|  |
| --- |
| [键入公司名称] |
| 酒店预定系统（RSH）软件详细设计文档描述 |
| CMBT工作组 |
|  |
| **吕丹**  **沈丽婷**  **王馨雨**  **蔚滢璐** |
| **2016/11/13** |

|  |
| --- |
|  |

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 吕丹  蔚滢璐 | 2016-10-25 | 确定文档格式  并填写文字描述内容 | V1.0 草稿 |
| 吕丹  王馨雨 | 2016-10-26 | 将部分的hotelbl模块上传  将部分的userbl模块上传 | V1.1阶段报告 |
| 吕丹  沈丽婷  蔚滢璐 | 2016-10-27 | 更新hotelbl模块  将部分orderbl模块上传  将部分webstaffbl模块上传  将部分promotionbl模块上传 | V1.2.0  阶段成果 |
| 王馨雨  蔚滢璐  沈丽婷 | 2016-10-29 | 大体完成 | v1.2.2.1  阶段成果 |
| 吕丹  沈丽婷  王馨雨  蔚滢璐 | 2016-10-29 | 完善各个模块的部分  完善接口视角，排版，校对 | V2.0  正式版 |
| 吕丹  沈丽婷  王馨雨  蔚滢璐 | 2016-11-13 | 重构代码，修改相应的名称 | V2.1修订版 |

目录

[1.引言 4](#_Toc465541481)

[1.1编制目的 4](#_Toc465541482)

[1.2词汇表 4](#_Toc465541483)

[1.3参考资料 4](#_Toc465541484)

[2.产品概述 4](#_Toc465541485)

[3.体系结构设计概述 4](#_Toc465541486)

[4.结构视角 4](#_Toc465541487)

[4.1业务逻辑层的分解 5](#_Toc465541488)

[4.2.1Login模块 5](#_Toc465541489)

[4.1.2User模块 10](#_Toc465541490)

[4.1.3Hotel模块 21](#_Toc465541491)

[4.1.4Order模块 44](#_Toc465541492)

[4.1.5Promotion模块 58](#_Toc465541493)

[4.1.6WebStaff模块 71](#_Toc465541494)

[5、依赖视角 75](#_Toc465541495)

# 1.引言

1.1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

## 1.2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） |

## 1.3参考资料

《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》

# 2.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。

# 3.体系结构设计概述

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。

# 4.结构视角

## 4.1业务逻辑层的分解

业务逻辑层的开发包图请查看体系结构设计文档。

### ­­4.2.1Login模块

1. 模块概述：

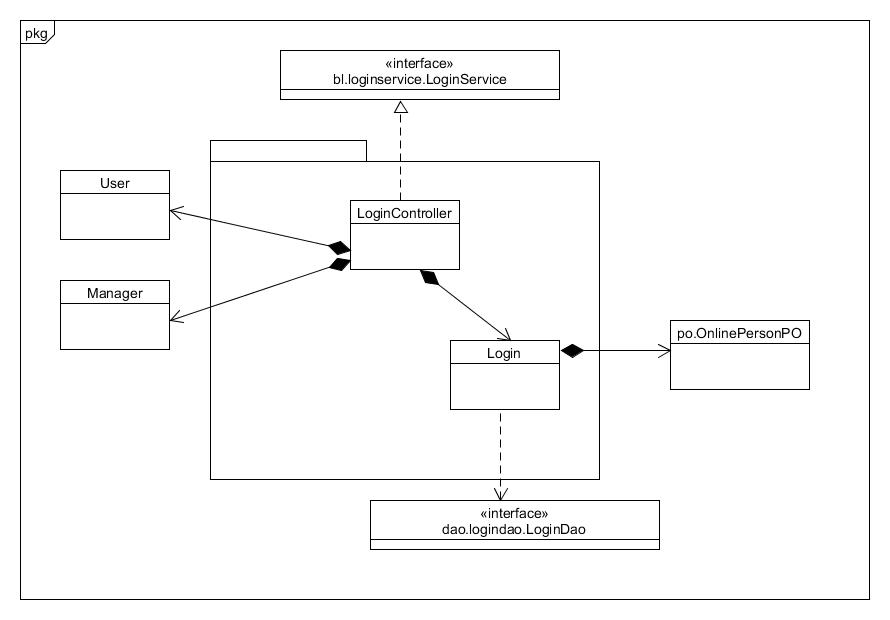
login模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

login模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

1. 整体结构：

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。在展示层和业务逻辑层之间添加LoginblService接口。在业务逻辑层和数据层之间添加Logindao接口。为了隔离业务逻辑职责和逻辑控制职责，添加LoginController，这样LoginController会将对注册的逻辑委托给Register对象，把登陆退出的逻辑委托给Login对象。而OnlinePersonPO是作为在线人员的持久化对象被添加到设计模型中去的。

**Loginbl模块的设计如图：**



**LoginBL模块中各个类的设计**

**Loginbl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 |
| register | 处理注册的逻辑 |

3、模块内部类的接口规范

**LoginController的接口规范**

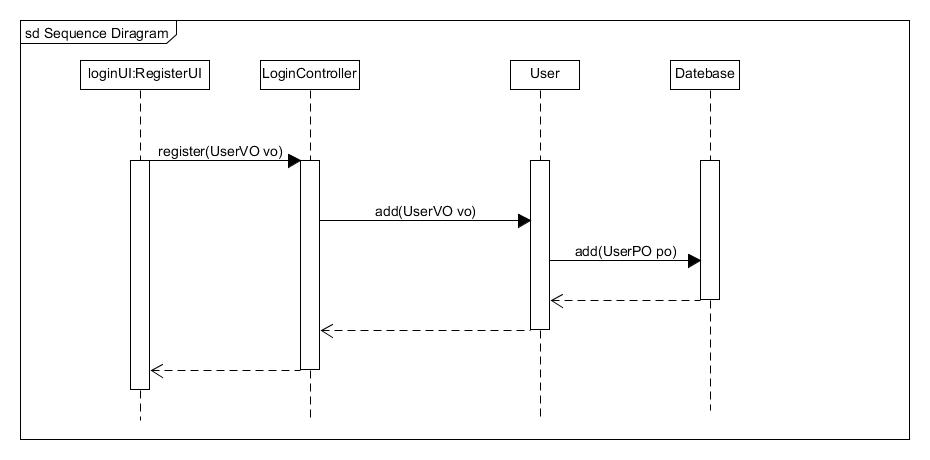
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| loginController.register | 语法 | | public ResultMessage register(UserVO vo) |
| 前置条件 | | 输入的信息符合语法规则 |
| 后置条件 | | 系统增加该用户的信息 |
| LoginController.register | 语法 | | Public ResultMessage register(WebManangerVO vo) |
| 前置条件 | | 输入的信息符合语法规则 |
| 后置条件 | | 系统增加网站管理人员信息 |
| loginController.checkOnline | 语法 | | public ResultMessage checkOnline(Role role, String id, String password) |
| 前置条件 | | 身份确认后，输入的账号、密码符合语法规则 |
| 后置条件 | | 系统显示登录验证结果。若账号与密码相匹配，系统更新在线人员记录。 |
| loginController.logout | 语法 | | public ResultMessage logout(Role role, String id) |
| 前置条件 | | 人员已登陆并选择注销登录 |
| 后置条件 | | 系统更新在线人员记录。 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| User.add(UserVO vo) | | 检查此账号是否存在，若不存在，创建该UserPO，在数据库中增加该用户的持久化对象 | |
| WebManager.add(WebManagerVO vo) | | 检查此账号是否存在，若不存在，创建该WebManagerPO，在数据库中增加该网站工作人员的持久化对象 | |
| Login.checkOnLine(Role role,String id,String password) | | 检查账号与密码是否匹配，若匹配，检查登陆冲突，返回登陆验证的结果 | |
| Login.logout(Role role,String id) | | 从数据库中删除该在线人员记录 | |

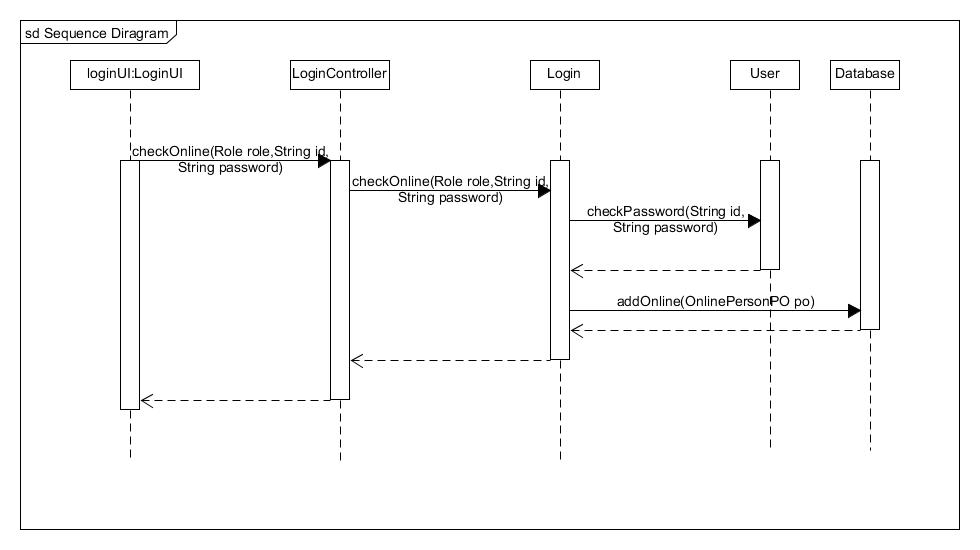
**Login的接口规范**

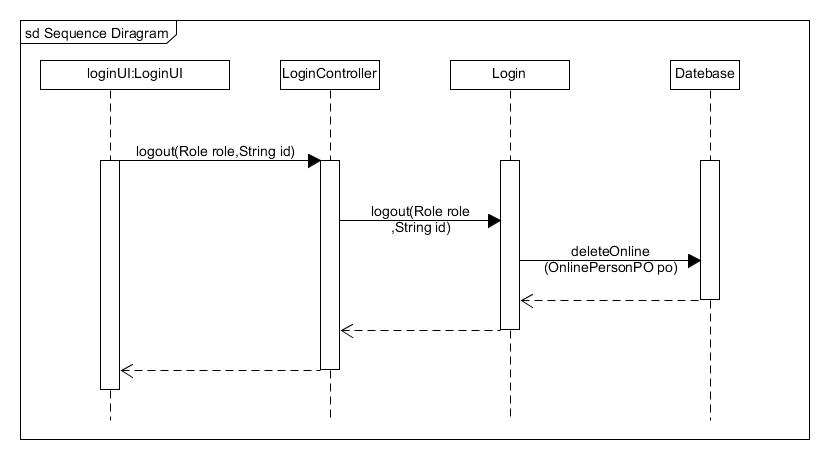
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| login.checkOnline | 语法 | | public static ResultMessage checkOnline(Role role, String id, String password) |
| 前置条件 | | 身份确认后，输入的账号、密码符合语法规则 |
| 后置条件 | | 系统显示登录验证结果。若账号与密码相匹配，系统更新在线人员记录。 |
| login.logout | 语法 | | public static ResultMessage logout(Role role, String id) |
| 前置条件 | | 人员已登陆并选择注销登录 |
| 后置条件 | | 系统更新在线人员记录。 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| User.checkPassword(String id,String password) | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 | |
| Hotel.checkPassword(String id,String password) | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 | |
| WebManager.checkPassword(String id,String password) | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 | |
| WebSalesman.checkPassword(String id,String password) | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 | |
| LoginDao.addOnline(OnlinePersonPO po) | | 检查是否已有该在线人员持久化对象，如果没有，增加在线人员持久化对象，返回登陆验证结果 | |
| LoginDao.deleteOnline(OnlinePersonPO po) | | 删除在线人员持久化对象 | |

4、业务逻辑层的动态模型

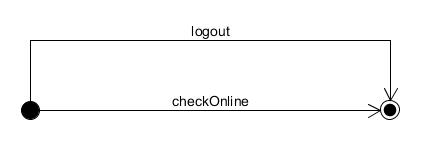
在loginbl模块中，业务分为注册和登陆，其中注册登陆的顺序图如下图所示：

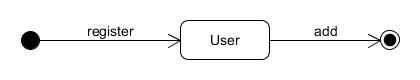






下图描述了Account对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。





5、业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.2User模块

1. 模块概述：

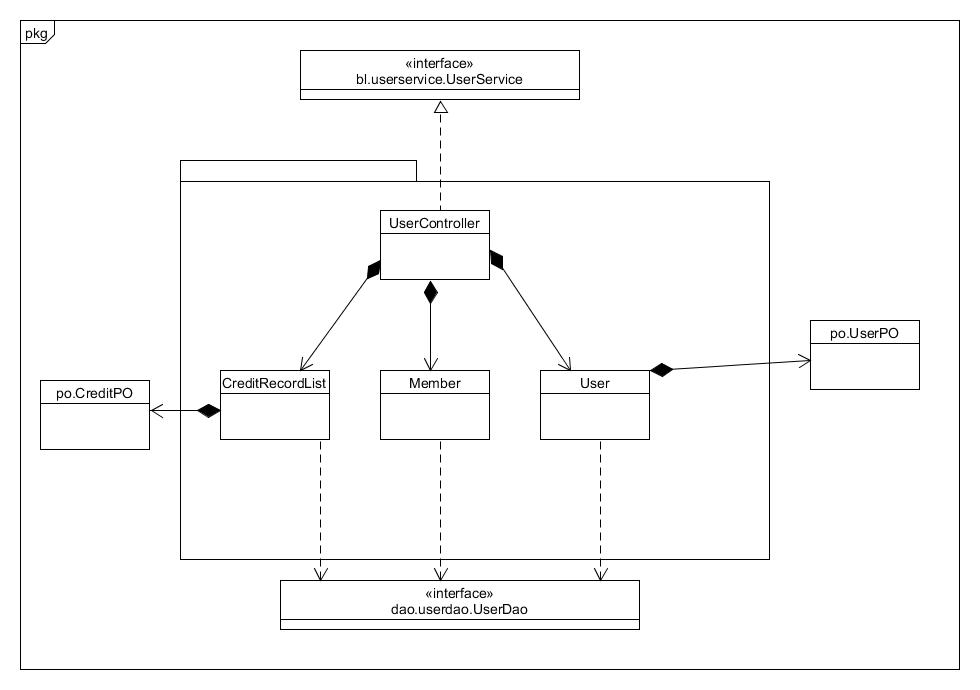
user模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

user模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

1. 整体结构：

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。在展示层和业务逻辑层之间添加UserblService接口。在业务逻辑层和数据层之间添加Userdao接口。为了隔离业务逻辑职责和逻辑控制职责，添加UserController，这样UserController会将对用户信息处理的逻辑委托给User对象。而UserPO是作为用户的持久化对象被添加到设计模型中去的，CreditRecordPO是作为信用记录的持久化对象被添加到设计模型中去的。

