

电子邮件

Email的历史比Web还要久远，直到现在，Email也是互联网上应用非常广泛的服务。

几乎所有的编程语言都支持发送和接收电子邮件，但是，先等等，在我们开始编写代码之前，有必要搞清楚电子邮件是如何在互联网上运作的。

我们来看看传统邮件是如何运作的。假设你现在在北京，要给一个香港的朋友发一封信，怎么做呢？

首先你得写好信，装进信封，写上地址，贴上邮票，然后就近找个邮局，把信仍进去。

信件会从就近的小邮局转运到大邮局，再从大邮局往别的城市发，比如先发到天津，再走海运到达香港，也可能走京九线到香港，但是你别担心具体路线，你只需要知道一件事，就是信件走得很慢，至少要几天时间。

信件到达香港的某个邮局，也不会直接送到朋友的家里，因为邮局的叔叔是很聪明的，他怕你的朋友不在家，一趟一趟地白跑，所以，信件会投递到你的朋友的邮箱里，邮箱可能在公寓的一层，或者家门口，直到你的朋友回家的时候检查邮箱，发现信件后，就可以取到邮件了。

电子邮件的流程基本上也是按上面的方式运作的，只不过速度不是按天算，而是按秒算。

现在我们回到电子邮件，假设我们自己的电子邮件地址是 `me@163.com`，对方的电子邮件地址是 `friend@sina.com`（注意地址都是虚构的哈），现在我们用 Outlook 或者 Foxmail 之类的软件写好邮件，填上对方的Email地址，点“发送”，电子邮件就发出去了。这些电子邮件软件被称为**MUA**：Mail User Agent——邮件用户代理。

Email从MUA发出去，不是直接到达对方电脑，而是发到**MTA**：Mail Transfer Agent——邮件传输代理，就是那些Email服务提供商，比如网易、新浪等等。由于我们自己的电子邮件是 `163.com`，所以，Email首先被投递到网易提供的MTA，再由网易的MTA发到对方服务商，也就是新浪的MTA。这个过程中间可能还会经过别的MTA，但是我们不关心具体路线，我们只关心速度。

Email到达新浪的MTA后，由于对方使用的是 `@sina.com` 的邮箱，因此，新浪的MTA会把Email投递到邮件的最终目的地**MDA**：Mail Delivery Agent——邮件投递代理。Email到达MDA后，就静静地躺在新浪的某个服务器上，存放在某个文件或特殊的数据库里，我们将这个长期保存邮件的地方称之为电子邮箱。

同普通邮件类似，Email不会直接到达对方的电脑，因为对方电脑不一定开机，开机也不一定联网。对方要取到邮件，必须通过MUA从MDA上把邮件取到自己的电脑上。

所以，一封电子邮件的旅程就是：

```
发件人 -> MUA -> MTA -> MTA -> 若干个MTA -> MDA <- MUA <- 收件人
```

有了上述基本概念，要编写程序来发送和接收邮件，本质上就是：

1. 编写MUA把邮件发到MTA;
2. 编写MUA从MDA上收邮件。

发邮件时，MUA和MTA使用的协议就是SMTP：Simple Mail Transfer Protocol，后面的MTA到另一个MTA也是用SMTP协议。

收邮件时，MUA和MDA使用的协议有两种：POP：Post Office Protocol，目前版本是3，俗称POP3；IMAP：Internet Message Access Protocol，目前版本是4，优点是不但能取邮件，还可以直接操作MDA上存储的邮件，比如从收件箱移到垃圾箱，等等。

邮件客户端软件在发邮件时，会让你先配置SMTP服务器，也就是你要发到哪个MTA上。假设你正在使用163的邮箱，你就不能直接发到新浪的MTA上，因为它只服务新浪的用户，所以，你得填163提供的SMTP服务器地址：`smtp.163.com`，为了证明你是163的用户，SMTP服务器还要求你填写邮箱地址和邮箱口令，这样，MUA才能正常地把Email通过SMTP协议发送到MTA。

