/\*

说明：这次作业需要

人机交互设计和各模块的初步详细设计   
  
提交制品：   
人机交互设计包括每个界面的原型（可以用代码写、工具拖拽或者手画）。补充到设计文档中去。   
初步的详细设计模型（设计类图、顺序图等。可选择重要的先描述，其它以后补充。）   
详细设计文档（按照书后模板写）。

根据界面我们分为以下部分：

寄件人 订单查询界面(OrderQueryPanel)1

快递员的 寄件单界面(postPanel)1，收件单界面(RecipientPanel)1

营业厅的 车辆管理界面(BusinessVehiclePanel)1，司机管理界面(BusinessDriverPanel)1, 收款单界面(BusinessColletionPanel)，装车管理界面(BItransportPanel), 派件单界面(BusinessDelieryPanel)

中转中心的 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,IBtransportPanel)，

财务中心的 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)

总经理的 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面(MakeConstantPanel)

查询经营情况界面（同上），

管理员的 用户账户权限管理的界面(AdminPanel)

blservice对应多个Dataservice，所有代码里没有其他blservice和dataservice，命名按之前的po对象命名

我已经写好了寄件单界面（代码）和他的文档描述（如下），和书上的controller不同的是，控制部分我写到了panel里，因为这个分层结构而不是mvc结构

