Vamei

编程，数学，设计

Python网络01 原始Python服务器

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

之前我的Python教程中有人留言，表示只学Python没有用，必须学会一个框架(比如Django和web.py)才能找到工作。而我的想法是，掌握一个类似于框架的高级工具是有用的，但是基础的东西可以让你永远不被淘汰。不要被工具限制了自己的发展。今天，我在这里想要展示的，就是不使用框架，甚至不使用Python标准库中的高级包，只使用标准库中的socket接口(我不是很明白套接字这个翻译，所以使用socket的英文名字)，写一个Python服务器。

在当今Python服务器框架 (framework, 比如Django, Twisted, web.py等等) 横行的时代，从底层的socket开始写服务器似乎是一个出力不讨好的笨方法。框架的意义在于掩盖底层的细节，提供一套对于开发人员更加友好的API，并处理诸如MVC的布局问题。框架允许我们快速的构建一个成型而且成熟的Python服务器。然而，框架本身也是依赖于底层(比如socket)。对于底层socket的了解，不仅可以帮助我们更好的使用框架，更可以让我们明白框架是如何设计的。更进一步，如果拥有良好的底层socket编程知识和其他系统编程知识，你完全可以设计并开发一款自己的框架。如果你可以从底层socket开始，实现一个完整的Python服务器，支持用户层的协议，并处理好诸如MVC(Model-View-Control)、多线程(threading)等问题，并整理出一套清晰的函数或者类，作为接口(API)呈现给用户，你就相当于设计了一个框架。

socket接口是实际上是操作系统提供的系统调用。socket的使用并不局限于Python语言，你可以用C或者JAVA来写出同样的socket服务器，而所有语言使用socket的方式都类似(Apache就是使用C实现的服务器)。而你不能跨语言的使用框架。框架的好处在于帮你处理了一些细节，从而实现快速开发，但同时受到Python本身性能的限

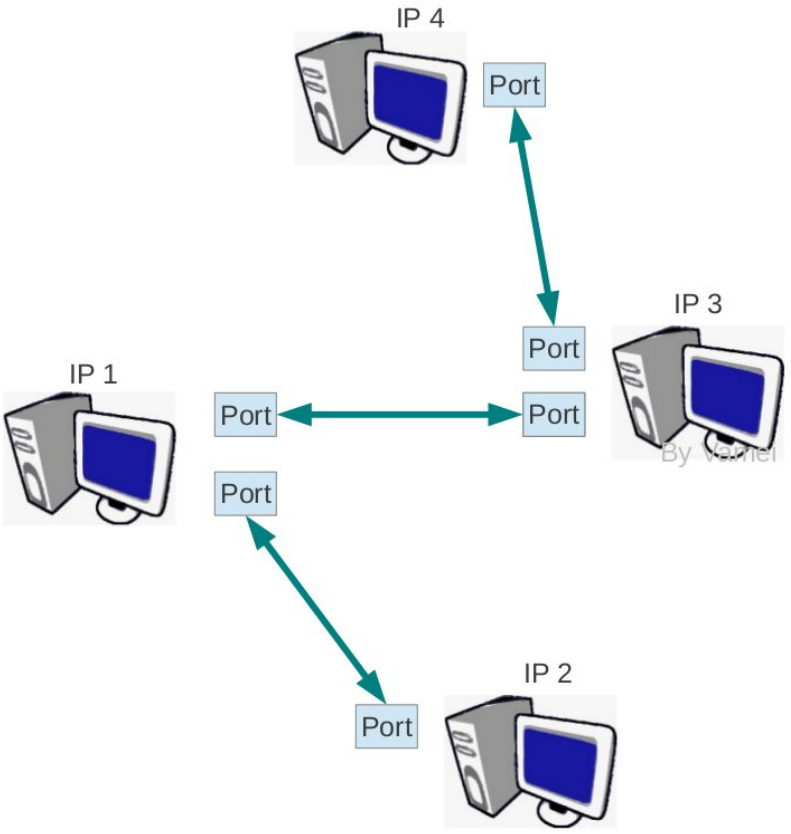
制。我们已经看到，许多成功的网站都是利用动态语言(比如Python，Ruby或者PHP，比如twitter和facebook)快速开发，在网站成功之后，将代码转换成诸如C和JAVA这样一些效率比较高的语言，从而让服务器能更有效率的面对每天亿万次的请求。在这样一些时间，底层的重要性，就远远超过了框架。

