数据逻辑IItransportDataService II运输表数据逻辑IBtransportDataService IB运输表数据逻辑stockDataService 库存单数据逻辑deliveryDataService 派件单数据逻辑paymentDataService 付款单数据逻辑collectionDataService 收款单数据逻辑8个对象引用，以及他们的共5\*8=40个方法。

所以CBO为40，DAC为8，DocumentApprovalServiceImpl实现该接口，NOC为1，DIT为0

# 10制定常量模块设计文档

## 10.1制定常量原型界面

### 10.2.1说明

制定价格工资常量的界面MakeConstantPanel是总经理界面ManagerPanel的一部分，总经理界面通过JtabblePanel分为人员机构管理界面、审批单据界面、制定价格工资常量的界面。下面详细说明人员机构管理界面。

制定价格工资常量的界面MakeConstantPanel需要在构造方法里传入用户管理业务逻辑接口MakeConstantblservice

MakeConstantblservice定义了基础价格常量的增删改查的方法。

其具体实现类MakeConstantblserviceimpl需要在构造方法中传入多个数据接口

业务实现需要1个数据接口

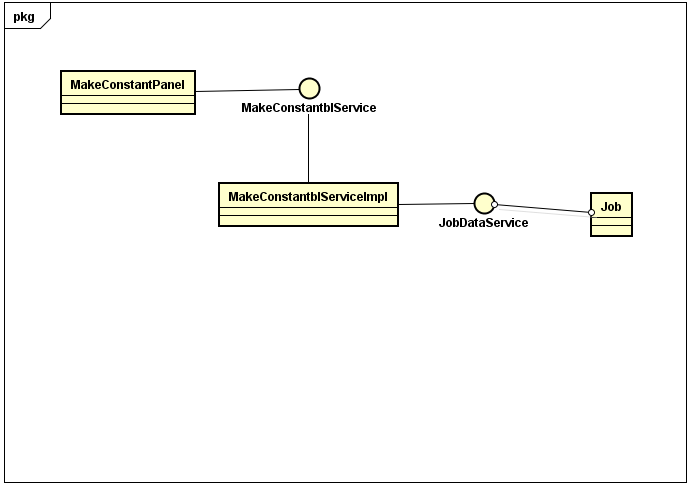
* JobDataService 工作数据逻辑

以获得方法为例，从数据库获得一个Job对象，需要通过JobDataService获得Job对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得Job里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 10.2.2人员机构管理详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| MakeConstantPanel | MakeConstant的用户界面 |
| MakeConstantrblserviceImpl | 负责有关MakeConstant的业务逻辑实现 |
| MakeConstantblservice | MakeConstant业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于MakeConstant的controller类，而是封装到Panel中

**MakeConstantPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| MakeConstantPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| MakeConstantPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的Job显示在表格中 |
| MakeConstantPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将结果的Job显示在表格中 |
| MakeConstantPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击取消按钮，或者Jobr添加，修改，删除，查询操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| MakeConstantPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果JobId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| MakeConstantPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用MakeConstantblservice的add方法，然后clear清空 |
| MakeConstantPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和Job的外键值，调用MakeConstantblService的update方法，然后clear清空 |
| MakeConstantPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的Job的id，调用MakeConstantblservice的delete方法，然后clear清空 |
| MakeConstantPanel.getJob | 语法 | private Job getJob() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装Job对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| MakeConstantblservice | User业务逻辑接口 |
|  |  |

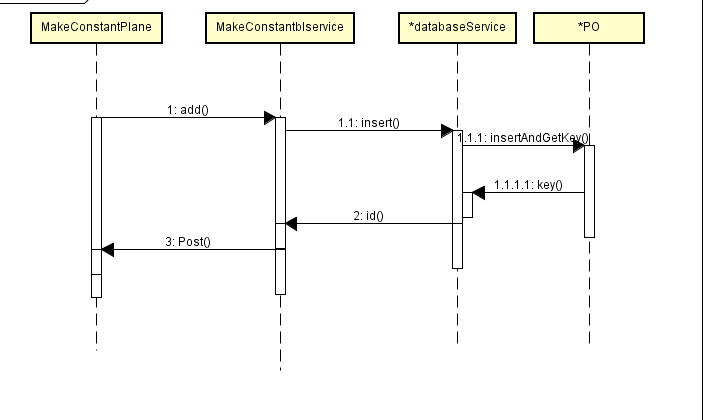
**MakeConstantblservicImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| MakeConstantblservice Impl.getById | 语法 | public Job getById(int id) |
| 前置条件 | 需要根据id查询Job时 |
| 后置条件 | 调用JobDataService的getById方法 |
| MakeConstantblservice Impl.add | 语法 | public Post add(Post post) |
| 前置条件 | 需要新增一个Job时 |
| 后置条件 | 调用JobDataService的insert方法，  获得新增的其他实体类的主键id，设置Job对应的外键的属性的值，调用JobDataService的insert方法，获得新增Job的主键，根据该主键id查询Job并返回这个新增的Job |
| MakeConstantblservice Impl.update | 语法 | public Job update(Job job) |
| 前置条件 | 需要更新Job时 |
| 后置条件 | 调用JobDataService的update方法，  调用JobDataService的update方法，获得返回的Job的主键，根据该主键id查询Job并返回这个Job |
| MakeConstantblservice Impl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个Job时 |
| 后置条件 | 调用JobDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| jobDataservice.getList() | 获得所有的job持久化对象 |
| jobDataservice.getById() | 根据id获得job持久化对象 |
| jobDataservice.insert() | 增加一个job持久化对象 |
| jobDataservice.update() | 更新一个job持久化对象 |
| jobDataservice.delete() | 删除一个job持久化对象 |

### 10.2.3人员机构管理动态模型

以新增Job为例，当用户保存时Job主键为空，则为新增操作，调用add方法，调用MakeConstantblservice的add方法，调用各个Jobservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 10.2.4人员机构管理模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 10.2.5度量分析

MakeConstantPanel这个类调用的只有MakeConstantblservice的4个方法，和一个MakeConstantblservice对象引用，

所以CBO为4，DAC为1，继承一个commonpanel，无子类，NOC为0，DIT为1

MakeConstantblservice有JobDataService 工作数据逻辑1个对象引用，以及他们的共5个方法。

所以CBO为5，DAC为1，MakeConstantrblserviceImpl实现该接口，NOC为1，DIT为0

# 11.管理员模块设计文档

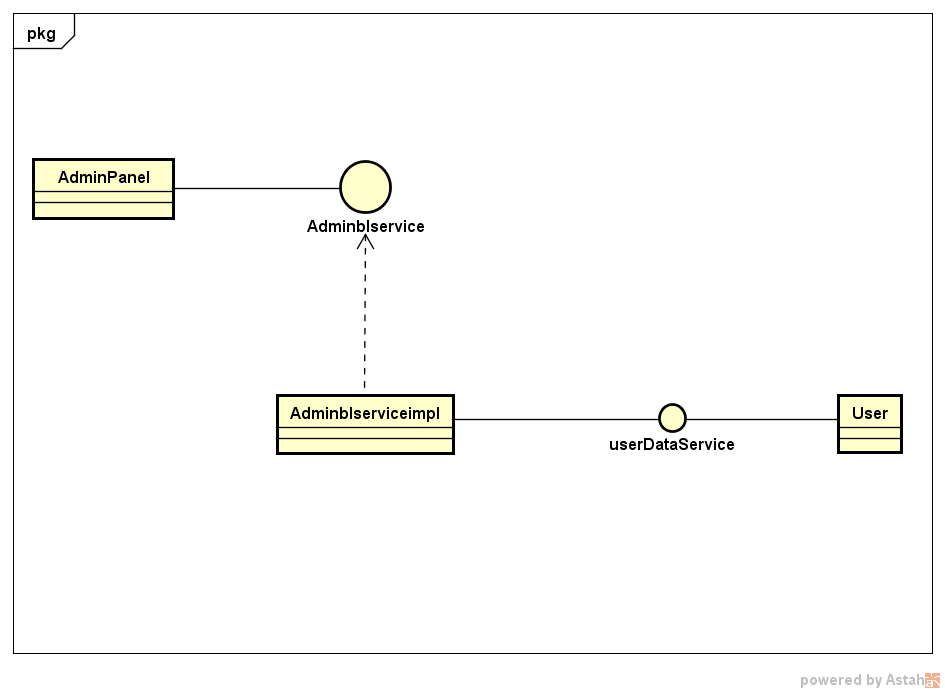
## 11.1管理员模块原型界面

### 11.2.1说明

管理员的用户权限管理界面AdminPanel需要在构造方法里面传入管理员业务逻辑接口Userblservice .Userblservice定义了UserData以及User的权限

