### Orderbl模块

1. 模块描述

orderbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

orderbl模块承担的职责和接口见体系设计文档中的orderbl相关的职责和接口

1. 整体结构

依先前体系设计逻辑模型和相关文档的说明，为了增加灵活性降低耦合，增加可复用性，在界面层和业务逻辑层增加接口层，在数据层和业务逻辑层也增加接口层，通过接口进行调用。

将系统分为展示层、业务逻辑层和数据层。每层之间添加接口。

在展示层和业务逻辑层之间，我们添加businesslogicservice.OrderBLservice.OrderBLservice接口。

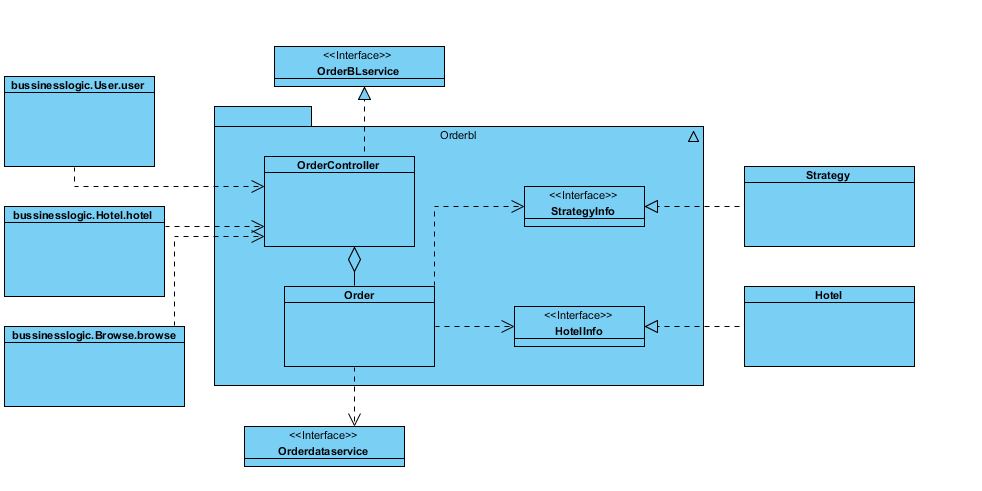
在业务逻辑层和数据层之间，我们添加dataservice.OrderDataservice接口。

为了隔离业务逻辑职责和逻辑控制职责，我们增加了OrderController，这样OrderController会将对销售的业务逻辑处理委托给Order对象。

OrderList是Order的容器类。

OrderVO是作为订单的持久化对象被添加到设计模型中去的。

Orderbl 模块设计如下：



Orderbl模块各个类的职责入下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderController | 负责Orderbl模块的逻辑操作，实现和订单相关界面所需要的服务 |
| Order | 订单的领域模型对象，拥有订单数据的状态、开始时间、结束时间、入住人数、有无儿童等，可以帮助完成订单相关的操作 |

1. 模块内部类的接口规范

OrderController和Order的接口规范如下表所示

OrderController：

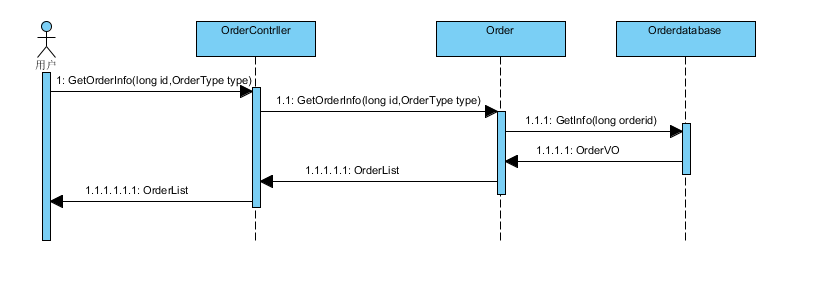
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| OrderController.getuserInfo | 语法 | Public List<OrderVO> getuserInfo(long userid,OrderType type); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的getuserInfo方法 |
| OrderController.getHotelInfo | 语法 | Public List<OrderVO> getHotelInfo(long hotelid,OrderType type); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的getHotelInfo方法 |
| OrderController.getWEBInfo | 语法 | Public List<OrderVO> getWEBInfo(); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的getWEBInfo方法 |
| OrderController.create | 语法 | Public ResultMessage create(OrderVO vo); |
| 前置条件 | User对象发出create请求 |
| 后置条件 | 调用Order领域对象的create方法 |
| OrderController.Abnormal | 语法 | Public void abnormal(OrderVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的Abnormal方法 |
| OrderController.userCancel | 语法 | Public ResultMessage userCancel(OrderVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的userCancel方法 |
| OrderController.execute | 语法 | Public ResultMessage execute(OrderVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的execute方法 |
| OrderController.WEBCancel | 语法 | Public ResultMessage WEBCancel(OrderVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的WEBCancel方法 |
| OrderController. isUsed | 语法 | Public boolean isUsed (StrategyVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的isUsed方法 |
| OrderController.getTotal | 语法 | Public double getTotal(OrderVO vo); |
| 前置条件 | 已创建一个Order领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Order领域对象的getTotal方法 |
| 需要的服务(需接口) | | |
| Order.getuserInfo(long userid,OrderType type) | 用户获取一个对应类型的订单信息 | |
| Order.getHotelInfo(long hotelid,OrderType type) | 酒店工作人员获取一个对应类型的订单信息 | |
| Order.getWEBInfo() | 网站营销人员获取一个对应类型的订单信息 | |
| Order.create(OrderVO vo) | 创建一个订单对象 | |
| Order.Abnormal(OrderVO vo) | 改变订单状态为异常 | |
| Order.userCancel(OrderVO vo) | 用户删除一个订单 | |
| Order.execute(OrderVO vo) | 改变订单状态为执行 | |
| Order.WEBCancel(OrderVO vo) | 网站营销人员删除一个订单 | |
| Order.getTotal(OrderVO vo) | 获得订单总价 | |

