酒店预订系统

体系结构设计文档

V1.1.3

南京大学软件学院

**Leftovers**工作组

2016-10

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 林志和 | 2016-10-10 | 确定整体模板，目录等 | V1.0 |
| 郭浩滨 | 2016-10-10 | 添加逻辑设计方案及客户端、服务器端开发包图 | V1.0.1 |
| 林志和 | 2016-10-10 | 添加词汇表、开发（物理）包图的表 | V1.0.2 |
| 吴志成 | 2016-10-10 | 添加界面跳转图，部署图，客户端、服务器端模块视图 | V1.0.3 |
| 吴游杰 | 2016-10-10 | 添加业务逻辑层设计图 | V1.0.4 |
| 郭浩滨 | 2016-10-11 | 修改部分接口表，审查与修改模板 | V1.0.5 |
| 郭浩滨、林志和 | 2016-10-12 | 增加userbl的接口规范 | V1.0.6 |
| 郭浩滨 | 2016-10-13 | 增加promotionbl, orderbl的接口规范，删除credit包，并修改逻辑设计方案及客户端、服务器端开发包、业务层设计图 | V1.0.7 |
| 林志和 | 2016-10-14 | 增加hotelbl, memberbl,  HotelDataService,MemberDataService | V1.0.8 |
| 郭浩滨、林志和 | 2016-10-14 | 修改userbl, memberbl | V1.0.9 |
| 林志和 | 2016-10-14 | 修改hotelbl, promotionbl | V1.0.10 |
| 郭浩滨 | 2016-10-15 | 修改orderbl，修改promotionbl，增加OrderDataService，PromotionDataService | V1.0.11 |
| 林志和 | 2016-10-15 | 增加UserDataService接口规范，数据层模块的职责，DataFactory接口规范，数据层模块描述图  修改MemberDataService接口规范，HotelDataService接口规范  美化文档，统一文档相同意思的用词 | V1.0.12 |
| 郭浩滨 | 2016-10-15 | 修改promotionbl,hotelbl，orderbl | V1.1.0 |
| 郭浩滨 | 2016-10-16 | 修改init接口规范及整理小部分的接口 | V1.1.1 |
| 吴志成 | 2016-10-19 | 修改界面跳转图：增加了客户、酒店工作人员、网站营销人员、网站管理人员主界面 | V1.1.2 |
| 郭浩滨 | 2016-10-27 | 增加了hotel对promotion的依赖 | V1.1.3 |

目录

[1. 引言 4](#_Toc463824208)

[1.1. 编制目的 4](#_Toc463824209)

[1.2. 词汇表 4](#_Toc463824210)

[1.3. 参考资料 4](#_Toc463824211)

[2. 产品描述 4](#_Toc463824212)

[3. 逻辑视角 5](#_Toc463824213)

[4. 组合视角 6](#_Toc463824214)

[4.1. 开发包图 6](#_Toc463824215)

[4.2. 运行时进程 9](#_Toc463824216)

[4.3. 物理部署 10](#_Toc463824217)

[5. 接口视角 10](#_Toc463824218)

[5.1. 模块的职责 10](#_Toc463824219)

[5.2. 用户界面层的分解 12](#_Toc463824220)

[5.2.1. 用户界面层模块的职责 13](#_Toc463824221)

[5.2.2. 用户界面模块的接口规范 14](#_Toc463824222)

[5.2.3. 用户界面模块设计原理 14](#_Toc463824223)

[5.3. 业务逻辑层的分解 14](#_Toc463824224)

[5.3.1. 业务逻辑层模块的职责 14](#_Toc463824225)

[5.3.2. 业务逻辑模块的接口模范 15](#_Toc463824226)

[5.4. 数据层的分解 33](#_Toc463824227)

[5.4.1. 数据层模块的职责 34](#_Toc463824228)

[5.4.2. 数据层模块的接口规范 34](#_Toc463824229)

[6. 信息视角 48](#_Toc463824230)

[6.1. 数据持久化对象 48](#_Toc463824231)

[6.2. 文件格式 49](#_Toc463824232)

