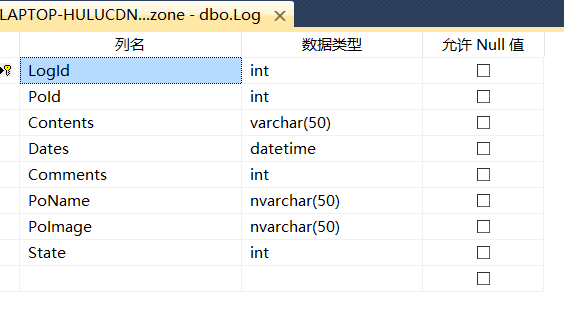
技术文档

——qq空间

**一、数据库设计**

1. 记录说说内容的表

表名：Log



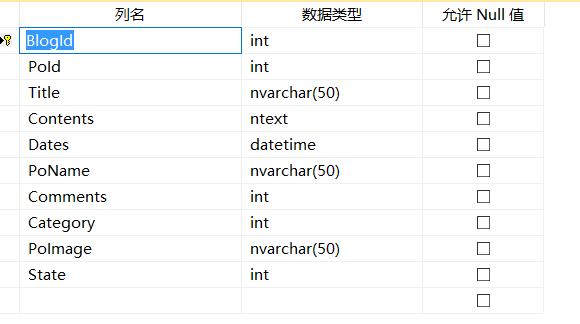
列名：

更新：Contents的类型改为nvarchar（1000）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LogId | PoId | Contents | Dates | Comments | PoName | PoImage | State |
| 主键  每条说说的id名 | 发表说说的用户的id | 发表的内容 | 发表的时间 | 该说说评论 | 发表人名字 | 发表人的头像 | 该说说的状态（若为1则是所有人可见，若为0则仅自己可见） |

1. 记录日志内容表

表名（Blog）



更新：Title的类型改为nvarchar（50）

列名：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BlogId | PoId | Title | Contents | Dates | PoName | Comments | Category | PoImage | State |
| 主键  每条日志的id 唯一标识 | 发表人的id | 日志标题 | 日志内容 | 发表日期 | 发表人名字 | 该日志的评论数量 | 日志的分类 | 发表人的头像 | 该日志的状态（若为1则是所有人可见，若为0则仅自己可见） |

1. 记录留言内容的表

表名（message）



更新：Contents的类型改为nvarchar（1000）

列名：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MessageId | PoId | Contents | Dates | PoName |
| 主键 每条留言的id 唯一标识 | 发表者的id | 留言内容 | 发表日期 | 发表者名字 |

|  |  |  |  |
| --- | --- | --- | --- |
| Comments | ParentId | ParentName | PoImange |
| 留言评论数量 | 被留言的用户id（即在谁的空间留言） | 被留言的用户姓名（即在谁的空间留言） | 发表者的头像 |

1. 记录相册的表

表名（Album）



更新：Describe的类型改为nvarchar（100）

列名

|  |  |  |  |
| --- | --- | --- | --- |
| AlbumId | AlbumName | OwnId | Describe |
| 主键 相册id 唯一标识 | 相册姓名 | 相册所有者的id | 该相册的简介 |

|  |  |  |  |
| --- | --- | --- | --- |
| AlbumImage | State | Dates | Category; |
| 存放相册封面的图片的路径 | 该相册的状态（若为1则是所有人可见，若为0则仅自己可见） | 创建相册的时间 | 该相册的分类 |

1. 记录照片的表

表名（Photo）



更新：Message的类型改为nvarchar（100）

列名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PhotoId | PoId | Contents | Dates | Message |
| 该照片的id 主键 唯一标识 | 发表者id | 存放该照片的路径 | 发表时间 | 该照片的简介 |

|  |  |  |  |
| --- | --- | --- | --- |
| Comments | PoName | AlbumId | PoImage |
| 评论数量 | 发表者的名字 | 该照片所属相册的id | 发表者的头像 |

1. 记录说说评论的表

表名（ComLog）



更新：Contents的类型改为nvarchar（100）

列名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentId | ToId | ParentId | UserId | Contents |
| 该条评论的id 主键 唯一标识 | 该条评论对应的说说的id | 被回复人的id（如果是直接在说说下面评论，则是发表说说的人的id 若是回复某条评论，则为这条评论发表人的id） | 发布这条评论的用户的id | 评论的内容 |

|  |  |  |  |
| --- | --- | --- | --- |
| Dates | UserName | ParentName | Information |
| 发表时间 | 发布这条评论的用户的姓名 | 被回复人的姓名（如果是直接在说说下面评论，则是发表说说的人的姓名 若是回复某条评论，则为这条评论发表人的姓名） | 如果该条评论不是自己评论自己，那么information设定为1，直至这条消息被 被评论的人看见，information则变为-0。用来标识信息是否已读 |

1. 记录日志评论的表

表名（ComBlog）



更新：Contents的类型改为nvarchar（100）

列名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentId | ToId | ParentId | UserId | Contents |
| 该条评论的id 主键 唯一标识 | 该条评论对应的日志的id | 被回复人的id（如果是直接在日志下面评论，则是发表日志的人的id 若是回复某条评论，则为这条评论发表人的id） | 发布这条评论的用户的id | 评论的内容 |

|  |  |  |  |
| --- | --- | --- | --- |
| Dates | UserName | ParentName | Information |
| 发表时间 | 发布这条评论的用户的姓名 | 被回复人的姓名（如果是直接在日志下面评论，则是发表日志的人的姓名 若是回复某条评论，则为这条评论发表人的姓名） | 如果该条评论不是自己评论自己，那么information设定为1，直至这条消息被 被评论的人看见，information则变为-0。用来标识信息是否已读 |

1. 记录照片评论的表

表名（ComPhoto）



更新：Contents的类型改为nvarchar（100）

列名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentId | ToId | ParentId | UserId | Contents |
| 该条评论的id 主键 唯一标识 | 该条评论对应的照片的id | 被回复人的id（如果是直接在照片下面评论，则是发表照片的人的id 若是回复某条评论，则为这条评论发表人的id） | 发布这条评论的用户的id | 评论的内容 |

|  |  |  |  |
| --- | --- | --- | --- |
| Dates | UserName | ParentName | Information |
| 发表时间 | 发布这条评论的用户的姓名 | 被回复人的姓名（如果是直接在照片下面评论，则是发表照片的人的姓名 若是回复某条评论，则为这条评论发表人的姓名） | 如果该条评论不是自己评论自己，那么information设定为1，直至这条消息被 被评论的人看见，information则变为-0。用来标识信息是否已读 |

1. 记录留言评论的表

表名（ComMessage）



更新：Contents的类型改为nvarchar（100）

列名

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentId | ToId | ParentId | UserId | Contents |
| 该条评论的id 主键 唯一标识 | 该条评论对应的留言的id | 被回复人的id（如果是直接在留言下面评论，则是发表留言的人的id 若是回复某条评论，则为这条评论发表人的id） | 发布这条评论的用户的id | 评论的内容 |

|  |  |  |
| --- | --- | --- |
| Dates | UserName | ParentName |
| 发表时间 | 发布这条评论的用户的姓名 | 被回复人的姓名（如果是直接在留言下面评论，则是发表留言的人的姓名 若是回复某条评论，则为这条评论发表人的姓名） |

