Love Space

《软件设计文档》

文件状态： [ ] 草稿 [√] 正式发布	文件标识：	软件设计文档
	当前版本：	1.0
	作 者	邓乐、张寅、胡伟特
	完成日期：	2013-7-21

1.技术选型及其理由

1.1 技术目标

快速开发一个前端交互频繁、后端处理方便且数据库处理高效的社交网站。

1.2 技术选型

后端：选用 Django 作为开发的后端框架，Mysql 作为数据库。

前端：选用 Jquery、Backbone、Uploadify、Bootstrap、FancyBox、FontAwesome 用于前端开发。

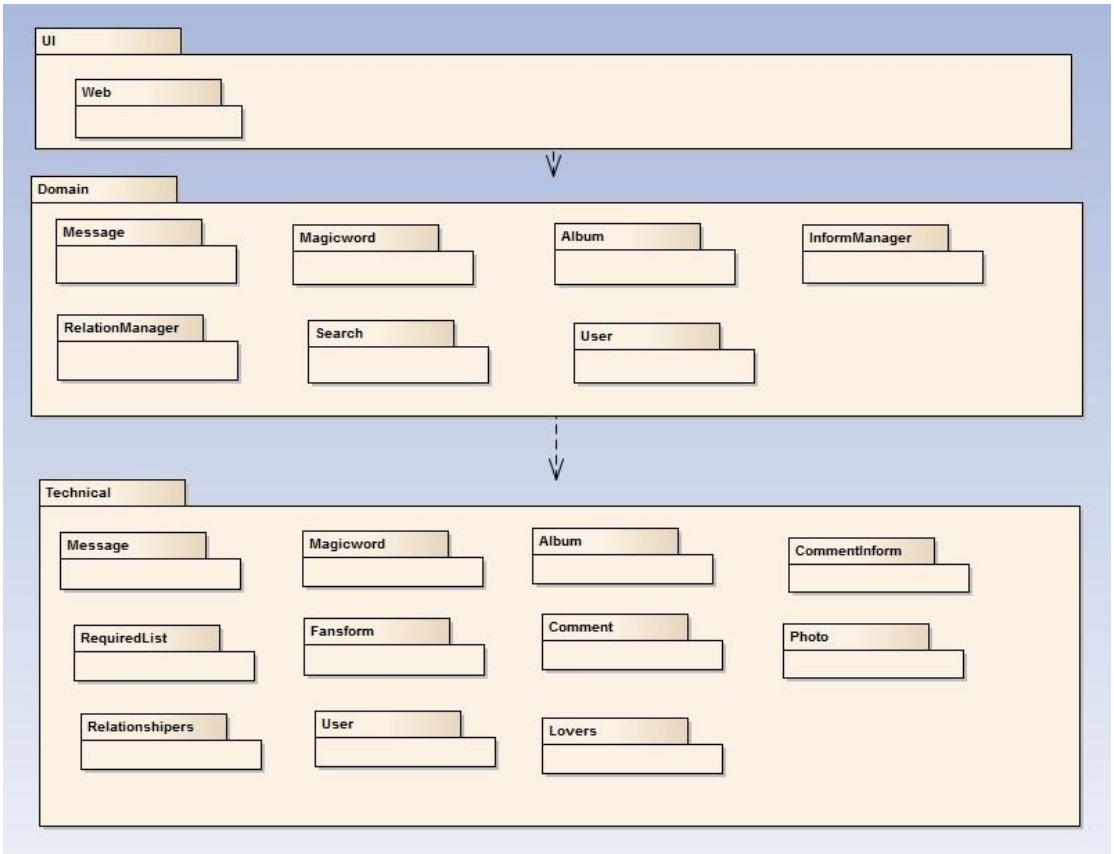
1.3 选型理由

后端：Django 是当前成熟的一款 MTV 框架，将模型、逻辑、视图很好的分离便于程序高效地开发，并且提供了简单的 ORM 和利于动态语言开发的模板视图。Mysql 是当前最好的开源关系型数据库管理系统之一，轻量高效并且易于配置。

