PandoraBox 网站说明文档

本文档是《程序设计实践》课程的大作业，“旗木卡卡东”小组作品“PandoraBox”的技术说明文档。

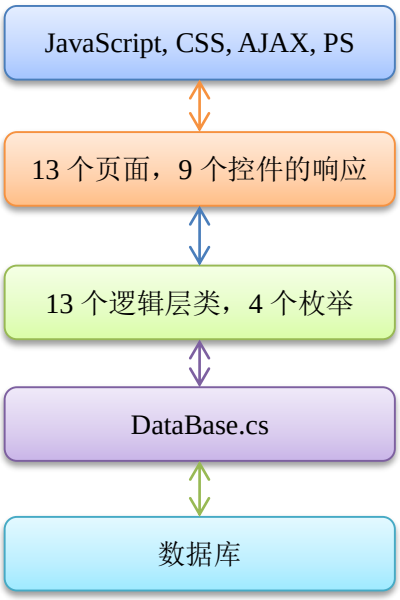
清华大学
软件 12
王思伦 江林楠 洪宇
2011013245 2011013260 2011013270
2013 年 7 月 19 日

目录

一、网站结构

1、整体结构

我们网站的整体如下图所示：

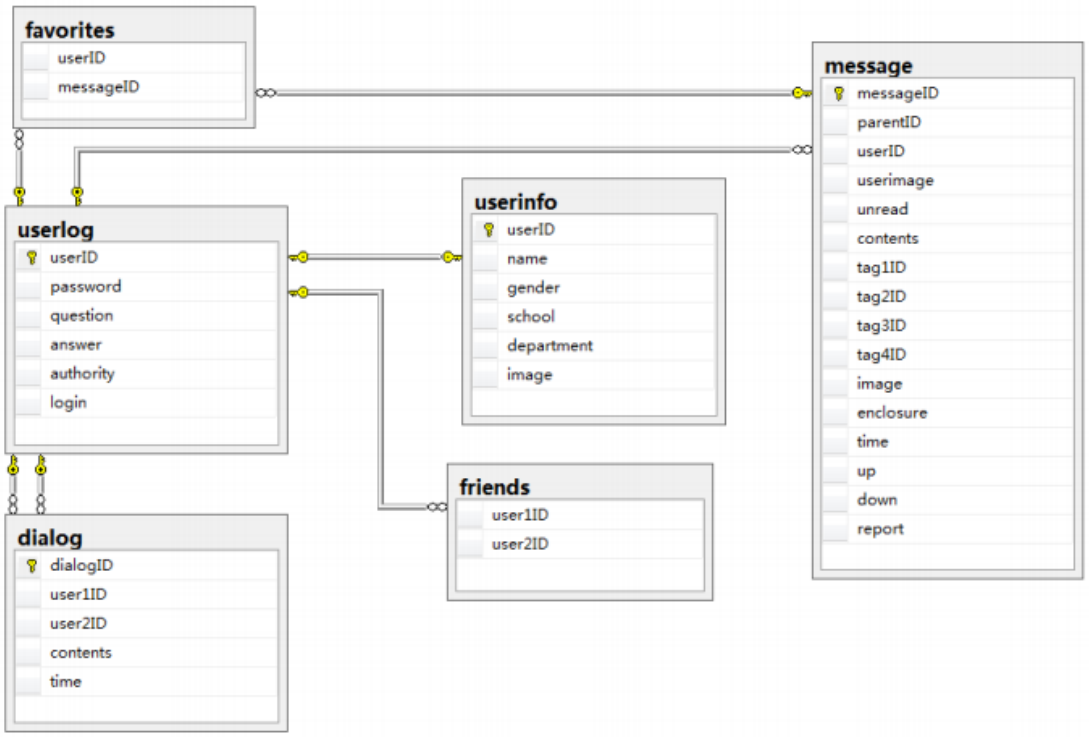


最顶一层是页面的设计，使用了 CSS，JavaScript 等多种方式美化页面的效果。第二层是页面的逻辑部分，为按钮赋予正确的响应。第三层是业务逻辑部分，与数据层进行沟通，获取页面上需要的数据。第四层是数据层，将读写数据库的操作封装起来，方便调用，最后一层是数据库。

在这种结构的设计下，可以保证页面逻辑部分与数据层完全隔离，页面所需要的数据完全从逻辑层得到。

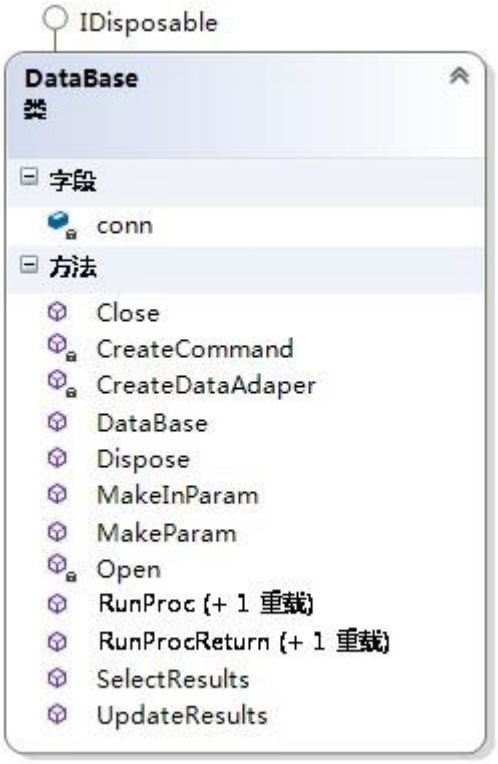
2、数据库设计与数据层

数据库的设计图如下所示：


favorites
userID
messageID
message
messageID
parentID
userID
userimage
unread
contents
tag1ID
tag2ID
tag3ID
tag4ID
image
enclosure
time
up
down
report
userlog
userID
password
question
answer
authority
login
userinfo
userID
name
gender
school
department
image
friends
user1ID
user2ID
dialog
dialogID
user1ID
user2ID
contents
time


tag
tagID
name
rank
parentID
image
course
tagID
name
englishname
teacher
exam
difficulty
reputation
difficulty1
difficulty2
difficulty3
difficulty4
difficulty5
reputation1
reputation2
reputation3
reputation4
reputation5
dish
tagID
name
reputation
reputation1
reputation2
reputation3
reputation4
reputation5

