

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

# 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 |

# 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。

RMI

Client1

RMI

Client2

Server

RMI

Client3

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 |

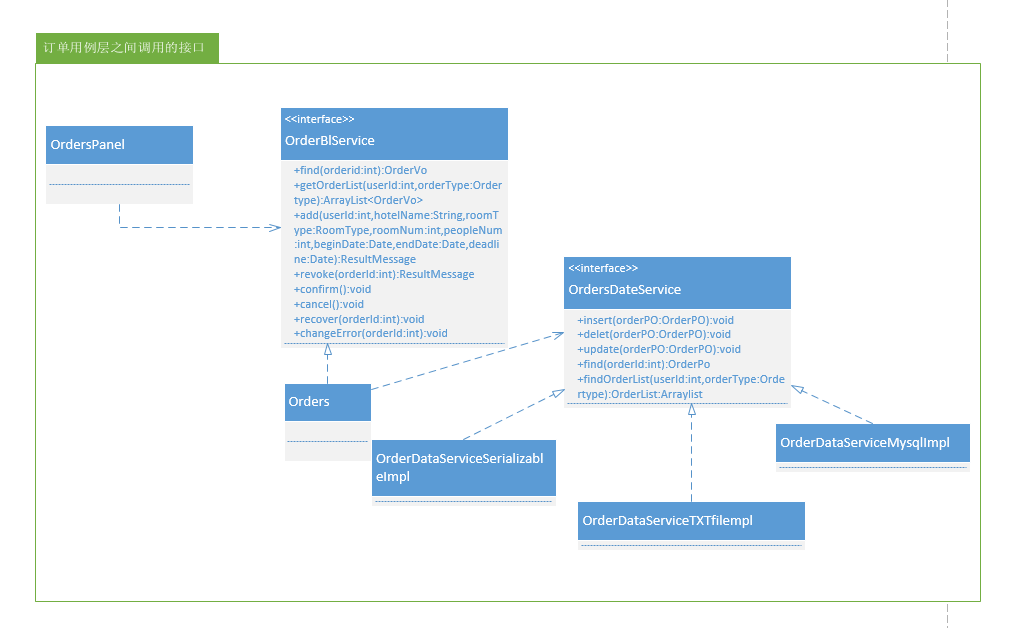
表3 服务器端各层的职责

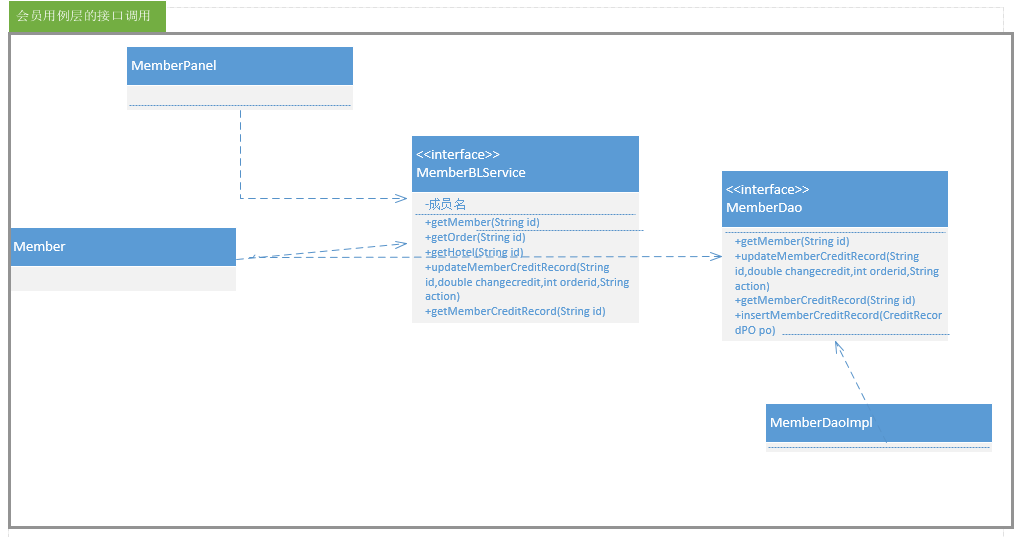
|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用JavaRMI机制开启RMI服务，注册RMI服务 |

