

F.A.F酒店预订系统

详细设计文档



2017-1-1

NJU F.A.F 万年杰 田原 曾虎双 陆茹茹

V1.4

**[更新历史 4](#_Toc22678)**

**[1. 引言 5](#_Toc19089)**

**[1.1 编制目的 5](#_Toc28651)**

**[1.2 词汇表 5](#_Toc490)**

**[1.3 参考资料 5](#_Toc10748)**

**[2. 产品概述 5](#_Toc11018)**

**[3. 体系结构设计概述 5](#_Toc16830)**

**[4. 结构视角 5](#_Toc24311)**

**[4.1 业务逻辑层的分解 5](#_Toc5762)**

**[4.1.1 userbl模块 5](#_Toc26796)**

[4.1.1.1 LoginAndSignUp 6](#_Toc21627)

[4.1.1.2 ModifyClientInfo 9](#_Toc17430)

[4.1.1.3 SignVip 10](#_Toc3681)

[4.1.1.4 QueryClientCreditRecord 12](#_Toc3486)

[4.1.1.5 AddCreditValue 14](#_Toc14655)

[4.1.1.6 ManageUserInfo 16](#_Toc12098)

**[4.1.2 orderbl模块 19](#_Toc17332)**

[4.1.2.1 BrowseHotelOrder 20](#_Toc16217)

[4.1.2.2 BrowseUserOrder 22](#_Toc3116)

[4.1.2.3 CreateNewOrder 25](#_Toc26213)

[4.1.2.4 GetOrderDone 29](#_Toc21971)

[4.1.2.5 CheckAbnormalOrder 33](#_Toc3246)

[4.1.2.6 WithdrawOrder 36](#_Toc6750)

**[4.1.3 hotelbl模块 38](#_Toc21880)**

[4.1.3.1 QueryHotel 40](#_Toc18140)

[4.1.3.2 SearchHotel 42](#_Toc23818)

[4.1.3.3 CheckOrderedHotel 44](#_Toc9248)

[4.1.3.4 CommentOnHotel 47](#_Toc6123)

[4.1.3.5 ImportNewRoom 49](#_Toc28562)

[4.1.3.6 MaintainHotelBasicInfo 52](#_Toc25849)

[4.1.3.7 ManageHotelInfo 54](#_Toc3936)

**[4.1.4 roombl模块 56](#_Toc12615)**

[4.1.4.1 BrowseSpareRoom 57](#_Toc25963)

[4.1.4.2 UpdateCheckIn 60](#_Toc17385)

[4.1.4.3 UpdateCheckOut 65](#_Toc7777)

**[4.1.5 strategybl模块 70](#_Toc14610)**

[4.1.5.1 UpdateStrategy 71](#_Toc1764)

**[4.2 业务逻辑层的设计原理 76](#_Toc12081)**

**[5. 依赖视角 76](#_Toc32432)**

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **更新人员** | **日期** | **变更原因** | **版本号** |
| **陆茹茹** | **2016.10.29** | **草稿** | **V0.0** |
| **陆茹茹** | **2016.10.29** | **正式版** | **V1.0** |
| **陆茹茹** | **2017.01.01** | **修改user部分** | **V1.1** |
| **万年杰** | **2017.01.01** | **修改order部分** | **V1.2** |
| **田原** | **2017.01.01** | **修改hotel部分** | **V1.3** |
| **曾虎双** | **2017.01.01** | **修改room、strategy部分** | **V1.4** |

1. **引言**
   1. **编制目的**

本报告详细完成对酒店预订系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

* 1. **词汇表**

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
|  |  |  |

* 1. **参考资料**

参考酒店预订系统用例文档和体系结构文档。

1. **产品概述**

参考酒店预订系统用例文档和软件需求规格说明文档中对产品的概括描述。

1. **体系结构设计概述**

参考酒店预订系统体系结构文档中对体系结构设计的概述。

1. **结构视角**
   1. **业务逻辑层的分解**
      1. **userbl模块**

**（1）模块概述**

userbl模块承担的需求参见需求规格说明文档功能需求和相关非功能需求。

userbl模块的职责及接口参见软件体系结构描述文档。

**（2）整体结构**

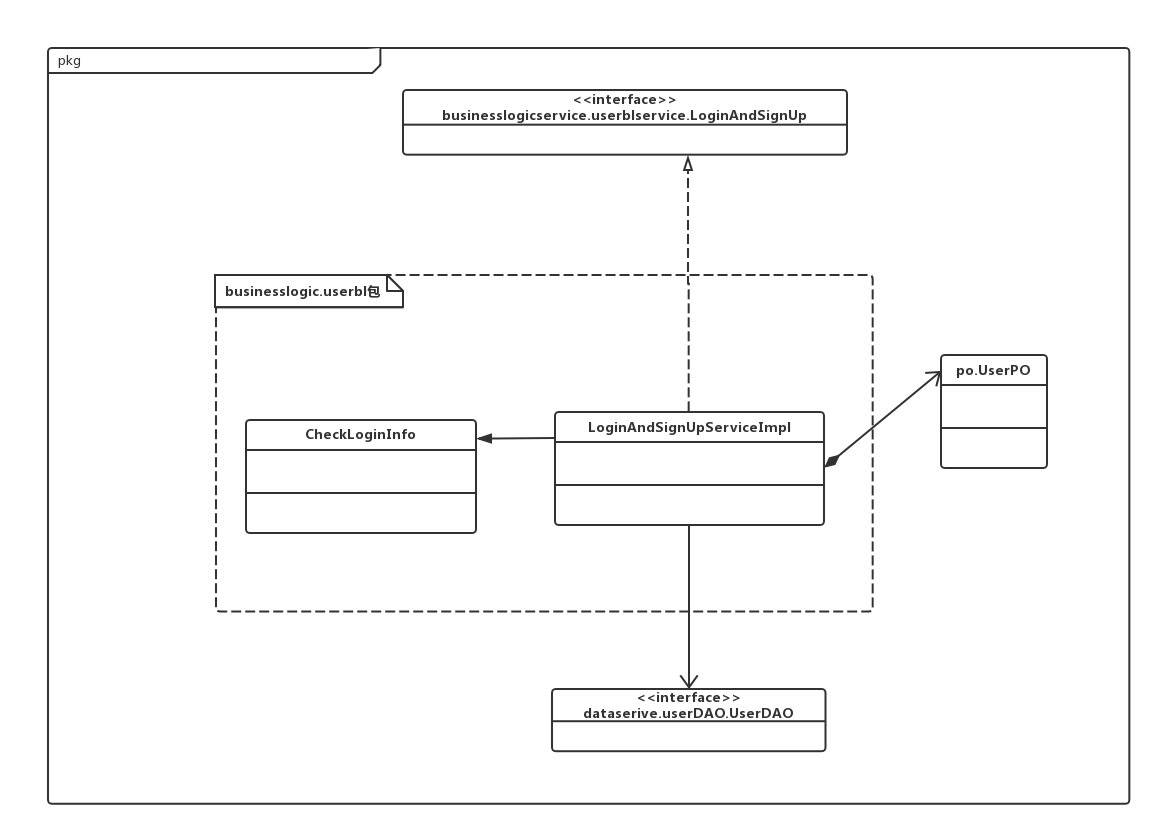
展示层和业务逻辑层之间，我们添加bussinesslogicservice. userblservice.LoginAndSignUpService接口、bussinesslogicservice. userblservice.ModifyClientInfoService接口、bussinesslogicservice. userblservice. SignVipService接口、bussinesslogicservice. Userblservice.QueryClientCreditRecordService接口、bussinesslogicservice. userblservice.AddCreditValueService接口、bussinesslogicservice. userblservice.ManageUserInfoService接口。业务逻辑层和数据层之间添加dataservice.userDAO.UserDAO接口。由于本模块涉及到的逻辑控制职责较为简单，将其与业务逻辑职责合并形成LoginAndSignUpServiceImpl, ModifyClientInfoServiceImpl, SignVipServiceImpl,QueryClientCreditRecordServiceImpl,AddCreditValueServiceImpl,ManageUserInfo类。UserPO, ClientInfoPO, WebMarketStaffInfoPO, HotelStaffInfoPO, VipInfoPO是作为用户的持久化对象被添加到涉及模型中去的。CheckLoginInfo的添加是Login的容器类。ClientCreditInfo, UserInfo, VipInfo都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

userbl模块各个类的职责

|  |  |
| --- | --- |
| 类名 | 职责 |
| loginAndSignUp.LoginAndSignUpServiceImpl | 负责实现对应于登录和注册所需要的服务 |
| loginAndSignUp.CheckLoginInfo | 检查登录信息对象，负责检查登录信息 |
| mdifyClientInfo.ModifyClientInfoServiceInpl | 负责实现对应于客户维护个人信息所需要的服务 |
| signVip.SignVipServiceImpl | 负责实现对应于注册会员所需要的服务 |
| queryClientCreditRecord.QueryClientCreditRecordServiceImpl | 负责实现对应于查看信用记录所需要的服务 |
| addCreditValue.AddCreditValueServiceImpl | 负责实现对应于信用充值所需要的服务 |
| manageUserInfo.ManageUserInfoServiceImpl | 负责实现对应于管理用户所需要的服务 |
| ClientCreditInfoImpl | 给同层的其他模块提供有关客户信用值的服务 |
| UserInfoImpl | 给同层的其他模块提供有关用户信息的服务 |
| VipInfoImpl | 给同层的其他模块提供有关会员信息的服务 |

* + - 1. **LoginAndSignUp**

LoginAndSignUp的设计类图如下图所示。

****

**接口规范：**

LoginAndSignUpServiceImpl和CheckLoginInfo的接口规范如下表所示。

LoginAndSignUpServiceImpl的接口规范

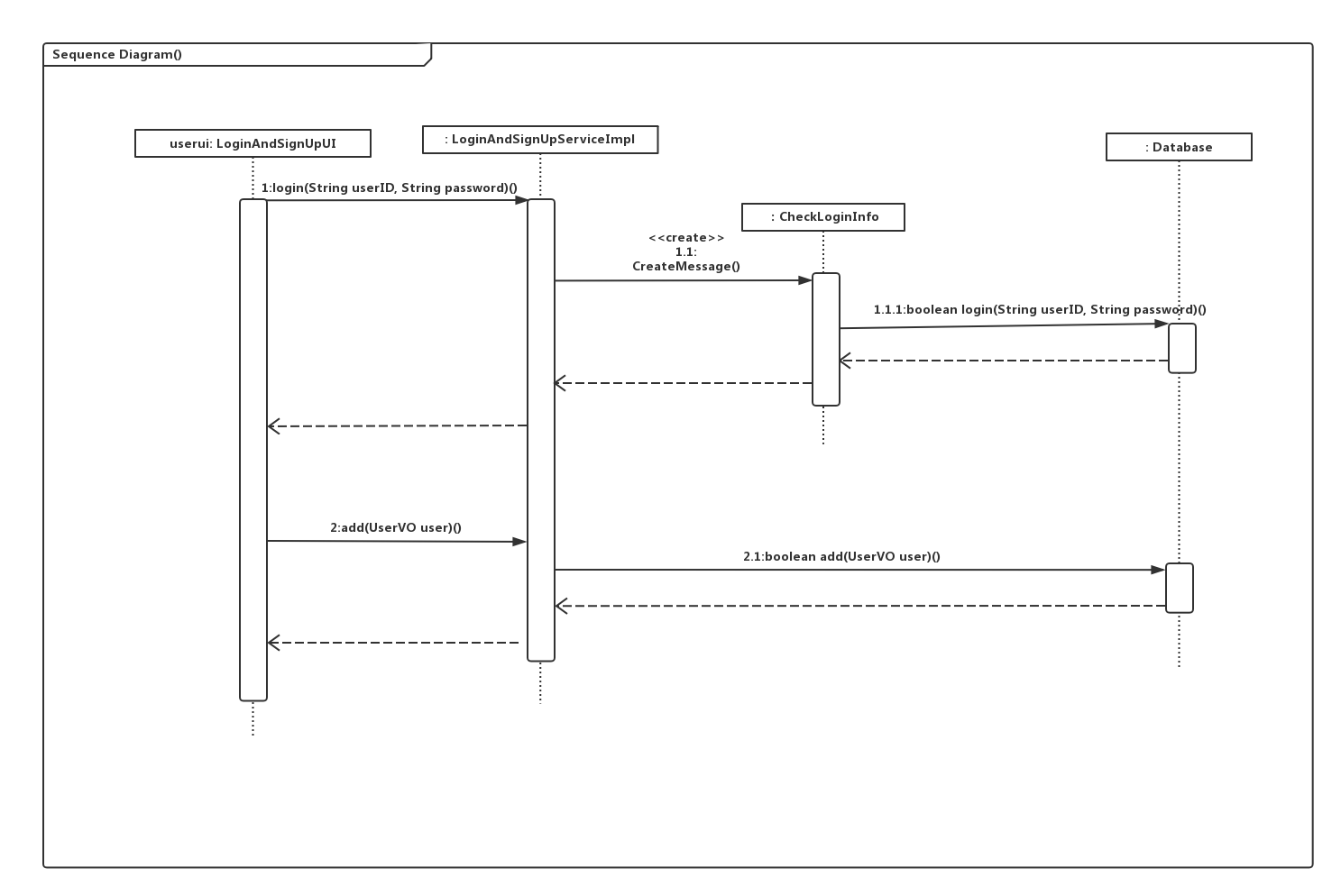
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginAndSignUpServiceImpl.login | 语法 | public boolean login(String ID, String password) |
| 前置条件 | ID和password符合输入规范 |
| 后置条件 | 调用CheckLoginInfo方法 |
| LoginAndSignUpServiceImpl.add | 语法 | public boolean add(UserVO user) |
| 前置条件 | 用户注册账号 |
| 后置条件 | 返回是否注册成功，若成功则返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.insert(UserPO po) | 插入单一持久化对象 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

CheckLoginInfo的接口规范

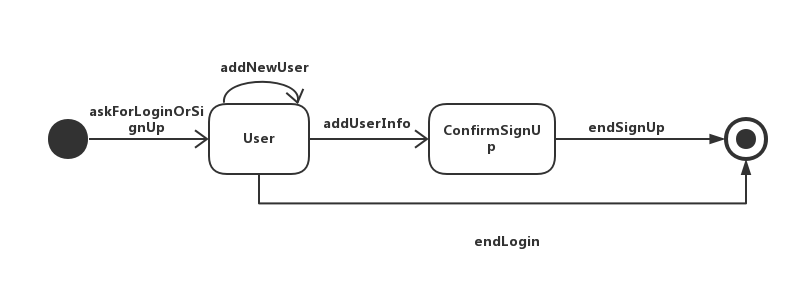
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckLoginInfo.checkUser | 语法 | public UserType checkUser(String userID,String password) |
| 前置条件 | userID和password符合输入规范 |
| 后置条件 | 查找是否存在相应的ID，根据输入的password返回输入的结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(String userID) | 得到userID对应的用户信息 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

**业务逻辑层的动态模型**

LoginAndSignUp的顺序图如下图所示。

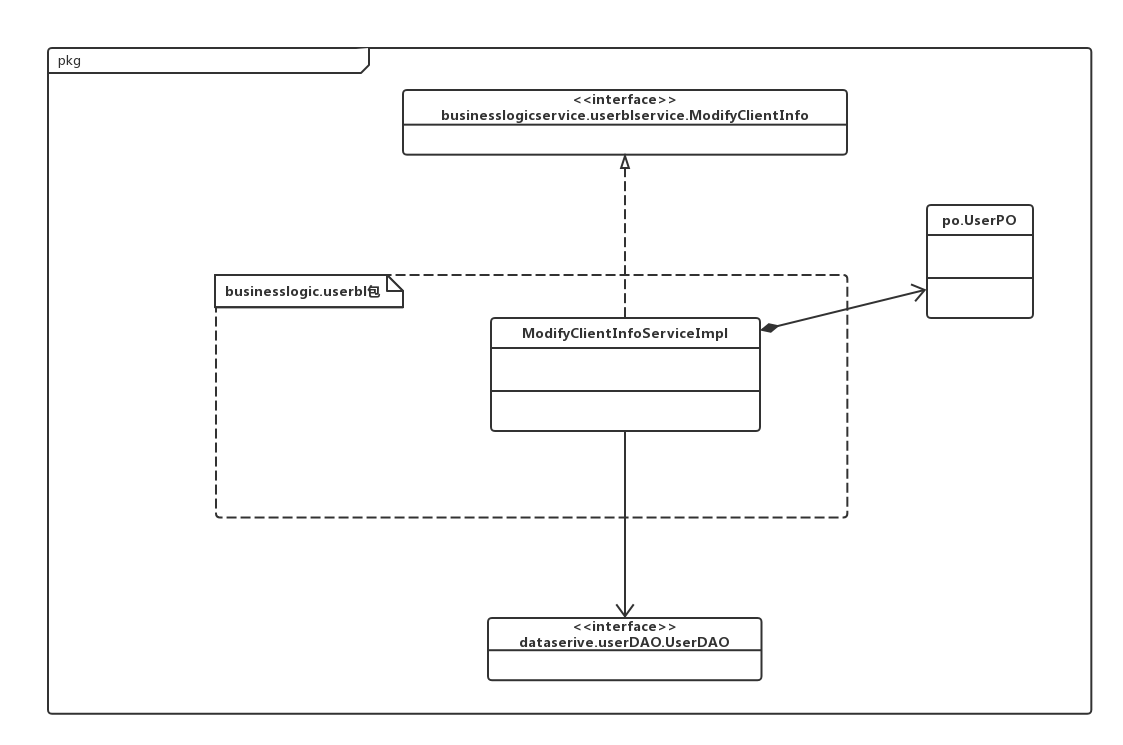
****

LoginAndSignUp的状态图如下图所示。

****

* + - 1. **ModifyClientInfo**

ModifyClientInfo的设计类图如下图所示。

****

**接口规范：**

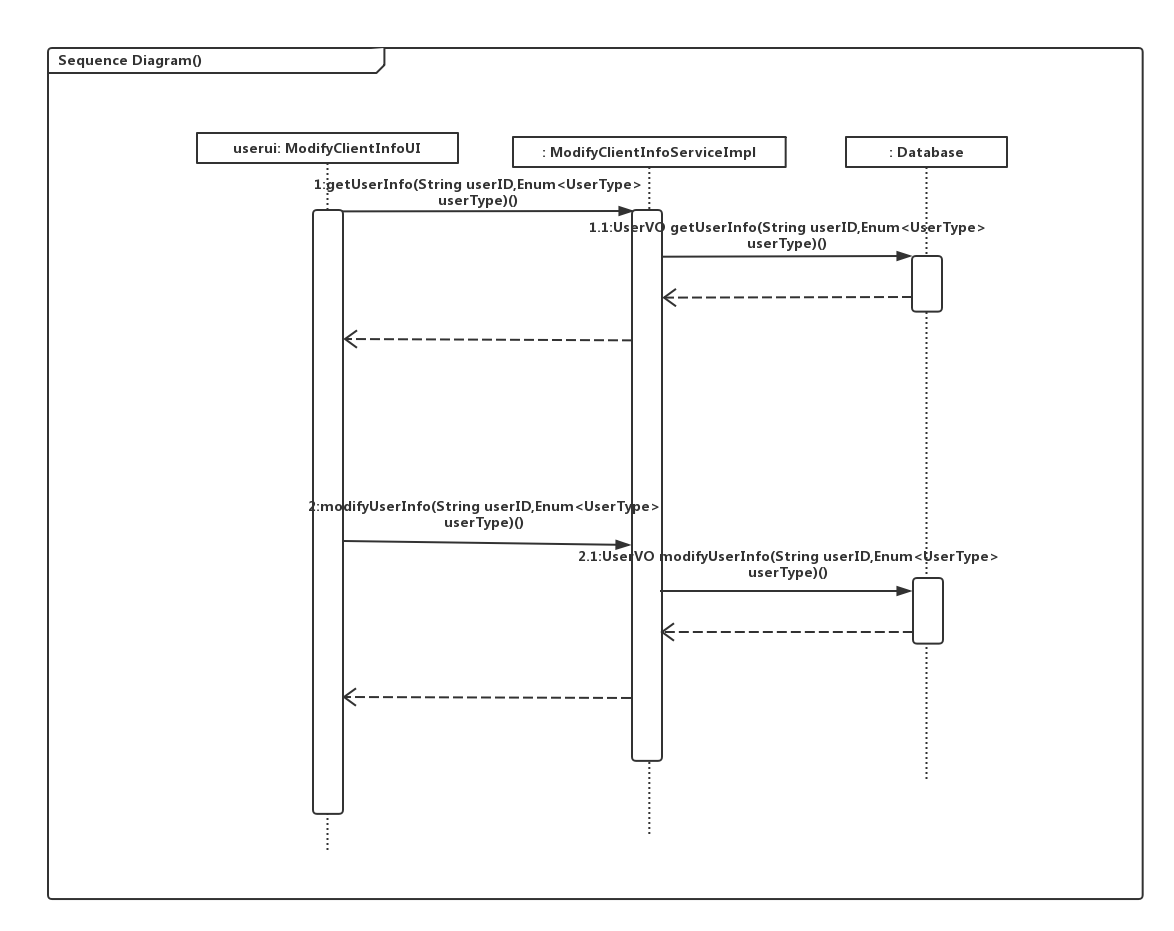
MoodifyClientInfoServiceImpl的接口规范如下表所示。

MoodifyClientInfoServiceImpl的接口规范

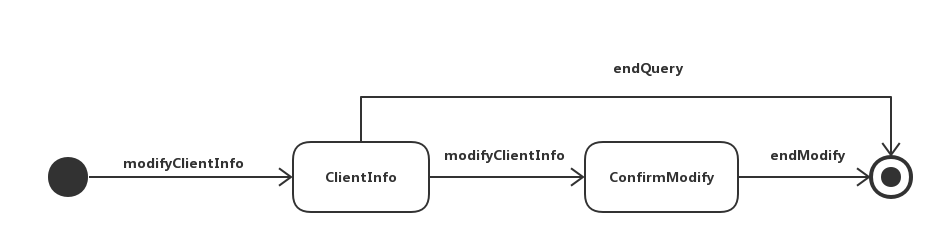
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ModifyClientInfoServiceImpl.modifyClientInfo | 语法 | public boolean modifyClientInfo(UserVO user, String oldUserID) |
| 前置条件 | 客户修改用户信息 |
| 后置条件 | 返回是否修改成功 |
| ModifyClientInfoServiceImpl.getClientInfo | 语法 | public UserVO getClientInfo(String userID) |
| 前置条件 | 客户查看用户信息 |
| 后置条件 | 返回该客户信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(String userID) | 得到userID对应的用户信息 | |
| userDAO.update(UserPO po) | 更新单一持久化对象 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

**业务逻辑层的动态模型**

ModifyClientInfo的顺序图如下图所示。

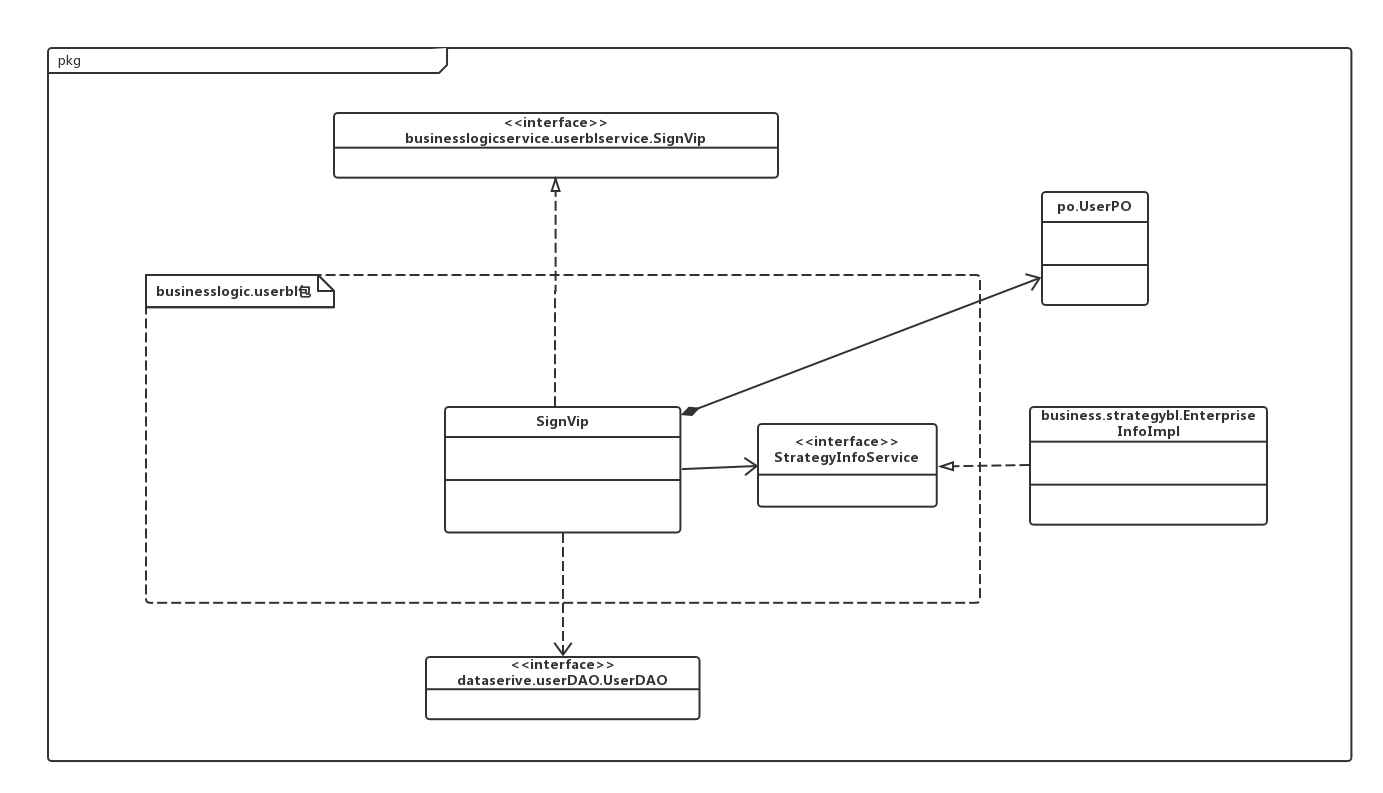
****

ModifyClientInfo的状态图如下图所示。

****

* + - 1. **SignVip**

SignVip的设计类图如下图所示。

****

**接口规范：**

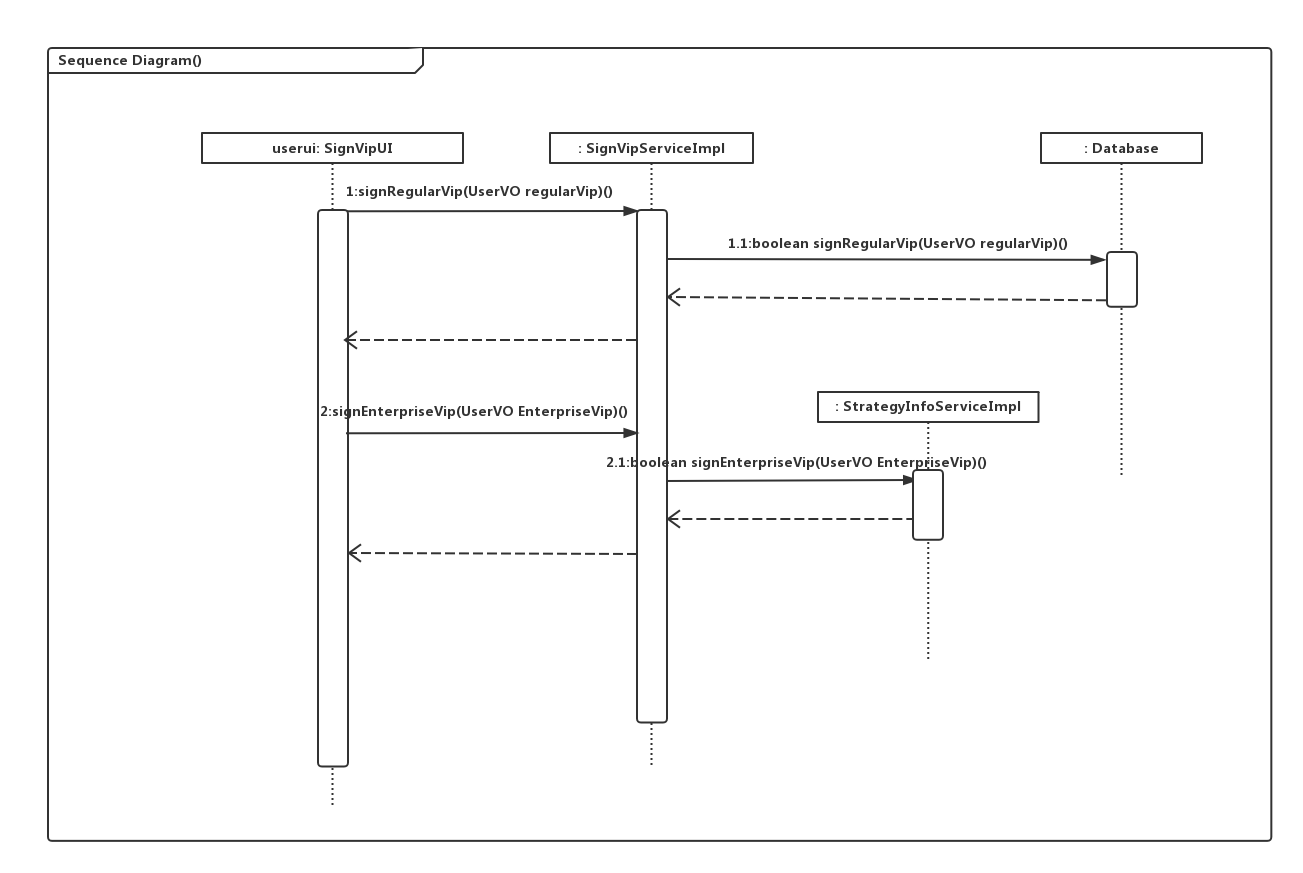
SignVipImpl的接口规范如下表所示。

SignVipServiceImpl的接口规范

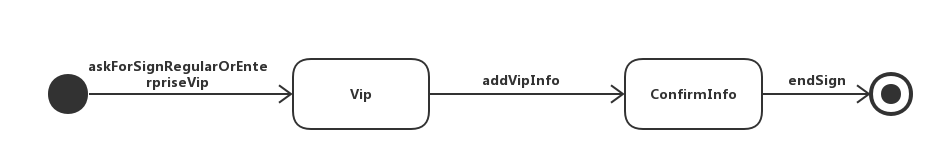
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SignVipServiceImpl.signEnterpriseVip | 语法 | public boolean signEnterpriseVip(VipInfoVO enterpriseVip) |
| 前置条件 | 客户申请企业会员 |
| 后置条件 | 返回是否申请成功 |
| SignVipServiceImpl.signRegularVip | 语法 | public boolean signRegularVip(VipInfoVO regularVip) |
| 前置条件 | 客户申请企业会员 |
| 后置条件 | 返回是否申请成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.update(UserPO po) | 更新单一持久化对象 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |
| StrategyInfo.verifyEnterpriseMember(String enterpriseName, String securityCode) | 验证企业会员注册的信息 | |