前端：Jquery 是目前最成熟的 js 前端框架之一，实现了许多底层常用的 js 代码。Backbone 框架便于将大量的前端异步请求模块化便于代码编写与管理，Uploadify 提供异步传送图片的方法，Bootstrap 提供了丰富的前端样式，便于快速地实现布局样式和 js 效果，FancyBox 提供了漂亮的图片阅览，使浏览单张图片时其大小适应屏幕大小而随时变化，FontAwesome 提供了大量的图标。

2.架构设计

2. 1 包图



2.2 数据库

2.2.1.表设计

数据库共有 11 张表，分别是：

表名	作用
User	存放用户基本信息
Lovers	存放情侣空间的基本信息
Album	存放相册的基本信息

Photo	存放相册图片的基本信息
Comment	存放状态的评论的基本信息
CommentInform	存放评论提醒的信息
FansInform	存放粉丝提醒的信息
Magicword	存放双人密语的信息
Message	存放状态的信息
Relation	存放关系（粉丝、关注）的信息
RequestList	存放邀请（开通情侣空间）的信息

2.2.2.详细信息

⑴用户表 User 详细信息

表项	类型	备注
Id	int(11) auto_increnment	用户 id
name	varchar(15)	用户名
password	varchar(50)	用户密码
email	varchar(75)	用户邮箱
sex	varchar(1)	用户性别
age	int(11)	用户年龄
spouseId	int(11)	用户伴侣 id
avatar	varchar(11)	用户头像信息

⑵情侣空间表 **Lovers** 详细信息

表项	类型	备注
Id	int(11) auto_increnment	情侣空间 id
lover1_id	int(11)	恋人 1 用户 id
lover2_id	int(11)	恋人 2 用户 id
name	varchar(20)	情侣空间名字
notice	int(11)	情侣消息数
fans	int(11)	情侣粉丝数量
time	datetime	情侣相爱开始时间
avatar	varchar(11)	情侣空间头像信息

⑶相册表 **Album** 详细信息

表项	类型	备注
id	int(11) auto_increnment	相册 id
loversId	int(11) foreign_key	相册所属情侣空间 Id
name	varchar(100)	相册名字
descri	varchar(100)	相册描述
time	datetime	相册建立时间

photonum	int(11)	相册含有的相片数量

⑷相片表 **Photo** 详细信息

表项	类型	备注
id	int(11) auto_increnment	相片对应 id
album_id	int(11) foreign_key	相片所属相册 Id
name	varchar(100)	图片名字
descri	varchar(100)	图片描述
time	datetime	图片发布时间

⑸评论表 **Comment** 详细信息

表项	类型	备注
id	int(11) auto_increnment	评论对应 id
userId	int(11)	评论对应用户 Id
messageId	int(11) foreign_key	对应状态的 id
content	varchar(100)	评论内容
time	datetime	评论的时间

⑹评论提醒表 **CommentInform** 详细信息

表项	类型	备注
uid	int(11) foreign_key	用户 id
num	int(11)	用户收到的评论数量

⑺粉丝提醒表 **FansInform** 详细信息

表项	类型	备注
uid	int(11)	用户 id
num	int(11)	用户情侣空间粉丝的数量

⑻双人密语表 **Magicword** 详细信息

表项	类型	备注
id	int(11) auto_increnment	密语 id
userId	int(11)	发布密语的用户 Id
loverId	int(11) foreign_key	密语所属的情侣空间 Id
content	varchar(100)	密语的内容
time	datetime	密语发布的时间

⑼状态表 **Message** 详细信息

表项	类型	备注
id	int(11) auto_increnment	状态 id
userId	int(11)	发布状态的用户 Id
loverId	int(11) foreign_key	状态所属的情侣空间 Id
content	varchar(100)	状态的内容
time	datetime	状态发布的时间
image	varchar(1)	状态含有的图片信息

⑽关系表 **Relation** 详细信息

表项	类型	备注
id	int(11) auto_increnment	粉丝关注 id
from_uid	int(11)	粉丝的用户 id
to_lid	int(11)	关注情侣空间的 id

⑾请求列表 **RequestList** 详细信息

表项	类型	备注
id	int(11) auto_increnment	请求的 id
from_uid	int(11)	请求用户的 id
to_lid	int(11)	被请求用户的 id

3.模块划分

3.1 用户管理

功能：用户注册、登录、退出、单身用户登录判断、用户分类、邀请伴侣、拒绝伴侣、接受伴侣、单身用户关键字搜索单身用户和情侣空间、有偶用户关键字搜索所有用户和情侣空间等。

