互联网酒店预订系统

软件体系结构描述文档

V1.2

南京大学软件学院

小组成员：徐佳炜，徐杨晨，茅力文，袁颜颜

2016年10月16日

**目录**

1、引言 4

1.1 编制目的 4

1.2 词汇表 4

1.3 参考资料 4

1.4更新历史 4

2、产品概述 4

3、逻辑视角 4

4、组合视角 7

4.1 开发包图 7

4.2 运行时进程 9

4.3 物理部署 10

5、接口视角 12

5.1 模块的职责 12

5.2 用户界面层的分解 14

5.2.1 用户界面层模块的职责 15

5.2.2用户界面层模块的接口规范 15

用户界面层需要的服务接口如表5.2.2-2所示。 16

5.2.3 用户界面模块设计原理 16

5.3 业务逻辑层的分解 16

5.3.1 业务逻辑层模块的职责 17

5.3.2 业务逻辑层模块的接口规范 17

5.4 数据层的分解 25

5.4.1 数据层模块的职责 26

5.4.2 数据层模块的接口规范 26

6、信息视角 31

6.1 数据持久化对象 31

6.2 TXT持久化对象 44

6.3 数据库表 44

**1.引言**

1.1编制目的

本报告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和用户的沟通。

本报告面向开发人员，测试人员以及最终用户而编写，是了解系统的导航。

1.2词汇表

1.3参考资料

连锁商店管理系统（MSCS）软件体系结构描述文档

1.4更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 更新人员 | 更新时间 | 更新概述 | 版本号 |
| 徐佳炜 | 2016/10/9 | 第一次编写 | V1.0 |
| 徐杨晨 | 2016/10/12 | 添加：业务逻辑层的设计 | V1.1 |
| 徐杨晨 | 2016/10/13 | 添加：业务逻辑层模块的职责、接口规范 | V1.2 |

**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中对产品的概括描述。

**3.逻辑视角**

在互联网酒店预订系统中，选择了分层风格，将系统分为三层（展示层，业务逻辑层，数据层）能够很好的展示整个高层抽象。展示层包含JAVAFX（或者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。