其具体实现类Userblserviceimpl需要在构造方法中传入多个数据接口

### 11.2.2用户权限管理详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| AdminPanel | Admin的用户界面 |
| AdminblserviceImpl | 负责有关Admin的业务逻辑实现 |
| Adminblservice | Admin业务逻辑接口，定义需要的业务方法 |
| \*Dataservice | 各种实体类的数据接口，定义数据库操作方法 |

**postPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AdminPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| AdminPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的UserData显示在表格中 |
| AdminPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的UserData显示在表格中 |
| 提供的服务（供接口） | | |
| PostPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者post添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| AdminPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果UserId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| AdminPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用Userblservice的add方法，然后clear清空 |
| AdminPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和post的外键值，调用Adminblservice的update方法，然后clear清空 |
| AdminPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的post的id，调用Adminblservice的delete方法，然后clear清空 |
| AdminPanel.getUser | 语法 | private Post getPost() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装User对象 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| postblservice | post业务逻辑接口 |
|  |  |

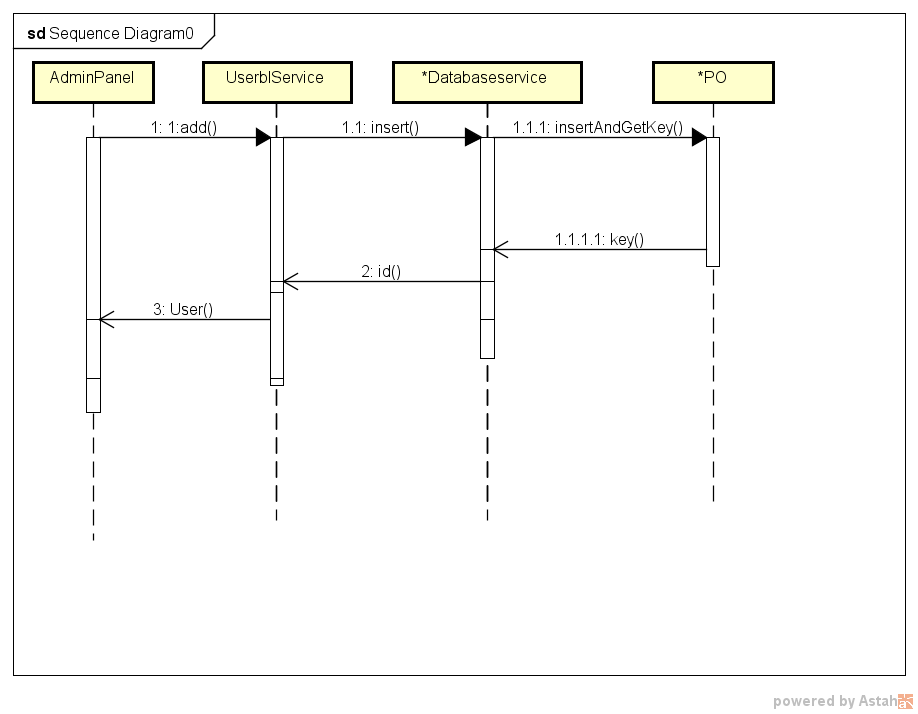
Admin**blserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AdminblserviceImpl.getAll | 语法 | public Vector<Post> getAll() |
| 前置条件 | 需要查询所有的User时 |
| 后置条件 | 调用UserDataService的getList方法，对获得的每一个User调用UserDataDataservice的getById方法 |
| AdminblserviceImpl.getById | 语法 | public Post getById(int id) |
| 前置条件 | 需要根据id查询User时 |
| 后置条件 | 调用UserDataService的getById方法 |
| AdminblserviceImpl.add | 语法 | public User add(Post post) |
| 前置条件 | 需要新增一个post时 |
| 后置条件 | 调用UserDataService的insert方法，  获得新增的其他实体类的主键id，设置User对应的外键的属性的值，调用UserDataService的insert方法，获得新增的User的主键，根据该主键id查询User并返回这个新增的User |
| AdminblserviceImpl.update | 语法 | public Post update(Post post) |
| 前置条件 | 需要更新User时 |
| 后置条件 | 调用senderDataService，receiverDataservice，goodsDataservice，collectionDatas的update方法，  调用postDataService的update方法，获得返回的post的主键，根据该主键id查询post并返回这个post |
| AdminblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个User时 |
| 后置条件 | 调用UserDataservice的delete方法 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| UserDataservice.getList() | 获得所有的User持久化对象 |
| UserDataservice.getById() | 根据id获得User持久化对象 |
| UserDataservice.insert() | 增加一个User持久化对象 |
| UserDataservice.update() | 更新一个User持久化对象 |
| UserDataservice.delete() | 删除一个User持久化对象 |

### 11.2.3 动态模型

新增一个User，调用add方法，调用UserDataService的insert方法，通过数据库insert每个PO对象，返回id

****

### 11.2.4模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 11.2.5度量计算部分：

adminpanel 这个类调用的只有userblservice的六个方法，和一个userblservice对象引用

CBD=6、DAC=1。继承一个commonpanel 无子类，NOC=0，DIT=1

adminblService有userDataservice一个对象引用，和这个对象的四个方法。

CBO=4,DAC1。userblseviceImpl实现该接口，NOC=1，DIT=0.

# 12车辆管理模块

## 12.1车辆管理原型界面



### 12.2.1说明

车辆管理界面BusinessVehiclePanel是营业厅业务员界面BusinessPanel的一部分,营业厅业务员界面通过JtabblePanel分为多个界面。下面详细说明车辆管理界面界面。

车辆管理界面BusinessVehiclepanel需要在构造方法里传入车辆管理业务逻辑接口BusinessVehicleblservice

BusinessVehicleblService定义了BusinessVehicle 的增删改查，以及获得快递员user的方法。

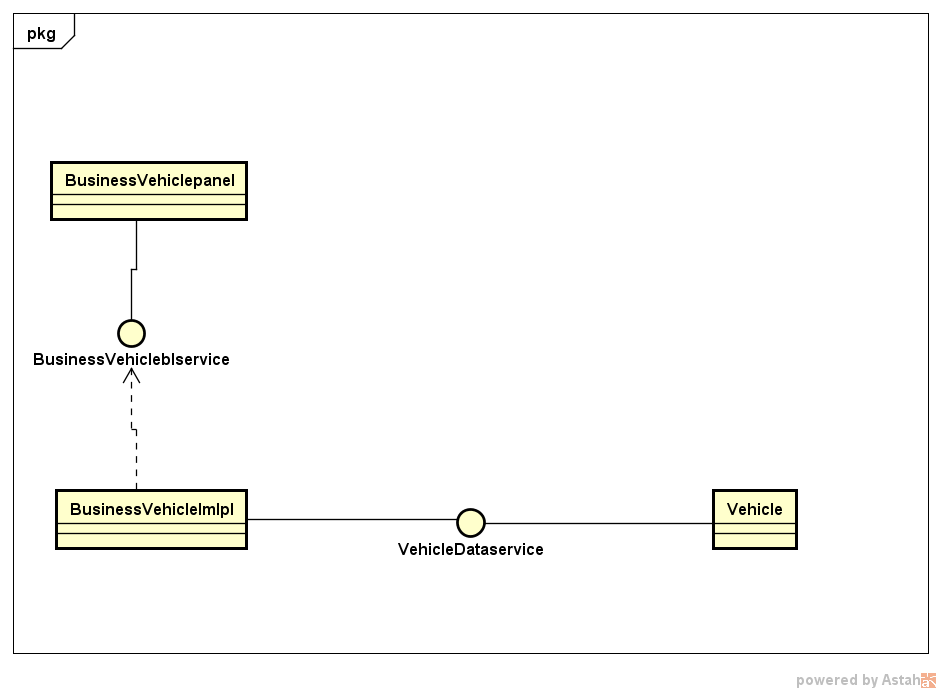
其具体实现类BusinessVehicleblserviceimpl需要在构造方法中传入多个数据接口

