酒店预订管理系统

System of Booking Hotel

详细设计文档

V 2.1

小组37 陈统 刚昭 高岳 贺云青

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 全体组员 | 2016-10-29 | 建立整体框架，完成部分文档 | V0.1 |
| 贺云青 | 2016-11-3 | 为Order逻辑层添加getHistoryHotels的供接口，为Hotel的逻辑层添加对应需接口，修正部分格式不统一问题，更新Hotel模块，调整Orderbl模块的职责分配 | V0.2 |
| 陈统 | 2016-11-5 | 更改Order模块中OrderController的职责分配。  更改Order模块中Order OrderUtil OrderList的部分接口。 | V0.3 |
| 贺云青 | 2016-11-5 | 更新Hotel模块内容，调整hotelbl模块的格式，为Promotion模块添加计算价格的方法，修改了Promotion模块部分接口返回值 | V0.4 |
| 陈统 | 2016-11-6 | 更改Order模块中OrderController的职责分配。  更改Order模块中OrderUtil的接口。  删去Order模块中Order与OrderList类的部分接口。 | V0.5 |
| 贺云青 | 2016-11-6 | 细化展示层的分解，修改少部分展示层类的职责分配  重制Userbl模块的设计图，添加CreditRecordLis类  重新分配Userbl模块的职责  更新依赖视角客户端开发包图  优化排版，调整对齐方式 | V1.0 |
| 贺云青 | 2016-11-23 | Promotion模块添加计算某用户等级的方法接口 | V1.1 |
| 贺云青 | 2016-12-2 | Promotion模块添加ID和delete方法 | V1.2 |
| 刚昭、贺云青 | 2016-12-24 | 全面修改文档hotel、promotion、comment模块一致性 | V2.0 |
| 陈统、高岳 | 2016-12-31 | 全面修改文档User、Order模块一致性 | V2.1 |

**目录**

[1 引言 3](#_Toc465525992)

[1.1 编制目的 3](#_Toc465525993)

[1.2 词汇表 3](#_Toc465525994)

[1.3 参考资料 3](#_Toc465525995)

[2 产品概述 3](#_Toc465525996)

[3 体系结构设计概述 3](#_Toc465525997)

[4 结构视角 4](#_Toc465525998)

[4.1 展示层的分解 4](#_Toc465525999)

[4.1.1 UI层设计概要 4](#_Toc465526000)

[4.1.2 UI层内部包的职责与划分 4](#_Toc465526001)

[4.2 业务逻辑层的分解 5](#_Toc465526002)

[4.2.1 Userbl模块 6](#_Toc465526003)

[4.2.2 order模块 15](#_Toc465526004)

[4.2.3 Hotelbl模块 24](#_Toc465526005)

[4.2.4 Promotion模块 29](#_Toc465526008)

[4.2.5 Comment模块 35](#_Toc465526009)

[4.3 数据层的分解 37](#_Toc465526010)

[4.3.1 UserDataService模块 37](#_Toc465526011)

[4.3.2 OrderDataService模块 38](#_Toc465526012)

[4.3.3 HotelDataService模块 39](#_Toc465526013)

[4.3.4 PromotionDataService模块 41](#_Toc465526014)

[4.3.5 CommentDataService模块 42](#_Toc465526015)

[5 依赖视角 42](#_Toc465526016)

# 引言

## 编制目的

本报告详细完成对酒店在线预订系统的详细设计，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。

## 词汇表

|  |  |
| --- | --- |
| SBH | 酒店预订管理系统 |
| SBH | 酒店预订管理系统 |
| user | 用户 |
| hotel | 酒店 |
| order | 订单 |
| promotion | 促销策略 |
| comment | 评价 |
| level | 用户等级 |
| HotelFilter | 酒店筛选信息 |

## 参考资料

《酒店在线预订系统用例文档V1.8》

《酒店在线预订系统需求规格说明文档V0.94》

《酒店在线预订系统体系结构设计文档V1.7》

# 产品概述

参考酒店在线预订系统用例文档及酒店在线预订系统规格说明文档对产品的概括描述。

# 体系结构设计概述

参考酒店在线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。

# 结构视角

## 展示层的分解

### UI层设计概要

UI层主要以人员的性质来分，包括客户、酒店工作人员、网站管理人员、网站营销人员。

Ui层中基本是每一种订单对应两个界面：浏览和修改，虽然两种界面风格差不多，但是还有略有区别。其余的基本是一个功能一个类。

Ui层的入口是mainframe，显示的是打开系统的login界面，然后通过不同身份的人的账户登录，跳转到各个职位所对应的ui包内，之后职位所对应的各个功能的界面可以最大程度在包内跳转，更好的实现高内聚、低耦合。

### UI层内部包的职责与划分

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 包名 | 类名 | 功能 |
| Common | Userui | Register | 客户注册 |
| Login | 登陆 |
| Customer  客户 | Userui | CustomerCheckInfo | 查看个人信息 |
| CustomerModifyInfo | 修改个人信息 |
| CustomerCheckCredit | 查看信用记录 |
| Hotelui | CustomerCheckHotelList | 浏览酒店 |
| CustomerCheckHotel | 酒店详细信息 |
| Orderui | CustomerGenerateOrder | 生成订单 |
| CustomerConfirmOrder | 生成订单确认 |
| CustomerUnexecutedOrder | 浏览客户未执行订单 |
| CustomerUnexecutedOrder | 未执行订单详细信息 |
| CustomerHistoryOrder | 浏览客户历史订单 |
| CustomerExecutedOrder | 已执行订单详细信息 |
| CustomerCancelledOrder | 已撤销订单详细信息 |
| CustomerAbnormalOrder | 异常订单详细信息 |
| Commentui | CustomerComment | 评价订单 |
| HotelManager  酒店工作人员 | Hotelui | HotelManagerModifyHotel | 维护酒店信息 |
| Promotionui | HotelManagerCheckPromotion | 管理酒店促销策略 |
| Orderui | HotelManagerModifyPromotion | 修改酒店促销策略 |
| HotelManagerUnexecuteOrderList | 浏览未执行订单 |
| HotelManagerUnexecuteOrder | 未执行订单详细信息 |
| HotelManagerHistoryOrder | 浏览酒店历史订单 |
| HotelManagerExecutedOrder | 已执行订单详细信息 |
| HotelManagerCancelledOrder | 已撤销订单详细信息 |
| WebMarketer  网站营销人员 | Userui | WebMarketerAddCredit | 信用充值 |
| Promotionui | WebMarketerModifyLevel | 修改等级制度 |
| WebMarketerCheckPromotion | 管理网站促销策略 |
| WebMarketerModifyPromotion | 修改网站促销策略 |
| Orderui | WebMarketerAbnormalOrderList | 浏览异常订单 |
| WebMarketerAbnormalOrder | 异常订单详细信息 |
| WebManager  网站管理人员 | Userui | WebManagerCheckCustomer | 浏览客户信息 |
| WebManagerModifyCustomer | 修改客户信息 |
| WebManagerCheckWebMarketer | 浏览网站营销人员信息 |
| WebManagerModifyWebMarketer | 修改网站营销人员信息 |
| WebManagerAddWebMarketer | 增加网站营销人员 |
| WebManagerCheckHotelManager | 浏览酒店工作人员信息 |
| WebManagerModifyHoelrManager | 修改酒店工作人员信息 |
| Hotelui | WebManagerAddHotel | 增加酒店 |

## 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。

### Userbl模块

1、模块概述：

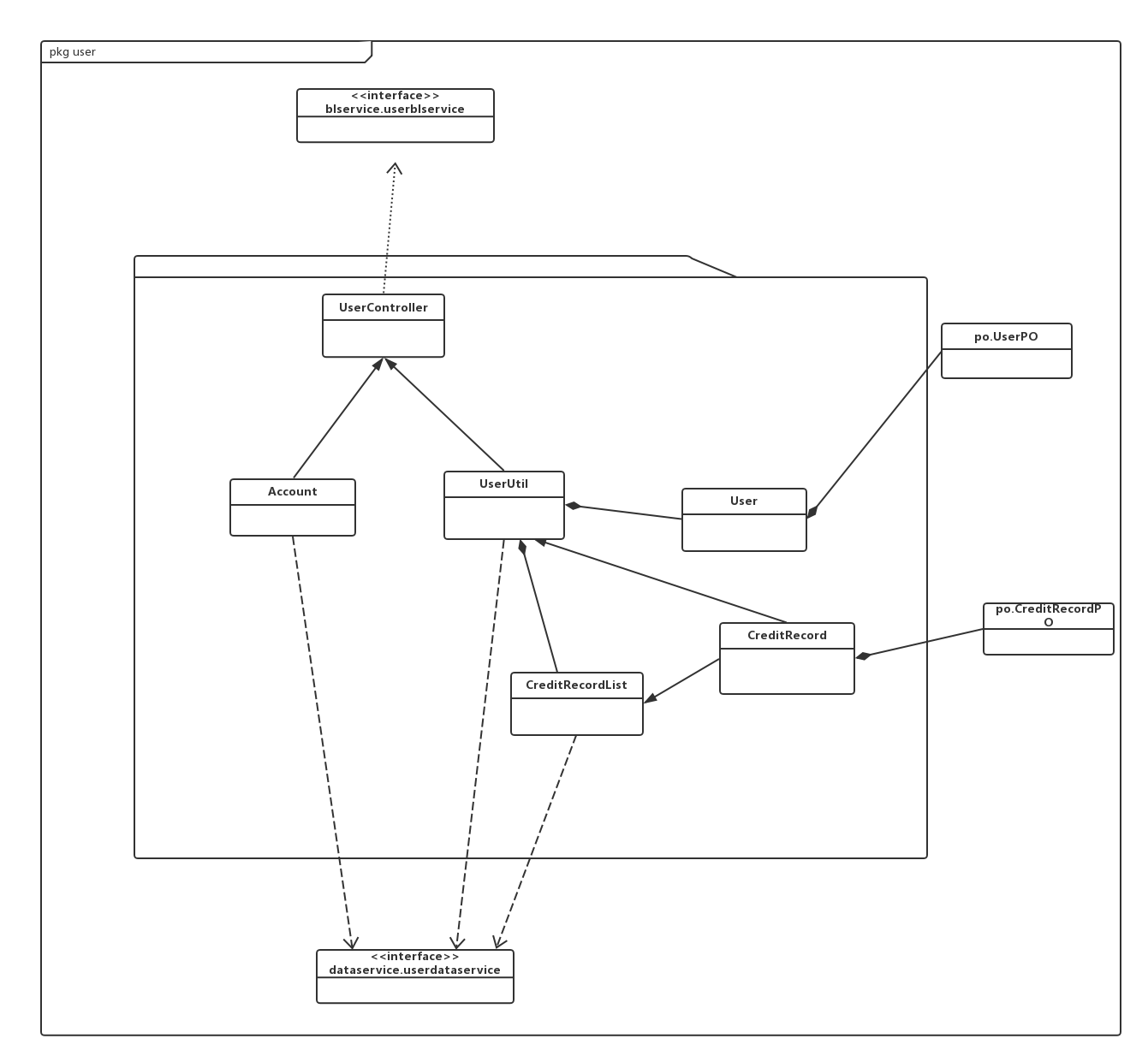
User模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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。

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UserblService接口。在业务逻辑层和数据层之间添加UserDataService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会将对用户管理的业务逻辑处理委托给UserUtil对象。UserPO和CreditRecordPO是作为账户信息的持久性对象与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模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserController | 负责对应于管理员账户管理所需要的服务 |
| UserUtil | 工具类，可以帮助完成账户管理界面所需要的服务 |
| User | 领域对象，拥有user的各项信息，帮助生成po与解包vo |
| Account | 工具类，负责登入登出相关的功能 |
| CreditRecord | 领域对象，拥有creditrecord的各项信息，帮助生成po与解包vo |
| CreditRecordlist | 负责生成和维护领域对象列表 |

3、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.login | 语法 | public ResultMessage login(String userName, String password) |
| 前置条件 | 无 |
| 后置条件 | 调用Account类，系统确认账号密码是否正确，若正确，进入主界面并记录登录信息，否则提示错误信息 |
| UserController.logout | 语法 | public ResultMessage logout(String ID) |
| 前置条件 | 系统已登录 |
| 后置条件 | 调用Account类，系统退出登录，记录登出信息 |
| UserController.add | 语法 | public ResultMessage add(UserVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 调用UserUtil类，系统创建该账户并持久化增加该账户数据 |
| UserController.findbyID | 语法 | public UserVO findbyID(String ID) |
| 前置条件 | 账户ID符合输入语法要求 |
| 后置条件 | 调用UserUtil类，系统返回查找到的搜索结果 |
| UserController.findbyUserName | 语法 | public UserVO findbyUserName(String userName) |
| 前置条件 | 账户名符合输入语法要求 |
| 后置条件 | 调用User类，系统返回查找到的搜索结果 |
| UserController.modifyPassword | 语法 | public ResultMessage modifyPassword(String userName, String oldPassword, String newPassword) |
| 前置条件 | 新密码符合输入语法要求，旧密码匹配正确 |
| 后置条件 | 调用User类，系统修改该账户的密码数据 |
| UserController.modify | 语法 | public ResultMessage modify(UserVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 调用User类，系统修改该账户并持久化保存该账户数据 |
| UserController.addCreditRecord | 语法 | public ResultMessage addCreditRecord (CreditRecordVO vo) |
| 前置条件 | 该账户ID为客户 |
| 后置条件 | 调用UserUtil类，系统增加该客户信用记录并持久化保存数据 |
| UserController.findCreditRecord | 语法 | public List<CreditRecordVO> findCreditRecord(String ID) |
| 前置条件 | 该账户ID为客户且输入符合语法要求 |
| 后置条件 | 调用CreditRecordList类，返回该账户所有信用 |
| UserController.getLevel | 语法 | public UserLevelVO getLevel(String userName) |
| 前置条件 | 该用户名正确且存在 |
| 后置条件 | 无 |
| UserController.deleteUser | 语法 | public ResultMessage deleteUser(String userName) |
| 前置条件 | 该用户名正确且存在 |
| 后置条件 | 调用UserUtil类，系统删除该用户信息 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Account.Login(String userName, String password) | 用户登录 | |
| Account.Logout(String ID) | 用户登出 | |
| UserUtil.add(UserVO vo) | 增加用户 | |
| UserUtil.findbyID(String ID) | 根据用户ID查找用户 | |
| UserUtil.findbyUserName(String userName) | 根据UserName查找用户 | |
| UserUtil.modifyPassword(String userName, String oldPassword, String newPassword) | 修改用户密码 | |
| UserUtil.modify(UserVO vo) | 修改用户信息 | |
| UserUtil.addCreditRecord  (CreditRecordPO po) | 增加一条信用记录 | |
| UserUtil.findCreditRecord  (String ID) | 找到信用记录 | |
| UserUtil.getLevel(String userName) | 获得用户等级 | |
| UserUtil.deleteUser(String UserName) | 根据用户名删除相应用户 | |