**Userbl模块的设计如图：**



**UserBL模块中各个类的设计**

**Userbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserController | 负责对应于用户界面所需要的服务 |
| User | 处理用户信息管理的逻辑 |
| CreditRecordList | 处理信用记录的逻辑 |
| Member | 处理注册普通会员的逻辑 |

1. 模块内部类的接口规范

**UserController的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| UserController.getInfo | 语法 | | public UserVO getInfo(String userid) |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回该用户的信息 |
| UserController.update | 语法 | | Public ResultMessage update(UserVO vo) |
| 前置条件 | | 输入的信息符合语法规则 |
| 后置条件 | | 系统更新该用户的信息 |
| UserController.addCreditRecord | 语法 | | Public ResultMessage addCreditRecord(CreditRecordVO vo) |
| 前置条件 | | 信用记录发生变化 |
| 后置条件 | | 增加该信用记录 |
| UserController.getCreditRecordList | 语法 | | public ArrayList<CreditRecordVO>getCreditRecordList(String id) |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回该用户信用记录列表 |
| UserController.registerMember | 语法 | | public ResultMessage registerMember(String id) |
| 前置条件 | | 无 |
| 后置条件 | | 系统将该用户设置成普通会员。 |
| UserController.registerMember | 语法 | | public ResultMessage registerMember(String id，String commerceName) |
| 前置条件 | | 无 |
| 后置条件 | | 系统将该用户设置成企业会员。 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| User.getInfo（） | | 返回该用户的信息 | |
| User.update（UserVO vo） | | 更新该用户的信息 | |
| CreditRecordList.getCreditRecordList() | | 返回该用户的信用记录列表 | |
| CreditRecordList.addCreditRecord(CreditRecordVO vo) | | 增加该信用记录 | |
| Member.registerMember() | | 为该用户注册普通会员 | |
| Member.registerMember(String commerceName) | | 为该用户注册企业会员 | |

**User的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.getInfo | 语法 | | public UserVO getInfo() |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回该用户的信息 |
| User.update | 语法 | | Public ResultMessage update(UserVO vo) |
| 前置条件 | | 输入的信息符合语法规则 |
| 后置条件 | | 系统更新该用户的信息 |
| User.add | 语法 | | public static ResultMessage add(UserVO vo) |
| 前置条件 | | 无 |
| 后置条件 | | 检查此账号是否存在，若不存在，创建该UserPO，在数据库中增加该用户的持久化对象 |
| User.checkPassword | 语法 | | public static ResultMessage checkPassword(String id,String password) |
| 前置条件 | | 用户输入的账号和密码格式正确 |
| 后置条件 | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| UserDao.add(UserPO po) | | 在数据库中增加该用户的持久化对象 | |
| UserDao.checkPassword(String id,String password) | | 检查该账号是否存在，若存在，检查账号与密码是否匹配 | |
| UserDao.getInfo(String id) | | 返回用户持久化对象 | |
| UserDao.update(UserPO po) | | 更新用户持久化对象 | |

**CreditRecordList的接口规范**

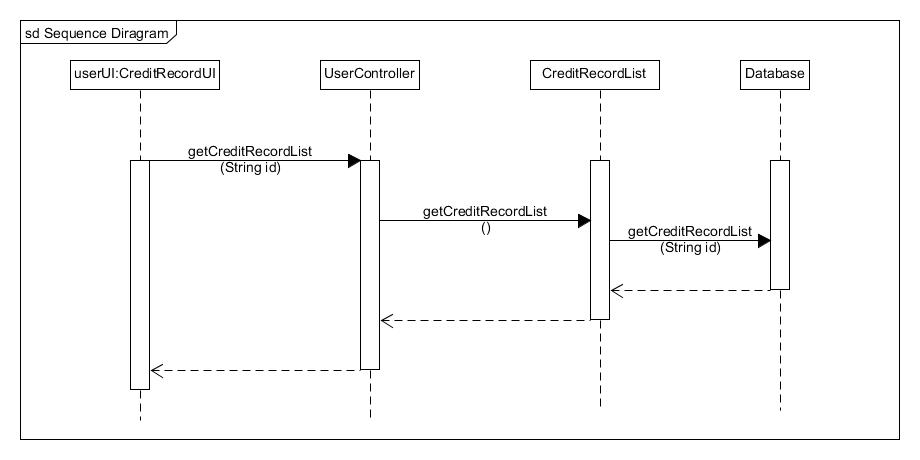
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CreditRecordList.getCreditRecordList | 语法 | | public ArrayList<CreditRecordVO> getCreditRecordList() |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回该用户信用记录列表 |
| CreditRecordList.addCreditRecord | 语法 | | Public int addCreditRecord(CreditRecordVO vo) |
| 前置条件 | | 信用记录发生变化 |
| 后置条件 | | 增加该信用记录,并返回用户累计信用值 |
| CreditRecordList.canOrder | 语法 | | public boolean canOrder() |
| 前置条件 | | 用户新建订单 |
| 后置条件 | | 检查该用户的信用值，返回该用户是否可以新建订单 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| CreditRecordDao.getCreditRecordList(String id) | | 返回该用户的信用记录列表 | |
| CreditRecordDao.addCreditRecord(CreditRecordPO po) | | 增加该条信用记录 | |

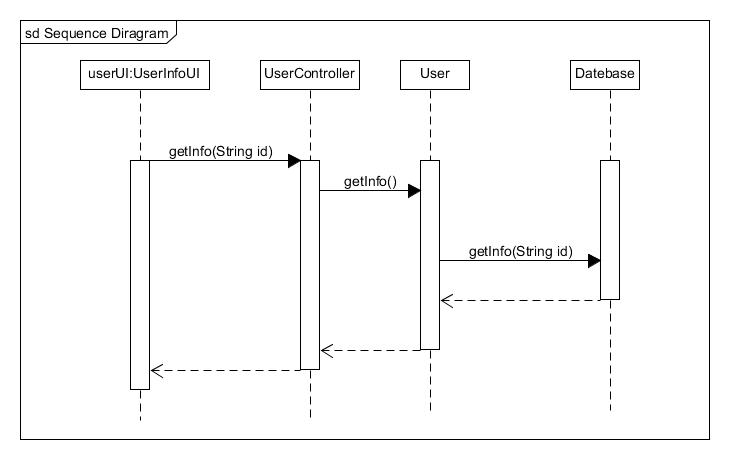
**Member的接口规范**

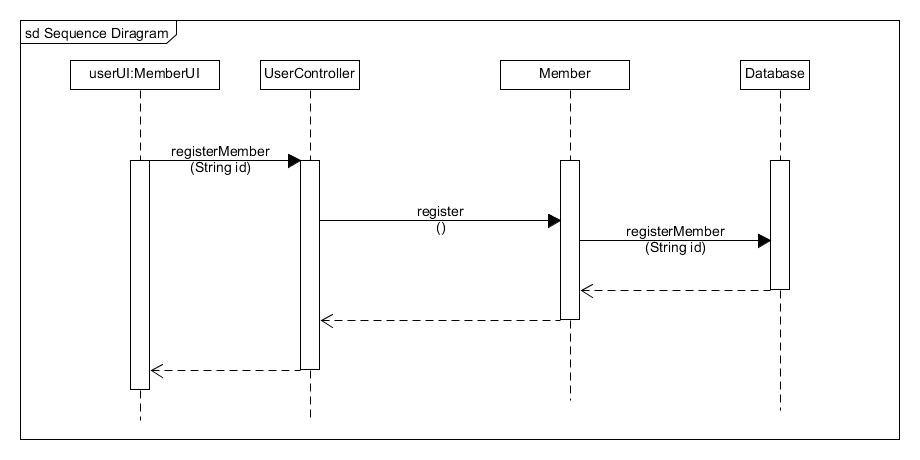
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Member.register | 语法 | | public ResultMessge register() |
| 前置条件 | | 无 |
| 后置条件 | | 系统将该用户更新为普通会员 |
| Member.register | 语法 | | public ResultMessge register(String commerName) |
| 前置条件 | | 无 |
| 后置条件 | | 系统将该用户更新为企业会员 |
| Member.setMemberLevel | 语法 | | public static ResultMessage setMemberLevel(int[][] gradeWithCredit) |
| 前置条件 | | 网站营销人员设置会员等级 |
| 后置条件 | | 更新所有会员的会员等级 |
| 需要的接口（需接口） | | | |
| 服务名 | | 服务内容 | |
| UserDao.register(String id) | | 在数据库将该用户信息更新为普通会员 | |
| UserDao.register(String id,String commerceName) | | 在数据库将该用户信息更新为企业会员 | |
| Userdao.setMemberLevel(int[][] gradeWithCredit) | | 更新所有会员的会员等级 | |

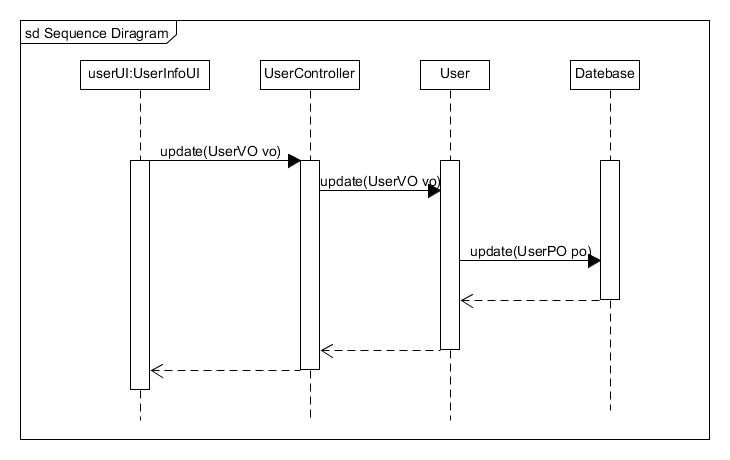
1. 业务逻辑层的动态模型

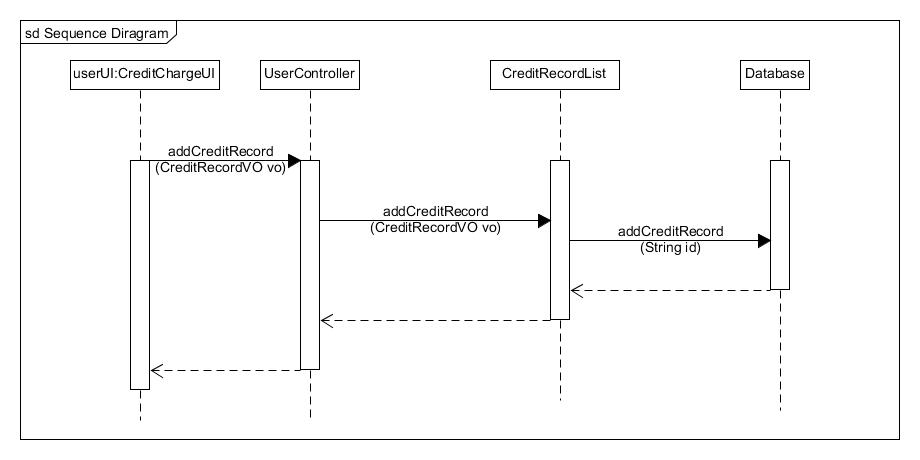
在userbl模块中，业务分为修改个人基本信息，查看信用记录，注册会员,信用充值，其顺序图如下图所示：



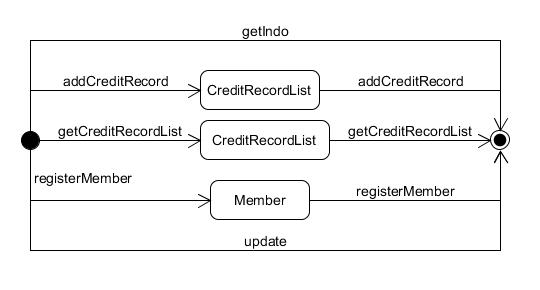








下图描述了User对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



1. 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.3Hotel模块

1、模块概述

hotelbl 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。

hotelbl 模块的职责及接口参见软件系统结构描述内对该模块的描述。

2、整体结构

hotelbl 模块的设计如下图所示

hotelbl模块中各个类的设计

**hotelbl 模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| HotelController | 负责实现对应于维护酒店个体信息的服务 |
| SearchHotelController | 负责实现对应于搜索酒店的服务 |
| WMHotelController | 负责实现对应于维护酒店全体信息的服务 |
| Hotel | 维护酒店个体信息的领域模型对象之一，拥有酒店的id等信息，可以帮助完成维护酒店个体信息所需要的服务 |
| SearchHotel | 搜索（酒店）的领域模型对象，可以帮助完成搜索界面所需要的服务 |
| WMHotel | 管理酒店全体信息的领域模型对象，拥有所有酒店的信息，可以帮助完成网站管理人员维护酒店信息、酒店工作人员安全登录的服务 |
| CommentImpl | 为酒店的添加评论 |

