酒店预订管理系统

System of Booking Hotel

详细设计文档

V 0.1

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

目录

1 引言

1.1 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。

1.2 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

1.3 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》

2 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。

3 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。

4 结构视角

4.1 展示层的分解

4.1.1 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。

4.1.2 UI 层内部包的职责与划分

包	包内项目	
Common	Register	客户注册
	Login	登陆
Customer 客户	PersonalInfo	查看个人信息
	PersonalInfoModify	修改个人信息
	Credit	查看信用记录
	Hotel	浏览酒店
	HotelDetail	酒店详细信息
	OrderGenerate	生成订单
	OrderConfirm	生成订单确认
	UnexecutedOrder	浏览客户未执行订单
	UnexecutedOrderDetail	未执行订单详细信息
	HistoryOrder	浏览客户历史订单
	ExecutedOrderDetail	已执行订单详细信息
	CancelledOrderDetail	已撤销订单详细信息
	AbnormalOrderDetail	异常订单详细信息
	Comment	评价订单
HotelManager 酒店工作人员	HotelInfoModify	维护酒店信息
	HotelPromotion	管理酒店促销策略
	HotelPromotionModify	修改酒店促销策略
	OrderExecute	执行订单
	HistoryOrder	浏览酒店历史订单
	ExecutedOrderDetail	已执行订单详细信息

	CancelledOrderDetail	已撤销订单详细信息
	AbnormalOrderDetail	异常订单详细信息
WebMarketer 网站营销人员	CreditAdd	信用充值
	LevelModify	修改等级制度
	WebPromotion	管理网站促销策略
	WebPromotionModify	修改网站促销策略
	AbnormalOrder	浏览异常订单
	AbnormalOrderDetail	异常订单详细信息
WebManager 网站管理人员	Customer	浏览客户信息
	CustomerModify	修改客户信息
	WebMarketer	浏览网站营销人员信息
	WebMarketerMofdify	修改网站营销人员信息
	WebMarketerAdd	增加网站营销人员
	HotelManager	浏览酒店工作人员信息
	HoelrManagerModify	修改酒店工作人员信息
	HotelAdd	增加酒店

4.2 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。

4.2.1 Userbl 模块

1、模块概述：

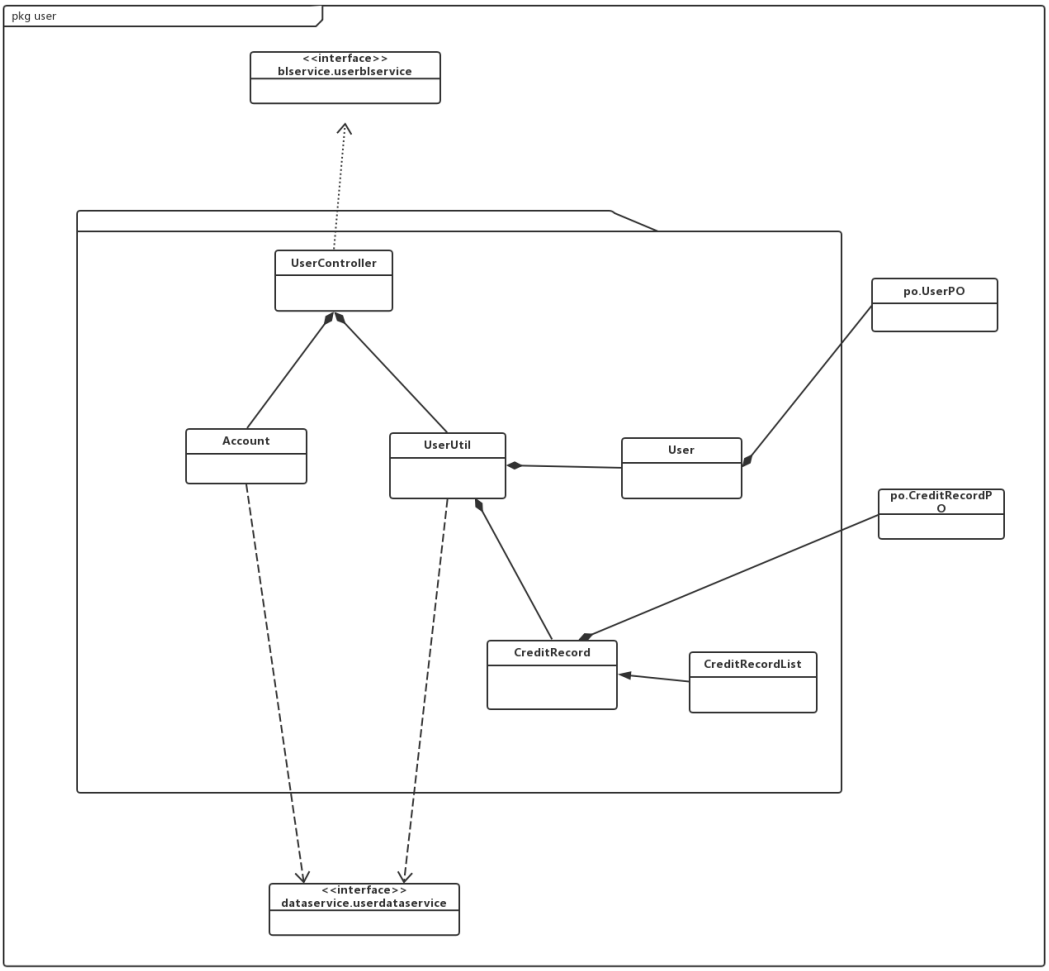
User 模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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

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	账号、密码符合语法规则
	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 UserUtil 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 CreditRecord 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
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	
User.modifyPassword(String userName, String oldPassword, String newPassword)	修改用户密码	
User.modify(UserVO vo)	修改用户信息	
CreditRecord. addCreditRecord (CreditRecordPO po)	增加一条信用记录	
CreditRecordList. findCreditRecord (String ID)	找到信用记录	

User 的接口规范

提供的服务（供接口）		
User.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	系统修改该账户的密码数据
User.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	系统修改该账户并持久化保存该账户数据
User.getUserPO	语法	public UserPo getUserPO(UserVo vo)
	前置条件	UserVO 信息完整且正确
	后置条件	无
需要的接口（需接口）		

服务名	服务内容
UserDataService.modify (UserPO po)	修改单一持久化对象
UserDataService.Check(String userName, String password)	检查账户密码是否正确

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
UserController.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
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	返回该账户所有信用
需要的接口（需接口）		
服务名	服务内容	
User.getUserPO(UserVO vo)	根据用户 vo 生成相应的用户 po	
CreditRecordList.getCreditPO（CreditRecordVO vo）	根据信用值 vo 生成相应的 po	
CreditRecordList.getCreditList（String id）	根据客户 id 得到该客户所有信用记录	
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