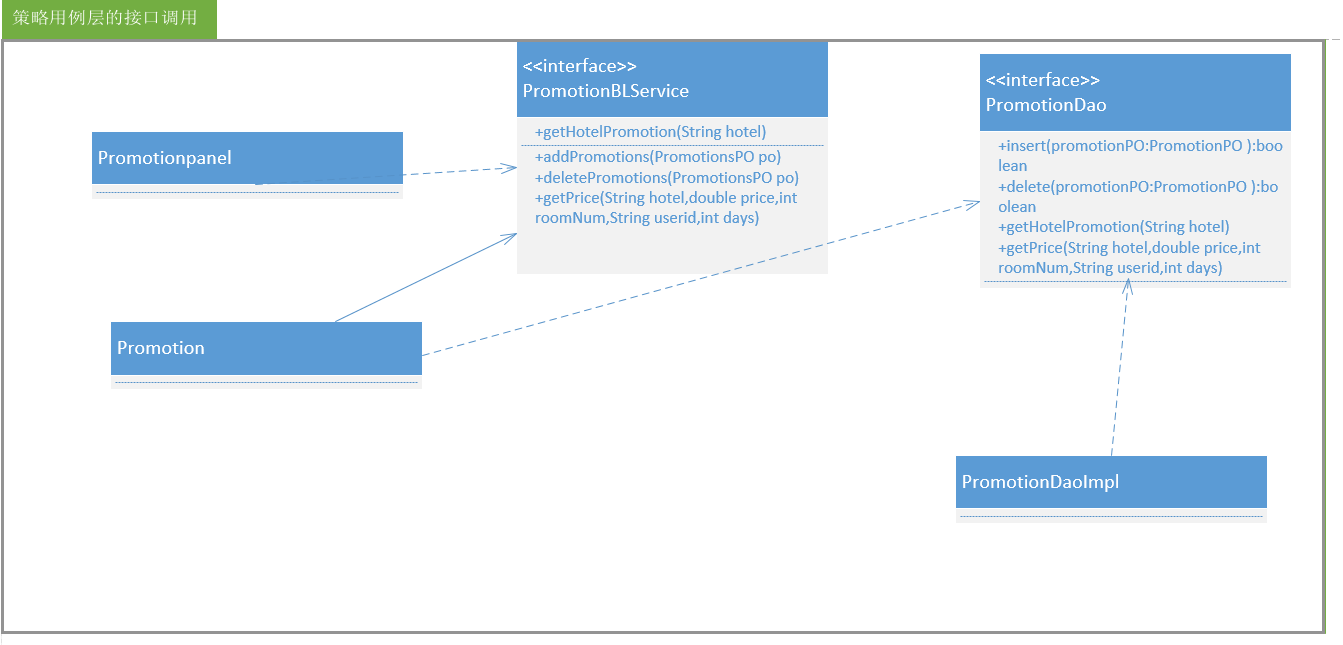
表4 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| OrderBLService  HotelBLService  RoomBLService  PromotionBLService  MemberBLService | 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。



