## strategy模块静态结构和动态行为

### 1.1.1 strategy模块概述

Strategy模块承担优惠策略的制定，对价格的计算方式的产生影响的功能。

### 1.1.2 strategy模块局部模块的职责

### 1.1.2.1结构视角

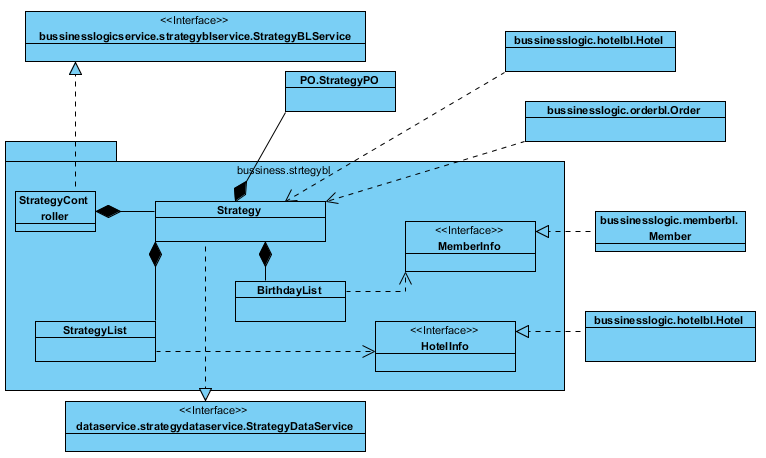
（1）模块概述

strategy模块承担的需求见需求规格说明书功能需求及相关非功能需求。

strategy模块的职责及接口参见软件体系结构描述文档表。

（2）整体结构

根据体系结构的设计，采用分层风格，将系统分为展示层、业务逻辑层、数据层。每一层之间未来灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了strategyblservice，strategydataservice两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了strategyController，这样strategyController将会将优惠策略相关的业务逻辑职责和逻辑控制委托给strategy对象。StrategyPO是作为优惠策略的持久化对象被添加到设计模型中的。StrategyList和StrategyLineItem的添加是RoomInfo的容器类。SaleLineItem保有房间的原始价格和折扣，以及计算当前价格的职责。MemberInfo和HotelInfo都是根据依赖倒置原则，为了消除循环依赖而产生的接口。



|  |  |
| --- | --- |
| 类 | 职责 |
| StrategyController | 负责优惠策略制定界面所需要的服务 |
| Strategy | 优惠策略的领域模型对象 |
| BirthdayList | 负责暂时存储该酒店下的所有会员名单 |
| StrategyList | 优惠策略下的客房信息集合的领域模型对象 |

（3）模块内部类的接口规范

StrategyController 、Strategy、StrategyLineItem和StrategyList的接口规范如以下所示。

表1.1.2.1-1 StrategyController的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| StrategyController.setMemberBirth | 语法 | | Public void setMemberBirth(long memberID) |
| 前置条件 | | Strategy领域对象已创建，优惠策略采用会员生日式。 |
| 后置条件 | | 调用Strategy领域对象的setMemberBirth方法 |
| StrategyController.setRoomInfo | 语法 | | Public ResultMessage setRoomInfo(long hotelID) |
| 前置条件 | | 已创建一个Strategy领域对象的addRoomInfo方法 |
| 后置条件 | | 调用Strategy领域对象的addRoomInfo方法 |
| StrategyController.deleteRoomInfo | 语法 | | Public ResultMessage deleteRoomInfo(long hotelID,filed roomStyle) |
| 前置条件 | | 已创建一个Strategy领域对象的deleteRoomInfo方法 |
| 后置条件 | | 调用Strategy领域对象的deleteRoomInfo方法 |
| StrategyController.addRoomInfo | 语法 | | Public ResultMessage addRoomInfo(long hotel,filed roomStyle) |
| 前置条件 | | 已创建一个Strategy领域对象的addRoomInfo方法 |
| 后置条件 | | 调用Strategy领域对象的addRoomInfo方法 |
| StrategyController.judgeDate | 语法 | | Public Boolean judgeDate(long hotel,field style,long memberID  ) |
| 前置条件 | | 已创建一个Strategy领域对象的judgeDate方法 |
| 后置条件 | | 调用Strategy领域对象的judgeDate方法 |
| StrategyController.changeRoomInfo | 语法 | | Public ResultMessage changeRoomInfo(long hotelID,filed roomStyle) |
| 前置条件 | | 已创建一个Strategy领域对象的changeRoomInfo方法 |
| 后置条件 | | 调用Strategy领域对象的changeRoomInfo方法 |
| StrategyController.getRoomInfo | 语法 | | Public RoomVO getRoomInfo(long hotelID) |
| 前置条件 | | 已创建一个Strategy领域对象的getRoomInfo对象 |
| 后置条件 | | 调用StrategyController 领域对象的getRoomInfo对象 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| Strategy.setMemberBirth | | | 得到会员生日信息 |
| Strategy.setRoomInfo | | | 初始化含有优惠策略的所有类型房间信息 |
| Strategy.deleteRoomInfo | | | 删除某一不含优惠策略的类型的房间信息 |
| Strategy.addRoomInfo | | | 增加某一含有优惠策略的类型的房间信息 |
| Strategy.judgeDate | | | 判断是否在策略时间区间内 |
| Strategy.changeRoomInfo | | | 改变某一类型房间的策略、价格信息 |
| Strategy.getRoomInfo | | | 获得某一类型的信息，包含策略信息 |

表1.1.2.1-2 Strategy的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Strategy.setMemberBirth | 语法 | Public ResultMessage setMemberBirth(long memberID) |
| 前置条件 | 获取到会员信息 |
| 后置条件 | 获取并存储所有会员的生日信息 |
| Strategy.setRoomInfo | 语法 | Public ResultMessage setRoomInfo(long hotelID) |
| 前置条件 | 进入酒店的策略制定 |
| 后置条件 | 获取酒店所有含有销售策略的房间信息 |
| Strategy.deleteRoomInfo | 语法 | Public ResultMessage deleteRoomInfo(long hotelID,filed roomStyle) |
| 前置条件 | 改类型的客房存在优惠销售策略 |
| 后置条件 | 将此类型客房的优惠销售策略去除，并从列表中删除 |
| Strategy.addRoomInfo | 语法 | Public ResultMessage addRoomInfo(long hotel,filed roomStyle) |
| 前置条件 | 此类型的客房不存在优惠销售策略 |
| 后置条件 | 设置此类型客房的优惠销售策略，并添加到列表中 |
| Strategy.judgeDate | 语法 | Public Boolean judgeDate(long hotel,field style,long memberID) |
| 前置条件 | 此客房的存在优惠策略且已明确优惠策略类型 |
| 后置条件 | 判断目前时间是否处在优惠时间段中并返回判断值 |
| Strategy.changeRoomInfo | 语法 | Public ResultMessage changeRoomInfo(long hotelID,filed roomStyle) |
| 前置条件 | 此类型的客房存在优惠策略 |
| 后置条件 | 更改此类型客房的优惠策略 |
| Stategy.getRoomInfo | 语法 | Public List< RoomVO >getRoomInfo(long hotelID) |
| 前置条件 | 存在此酒店的优惠销售策略客房信息 |
| 后置条件 | 返回此酒店的所有优惠类型客房信息 |
| Strategy.calcualtePrice | 语法 | Public int calculate<int price ,double discount> |
| 前置条件 | 此类型存在优惠策略 |
| 后置条件 | 返回打折后的客房价格 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| Hotel.getMemberInfo(long hotelID) | | 获得酒店所有会员信息的持久化对象 |
| Hotel.getRoomInfo() | | 获得酒店所有的客房信息的持久化对象 |
| Hotel.getRoomInfo(filed style) | | 获得酒店特定类型的客房信息的持久化对象 |
| StrategyDataService.save(long hotelID) | | 存储酒店所有存在优惠策略的客房类型信息的持久化对象 |
| StrategyDataService.insert(field style) | | 插入一条存在优惠策略的客房信息持久化对象 |

表1.1.2.1-3 BirthdayList的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BirthdayList.setBirthdayList | 语法 | Public void setBirthdayList(List<MemberInfo> lst) |
| 前置条件 | 已选择生日策略 |
| 后置条件 | 建立完会员生日列表 |
| BirthdayList.getMemberBirth | 语法 | Public Date getMemberBirth(long memberID) |
| 前置条件 | 会员存在 |
| 后置条件 | 返回会员的生日 |
| BirthdayList.insertMemberBirth | 语法 | Public ResultMessage insertMemberBirth(MemberInfo info) |
| 前置条件 | 此会员的生日记录不存在 |
| 后置条件 | 插入一条生日记录 |
| 需要的服务（需接口） | | |
| StrategyDataService.saveBirthday | | 存储列表中的所有生日持久化对象 |
| Hotel.getMemberBirth | | 获得酒店所有的会员生日记录 |

表1.1.2.1-4 StrategyList的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyList.setRoomInfo | 语法 | Public ResultMessage setRoomInfo(long hotelID) |
| 前置条件 | 进行酒店优惠策略制定 |
| 后置条件 | 初始化酒店所有的客房优惠策略 |
| StrategyList.deleteRoomInfo | 语法 | Public ResultMessage deleteRoomInfo(long hotelID,filed style) |
| 前置条件 | 存在此类客房的优惠策略记录 |
| 后置条件 | 删除优惠策略列表中此类客房的记录 |
| StrategyList.addRoomInfo | 语法 | Public ResultMessage addRoomInfo(long hotelID,field style) |
| 前置条件 | 不存在此类客房的优惠策略记录 |
| 后置条件 | 添加优惠策略了表中此类客房的记录 |
| StrategyList.chageRoomInfo | 语法 | Public ResultMessage changeRoomInfo(long hotelID,field style) |
| 前置条件 | 存在此类客房的优惠策略记录 |
| 后置条件 | 修改此类客房的优惠策略记录 |
| StrategyList.saveRoomInfo | 语法 | Public ResultMessage saveRoomInfo() |
| 前置条件 | 结束任务 |
| 后置条件 | 存储信息 |
| StrategyList.getRoomInfo | 语法 | Public RoomInfo getRoomInfo(long hotelID,filed style) |
| 前置条件 | 存在此类客房的优惠策略记录 |
| 后置条件 | 返回此类客房的优惠策略记录 |
| 需要的服务（需接口） | | |
| Hotel.getRoomInfoList | | 得到酒店的所有类型的客房信息 |
| Hotel.getRoomInfo | | 得到酒店的特定类型的客房信息 |
| StrategyDataService.saveRoomInfo | | 存储所有改变策略的或添加策略的客房类型持久化 |

（4）动态模型：

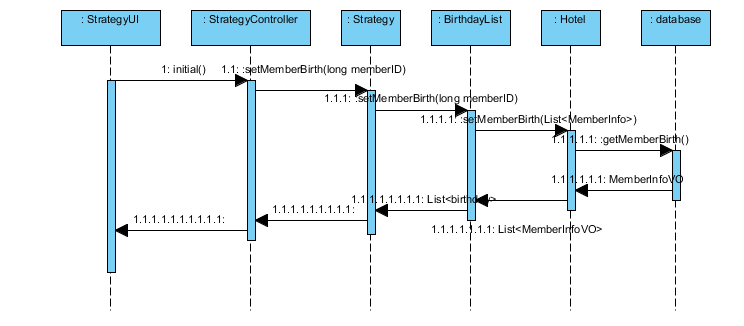


图1.1.2.1-1会员生日列表初始化顺序图

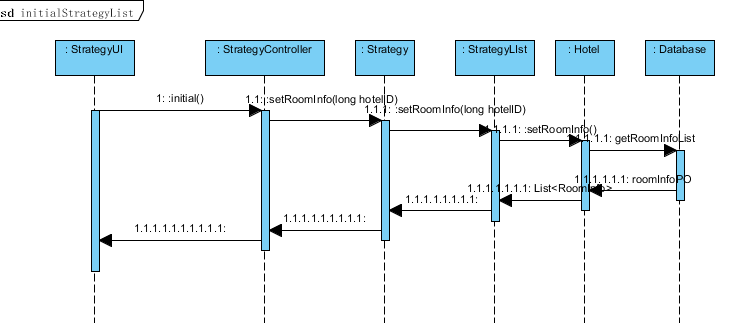


图1.1.2.1-2Strategy的策略列表初始化顺序图

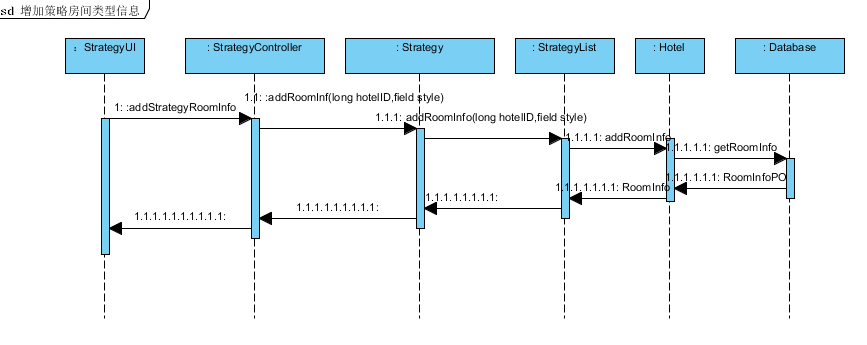


图1.1.2.1-3增加策略房间类型信息顺序图

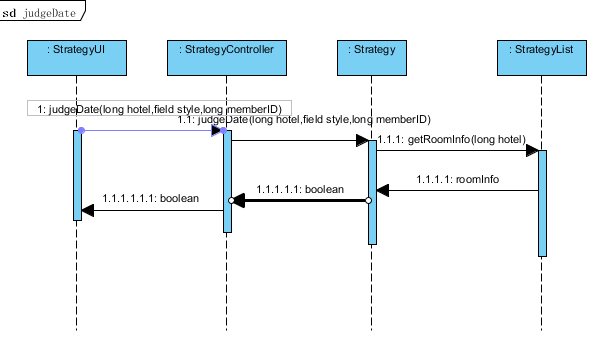


图1.1.2.1-4判断是否在优惠时间区间内顺序图

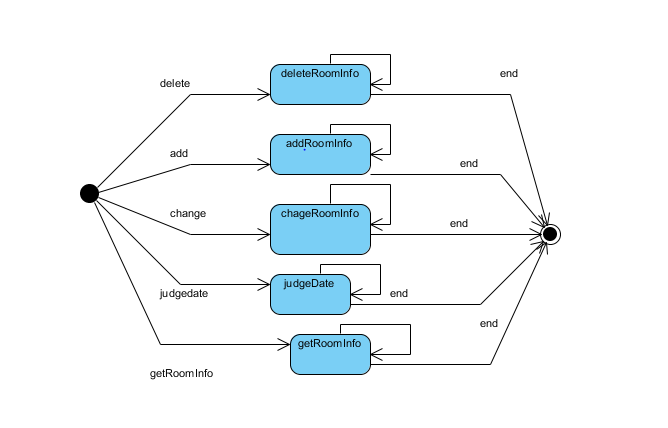


图1.1.2.1-5策略制定状态图

## credit模块静态结构和动态行为

### 1.2.1 credit模块概述

credit模块承担信用充值和查看的功能。

### 1.2.2 Credit模块局部模块的职责

### 1.2.2.1结构视角

（1）模块概述

credit模块承担的需求见需求规格说明书功能需求及相关非功能需求。

credit模块的职责及接口参见软件体系结构描述文档表。

（2）整体结构

根据体系结构的设计，采用分层风格，将系统分为展示层、业务逻辑层、数据层。每一层之间为了灵活性，添加了接口，以实现针对接口编程，隔离数据传输的职责，降低层与层之间耦合，添加了creditblservice，creditdataservice两个接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了creditController，这样creditController将会将信用值相关的业务逻辑职责和逻辑控制委托给credit对象。creditPO是作为信用值的持久化对象被添加到设计模型中的。StrategyList和StrategyLineItem的添加是RoomInfo的容器类。UserInfo是根据依赖倒置原则，为了消除循环依赖而产生的接口。

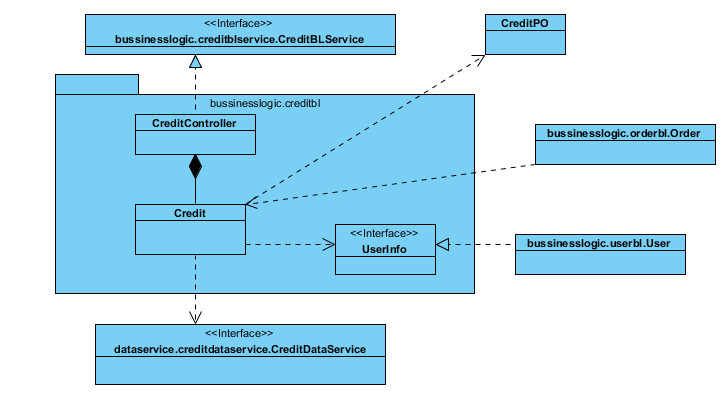


图1.2.2.1-1 credit模块各个类的设计

表1.2.2.1-1 credit模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| CreditController | 负责实现对应于信用值查看界面所需的方法 |
| Credit | 负责管理信用值 |