# 引言

## 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编写，是了解系统的导航。

## 词汇表

|  |  |
| --- | --- |
| **缩写或单词** | **解释** |
| HOS | 酒店预订系统 |
| member | 会员 |
| user | 用户 |
| hotel | 酒店 |
| room | 客房 |
| credit | 信用 |
| evaluation | 评价 |
| comment | 评论 |
| rank | 评分 |
| hotelpromotion | 酒店促销策略 |
| webpromotion | 网站营销策略 |
| order | 订单 |
| unexecuted | 未执行的（订单） |
| executed | 已执行的（订单） |
| abnormal | 异常的（订单） |
| canceled | 已撤销的（订单） |
| appeal | 异常订单申诉 |

## 参考资料

《酒店预订系统用例文档V1.2.11》

《酒店预订系统需求规格说明文档V1.1.5》

# 产品描述

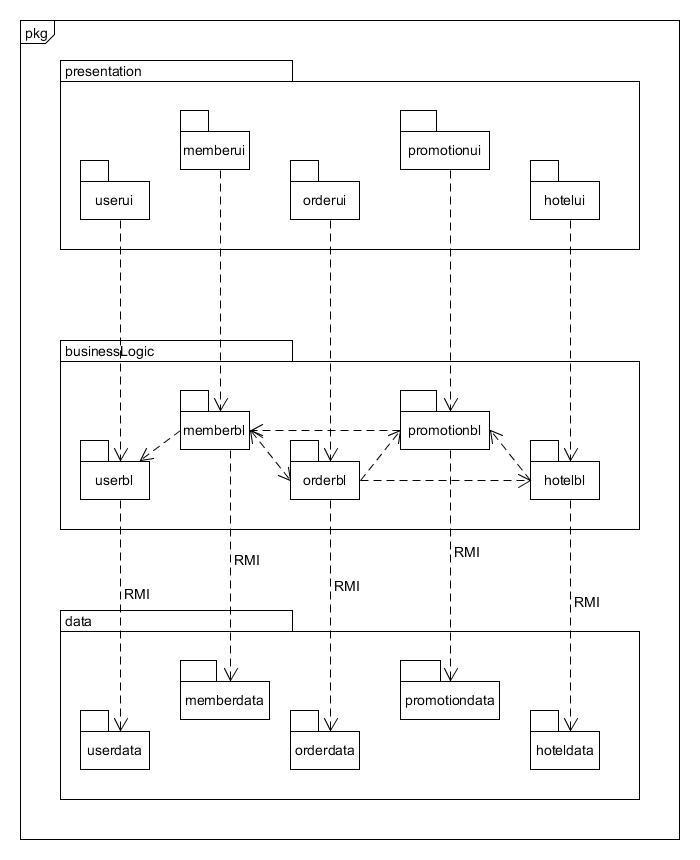
参考《酒店预订系统用例文档》和《酒店预订系统需求规格说明文档》中对产品的概括描述。

# 逻辑视角

酒店预订系统中，选择了分层体系结构的风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好的示意整个高层抽象。展示层包括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1：分层结构**



