# 酒店管理系统体系结构设计文档

学 院：南京大学软件学院

成 员：宋欣建 叶涵 姚政 黄程尔

完成日期：2016年10月14日

目录

[酒店管理系统体系结构设计文档 1](#_Toc464414343)

[更新历史 3](#_Toc464414344)

[1、引言 4](#_Toc464414345)

[1.1 编制目的 4](#_Toc464414346)

[1.2 词汇表 4](#_Toc464414347)

[1.3 参考资料 4](#_Toc464414348)

[2、产品概述 4](#_Toc464414349)

[3、逻辑视角 6](#_Toc464414350)

[4、组合视角 7](#_Toc464414351)

[4.1 开发包图 8](#_Toc464414352)

[4.2 运行时进程 14](#_Toc464414353)

[4.3 物理部署 14](#_Toc464414354)

[5、接口视角 15](#_Toc464414355)

[5.1 模块的职责 15](#_Toc464414356)

[5.2 用户界面层的分解 20](#_Toc464414357)

[5.3 业务逻辑层的分解 22](#_Toc464414358)

[5.4 数据层的分解 40](#_Toc464414359)

[6、信息视角 54](#_Toc464414360)

[6.1 数据持久化对象 54](#_Toc464414361)

[6.2 数据库表 56](#_Toc464414362)

## 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 全体人员 | 2016-10-12 | 最初草稿（创建文档模板） | V0.1 |
| 全体人员 | 2016-10-13 | 最终结果 | V0.2 |
| 宋欣建 | 2016-10-15 | 添加逻辑层Order的依赖 | V0.3 |
| 全体人员 | 2016-10-16 | 对逻辑层和数据层进行了重构 | V0.4 |
| 全体人员 | 2016-10-25 | 对界面层进行了重构 | V0.5 |
| 全体人员 | 2016-10-27 | 修改了逻辑层和数据层的接口 | V0.6 |
| 姚政 | 2016-11-19 | 逻辑层、数据层继续修改 | V0.7 |

## 1、引言

### 1.1 编制目的

本报表详细完成对酒店管理系统的概要设计，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

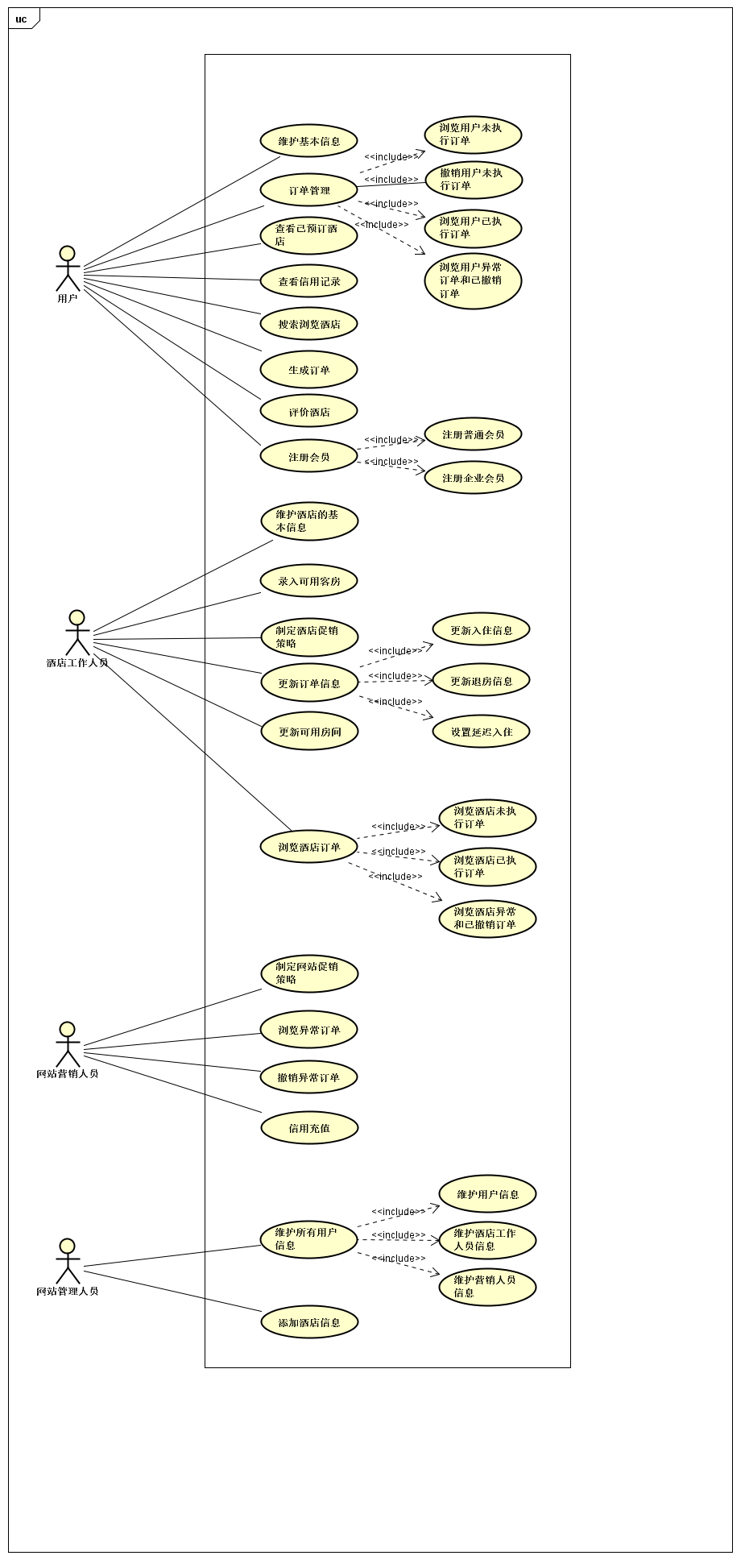
|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| UI | 表示某展示层 |  |
| BL | 表示某逻辑层 |  |
| Data | 表示某数据层 |  |
| RMI | 表示远程方法调用 |  |

