### member模块

1. 模块概述

member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。

member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memberBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加memberController，这样memberController会将对会员注册的业务逻辑处理委托给member对象。memberPO是作为会员信息的持久性对象被添加到设计模型中去的。



图1—member模块设计类图

1. 模块内部类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| memberController.memberSignUp | 语法 | Public ResultMessage memberSignUp (MemberVO memberVO) |
| 前置条件 | 客户输入的会员信息符合信息规范 |
| 后置条件 | 系统返回是否成功注册会员；是则提示添加成功，否则提示注册失败原因 |
| memberController.modifyMemberInfo | 语法 | Public ResultMessage modifyMemberInfo (String guestID, MemberVO memberVO) |
| 前置条件 | 用户输入客户编号，并且输入的信息符合信息规范 |
| 后置条件 | 系统返回是否成功修改会员；是则提示修改成功，否则提示修改失败原因 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Member.add | 系统增加会员信息 | |
| Member.modify | 系统修改会员信息 | |
| Member.getMemberInfo | 系统返回指定客户的会员信息 | |
| Member.isMember | 系统返回指定客户是否为会员 | |
| Member.getMemberType | 系统返回指定客户的会员类型 | |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Memeber.add | 语法 | Public ResultMessage add(MemberVO memberVO) |
| 前置条件 | 会员信息符合规则且齐全 |
| 后置条件 | 系统根据已指定会员类型持久化为该客户增加会员数据 |
| Member.modify | 语法 | Public ResultMessage modify(MemberVO memberVO) |
| 前置条件 | 会员信息符合规则且齐全 |
| 后置条件 | 系统修改该会员并持久化保存该客户的会员数据 |
| Member.getMemberInfo | 语法 | Public MemberVO getMemberInfo(String userID,MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统返回与该userID对应的指定会员类型 |
| Member.isMember | 语法 | public boolean isMember(String userID, MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 返回该userID的用户是否为会员 |
| Member.getMemberType | 语法 | public MemberType getMemberType(String userID, MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统返回该userID的会员类型 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| GuestDataService.ModifyMember(MemberPO memberPO) | 根据提供的memberPO修改单一持久化客户对象的信用值 | |
| GuestDataService.  getSingle (String userID) | 根据ID查找客户持久化对象 | |

1. 业务逻辑层的动态模型

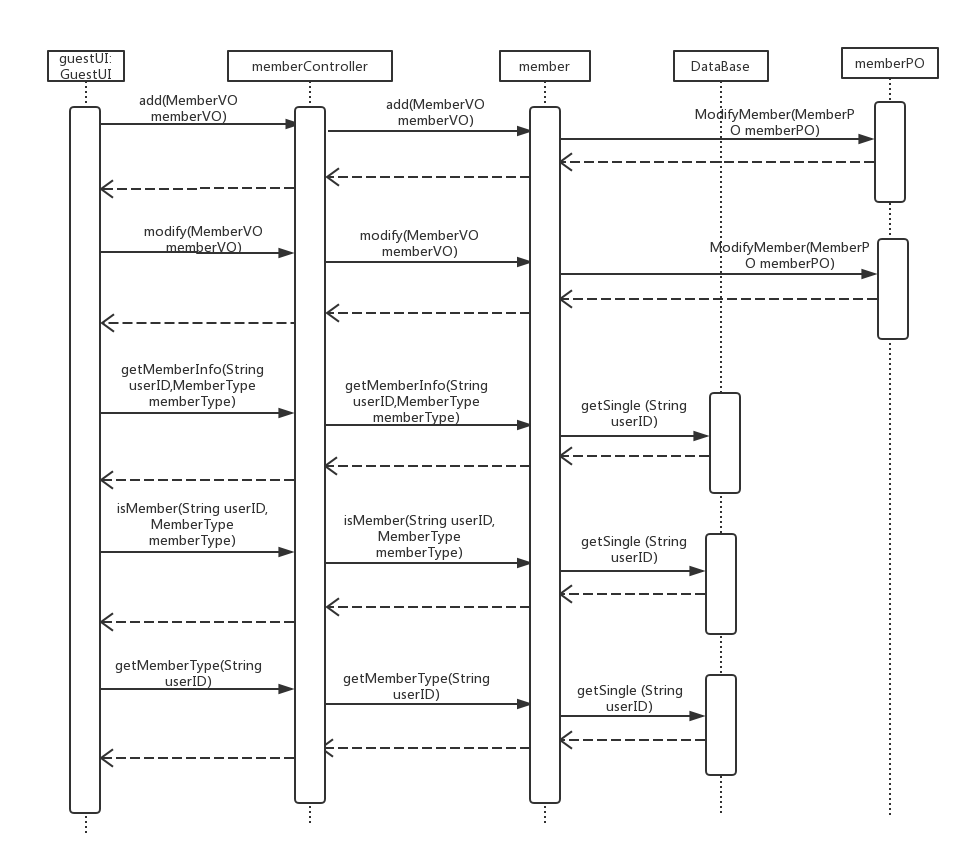


图2—member模块设计顺序图



图3—member模块设计状态图

1. 业务逻辑层的设计原理