

[文档标题]

[文档副标题]



[日期]

[公司名称]

[公司地址]

目录

[1 引言 2](#_Toc471073341)

[1.1 编制目的 2](#_Toc471073342)

[1.2 词汇表 2](#_Toc471073343)

[1.3 参考资料 2](#_Toc471073344)

[2 产品概述 2](#_Toc471073345)

[3 体系结构设计 2](#_Toc471073346)

[4 结构视角 2](#_Toc471073347)

[4.1 系统结构总述 2](#_Toc471073348)

[4.2 界面层的分解 2](#_Toc471073349)

[4.2.1 登陆模块 2](#_Toc471073350)

[4.2.2 用户模块 2](#_Toc471073351)

[4.2.3 酒店模块 2](#_Toc471073352)

[4.2.4 网站营销人员模块 2](#_Toc471073353)

[4.2.5 网站管理人员模块 2](#_Toc471073354)

[4.3 逻辑层的分解 2](#_Toc471073355)

[4.3.1 逻辑层模块综述 2](#_Toc471073356)

[4.3.2 Login 模块 2](#_Toc471073357)

[4.3.3 user 模块 2](#_Toc471073358)

[4.3.4 hotel 模块 2](#_Toc471073359)

[4.3.5 order 模块 7](#_Toc471073360)

[4.3.6 promotion 模块 11](#_Toc471073361)

[4.3.7 WebStaff 模块 21](#_Toc471073362)

[4.4 数据层的分解 21](#_Toc471073363)

[4.4.1 数据层设计思想 21](#_Toc471073364)

[4.4.2 Login 模块 21](#_Toc471073365)

[4.4.3 Hotel 模块 21](#_Toc471073366)

[4.4.4 Order 模块 21](#_Toc471073367)

[4.4.5 Promotion 模块 21](#_Toc471073368)

[4.4.6 WebStaff 模块 21](#_Toc471073369)

[5 依赖视角 21](#_Toc471073370)

1. 引言
   1. 编制目的

本报告详细完成对酒店预订管理系统的详细设计，达到指导后续软件构造的目的，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编写，是了解系统的导航

* 1. 词汇表

|  |  |
| --- | --- |
| 缩写或单词 | 解释 |
| Order | 订单 |
| Search | 搜索 |
| Hotel | 酒店 |
| Webstaff | 网站工作人员 |
| Bl | 业务逻辑 |
| Promotion | 促销 |
| WMHotel | 网站管理酒店（WebManagerHotel） |

* 1. 参考资料

《软件工程与计算（卷二）软件开发的技术基础》

《酒店预订系统用例文档V1.0》

《酒店预订系统需求规格说明文档V1.0》

1. 产品概述

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

1. 体系结构设计

参考《酒店预订系统体系结构设计文档》的描述。

主要采用分层的架构方式，采用分布式的部署方式，将数据存储在服务器端，客户端

通过RMI方式调用获取数据

1. 结构视角
   1. 系统结构总述

根据体系结构的设计，我们将系统分为展示层、业务逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。层与层之间通过接口单向调用。界面层只能调用逻辑层提供的接口，逻辑层只能调用数据层提供的接口。

逻辑层与界面层之间通过依赖vo模块传递值对象，数据层与逻辑层之间通过依赖po模块传递数据的持久化对象。

* 1. 界面层的分解

界面层的主体为四个不同的使用者模块，通过身份选择界面，登陆模块验证后可进入相应的使用者的模块。

* + 1. 登陆模块

1. 界面展示

登陆模块的界面参见软件人机交互文档

1. 逻辑展示

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 包名 | 类名 | 功能 |
| login | logincontroller | LoginUIController | 登陆界面，其中如果是用户，可以选择注册 |
| RoleChooseUIController | 身份选择 |

* + 1. 用户模块

1. 界面展示

用户模块的展示参见软件人机交互文档

1. 逻辑展示

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 包名 | 类名 | 功能 |
| user | usercontroller | AddCommentUIController | 添加评价 |
| BrowseHotelAnchorPane | 酒店浏览界面（用户视角）的gridpane的格子里，以实现翻页的功能 |
| BrowseHotelUIController | 酒店浏览 |
| CommentAnchorPane | 酒店详情界面中的一栏评价 |
| CommentUIController | 酒店详情中的浏览评价 |
| CommerceMemberRegisterUIController | 注册企业会员 |
| CommonMemberRegisterUIController | 注册普通会员 |
| CreateOrderUIController | 生成订单 |
| GuideUIController | 用户的导航栏，查看个人资料和退出以及返回 |
| HotelInfoUIController | 酒店详情 |
| LoadingUIController | 加载过渡 |
| ModifyPasswordUIController | 修改密码 |
| ModifyUserInfoUIController | 编辑个人资料 |
| MyIsMemberUIController | 显示我的会员信息 |
| MyIsnotMemberUIController | 提示用户注册企业会员或普通会员 |
| OrderInfoUIController | 订单详情 |
| RoomInfoUIController | 酒店详情下的客房信息，可以新建订单 |
| SearchHotelUIController | 用户首页，搜索酒店 |
| SelectConditionUIController | 筛选条件 |
| SingleCreditRecordAnchorPane | 信用记录上一栏信用记录信息 |
| SingleOrderOfBrowseAnchorPane | 订单浏览界面上一栏订单信息 |
| UserCreditRecordUIController | 信用记录浏览 |
| UserGuideUIController | 用户导航栏，可以跳转到个人资料，我的会员，我的订单，我的信用 |
| UserInfoUIController | 用户个人资料展示 |
| UserOrderUIController | 订单浏览 |
| UserRegisterUIController | 用户注册 |
| usercontrollertools | UIJumpTool | 用户视角下的界面跳转 |
| UserInfoUtil | 专门用来从逻辑层拿数据，并且存储当前用户id,当前地址和商圈，以及当前酒店名称（如果有的话） |
| UserInputFormCheckTool | 用户输入格式检查 |
| UserUIFXMLFactory | 加载fxml文件 |

* + 1. 酒店模块

1. 界面展示

酒店模块的界面参见人机交互文档

1. 逻辑展示
   * 1. 网站营销人员模块
2. 界面展示

网站营销人员的界面参见人机交互文档

1. 逻辑展示
   * 1. 网站管理人员模块
2. 界面展示

网站管理人员的界面参见人机交互文档

1. 逻辑展示

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 包名 | 类名 | 功能 |
| web manager | webmanagercontroller | AboutUsUIController | 展示关于我们的信息 |
| AddHotelUIController | 添加酒店 |
| AddWebSalesmanUIController | 添加网站营销人员 |
| CheckUserInfoUIController | 网管查看用户信息 |
| ManafeWebsalesmanUIController | 管理网站营销人员 |
| ManageHotelUIController | 管理酒店 |
| ManagerHomepageUIController | 网管首页 |
| ManageUserUIController | 管理用户 |
| ModifyHotelUIController | 修改酒店信息 |
| ModifyOwnPasswordUIController | 网管修改自己的密码 |
| ModifyUserInfoUIController | 修改用户信息 |
| ModifyWebSalesmanUIController | 网站管理人员修改营销人员信息界面 |
| SingleHotelAnchorPane | 管理酒店界面上的一栏酒店信息 |
| SingleUserAnchorPane | 管理用户界面上的一栏用户信息 |
| SingleWebSalesmanAnchorPane | 管理网站营销人员界面上的一栏信息 |
| SuccessAddHotelUIController | 提示添加酒店成功，显示账号和密码 |
| SuccessAddWebSalesmanUIController | 提示添加网站营销人员成功成功，显示账号和密码 |
| webmanagercontrollertools | WebManagerInfoUtil | 从逻辑层获取数据 |
| WebManagerUIFXMLFactory | 加载fxml |

