### 1.2.1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。

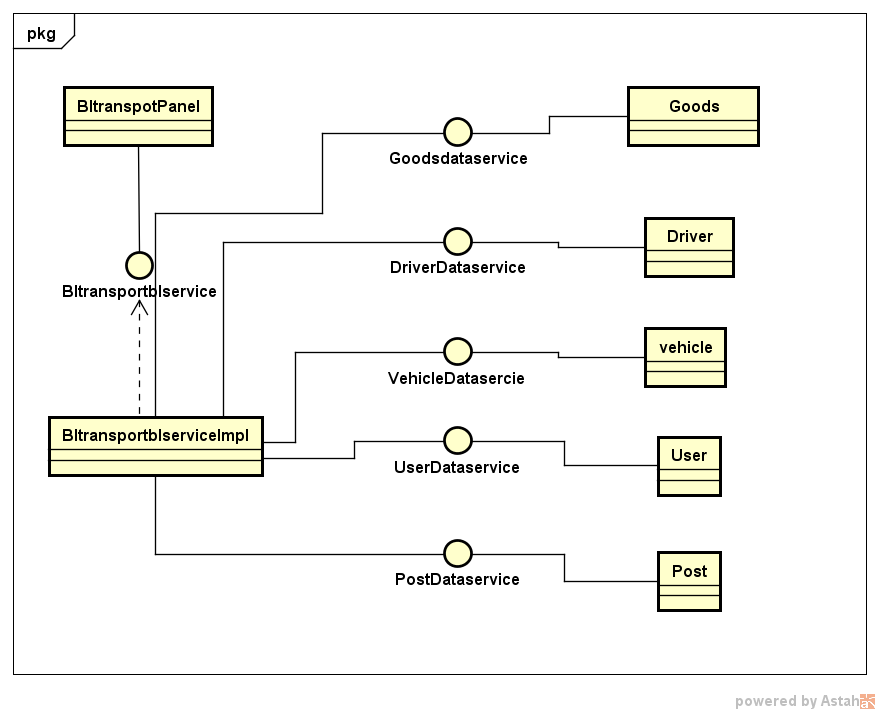
其具体实现类BItranspotblserviceimpl需要在构造方法中传入多个数据接口

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

### 1.2.2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 |
|  |  |

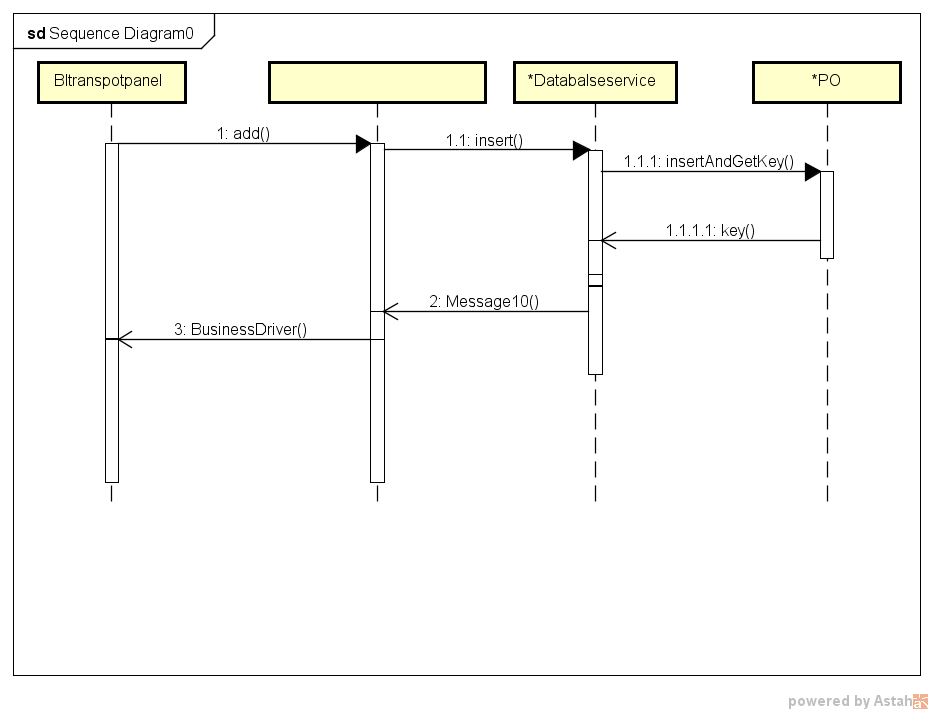
**BItranspotblserviceImpl的接口规范**

|  |  |  |
| --- | --- | --- |
| **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 |

### 1.2.3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。



### 1.2.4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。