**业务逻辑层的动态模型**

SignVip的顺序图如下图所示。

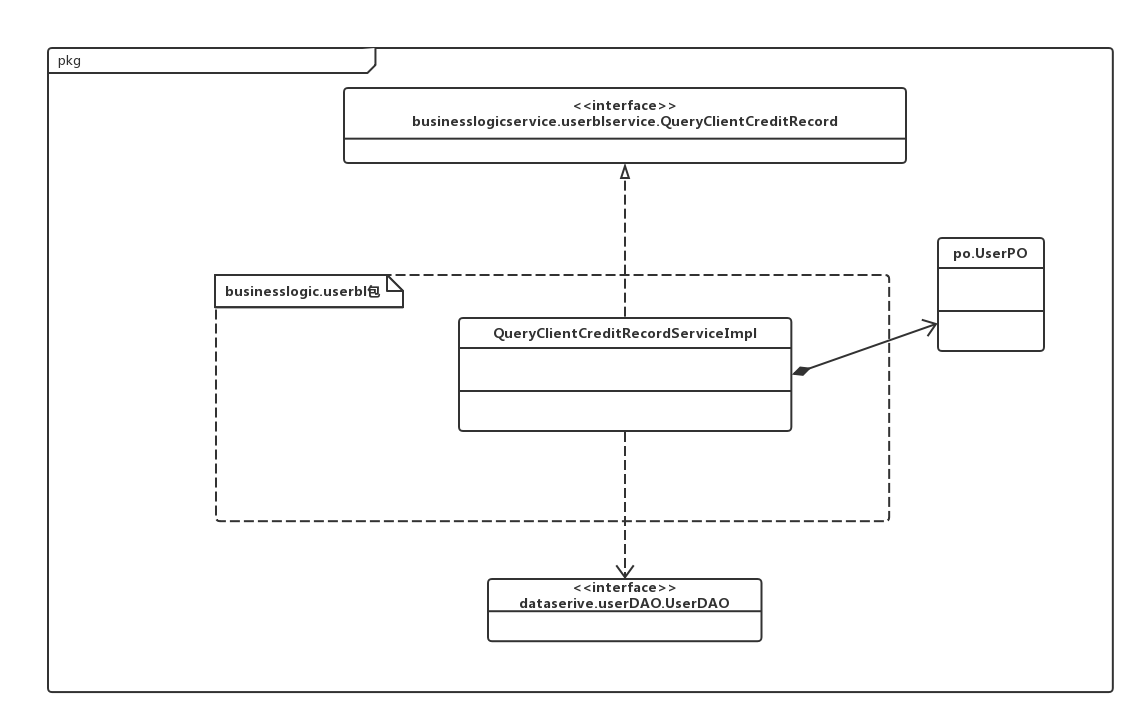
****

SignVip的状态图如下图所示。

****

* + - 1. **QueryClientCreditRecord**

QueryClientCreditRecord的设计类图如下图所示。

****

**接口规范：**

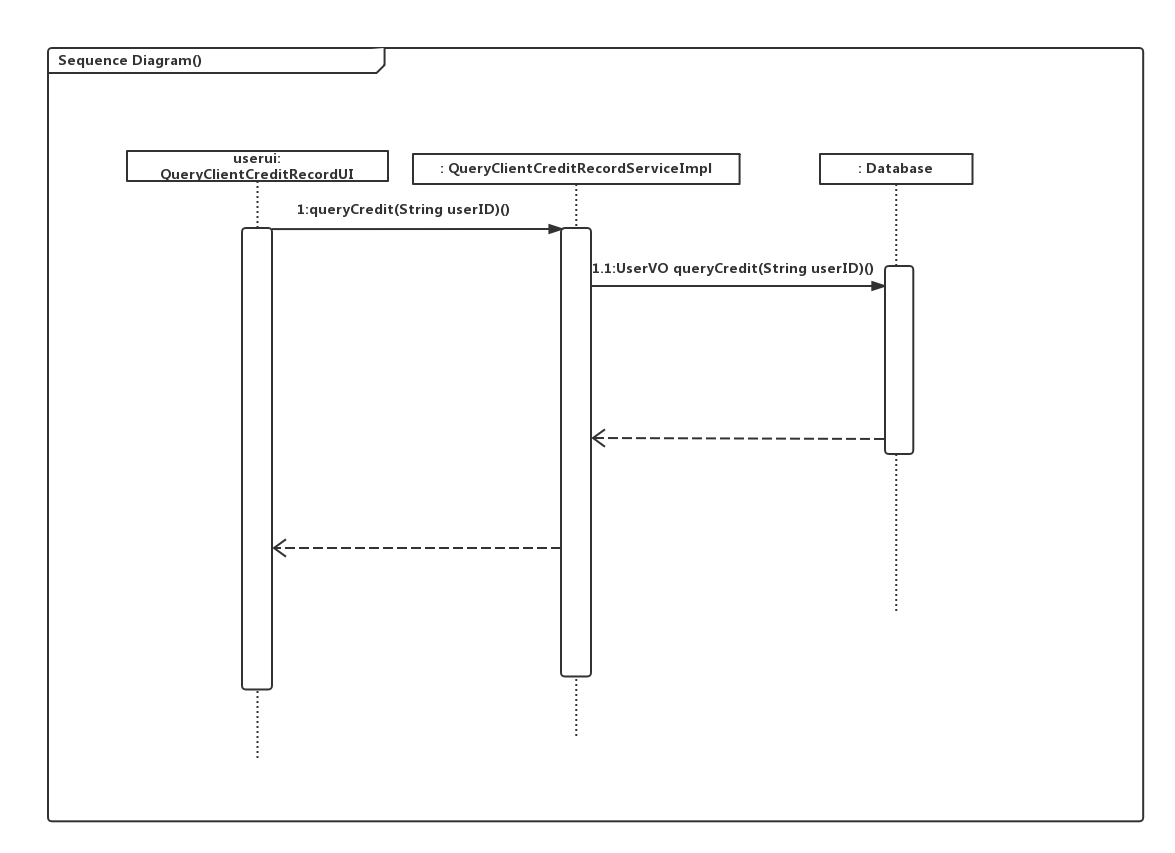
QueryClientCreditRecordServiceImpl的接口规范如下表所示。

QueryClientCreditRecordServiceImpl的接口规范

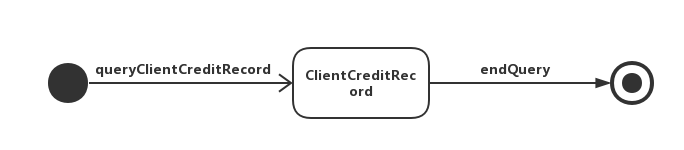
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| QueryClientCreditRecordServiceImpl.queryCreditRecord | 语法 | public ArrayList<ClientCreditRecordVO> queryCreditRecord(String userID) |
| 前置条件 | userID符合输入规范 |
| 后置条件 | 返回该客户的信用记录 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(String userID) | 得到userID对应的用户信息 | |
| userDAO.queryCreditRecord（String userID) | 得到userID对应的用户信用值及信用值变化记录 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

**业务逻辑层的动态模型**

QueryClientCreditRecord的顺序图如下图所示。

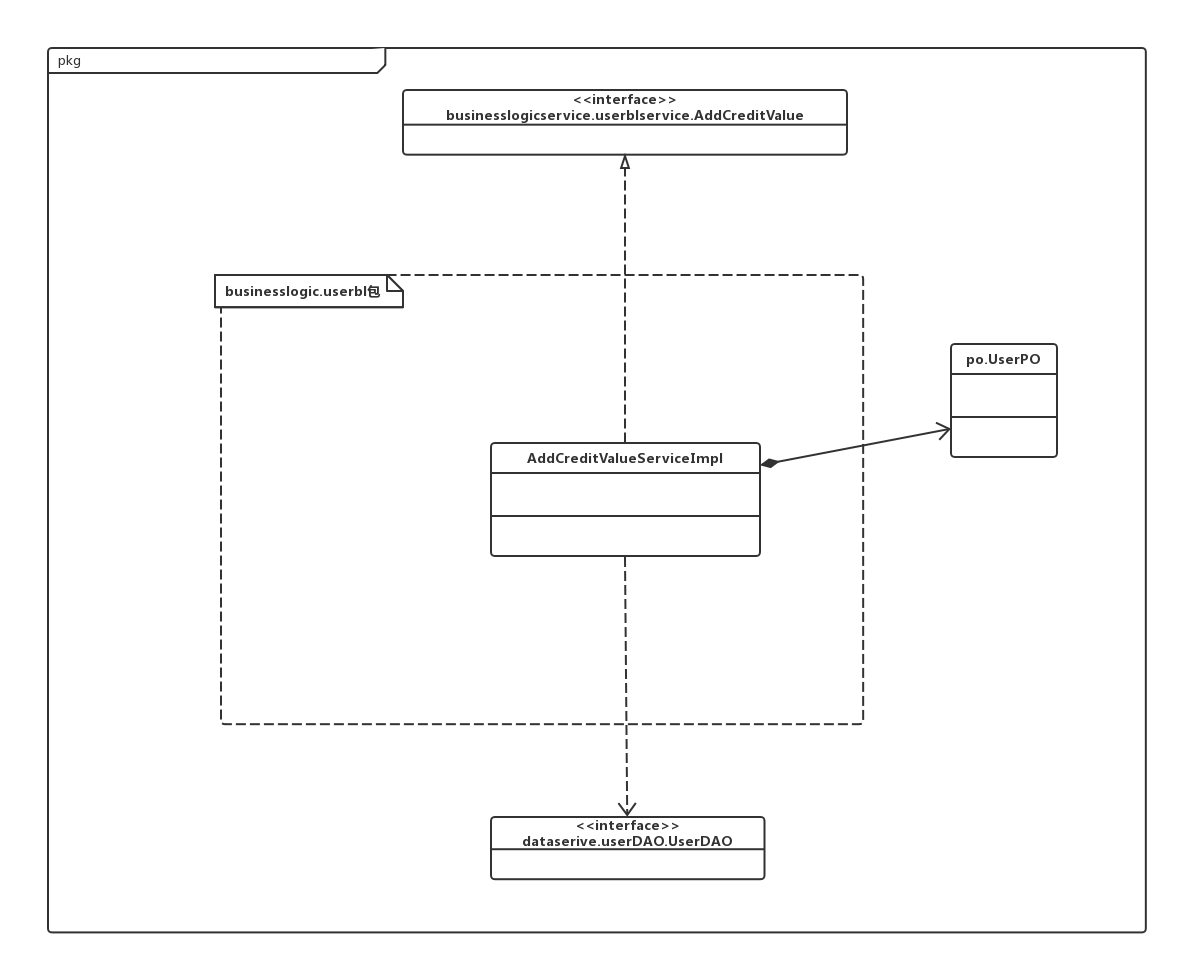
****

QueryClientCreditRecord的状态图如下图所示。

****

* + - 1. **AddCreditValue**

AddCreditValue的设计类图如下图所示。

****

**接口规范：**

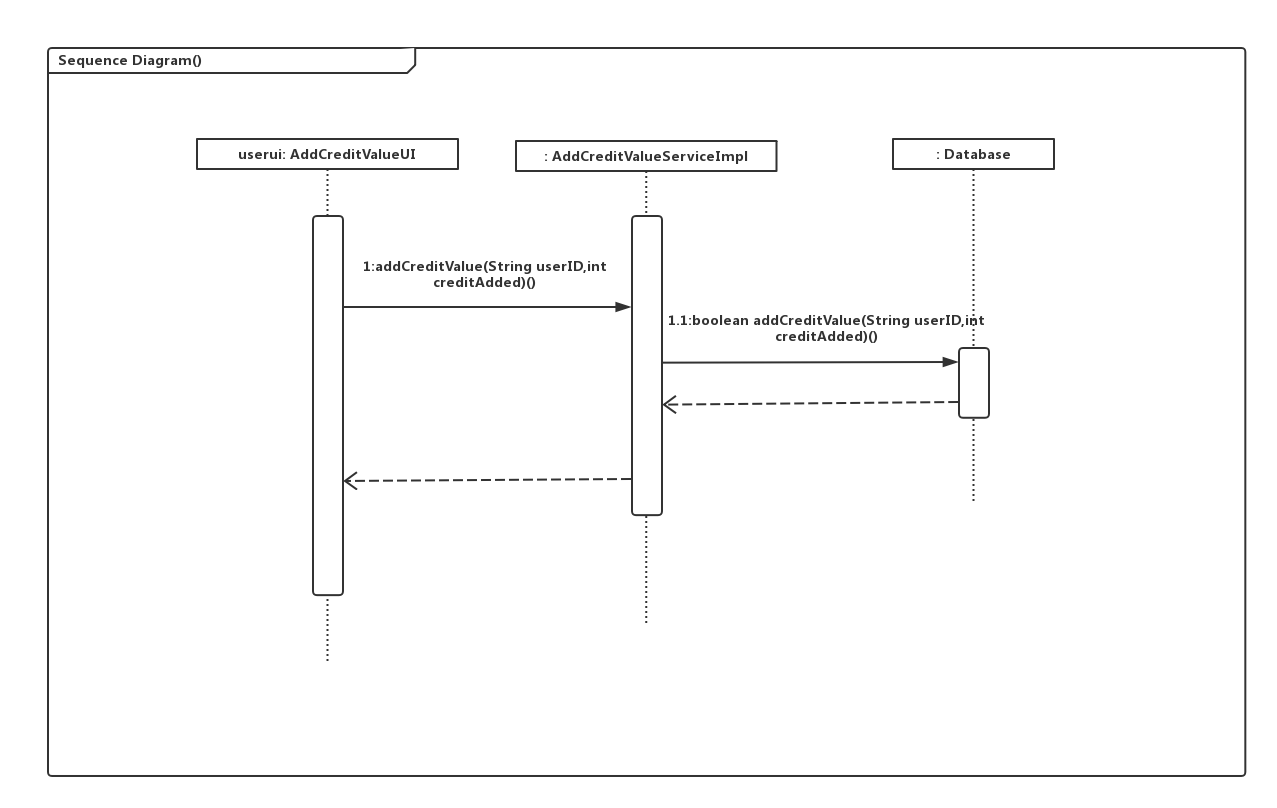
AddCreditValueServiceImpl的接口规范如下表所示。

AddCreditValueServiceImpl的接口规范

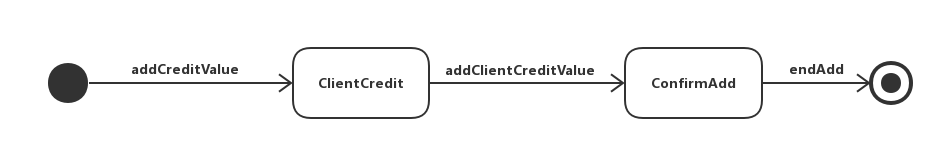
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AddCreditValueServiceImpl.addCreditValue | 语法 | public UserVO addCreditValue(String userID,int creditAdded) |
| 前置条件 | userID和creditAdded符合输入规范 |
| 后置条件 | 返回是否充值成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(String userID) | 得到userID对应的用户信息 | |
| UserDAO.getCreditValue(String userID) | 得到userID对应的用户的当前信用值 | |
| userDAO.update(UserPO po) | 更新单一持久化对象 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

**业务逻辑层的动态模型**

AddCreditValue的顺序图如下图所示。

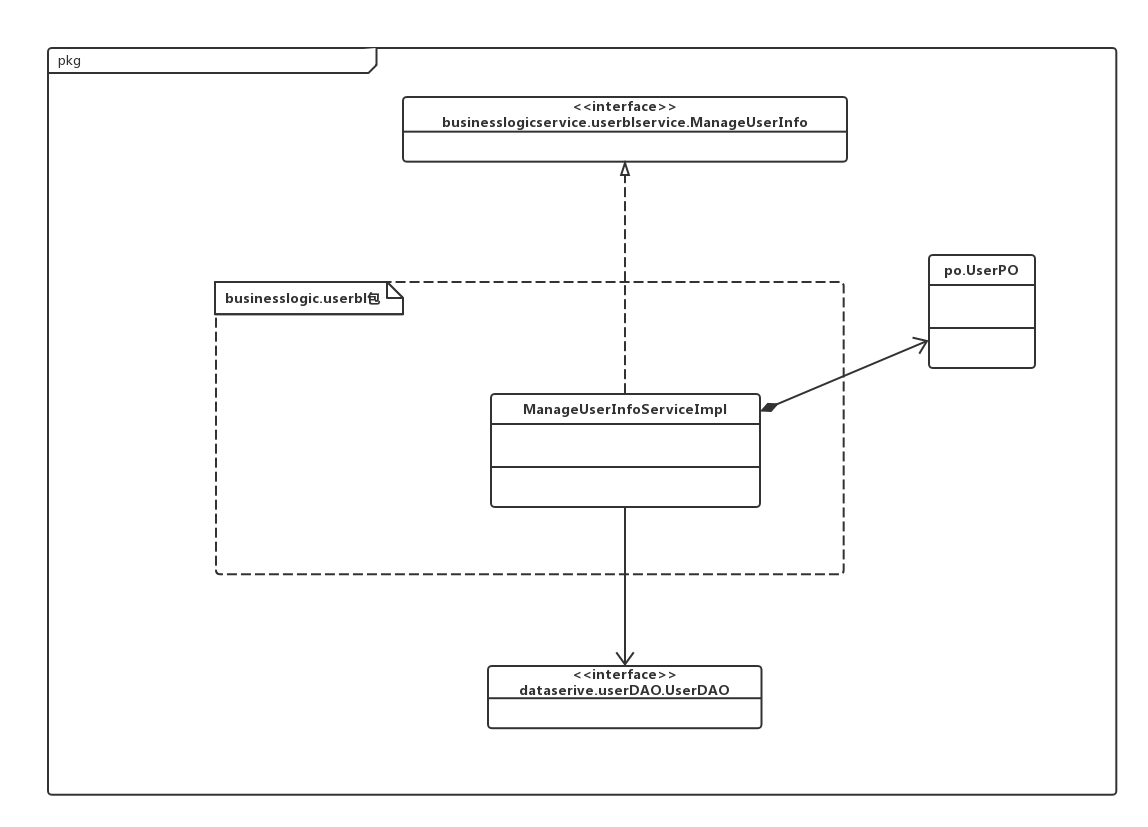
****

AddCreditValue的状态图如下图所示。

****

* + - 1. **ManageUserInfo**

ManageUserInfo的设计类图如下图所示。

****

**接口规范：**

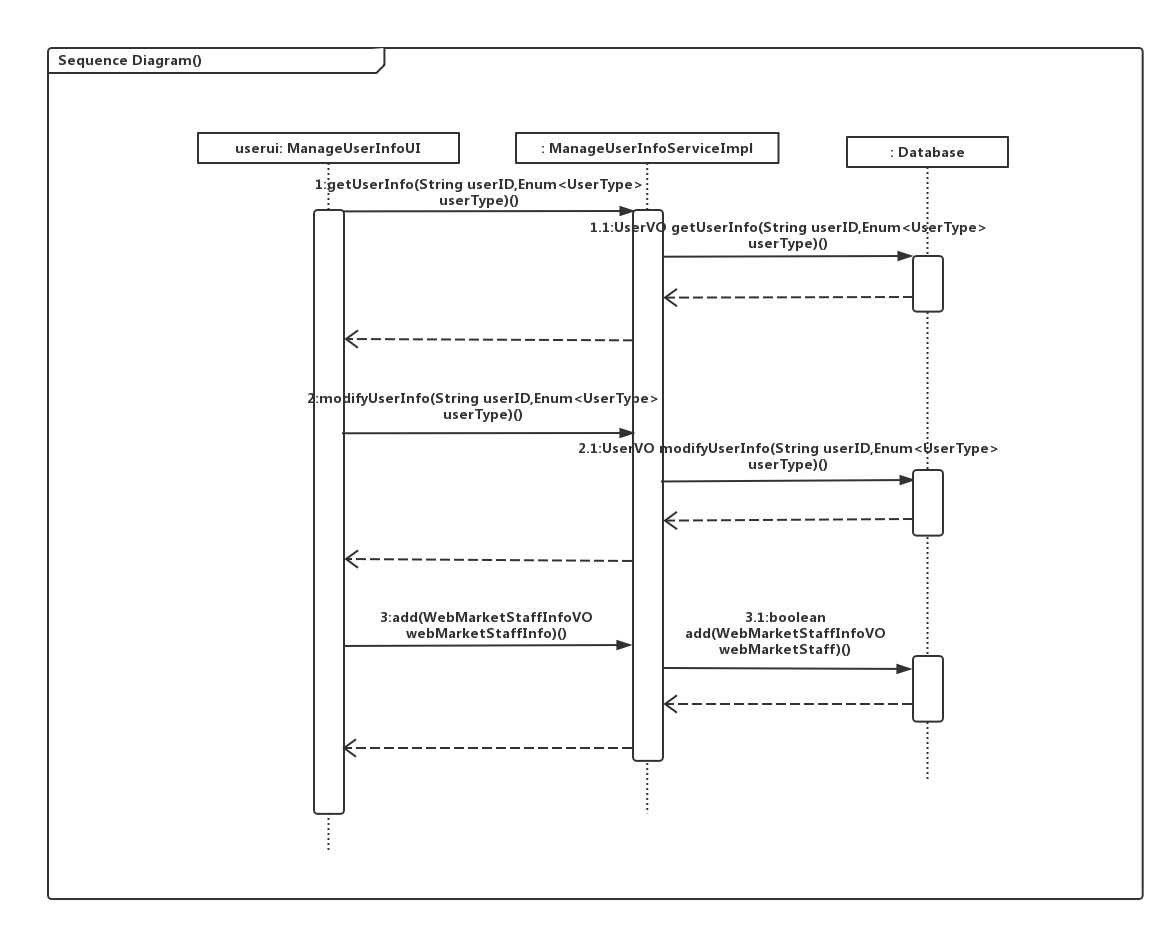
ManageUserInfoServiceInfo的接口规范如下表所示。

ManageUserInfoServiceInfo的接口规范

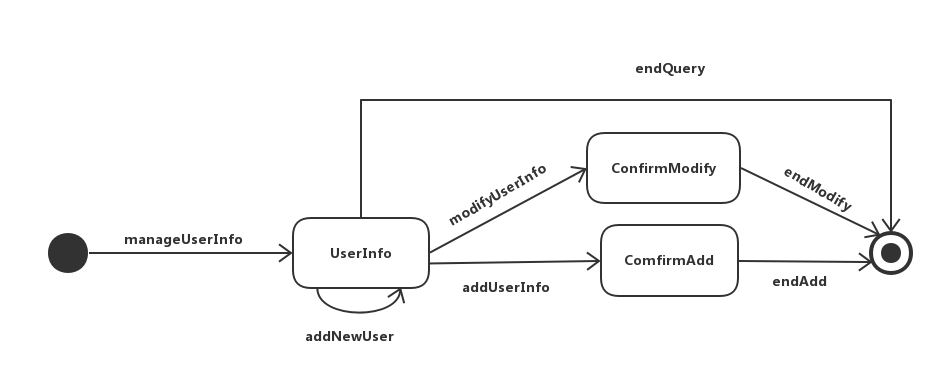
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageUserInfoServiceImpl.modifyUserInfo | 语法 | public boolean modifyUserInfo(UserVO user) |
| 前置条件 | 网站管理人员修改用户信息 |
| 后置条件 | 返回是否修改成功 |
| ManageUserInfoServiceImpl.getUserInfo | 语法 | public UserVO getUserInfo(String userID) |
| 前置条件 | 网站管理人员获取用户信息 |
| 后置条件 | 返回该客户信息 |
| ManageUserInfoServiceImpl.add | 语法 | public boolean add(WebMarketStaffInfoVO webMarketStaff) |
| 前置条件 | 网站管理人员添加网站营销人员 |
| 后置条件 | 返回是否添加成功 |
| ManageUserInfoServiceImpl.getHotelStaffInfo | 语法 | public UserVO getHotelStaffInfo(String userID) |
| 前置条件 | 网站管理人员获取用户信息 |
| 后置条件 | 返回该客户信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(String userID) | 得到userID对应的用户信息 | |
| userDAO.insert(UserPO po) | 插入单一持久化对象 | |
| userDAO.update(UserPO po) | 更新单一持久化对象 | |
| userDAO.finish() | 结束持久化数据库的使用 | |
| DatabaseFactory.getUserDatabase | 得到user数据库的服务的引用 | |

**业务逻辑层的动态模型**

ManageUserInfo的顺序图如下图所示。

****

ManageUserInfo的状态图如下图所示。

****

* + 1. **orderbl模块**

**（1）模块概述**

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模块的职责及接口参见软件体系结构描述文档5.3.2.2

**（2）整体结构**

展示层和业务逻辑层之间，我们添加businesslogicservice.orderblservice. BrowseHotelOrderService接口、businesslogicservice.orderblservice. BrowseUserOrderService接口、businesslogicservice.orderblservice. CheckAbnormalOrderService接口、businesslogicservice.orderblservice. CreateNewOrderService接口、businesslogicservice.orderblservice. GetOrderDoneService接口、businesslogicservice.orderblservice. WithdrawOrderService接口。业务逻辑层和数据层之间添加dataservice.orderDAO.OrderDAO接口。由于本模块涉及到的逻辑控制职责较为简单，将其与业务逻辑职责合并形成BrowseHotelOrderServiceImpl、BrowseUserOrderServiceImpl、CheckAbnormalOrderServiceImpl、CreateNewOrderServiceImpl、GetOrderDoneServiceImpl、WithdrawOrderServiceImpl类。OrderPO、BriefOrderInfoPO是作为订单信息的持久化对象被添加到涉及模型中去的。HotelOrderList的添加是HotelOrder的容器类、UserOrderList的添加是UserOrder的容器类、AbnormalOrderList是AbnormalOrder的容器类。