以获得方法为例，从数据库获得一个BusinessVehicle对象，需要通过BusinessVehicleDataService获得BusinessVehicle对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得BusinessVehicle里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 12.2.2车辆管理详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| BusinessVehiclePanel | BusinessVehicle的用户界面 |
| BusinessVehicleblserviceImpl | 负责有关BusinessVehicle的业务逻辑实现 |
| BusinessVehicleblservice | BusinessVehicle业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于BusinessVehicle的controller类，而是封装到Panel中

**BusinessVehiclePanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessVehiclePanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| BusinessVehiclePanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的BusinessVehicle显示在表格中 |
| BusinessVehiclePanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的BusinessVehicle显示在表格中 |
| **提供的服务（供接口）** | | |
| BusinessVehiclePanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者BusinessVehicle添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| BusinessVehiclePanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果BusinessVehicleId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| BusinessVehiclePanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用BusinessVehicleblservice的add方法，然后clear清空 |
| BusinessVehiclePanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和BusinessVehicle的外键值，调用BusinessVehicleblservice的update方法，然后clear清空 |
| BusinessVehiclePanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的BusinessVehicle的id，调用BusinessVehicleblservice的delete方法，然后clear清空 |
| BusinessVehiclePanel.getBusinessVehicle | 语法 | private BusinessVehicle getBusinessVehicle() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装BusinessVehicle对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessVehicleblservice | BusinessVehicle业务逻辑接口 |
|  |  |

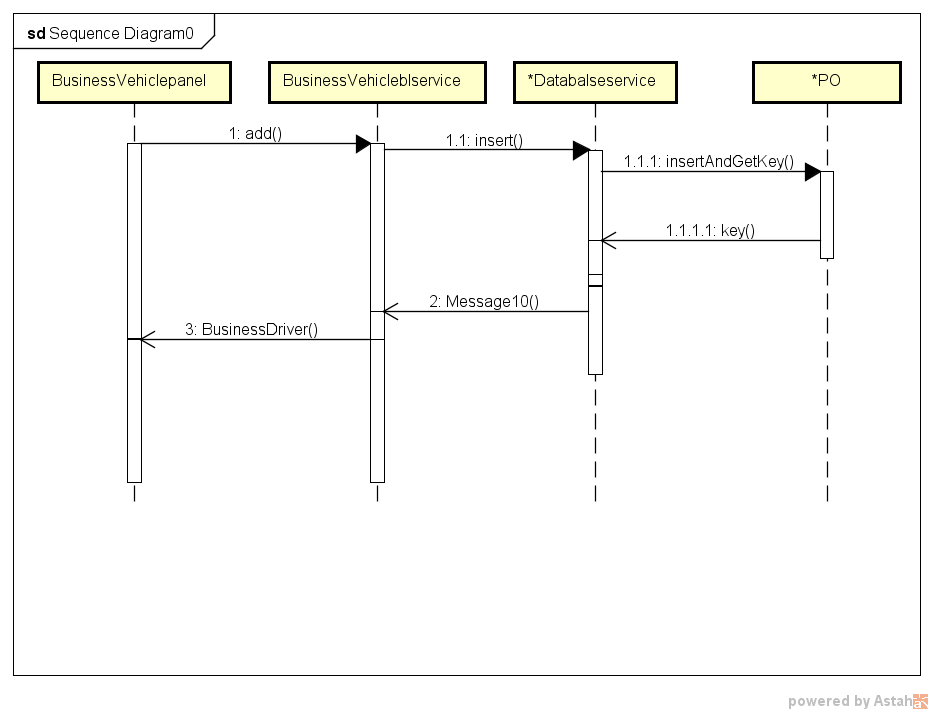
**BusinessVehicleblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessVehicleblserviceImpl.getAll | 语法 | public Vector<BusinessVehicle> getAll() |
| 前置条件 | 需要查询所有的BusinessVehicle时 |
| 后置条件 | 调用BusinessVehicleDataService的getList方法 |
| BusinessVehicleblserviceImpl.getById | 语法 | public BusinessVehicle getById(int id) |
| 前置条件 | 需要根据id查询BusinessVehicle时 |
| 后置条件 | 调用BusinessVehicleDataService的getById方法 |
| BusinessVehicleblserviceImpl.add | 语法 | public BusinessVehicle add(BusinessVehicle BusinessVehicle) |
| 前置条件 | 需要新增一个BusinessVehicle时 |
| 后置条件 | 设置BusinessVehicle对应的外键的属性的值，调用BusinessVehicleDataService的insert方法，获得新增的BusinessVehicle的主键，根据该主键id查询BusinessVehicle并返回这个新增的BusinessVehicle |
| BusinessVehicleblserviceImpl.update | 语法 | public BusinessVehicle update(BusinessVehicle BusinessVehicle) |
| 前置条件 | 需要更新BusinessVehicle时 |
| 后置条件 | 调用BusinessVehicleDataService的update方法，获得返回的BusinessVehicle的主键，根据该主键id查询BusinessVehicle并返回这个BusinessVehicle |
| BusinessVehicleblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个BusinessVehicle时 |
| 后置条件 | 调用BusinessVehicleDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessVehicleDataservice.getList() | 获得所有的BusinessVehicle持久化对象 |
| BusinessVehicleDataservice.getById() | 根据id获得BusinessVehicle持久化对象 |
| BusinessVehicleDataservice.insert() | 增加一个BusinessVehicle持久化对象 |
| BusinessVehicleDataservice.update() | 更新一个BusinessVehicle持久化对象 |
| BusinessVehicleDataservice.delete() | 删除一个BusinessVehicle持久化对象 |
|  |  |

### 12.2.3车辆管理单动态模型

以新增BusinessVehicle为例，当用户保存时BusinessVehicle主键为空，则为新增操作，调用add方法，调用BusinessVehicleblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 12.2.4车辆管理单模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 12.2.5度量计算部分：

businessVehiclepanel这个类调用的只有businessVehicleblservice的六个方法，和一个businessVehicleblservice对象引用