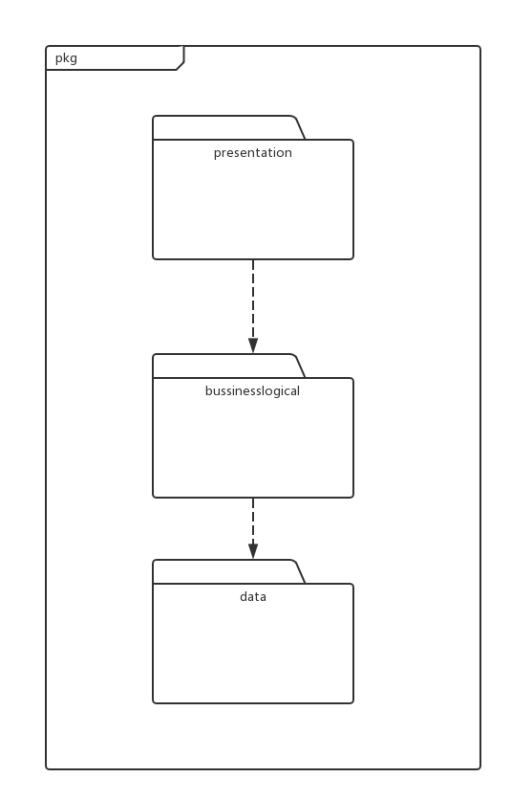


图1 参照体系结构风格的包图表达逻辑视角

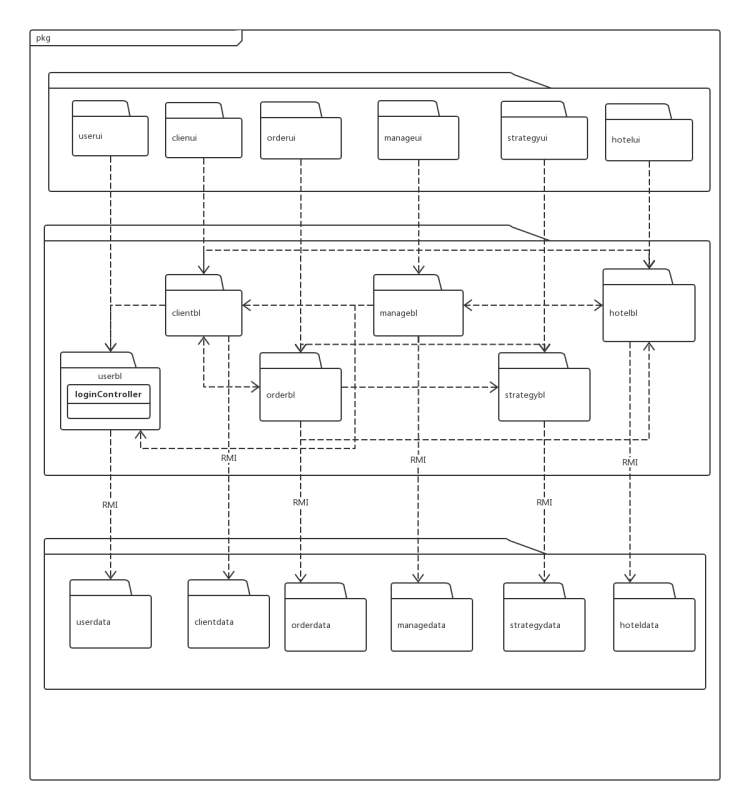


图2 软件体系结构逻辑设计方案

**4.组合视角**

4.1开发包图

互联网酒店预订系统的最终开发包设计如表1所示。

**表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,strategyui,hotelui,clientui,manageui,userui,vo |
| orderui | orderblservice,界面类库包,vo |
| orderblservice |  |
| orderbl | orderblservice,orderdataservice,po,strategybl,hotelbl |
| orderdataservice | Java RMI,po |
| orderdata | Java RMI,po,databaseutility |
| strategyui | strategyblservice,界面类库包 |
| strategyblservice |  |
| strategybl | strategyblservice,strategydataservice,vo |
| strategydataservice | Java RMI,po |
| strategydata | Java RMI,po,databaseutility |
| hotelui | hotelblservice,界面类库包 |
| hotelblservice |  |
| hotelbl | hotelblservice,hoteldataservice,po |
| hoteldataservice | Java RMI,po |
| hoteldata | Java RMI,po,databaseutility |
| clientui | clientblservice,界面类库包 |
| clientblservice |  |
| clientbl | clientblservice,clientdataservice,po,orderbl,hotelbl,userbl |
| clientdataservice | Java RMI,po |
| clientdata | Java RMI,po,databaseutility |
| manageui | manageblservice,界面类库包 |
| manageblservice |  |
| managebl | manageblservice,managedataservice,po,orderbl,  strategybl,hotelbl,clientbl,userbl |
| managedataservice | Java RMI,po |
| managedata | Java RMI,po,databaseutility |
| userui | userblservice,界面类库包 |
| userblservice |  |
| userbl | userblservice,userdataservice,po |
| userdataservice | Java RMI,po |
| userdata | Java RMI,po,databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

互联网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。

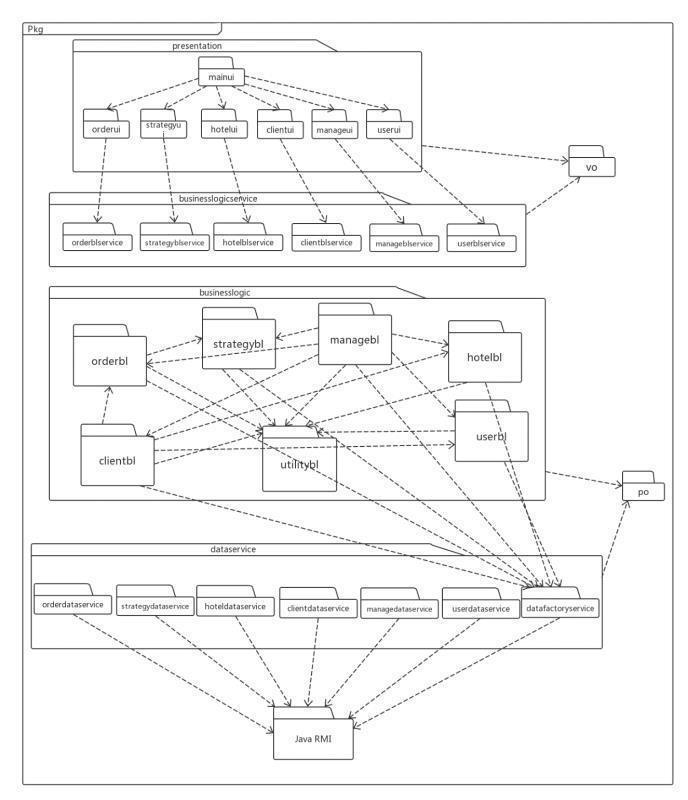


图3 客户端开发包图

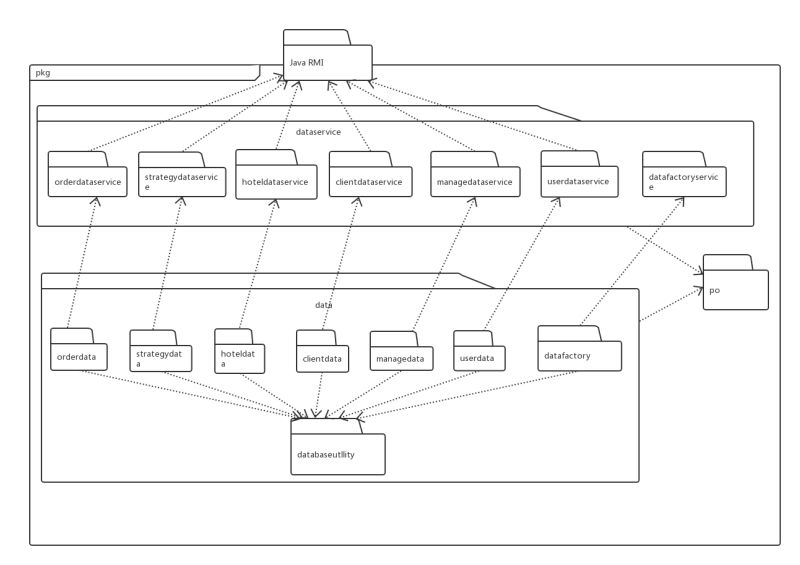


图4 服务器端开发包图

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。

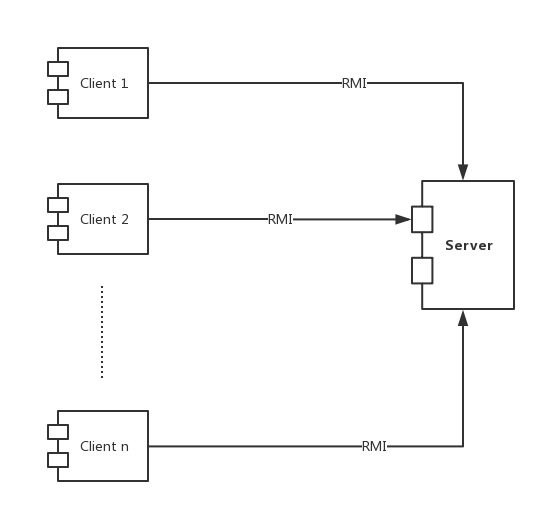


图5 进程图

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JDK8.0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6所示。