1. 记录用户信息的表

表名（Users）



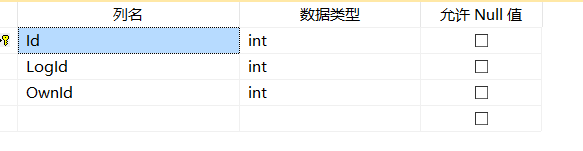
列名

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Name | Password | UsersName | Sex | Email |
| 主键 用户id | 用户姓名 | 用户密码 | 用户真实姓名 | 性别 | 邮箱 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Telephone | Birthday | Blood | QzoneName | State | Image |
| 电话号码 | 生日 | 血型 | Qq空间名 | 状态（刚注册的时候为0，当通过邮箱验证的时候边为1，每次进入登录页会删除数据库中state为0的用户 ） | 用户的头像 |

1. 记录说说的赞的表

表名（GoodLog）

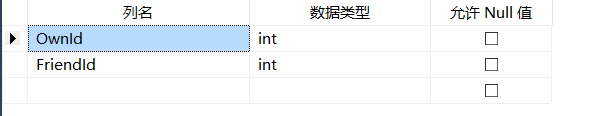


列名

|  |  |  |
| --- | --- | --- |
| Id | LogId | OwnId |
| 主键 每条赞的唯一标识 | 赞对应的说说的id | 点赞人的id |

1. 记录好友关系的表

表名（Friends）



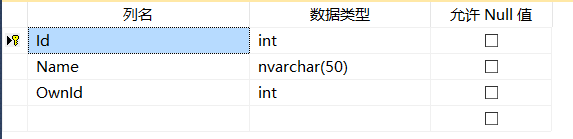
列名

|  |  |
| --- | --- |
| OwnId | FriendId |
| 用户id | 该用户关注好友的id |

（两个的关系为主键）

1. 记录相册分类的表

表名（AlbumCategory）

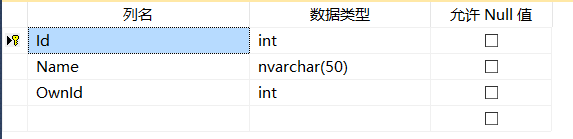


列名

|  |  |  |
| --- | --- | --- |
| Id | Name | OwnId |
| 主键 分类的唯一标识 | 分类名 | 该分类所属的用户的id |

1. 记录日志分类的表

表名（BLogCategory）

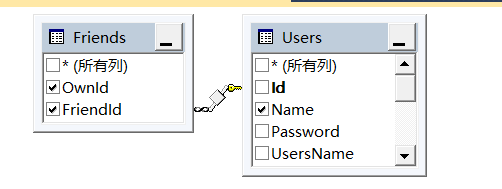


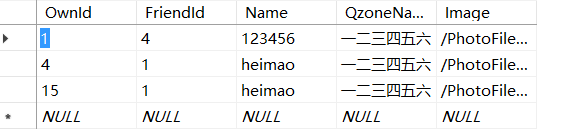
列名

|  |  |  |
| --- | --- | --- |
| Id | Name | OwnId |
| 主键 分类的唯一标识 | 分类名 | 该分类所属的用户的id |

1. 视图：有关好友的视图

视图名（View\_1）

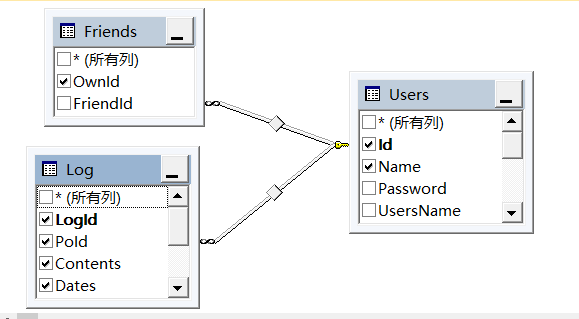




|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OwnId | FriendId | Name | QzoneName | Image |
| ；用户id | 用户关注好友的id | 用户好友的姓名 | 用户好友的空间名 | 用户好友的头像 |

1. 视图：有关好友说说动态的视图

视图名：LogF



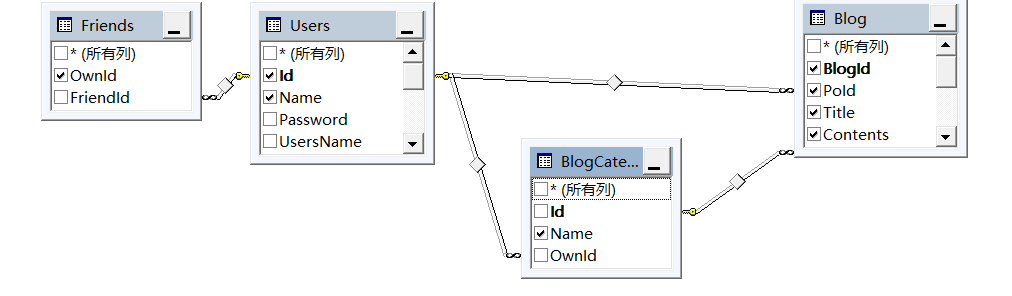


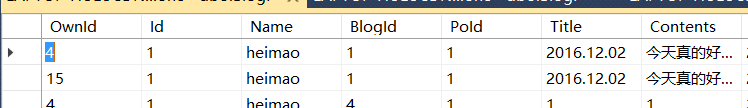
|  |  |  |
| --- | --- | --- |
| OwnId | Id | Name |
| 用户id | 用户好友的id | 用户好友名字 |

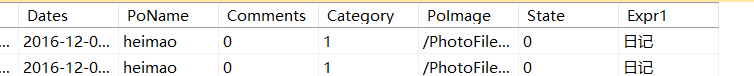
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LogId | PoId | Contents | Dates | Comments | PoName | PoImage | State |
| 好友说说的id名 | 发表说说的用户的id即用户好友的id | 发表的内容 | 发表的时间 | 该说说评论 | （用户好友）发表人名字 | （用户好友）发表人的头像 | 该说说的状态（若为1则是所有人可见，若为0则仅自己可见） |

