

F.A.F酒店预订系统

体系结构文档



2016-10-15

F.A.F

NJU

**目录**

[**更新历史 1**](#_Toc464310909)

[**1.引言 1**](#_Toc464310910)

[**1.1编制目的 1**](#_Toc464310911)

[**1.2 词汇表 1**](#_Toc464310912)

[**1.3 参考资料 1**](#_Toc464310913)

[**2.产品概述 1**](#_Toc464310914)

[**3.逻辑视角 2**](#_Toc464310915)

[**4.组合视角 3**](#_Toc464310916)

[**4.1开发包图 3**](#_Toc464310917)

[**4.2运行时进程 6**](#_Toc464310918)

[**4.3物理部署 7**](#_Toc464310919)

[**5.接口视角 7**](#_Toc464310920)

[**5.1模块的职责 7**](#_Toc464310921)

[**5.2用户界面层的分解 9**](#_Toc464310922)

[5.2.1用户界面层模块的职责 10](#_Toc464310923)

[5.2.2用户界面层模块的接口规范 10](#_Toc464310924)

[5.2.3用户界面模块设计原理 10](#_Toc464310925)

[**5.3业务逻辑层的分解 11**](#_Toc464310926)

[5.3.1业务逻辑层模块的职责 11](#_Toc464310927)

[5.3.2业务逻辑层模块的接口规范 12](#_Toc464310928)

[5.3.2.1 userbl模块的接口规范 12](#_Toc464310929)

[5.3.2.2 orderbl模块的接口规范 14](#_Toc464310930)

[5.3.2.3 hotelbl模块的接口规范 18](#_Toc464310931)

[5.3.2.4 roombl的模块接口规范 20](#_Toc464310932)

[5.3.2.5 strategybl的模块接口规范 23](#_Toc464310933)

[**5.4数据层的分解 25**](#_Toc464310934)

[5.4.1 数据层模块的职责 27](#_Toc464310935)

[5.4.2 数据层模块的接口规范 29](#_Toc464310936)

[**6.信息视角 35**](#_Toc464310937)

[**6.1数据持久化对象 35**](#_Toc464310938)

[**6.2持久化格式 35**](#_Toc464310939)

[**6.3数据库表 36**](#_Toc464310940)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 更新人员 | 日期 | 变更原因 | 版本号 |
| 万年杰 | 2016.10.11 | 草稿 | V0.0 |
| 田原 | 2016.10.14 | 客户端开发包图中的strategybl去掉原有的orderInfo接口，对调roombl与hotelbl的依赖关系 | V1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1.引言

## 1.1编制目的

本报告详细完成对酒店预订系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员和最终用户而编写，是了解系统的导航。

## 1.2 词汇表

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

## 1.3 参考资料

参考酒店预订系统用例文档和软件需求规格说明文档。

# 2.产品概述

参考酒店预订系统用例文档和酒店预订系统软件需求规格说明书中对产品的概括描述。

# 3.逻辑视角

酒店预订系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑包含业务逻辑处理的实现，数据层负责数据的持久化和访问。

