

[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
     1. 登陆模块
     2. 用户模块
     3. 酒店模块
     4. 网站营销人员模块
     5. 网站管理人员模块
  2. 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

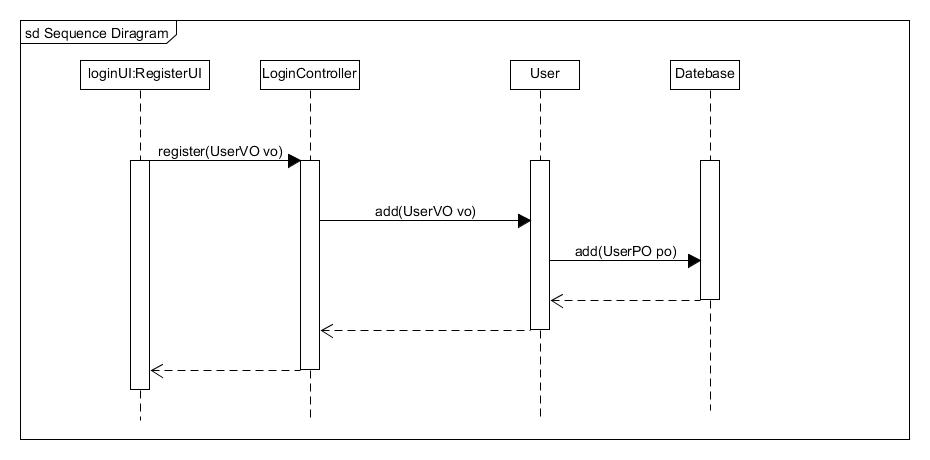
LoginController 的接口规范如下表所示：

|  |  |  |
| --- | --- | --- |
| 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(String reason, String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用promotion领域的add Promotion方法 |
| PromotionController.setScope | 语法 | public ResultMessage setScope (ScopeType type, String ID, RoomType type) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setScope方法 |
| PromotionController.setConditionType | 语法 | public ResultMessage setConditionType (ConditionType type, int requirement) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setConditionType方法 |
| PromotionController.setPromotionType | 语法 | public ResultMessage setPromotionType (PromotionType type, int discount) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setPromotionType方法 |
| PromotionController.setDate | 语法 | Public ResultMessage setDate (Date beginDate, Date endDate) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setBeginDate和setEndDate方法 |
| 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.getPromotionOfPeriod | 语法 | Public static ArrayList <PromotionVo> getPromotionOfPeriod (Date beginDate, Date endDate) |
| 前置条件 | 无 |
| 后置条件 | 调用show.getPromotionOfPeriod方法 |
| PromotionController.getPromotionOfRoom | 语法 | public static ArrayList <PromotionVo> getPromotionOfRoom (String hotelID, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 调用show.getPromotionOfRoom方法 |
| PromotionController.getPromotionOfHotel | 语法 | public static ArrayList <PromotionVo> getPromotionOfHotel (String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用show领域的getPromotionOfHotel方法 |
| PromotionController.getPromotionOfDistrict | 语法 | Public static ArrayList <PromotionVo> getPromotionOFDIstrict (String district) |
| 前置条件 | 无 |
| 后置条件 | 调用show领域的getPromotionOfDistrict方法 |
| PromotionController.countPromtionOfOrder | 语法 | Public String countPromotionOfOrder(Order order) |
| 前置条件 | 已创建该order引用的实体对象 |
| 后置条件 | 调用count领域的countPromotion方法 |
| PromotionController.countPromotionOfRoom | 语法 | public String countPromotionOfRoom(String hoteID, RoomType type, int num) |
| 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 | |
| Promotion.addPromotion | 增加一个促销策略 | |
| Promotion.setScope | 设置适用范围 | |
| Promotion.setConditionType | 设置适用条件 | |
| Promotion.setPromotionType | 设置促销方式 | |
| Promotion.setBeginDate | 设置开始日期 | |
| Promotion.setEndDate | 设置结束日期 | |
| Promotion.delPromotion | 删除促销策略 | |
| Show.getPromotionOfPeriod | 返回一定时间段促销策略 | |
| Show.getPromotionOfRoom | 返回某房间促销策略 | |
| Show.getPromotionOfHotel | 返回某酒店促销策略 | |
| Show.getPromotionOfDistrict | 返回某区域促销策略 | |
| Count.countPromtionOfOrder | 计算订单最优方案 | |
| Count.countPromotionOfRoom | 计算房间最优方案 | |

