## userbl模块

1. 模块概述

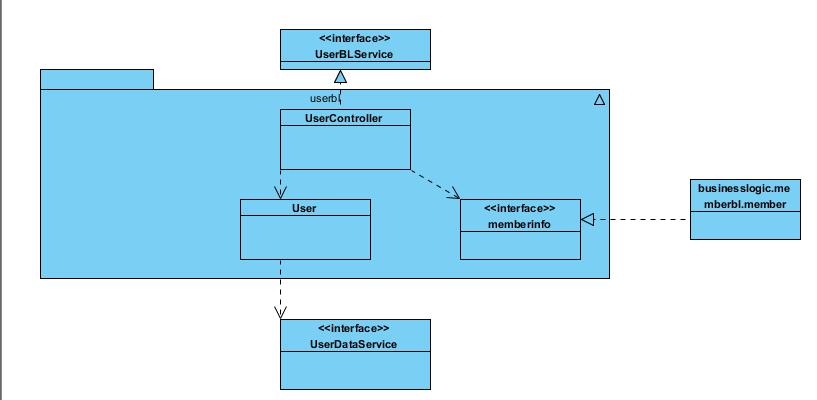
userbl模块承担的需求参见需求规格说明文档中登录/注册相关内容及相关非功能需求

userbl模块承担的职责和接口见体系设计文档中的userbl相关的职责和接口

1. 整体结构

依先前体系设计逻辑模型和相关文档的说明，为了增加灵活性降低耦合，增加可复用性，在界面层和业务逻辑层增加接口层，在数据层和业务逻辑层也增加接口层，通过接口进行调用，其中，使用usercontroller将模块里的业务逻辑和逻辑操作相分开，将登陆登出及注册委托给user，同时在登陆成功后调用相关member的接口进行初始化。

userbl 模块的设计如下图所示



userbl模块各个类的职责由下表所示

|  |  |
| --- | --- |
| 类 | 职责 |
| UserController | 负责Userbl模块的逻辑操作，实现对应于登陆登出和注册所需要的服务 |
| User | 用户登录相关的领域模型，拥有用户数据的用户名和密码，可以帮助完成登录登出相关的操作 |

1. 模块内部类的接口规范

UserController、User和Register的接口规范如下表所示

UserController:

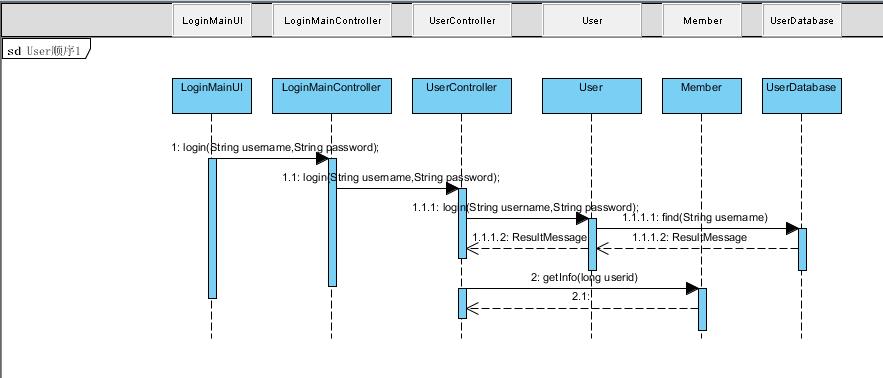
|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| UserController.login | 语法 | public ResultMessage login(String username,String password); |
| 前置条件 | 用户准备登陆且输入符合规范 |
| 后置条件 | 调用User的login方法 |
| UserController.register | 语法 | public ResultMessage logout(String username,String password); |
| 前置条件 | 用户选择登陆且输入符合规范 |
| 后置条件 | 调用User的register方法 |
| UserController.logout | 语法 | public ResultMessage logout(long userid); |
| 前置条件 | 用户已登录 |
| 后置条件 | 调用User的logout方法 |
| UserController.getInfo | 语法 | public MemberVO getInfo(long userid); |
| 前置条件 | 一个用户已登录，且登录成功并拿到相应userid |
| 后置条件 | 调用Member的getInfo方法 |
| 需要的接口 | | |
| 服务名 | 服务 | |
| User.login(String username,String password); | 提供用户登录服务 | |
| User.register(String username,String password); | 加入一个用户对象 | |
| User.logout(long userid); | 登出一个用户 | |
| Member.getInfo(long userid); | 获取一个用户的信息 | |