**3.1参照体系结构风格的包图表达逻辑视角**

****

3.2软件体系结构逻辑设计方案

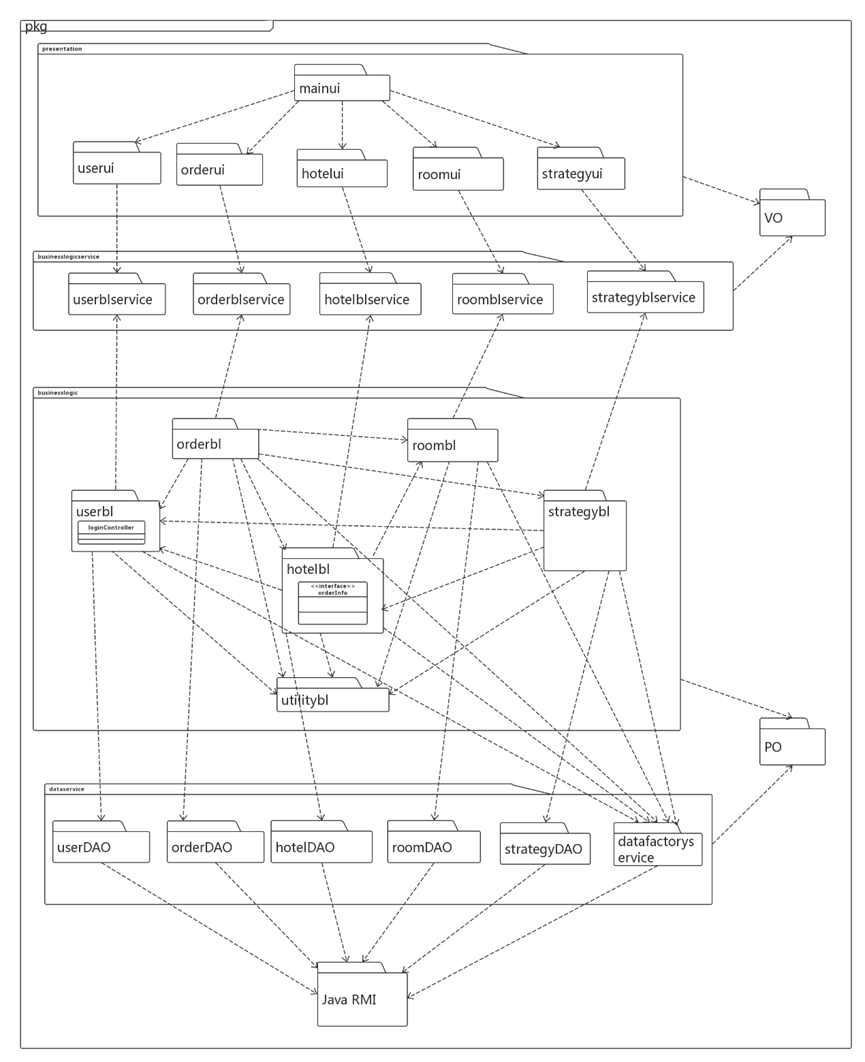
# 4.组合视角

## 4.1开发包图

酒店预定系统最终开发包设计

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| mainui | userui,orderui,hotelui,roomui,strategyui,vo |
| userui | userblservice,界面类库包 |
| userblservice |  |
| userbl | userblservice , userDao ,po, vo |
| userDao | Java RMI,po |
| userdata | userDao ,Java RMI,po ,databaseutility |
| orderui | orderblservice,界面类库包 |
| orderblservice |  |
| orderbl | orderblservice , orderDao ,po,vo, userbl, hotelbl, roombl, strategybl |
| orderDao | Java RMI,po |
| orderdata | orderDao ,Java RMI,po, databaseutility |
| hotelui | hotelblservice,界面类库包 |
| hotelblservice |  |
| hotelbl | hotelblservice , hotelDao ,po,vo,userbl |
| hotelDao | Java RMI,po |
| hoteldata | hotelDao ,Java RMI,po, databaseutility |
| roomui | roomblservice,界面类库包 |
| roomblservice |  |
| roombl | roomblservice , roomDao ,po,vo,hotelbl |
| roomDao | Java RMI,po |
| roomdata | roomDao ,Java RMI,po, databaseutility |
| strategyui | strategyblservice,界面类库包 |
| strategyblservice |  |
| strategybl | strategyblservice , strategyDao ,po,vo, userbl, hotelbl |
| strategyDao | Java RMI,po |
| strategydata | strategyDao ,Java RMI,po, databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

**酒店预订系统客户端开发包图**



**酒店预订系统服务器端开发包图**

## 4.2运行时进程

在酒店预订系统中，会有多个客户端进程和一个服务器端进程，其进程图如图5所示。结合部署图，客户端在客户端机器上运行，服务器端进程在服务器端机器上运行。



## 4.3物理部署

**酒店预订**系统中客户构件是放在客户端机器上，服务器端构件时放在服务器端机器上。在客户端节点上，还要部署\*\*构件。由于Java RMI构件属于JDk6.0的一部分。所以，在系统JDk环境已经设置好的情况下，不要再独立部署。

# 5.接口视角

## 5.1模块的职责

客户端模块和服务器模块视图分别如图7和图8所示。客户端各层和服务器各层的职责分别如表2和表3所示。

**网络模块**

**业务逻辑层**

**展示层**

**启动模块**

**网络模块**

**数据层**

**启动模块**

**客户端模块视图**  **服务器模块图**

**客户端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 基于窗口的快递物流系统客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用JAVA Socket机制实现数据通讯 |

**服务器端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java Socket机制实现数据通讯 |

**层之间调用的接口**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
|  | 客户端展示层 | 客户端业务逻辑层 |
|  | 客户端业务逻辑层 | 服务器端数据层 |

## 5.2用户界面层的分解

 根据需求，系统存在33个用户界面：登录界面、客户主界面、酒店工作人员主界面、网站营销人员主界面、网站管理人员主界面、维护个人基本信息界面、注册会员界面、查看信用记录界面、查看预定过的酒店界面、搜索酒店界面、查看酒店信息界面、生成订单界面、浏览自己的订单界面、撤销订单界面、评价酒店界面、录入可用客房界面、维护酒店基本信息界面、浏览酒店订单界面、执行订单界面、更新退房信息界面、更新入住信息界面、浏览空房列表界面、指定酒店促销策略界面、指定网站营销策略界面、管理信用充值界面、管理异常订单界面、管理用户界面、管理客户界面、管理网站营销人员界面、管理酒店工作人员界面、管理酒店信息界面、管理已存在酒店信息界面、添加新酒店及工作人员界面。

服务端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如图所示。



**5.2.1用户界面层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和界面的跳转 |

**5.2.2用户界面层模块的接口规范**

**用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel |

**用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| businessloicservice.LoginBLService | 登录界面的业务逻辑接口 |
| Businesslogcservice.\*BLService | 每个界面都有一个相应的业务逻辑接口 |

**5.2.3用户界面模块设计原理**

用户界面利用JAVA的Swing和AWT库来实现。

## 5.3业务逻辑层的分解

**业务逻辑层的设计**

****

**5.3.1业务逻辑层模块的职责**

**业务逻辑层模块的职责**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
| 模块 | 职责 |
| userbl | 负责实现与用户相关的服务 |
| orderbl | 负责实现与订单信息及处理相关的服务 |
| hotelbl | 负责实现酒店界面所需要的服务 |
| roombl | 实现与更新空房信息，更新入住和退房信息相关的服务 |
| strategybl | 实现与更新酒店促销策略和网站营销策略相关的服务 |