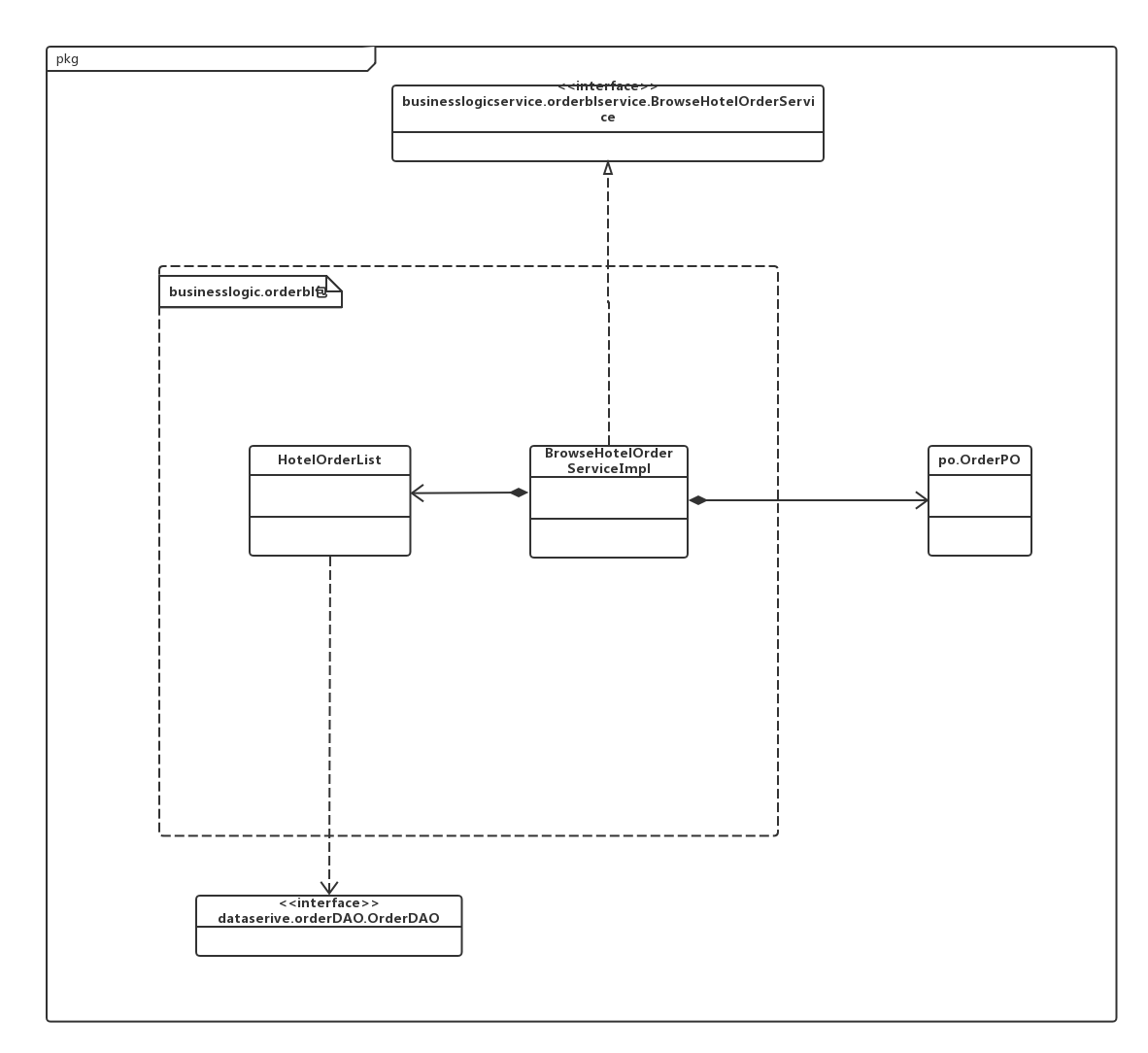
hotelbl模块各个类的职责如下表所示。

hoetlbl模块各个类的职责

|  |  |
| --- | --- |
| **类名** | **职责** |
| browseHotelOrder.  BrowseHotelOrderServiceImpl | 负责实现对于浏览酒店订单界面所需要的服务 |
| browseHotelOrder.  HotelOrderList | 酒店订单列表对象，拥有OrderDAO对象引用，提供查看、搜索订单的服务 |
| browseUserOrder.  BrowseUserOrderServiceImpl | 负责实现对于浏览用户订单界面所需要的服务 |
| browseUserOrder.  UserOrderList | 客户订单列表对象，拥有OrderDAO对象引用，提供查看、搜索订单的服务 |
| checkAbnormalOrder.  CheckAbnormalOrderServiceImpl | 负责实现对于处理异常订单界面所需要的服务 |
| checkAbnormalOrder.  AbnormalOrderList | 异常订单列表对象，拥有OrderDAO对象引用，提供查看、搜索订单的服务 |
| checkAbnormalOrder.  SystemOrderWithdrawer | 负责完成网站营销人员撤销订单操作的各种后置条件，拥有OrderDAO、ClientCreditInfo、RoomInfoService对象的引用 |
| createNewOrder.  CreateNewOrderServiceImpl | 负责实现对于生成订单界面所需要的服务 |
| createNewOrder.  Checker | 负责检查某客户是否可以生成订单以及某订单在当前条件下能否被生成 |
| createNewOrder.  NewOrder | 负责初始化订单，计算订单价格，创建订单，拥有OrderDAO的引用，可以向数据库中增加新的订单信息 |
| getOrderDone.  GetOrderDoneServiceImpl | 负责实现对于执行订单界面所需要的服务 |
| getOrderDone.  OrderTerminator | 负责完成执行订单操作的各种后置条件，拥有OrderDAO、ClientCreditInfo、RoomInfoService对象的引用 |
| withdrawOrder.  WithdrawOrderServiceImp | 负责实现对于撤回订单界面所需要的服务 |
| withdrawOrder.  OrderWithdrawer | 负责完成客户撤销订单操作的各种后置条件，拥有OrderDAO、ClientCreditInfo、RoomInfoService对象的引用 |

**4.1.2.1 BrowseHotelOrder**

BrowseHotelOrder的设计如下图所示。



**接口规范：**

BrowseHotelOrderServiceImpl和HotelOrderList的接口规范如下表所示。

BrowseHotelOrderServiceImpl的接口规范

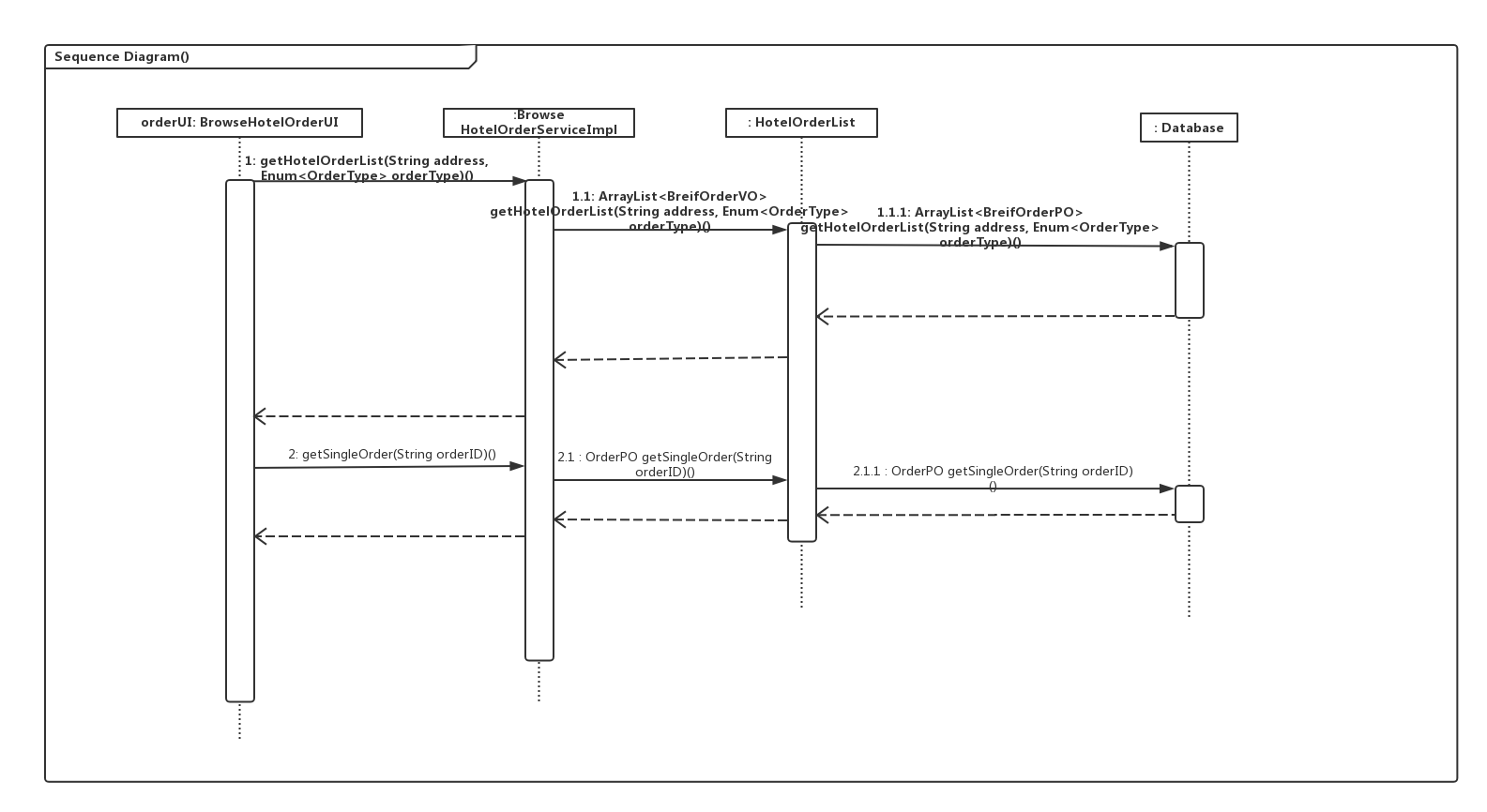
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BrowseHotelOrderServiceImpl.getHotelOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getHotelOrderList (String address, Enum<OrderType> orderType); |
| 前置条件 | 酒店工作人员启动浏览酒店订单流程，，且一个HotelOrderList对象已经被创建 |
| 后置条件 | 调用HotelOrderList的getHotelOrderList方法 |
| BrowseHotelOrderServiceImpl.getSingleOrder | 语法 | public OrderVO getSingleOrder(String address, String orderID); |
| 前置条件 | 酒店工作人员启动浏览酒店订单流程，，且一个HotelOrderList对象已经被创建 |
| 后置条件 | 调用HotelOrderList的getSingleOrder方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelOrderList.getHotelOrderList(String address, Enum<OrderType> orderType); | 得到该酒店所有订单的列表 | |
| HotelOrderList.getSingleOrder(String address, String orderID); | 检测该订单号是否存在，检测该订单号对应的订单是否属于该酒店，得到该订单号对应的订单详情 | |

HotelOrderList的接口规范

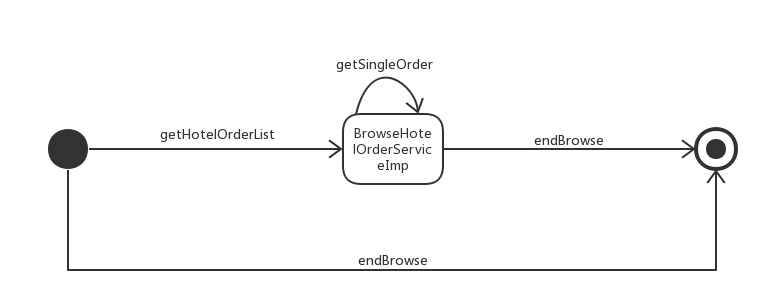
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelOrderList.getHotelOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getHotelOrderList (String address, Enum<OrderType> orderType); |
| 前置条件 | 酒店工作人员启动浏览酒店订单流程 |
| 后置条件 | 按orderType的值进行查找并按生成时间顺序返回相应的订单列表 |
| HotelOrderList.getSingleOrder | 语法 | public OrderVO getSingleOrder(String address, String orderID); |
| 前置条件 | 酒店工作人员已经输入订单号 |
| 后置条件 | 检测该订单号是否存在，如果存在，检测是否是该酒店的订单，若是则返回对应订单的VO，否则返回null |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.getHotelOrderList(String address, Enum<OrderType> orderType) | 得到按orderType的值进行查找并按时间顺序返回该酒店的相应订单列表 | |
| OrderDAO.getSingleOrder(String address, String orderID) | 检测该酒店的订单号是否存在，如果存在，则返回对应订单的VO，否则返回null | |

**业务逻辑层的动态模型**

BrowseHotelOrder的顺序图如下图所示。

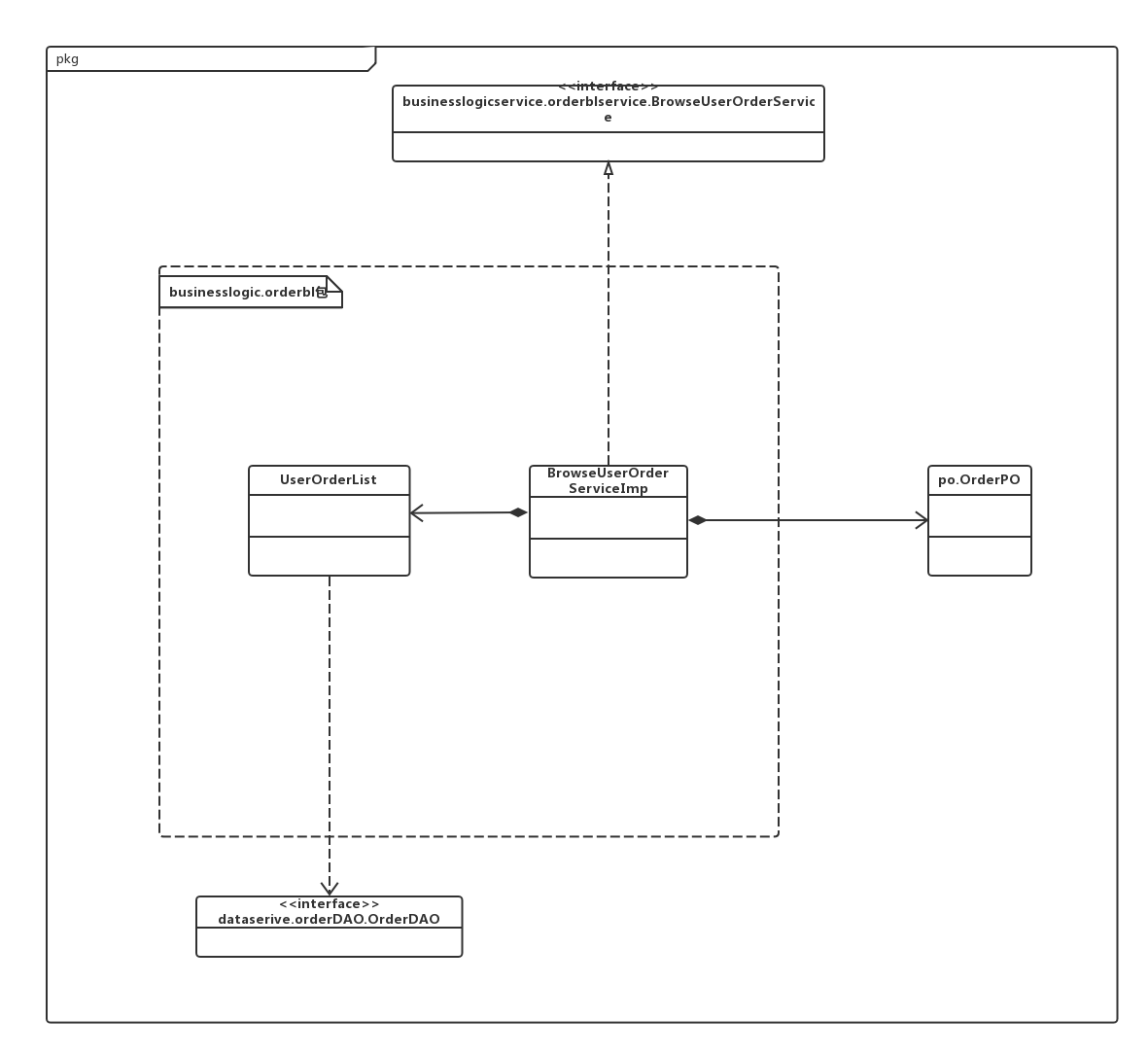
****

BrowseHotelOrder的状态图如下图所示。

****

**4.1.2.2 BrowseUserOrder**

BrowseUserOrder的设计如下图所示。



**接口规范：**

BrowseUserOrderServiceImpl和UserOrderList的接口规范如下表所示。

BrowseUserOrderServiceImpl的接口规范

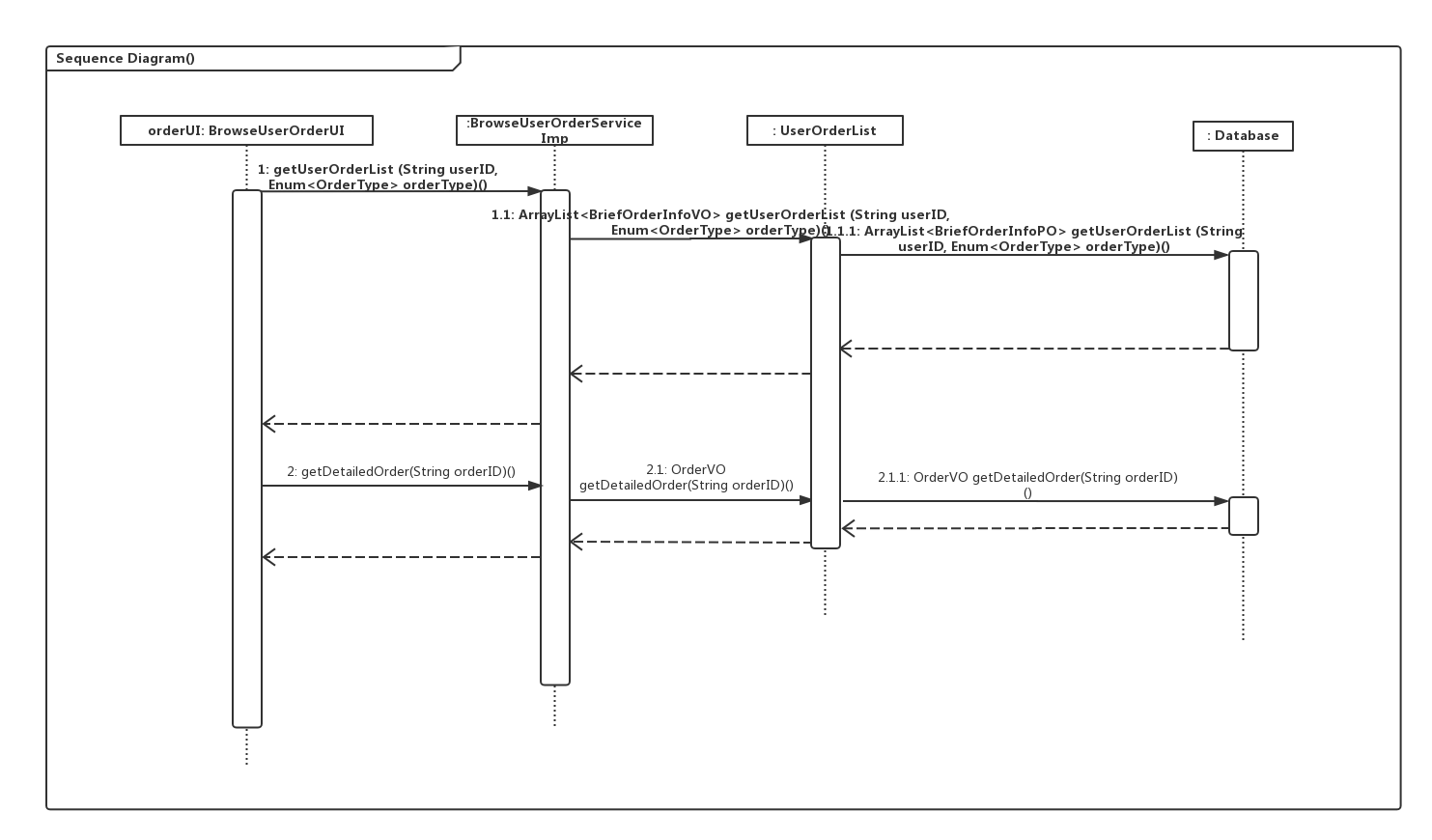
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BrowseUserOrderServiceImpl.getUserOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getUserOrderList (String userID, Enum<OrderType> orderType); |
| 前置条件 | 客户启动浏览订单流程，且一个UserOrderList对象已经被创建 |
| 后置条件 | 调用UserOrderList的getUserOrderList方法 |
| BrowseUserOrderServiceImpl.getDetailedOrder | 语法 | public OrderVO getDetailedOrder (String orderID); |
| 前置条件 | 客户启动浏览订单流程，且一个UserOrderList对象已经被创建 |
| 后置条件 | 调用UserOrderList的getDetailedOrder方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserOrderList.getUserOrderList(String userID, Enum<OrderType> orderType); | 得到该客户所有订单的列表 | |
| UserOrderList.getDetailedOrder(String orderID); | 检测该订单号是否存在，得到该订单号对应的订单详情 | |

UserOrderList的接口规范

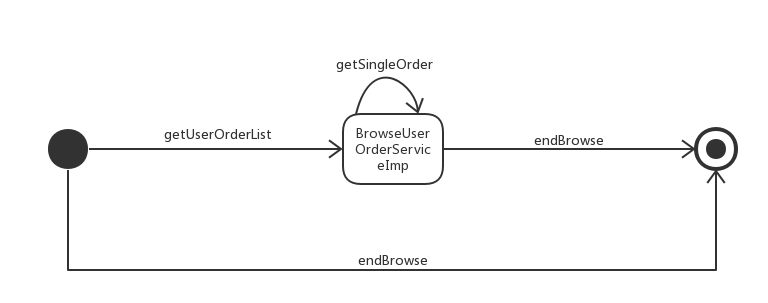
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserOrderList.getUserOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getUserOrderList (String userID, Enum<OrderType> orderType); |
| 前置条件 | 客户启动浏览订单流程 |
| 后置条件 | 按orderType的值进行查找并按生成时间顺序返回相应的订单列表 |
| Order.getDetailedOrder | 语法 | public OrderVO getDetailedOrder(String orderID); |
| 前置条件 | 客户请求展开订单详情 |
| 后置条件 | 检测订单号是否存在，如果存在，则返回对应订单的VO，否则返回null |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.getUserOrderList(String userID, Enum<OrderType> orderType) | 根据orderType的值得到当前用户的对应订单持久化对象列表 | |
| OrderDAO.getDetailedOrder(String orderID) | 检测该订单号是否存在，如果存在，则返回对应订单的VO，否则返回null | |

**业务逻辑层的动态模型**

BrowseUserOrder的顺序图如下图所示。

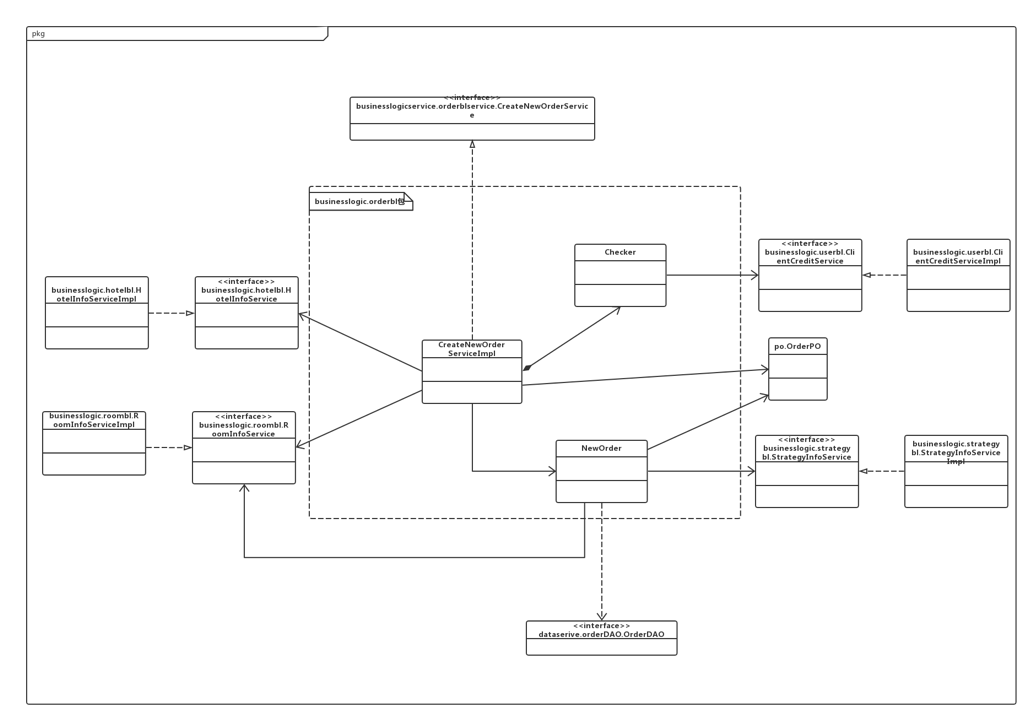
****

BrowseUserOrder的状态图如下图所示。

****

**4.1.2.3 CreateNewOrder**

CreateNewOrder的设计如下图所示。



**接口规范：**

CreateNewOrderServiceImpl、NewOrder和Checker的接口规范如下表所示。

CreateNewOrderServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreateNewOrderServiceImpl.initNewOrder | 语法 | public OrderVO initNewOrder(String userID, String address); |
| 前置条件 | 客户确认即将生成订单的酒店信息，且一个Checker和一个NewOrder对象均被创建 |
| 后置条件 | 调用Checker的canUserCreateNewOrder方法，调用NewOrder的initNewOrder方法 |
| CreateNewOrderServiceImpl. getAvailableRoomNum | 语法 | public int getAvailableRoomNum(String address, Enum<RoomType> roomType); |
| 前置条件 | 客户已经选择房型，且一个RoomInfoService 实现类已经被创建 |
| 后置条件 | 调用RoomInfoService的getAvailableRoomNum方法 |
| CreateNewOrderServiceImpl. checkNewOrder | 语法 | public boolean checkNewOrder( OrderVO vo); |
| 前置条件 | 客户确认生成订单，且一个Checker已被创建 |
| 后置条件 | 调用checker的checkNewOrder方法 |
| CreateNewOrderServiceImpl. getPrice | 语法 | public int getPrice( OrderVO vo); |
| 前置条件 | Checker已经检查过可以生成订单，且一个NewORder对象已被创建 |
| 后置条件 | 调用NewOrder的getPrice方法 |
| CreateNewOrderServiceImpl. addNewOrder | 语法 | public boolean addNewOrder( OrderVO vo); |
| 前置条件 | Checker已经检查过可以生成订单，且一个NewORder对象已被创建 |
| 后置条件 | 调用NewOrder的addNewOrder方法 |
| CreateNewOrderServiceImpl. getOriginalPrice | 语法 | public int getOriginalPrice(String hotelAddress, RoomType roomType); |
| 前置条件 | 一个NewOrder对象已被创建 |
| 后置条件 | 调用NewOrder的getOriginalPrice方法 |
| CreateNewOrderServiceImpl. getAvailblePromotionName | 语法 | public String getAvailblePromotionName(OrderVO order); |
| 前置条件 | 其他订单条件已经被填写，且一个NewORder对象已被创建 |
| 后置条件 | 调用StrategyInfoService的getAvailblePromotionName方法 |
| CreateNewOrderServiceImpl. getAvailbleMarketStrategyName | 语法 | public String getAvailbleMarketStrategyName(OrderVO order); |
| 前置条件 | 其他订单条件已经被填写，且一个NewORder对象已被创建 |
| 后置条件 | 调用StrategyInfoService的getAvailbleMarketStrategyName方法 |
|  | | |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Checker.canUserCreateNewOrder(String userID) | 判断该客户能否生成订单 | |
| Checker. checkNewOrder(OrderVO vo) | 检查该订单在当前时刻能否被生成 | |
| NewOrder.initNewOrder(String userID, String address) | 初始化订单信息，返回一个被初始化的OrderVO | |
| NewOrder.getPrice(OrderVO vo) | 计算并返回订单价格 | |
| NewOrder.addNewOrder(OrderVO vo) | 增加新的订单 | |
| RoomInfoService.getAvailableRoomNum(String address, Enum<RoomType> roomType) | 获取当前酒店roomType房型的空房数量 | |
| NewOrder. getOriginalPrice(String hotelAddress, RoomType roomType); | 得到酒店各房型的原始价格 | |
| StrategyInfoService. getAvailblePromotionName(OrderVO order); | 获得当前条件下可用的唯一可用酒店促销策略 | |
| StrategyInfoService. getAvailbleMarketStrategyName(OrderVO order); | 获得当前条件下可用的唯一可用网站营销策略 | |

Checker的接口规范

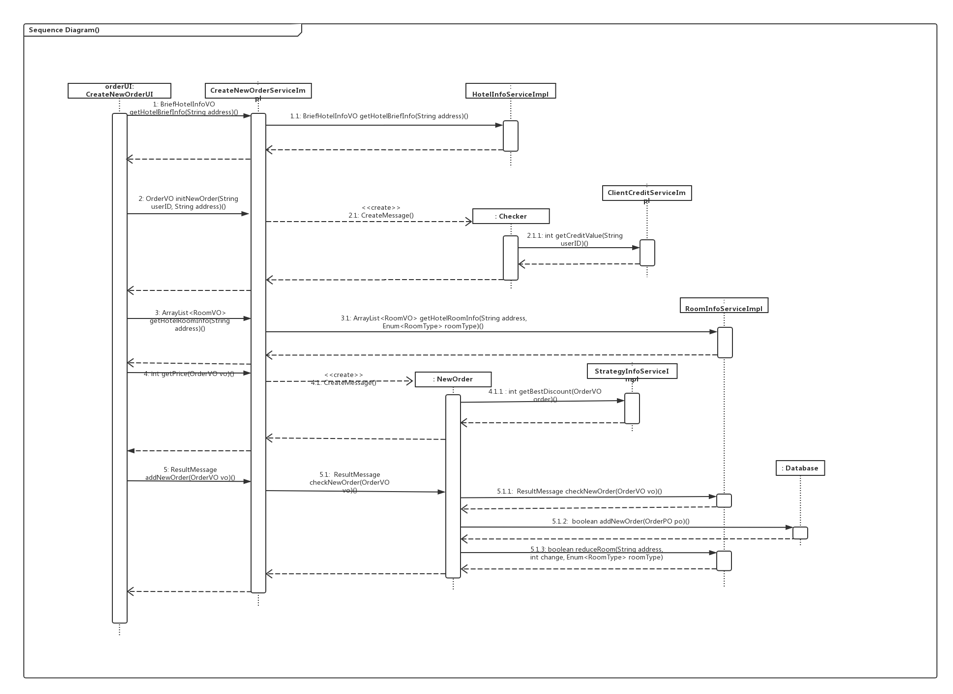
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Checker.canUserCreateNewOrder | 语法 | public boolean canUserCreateNewOrder(String userID) |
| 前置条件 | 客户启动浏览订单流程 |
| 后置条件 | 判断该客户能否生成订单，返回一个布尔值 |
| Checker. checkNewOrder | 语法 | public ResultMessage checkNewOrder(OrderVO vo) |
| 前置条件 | 客户确认生成订单 |
| 后置条件 | 检查该订单在当前时刻能否被生成，返回一个布尔值 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ClientCreditInfo. getCreditValue(String userID); | 检查该订单在当前时刻能否被生成，返回一个布尔值 | |
| RoomInfoService.checkNewOrder(OrderVO vo) | 检查该订单在当前时刻能否被生成，返回一个布尔值 | |