**图2：逻辑设计方案**

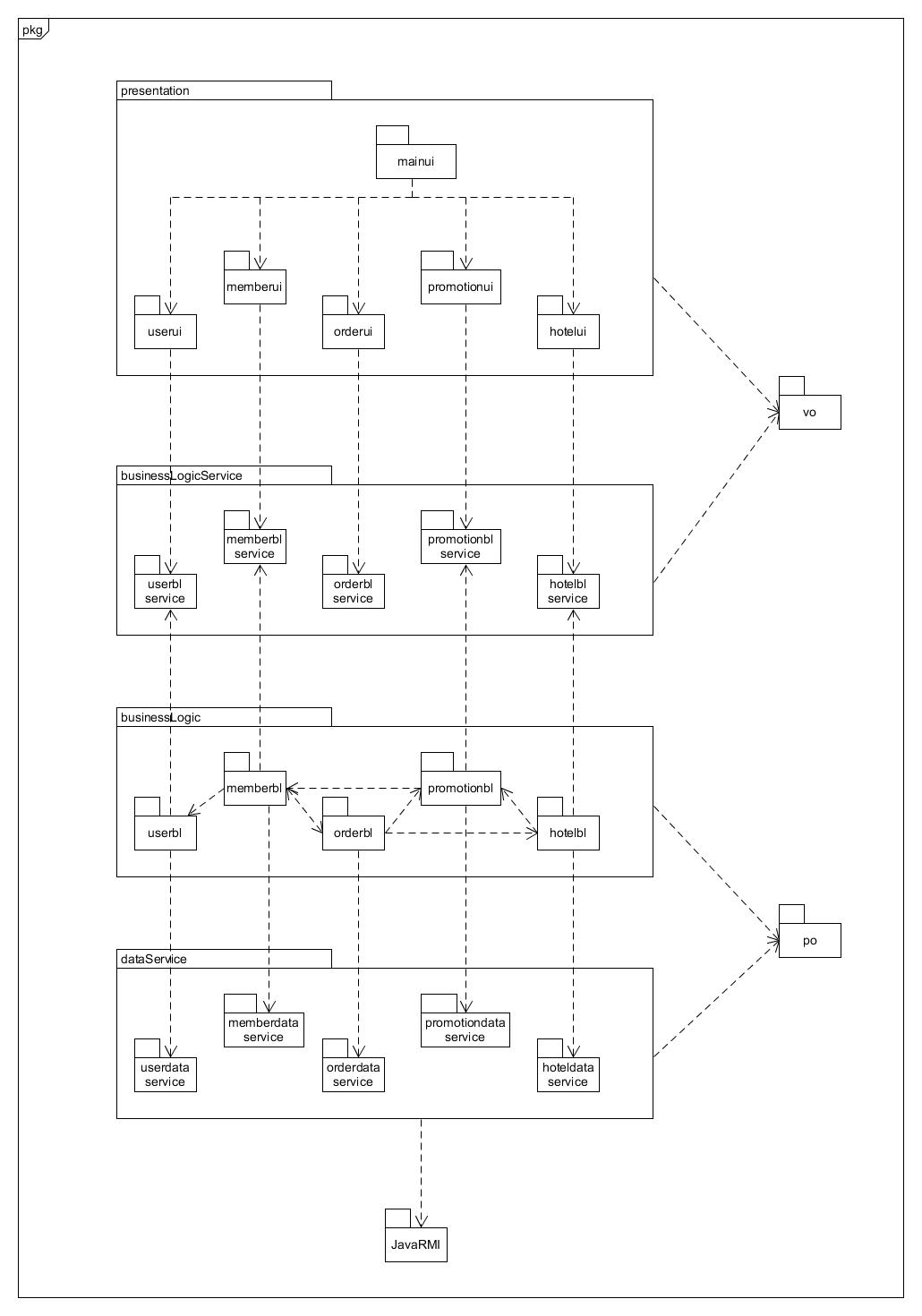
# 组合视角

## 开发包图

酒店预订系统的最终开发包设计如表1所示：

**表1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 | userui, memberui, orderui, promotionui, hotelui, creditui, vo |
| userui | userblservice, 界面类库包, vo |
| userblservice |  |
| userbl | userdataservice, userblservice, po |
| userdataservice | Java RMI, po |
| userdata | userdataservice, databaseseutility, po |
| memberui | memberblservice, 界面类库包, vo |
| memberblservice |  |
| memberbl | memberdataservice, memberblservice, userbl, orderbl, po |
| memberdataservice | Java RMI,po |
| memeberdata | memberdataservice, databaseseutility, po |
| orderui | orderblservice, 界面类库包, vo |
| orderblservice |  |
| orderbl | orderdataservice, orderblservice, memberbl, promotionbl, hotelbl, po |
| orderdataservice | Java RMI, po |
| orderdata | orderdataservice, databaseseutility, po |
| promotionui | promotionblservice, 界面类库包, vo |
| promotionblservice |  |
| promotionbl | promotiondataservice, promotionblservice, memberbl, po |
| promotiondataservice | Java RMI, po |
| promotiondata | promotiondataservice, databaseseutility, po |
| hotelui | hotelblservice, 界面类库包, vo |
| hotelblservice |  |
| hotelbl | hoteldataservice, hotelblservice, po |
| hoteldataservice | Java RMI, po |
| hoteldata | hoteldataservice, databaseseutility, po |
| creditui | creditblservice, 界面类库包, vo |
| creditblservice |  |
| creditbl | creditdataservice, creditblservice, po |
| creditdataservice | Java RMI, po |
| creditdata | creditdataservice, databaseseutility, po |
| vo |  |
| po |  |
| 界面类库包 |  |
| databaseseutility | JDBC |
| Java RMI |  |



