# 1收件单模块

## 1.1收件单界面的原型



## 1.2收件单模块设计文档

### 1.2.1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。

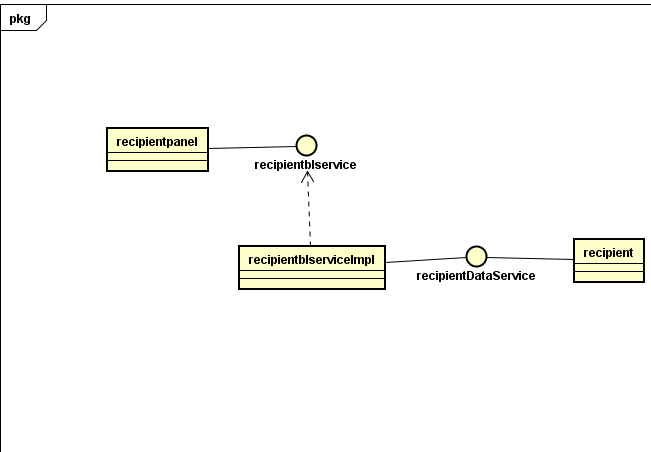
收件单界面RecipientPanel需要在构造方法里传入寄件单业务逻辑接口recipientblservice

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

### 1.2.2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 |
|  |  |

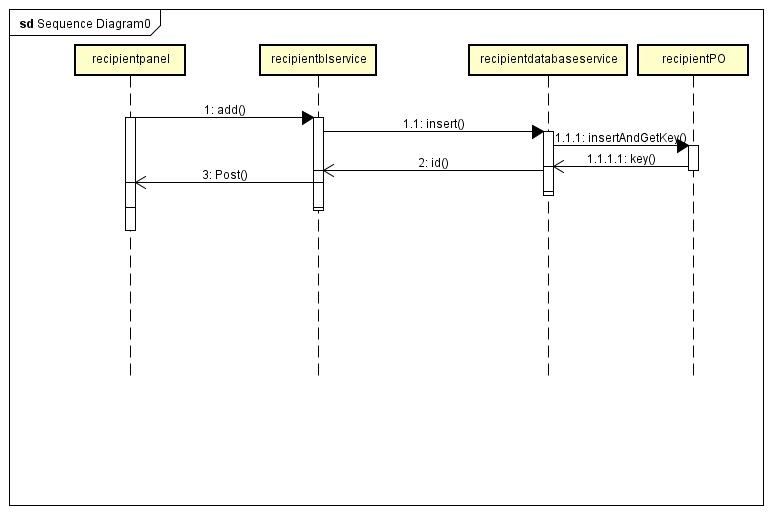
**recipientblserviceImpl的接口规范**

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

### 1.2.3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。



### 1.2.4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。