NewOrder的接口规范

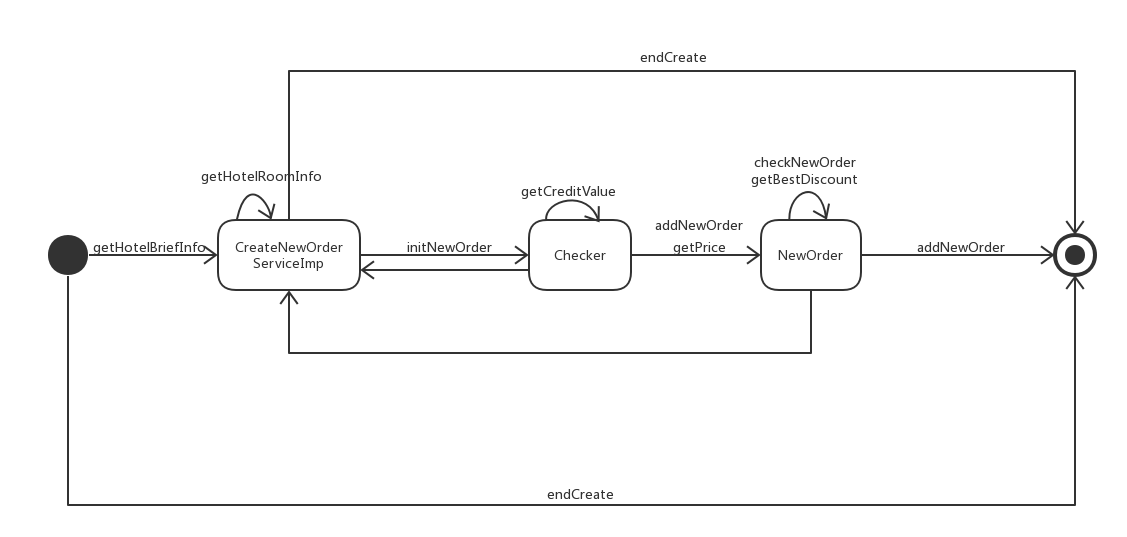
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| NewOrder.initNewOrder | 语法 | public OrderVO initNewOrder(String userID, String address); |
| 前置条件 | 客户选择生成订单 |
| 后置条件 | 初始化订单信息，返回一个被初始化的OrderVO |
| NewOrder.getPrice | 语法 | public int getPrice(OrderVO vo); |
| 前置条件 | 客户确认生成订单 |
| 后置条件 | 调用StrategyInfoService的getBestDiscount，计算并返回订单价格 |
| NewOrder.addNewOrder | 语法 | public boolean addNewOrder(OrderVO vo); |
| 前置条件 | 客户确认生成订单 |
| 后置条件 | 增加新的订单 |
| NewOrder. getOriginalPrice | 语法 | public int getOriginalPrice(String hotelAddress, RoomType roomType); |
| 前置条件 | 客户选择生成订单 |
| 后置条件 | 返回对应酒店各个房型对应的原始价格 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.insertOrder(OrderPO po) | 插入单一持久化对象 | |
| DatabaseFactory.getOrderDatabase | 得到Order数据库的服务的引用 | |
| StrategyInfoService.getBestDiscount(OrderVO vo) | 得到当前最终的最优的优惠折扣 | |
| HotelInfoService. getOriginalPrice(String hotelAddress, RoomType roomType) | 得到对应酒店各个房型对应 | |

**业务逻辑层的动态模型**

CreateNewOrder的顺序图如下图所示。

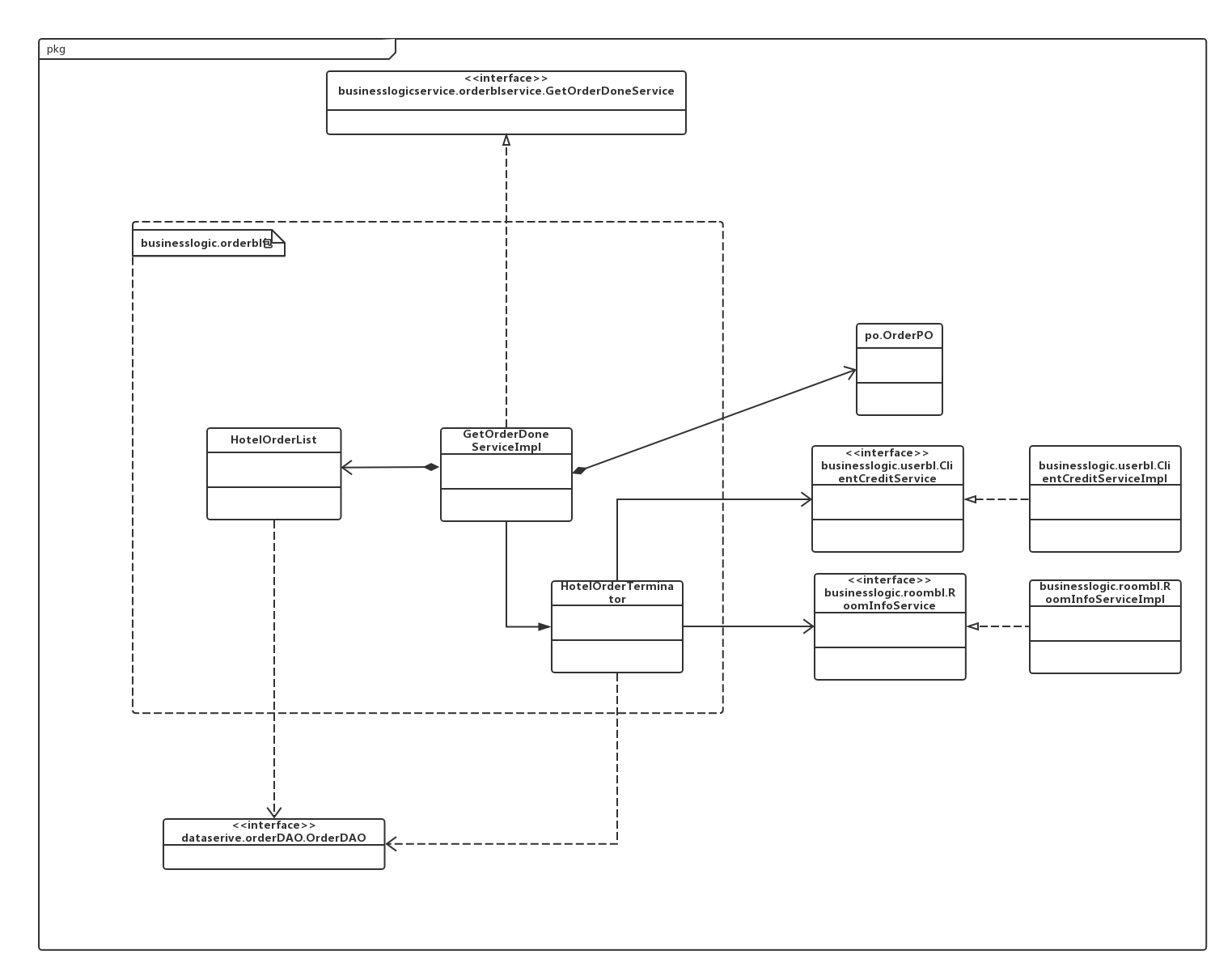
****

CreateNewOrder的状态图如下图所示。

****

**4.1.2.4 GetOrderDone**

GetOrderDone的设计如下图所示。



**接口规范：**

GetOrderDoneServiceImpl和OrderTerminator的接口规范如下表所示

GetOrderDoneServiceImpl的接口规范

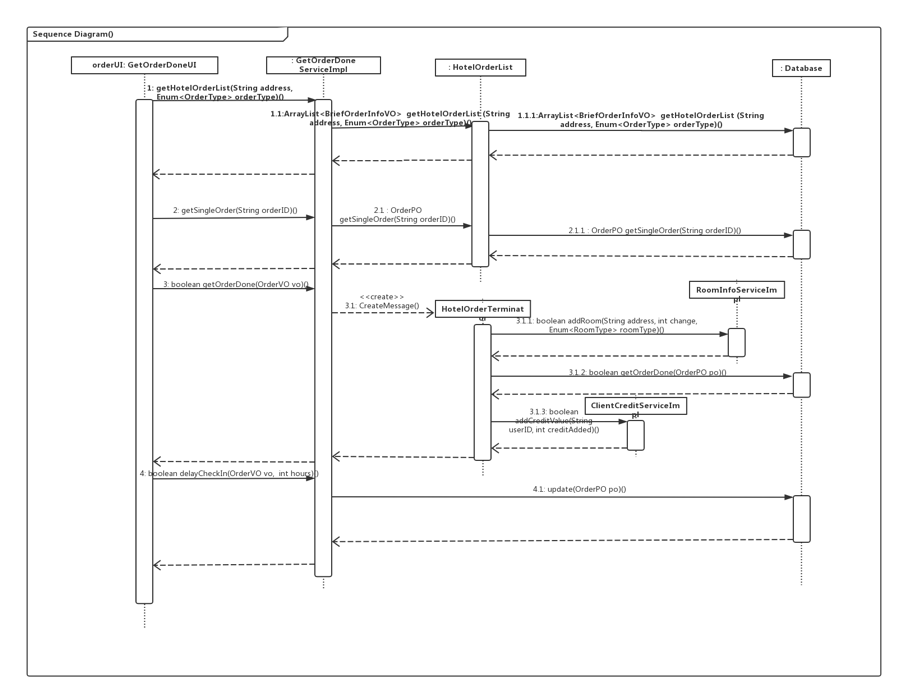
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| GetOrderDoneServiceImpl.getHotelNotDoneOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getHotelNotDoneOrderList(String address); |
| 前置条件 | 酒店工作人员启动执行酒店订单流程，且一个HotelOrderList对象已经被创建 |
| 后置条件 | 调用HotelOrderList的getHotelOrderList方法 |
| GetOrderDoneServiceImpl. getSingleOrder | 语法 | public OrderVO getSingleOrder(String address, String orderID); |
| 前置条件 | 酒店工作人员启动执行酒店订单流程，且一个HotelOrderList对象已经被创建 |
| 后置条件 | 调用HotelOrderList的getSingleOrder方法 |
| GetOrderDoneServiceImpl. getOrderDone | 语法 | public boolean getOrderDone(OrderVO vo); |
| 前置条件 | 酒店工作人员选择执行酒店订单，且一个OrderTerminator对象已经被创建 |
| 后置条件 | 调用OrderTerminator的getOrderDone方法 |
| GetOrderDoneServiceImpl. delayCheckIn | 语法 | public boolean delayCheckIn (OrderVO vo); |
| 前置条件 | 酒店工作人员选择延迟入住，且一个OrderTerminator对象已经被创建 |
| 后置条件 | 调用OrderTerminator的delayCheckIn方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelOrderList.getHotelOrderList(String address, Enum<OrderType> orderType); | 得到该酒店对应类型订单的列表（此处为未执行订单） | |
| HotelOrderList.getSingleOrder(String address, String orderID); | 检测该订单号是否存在，检测该订单号对应的订单是否属于该酒店，得到该订单号对应的订单详情 | |
| OrderTerminator.getOrderDone(OrderVO vo) | 将该订单改为已执行状态，记录入住时间，然后为该客户增加与订单价值等额的信用值, 并增加可用客房 | |
| OrderTerminator.delayCheckIn(OrderVO vo) | 将该订单置为已执行订单，记录入住时间，恢复扣除的信用值, 并增加可用客房 | |

OrderTerminator的接口规范

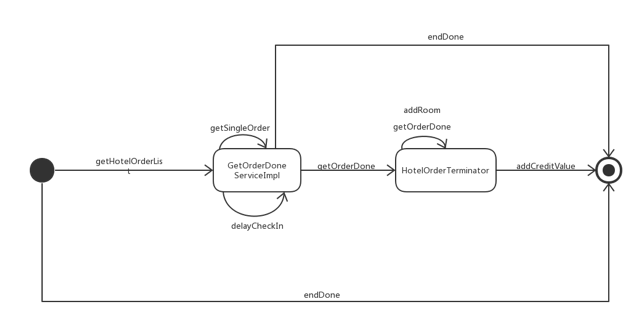
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| GetOrderDoneServiceImpl. getOrderDone | 语法 | public boolean getOrderDone(OrderVO vo); |
| 前置条件 | 酒店工作人员选择执行酒店订单 |
| 后置条件 | 将该订单改为已执行状态，记录入住时间，然后为该客户增加与订单价值等额的信用值, 并增加可用客房 |
| GetOrderDoneServiceImpl. delayCheckIn | 语法 | public boolean delayCheckIn (OrderVO vo); |
| 前置条件 | 酒店工作人员选择延迟入住 |
| 后置条件 | 将该订单置为已执行订单，记录入住时间，恢复扣除的信用值, 并增加可用客房 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.updateOrder(OrderPO po) | 更新单一持久化对象 | |
| RoomInfoService.addSpareRoom(String address, int change, Enum<RoomType> roomType) | 更新（增加）当前酒店被预定房型的可用房间数据 | |
| ClientCreditInfo.changeCreditValue(String userID, int num) | 根据num(可正可负)的值，增加或减少当前客户的信用值 | |

**业务逻辑层的动态模型**

GetOrderDone的顺序图如下图所示。

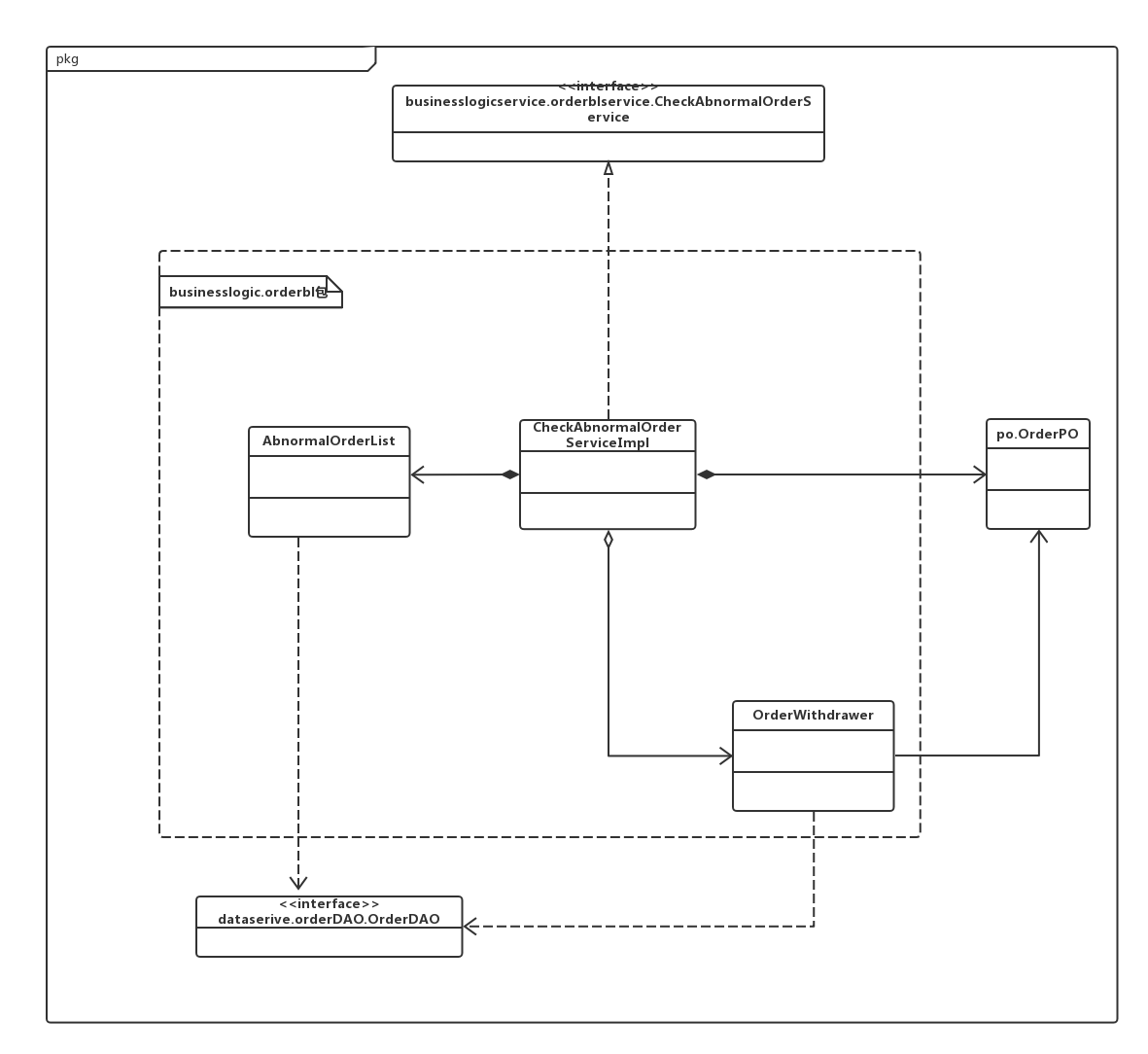
****

GetOrderDone的状态图如下图所示。

****

**4.1.2.5 CheckAbnormalOrder**

CheckAbnormalOrder的设计如下图所示。



**接口规范：**

CheckAbnormalOrderServiceImpl、AbnormalOrderList和SystemOrderWithdrawer的接口规范如下表所示。

CheckAbnormalOrderServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckAbnormalOrderServiceImpl.getAbnormalOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getAbnormalOrderList(Date date); |
| 前置条件 | 网站营销人员启动浏览每日异常订单流程，且一个AbnormalOrderList对象已经被创建 |
| 后置条件 | 调用AbnormalOrderList的getAbnormalOrderList方法 |
| CheckAbnormalOrderServiceImpl.getDetailedOrder | 语法 | public OrderVO getDetailedOrder (String orderID); |
| 前置条件 | 网站营销人员请求浏览订单详情，且一个AbnormalOrderList对象已经被创建 |
| 后置条件 | 调用AbnormalOrderList的getDetailedOrder方法 |
| CheckAbnormalOrderServiceImpl.systemWithdrawOrder | 语法 | **public** **boolean** systemWithdrawOrder(OrderVO vo, **boolean** isRecoverHalfCredit); |
| 前置条件 | 网站营销人员请求撤销订单详情，且一个SystemOrderWithdrawer对象已经被创建 |
| 后置条件 | 调用SystemOrderWithdrawer的systemWithdrawOrder方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AbnormalOrderList. getAbnormalOrderList(Date date) | 得到该日期所有异常订单的列表 | |
| AbnormalOrderList.getDetailedOrder(String orderID) | 检测该订单号是否存在，得到该订单号对应的订单详情 | |
| SystemOrderWithdrawer. systemWithdrawOrder(OrderVO vo, **boolean** isRecoverHalfCredit) | 根据订单VO中信息和是否恢复一半的信用值，来撤销订单，并恢复用户全部或一半的信用值 | |

AbnormalOrderList的接口规范

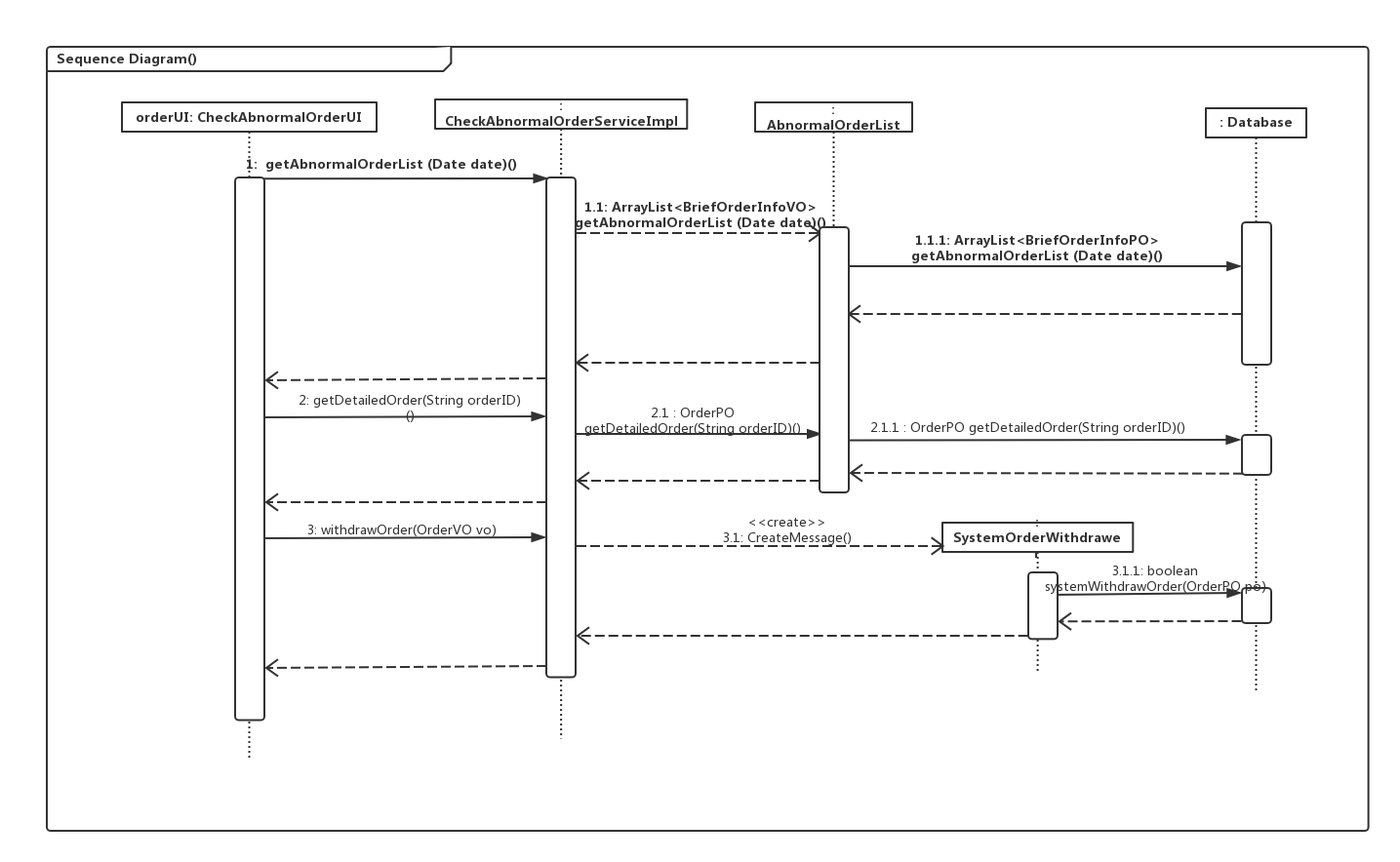
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AbnormalOrderList. getAbnormalOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getAbnormalOrderList (Date date); |
| 前置条件 | 网站营销人员启动浏览每日异常订单流程 |
| 后置条件 | 得到改日期所有异常订单的列表 |
| AbnormalOrderList.getDetailedOrder | 语法 | public OrderVO getDetailedOrder(String orderID); |
| 前置条件 | 网站营销人员请求浏览订单详情 |
| 后置条件 | 检测订单号是否存在，如果存在，则返回对应订单的VO，否则返回null |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.getAllAbnormalList (Date date) | 得到所查询日期当天所有异常订单的列表 | |
| OrderDAO.getDetailedOrder(String orderID) | 检测该订单号是否存在，如果存在，则返回对应订单的VO，否则返回null | |

SystemOrderWithdrawer的接口规范

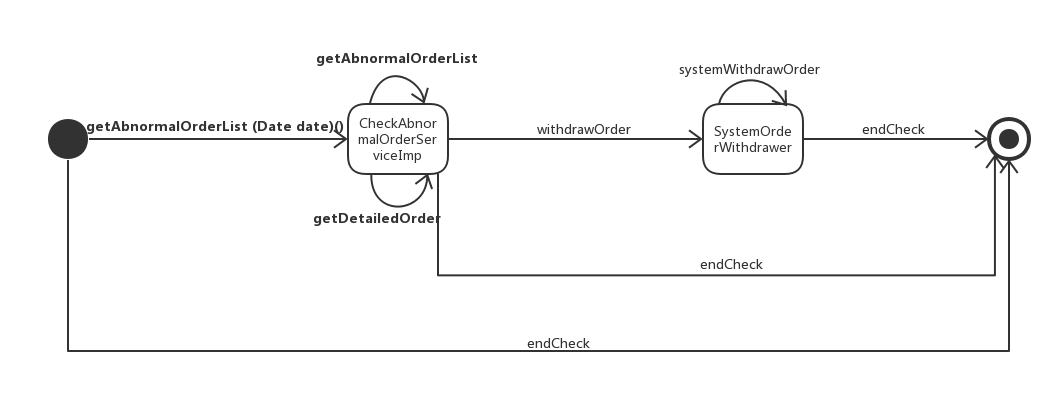
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SystemOrderWithdrawer.systemWithdrawOrder(OrderVO vo, boolean isRecoverHalfCredit) | 语法 | public boolean systemWithdrawOrder(OrderVO vo, boolean isRecoverHalfCredit); |
| 前置条件 | 网站营销人员确认撤销订单，且已经选择了恢复客户已经扣除的信用值的全部或一半 |
| 后置条件 | 撤销此异常订单并将其状态置为已撤销、记录撤销时间，恢复此客户信用值的全部或一半 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.updateOrder(OrderPO po) | 更新单一持久化对象 | |
| RoomInfoService.addSpareRoom(String address, int change, Enum<RoomType> roomType) | 更新（增加）当前酒店被预定房型的可用房间数据 | |
| ClientCreditInfo.changeCreditValue(String userID, int num) | 根据num(可正可负)的值，增加或减少当前客户的信用值 | |