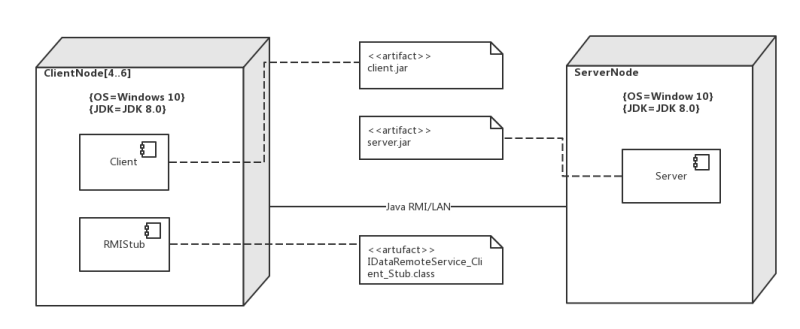


图6 部署图

**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。

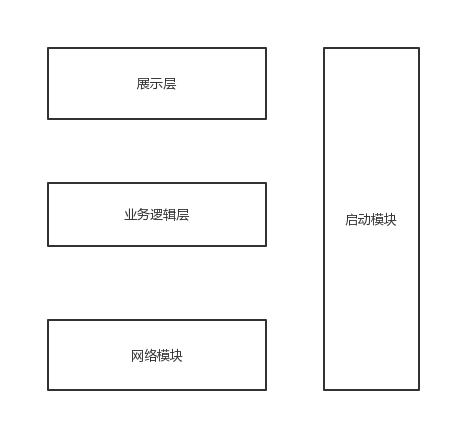


图7 客户端模块视图

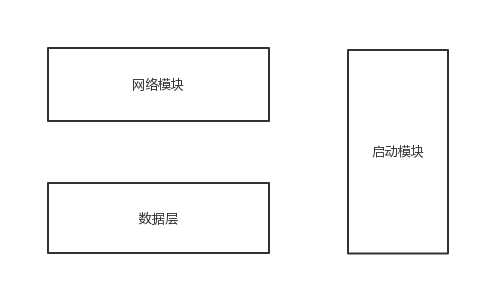


图8 服务器端模块视图

**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 | 基于窗口的互联网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借口 |
| 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** |
|  | 客户端展示层 | 客户端业务逻辑层 |
|  | 客户端业务逻辑层 | 服务器端数据层 |

5.2用户界面层的分解

根据需求，系统存在22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、客户基本信息界面、客户订单界面、酒店列表界面、酒店信息界面、生成订单界面、酒店评价界面、注册会员界面、酒店工作人员的酒店信息界面、制定酒店促销策略界面、更新客房信息界面、酒店订单界面、制定网站促销策略界面、浏览订单界面、信用充值界面、用户列表界面、用户信息界面、调整用户（添加人员）界面。界面跳转如下图所示。

图10 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。用户界面类如下图所示。

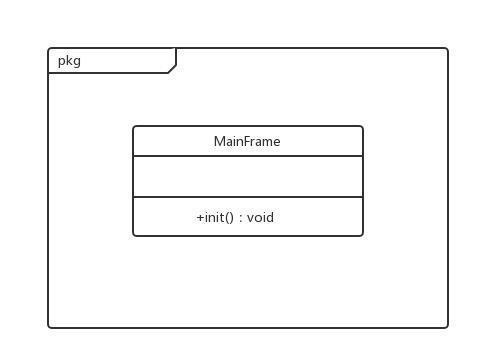
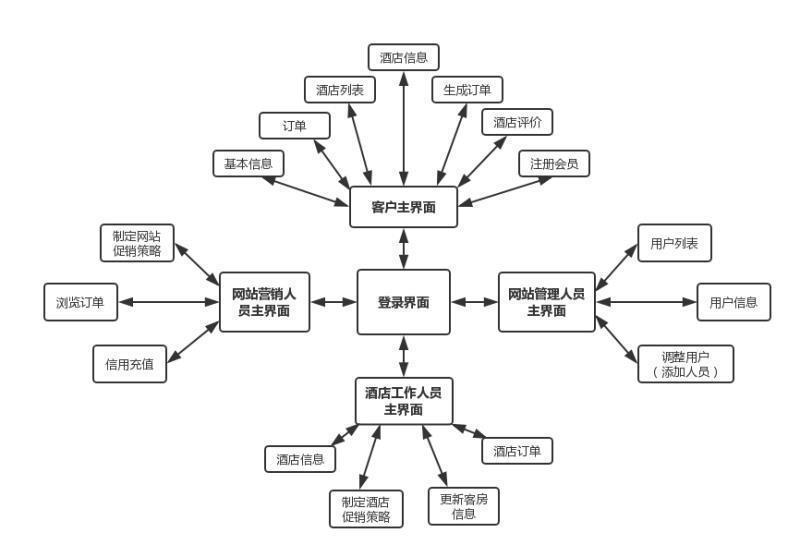


图11 用户界面类

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。

**表5 用户界面模块的职责**

|  |  |
| --- | --- |
| 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

**表6 用户界面模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | init(args：String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel |
| order\_ui | 语法 | init(args：String[]) |
| 前置条件 | 有用户选择查看订单或生成订单操作 |
| 后置条件 | 显示orderPanel |
| strategy\_ui | 语法 | init(args：String[]) |
| 前置条件 | 网站营销人员或酒店工作人员选择生成促销策略 |
| 后置条件 | 显示strategyPanel |
| hotel\_ui | 语法 | init(args：String[]) |
| 前置条件 | 有用户选择查看酒店 |
| 后置条件 | 显示hotelPanel |
| client\_ui | 语法 | init(args：String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示clientPanel |
| manage\_ui | 语法 | init(args：String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示managePanel |
| user\_ui | 语法 | init(args：String[]) |
| 前置条件 | 网站管理人员选择查询更改或添加用户 |
| 后置条件 | 显示userPanel |

用户界面层需要的服务接口如下表所示

**表7 用户界面模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| businesslogicservice.LoginBLService | 登录界面的业务逻辑接口 |
| Businesslogicservice.  order\_blservice | 处理订单信息的接口 |
| Businesslogicservice.  strategy\_blservice | 处理销售策略信息的接口 |
| Businesslogicservice.  hotel\_blservice | 处理酒店信息的接口 |
| Businesslogicservice.  client\_blservice | 处理客户信息的接口 |
| Businesslogicservice.  manage\_blservice | 处理网站管理的接口 |
| Businesslogicservice.  user\_blservice | 负责用户调整的接口 |