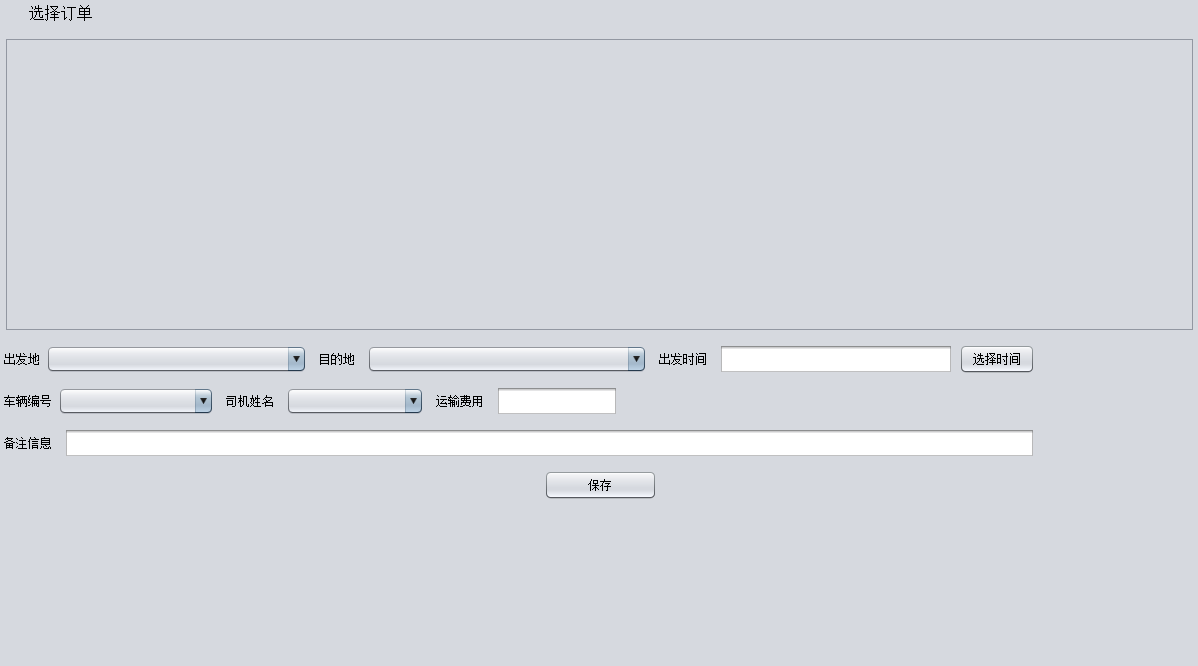
CBD=6、DAC=1。继承一个commonpanel 无子类，NOC=0，DIT=1

vehicleblService有vehicleDataservice一个对象引用，和这个对象的四个方法。

CBO=4,DAC1。vehicleblseviceImpl实现该接口，NOC=1，DIT=0.

# 13车辆装车管理模块

## 13.1车辆装车管理原型界面



### 13.2.1说明

车辆装车管理界面BItranspotPanel是营业厅业务员界面BusinessPanel的一部分，营业厅业务员界面通过JtabblePanel分为多个界面。下面详细说明车辆装车管理单界面。

车辆装车管理界面BItranspotpanel需要在构造方法里传入车辆装车管理业务逻辑接口BItranspotblservice

BItranspotblService定义了BItranspot装车单的增删改查，以及获得快递员user的方法。

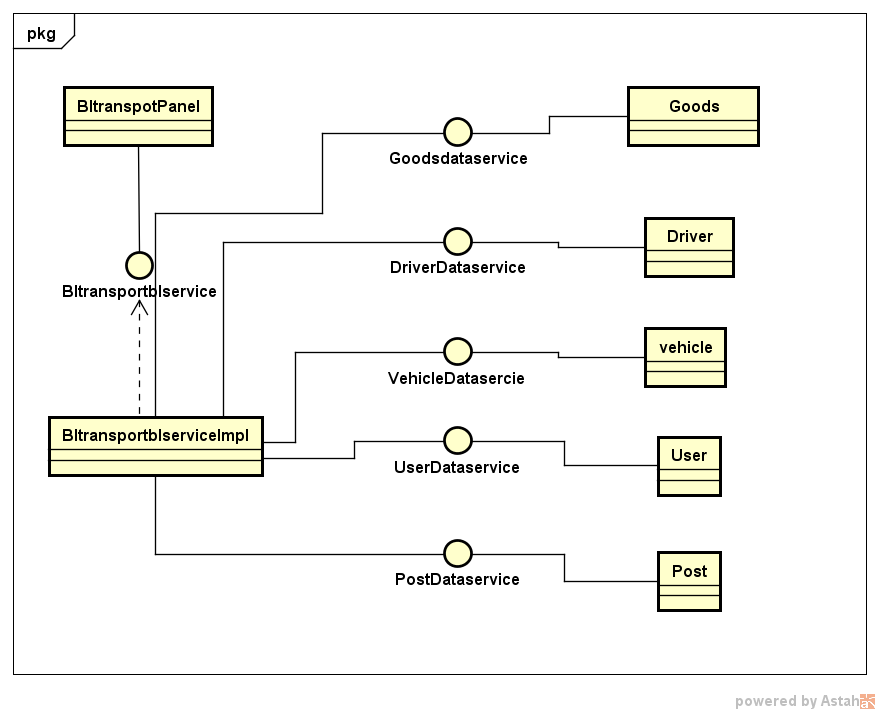
其具体实现类BItranspotblserviceimpl需要在构造方法中传入多个数据接口

以获得方法为例，从数据库获得一个BItranspot对象，需要通过BItranspotDataService获得BItranspot对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得BItranspot里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 13.2.2装车单详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| BItranspotPanel | BItranspot的用户界面 |
| BItranspotblserviceImpl | 负责有关BItranspot的业务逻辑实现 |
| BItranspotblservice | BItranspot业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于BItranspot的controller类，而是封装到Panel中

**BItranspotPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BItranspotPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| BItranspotPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的BItranspot显示在表格中 |
| BItranspotPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的BItranspot显示在表格中 |
| **提供的服务（供接口）** | | |
| BItranspotPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者BItranspot添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| BItranspotPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果BItranspotId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| BItranspotPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用BItranspotblservice的add方法，然后clear清空 |
| BItranspotPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和BItranspot的外键值，调用BItranspotblservice的update方法，然后clear清空 |
| BItranspotPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的BItranspot的id，调用BItranspotblservice的delete方法，然后clear清空 |
| BItranspotPanel.getBItranspot | 语法 | private BItranspot getBItranspot() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装BItranspot对象 |
| BItranspotPanel.getVehicle | 语法 | private Vehicle getVehicle() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装Vehicle对象 |
| BItranspotPanel.getVehicle | 语法 | private Receiver getVehicle() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装Vehicle对象 |
| BItranspotPanel.getGoods | 语法 | private Goods getGoods() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装goods对象 |
| BItranspotPanel.getPost | 语法 | private Post getPost() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装post对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BItranspotblservice | BItranspot业务逻辑接口 |
|  |  |