**业务逻辑层的动态模型**

CheckAbnormalOrder的顺序图如下图所示。

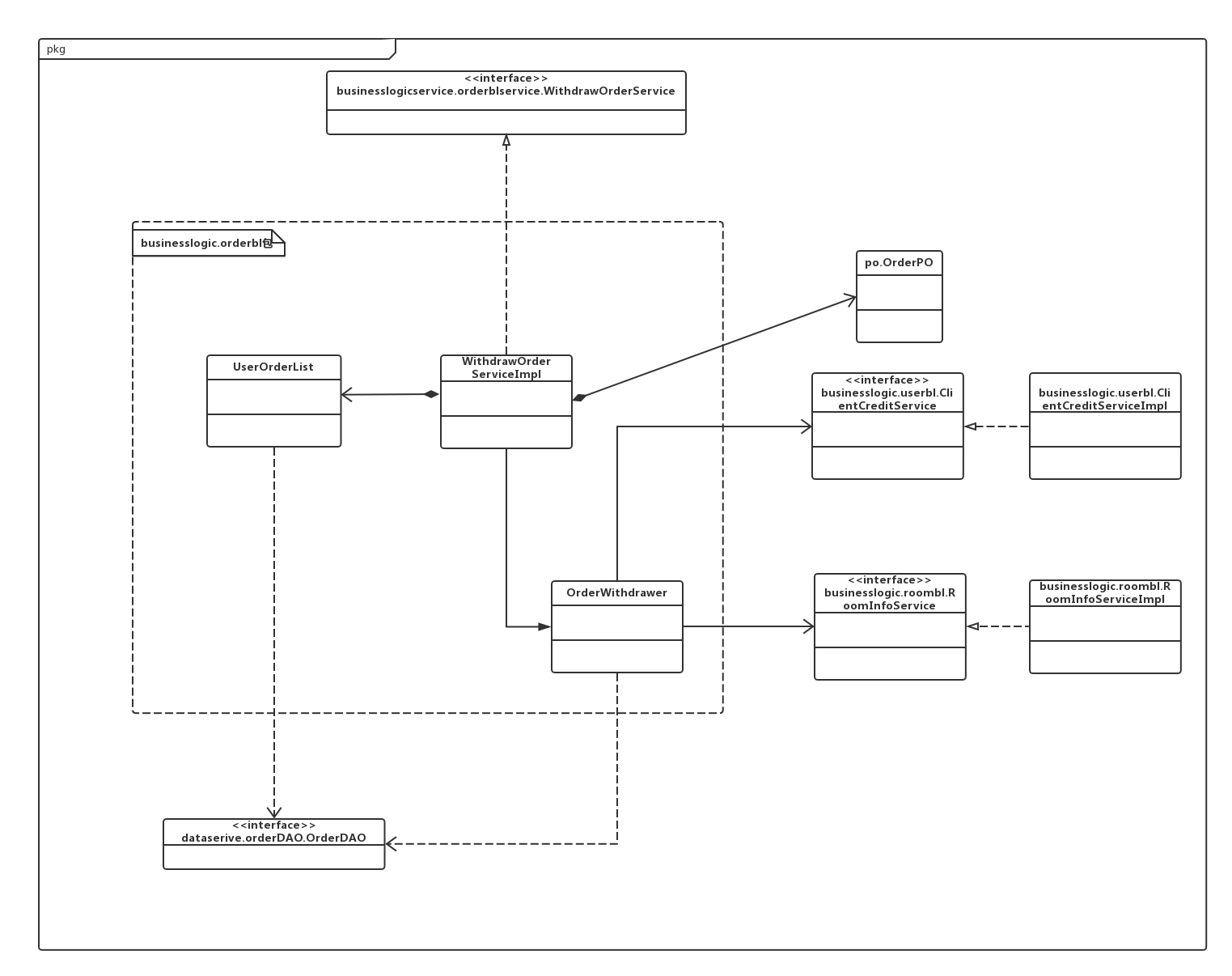
****

CheckAbnormalOrder的状态图如下图所示。

****

**4.1.2.6 WithdrawOrder**

WithdrawOrder的设计如下图所示。



**接口规范：**

WithdrawOrderServiceImpl和OrderWithdrawer的接口规范如下表所示

WithdrawOrderServiceImpl的接口规范

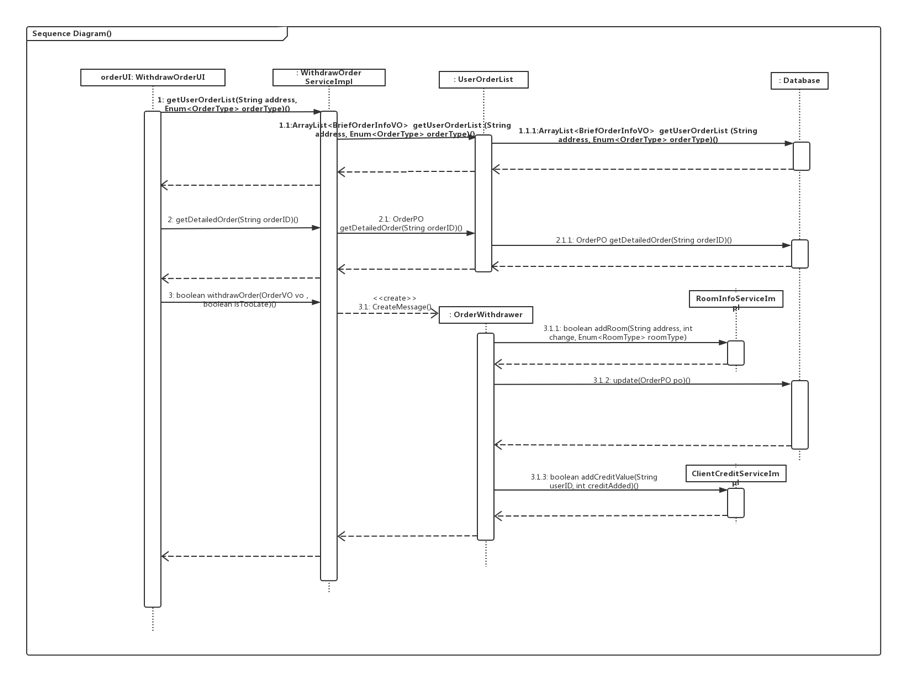
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WithdrawOrderServiceImpl.getUserOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getUserOrderList (String userID, Enum<OrderType> orderType); |
| 前置条件 | 客户启动浏览订单流程，，且一个UserOrderList对象已经被创建 |
| 后置条件 | 调用UserOrderList的getUserOrderList方法 |
| WithdrawOrderServiceImpl.getDetailedOrder | 语法 | public OrderVO getDetailedOrder (String orderID); |
| 前置条件 | 客户启动浏览订单流程，，且一个UserOrderList对象已经被创建 |
| 后置条件 | 调用UserOrderList的getDetailedOrder方法 |
| WithdrawOrderServiceImpl. withdrawOrder | 语法 | public boolean withdrawOrder(OrderVO vo, boolean isTooLate); |
| 前置条件 | 客户确认撤销订单，且一个OrderWithdrawer对象已经被创建 |
| 后置条件 | 调用OrderWithdrawer的withdrawOrder方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserOrderList.getUserOrderList(String userID, Enum<OrderType> orderType); | 得到该客户所有订单的列表 | |
| UserOrderList.getSingleOrder(String orderID); | 检测该订单号是否存在，得到该订单号对应的订单详情 | |
| OrderWithdrawer. withdrawOrder(OrderVO vo, boolean isTooLate) | 更改订单信息，置为已撤销状态，记录撤销时间, 根据isTooLate更改客户信用值 | |

OrderWithdrawer的接口规范

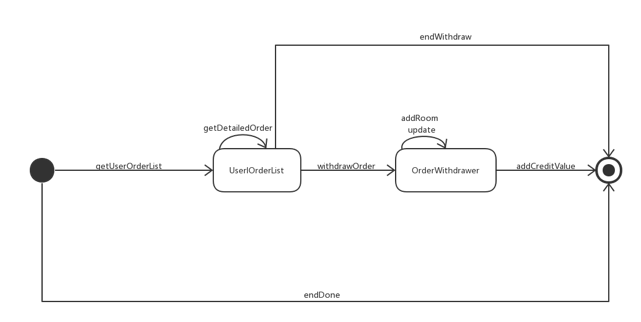
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderWithdrawer. withdrawOrder | 语法 | public boolean withdrawOrder(OrderVO vo, boolean isTooLate); |
| 前置条件 | 客户确认撤销订单 |
| 后置条件 | 更改订单信息，置为已撤销状态，记录撤销时间, 根据isTooLate更改客户信用值 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.updateOrder(OrderPO po) | 更新单一持久化对象 | |
| RoomInfoService.addSpareRoom(String address, int change, Enum<RoomType> roomType) | 更新（增加）当前酒店被预定房型的可用房间数据 | |
| ClientCreditInfo.changeCreditValue(String userID, int num) | 根据num(可正可负)的值，增加或减少当前客户的信用值 | |

**业务逻辑层的动态模型**

OrderWithdrawer的顺序图如下图所示。

****

OrderWithdrawer的状态图如下图所示。

****

**4.1.3** **hotelbl模块**

**（1）模块概述**

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模块的职责及接口参见软件体系结构描述文档5.3.2.3

**（2）整体结构**

展示层和业务逻辑层之间，我们添加businesslogicservice.hotelblservice. CheckOrderedHotelService接口、businesslogicservice.hotelblservice. CommentOnHotelService接口、businesslogicservice.hotelblservice. ImportNewRoomService接口、businesslogicservice.hotelblservice. MaintainHotelBasicInfoService接口、businesslogicservice.hotelblservice. ManageHotelInfoService接口、businesslogicservice.hotelblservice. QueryHotelService接口、businesslogicservice.hotelblservice. SearchHotelService接口。业务逻辑层和数据层之间添加dataservice.hotelDAO.HotelDAO接口。由于本模块涉及到的逻辑控制职责较为简单，将其与业务逻辑职责合并形成CheckOrderedHotelServiceImpl、CommentOnHotelServiceImpl、ImportNewRoomServiceImpl、MaintainHotelBasicInfoServiceImpl、ManageHotelInfoServiceImpl、QueryHotelServiceImpl、SearchHotelServiceImpl类。HotelPO、BriefHotelInfoPO是作为酒店信息的持久化对象被添加到涉及模型中去的。OrderedHotelList和OrderedHotelItem的添加是OrderedHotelInfo的容器类、CommentableHotelList的添加是CommentableHotel的容器类、AvailableRoomList和AvailableRoomItem的添加是AvailableRoom的容器类、QueryHotelList的添加是QueryHotel的容器类。HotelInfoService、OrderInfo都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

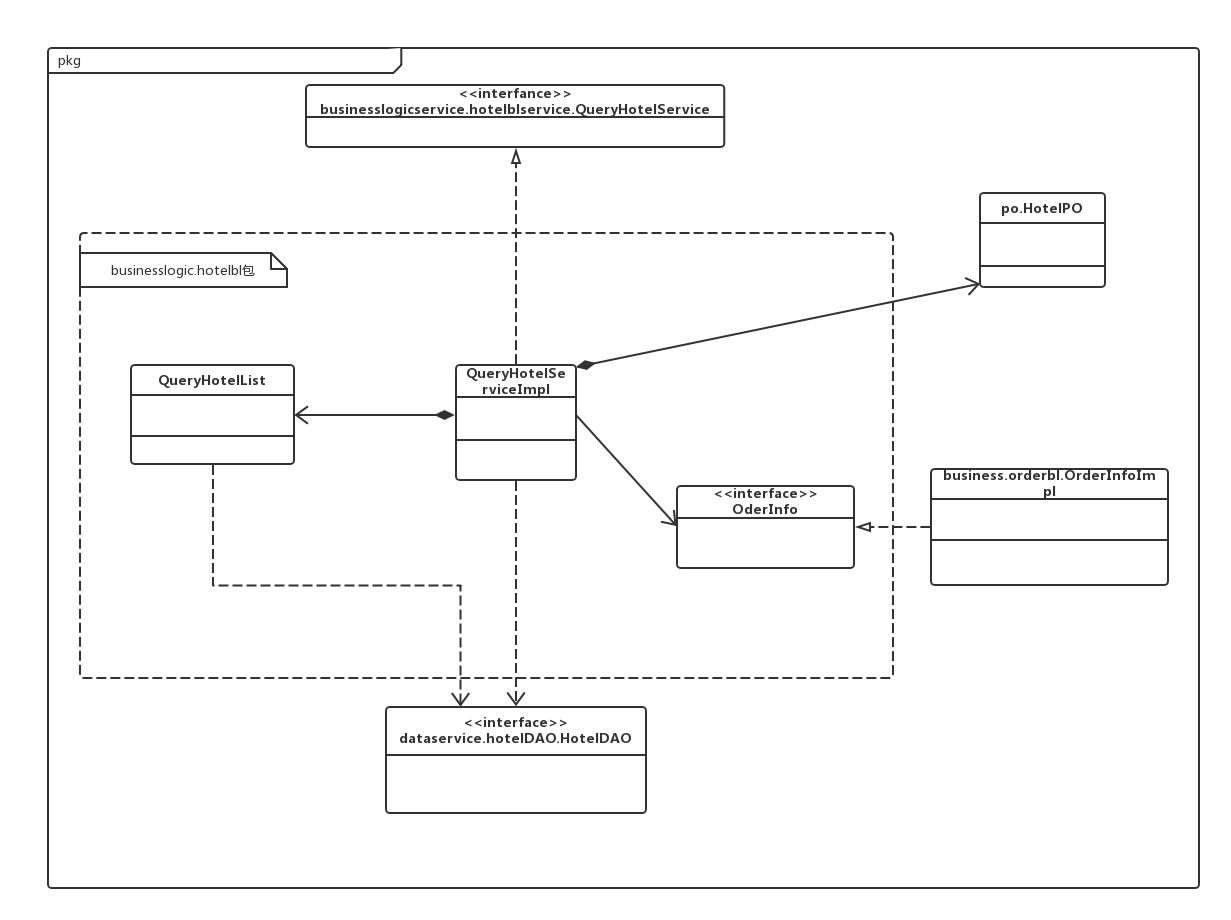
hotelbl模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| **类名** | **职责** |
| checkOrderedHotel. CheckOrderedHotelServiceImpl | 负责实现对于浏览预定过的酒店界面所需要的服务 |
| checkOrderedHotel. OrderedHotelList | 负责保存预定过的酒店列表 |
| checkOrderedHotel. OrderedHotelItem | 负责获得相应酒店的简要信息负责 |
| commentOnHotel. CommentOnHotelServiceImpl | 负责实现对于评价酒店界面所需要的服务 |
| commentOnHotel. CommentableOrderList | 负责获取可以评价的订单列表 |
| importNewRoom. ImportNewRoomServiceImpl | 负责实现对于录入可用客房界面所需要的服务 |
| importNewRoom. AvailableRoomList | 负责获取可用客房列表 |
| importNewRoom. AvailableRoomItem | 负责更新可用客房 |
| maintainHotelBasicInfo. MaintainHotelBasicInfoServiceImpl | 负责实现对于维护酒店基本信息界面所需要的服务 |
| manageHotelInfo. ManageHotelInfoServiceImpl | 负责实现对于管理酒店信息界面所需要的服务 |
| queryHotel. QueryHotelList | 负责获得有序的酒店列表 |
| queryHotel. QueryHotelServiceImpl | 负责实现对于浏览酒店信息详细所需要的服务 |
| searchHotel. SearchHotelServiceImpl | 负责实现对于搜索酒店界面所需要的服务 |

hoetlbl模块各个类的职责

**4.1.3.1 QueryHotel**

QueryHotel的设计如下图所示。



**接口规范：**

QueryHotelServiceImpl和QueryHotelList的接口规范如下表所示。

QueryHotelServiceImpl的接口规范

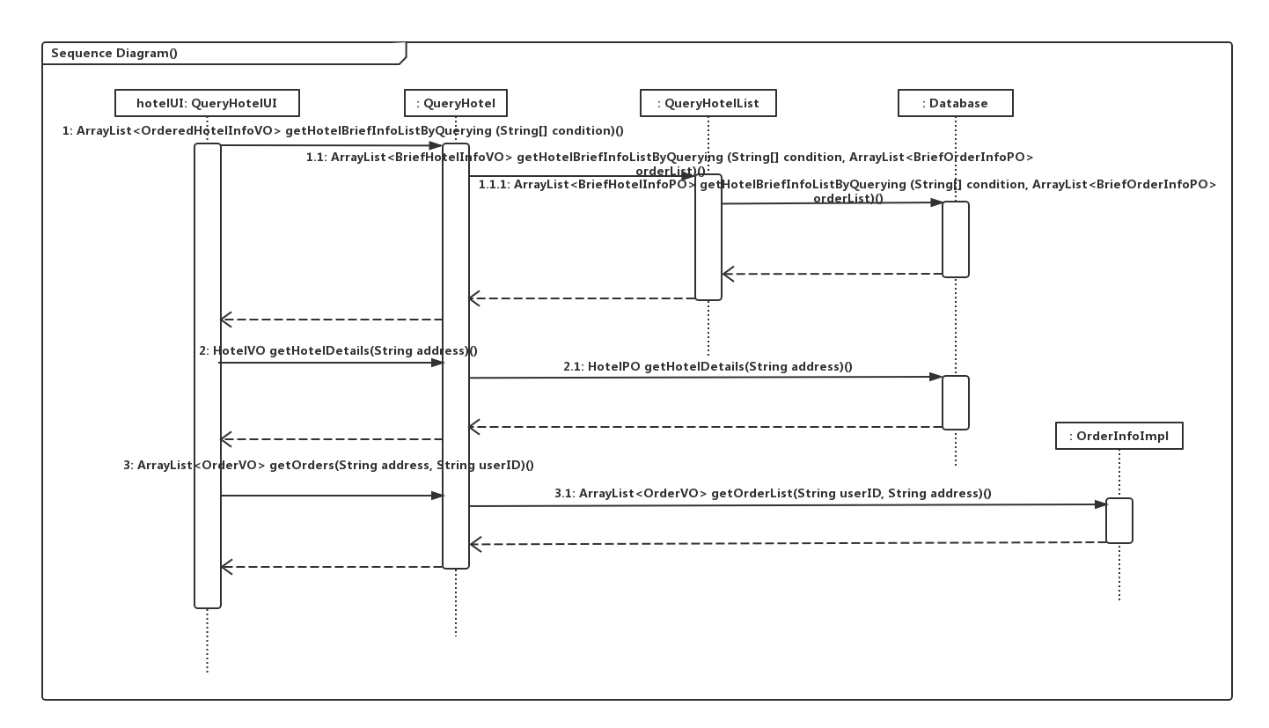
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| QueryHotelServiceImpl.getHotelBriefInfoListByQuerying | 语法 | public ArrayList<OrderedHotelInfoVO> getHotelBriefInfoListByQuerying(String[] condition) throws RemoteException |
| 前置条件 | 已创建一个QueryHotelList对象，并且condition符合输入规则 |
| 后置条件 | 调用QueryHotelList对象的getHotelBriefInfoListByQuerying方法 |
| QueryHotelServiceImpl. getHotelDetails | 语法 | public HotelVO getHotelDetails(String address) throws RemoteException |
| 前置条件 | 已创建一个HotelDAO对象 |
| 后置条件 | 调用HotelDAO对象的getHotelDetails方法 |
| QueryHotelServiceImpl.getOrders | 语法 | public ArrayList<OrderVO> getOrders(String address, String userID) throws RemoteException |
| 前置条件 | 已创建一个OrderInfo对象 |
| 后置条件 | 调用OrderInfo对象的getOrderList方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderInfo.getOrderList(String userID, String address) | 得到某用户在某酒店所有订单的列表 | |
| OrderInfo.getReservedOrderList(String userID) | 得到某用户的所有已执行、已撤销和异常订单 | |
| QueryHotelList.getHotelBriefInfoListByQuerying(String[] condition, ArrayList<BriefOrderInfoPO> orderedHotelList) | 获得符合条件的酒店列表 | |
| HotelDAO.getHotelDetails(String address) | 获取酒店详情 | |

QueryHotelList的接口规范

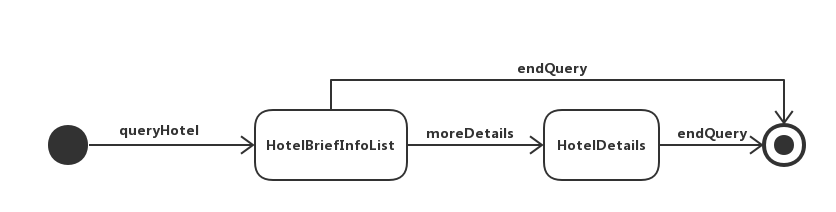
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| QueryHotelList. getHotelBriefInfoListByQuerying | 语法 | public ArrayList<BriefHotelInfoPO> getHotelBriefInfoListByQuerying(String[] condition, ArrayList<BriefOrderInfoPO> orderedHotelList) throws RemoteException |
| 前置条件 | 已创建一个HotelDAO对象 |
| 后置条件 | 返回有序的酒店列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.getHotelBriefInfoListByQuerying(String[] condition, ArrayList<BriefOrderInfoPO> orderedHotelList) | 通过查看获取酒店基本信息列表 | |

**业务逻辑层的动态模型**

QueryHotel的顺序图如下图所示。

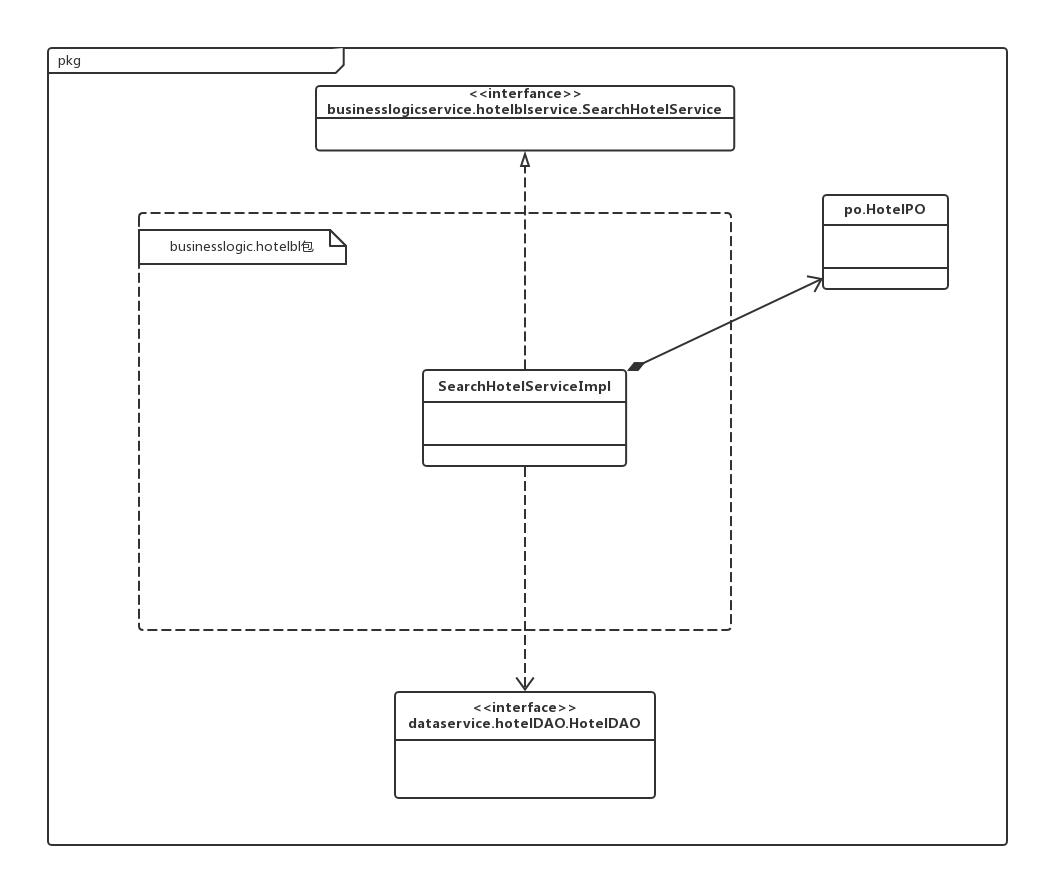
****

QueryHotel的状态图如下图所示。

****

**4.1.3.2 SearchHotel**

SearchHotel的设计如下图所示。

**接口规范：**

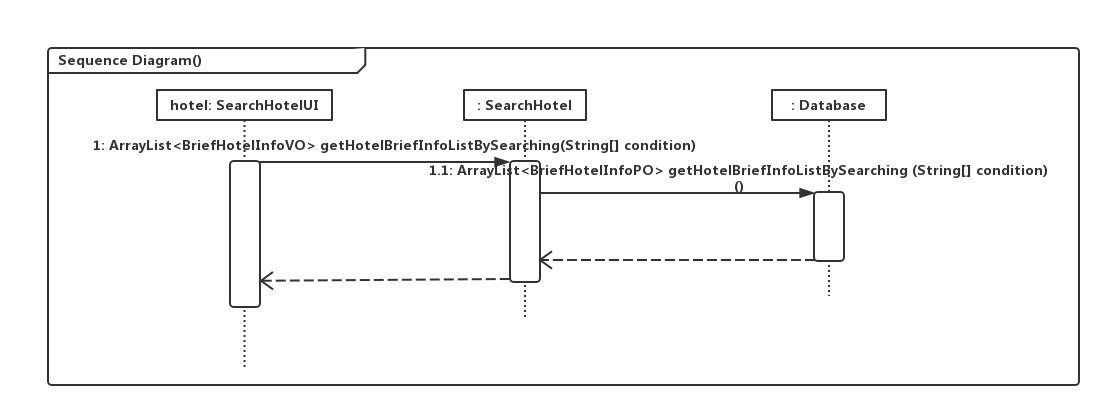
SearchHotelServiceImpl的接口规范如下表所示。

SearchHotelServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SearchHotelServiceImpl.getHotelBriefInfoListBySearching | 语法 | public ArrayList<OrderedHotelInfoVO> getHotelBriefInfoListByQuerying(String[] condition) throws RemoteException |
| 前置条件 | 拥有HotelDAO对象 |
| 后置条件 | 返回符合搜索条件的酒店列表 |
| SearchHotelServiceImpl.getBusinessDistrictList | 语法 | public ArrayList<BusinessDistrictPO> getBusinessDistrictList(String city) throws RemoteException |
| 前置条件 | city不为空 |
| 后置条件 | 返回该city所有的商圈 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.getHotelBriefInfoListBySearching(String[] condition) | 搜索符合输入条件的酒店 | |

**业务逻辑层的动态模型**

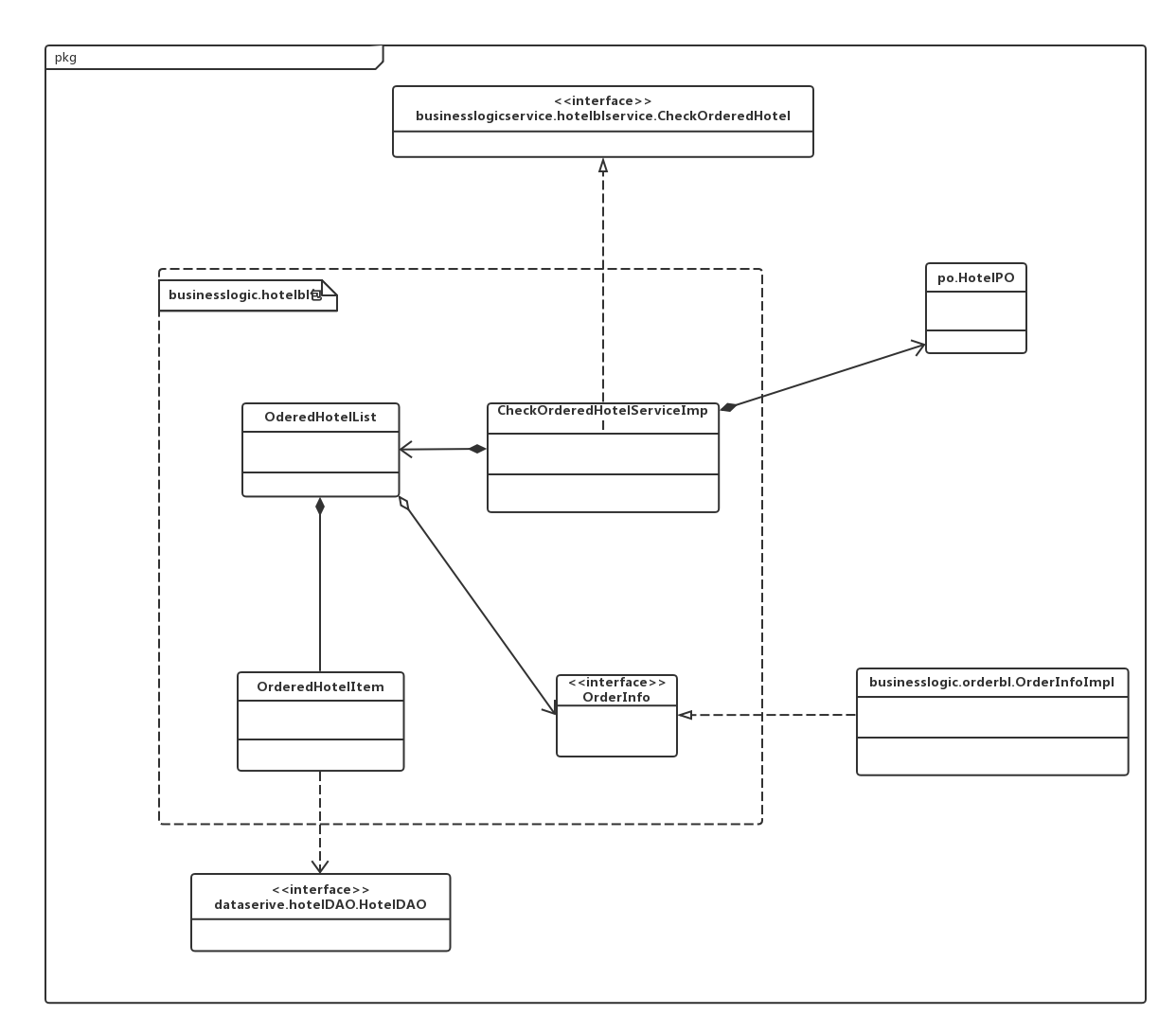
SearchHotel的顺序图如下图所示。

**** SearchHotel的状态图如下图所示。

****

**4.1.3.3 CheckOrderedHotel**

CheckOrderedHotel的设计如下图所示。

**接口规范：**

CheckOrderedHotelServiceImpl、OrderedHotelList、OrderedHotelItem的接口规范如下表所示。

CheckOrderedHotelServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckOrderedHotelServiceImpl.enrollHotelBreifInfoList | 语法 | public ArrayList<OrderedHotelInfoVO> enrollHotelBreifInfoList(String userID) throws RemoteException |
| 前置条件 | 拥有OrderedHotelList对象 |
| 后置条件 | 使用Ordered Hotel List对象的enrollHotelBreifInfoList方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderedHotelList.enrollHotelBreifInfoList() | 获得预定过的酒店的简要信息列表 | |