类似的，从MDA收邮件时，MDA服务器也要求验证你的邮箱口令，确保不会有人冒充你收取你的邮件，所以，Outlook之类的邮件客户端会要求你填写POP3或IMAP服务器地址、邮箱地址和口令，这样，MUA才能顺利地通过POP或IMAP协议从MDA取到邮件。

在使用Python收发邮件前，请先准备好至少两个电子邮件，如 `xxx@163.com`，`xxx@sina.com`，`xxx@qq.com` 等，注意两个邮箱不要用同一家邮件服务商。

最后 **特别注意**，目前大多数邮件服务商都需要手动打开SMTP发信和POP收信的功能，否则只允许在网页登录：



SMTP发送邮件

SMTP是发送邮件的协议，Python内置对SMTP的支持，可以发送纯文本邮件、HTML邮件以及带附件的邮件。

Python对SMTP支持有 `smtplib` 和 `email` 两个模块，`email` 负责构造邮件，`smtplib` 负责发送邮件。

首先，我们来构造一个最简单的纯文本邮件：

```
from email.mime.text import MIMEText
msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
```

注意到构造 `MIMEText` 对象时，第一个参数就是邮件正文，第二个参数是MIME的subtype，传入 `'plain'` 表示纯文本，最终的MIME就是 `'text/plain'`，最后一定要用 `utf-8` 编码保证多语言兼容性。

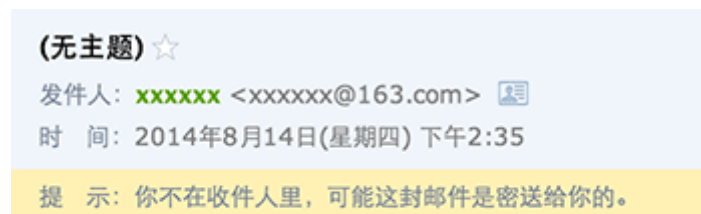
然后，通过SMTP发出去：

```
# 输入Email地址和口令：
from_addr = input('From: ')
password = input('Password: ')
# 输入收件人地址：
to_addr = input('To: ')
# 输入SMTP服务器地址：
smtp_server = input('SMTP server: ')

import smtplib
server = smtplib.SMTP(smtp_server, 25) # SMTP协议默认端口是25
server.set_debuglevel(1)
server.login(from_addr, password)
server.sendmail(from_addr, [to_addr], msg.as_string())
server.quit()
```

我们用 `set_debuglevel(1)` 就可以打印出和SMTP服务器交互的所有信息。SMTP协议就是简单的文本命令和响应。`login()` 方法用来登录SMTP服务器，`sendmail()` 方法就是发邮件，由于可以一次发给多个人，所以传入一个 `list`，邮件正文是一个 `str`，`as_string()` 把 `MIMEText` 对象变成 `str`。

如果一切顺利，就可以在收件人信箱中收到我们刚发送的Email：



hello, send by Python...

仔细观察，发现如下问题：

1. 邮件没有主题；
2. 收件人的名字没有显示为友好的名字，比如 `Mr Green`；
3. 明明收到了邮件，却提示不在收件人中。

这是因为邮件主题、如何显示发件人、收件人等信息并不是通过SMTP协议发给MTA，而是包含在发给MTA的文本中的，所以，我们必须把 `From`、`To` 和 `Subject` 添加到 `MIMEText` 中，才是一封完整的邮件：

```
from email import encoders
from email.header import Header
```

```

from email.mime.text import MIMEText
from email.utils import parseaddr, formataddr

import smtplib

def _format_addr(s):
    name, addr = parseaddr(s)
    return formataddr((Header(name, 'utf-8').encode(), addr))

from_addr = input('From: ')
password = input('Password: ')
to_addr = input('To: ')
smtp_server = input('SMTP server: ')

msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')
msg['From'] = _format_addr('Python爱好者 <%s>' % from_addr)
msg['To'] = _format_addr('管理员 <%s>' % to_addr)
msg['Subject'] = Header('来自SMTP的问候.....', 'utf-8').encode()

server = smtplib.SMTP(smtp_server, 25)
server.set_debuglevel(1)
server.login(from_addr, password)
server.sendmail(from_addr, [to_addr], msg.as_string())
server.quit()