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	

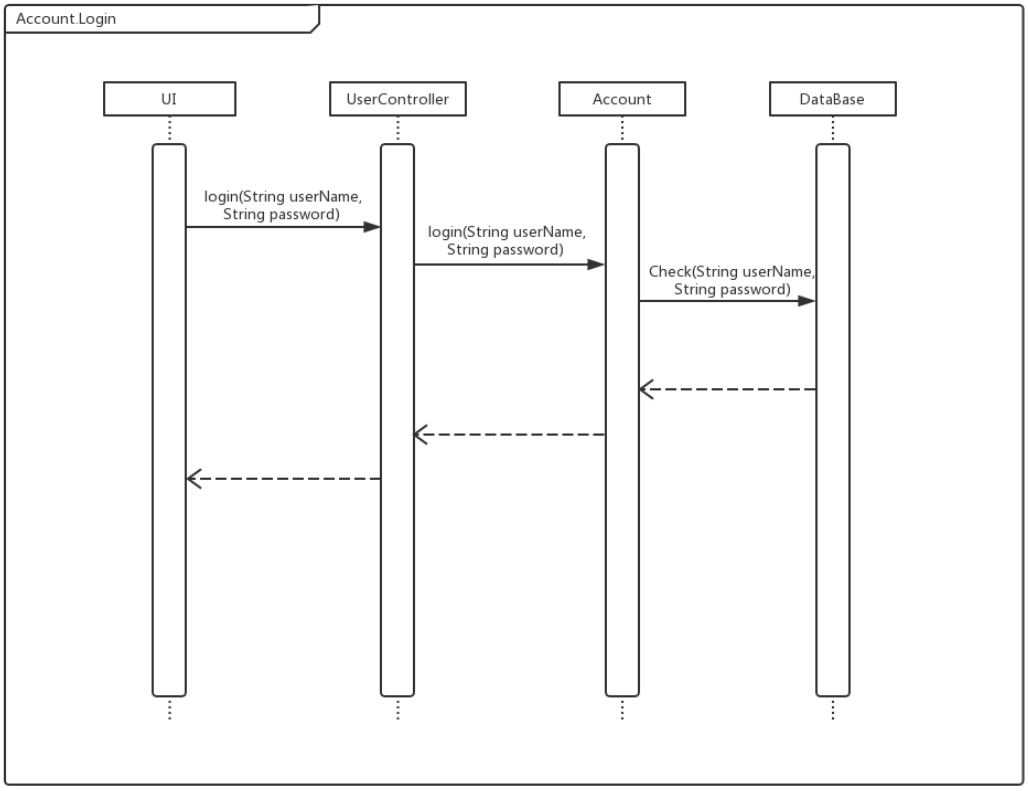
CreditRecord 的接口规范

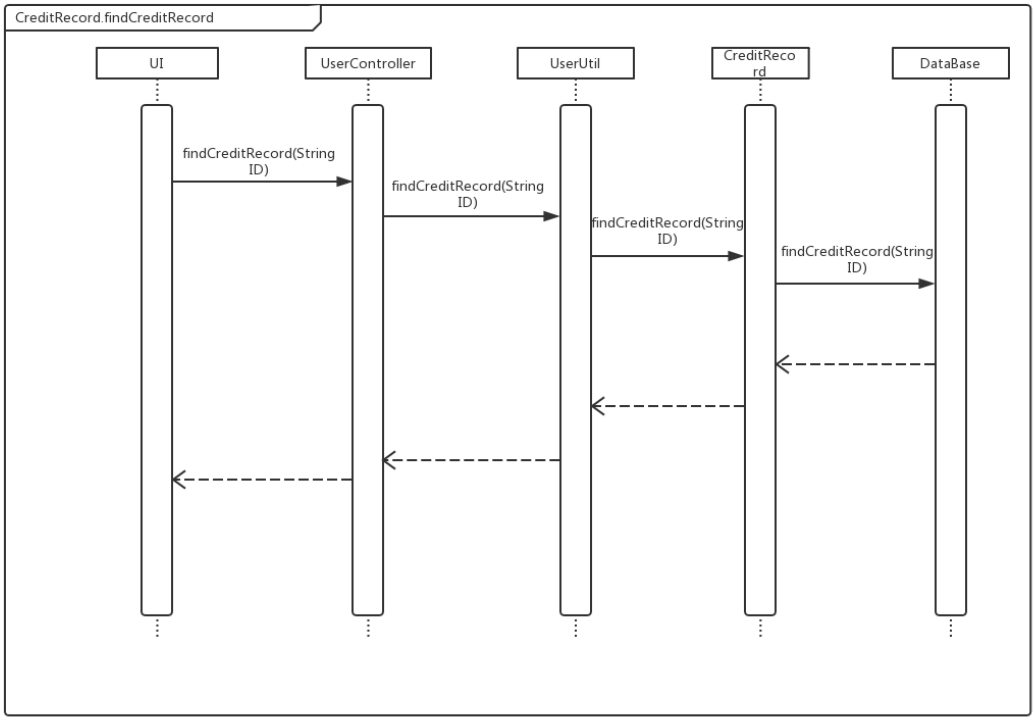
提供的服务（供接口）		
CreditRecord. getCreditRecordPO	语法	public CreditRecordPO getCreditRecordPO(CreditRecordVO vo)
	前置条件	vo 信息完整且正确
	后置条件	无
User.findCreditRecord	语法	public List<CreditRecordVO> findCreditRecord(String ID)
	前置条件	该账户 ID 为客户且输入符合语法要求
	后置条件	返回该账户所有信用
需要的接口（需接口）		
服务名	服务内容	
UserDataService. findCreditRecord	根据用户 ID 查找所有信用记录的持久化对象	

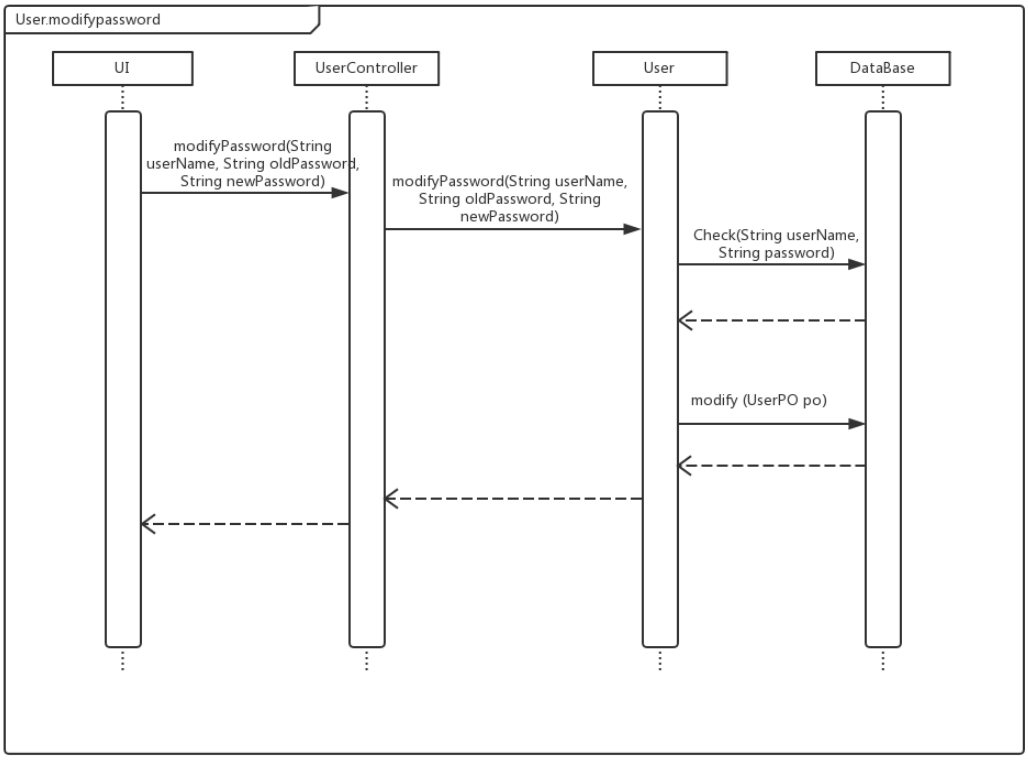
(String ID)