下面的一篇文章虽然是在谈JAVA，但我觉得也适用于Python的框架之争。

http://yakovfain.com/2012/10/11/the-degradation-of-java-developers/

TCP/IP和socket

我们需要对网络传输，特别是TCP/IP协议和socket有一定的了解。socket是进程间通信的一种方法 (参考Linux进程间通信)，它是基于网络传输协议的上层接口。socket有许多种类型，比如基于TCP协议或者UDP协议(两种网络传输协议)。其中又以TCP socket最为常用。TCP socket与双向管道(duplex PIPE)有些类似，一个进程向socket的一端写入或读取文本流，而另一个进程可以从socket的另一端读取或写入，比较特别是，这两个建立socket通信的进程可以分别属于两台不同的计算机。所谓的TCP协议，就是规定了一些通信的守则，以便在网络环境下能够有效实现上述进程间通信过程。双向管道(duplex PIPE)存活于同一台电脑中，所以不必区分两个进程的所在计算机的地址，而socket必须包含有地址信息，以便实现网络通信。一个socket包含四个地址信息： 两台计算机的IP地址和两个进程所使用的端口(port)。IP地址用于定位计算机，而port用于定位进程 (一台计算机上可以有多个进程分别使用不同的端口)。



一个TCP socket连接的网络

TCP socket

在互联网上，我们可以让某台计算机作为服务器。服务器开放自己的端口，被动等待其他计算机连接。当其他计算机作为客户，主动使用socket连接到服务器的时候，服务器就开始为客户提供服务。

在Python中，我们使用标准库中的socket包来进行底层的socket编程。

首先是服务器端，我们使用bind()方法来赋予socket以固定的地址和端口，并使用listen()方法来被动的监听该端口。当有客户尝试用connect()方法连接的时候，服务器使用accept()接受连接，从而建立一个连接的socket：

```
# Written by Vamei
# Server side
import socket
```

```
# Address
HOST = ''
PORT = 8000

reply = 'Yes'

# Configure socket
s       = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))

# passively wait, 3: maximum number of connections in the
queue
s.listen(3)
# accept and establish connection
conn, addr = s.accept()
# receive message
request    = conn.recv(1024)

print 'request is: ',request
print 'Connected by', addr
# send message
conn.sendall(reply)
# close connection
conn.close()
```

socket.socket()创建一个socket对象，并说明socket使用的是IPv4(AF_INET，IP version 4)和TCP协议(SOCK_STREAM)。

然后用另一台电脑作为客户，我们主动使用connect()方法来搜索服务器端的IP地址(在Linux中，你可以用$ifconfig来查询自己的IP地址)和端口，以便客户可以找到服务器，并建立连接：

```
# Written by Vamei
# Client side
import socket
```

```
# Address
HOST = '172.20.202.155'
PORT = 8000

request = 'can you hear me?'

# configure socket
s        = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

# send message
s.sendall(request)
# receive message
reply    = s.recv(1024)
print 'reply is: ',reply
# close connection
s.close()
```

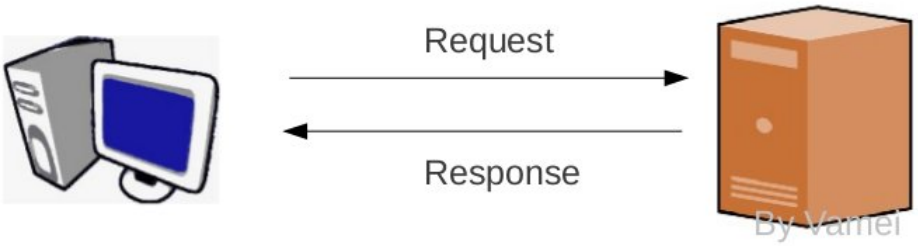
在上面的例子中，我们对socket的两端都可以调用recv()方法来接收信息，调用sendall()方法来发送信息。这样，我们就可以在分处于两台计算机的两个进程间进行通信了。当通信结束的时候，我们使用close()方法来关闭socket连接。

(如果没有两台计算机做实验，也可以将客户端IP想要connect的IP改为"127.0.0.1"，这是个特殊的IP地址，用来连接当地主机。)

基于TCP socket的HTTP服务器

上面的例子中，我们已经可以使用TCP socket来为两台远程计算机建立连接。然而，socket传输自由度太高，从而带来很多安全和兼容的问题。我们往往利用一些应用层的协议(比如HTTP协议)来规定socket使用规则，以及所传输信息的格式。