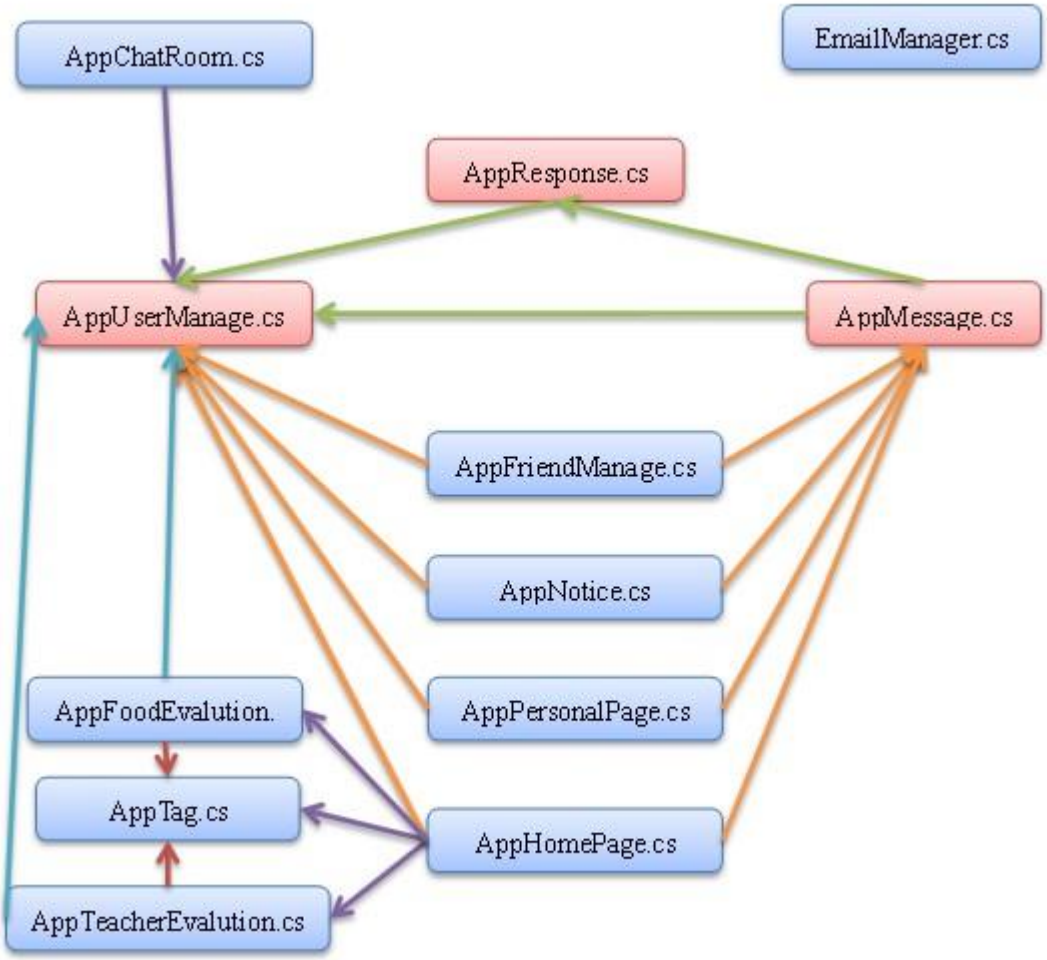
数据层的类图如下所示：



该类将数据库的相关操作以函数的方式封装起来，在执行数据库操作时，可以自动打开，关闭数据库。其中，MakeInParam 可以将 C#参数类型转换为数据库参数类型，RunProc 和 RunProcReturn 则封装了 SQL 语句，后者返回 C#中的 DataSet 类型供逻辑层进行后续的处理，取得界面所需的数据。

3、 逻辑层结构

逻辑层包含 13 个类，4 个枚举。其中，AppUser 类和 AppUserManage 类共用 AppUserManage.cs 文件，4 个枚举共用一个文件。4 个枚举分别是 AppUserAuthority，AppUserGender，AppEvaluationType，AppSortType，代表用户的权限，用户的性别，评分模块的种类，排序的方式。逻辑层中，类之间的关系如下图所示：