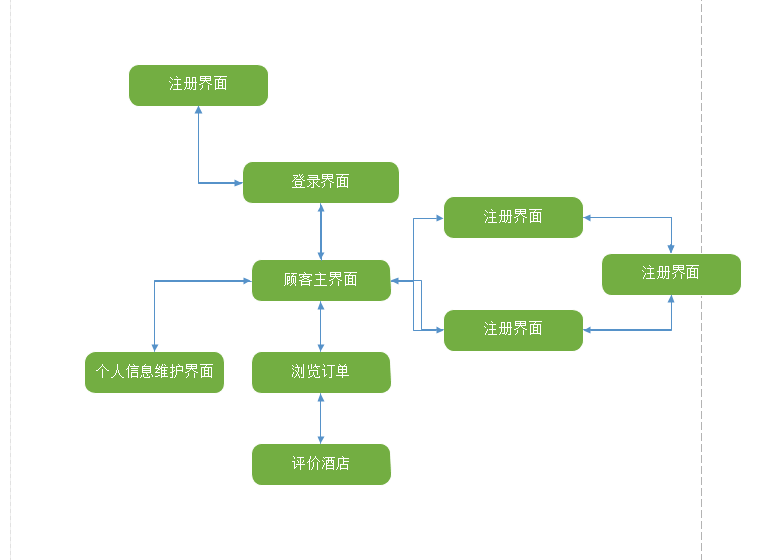


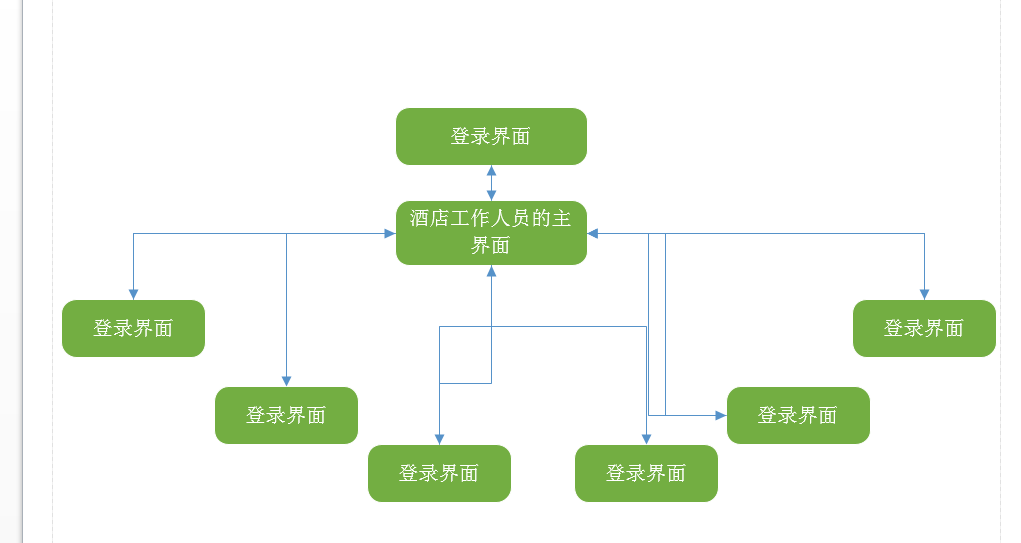


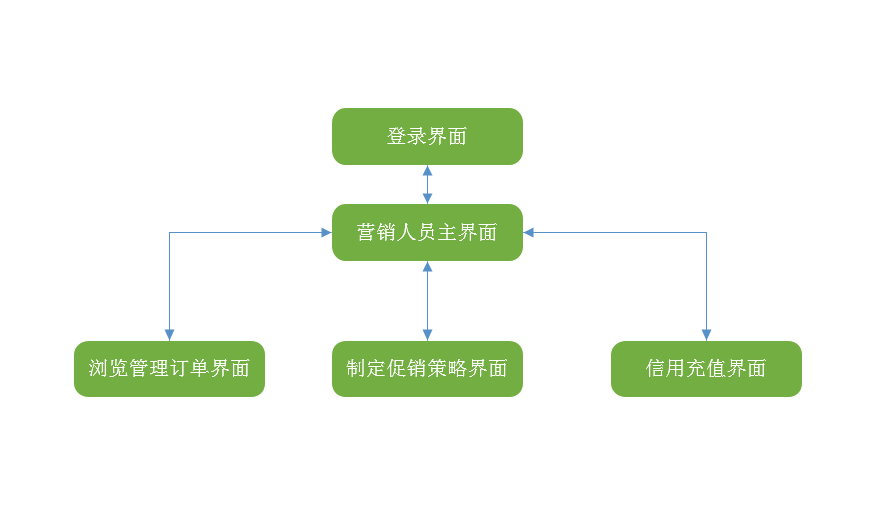
**5.2用户界面层的分解**

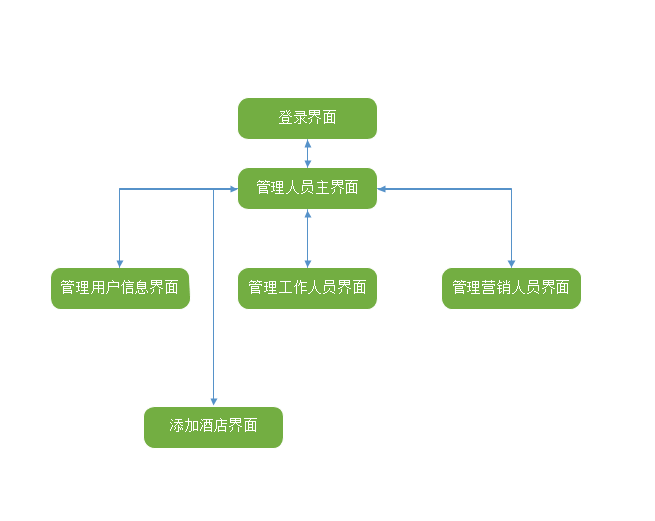
根据需求，系统存在18个界面：客户登录界面，客户主界面，浏览酒店界面，搜索界面，浏览订单，生成订单，个人信息界面，注册界面，评价界面；酒店工作人员登录界面，

酒店工作人员主界面，酒店信息界面，浏览执行订单，录入客房，制定促销策略，入住，退房；网站营销人员登录界面，网站营销人员主界面，浏览管理订单，制定促销策略，信用充值；酒店工作人员登录界面，酒店工作人员主界面，用户信息管理，酒店工作人员信息管理，网站营销人员信息管理，添加酒店。如下图所示

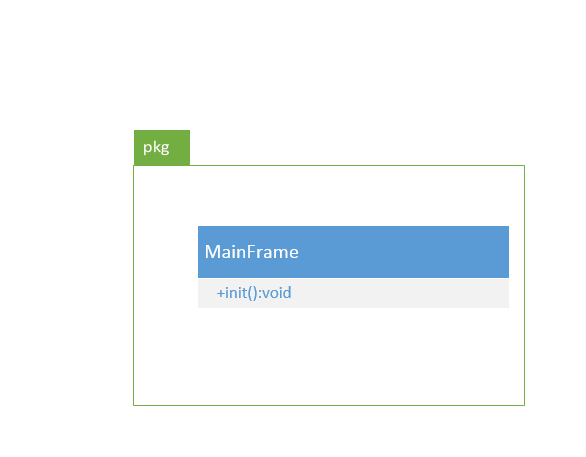








服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如下图所示。



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 |

### 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模块的借口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.view | **语法** | **public orderlist view(int Userid, Type Asktype)** |
| **前置条件** | **Userid存在，Asktype符合规则** |
| **后置条件** | **查找响应的User，根据Asktype返回查找响应的订单，返回订单序列** |
| Order.add | **语法** | **public ResultMessage add(int userId,String hotelName, String roomType,int roomNum, int peopleNum, Date beginDate, Date endDate, Date deadline** |