HTTP协议利用请求-回应(request-response)的方式来使用TCP socket。客户端向服务器发一段文本作为request，服务器端在接收到request之后，向客户端发送一段文本作为response。在完成了这样一次request-response交易之后，TCP socket被废弃。下次的request将建立新的socket。request和response本质上说是两个文本，只是HTTP协议对这两个文本都有一定的格式要求。



request-response cycle

现在，我们写出一个HTTP服务器端：

```
# Written by Vamei

import socket

# Address
HOST = ''
PORT = 8000

# Prepare HTTP response
text_content = '''HTTP/1.x 200 OK
Content-Type: text/html

<head>
<title>WOW</title>
</head>
<html>
<p>Wow, Python Server</p>
<IMG src="test.jpg"/>
</html>
'''

# Read picture, put into HTTP format
f = open('test.jpg','rb')
pic_content = '''
HTTP/1.x 200 OK
```

```
Content-Type: image/jpg

'''
pic_content = pic_content + f.read()
f.close()

# Configure socket
s    = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))

# infinite loop, server forever
while True:
    # 3: maximum number of requests waiting
   s.listen(3)
    conn, addr = s.accept()
    request     = conn.recv(1024)
    method     = request.split(' ')[0]
    src              = request.split(' ')[1]

    # deal with GET method
   if method == 'GET':
        # ULR
      if src == '/test.jpg':
            content = pic_content
        else: content = text_content

        print 'Connected by', addr
        print 'Request is:', request
        conn.sendall(content)
    # close connection
   conn.close()
```

深入HTTP服务器程序

如我们上面所看到的，服务器会根据request向客户传输的两条信息text_content和pic_content中的一条，作为response文本。整个response分为起始行(start line)，头信息(head)和主体(body)三部分。起始行就是第一行：

```
HTTP/1.x 200 OK
```

它实际上又由空格分为三个片段，HTTP/1.x表示所使用的HTTP版本，200表示状态(status code)，200是HTTP协议规定的，表示服务器正常接收并处理请求，OK是供人来阅读的status code。

头信息跟随起始行，它和主体之间有一个空行。这里的text_content或者pic_content都只有一行的头信息，text_content用来表示主体信息的类型为html文本：

```
Content-Type: text/html
```

而pic_content的头信息(Content-Type: image/jpg)说明主体的类型为jpg图片(image/jpg)。

主体信息为html或者jpg文件的内容。

(注意，对于jpg文件，我们使用'rb'模式打开，是为了与windows兼容。因为在windows下，jpg被认为是二进制(binary)文件，在UNIX系统下，则不需要区分文本文件和二进制文件。)

我们并没有写客户端程序，后面我们会用浏览器作为客户端。request由客户端程序发给服务器。尽管request也可以像response那样分为三部分，request的格式与response的格式并不相同。request由客户发送给服务器，比如下面是一个request：

```
GET /test.jpg HTTP/1.x
Accept: text/*
```

起始行可以分为三部分，第一部分为请求方法(request method)，第二部分是URL，第三部分为HTTP版本。request method可以有GET， PUT， POST，DELETE， HEAD。最常用的为GET和POST。GET是请求服务器发送资源给客户，POST是请求服务器接收客户送来的数据。当我们打开一个网页时，我们通常是使用GET方法；当我们填写表格并提交时，我们通常使用POST方法。第二部分为URL，它通常指向一个资源(服务器上的资源或者其它地方的资源)。像现在这样，就是指向当前服务器

的当前目录的test.jpg。

按照HTTP协议的规定，服务器需要根据请求执行一定的操作。正如我们在服务器程序中看到的，我们的Python程序先检查了request的方法，随后根据URL的不同，来生成不同的response(text_content或者pic_content)。随后，这个response被发送回给客户端。

