

酒店预订系统体系结构描述文档

Winter Studio



2016-10-16

编制：朱润之 徐天泽 杨凯 张磊

南京大学软件学院

目录

[1.引言 1](#_Toc470357855)

[1.1编制目的 1](#_Toc470357856)

[1.2 词汇表 1](#_Toc470357857)

[1.3 参考资料 1](#_Toc470357858)

[2.产品概述 2](#_Toc470357859)

[3.逻辑视角 2](#_Toc470357860)

[4.组合视角 2](#_Toc470357861)

[4.1开发包图 2](#_Toc470357862)

[4.2运行时进程 5](#_Toc470357863)

[4.3物理部署 5](#_Toc470357864)

[5.接口视角 6](#_Toc470357865)

[5.1 模块的职责 6](#_Toc470357866)

[**5.2用户界面层的分解** 10](#_Toc470357867)

[5.3业务逻辑层的分解 14](#_Toc470357868)

[5.3.1 业务逻辑层模块的职责 14](#_Toc470357869)

[5.3.2业务逻辑层的接口规范 15](#_Toc470357870)

[5.4数据层的分解 33](#_Toc470357871)

[5.4.1数据层模块的职责 33](#_Toc470357872)

[5.4.2数据层模块的接口规范 33](#_Toc470357873)

[6 信息视角 37](#_Toc470357874)

[6.1数据持久化对象 37](#_Toc470357875)

[6.2 数据库表 61](#_Toc470357876)

# 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、测试人员及最终用户而编写，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

#### 1.4 版本变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 修改时间 | 修改人员 | 修改内容 | 版本号 |
| 10.16 | 小组四人 | 整合 | V1.0 |
| 12.23 | 张磊 | 修改了member，promotion | V2.0 |
| 12.24 | 朱润之 | 修改了order，增加了login,修改了po | V3.0 |
| 12.24 | 徐天泽 | 修改了room,hotel | V4.0 |
| 12.24 | 张磊 | 增加部分data层代码 | V4.1 |
| 12.24 | 杨凯 | 修改界面跳转 | V4.2 |
| 12.24 | 杨凯 | 修改user接口 | V5.0 |

# 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，将系统分为三层（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界面的实现，业务逻辑层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。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。

# 4.组合视角

## 4.1开发包图

酒店预订系统的最终开发包设计如表1所示。

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他的开发包 |
| mainui | Orderui,Userui,Hotelui,Roomui,Promotionui,Memberui,vo |
| Orderui | 界面类库包，Orderblservice,vo |
| Orderbl | Orderblservice,Orderdataservice, Promotionbl,memberbl |
| Orderblservice |  |
| Orderdata | Databaseutility,po,Orderdataservice, |
| Orderdataservice | Java RMI, po |
| Userui | Userblservice,界面类库包 |
| Userbl | Userblservice,Userdataservice,hotelbl,po |
| Userblservice |  |
| Userdata | Databaseutility,po,Userdataservice, |
| Userdataservice | Java RMI, po |
| Hotelui | 界面类库包，Hotelblservice |
| Hotelbl | Hotelblservice,Hoteldataservice,po,roombl,orderbl |
| Hotelblservice |  |
| Hoteldata | Databaseutility,po,Hoteldataservice, |
| Hoteldataservice | Java RMI, po |
| Roomui | 界面类库包，Roomblservice |
| Roombl | Roomblservice,Roomdataservice,po,hotelbl,orderbl |
| Roomblservice |  |
| Roomdata | Databaseutility,po,Roomdataservice, |
| Roomdataservice | Java RMI, po |
| Promotionui | 界面类库包，Promotionblservice |
| Promotionbl | Promotionblservice,Promotiondataservice,hotelbl,memberbl |
| Promotionblservice |  |
| Promotiondata | Databaseutility,po,Promotiondataservice, |
| Promotiondataservice | Java RMI, po |
| Memberui | 界面类库包，Memberblservice |
| Memberbl | Memberblservice,Memberdataservice,orderbl |
| Memberblservice |  |
| Memberdata | Databaseutility,po,Memberdataservice, |
| Memberdataservice | Java RMI, po |
| Vo |  |
| Po |  |
| Utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

表1

酒店预订系统客户端开发包图如图3所示，服务器端开发包图如图4所示





## 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，客户端进程是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。

Client1

RMI

RMI

Client2

RMI

Client3

Server

RMI

Client4

图5

## 4.3物理部署

酒店预订系统中客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI与应用GUI技术Java Fx属于JDK1.8的一部分，所以，系统需要安装Java运行环境。部署图如图6所示。

<<artifact>>

IDataRemoteSeverice\_Client\_Stub.class

<<artifact>>

server.jar

<<artifact>>

client.jar

Server

ServerNode[4.6]

{OS=Windows7以上}

{JDK=JDK1.8}

RMIStub

Client

ClientNode[4.6]

{OS=Windows7以上}

{JDK=JDK1.8}

# 5.接口视角

## 5.1 模块的职责

客户端和服务器端模块视图以及职责分别如下图所示

启动模块

展示层

启动模块

网络模块

业务逻辑层

数据层

网络模块

图7，客户端模块视图 图8，服务器端模块视图

表2 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 | 基于窗口的连锁商店客户端用户界面 |
| 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javaRMI机制查找RMI服务 |