```

我们编写了一个函数 `_format_addr()` 来格式化一个邮件地址。注意不能简单地传入 `name`，因为如果包含中文，需要通过 `Header` 对象进行编码。

`msg['To']` 接收的是字符串而不是list，如果有多个邮件地址，用 `,` 分隔即可。

再发送一遍邮件，就可以在收件人邮箱中看到正确的标题、发件人和收件人：

来自SMTP的问候..... ☆

发件人: Python爱好者 <xxxxxx@163.com> 

时 间: 2014年8月14日(星期四) 下午3:45

收件人: 管理员 <xxxxxx@qq.com>

hello, send by Python...

你看到的收件人的名字很可能不是我们传入的 `管理员`，因为很多邮件服务商在显示邮件时，会把收件人名字自动替换为用户注册的名字，但是其他收件人名字的显示不受影响。

如果我们查看Email的原始内容，可以看到如下经过编码的邮件头：

```

From: =?utf-8?b?UH10aG9u54ix5aW96ICF?= <xxxxxx@163.com>
To: =?utf-8?b?566h55CG5ZGY?= <xxxxxx@qq.com>
Subject: =?utf-8?b?5p2l6IeqU01UU0eah0mXruWAmekApuKApg==?=

```

这就是经过 `Header` 对象编码的文本，包含utf-8编码信息和Base64编码的文本。如果我们自己来手动构造这样的编码文本，显然比较复杂。


发送HTML邮件

如果我们要发送HTML邮件，而不是普通的纯文本文件怎么办？方法很简单，在构造 `MIMEText` 对象时，把HTML字符串传进去，再把第二个参数由 `plain` 变为 `html` 就可以了：

```
msg = MIMEText('<html><body><h1>Hello</h1>' +
               '<p>send by <a href="http://www.python.org">Python</a>...</p>' +
               '</body></html>', 'html', 'utf-8')
```

再发送一遍邮件，你将看到以HTML显示的邮件：

来自SMTP的问候..... ☆

发件人: Python爱好者 <xxxxxx@163.com> 

时 间: 2014年8月14日(星期四) 下午4:06

收件人: 管理员 <xxxxxx@qq.com>

Hello

send by [Python...](#)

发送附件

如果Email中要加上附件怎么办？带附件的邮件可以看做包含若干部分的邮件：文本和各个附件本身，所以，可以构造一个 `MIMEMultipart` 对象代表邮件本身，然后往里面加上一个 `MIMEText` 作为邮件正文，再继续往里面加上表示附件的 `MIMEBase` 对象即可：

```
# 邮件对象：
msg = MIMEMultipart()
msg['From'] = _format_addr('Python爱好者 <%s>' % from_addr)
msg['To'] = _format_addr('管理员 <%s>' % to_addr)
msg['Subject'] = Header('来自SMTP的问候.....', 'utf-8').encode()

# 邮件正文是MIMEText：
msg.attach(MIMEText('send with file...', 'plain', 'utf-8'))

# 添加附件就是加上一个MIMEBase，从本地读取一个图片：
with open('/Users/michael/Downloads/test.png', 'rb') as f:
    # 设置附件的MIME和文件名，这里是png类型：
    mime = MIMEBase('image', 'png', filename='test.png')
    # 加上必要的头信息：
    mime.add_header('Content-Disposition', 'attachment', filename='test.png')
    mime.add_header('Content-ID', '<0>')
    mime.add_header('X-Attachment-Id', '0')
```


```
# 把附件的内容读进来:
mime.set_payload(f.read())

# 用Base64编码:
encoders.encode_base64(mime)

# 添加到MIMEMultipart:
msg.attach(mime)
```


然后, 按正常发送流程把 `msg` (注意类型已变为 `MIMEMultipart`) 发送出去, 就可以收到如下带附件的邮件:

来自SMTP的问候..... ☆


发件人: Python爱好者 <xxxxxx@163.com> 

时 间: 2014年8月14日(星期四) 下午5:08

收件人: 管理员 <xxxxxx@qq.com>

附 件: 1 个 ( test.png)

send with file...