1. 视图：有关好友日志动态的视图

视图名（BLogF）







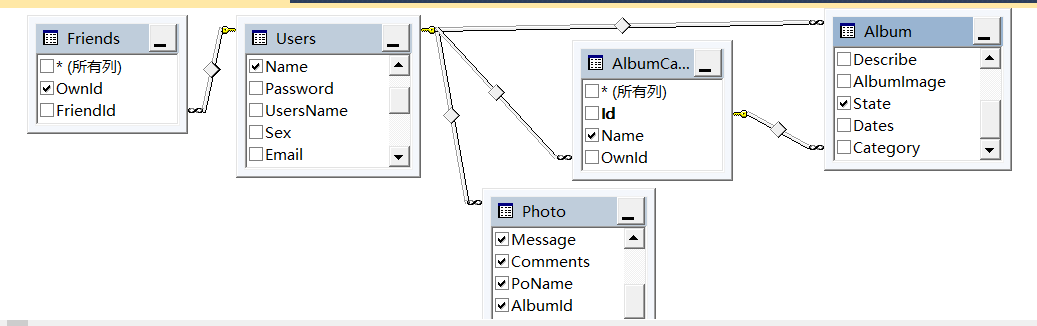
列名：

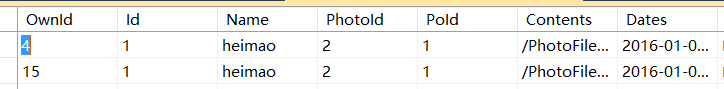
|  |  |  |
| --- | --- | --- |
| OwnId | Id | Name |
| 用户id | 用户好友的id | 用户好友名字 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BlogId | PoId | Title | Contents | Dates | PoName | Comments | Category | PoImage | State | Expr1 |
| 用户好友发表的日志的id | （用户好友）发表人的id | 日志标题 | 日志内容 | 发表日期 | （用户好友）发表人名字 | 该日志的评论数量 | 日志的分类 | （用户好友）发表人的头像 | 该日志的状态（若为1则是所有人可见，若为0则仅自己可见） | 日志的分类名 |

1. 视图 有关好友照片动态的视图

视图名（PhotoF）







列名

|  |  |  |
| --- | --- | --- |
| OwnId | Id | Name |
| 用户id | 用户好友的id | 用户好友名字 |

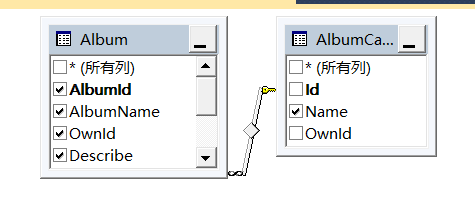
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PhotoId | PoId | Contents | Dates | Message |
| 用户好友发表的照片的id | （用户好友）发表者id | 存放该照片的路径 | 发表时间 | 该照片的简介 |

|  |  |  |  |
| --- | --- | --- | --- |
| Comments | PoName | AlbumId | AlbumName |
| 评论数量 | （用户好友）发表者的名字 | 该照片所属相册的id | 所属相册名 |

|  |  |  |
| --- | --- | --- |
| State | Category | PoImage |
| 该相册状态（若为1则是所有人可见，若为0则仅自己可见） | 所属相册分类 | （用户好友）发表者的头像 |

1. 视图 有关相册分类的视图

视图名（AlbumC）





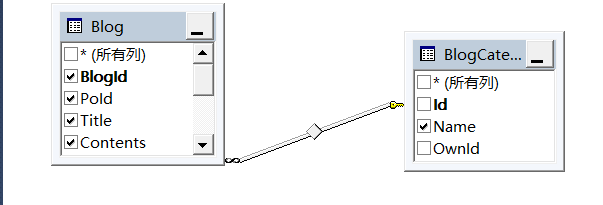
列名：

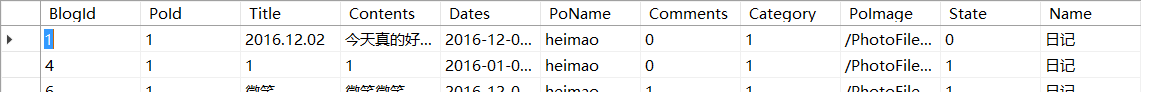
|  |  |  |  |
| --- | --- | --- | --- |
| AlbumId | AlbumName | OwnId | Describe |
| 主键 相册id 唯一标识 | 相册姓名 | 相册所有者的id | 该相册的简介 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AlbumImage | State | Dates | Category; | Name |
| 存放相册封面的图片的路径 | 该相册的状态（若为1则是所有人可见，若为0则仅自己可见） | 创建相册的时间 | 该相册的分类id | 该相册分类名 |

1. 视图 有关日志分类的视图

视图名（PhotoC）

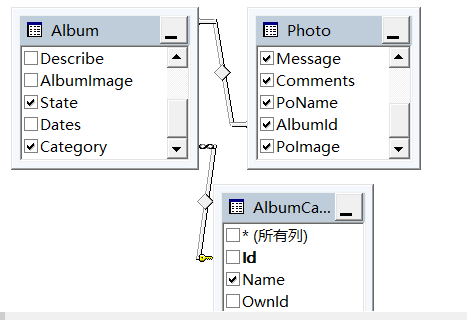


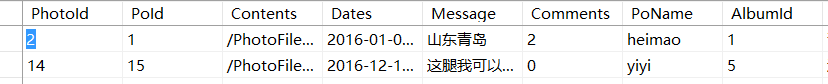


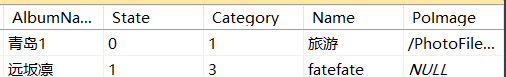
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BlogId | PoId | Title | Contents | Dates | PoName | Comments | Category | PoImage | State | Name |
| 主键  每条日志的id 唯一标识 | 发表人的id | 日志标题 | 日志内容 | 发表日期 | 发表人名字 | 该日志的评论数量 | 日志的分类id | 发表人的头像 | 该日志的状态（若为1则是所有人可见，若为0则仅自己可见） | 该日志分类名 |

1. 视图 有关照片和相册的视图

视图名（PhotoAlbum）





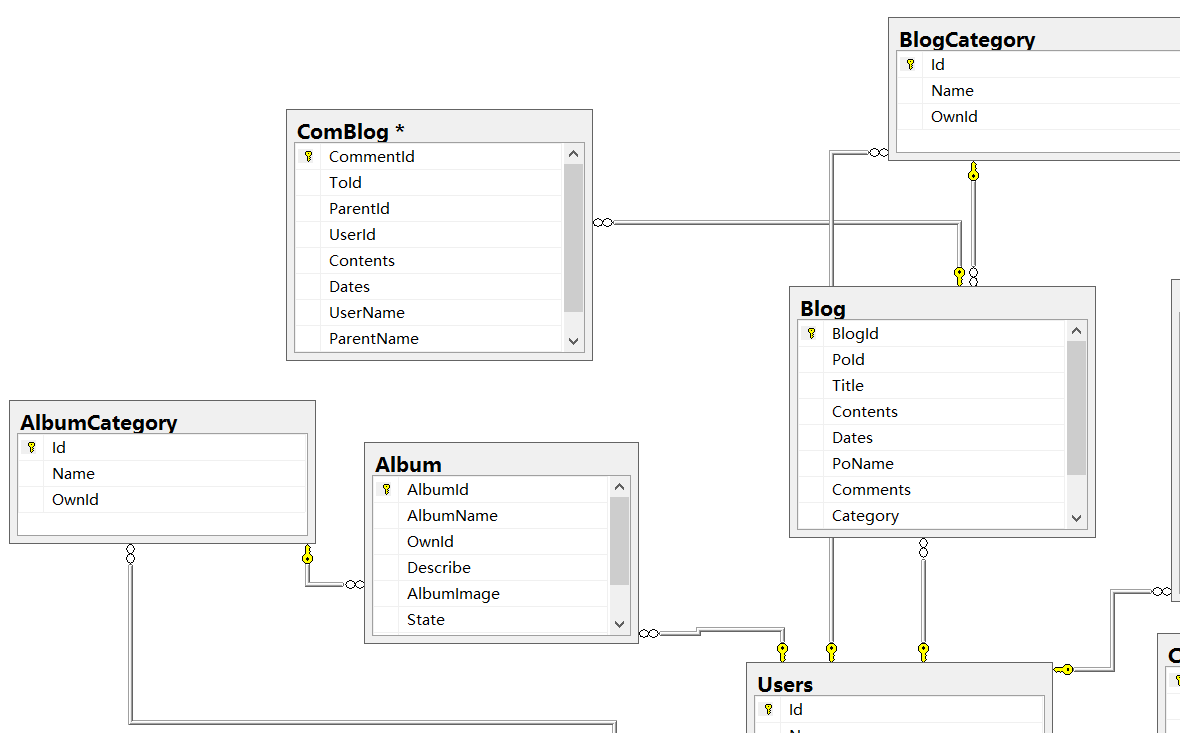


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PhotoId | PoId | Contents | Dates | Message |
| 该照片的id 主键 唯一标识 | 发表者id | 存放该照片的路径 | 发表时间 | 该照片的简介 |