文件说明：

文件名	函数名	功能
./users/views.py	register(request)	处理注册请求返回注册模板
	register_finish(request)	处理用户填写的注册表单信息完成注册
	login(request, error)	处理登录请求返回登录模板
	login_finish(request)	处理登录表单数据包括用户名、密码用于完成登录
	logout(request)	处理退出重定向到登录

	require_login(view)	使用 AOP 用于登录判定
	user_deal(request, *args, **kwargs)	用于判定用户是否单身用于登录或注册后的页面跳转
	lovers_deal(request, lid)	处理有偶用户的登录或注册后的跳转
	search(request, *agrs, **kwargs)	处理单身用户对用户和情侣空间的搜索
	request_lover(request, id, to_id)	处理用户邀请其他用户开通情侣空间
	request_delete(request, id, to_id)	处理用户删除自己发出的请求
	refuse_request(request, id, from_id)	处理用户拒绝其他用户的情侣邀请
	accept_request(request, id, from_id)	处理用户接受其他用户的情侣邀请
	avatar_process(request, dir_name, id)	处理头像图片，将原图裁剪为原图、中、小的规格
	lover_search(request, *agrs, **kwargs)	有偶用户对用户和情侣空间的搜索
./static/js/user_home.js	init_accept_button()	处理用户点击接受添加情侣按钮
	init_refuse_button()	处理用户点击拒绝添加情侣按钮
./static/js/search.js	Search(闭包)	用于搜索功能

	Search 的 Concern	关注的 Model，用于后续处理后端数据库的增删
	Search 的 ConcernView	集成关注的点击动作，包括触发函数
	Search 的 add_request	处理点击邀请情侣按钮
	Search 的 delete_request	处理撤销邀请情侣按钮
	Info(闭包)	单实例，用于保存全局变量

3.2 状态功能

功能：异步发送图片文字以及包含表情的消息、异步评论消息、获取用户个人情侣空间消息、获取用户关注空间消息。

文件说明：

文件名	函数名	功能
./message/view.py	msg(request, *args, **kwargs)	处理空间主页获取自己情侣空间状态的请求
	concern_msg(request, *args, **kwargs)	处理用户个人关注情侣空间的状态的请求
./message/ajax.py	send_text(request, *args, **kwargs)	处理异步发送文字状态的请求
	get_comment(request, *args, **kwargs)	处理异步发送文件的请求
	delete_comment(request, *args, **kwargs)	处理异步删除评论的请求
	move_image_to_dir(request, mid)	将发布状态前异步发送的图

		片从临时文件正式存储到对应的位置，并将图片处理为原图、中图、小图、超小图
	uploadify_script(request, *args, **kwargs)	处理发布状态前异步发送图片并将图片裁剪成原图、小图规格存放为临时文件名
	delete_message(request, *args, **kwargs)	处理异步删除状态
./static/js/message.js	Message(闭包)	处理状态消息的闭包
	Message 的 init_uploadify	初始化 uploadify 用于异步发送图片
	Message 的 Msg	Message 的 Model 用于后续后端数据库增删
	Message 的 Comment	Comment 的 Model 用于后续后端数据库增删
	Message 的 MessageView	用于集成各种事件触发处理，包括异步发送状态、获得评论等
	Message 的 delete_msg_or_comment	异步删除状态或者评论
	Message 的 send_message:	异步发送状态
	Message 的 get_comment	异步获得评论
	Message 的 send_comment	异步发送评论
	Message 的 image_click	异步点击图片获得大图

	Info(闭包)	用于存储全局变量
	Init_qqFace	用于初始化 qq 表情，用于发布消息时可添加表情

3.3 双人密语

功能：发送双人密语，获取双人密语。

文件说明：

文件名	函数名	功能
./magicword/view.py	magic_word(request, *args, **kwargs)	处理空间主页获取自己情侣空间双人密语的请求
./magicword/ajax.py	send_text(request, *args, **kwargs)	处理用户在自己情侣空间发送双人密语请求
	delete_magicword(request, *args, **kwargs)	处理用户在自己情侣空间删除双人密语请求
./static/js/magicword.js	init_send_magicword_button	初始化发送双人密语按钮
	init_close_button()	初始化删除获得的双人密语按钮
	submit_magicword_by_ajax	处理点击发送按钮异步发送双人密语
	close_click	处理点击删除双人密语按钮
	ajax_delete(id, Data, Url)	处理异步发送删除至后端请求