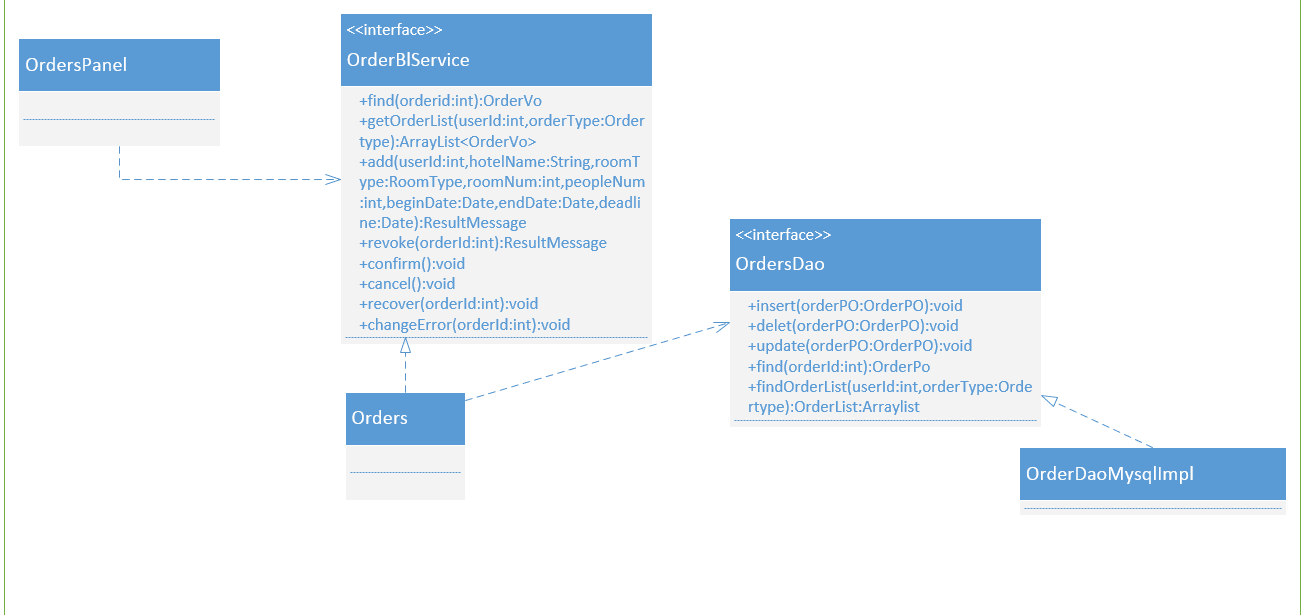
表3 服务器端各层的职责

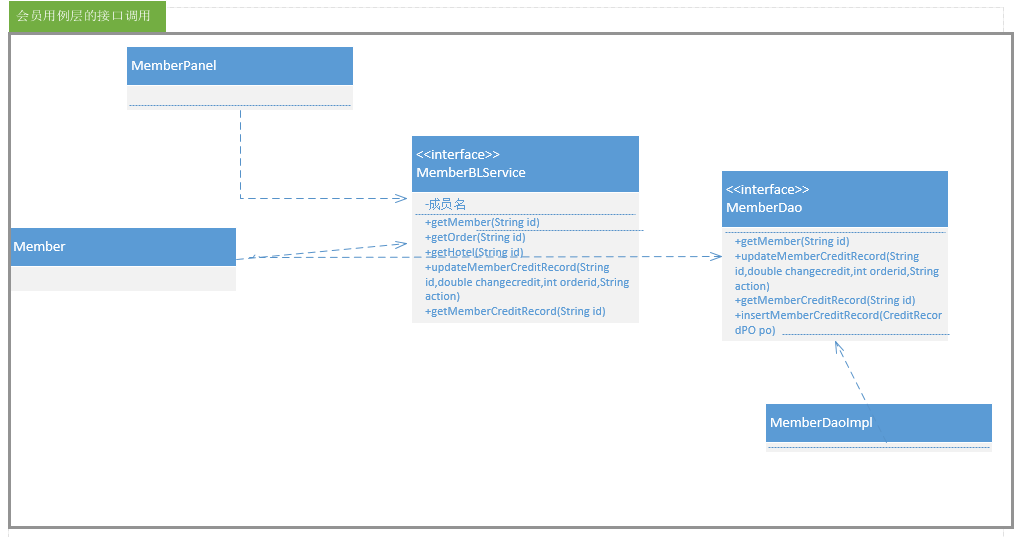
|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用JavaRMI机制开启RMI服务，注册RMI服务 |

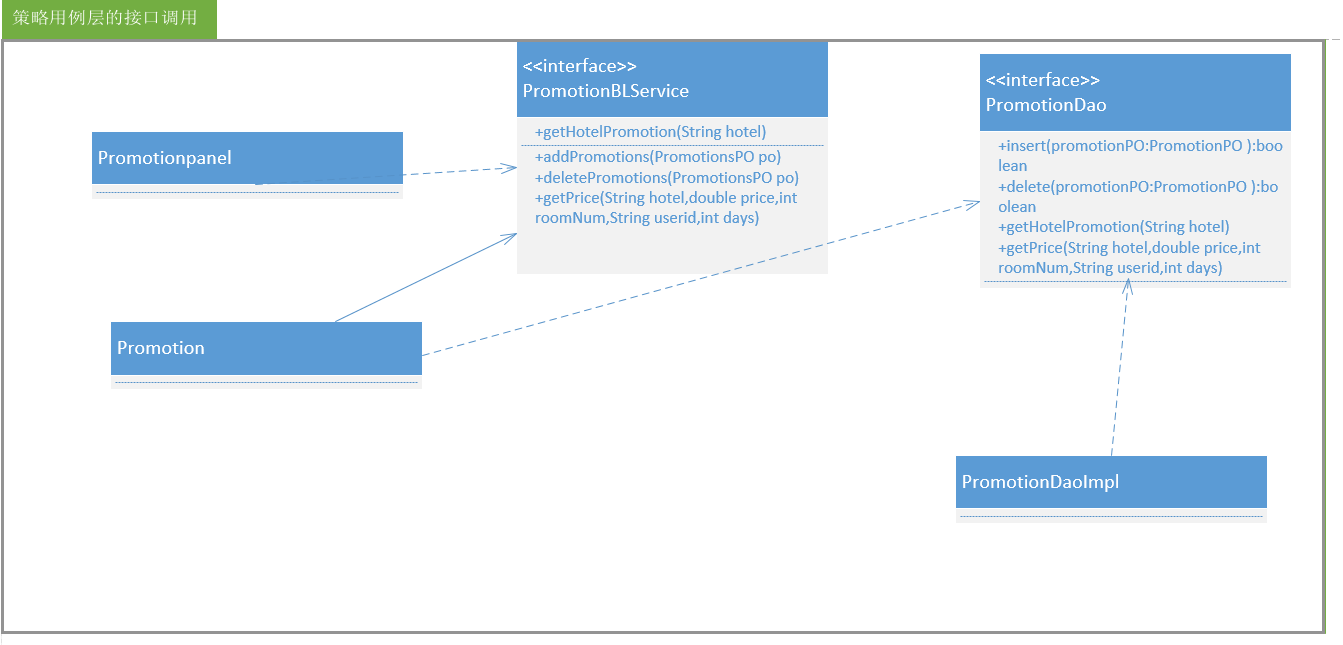
表4 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| OrderBLService  HotelBLService  RoomBLService  PromotionBLService  MemberBLService  LoginBLService  EvaluateBLService | 客户端展示层 | 客户端业务逻辑层 |
| OrderDataService  UserDataService  HotelDataService  RoomDataService  PromotionDataService  MemberDataService  DatabaseFactory | 客户端业务逻辑层 | 服务器端数据层 |

以下图表明了层之间的一个调用，如图所示，每一层之间都是由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。OrdersBlService提供了orders界面所需要的所有业务逻辑功能。OrdersDataService提供了对数据库的增删改查等操作。这样的实现就大大降低了层与层之间的耦合。







**5.2用户界面层的分解**

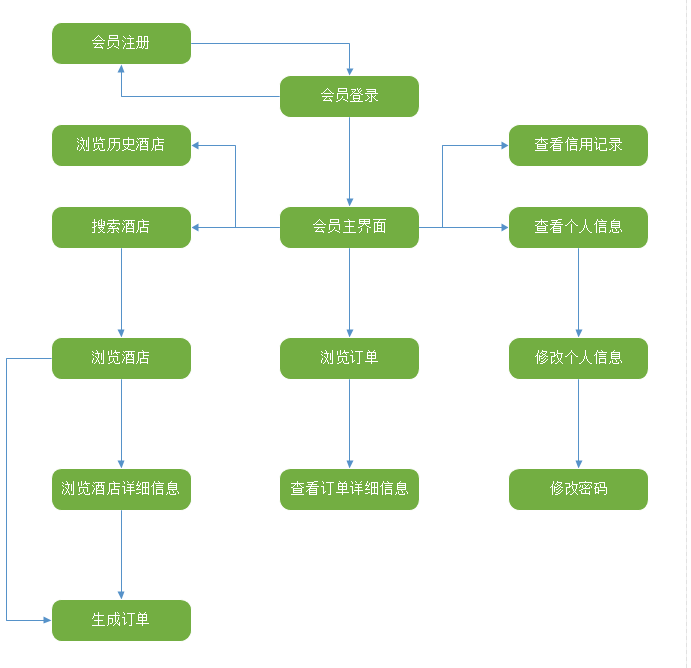
根据需求，系统存在50个界面：会员注册界面，会员登录界面，浏览历史酒店界面，搜索界面，会员主界面，查看信用记录界面，查看那个人信息界面，修改个人信息界面，修改密码界面，浏览酒店，浏览订单界面，浏览酒店详细信息，浏览订单详细信息界面，生成订单界面；