**User的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.modifyPassword | 语法 | public UserPO modifyPassword(String newPassword) |
| 前置条件 | 新密码符合输入语法要求，旧密码匹配正确 |
| 后置条件 | 返回修改好的po |
| User.modify | 语法 | public UserPO modify(UserVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 返回修改好的po |
| User.getUserPO | 语法 | public UserPo getUserPO() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| User.getUserVO | 语法 | public UserVO getUserVO() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| User.initbyPO | 语法 | public User initbyPO(UserPO po) |
| 前置条件 | Po信息完整且zhengque |
| 后置条件 | 存储po的信息 |
| User.initbyVO | 语法 | public User initbyVO(UserVO vo) |
| 前置条件 | Vo信息完整且正确 |
| 后置条件 | 存储vo的信息 |
| User.changeCredit | 语法 | public UserPO changeCredit(double change) |
| 前置条件 | 无 |
| 后置条件 | 无 |

**UserUtil的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserUtil.add | 语法 | public ResultMessage add(UserVO vo) |
| 前置条件 | 账户信息符合规则且齐全 |
| 后置条件 | 系统创建该账户并持久化增加该账户数据 |
| UserUtil.findbyID | 语法 | public UserVO findbyID(String ID) |
| 前置条件 | 账户ID符合输入语法要求 |
| 后置条件 | 系统返回查找到的搜索结果 |
| UserUtil.findbyUserName | 语法 | public UserVO findbyUserName(String userName) |
| 前置条件 | 账户名符合输入语法要求 |
| 后置条件 | 系统返回查找到的搜索结果 |
| UserUtil.addCreditRecord | 语法 | public ResultMessage addCreditRecord (CreditRecordVO vo) |
| 前置条件 | 该账户ID为客户 |
| 后置条件 | 系统增加该客户信用记录并持久化保存数据 |
| UserUtil.findCreditRecord | 语法 | public List<CreditRecordVO> findCreditRecord(String ID) |
| 前置条件 | 该账户ID为客户且输入符合语法要求 |
| 后置条件 | 返回该账户所有信用 |
| UserUtil.modifyPassword | 语法 | public ResultMessage modifyPassword(String userName, String oldPassword, String newPassword) |
| 前置条件 | 无 |
| 后置条件 | 系统将该用户的密码改成新的密码 |
| UserUtil.modify | 语法 | public ResultMessage modify(UserVO vo) |
| 前置条件 | Vo完整且信息正确 |
| 后置条件 | 系统修改存储在数据库的user信息 |
| UserUtil.getLevel | 语法 | public UserLevelVO getLevel(String userName) |
| 前置条件 | 该用户名正确且存在 |
| 后置条件 | 无 |
| UserUtil.deleteUser | 语法 | public ResultMessage deleteUser(String userName) |
| 前置条件 | 该用户名正确且存在 |
| 后置条件 | 系统删除该用户信息 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| User.initbyPO(UserPO po) | 根据po生成相应的领域对象 | |
| User.initbyVO(UserVO vo) | 根据vo生成相应的领域对象 | |
| User.modify(UserVO vo) | 将user的属性修改成vo中对应的信息 | |
| User.modifyPassword(String newPassword) | 将user的密码修改成新密码 | |
| User.getUserVO() | 得到该领域对象对应的vo | |
| User.getUserPO(UserVO vo) | 得到该领域对象对应的po | |
| User.changeCredit(double change) | 修改该领域对象的信用值 | |
| CreditRecordList.initlist(String ID) | 根据用户id生成领域对象列表 | |
| CreditRecordList.getVOs() | 获得信用记录的领域对象列表 | |
| CreditRecord.initbyVO  (CreditRecordVO vo) | 根据vo生成对应的领域对象 | |
| CreditRecord.getCreditRecordPO  (CreditRecordPO po) | 获得该领域对象对应的po | |
| UserDataService.add(UserPO po) | 根据提供的用户po增加持久化对象 | |
| UserDataService.findbyID(String ID) | 根据用户ID查找持久化对象 | |
| UserDataService.  findbyUserName(String userName) | 根据UserName查找持久化对象 | |
| UserDataService.  deleteUser(String userName) | 根据用户名删除相应用户 | |
| Promotionbl.showLevel() | 获得当前等级制度 | |

**Account的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Account.login | 语法 | public ResultMessage login(String userName, String password) |
| 前置条件 | 账号、密码符合语法规则 |
| 后置条件 | 系统确认账号密码是否正确，若正确，进入主界面并记录登录信息，否则提示错误信息 |
| Account.logout | 语法 | public ResultMessage logout(String ID) |
| 前置条件 | 系统已登录 |
| 后置条件 | 系统退出登录，记录登出信息 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| UserDataService.  Check(String userName, String password) | 检查账户密码是否正确 | |

**CreditRecord的接口规范**

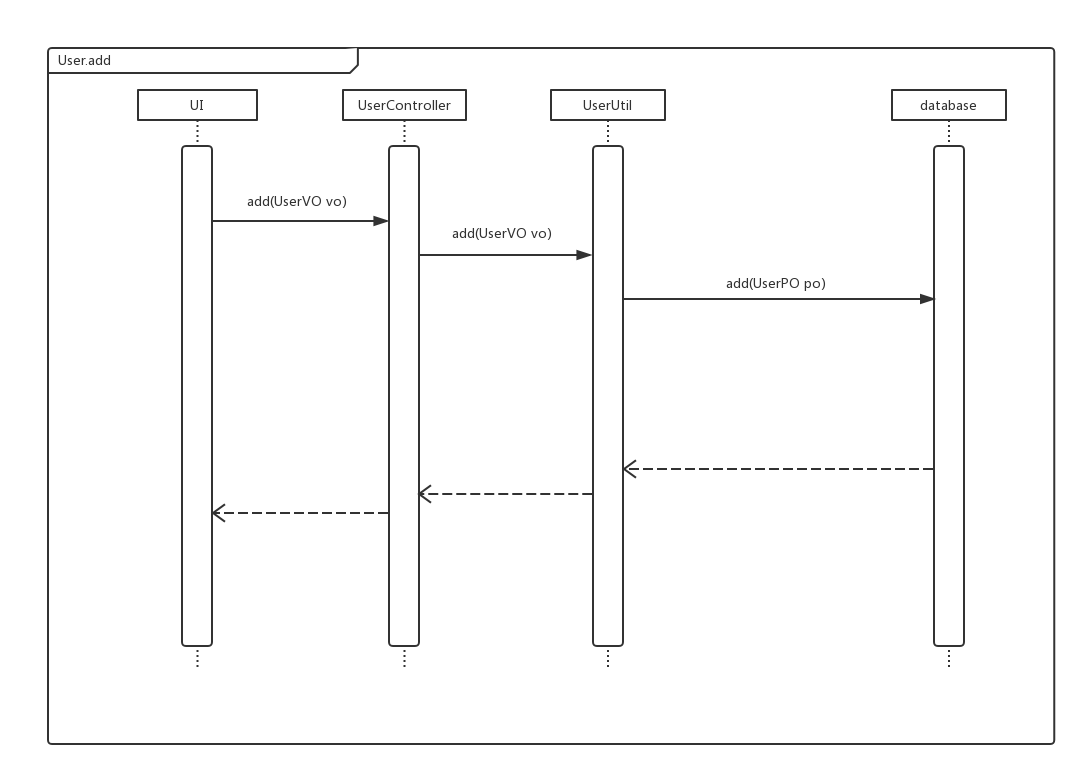
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditRecord.  getCreditRecordPO | 语法 | public CreditRecordPO getCreditRecordPO(CreditRecordVO vo) |
| 前置条件 | vo信息完整且正确 |
| 后置条件 | 无 |
| CreidtRecord.  getCreditRecordVO | 语法 | public List<CreditRecordVO> findCreditRecord(String ID) |
| 前置条件 | 该账户ID为客户且输入符合语法要求 |
| 后置条件 | 返回该账户所有信用 |
| CreditRecord.initbyPO | 语法 | public CreditRecord initbyPO(CreditRecordPO po) |
| 前置条件 | Po信息完整且正确 |
| 后置条件 | 无 |
| CreditRecord.initbyVO | 语法 | public CreditRecord initbyVO(CreditRecordVO vo) |
| 前置条件 | Vo信息完整切正确 |
| 后置条件 | 无 |

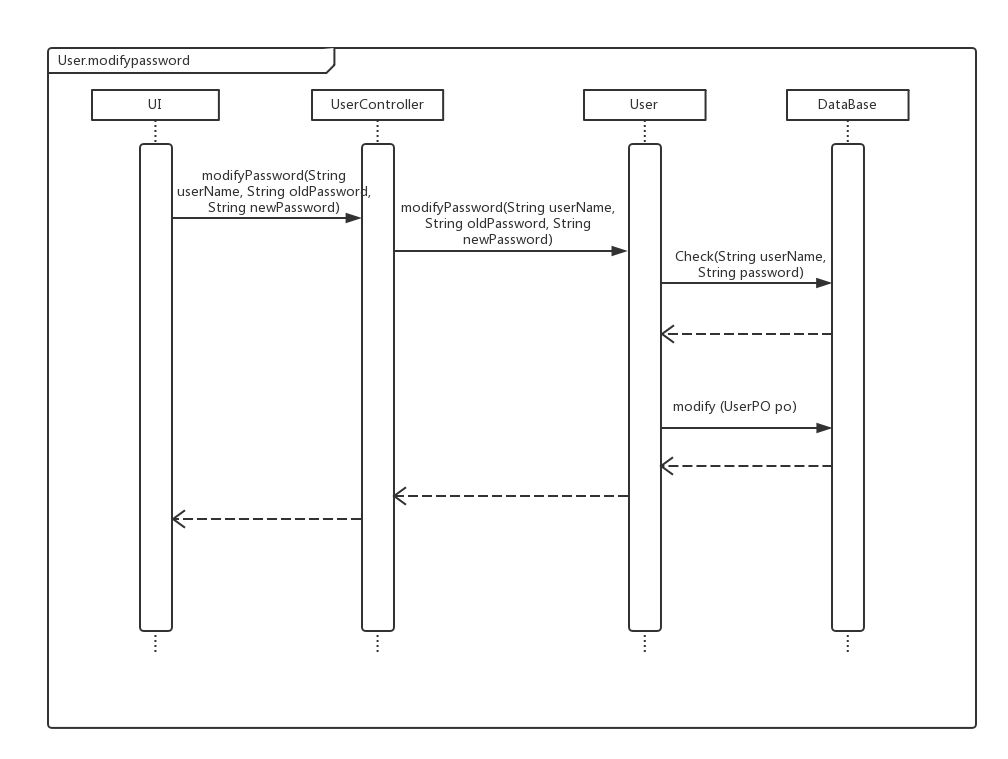
**CreditRecordList的接口规范**

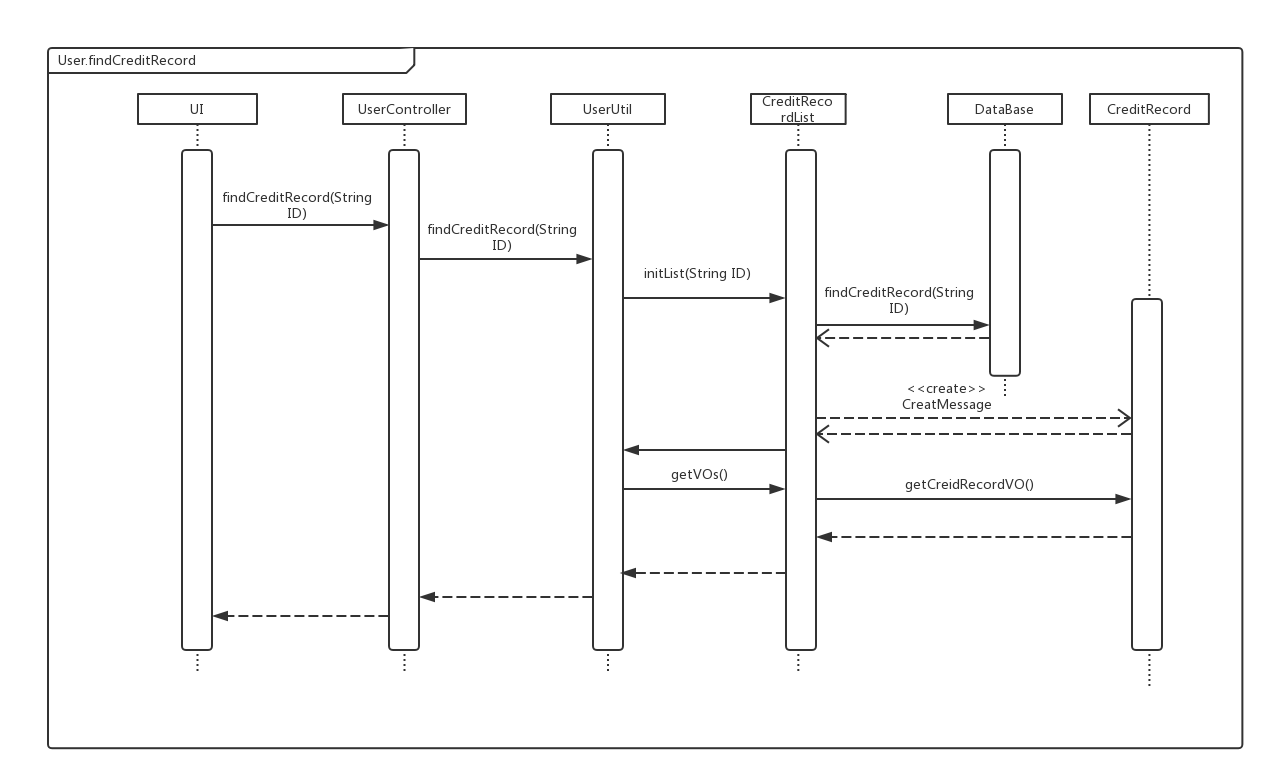
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditRecordList.  getVOs | 语法 | public List<CreditRecordVO> getVOs() |
| 前置条件 | 无 |
| 后置条件 | 返回信用记录领域对象列表 |
| CreditRecordList.initList | 语法 | public CreditRecordList initList(String ID) |
| 前置条件 | Id存在 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| CreditRecord.initbyPO | 根据po生成领域对象 | |
| CreditRecord.  getCreditRecordVO | 获得信用记录领域对象对应的vo | |