User:

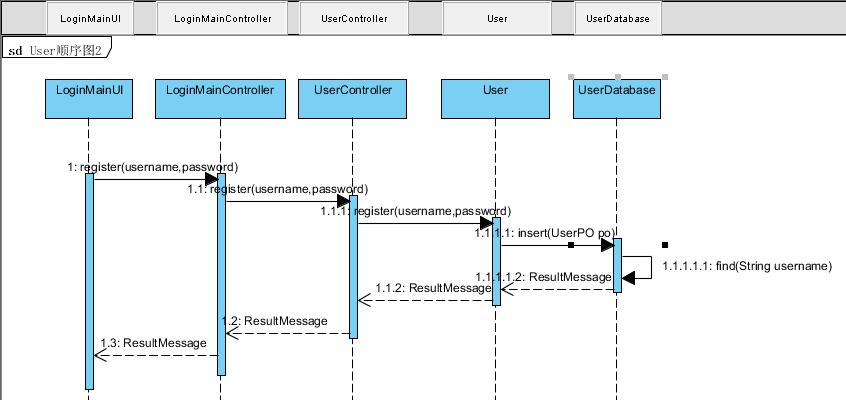
|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| User.login | 语法 | public ResultMessage login(String username,String password); |
| 前置条件 | 已经开始一个登录行为 |
| 后置条件 | 根据传入的参数进行判断，并返回登录结果 |
| User.register | 语法 | public ResultMessage logout(String username,String password); |
| 前置条件 | 已经开始一个注册行为 |
| 后置条件 | 若该用户名没有被注册，新增一个User到数据库中 |
| User.logout | 语法 | public ResultMessage logout(long userid); |
| 前置条件 | 用户已登录 |
| 后置条件 | 数据库中对应的用户状态置为未登录 |
| 需要的接口 | | |
| 服务名 | 服务 | |
| UserDataService.insert(UserPO po); | 插入单一持久化对象 | |
| UserDataService.update(UserPO po); | 更新单一持久化对象 | |
| UserDataService.find(String username); | 根据username查找单一持久化对象 | |
| UserDataService.find(long userid); | 根据Id查找单一持久化对象 | |

1. 业务逻辑层的动态模型

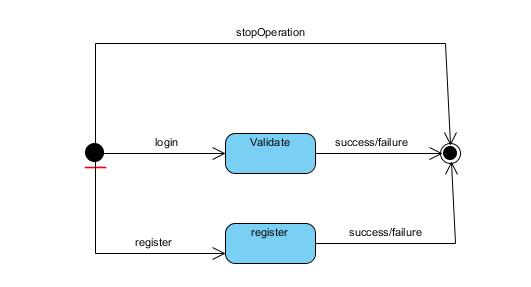
下图显示当用户输入登录的用户名和密码之后，User业务逻辑层处理的各个对象之间的协作：



下图为用户注册输入需要注册的用户名和密码之后，User业务逻辑层处理各个对象之间的协作：



下图描述了User对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作，当有用户执行登入或登出，sale会进行验证并返回结果，停止操作也可以直接进入终态：