3.4 相册功能

功能：相册、照片分页显示、删除，单张照片的浏览（适应屏幕的大小而变化尺寸）。

文件说明：

文件名	函数名	功能
./album/view.py	album_show(request, *args, **kwargs):	相册分页显示,每页 6 个相册
	album_add(request, *args, **kwargs):	添加相册
	delete_album(request, *args, **kwargs):	异步删除相册
	photos_show(request, *args, **kwargs):	照片分页显示，每页 12 张照片,点击一张图片后显示该张图片,且大小适应屏幕大小而变化，可左右切换
	photos_add(request, *args, **kwargs):	异步添加照片
	delete_photo(request, *args, **kwargs):	异步删除照片
	albumAddValid(request):	验证相册资料是否合法
	photoAddValid(photo, errors):	验证上传的照片格式等是否合法
	randomstr(n):	对上传的照片重命名为 16 个随机数字、字母字符

./static/js/uploadPhoto.js	init_uploadify = function()	初始化上传照片
	delete_album_or_photo: function(e)	得到删除相册或照片所需变量
	ajax_delete = function(id, Data, Url)	异步删除相册或照片

3.5 粉丝关注管理

功能：用户添加情侣空间为关注、取消关注、获取关注列表、获取粉丝列表。

文件说明：

文件名	函数名	功能
./users/RelationManager.py	get_fans_list(lid)	用于获取情侣空间的粉丝列表
	get_concern_list(uid)	用于获取用户的关注列表
	get_search_list(uid, lid, single, content)	用于获取搜索得到的用户和情侣空间以及与用户自身的关系(如是否关注对应情侣空间)
./users/view.py	add_concern(request, *args, **kwargs)	处理用户添加情侣空间为关注
	delete_concern(request, *args, **kwargs)	用于用户取消对已关注情侣空间的关注

3.6 消息管理

功能：用户个人情侣空间的新评论提醒消息、新粉丝提醒消息、获取评论消息信息、获取粉丝消息信息。

文件说明：

文件名	函数名	功能
./loveSpace/public.py	init_inform(uid)	用户注册成功后为用户在数据库添加对应消息提醒条目
	add_inform(tbl_name, lid)	用户情侣空间获得新的评论或者有了新粉丝时更新对应消息提醒的数据库条目
	get_inform(uid)	获取用户情侣空间的消息提醒
	delete_inform(tbl_name, uid)	删除用户情侣空间的消息提醒

3.7 其它

功能：对用户访问页面的分类

文件说明：

文件名	函数名	功能
./loveSpace/public.py	user_dispatch(process):	用 AOP 技术将用户访问情侣空间分为四类，未登录用户访问、单身用户访问、有偶用户访问自己空间、有偶用户访问他人空间
	get_lovers_info(lid):	获取情侣空间基本信息

	get_user_info(uid):	获取单身用户基本信息

4. 设计技术

4.1 使用 AOP 技术实现单身用户登录判断以及用户分类

位置：./users/view.py

说明：使用修饰模式对单身用户登录进行判定，若 session 中不包含 uid(登录或注册后生成对应 session)则重定向到登录，否则先获取一些基本页面信息，包括用户个人信息(名字、头像信息、性别等)、关注列表、邀请消息(其他单身用户的请求)，最后返回处理后的结果。

此外，python 中还可以使用@修饰符来实现修饰模式，这里未添加以方便 url 中的调试。

```
82
83 # AOP to reduce login test
84 def require_login(view):
85     def new_view(request, *args, **kwargs):
86         if not request.session.get('uid', None):
87             HttpResponseRedirect('/login/')
88         else:
89             data = {}
90             data['user'] = get_user_info(request.session['uid'])
91             data['concern'] = get_concern(request.session['uid'])
92             data['invitions'] = get_notify_from_request_list(request)
93             kwargs = dict(kwargs, **data)
94         return view(request, *args, **kwargs)
95     return new_view
```