5.2.3用户界面模块设计原理

用户界面利用Java的fx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，Order对象负责处理订单浏览、生成、撤销界面的业务逻辑；Hotel对象负责处理酒店基本信息、录入可用房间、更新房间信息界面的业务逻辑。业务逻辑层的设计如图12所示。

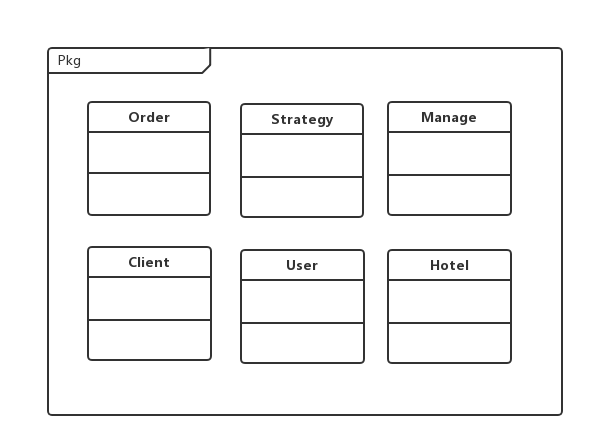


图12 业务逻辑层的设计

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8所示。

**表8 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| orderbl | 负责实现订单相关界面所需要的服务 |
| strategybl | 负责实现促销策略界面所需要的服务 |
| managebl | 负责实现网站管理相关界面所需要的服务 |
| clientbl | 负责实现客户相关界面所需要的服务 |
| userbl | 负责实现调整用户界面所需要的服务 |
| hotelbl | 负责实现酒店相关界面所需要的服务 |

5.3.2 业务逻辑层模块的接口规范

orderbl,strategybl,managebl,clientbl,userbl,hotelbl模块的接口规范如表9,10,11,12,13和14所示。

**表9 orderbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.client\_browse | 语法 | public ArrayList<OrderVO> order\_client\_browse(); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 按照时间由近到远的顺序，以列表形式返回当前客户的所有订单，同时标记订单状态 |
| Order.hotel\_browse | 语法 | public ArrayList<OrderVO> order\_hotel\_browse(); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 按照时间由近到远的顺序，以列表形式返回当前酒店的所有订单，同时标记订单状态 |
| Order.client\_cancel | 语法 | public ResultMessage order\_client\_cancel(OrderVO vo); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 将选择的订单状态置为已撤销，同时系统计算撤销订单时间与最晚订单执行时间的时间差，若不足6个小时，扣除客户信用值(信用值为订单总价值的1/2)。持久化更新涉及的领域对象的数据 |
| Order.client\_generate | 语法 | public OrderVO order\_client\_generate(); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统读取用户输入的订单信息，自动为其选择最合适的优惠策略，生成订单。持久化更新涉及的领域对象的数据 |
| Order.hotel\_execute | 语法 | public ResultMessage order\_hotel\_execute(ClientVO vo); |
| 前置条件 | 酒店工作人员已被识别且授权；客户到达酒店 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.  getOrderDatabase | 得到Order数据库的服务的引用 | |
| OrderDataService.  find(int clientid) | 查找对应客户的所有持久化对象 | |
| OrderDataService.  find(String hotelname) | 查找对应酒店的所有持久化对象 | |
| OrderDataService.  insert(OrderPO po) | 在数据库中插入OrderPO对象 | |
| OrderDataService.  update(OrderPO po) | 更新持久化对象 | |
| DatabaseFactory.  getClientDatabase | 得到Client数据库的服务的引用 | |
| ClientDatabase.  updateCredit(int clientid) | 更新对应客户的持久化对象 | |
| DatabaseFactory.  getStrategyDatabase | 得到Strategy数据库的服务的引用 | |
| StrategyDatabase.  getStrategyList\_hotel(String hotelname) | 查找对应酒店的所有持久化对象 | |
| StrategyDatabase.  getStrategyList\_web() | 查找网站的所有持久化对象 | |

**表10 strategybl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Strategy.hotel\_make | 语法 | public HotelStrategyVO strategy\_hotel\_make(); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 系统持久化更新涉及的领域对象的数据 |
| Strategy.web\_make | 语法 | public WebStrategyVO strategy\_web\_make(); |
| 前置条件 | 网站营销人员已被识别且授权 |
| 后置条件 | 系统持久化更新涉及的领域对象的数据 |
| Strategy.hotel\_update | 语法 | public ResultMessage strategy\_hotel\_update(HotelStrategyVO vo); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 系统持久化更新涉及的领域对象的数据 |
| Strategy.web\_update | 语法 | public ResultMessage strategy\_web\_update(WebStrategyVO vo); |
| 前置条件 | 网站营销人员已被识别且授权 |
| 后置条件 | 系统持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.  getStrategyDatabase | 得到Strategy数据库的服务的引用 | |
| StrategyDataService.  insert(StrategyPO po) | 在数据库中插入StrategyPO对象 | |
| StrategyDatabase.  getStrategyList\_hotel(String hotelname) | 查找对应酒店的所有持久化对象 | |
| StrategyDatabase.  getStrategyList\_web() | 查找网站的所有持久化对象 | |
| StrategyDataService.  update\_hotel(String hotelname,HotelStrategyPO po) | 更新对应酒店的持久化对象 | |
| StrategyDataService.  update\_web(WebStrategyPO po) | 更新网站的持久化对象 | |
| StrategyDataService.  delete\_hotel(String hotelname,HotelStrategyPO po) | 删除对应酒店的持久化对象 | |
| StrategyDataService.  delete\_web(WebStrategyPO po) | 删除网站的持久化对象 | |