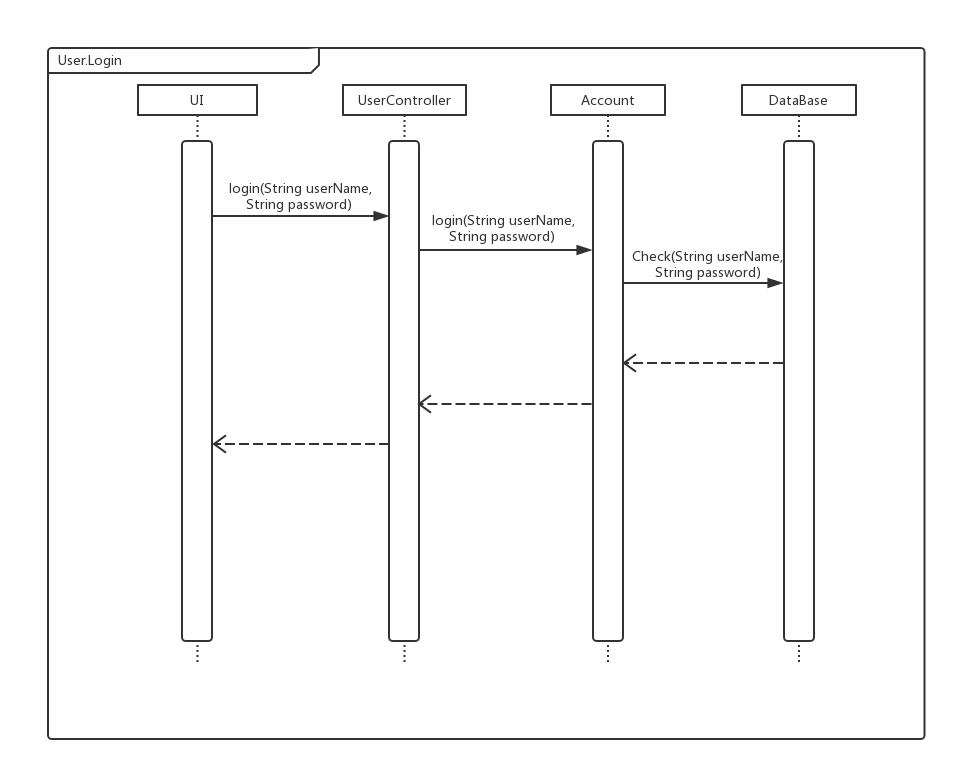
4、业务逻辑层的动态模型

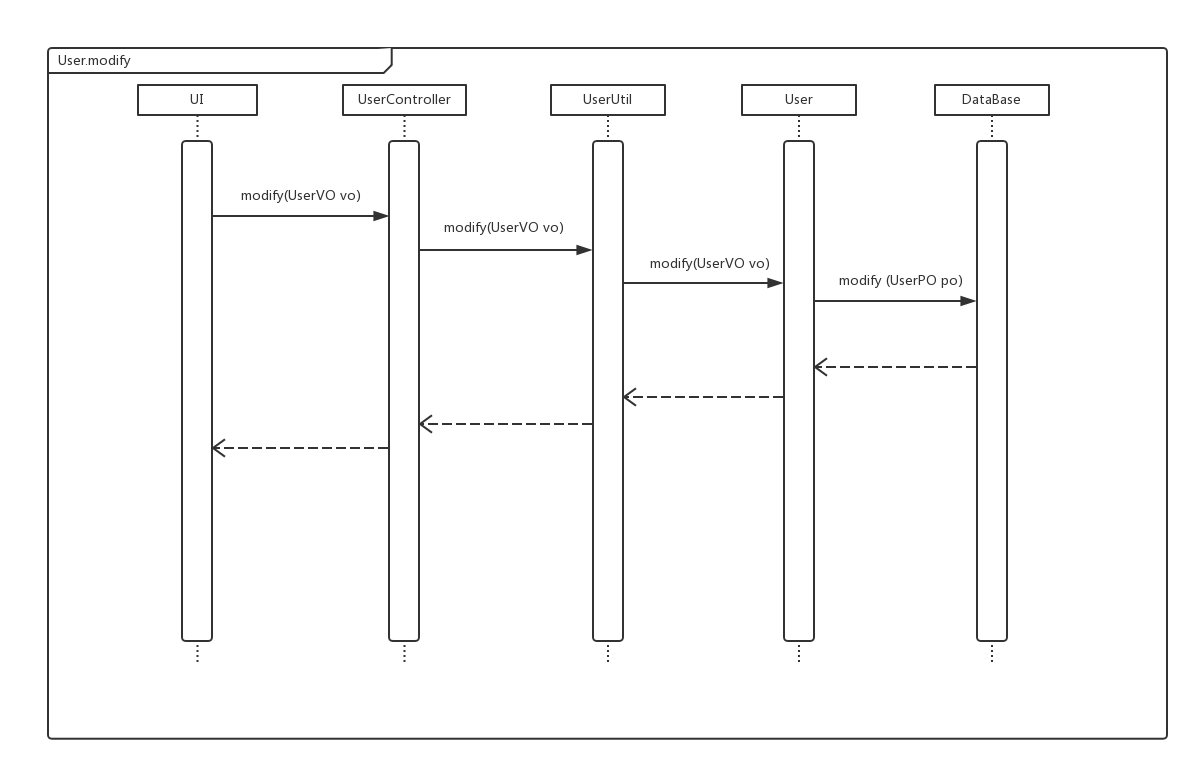
在系统账户管理中，登录、查找信用记录、修改密码、增加、修改个人信息的顺序图如下图所示：











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。

### order模块

1. 模块概述：

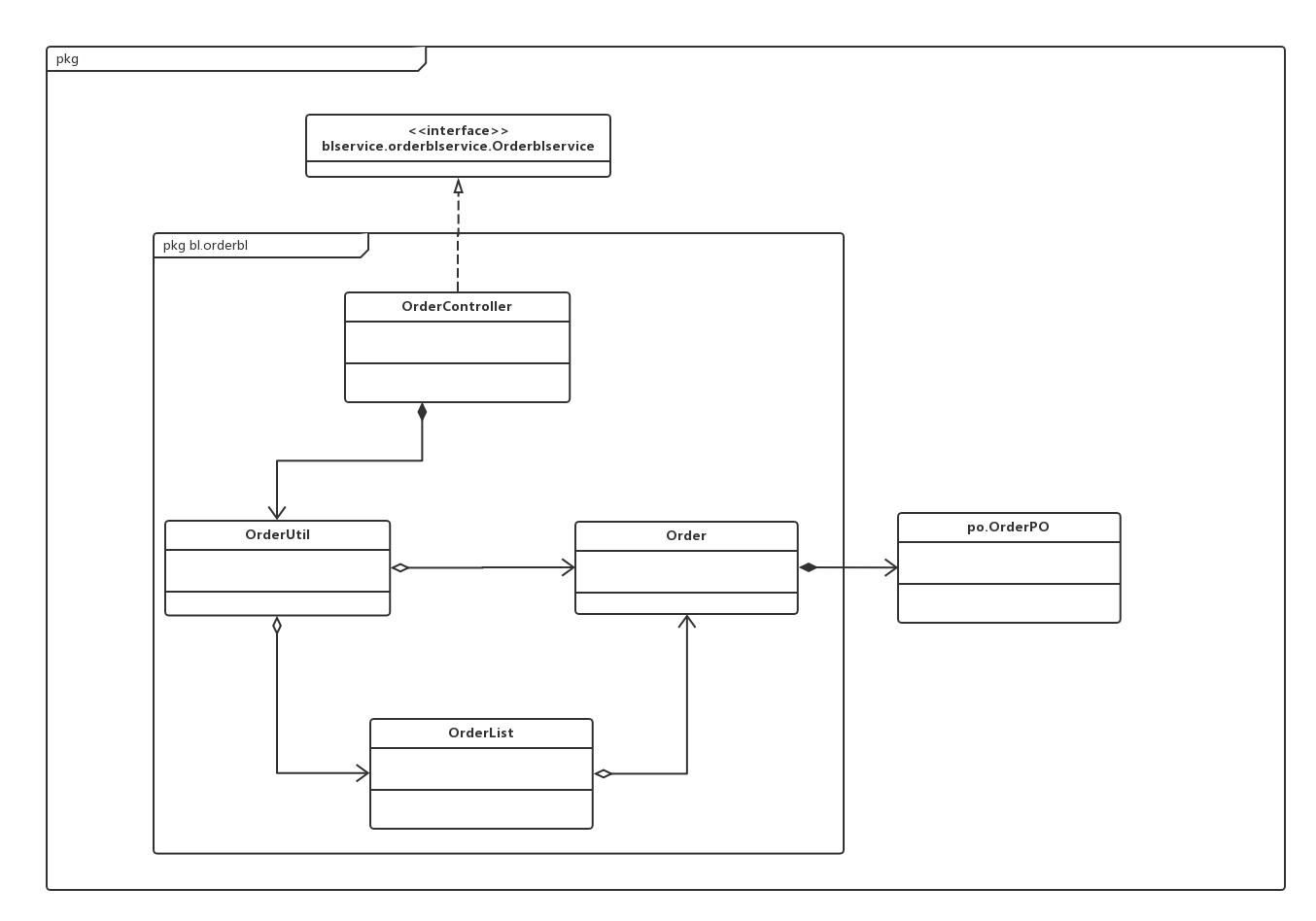
order模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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。

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Orderblservice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会将对商品管理的业务逻辑处理委托给OrderUtil和Order对象。OrderPO是作为账户信息的持久性对象被添加到设计模型中去的。

**Orderbl模块的设计如图：**



**Orderbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderController | 负责对应于Order模块管理所需要的服务 |
| OrderUtil | 订单的工具操作实现类 |
| Order | 订单的领域模型对象，拥有订单的所有信息 |
| OrderList | 订单的列表对象，拥有当前所需订单集合 |

1. 模块内部类的接口规范

**OrderController类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderController.createOrder | 语法 | public ResultMessage createOrder(OrderVO vo) |
| 前置条件 | orderVO包含完整且正确的信息 |
| 后置条件 | 调用OrderUtil的createOrder方法 |
| OrderController.cancelOrder | 语法 | public ResultMessage cancelOrder (String orderID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的cancelOrder方法 |
| OrderController.getOrder | 语法 | public OrderVO getOrder (String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getOrder方法 |
| OrderController.  getAbnormalOrder | 语法 | public List<OrderVO> getAbnormalOrder () |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getAbnormalOrder方法 |
| OrderController.  getUserOrderList | 语法 | public List<OrderVO> getUserOrderList (String userID, OrderType type) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getUserOrder方法 |
| OrderController.  getHotelOrderList | 语法 | public List<OrderVO> getHotelOrderList(String hotelID, OrderType type) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getHotelOrder方法 |
| OrderController.  executeOrder | 语法 | public ResultMessage executeOrder(String orderID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的executeOrder方法 |
| OrderController.  cancelAbnormalOrder | 语法 | public ResultMessage cancelAbnormalOrder (String orderID, Boolean isHalf) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的cancelAbnormalOrder方法 |
| OrderController.  getHistoryHotels | 语法 | public List<String> getHistoryHotels (String userId) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getHistoryHotels方法 |
| OrderController.  checkOut | 语法 | public ResultMessage checkOut(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getHistoryHotels方法 |
| OrderController.  comment | 语法 | public ResultMessage getHistoryHotels (String orderID) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getHistoryHotels方法 |
| OrderController.  getBookedRoomNum | 语法 | public int getBookedRoomNum(String hotelID, RoomType roomType, Date fromdate, Date todate) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的getBookedRoomNum方法 |
| OrderController.  createOffLineOrder | 语法 | public ResultMessage createOffLineOrder(OrderVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用OrderUtil的createOffLineOrder方法 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| OrderUtil.createOrder  (OrderVO vo) | 增加一个订单 | |
| OrderUtil.getOrder  (String ID) | 系统返回一条订单信息 | |
| OrderUtil.  getAbnormalOrder() | 返回当日全部异常订单vo | |
| OrderUtil.getUserOrderList  (String userID, OrderType type) | 返回某客户某种类全部订单vo | |
| OrderUtil.getHotelOrderList  (String hoteladdress, OrderType type) | 返回某酒店某种类全部订单vo | |
| OrderUtil.cancelOrder() | 系统实现客户撤销未执行订单 | |
| OrderUtil.executeOrder() | 系统实现订单执行 | |
| OrderUtil.cancelAbnormalOrder (Boolean isHalf) | 系统实现网站营销人员撤销异常订单 | |
| OrderUtil.getHistoryHotels (String userID) | 返回某客户的所有历史住过的酒店 | |
| OrderUtil.checkOut(String ID) | 系统实现订单退房 | |
| OrderUtil.comment(String orderID) | 系统实现记录订单评论 | |
| OrderUtil.getBookedRoomNum (String hotelID, RoomType roomType, Date fromDate, Date toDate) | 系统返回某酒店某房间类型在一段日其中的剩余房间数量 | |
| OrderUtil.createOffLineOrder (OrderVO vo) | 系统生成线下订单 | |