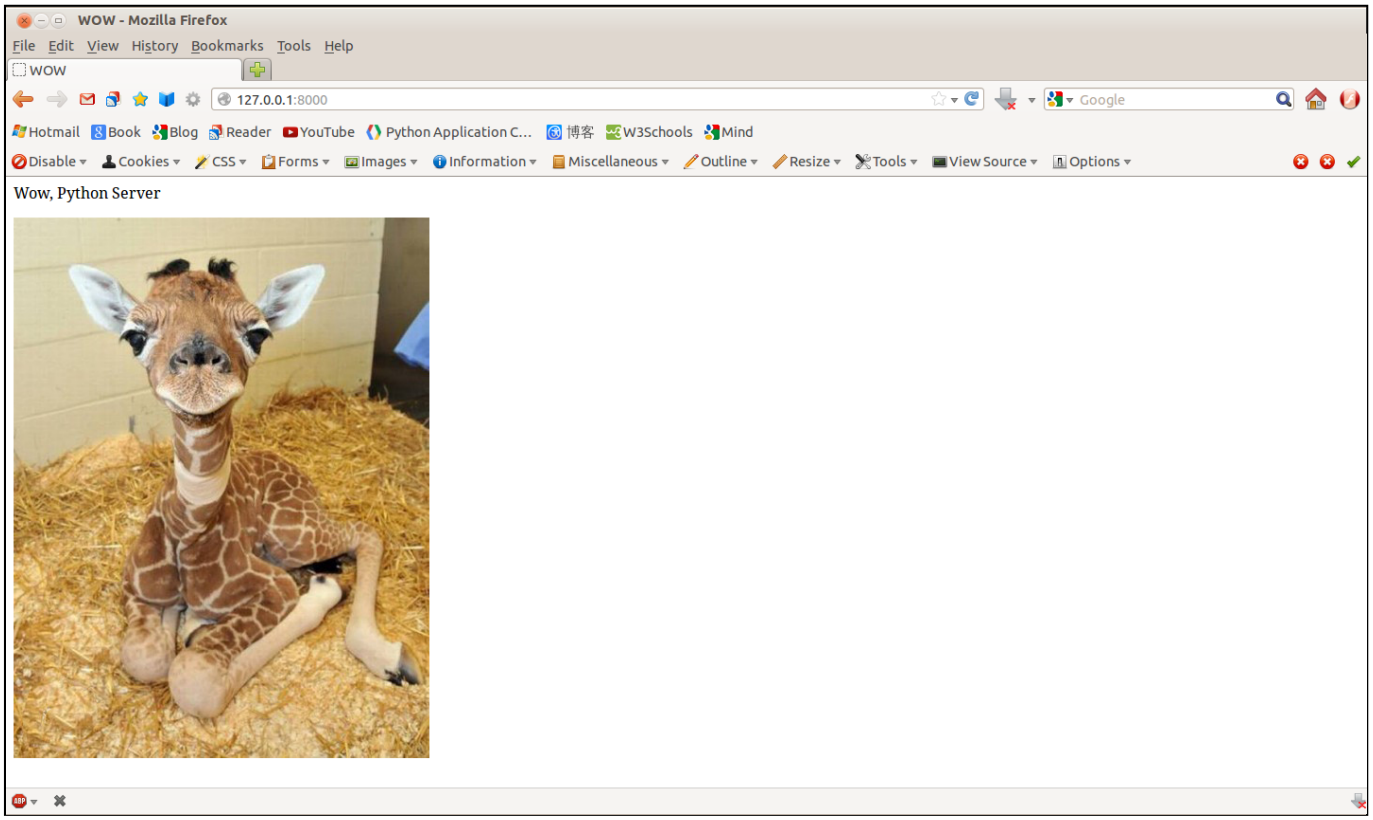
使用浏览器实验

为了配合上面的服务器程序，我已经在放置Python程序的文件夹里，保存了一个test.jpg图片文件。我们在终端运行上面的Python程序，作为服务器端，再打开一个浏览器作为客户端。(如果有时间，你也完全可以用Python写一个客户端。原理与上面的TCP socket的客户端程序相类似。)

在浏览器的地址栏输入:

```
127.0.0.1:8000
```

(当然，你也可以用令一台电脑，并输入服务器的IP地址。) 我得到下面的结果:



OK，我已经有了一个用Python实现的，并从socket写起的服务器了。

从终端，我们可以看到，浏览器实际上发出了两个请求。第一个请求为（关键信息在起始行，这一个请求的主体为空）：

```
GET / HTTP/1.1
Host: 127.0.0.1:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

我们的Python程序根据这个请求，发送给服务器text_content的内容。

浏览器接收到text_content之后，发现正文的html文本中有<IMG src="text.jpg" />，知道需要获得text.jpg文件来补充为图片，立即发出了第二个请求：

