**互联网酒店预订系统**

**体系结构设计文档**

学 院：南京大学软件学院

成 员：谢铠联 谢凯航 夏沐天 武秀峰

完成日期：2016年10月15日

**目录**

[更新历史 3](#_Toc464254514)

[1、引言 4](#_Toc464254515)

[1.1 编制目的 4](#_Toc464254516)

[1.2 词汇表 4](#_Toc464254517)

[1.3 参考资料 5](#_Toc464254518)

[2、产品概述 5](#_Toc464254519)

[3、逻辑视角 6](#_Toc464254520)

[4、组合视角 8](#_Toc464254521)

[4.1 开发包图 9](#_Toc464254522)

[4.2 运行时进程 13](#_Toc464254523)

[4.3 物理部署 14](#_Toc464254524)

[5、接口视角 14](#_Toc464254525)

[5.1 模块的职责 14](#_Toc464254526)

[5.2 用户界面层的分解 17](#_Toc464254527)

[5.2.1 用户界面层模块的职责 19](#_Toc464254528)

[5.2.2用户界面层模块的接口规范 19](#_Toc464254529)

[5.2.3 用户界面模块设计原理 21](#_Toc464254530)

[5.3 业务逻辑层的分解 21](#_Toc464254531)

[5.3.1 业务逻辑层模块的职责 21](#_Toc464254532)

[5.3.2 业务逻辑层模块的接口规范 22](#_Toc464254533)

[5.4 数据层的分解 29](#_Toc464254534)

[5.4.1 数据层模块的职责 30](#_Toc464254535)

[5.4.2 数据层模块的接口规范 30](#_Toc464254536)

[6、信息视角 35](#_Toc464254537)

[6.1 数据持久化对象 35](#_Toc464254538)

[6.2 数据库表 36](#_Toc464254539)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 全体人员 | 2016-10-14 | 最初草稿（创建文档模板） | V1.0 |
| 全体人员 | 2016-10-15 | ResultMesage用布尔类型表示，  更改了命名方式合并Orderhan与OrderView成Order。 | V2.0 |

# 1、引言

## 1.1 编制目的

本报表详细完成对互联网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 |
| \_ui | 表示某展示层 |  |
| \_bl | 表示某逻辑层 |  |
| \_data | 表示某数据层 |  |
| RMI | 表示远程方法调用 |  |

## 1.3 参考资料

1. IEEE std 1471-2000
2. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182

# 2、产品概述

参考互联网酒店预订系统用例文档和互联网酒店预订系统软件需求规格说明文档中对产品的概括描述。互联网酒店预订系统主要是应用于客户方便快捷的在网络预订酒店，主要功能见用例图如下。****

# 3、逻辑视角

互联网酒店预订系统中，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。



图一 参照体系结构风格的包图表达逻辑视角



图二 软件体系结构逻辑设计方案

# 4、组合视角

[软工2 166]与抽象的逻辑设计相比，实现物理设计要考虑更多的实现细节，这些细节有：

1. presentation层与logic层被置于客户端，data层被置于服务器端，那么logic层的开发包不可能依赖于data层的开发包。使用RMI技术，将data层开发包分解为置于客户端的dataservice接口包和置于服务器端的data层开发包。这样一来，logic层开发包依赖于dataservice包，dataservice包和data层的开发包都依赖于RMI类库包。
2. 所有的data层开发包都需要进行数据持久化（例如读写数据库、读写文件等），所以它们会有一些重复代码，可以将重复代码独立为新的开发包，然后所有的data层开发包都依赖于databaseutility。datavaseutility会依赖于JDBC类库包。
3. 所有的presentation层开发包都需要使用图形类型建立界面，都要依赖于图形界面类库包。
4. 此外，presentation层实现时，由mainui包负责整个页面之间的跳转逻辑。其他各包负责各自页面自身的功能。
5. 在分层风格的典型设计中，不希望高层直接依赖于低层，而是为低层建立接口包，实现依赖倒置原则，所以应该调整为：各presentation层开发包（调用）依赖于logic层接口包businesslogicservice包，logic层开发包也依赖于（实现了）logic层接口包businesslogicservice包。
6. 在分层风格的典型设计中，presentation层与logic层之间、logic层与data层之间可能会传递复杂数据对象，那么相邻两层都需要使用数据对象声明，所以需要将数据对象声明独立为开发包（VO包和PO包）。