3、类内部的接口规范

HotelController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelController.checkPassword | 语法 | Public ResultMessage checkPassword(String id, String password) |
| 前置条件 | 该id存在 |
| 后置条件 | 返回验证结果 |
| HotelController.getHotel | 语法 | Public HotelVO getHotel () |
| 前置条件 | 该酒店存在 |
| 后置条件 | 返回该酒店的基本信息 |
| HotelController.getRoomNorms | 语法 | Public ArrayList<RoomNormVO> getRoomNorms(); |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的所有房间规格（类型，价格） |
| HotelController.updateGrade | 语法 | Public ResultMessage updateGrade(double grade) |
| 前置条件 | 无 |
| 后置条件 | 更新数据库中该酒店的评分 |
| HotelController.updateHotel | 语法 | PublicResultMessage updateHotel (HotelVO vo) |
| 前置条件 | 酒店工作人员输入的信息正确 |
| 后置条件 | 系统更新数据库中该酒店的酒店基本信息 |
| HotelController.addSpecialRoom | 语法 | Public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中增加该特色客房数据 |
| HotelController.deleteSpecialRoom | 语法 | Public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中删除该特色客房数据 |
| HotelController.getRoomList | 语法 | Public ArrayList<RoomVO> getRoomList() |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的客房信息列表 |
| HotelController.updateRoomList | 语法 | Public ResultMessage updateRoomList(ArrayList<RoomVO> roomList) |
| 前置条件 | 酒店工作人员输入的数量和价格正确 |
| 后置条件 | 系统更新数据库中的该酒店的客房信息列表 |
| HotelController.getRoomAvailList | 语法 | Public ArrayList<RoomAvailableVO>getRoomAvailList(Date date) |
| 前置条件 | 无 |
| 后置条件 | 返回该日期下的可用客房信息列表 |
| HotelController.updateRoomAvailList | 语法 | Public ResultMessage updateRoomAvailList(ArrayList<RoomAvailableVO> availableRoomList); |
| 前置条件 | 传入参数合法 |
| 后置条件 | 更新该日期下的可用客房信息列表 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| Hotel.checkPassword(String id, String password) | 验证账号与密码是否匹配 | |
| Hotel.getHotel() | 得到该酒店的基本信息 | |
| Hotel.getRoomNorms() | 得到该酒店的所有房间规格（类型，价格） | |
| Hotel.updateGrade(double grade) | 更新该酒店的星级 | |
| Hotel.updateHotel (HotelVO vo) | 更新该酒店的信息 | |
| Hotel.addSpecialRoom(RoomVO vo) | 增加该酒店的特色客房 | |
| Hotel.deleteSpecialRoom(RoomVO vo) | 删除该酒店的特色客房 | |
| Hotel.getRoomList() | 得到该酒店的客房列表 | |
| Hotel.updateRoomList(ArrayList<RoomVO> roomList) | 更新该酒店的客房信息 | |
| Hotel.getRoomAvailList(Date date) | 得到该酒店给定日期下的可用客房列表 | |
| Hotel.updateRoomAvailList(ArrayList<RoomAvailableVO> availableRoomList) | 更新该酒店给定日期下的可用客房列表 | |

**Comment的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Comment.addComment | 语法 | Public ResultMessage addComment(String id, String userID, String comment) |
| 前置条件 | 酒店id，用户id输入合法 |
| 后置条件 | 系统更新数据库中该酒店的评价信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.addComment(String id, String userID, String comment) | 在数据库该酒店信息中加入一条评论 | |

**Hotel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Hotel.checkPassword | 语法 | Public ResultMessage checkPassword(String id, String password) |
| 前置条件 | 该id存在 |
| 后置条件 | 返回验证结果 |
| Hotel.getHotel | 语法 | Public HotelVO getHotel () |
| 前置条件 | 该酒店存在 |
| 后置条件 | 返回该酒店的基本信息 |
| Hotel.getRoomNorms | 语法 | Public ArrayList<RoomNormVO vo> getRoomNorms() |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的所有房间规格（类型，价格） |
| Hotel.numOfRoomAvail | 语法 | Public int numOfRoomAvail(RoomType roomType, DatecheckIn, Date checkOut) |
| 前置条件 | 输入合法 |
| 后置条件 | 返回该酒店指定日期下该房间类型的可用数量 |
| Hotel.updateGrade | 语法 | Public ResultMessage updateGrade(double grade) |
| 前置条件 | 无 |
| 后置条件 | 更新数据库中该酒店的评分 |
| Hotel.getCheckInDDL | 语法 | Public static String getCheckInDDL(String id) |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的最晚入住时间 |
| Hotel.changeRoomAvail | 语法 | Public ResultMessage changeRoomAvail(RoomType roomType, int num, Date checkIn, Date checkOut) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新系统的可用客房信息 |
| Hotel.updateHotel | 语法 | PublicResultMessage updateHotel (HotelVO vo) |
| 前置条件 | 酒店工作人员输入的信息正确 |
| 后置条件 | 系统更新数据库中该酒店的酒店基本信息 |
| Hotel.addSpecialRoom | 语法 | Public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中增加该特色客房数据 |
| Hotel.deleteSpecialRoom | 语法 | Public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中删除该特色客房数据 |
| Hotel.getRoomList | 语法 | Public ArrayList<RoomVO> getRoomList() |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的客房信息列表 |
| Hotel.updateRoomList | 语法 | Public ResultMessage updateRoomList(ArrayList<RoomVO> roomList) |
| 前置条件 | 酒店工作人员输入的数量和价格正确 |
| 后置条件 | 系统更新数据库中的该酒店的客房信息列表 |
| Hotel.getRoomAvailList | 语法 | Public ArrayList<RoomAvailableVO> getRoomAvailList(Date date) |
| 前置条件 | 无 |
| 后置条件 | 返回该日期下的可用客房信息列表 |
| Hotel.updateRoomAvailList | 语法 | Public ResultMessage updateRoomAvailList(ArrayList<RoomAvailableVO> availableRoomList) |
| 前置条件 | 传入参数合法 |
| 后置条件 | 更新该日期下的可用客房信息列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| HotelDao.checkPassword(String id, String password) | 验证账号与密码是否匹配 | |
| HotelDao.getHotel(String id) | 得到该酒店的基本信息 | |
| HotelDao.updateGrade(String id, double grade) | 更新该酒店的星级 | |
| HotelDao.getCheckInDDL(String id) | 得到该酒店最晚入住时间 | |
| RoomAvail.changeRoomAvail(String id, RoomType roomType, int num, Date checkIn, Date checkOut) | 更新系统的可用客房信息 | |
| RoomAvail.numOfRoomAvail(RoomType roomType, Date checkIn, Date checkOut) | 得到该酒店指定日期下该房间类型的可用数量 | |
| HotelManager.updateHotel (HotelVO vo) | 更新该酒店的信息 | |
| HotelManager.addSpecialRoom(RoomVO vo) | 增加该酒店的特色客房 | |
| HotelManager.deleteSpecialRoom(RoomVO vo) | 删除该酒店的特色客房 | |
| HotelManager.getRoomList() | 得到该酒店的客房列表 | |
| HotelManager.updateRoomList(ArrayList<RoomVO> roomList) | 更新该酒店的客房信息 | |
| RoomAvail.getRoomAvailList(Date date) | 得到该酒店给定日期下的可用客房列表 | |
| RoomAvail.updateRoomAvailList(ArrayList<RoomAvailableVO> availableRoomList) | 更新该酒店给定日期下的可用客房列表 | |

**HotelManager的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| HotelManager.updateHotel | 语法 | PublicResultMessage updateHotel (HotelVO vo) |
| 前置条件 | 酒店工作人员输入的信息正确 |
| 后置条件 | 系统更新数据库中该酒店的酒店基本信息 |
| HotelManager.addSpecialRoom(RoomVO vo) | 语法 | Public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中增加该特色客房数据 |
| HotelManager.deleteSpecialRoom(RoomVO vo) | 语法 | Public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中删除该特色客房数据 |
| HotelManager.getRoomList() | 语法 | Public ArrayList<RoomVO> getRoomList() |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的客房信息列表 |
| HotelManager.updateRoomList(ArrayList<RoomVO> roomList) | 语法 | Public ResultMessage updateRoomList(ArrayList<RoomVO> roomList) |
| 前置条件 | 酒店工作人员输入的数量和价格正确 |
| 后置条件 | 系统更新数据库中的该酒店的客房信息列表 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelPO.createHotelPO | 创建一个酒店持久化对象 | |
| HotelDao.updateHotel | 更新酒店信息 | |
| RoomManager.getRoomList() | 得到该酒店的客房列表 | |
| RoomManager.updateRoomList(ArrayList<RoomVO> roomList) | 更新该酒店的客房信息 | |

**RoomManager的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomManager.addSpecialRoom(RoomVO vo) | 语法 | Public ResultMessage addSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中增加该特色客房数据 |
| RoomManager.deleteSpecialRoom(RoomVO vo) | 语法 | Public ResultMessage deleteSpecialRoom(RoomVO vo) |
| 前置条件 | 无 |
| 后置条件 | 在客房数据库中删除该特色客房数据 |
| RoomManager.getRoomList() | 语法 | Public ArrayList<RoomVO> getRoomList() |
| 前置条件 | 无 |
| 后置条件 | 返回该酒店的客房信息列表 |
| RoomManager.updateRoomList(ArrayList<RoomVO> roomList) | 语法 | Public ResultMessage updateRoomList(ArrayList<RoomVO> roomList) |
| 前置条件 | 酒店工作人员输入的数量和价格正确 |
| 后置条件 | 系统更新数据库中的该酒店的客房信息列表 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelDao.addSpecialRoom(RoomVO vo) | 增加该酒店的特色客房 | |
| HotelDao.deleteSpecialRoom(RoomVO vo) | 删除该酒店的特色客房 | |
| HotelDao.getRoomList(String id) | 得到该酒店的客房列表 | |
| HotelDao.updateRoomList(ArrayList<RoomVO> roomList) | 更新该酒店的客房信息 | |

**RoomAvail的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RoomAvail.changeRoomAvail(String id, RoomType roomType, int num, Date checkIn, Date checkOut) | 语法 | Public ResultMessage changeRoomAvail(RoomType roomType, int num, Date checkIn, Date checkOut) |
| 前置条件 | 输入合法 |
| 后置条件 | 更新系统的可用客房信息 |
| RoomAvail.numOfRoomAvail(RoomType roomType, Date checkIn, Date checkOut) | 语法 | Public int numOfRoomAvail(RoomType roomType, DatecheckIn, Date checkOut) |
| 前置条件 | 输入合法 |
| 后置条件 | 返回该酒店指定日期下该房间类型的可用数量 |
| RoomAvail.getRoomAvailList(Date date) | 语法 | Public ArrayList<RoomAvailVO> getRoomAvailList(Date date) |
| 前置条件 | 无 |
| 后置条件 | 返回该日期下的可用客房信息列表 |
| RoomAvail.updateRoomAvailList(ArrayList<RoomAvailVO> availableRoomList) | 语法 | Public ResultMessage updateRoomAvailList(ArrayList<RoomAvailVO> availableRoomList) |
| 前置条件 | 传入参数合法 |
| 后置条件 | 更新该日期下的可用客房信息列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelDao.changeRoomAvail(String id, RoomType roomType, int num, Date checkIn, Date checkOut) | 更新系统的可用客房信息 | |
| HotelDao.numOfRoomAvail(RoomType roomType, Date checkIn, Date checkOut) | 得到该酒店指定日期下该房间类型的可用数量 | |
| HotelDao.getRoomAvailList（String id, Date date） | 得到该日期下的可用客房信息列表 | |
| HotelDao.updateRoomAvailList(String id, ArrayList<RoomAvailableVO> availableRoomList) | 更新该日期下的可用客房信息列表 | |

**SearchHotel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SearchHotelController.getHotelList | 语法 | Public ArrayList<HotelVO> getHotelList(String address,String businessArea); |
| 前置条件 | 该地址和商圈存在 |
| 后置条件 | 返回符合该地址和商圈的经过综合排序的酒店列表 |
| SearchHotelController.sort | 语法 | Public ArrayList<HotelVO> sort(SortBy sortBy,SortMethod sortM) |
| 前置条件 | 无 |
| 后置条件 | 返回排序后的酒店列表 |
| SearchHotelController.select | 语法 | Public ArrayList<HotelVO> select(SelectConditionVO vo) |
| 前置条件 | 已经从数据库中取得酒店列表 |
| 后置条件 | 返回筛选后的酒店列表 |
| SearchHotelController.getOrderStateOfUser | 语法 | Public HotelOrderStateOfUser getOrderStateOfUser(String id, String userID); |
| 前置条件 | 已经从数据库中取得酒店列表 |
| 后置条件 | 返回该用户在该酒店的订单状态 |
| SearchHotelController.getHotelInfo | 语法 | Public HotelVO getHotelInfo(String id) |
| 前置条件 | 用户选择查看酒店详情 |
| 后置条件 | 返回该酒店详情 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| SearchHotel.getHotelList(String address,String businessArea) | 得到符合该地址、商圈的酒店列表 | |
| SearchHotel.sort(SortBy sortBy,SortMethod sortM) | 将酒店列表按给定规则排序 | |
| SearchHotel.select(SelectConditionVO vo) | 将酒店列表按给定条件筛选 | |
| SearchHotel.getOrderStateOfUser(String userID,String hotelID) | 得到该用户在该酒店最近一笔订单的状态 | |
| SearchHotel.getHotelInfo(String id) | 得到该酒店详情 | |

**SearchHotel 的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SearchHotel.getHotelList | 语法 | Public ArrayList<HotelVO> getHotelList(String address,String businessArea); |
| 前置条件 | 该地址和商圈存在 |
| 后置条件 | 返回符合该地址和商圈的经过综合排序的酒店列表 |
| SearchHotel.sort | 语法 | Public ArrayList<HotelVO> sort(SortBy sortBy,SortMethod sortM) |
| 前置条件 | 无 |
| 后置条件 | 返回排序后的酒店列表 |
| SearchHotel.select | 语法 | Public ArrayList<HotelVO> select(SelectConditionVO vo) |
| 前置条件 | 已经从数据库中取得酒店列表 |
| 后置条件 | 返回筛选后的酒店列表 |
| SearchHotel.getOrderStateOfUser | 语法 | Public HotelOrderStateOfUser getOrderStateOfUser(String id, String userID); |
| 前置条件 | 已经从数据库中取得酒店列表 |
| 后置条件 | 返回该用户在该酒店的订单状态 |
| SearchHotel.getHotelInfo | 语法 | Public HotelVO getHotelInfo(String id) |
| 前置条件 | 用户选择查看酒店详情 |
| 后置条件 | 返回该酒店详情 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelDao.getHotelList(String address,String businessArea) | 得到符合该地址、商圈的酒店列表 | |
| SortHotel.sort(SortBy sortBy,SortMethod sortM) | 将酒店列表按给定规则排序 | |
| SelectHotel.select(SelectConditionVO vo) | 将酒店列表按给定条件筛选 | |
| HotelDao.getHotelInfo(String id) | 得到该酒店详情 | |