1. 业务逻辑层设计原理

采用委托式控制器风格，由controller统一调度，将任务委托给对应的对象进行处理。

### Browsebl模块

1. 模块概述

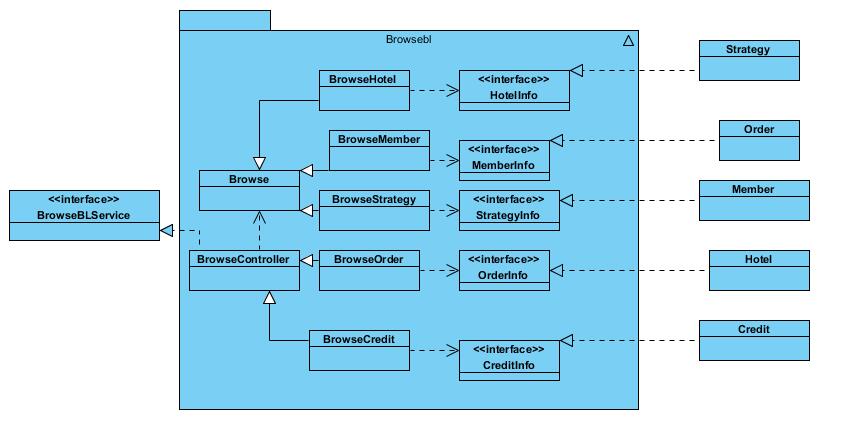
browseBl模块承担的需求参见需求规格说明文档中浏览订单信息（3个），浏览酒店信息，制定策略中隐含的浏览策略信息等相关隐含或非隐含的内容及相关非功能需求

browseBl模块承担的职责和接口见体系设计文档中的browseBl相关的职责和接口

1. 整体结构

具体总体概述见userbl模块的整体结构前半段描述，这里只介绍内部具体，使用BrowseController将模块里的业务逻辑和逻辑操作相分开，直接控制Browse（这里Browse是一个基类），以适配器的风格，将浏览订单信息委托给BrowseOrderList，将浏览策略信息委托给BrowseStrategyList,各具体类型的适配器分别调用其他模块接口完成自己的工作。具体的返回也有相应的基本列表包装！

browsebl 模块设计如下：



browsebl各个类的职责如下

|  |  |
| --- | --- |
| 类 | 职责 |
| BrowseController | 负责Browsebl模块的逻辑操作，实现对应于各类信息浏览所需要的服务 |
| Browse | 浏览信息相关的领域模型，是一个基类适配器，它能转化具体类型的接口 |
| BrowseHotel | 具体的适配Ataptee，适配对象，实现具体的浏览酒店信息服务 |
| BrowseMember | 具体的适配Ataptee，适配对象，实现具体的浏览会员信息服务 |
| BrowseStrategy | 具体的适配Ataptee，适配对象，实现具体的浏览策略信息服务 |
| BrowseOrder | 具体的适配Ataptee，适配对象，实现具体的浏览订单信息服务 |
| BrowseCredit | 具体的适配Ataptee，适配对象，实现具体的浏览信用信息服务 |

1. 模块内部类的接口规范

BrowseController、Browse和具体的Browse\*的接口规范如下表所示

BrowseController:

所有的对Browse的调用，类型会根据类型声明成具体的类型，而实际为相应的具体的adaptee的引用

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseController.getOrdersInfo | 语法 | Public List<OrderVO> getOrdersInfo(long userid,OrderType type); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 调用Browse的getData方法，参数为(Role role,long id,OrderType) |
| BrowseController.getHotelOrdersInfo | 语法 | Public List<OrderVO> getHotelOrdersInfo(long hotelid,OrderType type); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 调用Browse的getData方法，参数为(Role role,long id,OrderType) |
| BrowseController.getWEBOrdersInfo | 语法 | Public List<OrderVO> getWEBOrdersInfo(); |
| 前置条件 | 网站营销人员已登录 |
| 后置条件 | 调用Browse的getData方法，参数为(Role role,long id,OrderType) 其中id为-1，OrderType默认为异常 |
| BrowseController.getCreditInfo | 语法 | Public List<CreditVO> getCreditInfo(long userid); |
| 前置条件 | 客户已登录 |
| 后置条件 | 调用相应Browse的getData方法 参数为该id |
| BrowseController.getHotels | 语法 | Public List<HotelVO> getHotels(RuleVO vo); |
| 前置条件 | 客户已登录 |
| 后置条件 | 调用相应Browse的getData方法 参数为该RuleVO |
| BrowseController.getHotel | 语法 | Public HotelVO getHotel(long hotelid); |
| 前置条件 | 客户已登录 |
| 后置条件 | 调用相应Browse的getData方法 参数为该id |
| BrowseController.getStrategyInfo | 语法 | Public StrategyVO getStrategyInfo(int orderid); |
| 前置条件 | 用户选择一个订单 |
| 后置条件 | 调用相应Browse的getData方法 参数为该id |
| BrowseController.getMemberInfo | 语法 | Public MemberVO getMemberInfo(long memberid); |
| 前置条件 | 用户已登录 |
| 后置条件 | 调用相应Browse的getData方法 参数为该id |
| BrowseController.getBrowseInfo | 语法 | Public List<BrowseVO> getBrowseInfo(long userid); |
| 前置条件 | 客户已登录 |
| 后置条件 | 调用相应Browse的getData方法 参数为该id |
| BrowseController.clear | 语法 | Public void clear(long userid); |
| 前置条件 | 客户已登录 |
| 后置条件 | 调用Browse的clear方法 |
| 需要的服务(需接口) | | |
| Hotel.getHotels(RulePO po) | 得到基于rule的酒店信息列表 | |
| Hotel.getInfo(long Hotelid) | 得到hotel的详细信息 | |
| Credit.getInfo(long userID) | 得到user的信用信息 | |
| Order.getuserInfo(long userid,int method) | 得到user的对应method的对应类别的订单信息 | |
| Order.getHotelInfoloing hotelid,int method) | 得到hotel的对应method的对应类别的订单信息 | |
| Order.getWEBInfo() | 得到当天当前时间前所有的异常订单信息 | |
| Member.getInfo(long userid) | 得到对应的会员信息 | |
| Strategy.getInfo(long orderid) | 得到对应订单的策略信息 | |

BaseList 基列表类封装对应的基对象，包含大量对具体对象的集合，格式如下

BaseList<T> 其属性有 list<T>为引用（里面可以只为1个）

Browse类的接口规范如下：

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| Browse.getData | 语法 | Public BaseList<T> getData(Role role,long userid,OrderType type); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据其泛型T，jvm自己调用具体的adaptee的相应方法并返回 |
| Browse.getData | 语法 | Public BaseList<T> getData(long id); |
| 前置条件 | 酒店工作人员已登录 |
| 后置条件 | 根据其泛型T，jvm自己调用具体的adaptee的相应方法并返回（返回一个VO即对应BaseList只有一个VO对象） |
| Browse.getData | 语法 | Public BaseList<T> getData(RuleVO vo); |
| 前置条件 | 网站营销人员已登录 |
| 后置条件 | 根据其泛型T，jvm自己调用具体的adaptee的相应方法并返回 |
| Browse.clear | 语法 | Public void clear(long userid); |
| 前置条件 | 客户已登录 |
| 后置条件 | 清空数据库中相应的用户的浏览记录 |
| 需要的服务(需接口) | | |
| DatabaseFactory.getBrowseDatabase | 得到Browse数据库服务的引用 | |
| BrowseDataService.insert(BrowsePO po) | 在Browse数据库中插入po（搜索记录） | |
| BrowseDateService.getInfo(userid) | 返回一个用户的浏览/搜索记录 | |
| BrowseDataService.clear(long userid) | 清空一个用户的浏览搜索记录 | |
| DatabaseFactory.getBrowseDatabase | 得到Browse数据库服务的引用 | |

BrowseOrder的类的接口规范如下（其继承基类Browse，类型为OrderVO）（只描述实现的接口，继承下未改动的查看父类）：

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseOrder.getData | 语法 | Public BaseList<OrderVO> getData(Role role,long userid,OrderType type); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | Role，调用相应的Order方法，获得相应列表后包装并返回 |
| 需要的服务(需接口) | | |
| Order.getuserInfo(long userid,int method) | 得到user的对应method的对应类别的订单信息 | |
| Order.getHotelInfoloing hotelid,int method) | 得到hotel的对应method的对应类别的订单信息 | |
| Order.getWEBInfo() | 得到当天当前时间前所有的异常订单信息 | |