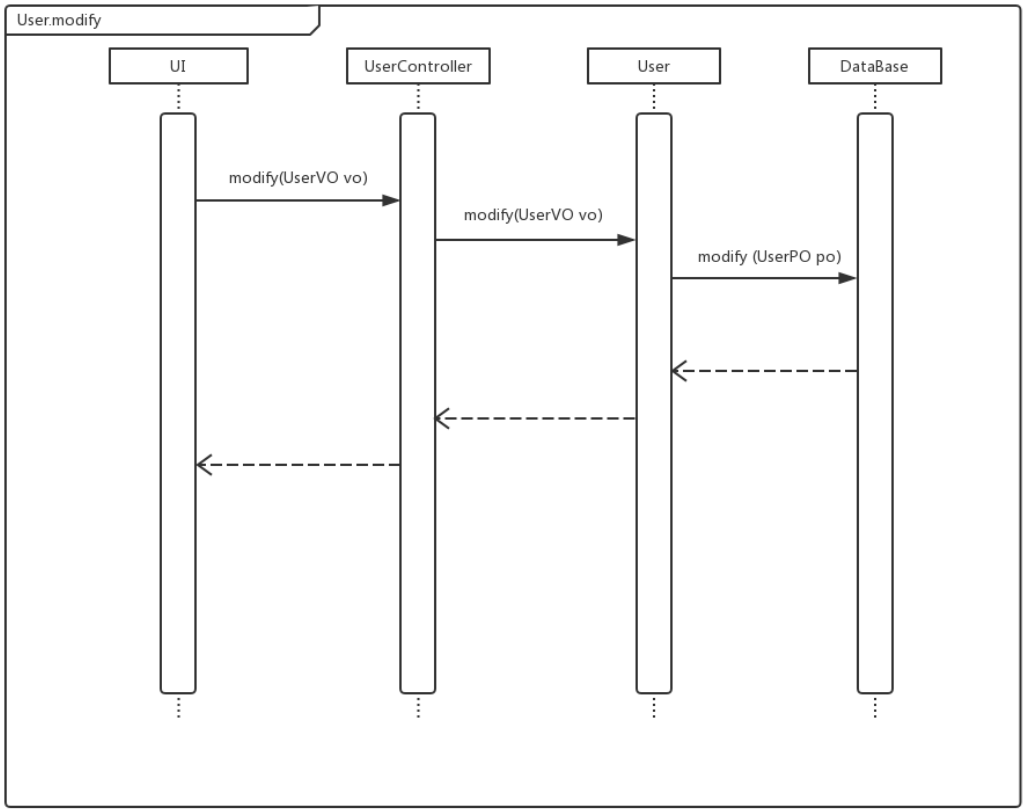
4、业务逻辑层的动态模型

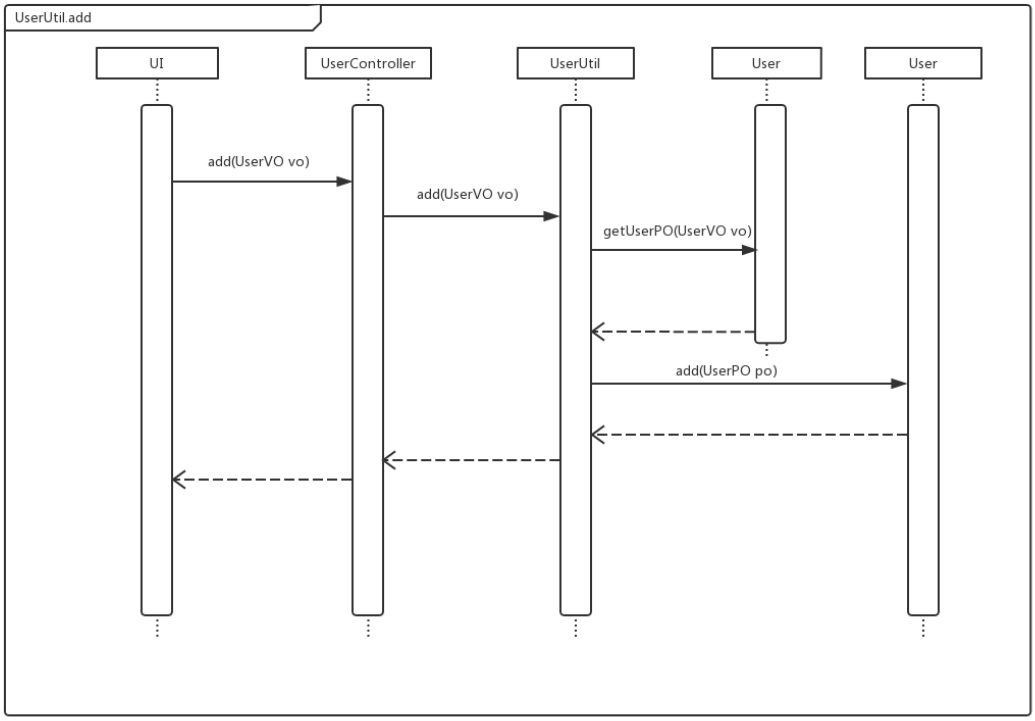
在系统账户管理中，登录、查找信用记录、修改密码、增加、修改个人信息的顺序图如下图所示：




CreditRecord.findCreditRecord
UI
UserController
UserUtil
CreditReco
rd
DataBase
findCreditRecord(String
ID)
findCreditRecord(String
ID)
findCreditRecord(String
ID)
findCreditRecord(String
ID)


User.modifypassword
UI
UserController
User
DataBase
modifyPassword(String
userName, String oldPassword,
String newPassword)
modifyPassword(String userName,
String oldPassword, String
newPassword)
Check(String userName,
String password)
modify (UserPO po)


User.modify
UI
UserController
User
DataBase
modify(UserVO vo)
modify(UserVO vo)
modify (UserPO po)


UserUtil.add
UI
UserController
UserUtil
User
User
add(UserVO vo)
add(UserVO vo)
getUserPO(UserVO vo)
add(UserPO po)

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。

4.2.2 order 模块

1、模块概述：

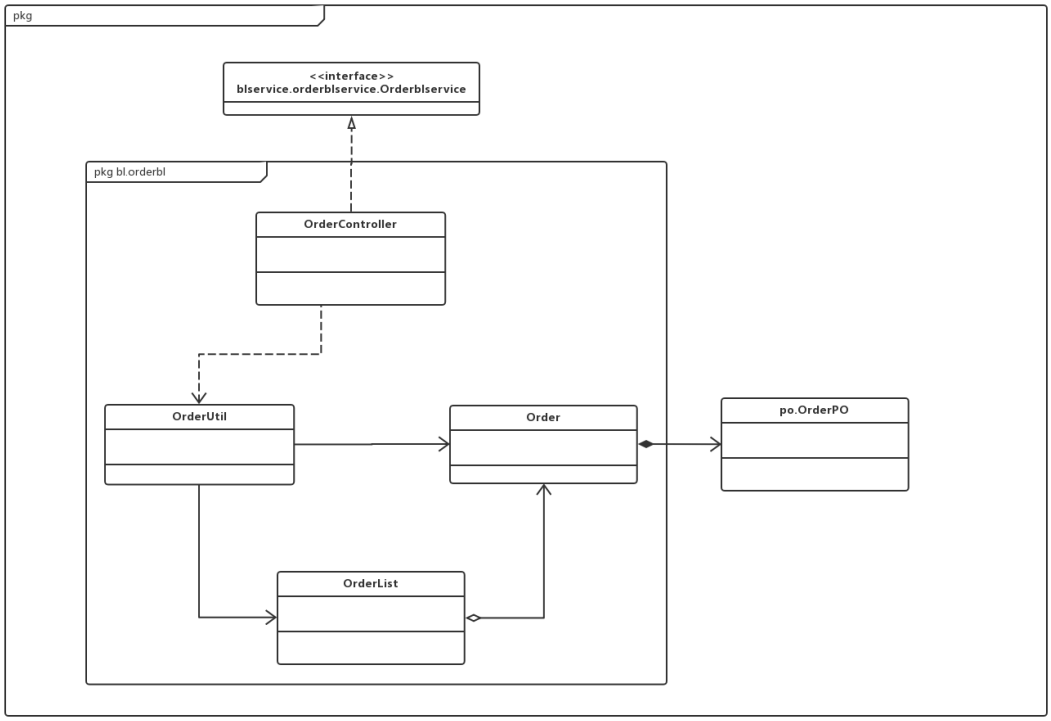
order 模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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。

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 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：



AccountBL 模块中各个类的设计

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

3、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 类的 createOrder 方法
OrderController.cancelOrder	语法	public ResultMessage cancelOrder (String ID)
	前置条件	ID 格式正确
	后置条件	调用 Order 对象的 cancelOrder 方法
OrderController.getOrder	语法	public OrderVO getOrder (String ID)
	前置条件	无
	后置条件	调用 OrderUtil 类的 getOrder 方法
OrderController. getAbnormalOrder	语法	public List<OrderVO> getAbnormalOrder ()
	前置条件	无
	后置条件	调用 OrderUtil 类的 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 类的 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 类的 getHotelOrder 方法
OrderController. executeOrder	语法	public ResultMessage executeOrder(String ID)
	前置条件	ID 格式正确
	后置条件	调用 Order 对象的 executeOrder 方法
OrderController. cancelAbnormalOrder	语法	public ResultMessage cancelAbnormalOrder (String orderID, Boolean isHalf)
	前置条件	已经存在一个 Order 对象
	后置条件	调用 Order 对象的 cancelAbnormalOrder 方法

需要的接口（需接口）	
服务名	服务内容
OrderUtil.createOrder (OrderVO vo)	增加 OrderVO
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
Order.cancelOrder()	系统实现客户撤销未执行订单
Order.executeOrder()	系统实现订单执行
Order. cancelAbnormalOrder (Boolean isHalf)	系统实现网站营销人员撤销异常订单

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 OrderDataService 的 add 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

		hoteladdress, OrderType type)
	前置条件	hoteladdress 格式正确
	后置条件	调用 OrderDataService 类的 getHotelOrderPO 方法
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	
User.findbyID (String ID)	获取一个 UserVO	
Promotion. getPromotion (String hotelAddress, String userID)	返回符合该酒店该客户的所有促销策略（包括酒店促销策略与网站促销策略）	
Hotel.getHotelVO(String hotelAddress)	返回一个酒店的 vo	
Hotel.changeRoom(RoomTypetype, int num)	系统更新酒店的可用客房信息并保存酒店的 po	
OrderDataService. add (OrderPO po)	系统添加新的订单记录	
OrderDataService. getOrderPO(String orderID)	系统返回一条订单信息	
OrderDataService. getAbnormalOrderPO (Date date)	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. 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.cancelOrder()	语法	public ResultMessage cancelOrder ()
	前置条件	已存在 Order 对象
	后置条件	调用 OrderDataSerivce 类的 modify 方法
Order.executeOrder()	语法	public ResultMessage executeOrder ()
	前置条件	已存在 Order 对象
	后置条件	调用 OrderDataSerivce 类的 modify 方法
Order. cancelAbnormalOrder (Boolean isHalf)	语法	public ResultMessage cancelAbnormalOrder (Boolean isHalf)
	前置条件	已存在 Order 对象
	后置条件	调用 OrderDataSerivce 类的 modify 方法
需要的接口（需接口）		
服务名	服务内容	
OrderDataService. modify (OrderPO po)	更新一个 po	
User.addCreditRecord (CreditRecordVO vo)	增加该客户信用记录并持久化保存数据	