位置：./loveSpace/public.py

说明：为了将用户分类到四种在情侣空间的操作页面，包括未登录用户访问情侣空间、单身用户访问情侣空间、有偶用户访问自己的情侣空间、有偶用户访问他人的情侣空间，使用修饰模式来先获取共同需要的数据，包括访问的情侣空间的情侣合照、情侣用户名字等等，再判定进行分类，分类之后可能还需要获取对应类别的一些数据，最后操作结束返回结果。

```python
def user_dispatch(process):
    def final_render_view(request, *args, **kwargs):
        print 'start ==========='
        print 'init: ', kwargs
        data = {}
        data['lovers'] = get_lovers_info(kwargs['lid'])
        kwargs = dict(kwargs, **data)
        data = dict(process(request, *args, **kwargs), **data)
        data['fans'] = get_fans(kwargs['lid'])
        template_prefix = kwargs['first'] + "_" + kwargs['second']
        print template_prefix
        print 'data to template: ', data
        if not request.session.get('uid', None):
        # without logging
            print "_am ==========="
            return render_to_response(template_prefix + "_am.html", data, context_instance = RequestContext(request))
        elif not request.session.get('lid', None):
        # user without lover
            print "_nlu ==========="
            return render_to_response(template_prefix + "_nlu.html", data, context_instance = RequestContext(request))
        elif request.session['lid'] == long(kwargs['lid']): # kwargs['lid'] is type of unicode
        # user's lover space
            data['inform'] = InfoManager.get_inform(request.session['uid'])
            data['concern'] = get_concern(request.session['uid'])
            data['inform'] = InfoManager.get_inform(request.session['uid'])
            data['concern'] = get_concern(request.session['uid'])
            print "_olu ==========="
            return render_to_response(template_prefix + "_olu.html", data, context_instance = RequestContext(request))
        else:
        # user to view other's lover space
            data['inform'] = InfoManager.get_inform(request.session['uid'])
            print "_ulu ==========="
            return render_to_response(template_prefix + "_ulu.html", data, context_instance = RequestContext(request))
    return final_render_view
```

4.2 使用 uploadify 异步传送图片至后端

位置：./static/js/message.js，./static/js/uploadPhoto.js

说明：为了使得异步发送包含图片的短消息，使用了 uploadify 来异步发送图片。

前端的实现为初始化指定一些基本参数，包括图片发送的 url、图片类型、大小等，以及一个成功返回的回调函数，来将一张经后端处理后产生的原图的小图返回到前端并加入到页面作为上传预览。

```javascript
var init_uploadify = function()
{
    $('#file_upload').uploadify({
        height: 30,
        width: 120,
        swf: '/static/uploadify/uploadify.swf',
        uploader: '/lovers/home/image_upload/',
        buttonText: '上传图片',
        sizeLimit: '20240', //kb, 20MB
        formData: {'lid': lid},
        fileTypeExts: '*.jpg;*.gif;*.png',
        onUploadSuccess: function(file, data, response) {
        // use user id to make it convienient to delete the temp image file
            $('#db').after("<img id='temp_img" + uid + "' src='/static/pic/" + lid + "/temp_s" + uid + ".jpg' />");
        }
    });
}
```

位置：./message/ajax.py

说明：后端的实现为将图片保存为一个临时文件的形式，而不是正式加入到对应的文件夹，因为一个状态发布包含了文字和图片，异步上传图片后未必保证一定点击状态发布，

因此这里将图片保存为临时文件，并且另存一个处理成小规格的图片用以返回给前端作为预览。

```python
@csrf_exempt
def uploadify_script(request, *args, **kwargs):
    result = '0'
    if request.method == 'POST' and request.FILES['Filedata'] and request.POST.get('lid', None):
        o_tempfile = '%s%s%s' % (settings.MEDIA_ROOT, request.POST['lid'], '/temp' + request.session['uid'] + '.jpg')
        s_tempfile = '%s%s%s' % (settings.MEDIA_ROOT, request.POST['lid'], '/temp_s' + request.session['uid'] + '.jpg')
        f = open(o_tempfile, 'wb+')
        for chunk in request.FILES['Filedata'].chunks():    # ensure the big file upload
            f.write(chunk)
#       f.write(request.FILES['Filedata'].read())
        f.close()
        o_img = Image.open(o_tempfile)
        width, height = o_img.size
        alpha = float(width) / float(height)
        s_img = o_img.resize((int(alpha * 60), 60))
        s_img.save(s_tempfile)
        result = '1'
    return HttpResponse(result)
```