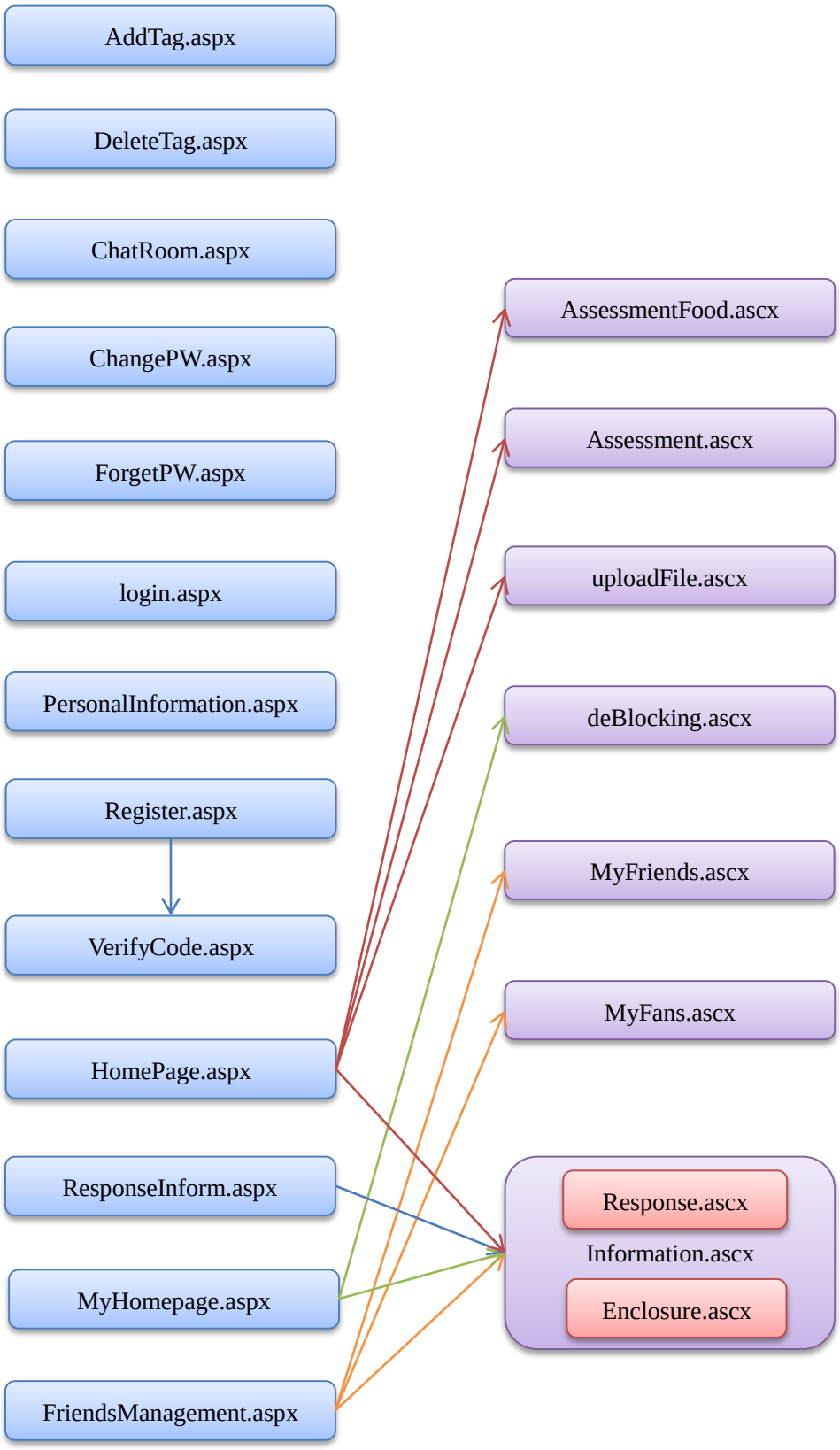


这里需要说明的是，箭头 a 指向 b 并不代表 a 继承自 b，而是表示 a 类中有 b 类型的成员或用到了 b 类的函数。

4、页面设计

页面的逻辑结构如下图所示。其中，AddTag.aspx 是增加标签页面，DeleteTag.aspx 是删除标签页面，ChatRoom.aspx 是聊天室页面，ChangePW.aspx 是更改密码页面，ForgetPW.aspx 是找回密码页面，login.aspx 是登录页面，PersonalInformation.aspx 是个人信息维护页面，Register.aspx 是注册页面，VerifyCode.aspx 是验证码页面，HomePage.aspx 是登录之后进入的主页，即“新吐槽”页面，ResponseInform.aspx 是回复提醒页面，MyHomepage.aspx 是个人主页，FriendsManagement.aspx 是好友管理主页。

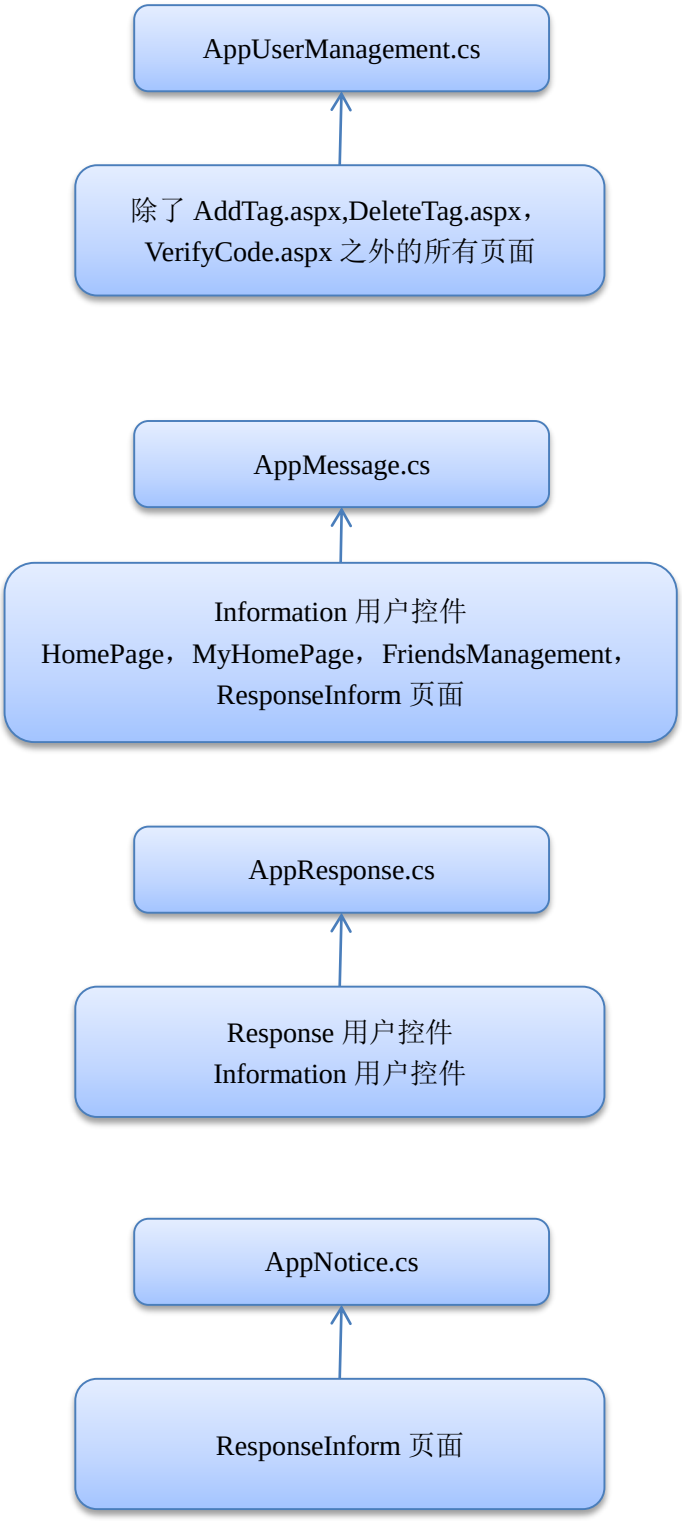
Assessment.ascx 是教师评价用户控件，AssessmentFood.ascx 是食品评价用户控件，deBlocking.ascx 是解封用户的用户控件，Enclosure.ascx 是附件提示用户控件，Information.ascx 是一条消息的用户控件，MyFans.ascx 是我的粉丝的用户控件，MyFriends.ascx 是我的好友的用户控件，Response.ascx 是回复的用户控件，uploadFile.ascx 是上传文件的用户控件。