### 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

参考酒店管理系统用例文档和酒店管理系统软件需求规格说明文档中对产品的概括描述。酒店管理系系统主要是应用于在线的互联网酒店预定系统，主要功能见用例图如下。



## 3、逻辑视角

酒店管理系统中，选择了分层体系结构风格，将系统分为3层(展示层、

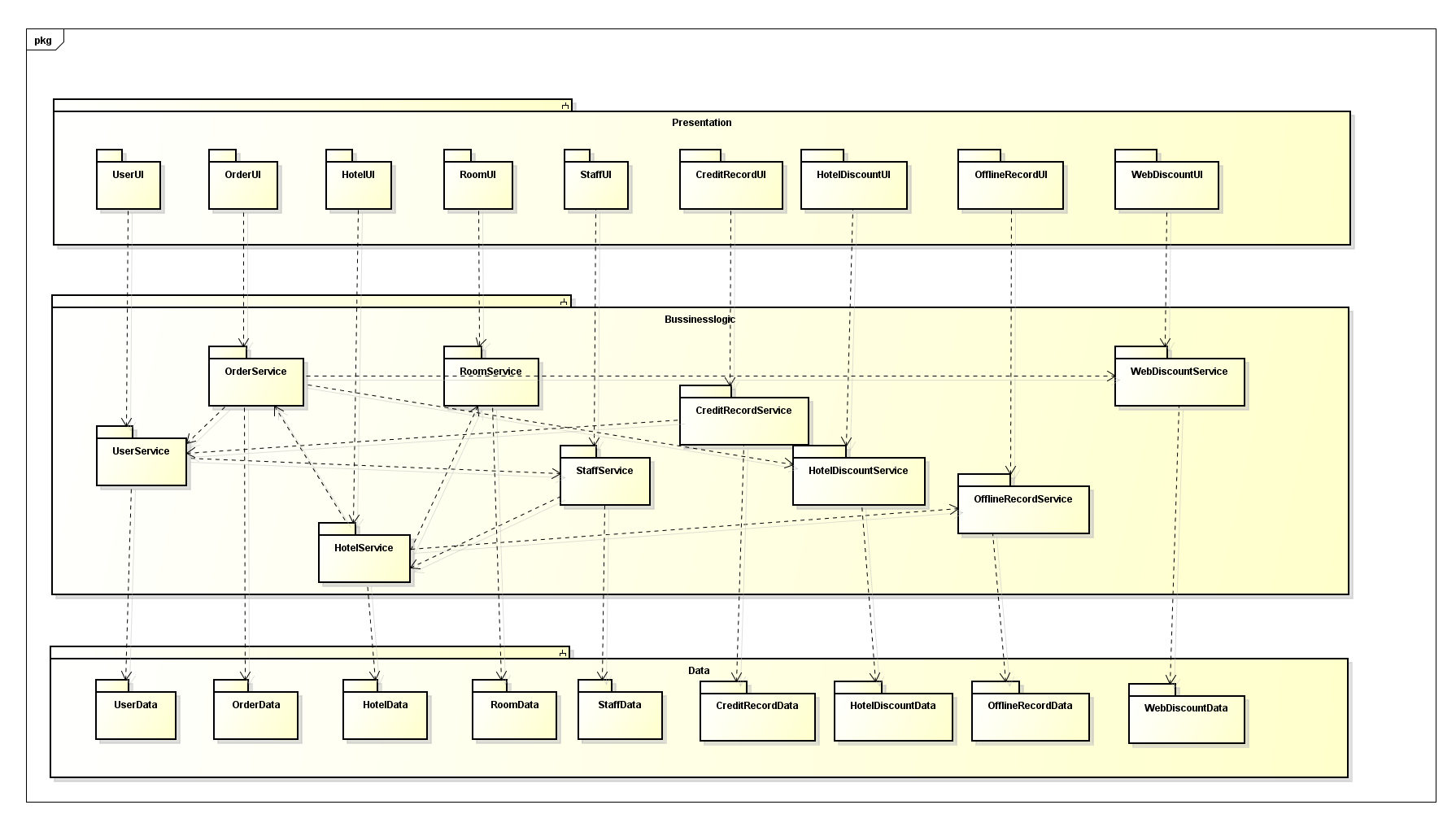
业务逻辑层、数据层)能够很好地示意整个高层抽象。展示层包含GUI页面

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层被置于客户端，logic层和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login层开发包分解为置于客户端的LogicService接口包和置于服务器端的LogicServiceImpl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databasetility。datavaseutility会依赖于Hibernate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loginui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blservice包，logic层开发包也依赖于（实现了）data层接口包dataservice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）。

7）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，最终建立酒店管理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 酒店管理系统的最终开发包设计

|  |  |
| --- | --- |
| UserUI | UserService,OrderService,HotelService,LocationService,CommCircleService,CreditRecordService,界面类库包,VO |
| HotelUI | OrderService,HotelService,RoomService,PromotionService,EnterpriseService,OfflineRecordService,界面类库包,VO |
| LoginUI | UserUI,HotelUI,UserService,StaffService,WebMarketUI,WebStaffUI,界面类库包,VO |
| WebStaffUI | UserService,HotelService,StaffService,界面类库包,VO |
| WebMarketUI | UserService,HotelServiceCreditRecordService,界面类库包,VO |
| UserService |  |
| UserServiceImpl | EnterpriseService，UserDAO,PO |
| OrderSearchService |  |
| OrderSearchServiceImpl | OrderDAO,PO |
| OrderService |  |
| OrderServiceImpl | CreditRecordService,  HotelDiscountService, WebDiscountService,OrderDAO,PO |
| HotelService |  |
| HotelServiceImpl | OrderService,RoomService,  HotelDAO,PO |
| LocationService |  |
| LocationServiceImpl | LocationDAO,PO |
| CommCircleService |  |
| CommCircleServiceImpl | CommCircleDAO,PO |
| RoomService |  |
| RoomServiceImpl | RoomDAO,PO |
| StaffService |  |
| StaffServiceImpl | StaffDAO ，PO |
| CreditRecordService |  |
| CreditRecordServiceImpl | UserService,CreditRecordDAO,PO |
| HotelDiscountService |  |
| HotelDiscountServiceImpl | HotelDiscountDAO,PO |
| EnterpriseService |  |
| EnterpriseService Impl | EnterpriseDAO,PO |
| WebDiscountService |  |
| WebDiscountServiceImpl | WebDiscountDAO,PO |
| OfflineRecordService |  |
| OfflineRecordServiceImpl | OfflineRecordDAO,PO |
| UserDAO | JAVA RMI,PO |
| UserDAOImpl | databaseutility,PO |
| OrderDAO | JAVA RMI,PO |
| OrderDAOImpl | databaseutility,PO |
| HotelDAO | JAVA RMI,PO |
| HotelDAOImpl | databaseutility,PO |
| LocationDAO | JAVA RMI,PO |
| LocationDAOImpl | databaseutility,PO |
| CommCircleDAO | JAVA RMI,PO |
| CommCircleDAOImpl | databaseutility,PO |
| RoomDAO | JAVA RMI,PO |
| RoomDAOImpl | databaseutility,PO |
| StaffDAO | JAVA RMI,PO |
| StaffDAOImpl | databaseutility,PO |
| CreditRecordDAO | JAVA RMI,PO |
| CreditRecordDAOImpl | databaseutility,PO |
| HotelDiscountDAO | JAVA RMI,PO |
| HotelDiscountDAOImpl | databaseutility,PO |
| EnterpriseDAO | JAVA RMI,PO |
| EnterpriseDAOImpl | databaseutility,PO |
| WebDiscountDAO | JAVA RMI,PO |
| WebDiscountDAOImpl | databaseutility,PO |
| OfflineRecordDAO | JAVA RMI,PO |
| OfflineRecordDAOImpl | databaseutility,PO |
| VO |  |
| PO |  |
| databaseutility | java.JDBC |
| 界面类库包 |  |
| JAVA RMI | java.swing,java.awt,java.2D,java.3D |