**5.3.2业务逻辑层模块的接口规范**

**5.3.2.1 userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserService.login | 语法 | public boolean login(long ID, String password) |
| 前置条件 | ID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 |
| UserInfoService.getUserInfo | 语法 | public UserVO getUserInfo(long ID) |
| 前置条件 | 用户或网站管理人员查看用户信息 |
| 后置条件 | 返回该用户信息 |
| UserInfoService.modifyUserInfo | 语法 | public boolean modifyUserInfo(UserVO user) |
| 前置条件 | 客户或网站管理人员修改用户信息 |
| 后置条件 | 返回是否修改成功 |
| UserService.signRegularVip | 语法 | public boolean signRegularVip(UserVO regularVip) |
| 前置条件 | 客户申请普通会员 |
| 后置条件 | 返回是否申请成功 |
| UserService.signEnterpriseVip | 语法 | public boolean signEnterpriseVip(UserVO 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 |
| ClientCreditService.queryCredit | 语法 | public UserVO queryCredit(long ID) |
| 前置条件 | ID符合输入规范 |
| 后置条件 | 返回该客户信用值及信用值变化记录 |
| ClientCreditService.addCreditValue | 语法 | public UserVO addCreditValue(long ID,int creditAdded) |
| 前置条件 | ID和creditAdded符合输入规范 |
| 后置条件 | 返回是否充值成功 |
| UserService.add | 语法 | public boolean add(UserVO user) |
| 前置条件 | 用户注册账号或网站管理人员添加用户账号 |
| 后置条件 | 返回是否注册或添加成功 |
| UserService.del | 语法 | public boolean del(UserVO user) |
| 前置条件 | 用户注销账号或网站管理人员删除用户账号 |
| 后置条件 | 返回是否注销或删除成功 |
| UserService.insert | 语法 | public boolean insert(UserVO staff) |
| 前置条件 | 网站管理人员添加酒店工作人员账号 |
| 后置条件 | 返回是否注册或添加成功 |
| ClientCreditService.getCreditValue | 语法 | public int getCreditValue(long ID) |
| 前置条件 | ID符合输入规范 |
| 后置条件 | 返回客户的当前信用值 |
| ClientCreditService.changeCreditValue | 语法 | public boolean changeCreditValue(long ID,int num) |
| 前置条件 | ID和num符合输入规范 |
| 后置条件 | 返回是否更改信用值成功 |
| UserInfoService.getRegularVipInfo | 语法 | public UserVO getGegularVipInfo(long ID) |
| 前置条件 | ID符合输入规范 |
| 后置条件 | 返回普通会员的信息 |
| UserInfoService.getEnterpriseVipInfo | 语法 | public UserVO getEnterpriseInfo(long ID) |
| 前置条件 | ID符合输入规范 |
| 后置条件 | 返回企业会员的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| userDAO.getUserInfo(long ID) | 得到ID对应的用户信息 | |
| userDAO.queryCredit（long ID) | 得到ID对应的用户信用值及信用值变化记录 | |
| UserDAO.getCreditValue(long ID) | 得到ID对应的用户的当前信用值 | |
| userDAO.insert(UserPO po) | 插入单一持久化对象 | |
| userDAO.del(UserPO po) | 删除单一持久化对象 | |
| 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 | |