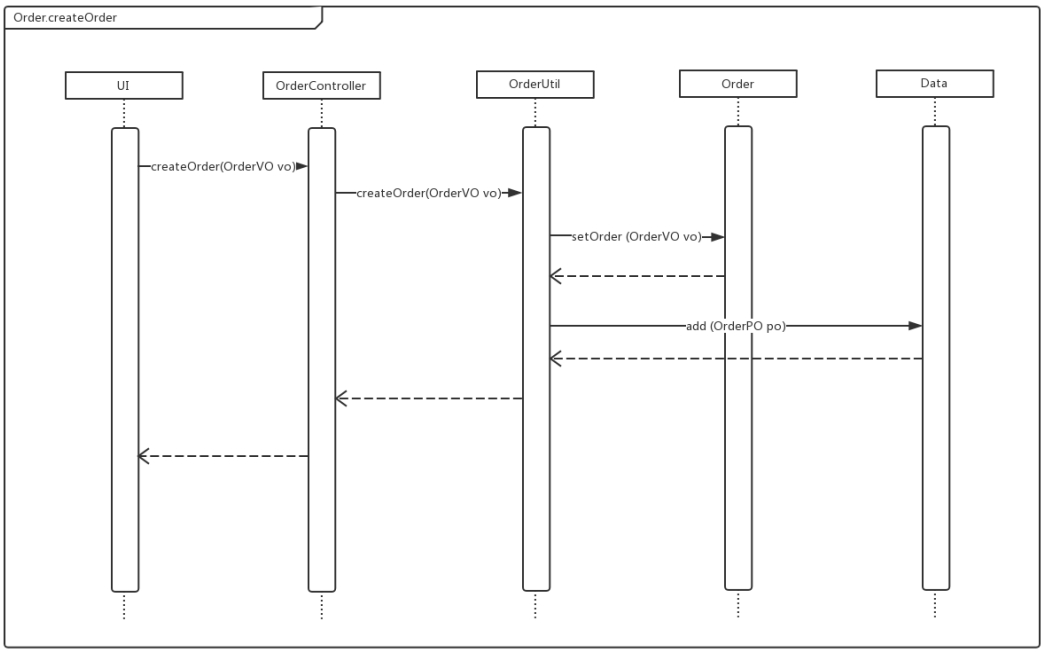
OrderList 类的接口规范

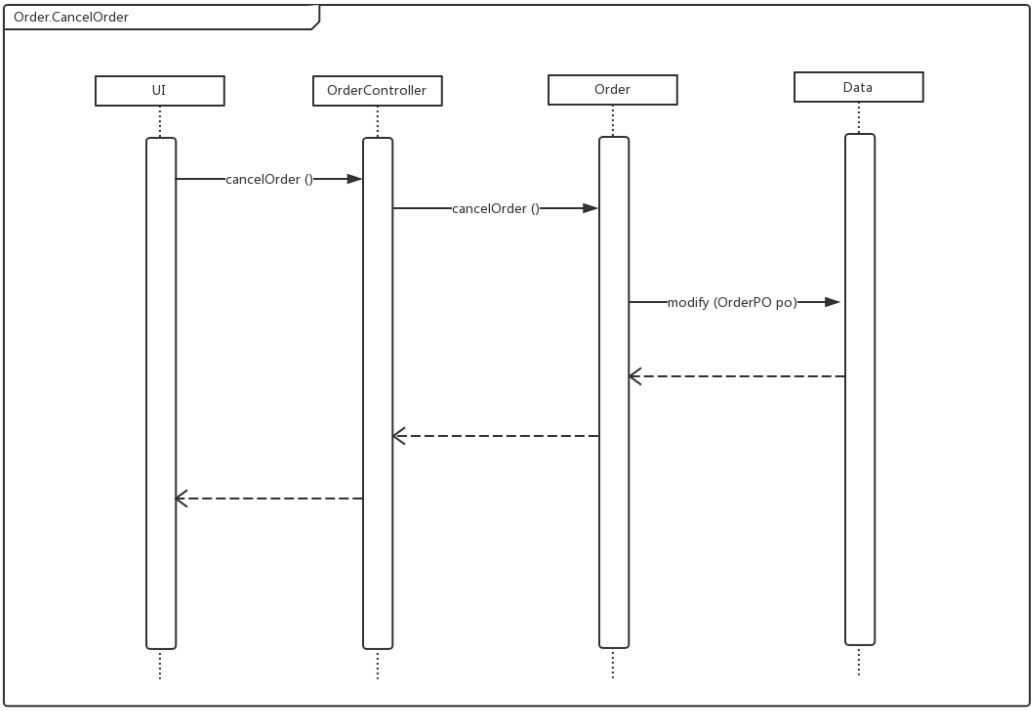
提供的服务（供接口）		
Order.setOrderList	语法	public ResultMessage setOrderList (List<OrderPO> polist)
	前置条件	已存在 OrderList 对象
	后置条件	无
Order.getOrderListVO	语法	public List<OrderVO> getOrderListVO ()
	前置条件	已存在 OrderList 对象
	后置条件	无
Order.getOrderVO	语法	public OrderVO getOrderVO (String OrderID)
	前置条件	OrderID 符合要求且存在于 List 中

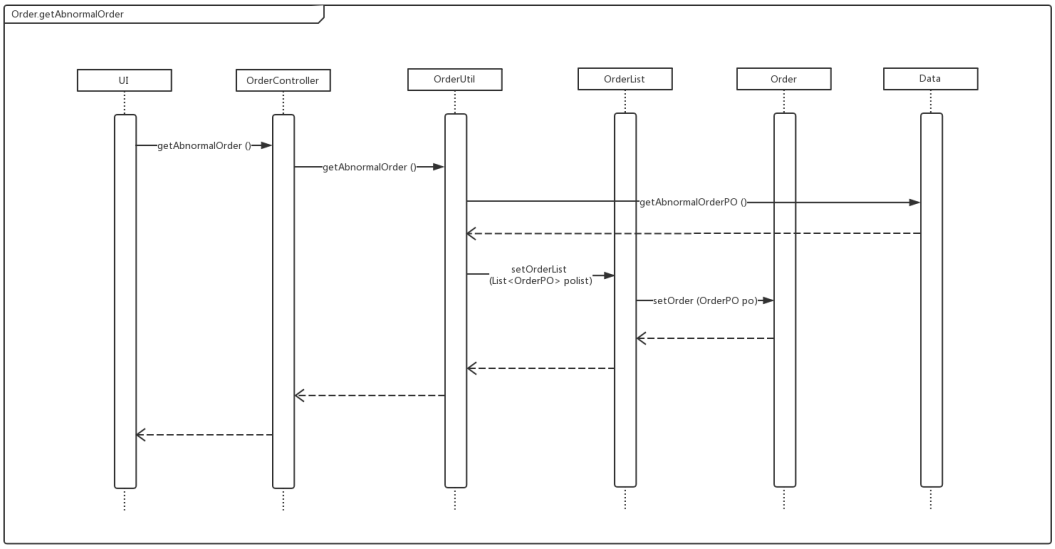
	后置条件	无
Order.getOrderPO	语法	public OrderPO getOrderPO (String OrderID)
	前置条件	OrderID 符合要求且存在于 List 中
	后置条件	无
需要的接口（需接口）		
服务名	服务内容	
Order.setOrder (OrderPO po)	通过 PO 增加 Order 对象	
Order.getOrderVO	获取 Order 的 VO 对象	