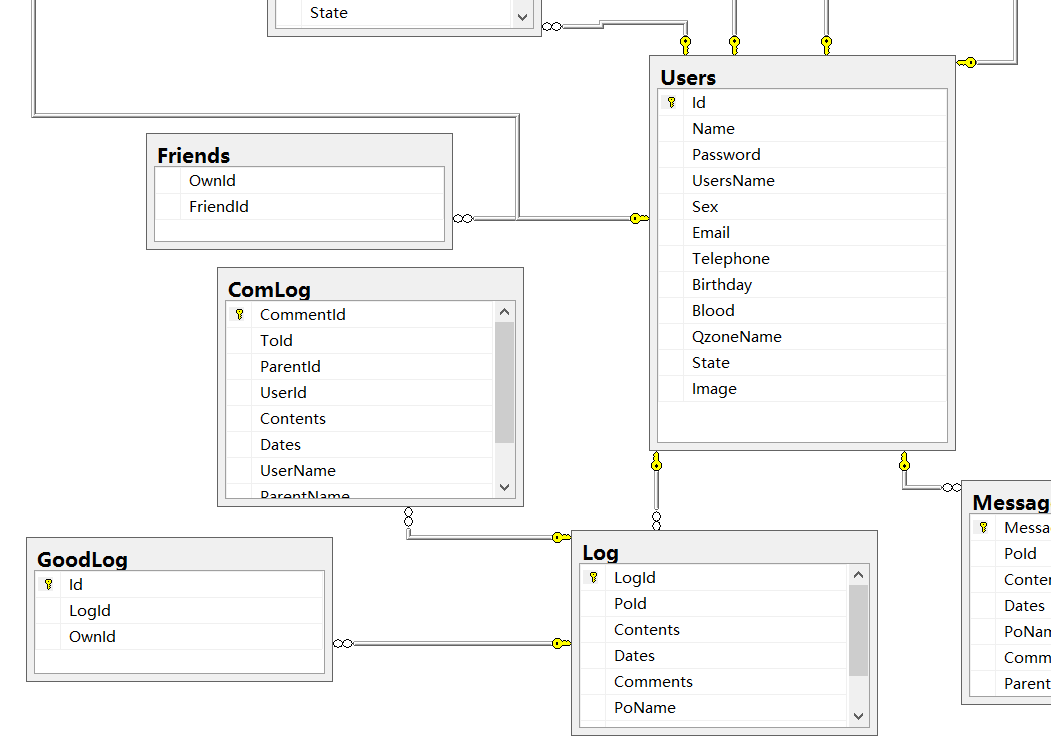
|  |  |  |  |
| --- | --- | --- | --- |
| Comments | PoName | AlbumId | AlbumName |
| 评论数量 | 发表者的名字 | 该照片所属相册的id | 所属相册名 |

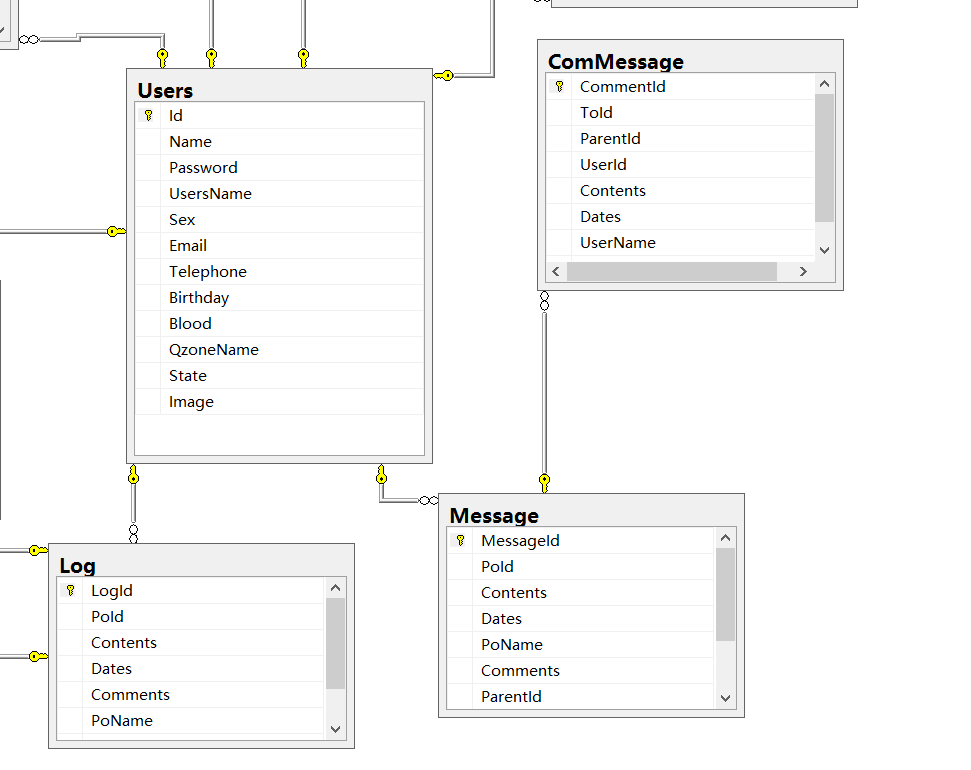
|  |  |  |
| --- | --- | --- |
| State | Category | PoImage |
| 该相册状态（若为1则是所有人可见，若为0则仅自己可见） | 所属相册分类 | 发表者的头像 |