**表11 managebl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Manage.searchClient | 语法 | public ClientVO mangage\_searchClient(int clientid); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统返回对应客户的信息 |
| Manage.updateClient | 语法 | public ResultMessage manage\_updateClient(ClientVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统更新对应客户信息 |
| Manage.addHotel | 语法 | public ResultMessage manage\_addHotel(HotelVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统添加对应酒店 |
| Manage.addHotelWorker | 语法 | public ResultMessage manage\_addHotelWorker(String hotelname，  HotelWorkerVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统添加对应酒店的工作人员 |
| Manage.searchHotel  Worker | 语法 | public HotelWorkerVO manage\_searchHotelWorker(String hotelname); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统返回对应酒店的工作人员信息 |
| Manage.updateHotel  Worker | 语法 | public ResultMessage manage\_updateHotelWorker(String hotelname,HotelWorkerVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统更新对应酒店的工作人员信息 |
| Manage.addWebMarket | 语法 | public ResultMessage manage\_addWebMarket(WebMarketVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统添加网站营销人员 |
| Manage.searchWebMarket | 语法 | public WebMarketVO manage\_web\_searchWebMarket(int marketWorkerid); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统更新对应的网站营销人员信息 |
| Manage.updateWebMarket | 语法 | public ResultMessage manage\_web\_updateWebMarket(WebMarketVO vo); |
| 前置条件 | 网站管理人员已被识别且授权 |
| 后置条件 | 系统更新对应的网站营销人员信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.  getClientDatabase | 得到Cilent数据库的服务的引用 | |
| DatabaseFactory.  getHotelDatabase | 得到Hotel数据库的服务的引用 | |
| DatabaseFactory.  getManageDatabase | 得到Manage数据库的服务的引用 | |
| ClientDataService.  find(int clientid) | 查找对应客户的信息 | |
| ClientDataService.  update(ClientPO po) | 更新对应客户的信息 | |
| HotelDataService.  findHotelWorker(String hotelname) | 查找对应酒店的工作人员信息 | |
| HotelDataService.  updateHotelWorker(String hotelname,HotelWorkerPO po) | 更新对应酒店的工作人员信息 | |
| HotelDataService.  insertHotelWorker(String hotelname,HotelWorkerPO po) | 在数据库中插入HotelWorkerPO对象 | |
| HotelDataService.  insertHotel(HotelPO po) | 在数据库中插入HotelPO对象 | |
| ManageDataService.  find(int webmarkerid) | 查找网站营销人员信息 | |
| ManageDataService.  update(WebMarketPO po) | 更新对应的网站营销人员信息 | |
| ManageDataService.  insert(WebMarketPO po) | 在数据库中插入MarketWorkerPO对象 | |
| ManageDataService.  delete(WebMarketPO po) | 删除网站营销人员信息 | |

**表12 clientbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Client.checkInfo | 语法 | public ClientVO client\_checkInfo(); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统返回当前客户的信息（姓名或名称，联系方式、信用） |
| Client.updateInfo | 语法 | public ResultMessage client\_updateInfo(ClientVO vo) |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统更新当前客户的信息（姓名或名称，联系方式、信用） |
| Client.getPreviousHotelList | 语法 | public ArrayList<HotelVO> client\_getpreviousHotelList (); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统按照时间顺序由近到远，以列表形式返回客户预定过的酒店，同时每个酒店标记上客户是否曾经预定过（若有，显示对应的订单信息） |
| Client.getCreditList | 语法 | public String client\_checkCredit(); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统返回当前客户的信用记录(时间，订单号，动作（订单执行、订单异常、订单撤销、充值），信用度变化、信用度结果) |
| Client.setLocation | 语法 | public ArrayList<HotelVO> client\_setLocation(String location); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统将客户地址/商圈更新为客户选择的地址/商圈 |
| Client.searchHotel | 语法 | public ArrayList<HotelVO> client\_searchHotel(Condition cond); |
| 前置条件 | 客户已设置地址/商圈 |
| 后置条件 | 系统以列表形式返回满足对应搜索条件(通过酒店名称、房间（类型、原始价格区间、有空房期间（房间数量、入住日期，退房日期））、星级、评分区间)的所有酒店的详细信息。同时每个酒店标记上客户是否曾经预定过（若有，显示对应的订单信息）。同时提供列表排序选项（价格从低到高，星级，评分） |
| Client.checkHotelInfo | 语法 | public HotelVO client\_checkHotelInfo(); |
| 前置条件 | 客户在已显示的HotelList中选择某个酒店查看其详细信息 |
| 后置条件 | 系统返回对应酒店的详细信息。同时列出当前客户在该酒店所有的订单（正常订单、异常订单和撤销订单要分别标记） |
| Client.evaluateHotel | 语法 | public ResultMessage client\_evaluateHotel(Evaluation e); |
| 前置条件 | 客户选择的酒店对应订单状态为已执行 |
| 后置条件 | 系统更新对应酒店的评价信息 |
| Client.enrollVIP | 语法 | public ResultMessage client\_enrollVIP(String vipinfo); |
| 前置条件 | 客户已被识别且授权 |
| 后置条件 | 系统更新当前客户的会员信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.  getClientDatabase | 得到Client数据库的服务的引用 | |
| ClientDataService.  find(int clientid) | 根据ID查找对应客户的持久化对象 | |
| ClientDataService.  update(ClientPO po) | 更新对应客户的持久化对象 | |
| DatabaseFactory.  getHotelDatabase | 得到Hotel数据库的服务的引用 | |
| HotelDataService.  find(String hotelname) | 根据酒店名查找对应酒店的持久化对象 | |
| HotelDataService.  update(HotelPO po) | 更新对应酒店的持久化对象 | |
| DatabaseFactory.  getOrderDatabase | 得到Order数据库的服务的引用 | |
| OrderDataService.  find(int clientid) | 根据ID查找对应客户的持久化对象 | |

**表13 userbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.login | 语法 | | public ResultMessage login(long id,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user，根据输入的password返回登录验证的结果 |
| 需要的服务(需接口） | | | |
| DatabaseFactory.getUser | | 得到User数据库的服务的引用 | |