8）在logic层中，初始化和业务逻辑层上下文的工作被分配到utility包中。

经过细节改进，最终建立的互联网酒店预订系统开发包设计如表4.1-1，其局部包图如图4.1-1和4.1-2所示：

## 4.1 开发包图

表4.1-1 互联网酒店预订系统的最终开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| mainui | Orderui ,Userui,Promotionui,Searchui,Hotelui,vo |
| orderui | Orderblservice,界面类库包,vo |
| orderbl | Orderblservice,Oderdataservice,po,Userbl,Promotionbl,Hotelbl |
| orderblservice |  |
| orderdata | databaseutility,po, Orderdataservice |
| orderdataservice | Java RMI,po |
| userui | Userblservice,界面类库包,vo |
| userbl | Userblservice,Userdataservice,po, |
| userblservice |  |
| userdata | databaseutility,po, Userdataservice |
| userdataservice | Java RMI,po |
| promotionui | Promotionblservice,界面类库包,vo |
| promotionbl | Promotionblservice,Promotiondataservice,po, |
| promotionblservice |  |
| promotiondata | databaseutility,po, Promotiondataservice |
| promotiondataservice | Java RMI,po |
| searchui | Searchblservice,界面类库包,vo |
| searchbl | Searchblservice,Searchdataservice,po,Userbl,Hotelbl |
| searchblservice |  |
| searchdata | databaseutility,po, Searchdataservice |
| searchdataservice | Java RMI,po |
| hotelui | Hotelblservice,界面类库包,vo |
| hotelbl | Hotelblservice,Hoteldataservice,po, |
| hotelblservice |  |
| hoteldata | databaseutility,po, Hoteldataservice |
| hoteldataservice | Java RMI,po |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| Databaseutility | JDBC |

图4.1-1 互联网酒店预订系统客户端开发包图



图4.1-2 互联网酒店预订系统服务器端开发包图

## 4.2 运行时进程

在互联网酒店预订系统中，会有多个客户端进程和一个服务器端进程，其进程图如图4.2所示。结合部署图，客户端进程实在客户端机器上运行，服务器端进程是在服务器端机器上运行。

图4.2 进程图

## 4.3 物理部署

互联网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JavaRMI构件属于JDK6.0的一部分。所以，在系统JDK环境未设置好的情况下，需要将系统打包成exe文件。部署图如图4.3所示。



4.3 部署图

# 5、接口视角

## 5.1 模块的职责

客户端模块和服务器端模块视图分别如图5.1-1和图5.1-2所示。客户端各层和服务器端各层的职责分别如表5.1-1和表5.1-2所示。



图5.1-1 客户端模块视图



图5.1-2 服务器端模块视图

表5.1-1 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 | 基于窗口的进销存系统客户端用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

表5.1-2 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 RMI机制开启RMI服务，注册RMI服务 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表5.1-3所示。

表5.1-3 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| orderblService  userblService  promotionblService  searchblService  hotelblService | 客户端展示层 | 客户端业务逻辑层 |
| orderDataService  userDataService  promotionDataService  searchDataService  hotelDataService | 客户端业务逻辑层 | 服务器端数据层 |

借用用户管理用例来说明层之间的调用，如图5.1-3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）。userblService提供了User界面所需要的所有业务逻辑功能UserDataService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。



图5.1-3 用户管理用例层之间调用的接口

## 5.2 用户界面层的分解

根据需求，系统存在19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、个人信息界面、酒店详情界面、酒店搜索界面、酒店搜索记录界面、酒店评价界面、会员登记界面、酒店信息维护界面、客房信息维护界面、酒店优惠政策制定界面、信用值修改界面、网站优惠政策制定界面、调整用户界面、订单浏览界面、订单处理界面。

界面跳转如图5.2所示。



图5.2 用户界面跳转

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。用户界面类如图5.2-2所示。



图5.2-2 用户界面类

### 5.2.1 用户界面层模块的职责

如表5.2.1-1所示为用户界面层模块的职责。

表5.2.1-1 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如表5.2.2-1所示。

表5.2.2-1 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| mainui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| orderui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示OrderPanel |
| user\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示UserPanel |
| promotion\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示PromotionPanel |
| search\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示SearchPanel |
| hotelui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示HotelPanel |