**SortHotel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SortHotel.sort | 语法 | Public ArrayList<HotelVO> sort(SortBy sortBy,SortMethod sortM) |
| 前置条件 | 无 |
| 后置条件 | 返回排序后的酒店列表 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| HotelDao.sort(SortBy sortBy,SortMethod sortM) | 将酒店列表按给定规则排序 | |

**SelectHotelService**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **SelectHotel.select** | 语法 | Public ArrayList<HotelVO> select(SelectConditionVO vo) |
| 前置条件 | 已经从数据库中取得酒店列表 |
| 后置条件 | 返回筛选后的酒店列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| HotelDao.select(SelectConditionVO vo) | 将酒店列表按给定条件筛选 | |

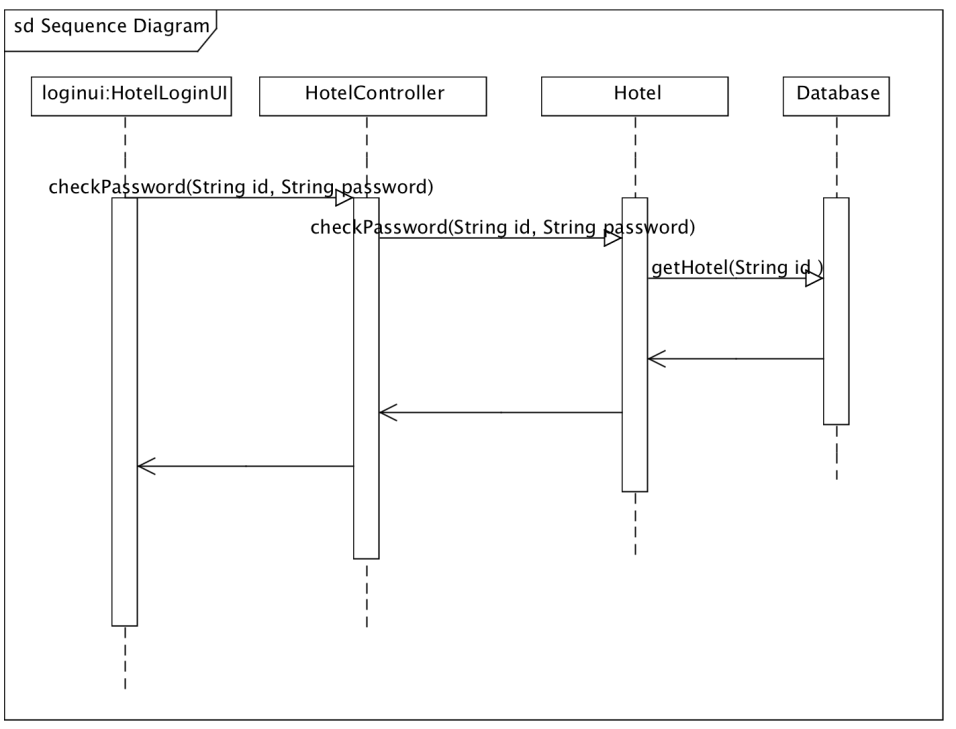
**WMHotel的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| WMHotel.getHotelNum | 语法 | Public int getHotelNum(String address); |
| 前置条件 | 无 |
| 后置条件 | 返回符合该地址的酒店数目 |
| WMHotel.addHotel | 语法 | Public ResultMessage addHotel(String id,String password) |
| 前置条件 | 该id不存在 |
| 后置条件 | 在酒店数据库中增加该酒店数据，在客房数据库中初始化该酒店的客房数据 |
| WMHotel.deleteHotel | 语法 | Public ResultMessage deleteHotel(String id) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 在酒店数据库中删除该酒店数据，在可用客房数据库中删除该酒店的可用客房数据 |
| WMHotel.updateHotelStaff | 语法 | Public ResultMessage updateHotelStaff(HotelStaffPO po); |
| 前置条件 | 该id存在 |
| 后置条件 | 更新酒店工作人员信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| HotelPO.createHotelPO | 创建一个酒店持久化对象 | |
| HotelDao. getHotelNum(String address) | 得到数据库中符合该地址的酒店数量 | |
| HotelDao. addHotel(String id,String password) | 增加一个酒店 | |
| HotelDao. deleteHotel(String id) | 删除一个酒店 | |
| HotelDao.updateHotelStaff(HotelStaffPO po) | 更新酒店工作人员信息 | |

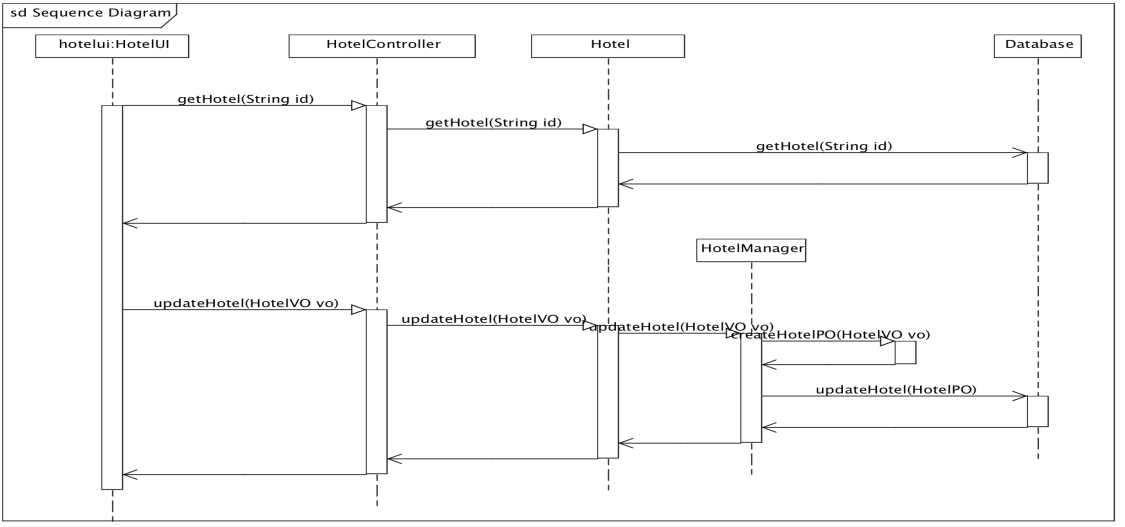
4、业务逻辑层的动态模型

酒店业务中，（部分业务逻辑：）登录验证、获得与更新酒店基本信息、酒店排序和筛选、获得酒店列表、获得用户在该酒店最近一笔订单状态、获得酒店详情、获得指定地址酒店数量及添加酒店、更新酒店工作人员信息、删除酒店的顺序图