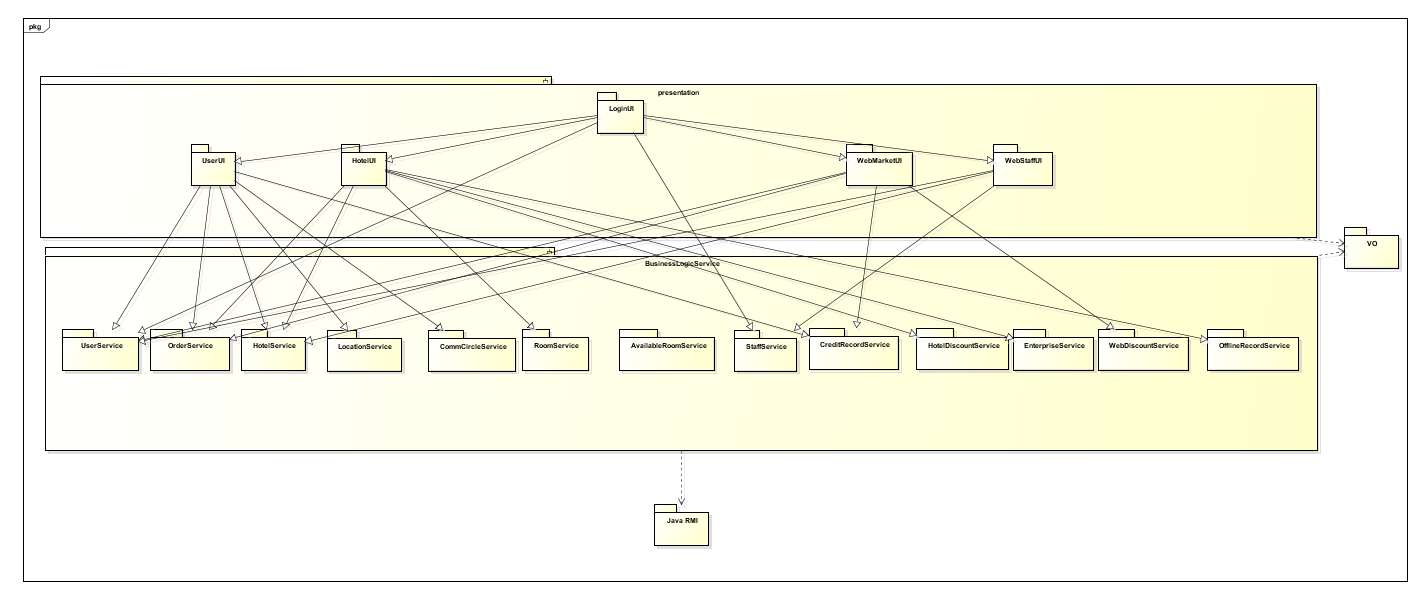


图4.1-1 酒店管理系统客户端开发包图

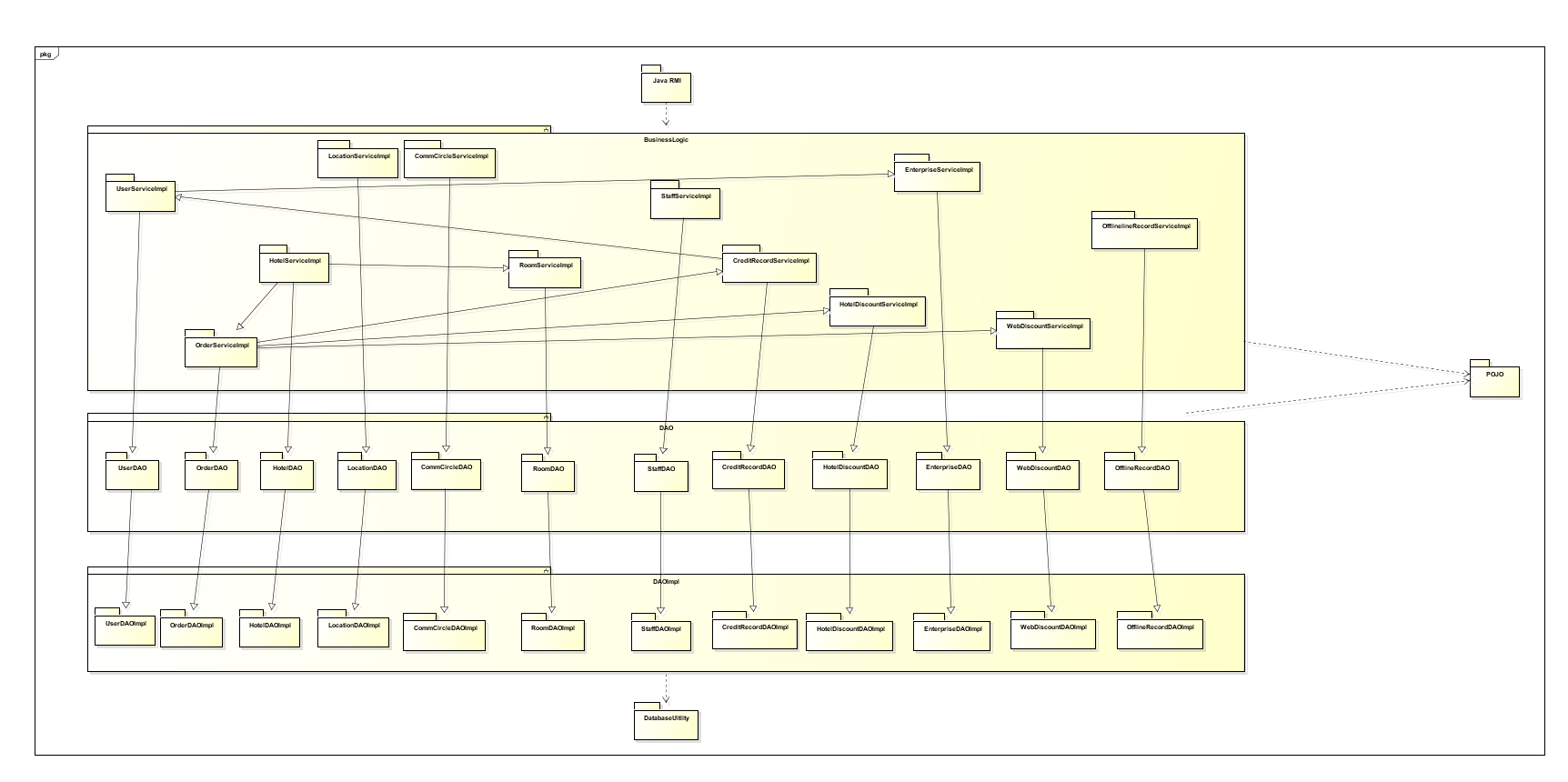


图4.1-2 酒店管理系统服务器端开发包图

### 4.2 运行时进程

在酒店管理系统中，会有多个客户端进程和一个服务器端进程，其进程图

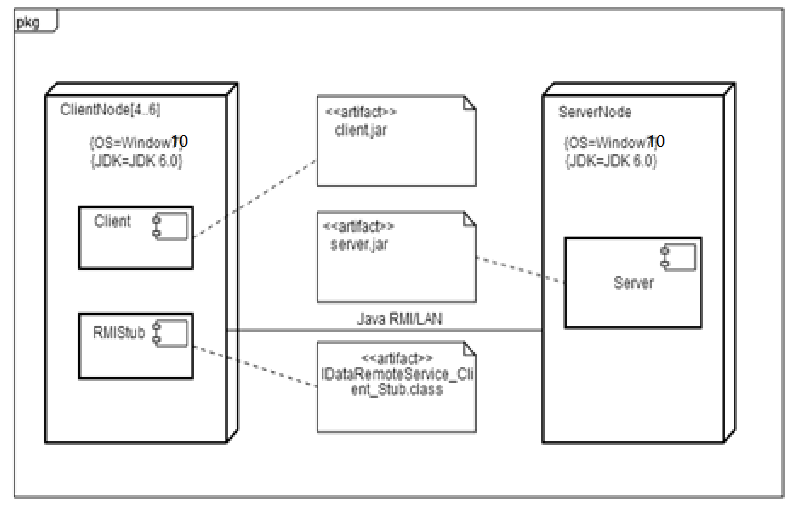
如图4.2所示。结合部署图，客户端进程实在客户端机器上运行，服务器端进程是在服务器端机器上运行。

 图4.2 进程图

### 4.3 物理部署

酒店管理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RMI构件。由于JavaRMI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，不需要再独立部署。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。

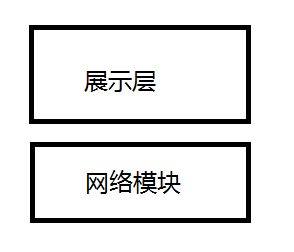


图5.1-1 客户端模块视图

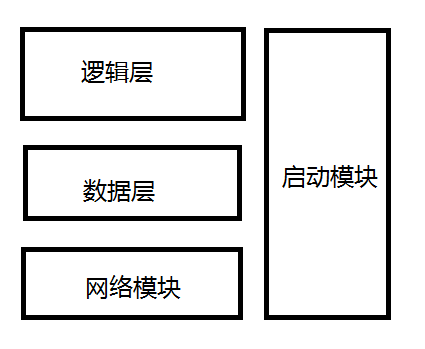


图5.1-2 服务器端模块视图

表5.1-1 客户端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 用户界面层 | 基于窗口的进销存系统客户端用户界面。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务,检测网络连接状态，进行断线重连 |