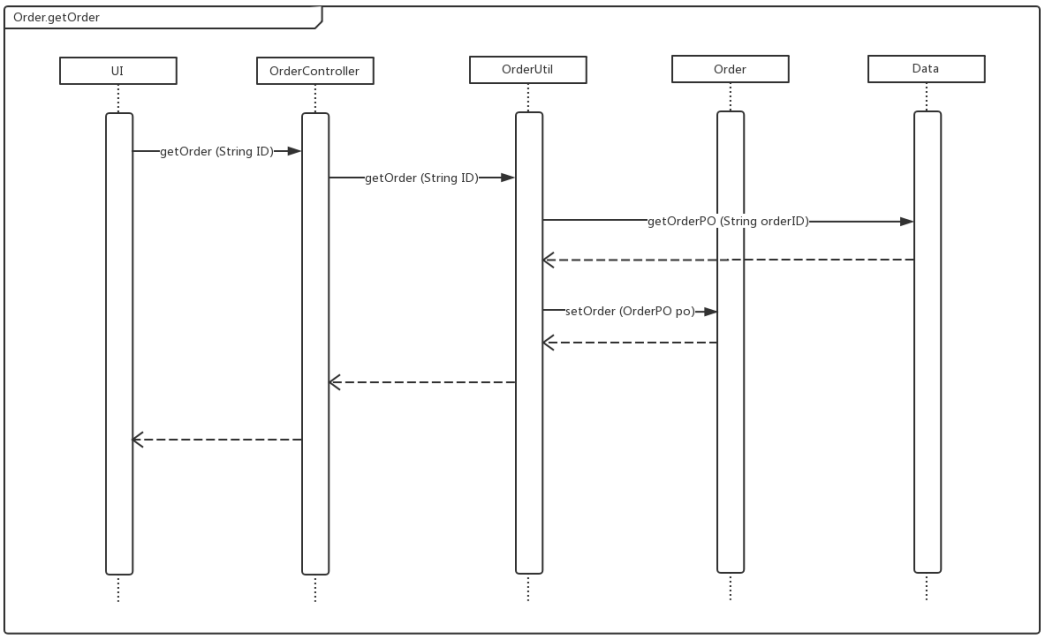
4、业务逻辑层的动态模型

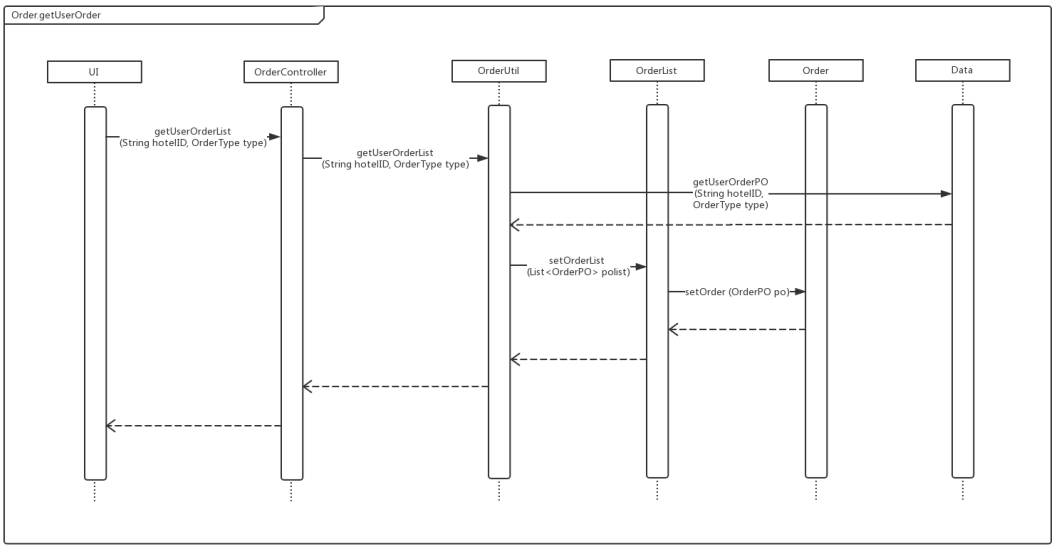
在系统订单管理中，业务分为增加、查找、撤销未执行订单与异常订单、查找账户信息，其中的顺序图如下图所示：

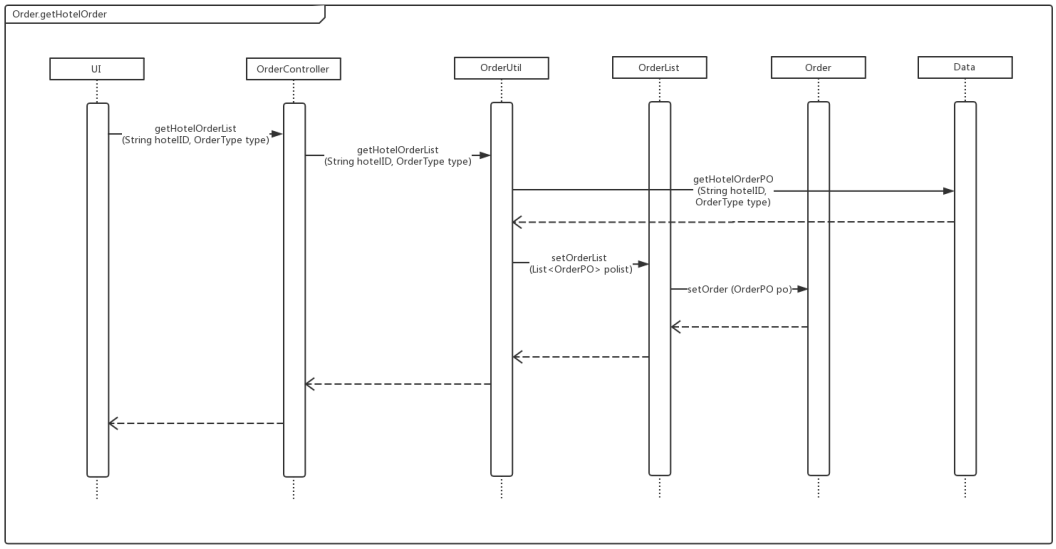


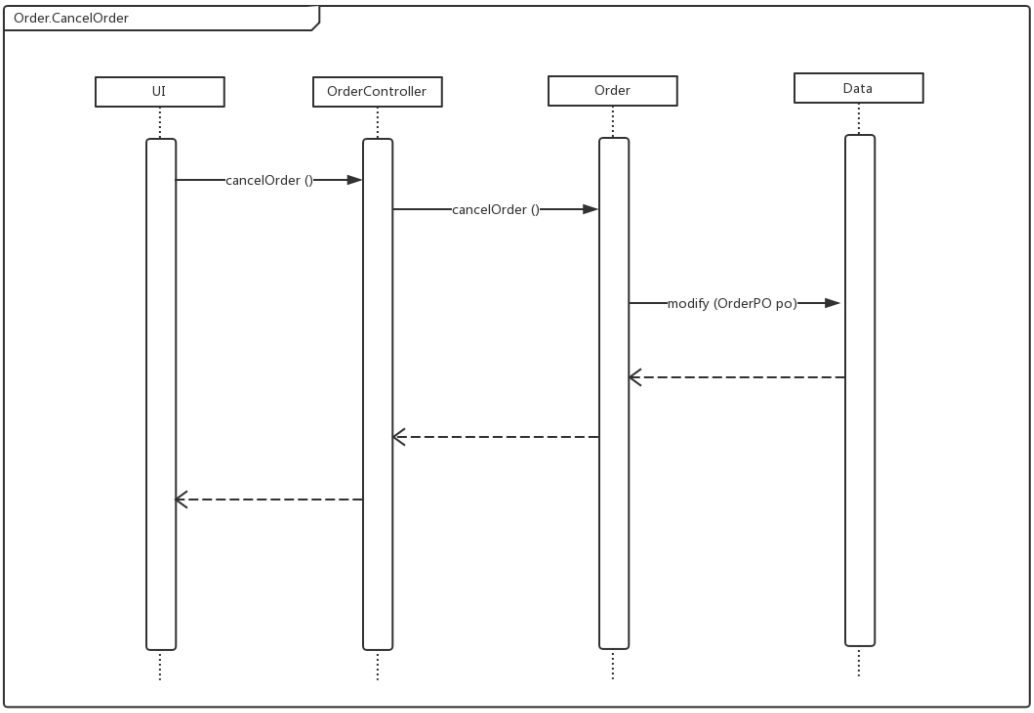

Order.CancelOrder
UI
OrderController
Order
Data
cancelOrder ()
cancelOrder ()
modify (OrderPO po)

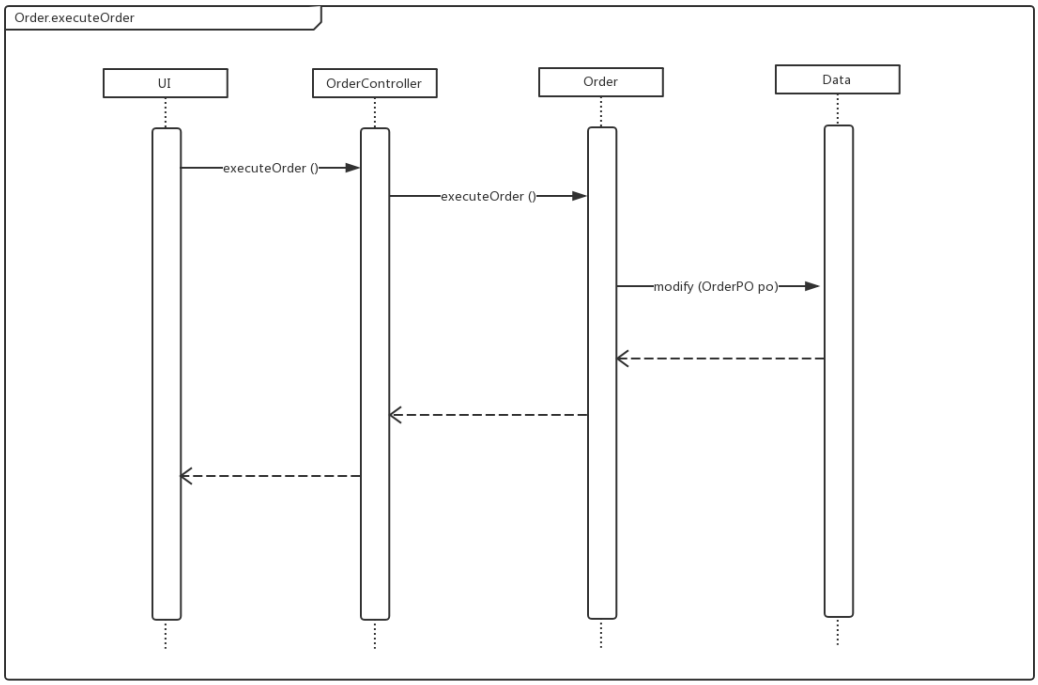

Order.getAbnormalOrder
UI
OrderController
OrderUtil
OrderList
Order
Data
getAbnormalOrder ()
getAbnormalOrder ()
getAbnormalOrderPO ()
setOrderList
(List<OrderPO> polist)
setOrder (OrderPO po)