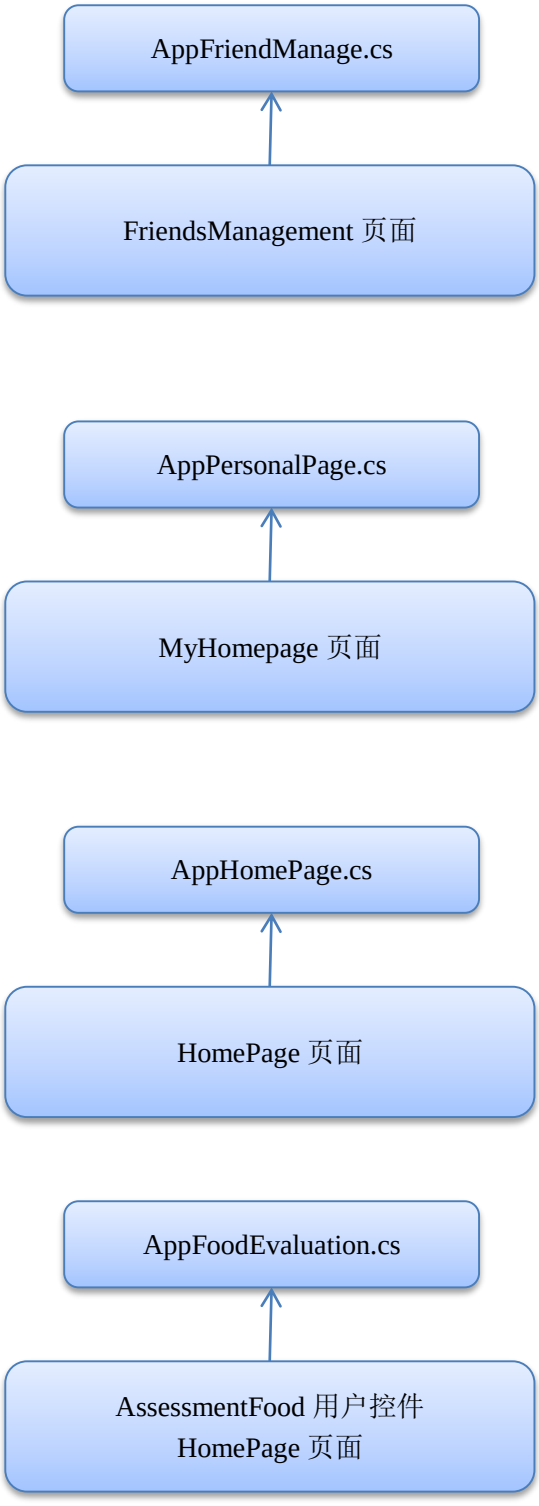


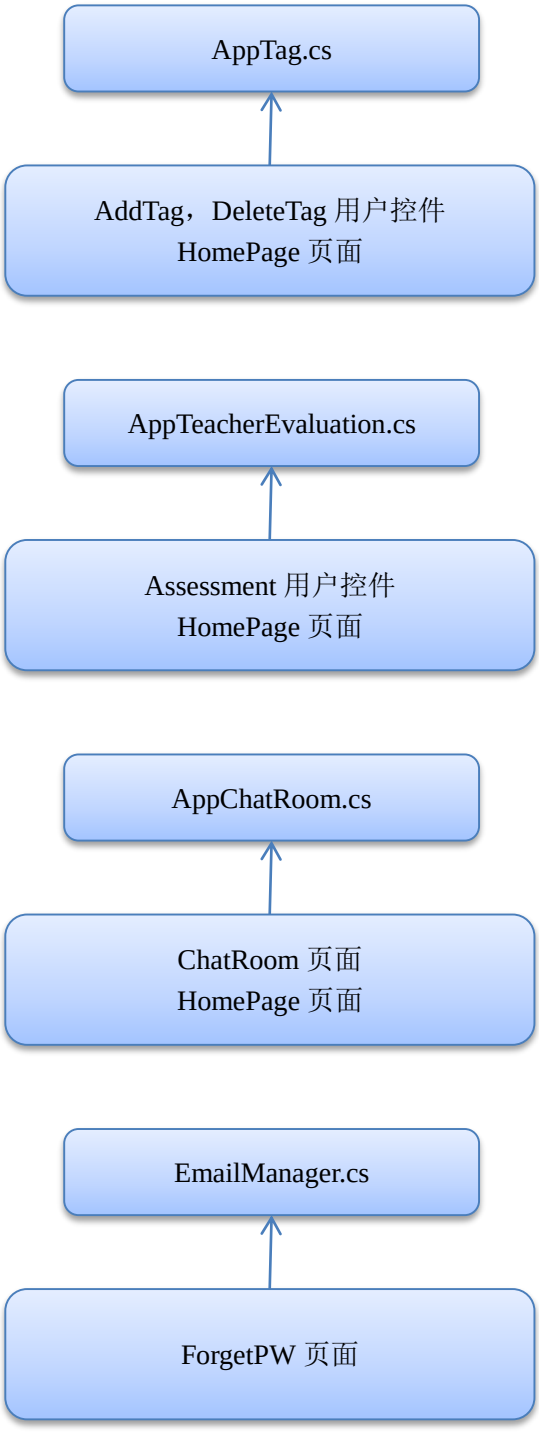
与上边一样，箭头 A 指向 B 表示 A 页面用到了 B 控件。另外，用户控件之间的包含关系也已经画出。

5、页面与逻辑层的关系

页面逻辑与业务逻辑之间的关系可以用下边几个图来表示。




AppFriendManage.cs
FriendsManagement 页面
AppPersonalPage.cs
MyHomepage 页面
AppHomePage.cs
HomePage 页面
AppFoodEvaluation.cs
AssessmentFood 用户控件
HomePage 页面