* 1. 逻辑层的分解
     1. 逻辑层模块综述

逻辑层分为login，user，hotel，order，promotion，webstaff六个模块。每个模块向上层提供接口，并通过相应的controller类实现，实现过程中可能需要调用模块内部的其他类或数据层提供的接口。逻辑层中不同模块中向上层提供的接口类可能不止一个，相应的controller类也可能不止一个。

数据层内部，模块与模块之间的调用通过调用其他模块中的controller类实现。Controller类中不止有实现向上层提供的接口的方法，还包括想逻辑层内其他模块提供的方法，以及类内部调用的方法。

* + 1. Login 模块

1. 模块概述

逻辑层login模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

逻辑层login模块的职责和接口参见软件体系结构文档对该模块的描述。

1. 整体结构

逻辑层login模块具有用户、酒店、网站营销人员与网站管理人员的登陆、离线职责，以及用户的注册职责。

LoginController类会将登陆、离线的业务逻辑处理委托给Login类，将用户的注册的逻辑处理委托给user模块。

Login类会将检查密码的业务逻辑处理委托给user、hotel、webstaff模块，通过调用数据曾提供的接口修改在线人员持久化对象。

逻辑层login模块的设计如下图所示：

逻辑层login模块各个类的职责如下表所示：

表 1 逻辑层login模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责对应于登陆注册界面所需要的服务 |
| Login | 处理登陆和退出的逻辑 |

1. 模块内部类的接口规范

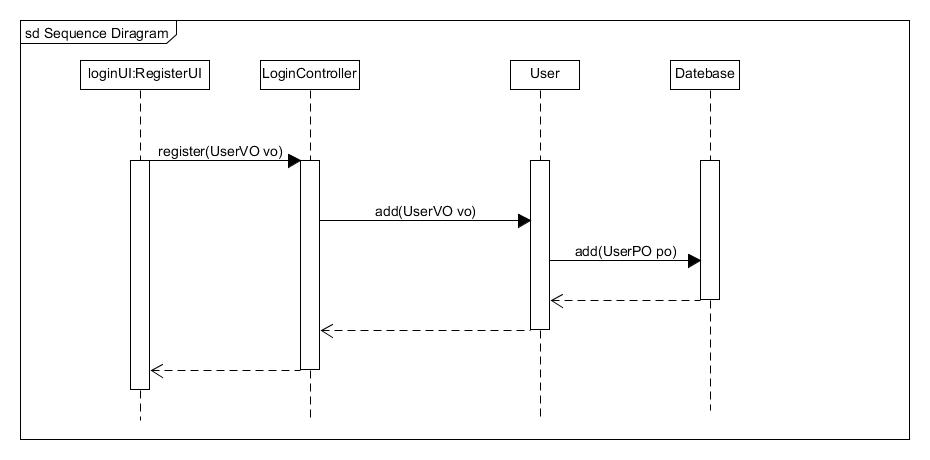
LoginController 的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginController.register | 语法 | public ResultMessage register(UserVO vo) |
| 前置条件 | 输入符合输入规则 |
| 后置条件 | 调用UserController类的add方法 |
| LoginController.checkOnline | 语法 | public ResultMessage checkOnline(Role role, String id, String password) |
| 前置条件 | 无 |
| 后置条件 | 调用Login类的checkOnline方法 |
| LoginController.logout | 语法 | public ResultMessage logout(Role role, String id) |
| 前置条件 | 无 |
| 后置条件 | 调用Login类的logout方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserController.add(UserVO userVO) | 注册对象并返回注册的结果 | |
| Login.checkOnline(Role role, String id, String password | 用户登陆并返回登陆的结果 | |
| Login.logout(Role role, String id) | 用户离线并返回离线的结果 | |

Login的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Login.checkOnline | 语法 | public static ResultMessage checkOnline(Role role, String id, String password) |
| 前置条件 | 账号与密码匹配，且账号处于离线状态 |
| 后置条件 | 增加该账号的在线持久化对象。 |
| Login.logout | 语法 | public static ResultMessage logout(Role role, String id) |
| 前置条件 | 该账号处于登陆状态 |
| 后置条件 | 删除该账号的在线持久化对象 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserController.checkPassword(String userID, String password) | 判断用户账号与密码是否匹配并返回结果 | |
| HotelController.checkPassword(String hotelID, String password) | 判断酒店账号与密码是否匹配并返回结果 | |
| WebStaffController.checkPassword(String id, String password) | 判断网站工作人员（营销人员与管理人员）的账号与密码是否匹配并返回结果 | |
| LoginDao.addOnline(OnlinpersonPO po) | 增加在线人员持久化对象，并返回增加结果（包括是否有登陆冲突） | |
| LoginDao.deleteOnline(Role role, String id) | 删除在线人员持久化对象，并返回删除结果（包括是否已经登陆） | |

1. 业务逻辑层的动态模型

其中用户注册的相关对象之间的协作为：

* + 1. user 模块

1. 模块概述

逻辑层user模块承担的需求参见需求规格说明文档。

逻辑层user模块的职责及接口参见软件体系结构说明文档。

1. 整体结构

逻辑层user模块向上层提供UserService接口，由UserController类实现。为消除user与order模块的双向依赖，在order模块内部增加了相应的接口，并在user模块内增加了相应的实现类UserForOrderController。

模块内部有User对象处理与User数据有关的逻辑职责，User对象持有一个UserPO对象的引用以记录对象的相关属性。CreditRecordList对象处理与信用记录有关的职责，MemberHelper类处理信用与会员等级之间的变化关系。

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserController | 实现对应与user界面所需要的服务 |
| UserForOrderController | 实现为了消除双向依赖而增加的UserForOrderService 接口 |
| User | 用户的领域模型对象，拥有用户的账号和相应po的引用，完成对user的增删改查操作。 |
| CreditRecordList | 信用记录的领域模型对象，完成对用户信用记录的修改 |
| MemberHelper | 辅助完成信用与会员等级之间的关系计算。 |

逻辑层user模块各个类的职责如下表所示：

1. 模块内部类的接口规范

UserController 的接口规范如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| UserController.getInfo | 语法 | | public userVO getInfo(String userID) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 返回该用户的值对象 |
| UserController.update | 语法 | | public ResultMessage update(UserVO userVO) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 更新用户信息 |
| UserController.addCreditRecord | 语法 | | public ResultMessage addCreditRecord (CreditRecordVO vo) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 增加一条信用记录 |
| UserController.gerCreditRecordList | 语法 | | public Iterator<CreditRecordVO> getCreditRecordList(String userID) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 得到该用户所有的信用记录 |
| UserController.registerMember | 语法 | | public ResultMessage registerMember(String userID)  public ResultMessage registerMember(String userID, String commerceName) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 调用user领域对象的registerMember方法 |
| UserController.setMemberStandard | 语法 | | public ResultMessage serMemberStandard(int boundaryForLevel) |
| 前置条件 | | 无 |
| 后置条件 | | 调用MemberHelper.setMemberStandard |
| UserController.getMemberStandard | 语法 | | public int getMemberStandard() |
| 前置条件 | | 无 |
| 后置条件 | | 调用MemberHelper.getBoundaryForLevel |
| UserController.getMemberLevel | 语法 | | public int getMemberLevel(int credit) |
| 前置条件 | | 无 |
| 后置条件 | | 调用MemberHelper.getMemberLevel(credit) |
| UserController.changePassword | 语法 | | public ResultMessage(String userID, String oldPassword, String newPassword) |
| 前置条件 | | 账号与密码匹配 |
| 后置条件 | | 调用user.changePassword |
| UserController.checkPassword | 语法 | | public ResultMessage(String userID, String password) |
| 前置条件 | | 该用户存在 |
| 后置条件 | | 调用user.checkPassword |
| UserController.getUserVOS | 语法 | | public ArrayList<UserVO> getUserVOS() |
| 前置条件 | | 无 |
| 后置条件 | | 返回所有用户值对象列表 |
| UserController.addCredit | 语法 | | public ResultMessage addCredit(int value, String userID) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 调用creditRecordList.add |
| UserController.hasReserved | 语法 | | public Boolean hasReserved(String userID, String hotelID) |
| 前置条件 | | 该用户已存在 |
| 后置条件 | | 调用user.hasReserved |
| UserController.add | 语法 | | public ResultMessage add(UserVO userVO) |
| 前置条件 | | 该用户不存在 |
| 后置条件 | | 调用User.add |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| user.getInfo | | 返回用户信息的值对象 | |
| userDao.update | | 更新用户单一持久化对象 | |
| creditRecordList.addCreditRecord | | 增加用户信用记录 | |
| creditRecordList.getCreditRecordList | | 返回用户信用记录列表 | |
| user.registerMember | | 注册用户和企业会员 | |
| MemberHelper.setMemberStandard | | 设置会员升级所需的信用值 | |
| MemberHelper.getBoundaryForLevel | | 返回会员升级所需的信用值 | |
| MemberHelper.getMemberLevel | | 返回当前信用值所在的会员等级 | |
| User.changePassword | | 修改密码并返回结果 | |
| User.checkPassword | | 检查密码并返回结果 | |
| CreditRecordList.add | | 增加信用记录 | |
| user.hasReserved | | 返回用户是否预定过该酒店 | |
| User.add | | 增加用户 | |