Order.getOrder
UI
OrderController
OrderUtil
Order
Data
getOrder (String ID)
getOrder (String ID)
getOrderPO (String orderID)
setOrder (OrderPO po)


Order.getUserOrder
UI
OrderController
OrderUtil
OrderList
Order
Data
getUserOrderList
(String hotelID, OrderType type)
getUserOrderList
(String hotelID, OrderType type)
getUserOrderPO
(String hotelID,
OrderType type)
setOrderList
(List<OrderPO> polist)
setOrder (OrderPO po)


Order.getHotelOrder
UI
OrderController
OrderUtil
OrderList
Order
Data
getHotelOrderList
(String hotelID, OrderType type)
getHotelOrderList
(String hotelID, OrderType type)
getHotelOrderPO
(String hotelID,
OrderType type)
setOrderList
(List<OrderPO> polist)
setOrder (OrderPO po)


Order.CancelOrder
UI
OrderController
Order
Data
cancelOrder ()
cancelOrder ()
modify (OrderPO po)



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。

4.2.3 Hotelbl 模块

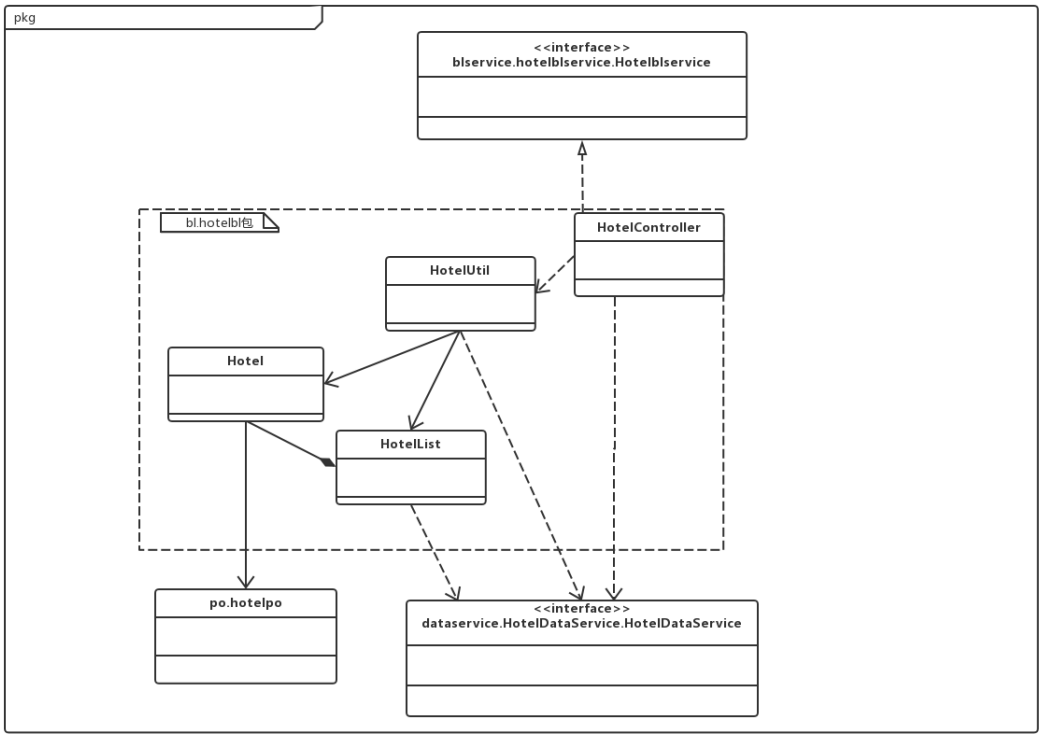
1、模块概述：

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 模块各个类的设计

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)
	前置条件	无
	后置条件	调用 HotelList 类的 getHotelList 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

HotelController.modifyHotel	语法	public ResultMessage modifyHotel(HotelVO vo)
	前置条件	已经创建一个 Hotel 领域对象，且输入符合格式
	后置条件	调用 Hotel 类的 modify 方法
HotelController.changeRoom	语法	public ResultMessage changeRoom(RoomType type, int num)
	前置条件	已经创建一个 Hotel 领域对象，且输入符合格式
	后置条件	调用 Hotel 类的 modify 方法
HotelController.getHotelVO	语法	public HotelVO getHotelVO(String hotelAddress)
	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	
Promotion.getPromotion	根据酒店地址和客户 ID 查找促销策略	
HotelUtil.addHotel	增加酒店的持久化对象	
HotelUtil.modify	修改酒店的领域对象并返回修改后的 po	
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	
User.add	增加酒店工作人员的持久化对象	
Comment.getComments	根据酒店地址返回该酒店的所有评论的持久化对象	
HotelDataService.modifyHotelInfo	修改酒店的持久化对象	
HotelDataService.getCitys	在数据库中获得所有 CityPO	
HotelDataService.getAreas	根据 CityPO 返回所有 AreaPO	

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)
	前置条件	无
	后置条件	创建并返回一个酒店领域类对象
Hotel.modify	语法	public HotelPO modify(HotelVO vo)
	前置条件	无
	后置条件	返回一个修改后的酒店的 po

需要的服务（需接口）	
服务名	服务

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	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
HotelDataService.findHotels	返回多个酒店 po

HotelUtil 的接口规范

提供的服务（供接口）		
HotelUtil.addHotel	语法	public ResultMessage addHotel

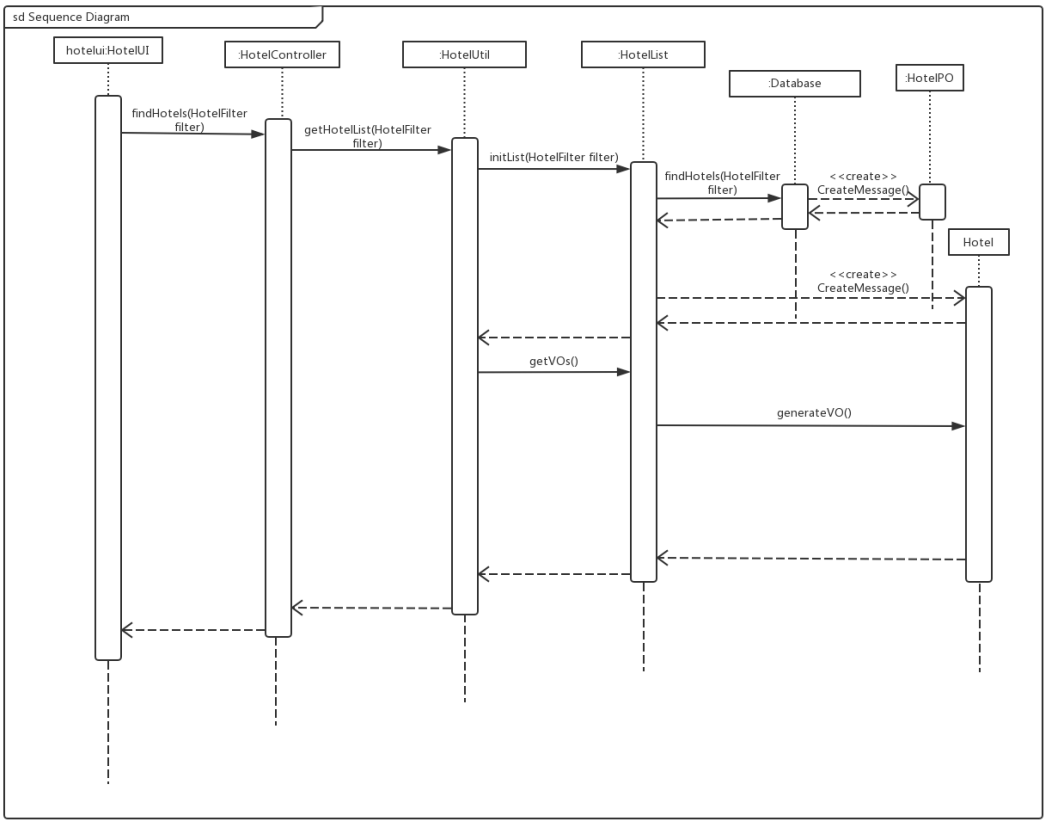
		(HotelVO vo)
	前置条件	无
	后置条件	调用 HotelDataService 类的 addHotelInfo 方法
HotelUtil.getHotel	语法	public HotelVO getHotel(String hotelAddress)
	前置条件	输入符合格式
	后置条件	返回一个酒店的 vo
HotelUtil.getHotelList	语法	public List<HotelVO> getHotelList (HotelFilter filter)
	前置条件	无
	后置条件	调用 HotelList 类的 initList 和 getVOs 方法，系统返回 HotelVO 的列表

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
Hotel.modify	修改领域对象并返回一个 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。