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。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 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 |
| UserService  OrderService  OrderSearchService  HotelService  LocationService  CommCircleService  RoomService  AvailableRoomService  StaffService  CreditRecordService  HotelDiscountService  EnterpriseService  WebDiscountService  OfflineRecordService | 客户端展示层 | 客户端业务逻辑层 |
| UserDAO  OrderDAO  HotelDAO  LocationDAO  CommCircleDAO AvailableRoomDAO  RoomDAO  StaffDAO  CreditRecordDAO  HotelDiscountDAO  EnterpriseDAO  WebDiscountDAO  OfflineRecordDAO | 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。

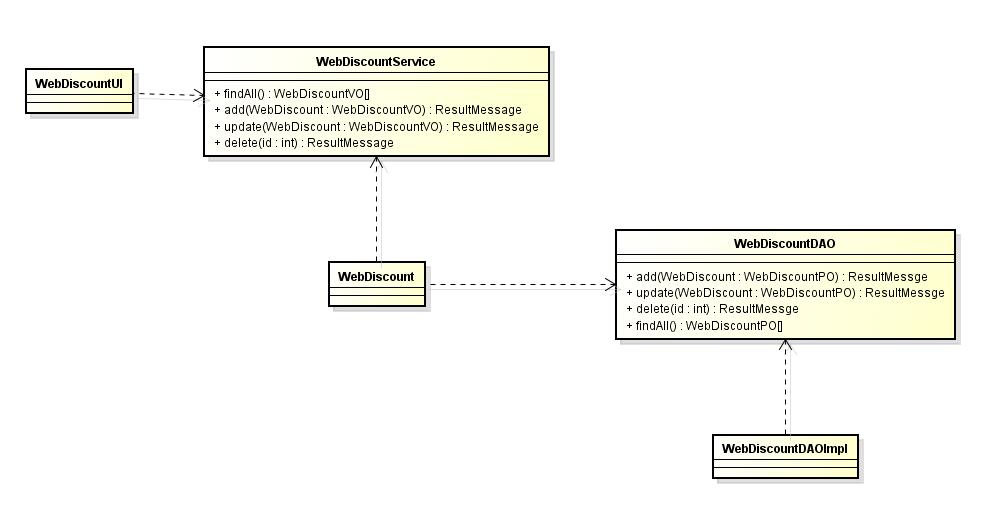


图5.1-3 网站促销策略用例层之间调用的接口

### 5**.2 用户界面层的分解**

根据需求，系统存在25个用户界面：注册登录界面，用户个人中心界面，个人信息界面，订单信息界面，酒店信息界面，信用记录界面，酒店查询界面，酒店信息展示界面，下单界面，酒店管理中心界面，酒店信息维护界面，酒店订单管理界面，录入客房界面，酒店促销策略界面，线下入住记录界面，网站营销人员界面，网站促销策略界面，异常订单管理界面，信用充值界面，网站信息管理界面，用户信息管理界面，用户信息界面，酒店工作人员界面，网站营销人员界面，酒店信息管理界面。

界面跳转如图5.2所示。

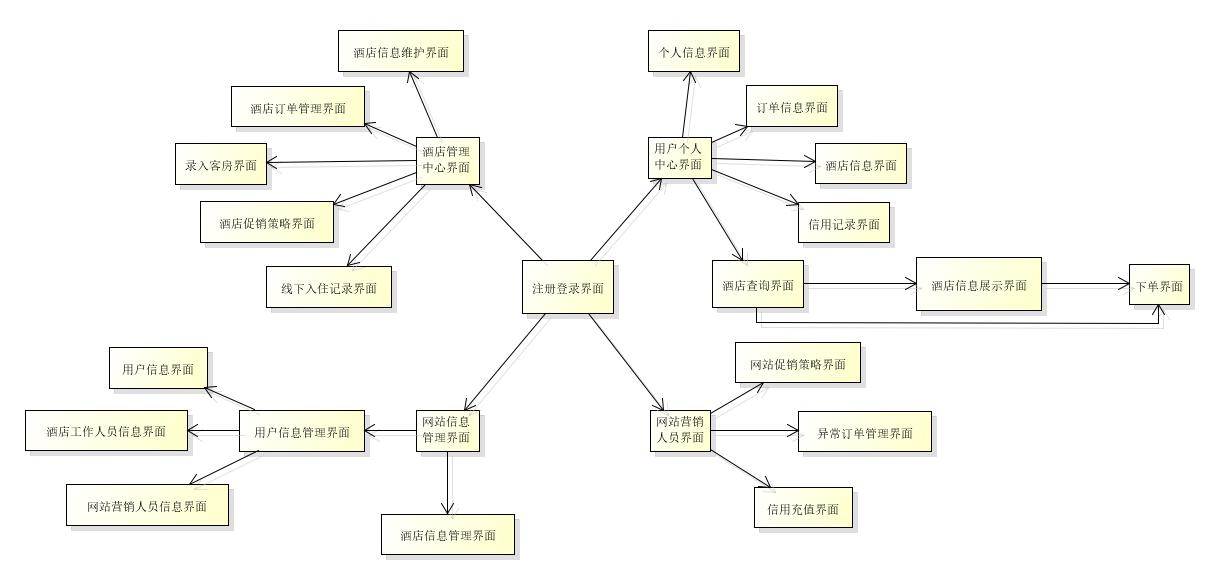


图5.2 用户界面跳转

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

|  |  |  |
| --- | --- | --- |
| Main\_ui | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel，登录成功后显示登录人员对应的功能界面。 |

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.UserBLService | 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。