OrderedHotelList的接口规范

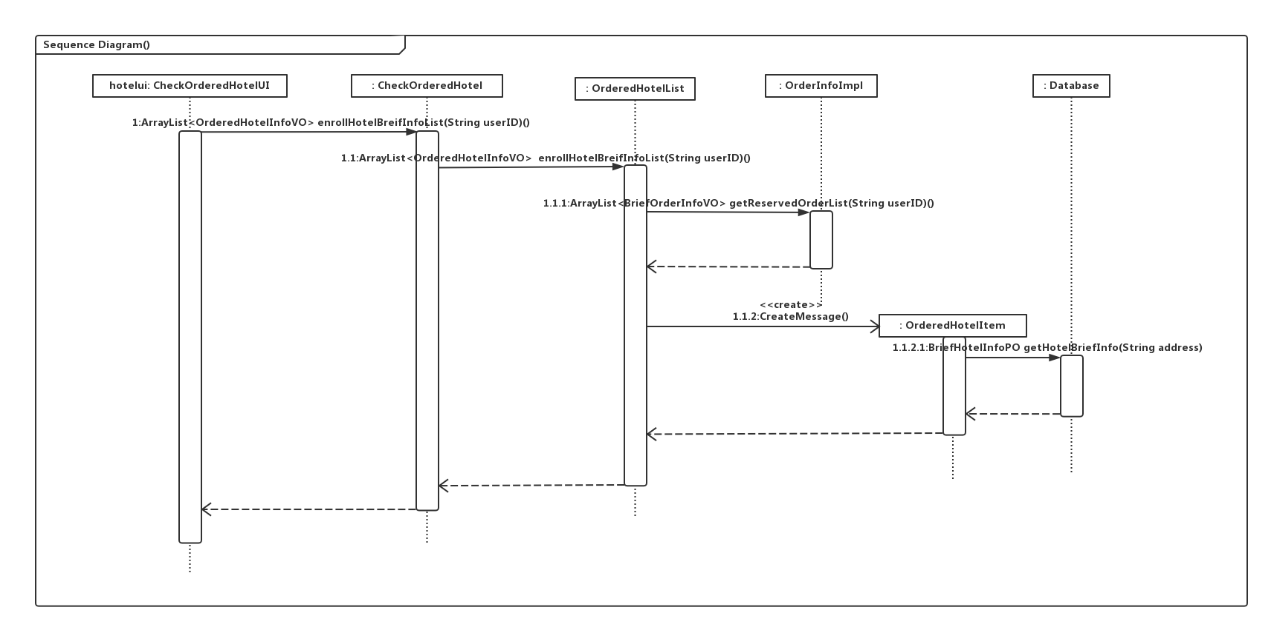
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderedHotelList. enrollHotelBreifInfoList | 语法 | public ArrayList<OrderedHotelInfoVO> enrollHotelBreifInfoList() throws RemoteException |
| 前置条件 | 拥有Order Info和OrderedHotelItem对象 |
| 后置条件 | 使用Order Info对象的getReservedOrderList方法和OrderedHotelItem对象的getbriefHotelInfo方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderInfo.getReservedOrderList(String userID) | 得到某用户的所有已执行、已撤销和异常订单 | |
| OrderedHotelItem.getbriefHotelInfo() | 得到相应酒店的信息 | |

OrderedHotelItem的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderedHotelItem. getbriefHotelInfo | 语法 | public BriefHotelInfoVO getBriefHotelInfo() throws RemoteException |
| 前置条件 | 已得知酒店地址 |
| 后置条件 | 返回相应酒店的简要信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.getHotelBriefInfo(String address) | 获取酒店的基本信息 | |

**业务逻辑层的动态模型**

CheckOrderedHotel的顺序图如下图所示。

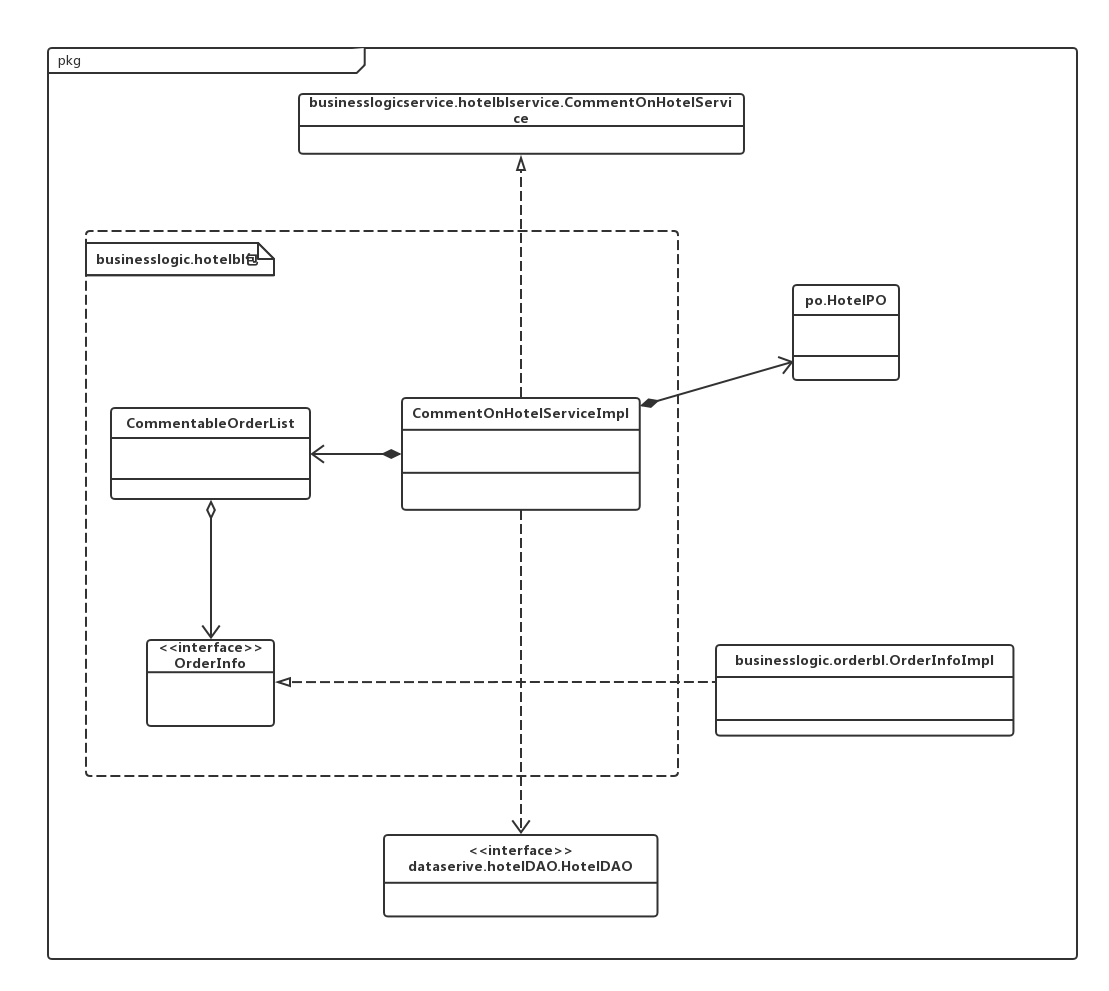
****

CheckOrderedHotel的状态图如下图所示。

****

**4.1.3.4 CommentOnHotel**

CommentOnHotel的设计如下图所示。

**接口规范：**

CommentOnHotelServiceImpl、CommentableOrderList的接口规范如下表所示。

CommentOnHotelServiceImpl的接口规范

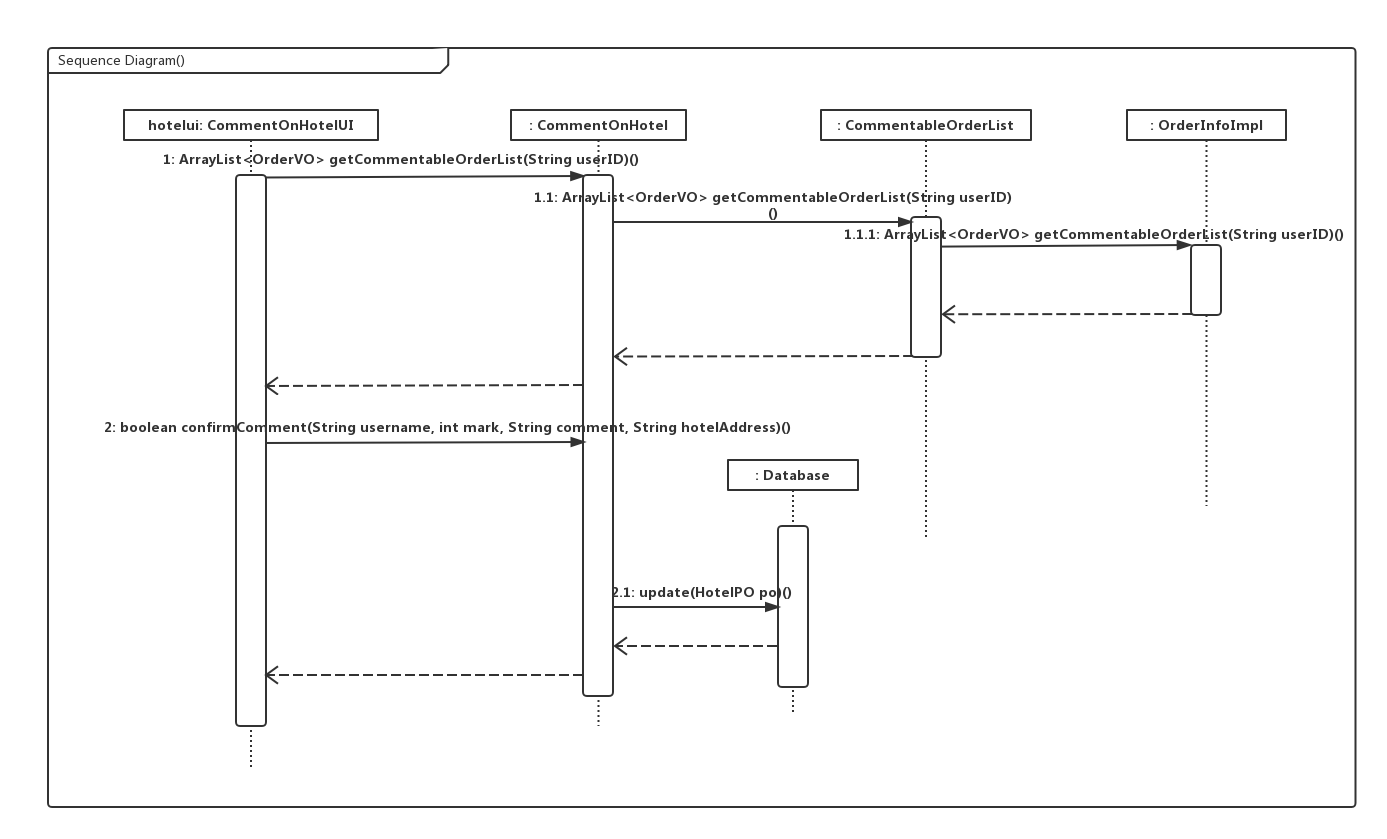
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommentOnHotelServiceImpl.getCommentableOrderList | 语法 | public ArrayList<OrderVO> getCommentableOrderList(String userID) throws RemoteException |
| 前置条件 | 已知用户名 |
| 后置条件 | 返回可评价的订单列表 |
| CommentOnHotelServiceImpl.confirmComment | 语法 | public boolean confirmComment(String username, float, mark, String comment, String hotelAddress, String orderID) throws RemoteException |
| 前置条件 | 所有输入均符合输入规则 |
| 后置条件 | 返回是否评价成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CommentableOrderList.getCommentableOrderList() | 得到所有可评价酒店的订单列表 | |
| HotelDAO.update(HotelPO po) | 更新酒店信息 | |

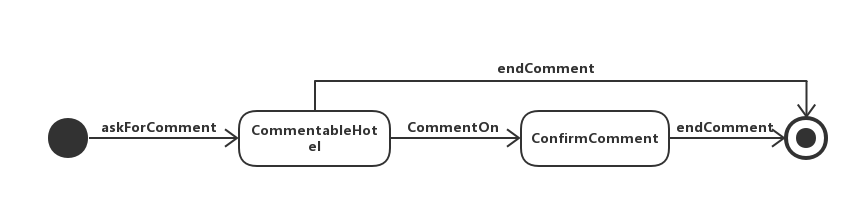
CommentableOrderList的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommentableOrderList.getCommentableOrderList | 语法 | public ArrayList<OrderVO> getCommentableOrderList() throws RemoteException |
| 前置条件 | 拥有OrderInfo对象和用户名 |
| 后置条件 | 返回可评价订单列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderInfo.getCommentableOrderList(String userID) | 得到某用户所有可评价订单的列表 | |

**业务逻辑层的动态模型**

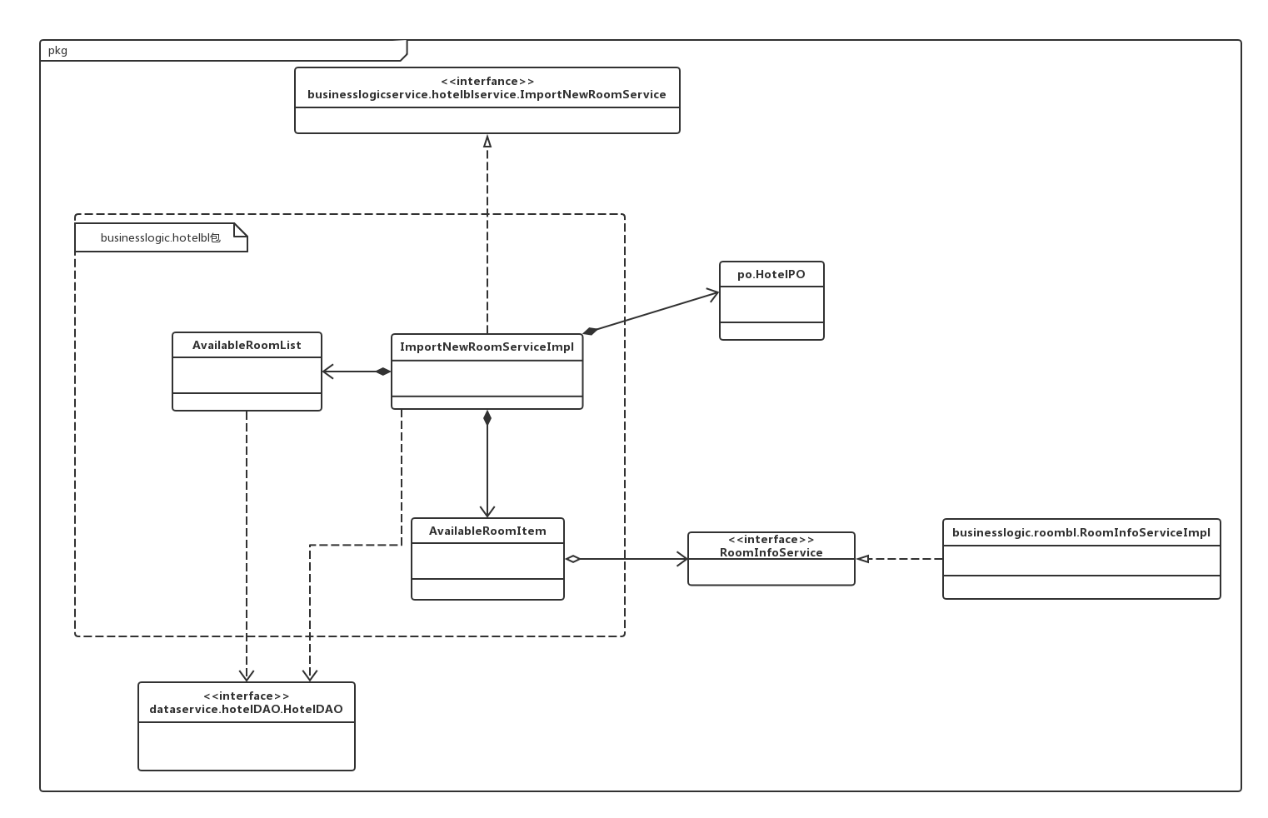
CommentOnHotel的顺序图如下图所示。

**** CommentOnHotel的状态图如下图所示。

****

**4.1.3.5 ImportNewRoom**

ImportNewRoom的设计如下图所示。

**接口规范：**

ImportNewRoomServiceImpl、AvailableRoomList、AvailableRoomItem的接口规范如下表所示。

ImportNewRoomServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ImportNewRoomServiceImpl.getAvailableRoomList | 语法 | public ArrayList<RoomVO> getAvailableRoomList(String address) throws RemoteException |
| 前置条件 | 拥有AvailableRoomList对象 |
| 后置条件 | 使用AvailableRoomList对象的getAvailableRoomList方法 |
| ImportNewRoomServiceImpl. addRoom | 语法 | public boolean addRoom(RoomVO room) throws RemoteException |
| 前置条件 | 拥有AvailableRoomItem对象 |
| 后置条件 | 使用AvailableRoomItem对象的addRoom方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AvailableRoomList.getAvailableRoomList() | 得到可用客房列表 | |
| AvailableRoomItem.addRoom(RoomVO room) | 增加可用客房 | |

AvailableRoomList的接口规范

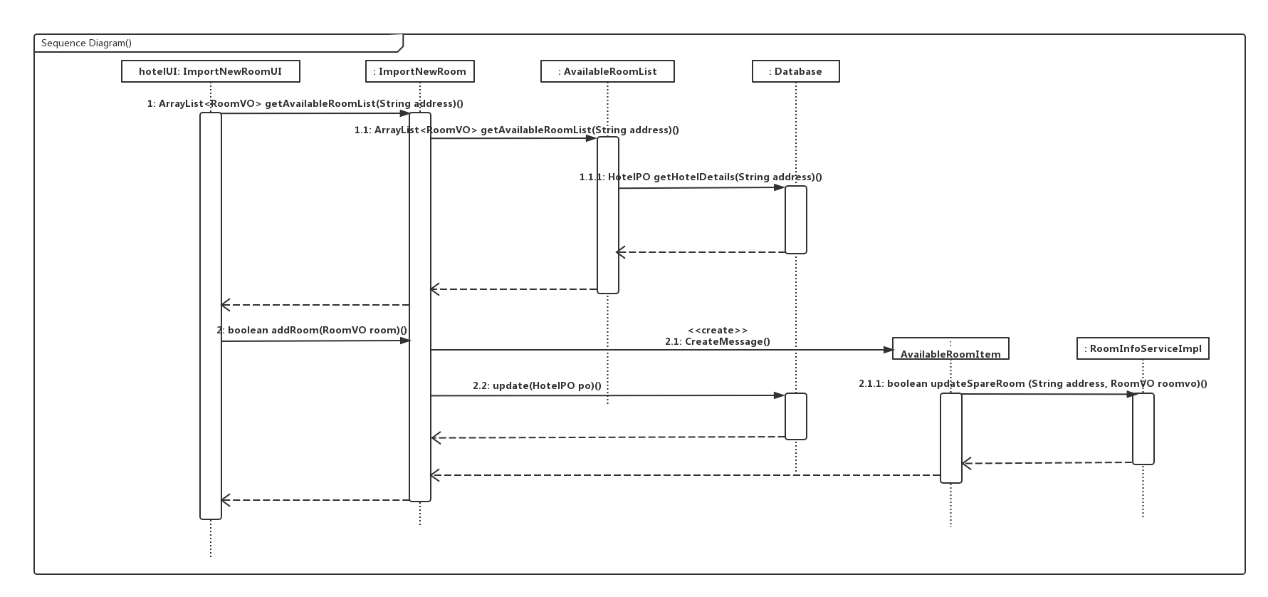
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AvailableRoomList. getAvailableRoomList | 语法 | public ArrayList<RoomVO> getAvailableRoomList() throws RemoteException |
| 前置条件 | 已获得酒店地址 |
| 后置条件 | 返回可用客房列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.getHotelDetails(String address) | 获取酒店详情 | |

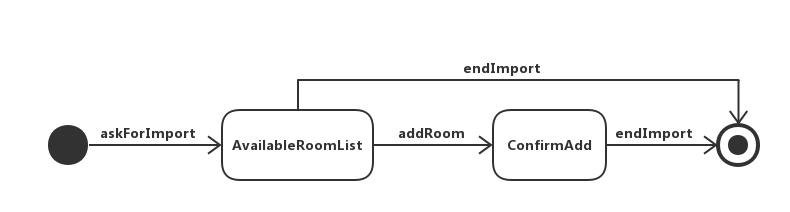
AvailableRoomItem的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AvailableRoomItem. addRoom | 语法 | public boolean addRoom(RoomVO room) throws RemoteException |
| 前置条件 | 已获得酒店地址 |
| 后置条件 | 返回是否录入成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| RoomInfoService.updateSpareRoom(String address, RoomVO roomvo) | 在更新可用客房后更新空房信息 | |

**业务逻辑层的动态模型**

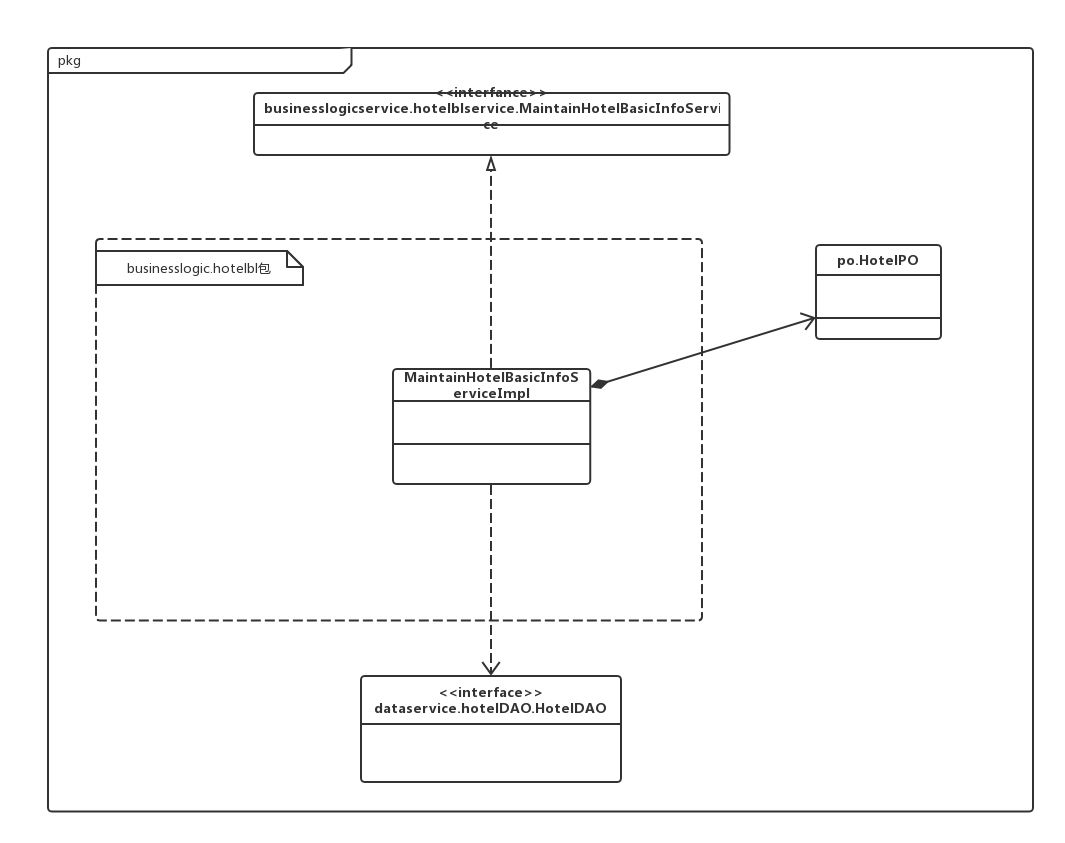
ImportNewRoom的顺序图如下图所示。

****ImportNewRoom的状态图如下所示。

****

**4.1.3.6 MaintainHotelBasicInfo**

MaintainHotelBasicInfo的设计如下图所示。

**接口规范：**

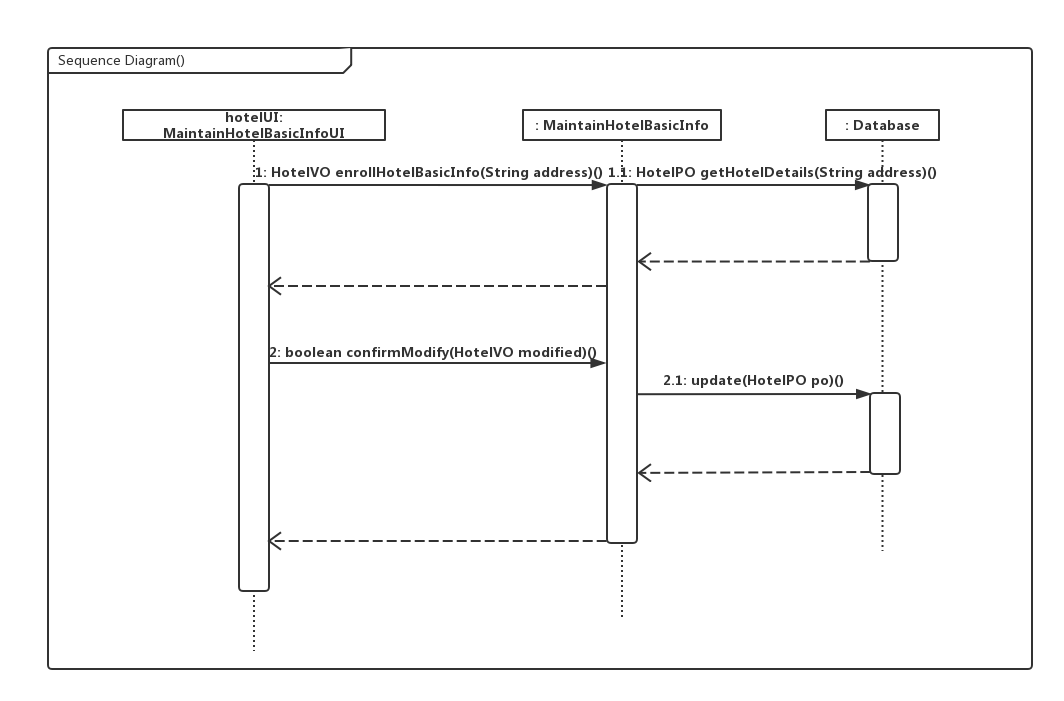
MaintainHotelBasicInfoServiceImpl的接口规范如下表所示。

MaintainHotelBasicInfoServiceImpl的接口规范

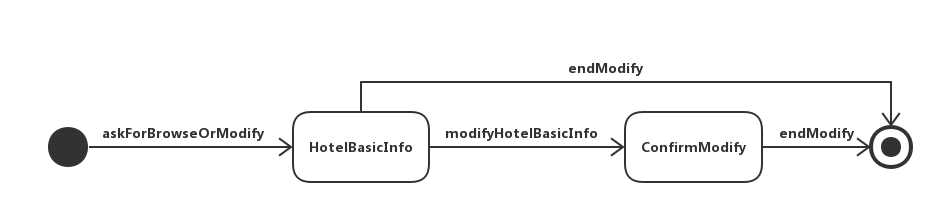
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MaintainHotelBasicInfoServiceImpl.enrollHotelBasicInfo | 语法 | public HotelVO enrollHotelBasicInfo(String address) throws RemoteException |
| 前置条件 | 已获得酒店地址 |
| 后置条件 | 返回酒店基本信息 |
| MaintainHotelBasicInfoServiceImpl. confirmModify | 语法 | public boolean confirmModify(HotelVO modified) throws RemoteException |
| 前置条件 | 已获得酒店地址以及修改后的酒店信息 |
| 后置条件 | 返回是否修改成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.getHotelDetails(String address) | 获取酒店详情 | |
| HotelDAO.update(HotelPO po) | 更新酒店信息 | |