**二、数据库关系**









图可能比较乱 我会一个个列出关系来

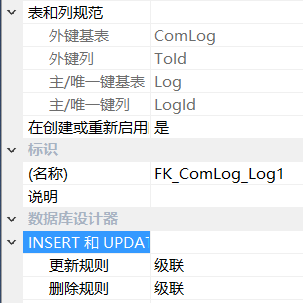
1. 主键：用户的id ；外键：说说表中发布说说者的id

当用户被删除时，该用户所有说说都会被删除



1. 主键：说说的id ；外键：说说的评论表中所对应说说的id

当该说说被删除时，该说说所有评论都会被删除



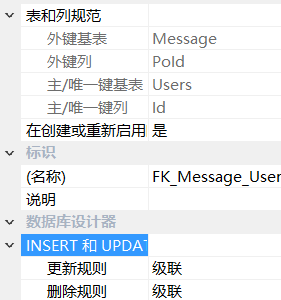
1. 主键：说说的id ；外键：说说的点赞表中所对应说说的id

当该说说被删除时，该说说所有赞都会被删除



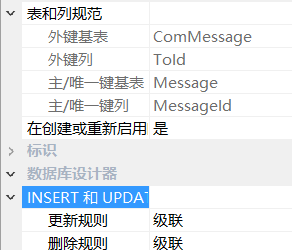
1. 主键：用户的id ；外键：留言表中发布留言者的id

当用户被删除时，该用户所有留言都会被删除



1. 主键：留言的id ；外键：留言的评论表中所对应留言的id

当该留言被删除时，该留言所有评论都会被删除



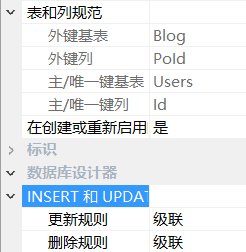
1. 主键：用户id 外键：用户关注好友的id

当该用户被删除时，该用户的被关注的关系会被删除



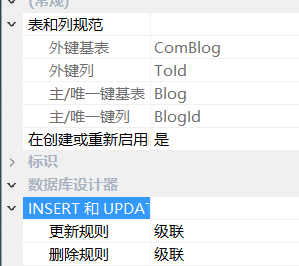
7、主键：用户的id ；外键：日志表中发布日志者的id

当用户被删除时，该用户所有日志都会被删除



1. 主键：日志的id ；外键：日志的评论表中所对应日志的id

当该日志被删除时，该日志所有评论都会被删除



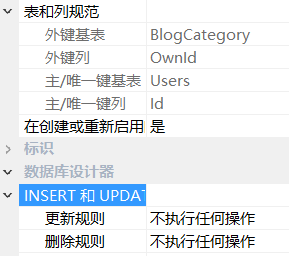
1. 主键：日志分类表中分类的id 外键：日志表中的日志分类id

当日志分类表中的某个分类被删除了 那么所属该分类的日志都将被删除



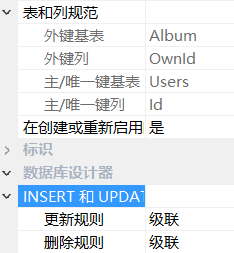
1. 主键：用户表中的用户id； 外键：日志分类表中该分类所属用户的id

当某个用户被删除了，该用户的所有分类被删除；（由于在sql server中这个关系的级联关系不能够被建立，于是我只是建立了关系并没有设置级联）



1. 主键：用户的id ；外键：相册表中发布相册者的id

当用户被删除时，该用户所有相册都会被删除



1. 主键：相册分类表中分类的id 外键：相册表中的相册分类id

当相册分类表中的某个分类被删除了 那么所属该分类的相册都将被删除



1. 主键：用户表中的用户id； 外键：相册分类表中该分类所属用户的id

当某个用户被删除了，该用户的所有分类被删除；（由于在sql server中这个关系的级联关系不能够被建立，于是我只是建立了关系并没有设置级联）



1. 主键：相册id 外键：照片表中该照片所属相册的id

当该相册被删除时，该相册下的所有照片都将被删除（由于在sql server中这个关系的级联关系不能够被建立，于是我只是建立了关系并没有设置级联，于是我选择了在代码里进行操作）