**表14 hotelbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.checkInfo | 语法 | public HotelVO hotel\_checkInfo(HotelWorkerVO vo); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 系统返回当前酒店的信息（地址、所属商圈、简介、设施服务、星级） |
| Hotel.updateInfo | 语法 | public ResultMessage hotel\_updateInfo(HotelVO vo); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 系统更新当前酒店的信息（地址、所属商圈、简介、设施服务、星级） |
| Hotel.importRoom | 语法 | public ResultMessage hotel\_importRoom(ArrayList<Room> a); |
| 前置条件 | 酒店工作人员已被识别且授权 |
| 后置条件 | 系统更新当前的酒店的可用房间信息（类型、数量、原始价格） |
| Hotel.updateAccommodation | 语法 | public ResultMessage hotel\_updateAccomodation(AccomdationInfo ai); |
| 前置条件 | 酒店工作人员已被识别且授权；有订单执行情况；有客户退房；线下入住/退房 |
| 后置条件 | 系统更新入住信息（房间号，入住时间、预计离开时间）/退房信息（实际离开时间） |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DatabaseFactory.  getHotelDatabase | 得到Hotel数据库的服务的引用 | |
| HotelDataService.  find(String hotelname) | 根据酒店名查找对应酒店的持久化对象 | |
| HotelDataService.  update(HotelPO po) | 更新对应酒店的持久化对象 | |

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。Order业务逻辑需要的服务有OrderDataService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图13所示。

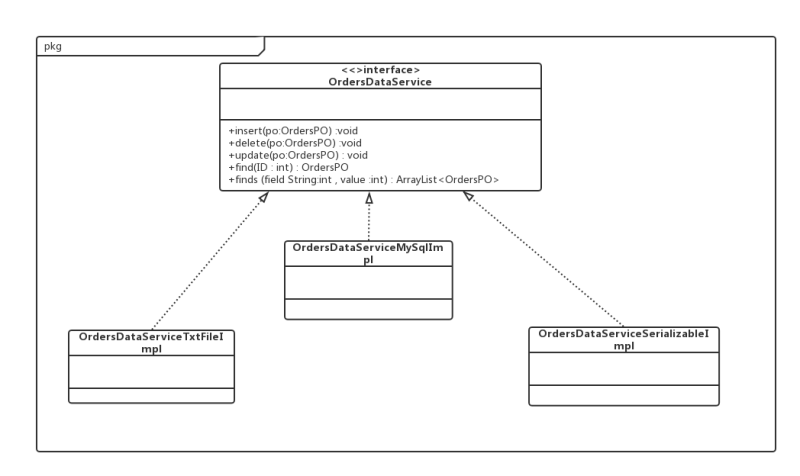


图13 数据层模块的描述

5.4.1 数据层模块的职责

数据层模块的职责如表2所示

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderDataService | 持久化数据的接口，提供集体载入、集体保存、增、删、改、查服务 |
| OrderDataServiceTxtFileImpl | 基于Txt文件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 |
| OrderDataServiceSerializableImpl | 基于序列化文件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 |
| OrderDataServiceMySqlImpl | 基于MySql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 |

表2

5.4.2数据层模块的接口规范

表15数据层orderdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDataService.  find(int client\_id) | 语法 | public OrderPO find (int id) ; |
| 前置条件 | 数据库中存在相同ID的po |
| 后置条件 | 按ID进行查找返回相应的OrderPO结果 |
| OrderDataService.  find(String hotel\_name) | 语法 | public List<OrderPO> find (String name) ; |
| 前置条件 | 数据库中存在相同name的po |
| 后置条件 | 按name进行查找返回所有该酒店的OrderPO结果 |
| OrderDataService.  insert(OrderPO po) | 语法 | public ResultMessage add(OrderPO po); |
| 前置条件 | 同样ID的po不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| OrderDataService.  update(OrderPO po) | 语法 | public ResultMessage update(OrderPO po); |
| 前置条件 | 在数据库中存在相同ID的po |
| 后置条件 | 更新一个po |
| OrderDataService.  delete(OrderPO po) | 语法 | public ResultMessage delete(OrderPO po); |
| 前置条件 | 在数据库中不存在相同ID的po |
| 后置条件 | 删除一个po |

表16数据层strategydata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyDataService.find\_hotel | 语法 | public StrategyHotelPO find (String name) ; |
| 前置条件 | 数据库中存在相同name的po |
| 后置条件 | 按name进行查找返回相应的StrategyHotelPO结果 |
| StrategyDataService.insert\_hotel | 语法 | public ResultMessage insert(StrategyHotelPO po); |
| 前置条件 | 同样name的po不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| StrategyDataService.delete\_hotel | 语法 | public ResultMessage delete(StrategyHotelPO po); |
| 前置条件 | 在数据库中存在name相同的po |
| 后置条件 | 删除一个po |
| StrategyDataService.update\_hotel | 语法 | public ResultMessage update(StrategyHotelPO po); |
| 前置条件 | 在数据库中存在相同name的po |
| 后置条件 | 更新一个po |
| StrategyDataService.getStrategyList\_hotel | 语法 | public ArrayList<StrategyHotelPO> show() ; |
| 前置条件 | 逻辑层请求获得酒店策略列表 |
| 后置条件 | 无 |
| StrategyDataService.find\_web | 语法 | public StrategyWebPO find (String name) ; |
| 前置条件 | 数据库中存在相同name的po |
| 后置条件 | 按name进行查找返回相应的StrategyWebPO结果 |
| StrategyDataService.insert\_web | 语法 | public ResultMessage insert(StrategyWebPO po); |
| 前置条件 | 同样name的po不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| StrategyDataService.delete\_web | 语法 | public ResultMessage delete(StrategyWebPO po); |
| 前置条件 | 在数据库中存在name相同的po |
| 后置条件 | 删除一个po |
| StrategyDataService.update\_web | 语法 | public ResultMessage update(StrategyWebPO po); |
| 前置条件 | 在数据库中存在相同name的po |
| 后置条件 | 更新一个po |
| StrategyDataService.getStrategyList\_web | 语法 | Public List<StrategyWebPO> show() ; |
| 前置条件 | 逻辑层请求获得网站策略列表 |
| 后置条件 | 无 |