**业务逻辑层的动态模型**

MaintainHotelBasicInfo的顺序图如下图所示。

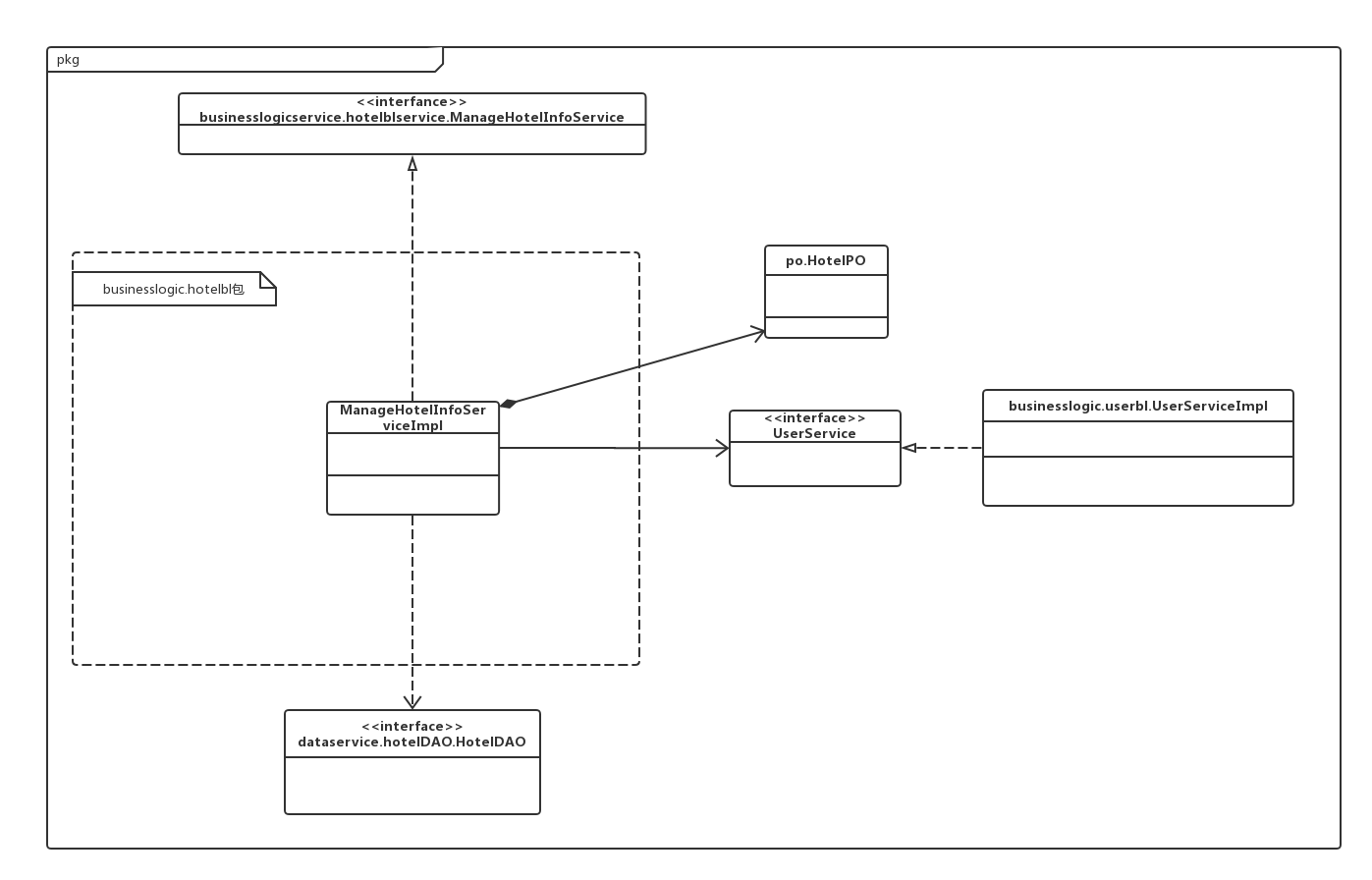
****

MaintainHotelBasicInfo的状态图如下图所示。



**4.1.3.7 ManageHotelInfo**

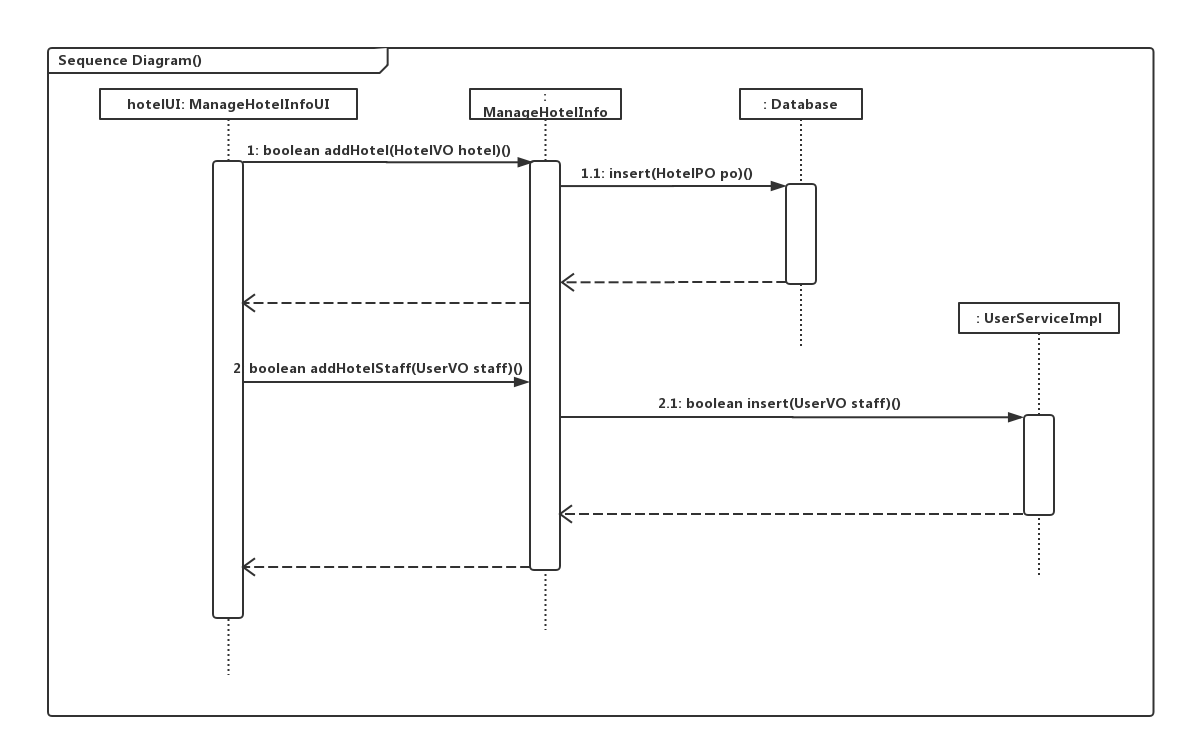
ManageHotelInfo的设计如下图所示。

**接口规范：**

ManageHotelInfoServiceImpl的接口规范如下表所示。

ManageHotelInfoServiceImpl的接口规范

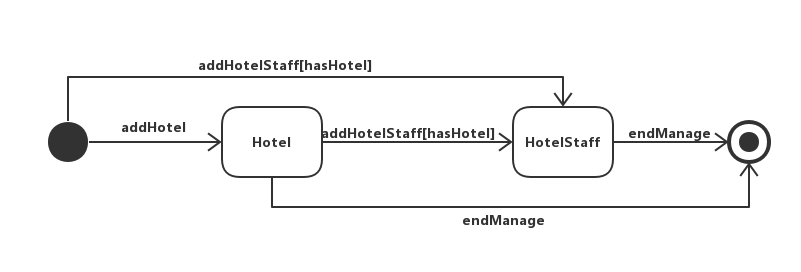
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageHotelInfoServiceImpl.addHotel | 语法 | public boolean addHotel(HotelVO hotel) throws RemoteException |
| 前置条件 | 已获得新增的酒店的信息 |
| 后置条件 | 返回是否添加成功 |
| ManageHotelInfoServiceImpl.addHotelStaff | 语法 | public boolean addHotelStaff(UserVO staff) throws RemoteException |
| 前置条件 | 拥有UserInfo对象 |
| 后置条件 | 返回是否添加成功 |
| ManageHotelInfoServiceImpl.getHotelInfo | 语法 | public HotelVO getHotelInfo(String hotelAddress) throws RemoteException |
| 前置条件 | 酒店地址不为空 |
| 后置条件 | 返回对应的酒店的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDAO.insert(HotelPO po) | 增加新的酒店 | |
| UserInfo.insert(UserVO staff) | 酒店管理人员添加酒店工作人员 | |
| HotelDAO.getHotelDetails(String address) | 获得酒店信息 | |

****

**业务逻辑层的动态模型**

ManageHotelInfoServiceImpl的顺序图如下图所示。

ManageHotelInfoServiceImpl的状态图如下图所示。



**4.1.4** **roombl模块**

1. **模块概述**

roombl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。

roombl模块的职责及接口参见软件体系结构文档。

1. **整体结构**

展示层和业务逻辑层之间，我们添加businesslogicservice.roomblservice.BrowseSpareRoomService接口、businesslogicservice.roomblservice.UpdateCheckInService接口、businesslogicservice.roomblservice. UpdateCheckOutService接口。业务逻辑层和数据层之间添加dataservice.roomDAO.RoomDAO接口。由于本模块涉及到的逻辑控制职责较为简单，将其与业务逻辑职责合并形成BroseSpareRoomServiceImpl, UpdateCheckInServiceImpl, UpdateCheckOutServiceImpl类。RoomPO、CheckInOutPO是作为房间的持久化对象被添加到涉及模型中去的。SpareRoomList和SpareRoomItem的添加是SpareRoom的容器类、CheckInList和CheckInItem的添加是CheckIn的容器类、CheckOutList和CheckOutItem的添加是CheckOut的容器类。RoomInfoService都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

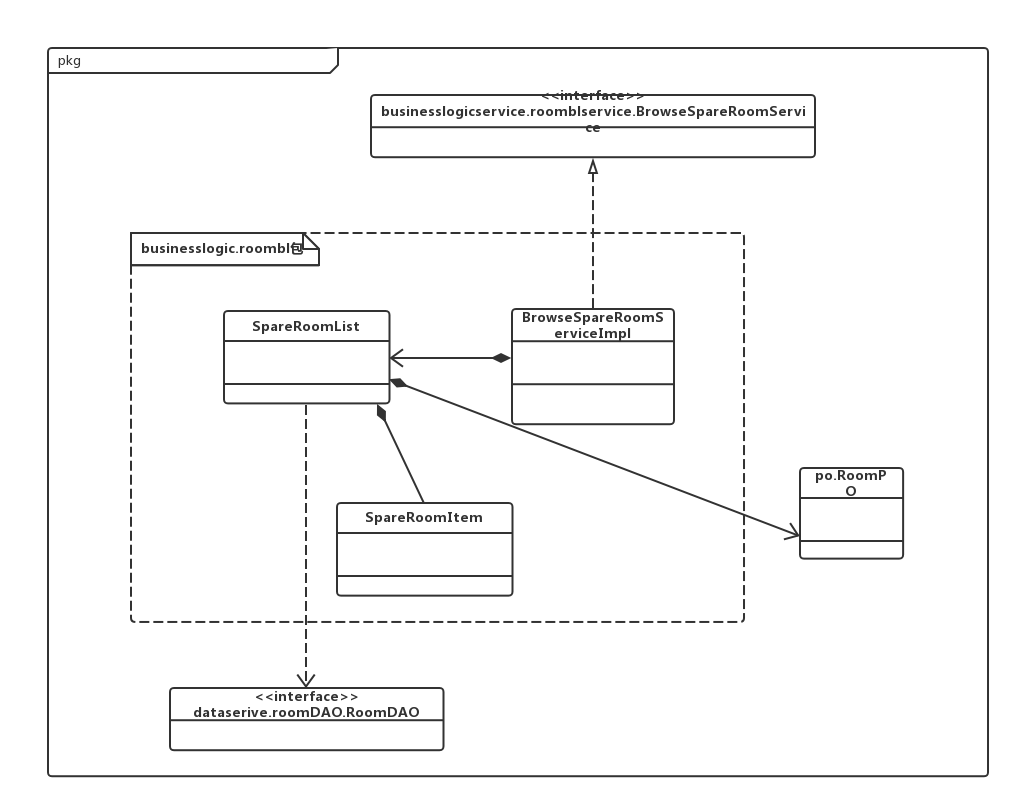
roombl模块各个类的职责如下表所示。

roombl模块各个类的职责

|  |  |
| --- | --- |
| **类名** | **职责** |
| browseSpareRoom.BrowseSpareRoomServiceImpl | 负责实现对应于浏览空房所需要的服务 |
| updateCheckIn.UpdateCheckInServiceImpl | 负责实现对应于更新入住信息所需要实现的服务 |
| updateCheckOut.UpdateCheckOutServiceImpl | 负责实现对应于更新退房信息所需要实现的服务 |
| browseSpareRoom.SpareRoomList | 空房列表对象，拥有空房对象引用，可以提供空房对象的列表 |
| browseSpareRoom.SpareRoomItem | 空房对象，拥有空房类型，价格，数量，可以提供空房信息 |
| updateCheckIn.CheckInList | 入住信息列表对象，拥有入住信息对象引用，可以提供入住信息列表，提供搜索，增加，修改，删除，验证入住信息的服务 |
| updateCheckIn.CheckInItem | 入住信息对象，拥有房间类型，数量，入住时间和预计离开时间，帮助完成增加，修改，删除，验证入住信息的服务 |
| updateCheckOut.CheckOutList | 退房信息列表对象，拥有退房信息对象引用，可以提供退房信息列表，提供搜索，增加，修改，删除，验证退房信息的服务 |
| updateCheckOut.CheckOutItem | 退房信息对象，拥有房间类型，数量，实际离开时间，帮助完成增加，修改，删除，验证退房信息的服务 |
| RoomInfoServiceImpl | 给同层的其他模块提供有关空房的服务 |

**4.1.4.1 BrowseSpareRoom**

browseSpareRoom模块的设计如下图所示。



**接口规范：**

BrowseSpareRoomServiceImpl, SpareRoomList和SpareRoomItem的接口规范如下表所示。

BrowseSpareRoomServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BrowseSpareRoomServiceImpl.getRoomInfoList | 语法 | public ArrayList<RoomVO> getRoomInfoList (String address) |
| 前置条件 | 已创建一个SpareRoomList对象 |
| 后置条件 | 调用SpareRoomList对象的getRoomInfoList方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| SpareRoomList.getRoomInfoList(String address) | 提供空房列表 | |
| SpareRoomItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |

SpareRoomList的接口规范

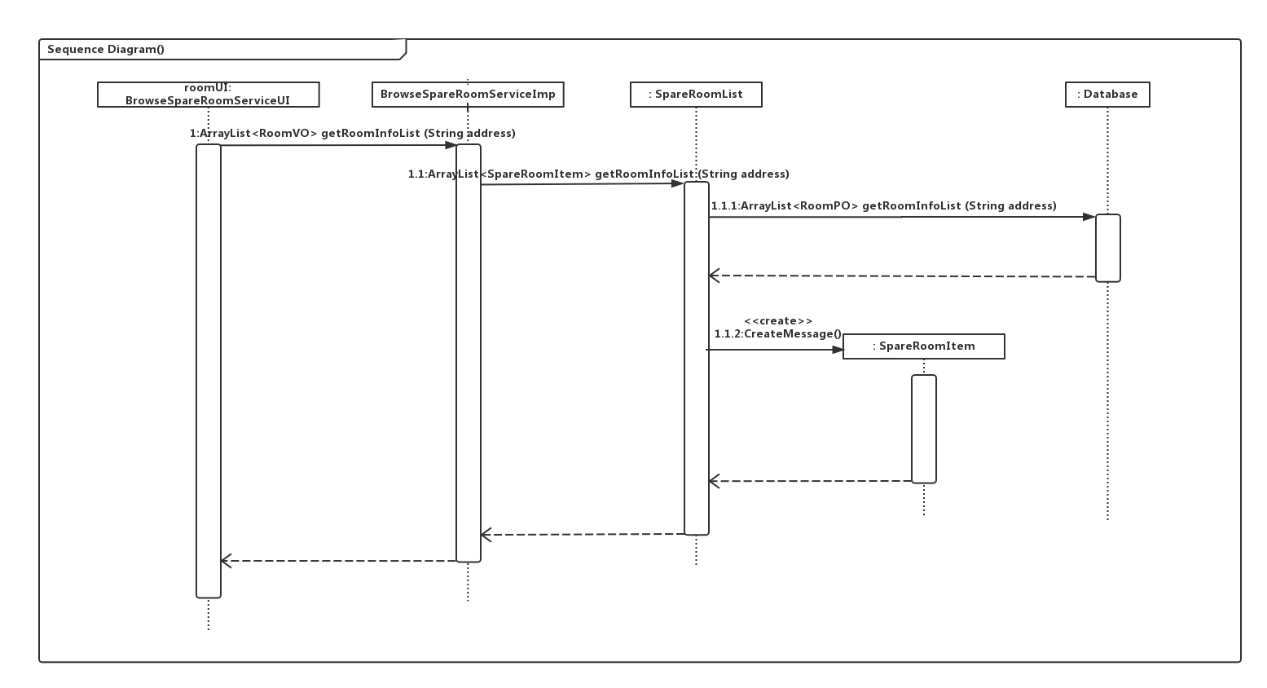
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SpareRoomList.getRoomInfoList | 语法 | public ArrayList<SpareRoomItem> getRoomInfoList (String address) |
| 前置条件 | 启动一个得到空房列表回合 |
| 后置条件 | 在一个得到空房列表回合中，返回空房列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| RoomDAO. getSpareRoomInfoList(String address) | 得到空房列表持久化对象 | |

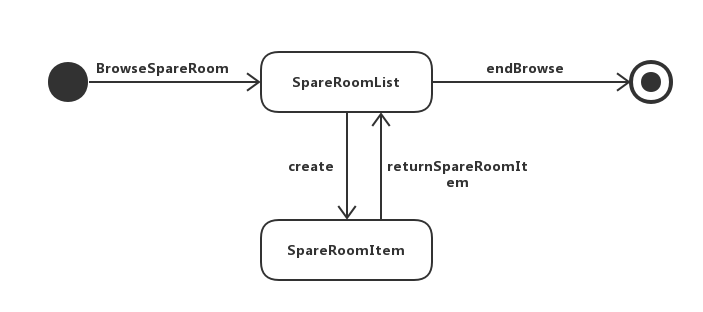
SpareRoomItem的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SpareRoomItem.toVO | 语法 | public RoomVO toVO() |
| 前置条件 | 无 |
| 后置条件 | 返回空房RoomVO |

**业务逻辑层的动态模型**

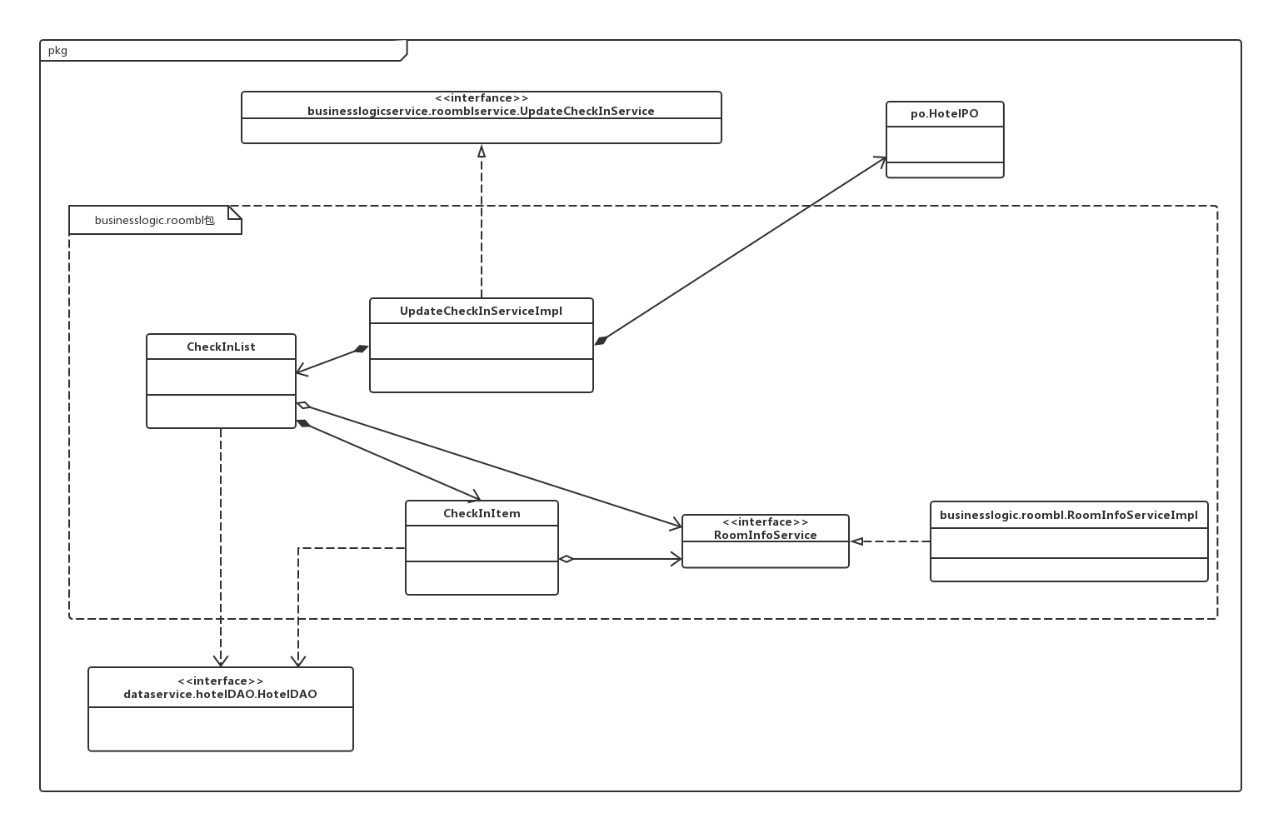
BrowseSpareRoom的顺序图如下图所示。

 BrowseSpareRoom的状态图如下图所示。



**4.1.4.2 UpdateCheckIn**

UpdateCheckIn模块的设计如下图所示。



**接口规范：**

UpdateCheckInServiceImpl, CheckInList和CheckInItem的接口规范如下表所示。

UpdateCheckInServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UpdateCheckInServiceImpl. getCheckInList | 语法 | public ArrayList<RoomVO> getCheckInList(String address) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的getRoomInfoList方法 |
| UpdateCheckInServiceImpl. searchCheckInInfo | 语法 | public ArrayList<RoomVO> searchCheckInInfo(String address ,Date time) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的searchCheckInInfo方法 |
| UpdateCheckInServiceImpl. searchCheckInInfo | 语法 | public ArrayList<RoomVO> searchCheckInInfo(String address ,Enum<RoomType> roomType) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的searchCheckInInfo方法 |
| UpdateCheckInServiceImpl. addCheckIn | 语法 | public boolean addCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的addCheckIn方法 |
| UpdateCheckInServiceImpl. modifyCheckIn | 语法 | public boolean modifyCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的modifyCheckIn方法 |
| UpdateCheckInServiceImpl.delCheckIn | 语法 | public boolean delCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的delCheckIn方法 |
| UpdateCheckInServiceImpl.validCheckIn | 语法 | public boolean validCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInList对象 |
| 后置条件 | 调用CheckInList对象的validCheckIn方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CheckInList.getCheckInList( String address) | 提供入住信息列表 | |
| CheckInItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |
| CheckInList. searchCheckInInfo | 提供根据入住时间搜索得到的入住信息列表 | |
| CheckInList. searchCheckInInfo | 提供根据房间类型搜索得到的入住信息列表 | |
| CheckInList. addCheckIn | 提供增加入住信息的服务 | |
| CheckInList. modifyCheckIn | 提供修改入住信息的服务 | |
| CheckInList. delCheckIn | 提供删除入住信息的服务 | |
| CheckInList. validCheckIn | 提供验证入住信息的服务 | |

CheckInList的接口规范

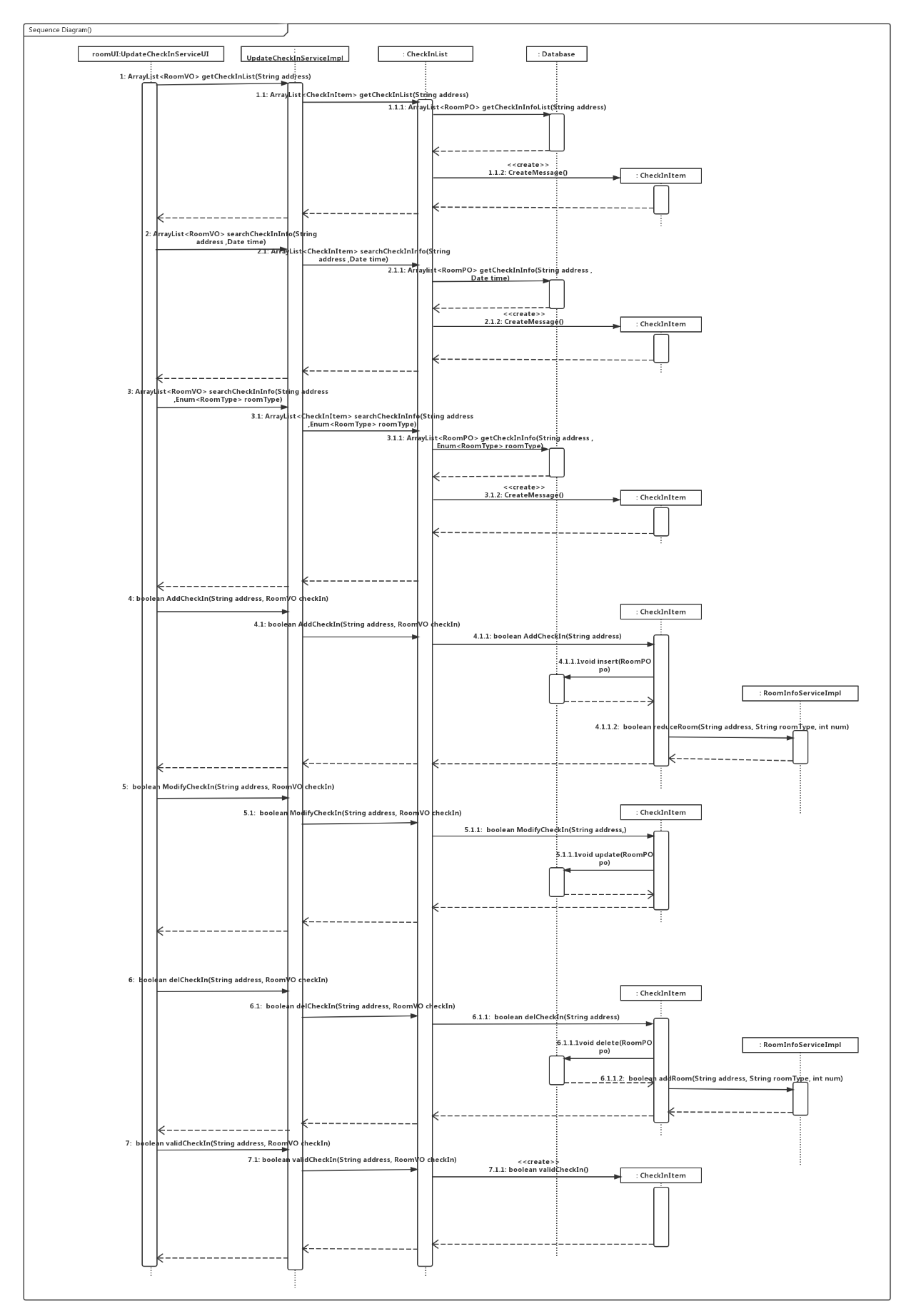
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckInList.getCheckInList | 语法 | public ArrayList<CheckInItem> getCheckInList(String address) |
| 前置条件 | 启动一个得到入住信息列表回合 |
| 后置条件 | 在一个得到入住信息列表回合中，返回入住信息列表 |
| CheckInList. searchCheckInInfo | 语法 | public ArrayList<CheckInItem> searchCheckInInfo(String address ,Date time) |
| 前置条件 | 启动一个搜索入住信息回合 |
| 后置条件 | 在一个搜索入住信息回合中，返回搜索到的入住信息列表 |
| CheckInList. searchCheckInInfo | 语法 | public ArrayList<CheckInItem> searchCheckInInfo(String address ,Enum<RoomType> roomType) |
| 前置条件 | 启动一个搜索入住信息回合 |
| 后置条件 | 在一个搜索入住信息回合中，返回搜索到的入住信息列表 |
| CheckInList. addCheckIn | 语法 | public boolean addCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInItem对象 |
| 后置条件 | 调用CheckInItem对象的addCheckIn方法 |
| CheckInList. modifyCheckIn | 语法 | public boolean modifyCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInItem对象 |
| 后置条件 | 调用CheckInItem对象的modifyCheckIn方法 |
| CheckInList. delCheckIn | 语法 | public boolean delCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInItem对象 |
| 后置条件 | 调用CheckInItem对象的delCheckIn方法 |
| CheckInList. validCheckIn | 语法 | public boolean validCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 已创建一个CheckInItem对象 |
| 后置条件 | 调用CheckInItem对象的validCheckIn方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CheckInItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |
| RoomDAO. getCheckInInfo(String address , Date time) | 得到对应入住时间的入住信息列表 | |
| RoomDAO. getCheckInInfo(String address ,Enum<RoomType> roomType) | 得到对应房间类型的入住信息列表 | |
| CheckInItem.addCheckIn | 提供增加入住信息的服务 | |
| CheckInItem. modifyCheckIn | 提供修改入住信息的服务 | |
| CheckInItem. delCheckIn | 提供删除入住信息的服务 | |
| CheckInItem. validCheckIn | 提供验证入住信息的服务 | |

CheckInItem的接口规范

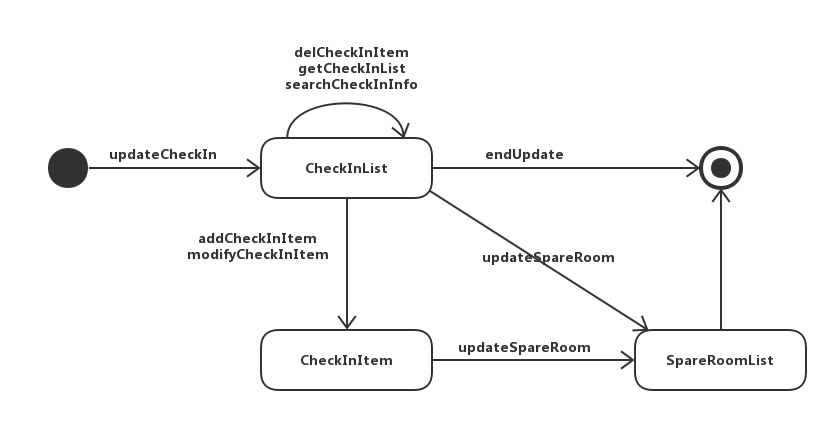
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckInItem.toVO | 语法 | public RoomVO toVO() |
| 前置条件 | 无 |
| 后置条件 | 返回入住信息CheckInOutVO |
| CheckInItem.addCheckIn | 语法 | boolean addCheckIn(String address) |
| 前置条件 | 启动一个增加入住信息回合 |
| 后置条件 | 返回是否增加成功 |
| CheckInItem. modifyCheckIn | 语法 | boolean modifyCheckIn(String address) |
| 前置条件 | 启动一个修改入住信息回合 |
| 后置条件 | 返回是否修改成功 |
| CheckInItem. delCheckIn | 语法 | boolean delCheckIn(String address) |
| 前置条件 | 启动一个删除入住信息回合 |
| 后置条件 | 返回是否删除成功 |
| CheckInItem. validCheckIn | 语法 | boolean validCheckIn(String address) |
| 前置条件 | 启动一个验证入住信息回合 |
| 后置条件 | 返回是否验证成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| RoomDAO.insert(RoomPO po) | 插入单一持久化对象 | |
| RoomDAO.update(RoomPO po) | 更新单一持久化对象 | |
| RoomDAO.delete(RoomPO po) | 删除单一持久化对象 | |