**5.3.2.2 orderbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderInfo.getAllOrders | 语法 | public ArrayList<OrderVO> getAllOrders(long ID); |
| 前置条件 | ID有效 |
| 后置条件 | 返回对应客户的所有订单的VO，否则返回null |
| OrderInfo.isReserved | 语法 | public boolean isReserved(long ID, String address); |
| 前置条件 | ID和address有效 |
| 后置条件 | 返回该酒店是否被该用户预定过 |
| OrderInfo.getCommentableOrderList | 语法 | public ArrayList<OrderVO> getCommentableOrderList (long ID); |
| 前置条件 | ID有效 |
| 后置条件 | 返回当前用户可评价酒店对应的订单列表 |
| OrderInfo.getOrderList | 语法 | getOrderList(long ID, String address); |
| 前置条件 | ID和address有效 |
| 后置条件 | 返回该用户在该酒店的所有订单 |
| 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 |
| Order.getSingleOrder | 语法 | public OrderVO getSingleOrder(String address, String orderID); |
| 前置条件 | 酒店工作人员输入订单号 |
| 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 |
| Order.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.getUserOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getUserOrderList (long ID, 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.getAbnormalOrderList | 语法 | public ArrayList<BriefOrderInfoVO> getAbnormalOrderList (Date date); |
| 前置条件 | 网站营销人员发起浏览每日异常订单情况操作 |
| 后置条件 | 返回所查询日期当天所有异常订单列表 |
| Order.initNewOrder | 语法 | public OrderVO initNewOrder(long ID, String address); |
| 前置条件 | 客户确认即将生成订单的酒店信息 |
| 后置条件 | 初始化订单信息，返回一个被初始化的OrderVO |
| Order.checkNewOrder | 语法 | public ResultMessage checkNewOrder(OrderVO vo); |
| 前置条件 | 客户确认生成订单 |
| 后置条件 | 检查订单要求是否可以得到满足，格式是否正确，并返回结果 |
| Order.addNewOrder | 语法 | public boolean addNewOrder(OrderVO vo); |
| 前置条件 | 订单要求可以得到满足 |
| 后置条件 | 增加新的订单 |
| Order.getPrice | 语法 | public int getPrice(OrderVO vo); |
| 前置条件 | 用户已填写房间类型和房间数量 |
| 后置条件 | 计算并返回订单价格 |
| Order.setOrderState | 语法 | public boolean setOrderState(Enum<OrderState> orderstate, OrderVO vo); |
| 前置条件 | 客户选择撤销订单或订单已执行或需要管理异常订单执行情况 |
| 后置条件 | 将当前订单状态置为orderstate所示，并更新数据库 |
| Order.isTooLate | 语法 | public boolean isTooLate(OrderVO vo); |
| 前置条件 | 客户发起撤销订单动作 |
| 后置条件 | 计算此次撤销操作是否需要扣除信用值 |
| Order.delayCheckIn | 语法 | public boolean delayCheckIn (OrderVO vo, int hours); |
| 前置条件 | 酒店工作人员确定延迟入住 |
| 后置条件 | 延迟最晚入住时间，返回是否成功 |
| Order.getOrderDone | 语法 | public boolean getOrderDone(OrderVO vo); |
| 前置条件 | 酒店工作人员确定执行订单 |
| 后置条件 | 更新当前订单状态为已执行订单，并为客户增加与订单金额等值的信用值 |
| Order.isWithdrawable | 语法 | public boolean isWithdrawable(OrderVO vo); |
| 前置条件 | 客户选择撤销订单 |
| 后置条件 | 返回该订单是否可以被撤销 |
| Order.withdrawOrder | 语法 | public boolean withdrawOrder(OrderVO vo); |
| 前置条件 | 客户确认撤销订单 |
| 后置条件 | （扣除信用值），系统将订单置为撤销状态，记录撤销时间，更新空房列表 |
| Order.systemWithdrawOrder | 语法 | public boolean systemWithdrawOrder(OrderVO vo); |
| 前置条件 | 网站营销人员确认撤销订单 |
| 后置条件 | 系统撤销此异常订单并将其状态置为已撤销、记录撤销时间，并提示选择恢复此客户信用值的全部或一半 |
| Order.getLatestDoneTime | 语法 | public Time getLatestDoneTime(OrderVO vo); |
| 前置条件 | 客户发起撤销订单动作 |
| 后置条件 | 返回当前订单的最晚执行时间 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDAO.getUserOrderList(long ID, 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.getUserAllOrders(long ID) | 返回对应客户的所有订单的PO，否则返回null | |
| OrderDAO.insert(OrderPO po) | 插入单一持久化对象 | |
| OrderDAO.delete(OrderPO po) | 删除单一持久化对象 | |
| OrderDAO.update(OrderPO po) | 更新单一持久化对象 | |
| OrderDAO.finish() | 结束持久化数据库的使用 | |
| 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.getAllAbnormalList(Date date) | 得到查询日期当天所有异常订单列表 | |
| 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 | |
| OrderDAO.isReserved(long ID, String address); | 返回该酒店是否被该用户预定过 | |
| OrderDAO.getCommentableOrders (long ID); | 返回当前用户可评价酒店对应的订单列表 | |
| DatabaseFactory.getOrderDatabase | 得到Order数据库服务的引用 | |
| RoomInfoService.getAvailableRoomNum(String address, Enum<RoomType> roomType) | 获取当前酒店roomType房型的空房数量 | |
| RoomInfoService.isTimeAvailable (String address, Enum<RoomType> roomType, Date beginDate, Date finishDate) | 查询用户预定时间段有无用户要求的房型剩余 | |
| 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 | |
| RoomInfoService.reduceSpareRoom(String address , int change, Enum<RoomType> roomType) | 更新（减少）当前酒店被预定房型的可用房间数据 | |
| Hotel.getHotelBriefInfo(String address) | 得到当前酒店的简要信息 | |
| Hotel.getHotelDetails(String address) | 得到当前酒店的详细信息 | |
| User.getCreditValue(long ID) | 得到当前登陆客户的信用值 | |
| User.changeCreditValue(long ID, int num) | 根据num(可正可负)的值，增加或减少当前客户的信用值 | |
| Strategy.getAvailblePromotionName(OrderVO vo) | 得到当前最优酒店优惠策略的名称 | |
| Strategy.getAvailbleMarketStrategyName(OrderVO vo) | 得到当前最优网站优惠策略的名称 | |
| Strategy.getBestDiscount(OrderVO vo) | 得到当前最终的最优的优惠折扣 | |