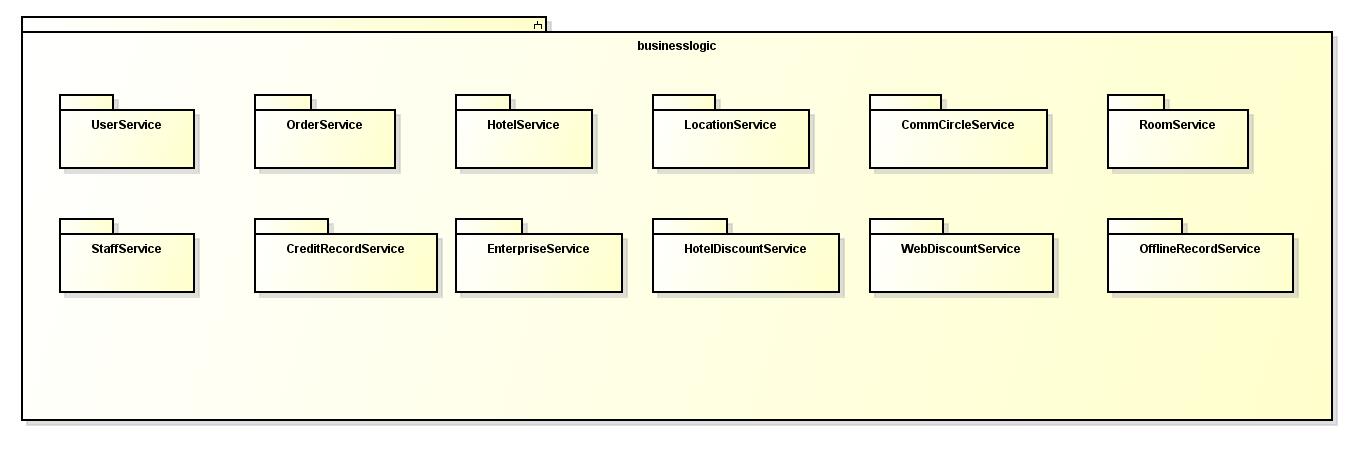


图5.3-1 业务逻辑层

#### 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 |
| UserService | 负责用户注册和所有用户的登录；负责用户信息管理 |
| OrderService | 负责订单的浏览、下单、撤销、评价，酒店的更新入住、退房、延迟入住信息、浏览订单，网站促销人员的浏览、撤销异常订单 |
| HotelService | 负责搜索浏览酒店，酒店基本信息管理；网站管理人员添加酒店；用户浏览酒店信息 |
| RoomService | 负责录入可用客房和可用客房管理 |
| StaffService | 负责网站管理人员管理用户、酒店工作人员、网站营销人员信息 |
| CreditRecordService | 负责用户查看信用记录，网站营销人员信用充值 |
| HotelDiscountService | 负责酒店工作人员制定酒店促销策略 |
| OfflineRecordService | 负责酒店工作人员更新线下入住信息 |
| WebDiscountService | 负责网站营销人员制定促销策略 |

#### 5.3.2 业务逻辑层模块的接口规范

表5.3.2-1 UserService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| UserService.findByUsername | 语法 | public UserVO findByUsername(String username) | |
| 前置条件 | 已知用户名，需要获得该用户的详细信息 | |
| 后置条件 | 无 | |
| UserService.register | 语法 | public Void register(UserVO) | |
| 前置条件 | 已知用户信息，需要将其注册 | |
| 后置条件 | 无 | |
| UserService.update | 语法 | public Void update(UserVO) | |
| 前置条件 | 已知该用户修改后的信息，需要更新该用户信息 | |
| 后置条件 | 更新数据库中用户的信息 | |
| UserService.login | 语法 | public UserVO login(String username,String password) | |
| 前置条件 | 已知该用户的用户名和密码，需要将其登录 | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| UserDAO.findByUsername(String username) | | | 查找单一持久化序列 |
| UserDAO.add(UserPO) | | | 添加单一持久化序列 |
| UserDAO.update(UserPO) | | | 更新单一持久化序列 |

表5.3.2-2 OrderService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderSearchService.findByID | 语法 | public OrderVO findByID(int id) | |
| 前置条件 | 已知该订单的ID，需要得到该订单的详细信息 | |
| 后置条件 | 返回该订单的详细信息 | |
| OrderSearchService.fingByHotelAndStatus | 语法 | Public List<OrderVO> findByHotelAndStatus(int hotelID,OrderStatus type) | |
| 前置条件 | 已知酒店的ID和订单状态，需要获得符合条件的订单列表 | |
| 后置条件 | 无 | |
| OrderSearchService.findByUsernameAndStatus | 语法 | public List<OrderVO>  findByUsernameAndStatus(String username,OrderStatus type) | |
| 前置条件 | 已知订单的所属用户和订单类型，需要获得符合条件的订单列表 | |
| 后置条件 | 无 | |
| OrderSearchService. findByUsername | 语法 | public List<OrderVO> findByUsername(String username) | |
| 前置条件 | 已知订单的所属用户，需要获得该用户的所有订单 | |
| 后置条件 | 无 | |
| OrderSearchService.findByHotelAndUsername | 语法 | public List<OrderVO> findByHotelAndUsername(int hotelID,String username) | |
| 前置条件 | 已知订单的所属酒店和所属用户，需要获得所有订单 | |
| 后置条件 | 无 | |
| OrderSearchService. findByOrderStatus | 语法 | public List<OrderVO> findByOrderStatus(OrderStatus status) | |
| 前置条件 | 已知订单的类型，需要获得所有该类型的订单 | |
| 后置条件 | 无 | |
| OrderSearchService. findByHotelAndTime | 语法 | public List<OrderVO> findByHotelAndTime(int hotelID,Date begin,Date end) | |
| 前置条件 | 已知酒店和时间，需要得到符合该条件的所有订单 | |
| 后置条件 | 无 | |
| OrderService.  placeOrder | 语法 | public OrderVO placeOrder(OrderVO) | |
| 前置条件 | 已知该订单部分信息，需要将订单信息进行完善 | |
| 后置条件 | 更新订单对象的信息 | |
| OrderService.add | 语法 | public void add(OrderVO) | |
| 前置条件 | 已知该订单信息，需要将其添加到数据库 | |
| 后置条件 | 在数据库中添加该订单信息 | |
| OrderService.checkin | 语法 | public void checkin(OrderVO ) | |
| 前置条件 | 已知订单信息，需要将该订单状态改为已入住 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| OrderService.checkout | 语法 | public void checkout(OrderVO ordervo) | |
| 前置条件 | 已知订单信息，需要将该订单的状态改为已退房 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| OrderService.revokeByUser | 语法 | public void revokeByUser(OrderVO ordervo) | |
| 前置条件 | 已知订单信息，用户需要将该订单的状态改为已撤销 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| OrderService.revokeByWebMarketer | 语法 | Public void revokeByWebMarketer(OrderVO ordervo,RestoreValueType type) | |
| 前置条件 | 已知订单信息，网站营销人员需要将该订单的状态改为已撤销 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| OrderService.checkAbNormal | 语法 | Public void checkAbNormalOrder() | |
| 前置条件 | 需要定期检查是否存在用户的为执行订单超时 | |
| 后置条件 | 无 | |
| OrderService.remark | 语法 | public void remark(OrderVO ordervo,int score ,String evaluation) | |
| 前置条件 | 已知订单信息，需要增加该订单的评价信息 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| OrderService.delayCheckin | 语法 | public void delayCheckin(OrderVO ordervo) | |
| 前置条件 | 已知订单信息，需要将该订单的状态改为已入住 | |
| 后置条件 | 在数据库中修改该订单状态 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| OrderDAO. findByID(int ID) | | | 查找单一持久化序列 |
| OrderDAO. findByUsernameAndStatus(String username,OrderStatus) | | | 查找单一持久化列表 |
| OrderDAO. findByUsername(String username) | | | 查找单一持久化列表 |
| OrderDAO. findByHotelAndUsername(int hotelID,String username) | | | 查找单一持久化列表 |
| OrderDAO.findByOrderStatus(OrderStatus) | | | 查找单一持久化列表 |
| OrderDAO. findByHotelAndTime(int hotelID,Date begin,Date end); | | | 查找单一持久化列表 |
| OrderDAO. add(OrderPO) | | | 添加单一持久化列表 |
| OrderDAO. update(OrderPO) | | | 更新单一持久化序列 |
| HotelDiscountService.findAllByHotelID(int hotelID) | | | 返回该酒店的所有优惠策略 |
| CreditRecordService. add(CreditRecordVO) | | | 添加当前用户的一条信用记录 |
| WebDiscountService.findAll() | | | 返回网站的所有优惠策略 |