**OrderUtil类的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderUtil.createOrder | 语法 | public ResultMessage createOrder(OrderVO vo) |
| 前置条件 | orderVO包含完整且正确的信息 |
| 后置条件 | 调用Order的create方法 |
| OrderUtil.getOrder | 语法 | public OrderVO getOrder (String orderID) |
| 前置条件 | orderID格式正确 |
| 后置条件 | 调用OrderDataService对象的getOrderPO方法 |
| OrderUtil.  getAbnormalOrder | 语法 | public List<OrderVO> getAbnormalOrder () |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDataService类的getAbnormalOrderPO方法 |
| OrderUtil.getUserOrderList | 语法 | public List<OrderVO> getUserOrderList (String UserID, OrderType type) |
| 前置条件 | UserID格式正确 |
| 后置条件 | 调用OrderDataService类的getUserOrderPO方法 |
| OrderUtil.getHotelOrderList | 语法 | public List<OrderVO> getHotelOrderList (String hotelID, OrderType type) |
| 前置条件 | hoteladdress格式正确 |
| 后置条件 | 调用OrderDataService类的getHotelOrderPO方法 |
| OrderUtil.getHistoryHotels | 语法 | List<String> getHistoryHotels(String userID) |
| 前置条件 | userID格式正确 |
| 后置条件 | 调用OrderDataService类的getUserOrderPO方法 |
| OrderUtil.cancelOrder | 语法 | public ResultMessage cancelOrder (String userID) |
| 前置条件 | userID格式正确 |
| 后置条件 | 调用Order的calncel方法 |
| OrderUtil.  executeOrder | 语法 | public ResultMessage executeOrder (String userID) |
| 前置条件 | userID格式正确 |
| 后置条件 | 调用Order的execute方法 |
| OrderUtil.  cancelAbnormalOrder | 语法 | public ResultMessage cancelAbnormalOrder (String UserID,Boolean isHalf) |
| 前置条件 | userID格式正确 |
| 后置条件 | 调用Order的cancelAbnormal方法 |
| OrderUtil.  createOffLineOrder | 语法 | public ResultMessage createOffLineOrder(OrderVO vo) |
| 前置条件 | Ordervo 信息完整 |
| 后置条件 | 调用Order类的create方法 |
| OrderUtil.  checkOut | 语法 | public ResultMessage checkOut(String ID) |
| 前置条件 | 无 |
| 后置条件 | 调用Order类的checkOut方法 |
| OrderUtil.  comment | 语法 | public ResultMessage comment(String orderID) |
| 前置条件 | 无 |
| 后置条件 | 调用Order类的comment方法 |
| OrderUtil.  getBookedRoomNum | 语法 | public int getBookedRoomNum(String hotelID, RoomType roomType, Date fromDate, Date toDate) |
| 前置条件 | 无 |
| 后置条件 | 返回剩余房间数量 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Order.setOrder (OrderVO vo) | 通过OrderVO重设Order对象内容 | |
| Order.setOrder (OrderPO po) | 通过OrderPO重设Order对象内容 | |
| Order.getOrderVO () | 获取Order对象的VO | |
| Order.getOrderPO () | 获取Order对象的PO | |
| Order.create () | 执行创建订单的操作 | |
| Order.checkOut () | 执行订单房间退房的操作 | |
| Order.cancel () | 执行取消订单的操作 | |
| Order.execute () | 执行执行订单的操作 | |
| Order.cancelAbnormal () | 执行撤销异常订单的操作 | |
| Order.comment () | 执行标记评论订单的操作 | |
| Order. addCreditRecord(Double value, String type) | 执行修改该订单导致用户信用值变化的操作 | |
| Order.setOrderList (List<OrderPO> polist) | 由OrderPO组成的List初始化OrderList | |
| Order.  getOrderListVO () | 获得由OrderVO组成的List | |
| Order.  getHotelList() | 获得订单列表中的酒店名称列表 | |
| Order. getTypeRoomNum(RoomType roomType) | 返回订单列表中的某类型房间数量 | |
| OrderDataService.  getOrderPO(String orderID) | 系统返回一条订单信息 | |
| OrderDataService.  getAbnormalOrderPO () | 返回当日全部异常订单po | |
| OrderDataService.  getUserOrderPO (String userID, OrderType type) | 返回某客户某种类全部订单po | |
| OrderDataService .  getDateOrderPO | 返回某日某酒店某房间类型的订单 | |
| OrderDataService.  getHotelOrderPO (String hotelID, OrderType type) | 返回某酒店某种类全部订单po | |

**Order类的接口规范**

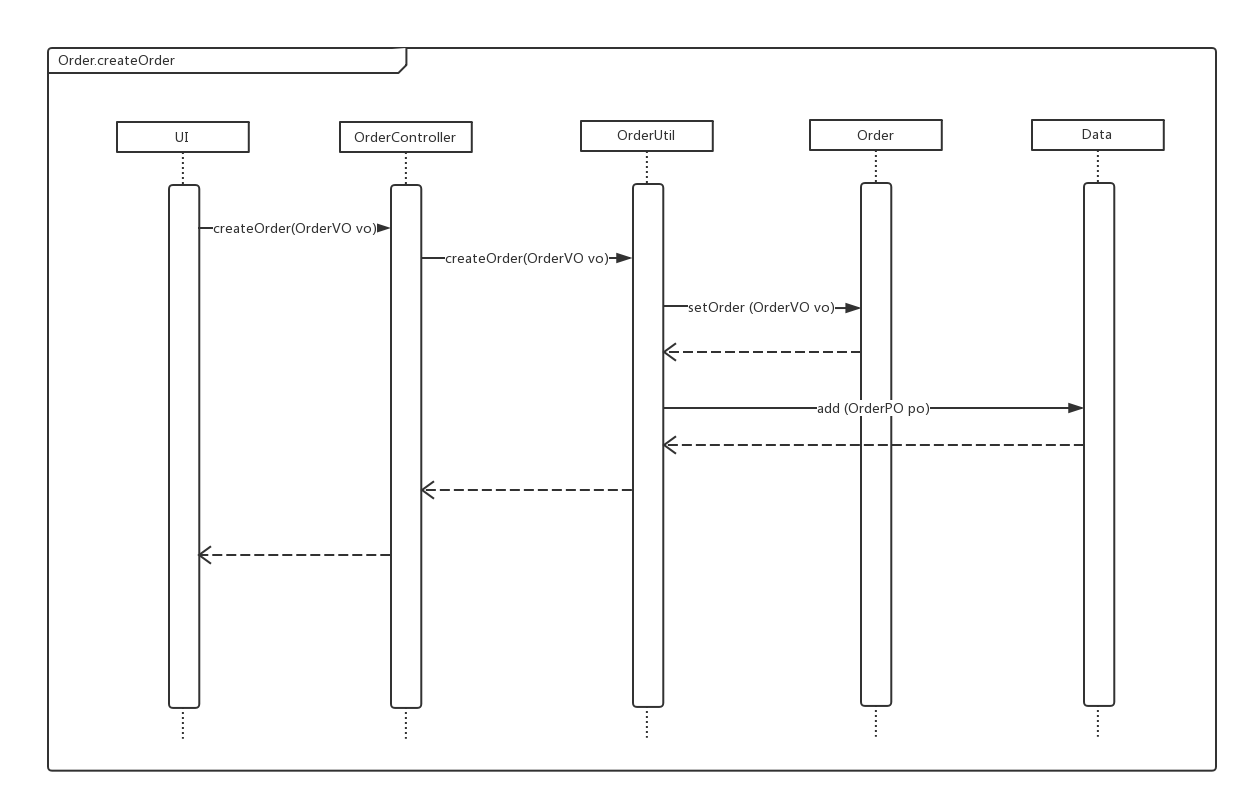
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.setOrder (OrderVO vo) | 语法 | public ResultMessage setOrder (OrderVO vo) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.setOrder (OrderPO po) | 语法 | public ResultMessage setOrder (OrderPO po) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.getOrderVO | 语法 | public OrderVO getOrderVO () |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.getOrderPO | 语法 | public OrderPO getOrderPO () |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Order.modify | 语法 | public ResultMessage modify() |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDataService.modify |
| Order.create | 语法 | public ResultMessage create () |
| 前置条件 | 无 |
| 后置条件 | 调用OrderDataService.add |
| Order.checkOut | 语法 | public ResultMessage checkOut () |
| 前置条件 | 无 |
| 后置条件 | 调用modify () |
| Order.cancel | 语法 | public ResultMessage cancel () |
| 前置条件 | 无 |
| 后置条件 | 调用modify () |
| Order.execute | 语法 | public ResultMessage excute () |
| 前置条件 | 无 |
| 后置条件 | 调用modify () |
| Order.cancelAbnormal | 语法 | public ResultMessage cancelAbnormal () |
| 前置条件 | 无 |
| 后置条件 | 调用modify () |
| Order.comment | 语法 | public ResultMessage comment () |
| 前置条件 | 无 |
| 后置条件 | 调用modify () |
| Order. addCreditRecord | 语法 | public ResultMessage addCreditRecord(Double value, String type) |
| 前置条件 | 无 |
| 后置条件 | 调用User.addCreditRecord |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| User.findbyID (String ID) | 获取一个UserVO | |
| User.addCreditRecord (CreditRecordVO vo) | 增加该客户信用记录并持久化保存数据 | |
| OrderDataService.  add (OrderPO po) | 系统添加新的订单记录 | |
| OrderDataService.  modify (OrderPO po) | 系统修改订单记录 | |
| TimeService. getDate () | 获取服务器当前时间 | |