4.2.4 Promotion 模块

1、模块概述：

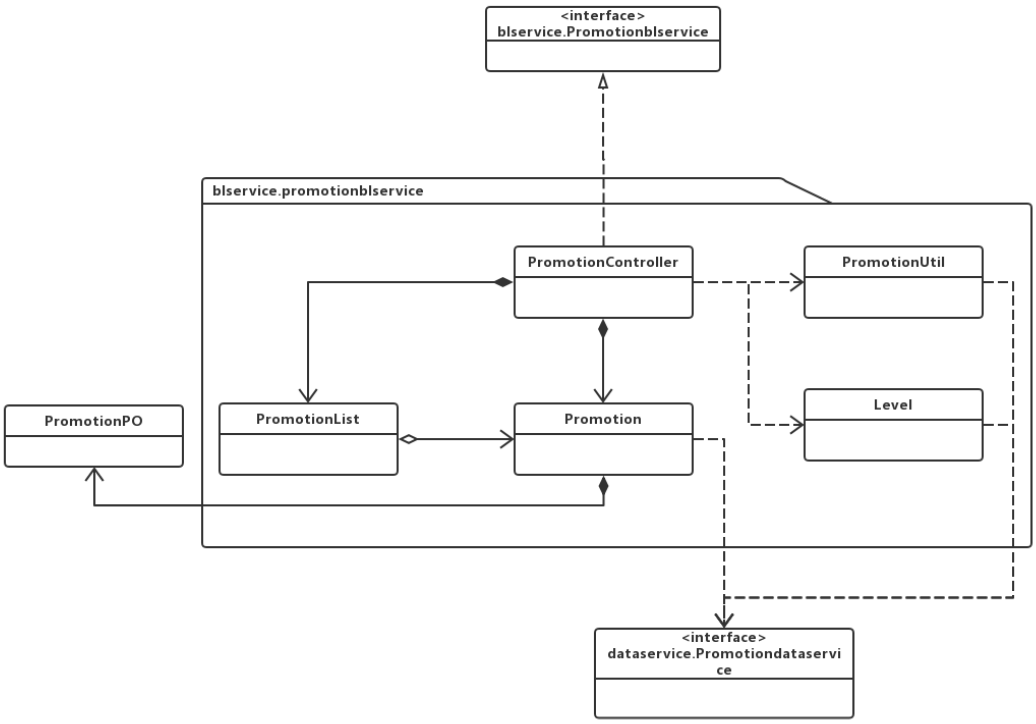
Promotion 模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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。

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 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。PromoionList 对象在需要大量 Promotion 对象时生成，是 Promotion 的容器类。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 是用户等级制度的枚举类。

Promotionbl 模块的设计如图：



PromotionBL 模块中各个类的设计

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

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.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	系统持久化增加该促销策略数据
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）
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	系统根据酒店地址返回该酒店的所有促销策略
PromotionController.showWebsitePromotion	语法	public List<PromotionVO> showWebsitePromotion()
	前置条件	无
	后置条件	系统返回所有网站促销策略
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	系统修改该策略并持久化存储该策略数据
PromotionController.showLevel	语法	public LevelVO showLevel ()
	前置条件	无
	后置条件	系统返回用户等级信息
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	系统修改用户等级策略并持久化存储该策略数据

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.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

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.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<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	系统根据酒店地址返回该酒店的所有促销策略
PromotionUtil.showWebsitePromotion	语法	static List<PromotionVO> showWebsitePromotion()
	前置条件	无
	后置条件	系统返回所有网站促销策略
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	

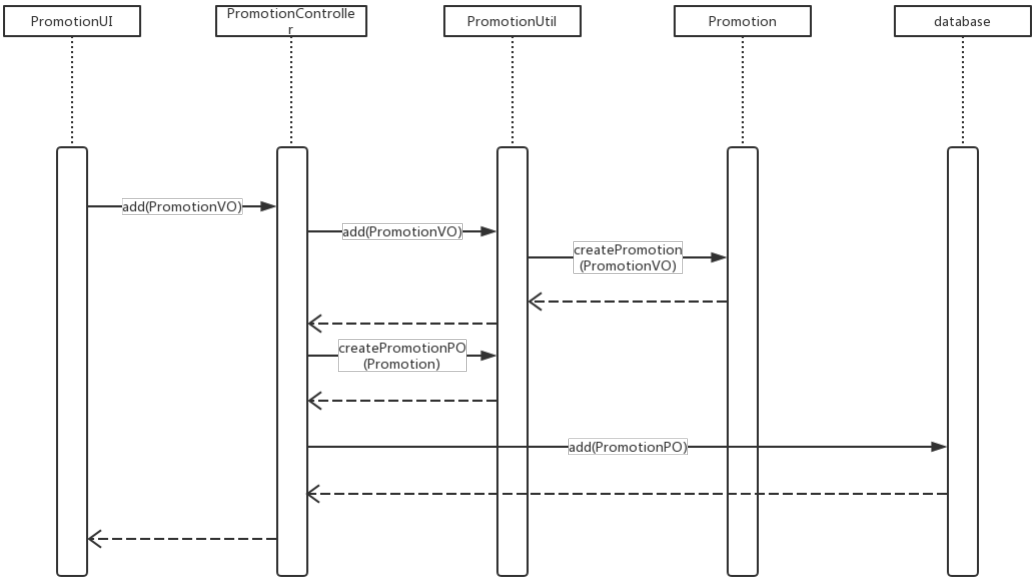
PromotionDataService.showHotelPromotion(String hotelAddress)	根据 hotelAddress 查找该酒店的所有持久化对象
PromotionDataService.showWebsitePromotion()	查找所有网站营销策略

Level 的接口规范

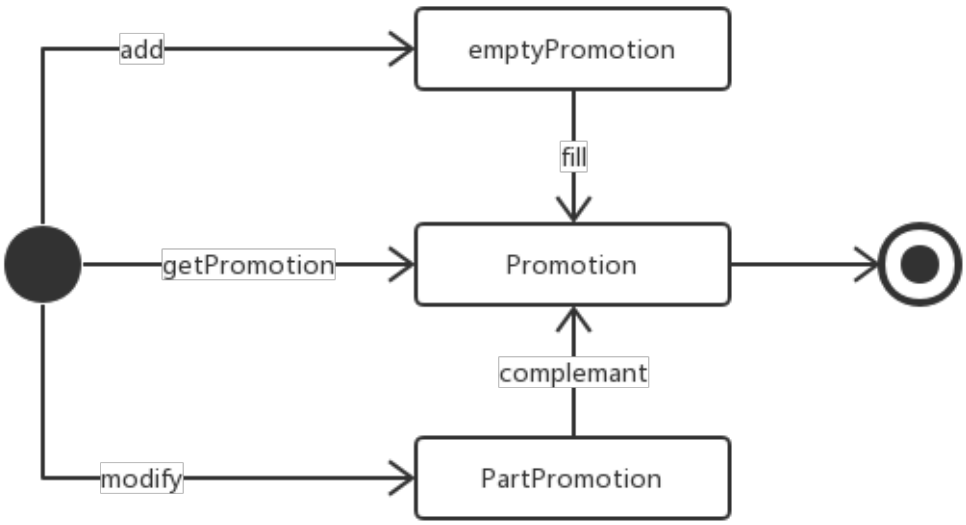
提供的服务（供接口）		
Level.showLevel	语法	LevelVO showLevel ()
	前置条件	无
	后置条件	系统返回用户等级信息
Level.modifyLevel	语法	ResultMessage modifyLevel (LevelVO vo)
	前置条件	输入的内容符合规范且齐全
	后置条件	系统修改用户等级策略并持久化存储该策略数据
需要的接口（需接口）		
服务名	服务内容	
PromotionDataService.showLevel ()	返回用户等级策略的持久化对象	
PromotionDataService.modifyLevel (LevelPO po)	修改单一用户等级策略的持久化对象	