表5.3.2-3 HotelService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelService. findByID | 语法 | public HotelVO findByID(int hotelID) | |
| 前置条件 | 已知该酒店的编号，需要该酒店的详细信息 | |
| 后置条件 | 无 | |
| HotelService.update | 语法 | public Void update(HotelVO) | |
| 前置条件 | 已知该酒店的更新后的信息，需要更新该酒店信息 | |
| 后置条件 | 在数据库中更新该酒店信息 | |
| HotelService.add | 语法 | public Void add(HotelVO) | |
| 前置条件 | 已知该酒店的信息，需要将其添加到数据库 | |
| 后置条件 | 在数据库中增加该酒店信息 | |
| HotelService.findOrderedHotelAndOrder | 语法 | public Map<HotelVO,List<OrderVO>>>  findOrderedHotelAndOrder(String username) | |
| 前置条件 | 已知用户名，需要用户曾预订的酒店列表以及每个酒店对应的订单信息 | |
| 后置条件 | 返回酒店列表 | |
| HotelService. find | 语法 | public Map<HotelVO,List<RoomVO>>>  find (int loc,int circle,Date begin,Date end) | |
| 前置条件 | 已知该酒店的地址、商圈、订单时间，得到可用房间 | |
| 后置条件 | 无 | |
| HotelService. filter | 语法 | public Map<HotelVO,List<RoomVO>>  filter(List<FilterCondition>) | |
| 前置条件 | 已知酒店列表，需要按照给定的过滤条件进行筛选 | |
| 后置条件 | 无 | |
| HotelService. order | 语法 | public Map<HotelVO,List<RoomVO>>  order(OrderRule rule,boolean isDecrease); | |
| 前置条件 | 已知酒店列表，需要按照给定条件进行排序 | |
| 后置条件 | 返回排好序的酒店列表 | |
| HotelService. getRoomDetail | 语法 | public List<RoomVO>  getRoomDetail(,int hotelID) | |
| 前置条件 | 已知酒店编号，需要取出该酒店的对应的房间信息 | |
| 后置条件 | 无 | |
| HotelService.addRemark | 语法 | Public void addRemark(HotelVO hotel,int score) | |
| 前置条件 | 需要评价酒店 | |
| 后置条件 | 无 | |
| LocationService.findAll | 语法 | public List<LocationVO> findAll() | |
| 前置条件 | 需要所有的地点信息 | |
| 后置条件 | 无 | |
| CommCircleService.findByLoc | 语法 | public List<CommercialCircleVO> findByLocation(int locID) | |
| 前置条件 | 已知地点，需要该地点的所有商圈 | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| HotelDAO.findByID(int hotelID) | | | 查找单一持久化序列 |
| HotelDAO.update(HotelPO) | | | 更新单一持久化序列 |
| HotelDAO.add(HotelPO) | | | 增加单一持久化序列 |
| HotelDAO. find (int loc,int circle,Date begin,Date end) | | | 查找单一持久化列表 |
| LocationDAO. findAll() | | | 查找单一持久化列表 |
| CommCircleDAO. findByLoc(int locID) | | | 查找单一持久化列表 |
| OrderService.findByUsername(String username) | | | 返回该用户的所有订单列表 |
| OrderService. findByHotelAndUsername(int hotelID,String username) | | | 返回该酒店的该用户的订单记录列表 |
| RoomService. findAvailableByHotelID (int hotelID) | | | 返回该酒店的所有可用房间信息 |

5.3.2-4 RoomService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| RoomService. findByHotelID | 语法 | public List<RoomVO> findByHotelID (int hotelID) | |
| 前置条件 | 已知酒店，需要房间详细信息列表 | |
| 后置条件 | 无 | |
| RoomService.findByHotelAndType | 语法 | Public RoomVO findByHotelAndType(int hotelID,RoomType type) | |
| 前置条件 | 已知酒店和房间类型，需要符合条件的有效房间信息 | |
| 后置条件 | 无 | |
| RoomService.findAvailableByHotelID | 语法 | public List<RoomVO> findAvailableByHotelID(int hotelID,Date begin,Date end) | |
| 前置条件 | 已知酒店和时间，需要在该时间段内有效的房间信息 | |
| 后置条件 | 无 | |
| RoomService.findNotAddedRoomType | 语法 | List<RoomType> findNotAddedRoomType(int hotelID) | |
| 前置条件 | 已知酒店编号，需要该酒店尚未添加过的房间类型列表 | |
| 后置条件 | 无 | |
| RoomService.update | 语法 | public Void update(RoomVO) | |
| 前置条件 | 已知房间更新后的信息，需要更新数据库中的信息 | |
| 后置条件 | 在数据库中更新房间信息 | |
| RoomService.add | 语法 | public Void add(RoomVO) | |
| 前置条件 | 已知房间信息，需要将其添加到数据库中 | |
| 后置条件 | 在数据库中保存新添加的房间信息 | |
| RoomService. findAvailableRoomNum | 语法 | public int findAvailableRoomNum(int hotelID,RoomType type,Date begin,Date end) | |
| 前置条件 | 已知酒店、房间类型和时间，需要可用房间数量 | |
| 后置条件 | 返回可用房间数量 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| RoomDAO. findByHotelAndType(int hotelID,RoomType) | | | 查找单一持久化序列 |
| RoomDAO. findByHotelID(int hotelID) | | | 查找单一持久化列表 |
| RoomDAO. add(RoomPO) | | | 增加单一持久化序列 |
| RoomDAO. update(RoomPO) | | | 更新单一持久化序列 |
| RoomDAO. findAvailableRoomNum (int hotelID,RoomType type,Date begin,Date end) | | | 返回在当前酒店和房间类型下一段时间内可用房间数量 |

表5.3.2-5 StaffService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| StaffService.login | 语法 | public StaffVO login(String username,String password) | |
| 前置条件 | 已知用户名和密码，需要进行登录 | |
| 后置条件 | 无 | |
| StaffService.update | 语法 | public Void update(StaffVO) | |
| 前置条件 | 已知更新后的职员信息，需要更新数据库中的信息 | |
| 后置条件 | 在数据库中更新职员信息 | |
| StaffService.add | 语法 | public Void add(StaffVO) | |
| 前置条件 | 已知职员信息，需要将其添加到数据库中 | |
| 后置条件 | 在数据库中保存新添加的职员信息 | |
| StaffService.findByUsername | 语法 | public StaffVO findByUsername(String username) | |
| 前置条件 | 已知用户名，需要返回对应的职员信息 | |
| 后置条件 | 无 | |
| StaffService.findByHotelName | 语法 | public StaffVO findByHotelName(String hotelName) | |
| 前置条件 | 已知酒店名称，需要返回对应的职员信息（如果是酒店工作人员） | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| StaffDAO.update(StaffPO) | | | 更新单一持久化序列 |
| StaffDAO. add(StaffPO) | | | 添加单一持久化序列 |
| StaffDAO. findByUsername(String username) | | | 查找单一持久化序列 |
| StaffDAO.findByHotelName(String hotelName); | | | 查找单一持久化序列 |

表5.3.2-6 CreditRecordService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CreditRecordService. findByUsername | 语法 | public List<CreditRecordVO> findByUsername(String username) | |
| 前置条件 | 已知用户名，需要该用户的所有信用记录 | |
| 后置条件 | 返回该用户的所有信用记录 | |
| CreditRecordService.add | 语法 | public Void add(CreditRecordVO) | |
| 前置条件 | 已知信用记录的信息，需要将其添加到数据库 | |
| 后置条件 | 返回添加的结果信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| CreditRecordDAO. findByUsername(String username) | | | 查找单一持久化列表 |
| CreditRecordDAO. add(CreditRecordPO) | | | 增加单一持久化序列 |
| UserService. findByUsername(String username) | | | 查找用户的原始信用值 |
| UserService. update(UserVO) | | | 更新用户的信用值信息 |