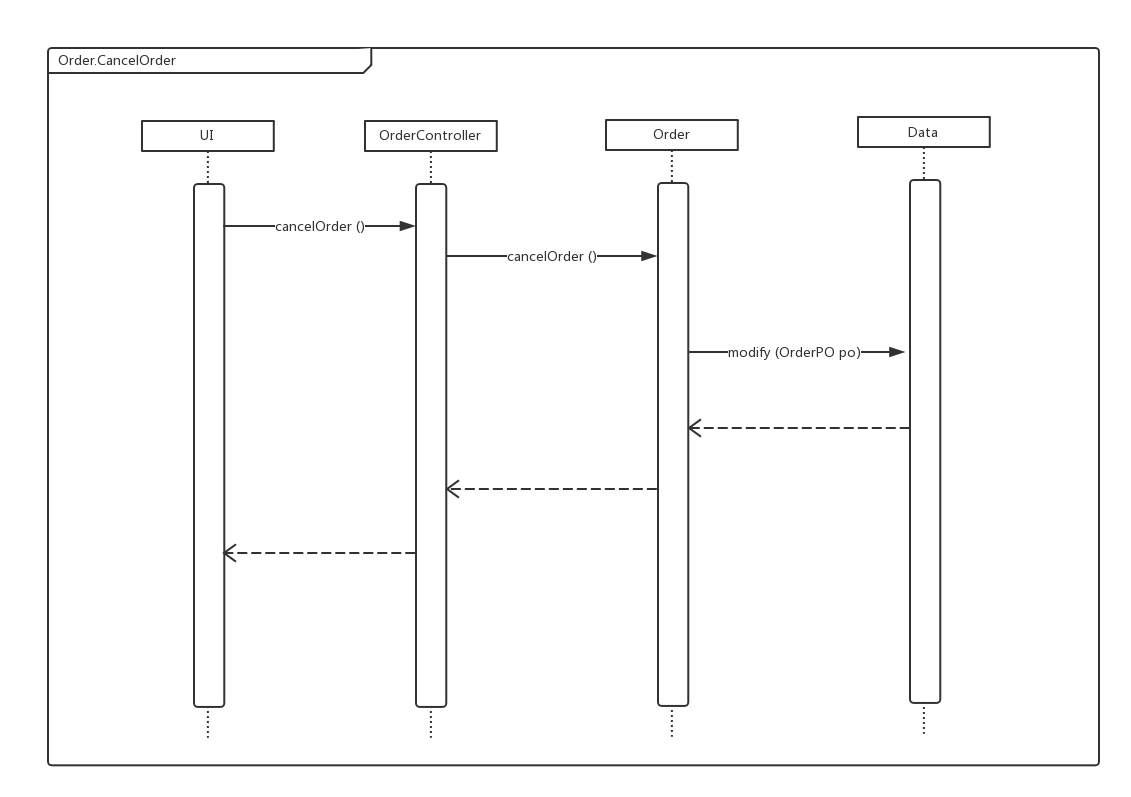
**OrderList 类的接口规范**

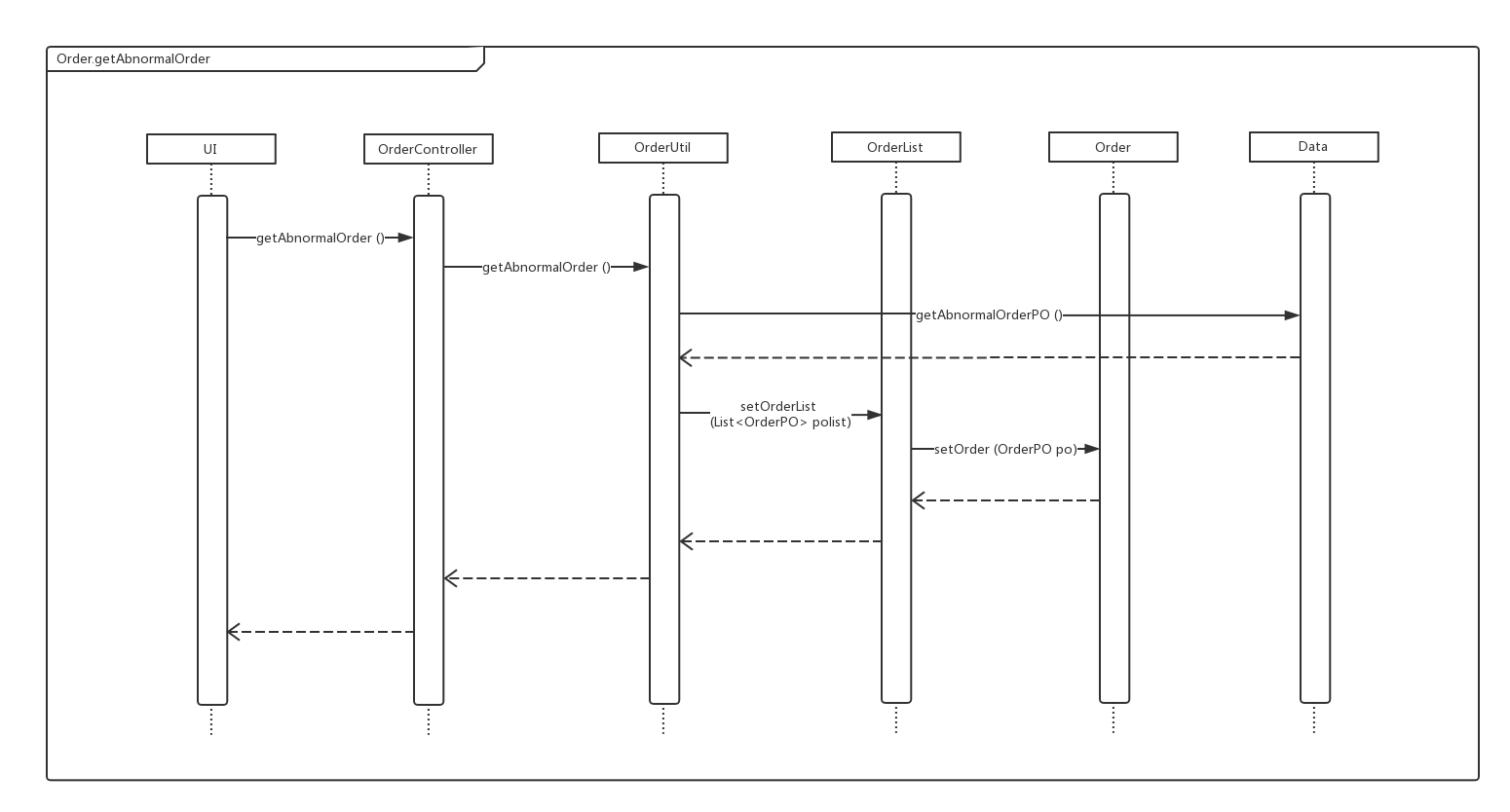
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderList.setOrderList | 语法 | public ResultMessage setOrderList (List<OrderPO> polist) |
| 前置条件 | 已存在OrderList对象 |
| 后置条件 | 无 |
| OrderList.getOrderListVO | 语法 | public List<OrderVO> getOrderListVO () |
| 前置条件 | 已存在OrderList对象 |
| 后置条件 | 无 |
| OrderList.getHotelList | 语法 | public List<String> getHotelList() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderList.getTypeRoomNum | 语法 | public int getTypeRoomNum(RoomType roomType) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| Order.setOrder (OrderPO po) | 通过PO增加Order对象 | |
| Order.getOrderVO | 获取Order的VO对象 | |

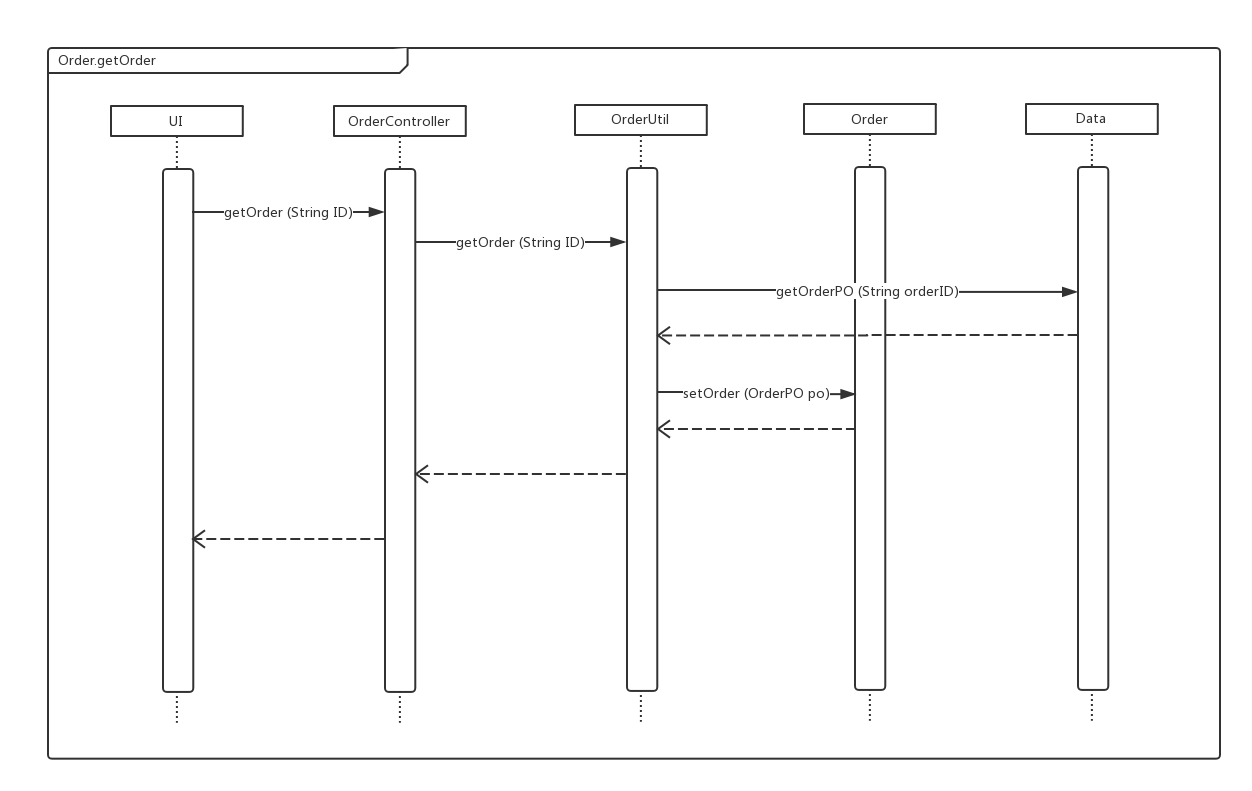
1. 业务逻辑层的动态模型

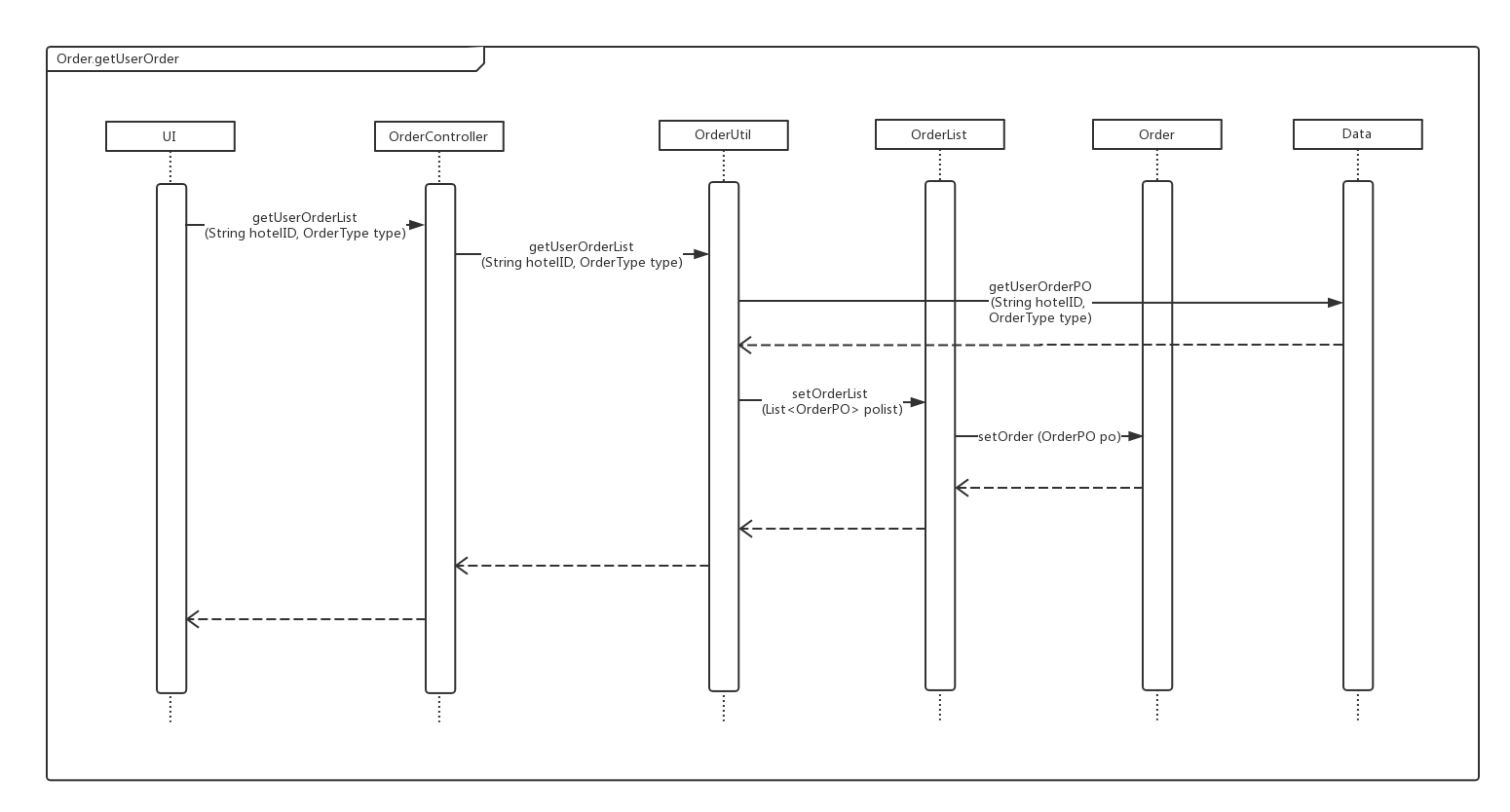
在系统订单管理中，业务分为增加、查找、撤销未执行订单与异常订单、查找账户信息，其中的顺序图如下图所示：