```
GET /test.jpg HTTP/1.1
Host: 127.0.0.1:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://127.0.0.1:8000/
```

我们的Python程序分析过起始行之后，发现/test.jpg符合if条件，所以将pic_content发送给客户。

最后，浏览器根据html语言的语法，将html文本和图画以适当的方式显示出来。

(html可参考http://www.w3schools.com/html/default.asp)

探索的方向

1） 在我们上面的服务器程序中，我们用while循环来让服务器一直工作下去。实际上，我们还可以根据我之前介绍的多线程的知识，将while循环中的内容改为多进程或者多线程工作。(参考Python多线程与同步，Python多进程初步，Python多进程探索)

2） 我们的服务器程序还不完善，我们还可以让我们的Python程序调用Python的其他功能，以实现更复杂的功能。比如说制作一个时间服务器，让服务器向客户返回日期和时间。你还可以使用Python自带的数据库，来实现一个完整的LAMP服务器。

3） socket包是比较底层的包。Python标准库中还有高层的包，比如SocketServer，SimpleHTTPServer，CGIHTTPServer，cgi。这些都包都是在帮助我们更容易的使用socket。如果你已经了解了socket，那么这些包就很容易明白了。利用这些高层的包，你可以写一个相当成熟的服务器。

4） 在经历了所有的辛苦和麻烦之后，你可能发现，框架是那么的方便，所以决定去使用框架。或者，你已经有了参与到框架开发的热情。

更多内容

TCP/IP和port参考: TCP/IP illustrated
http://book.douban.com/subject/1741925/

socket参考: UNIX Network Programming
http://book.douban.com/subject/1756533/

Python socket 官方文档
http://docs.python.org/2/library/socket.html

HTTP参考: HTTP, the definitive guide
http://book.douban.com/subject/1440226/

分类: Python

标签: Python

好文要顶 关注我 收藏该文

Vamei
关注 - 26
粉丝 - 4985
荣誉：推荐博客
+加关注

15 0

(请您对文章做出评价)

« 上一篇：Python标准库12 数学与随机数 (math包，random包)
» 下一篇：Python网络02 Python服务器进化

posted @ 2012-10-30 15:38 Vamei 阅读(22205) 评论(46) 编辑 收藏

评论列表

#1楼 2012-10-30 16:26 竹风抚荷塘

框架什么的我现在还是听得云里雾里的...

谢谢楼主~~

支持(0) 反对(0)

#2楼 2012-10-30 18:19 zhuangzhuang1988

还有个wsgi...

支持(0) 反对(0)

#3楼[楼主] 2012-10-30 19:22 Vamei

@ zhuangzhuang1988
没错，还有wsgiref。我没有写全。

支持(0) 反对(0)

#4楼 2012-10-30 20:51 玉菜园

收藏

支持(0) 反对(0)

#5楼 2012-11-16 23:34 msheng.yeb

http服务器那个挺好玩的。博主，我发现代码里面那个test.jpg文件句柄没有关闭

支持(0) 反对(0)

#6楼[楼主] 2012-11-17 09:50 Vamei

@ msheng.yeb

是哦，应该关闭的。

支持(0) 反对(0)

#7楼[楼主] 2012-11-17 13:34 Vamei

@ msheng.yeb

原来file descriptor是翻译成为句柄啊。

支持(0) 反对(0)

#8楼 2012-11-17 23:38 msheng.yeb

@ Vamei

如果要说file descriptor，那应该译为"文件描述符"合适一点吧，叫句柄也行

支持(0) 反对(0)

#9楼[楼主] 2012-12-12 15:06 Vamei

@ msheng.yeb

我一般称它为文件描述符

支持(0) 反对(0)

#10楼 2012-12-12 18:21 msheng.yeb

@ Vamei

最近看书，说在最后的文件描述符，不关闭也行。因为进程退出的时候，该进程的所有文件描述符会自动被关闭

支持(0) 反对(0)

#11楼[楼主] 2012-12-12 19:17 Vamei

@ msheng.yeb

是这样的。

支持(0) 反对(1)

#12楼 2013-03-26 21:28 求知的年轻人

@Vamei

博主，Client 中为什么不能传递汉字呢

支持(0) 反对(0)

#13楼 2013-03-26 22:06 求知的年轻人

@ Vamei

Client 中为什么不能传递汉字呢？

支持(0) 反对(0)

#14楼[楼主] 2013-03-26 22:16 Vamei

@ 求知的年轻人

不好意思，我没有尝试过。

支持(0) 反对(0)

#15楼[楼主] 2013-03-26 22:20 Vamei

@ 求知的年轻人

Python的中文支持需要一些摸索，可惜我在这方面下的功夫不够，所以可能无法回答你的问题。希望有人能来回答一下。