**5.3.2.3 hotelbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CheckOrderedHotelSerivice.enrollHotelBriefInfoList | 语法 | public ArrayList<HotelBriefInfoVO> enrollHotelBriefInfoList (long ID); |
| 前置条件 | 启动一个浏览预订过的酒店请求 |
| 后置条件 | 返回当前用户所有预订过的酒店简要信息列表 |
| SearchHotelService.getHotelBriefInfoListBySearching | 语法 | public ArrayList<HotelBriefInfoVO> getHotelBriefInfoListBySearching (String[] condition); |
| 前置条件 | condition符合输入规范 |
| 后置条件 | 查找是否存在符合该输入条件的酒店，如果存在则返回所有满足条件的酒店简要信息列表 |
| QueryHotelService.getHotelBriefInfoListByQuerying | 语法 | public ArrayList<HotelBriefInfoVO> getHotelBriefInfoListByQuerying (String[] condition); |
| 前置条件 | condition符合输入规范 |
| 后置条件 | 查找是否存在符合该输入条件的酒店，如果存在则返回所有满足条件的酒店简要信息列表 |
| QueryHotelSevice.getHotelDetails | 语法 | public HotelVO getHotelDetails(String address); |
| 前置条件 | address符合输入规范 |
| 后置条件 | 返回在该地址处的酒店的详细信息 |
| QueryHotelService.getOrders | 语法 | public ArrayList<OrderVO> getOrders(String address, long ID); |
| 前置条件 | address和ID符合输入规范 |
| 后置条件 | 返回该用户在该酒店的所有订单 |
| CommentOnHotelService.getCommentableOrderList | 语法 | public ArrayList<OrderVO> getCommentableOrderList(long ID); |
| 前置条件 | 启动一个评价酒店请求 |
| 后置条件 | 返回可以评价的酒店对应的已执行订单列表 |
| CommentOnHotelService.confirmComment | 语法 | public boolean confirmComment(String username, int mark, String comment); |
| 前置条件 | mark和comment符合输入规范 |
| 后置条件 | 返回是否评论成功 |
| MaintainHotelBasicInfoService.enrollHotelBasicInfo | 语法 | public HotelVO enrollHotelBasicInfo(String address); |
| 前置条件 | 启动一个修改或浏览酒店的基本信息的请求 |
| 后置条件 | 返回该酒店的基本信息 |
| MaintainHotelBasicInfoService.confirmModify | 语法 | public boolean confirmModify(HotelVO modified); |
| 前置条件 | modified符合输入规范 |
| 后置条件 | 返回是否修改成功 |
| ImportNewRoomService.getAvailableRoomList | 语法 | public ArrayList<RoomVO> getAvailableRoomList(String address); |
| 前置条件 | 启动一个查看酒店可用客房或者录入可用客房的请求 |
| 后置条件 | 返回该酒店的可用客房列表 |
| ImportNewRoomService.addRoom | 语法 | public boolean addRoom(RoomVO room); |
| 前置条件 | room符合输入规范 |
| 后置条件 | 返回是否录入成功 |
| ManageHotelInfoService.addHotel | 语法 | public boolean addHotel(HotelVO hotel); |
| 前置条件 | hotel符合输入规范 |
| 后置条件 | 返回是否添加酒店成功 |
| ManageHotelInfoService.addHotelStaff | 语法 | public boolean addHotelStaff(UserVO staff); |
| 前置条件 | staff符合输入规范 |
| 后置条件 | 返回是否添加酒店工作人员成功 |
| HotelInfoService.getHotelBriefInfo | 语法 | public HotelBriefInfoVO getHotelBriefInfo(String address); |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的简要信息 |
| HotelInfoService. getHotelDetails | 语法 | public HotelVO getHotelDetails(String address); |
| 前置条件 | address符合输入规范 |
| 后置条件 | 返回在该地址处的酒店的详细信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderInfo.getAllOrders(long ID) | 得到当前用户所有订单 | |
| HotelDAO.getHotelBriefInfo(String address) | 得到单一的酒店简要信息持久化对象 | |
| OderInfo.isReserved(String ID, String address) | 得到该酒店是否被该用户预定过 | |
| HotelDAO.getHotelBriefInfoListBySearching(String[] condition) | 根据搜索条件得到酒店简要信息持久化对象列表 | |
| HotelDAO.getHotelBriefInfoListByQuerying(String[] condition) | 根据查看条件得到酒店简要信息持久化对象列表 | |
| HotelDAO.getHotelDetails(String address) | 得到该酒店详细信息的持久化对象 | |
| OrderInfo.getOrderList(String ID, String address) | 得到当前用户在该酒店的所有订单 | |
| OrderInfo.getCommentableOrderList(long ID) | 得到当前用户可评价酒店对应的订单列表 | |
| HotelDAO.update((HotelPO hotel) | 更新单一持久化对象 | |
| HotelDAO.find(String address) | 根据address查找单一持久化对象 | |
| HotelDAO.insert(HotelPO hotel) | 在数据库中插入hotelPO对象 | |
| userbl.insert(UserPO staff) | 在数据库中插入UserPO对象 | |
| DatabaseFactory.getHotelDatabase | 得到Hotel数据库的服务的引用 | |
| roombl.updateSpareRoom(RoomVO roomvo) | 更新酒店空房信息 | |