表5.3.2-7 HotelDiscountService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelDiscountService. add | 语法 | public Void add (HotelDiscountVO) | |
| 前置条件 | 已知酒店促销策略信息，需要将其添加到数据库 | |
| 后置条件 | 返回添加的结果信息 | |
| HotelDiscountService.update | 语法 | public Void update(HotelDiscountVO) | |
| 前置条件 | 已知更新后的酒店促销策略信息，需要更新数据库 | |
| 后置条件 | 返回更新的结果信息 | |
| HotelDiscountService.delete | 语法 | public Void delete(int id) | |
| 前置条件 | 已知酒店优惠策略的编号，需要将其删除 | |
| 后置条件 | 返回删除的结果信息 | |
| HotelDiscountService.findAllByHotelID | 语法 | public List<HotelDiscountVO> findAllByHotelID(int hotelID) | |
| 前置条件 | 已知酒店编号，需要该酒店所有的优惠策略 | |
| 后置条件 | 返回该酒店的所有优惠策略 | |
| HotelDiscountService.createAllStrategies | 语法 | public List<HotelDiscount> createAllStrategies(int hotelID); | |
| 前置条件 | 已知酒店编号，需要返回该酒店的所有策略领域对象 | |
| 后置条件 | 无 | |
| EnterpriseService.getAllEnterprises | 语法 | public List<EnterpriseVO> getAllEnterprises() | |
| 前置条件 | 无，需要所有的企业信息 | |
| 后置条件 | 无 | |
| EnterpriseService.add | 语法 | public Void add(EnterpriseVO ); | |
| 前置条件 | 已知企业信息，需要将其添加到数据库 | |
| 后置条件 | 在数据库中保存新添加的酒店信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| HotelDiscountDAO.add (HotelDiscountPO) | | | 添加单一持久化序列 |
| HotelDiscountDAO update(HotelDiscountPO) | | | 更新单一持久化序列 |
| HotelDiscount.delete(int id) | | | 删除单一持久化序列 |
| HotelDiscount.findAllByHotelID(int hotelID); | | | 查询单一持久化列表 |
| EnterpriseDAO.findAll(); | | | 返回企业信息列表 |
| EnterpriseDAO.add(EnterprisePO ) | | | 新增单一持久化序列 |

表5.3.2-8 OfflineRecordService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OfflineRecordService. findByID | 语法 | public OfflineRecordVO findByID(int id) | |
| 前置条件 | 已知线下记录编号，需要线下记录的信息 | |
| 后置条件 | 返回该编号对应的线下记录信息 | |
| OfflineRecordService.checkin | 语法 | public Void checkin(OfflineRecordVO) | |
| 前置条件 | 已知线下记录信息，需要将该记录状态改为已入住 | |
| 后置条件 | 在数据库中更新线下记录信息 | |
| OffLineRecordService.checkout | 语法 | public Void checkout(OfflineRecordVO) | |
| 前置条件 | 已知线下记录信息，需要将该记录状态改为已退房 | |
| 后置条件 | 在数据库中更新线下记录信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| OfflineRecordDAO. findByID(int id) | | | 查找单一持久化序列 |
| OfflineRecordDAO.add(OfflineRecordPO) | | | 添加单一持久化序列 |
| OfflineRecordDAO.update(OfflineRecordPO) | | | 更新单一持久化序列 |

表5.3.2-9 WebDiscountService模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| WebDiscountService.findAll | 语法 | public List<WebDiscountVO> findAll() | |
| 前置条件 | 无，需要网站所有促销策略信息 | |
| 后置条件 | 返回网站所有促销策略信息 | |
| WebDiscountService.add | 语法 | public Void add (WebDiscountVO) | |
| 前置条件 | 已知网站促销策略信息，需要将其添加到数据库 | |
| 后置条件 | 返回添加的结果信息 | |
| WebDiscountService. update | 语法 | public Void update(WebDiscountVO) | |
| 前置条件 | 已知更新后的网站促销策略信息，需要将数据库中的信息进行更新 | |
| 后置条件 | 返回更新的结果信息 | |
| WebDiscountService.createAllStrategies | 语法 | public List<WebDiscount> createAllStrategies(); | |
| 前置条件 | 需要返回网站的所有策略领域对象 | |
| 后置条件 | 无 | |
| WebDiscount.delete | 语法 | public Void delete(int id) | |
| 前置条件 | 已知网站促销策略编号，需要将其从数据库中删除 | |
| 后置条件 | 返回删除的结果信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| WebDiscountDAO. add (WebDiscountPO) | | | 添加单一持久化序列 |
| WebDiscountDAO.update | | | 更新单一持久化序列 |
| WebDiscount. delete | | | 删除单一持久化序列 |
| WebDiscount.findAll | | | 查找单一持久化列表 |

### 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 |
| DataServiceSpringImpl | 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UserDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDAO.findByUsername | 语法 | Public UserPO findByUsername  (String username) |
| 前置条件 | 无 |
| 后置条件 | 返回对应username的PO记录 |
| UserDAO.add | 语法 | public Void add(UserPO) |
| 前置条件 | 逻辑层请求添加新的user信息，且同样的username的PO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个user的PO记录 |
| UserDAO.update | 语法 | Public Void update(UserPO) |
| 前置条件 | 逻辑层请求更新user的信息，且对应的userPO在数据文件中已存在 |
| 后置条件 | 在数据文件中更新该user的PO记录 |

表5.4.2-2数据层OrderDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDAO.findByID | 语法 | public OrderPO findByID(int ID); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByUsernameAndStatus | 语法 | Public List<OrderPO> findByUsernameAndType(String username,OrderStatus) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByHotelAndUsername | 语法 | public List<OrderPO> findByHotelAndUsername(int hotelID,String username) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByUsername | 语法 | Public List<OrderPO> findByUsername(String username) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByHotelAndStatus | 语法 | public List<OrderPO>  findByHotelAndStatus(int hotelID,OrderStatus type) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByOrderStatus | 语法 | public List<OrderPO>  findByOrderStatus(OrderStatus) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.findByHotelAndTime | 语法 | public List<OrderPO> findByHotelAndTime(int hotelID,Date begin,Date end); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDAO.add | 语法 | public Void add  (OrderPO); |
| 前置条件 | 逻辑层请求添加订单记录，且同样ID的PO在数据文件中不存在 |
| 后置条件 | 在数据库中添加一个Order记录 |
| OrderDAO.update | 语法 | public Void update  (OrderPO) |
| 前置条件 | 逻辑层请求更新订单记录，且对应的  OrderPO记录在数据文件中已存在 |
| 后置条件 | 在数据文件中更新OrderPO记录 |

表5.4.2-3数据层HotelDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelDAO.findByID | 语法 | public HotelPO findByID(int hotelID); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| HotelDAO.update | 语法 | public Void update  (HotelPO); |
| 前置条件 | 逻辑层请求更新hotelPO记录，且对应hotelPO记录在数据文件中已存在 |
| 后置条件 | 在数据文件中更新对应hotel的PO记录 |
| HotelDAO.add | 语法 | public Void add  (HotelPO); |
| 前置条件 | 逻辑层请求添加新的hotel记录，且相同ID的hotel在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| HotelDAO.find | 语法 | public List<HotelPO> find (int loc,int circle); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-4数据层LocationDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LocationDAO.findAll | 语法 | public List<LocationPO> findAll(); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-5数据层CommCircleDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommCircleDAO. findByLoc | 语法 | public List<CommercialCirclePO>  findByLoc(int locID); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-6数据层RoomDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomDAO.findAvailableRoom(int hotelID,RoomType,Data data) | 语法 | Public int findAvailableRoom(int hotelID,RoomType,Data data) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| RoomDAO.findByHotelAndType | 语法 | public RoomPO find(int hotelID,  RoomType); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| RoomDAO.update | 语法 | public Void update  (RoomPO); |
| 前置条件 | 逻辑层请求更新Room的PO记录，且对应ID的PO在数据文件中已存在 |
| 后置条件 | 在数据文件中更新PO |
| RoomDAO.add | 语法 | public Void add  (RoomPO); |
| 前置条件 | 逻辑层请求添加新的room记录，且相同ID的PO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| RoomDAO.findByHotelID | 语法 | Public LIst<RoomPO>  findByHotelID(int hotelID) |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-7数据层StaffDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StaffDAO.update | 语法 | public Void update(StaffPO); |
| 前置条件 | 逻辑层请求更新Staff的PO记录，且对应ID的PO在数据文件中已存在 |
| 后置条件 | 在数据文件中更新PO |
| StaffDAO.add | 语法 | public Void add(StaffPO); |
| 前置条件 | 逻辑层请求添加新的staff记录，且相同ID的PO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| StaffDAO.findByHotelName | 语法 | public StaffPO findByHotelName(String hotelname); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| StaffDAO.findByUsername | 语法 | public StaffPO findByUsername(String username); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-8数据层CreditRecordDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditRecordDAO.  findByUsername | 语法 | public List<CreditRecordPO> findByUsername(String username); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| CreditRecordDAO.add | 语法 | public Void add  (CreditRecordPO); |
| 前置条件 | 逻辑层请求增加新的信用记录 |
| 后置条件 | 在数据文件中增加一条信用记录 |