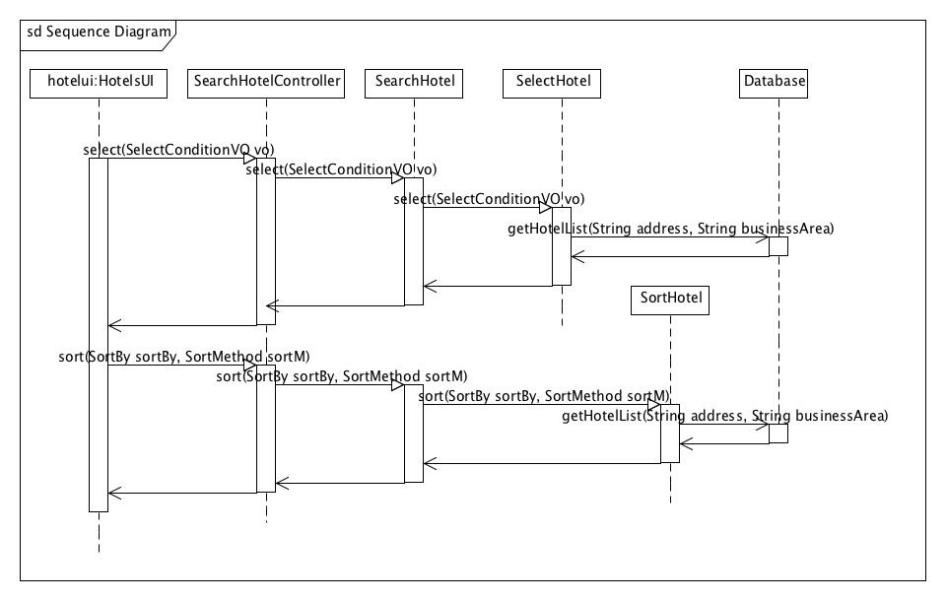
分别如下所示：



登录验证的顺序图



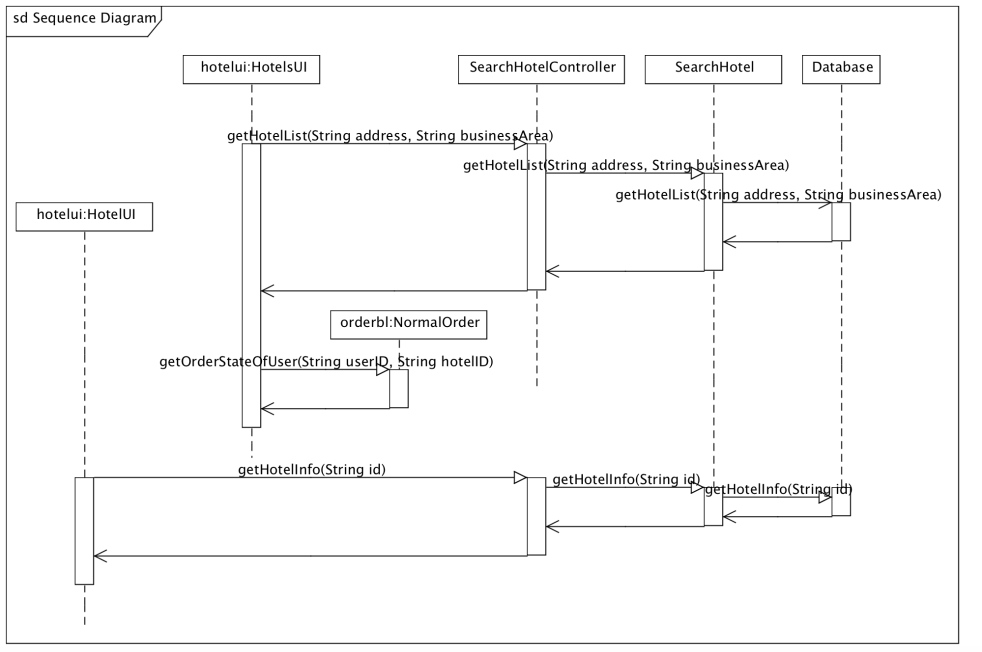
获得与更新酒店基本信息的顺序图



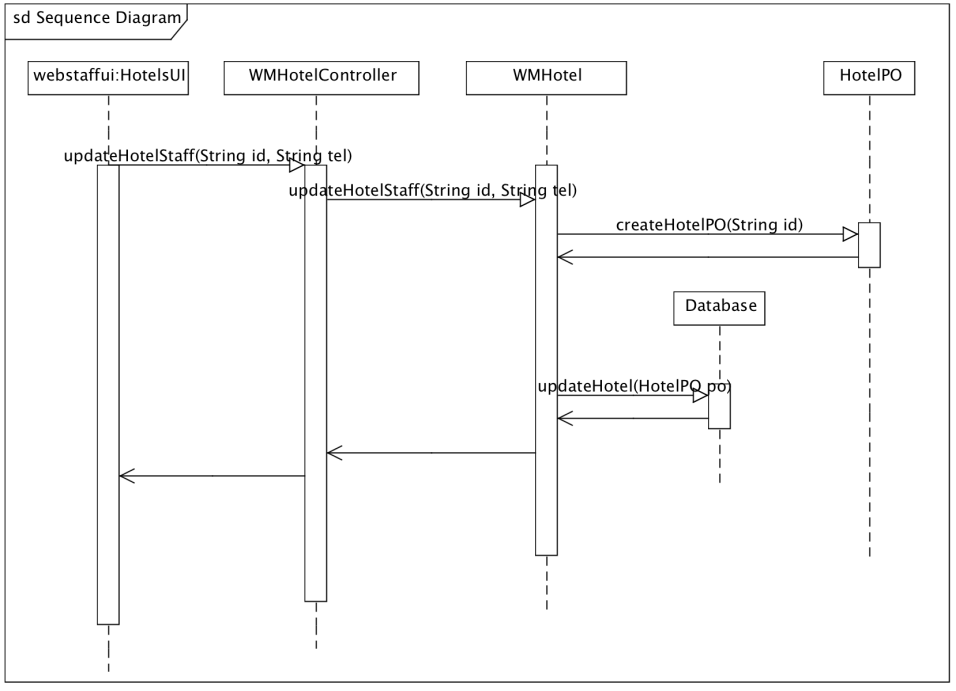
酒店排序和筛选的顺序图



获得酒店列表、得到用户在该酒店最近一笔订单状态、获得酒店详情的顺序图



获得指定地址酒店数量及添加酒店

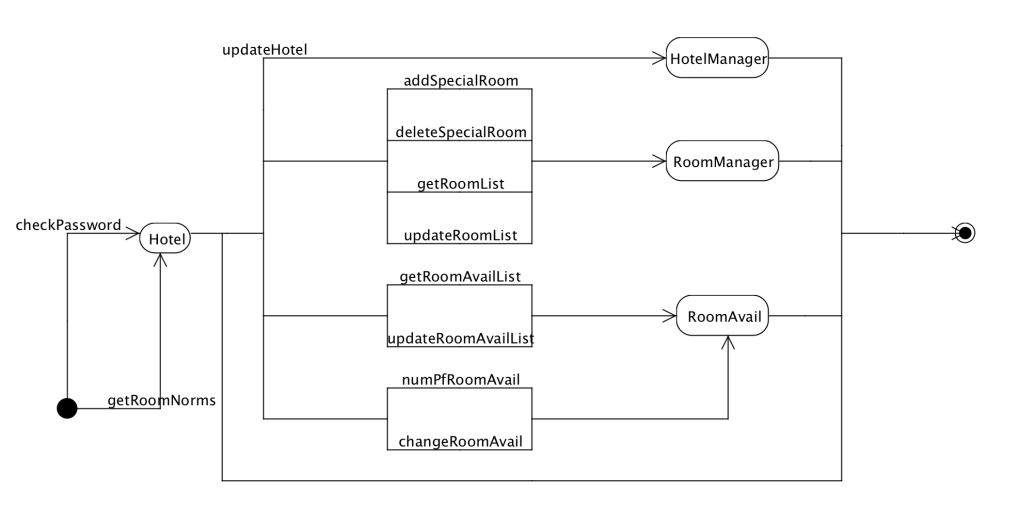


更新酒店工作人员信息的顺序图

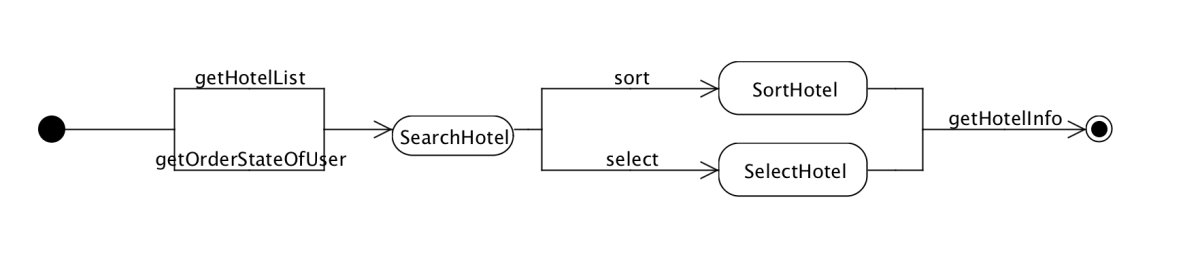
删除酒店的顺序图

Hotel对象的生存期间状态序列、引起转移的事件，以及因状态转移而伴随的动作如以下

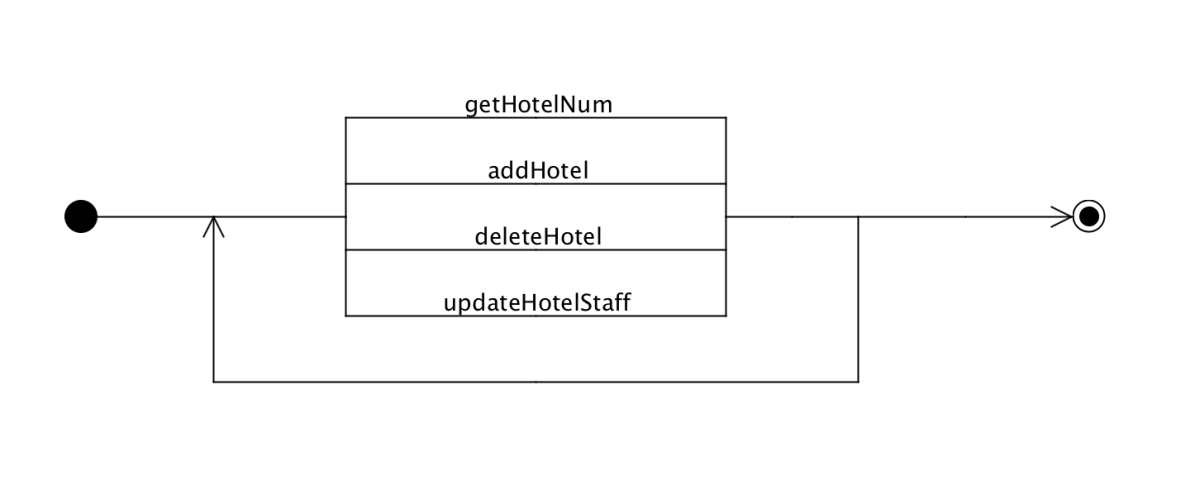
状态图所示：



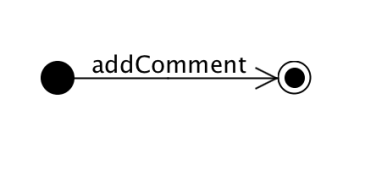
SearchHotel对象的生存期间状态序列、引起转移的事件，以及因状态转移而伴随的动作如以下状态图所示：



WMHotel对象的生存期间状态序列、引起转移的事件，以及因状态转移而伴随的动作如以下状态图所示：



Comment对象的生存期间状态序列、引起转移的事件，以及因状态转移而伴随的动作如以下状态图所示：



5、业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.4Order模块

1、模块概述：

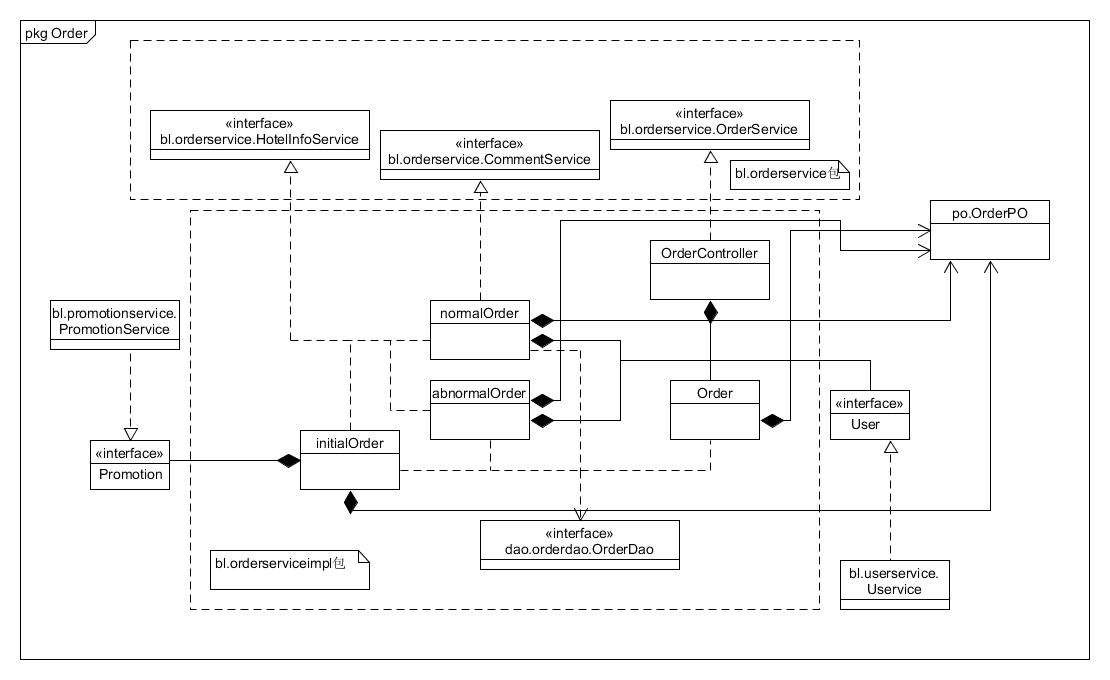
Order模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Order模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

2、整体结构：

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。在展示层和业务逻辑层之间添加OrderService接口。在业务逻辑层和数据层之间添加OrderdataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加OrderController，这样OrderController会将逻辑委托给initialOrder、Order、normalOrder、abnormalOrder对象。而OrderPO是作为在线人员的持久化对象被添加到设计模型中去的。

**OrderService模块的设计如图：**

****

**OrderService模块中各个类的设计**

**OrderService模块各个类的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| OrderController | 负责对应于订单管理所需要的服务 |
| initialOrder | 负责对应生成订单所需要的服务 |
| Order | 订单管理的领域模型对象，拥有订单的信息，可以帮助完成订单管理界面所需的服务 |
| normalOrder | 正常订单 |
| abnormalOrder | 异常订单 |

3、模块内部类的接口规范

**OrderController的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| OrderController.getHotelRoom | 语法 | public ArrayList<RoomNormVO> getHotelRoom(String hotelid) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 无 |
| OrderController.getRoomInfo | 语法 | public int[] getRoomInfo(String hotelid,Date checkIn,Date checkOut) |
| 前置条件 | 该酒店存在 |
| 后置条件 | 无 |
| OrderController.getOrder | 语法 | public OrderPO getOrder(String hotelid,String userid,Date checkIn,Date checkout,int roomNum[]) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderController.getDiscounted | 语法 | public getDiscounted(OrderPO order) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderController.add | 语法 | public void add(OrderPO order) |
| 前置条件 | 输入符合语法规则 |
| 后置条件 | 调用Order类，将新的订单添加到数据库中，持久化更新相关数据 |
| OrderController.userBrowse | 语法 | public OrderPO[] browse(String userid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderController.cancelMyOrder | 语法 | public void cancelMyOrder(String orderid) |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 订单的未执行状态改为已撤销状态，持久化更新相关数据 |
| OrderController.detail | 语法 | public OrderPO detail(String orderid) |
| 前置条件 | 已经创建一个Order领域对象，输入符合语法规范 |
| 后置条件 | 无 |
| OrderController.hotelBrowse | 语法 | public OrderPO[] hotelBrowse(String hotelid) |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 无 |
| OrderController.execute | 语法 | public ResultMessage execute(String orderid) |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 订单的未执行状态改为已执行状态，持久化更新相关数据 |
| OrderController.comment | 语法 | public ResultMessage comment(String orderid,double score,String coomment) |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 更新酒店的评分、评论信息 |
| OrderController.hotelCancelAbnormal | 语法 | public ResultMessage hotelCancelAbnormal(String orderid) |
| 前置条件 | 已经创建一个abnormalOrder领域对象，输入符合语法规范 |
| 后置条件 | 订单的异常状态改为已执行状态，  持久化更新相关数据 |
| OrderController.specificOrder | 语法 | public OrderPO[] specificOrder(String userid,String hotelid) |
| 前置条件 | 已经创建一个Order领域对象，输入符合语法规范 |
| 后置条件 | 无 |
| OrderController.leaveUpdate | 语法 | public ResultMessage leaveUpdate(String orderid) |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 更新订单的实际离开时间信息，  持久化更新相关数据 |
| OrderController.classify | 语法 | public OrderPO[] classify(OrderPO[] list,StateOfOrder state) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderController.browseUnperformed | 语法 | public OrderPO[] browseUnperformed() |
| 前置条件 | 已经创建一个normalOrder领域对象，输入符合语法规范 |
| 后置条件 | 无 |
| OrderController.browseAbnormal | 语法 | public OrderPO browseAbnormal() |
| 前置条件 | 已经创建一个abnormalOrder领域对象，输入符合语法规范 |
| 后置条件 | 无 |
| OrderController.web  CancelAbnormal | 语法 | public ResultMessage webCancelAbnormal(String orderid) |
| 前置条件 | 已经创建一个abnormalOrder领域对象，输入符合语法规范 |
| 后置条件 | 订单的异常状态改为已撤销状态，持久化更新相关数据 |
| OrderController.getOrderStateOfUser | 语法 | public StateOfOrder getOrderStateOfUser(String userid,String hotelid) |
| 前置条件 | 存在该用户在酒店下的订单 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| initialOrder.add | 新增一个订单 | |
| Order.userBrowse | 浏览用户的所有订单 | |
| normalOrder.cancelMyOrder | 用户撤销未执行订单 | |
| Order.detail | 显示订单详情 | |
| Order.hotelBrowse | 浏览酒店的所有订单 | |
| normalOrder.execute | 订单执行 | |
| normalOrder.comment | 订单评论 | |
| abnormalOrder.hotel  CancelAbnormal | 酒店手动补登记 | |
| Order.specificOrder | 具体酒店中具体用户所有的订单 | |
| normalOrder.leaveUpdate | 用户退房时间记录更新 | |
| Order.classify | 订单的分类 | |
| normalOrder.browse  Unperformed | 营销人员查看未执行订单 | |
| abnormalOrder.browseAbnormal | 营销人员查看异常订单 | |
| abnormalOrder.web  CancelAbnormal | 营销人员撤销异常订单 | |

**initialOrder的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| initialOrder.getHotelRoom | 语法 | public ArrayList<RoomNormVO> getHotelRoom() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| initialOrder.getRoomInfo | 语法 | public int[] getRoomInfo(String hotelid,Date checkIn,Date checkOut) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| initialOrder.getOrder | 语法 | public OrderPO getOrder(String hotelid,String userid,Date checkIn,Date checkout,int roomNum[]) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| initialOrder.add | 语法 | public void add(OrderPO order) |
| 前置条件 | 无 |
| 后置条件 | 根据条件新增订单的一个持久化对象或无变动 |
| initialOrder.getDiscounted | 语法 | public int getDiscounted(OrderPO order) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| initialOrder.check | 语法 | public ResultMessage check(OrderPO order) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| OrderDataService.insert | 插入单一持久化对象 | |
| Hotel.changeRoomAvail | 生成订单时，酒店的客房信息变化  订单被取消后，酒店的客房信息变化  线下退房后，酒店的客房信息变化 | |
| Hotel.getCheckInDDL() | 生成订单时，出现在订单提示信息上  判断订单是否设为异常时，得到酒店的最晚入住时间  撤销未执行订单时，判断是否扣除信用值，得到酒店的最晚入住时间 | |
| Hotel.numOfRoomAvail | 要求生成订单时，返回酒店特定房间类型在一定时期内的可用客房数量 | |
| Hotel.getRoomNorms | 要求生成订单时，返回酒店的所有房间类型 | |
| Promotion.countPromotionOfOrder | 生成订单时，计算总价的优惠后价格 | |
| Promotion.countPromotionOfRoom | 生成订单时，计算特殊房间优惠后的价格 | |

**Order的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Order.userBrowse | 语法 | public OrderPO[] browse(String userid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.detail | 语法 | public OrderPO detail(String orderid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.hotelBrowse | 语法 | public OrderPO[] hotelBrowse(String hotelid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.specificOrder | 语法 | public OrderPO[] specificOrder(String userid,String hotelid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.classify | 语法 | public OrderPO[] classify(OrderPO[] list,StateOfOrder state) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.getOrderStateOfUser | 语法 | public stateOfOrder getOrderStateOfUser(String userid, String hotelid) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| **服务名** | **服务** | |
| orderdao.find | 根据订单ID查找单一持久化对象 | |
| orderdao.userFind | 根据用户ID查找多个持久化对象 | |
| orderdao.hotelFind | 根据酒店ID查找多个持久化对象 | |
| orderdao.stateFind | 根据订单状态查找多个持久化对象 | |
| orderdao.delete | 删除单一持久化对象 | |
| orderdao.init | 初始化持久化数据库 | |
| orderdao.finish | 结束持久化数据库的使用 | |