同样，箭头 A 指向 B 表示 A 使用过 B。

二、功能实现

1、个人信息维护部分

此部分内容包括登录，注册，修改个人信息，修改密码，找回密码五个部分。数据库中，有关用户信息的字段存储在两个表上，如下图所示：

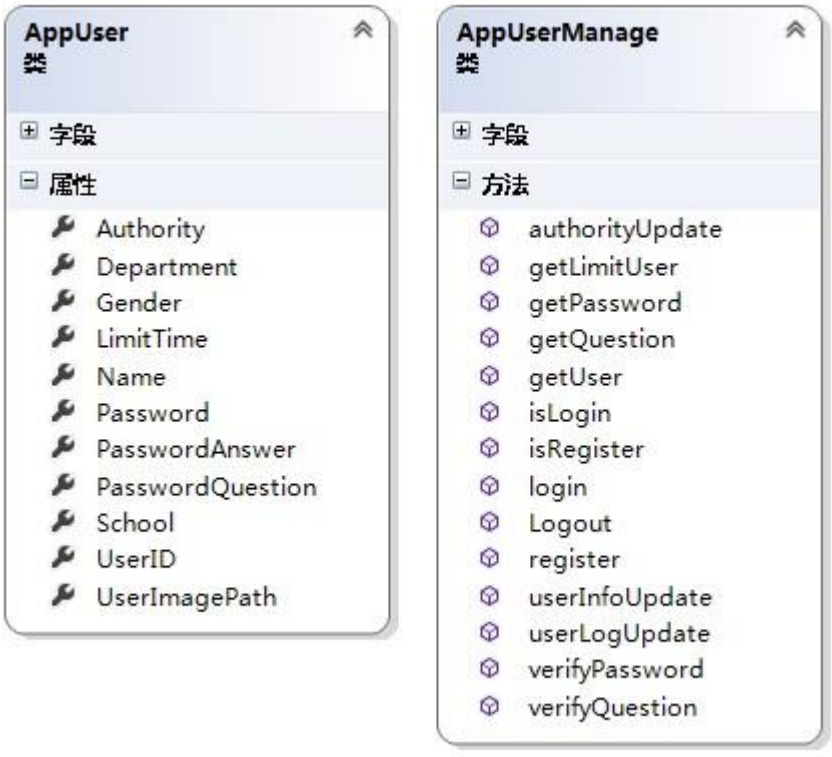
	名称	数据类型	允许 Null	默认值
🔑	userID	nvarchar(50)	☐	
	name	nvarchar(50)	☑	(N'')
	gender	nvarchar(50)	☑	(N'男')
	school	nvarchar(50)	☑	(N'清华大学')
	department	nvarchar(50)	☑	(N'')
	image	nvarchar(50)	☑	(N'')

- 键 (1)
  - PK_UserInfo (主键, Clustered: userID)
- CHECK 约束 (1)
  - CK_userinfo (gender)
- 索引 (0)
- 外键 (0)
- 触发器 (0)

	名称	数据类型	允许 Null	默认值
🔑	userID	nvarchar(50)	☐	
	password	nvarchar(50)	☐	
	question	nvarchar(50)	☐	
	answer	nvarchar(50)	☐	
	authority	int	☐	
	login	bit	☐	

- 键 (1)
  - PK_User (主键, Clustered: userID)
- CHECK 约束 (0)
- 索引 (0)
- 外键 (0)
- 触发器 (0)

核心逻辑都在 AppUserManagement.cs 中，其类视图如下所示：



如上左图，在 AppUser 类中，依次存储用户权限，院系，性别，被封的开始时间，昵称，密码，密码提示答案，密码提示问题，学校，用户邮箱，用户头像路径。

如上右图，在 AppUserManage 中，方法依次为：用户权限更新，获取所有被封用户，根据 UserID 获得密码，根据 UserID 获得密码提示问题，根据 UserID 获得用户，判断用户是否登录，判断用户是否注册，用户登录，用户注销，用户注册，用户个人信息更新，用户密码等关键信息更新，检查密码是否正确，检查回答密保问题是否正确。

AppUserManage 主要完成如下几个功能：第一，界面响应函数将参数传入这些类，然后这些类与数据层沟通，更新数据库。第二、界面响应函数将参数传入这些类，然后这些类与数据层沟通，获得构建界面的信息。

2、消息部分

本部分只介绍每一条消息的组成。该部分在表现层中对应三个用户控件，包括 Information 控件，Enclosure 控件，Response 控件，而后两个是 Information 的子控件。对应的类则包含 AppMessage 和 AppResponse 类两个类，其中前者以 List 容器的方式存有后者的引用。

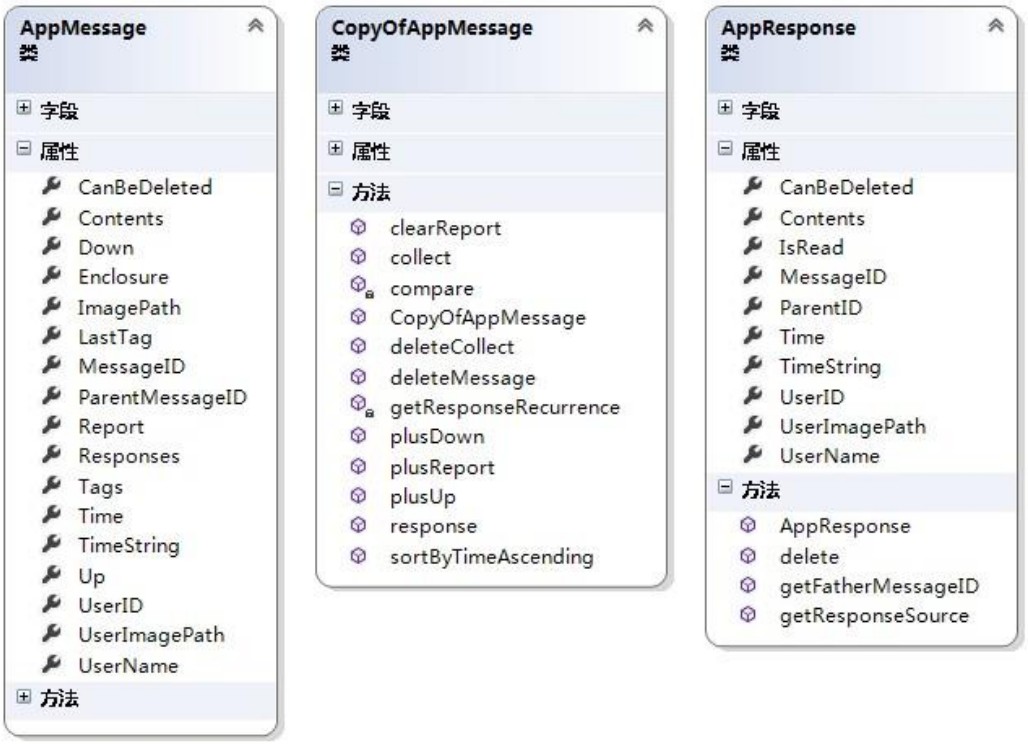
而在数据库中，一条消息和一个回复没有本质的区别，这是考虑到回复不仅仅可以针对消息，还可以针对回复。数据库的设计表为：