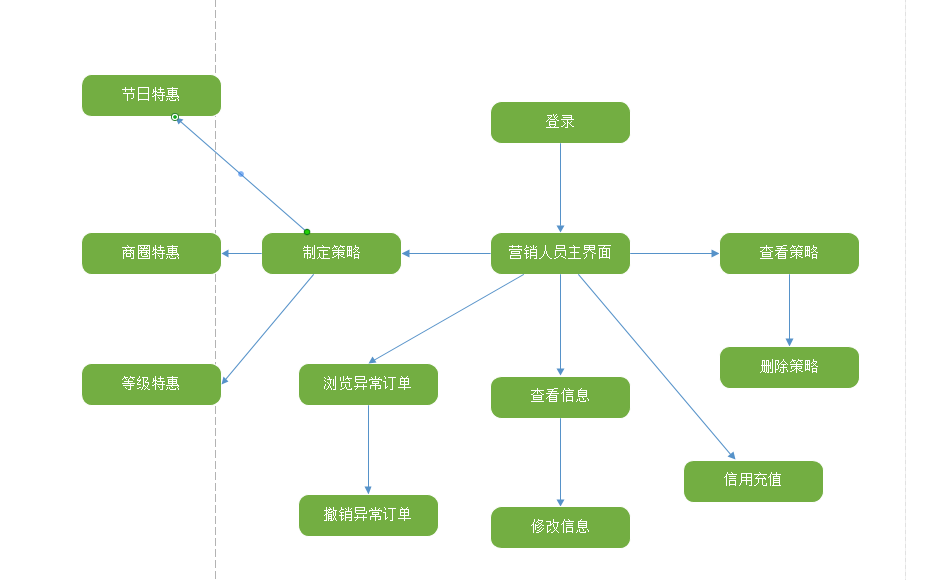
酒店工作人员主界面，酒店工作人员登录界面，浏览异常订单界面，制定策略，查看策略界面，查看订单信息界面，登记入住，添加房间，维护酒店信息，登记退房界面；

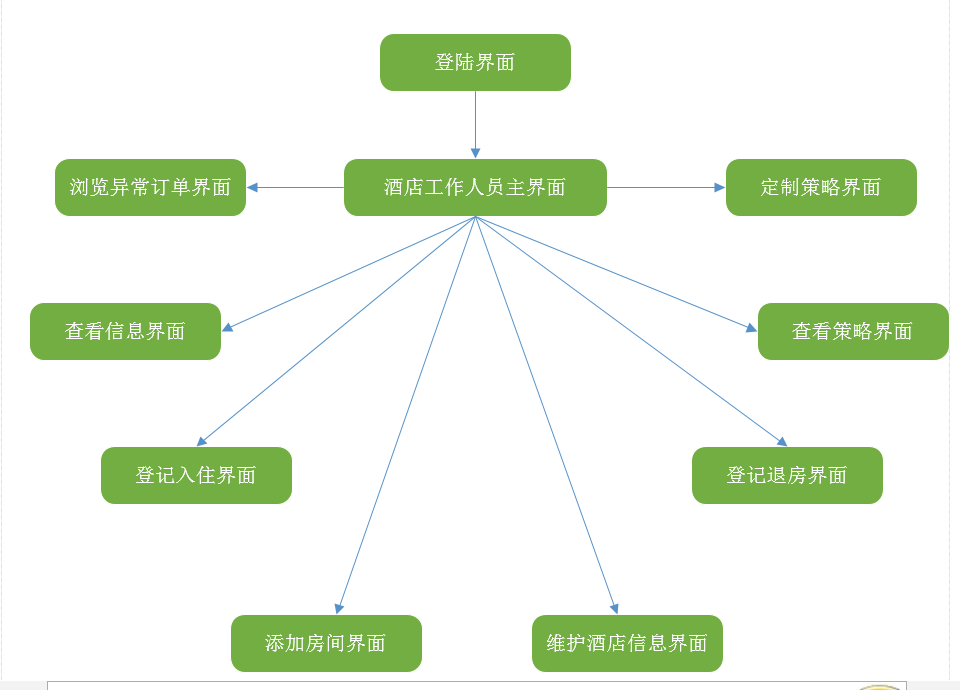
营销人员主界面，营销人员登录界面，制定策略，节日特惠，商圈特惠，等级特惠界面，查看策略，删除策略界面，查看信息，修改信息界面，浏览异常订单，撤销异常订单界面，信用充值界面；

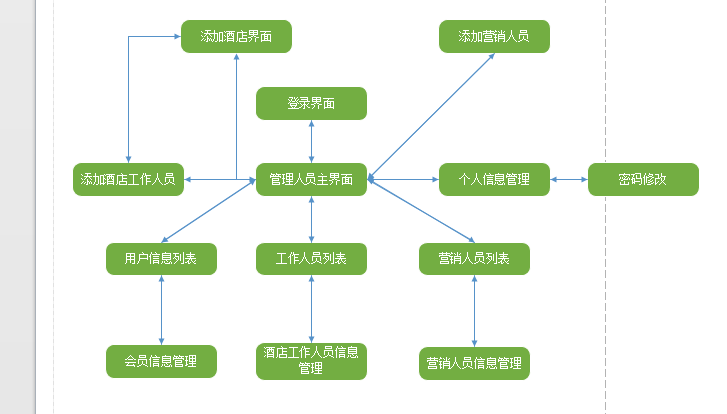
网站管理人员主界面，网站管理人员登录界面，添加酒店工作人员，添加酒店界面，添加营销人员界面，个人信息管理，密码修改界面，用户信息列表，会员信息管理，工作人员列表，酒店工作人员信息管理，营销人员列表，营销人员信息管理界面。

如下图所示

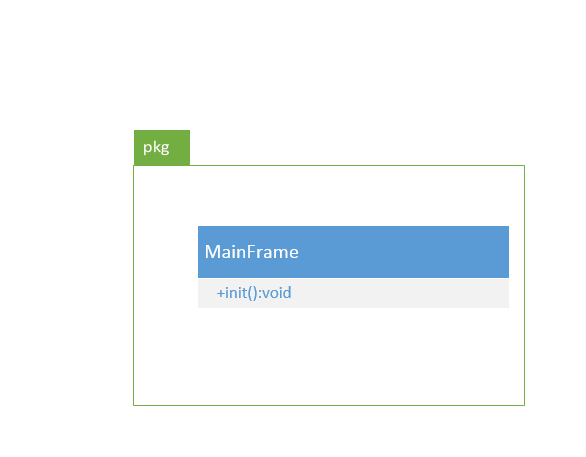








服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如下图所示。



5.2.1用户界面层模块的职责

如下表所示为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如下表所示。

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

用户界面层需要的服务接口如下表所示。

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Businesslogicservice.LoginBLService | 登录界面的业务逻辑接口 |
| Businesslogicservice.\*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javaFx库来实现。

## 5.3业务逻辑层的分解

业务逻辑层包括针对页面的业务逻辑处理对象，设计如下图所示。



### 5.3.1 业务逻辑层模块的职责

业务逻辑层模块的职责如下表所示

表8 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| Promotionbl | 负责实现制定促销策略所需要的服务 |
| Memberbl | 负责实现会员界面所需要的服务 |
| Orderbl | 负责实现订单相关界面所需要的服务 |
| Userbl | 负责实现用户相关界面所需要的服务 |
| Hotelbl | 负责实现酒店相关界面所需要的服务 |
| Roombl | 负责实现房间相关界面所需要的服务 |
| Loginbl | 负责实现登陆界面所需要的服务 |
| Evaluatebl | 负责实现有关评价界面所需要的服务 |