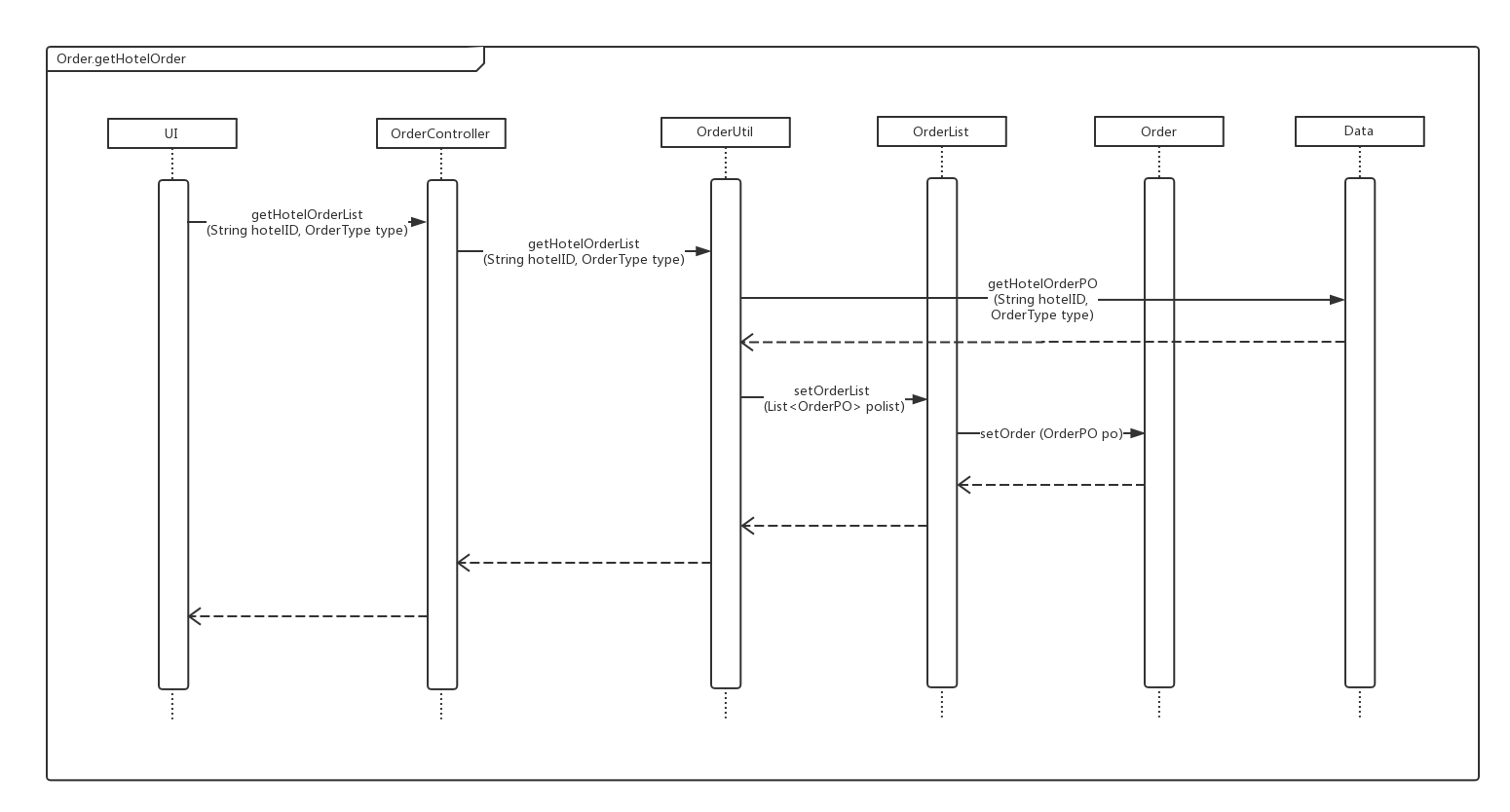


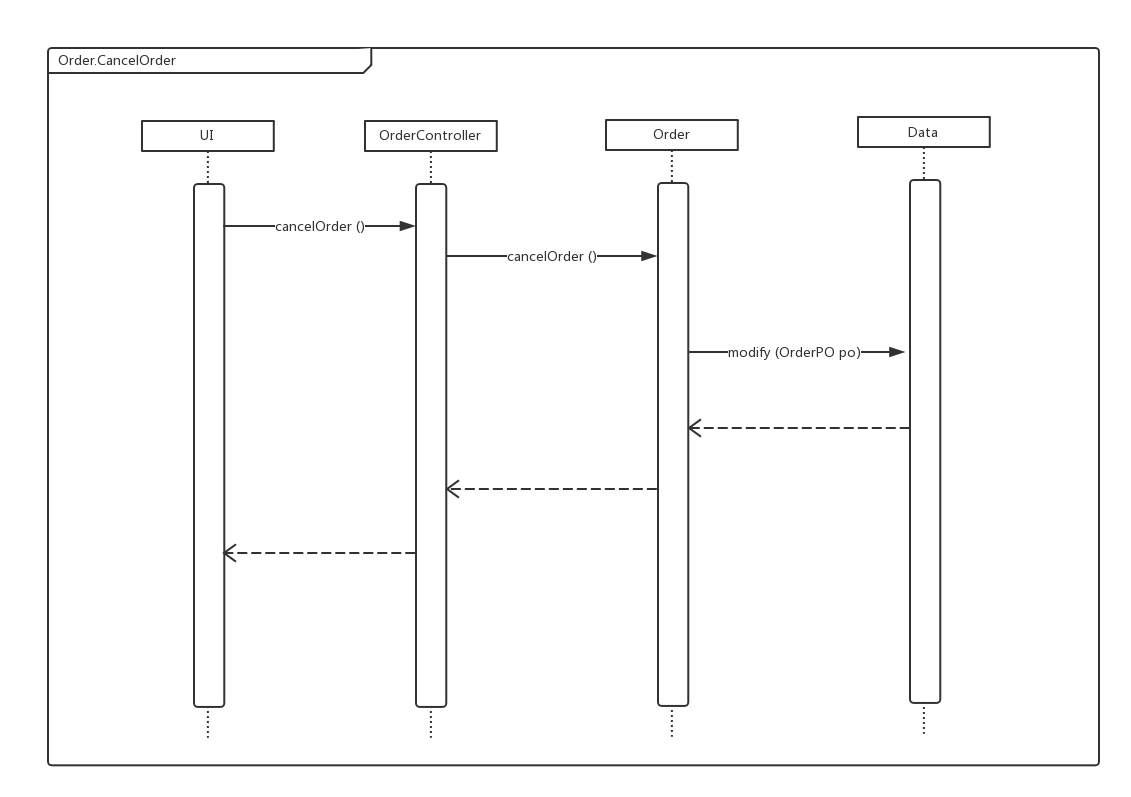


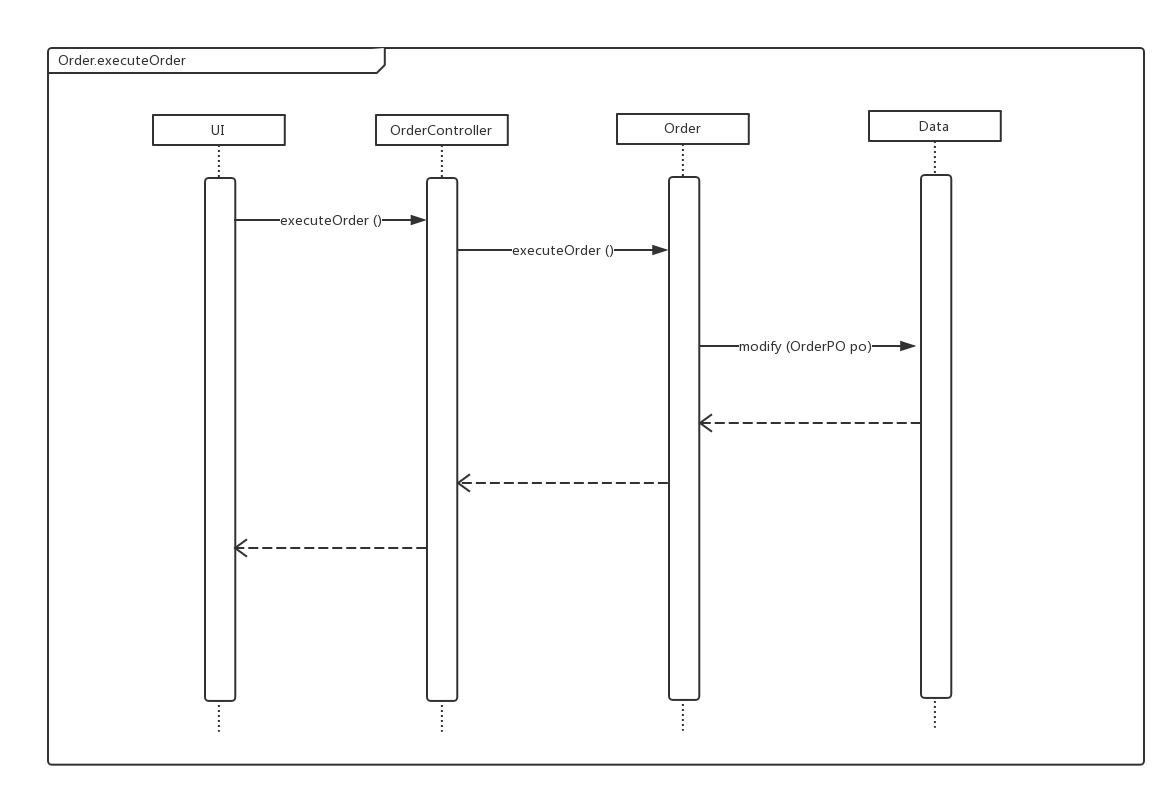












1. 业务逻辑层的设计原理

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。

### Hotelbl模块

1、模块概述：

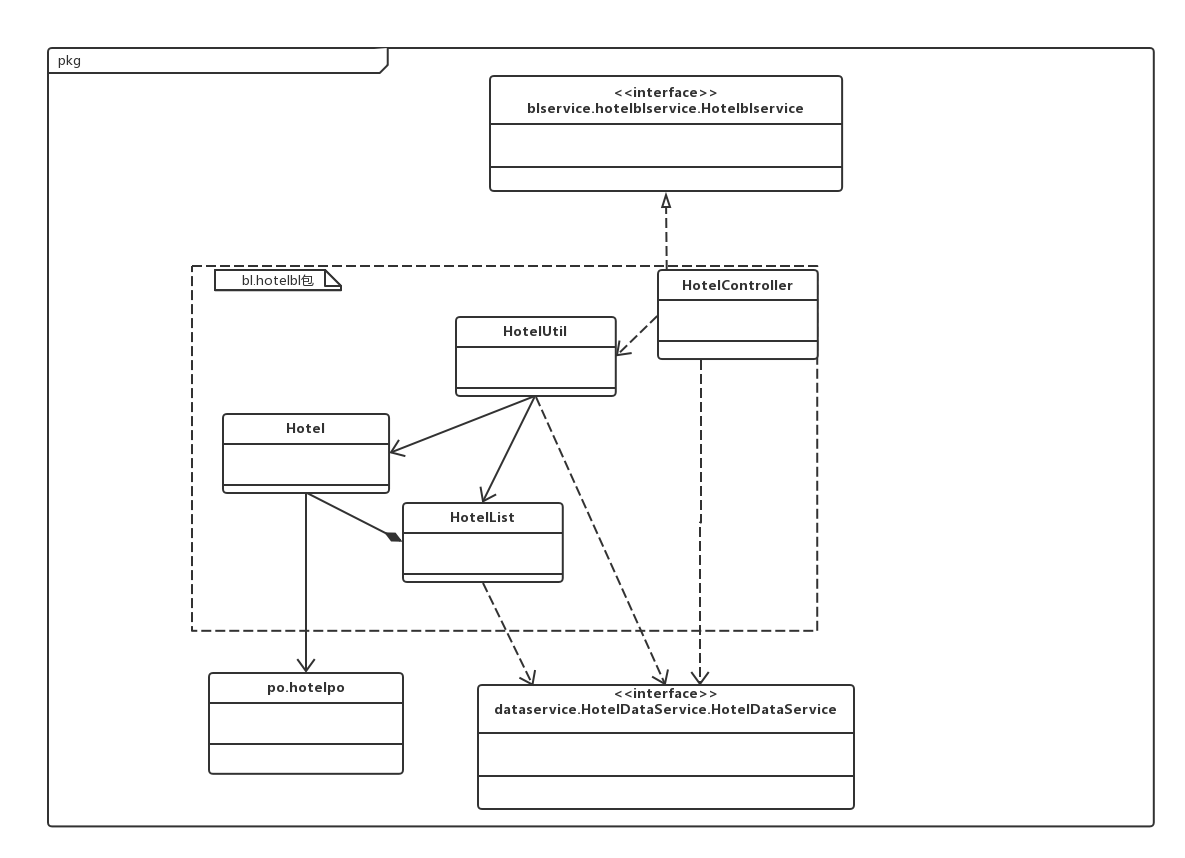
**Hotelbl**模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

**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HotelblService接口。在业务逻辑层和数据层之间添加HotelDataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加HotelController，这样HotelController会将对酒店管理的业务逻辑处理委托给Hotel对象。HotelPO是作为账户信息的持久性对象被添加到设计模型中去的。

**Hotelbl模块的设计如图：**



**Hotelbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| HotelController | 负责实现酒店界面所需要的服务 |
| Hotel | 酒店的领域模型对象，拥有一所酒店的名称、地址、评价等信息，可以帮助完成酒店界面所需要的服务 |
| HotelList | 负责生成和维护领域对象列表 |
| HotelUtil | 负责协助创建和修改领域对象和领域对象列表 |

3、模块内部类的接口规范

**HotelController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelController.findHotels | 语法 | public List findHotels(HotelFilter filter, String userID, Boolean historyOnly) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelUtil类的getHotelList方法,如果historyOnly,调用Orderblservice类的getHistoryOrders()方法 |
| HotelController.addHotel | 语法 | public ResultMessage addHotel(HotelVO hotelVO, UserVO userVO) |
| 前置条件 | 输入符合格式 |
| 后置条件 | 调用HotelUtil类的addHotel方法和Userblservice类的add、deleteUser方法和IDUtil类的generateNewHotelID方法 |
| HotelController.modifyHotel | 语法 | public ResultMessage modifyHotel(HotelVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用Hotelutil类的modify方法 |
| HotelController.getAvailable  RoomNum | 语法 | public int getAvailableRoomNum(Date schFrom, Date schTo, String hotelId, RoomType type) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDataService类的getRooms方法和Orderblservice类的getBookedRoomNum方法 |
| HotelController.getHotelVO | 语法 | public HotelVO getHotelVO(String hotelID) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelUtil类的getHotel方法 |
| HotelController.getCitys | 语法 | public List getCitys() |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDataService类的getCitys方法 |
| HotelController.getAreas | 语法 | public List getAreas(CityVO city) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDataService类的getAreas方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelUtil.addHotel | 增加酒店的持久化对象 | |
| HotelUtil.modify | 修改酒店和房间的持久化对象 | |
| HotelUtil.getHotel | 根据酒店地址返回单一的酒店vo | |
| HotelUtil.getHotelList | 系统返回该筛选条件下的所有酒店 vo | |
| Userblservice.add | 增加酒店工作人员的持久化对象 | |
| Userblservice.deleteUser | 删除酒店工作人员的持久化对象 | |
| HotelDataService.getCitys | 在数据库中获得所有CityPO | |
| HotelDataService.getAreas | 根据CityPO返回所有AreaPO | |
| HotelDataService.getRooms | 获得房间的持久化对象 | |
| Orderblservice. getHistoryHotels | 根据userID返回酒店地址列表 | |
| Orderblservice.getBooked  RoomNum | 返回酒店某类型房间在某段时间被预订的数量 | |
| IDUtil.generateNewHotelID | 为新增的酒店分配ID | |

**Hotel的接口规范**

**Hotel的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.generateVO | 语法 | public HotelVO generateVO  () |
| 前置条件 | 无 |
| 后置条件 | 返回一个酒店的vo |
| Hotel.initByPO | 语法 | public Hotel  initByPO(HotelPO po, List<RoomPO> rooms) |
| 前置条件 | 无 |
| 后置条件 | 创建并返回一个酒店领域类对象 |
| Hotel.modifyAndReturnPO | 语法 | public HotelPO  modifyAndReturnPO(HotelVO vo) |
| 前置条件 | 无 |
| 后置条件 | 返回一个修改后的酒店的po |

**HotelList的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelList.initList | 语法 | public HotelList initList  (HotelFilter filter) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelDataService类的findHotels、getRooms方法，调用Hotel类的initByPO方法，创建一个领域对象的列表 |
| HotelList.getVOs | 语法 | public List< HotelVO >  getVOs() |
| 前置条件 | 已经创建一个领域对象列表 |
| 后置条件 | 调用Hotel类的generateVO方法，系统返回HotelVO的列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Hotel.generateVO | 返回一个酒店vo | |
| Hotel.initByPO | 返回一个酒店领域对象 | |
| HotelDataService.findHotels | 返回多个酒店 po | |
| HotelDataService.getRooms | 返回多个房间po | |