	名称	数据类型	允许 Null	默认值
🔑	messageID	int	☐	
	parentID	int	☐	((-1))
	userID	nvarchar(50)	☐	
	userimage	nvarchar(50)	☑	(N'')
	isread	bit	☐	((0))
	contents	nvarchar(200)	☐	
	tag1ID	int	☑	((-1))
	tag2ID	int	☑	((-1))
	tag3ID	int	☑	((-1))
	tag4ID	int	☑	((-1))
	image	nvarchar(50)	☑	(N'')
	enclosure	nvarchar(50)	☑	(N'')
	time	datetime	☐	
	up	int	☑	((0))
	down	int	☑	((0))
	report	int	☑	((0))

- 键 (1)
  - PK_Message (主键, Clustered: messageID)
- CHECK 约束 (0)
- 索引 (0)
- 外键 (1)
  - FK_message_userlog (userID)
- 触发器 (0)

字段从上到下分别代表：这条信息/回复的 ID，这条信息/回复的双亲 ID（例：若某一条回复是针对 ID 为 3 的信息的，则该回复的 ParentID 为 3），这条信息/回复对应的用户头像，这条信息/回复是否已被其双亲的主人读过，这条信息/回复的内容，这条信息对应的标签*4，这条信息所含的图像，这条信息所含的附件，这条信息/回复发表的时间，这条信息顶，踩，举报的个数。

AppMessage 和 AppResponse 的两个类的结构图如下所示：

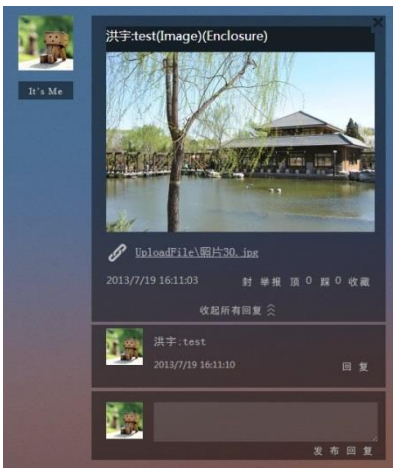


AppResponse 类的属性分别为：是否可删除，回复的内容，是否已读，这条回复的 ID，这条回复的父消息（回复）的 ID，回复的时间，回复的时间-字符串形式，回复的用户的 UserID，回复的用户的头像，回复的用户的昵称或 UserID（如果昵称不为空则该属性为昵称，否则为 UserID，即注册邮箱）。

该类的方法为：构造方法（给定回复 ID，当前用户，初始化里边的字段），删除回复，得到这条回复的双亲 ID（也就是这条回复究竟回复的是什么），得到这条回复所在的消息 ID（即顺着这个回复往回找，直到找到其隶属的消息）。

而 AppMessage 类中，除了回复所需的属性之外，还有如下属性：消息的附件、图片地址，顶、踩、举报的个数，消息所对应的标签，消息中所有的回复。而相关的方法为：清除举报个数，收藏，比较时间大小，构造方法，删除收藏，删除这条消息，递归的获得所有回复（因为考虑到回复的回复），增加消息的顶、踩、举报，回复这条消息，将一些消息按照时间降序排列。

页面中，一条消息完整的样子如下图所示：

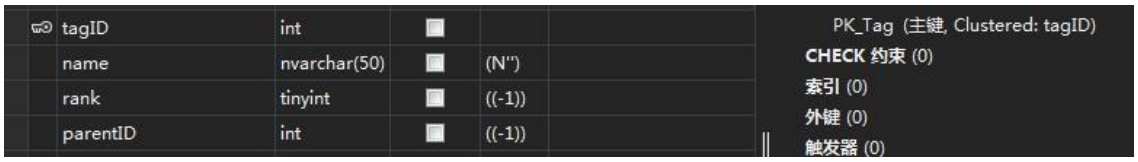


在页面逻辑的实现中，比较难的部分有两处：第一是对勾和叉子的实现。仅当管理员处在举报提醒页面中，对勾显示出来，此时点击对勾将标记这条消息的举报数量为 0，同时删除消息的显示。而删除消息的显示与消息所在的容器有关，因此采用了 C#反射机制，将删除消息的代码写到了父页面上。类似的，点击叉子消除消息也是如此，不过需要判断当前所在的页面，因为在收藏页面，点击删除的意义是删除收藏而不是删除消息本身。

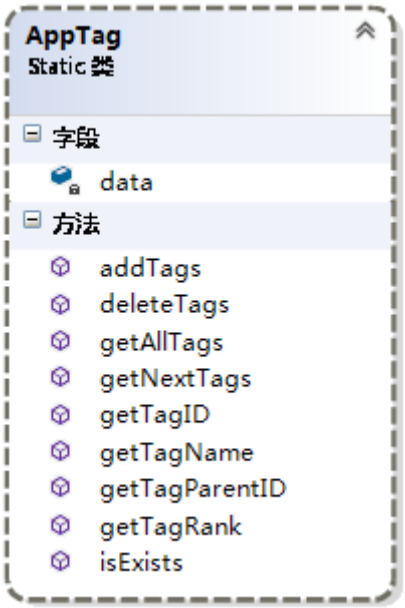
第二个难点是回复的回复。一方面，点击其他回复中的“回复”，将会在回复框中给予相应的提示。由于“回复”按钮在 Response 用户控件中，而提示应当在 Information 控件中，前者是后者的子控件，因此同样需要利用反射机制来实现。此外，需要存储一些必要的信息，来确认这条回复针对的是什么，才可以调用业务逻辑层来写入数据库。