**图3 客户端开发包图**



**图4 服务器端开发包图**

## 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，服务器端进程是在服务器端机器上运行。

**图5 进程图**



## 物理部署

酒店预订系统中客户端构件是放在客户端机上，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 构件属于JDK 1.8的一部分。所以，在系统JDK环境已经设置好的情况下，不需要独立部署。部署图如图6所示。



**图 6 部署图**

# 接口视角

## 模块的职责

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

**启动模块**

**业务逻辑层**

**展示层**

**网络模块**

**图7 客户端模块视图**

**网络模块**

**启动模块**

**业务逻辑层**

**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 | 基于窗口的酒店预订系统客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用java RMI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4所示。

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| Userbl  Memberbl  Hotelbl  Orderbl  Promotionbl | 客户端展示层 | 客户端业务逻辑层 |
| UserDataService  MemberDataService  HotelDataService  OrderDataService  PromotionDataService | 客户端业务逻辑层 | 服务器端数据层 |

## 用户界面层的分解

根据需求，系统存在37个用户界面（加粗为主界面）：**客户登录界面、客户界面**（客户）基本信息界面、（客户）编辑信息界面、信用历史记录界面、注册会员界面、（客户）订单列表界面、评价订单界面、撤销订单界面、预订过的酒店界面、酒店信息详情界面、搜索酒店界面、生成订单界面、**酒店工作人员登录界面、酒店工作人员界面**（酒店）基本信息界面、（酒店）编辑信息界面、录入可用客房界面、（酒店）制定促销策略界面、更新入住信息界面、更新退房信息界面、（酒店）订单列表界面、**网站营销人员登录界面、网站营销人员界面、**（网站）制定促销策略界面、（网站）订单列表界面、处理订单申诉界面、信用充值界面、**网站管理人员登录界面、网站管理人员界面、**酒店管理界面、用户管理界面、客户管理界面、酒店工作人员管理界面、网站营销人员管理界面。



**图9 酒店预订系统界面跳转图**

### 用户界面层模块的职责

**表5 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 |

### 用户界面模块的接口规范

**表6 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | init(args:String[]) |
| 前置条件 | 用户触发相应需要显示界面的条件 |
| 后置条件 | 显示Frame并且加载LoginPanel |

**表7 用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| bl.accountservice | 登录界面的业务逻辑接口 |
| bl.\*blservice | 每个ui模块会有对应的数据逻辑接口 |

### 用户界面模块设计原理

用户界面拟用java的javaFx库来实现。

## 业务逻辑层的分解

业务逻辑层包括多个针对界面及业务逻辑处理对象。例如User对象负责处理登录界面

的业务逻辑。业务逻辑层的设计如下图所示：

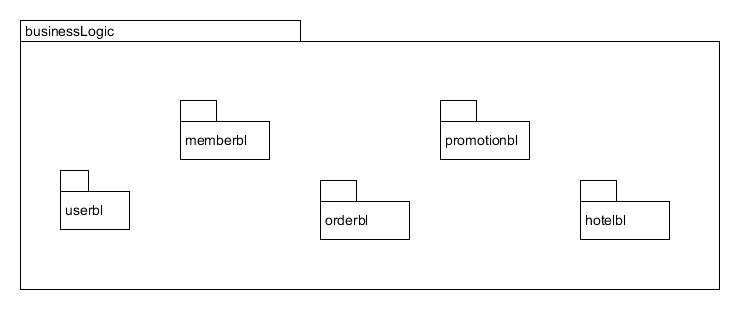


图10 业务逻辑层的设计

### 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| Userbl | 实现用户登录与个人账户信息修改的服务 |
| Memberbl | 实现与会员相关操作有关的服务 |
| Hotelbl | 实现对酒店信息增删改查相关的服务 |
| Orderbl | 实现与订单操作相关的服务 |
| Promotionbl | 实现促销策略的制定、实施相关的服务 |