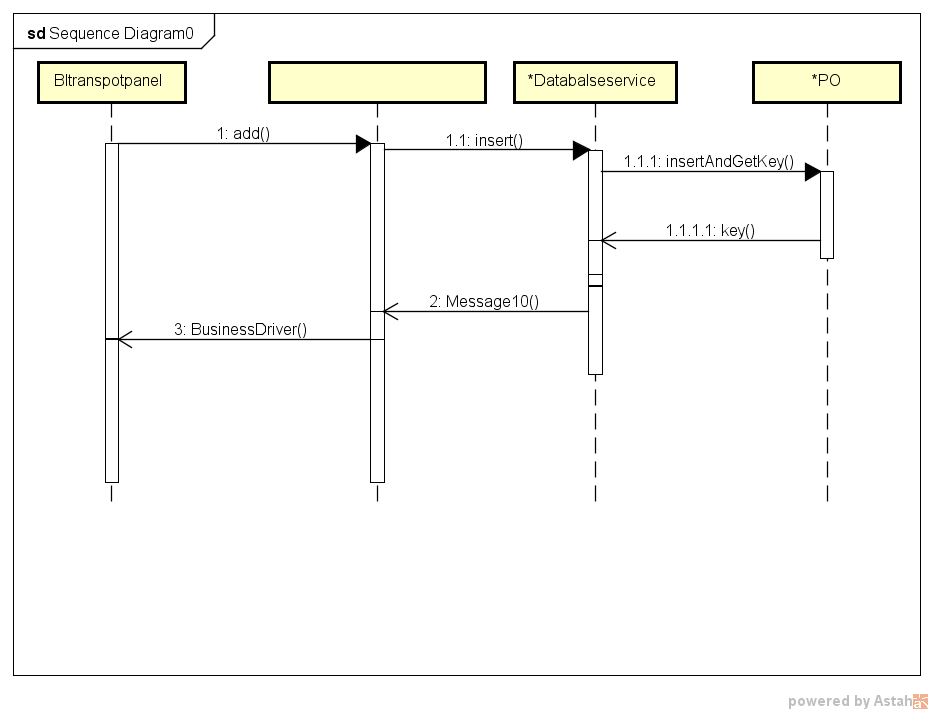
**BItranspotblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BItranspotblserviceImpl.getAll | 语法 | public Vector<BItranspot> getAll() |
| 前置条件 | 需要查询所有的BItranspot时 |
| 后置条件 | 调用BItranspotDataService的getList方法，对获得的每一个BItranspot调用VehicleDataService，VehicleDataservice，goodsDataservice，postDatas的getById方法，对获得的post调用userDataservice的getById方法 |
| BItranspotblserviceImpl.getById | 语法 | public BItranspot getById(int id) |
| 前置条件 | 需要根据id查询BItranspot时 |
| 后置条件 | 调用BItranspotDataService的getById方法，对获得的BItranspot调用VehicleDataService，VehicleDataservice，goodsDataservice，postDatas的getById方法，对获得的post调用userDataservice的getById方法 |
| BItranspotblserviceImpl.add | 语法 | public BItranspot add(BItranspot BItranspot) |
| 前置条件 | 需要新增一个BItranspot时 |
| 后置条件 | 调用VehicleDataService，VehicleDataservice，goodsDataservice，postDatas的insert方法，  获得新增的其他实体类的主键id，设置BItranspot对应的外键的属性的值，调用BItranspotDataService的insert方法，获得新增的BItranspot的主键，根据该主键id查询BItranspot并返回这个新增的BItranspot |
| BItranspotblserviceImpl.update | 语法 | public BItranspot update(BItranspot BItranspot) |
| 前置条件 | 需要更新BItranspot时 |
| 后置条件 | 调用VehicleDataService，VehicleDataservice，goodsDataservice，postDatas的update方法，  调用BItranspotDataService的update方法，获得返回的BItranspot的主键，根据该主键id查询BItranspot并返回这个BItranspot |
| BItranspotblserviceImpl.getCourier | 语法 | public Vector<User> getCourier() |
| 前置条件 | 需要获得BItranspot中post的快递员user时 |
| 后置条件 | 调用userDataservice的getUserOfJob方法，获得返回的user集合，说明：该方法本应该是postblservice的方法，但BItranspotblservice也需要这个方法，写在这里可以节省代码 |
| BItranspotblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个BItranspot时 |
| 后置条件 | 调用BItranspotDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BItranspotDataservice.getList() | 获得所有的BItranspot持久化对象 |
| BItranspotDataservice.getById() | 根据id获得BItranspot持久化对象 |
| BItranspotDataservice.insert() | 增加一个BItranspot持久化对象 |
| BItranspotDataservice.update() | 更新一个BItranspot持久化对象 |
| BItranspotDataservice.delete() | 删除一个BItranspot持久化对象 |
|  |  |
| VehicleDataservice.getList() | 获得所有的Vehicle持久化对象 |
| VehicleDataservice.getById() | 根据id获得Vehicle持久化对象 |
| VehicleDataservice.insert() | 增加一个Vehicle持久化对象 |
| VehicleDataservice.update() | 更新一个Vehicle持久化对象 |
| VehicleDataservice.delete() | 删除一个Vehicle持久化对象 |
|  |  |
| DriverDataservice.getList() | 获得所有的Driver持久化对象 |
| DriverDataservice.getById() | 根据id获得Driver持久化对象 |
| DriverDataservice.insert() | 增加一个Driver持久化对象 |
| DriverDataservice.update() | 更新一个Driver持久化对象 |
| DriverDataservice.delete() | 删除一个Driver持久化对象 |
|  |  |
| goodsDataservice.getList() | 获得所有的goods持久化对象 |
| goodsDataservice.getById() | 根据id获得goods持久化对象 |
| goodsDataservice.insert() | 增加一个goods持久化对象 |
| goodsDataservice.update() | 更新一个goods持久化对象 |
| goodsDataservice.delete() | 删除一个goods持久化对象 |
|  |  |
| postDataservice.getList() | 获得所有的post持久化对象 |
| postDataservice.getById() | 根据id获得post持久化对象 |
| postDataservice.insert() | 增加一个post持久化对象 |
| postDataservice.update() | 更新一个post持久化对象 |
| postDataservice.delete() | 删除一个post持久化对象 |
|  |  |
| userDataservice.getList() | 获得所有的user持久化对象 |
| userDataservice.getById() | 根据id获得user持久化对象 |
| userDataservice.insert() | 增加一个user持久化对象 |
| userDataservice.update() | 更新一个user持久化对象 |
| userDataservice.delete() | 删除一个user持久化对象 |

### 13.2.3装车单动态模型

以新增BItranspot为例，当用户保存时BItranspot主键为空，则为新增操作，调用add方法，调用BItranspotblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 13.2.4装车单模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 13.2.5度量计算部分：

BlTransportpanel 这个类调用的只有BITransportblservice的六个方法，和一个BITransportblservice对象引用

CBD=6、DAC=1。继承一个commonpanel 无子类，NOC=0，DIT=1

BITransportblService有userDataservice，goodsDataservice,vehicleDataservice,userDataservice,

DriverDataservice五个对象引用，和这个对象的四个方法。

CBO=20,DAC=1。BITransportblseviceImpl实现该接口，NOC=1，DIT=0.

# 14派件单模块

## 14.1派件单原型界面



### 14.2.1说明

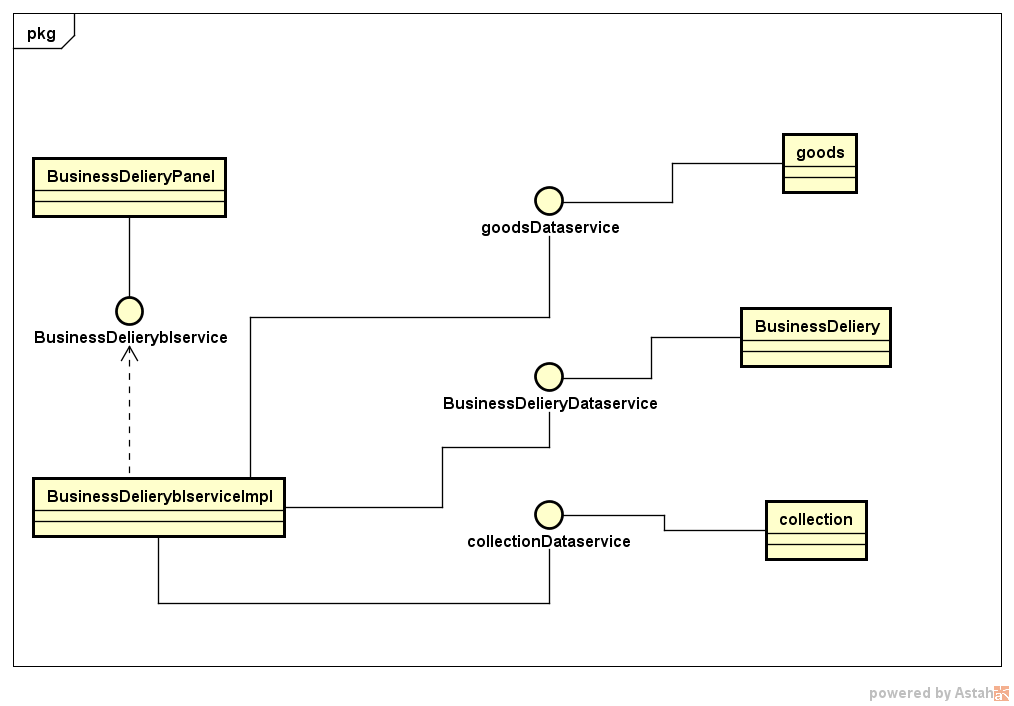
派件单界面BusinessDelieryPanel营业厅业务员界面BusinessPanel的一部分，营业厅业务员通过JtabblePanel分为多个界面（车辆管理界面(BusinessVehiclePanel)，司机管理界面(BusinessDriverPanel), 收款单界面(BusinessColletionPanel)，装车管理界面(BItransportPanel), 派件单界面(BusinessDelieryPanel)），下面详细说明派件单界面