Order：

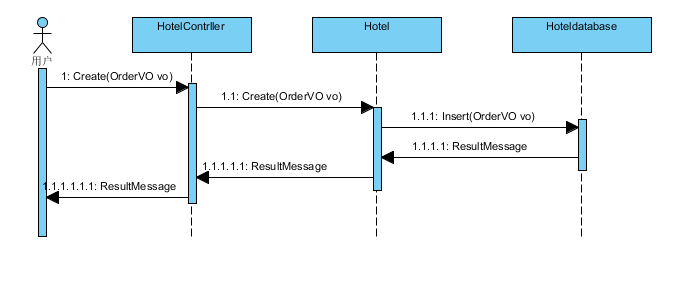
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Order.getuserInfo | 语法 | Public List<OrderVO> getuserInfo(long userid,OrderType type); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据对应的type 查找对应类型的对应用户订单信息 |
| Order.getHotelInfo | 语法 | Public List<OrderVO> getHotelInfo(long hotelid,OrderType type); |
| 前置条件 | 酒店管理人员已登录 |
| 后置条件 | 根据对应的type 查找对应类型的对应用户订单信息 |
| Order.getWEBInfo | 语法 | Public List<OrderVO> getWEBInfo(); |
| 前置条件 | 网站营销人员已登录 |
| 后置条件 | 根据对应的TYPE 查找对应类型的当天当时间异常订单信息 |
| Order.create | 语法 | Public ResultMessage create(OrderVO vo); |
| 前置条件 | 客户已登录 |
| 后置条件 | 创建订单并存储 |
| Order.Abnormal | 语法 | Public void abnormal(OrderVO vo); |
| 前置条件 | 订单由未执行变为异常 |
| 后置条件 | 改变订单状态并存储 |
| Order.userCancel | 语法 | Public ResultMessage userCancel(OrderVO vo); |
| 前置条件 | 用户已登录 |
| 后置条件 | 所orderid所对应订单为未执行订单，撤销该订单 |
| Order.execute | 语法 | Public ResultMessage execute(OrderVO vo); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 根据对应订单，判断订单状态，执行或补登记执行订单，并作出信用值的相应改变 |
| Order.WEBCancel | 语法 | Public ResultMessage WEBCancel(OrderVO vo); |
| 前置条件 | 网站营销人员已登录且线下已判断订单可撤销 |
| 后置条件 | 根据对应的订单号，撤销订单，并返回状态 |
| Order.isUsed | 语法 | Public boolean isUsed (StrategyVO vo); |
| 前置条件 | 用户要求计算订单价格 |
| 后置条件 | 判断是否符合促销策略条件，并返回 |
| Order.getTotal | 语法 | Public double getTotal(OrderVO vo); |
| 前置条件 | 无 |
| 后置条件 | 计算订单总价，并返回 |
| 需要的服务(需接口) | | |
| DatabaseFactory.getOrderDatabase | 得到Order数据库服务的引用 | |
| OrderDataService.insert(OrderPO po) | 在Order数据库中插入po | |
| OrderDataService.changeStatus(OrderPO po); | 改变订单的状态 | |
| OrderDataService.getInfo(long orderid); | 获取订单的信息 | |
| Hotel.getRooms(long hotelid) | 得到该hotel的房间信息 | |
| Hotel.setRoomInfo(RoomVO vo) | 锁定该酒店的相应的房间 | |
| Credit.add/decrese(long userid,int value); | 增加/减少该user的信用值信息 | |
| Member.getInfo(userid) | 得到对应的会员信息 | |
| Strategy.getInfo(orderid) | 得到对应订单的策略信息 | |