1. 主键：用户的id ；外键：照片表中发布照片者的id

当用户被删除时，该用户所有照片都会被删除

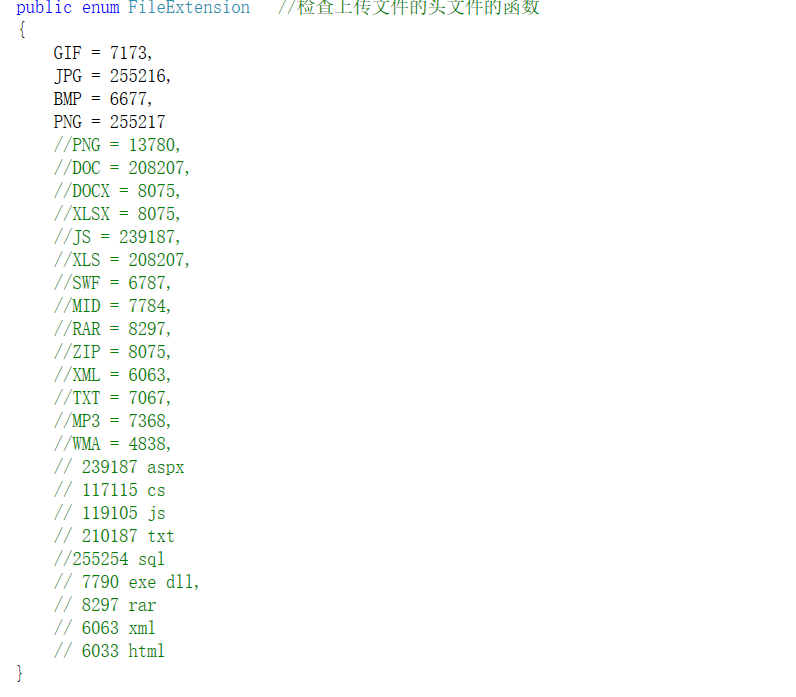


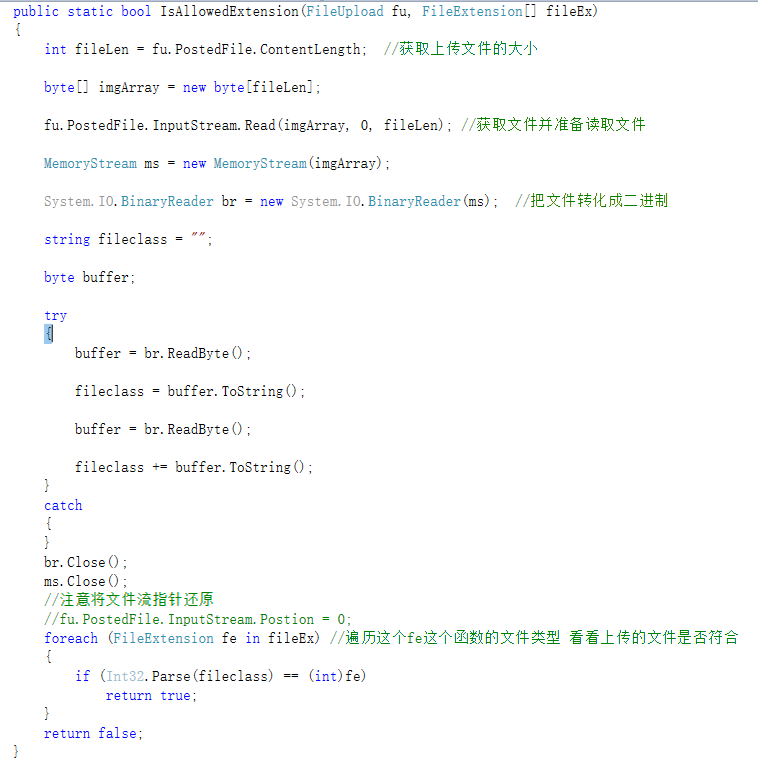
15、主键：照片的id ；外键：照片的评论表中所对应照片的id

当该照片被删除时，该照片所有评论都会被删除



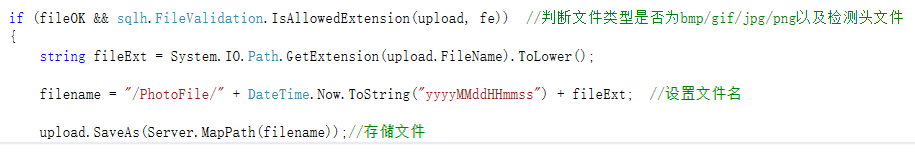
1. **所使用的一些方法类接口函数等等**
2. 有关上传图片的函数





原理是先解析上传文件的头文件，然后再遍历自己定义的fe函数中的文件类型，与上传文件进行对比，若相符（这个文件的类型是在这个fe函数里面的），则返回true

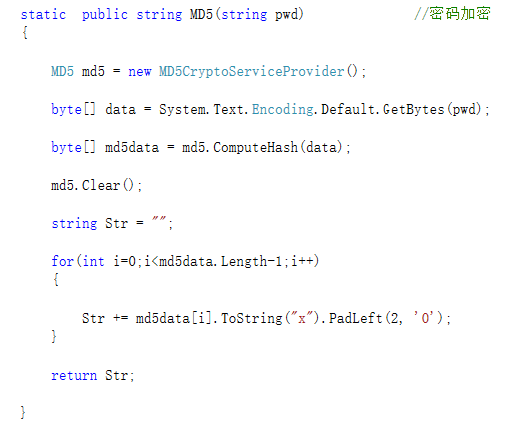




我选择了两种方法进行验证（双重保险）：

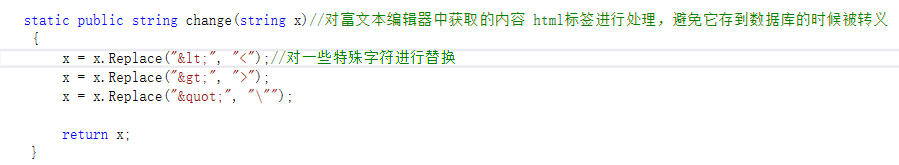
先判断是否选取文件再是检查文件后缀名是否是正确的再检查文件的头文件是否是自己想要的文件类型，当两种都通过时才可以上传文件并存储

1. 密码加密（MD5）



读取遍历字符串并对字符串的字符进行替换，这就起到了密码加密的作用

3、对富文本编辑器里的内容进行处理的函数



4、发送邮件



发送邮件需要在发送邮件的邮箱中设定smtp，在这个函数中设定好发送邮件的邮箱，以及密码（虽然没太大作用，但是我还是打了马赛克）。我是通过把被发送的邮件与邮件标题和邮件内容都传进来，来发送邮件（这样子比较通用），这次我只在注册，确认邮箱是否存在的时候使用了发送邮件，因为有个很重要的地方，发送邮件给同一个人同一个内容发送过多会被系统认定为垃圾文件，这就是一个无法解决的bug

5、分页



传进来的当前页的页数以及sql语句和对sql进行参数化处理的sqlparameter[] 数组，在函数里找出返回的datatable 再对这个datatable进行分页，返回一个分页对象