支持(0) 反对(0)

#16楼 2013-05-09 20:14 ___kevin

楼主，那个图片显示不出来是和网页的编码有关么？

支持(0) 反对(0)

#17楼[楼主] 2013-05-09 20:50 Vamei

@ ___kevin

html可以显示么？

支持(0) 反对(0)

#18楼 2013-05-23 11:16 dacoolbaby

非常好的文章，简明扼要地解释了服务器的根本原理。

支持(0) 反对(0)

#19楼 2013-06-19 16:18 Eve.月

楼主你出书吧...

支持(0) 反对(0)

#20楼[楼主] 2013-06-20 00:28 Vamei

@ Eve.月

我是有计划出书。我最近去北京了解一下。

支持(0) 反对(0)

#21楼 2013-06-20 00:30 Eve.月

支持楼主出书啊！

支持(0) 反对(0)

#22楼 2013-06-24 17:08 薛定谔的破猫

请教一个小问题

server端conn.sendall(reply)后，如果在客户端recv()之前，server端的socket就close掉了，此时客户端再去recv()响应会怎样呢？

试验（客户端recv前，用time.sleep()做了延迟）证明，依然可以收到响应数据，但是这里的原理应该怎样理解呢？是说虽然服务端的socket不存在了，但是这个通信的通道依然存在？

支持(0) 反对(0)

#23楼 2013-09-26 20:11 开心星

楼主的文章写的浅显易懂还到位，特别是一些细节地方的备注对我这类基础很少的人来说特别有用，以前好多不懂的疑惑一下子就解开了，感谢！

支持(0) 反对(0)

#24楼 2013-10-22 20:00 cirfe

楼主，为什么我复制了你的代码运行之后，却只能本机访问，局域网内无法访问？

支持(0) 反对(0)

#25楼 2013-10-22 20:23 cirfe

我自己搞明白了，是没有开放端口。。。。。

支持(0) 反对(0)

#26楼[楼主] 2013-10-28 10:02 Vamei

@ cirfe

不好意思，前一段没怎么看博客，没能及时回复。

支持(0) 反对(0)

#27楼 2013-11-24 18:31 wooanson

你好,

我在Windows 7, python 3 .3 .2环境下执行"http客户端程序"的时候总是会报如下的错误:

b'\x16\x03\x01\x00\xa1\x01\x00\x00\x9d\x03\x01R\x91\xd4XB)\x85u\x85U7\xd3\x9eAc\xf3\x03)V\xf3\xa6\xfe\x80t\x11z\xc2\x93,j\x9f\xb0\x00\x00H\x00\xff\xc0\n\xc0\x14\x00\x88\x00\x87\x009\x008\xc0\x0f\xc0\x05\x00\x84

\x005\xc0\t\xc0\x07\xc0\x13\xc0\x11\x00E\x00D\x003\x002\xc0\x0e\xc0\x0c\xc0\x04\xc0\x02\x00\x96\x00A\x00/\x00\x05\x00\x04\xc0\x08\xc0\x12\x00\x16\x00\x13\xc0\r\xc0\x03\xfe\xff\x00\n\x01\x00\x00,\x00\x00\x00\x0e\x00\x0c\x00\x00\tlocalhost\x00\n\x00\x08\x00\x06\x00\x17\x00\x18\x00\x19\x00\x0b\x00\x02\x01\x00\x00#\x00\x003t\x00\x00'
Traceback (most recent call last):
File "E:\Eclipse\Workshops\pyTest\pyTest\HttpServer.py", line 47, in <module>
method = request1.split(' ')[0]
TypeError: Type str doesn't support the buffer API

我googled了一下, 有的说是bytes和string的问题, 所以尝试转换成unicode string, 然后再调用split函数:

request1=request.decode('utf8')

还是没有成功,报的错是:
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb0 in position 0: invalid start byte

问题看起来似乎是, decode函数库还不够完善, 不能很好地解析二进制的字符串

请问我的理解对不对.

Anson

支持(0) 反对(0)

#28楼 2013-11-27 11:21 孔仔
特意注册了个号，给个赞，lz继续加油，

支持(0) 反对(0)

#29楼[楼主] 2013-11-27 20:09 Vamei
@ wooanson
不好意思，我对unicode部分没有深入研究。我回头查一下。

支持(0) 反对(0)

#30楼 2014-03-27 15:04 gweiwei01

写的太好了，一直没接触过python，现在研二，找工作想找测试方向的，听说得会一个脚本语言，最近开始恶补python，看这个很快了解大概，待深入研究；

支持(0) 反对(0)

#31楼 2014-04-02 23:13 刘二枣

你好 我在用另一台电脑做客户机时出现了如下错误