其他的文档需要都需要这4个部分，

其实大部分都是对实体类的增删改查，大体相似

我会把界面都画出来，我写的文档仅供参考，欢迎大家指正

分工上，首先我会把界面写出来，

设计文档

我：寄件人和快递员的

王亚杰：营业厅，管理员

夏志伟：财务中心，和中转中心

熊定波： 总经理

我没有偷懒。。。界面我是真的写代码，我还要汇总的。。。你想，我都给出模板了。。

我这几天每天都写十几个小时，快递都忘记拿了

关于代码分工，你们怎么看。。。现在还不急，我这是出于兴趣写写，6800行了。。。

首先你们得把mysql装好。。。

周四晚饭之前给我，谁晚上给我我xxoo死他，

\*/

# 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

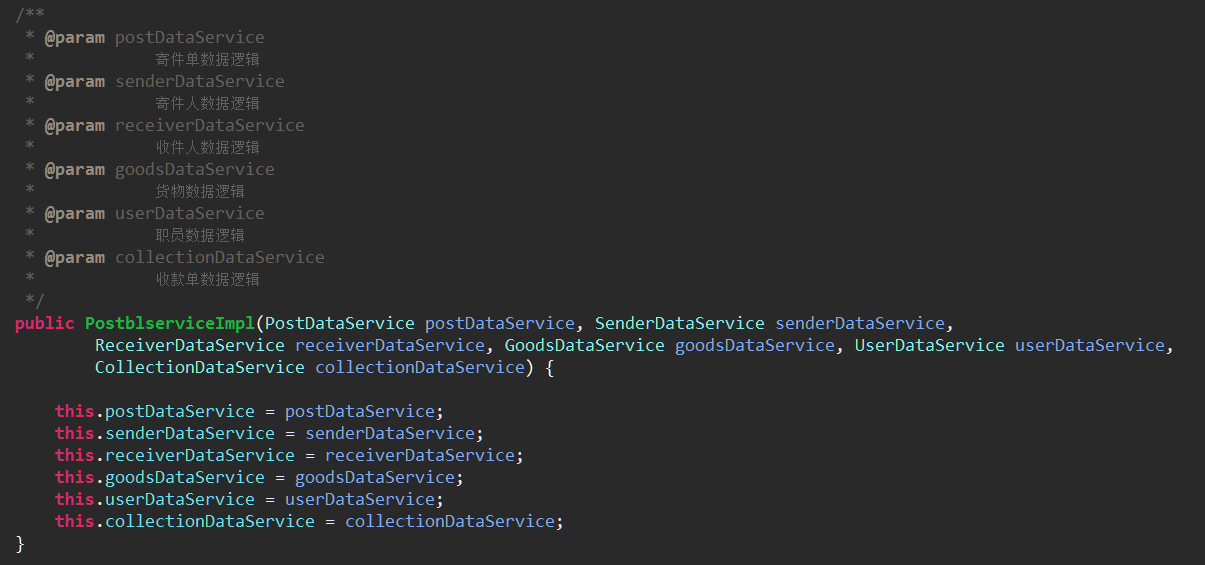
### 1.2.1说明

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

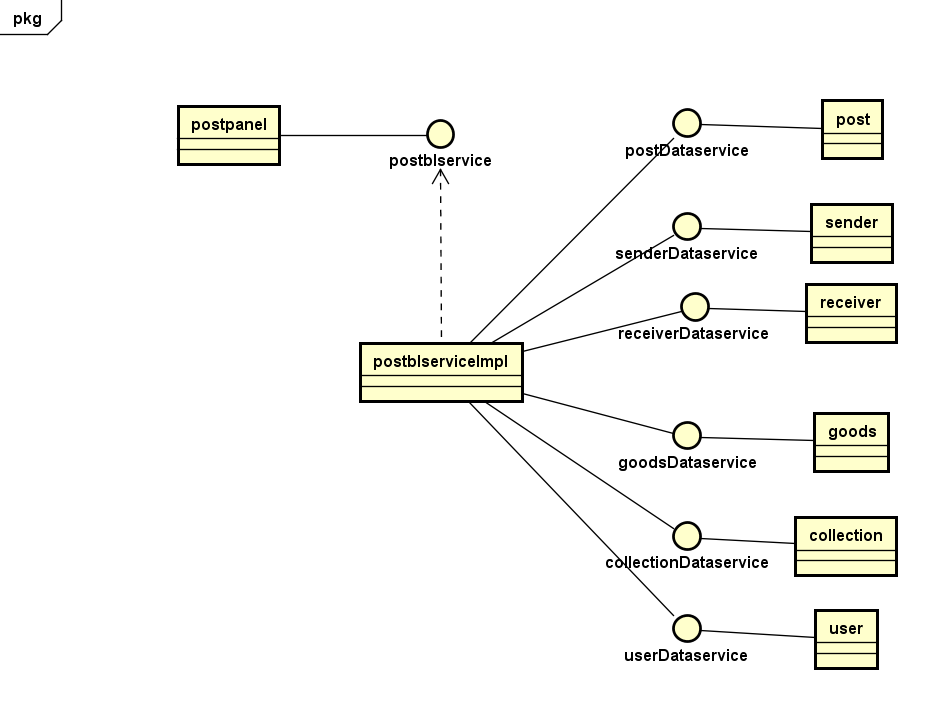
以获得方法为例，从数据库获得一个post对象，需要通过postDataService获得post对象（只包含其本身的属性，而不包含其他的对象），

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 |
|  |  |

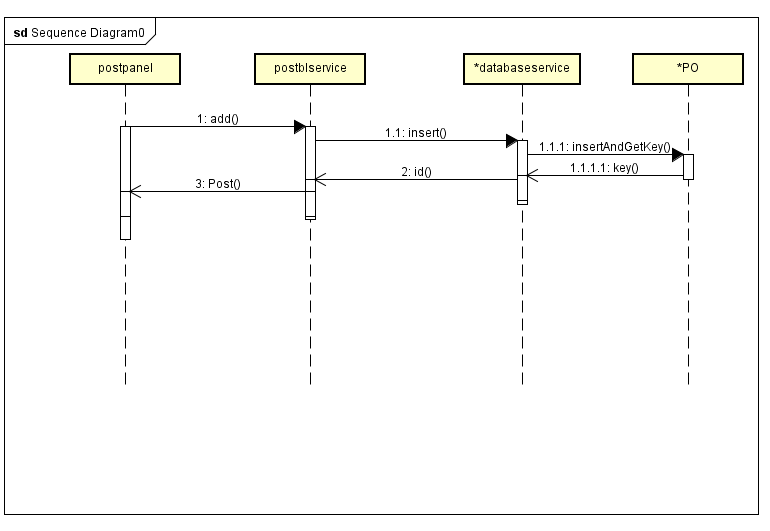
**postblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **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。