表5.4.2-9数据层PromotionDAO模块的接口规范(HotelDiscount)

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionDAO.addHotelDiscount | 语法 | public Void addHotelDiscount  (HotelDiscountPO); |
| 前置条件 | 逻辑层请求添加新的HotelDiscount记录，且相同ID的PO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| PromotionDAO.updateHotelDiscount | 语法 | public Void updateHotelDiscount  (HotelDiscountPO) |
| 前置条件 | 逻辑层请求更新HotelDiscount的PO记录，且对应ID的PO在数据文件中已存在 |
| 后置条件 | 在数据文件中更新PO |
| PromotionDAO.deleteHotelDiscount | 语法 | public Void deleteHotelDiscount (int id); |
| 前置条件 | 相同ID的PO在数据文件中已存在 |
| 后置条件 | 删除一个PO记录 |
| PromotionDAO.findHotelDiscountByHotelID | 语法 | public List<HotelDiscountPO>  findHotelDiscountByHotelID(int hotelID); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-10数据层EnterpriseDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| EnterpriseDAO. findAll | 语法 | public List< EnterprisePO > findAll(); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| EnterpriseDAO.add | 语法 | Public Void add(EnterprisePO po) |
| 前置条件 | 逻辑层请求添加新的Enterprise记录，且同样的EnterprisePO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |

表5.4.2-11数据层WebDiscountDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WebDiscountDAO.add | 语法 | public Void add  (WebDiscountPO); |
| 前置条件 | 逻辑层请求添加新的WebDiscount记录，且同样的WebDiscountPO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| WebDiscountDAO  .update | 语法 | public Void update  (WebDiscountPO); |
| 前置条件 | 逻辑层请求更新WebDiscount,且对应ID的PO记录在数据文件中已存在 |
| 后置条件 | 在数据文件中更新 WebDiscou的PO记录 |
| WebDiscountDAO  .delete | 语法 | public Void delete(int id); |
| 前置条件 | 相同ID的PO在数据文件中已存在 |
| 后置条件 | 删除一个PO记录 |
| WebDiscountDAO  .findAll | 语法 | public List<WebDiscountPO>  findAll(); |
| 前置条件 | 无 |
| 后置条件 | 无 |

表5.4.2-12数据层OfflineRecordDAO模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OfflineRecordDAO.findByID | 语法 | public OfflineRecordPO findByID(int id); |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OfflineRecordDAO.add | 语法 | public Void add  (OfflineRecordPO); |
| 前置条件 | 逻辑层请求添加新的OfflineRecord记录，且同样的线下记录PO在数据文件中不存在 |
| 后置条件 | 在数据文件中增加一个PO记录 |
| OfflineRecordDAO.update | 语法 | Public Void update(OfflineRecordPO); |
| 前置条件 | 逻辑层请求更新线下记录，且对应的PO在数据文件中已存在 |
| 后置条件 | 数据文件中更新线下记录PO |

## 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，如表6.1所示：

表6.1 PO类简单介绍

|  |  |
| --- | --- |
| 类名 | 包含的属性 |
| CommercialCirclePO | 商圈信息：商圈名称、编号，商圈位置编号 |
| CreditRecordPO | 信用值变化信息：编号；用户信息；订单信息；信用值变化类型、信用值变化量；当前信用值 |
| EnterprisePO | 企业信息：企业名称、企业编号 |
| HotelDiscountPO | 酒店促销策略信息：折扣信息编号；酒店信息；合作企业信息；享受折扣房间数；优惠策略、折扣力度；优惠开始时间、结束时间； |
| HotelPO | 酒店信息：酒店编号、名称、星级、评分、地址信息、商圈信息；街道名称、服务设施、简介；员工信息 |
| LocationPO | 位置信息：位置编号、名称 |
| OfflineRecord | 线下入住信息：线下入住事务编号；酒店信息；入住日期、预计离开日期、离开日期；房间类型、数量 |
| OrderPO | 订单信息：订单编号、状态、价值； 订单生成时间、最晚订单执行时间、入住时间、预期退房时间、退房时间、撤销时间；酒店信息；房间类型、数量；是否有孩子；用户信息；用户评分、评价；人数 |
| RoomPO | 房间信息：编号；酒店编号；房间类型、数量、价格 |
| StaffPO | 员工信息：用户名、密码、姓名；员工类型、酒店信息 |
| UserPO | 用户信息：用户名、密码、姓名、手机号、信用值、生日、所在企业、会员等级、用户类型 |
| WebDiscountPO | 网站促销策略信息：编号；折扣力度；策略类型、开始时间、结束时间；商圈信息；享受策略的会员等级 |

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

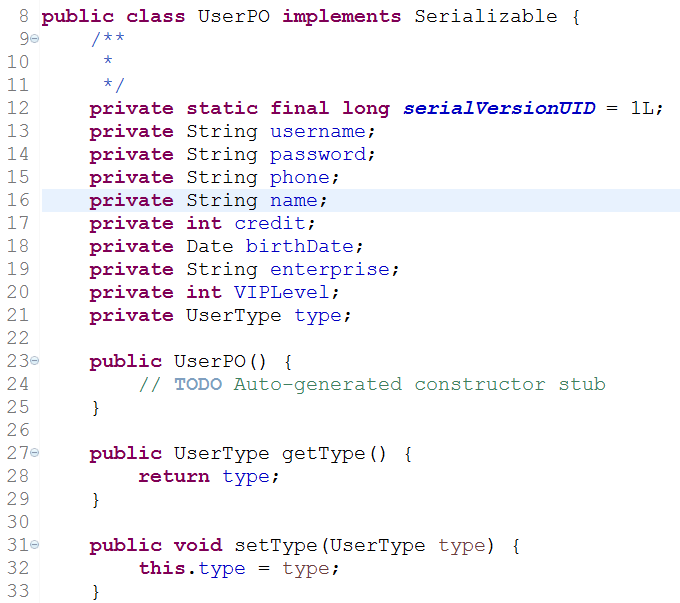


图6.1 持久化用户对象UserPO的定义

**6.2 数据库表**

|  |  |  |
| --- | --- | --- |
| 表名 | 内容 | 字段设计 |
| available\_room | 空房信息 |  |
| commercial\_circle | 商圈信息 |  |
| credit\_record | 信用变更记录 |  |
| enterprise | 企业信息 |  |
| hotel | 酒店信息 |  |
| hotel\_discount | 酒店促销策略 |  |
| hotel\_order | 酒店订单 |  |
| location | 地区 |  |
| offline\_record | 线下入住记录 |  |
| room | 房间信息 |  |
| room\_type | 房间类型 |  |
| staff | 员工信息 |  |
| user | 用户信息 |  |
| website\_discount | 网站促销策略 |  |