**5.3.2.4 roombl的模块接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomInfoService.getSpareRoomInfoList | 语法 | public ArrayList<RoomVO> getRoomInfoList (String address) |
| 前置条件 | 启动一个查看空房列表的请求 |
| 后置条件 | 返回空房列表 |
| RoomInfoService.isTimeAvailable (Enum roomType, OrderTime time) | 语法 | public boolean isTimeAvailable (String addresss, Enum roomType, OrderTime time) |
| 前置条件 | 无 |
| 后置条件 | 返回用户预定时间段有无用户要求的房型剩余 |
| RoomInfoService.updateSpareRoom | 语法 | public boolean updateSpareRoom (String address, RoomVO roomvo) |
| 前置条件 | 每次更新可用客房后 |
| 后置条件 | 返回是否更新空房信息成功 |
| RoomInfoService.reduceSpareRoom | 语法 | public boolean reduceRoom(String address, String roomType, int num) |
| 前置条件 | （1）用户生成订单时  （2）线下入住并更新入住信息时  （3）异常订单通过酒店工作人员手动延迟入住变为已执行订单时 |
| 后置条件 | 返回是否减少空房成功 |
| RoomInfoService.AddSpareRoom | 语法 | public boolean addRoom(String address, String roomType, int num) |
| 前置条件 | （1）更新退房信息时  （2）当正常订单变成异常订单时  （3）用户撤销订单时 |
| 后置条件 | 返回是否增加空房成功 |
| BrowseSpareRoomService.getRoomInfoList | 语法 | public ArrayList<RoomVO> getRoomInfoList (String address) |
| 前置条件 | 启动一个浏览空房信息的请求 |
| 后置条件 | 返回空房信息列表 |
| UpdateCheckInService.getCheckInList | 语法 | public ArrayList<RoomVO> getCheckInList(String address) |
| 前置条件 | 启动一个更新入住信息请求 |
| 后置条件 | 返回入住信息列表 |
| UpdateCheckInService.searchCheckInInfo | 语法 | public RoomVO searchCheckInInfo(String address ,String time) |
| 前置条件 | 无 |
| 后置条件 | 按入住时间进行查找返回相应的入住信息 |
| UpdateCheckInService.searchCheckInInfo | 语法 | public RoomVO searchCheckInInfo (String address ,String roomType) |
| 前置条件 | 无 |
| 后置条件 | 按房间类型进行查找返回相应的入住信息 |
| UpdateCheckInService.AddCheckIn | 语法 | public boolean AddCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 启动增加入住信息请求，checkIn符合规范 |
| 后置条件 | 返回是否更新成功 |
| UpdateCheckInService.modifyCheckIn | 语法 | public boolean ModifyCheckIn(String address, RoomVO checkIn) |
| 前置条件 | checkIn符合规范 |
| 后置条件 | 启动修改入住信息请求，返回是否修改成功 |
| UpdateCheckInService.delCheckIn | 语法 | public boolean delCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 启动删除入住信息请求，checkIn在入住信息列表种已存在 |
| 后置条件 | 返回是否删除成功 |
| UpdateCheckInService.validCheckIn | 语法 | public boolean validCheckIn(String address, RoomVO checkIn) |
| 前置条件 | 无 |
| 后置条件 | 返回checkIn的信息是否符合规范 |
| UpdateCheckOutService.getCheckOutList | 语法 | public ArrayList<RoomVO> getCheckOutList(String address) |
| 前置条件 | 启动一个更新退房信息请求 |
| 后置条件 | 返回退房信息列表 |
| UpdateCheckOutService.searchCheckOutInfo | 语法 | public RoomPO getCheckOutInfo(String address, String time) |
| 前置条件 | 无 |
| 后置条件 | 按实际离开时间进行查找返回相应的退房信息 |
| UpdateCheckOutService.searchCheckOutInfo | 语法 | public RoomPO getCheckOutInfo(String address, String roomType) |
| 前置条件 | 无 |
| 后置条件 | 按房间类型进行查找返回相应的退房信息 |
| UpdateCheckOutService.addCheckOut | 语法 | public boolean AddCheckOut(String address, RoomVO roomvo) |
| 前置条件 | 启动增加退房信息请求，checkOut符合规范 |
| 后置条件 | 返回是否更新成功 |
| UpdateCheckOutService.modifyCheckOut | 语法 | public Boolean ModifyCheckOut(String address, RoomVO roomvo) |
| 前置条件 | 启动修改退房信息请求，checkOut符合规范 |
| 后置条件 | 返回是否修改成功 |
| UpdateCheckOutService.delCheckOut | 语法 | public boolean delCheckOut (String address, RoomVO roomvo) |
| 前置条件 | 启动删除退房信息请求，checkOut在入住信息列表种已存在 |
| 后置条件 | 返回是否删除成功 |
| UpdateCheckOutService.validCheckOut | 语法 | public boolean validCheckIn(String address, RoomVO checkOut) |
| 前置条件 | 无 |
| 后置条件 | 返回checkOut的信息是否符合规范 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| RoomDAO.getSpareRoomInfoList(String address) | 得到空房信息持久化对象列表 | |
| RoomDAO.getSpareRoomInfo(String address, String roomType) | 得到对应房间类型的空房信息持久化对象 | |
| RoomDAO.getCheckInInfoList(String address) | 得到入住信息持久化对象列表 | |
| RoomDAO.getCheckInInfo(String address, String time) | 得到对应入住时间的入住信息持久化对象 | |
| RoomDAO.getCheckInInfo(String address, String roomType) | 得到对应房间类型的入住信息持久化对象 | |
| RoomDAO.getCheckOutInfoList(String address) | 得到退房信息持久化对象列表 | |
| RoomDAO.getCheckOutInfo(String address, String time) | 得到对应实际离开时间的退房信息持久化对象 | |
| RoomDAO.getCheckOutInfo(String address, String roomType) | 得到对应房间类型的退房信息持久化对象 | |
| RoomDAO.find(long id) | 根据ID查找单一持久化对象RoomPO | |
| RoomDAO.update(RoomPO po) | 更新单一持久化对象 | |
| RoomDAO.insert(RoomPO po) | 在数据库插入Room PO对象 | |
| RoomDAO.delete(RoomPO po) | 在数据库删除某Room PO对象 | |
| DatabaseFactory.getRoomDatabase | 得到Room数据库的服务的引用 | |