UserForOrderController的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserForOrderController.get  Credit | 语法 | public int getCredit(String userID) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回用户信用值 |
| UserForOrderController.addCreditRecordForExecute | 语法 | public ResultMessage addCreditRecordForExecute(String userID, String orderID, int change, Date executeTime) |
| 前置条件 | 该用户和订单已存在 |
| 后置条件 | 调用CreditRecordList.addCreditRecord |
| UserForOrderController.addCreditRecordForCancel Abnormal | 语法 | public ResultMessage addCreditRecordFor CancelAbnormal(String userID, String orderID, int change, Date cancelTime) |
| 前置条件 | 该用户和订单已存在 |
| 后置条件 | 调用CreditRecordList.addCreditRecord |
| OrderForUser.minusCredit RecordForCancel | 语法 | public ResultMessage minusCreditRecordFor Cancel(String userID, String orderID, int value, Date cancelTime) |
| 前置条件 | 该用户和订单已存在 |
| 后置条件 | 调用CreditRecordList.addCreditRecord |
| OrderForUser.canGenerate | 语法 | public Boolean can Generate(String userID) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 调用user.canGenerate |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| User.getInstance | 得到user的领域模型对象的引用 | |
| CreditRecordList.addCreditRecord | 增加信用记录 | |
| User.canGenerate | 判断用户是否可以生成订单 | |

User的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.getInstance | 语法 | public static User getInstance(String userID) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回该用户领域模型对象的引用 |
| User.getInfo | 语法 | public UserVO getInfo() |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回该用户的值对象 |
| User.add | 语法 | public static ResultMessage add(UserVO userVO) |
| 前置条件 | 该用户不存在 |
| 后置条件 | 调用userDao.insert |
| User.checkPassword | 语法 | public ResultMessage checkPassword(String password) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回密码是否匹配 |
| User.changePassword | 语法 | public ResultMessage changePassword(String oldPassword, String new Password) |
| 前置条件 | 该用户已存在，且旧密码匹配 |
| 后置条件 | 修改密码 |
| User.canGenerate | 语法 | Public Boolean canGenerate() |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回用户是否可以生成新的订单 |
| User.registerMember | 语法 | public ResultMessage registerMember();  public ResultMessage registerMember(String commerceName) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回用户的会员等级 |
| User.changeCredit | 语法 | public void changeCredit(int credit) |
| 前置条件 | 该会员已存在 |
| 后置条件 | 更新用户的信用值和会员等级（如果已经是会员） |
| User.update | 语法 | private ResultMessage update() |
| 前置条件 | 该用户已存在 |
| 后置条件 | 调用UserDao.update |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDao.getInfo | 得到用户单一持久化对象 | |
| UserDao.insert | 增加用户单一持久化对象 | |
| UserDao.update | 更新用户单一持久化对象 | |

CreditRecordList的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditRecordList.get Instance | 语法 | protecte static CreditRecordList getInstance(String userID) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回相应用户的信用记录领域模型对象的引用 |
| CreditRecordList.add | 语法 | public ResultMessage add (int value) |
| 前置条件 | 网站营销人员为用户线下充值时调用，该用户已存在 |
| 后置条件 | 调用CreditRecordList.addCreditRecord |
| CreditRecordList.getCredit RecordList | 语法 | public Iterator<CreditRecordVO> getCreditRecord List() |
| 前置条件 | 该用户已存在 |
| 后置条件 | 返回该用户信用记录 |
| CreditRecord.addCredit Record | 语法 | public ResultMessage addCreditRecord (CreditAction creditAction, String userID, String orderID, int change, Date changeTime);  public ResultMessage addCreditRecord (CreditRecordVO vo) |
| 前置条件 | 该用户已存在 |
| 后置条件 | 调用creditRecordListDao.addCreditRecord 增加信用记录 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CreditRecordListDao.get CreditRecordList | 得到特定用户的信用记录列表 | |
| CreditRecordListDao.add | 增加信用记录的单一持久化数据对象 | |

MemberHelper的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MemberHelper.get Instance | 语法 | public static MemberHelper getInstance() |
| 前置条件 | 无 |
| 后置条件 | 返回MemberHelper的对象的引用 |
| MemberHelper.serMember Standard | 语法 | public ResultMessage setMemberStandard(int boundaryForLevel) |
| 前置条件 | 网站营销人员制定会员等级 |
| 后置条件 | 调用MemberHelper.update() |
| MemberHelper.get BoundaryForLevels | 语法 | public int get BoundaryForLevels() |
| 前置条件 | 无 |
| 后置条件 | 返回现有酒店的 |
| MemberHelper.getMember  Level | 语法 | public int getMemberLevel(int credit) |
| 前置条件 | 无 |
| 后置条件 | 返回信用值对应的会员等级 |
| MemberHelper.update | 语法 | private ResultMessage update() |
| 前置条件 | 无 |
| 后置条件 | 调用userDao.setMemberLevel |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDao.getMemberLevel | 更新持久化数据中的会员升级条件 | |
| UserDao.setMemberLevel | 得到持久化数据中的会员升级条件 | |

* + 1. hotel 模块

1. 模块概述

逻辑层hotel模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

逻辑层hotel模块的职责及接口参见软件体系结构描述文档

1. 整体结构

hotel模块向界面层提供HotelInfoService, HotelService, ManagerHotelService, SearchHotelService 的接口，分别提供得到酒店信息，对酒店信息管理，新增酒店，搜索酒店的服务。相应有HotelInfoController, HotelController, ManagerHotelController, SearchHotelController 作为实现类。另有Hotel的领域模型对象处理对Hotel信息的处理以及RoomAvail和RoomManager处理酒店可用房间的操作与对酒店房间的操作。

逻辑层hotel模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| HotelController | 实现修改酒店信息所需要的服务 |
| HotelInfoController | 实现得到酒店信息所需要的服务 |
| ManagerHotelController | 实现网站管理人员添加酒店所需要的服务 |
| SearchHotelController | 实现搜索酒店所需要的服务 |
| Hotel | 酒店的领域模型对象，实现对酒店信息的修改服务 |
| RoomManager | 实现对酒店房间的增加和删除服务 |
| RoomAvail | 实现对酒店可用房间数量的线下修改服务 |