3、标签部分

每一个消息都有相应的标签来记录消息的类别，而标签之间以树形结构组织起来。在数据库中，标签的存储如下图所示：



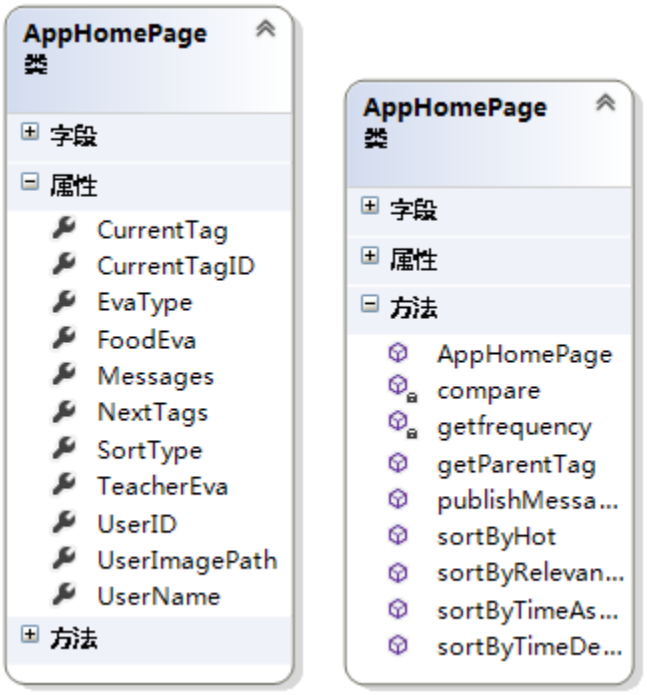
上图中的字段分别表示：标签 ID、标签名、标签等级、父标签的 ID。而标签对应的类为 AppTag 类，其类图如下：



这是一个静态类，里边的方法依次为：增加一个标签（如果必要，连带在 course 表或 dish 表中增加一条记录，给评分模块提供必要的信息），删除一个标签（连带删除子标签），得到所有的标签，得到该标签下的子标签，根据标签名字得到 ID，根据 ID 得到标签名、父标签 ID、标签等级，判断标签是否存在。

4、主页部分

主页部分对应的业务逻辑层为 AppHomePage.cs，该类提供显示主页所需的一切数据，为主页操作引发的数据更新提供接口。该类的类视图为：



该类的属性分别为：当前页面的标签、标签 ID，当前页面的评价种类（无/教师评价/食品评价），食品评价的引用，页面上的所有消息，下一级标签的名字，消息的排序方式，教师评价的引用，当前用户名，用户头像，昵称或邮箱。

相关的方法为：构造方法（根据用户，当前位置等信息构造这个页面所需的一切内容），比较两个时间的大小，得到某个关键词出现的频率，得到父标签的名字，发表一条信息，按照热度、相关性（适用于搜索，需提供关键词），对消息进行排序。

页面上的响应函数都可以通过调用 AppHomePage 类的函数来完成必须的操作，为了防止点击时刷新，采用了 AJAX 技术。另外，主页上还有一些内部控件利用反射机制而产生的函数。

5、上传文件与图像

在这里我们使用了不错的 Uploadify 上载控件，可以在 js 脚本中对他进行控制，配置参数完成不同的功能：

‘uploader’：控件图片路径

‘script’：Handler 文件

‘cancelImg’：取消图片路径

‘folder’：上载文件存放路径

‘queueID’ 上载队列所对应的控件 ID

‘multi’：支持多文件同时上传

‘OnUploadStart’：上载开始时执行的函数

‘onComplete’：文件上传结束后调用的函数

```
$("#uploadify").uploadify({
                'uploader': '../JS/jquery.uploadify/uploadify.swf',
                'script': 'UploadHandler.ashx',
                'cancelImg': '../JS/jquery.uploadify/cancel.png',
                //存放路径：Pages/UploadFile
                'folder': 'UploadFile',
                'queueID': 'fileQueue',
                'auto': false,
                'multi': false,
                'OnUploadStart': function () {
                    $('#uploadify').uploadifyUpload();
                },
                'onComplete': function (event, queueID, fileObj, response, data) {
                }
            });
```

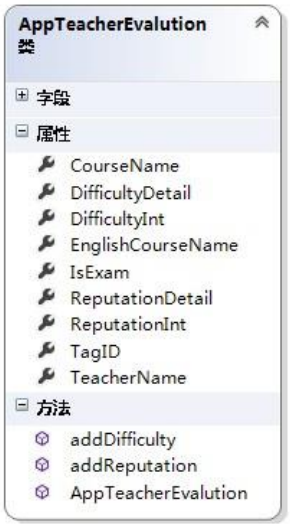
6、聊天室

多人聊天时，用户发布的信息能够被所有人看见，这也就是说用户的信息能够呈现在多人聊天窗口。当多人聊天时，不同的用户所打开的页面是不相同的，这样就造成可能信息呈现的时间不一致，为了保证信息的一致性，这里使用了 AJAX 的 Time 控件进行刷新。

聊天室的设计中使用了 Application，他的生命周期在整个网站的运行中。当用户发送一条信息时，利用 Session 记录用户的 ID，利用 Application 暂时存取聊天记录，Application 对象是页面中的公共对象，就算是不同的用户之间也能够共享 Application 对象，在页面进行聊天信息发布时，可以将值添加到 Application 对象中被其他用户读取。当用户单击按钮控件进行消息发布时，需要在页面中的相应位置进行呈现，在多人聊天代码实现中，可以直接将信息内容增加到文本框中，所以使用定时更新或者心跳感测可以实现群聊的功能。

7、评价部分

评价包括教师评价和食品评价，包含两个用户控件 Assessment.ascx，AssessmentFood.ascx。它们对应的两个后台为：AppTeacherEvalution.cs、AppFoodEvalution.cs。后台类的类视图为：