| **前置条件** | **Userid存在，hotelName，roomType存在，输入的信息符合规则** |
| **后置条件** | **根据Userid查找User,通过输入的信息生成一个Order对象，若成功，则计算价格，返回金额，失败返回失败信息** |
| Order.revoke | **语法** | **public ResultMessage revoke(int orderId)** |
| **前置条件** | **orderId存在** |
| **后置条件** | **根据orderId查找对应订单，根据撤销时间，订单类型，进行撤销操作， 成功返回success以及扣除的信用值，失败返回结果信息** |
| Order.confirm | **语法** | **public void confirm()** |
| **前置条件** | **已完成订单信息填写** |
| **后置条件** | **生成订单，结束此次订单生成任务，持久化更新涉及的领域对象的数据** |
| Order.cancel | **语法** | **public void cancel()** |
| **前置条件** | **已完成订单信息填写** |
| **后置条件** | **删除order对象的信息，跳转到订单信息页面** |
| Order.recover | **语法** | **public void recover(int orderId)** |
| **前置条件** | **工作人员请求撤销** |
| **后置条件** | **根据orderId查找到指定订单，将状态变更为已执行** |
| Order.changeError | **语法** | **public void changeError(int orderId)** |
| **前置条件** | **网站营销人员在进行撤销异常订单** |
| **后置条件** | **根据orderId查找到指定订单，将状态变更为已撤销** |