**5.3.2.5 strategybl的模块接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StrategyInfoService.getAvailble PromotionName | 语法 | public String .getAvailblePromotionName (OrderVO order) |
| 前置条件 | 无 |
| 后置条件 | 返回某订单能享受的唯一酒店促销策略折扣名称，如果没有，则返回null |
| StrategyInfoService.getAvailbleMarketStrategyName | 语法 | public String getAvailbleMarketStrategyName (OrderVO order) |
| 前置条件 | 无 |
| 后置条件 | 返回某订单能享受的唯一网站营销策略折扣名称，如果没有，则返回null |
| StrategyInfoService. getBestDiscount | 语法 | public int getBestDiscount(OrderVO order) |
| 前置条件 | 无 |
| 后置条件 | 返回某订单的最终的折扣百分比 |
| UpdateStrategyService.getStrategyList | 语法 | public ArrayList<StrategyVO> getStrategyList(String address, String StrategyType) |
| 前置条件 | 启动一个制定策略的请求 |
| 后置条件 | 返回某种策略类型的折扣列表 |
| UpdateStrategyService.searchStrategyInfo | 语法 | public StrategyVO getStrategyInfo(String address, String name) |
| 前置条件 | 无 |
| 后置条件 | 返回某个折扣名称的具体折扣信息 |
| UpdateStrategyService.add | 语法 | public boolean add(String address, StrategyVO strategy) |
| 前置条件 | 该StrategyVO符合规范，通过Valid方法 |
| 后置条件 | 返回是否制定策略成功 |
| UpdateStrategyService.modify | 语法 | public boolean modify(String address, StrategyVO strategy) |
| 前置条件 | 该StrategyVO符合规范，通过Valid方法 |
| 后置条件 | 返回是否修改策略信息成功 |
| UpdateStrategyService.del | 语法 | public boolean delete(String address, StrategyVO strategy) |
| 前置条件 | 该StrategyVO已存在 |
| 后置条件 | 返回是否删除成功 |
| UpdateStrategyService.Valid | 语法 | public boolean Valid(String address, StrategyVO strategy) |
| 前置条件 | 无 |
| 后置条件 | 返回该策略信息是否符合规范 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StrategyDAO.getStrategyList(String address, String StrategyType) | 得到策略折扣持久化对象列表 | |
| StrategyDAO.getStrategyInfo(String address, String strategyName) | 得到对应折扣名称的折扣策略持久化对象 | |
| StrategyDAO.find(long id) | 根据ID查找单一持久化对象StrategyPO | |
| StrategyDAO.update(StrategyPO po) | 更新单一持久化对象 | |
| StrategyDAO.insert(StrategyPO po) | 在数据库插入StrategyPO对象 | |
| StrategyDAO.delete(StrategyPO po) | 在数据库删除某StrategyPO对象 | |
| DatabaseFactory.getStrategyDatabase | 得到Strategy数据库的服务的引用 | |
| HotelInfoService.getHotelBrifInfo(String address) | 得到酒店的简要信息(商圈名称) | |
| User.getRegularVipInfo(String ID) | 得到普通会员信息 | |
| User.getEnterpriseVipInfo(String ID) | 得到企业会员信息 | |

## 5.4数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对持久化数据的增、删、改、查。User业务逻辑需要的服务由userDAO接口提供,Order业务逻辑需要的服务由orderDAO接口提供,Hotel业务逻辑需要的服务由hotelDAO接口提供,Room业务逻辑需要的服务由roomDAO接口提供,strategy业务逻辑需要的服务由strategyDAO接口提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件等，所示抽象了数据服务。数据层模块的描述具体如下图所示。



**5.4.1 数据层模块的职责**

**数据层模块的职责**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

|  |  |
| --- | --- |
| **模块** | **职责** |
| userDAO | 持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| userDAOTxtFileImpl | 基于Txt文件的持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| userDAOSerializableFileImpl | 基于序列化文件的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| userDAOMysqlImpl | 基于Mysql数据库的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| orderDAO | 持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| orderDAOTxtFileImpl | 基于Txt文件的持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| orderDAOSerializableFileImpl | 基于序列化文件的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| orderDAOMysqlImpl | 基于Mysql数据库的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| hotelDAO | 持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| hotelDAOTxtFileImpl | 基于Txt文件的持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| hotelDAOSerializableFileImpl | 基于序列化文件的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| hotelDAOMysqlImpl | 基于Mysql数据库的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| roomDAO | 持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| roomDAOTxtFileImpl | 基于Txt文件的持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| roomDAOSerializableFileImpl | 基于序列化文件的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| roomDAOMysqlImpl | 基于Mysql数据库的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| strategyDAO | 持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| strategyDAOTxtFileImpl | 基于Txt文件的持久化数据的接口，提供集体载入、集体保存、增、改、查服务 |
| strategyDAOSerializableFileImpl | 基于序列化文件的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |
| strategyDAOMysqlImpl | 基于Mysql数据库的持久化数据接口，提供集体载入、集体保存、增、改、查服务 |