### 业务逻辑模块的接口模范

#### userbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | public ResultMessage login(UserVO userVO) |
| 前置条件 | password符合输入规则且对应的用户未登录 |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回登录验证的结果 |
| User.logout | 语法 | public ResultMessage logout() |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更改用户登录状态 |
| User.add | 语法 | public ResultMessage add(UserVO userVO) |
| 前置条件 | 账户信息符合规则且齐全，且不与已注册的用户名重复 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| User.del | 语法 | public ResultMessage del(UserVO userVO) |
| 前置条件 | 无（删除表明该数据必然时存在的） |
| 后置条件 | 系统永久性 |
| User.find | 语法 | public UserVO find(String username) |
| 前置条件 | 管理员已登录 |
| 后置条件 | 系统返回查找到的账户信息 |
| User.showListByType | 语法 | public ArrayList<UserVO> showListByType (UserType userType) |
| 前置条件 | 管理员已登录 |
| 后置条件 | 系统返回相应类型的账户列表 |
| User.getInfo | 语法 | public UserVO getInfo(String username) |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统返回目前账户信息 |
| User.setInfo | 语法 | public ResultMessage setInfo(UserVO userVO) |
| 前置条件 | 用户已登录，账户信息符合规则且齐全 |
| 后置条件 | 系统修改该账户并持久化保存该账户数据 |
| User.showList | 语法 | public ArrayList<UserVO> showList() |
| 前置条件 | 管理员已登陆 |
| 后置条件 | 系统返回目前用户列表 |
| User.getMemberType | 语法 | public MemberType getMemberType(String username) |
| 前置条件 | 酒店工作人员/网站营销人员已登录 |
| 后置条件 | 系统返回客户会员类型 |
| User.addCredit | 语法 | public ResultMessage addCredit(String username, double amount) |
| 前置条件 | 无 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| User.deductCredit | 语法 | public ResultMessage deductCredit(String username, double amount) |
| 前置条件 | 无 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| UserDataService.canLogin(UserPO userPO) | 根据提供的VO判断是否允许登录 | |
| UserDataService.findByUsername(String username) | 根据用户名查找持久化对象 | |
| UserDataService.findByType(UserType userType) | 根据userType搜索并返回相应的用户列表 | |
| UserDataService.insert(UserPO userPO) | 根据提供的VO判断是否允许注册 | |
| UserDataService.delete(UserPO userPO) | 删除单一持久化对象 | |
| UserDataService.update(UserPO userPO) | 修改单一持久化对象 | |
| UserDataService.getList() | 显示用户列表 | |
| UserDataService.addCredit(String username, double amount) | 持久化增加客户信用值 | |
| UserDataService.deductCredit(String username, double amount) | 持久化扣除客户信用值 | |

#### memberbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Member.register | 语法 | public ResultMessage register(MemberVO memberVO) |
| 前置条件 | 无 |
| 后置条件 | 创建一个新的会员对象并持久化该对象 |
| Member.showInfo | 语法 | public MemberVO showInfo(String username) |
| 前置条件 | 无 |
| 后置条件 | 根据username搜索并返回对应的MemberVO |
| Member.delete | 语法 | public ResultMessage delete(String username) |
| 前置条件 | 无 |
| 后置条件 | 根据username搜索并删除对应的持久化对象 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| MemberDataService.find (String username) | 根据username进行查找单一持久化对象 | |
| MemberDataService.insert(MemberPO memberPO) | 插入单一持久化对象 | |
| MemberDataService.delete(MemberPO memberPO) | 删除单一持久化对象 | |
| MemberDataService.update(MemberPO memberPO) | 更新单一持久化对象 | |

#### hotelbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.find | 语法 | public HotelVO find(long hotelID) |
| 前置条件 | 已打开某酒店列表 |
| 后置条件 | 返回hotelID对应的酒店的VO |
| Hotel.showList | 语法 | public ArrayList<HotelVO> showList(HotelVO hotelVO) |
| 前置条件 | 无 |
| 后置条件 | 返回符合提供的HotelVO包含的信息的酒店列表 |
| Hotel.add | 语法 | public ResultMessage add(HotelVO hotelVO) |
| 前置条件 | 管理员已登录 |
| 后置条件 | 创建一个新的HotelPO对象并持久化该对象 |
| Hotel.del | 语法 | public ResultMessage del(long hotelID) |
| 前置条件 | 管理员已登录 |
| 后置条件 | 根据hotelID搜索并删除对应的持久化对象 |
| Hotel.modify | 语法 | public ResultMessage modify(HotelVO) |
| 前置条件 | 启动一个维护酒店基本信息事件 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| HotelDataService.findByHotelID(long hotelID) | 根据ID进行查找单一持久化对象 | |
| HotelDataService.findByHotelPO(HotelPO hotelPO) | 根据提供的VO进行查找多个持久化对象 | |
| HotelDataService.insert(HotelPO hotelPO) | 插入单一持久化对象 | |
| HotelDataService.delete(HotelPO hotelPO) | 删除单一持久化对象 | |
| HotelDataService.update(HotelPO hotelPO) | 更新单一持久化对象 | |

#### orderbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.find | 语法 | public OrderVO find(long orderID) |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统返回订单基本信息 |
| Order.add | 语法 | public ResultMessage add(OrderVO orderVO) |
| 前置条件 | 用户已登录，订单信息符合规则且齐全 |
| 后置条件 | 系统增加该订单并持久化保存该订单数据 |
| Order.modify | 语法 | public ResultMessage modify(OrderVO orderVO) |
| 前置条件 | 用户已登录，订单信息符合规则且齐全 |
| 后置条件 | 系统修改该订单并持久化保存该订单数据 |
| Order.evaluate | 语法 | public ResultMessage evaluate(OrderVO orderVO) |
| 前置条件 | 用户已登录，订单状态为已执行 |
| 后置条件 | 系统修改该订单评价信息并持久化保存该订单数据 |
| Order.cancel | 语法 | public ResultMessage cancel() |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统修改该订单状态信息并持久化保存该订单数据 |
| Order.showList | 语法 | public ArrayList<OrderVO> showList(OrderVO orderVO) |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统返回相应条件下的订单列表 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| OrderDataService.findByOrderID(long orderID) | 显示订单信息 | |
| OrderDataService.find(OrderPO orderPO) | 显示相应条件下的订单列表 | |
| OrderDataService.insert(OrderPO orderPO) | 增加单一持久化对象 | |
| OrderDataService.update(OrderPO orderPO) | 修改单一持久化对象 | |
| OrderDataService.getList() | 显示订单列表 | |

#### promotionbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Promotion.getHotelPromList | 语法 | public ArrayList<PromotionVO> getHotelPromList(long hotelID) |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 系统返回酒店促销策略列表 |
| Promotion.getWebPromList | 语法 | public ArrayList<PromotionVO> getWebPromList(PromotionType type) |
| 前置条件 | 网站营销人员已登录 |
| 后置条件 | 系统返回网站营销策略列表 |
| Promotion.find | 语法 | public PromotionVO find(long promotionID) |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 系统搜索并返回相应的促销策略vo |
| Promotion.add | 语法 | public ResultMessage add(PromotionVO promotionVO) |
| 前置条件 | 酒店工作人员/网站营销人员已登录，营销策略信息符合规则且齐全 |
| 后置条件 | 系统创建该营销策略并持久化增加该营销策略数据 |
| Promotion.del | 语法 | public ResultMessage del(PromotionVO promotionVO) |
| 前置条件 | 酒店工作人员/网站营销人员已登录 |
| 后置条件 | 系统删除该营销策略并持久化增加该营销策略数据 |
| Promotion.modify | 语法 | public ResultMessage modify(PromotionVO promotionVO) |
| 前置条件 | 酒店工作人员/网站营销人员已登录，营销策略信息符合规则且齐全 |
| 后置条件 | 系统修改该营销策略并持久化增加该营销策略数据 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| PromotionDataService.findByHotelID(long hotelID) | 系统根据hotelID查找多个持久化对象 | |
| PromotionDataService.findByPromotionType(PromotionType type) | 系统根据营销策略类型查找多个营销策略持久化对象 | |
| PromotionDataService.findByPromotionID(long promotionID) | 系统根据promotionID查找单个持久化对象 | |
| PromotionDataService.insert(PromotionPO po) | 系统更新单一持久化对象 | |
| PromotionDataService.delete(PromotionPO po) | 系统删除单一持久化对象 | |
| PromotionDataService.update(PromotionPO po) | 系统更新单一持久化对象 | |

## 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。例如，User业务逻辑需要的服务主要由UserDataService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。以UserDataService为例，数据层模块的具体描述下图所示：

**图11 数据层模块的描述**

### 数据层模块的职责

数据层模块职责如下表所示

**数据层模块职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserDataservice | 负责保存用户的信息，进行增删改查的操作 |
| MemberDataService | 负责保存会员的信息，供读取、写入和删除 |
| HotelDataService | 负责保存酒店的信息，进行增删改查的操作 |
| OrderDataservice | 负责保存订单的信息，供读取、写入和修改 |
| PromotionDataservice | 负责保存促销策略的信息，进行增删改查的操作 |

### 数据层模块的接口规范

数据层模块的接口规范如下表

#### DataFactory

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DataFactory.getUserDataService | 语法 | public UserDataService getUserDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回UserDataService对象 |
| DataFactory.getMemberDataService | 语法 | public MemberDataService getMemberDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回MemberDataService对象 |
| DataFactory.getHotelDataService | 语法 | public HotelDataService getHotelDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回HotelDataService对象 |
| DataFactory.getOrderDataService | 语法 | public OrderDataService getOrderDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回OrderDataService对象 |
| DataFactory.getPromotionDataService | 语法 | public PromotionDataService getPromotionDataImpl() |
| 前置条件 | 无 |
| 后置条件 | 返回PromotionDataService对象 |

#### UserDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| UserDataService.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 |
| UserDataService.canLogin | 语法 | public boolean canLogin(UserPO userPO) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 返回对应username的用户的是否可登录属性值 |
| UserDataService.findByUsername | 语法 | public UserPO findByUsername(String username) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按username进行查找返回相应的UserPO结果 |
| UserDataService.findByType | 语法 | public ArrayList<UserPO> findByType(UserType userType) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回对应类型的UserPO列表 |
| UserDataService.insert | 语法 | public void insert(UserPO userPO) throws RemoteException; |
| 前置条件 | 同样的username的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| UserDataService.delete | 语法 | public void delete(UserPO userPO) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 删除一个po |
| UserDataService.update | 语法 | public void update(UserPO userPO) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 更新一个po |
| UserDataService.getList | 语法 | public ArrayList<UserPO> getList() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回所有UserPO的列表 |
| UserDataService.addCredit | 语法 | public void addCredit(String username, double amount) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 持久化增加对应username的客户的信用值 |
| UserDataService.deductCredit | 语法 | public void deductCredit(String username, double amount) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 持久化扣除对应username的客户的信用值 |
| UserDataService.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 |

#### MemberDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MemberDataService.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 |
| MemberDataService.find | 语法 | public MemberPO find (String username) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按username进行查找返回相应的MemberPO结果 |
| MemberDataService.insert | 语法 | public void insert(MemberPO memberPO) throws RemoteException; |
| 前置条件 | 同样的username的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| MemberDataService.delete | 语法 | public void delete(MemberPO memberPO) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 删除一个po |
| MemberDataService.update | 语法 | public void update(MemberPO memberPO) throws RemoteException; |
| 前置条件 | 在数据库中存在同样username的po |
| 后置条件 | 更新一个po |
| MemberDataService.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 |

#### HotelDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelDataService.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 |
| HotelDataService.findByHotelID | 语法 | public HotelPO findByHotelID(long hotelID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的HotelPO结果 |
| HotelDataService.findByHotelPO | 语法 | public ArrayList<HotelPO> findByHotelPO(HotelPO hotelPO) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按提供的HotelPO进行查找返回相应的HotelPO结果列表 |
| HotelDataService.insert | 语法 | public void insert(HotelPO hotelPO) throws RemoteException; |
| 前置条件 | 同样的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 |
| HotelDataService.delete | 语法 | public void delete(HotelPO hotelPO) 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 |
| HotelDataService.update | 语法 | public void update(HotelPO hotelPO) 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 |
| HotelDataService.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 |

#### OrderDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| OrderDataService.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 |
| OrderDataService.findByOrderID | 语法 | public OrderPO findByOrderID(long orderID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的OrderPO结果 |
| OrderDataService.find | 语法 | public ArrayList<OrderPO> findByType(OrderPO orderPO) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回相应条件下的ArrayList<OrderPO>结果 |
| OrderDataService.insert | 语法 | public void insert(OrderPO orderPO) 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 |
| OrderDataService.update | 语法 | public void update(OrderPO orderPO) 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 |
| OrderDataService.getList | 语法 | public ArrayList<OrderPO> getList() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回ArrayList<OrderPO>结果 |
| OrderDataService.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 |

#### PromotionDataService

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionDataService.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 |
| PromotionDataService.findByHotelID | 语法 | public ArrayList<PromotionPO> findByHotelID(long hotelID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按hotelID进行查找返回相应的ArrayList<PromotionPO>结果 |
| PromotionDataService.findByPromotionType | 语法 | public ArrayList<PromotionPO> findByPromotionType(PromotionType type) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按PromotionType进行查找返回相应的ArrayList<PromotionPO>结果 |
| PromotionDataService.findByPromotionID | 语法 | public PromotionPO findByPromotionID(long promotionID) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的PromotionPO结果 |
| PromotionDataService.insert | 语法 | public void insert(PromotionPO promotionPO) 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 |
| PromotionDataService.delete | 语法 | public void delete(PromotionPO promotionPO) 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 |
| PromotionDataService.update | 语法 | public void update(PromotionPO promotionPO) 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 |
| PromotionDataService.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 |

# 信息视角

## 数据持久化对象

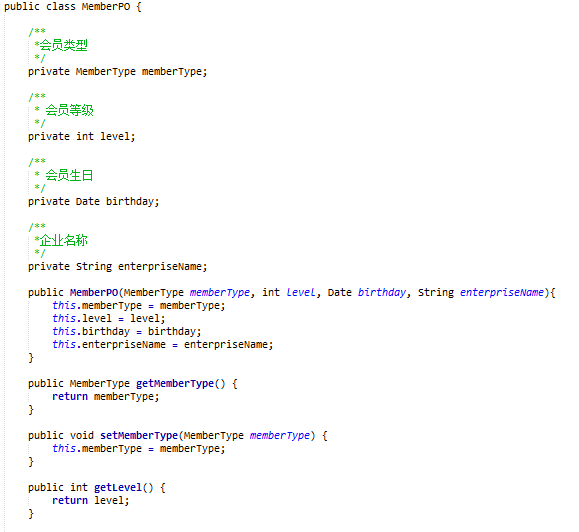
系统的PO类就是对应的相关的实体类，如下所示。本系统共包含7个PO类，在此

不一一赘述，只做部分介绍：

UserPO类为账户信息，包括用户名、密码、账户类型、个人其他信息。

MemberPO类为会员信息，包括会员类型，等级，生日等等。

HotelPO类为酒店信息，包括酒店ID，酒店地址商圈，酒店客房信息，历史评价等等。



## 文件格式

本系统依据不同情况分别采用数据库、序列化、txt文件持久化保存三种方式保存文件