|  |  |
| --- | --- |
| 需要的接口 | |
| 服务名 | **服务** |
| OrderDataService.find(int id) | **根据ID进行查找单一持久化对象** |
| OrderDataService.find(String field,int orderType) | **根据字段名和类型查找多个持久化对象** |
| OrderDataService.insert(OrdersPO po) | **插入单一持久化对象** |
| OrderDataService.delete(OrdersPO po) | **删除单一持久化对象** |
| OrderDataService.update(OrdersPO po) | **更新单一持久化对象** |
| DatabaseFactory.getOrdersDatabase | **得到Orders数据库的服务引用** |
| DatabaseFactory.getPromotionDatabase | **得到Promotion数据库的服务引用** |
| UserinfoService.findUser(int Userid) | **根据ID查找User，得到UserPO对象** |
| PromotioninfoService.getWebPromtion | **得到网站促销策略** |
| PromotioninfoService.getHotelPromtion(String hotelName) | **根据hotelName得到酒店促销策略** |
| HotelinfoService.findHotel(String hotelName) | **根据hotelName得到HotelPO对象** |
| RoominfoService.findRoom(String hotelName) | **根据hotelName得到RoomPO对象** |
| MemberinfoService.findMember(String userID) | **根据userID得到MemberPO对象** |
| DatabaseFactory.getHotelDatabase | **得到Hotels数据库的服务引用** |
| DatabaseFactory.getRoomDatabase | **得到Rooms数据库的服务引用** |
| DatabaseFactory.getMembersDatabase | **得到Members数据库的服务引用** |
| OrderDataService.insert(OrdersPO po) | **插入OrderPO对象** |
| OrderDataService.delete(OrdersPO po) | **删除OrderPO对象** |
| OrderDataService.update(OrdersPO po) | **更新OrderPO对象** |

**userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **User.staffLogin** | **语法** | **public ResultMessage staffLogin(long staffId, String password);** |
| **前置条件** | **password符合输入规则** |
| **后置条件** | **查找是否存在相应的staff，根据输入的password返回登录验证的结果** |
| **User.marketerLogin** | **语法** | **public ResultMessage marketerLogin(long marketerId, String password);** |
| **前置条件** | **password符合输入规则** |
| **后置条件** | **查找是否存在相应的marketer，根据输入的password返回登录验证的结果** |
| **User.managerLogin** | **语法** | **public ResultMessage managerLogin(long managerId, String password);** |
| **前置条件** | **password符合输入规则** |
| **后置条件** | **查找是否存在相应的manager，根据输入的password返回登录验证的结果** |
| **User.findStaff** | **语法** | **Public ResultMessage findStaff(long staffId)** |
| **前置条件** | **Manager已经验证登录** |
| **后置条件** | **查找是否存在相应的Staff，返回他的对象信息** |
| **User.findMarketer** | **语法** | **Public ResultMessage findMarketer(long maketerId)** |
| **前置条件** | **Manager已经验证登录** |
| **后置条件** | **查找是否存在相应的Marketer，返回他的对象信息** |
| **User.revampStaff** | **语法** | **Public ResultMessage revampStaff(String s)** |
| **前置条件** | **Manager已经验证登录** |
| **后置条件** | **保存成功返回True，失败返回False** |
| **User.revampMarketer** | **语法** | **Public ResultMessage revampMarketer(String s)** |
| **前置条件** | **Manager已经验证登录** |
| **后置条件** | **保存成功返回True，失败返回False** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **服务名** | **服务** |
| **UsersDataService.find(int id)** | **根据ID进行查找单一持久化对象** |
| **UsersDataService.finds(String field, int value)** | **根据字段名和值进行查找多个持久化对象** |
| **UsersDataService.insert(UsersPO po)** | **插入单一持久化对象** |
| **UsersDataService.delete(UsersPO po)** | **删除单一持久化对象** |
| **UsersDataService.update(UsersPO po)** | **更新单一持久化对象** |
| **DatebaseFacory.getSalesDatabase** | **得到Users数据库的服务的引用** |
| **UsersDataService.insert(UsersPO po)** | **在数据库中插入UsersPO对象** |
| **…** | **…** |

**表9 Promotionbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **PromotionImpl.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** |
| **PromotionImpl.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** |
| **PromotionImpl.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** |
| **PromotionImplgetHotelPromotions** | **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)** | **插入单一持久化对象** |
| **PromotionDao.deletePromotions(PromotionPO po)** | **删除单一持久化对象** |
| **PromotionDao.getPrice()** | **得到单一持久化对象** |
| **PromotionsDao.getHotelPromotions()** | **获得单一持久化对象** |