食品评价类的属性为：食品的名称，评价的详细内容，评价的整数值（用于前段显示），食品的标签，地点。方法为：增加一个评价，根据当前标签构造评价模块。

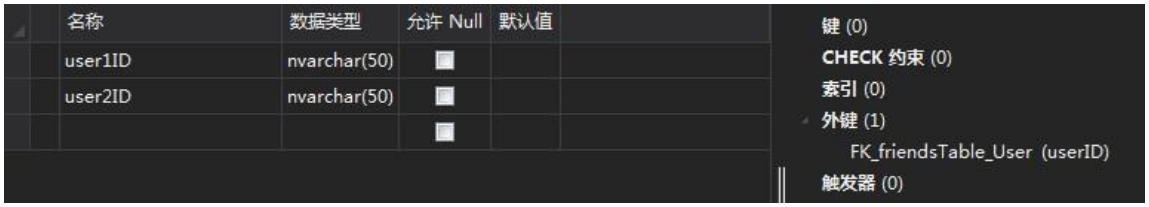
教师评价则包含课程信息，课程难度，教师评价等内容，与食品评价类似，不再赘述。

食品评价和教师评价的用户控件如下所示：

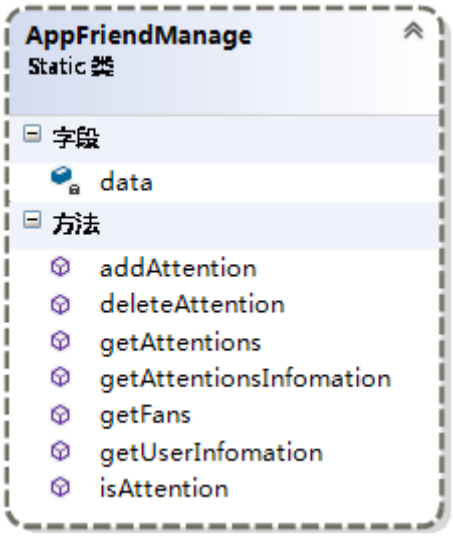
其中，您的评分这部分的 5 个星星都是 ImageButton，每个星星对应一个响应函数，它们之间的区别只是调用 addReputation 的参数和后边的显示不同。大众评分由 5 个 Image 组成，相邻两级差别是半个星星，每个 Image 显示的内容根据页面的 Page_Load 函数从业务逻辑层获得的数据而定。

8、好友管理部分

与好友管理页面相关的数据库表如下所示：



两个字段为 user1 的 ID 和 user2 的 ID，表示 user1 关注 user2。而该部分的后台为 AppFriendManage.cs，其类视图为：



字段是一个数据层 DataBase 对象的引用，而方法依次为：添加一个关注，删除一个关注，得到所有的关注，得到所有关注发送的所有信息，得到所有的粉丝，得到某一个关注所发的所有信息，判断是否关注。

对于页面部分，包括一个主页和两个小控件 MyFans 和 MyFriends。每个 MyFans 小控件代表一个粉丝，每个 MyFriends 代表一个关注，小控件中的用户信息可以从 AppUserManage 中获得，而加关注，取消关注等操作需要调用 AppFriendsManage 函数，并且需要通过反射机制来更新父页面 UpdatePanel 的显示以避免刷新。

而父页面 FriendsManagement.aspx 也需要通过 AppFriendsManage 获得关注信息等必要的信息，为了保证 Information 模块的完整性，父页面同样需要实现 Information 控件的反射调用，否则点击控件上的叉子（删除按钮）之后必须刷新页面才有效果。

9、回复提醒部分

在数据库的 Message 表中，回复和消息的唯一区别是双亲 ID 不同，消息的双亲 ID 是-1，而回复的双亲 ID 一定不是-1，而 Message 表中存有 isread，代表该消息/回复是否已读，这个标记也就成为回复提醒的基础。

回复提醒的后台类为 AppNotice.cs，其类视图如下所示：

Messages 属性存储了当前所有未读回复，UserID 存储当前用户，UserImagePath 存储当前用户的头像地址，UserName 存储当前用户的邮箱或昵称。

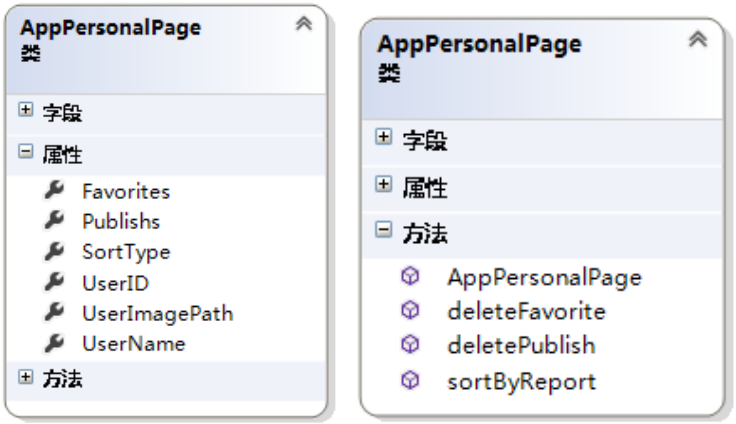
获得 Messages 对象需要检索数据库，找到未读消息/回复中，有关当前用户的消息/回复。然后再查找这些消息/回复所在的消息列表，整理后返回。

三个方法分别是：构造方法，删除一个提醒，统计提醒的数量。

前台的响应函数相对比较简单，一是通过 AppNotice 类获取未读消息，然后显示在前台，同时显示未读回复数量。二是实现“全部标记为已读”，需借助 AppNotice 类的 deleteNotice。三是实现 Information 的反射调用，给叉子赋予“删除这个未读消息”的响应。

10、 个人主页部分

个人主页包括我的发布，我的收藏，举报提醒，封号处理，添加标签，删除标签六个模块构成。添加标签，删除标签是单独的两个页面，对应的逻辑层是 AppTag 类，在个人主页上只需要做个链接即可。封号处理对应的后台是 AppUserManage 类，里边有获得所有被封用户，更改用户权限两个方法。而剩余的功能对应的业务逻辑为 AppPersonalPage 类，其类视图如下所示：



属性中：Favorites 是一个 List<AppMessage>，存储用户所有的收藏，Publish 也是上述类型，存储用户所有的发布，剩下的属性分别为排序的种类，用户的邮箱，用户的头像路径，用户的头像地址。

方法中：AppPersonalPage 通过 userID 获得该用户的个人主页内容。deleteFavorite 为删除用户的某一条收藏，deletePublish 为删除用户的某一条发布，sortByReport 将会返回所有被举报的消息并按照举报数量排序。

Thank you for your patience !