收款单界面BusinessDelieryPanel需要在构造方法里传入收款单业务逻辑接口BusinessDelieryblservice，它定义了派件单的增删改查。其具体实现类BusinessDelieryblserviceimpl需要在构造方法中传入

以获得方法为例，从数据库获得一个BusinessDeliery对象，需要通过BusinessDelieryDataService获得BusinessDeliery对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得BusinessDeliery里包含的其他对象

若其他对象还包含着其他对象，需要继续封装



### 14.2.2派件单详细设计描述

|  |  |
| --- | --- |
| 模块 | 职责 |
| BusinessDelieryPanel | BusinessDeliery的用户界面 |
| BusinessDelieryblserviceImpl | 负责有关BusinessDeliery的业务逻辑实现 |
| BusinessDelieryblservice | BusinessDeliery业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

BusinessDelieryPanel**的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessDelieryPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| BusinessDelieryPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的BusinessDeliery显示在表格中 |
| BusinessDelieryPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的BusinessDeliery显示在表格中 |
| **提供的服务（供接口）** | | |
| BusinessDelieryPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者BusinessDeliery添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| BusinessDelieryPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果BusinessDelieryId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| BusinessDelieryPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用BusinessDelieryblservice的add方法，然后clear清空 |
| BusinessDelieryPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和BusinessDeliery的外键值，调用BusinessDelieryblservice的update方法，然后clear清空 |
| BusinessDelieryPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的BusinessDeliery的id，调用BusinessDelieryblservice的delete方法，然后clear清空 |
| BusinessDelieryPanel.getBusinessDeliery | 语法 | private BusinessDeliery getBusinessDeliery() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装BusinessDeliery对象 |
| BusinessDelieryPanel.getGoods | 语法 | private Goods getGoods() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装goods对象 |
| BusinessDelieryPanel.getCollection | 语法 | private Collection getCollection() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装collection对象 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessDelieryblservice | BusinessDeliery业务逻辑接口 |
|  |  |

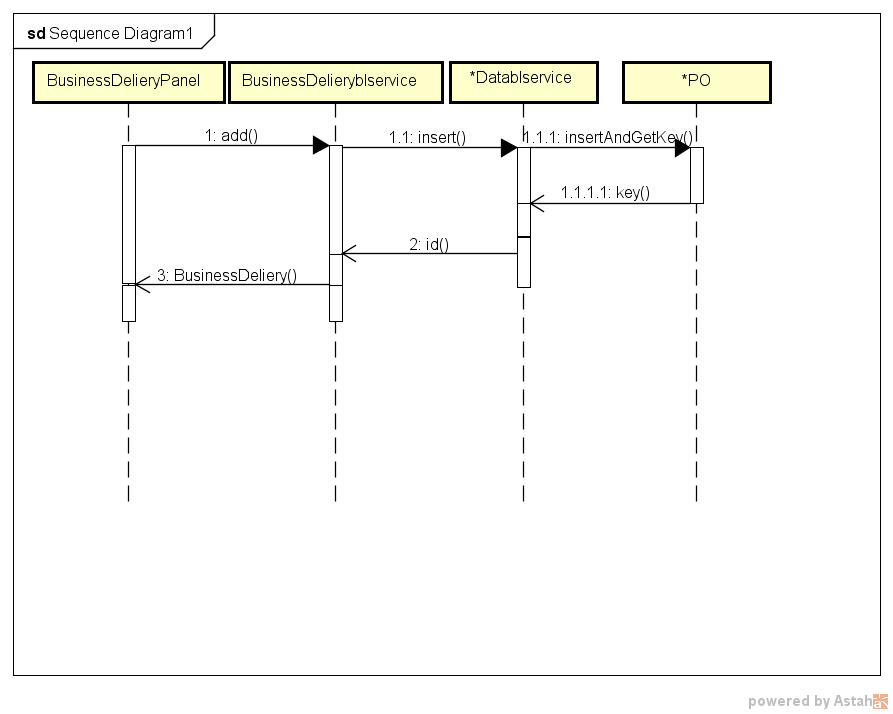
**BusinessDelieryblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessDelieryblserviceImpl.getAll | 语法 | public Vector<BusinessDeliery> getAll() |
| 前置条件 | 需要查询所有的BusinessDeliery时 |
| 后置条件 | 调用BusinessDelieryDataService的getList方法， collectionDatas的getById方法，对获得的collection调用userDataservice的getById方法 |
| BusinessDelieryblserviceImpl.getById | 语法 | public BusinessDeliery getById(int id) |
| 前置条件 | 需要根据id查询BusinessDeliery时 |
| 后置条件 | 调用BusinessDelieryDataService的getById方法 goodsDataservice，collectionDatas的getById方法， |
| BusinessDelieryblserviceImpl.add | 语法 | public BusinessDeliery add(BusinessDeliery BusinessDeliery) |
| 前置条件 | 需要新增一个BusinessDeliery时 |
| 后置条件 | 调用 goodsDataservice，collectionDatas的insert方法，  获得新增的其他实体类的主键id，设置BusinessDeliery对应的外键的属性的值，调用BusinessDelieryDataService的insert方法，获得新增的BusinessDeliery的主键，根据该主键id查询BusinessDeliery并返回这个新增的BusinessDeliery |
| BusinessDelieryblserviceImpl.update | 语法 | public BusinessDeliery update(BusinessDeliery BusinessDeliery) |
| 前置条件 | 需要更新BusinessDeliery时 |
| 后置条件 | 调用 goodsDataservice，collectionDatas的update方法，  调用BusinessDelieryDataService的update方法，获得返回的BusinessDeliery的主键，根据该主键id查询BusinessDeliery并返回这个BusinessDeliery |
| BusinessDelieryblserviceImpl.getCourier | 语法 | public Vector<User> getCourier() |
| 前置条件 | 需要获得BusinessDeliery中collection的快递员user时 |
| 后置条件 | 调用userDataservice的getUserOfJob方法，获得返回的user集合，说明：该方法本应该是collectionblservice的方法，但BusinessDelieryblservice也需要这个方法，写在这里可以节省代码 |
| BusinessDelieryblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个BusinessDeliery时 |
| 后置条件 | 调用BusinessDelieryDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessDelieryDataservice.getList() | 获得所有的BusinessDeliery持久化对象 |
| BusinessDelieryDataservice.getById() | 根据id获得BusinessDeliery持久化对象 |
| BusinessDelieryDataservice.insert() | 增加一个BusinessDeliery持久化对象 |
| BusinessDelieryDataservice.update() | 更新一个BusinessDeliery持久化对象 |
| BusinessDelieryDataservice.delete() | 删除一个BusinessDeliery持久化对象 |
|  |  |
|  |  |
| goodsDataservice.getList() | 获得所有的goods持久化对象 |
| goodsDataservice.getById() | 根据id获得goods持久化对象 |
| goodsDataservice.insert() | 增加一个goods持久化对象 |
| goodsDataservice.update() | 更新一个goods持久化对象 |
| goodsDataservice.delete() | 删除一个goods持久化对象 |
|  |  |
| collectionDataservice.getList() | 获得所有的collection持久化对象 |
| collectionDataservice.getById() | 根据id获得collection持久化对象 |
| collectionDataservice.insert() | 增加一个collection持久化对象 |
| collectionDataservice.update() | 更新一个collection持久化对象 |
| collectionDataservice.delete() | 删除一个collection持久化对象 |
|  |  |