用户界面层需要的服务接口如表5.2.2-2所示。

表5.2.2-2 用户界面层模块需要的服务接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| businessLogicService.order\_blservice | 处理与订单查看、订单处理相关的接口 |
| businesslLogicService.user \_blservice | 处理用户信息的接口 |
| businessLogicService.promotion\_blservice | 处理酒店、网站促销策略制定的接口 |
| businesslogicservice.search\_blservice | 负责用户信息查询、酒店搜索的接口 |
| businesslogicservice.hotel\_blservice | 处理酒店信息的接口 |

### 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

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，User对象负责处理登陆界面的业务逻辑；User对象负责处理管理用户的业务逻辑。业务逻辑层的设计如图5.3-1所示。



图5.3-1 业务逻辑层

### 5.3.1 业务逻辑层模块的职责

业务逻辑层模块的职责如表5.3.1-1所示。

表5.3.1-1 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| orderbl | 负责与订单处理与订单查看相关的服务 |
| userbl | 负责处理与用户信息管理相关的服务 |
| promotionbl | 负责与促销策略相关的服务 |
| searchbl | 负责处理与搜索酒店和查询用户信息的服务 |
| hotelbl | 负责处与酒店信息相关的服务 |

### 5.3.2 业务逻辑层模块的接口规范

表5.3.2-1 orderbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| orderbl.handleAbnormalOrder | 语法 | public boolean handleAbnormalOrder(OrderVO order); | |
| 前置条件 | 处理异常订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已执行并且增加用户被扣除的信用值 | |
| orderbl.reverseOrder | 语法 | public boolean reverseOrder(OrderVO order); | |
| 前置条件 | 撤销订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已撤销，根据撤销时间减少用户信用值 | |
| orderbl.finishOrder | 语法 | public boolean finishOrder(OrderVO order); | |
| 前置条件 | 执行订单，需要改变订单状态和用户信息 | |
| 后置条件 | 将订单状态变为已执行并且增加客户信用值 | |
| orderbl.createOrder | 语法 | public boolean  createOrder(OrderVO order); | |
| 前置条件 | 生成订单，已知完整订单信息 | |
| 后置条件 | 提交订单，持久化保存订单信息 | |
| orderbl.personOrders | 语法 | Public List <OrderVO> personOrders(String username); | |
| 前置条件 | 查看个人订单，已知用户名 | |
| 后置条件 | 根据用户名返回个人订单列表 | |
| orderbl.hotelOrders | 语法 | public List<OrderVO> hotelOrders(String hotelname); | |
| 前置条件 | 浏览酒店订单，已知酒店名称 | |
| 后置条件 | 根据酒店名称返回酒店订单列表 | |
| orderbl.netOrders | 语法 | public List<OrderVO> netOrders(); | |
| 前置条件 | 浏览网站订单，已经登录验证 | |
| 后置条件 | 返回网站未执行和异常订单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| order\_data.add(OrderPO order) | | | 新增一个订单信息 |
| order\_data.person\_find(OrderPO order) | | | 返回个人订单列表 |
| order\_data.modify(OrderPO order) | | | 改变一个订单的状态 |
| user\_bl.add\_credit(String username,int credit) | | | 改变个人信息中的信用值 |
| order\_data.hotel\_find (String hotelname) | | | 返回单酒店订单列表 |
| order\_data.exception\_find () | | | 返回未执行和异常订单列表 |
| promotion\_bl.priceCut(hotelPO price ,orderVO number,orderVO type) | | | 返回折后价格 |