BrowseCredit的类的接口规范如下（其继承基类Browse，类型为CreditVO）（只描述实现的接口，继承下未改动的查看父类）：

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseCredit.getData | 语法 | Public BaseList<CreditVO> getData(long userid); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据传入的userid调用credit的getInfo方法 |
| 需要的服务(需接口) | | |
| Credit.getInfo(long userID) | 得到user的信用信息 | |

BrowseHotel的类的接口规范如下（其继承基类Browse，类型为HotelVO）（只描述实现的接口，继承下未改动的查看父类）：

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseHotel.getData | 语法 | Public BaseList<HotelVO> getData(long hotelid); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据传入的hotelid,调用hotel 的getinfo方法，包装信息后返回包装的信息 |
| 需要的服务(需接口) | | |
| Hotel.getHotels(RulePO po) | 得到基于rule的酒店信息列表 | |
| Hotel.getInfo(long Hotelid) | 得到hotel的详细信息 | |

BrowseMember的类的接口规范如下（其继承基类Browse，类型为MemberVO）（只描述实现的接口，继承下未改动的查看父类）：

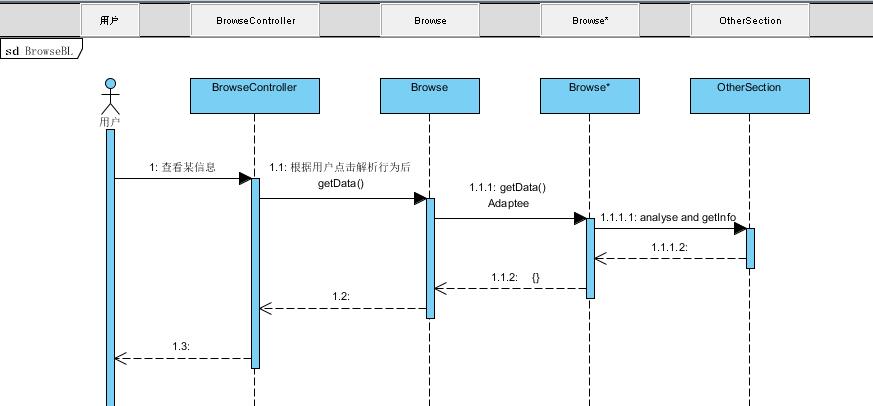
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseMember.getData | 语法 | Public BaseList<MemberVO> getData(long userid); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据传入的userid调用member的方法  长度为13，则为getInfo  长度为1，则为getList |
| 需要的服务(需接口) | | |
| Member.getInfo(long userid) | 得到对应的会员信息 | |
| Member.getList(int type); | 得到对应的用户列表 | |

BrowseStrategy的类的接口规范如下（其继承基类Browse，类型为StrategyVO）（只描述实现的接口，继承下未改动的查看父类）：

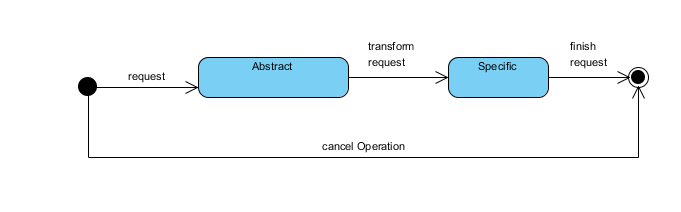
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| BrowseStrategy.getData | 语法 | Public BaseList<StrategyVO> getData(long orderid); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 根据传入的orderid，调用Strategy的getInfo方法 |
| 需要的服务(需接口) | | |
| Strategy.getInfo(long orderid) | 得到对应订单的策略信息 | |

(4)业务逻辑层的动态模型

下图显示当用户请求查看具体的信息时，Browsebl模块各个类之间的协作时序图



下图描述了Browse对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作。当客户发起了一个请求，Browse处于抽象状态，控制器解析后调用相应方法（并声明了具体类型），Browse变为具体状态并处理请求，处理完后结束



（5）业务逻辑层设计原理

采用适配器模式风格，由controller统一调度，将任务委托给Browse，Browse对类型进行转换后处理结束。