### 14.2.3派件单动态模型

以新增派件单BusinessDeliery为例，当用户保存时BusinessDeliery主键为空，则新增操作，调用调用add方法，调用BusinessDelieryblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。



### 14.2.4模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合的特点。

### 14.2.5度量计算部分：

BusinessDeliverypanel 这个类调用的只有BusinessDeliveryblservice的六个方法，和一个BusinessDeliveryblservice对象引用

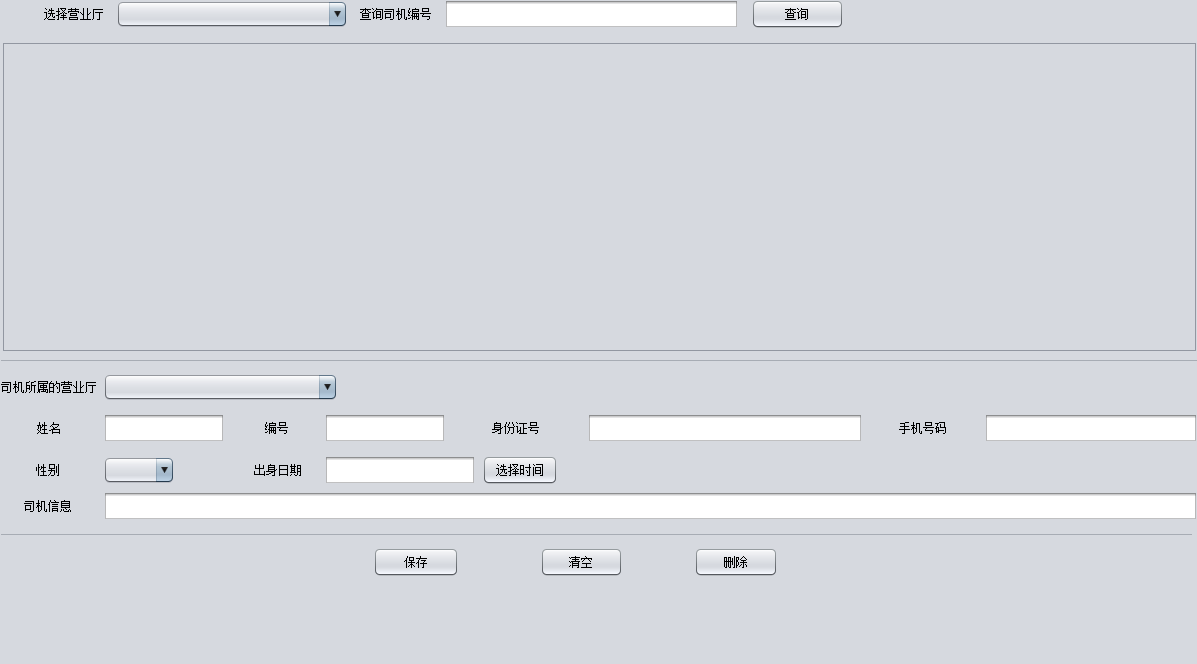
CBD=6、DAC=1。继承一个commonpanel 无子类，NOC=0，DIT=1

BusinessDeliveryblServic有BusinessDeliveryDataservice,goodsDataservice,collectionDataservice三个对象引用，和这个对象的三个方法。

CBO=9,DAC=1。BusinessDeliveryblseviceImpl实现该接口，NOC=1，DIT=0.

# 15司机信息管理模块

## 15.1司机信息管理原型界面



### 15.2.1说明

司机信息管理界面BusinessDriverPanel是营业厅业务员界面BusinessPanel的一部分,营业厅业务员界面通过JtabblePanel分为多个界面。下面详细说明司机信息管理界面界面。

司机信息管理界面BusinessDriverpanel需要在构造方法里传入司机信息管理业务逻辑接口BusinessDriverblservice

BusinessDriverblService定义了BusinessDriver 的增删改查。

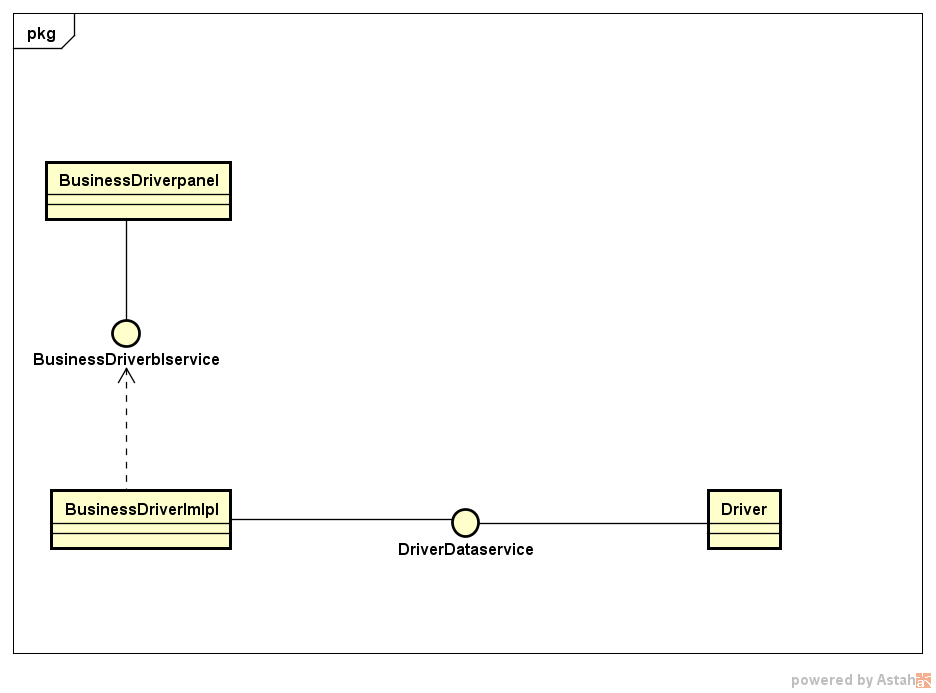
其具体实现类BusinessDriverblserviceimpl需要在构造方法中传入多个数据接口

以获得方法为例，从数据库获得一个BusinessDriver对象，需要通过BusinessDriverDataService获得BusinessDriver对象（只包含其本身的属性，而不包含其他的对象），

再根据其外键的id获得BusinessDriver里包含的其他对象

若其他对象还包含着其他对象，需要继续封装

### 15.2.2寄件单详细设计描述



|  |  |
| --- | --- |
| 模块 | 职责 |
| BusinessDriverPanel | BusinessDriver的用户界面 |
| BusinessDriverblserviceImpl | 负责有关BusinessDriver的业务逻辑实现 |
| BusinessDriverblservice | BusinessDriver业务逻辑接口，定义需要的业务方法 |
| \*Dateservice | 各种实体类的数据接口，定义数据库操作方法 |