表5.3.2-2 userbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| userbl.  register | 语法 | public boolean register(UserVO userinfo) | |
| 前置条件 | 客户注册，已知注册信息 | |
| 后置条件 | 根据注册信息更新网站用户信息并返回注册结果 | |
| userbl.userView | 语法 | public UserVO userView(String username,int usertype); | |
| 前置条件 | 显示用户信息，已知用户类型和名 | |
| 后置条件 | 根据用户类型和名返回用户信息 | |
| userbl.userSave | 语法 | public boolean userSave(UserVO user\_info); | |
| 前置条件 | 保存更改的用户信息，已知更改后用户信息 | |
| 后置条件 | 更新用户信息并返回保存结果 | |
| userbl.addCredit | 语法 | public boolean addCredit (String username,int credit) | |
| 前置条件 | 已知增加的信用值，需要更新个人信息内的信用值 | |
| 后置条件 | 更新个人信息并返回结果 | |
| userbl.registerMember | 语法 | public boolean registerMember(UserVO user\_info); | |
| 前置条件 | 已知会员信息，需要向个人信息中加入会员信息 | |
| 后置条件 | 更新个人信息并返回结果 | |
| userbl.getPersonInfo | 语法 | public PersonPO getPersonInfo(String username); | |
| 前置条件 | 已知客户名，需要取得用户信息 | |
| 后置条件 | 根据客户名返回客户信息 | |
| userbl.userDelete | 语法 | Public boolean userDelete(UserVO userinfo) | |
| 前置条件 | 已知用户信息，需要删除用户信息 | |
| 后置条件 | 根据用户信息删除用户信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| user\_data.add(userPO userInfo) | | | 新增客户信息 |
| user\_data.find(String username, int usertype) | | | 返回客户信息 |
| user\_data.modify(userPO userInfo) | | | 改变客户信息 |
| user\_data.addCredit(userPO userInfo) | | | 增加客户信用值 |

表5.3.2-3 promotionbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| promotionbl.priceCut | 语法 | public double priceCut(HotelPO price ,OrderVO numberAndType); | |
| 前置条件 | 已知客户所需房间数量与类型，已知该酒店房间价格 | |
| 后置条件 | 返回考虑优惠政策后的价格 | |
| promotionbl.changeProm | 语法 | public Boolean  changeProm(PromotionVO prom,int operation); | |
| 前置条件 | 酒店工作人员或网站营销人员输入新优惠政策与操作选择 | |
| 后置条件 | 改变存储的优惠政策 | |
| promotionbl.getProm | 语法 | public Boolean List<PromotionPO>getProm(String hotelname); | |
| 前置条件 | 需要获取当前的优惠政策内容 | |
| 后置条件 | 返回当前优惠政策PO列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| PromotionData.add(PromotionPO promotion); | | | 增加一个促销策略 |
| promotionData.find(String hotelname,int promotiontype) | | | 查找一个促销策略 |
| promotionData.delete(PromotionPO promotion) | | | 删除一个促销策略 |
| promotionData.modify(PromotionPO promotion) | | | 改变一个促销策略 |
| userbl.userView(String personName,int usertype) | | | 获得客户的信息 |

表5.3.2-4 searchbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| search\_bl.findHotel | 语法 | public List<HotelVO>findHotel(hotelPO condition); | |
| 前置条件 | 已知客户的筛选条件,需要查找符合条件的酒店 | |
| 后置条件 | 返回符合条件的酒店VO列表 | |
| search\_bl.saveHistory | 语法 | public Boolean saveHistory (Search VO); | |
| 前置条件 | 保存搜索酒店记录 | |
| 后置条件 | 持久化保存搜索酒店的记录 | |
| search\_bl.deleteHistory | 语法 | public Boolean deleteHistory(String hotelname) | |
| 前置条件 | 已知客户需要删除的酒店编号 | |
| 后置条件 | 数据层删除选定的搜索记录 | |
| search\_bl.searchInfo | 语法 | public userVO searchInfo(String username,int usertype) | |
| 前置条件 | 已知需要查询的用户类型和名称，需要返回查找结果 | |
| 后置条件 | 返回查找的对象的PO类 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| search\_data.add(SearchPO); | | | 增加搜索记录 |
| hotel\_ data.findWithReq(hotelPO condition); | | | 依据条件查找酒店 |
| search\_ data.Delete(SearchPO) | | | 删除搜索记录 |
| user\_data.userView(String userID, int usertype) | | | 依据条件查找用户信息 |