1. 业务逻辑层的动态模型

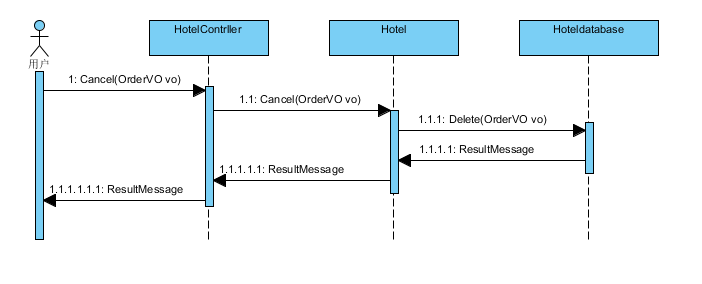
下图显示当客户或酒店工作人员或网站营销人员请求查看订单信息时，Orderbl模块各个类之间的协作时序图：



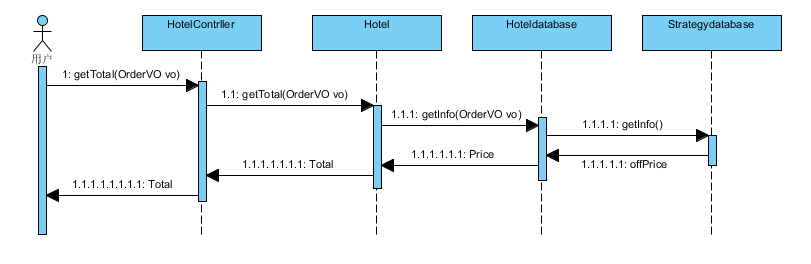
下图显示当客户要求创建订单时，Orderbl模块各个类之间的协作时序图：



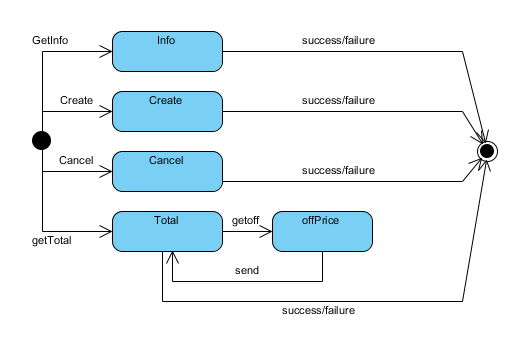
下图显示当客户或网站营销人员要求撤销订单时，Orderbl模块各个类之间的协作时序图：



下图显示当客户要求计算订单总价时，Orderbl模块各个类之间的协作时序图：



下图描述了Hotel对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作。随着Getinfo、SetInfo、Cancel或getTotal方法被UI调用，Order处理请求并结束。



1. 业务逻辑层的设计原理

采用适配器模式风格，由controller统一调度，将任务委托给Order，Order处理后结束。

### Hotelbl模块

（1）模块描述

hotelbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

hotelbl模块承担的职责和接口见体系设计文档中的hotelbl相关的职责和接口

（2）整体结构

依先前体系设计逻辑模型和相关文档的说明，为了增加灵活性降低耦合，增加可复用性，在界面层和业务逻辑层增加接口层，在数据层和业务逻辑层也增加接口层，通过接口进行调用。

将系统分为展示层、业务逻辑层和数据层。每层之间添加接口。

在展示层和业务逻辑层之间，我们添加businesslogicservice.HotelBLservice.HotelBLservice接口。

在业务逻辑层和数据层之间，我们添加dataservice.HotelDataservice接口。

为了隔离业务逻辑职责和逻辑控制职责，我们增加了HotelController，这样HotelController会将对销售的业务逻辑处理委托给Hotel对象。

Hotel对象在执行一些操作的情况下会委托给Room对象。

HotelVO是作为订单的持久化对象被添加到设计模型中去的。

Hotelbl模块各个类的职责入下表所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| HotelController | 负责Hotelbl模块的逻辑操作，实现和酒店相关界面所需要的服务 |
| Hotel | 酒店的领域模型对象，拥有酒店数据的酒店名称、商圈、地址、简介、星级、设施服务等，可以帮助完成酒店相关的操作 |
| Room | 房间的领域模型对象，包括房间号、房间类型、价格等，可以帮助完成房间相关的操作 |