**Promotion的接口规范**

|  |  |  |
| --- | --- | --- |
|  | | |
| Promotion.addPromotion | 语法 | public ResultMessage addPromotion(String reason, String ID) |
| 前置条件 | 无 |
| 后置条件 | 生成一个新的促销策略对象 |
| Promotion.setScope | 语法 | public ResultMessage setScope (ScopeType type, String id, RoomType 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 |
| Promotion.setConditionType | 语法 | public ResultMessage setConditionType (ConditionType type, int requirement) |
| 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 |
| Promotion.setPromotionType | 语法 | public ResultMessage setPromotionType (PromotionType type, int 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 |
| Promotion.setBeginDate | 语法 | public ResultMessage setBeginDate (Date beginDate) |
| 前置条件 | 该促销策略对象已生成 |
| 后置条件 | 设置该促销策略对象的开始日期 |
| Promotion.setEndDate | 语法 | public ResultMessage setEndDate (Date EndDate) |
| 前置条件 | 改促销策略对象已生成 |
| 后置条件 | 设置改促销策略对象的结束日期 |
| Promotion.end | 语法 | public ResultMessage end() |
| 前置条件 | 该促销策略对象属性不为空 |
| 后置条件 | 调用Promotion Data Service的insertPromotion方法 |
| Promotion.cancel | 语法 | public ResultMessage cancel() |
| 前置条件 | 该促销策略对象已存在 |
| 后置条件 | 删除该促销策略对象 |
| Promotion.delPromotion | 语法 | public ResultMessage delPromotion(String reason, String ID) |
| 前置条件 | 存在该促销策略对象 |
| 后置条件 | 将改促销策略对象删除，调用PromotioDataService的delPromotion方法 |
| 需要的服务（虚接口） | | |
| 服务名 | 服务 | |
| PromotionDataService.insertPromotion(PromotionPo po) | 插入促销策略单一持久化对象 | |
| PromotionDataService.delPromotion(PromotionPo po) | 删除促销策略单一持久化对象 | |

**Scope 的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Scope.check | 语法 | public boolean check(String scope) |
| 前置条件 | 该对象已存在 |
| 后置条件 | 检查是否符合领域要求 |

**RoomScope的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| RoomScope.RoomScoope | 语法 | public RoomScope(String hotelID, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 创建一个RoomScope对象 |
| RoomScope.check | 语法 | public ResultMessage check(String HotelAndRoom) |
| 前置条件 | 已存在该领域对象 |
| 后置条件 | 返回是否符合范围条件 |

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) |
| 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 |

**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 |

**ConditionType**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| ConditionType.check | 语法 | public boolean check(int i) |
| 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 |

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(int num) |
| 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(int 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(int total) |
| 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(int isBirthday) |
| 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 MemberCondition(int class) |
| 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 |

PromotionType

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Promotion.getPromotion | 语法 | public int getPromotion(int 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 |

ReducePromotion

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReducePromotion.ReducePromotion | 语法 | public ReducePromotion(int reduce) |
| 前置条件 | 无 |
| 后置条件 | 生成该领域实体对象 |
| ReducePromotion.getPromotion | 语法 | public int getPromotion(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 |

DiscountPromotion

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DiscountPromotion.DiscountPromotion | 语法 | public DiscountPromotion(int discount) |
| 前置条件 | 无 |
| 后置条件 | 生成一个领域实体对象 |
| DiscountPromotion.getPromotion | 语法 | public int getPromotion(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 |

CouponPromotion

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CouponPromotion | 语法 | public CouponPromotion(Coupon coupon) |
| 前置条件 | 无 |
| 后置条件 | 生成一个领域实体对象 |
| CouponPromotion.getPromotion | 语法 | public int getPromotion(int total) |
| 前置条件 | 该领域实体对象已存在 |
| 后置条件 | 返回赠送券的编号 |