6、进行sql增删查改的语句

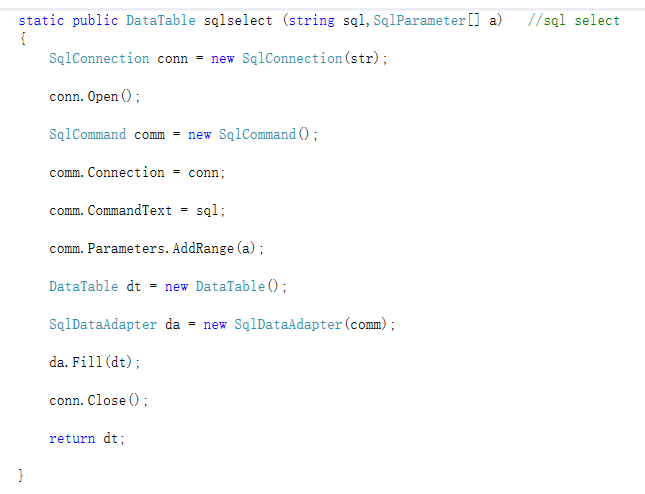
（1）增删改

传进来sql语句和对sql进行参数化处理的sqlparameter[] 数组，在函数里对数据库进行增删改了，返回被影响的行数

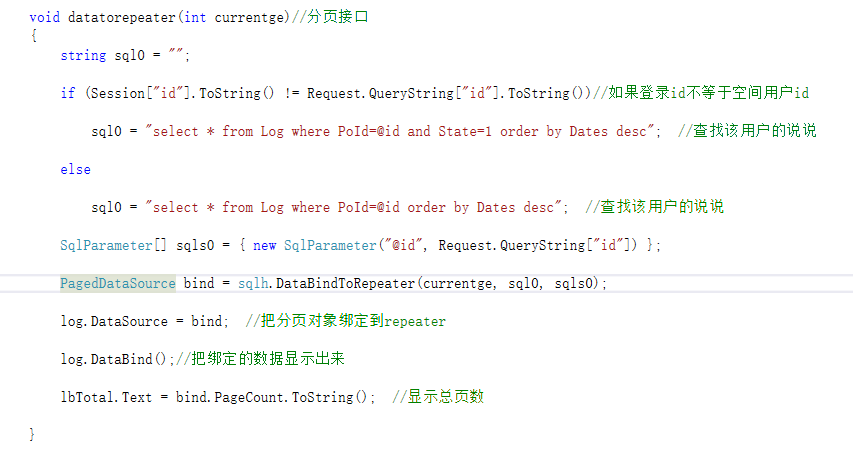


(2)sql查询

传进来sql语句和对sql进行参数化处理的sqlparameter[] 数组，在函数里对数据库进行查询了，返回查询出来的datatable



7、说说（Log）页里的分页接口

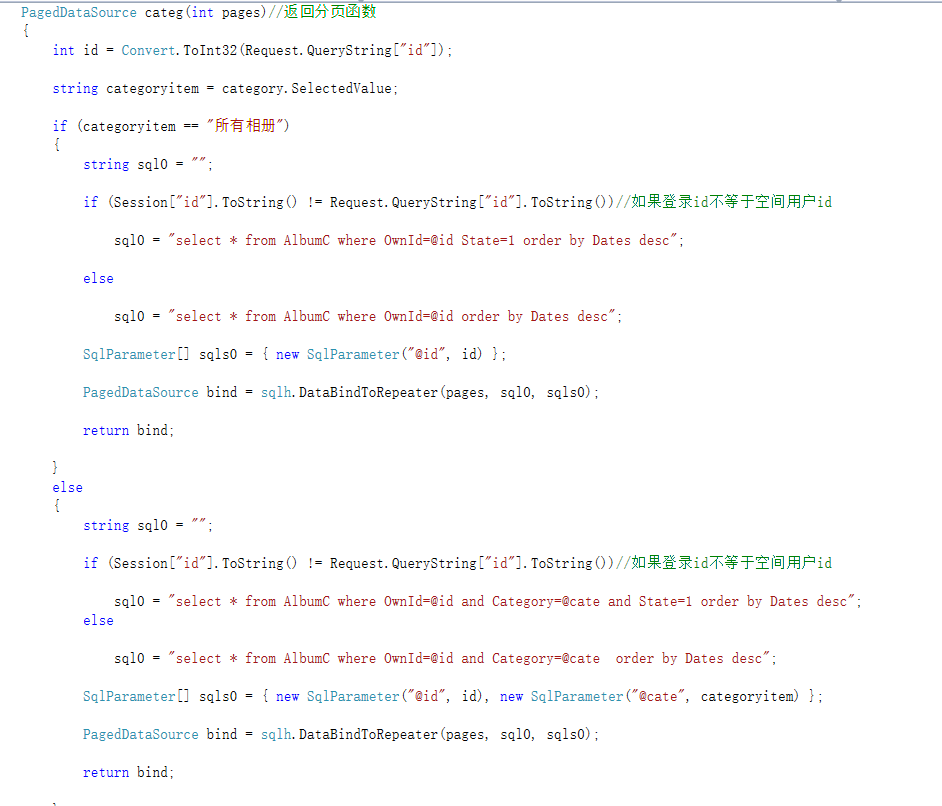


为了代码的简练性，写了一个接口

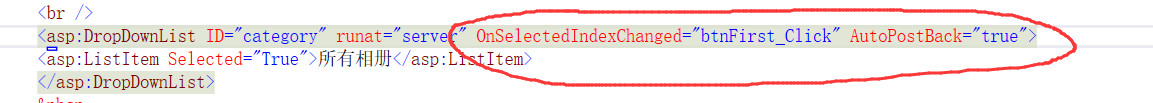
通过判定该空间的主人是否是登录用户，设定不同的sql语句（访问权限问题），对其进行分页并绑定，并显示总页数（因为不同的sql语句页数不同）

8、有关分类和权限的接口

以相册为例



通过判定该空间的主人是否是登录用户以及选择的分类，设定不同的sql语句（访问权限问题），对其进行分页并绑定，并显示总页数（因为不同的sql语句页数不同）



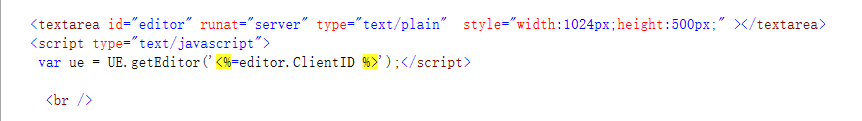
这里使用了dropdownlist的一个属性，可通过选择直接刷新页面

**四、一些处理**

1、富文本编辑器的使用



首先要下载百度ueditor这个.net的包，把这个包引进这个项目中，然后再在需要使用富文本编辑器的地方引用这些文件，并且修改这些文件的路径，还需修改一下最上头的一些属性（用红笔圈出来的地方）

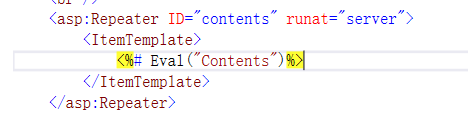


接下来就是实例化这个富文本编辑器，这个时候需要注意一个问题，因为我使用的是母版页，所以在实例化的时候需要获取id，这样子才能正确设置编辑器



要想把富文本编辑器中的内容显示出来

要用一个标签显示html的语言 可以使用repeater



2、对地址栏和模板页中的控件进行的处理





运用正则表达式对地址栏中的id进行判断，判断是否有id以及是否为数字

(用来防止地址栏被篡改)

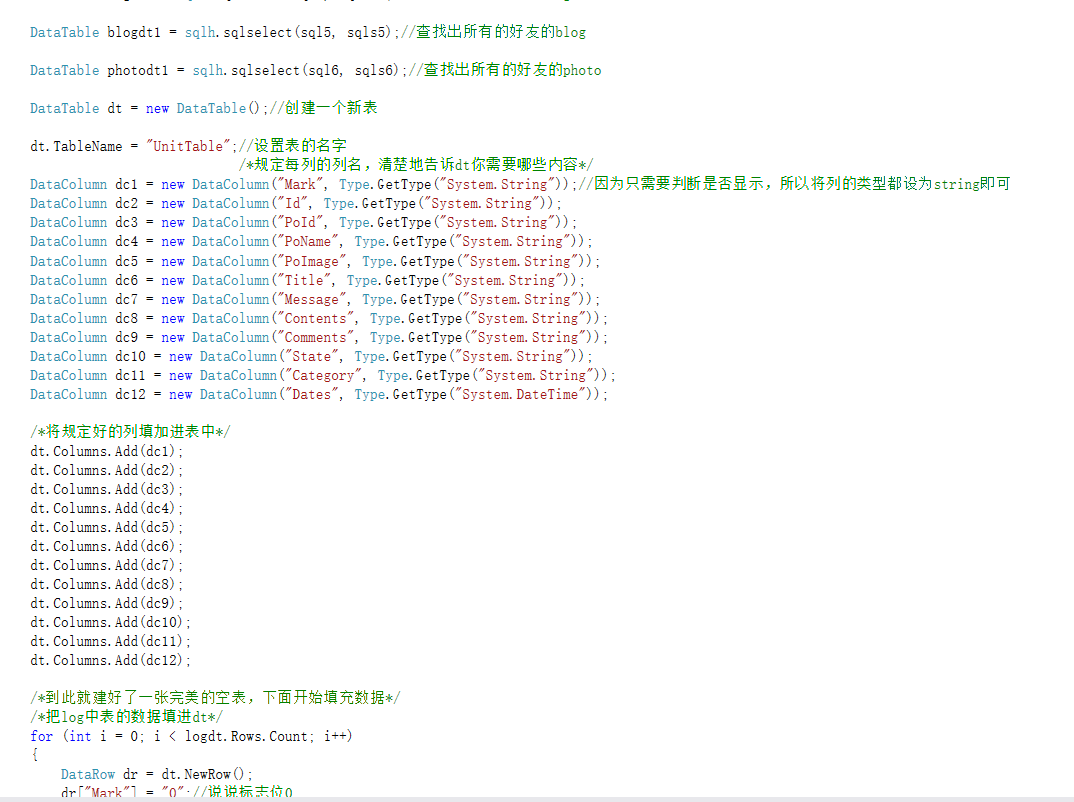
我是使用session来存登录用户的信息，然后用request querystring[“id”]来传递这个所在的空间主人的id,然后再根据一些情况设定request querystring[“id1”]来传递一些比如相册id之类的值,当有些页只能由登录用户访问的时候加入判断request querystring[“id”]是否等于session[“id”]