**normalOrder的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务(供接口)** | | |
| normalOrder.cancel  MyOrder | 语法 | public void cancelMyOrder(String orderid) |
| 前置条件 | 无 |
| 后置条件 | 订单的未执行状态改为已撤销状态 |
| normalOrder.execute | 语法 | public ResultMessage execute(String orderid) |
| 前置条件 | 用户线下入住 |
| 后置条件 | 订单的未执行状态改为已执行状态 |
| normalOrder.comment | 语法 | public ResultMessage comment(String orderid，String coomment) |
| 前置条件 | 订单为已执行状态 |
| 后置条件 | 更新酒店的评分、评论信息 |
| normalOrder.setAbnormal | 语法 | public void setAbnormal(String orderid) |
| 前置条件 | 时间离最晚执行时间小于6h |
| 后置条件 | 订单的未执行状态改为异常状态 |
| normalOrder.leaveUpdate | 语法 | public ResultMessage leaveUpdate(String orderid) |
| 前置条件 | 用户退房 |
| 后置条件 | 更新订单的实际离开时间信息 |
| normalOrder.browse  Unperformed | 语法 | public OrderPO[] browseUnperformed() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| orderdao.find | 根据订单ID查找单一持久化对象 | |
| orderdao.stateFind | 根据订单状态查找多个持久化对象 | |
| orderdao.stateUpdate | 更新单一持久化对象 | |
| orderdao.commentUpdate | 更新单一持久化对象 | |
| orderdao.leaveUpdate | 更新单一持久化对象 | |
| Hotel.updateGrade | 订单评分，酒店相关信息修改 | |
| Hotel.addComment | 订单评论，酒店相关信息修改 | |
| Hotel.changeRoomAvail | 生成订单时，酒店的客房信息变化  撤销订单后，酒店的客房信息变化  线下退房后，酒店的客房信息变化 | |
| User.addCreditRecord | 订单执行时，信用值添加  异常订单被酒店撤销时，信用值增加  未执行订单被用户撤销时，满足条件，信用值减少  异常订单被网站营销人员撤销时，信用值增加 | |
| Hotel.getCheckInDDL() | 判断订单是否设为异常时，得到酒店的最晚入住时间  撤销未执行订单时，判断是否扣除信用值，得到酒店的最晚入住时间 | |

**abnormalOrder的接口规范**

|  |  |  |
| --- | --- | --- |
| abnormalOrder.hotel  CancelAbnormal | 语法 | public ResultMessage hotelCancelAbnormal(String orderid) |
| 前置条件 | 订单状态为异常 |
| 后置条件 | 订单的异常状态改为已执行状态 |
| abnormalOrder.browseAbnormal | 语法 | public OrderPO browseAbnormal() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| abnormalOrder.handleAbnormalRoom | 语法 | public void handleAbnormalRoom() |
| 前置条件 | 无 |
| 后置条件 | 改变可用客房信息 |
| abnormalOrder.web CancelAbnormal | 语法 | public ResultMessage webCancelAbnormal(String orderid) |
| 前置条件 | 订单状态为异常 |
| 后置条件 | 订单的异常状态改为已撤销状态 |
| **需要的服务（需接口）** | | |
| orderdao.find | 根据订单ID查找单一持久化对象 | |
| orderdao.stateFind | 根据订单状态查找多个持久化对象 | |
| orderdao.stateUpdate | 更新单一持久化对象 | |
| User.addCreditRecord | 订单执行时，信用值添加  异常订单被酒店撤销时，信用值增加  未执行订单被用户撤销时，满足条件，信用值减少  异常订单被网站营销人员撤销时，信用值增加 | |
| Hotel.changeRoomAvail | 生成订单时，酒店的客房信息变化  订单被取消后，酒店的客房信息变化  线下退房后，酒店的客房信息变化 | |