（3）模块内部类的接口规范

HotelController和Hotel的接口规范如下表所示

HotelController：

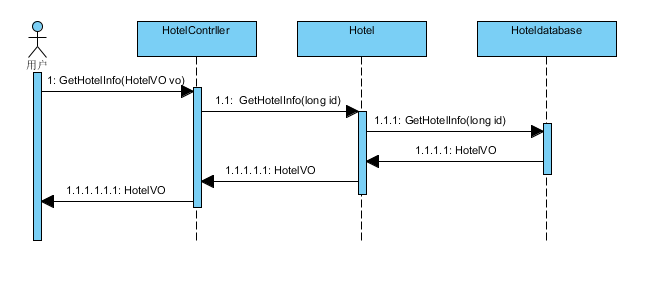
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| HotelController.GetHotelInfo | 语法 | Public HotelVO GetHotelInfo(long id); |
| 前置条件 | 已创建一个Hotel领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Hotel领域对象的GetHotelInfo方法 |
| HotelController.SetHotelInfo | 语法 | Public ResultMessage SetHotelInfo(HotelVO vo); |
| 前置条件 | 已创建一个Hotel领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Hotel领域对象的SetHotelInfo方法 |
| HotelController.SetRoomInfo | 语法 | Public ResultMessage SetRoomInfo(long id, RoomVO vo); |
| 前置条件 | 已创建一个Room领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Hotel领域对象的SetRoomInfo方法 |
| HotelController.AddHotel | 语法 | Public ResultMessage AddHotel(HotelVO vo); |
| 前置条件 | WebManager对象发出add请求 |
| 后置条件 | 调用Hotel领域对象的AddHotel方法 |
| HotelController.AddRoom | 语法 | Public ResultMessage AddRoom(RoomVO vo); |
| 前置条件 | Hotelworker对象发出add请求 |
| 后置条件 | 调用Hotel领域对象的AddRoom方法 |
| 需要的服务(需接口) | | |
| Hotel.GetHotelInfo(long id) | 获取一个酒店对象的信息 | |
| Hotel.SetHotelInfo(HotelVO vo) | 修改一个酒店对象的信息 | |
| Hotel.SetRoomInfo(long id, RoomVO vo) | 修改一个房间对象的信息 | |
| Hotel.AddHotel(HotelVO vo) | 创建一个酒店对象 | |
| Hotel.AddRoom(RoomVO vo) | 创建一个房间对象 | |

Hotel：

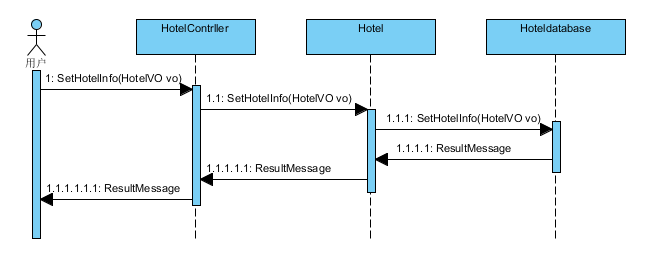
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Hotel.GetHotelInfo | 语法 | Public HotelVO GetHotelInfo(long id); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Hotel.SetHotelInfo | 语法 | Public ResultMessage SetHotelInfo(HotelVO vo); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 更新酒店信息 |
| Hotel.SetRoomInfo | 语法 | Public ResultMessage SetRoomInfo(long id, RoomVO vo); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 更新房间信息 |
| Hotel.AddHotel | 语法 | Public ResultMessage AddHotel(HotelVO vo); |
| 前置条件 | 网站管理人员已登录 |
| 后置条件 | 生成新的酒店信息 |
| Hotel.AddRoom | 语法 | Public ResultMessage AddRoom(RoomVO vo); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 生成新的房间信息 |
| 需要的服务(需接口) | | |
| DatabaseFactory.getHotelDatabase | 得到Hotel数据库服务的引用 | |
| HotelDataService.insertHotel(HotelPO po) | 在数据库中插入vo对象 | |
| HotelDataService.getHotelInfo(HotelID id) | 得到Hotel信息 | |
| HotelDataService.setHotelInfo(HotelPO po) | 更改Hotel信息 | |
| HotelDataService.insertRoom(RoomPO po) | 在数据库中插入vo对象 | |
| HotelDataService.getRoomInfo(HotelID id) | 得到Room信息 | |
| HotelDataService.setRoomInfo(RoomPO po) | 更改Room信息 | |

（4）业务逻辑层的动态模型

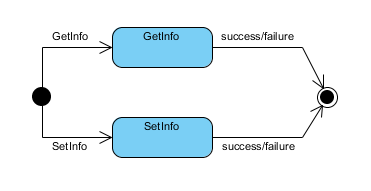
下图显示当酒店工作人员请求查看酒店信息时，Hotelbl模块各个类之间的协作时序图：



下图显示当酒店工作人员请求修改酒店信息时，Hotelbl模块各个类之间的协作时序图：



下图描述了Hotel对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作。随着Getinfo或SetInfo方法被UI调用，Hotel处理请求并结束。



（5）业务逻辑层的设计原理

采用适配器模式风格，由controller统一调度，将任务委托给Hotel，Hotel处理后结束。