1. 模块类内部的接口规范

HotelController 的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelController.getHotel Info | 语法 | public HotelVO getHotelInfo(String hotelID) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 返回酒店值对象 |
| HotelController.getRoom List | 语法 | public ArrayList<RoomVO> getRoomList(String hotelID) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 得到酒店房间的值对象列表 |
| HotelController.addSpecial Room | 语法 | public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 该酒店已存在且该房间类型不存在 |
| 后置条件 | 调用Hotel.addSpecialRoom |
| HotelController.delete SpecialRoom | 语法 | public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 该酒店与房间类型均已存在 |
| 后置条件 | hotel.deleteSpecialRoom |
| HotelController.getImage Addresses | 语法 | public ArrayList<String> getImageAddresses (String hotelID) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 返回该酒店所有图片地址列表 |
| HotelController.getImage Address | 语法 | public String getImageAddresses(String hotelID, String roomType) |
| 前置条件 | 该酒店及房间已经存在 |
| 后置条件 | 返回该酒店房间的图片地址 |
| HotelController.getRoom AvailList | 语法 | public ArrayList<RoomAvailVO> getRoomAvailList (String hotelID, Date checkIn) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 调用HotelDao.getRoomAvailList返回特定酒店特定日期所有房间的可用数量 |
| HotelController.plusRoom Avail | 语法 | public ResultMessage plusRoomAvail (String hotelID, String roomType, int num, Date checkIn, Date checkout) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 调用roomAvail.changeRoomAvail 修改可用数量 |
| HotelController.plusRoom Avail | 语法 | public ResultMessage minusRoomAvail (String hotelID, String roomType, int num, Date checkIn, Date checkout) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 调用roomAvail.changeRoomAvail 修改可用数量 |
| HotelController.numOfRoom Avail | 语法 | public int numOfRoomAvail( String hotelID, String roomType, Date checkIn, Date checkout) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 调用roomAvail.getRoomAvailNum |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Hotel.getInstance | 得到酒店领域模型对象的引用 | |
| Hotel.getHotelInfo | 得到酒店的值对象 | |
| Hotel.getRoomList | 得到酒店的房间列表 | |
| Hotel.addSpecialRoom | 增加房间类型 | |
| Hotel.getImageAddresses | 得到酒店图片地址 | |
| Hotel.deleteSpecialRoom | 减少房间类型 | |
| Hotel.checkPassword | 检查密码是否正确 | |
| RoomAvail.changeRoom Avail | 更改可用客房数量 | |
| RoomAvail.getRoomAvail Num | 得到可用客房数量 | |
| HotelDao.updateHotel | 更新酒店信息 | |
| HotelDao.getRoomAvailList | 得到可用房间数量值对象列表 | |

HotelInfoController 的接口规范如下图所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelInfoController.get RoomNorm | 语法 | public ArrayList<RoomNormVO> getRoomNorm (String hotelID) |
| 前置条件 | Hotel已存在 |
| 后置条件 | 调用Hotel.getRoomNorms |
| HotelInfoController.get CheckInDDL | 语法 | public String getCheckInDDL(String hotelID) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 调用hotel.checkInDDL |
| HotelInfoController.update Grade | 语法 | public ResultMessage updateGrade(String hotelID, int grade) |
| 前置条件 | 该酒店已存在 |
| 后置条件 | 调用hote.updateGrade(grade) |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Hotel.getRoomNorms | 得到酒店的房间类型值对象列表 | |
| Hotel.updateGrade | 更新酒店评分 | |
| Hotel.getCheckInDDL | 得到酒店最晚入住时间 | |

SearchHotelController 的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SearchHotelController.get HotelList | 语法 | public ArrayList<HotelVO> getHotelList (String province, String city, String area);  public ArrayList<HotelVO> getHoteList(); |
| 前置条件 | 无 |
| 后置条件 | 调用hotelDao.getHotelList 返回符合条件的酒店值对象列表。 |
| SearchHotelController. select | 语法 | public ArrayList<HotelVO> select (ArrayList <HotelVO> hotelList, SelectConditionVO selectConditionVO)；  public ArrayList<HotelVO> hotelList, String hotelName); |
| 前置条件 | 无 |
| 后置条件 | 返回符合条件的酒店列表，内部调用SearchHOtel.match |
| SearchHotelController.sort | 语法 | public ArrayList<HotelVO> sort(ArrayList<HotelVO> hotelVOs, SortBy sortBy, SortMethod sortM) |
| 前置条件 | 无 |
| 后置条件 | 返回符合排序要求的hotel值对象列表，内部调用SearchHotel.compare |
| SearchHotelController. match | 语法 | private Boolean match(HotelVO hotelVO, SelectConditionVO selectConditionVO) |
| 前置条件 | 无 |
| 后置条件 | 返回酒店是否符合排序要求 |
| SearchHotelController. compare | 语法 | private boolean compare(HotelVO hotelVO1, HotelVO hotelVO2, SortBy sortBy, SortMethod, SortM) |
| 前置条件 | 无 |
| 后置条件 | 返回两酒店的比较结果，调用OrderForHotel.has Reserved |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.getHotelList | 得到持久化数据对象列表 | |
| Hotel.hasEnoughRoom | 是否有足够多的可用客房 | |
| OrderForUserController.has reserved | 用户是否预定过特定酒店 | |

ManagerHotelController的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManagerHotelController. getHotelID | 语法 | public String getHotelID(String district) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.getNewHotelID |
| ManagerHotelController. addHotel | 语法 | public ResultMessage addHotel(HotelVO hotelVO) |
| 前置条件 | 无 |
| 后置条件 | 调用 Hotel.addHotel(hotelVO) |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.getNewHotelID | 得到某地区一个新的酒店编号 | |
| Hotel.addHotel | 增加一个酒店并返回结果 | |

Hotel的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.addHotel | 语法 | public static ResultMessage addHotel(HotelVo hotelVO) |
| 前置条件 | 该酒店不存在 |
| 后置条件 | 调用HotelDao.addHotel |
| Hotel.addSpecialRoom | 语法 | public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 调用RoomManager.addSpecialRoom |
| Hotel.deleteSpecialRoom | 语法 | public ResultMessage deleteSpecialRoom (RoomVO vo) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 调用RoomManager.deleteSpecialRoom |
| Hotel.getRoomList | 语法 | public ArrayList<RoomVO> getRoomList |
| 前置条件 | 无 |
| 后置条件 | 调用RoomManager.getRoomList |
| Hotel.updateRoomList | 语法 | public ResultMessage updateRoomList ( ArrayList <RoomVO> roomList) |
| 前置条件 | 无 |
| 后置条件 | 调用RoomManager.updateRoomList |
| Hotel.getRoomAvailList | 语法 | public ArrayList<RoomAvailVO> getRoomAvailList (Date checkIn, Date checkout) |
| 前置条件 | 无 |
| 后置条件 | 调用RoomAvail.getRoomAvailList(checkIn) |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.getHotel | 得到酒店的领域模型对象的引用 | |
| HotelDao.addHotel | 增加酒店 | |
| RoomManager.add SpecialRoom | 增加酒店房间类型 | |
| RoomManager.delete SpecialRoom | 删除酒店房间类型 | |
| RoomMabager.getRoom AvialList | 得到可用房间列表 | |
| RoomAvail.changeRoomAvail | 修改可用房间数量 | |
| HotelDao.updateGrade | 更新酒店评分 | |
| HotelDao.updateHotel | 更新酒店信息 | |

RoomAvail的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomAvail.getRoom AvailNum | 语法 | public int getRoomAvailNum(String roomType, Date checkIn, Date checkout) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.numOfRoomAvail |
| RoomAvail.change RoomAvail | 语法 | public ResultMessage changeRoomAvail( String roomType, Boolean isPlus, int num, Date checkIn, Date checkOut |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.changeRoomAvail |
| RoomAvail.getRoom AvailList | 语法 | public ArrayList<RoomAvailVO> getRoomAvailList ( Date checkIn) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.getRoomAvailList |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.numOfRoomAvail | 返回可用房间数量 | |
| HotelDao.changeRoomAvail | 修改可用房间数量 | |
| HotelDao.getRoomAvailList | 得到不同日期可用房间数量的列表 | |