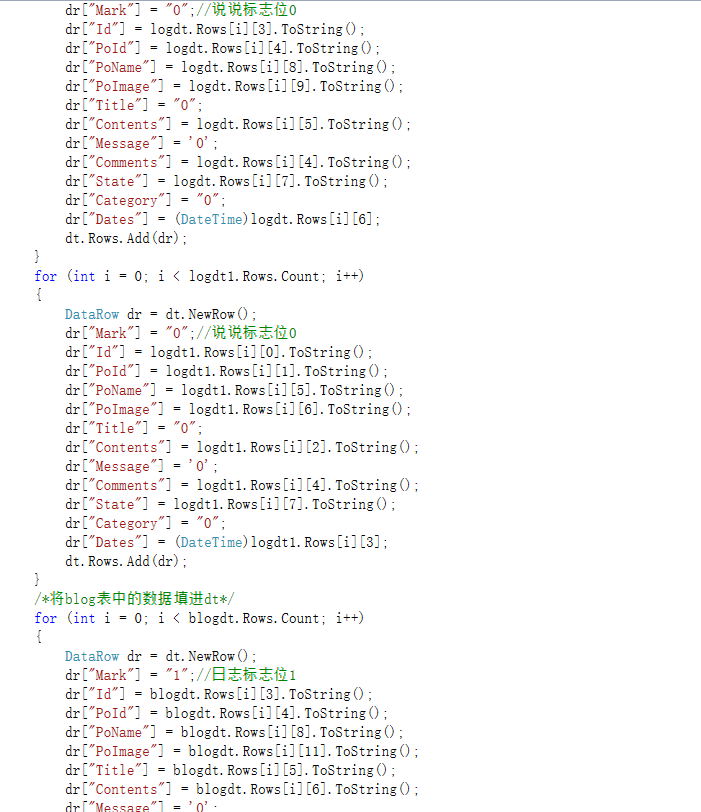
还会对传进来的id是否存在进行判断，并且根据查询出来的用户信息的值对模板页的控件进行赋值（因为母版页与内容的执行顺序是先执行内容页的pageload 再执行母版页的pageload，所以我选择了在每个内容页的pageload对母版页的控件进行赋值）。

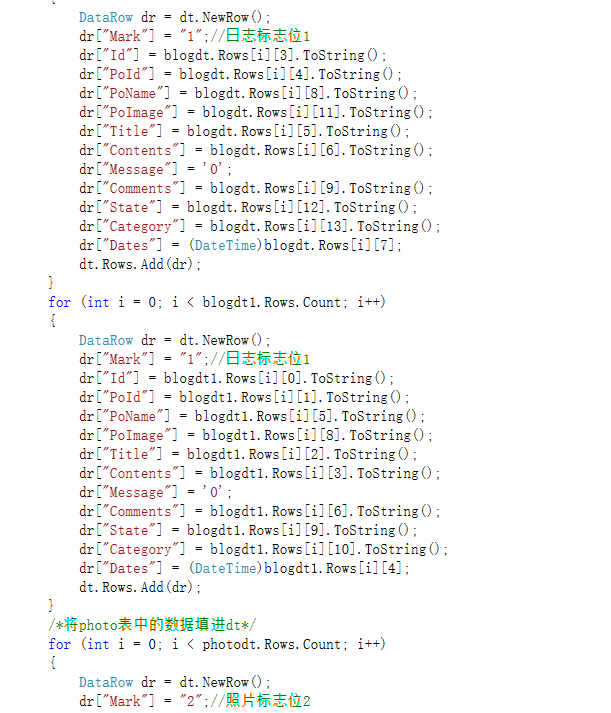
3、有关好友动态的一些处理！！！！！！

（1）在后台拼表











以好友动态的表为例

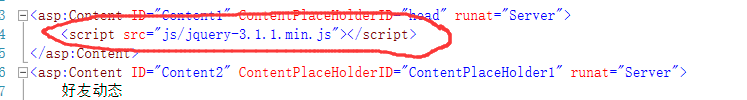
根据需求（好友的权限为所有人可见的说说日志照片的动态表，自己的所有的说说日志照片的动态表），我查找出了六个表，建立列名，设定标识位（判定是说说还是日志还是照片），把共有的列名以及某个表专有的列名（如日志表中的title）都建立出来。并对其进行填充（填充时，如果某个列没有值则设定为0）。

由于我要按时间进行排序，这里出现了一个问题（按照数据库里的datetime的类型存储的时间，如果以字符串的形式存在datatable中，字符就变了样式，这个时候在对新表进行排序就会出现bug，于是我采用了把数据按datetime的形式存进新表中，这个时候进行排序就没问题了）

（2）在前台进行显示

运用了一些js的东西，根据标识，显示不同的div

A、首先要引用了jquery文件，才能使用



B、设定一个repeater在repeater里面设定三个div,，分别显示说说 日志 照片先把他们的属性都设置为空

这个时候需要一个标签来设定标识

然后我是选择了每个div都嵌套了一个评论repeater 通过判断是按下了哪个评论repeater来判定这个动态是说说还是日志还是照片，来在进行不同repeater的嵌套

然后通过判定是按下哪个repeater的哪个按钮来判定对哪个评论表里的评论进行操作（这是一个大坑微笑）

这个时候顺便说一下我对说说日志照片进行的处理

评论

点击别人的评论内容 会跳出一个框对别人的评论进行评论。

直接点击评论的话，是对这条说说进行评论。

转发

转发说说日志照片可以选择权限

转发日志的时候需选择分类，若该主人的并没有设定好的分类那么就会有个linkbutton跳转到分类页对其进行添加

转发照片的时候也需要选择转发到哪个相册，若该主人的并没有相册那么就会有个linkbutton跳转到新建相册页对其进行添加。如果该用户没有相册分类，此时也会需要跳转到新建分类页对其进行操作

删除

可以删除自己回复的那条评论

以及自己发的动态下的评论

一些控件隐藏的处理

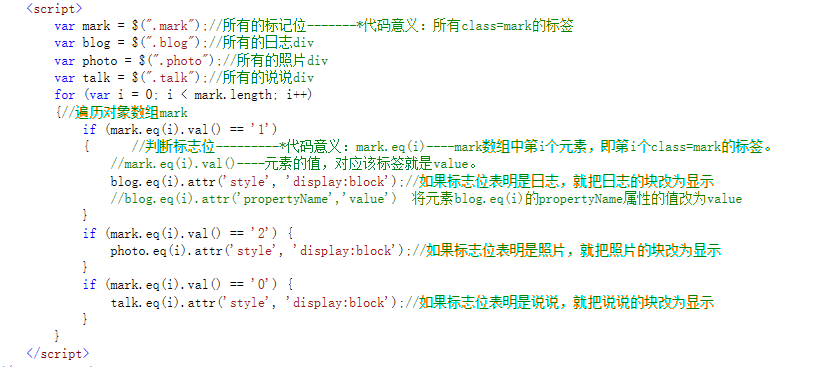






C、这个是js的坑了

注释写得很清楚就不多说了



1. 消息的处理

时间问题，我只设置了在说说，日志，照片下面被评论，或者说用户的评论被回复了，会被提示。以及点进去该条消息可以查看该条消息所对应动态的详情，并对其进行操作

1. 弹出提示选择是否的弹窗



只有按‘是‘的时候才会继续往下执行该按钮我触发的事件

1. 对用户状态的一些处理

由于我使用了邮件确认功能，没被确认的用户state=0 我在登录页进行更新数据

删除所有state=0的用户