4.3 使用正则表达式实现 qq 表情的消息推送

位置：./static/js/message.py

说明：为了使发布的状态包含表情，使用正则表达式来匹配特殊表情字符串，比如[/表情 3]就是一个流口水的表情，对应./static/pic/face/里的一个 gif 文件，先用正则表达式匹配这些特殊表情字符串，再根据对应关系来把这些特殊字符串替换为一个 img 标签的图片，从而实现了表情的展现。

```javascript
        send_message: function (e) {
            var content = $('#msg_content').val()
            var message_model = new Msg({content: content});

            message_model.save(null,
                //alter the accept type in header from json to html!!!
                { dataType: 'html',
                success: function(model, response) {
                    response = response.replace(/\[\/表情([0-9]*)\]/g,'<img src="/static/pic/face/$1.gif" border="0" />');
//                  $('#message').prepend(response).fadeIn();   //add message to show
//                  $(response).hide().prependTo('#message').fadeIn("slow");
                    $(response).prependTo('#message').hide().slideDown("slow");
                    $('#msg_content').val('');  //delete the content of input text
                    mid = $('#message').children().first().attr('id');  //!!pay to here
                    $('#temp_img' + uid).remove();
                },
                error: function(model, response) {
                    alert(response.status);
                }
            });
        },
```

4.4 使用外观模式用于粉丝评论消息管理

位置：./loveSpace/public.py

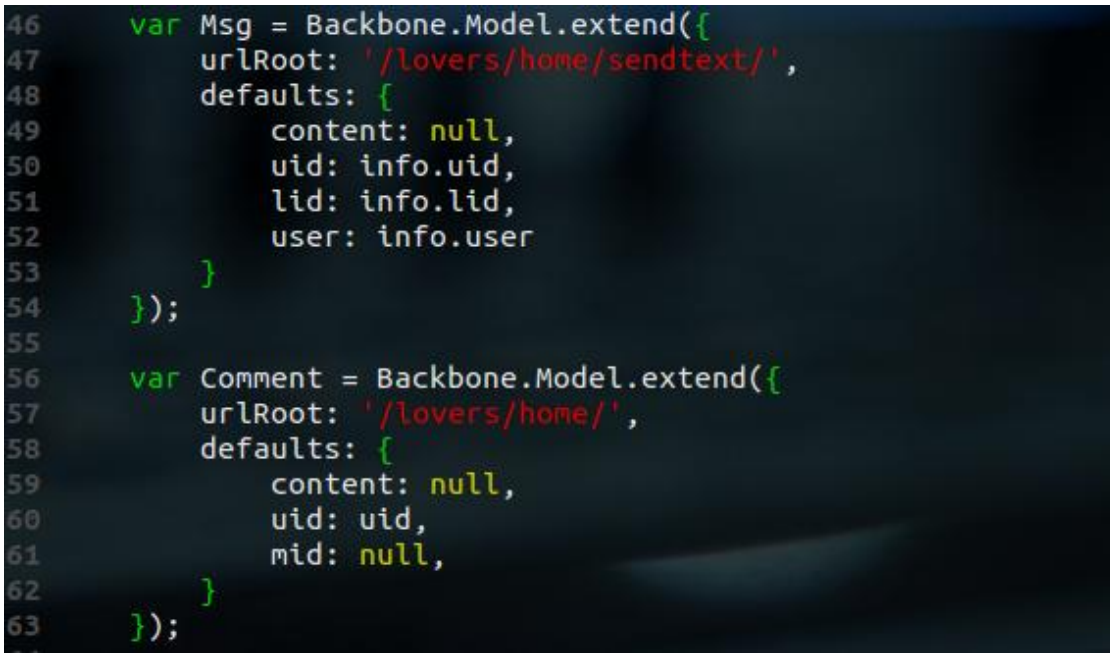
说明：为了降低耦合使用外观模式来管理粉丝评论消息，设计 InfoManager 类来实现，只需要知道该类便可以实现初始化用户粉丝消息和评论消息列表、添加与删除新消息或者新评论至消息列表、获取消息具体信息的功能。

```
119 class InfoManager:
120     @staticmethod
121     def init_inform(uid):
122         cursor = connection.cursor()
123         cursor.execute('insert into FansInform (uid) values (%s)' % (uid))
124         cursor.execute('insert into CommentInform (uid) values (%s)' % (uid))
125 
126     @staticmethod
127     def add_inform(tbl_name, lid):
128         tbl = tbl_name + 'Inform'
129         cursor = connection.cursor()
130         a = 'update %s, Lovers set %s.num=%s.num+1 where (%s.uid=Lovers.lover1_id or %s.uid=Lovers.lover2_id) and Lovers.id=%s' % (tbl, tbl, tbl
    , tbl, tbl, lid)
131         cursor.execute(a)
132         transaction.commit_unless_managed()
133 
134     @staticmethod
135     def get_inform(uid):
136         cursor = connection.cursor()
137         cursor.execute('select FansInform.num, CommentInform.num from FansInform, CommentInform where FansInform.uid=CommentInform.uid and FansI
    nform.uid=%s' % (uid))
138         row = cursor.fetchone()
139         row = {'fans': row[0], 'comment': row[1]}
140         return row
141 
142     @staticmethod
143     def delete_inform(tbl_name, uid):
144         cursor = connection.cursor()
145         cursor.execute('update %sInform set num=0 where uid=%s' % (tbl_name, uid))
146         transaction.commit_unless_managed()
```

4.5 使用 backbone.js 处理前端请求

位置：./static/js/message.js

说明：为了使前端更方便地管理异步请求，使用 backbone 来实现更好的模块划分，对于发布的状态以及评论，各自定义一个 Model 来记录 ajax 请求需要的数据以及 url。接着使用一个 View 来用于绑定界面操作的触发操作，并用之前的 Model 来快速地实现后端请求数据库增删查改的操作。



```
var Msg = Backbone.Model.extend({
    urlRoot: '/lovers/home/sendtext/',
    defaults: {
        content: null,
        uid: info.uid,
        lid: info.lid,
        user: info.user
    }
});

var Comment = Backbone.Model.extend({
    urlRoot: '/lovers/home/',
    defaults: {
        content: null,
        uid: uid,
        mid: null,
    }
});
```

```
var MessageView = Backbone.View.extend({
    el: $("body"),
    initialize: function () {
    },
    events: {
        "click #submit_message":  "send_message",
        "click .close":  "delete_msg_or_comment",
        "click .get_comment": "get_comment",
        "click .comment": "send_comment",
        "click .img_block": "image_click"
    },

    send_message: function (e) {
        var content = $('#msg_content').val()
        var message_model = new Msg({content: content});

        message_model.save(null,
            //alter the accept type in header from json to html!!!
            { dataType: 'html',
            success: function(model, response) {
                response = response.replace(/\[\/表情([0-9]*)\]/g,'<img src="/static/pic/face/$1.gif" border="0" />');
//              $('#message').prepend(response).fadeIn();   //add message to show
//              $(response).hide().prependTo('#message').fadeIn("slow");
                $(response).prependTo('#message').hide().slideDown("slow");
                $('#msg_content').val('');  //delete the content of input text
                mid = $('#message').children().first().attr('id');  //!!pay to here
                $('#temp_img' + uid).remove();
            },
            error: function(model, response) {
                alert(response.status);
            }
        });
    },
```

4.6 相册的图片处理

位置：./album/views.py

说明：为方便处理图片，对用户上传的图片做几点处理，1.重命名图片，’0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ’中随机产生 16 个字符组成　2.每张图片在服务器保存两份，一张为原图，一张截取中间 180*180 像素作为预览（小于 180*180 的将尺寸放大到 180*180）。

```
256             # cover
257             if photoWidth > photoHeight:
258                 tmp = int((photoWidth-photoHeight)*1.0/2)
259                 box = (tmp, 0, tmp+photoHeight, photoHeight)
260             else :
261                 tmp = int((photoHeight-photoWidth)*1.0/2)
262                 box = (0, tmp, photoWidth, tmp+photoWidth)
263             region = im.crop(box)
264             imS = region.resize((180, 180))
265             photoSPath = os.path.join(uploadFolder, 'c' + photoName + '.jpg')
266             imS.save(photoSPath, 'jpeg')
```

（具体设计在源代码中出现的位置，指明对应模块和代码）