RoomManager的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomManager.get RoomList | 语法 | public ArrayList<RoomVO> getRoomList() |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.getRoomList |
| RoomManager.addSpecial Room | 语法 | public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.addSpecialRoom |
| RoomManager.delete SpecialRoom | 语法 | public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDao.deleteSpecialRoom |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.getRoomList | 得到酒店房间的持久化对象列表 | |
| HotelDao.addSpecialRoom | 增加酒店房间类型 | |
| HotelDao.deleteSpecialRoom | 删除就i但房间类型 | |

* + 1. order 模块

1. 模块概述

逻辑层order模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

逻辑层order模块的职责及接口参见软件系统结构描述文档

1. 整体结构

order模块向上层提供OrderForUser, OrderForHotel, OrderForWebsite 三个接口，分别向用户、酒店、及网站提供服务。有相应的OrderForUser Cotroller, OrderForHotelController, OrderForWebsiteController 实现相应的服务。为了消除order模块与user模块的双向依赖，增加了UserForOrder的接口。另外有Order的领域模型对象处理与订单有关的生成、执行、撤销、撤销异常的操作。

逻辑层Order模块各个类的职责如下图所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderForUserController | 实现用户界面所需的服务 |
| OrderForHotelCotroller | 实现酒店界面所需的服务 |
| OrderForWebsiteController | 实现网站工作人员界面所需的服务 |
| Order | 订单的领域对象，处理对订单的生成执行等操作逻辑 |

1. 模块内部类的接口规范、

OrderForUserController的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderForUserController. userClassify | 语法 | public ArrayList<OrderVO> userClassify (String userID, StateOfOrder state) |
| 前置条件 | 无 |
| 后置条件 | 调用UserDao.searchByUser |
| OrderForUser.detatil | 语法 | public OrderVO detail(String orderID) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用orderDao.searchByID |
| OrderForUser.cancel MyOrder | 语法 | public int cancelMyOrder(String orderID) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用Order.cancelUnexecuted |
| OrderForUser.getOrder StateOfUser | 语法 | public StateOfOrder getOrderStateOfUser( String userID, String hotelID, |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByUserWithHotel; |
| OrderForUser. specific0rder | 语法 | public ArrayList<OrderVO> specificOrder( String userID, String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByUserWithHotel |
| OrderForUser.getTrueValue | 语法 | public String getTrueValue(OrderInfo orderInfo) |
| 前置条件 | 无 |
| 后置条件 | 调用Count.countPromotionOfRoom |
| OrderForUser.confirm Researvation(OrderVO orderVO) | 语法 | public ResultMessage confirmReservation(OrderVO orderVO) |
| 前置条件 | 无 |
| 后置条件 | 调用Order.generateOrder |
| OrderForUser.add Comment | 语法 | public ResultMessage addComment(String userId, int grade, String comment) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用Order.addComment |
| OrderForUser.getCredit Reduced | 语法 | public int getCreditReduced(OrderVO orderVO) |
| 前置条件 | 无 |
| 后置条件 | 调用Order.getCreditReduced(orderVO) |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDao.searchByUser | 返回该用户生成过的订单 | |
| OrderDao.searchByID | 根据order ID得到订单的值对象 | |
| Order.cancelUnexeuted | 返回用户撤销异常订单的结果 | |
| OrderDao.searchByUser WIthHotel | 返回用户在该酒店生成过的订单 | |
| Count.countPromotionOf Room | 计算最优策略 | |
| Order.addComment | 评价订单 | |
| Order.getCreditReduced | 得到用户撤销订单时将要减去的信用值 | |

OrderForHotelController 的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderForHotelController. hotelClassify | 语法 | public ArrayList<OrderVO> hotelClassify (String hotelID, StateOfOrder state) |
| 前置条件 | 无 |
| 后置条件 | 调用orderDao.searchByHotel |
| OrderForHotelController. execute | 语法 | public ResultMessage execute(String orderID) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用Order.execute |
| OrderForHotelController. leaveUpdate | 语法 | public ResultMessage leaveUpdate (String orderID) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用Order.leaveUpdate |
| OrderForHotelController. hotelCancelAbnormal | 语法 | public ResultMessage hotelCancelAbnormal( String userID) |
| 前置条件 | 该订单已经存在且状态为异常 |
| 后置条件 | 调用Order.cancelAbnormal Order.execute |
| OrderForHotelController. getOrderOfHotel | 语法 | private ArrayList<OrderVO>getOrderOfHotel(String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByHotel |
| OrderForHotelController. hasReserved | 语法 | public Boolean hasReserved(String userID, String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByUserWithHotel |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Order.execute | 执行订单 | |
| Order.cancelAbnormal | 撤销订单异常状态并执行 | |
| Order.leaveUpdate | 实际离开时调用，修改订单信息 | |
| OrderDao.SearchByHotel | 得到酒店所有订单的持久化数据对象 | |
| OrderDao.searchByUser WithHotel | 返回用户在该酒店生成订单的所有持久化数据对象 | |

OrderForWebsiteController 的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderForWebsite Controller.browse Unperformed | 语法 | public ArrayList<OrderVO> browseUnperformed; |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByState |
| OrderForWebsite  Controller. browse  Abnormal | 语法 | public ArrayList<OrderVO>browseAbnormal |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.searchByState |
| OrderForWebsite. Controller.webCancel Abnormal | 语法 | public ResultMessage webCancelAbnormal(String orderID, Boolean isHalf) |
| 前置条件 | 该订单已存在 |
| 后置条件 | 调用Order.cancelAbnormal |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| OrderDao.searchByState | 返回符合订单状态条件的订单持久化数据对象 | |
| Order.cancelAbnormal | 订单撤销异常订单 | |

Order的接口规范：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDao.generateOrder | 语法 | public static ResultMessage generateOrder ( OrderVO orderVO) |
| 前置条件 | 用户可以生成订单 |
| 后置条件 | 调用OrderDao.insert |
| Order.execute | 语法 | public ResultMessage execute() |
| 前置条件 | 订单状态可以被执行 |
| 后置条件 | 更新订单信息，  调用UserForOrder.addCreditRecordForExecute |
| Order.leaveUpdate | 语法 | public ResultMessage leaveUpdate() |
| 前置条件 | 订单一杯执行 |
| 后置条件 | 更新订单信息 |
| Order.cancelUnexecuted | 语法 | public ResultMessage cancelUnexecuted() |
| 前置条件 | 订单状态可以被撤销 |
| 后置条件 | 更新订单信息  调用HotelController.plusRoomAvail, |
| Order.cancelAbnormal | 语法 | public RessultMessage cancelAbnormal( Boolean isHalf) |
| 前置条件 | 该订单状态可以被撤销 |
| 后置条件 | 更新订单信息  调用UserForOrder. addCreditRecordForCancelAbnormal |
| Order.update | 语法 | private ResultMessage update() |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDao.update |
| Order.addComment | 语法 | public ResultMessage addComment(int grade, String comment) |
| 前置条件 | 订单状态为可以评价 |
| 后置条件 | 更新订单信息  调用HotelInfoController.updateGrade |
| Order.getCreditReduced | 语法 | public static int getCreditReduced(OrderVO orderVO) |
| 前置条件 | 无 |
| 后置条件 | 返回将扣除的信用值 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDao.insert | 插入持久化数据对象 | |
| UserDao.update | 更新持久化数据对象 | |
| HotelController.minus RoomAvail | 减少可用客房数量 | |
| HotelController.plus RoomAvail | 增加可用客房数量 | |
| UserForOrder. addCredit RecordForExecute | 增加执行订单的信用变更记录 | |
| UserForOrder.addCredit RecordForCancelAbnormal | 增加撤销异常订单的信用变更记录 | |