 附件(1 个)

普通附件



test.png (80.13K)

[下载](#) [预览](#) [收藏](#) [转存](#) ▼

发送图片

如果要把一个图片嵌入到邮件正文中怎么做? 直接在HTML邮件中链接图片地址行不行? 答案是, 大部分邮件服务商都会自动屏蔽带有外链的图片, 因为不知道这些链接是否指向恶意网站。

要把图片嵌入到邮件正文中, 我们只需按照发送附件的方式, 先把邮件作为附件添加进去, 然后, 在HTML中通过引用 `src="cid:0"` 就可以把附件作为图片嵌入了。如果有多个图片, 给它们依次编号, 然后引用不同的 `cid:x` 即可。

把上面代码加入 `MIMEMultipart` 的 `MIMEText` 从 `plain` 改为 `html`, 然后在适当的位置引用图片:

```
msg.attach(MIMEText('<html><body><h1>Hello</h1>' +
    '<p></p>' +
    '</body></html>', 'html', 'utf-8'))
```

再次发送, 就可以看到图片直接嵌入到邮件正文的效果:

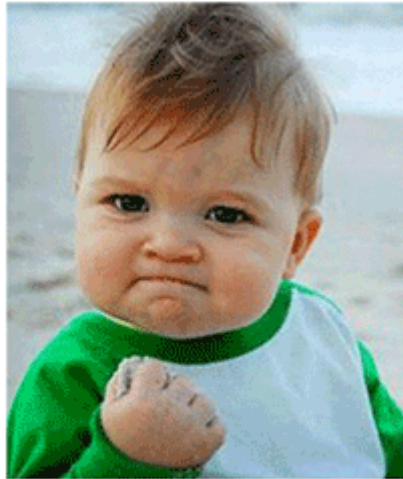
来自SMTP的问候..... ☆

发件人: Python爱好者 <asklxf@163.com> 

时 间: 2014年8月14日(星期四) 下午5:27

收件人: Xuefeng <18224514@qq.com>

Hello



同时支持HTML和Plain格式

如果我们发送HTML邮件，收件人通过浏览器或者Outlook之类的软件是可以正常浏览邮件内容的，但是，如果收件人使用的设备太古老，查看不了HTML邮件怎么办？

办法是在发送HTML的同时再附加一个纯文本，如果收件人无法查看HTML格式的邮件，就可以自动降级查看纯文本邮件。

利用 `MIMEMultipart` 就可以组合一个HTML和Plain，要注意指定subtype是 `alternative`：

```
msg = MIMEMultipart('alternative')
msg['From'] = ...
msg['To'] = ...
msg['Subject'] = ...

msg.attach(MIMEText('hello', 'plain', 'utf-8'))
msg.attach(MIMEText('<html><body><h1>Hello</h1></body></html>', 'html', 'utf-8'))
# 正常发送msg对象...
```

加密SMTP

使用标准的25端口连接SMTP服务器时，使用的是明文传输，发送邮件的整个过程可能会被窃听。要更安全地发送邮件，可以加密SMTP会话，实际上就是先创建SSL安全连接，然后再使用SMTP协议发送邮件。