表17数据层hoteldata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelDataService.find | 语法 | public HotelPO find (String name) ; |
| 前置条件 | 数据库中存在name相同的po |
| 后置条件 | 按name进行查找返回相应的HotelPO结果 |
| HotelDataService.insert | 语法 | public ResultMessage insert(HotelPO po); |
| 前置条件 | 同样ID的po不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| HotelDataService.delete | 语法 | public ResultMessage delete(OrderPO po); |
| 前置条件 | 在数据库中存在ID相同的po |
| 后置条件 | 删除一个po |
| HotelDataService.update | 语法 | public ResultMessage update(OrderPO po); |
| 前置条件 | 在数据库中存在相同ID的po |
| 后置条件 | 更新一个po |
| HotelDataService.findHotelWorkers | 语法 | Public List<HotelWorkers> findHotelWorkers(String hotelname); |
| 前置条件 | 在数据库中存在name相同的po |
| 后置条件 | 返回该酒店的工作人员列表 |

表18数据层clientdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ClientDataService.find | 语法 | public ClientPO find (int id) ; |
| 前置条件 | 数据库中存在name相同的po |
| 后置条件 | 按name进行查找返回对应的Clientpo结果 |
| ClientDataService.insert | 语法 | public ResultMessage insert(ClientPO po); |
| 前置条件 | 在数据库中不存在相同的po |
| 后置条件 | 在数据库中添加一条po |
| ClientDataService.update | 语法 | public ResultMessage update(ClientPO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 更新一个po |
| ClientDataService.delete | 语法 | public ResultMessage delete(ClientPO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 删除一个po |
| ClientDataService.showHotel | 语法 | public List<Hotel> showHotel(ClientPO po); |
| 前置条件 | 在数据库中存在相同的po |
| 后置条件 | 无 |
| ClientDataService.updateCredit | 语法 | public ResultMessage changeCredit(ClientPO po); |
| 前置条件 | 逻辑层请求更改用户信用值 |
| 后置条件 | 更改用户信用值 |

表19数据层managedata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService.find | 语法 | public ManagePO find (String name) ; |
| 前置条件 | 数据库中存在name相同的po |
| 后置条件 | 按name进行查找返回对应的Managepo结果 |
| ManageDataService.insert | 语法 | public ResultMessage insert(ManagePO po); |
| 前置条件 | 在数据库中不存在相同的po |
| 后置条件 | 在数据库中添加一条po |
| ManageDataService.update | 语法 | public ResultMessage update(ManagePO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 更新一个po |
| ManageDataService.delete | 语法 | public ResultMessage delete(ManagePO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 删除一个po |
| ManageDataService.showWebWorker | 语法 | public List<WebWorker> showWebWorker(ManagePO po); |
| 前置条件 | 在数据库中存在相同的po |
| 后置条件 | 返回所有的网站工作人员 |

表20数据层userdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.find | 语法 | public UserPO find (String name) ; |
| 前置条件 | 数据库中存在name相同的po |
| 后置条件 | 按name进行查找返回对应的Managepo结果 |
| UserDataService.insert | 语法 | public ResultMessage insert(UserPO po); |
| 前置条件 | 在数据库中不存在相同的po |
| 后置条件 | 在数据库中添加一条po |
| UserDataService.update | 语法 | public ResultMessage update(UserPO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 更新一个po |
| UserDataService.delete | 语法 | public ResultMessage delete(UserPO po); |
| 前置条件 | 在数据库里存在相同的po |
| 后置条件 | 删除一个po |
| UserDataService.showWebManager | 语法 | public List<WebManager> showWebManager(UserPO po); |
| 前置条件 | 在数据库中存在相同的po |
| 后置条件 | 返回所有的网站工作人员 |
| 后置条件 |  |

表21数据层datafactory模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DataBaseFactory.getClientDatabase | 语法 | public ClientPO getClientDatabase (); |
| 前置条件 | 逻辑层请求调用ClientData |
| 后置条件 | 按name进行查找返回对应的Managepo结果 |
| DataBaseFactory.getHotelDatabase | 语法 | public ClientPO getHotelDatabase (); |
| 前置条件 | 逻辑层请求调用HotelData |
| 后置条件 | 在数据库中添加一条po |
| DataBaseFactory.getManageDatabase | 语法 | public ClientPO getManageDatabase (); |
| 前置条件 | 逻辑层请求调用ManageData |
| 后置条件 | 更新一个po |
| DataBaseFactory.getUserDatabase | 语法 | public ClientPO getUserDatabase (); |
| 前置条件 | 逻辑层请求调用UserData |
| 后置条件 | 删除一个po |
| DataBaseFactory.getOrderDatabase | 语法 | public ClientPO getOrderDatabase (); |
| 前置条件 | 逻辑层请求调用OrderData |
| 后置条件 | 返回所有的网站工作人员 |
| DataBaseFactory.getStrategyDatabase | 语法 | public ClientPO getStrategyDatabase (); |
| 前置条件 | 逻辑层请求调用StrategyData |
| 后置条件 | 返回所有的网站工作人员 |