* + 1. promotion 模块

1. 模块概述

逻辑层promotion模块承担的需求参见需求规格说明文档功能需求及非功能需求。

逻辑层promotion 模块的职责及接口参见软件系统结构描述文档。

1. 整体结构

promotion模块的职责为销售策略的制定与计算。为了实现销售策略的可修改性，promotion类使用了策略模式：一个promotion对象，除了制定者等基本属性外，由Scope，Condition，Deduction三部分组成，分别代表该销售策略的地理上的适用范围，可以应用的条件，以及促销的方式。每个部分可由不同的类实现已达到促销条件的多样性和可修改行。

系统规定每个订单只可以使用一个促销策略，在所有可用的促销策略中选择最优的用来生成订单，计算过程由Count类进行。

逻辑层promotion模块各个类的职责如下图所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 实现模块想界面曾提供的方法，并向order模块提供最有促销策略的计算方法 |
| Promotion | 促销策略的领域模型对象，拥有一条促销策略所持有的制定者、适用日期、促销原因等信息，聚合适用范围、使用条件、促销方式等类，可以帮助完成促销界面所需要的服务 |
| Scope | 促销策略适用范围的领域模型对象 |
| RoomScope | 继承Scope类，是促销策略适用范围在特定房间类型促销上的具体实现 |
| HotelScope | 继承Scope类，是促销策略适用范围在单个酒店上的具体实现 |
| DistrictScope | 继承Scope类，是促销策略适用范围在一定地区内的具体实现 |
| ConditionType | 促销策略适用条件的领域模型对象 |
| NumCondition | 继承ConditionType类，是促销策略适用条件在数量要求上的具体实现 |
| TotalCondition | 继承ConditionType类，是促销策略适用条件在金额要求上的具体实现 |
| BirthdayCondition | 继承ConditionType类，是促销策略适用条件在生日特惠上的具体实现 |
| MemberCondition | 继承ConditionType类，是促销策略适用条件在普通会员优惠上的具体实现 |
| CommerceCondition | 继承ConditionType类，是促销策略适用条件在企业会员优惠上的具体实现 |
| SpecialPeriodCondition | 继承ConditionType类，是促销策略适用条件在特定日期优惠，及无其他条件上的具体实现。 |
| PromotionType | 促销方式的领域模型具体对象 |
| ReducePromotion | 继承PromotionType类，s是促销方式在减额策略上的具体实现 |
| DiscountPromotion | 继承PromotionType类，是促销方式在打折策略上的具体实现 |
| Count | 负责计算使用促销策略后的价格并选择最优 |

1. 模块内部类的接口规范

PromotionController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionController.addPromotion | 语法 | public ResultMessage addPromotion (PromotionVO promotionVO) |
| 前置条件 | 无 |
| 后置条件 | 调用promotionDao.insert |
| PromotionController.delPromotion | 语法  \\\\\\\\\\\\ | public static ResultMessagedelPromotion(String reason,String ID) |
| 前置条件 | 已存在promotion对象 |
| 后置条件 | 调用promotion.delPromotion方法 |
| PromotionController.get IDForNewPromotion | 语法 | Public String getIDForNewPromotion (String setterID |
| 前置条件 | 无 |
| 后置条件 | 调用promotionDao.getNewID(setterID) |
| PromotionController.getPromotionOfHotel | 语法 | public static ArrayList <PromotionVo> getPromotionOfHotel (String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用promotionDao.finds |
| PromotionController.getPromotionOfDistrict | 语法 | Public static ArrayList <PromotionVo> getPromotionofDistrict (String district) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionController.getPromotion ofHotel 内部方法 |
| PromotionController.countPromotionOfRoom | 语法 | public String countPromotionOfRoom(OrderInfo orderInfo) |
| 前置条件 | 输入合法 |
| 后置条件 | 调用count领域的countPromotionOfRoom方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| PromotionDao.insert | 增加promotion单一持久化对象 | |
| PromotionDao.delPromotion | 删除promotion单一持久化对象 | |
| PromotionDao.finds | 返回符合条件的多个持久化对象 | |
| Count.countPromtionOf Room | 计算订单最优方案 | |

**promotion类的接口规范如下表所示：**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口 | | |
| Promotion.setScope | 语法 | public void setScope (ScopeType type, String id, String type) |
| 前置条件 | 该促销策略对象已生成 |
| 后置条件 | 设置促销策略对象的适用范围,初始化scope对象 |
| Promotion.setCondition | 语法 | public ResultMessage setCondition (ConditionType conditionType,double conditionNum) |
| 前置条件 | 该促销策略对象已生成 |
| 后置条件 | 设置促销策略对象的适用条件,初始化condition 对象 |
| Deduction.setDeduction | 语法 | public ResultMessage setDeduction (DeductionType type, double discount) |
| 前置条件 | 该促销策略对象已生成 |
| 后置条件 | 设置促销策略对象的促销方式,初始化deduction 对象 |
| 需要的服务（虚接口） | | |
| 服务名 | 服务 | |

**Scope 类的接口规范如下表所示：**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Scope.check | 语法 | public abstract boolean check(String scope，Strint roomType) |
| 前置条件 | 该对象已存在 |
| 后置条件 | 检查是否符合领域要求 |
| Scope.getType | 语法 | public abstract ScopeType getType() |
| 前置条件 | 无 |
| 后置条件 | 返回scope中的种类属性 |
| Scope.getNum | 语法 | public abstract String getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回scope对象的scopeNum属性 |
| Scope.getRoomType | 语法 | public abstract String getRoomType() |
| 前置条件 | 无 |
| 后置条件 | 返回scop对象的roomType属性 |

**RoomScope类的接口规范如下表所示：**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomScope.check | 语法 | public Boolean check(String hotelID,String roomType) |
| 前置条件 | 无 |
| 后置条件 | 检查是否符合要求 |
| RoomScope.getType | 语法 | public ResultMessage check(String HotelAndRoom) |
| 前置条件 | 已存在该领域对象 |
| 后置条件 | 返回ScopeType.Room |
| RoomScope.getNum | 语法 | public String getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回scope对象的scopeNum属性 |
| RoomScope.getRoomType | 语法 | public String getRoomType() |
| 前置条件 | 无 |
| 后置条件 | 返回scope对象的roomType属性 |

HotelScope

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelScope.HotelScope | 语法 | public HotelScope(String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域对象 |
| HotelScope.check | 语法 | public ResultMessage check(String hotelID,String roomType) |
| 前置条件 | 该领域对象已存在 |
| 后置条件 | 返回是否符合范围条件 |
| HotelScope.getType | 语法 | public ScopeType getType() |
| 前置条件 | 无 |
| 后置条件 | 返回ScopeType.HOTEL |
| HotelScope.getNum | 语法 | public String getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回酒店编号 |

**DistrictScope**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DistrictScope.DistrictScope | 语法 | public DistrictScope(String districtID) |
| 前置条件 | 无 |
| 后置条件 | 创建该领域实体对象 |
| DistrictScope.check | 语法 | Public ResultMessage check() |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合范围条件 |
| DistrictScope.getType | 语法 | public Scope getType() |
| 前置条件 | 无 |
| 后置条件 | 返回 ScopeType.DISTRICT |
| DistrictScope.getNum | 语法 | public String getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回地区编号 |

**Condition类的接口规范：**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| ConditionType.check | 语法 | public abstract Boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合条件 |
| Condition.getType | 语法 | public anstract ConditionType getType() |
| 前置条件 | 无 |
| 后置条件 | 返回促销策略条件的类型 |
| Condition.getNum | 语法 | public abstract double getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回促销策略条件的数值（没有时返回0） |

NumCondition

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| NumCondition.NumCondition | 语法 | public NumCondition(int num) |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| NumCOndition.NumCondition | 语法 | public boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 创建一个领域实体对象 |

TotalCondition

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TotalCondition.TotalCondition | 语法 | public TotalCondition(double total) |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| TotalCondition.check | 语法 | public boolean TotalCondition(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合条件 |