**表10 Memberbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务(供接口)** | | |
| **MembersImpl.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** |
| **MembersImpl.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** |
| **MembersImpl.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** |
| **MembersImpl.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** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **Ordersdao.getOrder** | **得到Order数据库的服务的引用** |
| **Membersdao.getMember** | **得到member数据库的服务的引用** |
| **Membersdao.getCreditRecord** | **得到CreditRecord数据库的服务的引用** |
| **Membersdao.updateCreditRecord** | **更新数据库中信用记录** |

**hotelbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Hotel.searchHotel** | **语法** | **public ArrayList<HotelVO> searchHotel (HotelVO h);** |
| **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** |
| **Hotel.browseHotel** | **语法** | **public ArrayList <HotelVO> browseHotel(HotelVO h);** |
| **前置条件** | **已添加酒店所属商圈、地址** |
| **后置条件** | **返回相应地址、商圈的酒店列表** |
| **Hotel.browseHotelDetails** | **语法** | **public HotelVO browseHotelDetails (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** |
| **Hotel.evaluateHotel** | **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** |
| **Hotel.updateInfo** | **语法** | **public ResultMessage updateInfo(HotelVO h);** |
| **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** |
| **Hotel.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，添加酒店的信息** |

**roombl模块的接口规范**

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **HotelDataService.find**  **(HotelPo h)** | **根据输入的酒店信息查找多个持久化对象** |
| **HotelDataService.insert**  **(HotelPo h)** | **插入单一持久化对象** |
| **HotelDataService.delete**  **(HotelPo h)** | **删除单一持久化对象** |
| **HotelDataService.update (HotelPo h)** | **更新单一持久化对象** |
| **DataFactoryService.getHot**  **elDataService** | **得到酒店数据库的服务的引用** |
| **Room.getRoomInfo(string HotelName)** | **得到该酒店的房间信息** |
| **Promotion.getPromotionInfo**  **(string HotelName)** | **得到该酒店的折扣信息** |
| **Member.getMemberInfo**  **(string HotelName)** | **得到该酒店的工作人员信息** |
| **Order.getHotelHistory**  **(string UserName)** | **得到某一会员的酒店预订历史记录** |
| **Order.getOrderHistory**  **(string UserName,string HotelName)** | **得到某一会员在某一酒店的订单历史记录** |

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **Room.addRoom** | **语法** | **public ResultMessage addRoom(RoomVo r)** |
| **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** |
| **Room.recordOccupancy** | **语法** | **public ResultMessage recordOccupancy (OccupancyVo o)** |
| **前置条件** | **已添加入住类型、房间号、入住时间、预计离开时间** |
| **后置条件** | **在一个记录回合中，增加入住信息** |
| **Room.recordCheckout** | **语法** | **public ResultMessage recordCheckout (CheckoutVo c)** |
| **前置条件** | **已添加实际退房时间、房间号** |
| **后置条件** | **在一个记录回合中，增加退房信息** |

|  |  |
| --- | --- |
| **需要的服务（需接口）** | |
| **RoomDataService.find**  **(RoomPo h)** | **根据输入的房间信息查找多个持久化对象** |
| **RoomDataService.insert**  **(RoomPo h)** | **插入单一持久化对象** |
| **RoomDataService.delete**  **(RoomPo h)** | **删除单一持久化对象** |
| **RoomDataService.update (RoomPo h)** | **更新单一持久化对象** |
| **DataFactoryService.get**  **RoomDataService** | **得到房间数据库的服务的引用** |

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OrdersDataService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** |
| OrdersDataService | 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 |
| OrdersDataServiceTxtFileImpl | 基于txt文件的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 |
| OrdersDataServiceSerializableFileImpl | 基于序列化文件的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 |
| OrdersDataServiceMySqlImpl | 基于 MySql数据库的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 |