### 5.3.2业务逻辑层的接口规范

每个bl模块的接口规范如下图所示

**表9 Orderbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderService.getPrice | **语法** | **public PriceInfo getPrice(String hotel,double price,int roomNum,String userId,int days)** |
| **前置条件** | **信息输入完整** |
| **后置条件** | **返回计算出的最优价格以及适用的策略** |
| OrderService.completeOrder | **语法** | **public void completeOrder(int orderId,Date outDate)** |
| **前置条件** | **已生成订单，且该订单处于未执行状态** |
| **后置条件** | **将该订单置为已完成，记录当前时间为完成时间** |
| OrderService.getHotelOrder | **语法** | **public ArrayList<OrderVo> getHotelOrder(String hotelName)** |
| **前置条件** | **酒店存在** |
| **后置条件** | **得到酒店的全部订单信息** |
| OrderService.getOrderList | **语法** | **public ArrayList<OrderVo> getOrderList(String memberId,*OrderType* type);** |
| **前置条件** | **用户存在** |
| **后置条件** | **得到指定类型的全部订单列表** |
| OrderService.getUndoList | **语法** | **public ArrayList<OrderVo> getUndoList();** |
| **前置条件** | **存在未执行订单** |
| **后置条件** | **得到未执行订单列表** |
| OrderService.getHotelList | **语法** | **publicArrayList<String> getHotelList(String memberId);** |
| **前置条件** | **用户存在** |
| **后置条件** | **得到用户住过的全部酒店** |
| OrderService.getOrder | **语法** | **public OrderVo getOrder(int orderId)** |
| **前置条件** | **订单号存在** |
| **后置条件** | **根据订单号得到指定订单** |
| OrderService.recordIn | **语法** | **public void recordIn(int orderId,Date inDate)** |
| **前置条件** | **订单存在** |
| **后置条件** | **记录订单入住** |
| OrderService.recordOut | **语法** | **public void recordOut(int orderId,Date outDate)** |
| **前置条件** | **订单存在** |
| **后置条件** | **记录订单退房** |
| OrderService.setType | **语法** | **public void setType(int orderId,*OrderType* type)** |
| **前置条件** | **订单存在** |
| **后置条件** | **改变订单的状态** |
| OrderService.update | **语法** | **public boolean update(OrderVo info)** |
| **前置条件** | **订单存在** |
| **后置条件** | **更新订单信息** |
| OrderService.addpre | **语法** | **public double addPre(OrderVo info)** |
| **前置条件** | **无** |
| **后置条件** | **生成订单预览界面，显示价格等待确认** |
| OrderService.confirmAdd | **语法** | **public void confirmAdd(OrderVo info)** |
| **前置条件** | **addpre已执行** |
| **后置条件** | **将订单添加到数据库，改变房间状态** |
| OrderServicedelete | **语法** | **public void delete(int orderId)** |
| **前置条件** | **订单存在** |
| **后置条件** | **删除订单信息** |
| OrderService.revoke | **语法** | **public double revoke(int orderId)** |
| **前置条件** | **订单存在且为未执行状态** |
| **后置条件** | **将订单置为已撤销，根据时间判断是否扣除信用值** |
| OrderService.errorHandle | **语法** | **public void errorHandle(int orderId);** |
| **前置条件** | **订单存在** |
| **后置条件** | **将订单置为异常，同时扣除信用值** |
| OrderService.checkOrder | **语法** | **public void checkOrder();** |
| **前置条件** | **无** |
| **后置条件** | **检查所有订单，如有超时置为异常订单，扣除信用值** |
| OrderService.recover | **语法** | **public *ResultMessage* recover(int orderId,double recoverPer);** |
| **前置条件** | **订单存在且为异常订单** |
| **后置条件** | **将订单置为已撤销，根据选择恢复信用值** |
| OrderService.getHotelOrderList | **语法** | **public ArrayList<OrderVo> getHotelOrderList(String hotel,String userId,*OrderType* type);** |
| **前置条件** | **酒店名存在** |
| **后置条件** | **根据用户名以及酒店名以及订单状态得到订单列表** |
| OrderService.getOrderHistory | **语法** | **public ArrayList<OrderVo> getOrderHistory(String memberId, String HotelName);** |
| **前置条件** | **用户存在** |
| **后置条件** | **得到用户在该酒店的所有订单** |

|  |  |
| --- | --- |
| 需要的接口 | |
| 服务名 | **服务** |
| OrdersDao.getOrder(int orderId) | **根据id查找订单** |
| OrdersDao.getHotelOrderList | **根据酒店名查找订单** |
| OrdersDao.getOrderList(String userId) | **根据用户名查找订单列表** |
| OrdersDao.insert(OrderPo po) | **插入订单信息** |
| OrdersDao.update(OrderPo po) | **更新订单信息** |
| OrdersDao.delete(int orderId) | **删除指定订单** |
| OrdersDao.getAllOrderList() | **得到所有的订单列表** |
| MembersInfo.getMemberLevel(String memberId) | **根据id查找用户，得到等级** |
| MembersInfo.updateMemberCredit(String id,double changecredit,int orderid,String action); | **更新用户信用** |
| PromotionGetPrice.getPrice(String hotel,double price,int roomNum,String userId,int days); | **根据订单信息计算价格** |
| RoomReservationService.makeReservation(int orderId,String hotelName,*RoomType* roomType,Date startTime,Date endTime,int roomNum); | **为房间记录入住信息** |
| RoomReservationService.revokeReservatio(int orderId); | **删除房间的入住信息** |

**userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **UserService.find** | **语法** | **public UserVOfind(String Id,UserType type);** |
| **前置条件** |  |
| **后置条件** | **根据userType和id返回一个相应的user** |
| **UserService.revoke** | **语法** | **public ResultMessage revoke(String Id, UserType type ,UserVO vo);** |
| **前置条件** | **Vo的id和type和id，type一直** |
| **后置条件** | **修改相应id对应的user的信息** |
| **UserService.addMember** | **语法** | **public ResultMessage addMember(MemberInformationVO user);** |
| **前置条件** |  |
| **后置条件** | **增加新的user到数据库** |
| **UserService.addStaff** | **语法** | **public ResultMessage addStaff(StaffVO user)** |
| **前置条件** |  |
| **后置条件** | **增加新的User到数据库** |
| **UserService.addMarketer** | **语法** | **public ResultMessage addMarketer(MarketerVO user)** |
| **前置条件** | **Manager已经验证登录** |
| **后置条件** | **增加新的user到数据库** |
| **UserService.getAllUsers** | **语法** | **Public ArrayList<UserVO> getAllUsers(UserType type)** |
| **前置条件** |  |
| **后置条件** | **得到所有的相应的User信息** |
| **UserService.isStaffExist** | **语法** | **Public boolean isStaffExist(String HotelName)** |
| **前置条件** | **Hotel存在** |
| **后置条件** | **存在返回True，不存在返回False** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **服务名** | **服务** |
| **UsersDao.findUser(String UserId, UserType type)** | **根据type和id查找一个持久化对象** |
| **UsersDao.updateUser(UsersPO po)** | **更新单一持久化对象，修改User信息** |
| **UsersDao.addUser(UsersPO po)** | **插入单一持久化对象，增加User** |
| **UsersDao.getAllUsers(UserType type)** | **得到所有的用户信息** |
| **UsersDao.isHotelHasStaff(String HotelName)** | **查找酒店是否存在工作人员** |

**表9 Promotionbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **PromotionService.addPromotions** | **语法** | **public boolean addPromotionsI(PromotionsIVO vo)** |
| **前置条件** | **工作人员想要制定策略** |
| **后置条件** | **完成促销策略内容的制定，返回促销策略方案** |
| **PromotionService.deletePromotions** | **语法** | **Public boolean deletePromotions(PromotionsVO promotionsVO);** |
| **前置条件** | **存在促销策略** |
| **后置条件** | **删除已有的促销策略，返回促销策略初始界面** |
| **PromotionService.getPrice** | **语法** | **Public PriceInfo getPrice(String hotel,double price,int roomNum,String userId,int days);** |
| **前置条件** | **得到订单信息** |
| **后置条件** | **返回订单价格** |
| **PromotionServicegetHotelPromotions** | **语法** | **public ArrayList<String> getHotelPromotion(String hotel);** |
|  | **前置条件** | **存在该酒店** |
|  | **后置条件** | **返回酒店策略列表** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **PromotionDao.addPromotions(PromotionPO po)** | **插入新策略** |
| **PromotionDao.deletePromotions(PromotionPO po)** | **删除策略** |
| **PromotionDao.getPrice()** | **得到价格对象** |
| **PromotionsDao.getHotelPromotions()** | **获得酒店策略对象** |