BirthdayCondition

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BirthdayCondition.BirthdayCondition | 语法 | public BirthdayCondtion() |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| BirthdayCondtion.check | 语法 | public boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合条件 |

MemberCondition

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MemberCondition.MemberCondition | 语法 | public MemberConditoin(int class) |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| MemberCondition.check | 语法 | public boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合条件 |

CommerceCondition 类的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommerceCondition.CommerceCondition | 语法 | public CommerceConditoin(int class) |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| CommerceCondition.check | 语法 | public boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回是否符合条件 |

SpecialPeriod 类的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SpecialPeriiodCondition. SpecialPerionCondition | 语法 | public SpecialPerionCondition() |
| 前置条件 | 无 |
| 后置条件 | 创建一个领域实体对象 |
| SpecialPerionCondition.check | 语法 | public boolean check(OrderInfo orderInfo) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回true |

Deduction类的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Deduction.getDeduction | 语法 | public abstract double getDeduction(double total) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回优惠的价格 |
| Deduction.getType | 语法 | public abstract DeductionType getType() |
| 前置条件 | 无 |
| 后置条件 | 返回促销策略的优惠方式 |
| Deduction.getNum | 语法 | public abstract double getNum() |
| 前置条件 | 无 |
| 后置条件 | 返回促销策略的优惠力度 |

ReduceDeduction

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReduceDeduction.ReduceDeduction | 语法 | public ReduceDeduction(int reduce) |
| 前置条件 | 无 |
| 后置条件 | 生成该领域实体对象 |
| ReduceDeduction.getDeduction | 语法 | public int getDeduction(int total) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回优惠后实际价格 |

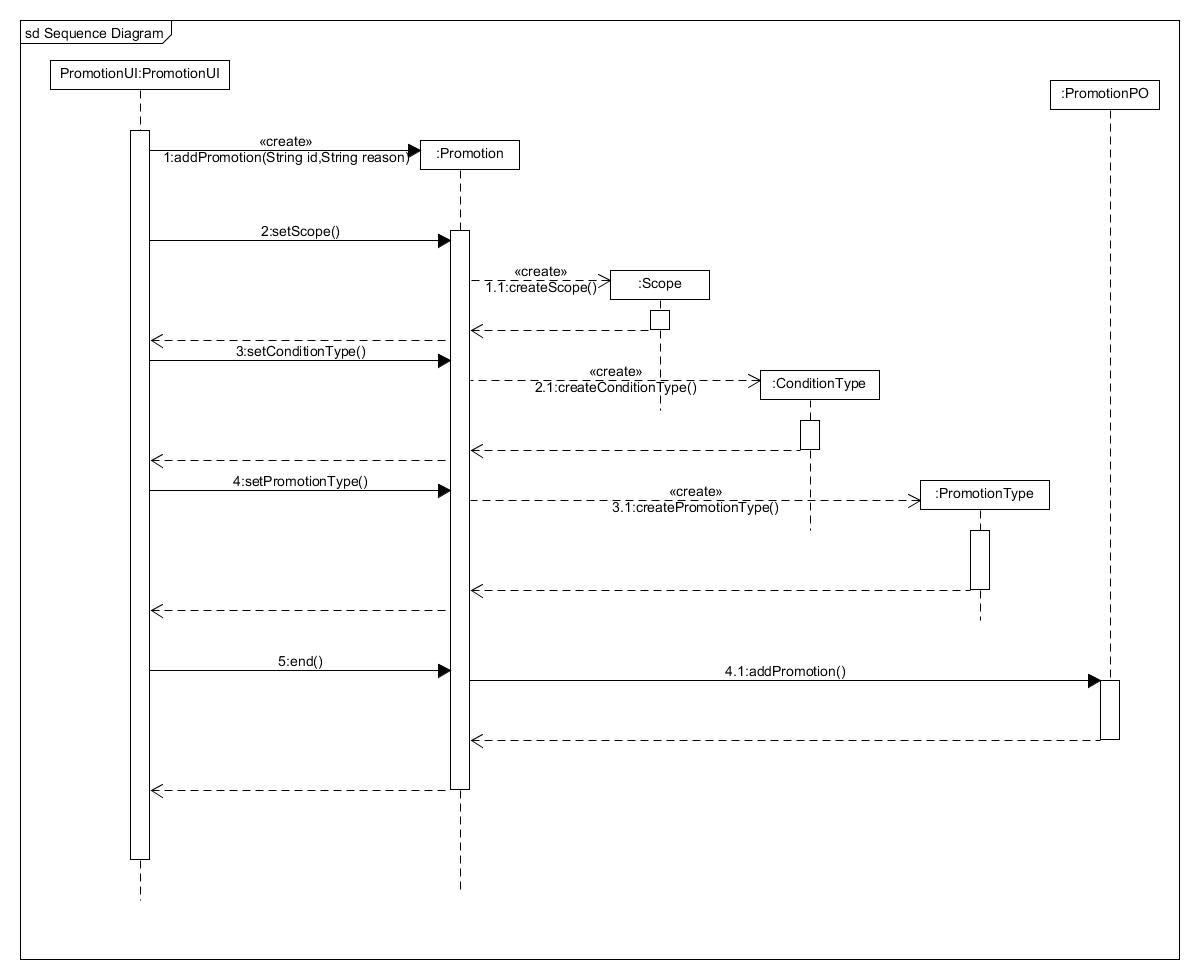
DiscountDeduction

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DiscountDeduction.DiscountDeduction | 语法 | public DiscountDeduction(int discount) |
| 前置条件 | 无 |
| 后置条件 | 生成一个领域实体对象 |
| DiscountDeduction.getDeduction | 语法 | public int getDeduction(int total) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回优惠后的实际价格 |

Count 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口) | | |
| Count.countPromotionOfRoom | 语法 | public StringcountPromotionOfRoom(String hotelID, RoomType type,int num) |
| 前置条件 | 无 |
| 后置条件 | 返回最优方案(reason#price) |
| 需要的服务（需接口） | | |
| Scope.check | 检查是否符合范围条件 | |
| Condition.check | 检查是否符合范围 | |
| Deduction.get | 得到优惠价格 | |
| promotionDao.finds | 查看符合条件的促销策略 | |

1. 业务逻辑层的动态模型



* + 1. WebStaff 模块

1. 模块概述
2. 整体结构

Web staff模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| WebstaffController | 负责实现网站工作人员所需要的服务 |
| WebSalesman | 网站营销人员的领域模型对象，拥有每个网站营销人员的账号、密码、姓名、联系方式等信息，可以帮助完成网站工作人员界面所需要的服务 |
| WebManager | 网站管理人员的领域模型对象，拥有每个网站管理人员的账号、密码等信息，可以帮助完成网站工作人员界面所需要的服务 |

1. 模块内部类的接口规范

webstaffController 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WebStaff.addWebSalesman | 语法 | public ResultMessage addWebSalesman(String id , String password) |
| 前置条件 | 没有相同id的持久化数据对象 |
| 后置条件 | 调用webSalesmanDao.insert |
| WebStaff.delWebSalesman | 语法 | public ResultMessage delWebSalesman(String id) |
| 前置条件 | 已存在相应的持久化数据对象 |
| 后置条件 | 调用WebSalesmanDao.delWebSalesman方法 |
| WebStaff.getWebSalesman | 语法 | public WebSalesmanVO getWebSalesman (String webSalesmanID ) |
| 前置条件 | 该账号存在 |
| 后置条件 | 返回值对象 |
| WebStaff.getWebSalesmanList | 语法 | public ArrayList<WebSalesmanPo> getSalesmanList() |
| 前置条件 | 无 |
| 后置条件 | 调用WebSalesmanDao.getAll() |
| WebStaffCheckPassword | 语法 | public static ResultMessage checkPassword (String ID, String password) |
| 前置条件 | 该账号存在 |
| 后置条件 | 调用checkPassword方法 |
| WebStaff.changePassword | 语法 | public static ResultMessage changePassword（String newPassword, String oldPassword，String id） |
| 前置条件 | 已存在该领域实体对象 |
| 后置条件 | 调用Salesman与WebManager领域的changePassword方法 |
| 需要的服务（需接口） | | |
| Web SalesmanDao. addSalesman | 增加一个持久化对象 | |
| WebSalesmanDao.delSalesman | 删除一个持久化对象 | |
| Web SalesmanDao. getAll | 得到网站营销人员列表 | |
| WebSalesman.changePassword | 改变网站营销人员账号密码 | |
| WebSalesman.checkPassword | 判断密码是否匹配 | |
| WebManager.changePassword | 改变网站管理人员密码 | |