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） | | |
| OrdersDataService.find | 语法 | Public OrdersPO 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OrdersPO结果 |
| OrdersDataService.insert | 语法 | Public void insert(OrdersPO 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 |
| OrdersDataService.delete | 语法 | Public void delete(OrdersPO 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 |
| OrdersDataService.update | 语法 | Public void update(OrdersPO 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 |
| OrdersDataService.init | 语法 | Public void init( ) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| OrdersDataService.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 |

# 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

持久化对象的定义依次如下图所示

**public** **class** OrderPO **implements** Serializable{

**int** orderId;

**int** userId;

String userNameString;

String hotelNameString;

RoomType roomType;

**int** roomNum;

**double** price;

OrderType orderType;

Date inDate;

Date outDate;

Date completeDate;

Date revokeDate;

Date deadLine;

**public** OrderPO(**int** orderId, **int** userId, String userNameString,

String hotelNameString, RoomType roomType, **int** roomNum,

**double** price, OrderType orderType, Date inDate, Date outDate,

Date completeDate, Date revokeDate, Date deadLine) {

**super**();

**this**.orderId = orderId;

**this**.userId = userId;

**this**.userNameString = userNameString;

**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**.revokeDate = revokeDate;

**this**.deadLine = deadLine;

}

**public** **int** getOrderId() {

**return** orderId;

}

**public** **int** userId(){

**return** userId;

}

**public** String getUserNameString(){

**return** userNameString;

}

**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 getRevokeDate(){

**return** revokeDate;

}

**public** Date getDeadLine(){

**return** deadLine;

}

}

import PromotionStub.Hotel;

import PromotionStub.PromotionRole;

public class PromotionPO {

Hotel hotel;

String type;

PromotionRole role;

public PromotionPO(PromotionRole role,Hotel hotel,String type ){

this.hotel=hotel;

this.type=type;

this.role=role;

}

public Hotel getHotel(){

return hotel;

}

public String getType(){

return type;

}

public PromotionRole getRole(){

return role;

}

}

**public** **class** UserPO {

**int** id;

String name;

String password;

String hotel;

UserType role;

**public** UserPO(**int** i,String n,String p ,UserType r,String hotel){

**this**.id=i;

**this**.name=n;

**this**.password=p;

**this**.role=r;

**this**.hotel=hotel;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** String getPassword() {

**return** password;

}

**public** UserType getRole() {

**return** **this**.role;

}

**public** String getHotel(){

**return** **this**.hotel;

}

}

HotelPO

**import** java.io.Serializable;

**public** **class** HotelPO **implements** Serializable{

**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;

}

}

oooooooo9

**import** java.io.Serializable;

**public** **class** RoomPO **implements** Serializable{

**private** String roomName;

**private** String hotelBelongTo;

**private** RoomType roomtype;

**private** String introdution;

**private** **double** price;

//status需要存储不可用的时间段，不应采用int数组，具体实现待讨论

**private** **int**[] unavailablePeriod;

**public** RoomPO(String r, String h, RoomType t,String i,**double** p, **int**[] s){

**this**.roomName= r;

**this**.hotelBelongTo =h;

**this**.roomtype =t;

**this**.introdution=i;

**this**.price =p;

**this**.unavailablePeriod =s;

}

**public** String getRoomName(){

**return** roomName;

}

**public** String getHotelBelongTo(){

**return** hotelBelongTo;

}

**public** RoomType getRoomType(){

**return** roomtype;

}

**public** String getIntroduction(){

**return** introdution;

}

**public** **double** getPrice(){

**return** price;

}

**public** **int**[] getRoomStatus(){

**return** unavailablePeriod;

}

}

**public** **class** MemberPO **implements** Serializable {

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** String getId() {

**return** id;

}

**public** String getPassword() {

**return** password;

}

**public** String getName() {

**return** name;

}

**public** String getTelephone() {

**return** telephone;

}

**public** String getSpecial() {

**return** special;

}

**public** **double** getCredit() {

**return** credit;

}

**public** **int** getType() {

**return** type;

}

**public** **int** getLevel() {

**return** level;

}

}

## 6.2 Txt持久化格式

Txt数据保持格式以user.txt为例，分别对应用户名，用户id，密码，种类，用：隔开

如下：

小明：0001：12345：user

## 6.3 数据库表

数据库表中包含Order表，User表，Member表，Promotion表，Hotel表，Room表