```
Traceback (most recent call last):
File "C:/Python34/socket_client.py", line 16, in <module>
s.sendall(request)
TypeError: 'str' does not support the buffer interface
```

请问是怎么回事？为什么说接口不支持str？是因为使用无线路由器的关系吗？

支持(0) 反对(0)

#32楼 2014-04-09 23:28 泰达希尔

楼主我下载了python2.7，把这两个程序分别保存并运行就可以吗？

支持(0) 反对(0)

#33楼 2014-05-18 00:29 bloodeyed

@ 刘二枣

查了下 好像3.* socket.send 传递的数据必须是bytes。

发送和接收数据时做下编码转换就可以了。修改如下：

```
# send message
s.sendall(request.encode())

#receive mesage
reply = s.recv(1024)
print ('reply is: ', reply.decode())
```

支持(0) 反对(0)

#34楼 2014-06-11 01:19 本本乱

提个小BUG：

「基于TCP socket的HTTP服务器」部分的例子。

http响应的text_content内容，最开头有一个换行。
这会引起一个兼容性问题。

测试环境：
服务器环境：Ubuntu
客户机环境：Mac OS X + Chrome/Safari

结果：
Chrome正确显示网页。
Safari因为开头的换行，把整个http响应当成了一个纯文本文件！
（如果装了迅雷插件之类的，还会帮你下载下来.....）

解决方法：
'''
xxx
'''.strip()
即可。

支持(0) 反对(0)

#35楼[楼主] 2014-06-11 07:53 Vamei
@ 本本乱
好的，谢谢你了。

支持(0) 反对(0)

#36楼 2014-07-07 19:57 心_心
winxp，本机实验，出现
Traceback (most recent call last):
File "C:\Python34\1.py", line 20, in <module>
pic_content = pic_content + f.read()
TypeError: Can't convert 'bytes' object to str implicitly

支持(0) 反对(0)

#37楼 2014-08-23 20:36 Nonikka
"""我们的Python程序根据这个请求，发送给服务器text_content的内容。"""

楼主这句话是发送给浏览器还是服务器啊？我有点糊涂了...

支持(0) 反对(0)

#38楼 2014-08-27 17:02 cquptzzq

写的很好

支持(0) 反对(0)

#39楼 2014-08-28 23:14 连城测

@ ___kevin

图片不能显示还有可能是因为你的文件没有传送完整，你测试一下读取图片文件和发送的时候是否读取和发送完整了，可以写个读取图片再写入一个新文件的小程序测试一下，我以前就是这里有问题导致图片显示不出来，希望可以帮到你

支持(0) 反对(0)

#40楼 2014-12-28 18:01 qbhust

你好，你的文章很精彩，我正在通过你的文章学习Python。

有个小疑问，http服务器的代码中

s.listen(3)

这一行是否应该放在while True:的外面？

支持(1) 反对(0)

#41楼 2015-01-09 17:32 Will.Hu

@ 连城测

我测试的时候，

```
1 pic_content += f.read()
```

是可以debug可以看到完整的内容的，但是测试127.0.0.1:8000还是看不到图片，图片的URL是

http://127.0.0.1:8000/RedHat.jpg

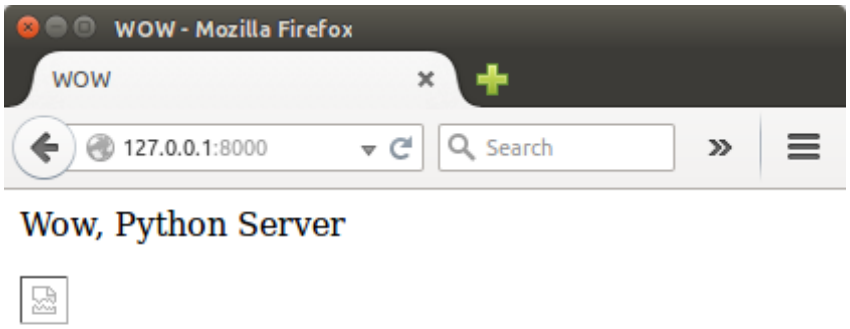
而且test.jpg和server.py在同一级目录下，

而且也可以看到server端的信息：

```
1 Connected by ('127.0.0.1', 45098)
2 Request is: GET /RedHat.jpg HTTP/1.1
3 Host: 127.0.0.1:8000
4 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:34.0) Gecko/20100101 Firefox/
5 Accept: image/png,image/*;q=0.8,*/*;q=0.5
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
```

```
Referer: http://127.0.0.1:8000/
Connection: keep-alive
```

但是就是不能是显示图片：



希望得到帮助，谢谢！！

支持(0) 反对(0)

#42楼 2015-01-09 20:21 Will.Hu

我知道什么原因了。

```
f = open('RedHat.jpg', "rb")
pic_content = '''
HTTP/1.x 200 OK
Content-Type: image/jpg