**6.信息视角**

6.1数据持久化对象

· ClinetPO类包含客户的姓名，联系方式，信用值，信用记录，会员ID,会员种类，生日，所在企业

* OrderPO类包含订单的编号，状态，执行情况，酒店，开始时间，退房时间，最晚订单执行时间，房间类型及数量，优惠策略，价格，预计入住人数，有无儿童

会员种类，生日，所在企业

* HotelStrategyPO类包含促销策略的名称，适用条件，开始时间，结束时间，实行方式，是否可叠加
* WebStrategyPO类包含网站促销策略的名称，适用条件，开始时间，结束时间，实行方式，是否可叠加
* HotelPO类包含酒店的地址，商圈，简介，设施服务，客房（状态，类型，数量，原始价格），订单，评价
* HotelWorkerPO类包含酒店工作人员的姓名，联系方式，所属酒店
* WebManagerPO类包含网站管理人员的姓名，联系方式
* WebMarketPO类包含网站营销人员的姓名，联系方式
* UserPO类包含客户的账号，密码，酒店工作人员的账号，密码，网站管理人员的账号，密码，网站营销人员的账号，密码

|  |  |
| --- | --- |
| 数据持久化对象 | 定义 |
| UserPO | public class UserPO {  int id;  String password;  public UserPO(int i,String p){  id=i;  password=p;  }  public int getID(){  return id;  }  public String getPassword(){  return password;  }  } |
| ClientPO | public class ClientPO implements Serializable{  String name;  String contact;  int credit;  String credit\_record;  int memberid;  String member\_type;  String birthday;  String company;    public ClientPO(String n,String c,int c1,String cr,int m,String mt,String b,String co){  name=n;  contact=c;  credit=c1;  credit\_record=cr;  memberid=m;  member\_type=mt;  birthday=b;  company=co;  }  public String getname(){  return name;  }  public String getcontact(){  return contact;  }  public int getcredit(){  return credit;  }  public String getcredit\_record(){  return credit\_record;  }  public int getmemberid(){  return memberid;  }  public String getmember\_type(){  return member\_type;  }  public String getbirthday(){  return birthday;  }  public String getcompany(){  return company;  }  } |
| OrderPO | public class OrderPO implements Serializable {  int id;  String state;  boolean execute;  String hotel;  String start\_time;  String end\_time;  String latest\_execute\_time;  String room\_type;  int room\_number;  String strategy;  int price;  int except\_number\_of\_people;    public OrderPO(int i,String s,boolean e,String h,String st,String et,String lt,String rt,int rm,String str,int p,int en){  id=i;  state=s;  execute=e;  hotel=h;  start\_time=st;  end\_time=et;  latest\_execute\_time=lt;  room\_type=rt;  room\_number=rm;  strategy=str;  price=p;  except\_number\_of\_people=en;  }  public int getid(){  return id;  }  public String getstate(){  return state;  }  public boolean getexecute(){  return execute;  }  public String gethotel(){  return hotel;  }  public String getstart\_time(){  return start\_time;  }  public String getend\_time(){  return end\_time;  }  public String getlatest\_execute\_time(){  return latest\_execute\_time;  }  public String getroom\_type(){  return room\_type;  }  public int getroom\_number(){  return room\_number;  }  public String getstrategy(){  return strategy;  }  public int getprice(){  return price;  }  public int getexcept\_number\_of\_people(){  return except\_number\_of\_people;  }  } |
| HotelStrategyPO | public class HotelStrategyPO implements Serializable{  String name;  String condition;  String start\_time;  String end\_time;  String executeway;  boolean superposition;    public HotelStrategyPO(String n,String con,String st,String et,String ew,boolean iss){  name=n;  condition=con;  start\_time=st;  end\_time=et;  executeway=ew;  superposition=iss;  }  public String getname(){  return name;  }  public String getcondition(){  return condition;  }  public String getstart\_time(){  return start\_time;  }  public String getend\_time(){  return end\_time;  }  public String getexecuteway(){  return executeway;  }  public boolean getsuperposition(){  return superposition;  }  } |
| WebStrategyPO | public class WebStrategyPO implements Serializable{  String name;  String condition;  String start\_time;  String end\_time;  String executeway;  boolean superposition;    public WebStrategyPO(String n,String con,String st,String et,String ew,boolean iss){  name=n;  condition=con;  start\_time=st;  end\_time=et;  executeway=ew;  superposition=iss;  }  public String getname(){  return name;  }  public String getcondition(){  return condition;  }  public String getstart\_time(){  return start\_time;  }  public String getend\_time(){  return end\_time;  }  public String getexecuteway(){  return executeway;  }  public boolean getsuperposition(){  return superposition;  }  } |
| HotelPO | public class HotelPO {  String address;  String bussiness\_address;  String introduction;  String service;  String room\_state;  String room\_type;  int room\_number;  int room\_price;  String order;  String evaluation;    public HotelPO(String a,String ba,String in,String s,String rs,String rt,int rn,int rp,String o,String e){  address=a;  bussiness\_address=ba;  introduction=in;  service=s;  room\_state=rt;  room\_type=rt;  room\_number=rn;  room\_price=rp;  order=o;  evaluation=e;  }  public String getaddress(){  return address;  }  public String getbussiness\_address(){  return bussiness\_address;  }  public String getintroduction(){  return introduction;  }  public String getservice(){  return service;  }  public String getroom\_state(){  return room\_state;  }  public String getroom\_type(){  return room\_type;  }  public int getroom\_number(){  return room\_number;  }  public int getroom\_price(){  return room\_price;  }  public String getorder(){  return order;  }  public String getevaluation(){  return evaluation;  }  } |
| HotelWorkerPO | public class HotelWorkerPO implements Serializable{  String name;  int contact;  String hotel;    public HotelWorkerPO(String n,int c,String h){  name=n;  contact=c;  hotel=h;  }  public String getname(){  return name;  }  public int getcontact(){  return contact;  }  public String gethotel(){  return hotel;  }  } |
| WebManagerPO | public class WebManagerPO implements Serializable{  String name;  int contact;    public WebManagerPO(String n,int c){  name=n;  contact=c;  }  public String getname(){  return name;  }  public int getcontact(){  return contact;  }  } |
| WebMarketPO | public class WebMarketPO implements Serializable{  String name;  int contact;    public WebMarketPO(String n,int c){  name=n;  contact=c;  }  public String getname(){  return name;  }  public int getcontact(){  return contact;  }  } |

6.2 Txt持久化格式

Txt数据保持格式以HotelWorker.txt为例。每行分别对应名字，联系方式，所属酒店。中间用“：”隔开，如下所示：

张三：11111111111：如家

李四：22222222222：汉庭

6.3数据库表

数据库表包含:

User表

Client表

Order表

HotelStrategy表

WebStrategy表

Hotel表

HotelWorker表

WebManger表

WebMarket表