表5.3.2-5 Hotelbl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| hotel\_bl.showHotelInfo | 语法 | public HotelVO getHotel(String Hotelname); | |
| 前置条件 | 已知酒店编号，需要返回酒店的信息 | |
| 后置条件 | 返回酒店的PO类 | |
| hotel\_bl.modifyHotelInfo | 语法 | public boolean modifyHotelinfo(HotelVO hotelinfo); | |
| 前置条件 | 酒店工作人员维护酒店的信息 | |
| 后置条件 | 数据层修改酒店的信息 | |
| Hotel\_bl.roomModify | 语法 | public Boolean roomModify(String roomtype,int number) | |
| 前置条件 | 酒店工作人员修改酒店房间数量 | |
| 后置条件 | 更新酒店的房间数量 | |
| hotel\_bl.addComment | 语法 | public boolean addcomment(String comment，String username,String hotelname); | |
| 前置条件 | 用户需要评论酒店 | |
| 后置条件 | 数据层增加酒店评论 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| hotel\_data.getHotelInfo(String Hotelname); | | | 根据酒店名返回酒店信息 |
| hotel\_ data.modify(HotelPO hotelinfo) | | | 修改酒店信息 |
| hotel\_data. roomModify | | | 修改酒店房间信息 |
| hotel\_data.addComment (String comment,String username,String hotelname) | | | 增加评论 |

## 5.4 数据层的分解

数据层主要给业务逻辑层提供数据防伪服务，包括对于持久化数据的增、删、改、查。User业务逻辑需要的服务由UserDataService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。数据层模块的具体描述如图5.4所示。

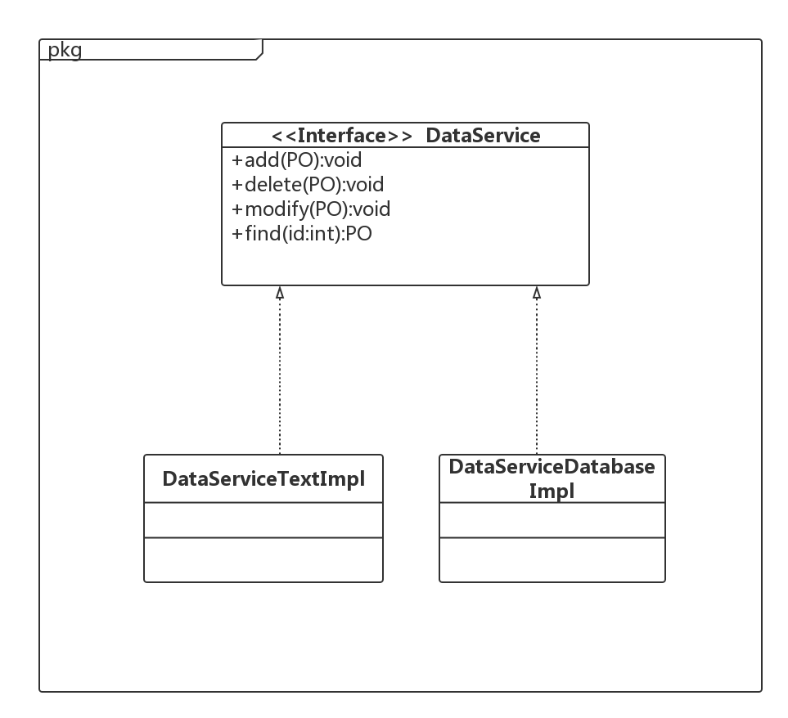


图5.4 数据层模块的描述

### 5.4.1 数据层模块的职责

数据层模块的职责如表5.4.1所示。

表5.4.1 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| dataService | 持久化数据库的接口，提供集体载入，集体保存、增、删、改、查服务。 |
| dataServiceTxtImpl | 基于TXT文件的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| dataServiceDatabaseImpl | 基于数据库的持久化数据的接口，提供集体载入、集体保存、增、删、改、查服务 |

### 5.4.2 数据层模块的接口规范

表5.4.2-1 数据层OrderData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| order\_data.add | 语法 | public boolean add(OrderPO order) |
| 前置条件 | 数据库中不存在username相同，时间段重合的订单 |
| 后置条件 | 数据库中增加一条订单记录并且返回结果 |
| order\_data.personfind | 语法 | Public List <OrderPO> person\_find(OrderPO order) |
| 前置条件 | 数据库中存在username相同的记录 |
| 后置条件 | 如果没有符合OrderID的订单记录则返回null |
| order\_data.modify | 语法 | public boolean modify(OrderPO order) |
| 前置条件 | 数据库中存在OrderID相同的订单 |
| 后置条件 | 数据库中更新订单记录并返回结果 |
| order\_data.hotelfind | 语法 | Public List<hotelPO>hotel\_find (String hotelname) |
| 前置条件 | 数据库中存在相同酒店名的订单记录 |
| 后置条件 | 如果不存在相同酒店名的订单记录返回null |
| order\_data.exceptionfind | 语法 | public List<OrderPO> exception\_find (); |
| 前置条件 | 数据库中存在异常订单和未执行订单记录 |
| 后置条件 | 如果不存在就返回null |