没有写关于BusinessDriver的controller类，而是封装到Panel中

**BusinessDriverPanel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessDriverPanel.view | 语法 | private void view() |
| 前置条件 | 用户点击表格的某一行 |
| 后置条件 | 根据选择的id调用查询方法，然后显示在下部的界面详细显示 |
| BusinessDriverPanel.query | 语法 | private void query() |
| 前置条件 | 用户点击查询按钮 |
| 后置条件 | 根据查询文本框调用查询方法，然后将符合查询条件的BusinessDriver显示在表格中 |
| BusinessDriverPanel.setViewData | 语法 | public void setViewData() |
| 前置条件 | 界面初始化 |
| 后置条件 | 调用查询方法，然后将所有的BusinessDriver显示在表格中 |
| **提供的服务（供接口）** | | |
| BusinessDriverPanel.clear | 语法 | private void clear() |
| 前置条件 | 用户点击清空按钮，或者BusinessDriver添加，修改，删除操作完成后 |
| 后置条件 | 清空下面的文本框和按钮，重新获得数据，重新获得提示信息 |
| BusinessDriverPanel.save | 语法 | private void save() |
| 前置条件 | 用户点击保存按钮 |
| 后置条件 | 如果BusinessDriverId的文本框(隐藏)的值为空, 则是新增,调用add方法，否则为修改，调用update方法 |
| BusinessDriverPanel.add | 语法 | private void add() |
| 前置条件 | 用户点击保存按钮后是新增操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 调用BusinessDriverblservice的add方法，然后clear清空 |
| BusinessDriverPanel.update | 语法 | private void update() |
| 前置条件 | 用户点击保存按钮后是更新操作，检查文本框内容是否存在以及合法 |
| 后置条件 | 封装其他实体类的主键值和BusinessDriver的外键值，调用BusinessDriverblservice的update方法，然后clear清空 |
| BusinessDriverPanel.delete | 语法 | private void delete() |
| 前置条件 | 用户点击删除按钮 |
| 后置条件 | 获得选择删除的BusinessDriver的id，调用BusinessDriverblservice的delete方法，然后clear清空 |
| BusinessDriverPanel.getBusinessDriver | 语法 | private BusinessDriver getBusinessDriver() |
| 前置条件 | add或update方法调用时 |
| 后置条件 | 根据文本框以及其他按钮封装BusinessDriver对象 |

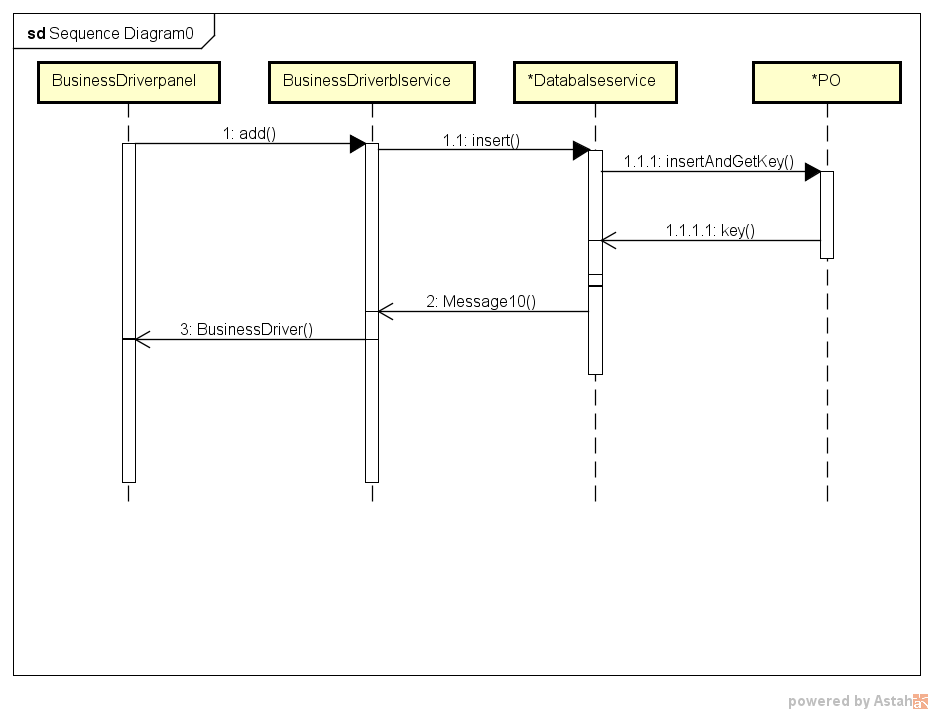
|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessDriverblservice | BusinessDriver业务逻辑接口 |
|  |  |

**BusinessDriverblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| BusinessDriverblserviceImpl.getAll | 语法 | public Vector<BusinessDriver> getAll() |
| 前置条件 | 需要查询所有的BusinessDriver时 |
| 后置条件 | 调用BusinessDriverDataService的getList方法 |
| BusinessDriverblserviceImpl.getById | 语法 | public BusinessDriver getById(int id) |
| 前置条件 | 需要根据id查询BusinessDriver时 |
| 后置条件 | 调用BusinessDriverDataService的getById方法 |
| BusinessDriverblserviceImpl.add | 语法 | public BusinessDriver add(BusinessDriver BusinessDriver) |
| 前置条件 | 需要新增一个BusinessDriver时 |
| 后置条件 | 设置BusinessDriver对应的外键的属性的值，调用BusinessDriverDataService的insert方法，获得新增的BusinessDriver的主键，根据该主键id查询BusinessDriver并返回这个新增的BusinessDriver |
| BusinessDriverblserviceImpl.update | 语法 | public BusinessDriver update(BusinessDriver BusinessDriver) |
| 前置条件 | 需要更新BusinessDriver时 |
| 后置条件 | 调用BusinessDriverDataService的update方法，获得返回的BusinessDriver的主键，根据该主键id查询BusinessDriver并返回这个BusinessDriver |
| BusinessDriverblserviceImpl.delete | 语法 | public boolean delete(int id) |
| 前置条件 | 需要删除一个BusinessDriver时 |
| 后置条件 | 调用BusinessDriverDataservice的delete方法 |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessDriverDataservice.getList() | 获得所有的BusinessDriver持久化对象 |
| BusinessDriverDataservice.getById() | 根据id获得BusinessDriver持久化对象 |
| BusinessDriverDataservice.insert() | 增加一个BusinessDriver持久化对象 |
| BusinessDriverDataservice.update() | 更新一个BusinessDriver持久化对象 |
| BusinessDriverDataservice.delete() | 删除一个BusinessDriver持久化对象 |
|  |  |

### 15.2.3司机管理动态模型



以新增BusinessDriver为例，当用户保存时BusinessDriver主键为空，则为新增操作，调用add方法，调用BusinessDriverblservice的add方法，调用各个Dataservice的insert方法，通过数据库insert每个PO对象，返回其主键id。

### 15.2.4司机管理模块设计原理

利用分层的结构，和委托式的控制风格，将其业务逻辑独立出来，通过不同的数据管理对象管理数据，实现了高内聚低耦合

### 15.2.5度量计算部分

BusinessCollectionpanel 这个类调用的只有BusinessCollectionblservice的六个方法，和一个BusinessCollectionblservice对象引用

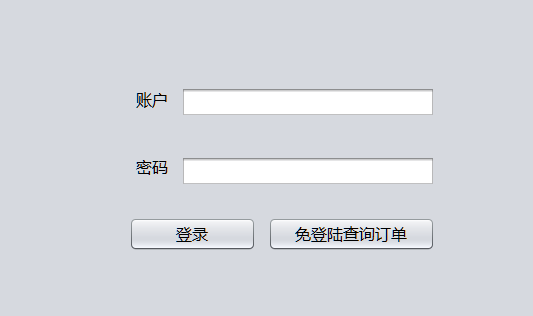
CBD=6、DAC=1。继承一个commonpanel 无子类，NOC=0，DIT=1

BusinessCollectionblService有userDataservice,postDataservice,goodsDataservice三个对象引用和这个对象的四个方法。

CBO=15,DAC=3。collectionblserviceImpl实现该接口，NOC=1，DIT=0.

# 其他原型界面设计

## 登录界面



## 欢迎界面

不同职业有不同的欢迎界面

说明：目前使用的nimbus的lookandfeel，以后会改用ps做界面