**表10 Memberbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务(供接口)** | | | |
| **MembersService.getOrder** | **语法** | **public ArrayList<OrderVO> getOrder(String id);** | |
| **前置条件** | **Id，符合输入规则,** | |
| **后置条件** | **查找该id下的用户，并返回该用户的历史订单列表** | |
| **MembersService.getHotel** | **语法** | **public Map<String,ArrayList<OrderType>> getHotel(String id);** | |
| **前置条件** | **Id符合规则存在该用户** | |
| **后置条件** | **查找该id下的用户的历史酒店，返回历史酒店信息** | |
| **MembersService.getMember** | **语法** | **Public MemberPO getMember(String id);** | |
| **前置条件** | **用户已登录,且id符合规则** | |
| **后置条件** | **查找此id的用户，返回用户信息** | |
| **MembersService.updateMemberCredit** | **语法** | **Public boolean updateMemberCredit(String id,double changecredit,int ordered,String action);** | |
| **前置条件** | **存在该用户** | |
| **后置条件** | **更新该用户的信用记录** | |
| **MembersService.getMemberCreditRecord** | **语法** | | **Public ArrayLits<CreditrecordPO> getMemberCreditRecord(String memberid)** |
| **前置条件** | | **存在该用户** |
| **后置条件** | | **得到该用户的信用记录** |
| **MembersService.insertCreditRecord** | **语法** | | **public void insertCreditRecord(CreditrecordPO po)** |
| **前置条件** | | **存在该用户** |
| **后置条件** | | **插入该用户的信用记录** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **Ordersblservice.getOrder** | **得到历史订单** |
| **Ordersblservice.getHotel** | **得到历史酒店** |
| **Membersdao.getMember** | **得到member信息** |
| **Membersdao.getCreditRecord** | **得到用户的CreditRecord** |
| **Membersdao.updateCreditRecord** | **更新数据库中信用记录** |

**Hotelbl层**的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelService. getHotelListInfo | 语法 | **public** ArrayList<HotelColumnVo> getHotelListInfo(HotelSearchVo hotelSearchVo); |
| 前置条件 | 已添加酒店所属商圈、地址以及酒店名称、房间、星级或评分区间 |
| 后置条件 | 返回符合条件的酒店列表 |
| HotelService. getHotelInfo | 语法 | **public** HotelVo getHotelInfo(String hotelName); |
| 前置条件 | 已添加酒店名称 |
| 后置条件 | 返回酒店的**地址，简介，设施服务，客房类型,历史订单情况** |
| HotelService. addHotelEvaluation | 语法 | public ResultMessage evaluateHotel(string evaluation,string HotelName); |
| 前置条件 | 已添加对酒店的评价信息 |
| 后置条件 | 在一个评论回合中，增加会员的评价信息 |
| HotelService. updateHotel | 语法 | **public** **boolean** updateHotel(HotelVo hotelVo); |
| 前置条件 | 已添加更新后的酒店信息 |
| 后置条件 | 在一个更新回合中，添加酒店的更新信息 |
| HotelService. addHotel | 语法 | public ResultMessage addHotel(HotelVO h); |
| 前置条件 | 已添加酒店信息 |
| 后置条件 | 在一个添加回合中，返回添加是否成功的信息 |
| HotelService. getRoomOfHotel | 语法 | **public** ArrayList<RoomVo> getRoomOfHotel(String hotelName,Date startTime,Date endTime); |
| 前置条件 | 已添加酒店名称 |
| 后置条件 | 在一个添加回合中，返回酒店酒店相关信息 |
| HotelService.getHotelComment | 语法 | **public ArrayList<String> getHotelComment(String hotelName);** |
| 前置条件 | 已添加评价信息 |
| 后置条件 | 在一个添加回合中，返回添加是否成功的信息 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| HotelDao.getHotelList  (**HotelPo hotelPo)** | 根据输入的酒店信息查找多个持久化对象 |
| HotelDao.judgeHotelExists  (String hotelName) | 根据输入的酒店名称判断酒店是否存在 |
| HotelDao.getHotelDetails  (String hotelName) | 根据输入的酒店名称得到酒店的细节信息 |
| HotelDao.insert  (HotelPo h) | 插入单一持久化对象 |
| HotelDao.update (HotelPo h) | 更新单一持久化对象 |
| RoomService.getHotelRoomType  (String hotelName,Date startTime,Date endTime) | 得到该酒店在某一时段的房间类型 |
| RoomService.getNumOfRoom  (String hotelName,Date startTime,Date endTime) | 得到该酒店在某一时段的房间数量 |
| OrderService.getHotelList  (string UserID,String hotelName) | 得到某一会员的酒店预订历史记录 |
| EvaluateService.getComments(String hotelName) | 得到某一酒店的分数 |

**Roombl层**的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomService.addNewRoom | 语法 | **public** **void** addNewRoom(ArrayList<RoomVo> roomList); |
| 前置条件 | 已添加房间的类型、数量、原始价格 |
| 后置条件 | 在一个添加回合中，增加房间信息 |
| RoomService. getRoomOfHotel | 语法 | **public** ArrayList<RoomVo> getRoomOfHotel(String hotelName,Date startTime,Date endTime); |
| 前置条件 | 已添加酒店名称 起始时间 退房时间 |
| 后置条件 | 在一个查询回合中，返回符合条件的房间列表 |
| RoomService.makeReservation | 语法 | public void makeReservation(int orderId, String hotelName, RoomType roomType, Date startTime, Date endTime,  int roomNum) |
| 前置条件 | 已添加订单号 酒店名称 房间类型  房间数目 起始时间 退房时间 |
| 后置条件 | 在一个记录回合中，对房间进行预订 |
| RoomService.makeCheckIn | 语法 | public void makeCheckIn(int orderId, Date startTime) |
| 前置条件 | 已添加订单号 入住日期 |
| 后置条件 | 在一个记录回合中，添加入住信息 |
| RoomService.makeCheckOut | 语法 | public void makeCheckOut(int orderId, int roomId, Date endTime) |
| 前置条件 | 已添加订单号 房间号 退房日期 |
| 后置条件 | 在一个记录回合中，添加退房信息 |
| RoomService.numberOfAllRooms | 语法 | public int numberOfAllRooms(String hotelName, Date startTime, Date endTime) |
| 前置条件 | 已添加酒店名称 入住日期 退房日期 |
| 后置条件 | 在一个查询回合中，返回房间总数 |
| RoomService.getHotelRoomType | 语法 | public ArrayList<RoomType> getHotelRoomType(String hotelName, Date startTime, Date endTime) |
| 前置条件 | 已添加酒店名称 入住日期 退房日期 |
| 后置条件 | 在一个查询回合中，返回酒店的可用房间类型列表 |
| RoomService.getPriceOfRoom | 语法 | public Map<RoomType, Double> getPriceOfRoom(String hotelName, Date startTime, Date endTime) |
| 前置条件 | 已添加酒店名称 入住日期 退房日期 |
| 后置条件 | 在一个查询回合中，返回酒店房间对应的价格 |
| RoomService. getNumOfRoom | 语法 | public Map<RoomType, Integer> getNumOfRoom(String hotelName, Date startTime, Date endTime) |
| 前置条件 | 已添加酒店名称 入住日期 退房日期 |
| 后置条件 | 在一个查询回合中，返回酒店房间对应的数量 |
| RoomService. revokeReservation | 语法 | public void revokeReservatio(int orderId) |
| 前置条件 | 已添加订单号 |
| 后置条件 | 在一个记录回合中，取消房间的预定 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| RoomsDao.getRoomList(String hotelName) | 根据输入的酒店名称返回多个房间持久化对象 |
| RoomsDao.insertRoom ArrayList<RoomPO> roomList | 插入多个持久化对象 |
| DataFactoryService.get  RoomDataService | 得到房间数据库的服务的引用 |
| roomsDao.recordReservation(int RoomID, Date startTime, Date endTime, int orderId); | 维护一个单一持久化对象的某一字段  将一次入住的预计入住时间/预计退房时间/对应订单号  记载在相应房间数据库中 |
| roomsDao.recordOrderRoom(int orderId, ArrayList<Integer> roomList) | 插入一个单一持久化对象  将订单对应的一组房间号记载在数据库中 |
| roomsDao.recordCheckIn(int roomId, int orderId, Date startTime); | 维护一个单一持久化对象的某一字段  将一次登记入住的实际时间记载在数据库中 |
| roomsDao.recordCheckOut(int roomId, int orderId); | 维护一个单一持久化对象的某一字段  删除某次已结束入住的入住时间/订单号 |
| ordersService.recordIn(int orderId, Date startTime); | 将一次入住信息记载在订单中 |
| ordersService.recordOut(int orderId, Date endTime) | 将一次退房信息记载在订单中 |