表5.4.2-2 数据层UserData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| user\_data.find | 语法 | public userPO find(String username, String usertype) |
| 前置条件 | 在usertype的数据库存在相同username的记录 |
| 后置条件 | 不存在则返回null |
| user\_data.add | 语法 | public boolean add(userPO userInfo); |
| 前置条件 | 在usertype的数据库不存在相同username的记录 |
| 后置条件 | 数据库中增加一条记录并返回ture |
| user\_data.modify | 语法 | public boolean modify(userPO userInfo) |
| 前置条件 | 在usertype的数据库存在相同username的记录 |
| 后置条件 | 数据库中更新记录并且返回结果 |
| user\_data.delete | 语法 | public boolean delete(UserPO userInfo) |
| 前置条件 | 在usertype的数据库存在相同username的记录 |
| 后置条件 | 数据库中删记录并且返回结果 |
| user\_data.addcredit | 语法 | public boolean addcredit(String username,int credit) |
| 前置条件 | 在数据库存在相同username的记录 |
| 后置条件 | 数据库中修改信用值并返回结果 |

表5.4.2-3 数据层PromotionData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| promotion\_data.add | 语法 | public boolean add(PromotionPO promotion) |
| 前置条件 | 数据库中不存在名字相同的记录 |
| 后置条件 | 在数据库中增加一条促销策略记录并返回结果 |
| promotion\_data.find | 语法 | public List <PromotionPO> find(String hotelnames,int Promotiontype) |
| 前置条件 | 无 |
| 后置条件 | 返回对应酒店名的PromotionPO，没有则返回null |
| promotion\_data.modify | 语法 | Public boolean modify(PromotionPO promotion) |
| 前置条件 | 数据库中存在名字相同的记录 |
| 后置条件 | 更新数据库中的记录并返回结果 |
| promotion\_data.delete | 语法 | Public boolean delete(PromotionPO promotion) |
| 前置条件 | 数据库中存在完全相同的记录 |
| 后置条件 | 删除数据库中的记录并返回结果 |

表5.4.2-4 数据SearchData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| search\_data.add | 语法 | public boolean add(String hotelID); |
| 前置条件 | 数据库中不存在完全相同的记录 |
| 后置条件 | 数据库中增加一条搜索记录并且返回结果 |
| search\_ data.delete | 语法 | Public boolean delete(String username) |
| 前置条件 | 无 |
| 后置条件 | 清空数据库中所有搜索记录并返回结果 |

表5.4.2-5 数据层HotelData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| hotel\_data.getHotelinfo | 语法 | public HotelPO getHotelinfo(String Hotelname); |
| 前置条件 | 在数据库中存在相同Hotelname的记录 |
| 后置条件 | 如果不存在就返回null |
| hotel\_ data.modify | 语法 | public boolean modify(HotelPO hotelinfo) |
| 前置条件 | 数据库中存在相同酒店名的记录 |
| 后置条件 | 数据库修改记录并且返回成功消息 |
| hotel\_data.findWithReq | 语法 | public List<HotelPO> findWithReq(HotelPO) |
| 前置条件 | 已知筛选条件，需要筛选符合条件的酒店 |
| 后置条件 | 返回一个List<HotelPO> |
| hotel\_data. addComment | 语法 | public boolean addComment (String comment,String username,Stirng hotelname); |
| 前置条件 | 数据库中存在酒店的记录 |
| 后置条件 | 数据库中增加一个评论并且返回结果 |
| hotel\_data. roomModify | 语法 | public boolean roomModify (String roomtype,int roomnumber); |
| 前置条件 | 数据库中存在酒店的记录 |
| 后置条件 | 数据库中增加一个评论并且返回结果 |