WebSalesman 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WebSalesman.checkPassword | 语法 | public ResultMessage checkPasword(String passaword) |
| 前置条件 | 该账号已存在 |
| 后置条件 | 返回检查结果 |
| WebSalesman.changePassword | 语法 | public static ResultMessage changePassword（String newPassword, String oldPassword，String id） |
| 前置条件 | 已存在相应领域对象 |
| 后置条件 | 修改领域对象、持久化对象数据 |
| 需要的服务（需接口） | | |
| WebSalesmanDao.update | 更新持久化对象 | |
| WebSalesmanDao.findByID | 根据ID持久化数据对象 | |

Manager的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manager.changePassword | | 语法 | public ResultMessage changePassword（String newPassword, String oldPassword） |
| 前置条件 | 已存在相应领域对象 |
| 后置条件 | 修改领域对象、持久化对象数据 |
| WebManager.checkPassword | | 语法 | public ResultMessage checkPassword(String password) |
| 前置条件 | 无 |
| 后置条件 | 检查密码是否匹配 |
| 需要的服务（需接口） | | | |
| WebManagerDao.getManagerInstance | 得到管理人员持久化数据对象 | | |
| WebManagerDao.updateManager | | 更新网站管理人员持久化数据对象 | |

1. 业务逻辑层的动态模型
   1. 数据层的分解
      1. 数据层设计思想

根据分层结构思想，数据层封装为软件的服务器端。在运行时，服务器端runner 调用 remoteHelper 中的init方法，这个方法里注册了端口通过rmi传递了逻辑层的dao的实现类，注册生成实现类对象是通过实现类内部调用的daohelperfactory里的单例方法，得到daohelper的实现daohelperimpl的一个固定对象来实现自己的方法。 数据库的初始化调用daohelperimpl的init方法，删除数据库调用daohelperimpl的finish方法。

* + 1. login 模块

1. 模块概述

数据层login模块为逻辑层login模块提供服务，

其职责及接口参见软件体系结构描述文档。

1. 整体结构

Login模块存储当前在线人员的信息，在不同类型的使用者登陆通过验证密码后，会在数据层login存储的信息中查看是否已有在线记录，并添加其在线记录。离线时，会将该使用者的在线记录删除。

在线人员的持久化数据对象中包括使用者的类型（分为用户、酒店、网站营销人员和网站管理人员）和账号。

由于在线人员持久化对象的特殊性，将在线人员持久化对象的信息记录全部保存在服务器端的内存中，通过链表的形式存储。使用者的登陆离线表现为链表中数据的增加和删除，同一条信息不可以重复增加两次，已有的记录才可以进行删除。在服务器端关闭时，所有在线人员的记录全部被清空。

* + 1. user 模块

1. 模块概述

数据层user模块为逻辑层user模块提供服务，

其职责及接口参见软件体系结构文档。

1. 整体结构

数据层user模块的数据利用MySql数据库表和txt文件进行存储，包括两张数据库表分别存储用户基本信息，和用户信用记录变更的记录。txt文件存储会员升级所需的信用值。

存储用户基本信息的表中的数据为：用户账号（加密存储），密码（加密存储），昵称（10位字符），头像的图片地址（tinytext），生日（10位字符），会员等级（tinyint），会员种类（tinyint），企业名称（20位字符），信用值（int整数），真实姓名（加密存储），性别（tinyint），邮箱（30位字符），信用记录的数量（int）。

信用记录的信息包括：用户的账号，变更日期，订单编号（26位字符），信用变化原因种类（tinyint），改变的值（int），变更后的信用值（int）。

* + 1. hotel模块

1. 模块概述

数据层hotel模块为逻辑层hotel提供服务，

其职责及接口参见软件体系结构描述文档。

1. 整体结构

数据层hotel模块的数据利用MySql数据库进行存储，包括两张数据库表分别存储酒店的基本信息和房间的基本信息（包括不同日期下可用房间的数量）。

酒店的基本信息包括酒店的名称（10位字符以内），密码（加密存储），联系方式（11位字符），地区编码（15位字符），详细地址（20位字符），标准间价格（双精度浮点型小数），酒店简介（tinytext），酒店基本设施编码（4位字符），酒店星级（tinyint），酒店评分（双精度浮点型小数），最晚入住时间（8位字符，形式为HH：mm：ss），酒店图片地址（tinytext），酒店被评价的次数（int整数），酒店不同房间类型的数量（tinyint）。

酒店房间的基本信息包括：酒店的账号（10位字符），房间的类型（10位字符），房间的数量（int整数），房间的单价（双精度浮点型小数），房间的图片地址（tinytext），可用房间数量列表（text，保存包括今天在内的未来180天的可用房间数量的记录，不同数字之间以‘，’隔开），可用房间数量列表的上次更新时间（日期类型）。

1. 特殊设计

由于酒店的可用房间数据列表随着日期的增加而变化，因而在房间信息的数据库表中增加了可用房间数量列表的上次更新时间，在逻辑层调用数据库中的房间信息时，检查上次更新时间是否是今天。若不是，将可用房间数量列表头部相应数量的数据删除，并在列表尾部将相应数量的数据（房间的物理数量）增加；将上次更新时间改为今天的日期。

* + 1. order 模块

1. 模块概述

数据层order模块为逻辑层order模块提供服务，

其职责及接口参见软件体系结构说明文档。

1. 整体结构

订单的基本信息利用MySql数据库表进行存储，其中每一个order存储的信息包括：订单编号（26位字符），用户编号（加密存储），用户姓名（加密存储），酒店编号（10位字符），酒店名称（15位字符），订单的状态（tinyint,与java enum数据类型相对应），房间的类型（10位字符），房间单价（双精度浮点小数），房间数量（tinyint），入住人数（tinyint）,原始价格（双精度浮点小数），实际价格（双精度浮点小数），使用促销策略的名称（20位字符），订单评价（tinytext），订单评分（tinyint），预计入住时间（时间类型），预计离开时间（时间类型），酒店最晚执行时间（8位字符），订单生成时间（时间类型），订单实际入住时间（时间类型），订单实际离开时间（时间类型），撤销订单时间（时间类型），撤销异常状态的时间（时间类型）

* + 1. promotion 模块

1. 模块概述

数据层promotion模块为逻辑层promotion提供服务，

其职责及接口参见软件体系结构设计文档

1. 整体结构

促销策略在数据库中利用MySql数据库表存储，表中的信息包括：制定者的编号（10位字符），订单的编号（3位字符），促销策略名称（10位字符），起始日期（时间类型），截止日期（时间类型），适用范围的种类（tinyint），适用范围的编号（20位字符），适用条件的种类（tinyint），促销方式的种类（tinyint），促销的力度（双精度浮点类型）。

* + 1. webStaff 模块

1. 模块概述

数据层webstaff模块为逻辑层web staff模块提供服务，

其职责和接口规范参见软件体系结构设计文档。

1. 整体结构、

网站工作人员信息在数据库中以MySql数据库表的形式存储，表中的信息包括：工作人员账号（10位字符），密码（加密存储），负责地区编号（6位字符）。

1. 依赖视角