某些邮件服务商，例如Gmail，提供的SMTP服务必须要加密传输。我们来看看如何通过Gmail提供的安全SMTP发送邮件。

必须知道，Gmail的SMTP端口是587，因此，修改代码如下：

```
smtp_server = 'smtp.gmail.com'
smtp_port = 587
server = smtplib.SMTP(smtp_server, smtp_port)
server.starttls()
# 剩下的代码和前面的一模一样：
server.set_debuglevel(1)
...
```

只需要在创建 SMTP 对象后，立刻调用 `starttls()` 方法，就创建了安全连接。后面的代码和前面的发送邮件代码完全一样。

如果因为网络问题无法连接Gmail的SMTP服务器，请相信我们的代码是没有问题的，你需要对你的网络设置做必要的调整。

小结

使用Python的smtplib发送邮件十分简单，只要掌握了各种邮件类型的构造方法，正确设置好邮件头，就可以顺利发出。

构造一个邮件对象就是一个 `Message` 对象，如果构造一个 `MIMEText` 对象，就表示一个文本邮件对象，如果构造一个 `MIMEImage` 对象，就表示一个作为附件的图片，要把多个对象组合起来，就用 `MIMEMultipart` 对象，而 `MIMEBase` 可以表示任何对象。它们的继承关系如下：

```
Message
+- MIMEBase
  +- MIMEMultipart
  +- MIMENonMultipart
    +- MIMEMessage
    +- MIMEText
    +- MIMEImage
```

这种嵌套关系就可以构造出任意复杂的邮件。你可以通过[email.mime文档](#)查看它们所在的包以及详细的用法。

POP3 收取邮件

SMTP用于发送邮件，如果要收取邮件呢？

收取邮件就是编写一个MUA作为客户端，从MDA把邮件获取到用户的电脑或者手机上。收取邮件最常用的协议是POP协议，目前版本号是3，俗称POP3。

Python内置一个 `poplib` 模块，实现了POP3协议，可以直接用来收邮件。

注意到POP3协议收取的不是一个已经可以阅读的邮件本身，而是邮件的原始文本，这和SMTP协议很像，SMTP发送的也是经过编码后的一大段文本。

要把POP3收取的文本变成可以阅读的邮件，还需要用 `email` 模块提供的各种类来解析原始文本，变成可阅读的邮件对象。

所以，收取邮件分两步：

第一步：用 `poplib` 把邮件的原始文本下载到本地；

第二部：用 `email` 解析原始文本，还原为邮件对象。

通过POP3下载邮件

POP3协议本身很简单，以下面的代码为例，我们来获取最新的一封邮件内容：

```
import poplib

# 输入邮件地址，口令和POP3服务器地址：
email = input('Email: ')
password = input('Password: ')
pop3_server = input('POP3 server: ')

# 连接到POP3服务器：
server = poplib.POP3(pop3_server)
# 可以打开或关闭调试信息：
server.set_debuglevel(1)
# 可选：打印POP3服务器的欢迎文字：
print(server.getwelcome().decode('utf-8'))

# 身份认证：
server.user(email)
server.pass_(password)

# stat()返回邮件数量和占用空间：
print('Messages: %s. Size: %s' % server.stat())
# list()返回所有邮件的编号：
resp, mails, octets = server.list()
# 可以查看返回的列表类似[b'1 82923', b'2 2184', ...]
print(mails)

# 获取最新一封邮件，注意索引号从1开始：
index = len(mails)
resp, lines, octets = server.retr(index)

# lines存储了邮件的原始文本的每一行，
# 可以获得整个邮件的原始文本：
msg_content = b'\r\n'.join(lines).decode('utf-8')
# 稍后解析出邮件：
msg = Parser().parsestr(msg_content)

# 可以根据邮件索引号直接从服务器删除邮件：
```

```
# server.dele(index)
# 关闭连接:
server.quit()
```

用POP3获取邮件其实很简单，要获取所有邮件，只需要循环使用 `retr()` 把每一封邮件内容拿到即可。真正麻烦的是把邮件的原始内容解析为可以阅读的邮件对象。

解析邮件

解析邮件的过程和上一节构造邮件正好相反，因此，先导入必要的模块：

```
from email.parser import Parser
from email.header import decode_header
from email.utils import parseaddr

import poplib
```