Login的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginService.login | **语法** | **public ResultMessage login(String id,String password, UserType type)** |
| **前置条件** | **id存在，输入符合规则** |
| **后置条件** | **进行登录操作，成功返回true，失败返回失败信息** |
| LoginService.logout | **语法** | **public void logout(String id)** |
| **前置条件** | **已执行login** |
| **后置条件** | **进行登出操作，状态变为登出** |
| LoginService.getMessage | **语法** | **public Loginpo getMessage(Sting id)** |
| **前置条件** | **已执行login，并且未执行logout方法** |
| **后置条件** | **查找并返回对应id的loginpo对象** |

|  |  |
| --- | --- |
| 需要的接口 | |
| 服务名 | **服务** |
| User.getUser（String id） | **得到对应的userpo对象信息** |

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，包括对于持久化数据的增、删、改、查。Orders业务逻辑需要的服务由OrdersDao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、数据库等，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

数据层模块的职责如下表所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| OrdersDao | 持久化数据库的接口，提供集体裁入、集体保存、增、删、改、查服务 |
| OrdersDaoImpl | 数据层的实现，实现OrdersDao接口 |
| OrdersDataHelper | 提供具体数据交互的接口 |
| OrdersDataSqlHelperImpl | 基于sql数据库实现数据交互的具体实现 |

5.4.2数据层模块的接口规范

数据层模块的接口规范如下表所示。

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrdersDao.getOrder | 语法 | **public** **OrderPO** **getOrder**(**int** orderId) **throws** **RemoteException**; |
| 前置条件 | 订单号存在，数据库已连接 |
| 后置条件 | 根据订单号得到订单 |
| OrdersDao.getHotelOrderList | 语法 | **public** **ArrayList**<OrderPO> **getHotelOrderList**(**String** hotelName) **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 根据酒店名，得到酒店的订单列表 |
| OrdersDao.getOrderList | 语法 | **public** **ArrayList**<OrderPO> **getOrderList**(**String** userId) **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 得到指定用户的所有订单 |
| OrdersDao.insert | 语法 | **public** **boolean** **insert**(**OrderPO** po) **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 插入一个orderpo |
| OrdersDao.update | 语法 | **public** **boolean** **updata**(**OrderPO** po) **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 更新一个orderpo |
| OrdersDao.getAllOrderList | 语法 | **public** **ArrayList**<OrderPO> **getAllOrderList**() **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 得到全部订单 |
| OrdersDao.delete | 语法 | **public** **boolean** **delete**(**int** orderId) **throws** **RemoteException**; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 删除一个指定订单 |

表 PromotionDao模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionsDao.addPromotionsI | 语法 | public boolean addPromotionsI(PromotionsIPO po) |
| 前置条件 | 工作人员想要制定策略 |
| 后置条件 | 完成促销策略内容的制定，返回促销策略方案 |
| PromotionsDao.deletePromotions | 语法 | Public boolean deletePromotions(PromotionsPO promotionsPO); |
| 前置条件 | 存在促销策略 |
| 后置条件 | 删除已有的促销策略，返回促销策略初始界面 |
| PromotionsDao.getHotelPromotions | 语法 | public ArrayList<PromotionsPO> getHotelPromotion(String hotel); |
|  | 前置条件 | 存在该酒店 |
|  | 后置条件 | 返回酒店策略列表 |

表 MemberDao模块的接口

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| MembersDao.getMemberCreditRecord | 语法 | public ArrayLits<CredittrecordPO> getMemberCreditRecord(String memberid) |
| 前置条件 | Id，符合输入规则, |
| 后置条件 | 查找该id下的用户，并返回该用户的历史信用记录 |
| MembersDao.insertCreditRecord | 语法 | public void insertCreditRecord(CreditrecordPO po) |
| 前置条件 | 存在数据库 |
| 后置条件 | 插入用户信用记录 |
| MembersDao.getMember | 语法 | Public MemberPO getMember(String id); |
| 前置条件 | 用户已登录,且id符合规则 |
| 后置条件 | 查找此id的用户，返回用户信息 |
| MembersDao.updateCredit | 语法 | Public boolean updateCredit(String id,double changecredit); |
| 前置条件 | 存在该用户 |
| 后置条件 | 更新该用户的信用记录 |

**1Hotel数据层模块的职责**

Hotel数据层模块的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| HotelsDao | 持久化数据库的接口，提供集体裁入、集体保存、增、删、改、查服务 |
| HotelsDaoImpl | 数据层的实现，实现HotelsDao接口 |
| HotelsDataHelper | 提供具体数据交互的接口 |
| HotelsDataSqlHelperImpl | 基于sql数据库实现数据交互的具体实现 |

**2数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelsDao.getHotelList | 语法 | **public ArrayList<HotelPO> getHotelList(HotelPO hotelPO) throws RemoteException;** |
| 前置条件 | 数据库已连接 |
| 后置条件 | 根据酒店信息得到符合条件的酒店列表 |
| HotelsDao. getHotelDetails | 语法 | **public** HotelPO getHotelDetails(String hotelName) **throws** RemoteException; |
| 前置条件 | 数据库已连接，酒店名存在 |
| 后置条件 | 根据酒店名得到酒店细节信息 |
| HotelsDao. insertHotel | 语法 | **public** **boolean** insertHotel(HotelPO hotelPO) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 向数据库中插入持久化对象 |
| HotelsDao. deleteHotel | 语法 | **public** **boolean** deleteHotel(String hotelName) **throws** RemoteException; |
| 前置条件 | 数据库已连接，酒店存在 |
| 后置条件 | 在数据库中删除持久化对象 |
| HotelsDao. updateHotel | 语法 | **public** **boolean** updateHotel(HotelPO hotelPO) **throws** RemoteException; |
| 前置条件 | 数据库已连接，酒店存在 |
| 后置条件 | 在数据库中更新持久化对象 |
| HotelsDao. getRoomOfHotel | 语法 | **public** ArrayList<RoomPO> getRoomOfHotel(String hotelName) **throws** RemoteException; |
| 前置条件 | 数据库已连接，酒店存在 |
| 后置条件 | 根据酒店名返回相应的房间 |
| HotelsDao. judgeHotelExists | 语法 | **public** **boolean** judgeHotelExists(String hotelName) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 根据酒店名判断酒店是否存在 |

**1Room数据层模块的职责**

Room数据层模块的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| RoomsDao | 持久化数据库的接口，提供集体裁入、集体保存、增、删、改、查服务 |
| RoomsDaoImpl | 数据层的实现，实现RoomsDao接口 |
| RoomsDataHelper | 提供具体数据交互的接口 |
| RoomsDataSqlHelperImpl | 基于sql数据库实现数据交互的具体实现 |