4、业务逻辑层的动态模型

在Orderservice模块中，业务分为

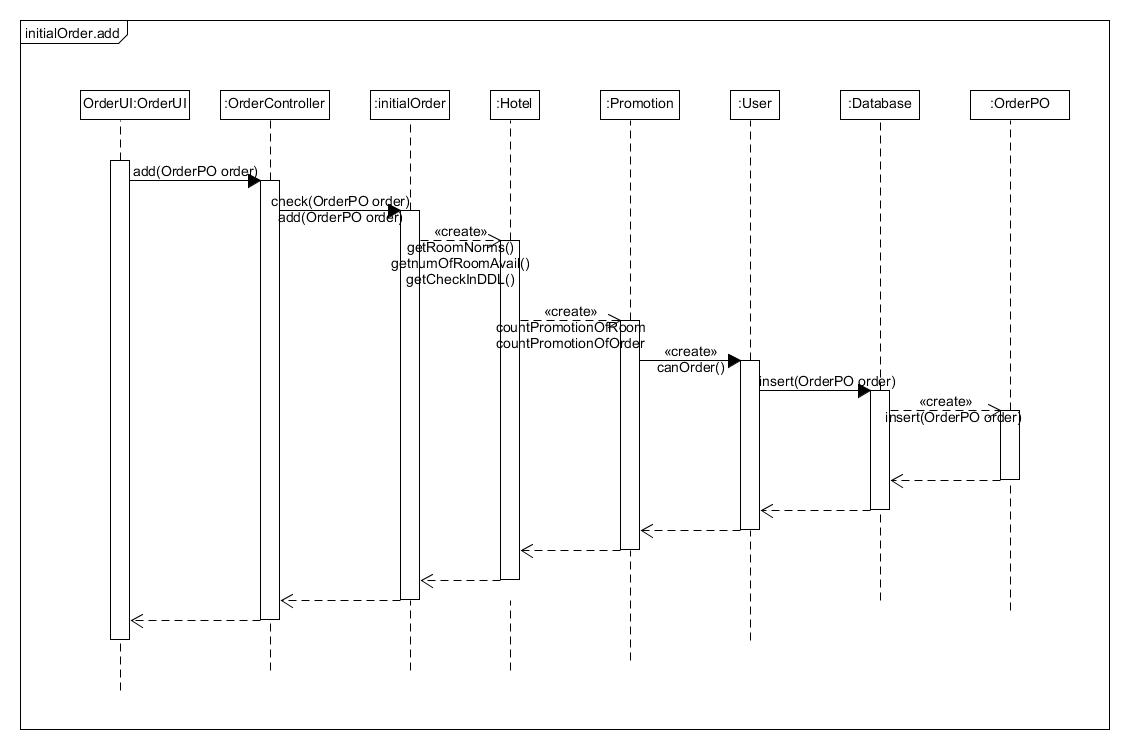
initialOrder：增加订单；

Order：浏览订单、订单状态分类、订单详情展示；

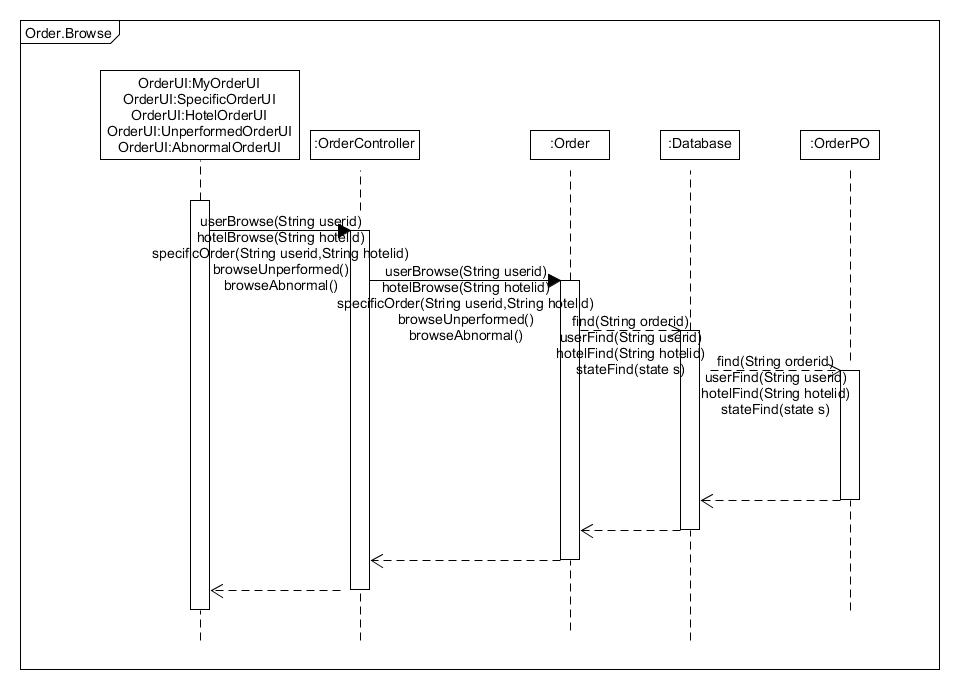
abnormalOrder：酒店手动补登记、网站营销人员撤销异常；

normalOrder：撤销我的订单、评论订单、执行订单、登记实际离开时间。

其顺序图如下列图所示：



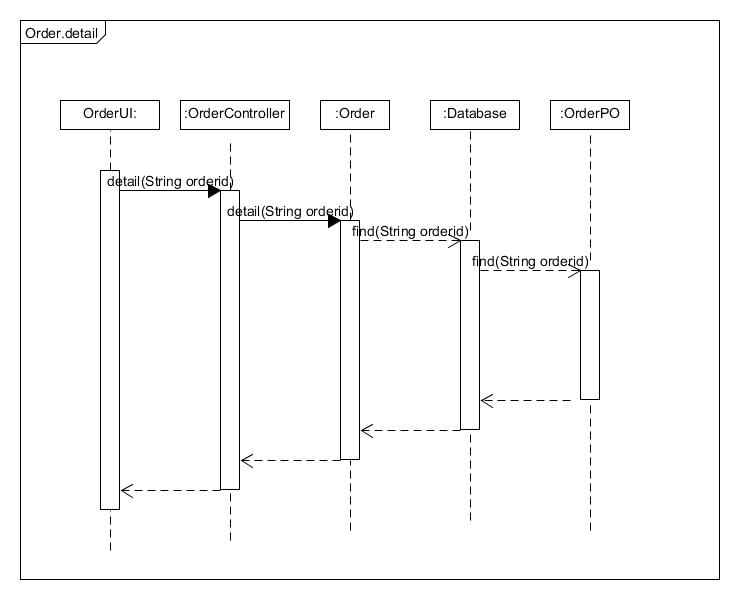
增加订单



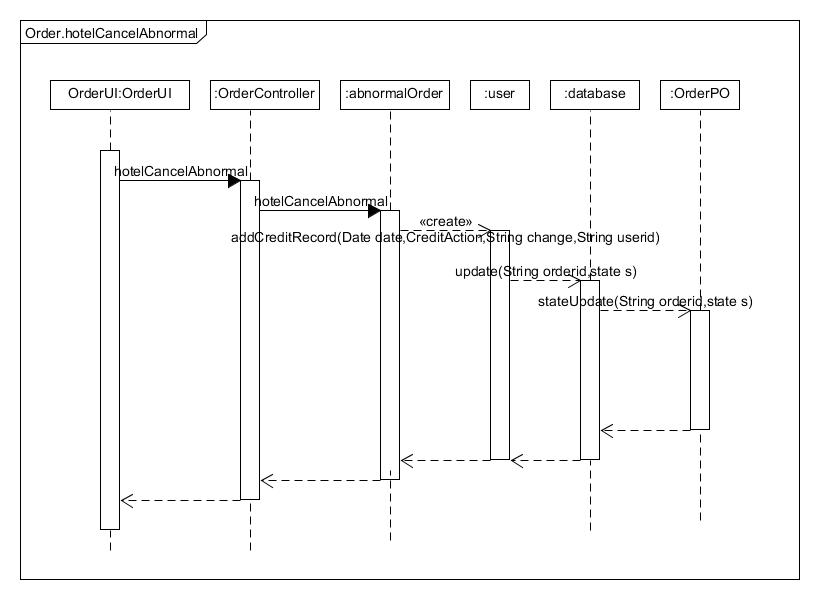
浏览订单



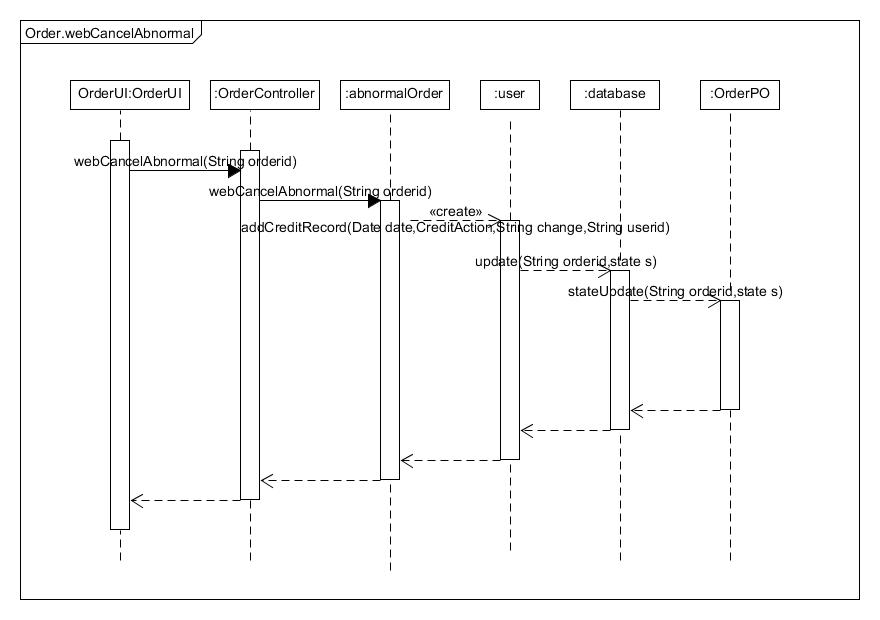
订单状态分类



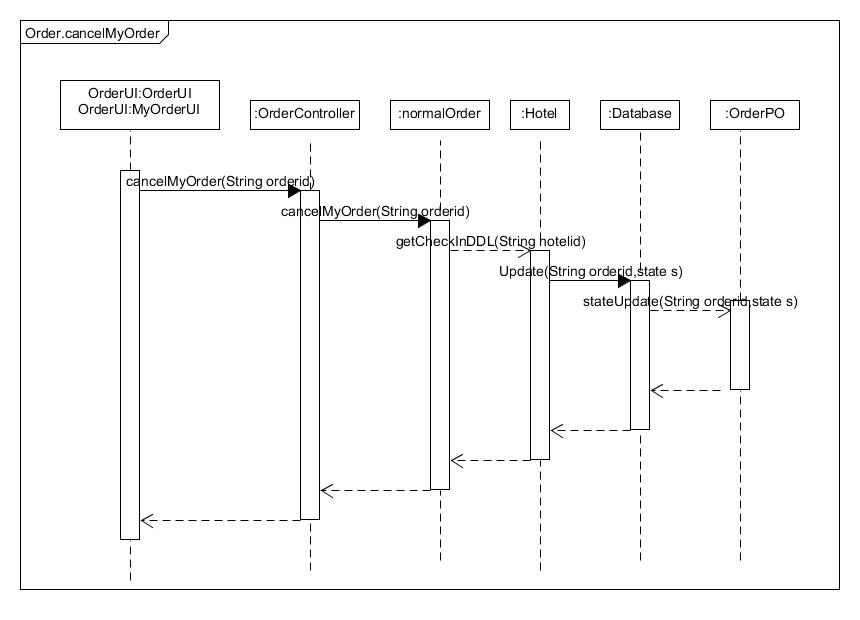
订单详情展示



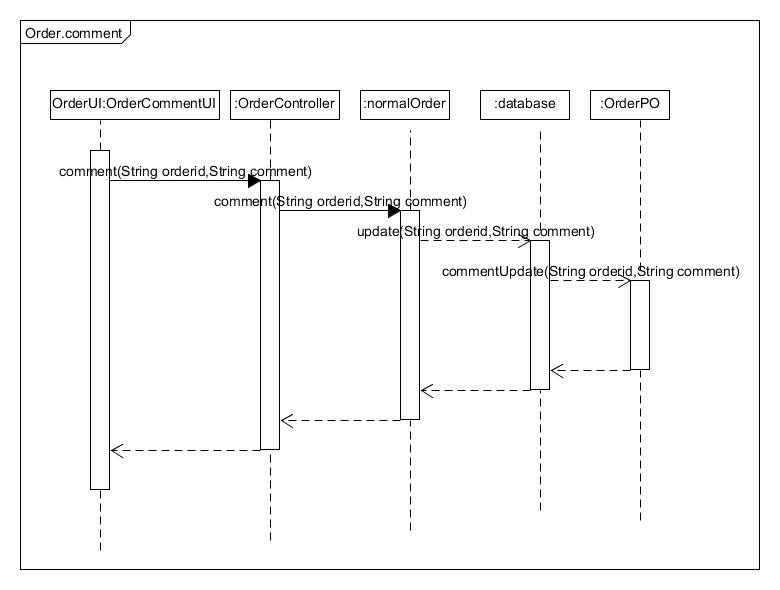
酒店手动补登记



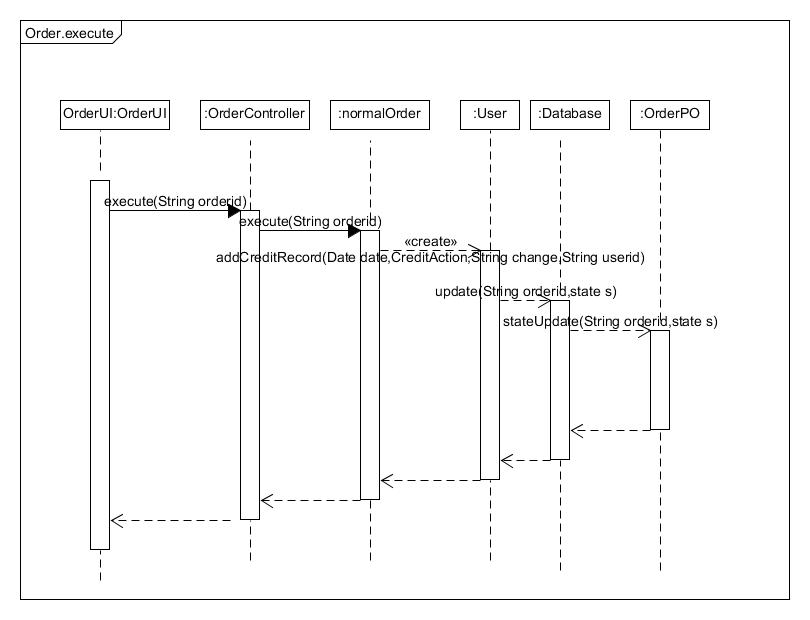
网站营销人员撤销异常



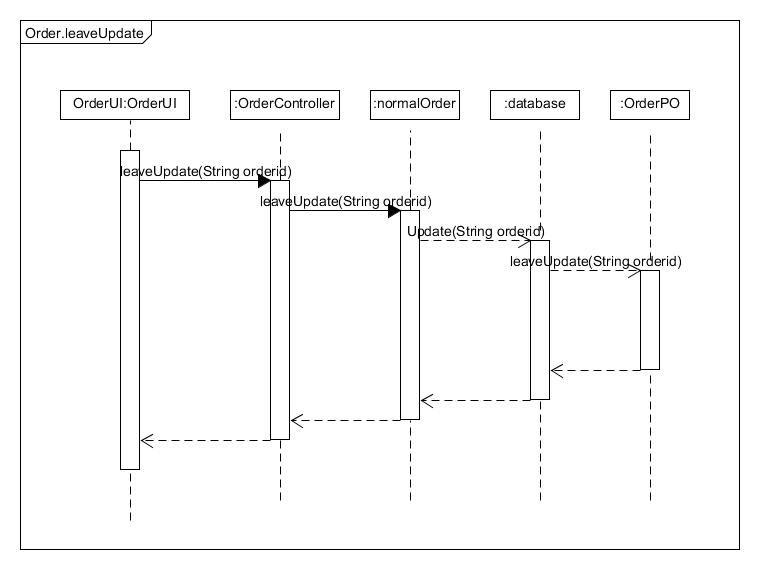
撤销我的订单



评论订单

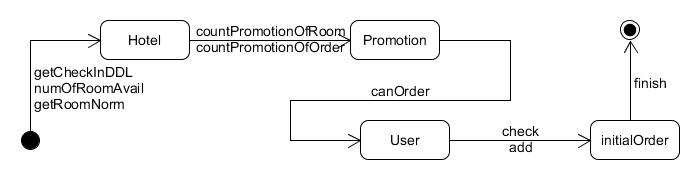


执行订单

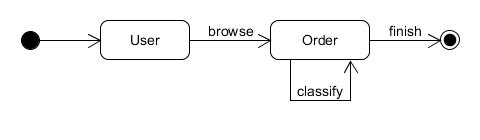


登记实际离开时间

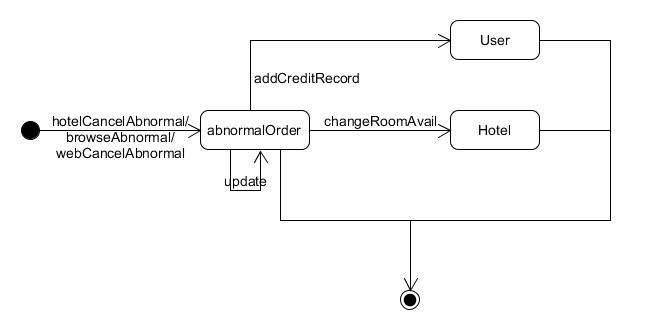
下列图分别描述了initialOrder对象、Order对象、abnormalOrder对象、normalOrder对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。



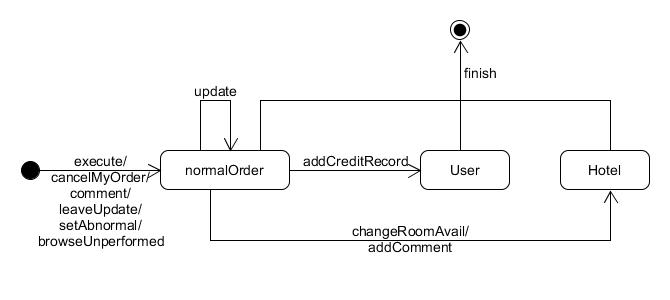
initialOrder



Order



abnormalOrder



normalOrder

5、业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

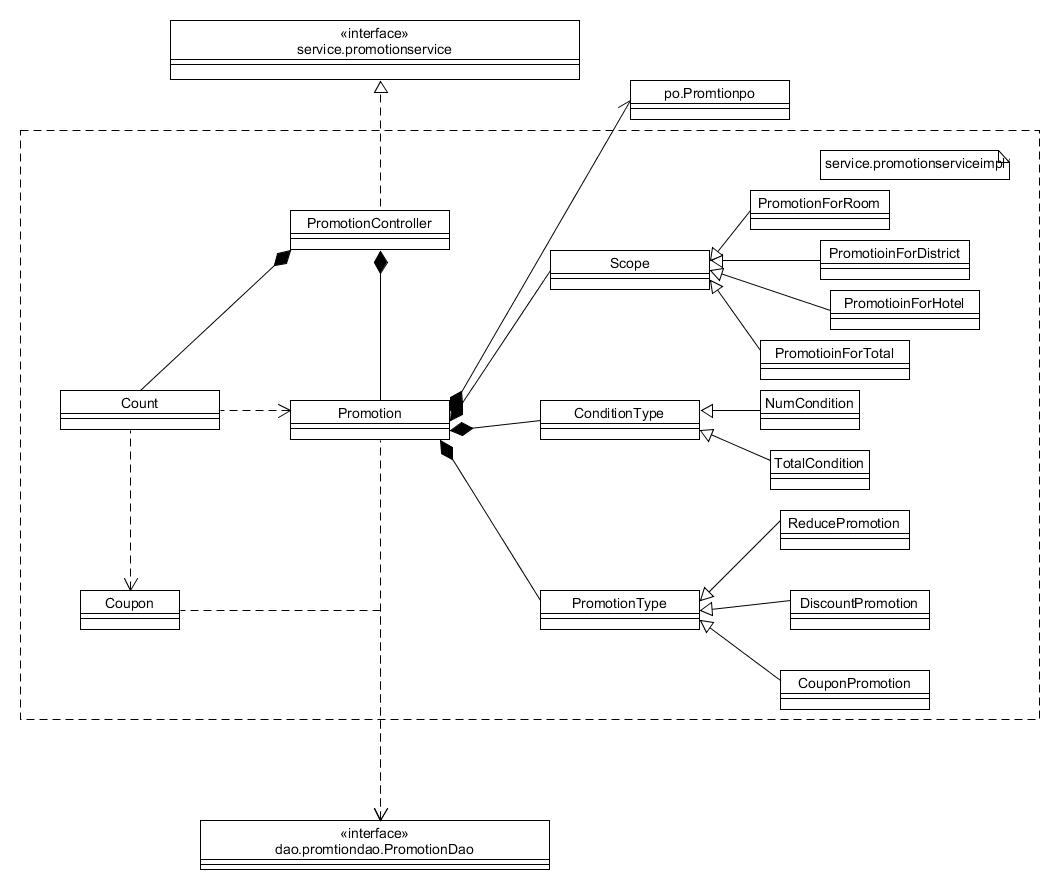
4.1.5Promotion模块

1. 模块概述

2、整体结构

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。在展示层和业务逻辑层之间添加PromotionblService接口。在业务逻辑层和数据层之间添加Promotiondao接口。为了隔离业务逻辑职责和逻辑控制职责，添加PromotionController，这样PromotionController会将对促销策略的逻辑委托给Promotion对象。而PromotionPO是作为促销策略的持久化对象被添加到设计模型中去的。

Promotionbl模块的设计如图



Promotionbl模块各个类的职责如表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 负责实现促销策略所需要的服务 |
| 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会员优惠上的具体实现 |
| PromotionType | 促销方式的领域模型具体对象 |
| ReducePromotion | 继承PromotionType类，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 |
| CoupoPromotion | 继承PromotionType类，是促销方式在赠送卡券策略上的具体实现 |
| Coupon | 卡券的领域模型对象，拥有一张卡券的制定者，优惠方式，适用范围，适用日期，卡券描述等信息，可以帮助完成促销界面所需要的服务 |
| Count | 负责计算使用促销策略后的价格并选择最优 |
| Show | 帮助完成对已有促销策略的查看 |

3、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.setBeginDate | 语法 | public ResultMessage setBiginDate (Date beginDate) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setBeginDate方法 |
| PromotionController.setEndDate | 语法 | public ResultMessage setEndDate (Date EndDate) |
| 前置条件 | 已创建一个promotion对象 |
| 后置条件 | 调用promotion领域的setEndDate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 ResultMessage derPromotion |
| 前置条件 | 已存在promotion对象 |
| 后置条件 | 调用promotion领域delPromotion方法 |
| PromotionController.getPromotionOfPeriod | 语法 | Public ArrayList <PromotionVo> getPromotionOfPeriod (Date beginDate, Date EndDate) |
| 前置条件 | 无 |
| 后置条件 | 调用show领域的getPromotionOfPeriod方法 |
| PromotionController.getPromotionOfRoom | 语法 | public ArrayList <PromotionVo> getPromotionOfRoom (String hotelID, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 调用show领域的getPromotionOfRoom方法 |
| PromotionController.getPromotionOfHotel | 语法 | public ArrayList <PromotionVo> getPromotionOfHotel (String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用show领域的getPromotionOfHotel方法 |
| PromotionController.getPromotionOfDistrice | 语法 | Public 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 setBiginDate (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 |
| 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PromotioDao的delPromotion方法 |
| 需要的服务（虚接口） | | |
| 服务名 | 服务 | |
| PromotionDao.insertPromotion(PromotionPo po) | 插入促销策略单一持久化对象 | |
| PromotionDao.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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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的接口规范**

|  |  |  |
| --- | --- | --- |
| 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**