**业务逻辑层的动态模型**

UpdateCheckIn的顺序图如下图所示。

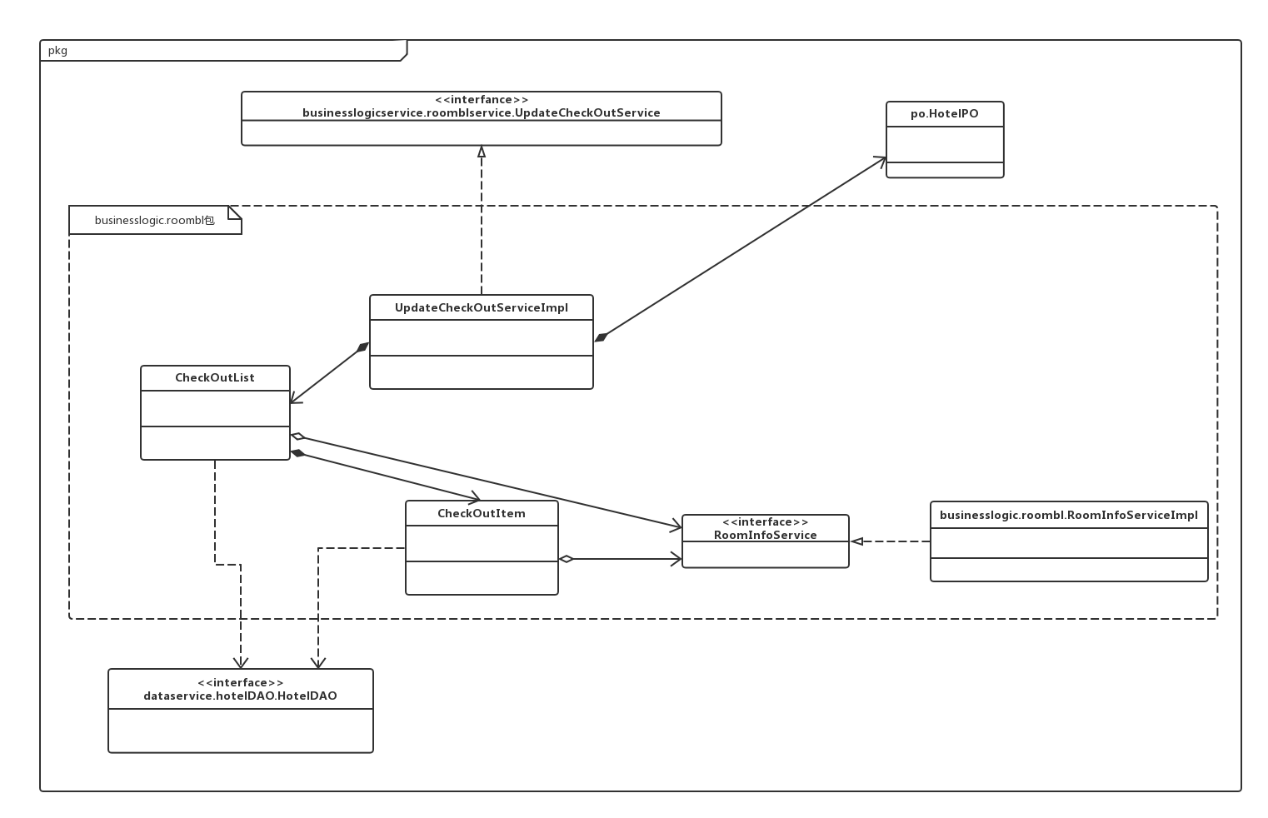


UpdateCheckIn的状态图如下图所示。



**4.1.4.3 UpdateCheckOut**

UpdateCheckOut模块的设计如下图所示。



**接口规范：**

UpdateCheckOutServiceImpl, CheckOutList和CheckOutItem的接口规范如下表所示。

UpdateCheckOutServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UpdateCheckOutServiceImpl. getCheckOutList | 语法 | public ArrayList<RoomVO> getCheckOutList(String address) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的getRoomInfoList方法 |
| UpdateCheckOutServiceImpl. searchCheckOutInfo | 语法 | public ArrayList<RoomVO> searchCheckOutInfo(String address ,Date time) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的searchCheckOutInfo方法 |
| UpdateCheckOutServiceImpl. searchCheckOutInfo | 语法 | public ArrayList<RoomVO> searchCheckOutInfo(String address ,Enum<RoomType> roomType) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的searchCheckOutInfo方法 |
| UpdateCheckOutServiceImpl. addCheckOut | 语法 | public boolean addCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的addCheckOut方法 |
| UpdateCheckOutServiceImpl. modifyCheckOut | 语法 | public boolean modifyCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的modifyCheckOut方法 |
| UpdateCheckOutServiceImpl.delCheckOut | 语法 | public boolean delCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的delCheckOut方法 |
| UpdateCheckOutServiceImpl.validCheckOut | 语法 | public boolean validCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutList对象 |
| 后置条件 | 调用CheckOutList对象的validCheckOut方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CheckOutList.getCheckOutList( String address) | 提供退房信息列表 | |
| CheckOutItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |
| CheckOutList. searchCheckOutInfo | 提供根据实际离开时间搜索得到的退房信息列表 | |
| CheckOutList. searchCheckOutInfo | 提供根据房间类型搜索得到的退房信息列表 | |
| CheckOutList. addCheckOut | 提供增加退房信息的服务 | |
| CheckOutList. modifyCheckOut | 提供修改退房信息的服务 | |
| CheckOutList. delCheckOut | 提供删除退房信息的服务 | |
| CheckOutList. validCheckOut | 提供验证退房信息的服务 | |

CheckOutList的接口规范

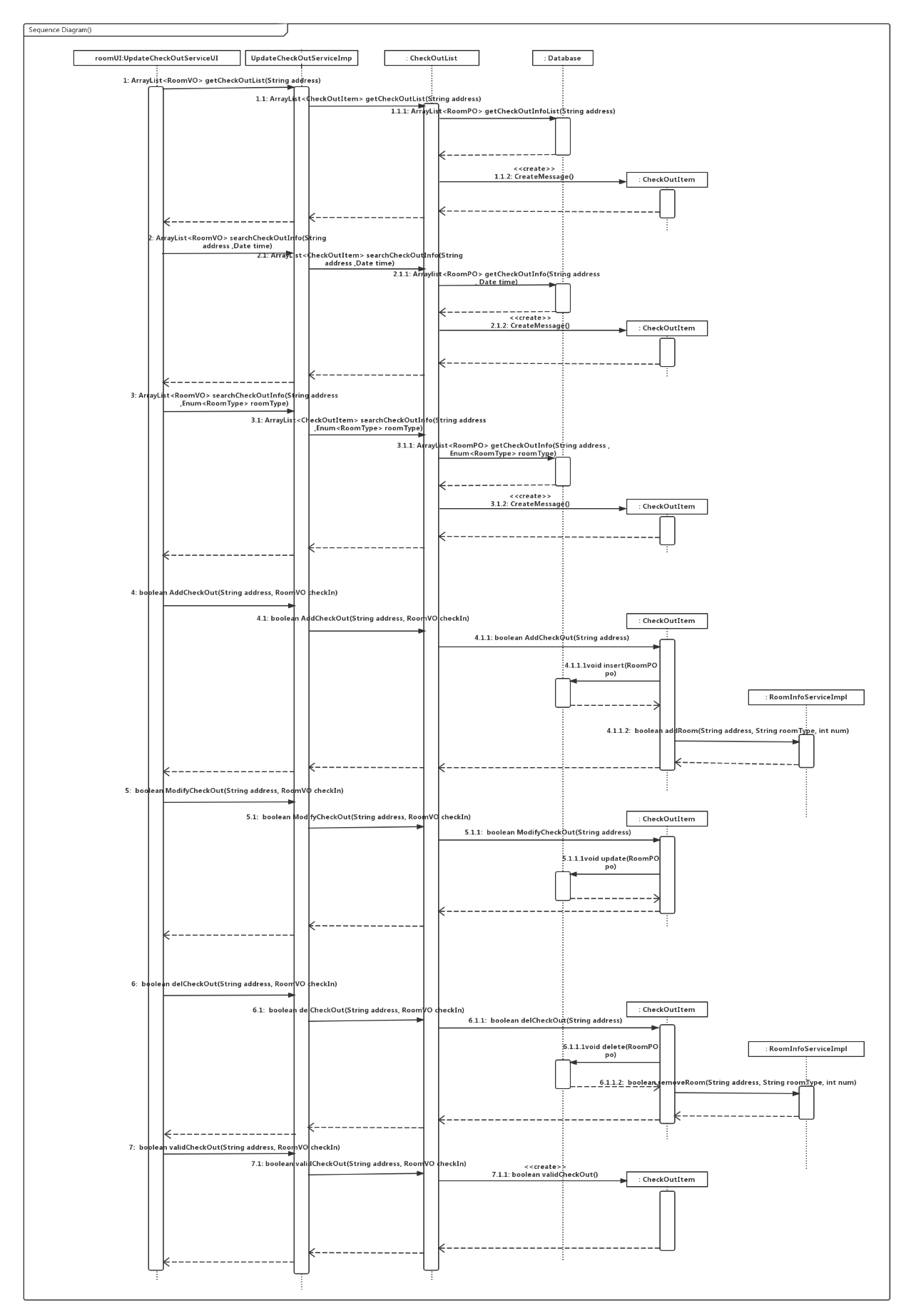
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckOutList.getCheckOutList | 语法 | public ArrayList<CheckOutItem> getCheckOutList(String address) |
| 前置条件 | 启动一个得到退房信息列表回合 |
| 后置条件 | 在一个得到退房信息列表回合中，返回退房信息列表 |
| CheckOutList. searchCheckOutInfo | 语法 | public ArrayList<CheckOutItem> searchCheckOutInfo(String address ,Date time) |
| 前置条件 | 启动一个搜索退房信息回合 |
| 后置条件 | 在一个搜索退房信息回合中，返回搜索到的退房信息列表 |
| CheckOutList. searchCheckOutInfo | 语法 | public ArrayList<CheckOutItem> searchCheckOutInfo(String address ,Enum<RoomType> roomType) |
| 前置条件 | 启动一个搜索退房信息回合 |
| 后置条件 | 在一个搜索退房信息回合中，返回搜索到的退房信息列表 |
| CheckOutList. addCheckOut | 语法 | public boolean addCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutItem对象 |
| 后置条件 | 调用CheckOutItem对象的addCheckOut方法 |
| CheckOutList. modifyCheckOut | 语法 | public boolean modifyCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutItem对象 |
| 后置条件 | 调用CheckOutItem对象的modifyCheckOut方法 |
| CheckOutList. delCheckOut | 语法 | public boolean delCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutItem对象 |
| 后置条件 | 调用CheckOutItem对象的delCheckOut方法 |
| CheckOutList. validCheckOut | 语法 | public boolean validCheckOut(String address, RoomVO CheckOut) |
| 前置条件 | 已创建一个CheckOutItem对象 |
| 后置条件 | 调用CheckOutItem对象的validCheckOut方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CheckOutItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |
| RoomDAO. getCheckOutInfo(String address , Date time) | 得到对应退房时间的退房信息列表 | |
| RoomDAO. getCheckOutInfo(String address ,Enum<RoomType> roomType) | 得到对应房间类型的退房信息列表 | |
| CheckOutItem.addCheckOut | 提供增加退房信息的服务 | |
| CheckOutItem. modifyCheckOut | 提供修改退房信息的服务 | |
| CheckOutItem. delCheckOut | 提供删除退房信息的服务 | |
| CheckOutItem. validCheckOut | 提供验证退房信息的服务 | |

CheckOutItem的接口规范

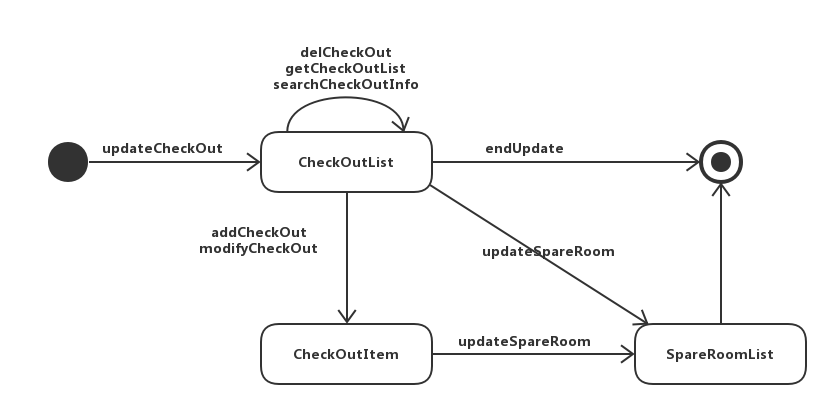
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckOutItem.toVO | 语法 | public RoomVO toVO() |
| 前置条件 | 无 |
| 后置条件 | 返回退房信息CheckInOutVO |
| CheckOutItem.addCheckOut | 语法 | boolean addCheckOut(String address) |
| 前置条件 | 启动一个增加退房信息回合 |
| 后置条件 | 返回是否增加成功 |
| CheckOutItem. modifyCheckOut | 语法 | boolean modifyCheckOut(String address) |
| 前置条件 | 启动一个修改退房信息回合 |
| 后置条件 | 返回是否修改成功 |
| CheckOutItem. delCheckOut | 语法 | boolean delCheckOut(String address) |
| 前置条件 | 启动一个删除退房信息回合 |
| 后置条件 | 返回是否删除成功 |
| CheckOutItem. validCheckOut | 语法 | boolean validCheckOut(String address) |
| 前置条件 | 启动一个验证退房信息回合 |
| 后置条件 | 返回是否验证成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| RoomDAO.insert(RoomPO po) | 插入单一持久化对象 | |
| RoomDAO.update(RoomPO po) | 更新单一持久化对象 | |
| RoomDAO.delete(RoomPO po) | 删除单一持久化对象 | |

**业务逻辑层的动态模型**

UpdateCheckOut的顺序图如下图所示。



UpdateCheckOut的状态图如下图所示。



**4.1.5** **strategybl模块**

**（1）模块概述**

strategybl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。

strategybl模块的职责及接口参见软件体系结构文档。

**（2）整体结构**

展示层和业务逻辑层之间，我们添加businesslogicservice. strategyblservice.UpdateStrategyService接口。业务逻辑层和数据层之间添加dataservice.strategyDAO.StrategyDAO接口。由于本模块涉及到的逻辑控制职责较为简单，将其与业务逻辑职责合并形成UpdateStrategyServiceImpl类。StrategyPO是作为促销策略的持久化对象被添加到涉及模型中去的。StrategyList和StrategyItem的添加是Strategy的容器类StrategyInfoService都是根据依赖倒置原则，为了消除循环依赖而产生的接口。

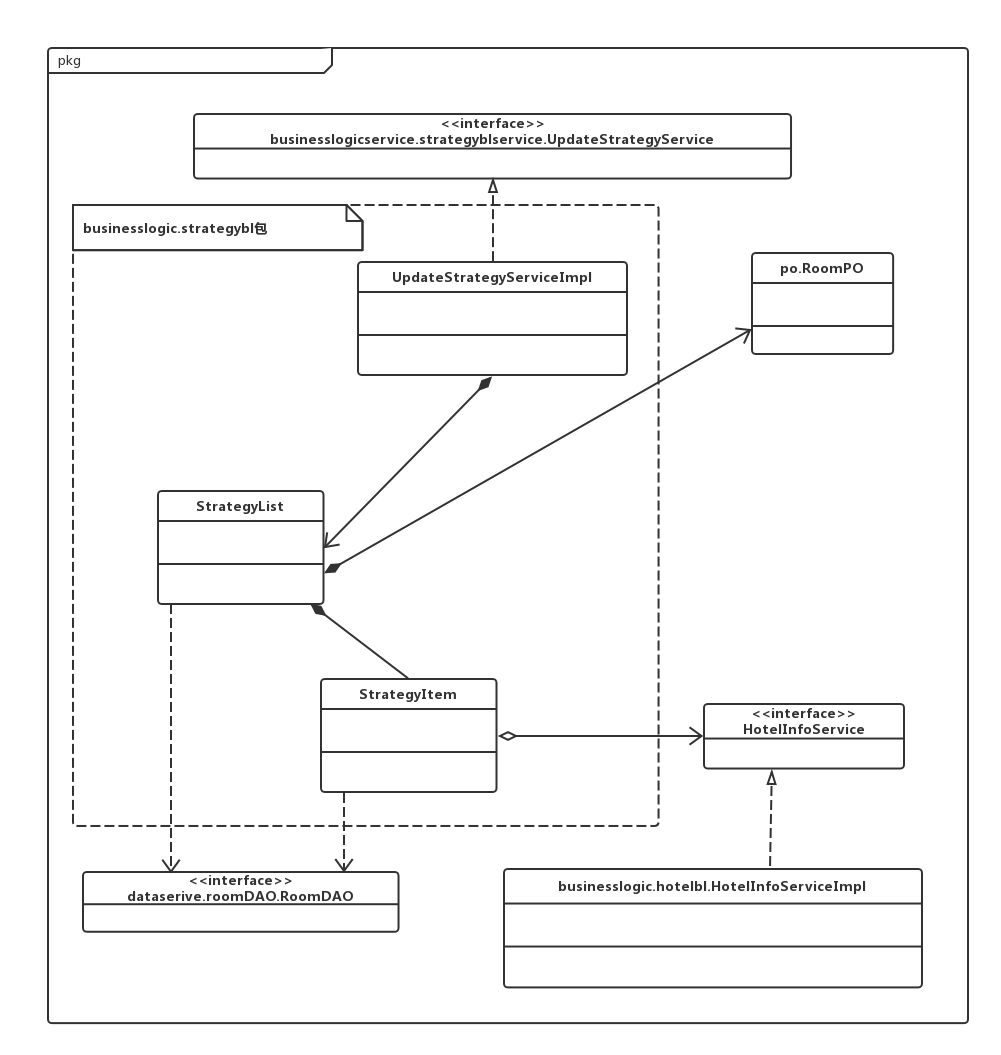
strategybl模块各个类的职责如下表所示。

Strategybl模块各个类的职责

|  |  |
| --- | --- |
| **类名** | **职责** |
| updateStrategy.UpdateStrategyServiceImpl | 负责实现对应于更新促销和营销策略所需要实现的服务 |
| updateStrategy.StrategyList | 策略列表对象，拥有策略对象引用，可以提供某种策略类型的策略列表，提供搜索，增加，修改，删除，验证策略信息的服务 |
| updateStrategy.StrategyItem | 策略对象，拥有酒店地址，折扣类型，折扣名称，折扣百分比，帮助完成增加，修改，删除，验证策略信息的服务 |
| StrategyInfoServiceImpl | 为同层的其他模块提供有关策略信息的服务 |

**4.1.5.1 UpdateStrategy**

UpdateStrategy模块的设计如下图所示。



**接口规范：**

UpdateStrategyServiceImpl, StrategyList和StrategyItem的接口规范如下表所示。

UpdateStrategyServiceImpl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UpdateStrategyServiceImpl. getStrategyList | 语法 | public ArrayList<RoomVO> getStrategyList(String address , Enum<StrategyType> StrategyType) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的getStrategyList方法 |
| UpdateStrategyServiceImpl. getStrategyInfo | 语法 | public ArrayList<RoomVO> getStrategyInfo(String address , String name) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的getStrategyInfo方法 |
| UpdateStrategyServiceImpl. add | 语法 | public boolean add(String address, StrategyVO strategy) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的add方法 |
| UpdateStrategyServiceImpl. modify | 语法 | public boolean add(String address, StrategyVO strategy) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的modify方法 |
| UpdateStrategyServiceImpl.delete | 语法 | public boolean delete (String address,StrategyVO Strategy) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的del方法 |
| UpdateStrategyServiceImpl.valid | 语法 | public boolean valid (String address, StrategyVO Strategy) |
| 前置条件 | 已创建一个StrategyList对象 |
| 后置条件 | 调用StrategyList对象的valid方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| StrategyList.getStrategyList( String address , Enum<StrategyType> strategyType) | 提供某种策略类型的策略列表 | |
| StrategyItem.toVO() | 把StrategyItem转换成StrategyVO类型 | |
| StrategyList.getStrategyInfo(String address, String name) | 提供根据折扣名称搜索到的策略 | |
| StrategyList. add | 提供增加策略的服务 | |
| StrategyList. modify | 提供修改策略的服务 | |
| StrategyList. delete | 提供删除策略的服务 | |
| StrategyList. valid | 提供验证策略的服务 | |

StrategyList的接口规范

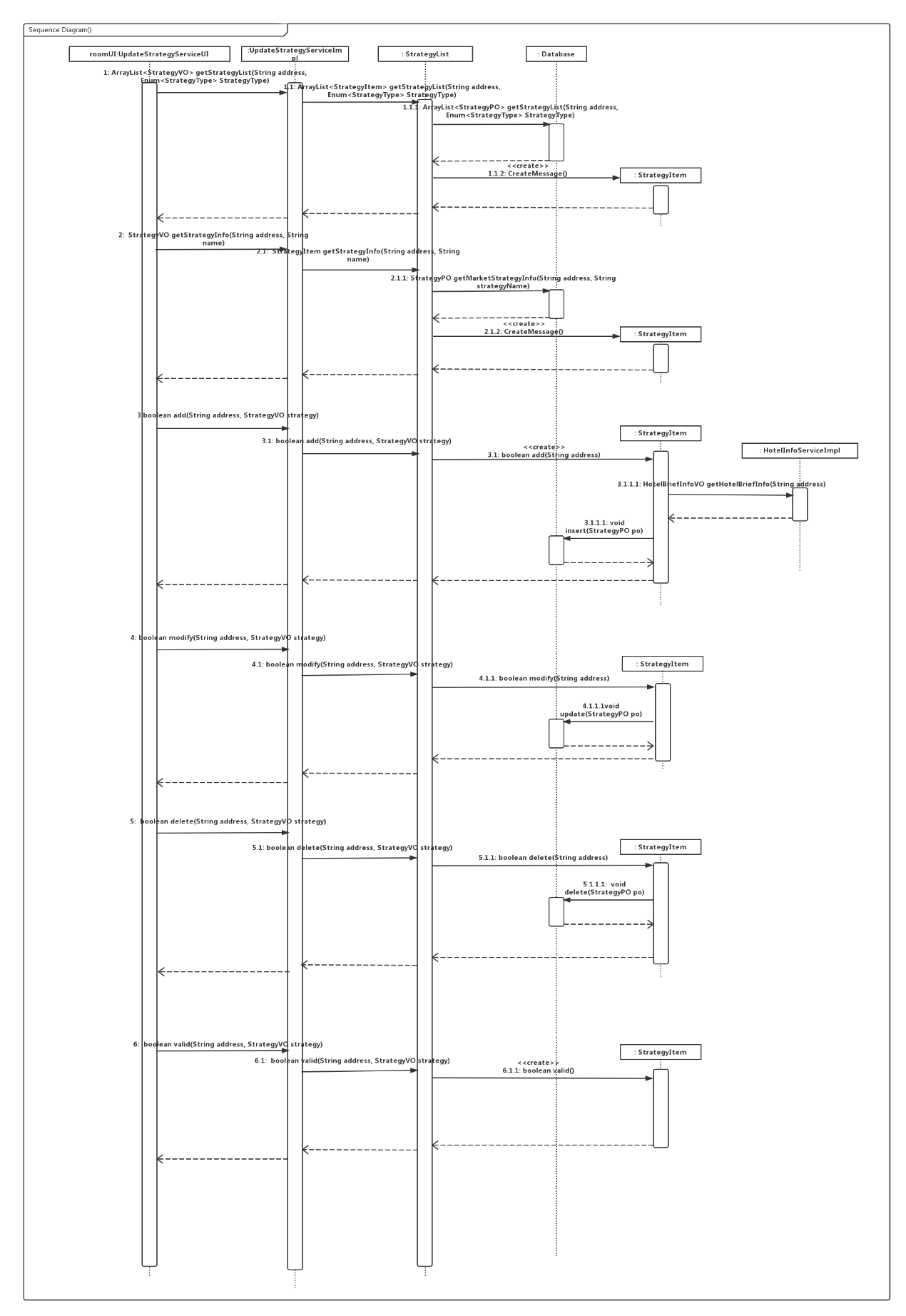
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyList.getStrategyList | 语法 | public ArrayList<StrategyItem> getStrategyList(String address, Enum<StrategyType> strategyType) |
| 前置条件 | 启动一个得到策略列表回合 |
| 后置条件 | 在一个得到策略列表回合中，返回策略列表 |
| StrategyList. getStrategyInfo | 语法 | public StrategyItem getStrategyInfo(String address, String name) |
| 前置条件 | 启动一个搜索策略回合 |
| 后置条件 | 在一个搜索策略回合中，返回搜索到的策略 |
| StrategyList. add | 语法 | public boolean add(String address, StrategyVO strategyVO) |
| 前置条件 | 已创建一个StrategyItem对象 |
| 后置条件 | 调用StrategyItem对象的add方法 |
| StrategyList. modify | 语法 | public boolean modify(String address, StrategyVO strategyVO) |
| 前置条件 | 已创建一个StrategyItem对象 |
| 后置条件 | 调用StrategyItem对象的modify方法 |
| StrategyList. delete | 语法 | public Boolean delete(String address, StrategyVO strategyVO) |
| 前置条件 | 已创建一个StrategyItem对象 |
| 后置条件 | 调用StrategyItem对象的delete方法 |
| StrategyList. valid | 语法 | public boolean valid(String address, StrategyVO strategyVO) |
| 前置条件 | 已创建一个StrategyItem对象 |
| 后置条件 | 调用StrategyItem对象的valid方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| StrategyItem.toVO() | 把SpareRoomItem转换成RoomVO类型 | |
| StrategyDAO.getMarketStrategyInfo(String address, String strategyName) | 得到对应策略名称的策略信息 | |
| StrategyItem.add | 提供增加策略的服务 | |
| StrategyItem. modify | 提供修改策略的服务 | |
| StrategyItem. del | 提供删除策略的服务 | |
| StrategyItem. valid | 提供验证策略的服务 | |

StrategyItem的接口规范

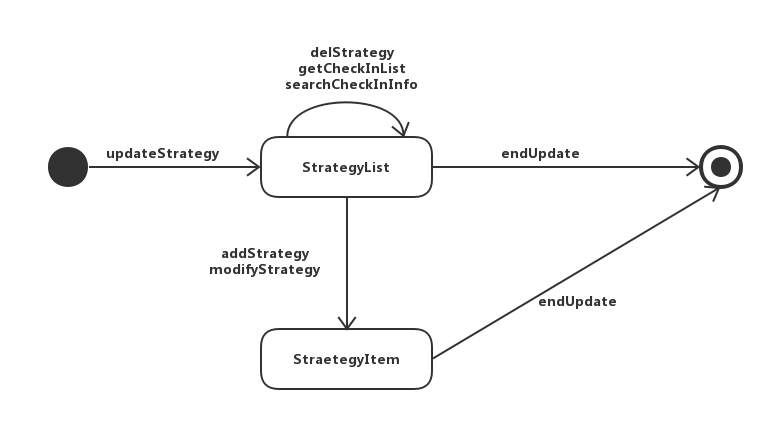
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyItem.toVO | 语法 | public RoomVO toVO() |
| 前置条件 | 无 |
| 后置条件 | 返回策略信息StrategyVO |
| StrategyItem.add | 语法 | boolean add(String address) |
| 前置条件 | 启动一个增加策略回合 |
| 后置条件 | 返回是否增加成功 |
| StrategyItem. modify | 语法 | boolean modify(String address) |
| 前置条件 | 启动一个修改策略回合 |
| 后置条件 | 返回是否修改成功 |
| StrategyItem. del | 语法 | boolean delete(String address) |
| 前置条件 | 启动一个删除策略回合 |
| 后置条件 | 返回是否删除成功 |
| StrategyItem. valid | 语法 | boolean valid(String address) |
| 前置条件 | 启动一个验证策略回合 |
| 后置条件 | 返回是否验证成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| StrategyDAO.insert(StrategyPO po) | 插入单一持久化对象 | |
| StrategyDAO.update(StrategyPO po) | 更新单一持久化对象 | |
| StrategyDAO.delete(StrategyPO po) | 删除单一持久化对象 | |

**业务逻辑层的动态模型**

UpdateStrategy的顺序图如下图所示。



UpdateStrategy的状态图如下图所示。



* 1. **业务逻辑层的设计原理**

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象.

1. **依赖视角**

下图是客户端和服务器端各自的包之间的依赖关系。

