### 1.2.1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。

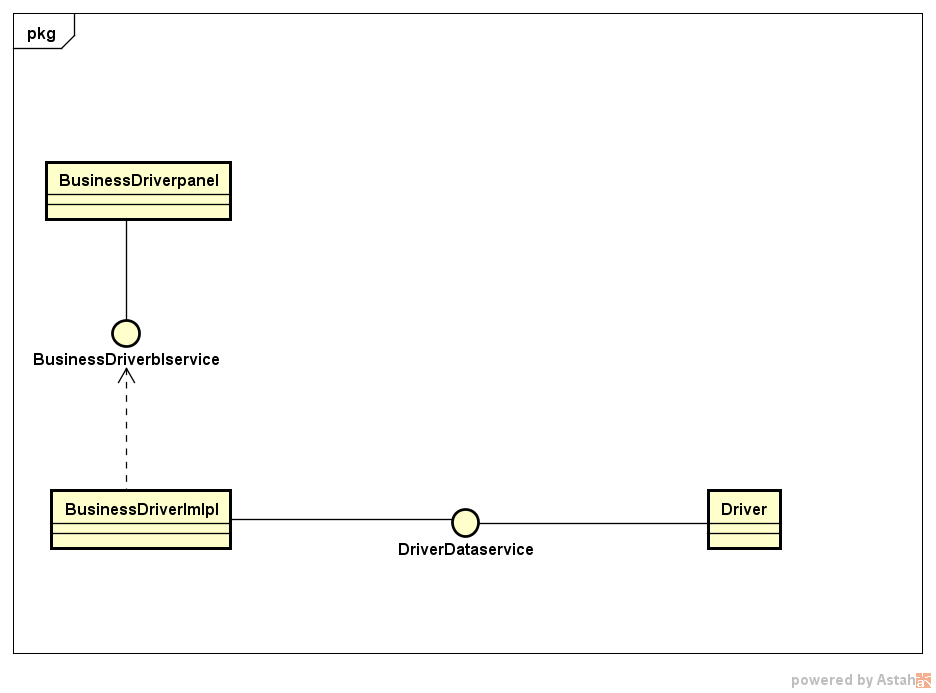
其具体实现类BusinessDriverblserviceimpl需要在构造方法中传入多个数据接口

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

### 1.2.2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 |

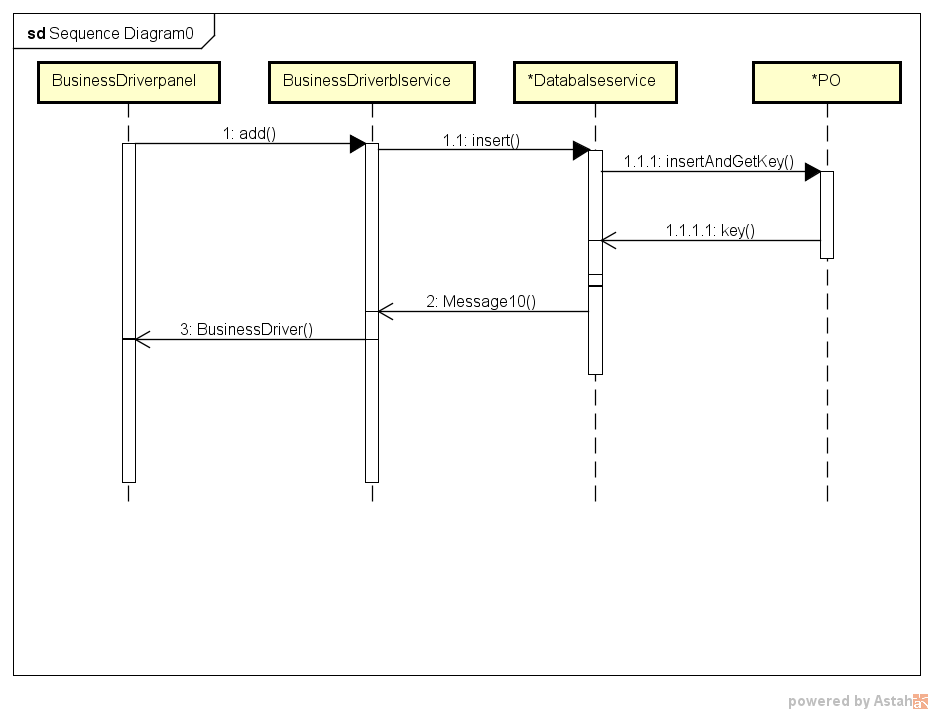
|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| 服务名 | 服务 |
| BusinessDriverblservice | BusinessDriver业务逻辑接口 |
|  |  |

**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 |
|  |  |

### 1.2.3寄件单动态模型



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。

### 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