**2Room数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomsDao. getRoomList | 语法 | **public ArrayList<RoomPO> getRoomList(String hotelName) throws RemoteException;** |
| 前置条件 | 数据库已连接，酒店名存在 |
| 后置条件 | 根据酒店名得到符合条件的房间列表 |
| RoomsDao. insertRoom | 语法 | **public** **boolean** insertRoom(RoomPO roomPO) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 向数据库中插入持久化对象 |
| RoomsDao. recordReservation | 语法 | **public** **boolean** recordReservation(**int** roomID, Date StartTime,Date EndTime, **int** orderId) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 记录一次预定的起始 结束时间和相应的订单号 |
| RoomsDao. recordCheckOut | 语法 | **public** **boolean** recordCheckOut(**int** roomID, **int** orderId) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 记录一次退房的起始 结束时间和相应的订单号 |
| RoomsDao. getOrderRoom | 语法 | **Public** ArrayList<Integer> getOrderRoom(**int** orderId) **throws** RemoteException; |
| 前置条件 | 数据库已连接，订单号存在 |
| 后置条件 | 得到一张订单对应的房间号 |
| RoomsDao. recordOrderRoom | 语法 | **public** **boolean** recordOrderRoom(**int** orderId, ArrayList<Integer> roomIdList) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 记录一次订单与其对应的房间号 |
| RoomsDao. recordCheckIn | 语法 | **public** **boolean** recordCheckIn(**int** roomID, **int** orderId, Date StartTime) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 记录一次入住的实际起始 |

**User数据层模块的职责**

User数据层模块的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| UsersDao | 持久化数据库的接口，提供集体裁入、集体保存、增、删、改、查服务 |
| UsersDaoImpl | 数据层的实现，实现UsersDao接口 |
| UsersDataHelper | 提供具体数据交互的接口 |
| UsersDataSqlHelperImpl | 基于sql数据库实现数据交互的具体实现 |

**数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UsersDao.getAllUsers | 语法 | **public ArrayList<UserPO> getUserList(UserType type) throws RemoteException;** |
| 前置条件 | 数据库已连接 |
| 后置条件 | 得到这种类型的所有用户 |
| UsersDao. findUser | 语法 | **public** UserPO findUser(String Userid UserType type) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 根据用户id和type得到相应的UserPO |
| UsersDao. updateUser | 语法 | **public** **ResultMessage** updateUser (UserPO UserPO )**throws** RemoteException; |
| 前置条件 | 数据库已连接，Id和Type和userPO相一致 |
| 后置条件 | 修改数据 |
| UsersDao. addUser | 语法 | **public ResultMessage** addUser UserPO UserPO) **throws** RemoteException; |
| 前置条件 | 数据库已连接 |
| 后置条件 | 在数据库中增加持久化对象 |
| UsersDao. isHotelHasStaff | 语法 | **public** **boolean** isHotelHasStaff (String HotelName) **throws** RemoteException; |
| 前置条件 | 数据库已连接，酒店存在 |
| 后置条件 | 返回酒店是否存在工作人员 |

**Evaluate数据层模块的职责**

Evaluate数据层模块的职责如下表所示

|  |  |
| --- | --- |
| **模块** | **职责** |
| EvaluatesDao | 持久化数据库的接口，提供集体裁入、集体保存、增、删、改、查服务 |
| EvaluatesDaoImpl | 数据层的实现，实现EvaluatesDao接口 |
| EvaluatesDataHelper | 提供具体数据交互的接口 |
| EvaluatesDataSqlHelperImpl | 基于sql数据库实现数据交互的具体实现 |

**数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| EvaluatesDao.addEvaluate | 语法 | **public ResultMessage addEvauate(EvaluatePO po)throws RemoteException;** |
| 前置条件 | 数据库已连接 |
| 后置条件 | 向数据库增加可持久化对象 |
| EvaluatesDao. getScore | 语法 | **public** double getScore(String HotelName)**throws** RemoteException; |
| 前置条件 | 数据库已连接,酒店存在 |
| 后置条件 | 得到酒店的评分 |
| EvaluatesDao. getComments | 语法 | **Public** ArrayList<String> getComments(String HotelNmae)**throws** RemoteException; |
| 前置条件 | 数据库已连接,酒店存在 |
| 后置条件 | 得到酒店的评分 |

# 6 信息视角

## 6.1数据持久化对象

系统的po类就是对应的相关的实体类，在此只做简单的介绍。

OrderPo包含订单编号，开始时间，退房时间，实际离店时间，最晚执行时间，用户名，用户编号，订单状态，酒店名，房间类型，房间数量，入住人数

UserPo包含用户名，ID，密码，预订过的酒店，角色

HotelPo包含酒店名，地址，商圈，介绍，设施和服务，星级

RoomPo包含房间id，房间所属，房间类型，介绍，价格

PromotionPo包含酒店，种类，促销人员

MemberPo包含id,密码，名字，手机号

LoginPo 包含用户id，用户密码，用户类型

持久化对象的定义依次如下图所示

OrderPo的定义

EvaluatePo包含酒店名，评分，具体评价，用户名

|  |
| --- |
| **package** po;  **import** java.io.Serializable;  **import** java.util.Date;  **import** RoomService.RoomType;  **import** ordersService.OrderType;  **public** **class** OrderPO **implements** Serializable{  /\*\*  \*  \*/  **private** **static** **final** **long** serialVersionUID = -4144960617559928505L;  **int** orderId;  String userId;  String name;  **int** peopleNum;  String hotelNameString;  RoomType roomType;  **int** roomNum;  **double** price;  OrderType orderType;  Date beginDate; //订单开始时间  Date inDate; //入住时间  Date outDate; //离店时间  Date completeDate; //订单完成时间  Date deadLine; //最晚执行时间  **boolean** hasChild;  **public** OrderPO(**int** orderId, String userId, String name,  String hotelNameString, RoomType roomType, **int** roomNum,  **double** price, OrderType orderType, Date inDate, Date outDate,  Date completeDate, Date deadLine,**int** peopleNum,Date beginDate,**boolean** hasChild) {  **super**();  **this**.orderId = orderId;  **this**.userId = userId;  **this**.name = name;  **this**.hotelNameString = hotelNameString;  **this**.roomType = roomType;  **this**.roomNum = roomNum;  **this**.price = price;  **this**.orderType = orderType;  **this**.inDate = inDate;  **this**.outDate = outDate;  **this**.completeDate = completeDate;  **this**.deadLine = deadLine;  **this**.peopleNum = peopleNum;  **this**.beginDate = beginDate;  }  **public** **void** setOrderId(**int** orderId) {  **this**.orderId = orderId;  }  **public** **void** setUserId(String userId) {  **this**.userId = userId;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **void** setPeopleNum(**int** peopleNum) {  **this**.peopleNum = peopleNum;  }  **public** **void** setHotelNameString(String hotelNameString) {  **this**.hotelNameString = hotelNameString;  }  **public** **void** setRoomType(RoomType roomType) {  **this**.roomType = roomType;  }  **public** **void** setRoomNum(**int** roomNum) {  **this**.roomNum = roomNum;  }  **public** **void** setPrice(**double** price) {  **this**.price = price;  }  **public** **void** setBeginDate(Date beginDate) {  **this**.beginDate = beginDate;  }  **public** **void** setInDate(Date inDate) {  **this**.inDate = inDate;  }  **public** **void** setOutDate(Date outDate) {  **this**.outDate = outDate;  }  **public** **void** setCompleteDate(Date completeDate) {  **this**.completeDate = completeDate;  }  **public** **void** setDeadLine(Date deadLine) {  **this**.deadLine = deadLine;  }  **public** **void** setHasChild(**boolean** hasChild) {  **this**.hasChild = hasChild;  }  **public** **void** setOrderType(OrderType orderType) {  **this**.orderType = orderType;  }  **public** String getName(){  **return** name;  }  **public** Date getBeginDate(){  **return** beginDate;  }  **public** **int** getPeopleNum(){  **return** peopleNum;  }  **public** **int** getOrderId() {  **return** orderId;  }  **public** String getUserId(){  **return** userId;  }    **public** String getHotelNameString() {  **return** hotelNameString;  }  **public** RoomType getRoomType(){  **return** roomType;  }  **public** **int** getRoomNum(){  **return** roomNum;  }  **public** **double** getPrice(){  **return** price;  }  **public** OrderType getOrderType(){  **return** orderType;  }  **public** Date getInDate(){  **return** inDate;  }  **public** Date getOutDate(){  **return** outDate;  }  **public** Date getCompleteDate(){  **return** completeDate;  }    **public** Date getDeadLine(){  **return** deadLine;  }  **public** **boolean** isHasChild(){  **return** hasChild;  }  } |