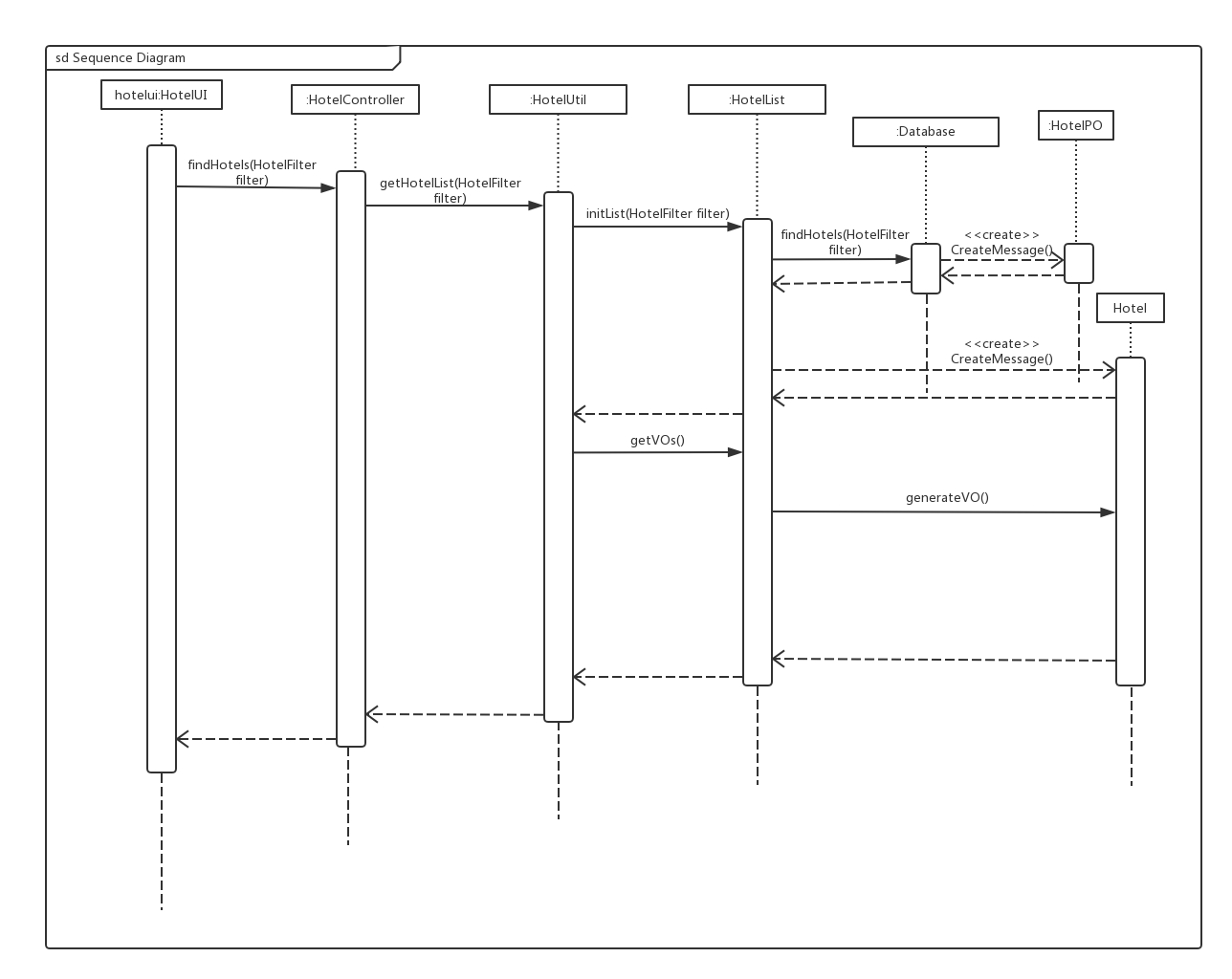
**HotelUtil的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelUtil.addHotel | 语法 | public ResultMessage addHotel  (HotelVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用Hotel类的modifyAndReturnPO方法和HotelDataService类的addHotelInfo方法 |
| HotelUtil.getHotel | 语法 | public HotelVO  getHotel(String hotelID) |
| 前置条件 | 输入符合格式 |
| 后置条件 | 调用HotelDataService类的getHotelInfo和getRooms方法，调用Hotel类的initByPO方法和generateVO方法，返回一个酒店的vo |
| HotelUtil.getHotelList | 语法 | public List<HotelVO> getHotelList  (HotelFilter filter) |
| 前置条件 | 无 |
| 后置条件 | 调用HotelList类的initList和getVOs方法，系统返回HotelVO的列表 |
| HotelUtil.modify | 语法 | public ResultMessage modify(HotelVO vo) |
| 前置条件 | 无 |
| 后置条件 | 调用Hotel类的modifyAndReturnPO方法，调用HotelDataService类的modifyRooms和modifyHotelInfo方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| HotelList.initList | 根据筛选条件创建酒店领域对象列表 | |
| HotelList.getVOs | 返回酒店信息的vo列表 | |
| HotelDataService.addHotelInfo | 增加酒店的持久化对象 | |
| HotelDataService.getHotelInfo | 返回一个酒店 po | |
| HotelDataService.modifyHotelInfo | 修改酒店的持久化对象 | |
| Hotel.initByPO | 返回一个酒店领域对象 | |
| Hotel.generateVO | 根据po返回一个酒店vo | |
| Hotel.modifyAndReturnPO | 根据vo返回一个修改后的酒店po | |
| HotelDataService.modifyRooms | 修改酒店房间的持久化对象 | |
| HotelDataService.getRooms | 返回该酒店的多个房间po | |

4、业务逻辑层的动态模型

在酒店管理业务中，分为浏览酒店列表，浏览酒店详细信息，新增酒店，维护酒店信息功能。

下图表明了浏览酒店列表的相关对象之间的协作。



**浏览酒店列表**

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。



### Promotion模块

1. 模块概述：

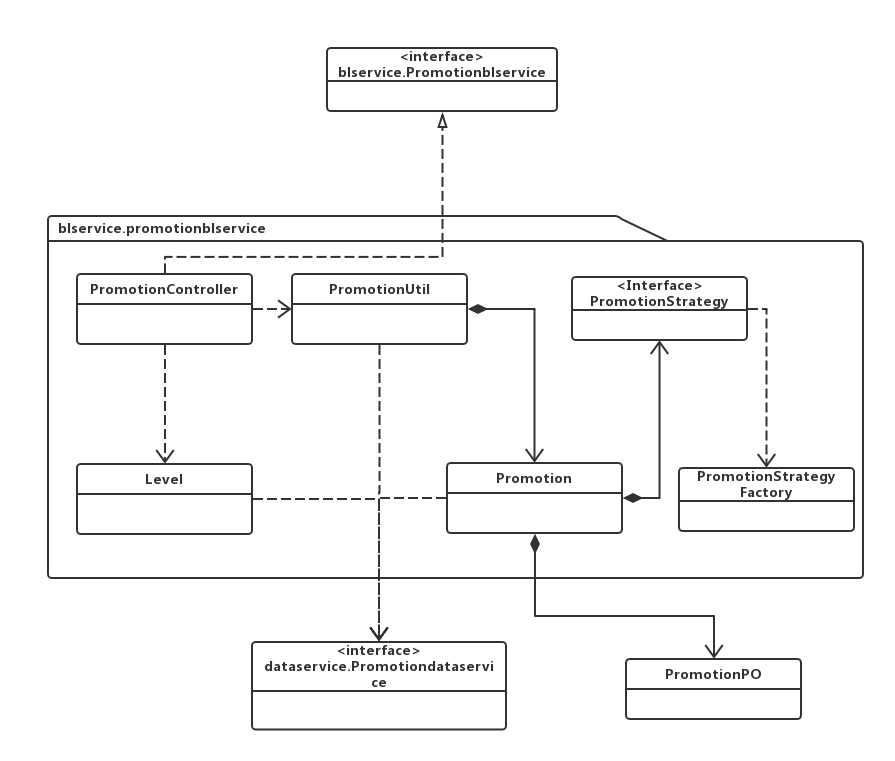
Promotion模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Promotion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promotionblservice接口。在业务逻辑层和数据层之间添加promotiondataservice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会将对促销策略管理的业务逻辑处理委托给Promotion对象和PromotionUtil工具类。PromotionStrategy是专门用于不同种类促销计算价格的策略接口，每个Promotion拥有一个策略。PromotionPO是作为账户信息的持久性对象被添加到设计模型中去的，Level是包含用户等级制度信息，并处理相关数据的类。

**Promotionbl模块的设计如图：**



**Promotionbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 负责控制促销策略模块内部的逻辑跳转 |
| PromotionUtil | 提供生成VO和PO的方法，和数据库交互的方法，增查Promotion数据的方法 |
| Promotion | 提供维持在内存中的Promotion信息，提供修改Promotion数据的方法 |
| PromotionStrategy | 提供计算价格方法的接口，多种策略类实现该接口 |
| PromotionStrategyFactory | 为Promotion创建不同PromotionStrategy的工厂 |

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.add | 语法 | public ResultMessage add(PromotionVO vo) |
| 前置条件 | 系统登录账户为酒店工作人员或网站营销人员 |
| 后置条件 | 系统调用PromotionUtil的add方法 |
| PromotionController.  getPromotion | 语法 | public List<PromotionVO> getPromotion (String hotelAddress, String userID) |
| 前置条件 | 输入的地址名称、客户ID符合文本规则 |
| 后置条件 | 系统调用PromotionUtil的getPromotion方法以及Promotion的toVO方法 |
| PromotionController.  showHotelPromotion | 语法 | public List<PromotionVO> showHotelPromotion (String hotelAddress) |
| 前置条件 | 输入的地址名称符合文本规则 |
| 后置条件 | 系统调用PromotionUtil的showHotelPromotion方法以及Promotion的toVO方法 |
| PromotionController.  showWebsitePromotion | 语法 | public List<PromotionVO> showWebsitePromotion() |
| 前置条件 | 无 |
| 后置条件 | 系统调用PromotionUtil的showWebsitePromotion方法以及Promotion的toVO方法 |
| PromotionController.  modify | 语法 | public ResultMessage modify (PromotionVO vo) |
| 前置条件 | 填写的内容符合规范且齐全 |
| 后置条件 | 系统调用Promotion的modify方法 |
| PromotionController.  delete | 语法 | public ResultMessage delete (String promotionID) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| PromotionController.  showLevel | 语法 | public LevelVO showLevel () |
| 前置条件 | 无 |
| 后置条件 | 系统调用Level的showLevel方法 |
| PromotionController.  modifyLevel | 语法 | public ResultMessage modifyLevel (LevelVO vo) |
| 前置条件 | 输入的内容符合规范且齐全 |
| 后置条件 | 系统调用Level的modifyLevel方法 |
| PromotionController.  getPrice | 语法 | public double modifyLevel (String userID, String hotelAdrress, double rawPrice) |
| 前置条件 | 无 |
| 后置条件 | 系统调用PromotionUtil的getPrice方法 |
| PromotionController.  calLevel | 语法 | public int calLevel (double credit) |
| 前置条件 | 无 |
| 后置条件 | 系统调用Level的calLevel方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Promotion.modify (PromotionVO vo) | 修改一个促销策略 | |
| PromotionUtil.add (PromotionVO vo) | 增加一个促销策略 | |
| PromotionUtil.delete  (String promotionID) | 删除一个促销策略 | |
| PromotionUtil.  getPromotion(String hotelAddress, String userID) | 获取所有促销策略 | |
| PromotionUtil.  showHotelPromotion(String hotelAddress) | 获取特定酒店全部促销策略 | |
| PromotionUtil.  showWebsitePromotion() | 获取网站全部促销策略 | |
| Level.showLevel() | 获取等级制度 | |
| Level.modifyLevel(LevelVO vo) | 修改等级制度 | |
| Level.calLevel(double credit) | 获取特定信用值对应的等级 | |
| PromotionDataService.  init() | 初始化持久化数据库 | |
| PromotionDataService.  finish() | 结束持久化数据库的使用 | |

**Promotion的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Promotion.modify | 语法 | ResultMessage modify (PromotionVO vo) |
| 前置条件 | 填写的内容符合规范且齐全 |
| 后置条件 | 系统修改该策略并持久化存储该策略数据 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| PromotionDataService.  modify (PromotionPO po) | 修改单一促销策略持久化对象 | |

**PromotionUtil的接口规范**

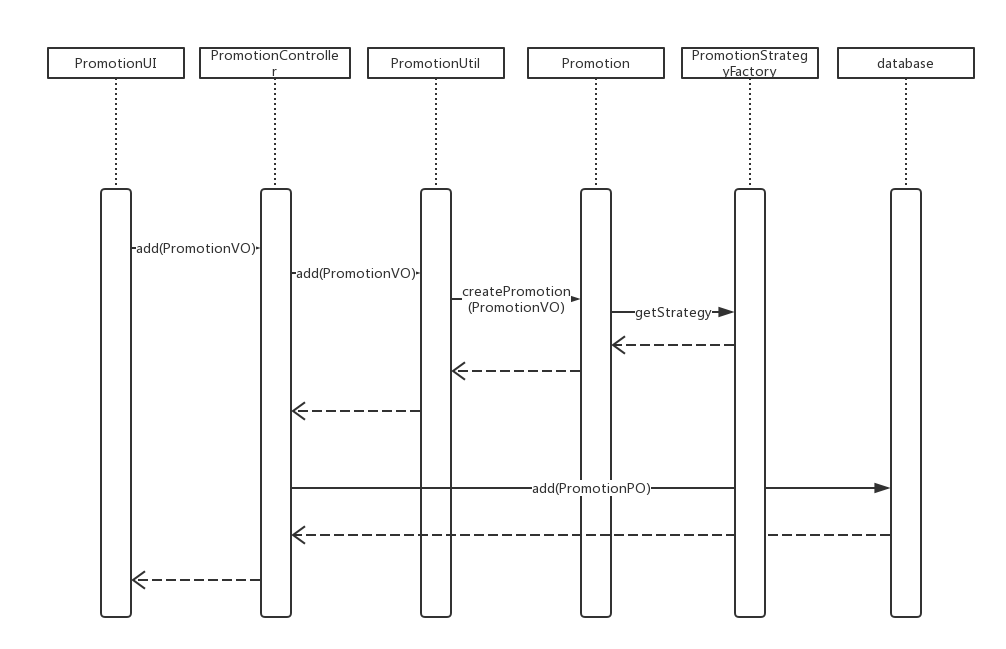
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PromotionUtil.add | 语法 | static ResultMessage add(PromotionVO vo) | |
| 前置条件 | 系统登录账户为酒店工作人员或网站营销人员 | |
| 后置条件 | 系统持久化增加该促销策略数据 | |
| PromotionUtil.delete | 语法 | public ResultMessage delete (String promotionID) | |
| 前置条件 | 无 | |
| 后置条件 | 无 | |
| PromotionUtil.  getPromotion | 语法 | static List<PromotionVO> getPromotion (String hotelAddress, String userID) | |
| 前置条件 | 输入的地址名称、客户ID符合文本规则 | |
| 后置条件 | 系统根据酒店地址返回符合该酒店该客户订单的所有促销策略（包括酒店促销策略与网站促销策略） | |
| PromotionUtil.  showHotelPromotion | 语法 | static List<Promotion> showHotelPromotion (String hotelAddress) | |
| 前置条件 | 输入的地址名称符合文本规则 | |
| 后置条件 | 系统根据酒店地址返回该酒店的所有促销策略 | |
| PromotionUtil.  showWebsitePromotion | 语法 | static List<Promotion> showWebsitePromotion() | |
| 前置条件 | | 无 |
| 后置条件 | | 系统返回所有网站促销策略 |
| PromotionUtil.getPrice | 语法 | | public double modifyLevel (String userID, String hotelAdrress, double rawPrice) |
| 前置条件 | | 无 |
| 后置条件 | | 系统计算并返回折扣后的价格 |
| 需要的接口（需接口） | | | |
| 服务名 | 服务内容 | | |
| PromotionDataService.  add (PromotionPO po) | 根据提供的促销策略po增加持久化对象 | | |
| PromotionDataService.  showHotelPromotion (String hotelAddress) | 根据hotelAddress查找该酒店的所有持久化对象 | | |
| PromotionDataService.  showWebsitePromotion () | 查找所有网站营销策略 | | |

