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 = raw_input('Email: ')
password = raw_input('Password: ')
pop3_server = raw_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())
# 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['1 82923', '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 = '\r\n'.join(lines)
# 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：

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

```
from email.utils import parseaddr
```

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:
        # 邮件的From, To, Subject存在于根对象上:
        for header in ['From', 'To', 'Subject']:
            value = msg.get(header, '')
            if value:
                if header=='Subject':
                    # 需要解码Subject字符串:
                    value = decode_str(value)
                else:
                    # 需要解码Email地址:
                    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()):
        # 如果邮件对象是一个MIMEMultipart,
        # get_payload()返回list，包含所有的子对象:
        parts = msg.get_payload()
        for n, part in enumerate(parts):
            print('%spart %s' % ('  ' * indent, n))
            print('%s--------------------' % ('  ' * indent))
            # 递归打印每一个子对象:
            print_info(part, indent + 1)
    else:
        # 邮件对象不是一个MIMEMultipart,
        # 就根据content_type判断:
        content_type = msg.get_content_type()
        if content_type=='text/plain' or content_type=='text/html':
            # 纯文本或HTML内容:
            content = msg.get_payload(decode=True)
            # 要检测文本编码:
            charset = guess_charset(msg)
            if charset:
                content = content.decode(charset)
            print('%sText: %s' % ('  ' * indent, content + '...'))
        else:
            # 不是文本,作为附件处理:
            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):
    # 先从msg对象获取编码:
    charset = msg.get_charset()
    if charset is None:
        # 如果获取不到，再从Content-Type字段获取:
```

```
        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：

Python可以使用POP3收取邮件.....

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。

源码参考：

https://github.com/michaelliao/learn-python/tree/master/email