MemberPo的定义

|  |
| --- |
| package po;  import java.io.Serializable;  import MembersService.MembersInfo;  public class MemberPO implements Serializable,MembersInfo{  /\*\*  \*  \*/  private static final long serialVersionUID = 1678038984150126521L;  String id;  String password;  String name;  String telephone;  String special;  double credit;  int type;  int level;  public MemberPO(String id,String name,String password,String telephone,double credit,String special,int type,int level){  this.id=id;  this.name=name;  this.password=password;  this.telephone=telephone;  this.credit =credit;  this.special=special;  this.type=type;  this.level=level;  }  public double getCredit(){  return credit;  }  public String getName(){  return name;  }  public String getID(){  return id;  }  public String getPassword(){  return password;  }  public String getTelephone(){  return telephone;  }  public String getSpecial(){  return special;  }  public int getType(){  return type;  }  public int getLevel(){  return level;  }  } |

UserPo的定义

|  |
| --- |
| package po;  import java.io.Serializable;  import java.util.Date;  import UsersService.UserInfo;  import UsersService.UserType;  public class UserPO implements Serializable{  /\*\*  \*  \*/  private static final long serialVersionUID = -2144004342974484005L;  String id;  String name;  String password;  // String hotel;  UserType role;  // Date birthday;  public UserPO(String i,String n,String p ,UserType r){  this.id=i;  this.name=n;  this.password=p;  this.role=r;  // this.hotel=hotel;  // this.birthday=birthday;  }    public String getId() {  return id;  }  public String getName() {  return name;  }  public String getPassword() {  return password;  }  public UserType getRole() {  return this.role;  }  public String getHotel() {  // TODO Auto-generated method stub  return null;  }    } |

EvaluatePo的定义

|  |
| --- |
| **package** po;  **import** java.io.Serializable;  **public** **class** **EvaluatePO** **implements** Serializable{    /\*\*  \*  \*/  **private** **static** **final** **long** ***serialVersionUID*** = 8811457577381021855L;  **private** **String** hotelName;  **private** **double** score;  **private** **String** s;  **private** **String** memberName;    **public** **EvaluatePO**(**String** hotelName,**double** score,**String** s,**String** memberName){  **this**.hotelName=hotelName;  **this**.score=score;  **this**.s=s;  **this**.memberName=memberName;  }    **public** **String** **getHotelName**() {  **return** hotelName;  }    **public** **String** **getName**(){  **return** memberName;  }    **public** **double** **getScore**() {  **return** score;  }    **public** **String** **getS**() {  **return** s;  }    } |

CreditRecordPo的定义

|  |
| --- |
| **package** po;  **import** java.io.Serializable;  **public** **class** **CreditrecordPO** **implements** Serializable{  **String** memberid;  **String** time;  **int** recordid;  **String** action;  **double** creditchange;  **double** creditresult;  **public** **String** **getMemberid**(){  **return** memberid;  }  **public** **String** **getTime**(){  **return** time;  }  **public** **int** **getRecordid**(){  **return** recordid;  }  **public** **String** **getAction**(){  **return** action;  }  **public** **double** **getCreditchange**(){  **return** creditchange;  }  **public** **double** **getCreditresult**(){  **return** creditresult;  }  **public** **CreditrecordPO**(**String** memberid,**String** time,**int** recordid,**String** action,**double** creditchange,**double** creditresult){  **this**.memberid=memberid;  **this**.time=time;  **this**.recordid=recordid;  **this**.action=action;  **this**.creditchange=creditchange;  **this**.creditresult=creditresult;    }  } |

LoginPo的定义

|  |
| --- |
| package po;  import java.io.Serializable;  import UsersService.UserInfo;  import UsersService.UserType;  public class LoginPO implements Serializable{  /\*\*  \*  \*/  private static final long serialVersionUID = -1443162645998603682L;  private String ID;  private String password;  private UserType usersType;  private UserInfo info;  public LoginPO(String iD, String password, UserType usersType, UserInfo info) {  ID = iD;  this.password = password;  this.usersType = usersType;  this.info = info;  }    } |

HotelPo的定义

|  |
| --- |
| package po;  import java.io.Serializable;  import HotelService.HotelRanking;  import HotelService.HotelTradeArea;  import HotelService.HotelsInfo;  public class HotelPO implements Serializable,HotelsInfo{  /\*\*  \*  \*/  private static final long serialVersionUID = -7595262996881300104L;  private String name;  private String address;  private HotelTradeArea tradeArea;  private String introduction;  private String serviceAndFacility;  private HotelRanking ranking;    public HotelPO(String n, String a, HotelTradeArea t,String i,String s, HotelRanking r ){  name = n;  address = a;  tradeArea = t;  introduction = i;  serviceAndFacility = s;  ranking = r;  }  public String getName(){  return name;  }  public String getAddress(){  return address;  }  public HotelTradeArea getTradeArea(){  return tradeArea;  }  public String getIntroduction(){  return introduction;  }  public String getServiceAndFacility(){  return serviceAndFacility;  }  public HotelRanking getRanking(){  return ranking;  }  } |

RoomPo的定义

|  |
| --- |
| package po;  import java.io.Serializable;  import java.util.Date;  import java.util.Map;  import RoomService.RoomType;  import RoomService.RoomsInfo;  public class RoomPO implements Serializable,RoomsInfo{  /\*\*  \*  \*/  private static final long serialVersionUID = 3336011436313431937L;  private int roomId;  private String hotelBelongTo;  private RoomType roomtype;  private String introdution;  private double price;  private Map<Date, Date> unavailablePeriod;      public RoomPO(int roomId, String hotelBelongTo, RoomType roomtype,String introdution,double price, Map<Date, Date> unavailablePeriod){  this.roomId= roomId;  this.hotelBelongTo =hotelBelongTo;  this.roomtype =roomtype;  this.introdution=introdution;  this.price =price;  this.unavailablePeriod =unavailablePeriod;  }    public int getId(){  return roomId;  }  public String getHotelBelongTo(){  return hotelBelongTo;  }  public RoomType getRoomType(){  return roomtype;  }  public String getIntroduction(){  return introdution;  }  public double getPrice(){  return price;  }  public Map<Date, Date> getUnavailablePeriod() {  return unavailablePeriod;  }  } |

PromotionsPo的定义

|  |
| --- |
| **package** po;  **import** java.io.Serializable;  **public** **class** **PromotionsPO** **implements** Serializable{  /\*\*  \*  \*/  **private** **static** **final** **long** ***serialVersionUID*** = -954887827818674317L;  **public** **int** type;  **public** **String** hotel;  **public** **double** discount;  **public** **String** introduction;  **public** **int** **getType**(){  **return** type;  }  **public** **String** **getHotel**(){  **return** hotel;  }  **public** **double** **getDiscount**(){  **return** discount;  }  **public** **String** **getIntroduction**(){  **return** introduction;  }  **public** **void** **setType**(**int** type) {  **this**.type = type;  }  **public** **void** **setHotel**(**String** hotel) {  **this**.hotel = hotel;  }  **public** **void** **setDiscount**(**double** discount) {  **this**.discount = discount;  }  **public** **void** **setIntroduction**(**String** introduction) {  **this**.introduction = introduction;  }  } |

## 6.2 数据库表

数据库表如下图所示