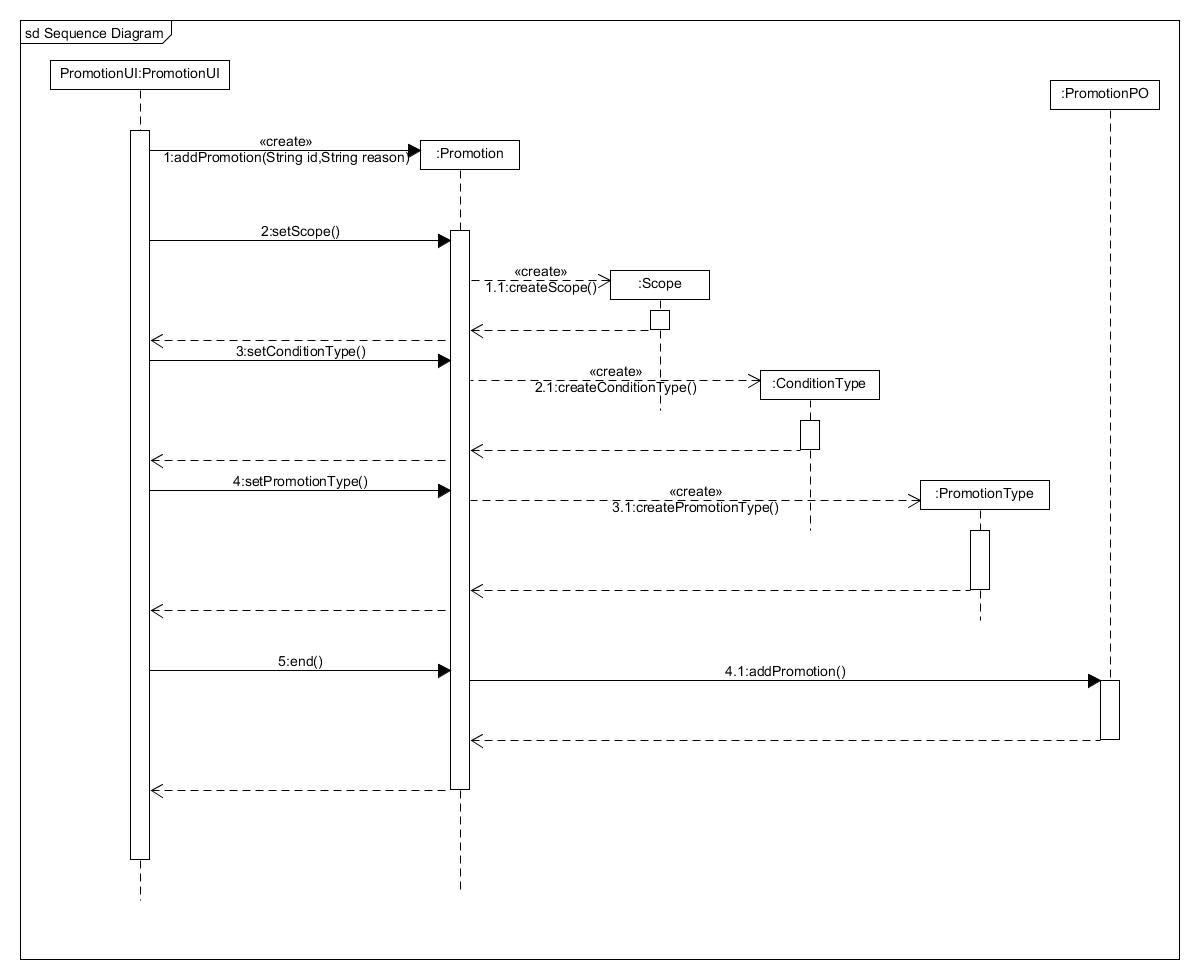
Show 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Show.getPromotionOfPeriod | 语法 | Public static ArrayList <Promotion> getPromotionOfPeriod(Date beginDate, Date EndDate) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionDataService的finds方法 |
| Show.getPromotionOfRoom | 语法 | public static ArrayList <Promotion> getPromotionOfRoom (String hotelID, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionDataService的finds方法 |
| Show.getPromotionOfDistrict | 语法 | public static ArrayList <Promotion> getPromotionOfDistrict (String district) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionDataService的finds方法 |
| 需要的服务（需接口） | | |
| PromotionDataService.finds(String field, String value) | 根据字段名和值进行查找多个持久化对象 | |