只需要一行代码就可以把邮件内容解析为 `Message` 对象：

```
msg = Parser().parsestr(msg_content)
```

但是这个 `Message` 对象本身可能是一个 `MIMEMultipart` 对象，即包含嵌套的其他 `MIMEBase` 对象，嵌套可能还不止一层。

所以我们要递归地打印出 `Message` 对象的层次结构：

```
# indent用于缩进显示:
def print_info(msg, indent=0):
    if indent == 0:
        for header in ['From', 'To', 'Subject']:
            value = msg.get(header, '')
            if value:
                if header=='Subject':
                    value = decode_str(value)
                else:
                    hdr, addr = parseaddr(value)
                    name = decode_str(hdr)
                    value = u'%s <%s>' % (name, addr)
                print('%s%s: %s' % (' ' * indent, header, value))
    if (msg.is_multipart()):
        parts = msg.get_payload()
        for n, part in enumerate(parts):
            print('%spart %s' % (' ' * indent, n))
            print('%s-----' % (' ' * indent))
            print_info(part, indent + 1)
    else:
        content_type = msg.get_content_type()
```

```

        if content_type=='text/plain' or content_type=='text/html':
            content = msg.get_payload(decode=True)
            charset = guess_charset(msg)
            if charset:
                content = content.decode(charset)
            print('%sText: %s' % (' ' * indent, content + '...'))
        else:
            print('%sAttachment: %s' % (' ' * indent, content_type))

```

邮件的Subject或者Email中包含的名字都是经过编码后的str，要正常显示，就必须decode：

```

def decode_str(s):
    value, charset = decode_header(s)[0]
    if charset:
        value = value.decode(charset)
    return value

```

decode_header() 返回一个list，因为像 Cc 、 Bcc 这样的字段可能包含多个邮件地址，所以解析出来的会有多个元素。上面的代码我们偷了个懒，只取了第一个元素。

文本邮件的内容也是str，还需要检测编码，否则，非UTF-8编码的邮件都无法正常显示：

```

def guess_charset(msg):
    charset = msg.get_charset()
    if charset is None:
        content_type = msg.get('Content-Type', '').lower()
        pos = content_type.find('charset=')
        if pos >= 0:
            charset = content_type[pos + 8:].strip()
    return charset

```

把上面的代码整理好，我们就可以来试试收取一封邮件。先往自己的邮箱发一封邮件，然后用浏览器登录邮箱，看看邮件收到没，如果收到了，我们就来用Python程序把它收到本地：



运行程序，结果如下：

```
+OK Welcome to coremail Mail Pop3 Server (163coms[...])
```

```
Messages: 126. Size: 27228317
```

```
From: Test <xxxxxx@qq.com>
```

```
To: Python爱好者 <xxxxxx@163.com>
```

```
Subject: 用POP3收取邮件
```

```
part 0
```

```
-----
```

```
part 0
```

```
-----
```

```
Text: Python可以使用POP3收取邮件.....
```

```
part 1
```

```
-----
```

```
Text: Python可以<a href="...">使用POP3</a>收取邮件.....
```

```
part 1
```

```
-----
```

```
Attachment: application/octet-stream
```

我们从打印的结构可以看出，这封邮件是一个 `MIMEMultipart`，它包含两部分：第一部分又是一个 `MIMEMultipart`，第二部分是一个附件。而内嵌的 `MIMEMultipart` 是一个 `alternative` 类型，它包含一个纯文本格式的 `MIMEText` 和一个HTML格式的 `MIMEText`。

小结

- 用Python的 `poplib` 模块收取邮件分两步：第一步是用POP3协议把邮件获取到本地，第二步是用 `email` 模块把原始邮件解析为 `Message` 对象，然后，用适当的形式把邮件内容展示给用户即可。