4、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。



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。

4.2.5 Comment 模块

1、模块概述：

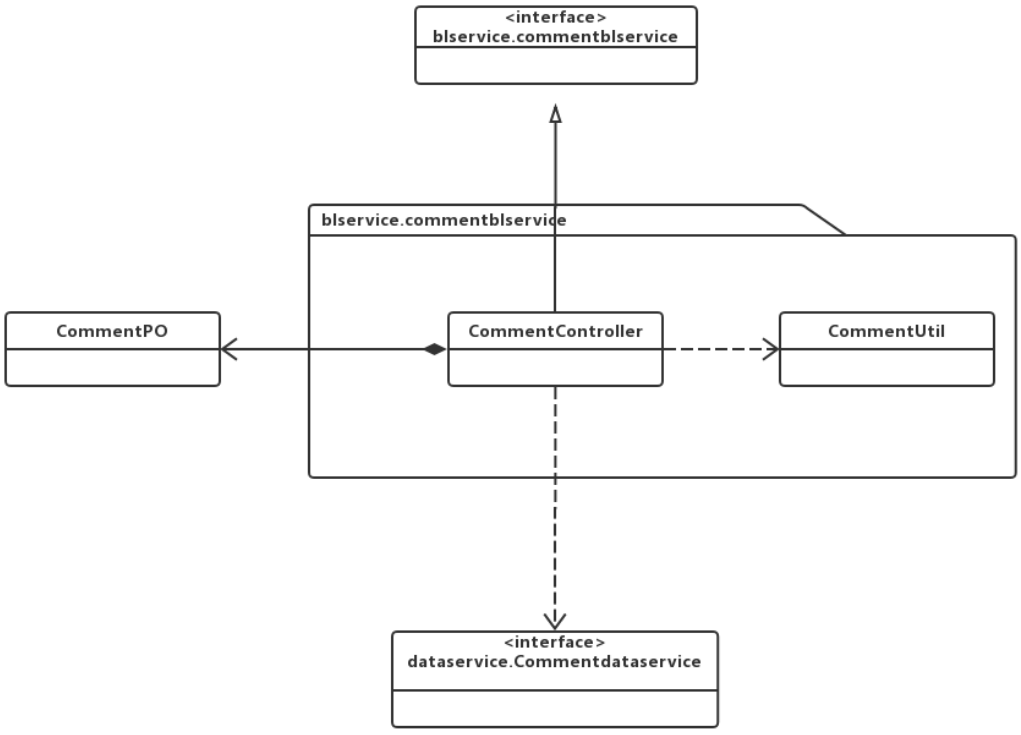
Comment 模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

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。

Approval 模块中各个类的设计：



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

3、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(CommentPO po)	在数据库中插入单一的评价的持久化对象	

4、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。

4.3 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

4.3.1 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.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

4.3.2 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.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

4.3.3 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	在数据库中添加一个 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标志的 po
	后置条件	在数据库中更新一个 po
HotelDataService.	语法	public HotelPO getHotelInfo(String

getHotelInfo		hotelAddress) 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	在数据库找到所有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.finish	语法	public void finish() throws RemoteException
	前置条件	无
	后置条件	结束对数据的使用

4.3.4 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.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

4.3.5 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。

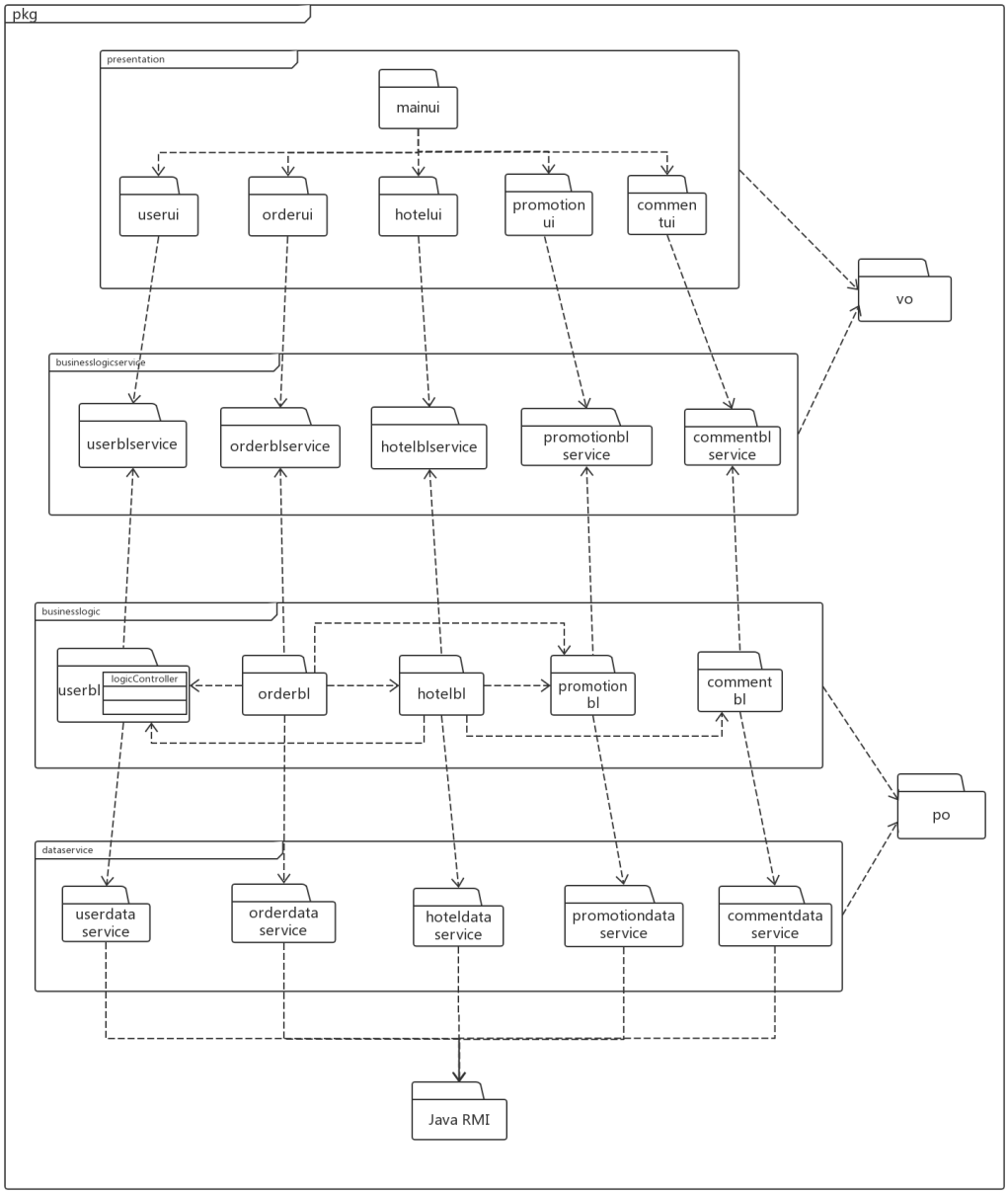
（2）模块内部接口规范

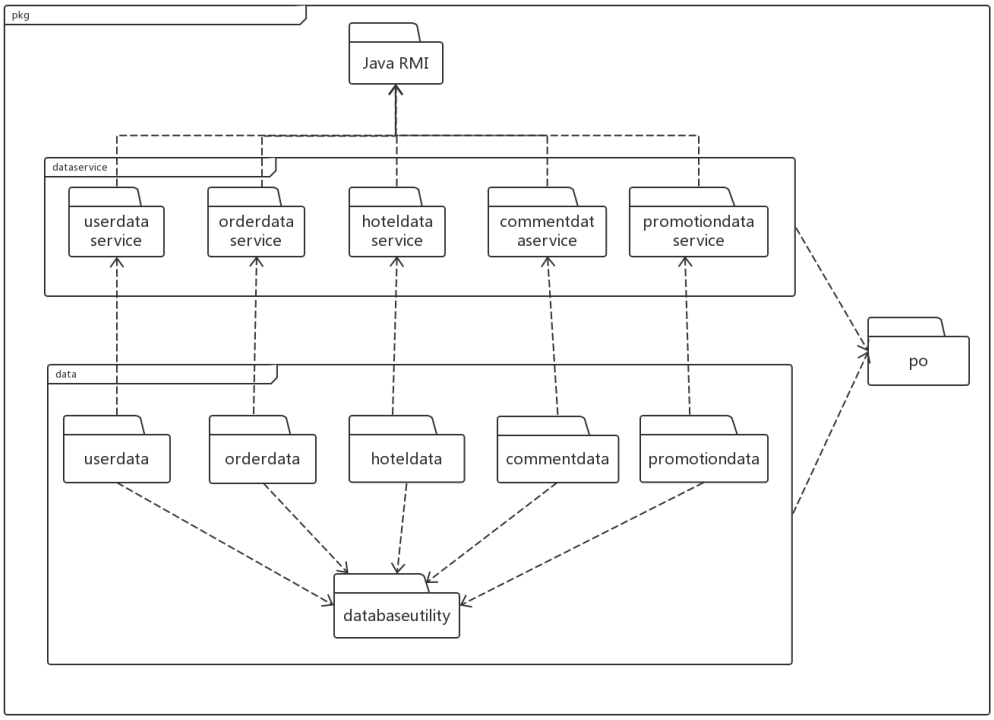
提供的服务（供接口）		
CommentDataService. init	语法	public void initial() throws RemoteException
	前置条件	无
	后置条件	初始化持久化数据库
CommentDataService. find	语法	public List<CommentPO> find(String hotelAddress) throws RemoteException

	前置条件	无
	后置条件	按地址查找返回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	结束对数据的使用

5 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：


pkg
presentation
mainui
userui
orderui
hotelui
promotion ui
commen tui
vo
businesslogicservice
userblservice
orderblservice
hotelblservice
promotionbl service
commentbl service
businesslogic
userbl
logicController
orderbl
hotelbl
promotion bl
comment bl
po
dataservice
userdata service
orderdata service
hoteldata service
promotiondata service
commentdata service
Java RMI


pkg
Java RMI
dataservice
userdata service
orderdata service
hoteldata service
commentdat aservice
promotiondata service
po
data
userdata
orderdata
hoteldata
commentdata
promotiondata
databaseutility