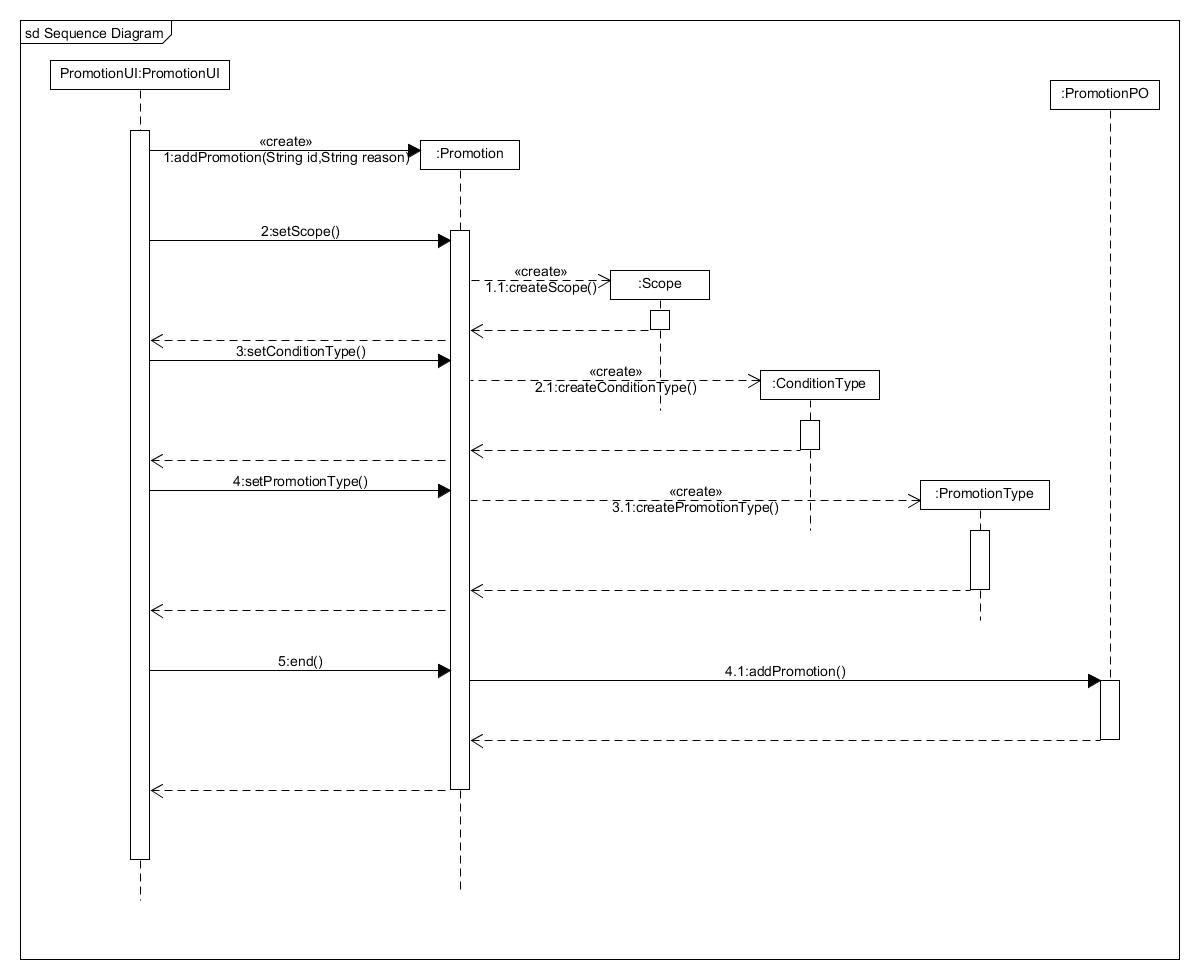
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Show.getPromotionOfPeriod | 语法 | Public static ArrayList <Promotion> getPromotionOfPeriod(Date beginDate, Date EndDate) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionDao的finds方法 |
| Show.getPromotionOfRoom | 语法 | public stattic ArrayList <Promotion> getPromotionOfRoom (String hotelID, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 调用PromotionDao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PromotionDao的finds方法 |
| 需要的服务（需接口） | | |
| PromotionDao.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**

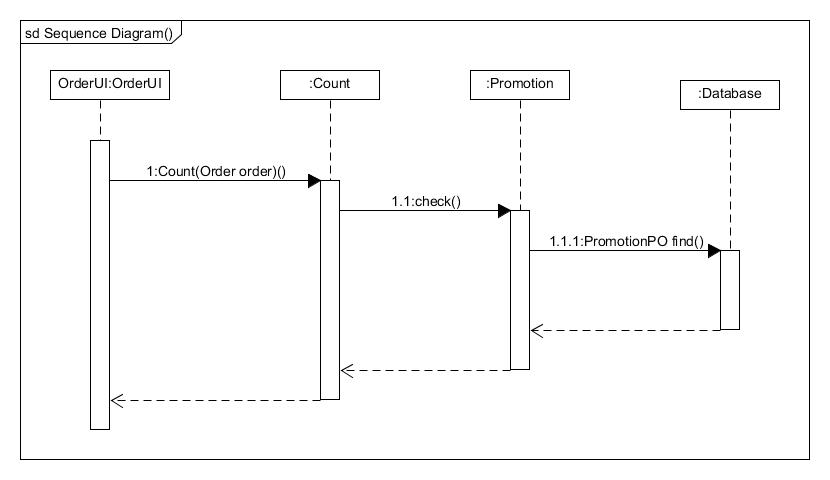
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口) | | |
| Count.countPromtionOfOrder | 语法 | public String countPromotionOfOrder(Order order) |
| 前置条件 | 已存在order的领域对象 |
| 后置条件 | 返回最优方案 |
| Count.countPromotionOfRoom | 语法 | public StringcountPromotionOfRoom(String hotelID, RoomType type,int num) |
| 前置条件 | 无 |
| 后置条件 | 返回最优方案 |
| 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 | |

4、业务逻辑层的动态模型

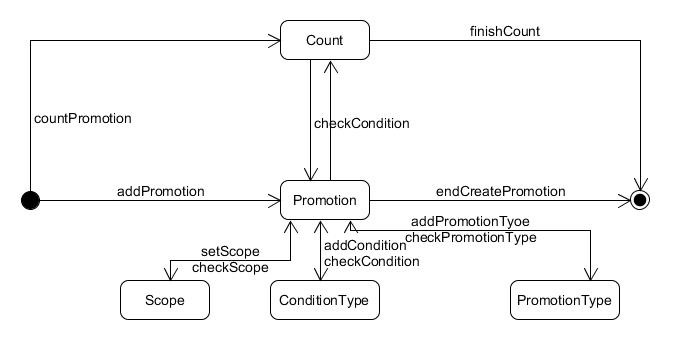
添加促销策略的顺序图



计算优惠策略的顺序图



Promotion对象状态图



4、业务逻辑层的设计原理

利用委托是风格，每个界面需要访问的业务逻辑由各自的委托控制器委托给不同的领域

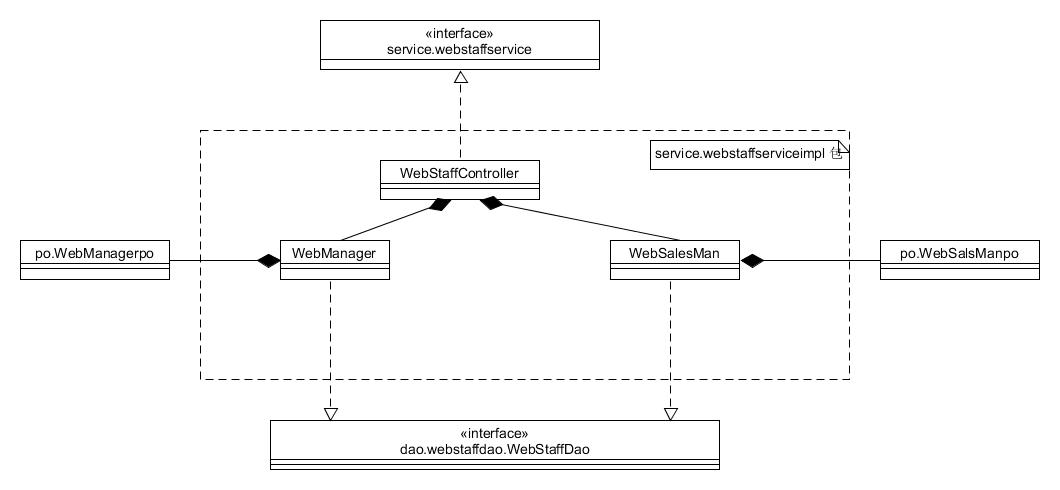
4.1.6WebStaff模块

1、模块概述

2、整体结构

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。在展示层和业务逻辑层之间添加WebStaffblService接口。在业务逻辑层和数据层之间添加WebStaffdao接口。为了隔离业务逻辑职责和逻辑控制职责，添加WebStaffController，这样WebStaffController会将对网站管理人员信息处理的逻辑委托给WebManager和WebSalseMan对象。而WebManager，WebSalseMan是作为用户的持久化对象被添加到设计模型中去的。

WebStaff模块设计图：



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 |
| SalesMan | 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 |
| Manager | 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 |

3、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**

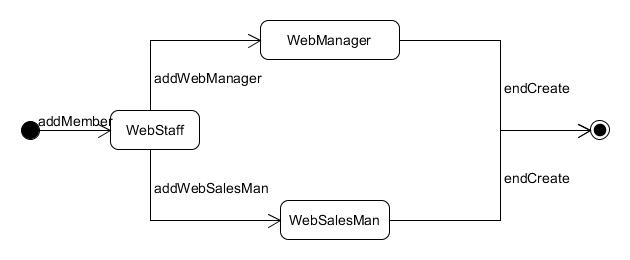
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SalesMan.addSalesMan | 语法 | public ResultMessage addSalesMan(String id , String password) |
| 前置条件 | 没有相同id的持久化数据对象 |
| 后置条件 | 创建一个SalesMan领域对象，持久化对象 |
| SalesMan.delSalesMan | 语法 | public ResultMessage delSalesMan(String id) |
| 前置条件 | 已存在相应的持久化数据对象 |
| 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） | | |
| WebStaffDao.insert(String id, String password) | 插入持久化对象 | |
| WebStaffDao.delete(String id) | 删除持久化对象 | |
| WebStaffDao.update(String id, String password) | 更新持久化对象 | |
| WebStaffDao.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 |
| 需要的服务（需接口） | | |
| WebStaffDao.insert | 插入一个持久化对象 | |
| WebStaddDao.update | 更新持久化对象 | |

4、业务逻辑层的动态模型

WebStaff状态图：

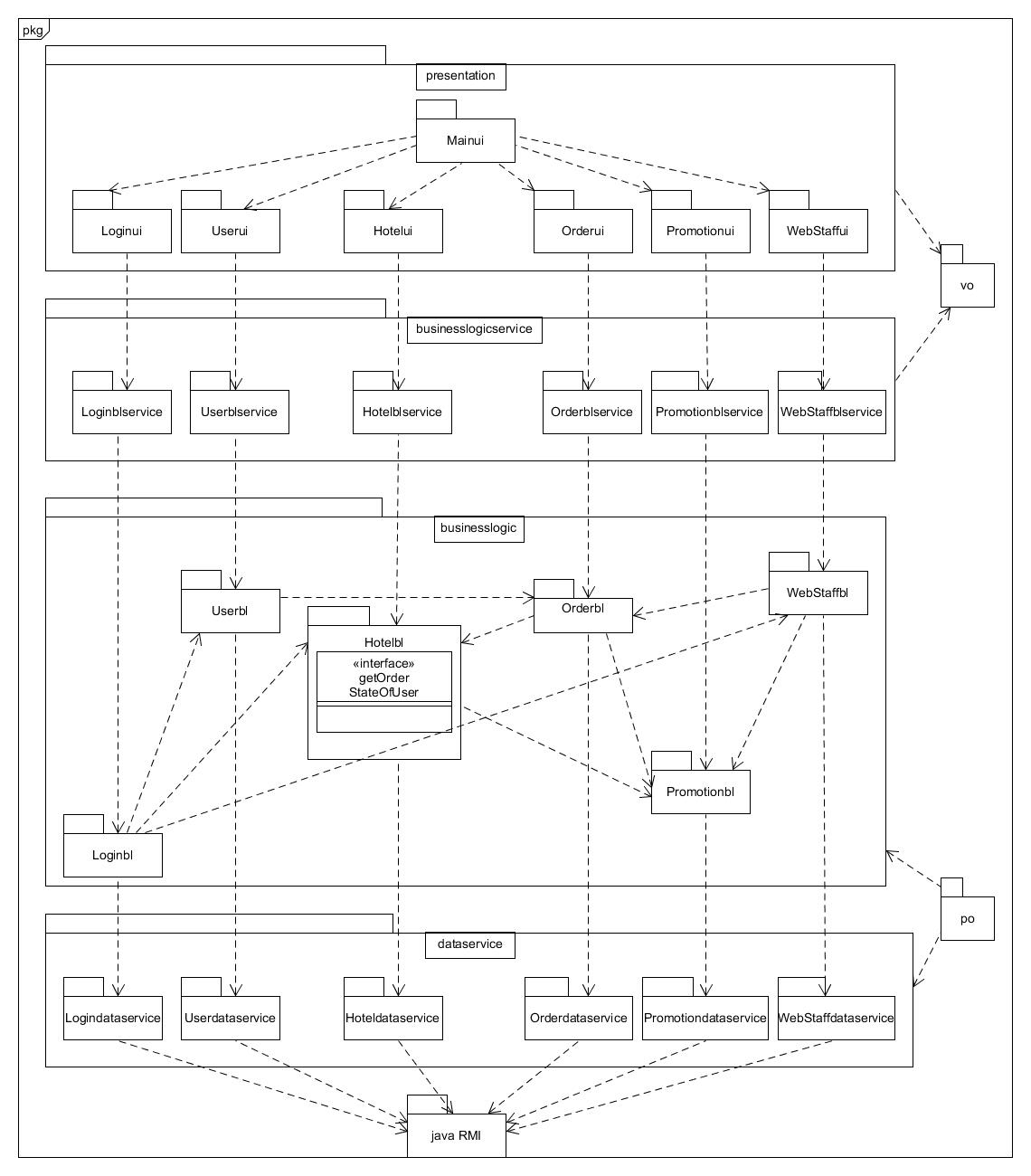


5、业务逻辑层的设计原理

利用委托是风格，每个界面需要访问的业务逻辑由各自的委托控制器委托给不同的领域

# 5、依赖视角

客户端包图：



服务器端包图：