**5.4.2 数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| userDAO.getUserInfo | 语法 | public UserPO getUserInfo(long ID) throws RemoteException |
| 前置条件 | 无 |
| 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 |
| userDAO.queryCredit | 语法 | public UserPO queryCredit(long ID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的ID，存在则返回相应的PO |
| userDAO.getCreditValue | 语法 | public int getCreditValue(long ID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的ID，存在则返回相应的信用值 |
| userDAO.insert | 语法 | public void insert(UserPO po) throws RemoteException |
| 前置条件 | 在数据库中不存在同样ID的po |
| 后置条件 | 在数据库中增加一个po记录 |
| userDAO.del | 语法 | public void delete(UserPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 删除一个po |
| userDAO.update | 语法 | public void update(UserPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| userDAO.finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| orderDAO.getUserOrderList | 语法 | public ArrayList<BriefOrderInfoPO> getUserOrderList(long ID, Enum<OrderType> orderType) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按orderType的值进行查找返回相应的OrderPO列表（全部订单，未执行订单，已执行订单，异常订单和已撤销订单4种） |
| orderDAO.getUserAllOrders | 语法 | public ArrayList<OrderPO> getUserAllOrders(long ID) throws RemoteException; |
| 前置条件 | ID有效 |
| 后置条件 | 返回对应客户的所有订单的PO，否则返回null |
| orderDAO.getSingleOrder | 语法 | public OrderPO getSingleOrder(String address, String orderID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 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 |
| orderDAO.isReserved | 语法 | public boolean isReserved(long ID, String address) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店是否被该用户预定过 |
| orderDAO. getCommentableOrders | 语法 | public ArrayList< OrderPO > getCommentableOrders (long ID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当前用户可评价酒店对应的订单列表 |
| OrderDAO.insert | 语法 | public void insert(OrderPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| OrderDAO.delete | 语法 | public void delete(OrderPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 删除一个po |
| OrderDAO.update | 语法 | public void update(OrderPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| OrderDAO.finish | 语法 | public void finish() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |
| OrderDAO.getAllAbnormalList | 语法 | public ArrayList<BriefOrderInfoPO> getAllAbnormalList (Date date) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回所查询日期当天所有异常订单列表，若无异常订单，则返回null（数据库中所有的） |
| OrderDAO.getHotelOrderList | 语法 | public ArrayList<BriefOrderInfoPO> getHotelOrderList(String address, Enum<OrderType> orderType) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按orderType的值进行查找并按生成时间顺序返回该酒店的相应的订单列表，若无对应类型，则返回null |
| **提供的服务（供接口）** | | |
| HotelDAO.getHotelBriefInfo | 语法 | public HotelBriefInfoPO getHotelBriefInfo(String address) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按address进行查找返回相应的HotelBriefInfoPO结果 |
| HotelDAO.getHotelBriefInfoListBySearching | 语法 | public ArrayList<HotelBriefInfoPO> getHotelBriefInfoListBySearching (String[] condition) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按照condition进行查找返回相应的所有HotelBriefInfoPO |
| HotelDAO.getHotelBriefInfoListByQuerying | 语法 | public ArrayList<HotelBriefInfoPO> getHotelBriefInfoListByQuerying (String[] condition) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按照condition进行查找返回相应的所有HotelBriefInfoPO |
| HotelDAO.getHotelDetails | 语法 | public HotelPO getHotelDetails(String address) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 根据address进行查找返回相应的HotelPO |
| HotelDAO.update | 语法 | public void update(HotelPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在相同ID的po |
| 后置条件 | 更新一个po |
| HotelDAO.find | 语法 | public HotelPO find(String address) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按address进行查找返回相应的HotelPO结果 |
| HotelDAO.insert | 语法 | public void insert(HotelPO po) throws RemoteException; |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| HotelDAO.init | 语法 | public void init() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据存储 |
| HotelDAO.finish | 语法 | public void finish() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据存储的使用 |

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RoomDAO.getSpareRoomInfoList | 语法 | public ArrayList<RoomPO> getSpareRoomInfoList(String address) |
| 前置条件 | 无 |
| 后置条件 | 返回空房列表 |
| RoomDAO.getSpareRoomInfo | 语法 | public RoomPO getSpareRoomInfo(String address , String roomType) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按房间类型进行查找返回相应的空房信息 |
| RoomDAO.getCheckInInfoList | 语法 | public ArrayList<RoomPO> getCheckInInfoList(String address) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回入住信息列表 |
| RoomDAO.getCheckInInfo | 语法 | public RoomPO getCheckInInfo(String address , String time) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按入住时间进行查找返回相应的入住信息 |
| RoomDAO.getCheckInInfo | 语法 | public RoomPO getCheckInInfo (String address, String roomType) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按房间类型进行查找返回相应的入住信息 |
| RoomDAO.getCheckOutInfoList | 语法 | public ArrayList<RoomPO> getCheckOutInfoList(String address) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回退房信息列表 |
| RoomDAO.getCheckOutInfo | 语法 | public RoomPO getCheckOutInfo(String address, String time) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按实际离开时间进行查找返回相应的退房信息 |
| RoomDAO.getCheckOutInfo | 语法 | public RoomPO getCheckOutInfo(String address, String roomType) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按房间类型进行查找返回相应的退房信息 |
| RoomDAO.find | 语法 | public RoomPO find(long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的RoomVO结果 |
| RoomDAO.update | 语法 | public void update(RoomPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| RoomDAO.insert | 语法 | public void insert(RoomPO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| RoomDAO.delete | 语法 | public void delete(RoomPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| RoomDAO.init | 语法 | public void init(RoomPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 持久化数据存储 |
| RoomDAO.finish | 语法 | public void finish(RoomPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |
| **提供的服务（供接口）** | | |
| StrategyDAO.getStrategyList | 语法 | public ArrayList<StrategyPO> getStrategyList(String address, String StrategyType) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回某种策略折扣列表 |
| StrategyDAO.getStrategyInfo | 语法 | public StrategyPO getMarketStrategyInfo(String address, String strategyName) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的策略折扣 |
| StrategyDAO.find | 语法 | public StrategyPO find(long id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的StrategyVO结果 |
| StrategyDAO.update | 语法 | public void update(StrategyPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| StrategyDAO.insert | 语法 | public void insert(StrategyPO po) throws RemoteException |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| StrategyDAO.delete | 语法 | public void delete(StrategyPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| StrategyDAO..init | 语法 | public void init(StrategyPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 持久化数据存储 |
| StrategyDAO.finish | 语法 | public void finish(StrategyPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 无结束持久化数据库的使用 |

# 6.信息视角

## 6.1数据持久化对象

系统的PO类就是对应的实体类，在此只做简单的介绍。

* UserPO类包含
* OrderPO类包含
* HotelPO类包含
* RoomPO类包含
* StrategyPO类包含

持久化用户对象**\*\*\***PO的定义如下所示。

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

## 6.2持久化格式

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**6.3数据库表**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***