**Level的接口规范**

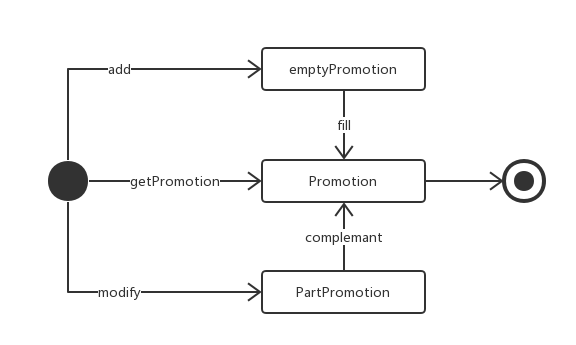
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Level.showLevel | 语法 | LevelVO showLevel () |
| 前置条件 | 无 |
| 后置条件 | 系统返回用户等级信息 |
| Level.modifyLevel | 语法 | ResultMessage modifyLevel (LevelVO vo) |
| 前置条件 | 无 |
| 后置条件 | 系统修改用户等级策略并持久化存储该策略数据 |
| Level.calLevel | 语法 | int calLevel (double credit) |
| 前置条件 | 无 |
| 后置条件 | 系统根据信用值计算等级 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| PromotionDataService.  showLevel () | 返回用户等级策略的持久化对象 | |
| PromotionDataService.  modifyLevel (LevelPO po) | 修改单一用户等级策略的持久化对象 | |

1. 业务逻辑层的动态模型

在促销策略模块中，业务分为增加、修改、查找促销策略和修改、查找会员等级制度，其中增加促销策略的顺序图如下图所示：



下图描述了Promotion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。

### Comment模块

1、模块概述：

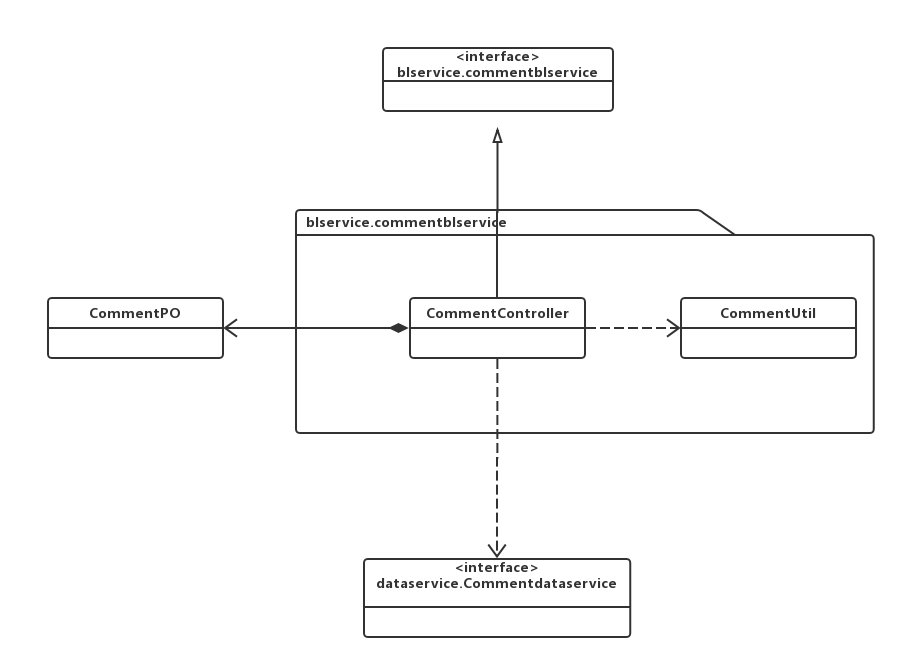
Comment模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Comment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commentblservice接口。在业务逻辑层和数据层之间添加commentdataservice接口。为了隔离业务逻辑职责和逻辑控制职责，添加CommentController，这样CommentController会调用CommentUtil实现的方法传递CommentVO和CommentPO。CommentPO是作为评价的持久性对象被添加到设计模型中去的。

**Comment模块的设计如图：**



**Comment模块中各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| CommentController | 负责对应于传递Comment信息所需要的服务 |
| CommentUtil | 实现传递Comment信息所需要的方法 |

1. 模块内部类的接口规范

**CommentController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommentController.  getComments | 语法 | public List<CommentVO>  getComment(String hotelAddress) |
| 前置条件 | 存在该酒店的评价 |
| 后置条件 | 返回该酒店的所有评价 |
| CommentController.  addComment | 语法 | public ResultMessage addComment(CommentVO vo) |
| 前置条件 | 评价输入合法 |
| 后置条件 | 系统增加该评价的持久化对象 |
| 需要的接口（需接口） | | |
| 服务名 | 服务内容 | |
| CommentUtil.getComments  (String hotelAddress) | 获得特定酒店全部评价 | |
| CommentUtil.addComment  (CommentVO vo) | 添加一条评价 | |

**CommentUtil的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| CommentUtil.getComments | | 语法 | static List<CommentVO>  getComment(String hotelAddress) |
| 前置条件 | 存在该酒店的评价 |
| 后置条件 | 返回该酒店的所有评价 |
| CommentUtil.addComment | | 语法 | static ResultMessage addComment(CommentVO vo) |
| 前置条件 | 评价输入合法 |
| 后置条件 | 系统增加该评价的持久化对象 |
| 需要的接口（需接口） | | | |
| 服务名 | 服务内容 | | |
| CommentDataService.  find(String hotelAddress) | 根据酒店地址返回多个评价的持久化对象 | | |
| CommentDataService.  insert(Com  mentPO po) | 在数据库中插入单一的评价的持久化对象 | | |

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。

## 数据层的分解

数据层模块的分解详见需求规格说明文档中的相关描述

### UserDataService模块

（1）模块概述

UserDataService模块负责保存用户账户的信息，进行增删改查的操作。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.add | 语法 | public ResultMessage add (UserPO po) throws RemoteException |
| 前置条件 | 同样的账户在数据库中不存在 |
| 后置条件 | 在数据库中增加一个UserPO记录 |
| UserDataService.findbyID | 语法 | public UserPO findbyID(String ID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的UserPO记录 |
| UserDataService.  findbyUserName | 语法 | public UserPO findbyUserName(String userName) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按UserName进行查找返回相应的UserPO记录 |
| UserDataService.modify | 语法 | public ResultMessage modify (UserPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样的PO |
| 后置条件 | 更新一个PO |
| UserDataService.Check | 语法 | public ResultMessage Check(String userName, String password) throws RemoteException |
| 前置条件 | 账号、密码合法 |
| 后置条件 | 系统检查是否存在账户账户密码是否对应 |
| UserDataService.  addCreditRecord | 语法 | public ResultMessage addCreditRecord(CreditRecordPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个CreditRecordPO记录 |
| UserDataService.  findCreditRecord | 语法 | public CreditRecordPO findCreditRecord (String ID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的CreditRecordPO记录 |
| UserDataService.  deleteUser | 语法 | public ResultMessage deleteUser(String userName) |
| 前置条件 | 该用户名正确 |
| 后置条件 | 系统删除这个用户的信息 |
| UserDataService.  addNowUser | 语法 | public void addNowUser(String userName) |
| 前置条件 | 该用户名正确 |
| 后置条件 | 系统增加这个用户已经链接到服务器的信息 |
| UserDataSerice.  deleteNowUser | 语法 | public ResultMessage deleteNowUser(String userName) |
| 前置条件 | 该用户名正确 |
| 后置条件 | 系统删除这个用户已连接到服务器的信息 |
| UserDataService.  userIsExist | 语法 | public ResultMessage userIsExist(String userName) |
| 前置条件 | 该用户名正确 |
| 后置条件 | 返回该用户是否已登陆的信息 |
| UserDataService.init | 语法 | public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| UserDataService.finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

### OrderDataService模块

（1）模块概述

OrderDataService模块负责保存各类订单，供读取修改写入。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderDataService.  getOrderPO | 语法 | public OrderPO getOrderPO(String orderID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 系统返回一条订单信息 |
| OrderDataService.  getUserOrderPO | 语法 | public List<OrderPO> getUserOrderPO (String userID, OrderType type) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回某客户某种类全部订单po |
| OrderDataService.  getHotelOrderPO | 语法 | public List<OrderPO> getHotelOrderPO (String hotelID, OrderType type) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回某酒店某种类全部订单po |
| OrderDataService.  getAbnormalOrderPO | 语法 | public List<OrderPO> getAbnormalOrderPO (Date date) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回当日全部异常订单po |
| OrderDataService.  add | 语法 | public ResultMessage add (OrderPO po) throws RemoteException |
| 前置条件 | 该订单在数据库中不存在 |
| 后置条件 | 系统返回添加结果记录 |
| OrderDataService.  modify | 语法 | public ResultMessage modify (OrderPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样订单号的po |
| 后置条件 | 更新一个po |
| OrderDataService.  getDateOrderPO | 语法 | public List<OrderPO> getDateOrderPO(String hotelID, Date date) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 无 |
| OrderDataService.init | 语法 | public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| OrderDataService.finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

### HotelDataService模块

### HotelDataService模块

（1）模块概述

HotelDataService模块负责保存酒店的信息，供读取修改写入。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelDataService.init | 语 法 | public void initial() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| HotelDataService.  addHotelInfo | 语法 | public ResultMessage  addHotelInfo(HotelPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中插入一个po |
| HotelDataService.  modifyHotelInfo | 语法 | public ResultMessage  modifyHotelInfo(HotelPO po) throws RemoteException |
| 前置条件 | 数据库中存在一个相同地址信息的po |
| 后置条件 | 在数据库中更新一个po |
| HotelDataService.  getHotelInfo | 语法 | public HotelPO getHotelInfo(String hotelID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回一个酒店po |
| HotelDataService.  findHotels | 语法 | public List<HotelPO>  findHotels(HotelFilter filter) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回多个酒店po |
| HotelDataService.getCitys | 语法 | public List<CityPO> getCitys() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中获得所有CityPO |
| HotelDataService.getAreas | 语法 | public List<AreaPO> getAreas(CityPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据CityPO返回所有AreaPO |
| HotelDataService.getRooms | 语法 | public List<RoomPO> getRooms(String hotelID) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据CityPO返回所有AreaPO |
| HotelDataService.  modifyRooms | 语法 | public ResultMessage modifyRooms(String hotelID, List<RoomPO> po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据CityPO返回所有AreaPO |
| HotelDataService.finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

### PromotionDataService模块

（1）模块概述

PromotionDataService模块负责保存促销策略，供读取修改写入。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionDataService.  add | 语法 | public ResultMessage add (PromotionPO po) throws RemoteException |
| 前置条件 | 同样的PromotionPO在数据库中不存在 |
| 后置条件 | 在数据库中增加一个PromotionPO记录 |
| PromotionDataService.  modify | 语法 | public ResultMessage modify (PromotionPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样的PO |
| 后置条件 | 更新一个PO |
| PromotionDataService.  delete | 语法 | public ResultMessage delete (String promotionID) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| PromotionDataService.  showHotelPromotion | 语法 | public List<PromotionPO> showHotelPromotion (String hotelAddress) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按hotelAddress进行查找返回该酒店的PromotionPO记录 |
| PromotionDataService.  showWebsitePromotion | 语法 | public List<PromotionPO> showWebsitePromotion () throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回网站策略的PromotionPO记录 |
| PromotionDataService.  showLevel | 语法 | public LevelPO showLevel () throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回LevelPO记录 |
| PromotionDataService.  modifyLevel | 语法 | public ResultMessage modifyLevel (LevelPO po) throws RemoteException |
| 前置条件 | 在数据库中存在同样的PO |
| 后置条件 | 更新一个PO |
| PromotionDataService.  init | 语法 | public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| PromotionDataService.  finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

### CommentDataService模块

（1）模块概述

CommentDataService模块负责保存评价信息，供读取和写入。

（2）模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommentDataService.  init | 语法 | public void initial() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| CommentDataService.  find | 语法 | public List<CommentPO>  find(String hotelAddress) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按地址进行查找返回相应的CommentPO结果 |
| CommentDataService.  insert | 语法 | public ResultMessage  insert(CommentPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中插入一个po |
| CommentDataService.  finish | 语法 | public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

# 依赖视角

下图分别为客户端和服务器端各自的包之间的联系：