Count 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口) | | |
| Count.countPromtionOfOrder | 语法 | public String countPromotionOfOrder(Order order) |
| 前置条件 | 已存在order的领域对象 |
| 后置条件 | 返回最优方案(reason#price) |
| 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） | | |
| Promotion.ScopeType.check() | 检查是否符合范围条件 | |
| Promotion.ConditionType.check | 检查是否符合范围 | |
| Promotion.PromotionType.get | 得到优惠方式 | |
| Show.getPromotionOfPeriod | 查看一定时期内的促销策略 | |
| Show.getPromotionOfRoom | 查看某房间促销策略 | |
| Show.getPromotionOfDistrict | 查看某区域内促销策略 | |

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.addSalesman | 语法 | public ResultMessage addSalesman(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Salesman领域的addSalesman方法 |
| WebStaff.delSalesman | 语法 | public ResultMessage delSalesman(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Salesman领域的delSalesman方法 |
| WebStaff.getSalesmanList | 语法 | public ArrayList<SalesmanPo> getSalesmanList() |
| 前置条件 | 无 |
| 后置条件 | 调用Salesman领域的getSalesman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 |
| WebStaff.addManager | 语法 | public static ResultMessage addManager(String id String password) |
| 前置条件 | 不存在相同id的领域实体对象 |
| 后置条件 | 调用WebManager领域的addManager方法 |
| 需要的服务（需接口） | | |
| Salesman.addSalesman | 增加一个持久化对象 | |
| Salesman.delSalesman | 删除一个持久化对象 | |
| Salesman.getSalesmanList | 得到网站营销人员列表 | |
| Salesman.changePassword | 改变网站营销人员账号密码 | |
| Manager.changePassword | 改变网站管理人员密码 | |
| Manager.addManager | 增加网站管理人员持久化对象 | |

Salesman 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Salesman.addSalesman | 语法 | public ResultMessage addSalesman(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 | 创建一个Salesman领域对象，持久化对象 |
| Salesman.delSalesman | 语法 | public ResultMessage delSalesman(String id) |
| 前置条件 | 已存在相应的持久化数据对象 |
| 后置条件 | 删除相应的领域对象，持久化对象 |
| Salesman.getSalesmanList | 语法 | public ArrayList <SalesmanPo> getSalesmanList() |
| 前置条件 | 无 |
| 后置条件 | 返回Salesman领域对象列表 |
| Salesman.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） | | |
| WebStaffDataService.insert(String id, String password) | 插入持久化对象 | |
| WebStaffDataService.delete(String id) | 删除持久化对象 | |
| WebStaffDataService.update(String id, String password) | 更新持久化对象 | |
| WebStaffDataService.find(String field, String id) | 根据字段名和值进行查找多个持久化数据对象 | |

Manager的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Manager.changePassword | 语法 | public 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 |
| Manager.addManager | 语法 | public ResultMessage addManager(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 | 创建一个WebManager领域对象，持久化对象 |
| WebManager.delManager | 语法 | public static ResultMessage (String ID) |
| 前置条件 | 已存在相应的持久化数据对象 |
| 后置条件 | 删除该持久化数据对象 |
| 需要的服务（需接口） | | |
|  | | |
| WebStaffDataService.insert | 插入一个持久化对象 | |
| WebStaffDataService.update | 更新持久化对象 | |
| WebStaffDataService.delManager | 删除一个持久化对象 | |

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
      2. Login 模块
      3. Hotel 模块
      4. Order 模块
      5. Promotion 模块
      6. WebStaff 模块
2. 依赖视角

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
|  | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |