## 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、查看以及酒店和订单相关功能的实现。

### 1.1.2 strategy模块局部模块的职责

### 1.1.2.1结构视角

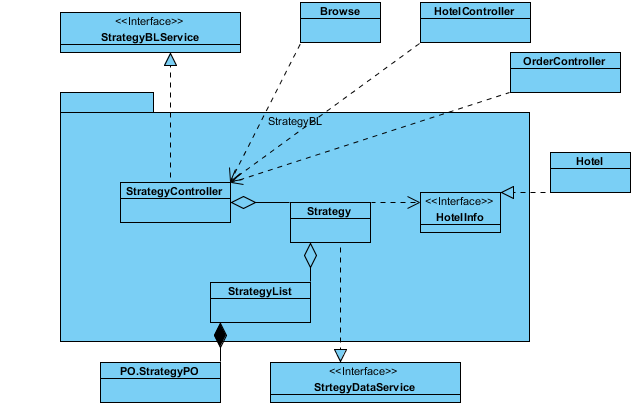
（1）模块概述

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的添加是StrategyInfo的容器类。HotelInfo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 |
| StrategyList | 优惠策略信息集合的领域模型对象 |

（3）模块内部类的接口规范

StrategyController 、Strategy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.getStrategyInfo | 语法 | Public List<StrategyVO> getStrategyInfo(long hotelID) |
| 前置条件 | 请求获取相关酒店的所有策略信息 |
| 后置条件 | 调用Strategy领域对象的getStrategyInfo方法 |
| StrategyController.deleteStrategyInfo | 语法 | Public ResultMessage deleteStrategyInfo(long hotelID,long StrategyID) |
| 前置条件 | 请求删除策略对象 |
| 后置条件 | 调用Strategy领域对象的deleteStrategyInfo方法 |
| StrategyController.addStrategyInfo | 语法 | Public ResultMessage addRoomInfo(StrategyVO,vo) |
| 前置条件 | 请求增加一条策略信息 |
| 后置条件 | 调用Strategy领域对象的addStrategyInfo方法 |
| StrategyController.changeStrategyInfo | 语法 | Public ResultMessage changeStrategyInfo(StrategyVO,vo) |
| 前置条件 | 已创建一个Strategy领域对象的changeStrategyInfo方法 |
| 后置条件 | 调用Strategy领域对象的changeStrategyInfo方法 |
| StrategyController.getRoomInfo | 语法 | Public List<RoomVO> getRoomInfo(long hotelID) |
| 前置条件 | 请求获取酒店的房间信息 |
| 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.getStrategyInfo | | 获取酒店所有的策略信息列表 |
| Strategy.deleteStrategyInfo | | 删除酒店的某一策略信息 |
| Strategy.addStrategyInfo | | 增加酒店的某一策略信息 |
| Strategy.changeStrategyInfo | | 改变酒店的某一策略信息 |
| Strategy.getRoomInfo | | 获取酒店的客房信息 |

表1.1.2.1-2 Strategy的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Strategy.getStrategyInfo | 语法 | Public List<StrategyVO,vo> getStrategyInfo(long hotelID) |
| 前置条件 | StrategyController请求调用 |
| 后置条件 | 获取酒店所有策略信息的列表 |
| Strategy.deleteStrategyInfo | 语法 | Public ResultMessage deleteStrategyInfo(long hotelID,long strategyID) |
| 前置条件 | StrategyController请求调用 |
| 后置条件 | 删除此酒店的此策略 |
| Strategy.addStrategyInfo | 语法 | Public ResultMessage addStrategyInfo(StrategyVO,vo) |
| 前置条件 | StrategyController请求调用 |
| 后置条件 | 增加一条策略信息 |
| Strategy.changeStrategyInfo | 语法 | Public ResultMessage changeStrategyInfo(StrategyVO,vo) |
| 前置条件 | 此类型的客房存在优惠策略且StrategyController请求调用 |
| 后置条件 | 更改相应的策略信息 |
| Stategy.getRoomInfo | 语法 | Public List<roomVO> getRoomInfo(long hotelID) |
| 前置条件 | StrategyController请求调用 |
| 后置条件 | 返回此酒店的所有客房信息 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| HotelController.getRoomInfo() | | 获得酒店所有的客房信息的持久化对象 |
| StrategyDataService.add(StrategyPO,po) | | 插入一条策略信息的持久化对象 |
| StrategyDataService.delete(StrategyPO,po) | | 删除一条策略信息的持久化对象 |
| StrategyDataService.change(StrategyPO,po) | | 修改一条策略信息的持久化对象 |

（4）动态模型：

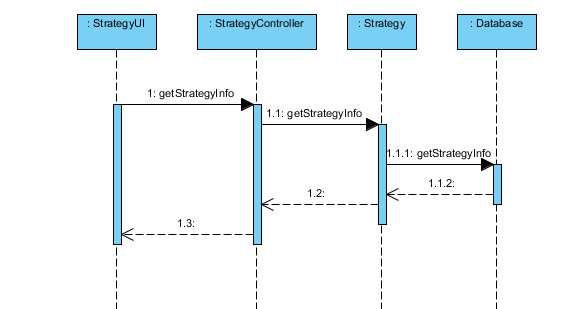


图1.1.2.1-1获取策略信息顺序图

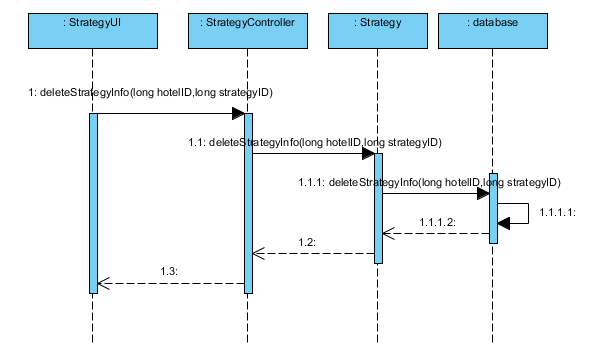


图1.1.2.1-2Strategy的删除策略信息顺序图

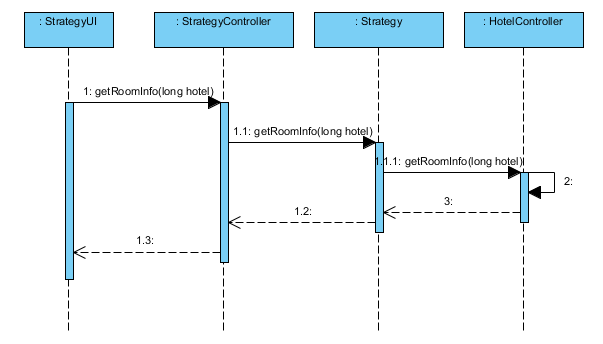


图1.1.2.1-3获取酒店客房信息顺序图

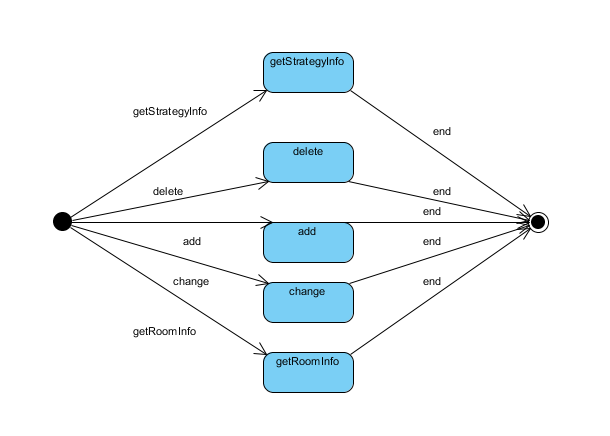


图1.1.2.1-5策略模块状态图

## 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、查看界面相关的功能。

### 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。

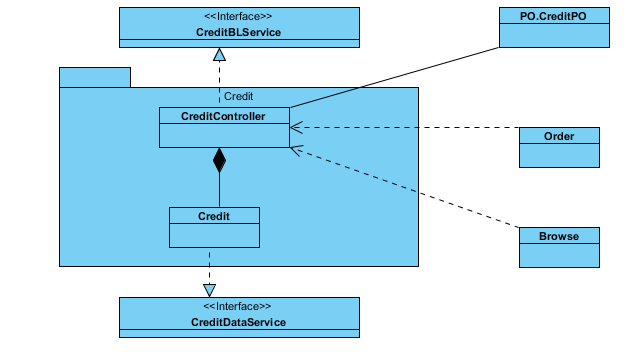


图1.2.2.1-1 credit模块各个类的设计

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和充值界面所需的方法 |
| 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 | 请求查看用户信用信息明细 |
| 后置条件 | 调用Credit领域对象的getInfo方法 |
| CreditController.update | 语法 | Public ResultMessage update(long userID,int value,String style) |
| 前置条件 | 请求对信用值进行操作 |
| 后置条件 | 调用Credit领域对象的update方法 |
| CreditController.insert | 语法 | Public ResultMessage(CreditPO,po) |
| 前置条件 | 请求插入一条信用信息 |
| 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.update | | 信用值更改 |
| Credit.insert | | 插入信用信息 |

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 | 请求查看用户信用明细 |
| 后置条件 | 返回用户信用明细信息 |
| Credit.update | 语法 | Public ResultMessage update(long userID,int value,long style) |
| 前置条件 | 请求对客户信用值发生更改 |
| 后置条件 | 更改客户信用值并存储 |
| Credit.insert | 语法 | Public ResultMessage insert(CreditPO,po) |
| 前置条件 | 请求插入一条信用记录 |
| 后置条件 | 插入一条信用记录并存储 |
| 需要的服务（需接口） | | |
| DatabaseFactory.getCreditDatabase | | 得到Credit数据库服务的引用 |
| CreditDataService.update(long userID,int value,string style) | | 更新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

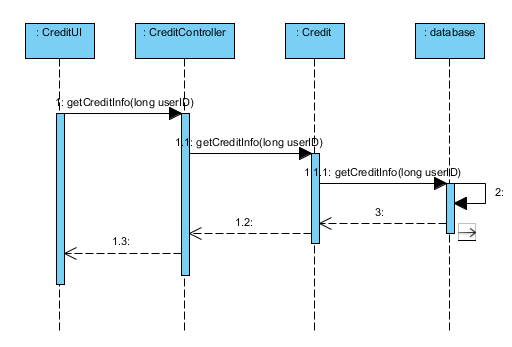


图1.1.2.1-2获取信用明细顺序图

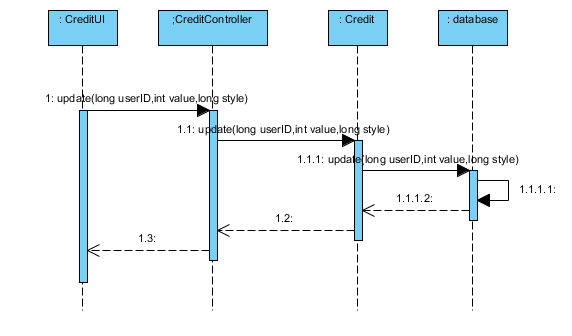


图1.1.2.1-3更新信用值顺序图

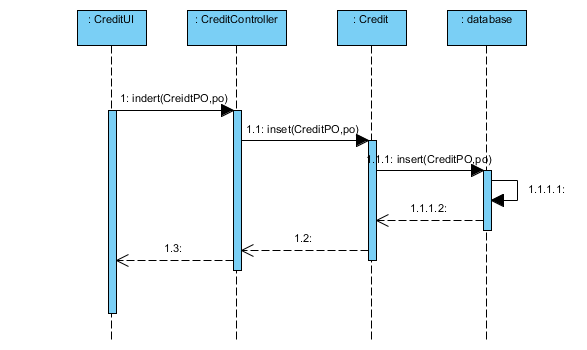


图1.1.2.1-4插入信用信息顺序图

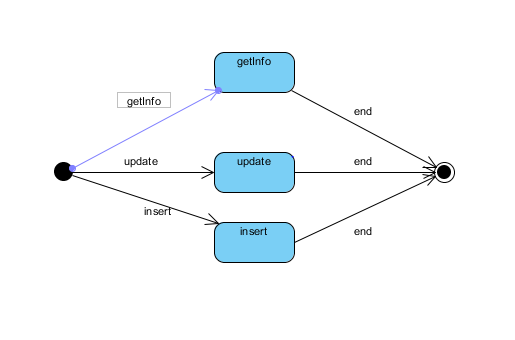


图1.1.2.1-4 Credit模块状态图