# 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，持久化序列化文件保存在ser文件中,如表6.1所示：

表6.1 PO类简单介绍

|  |  |
| --- | --- |
| 类名 | 包含的属性 |
| orderPO | 订单信息：客户名、时间、酒店名、房间类型、房间数量、入住人数、入住人信息（身份证号、姓名），订单价格、订单编号、订单状态（预设为未执行） |
| personPO | 客户信息：用户名、密码、会员等级、会员信息、信用值、用户类型 |
| promotionPO | 促销策略信息：促销名、促销起始时间、促销结束时间、促销类型（根据会员等级区分）、促销折扣、促销酒店 |
| searchPO | 搜索记录信息：客户名、时间、酒店名 |
| hotelPO | 酒店信息：酒店名、星级、酒店特色、服务设备、地址、商圈、房间类型、房间数量、房间价格、评价信息、酒店工作人员名称 |
| hotelWorkerPO | 酒店工作人员信息：酒店名、用户名、密码 |
| managerPO | 网站管理人员信息：用户名、密码、用户类型 |
| marketPO | 网站营销人员信息：用户名、密码、用户类型 |
| userPO | 用户信息：用户名、密码、用户类型 |

持久化用户对象如UserPO的定义如图6.1，更多定义见原型代码

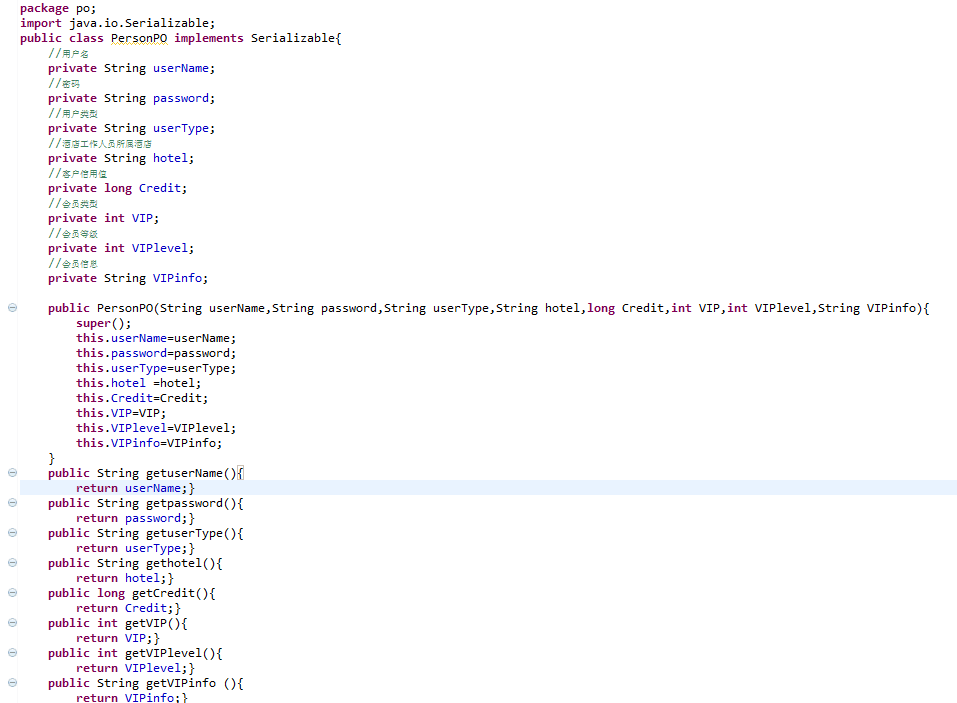




图6.1 持久化用户对象UserPO的定义

## 6.2 数据库表

|  |  |
| --- | --- |
| 表名 | 内容 |
| personInf | 客户信息 |
| manager | 网站管理人员信息 |
| hotelWroker | 酒店工作人员信息 |
| hotel | 酒店信息 |
| order | 订单信息 |
| search | 搜索记录信息 |
| promotion | 促销策略信息 |

|  |  |
| --- | --- |
| Txt文件 | 内容 |
| 酒店评价 | 保存用户名、酒店名、时间、以及评价内容，格式如下：  @用户名@酒店名@时间@酒店评价  @用户名@酒店名@时间@酒店评价 |