'''
pic_content += f.read()
f.close()
```

我实际在写的时候，把image/jpg下面的那空行给手动删除了，所以出现了问题。

支持(0) 反对(0)

#43楼 2015-04-10 02:26 piperck

@ 本本乱

@Vamei

我用的macos 这个问题我也遇见了 直接被 safari解析成了<pre> 标签的东西。。我用

了 .strip函数 但是貌似没有解决的了 请问你具体怎么解决 让他解析出来的？ 博主看到知道的话 也请能回复我一下吗0 0

支持(0) 反对(0)

#44楼 2015-08-03 09:29 focky

@ 心_心

我也出现了这个问题，请问解决了吗？

支持(0) 反对(0)

#45楼 2015-08-06 14:04 zouyeah

博主，提几个python3+win7下的几个bug哈：

1.f.read()要加上f.read().decode('utf-8', 'ignore'),貌似python3区分了bytes和str

2.request =conn.recv(1024).decode()需要对接收回来的数解码。

顺便回答下@focky

支持(0) 反对(0)

#46楼 2016-06-03 16:25 甩锅侠

楼主我爱你，多出好文章。

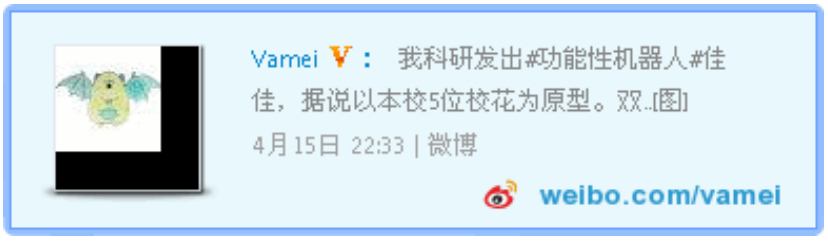
支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

公告

你好，这里是Vamei，一名编程爱好者。我在博客里写了**Python/Linux/**网络协议**/**算法**/Java/**数据科学系列文章，从这里开始阅读。非常期待和你的交流。

我的微博 

下列教程已经做成电子出版物，内容经过修订，也方便离线阅读：

协议森林

欢迎阅读我写的其他书籍：

现代小城的考古学家

天气与历史的相爱相杀

随手拍光影

昵称：Vamei
园龄：4年1个月
荣誉：推荐博客
粉丝：4985
关注：26
+加关注

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

Python(61)

Java(42)

大数据(22)

Linux(17)
网络(16)
算法(15)
文青(14)
技普(9)
系列索引(6)
开发工具(4)
更多

系列文章
Java快速教程
Linux的概念与体系
Python快速教程
数据科学
协议森林
纸上谈兵: 算法与数据结构

积分与排名
积分 - 659668
排名 - 122

最新评论
1. Re:Java基础11 对象引用
受教!

--MisslLost

2. Re:Python快速教程
看评论区一片喝彩！看来我得在此扎营了！

--测试小蚂蚁

3. Re:Python进阶06 循环对象
好好地列表解析变成了表推导

--ashic

4. Re:“不给力啊，老湿！”：RSA加密与破解
感谢楼主精彩分享

--worldball

5. Re:概率论04 随机变量
你写的这一系列太棒了，刚加入博客园就在你这里学到了，我要转载过去学习一下

--yixius

6. Re:Python基础03 序列

挺好的教程、、、、

--王小拽的号

7. Re:Python进阶07 函数对象

def func(x,y): print x**ydef test(f,a,b): print 'test' print f(a,b)test (func,3,2)输出的内容:tes.....

--M-edea

8. Re:Python进阶02 文本文件的输入输出

@coderXT换行符：\n...

--行者之印

9. Re:数据科学

博主啊，这里是一枚即将大二的计算机新人，大一学了python，java，还有一些算法，数据结构，图论了，感觉我对数学又一些反感，但是听说离散数学对计算机专业的很重要，不知道怎么去学比较好呢，我想像您写.....

--Acokil

10. Re:为什么要写技术博

楼主是用自己自定义的模板吗？在博客园里找不到这种风格的blog模板？

--行者之印

11. Re:来玩Play框架01 简介

挖煤哥,我补充了一下Windows下的搭建play框架,希望有点帮助,谢谢!

--Sungeek

12. Re:来玩Play框架07 静态文件

@helper.form(action = routes.Application.upload, 'enctype -> "multipart/form-data") {--action = rout.....

--quxiaozha

13. Re:来玩Play框架07 静态文件

该记录将/assets/下的URL，对应到项目的/public文件夹内的文件。比如在项目的/public/images/test.jpg，就可以通过/assests/images/test.jpg这一.....

--quxiaozha

14. Re:来玩Play框架06 用户验证

支持挖煤哥~~~

--quxiaozha

15. Re:“不给力啊，老湿！”：RSA加密与破解

@maanshancss请你仔细阅读了这个文章再来评价。...

--Vamei

推荐排行榜

1. “不给力啊，老湿！”：RSA加密与破解(218)
2. Python快速教程(140)
3. 野蛮生长又五年(91)
4. Java快速教程(88)
5. 协议森林01 邮差与邮局 (网络协议概观)(79)
6. 为什么要写技术博(71)
7. 编程异闻录(54)
8. 博客一年：心理之旅(49)
9. 协议森林08 不放弃 (TCP协议与流通信)(45)
10. Python快速教程 尾声(43)
11. 协议森林(42)
12. Java基础01 从HelloWorld到面向对象(42)
13. Python基础08 面向对象的基本概念(40)
14. 一天能学会的计算机技术(34)
15. 博客第二年，杂谈(33)