（3）credit模块内部接口规范

表1.2.2.1-2 CreditController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditController.getInfo | 语法 | Public List<CreditPO>getInfo(long userID) |
| 前置条件 | 已创建一个Credit领域对象的getInfo方法 |
| 后置条件 | 调用Credit领域对象的getInfo方法 |
| CreditController.recharge | 语法 | Public ResultMessage recharge(int value) |
| 前置条件 | 已创建一个Credit领域对象的recharge方法 |
| 后置条件 | 调用Credit领域对象的recharge方法 |
| CreditController.add | 语法 | Public void add(long userID,int value) |
| 前置条件 | 已创建一个Credit领域对象的add方法 |
| 后置条件 | 调用Credit领域对象的add方法 |
| CreditController.decrease | 语法 | Public void decrease(long userID,int value) |
| 前置条件 | 已创建一个Credit领域对象的decrease方法 |
| 后置条件 | 调用Credit领域对象的decrease方法 |
| CreditController.insert | 语法 | Public ResultMessage(long userID) |
| 前置条件 | 已创建一个Credit领域对象的insert方法 |
| 后置条件 | 调用Credit领域对象的insert方法 |
| 需要的服务（需接口） | | |
| Credit.getInfo | | 得到用户的信用信息 |
| Credit.recharge | | 充值信用 |
| Credit.add | | 增加信用值 |
| Credit.decrease | | 减少信用值 |
| Credit.insert | | 创建信用信息 |

表1.2.2.1-3 Credit的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Credit.getInfo | 语法 | Public List<CreditPO>getInfo(long userID) |
| 前置条件 | 启动查看信用回合 |
| 后置条件 | 返回用户信用信息 |
| Credit.recharge | 语法 | Public ResultMessage recharge(int value) |
| 前置条件 | 启动信用充值回合 |
| 后置条件 | 增加相应数值的信用值 |
| Credit.add | 语法 | Public void add(long userID,int value) |
| 前置条件 | 订单已执行或者酒店工作人员补登记订单或者网站工作人员撤销异常订单后，启动返还信用值回合 |
| 后置条件 | 客户的信用值增加相应的数值 |
| Credit.decrease | 语法 | Public void decrease(long uerID,int value) |
| 前置条件 | 订单由未执行状态变为异常状态，启动扣除信用值回合 |
| 后置条件 | 扣除相应数量的信用值 |
| Credit.insert | 语法 | Public ResultMessage insert(long usrID) |
| 前置条件 | 不存在此客户的信用记录 |
| 后置条件 | 创建此客户的信用记录 |
| 需要的服务（需接口） | | |
| DatabaseFactory.getCreditDatabase | | 得到Credit数据库服务的引用 |
| CreditDataService.add(long userID,int value) | | 增加数据库中的相应信用值 |
| CreditDataService.decrease(long userID,int value) | | 减少数据库中的相应信用值 |
| CreditDataService.insert(CreditPO,po) | | 在数据库中插入CreditPO对象 |

（4）动态模型

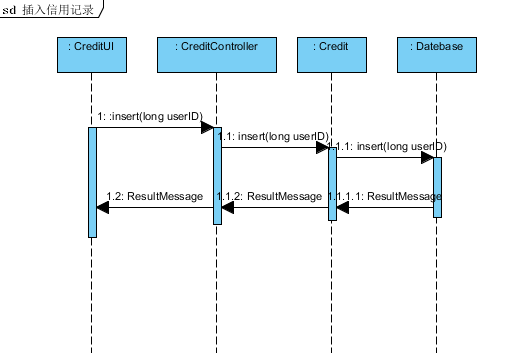


图1.1.2.1-2插入信用记录顺序图

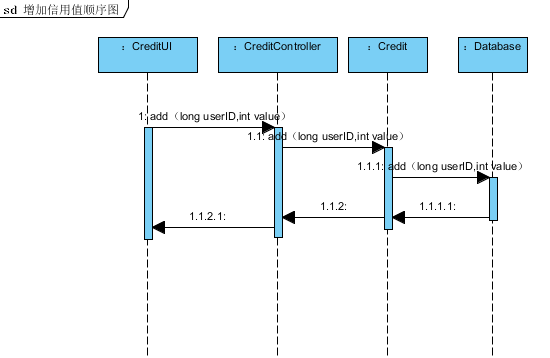


图1.1.2.1-3增加信用值顺序图

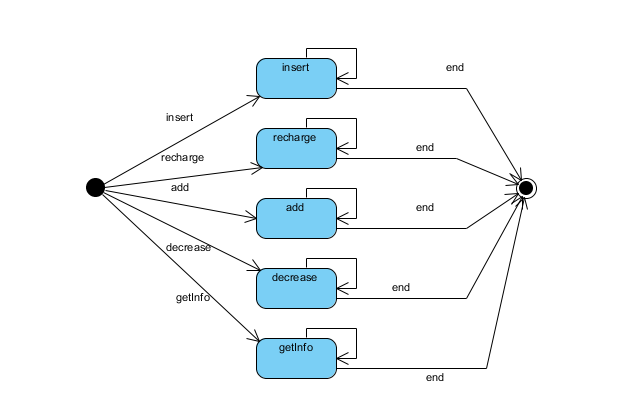


图1.1.2.1-4 Credit模块状态图