最近在几周在做手游崩溃信息收集和上传，拿到崩溃信息后，使用的是HTTP的POST方法上传到公司共用的服务器的，因此做简单总结。本文首先简单介绍了HTTP协议，主要说明了POST方法和GET方法的区别；然后用Python实现了 对POST方法和GET方法的响应；最后用Python模拟了POST方法和GET方法的请求。

**HTTP协议简介**

HTTP是Hyper Text Transfer Protocol（超文本传输协议）的缩写，简单来说它是一个应用层的协议，它允许将超文本标记语言(HTML)文档从Web服务器传送到客户端的浏览器。HTTP是一个无状态的协议，即同一个客户端的这次请求和上次请求是没有对应关系，对http服务器来说，它并不知道这两个请求来自同一个客户端，为了解决这个问题，Web程序引入了Cookie机制来维护状态。

HTTP协议通常基于TCP协议来实现的，有时也基于于TLS或SSL协议（这个两个协议也是基于TCP协议来说实现）来实现，这个时候，就成了我们常说的HTTPS，每次HTTP操作都至少有下面几个过程：首先客户端与服务端建立连接；建立建立后，客户端按照协议格式发送请求；服务端接到请求后，同样按照某个格式返回响应数据；最后客户端与服务端断开连接。

通常我们打开一个网页，需要浏览器发送多次请求，因为一个网页中可能引用了其他文件，比如图片等文件，这时候浏览器会自动再次发送请求去获取图片等数据，直到网页上的数据被完全显示出来。

**POST和GET区别**

HTTP协议定义了很多与服务器交互的方法，最基本的有4种，分别是GET,POST,PUT,DELETE. 一个URL地址用于描述一个网络上的资源，而HTTP中的GET, POST, PUT, DELETE就对应着对这个资源的查，改，增，删4个操作，其中最常见请求方式是GET和POST，并且现在浏览器一般只支持GET和POST方法。GET一般用于获取/查询资源信息，而POST一般用于更新资源信息，他们之间主要区别如下：

1）根据HTTP规范，GET用于信息获取，而且应该是安全的和幂等的，这里安全是指该操作用于获取信息而非修改信息，幂等是指对同一URL的多个请求应该返回同样的结果（这一点在实质实现时，可能并不满足）；

POST表示可能修改变服务器上的资源的请求。

2）GET请求的数据会附在URL之后（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，参数之间以&相连，如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，

则直接把字符串用BASE64编码；POST把提交的数据则放置在是HTTP包的包体中。

3）因为GET是通过URL提交数据，那么GET可提交的数据量就跟URL的长度有直接关系，理论上URL长度是没有限制的，即HTTP协议没有规定URL的长度，但在实质中，特定的浏览器可能对这个长度做了限制；理论上POST也是没有大小限制的，HTTP协议规范也没有进行大小限制，但在服务端通常会对这个大小做一个限制，当然这个限制比GET宽松的多，即使用POST可以提交的数据量比GET大得多。

最后，网上有人说，POST的安全性要比GET的安全性高，实质上POST跟GET都是明文传输，这可以通过类似WireShark工具看到。总之，Get是向服务器发索取数据的一种请求，而Post是向服务器提交数据的一种请求。

**POST和GET方法响应Python实现**

下面代码实现对POST方法和GET方法的响应：

#!/usr/bin/python

#coding=utf8

"""

import sys

reload(sys)

sys.setdefaultencoding('utf-8')

"""

from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer

from os import curdir, sep

import cgi

import logging

import time

PORT\_NUMBER = 8080

RES\_FILE\_DIR = "."

class myHandler(BaseHTTPRequestHandler):

def do\_GET(self):

if self.path=="/":

self.path="/index\_example3.html"

try:

#根据请求的文件扩展名，设置正确的mime类型

sendReply = False

if self.path.endswith(".html"):

mimetype='text/html'

sendReply = True

if self.path.endswith(".jpg"):

mimetype='image/jpg'

sendReply = True

if self.path.endswith(".gif"):

mimetype='image/gif'

sendReply = True

if self.path.endswith(".js"):

mimetype='application/javascript'

sendReply = True

if self.path.endswith(".css"):

mimetype='text/css'

sendReply = True

if sendReply == True:

#读取相应的静态资源文件，并发送它

f = open(curdir + sep + self.path, 'rb')

self.send\_response(200)

self.send\_header('Content-type',mimetype)

self.end\_headers()

self.wfile.write(f.read())

f.close()

return

except IOError:

self.send\_error(404,'File Not Found: %s' % self.path)

def do\_POST(self):

logging.warning(self.headers)

form = cgi.FieldStorage(

fp=self.rfile,

headers=self.headers,

environ={'REQUEST\_METHOD':'POST',

'CONTENT\_TYPE':self.headers['Content-Type'],

})

file\_name = self.get\_data\_string()

path\_name = '%s/%s.log' % (RES\_FILE\_DIR,file\_name)

fwrite = open(path\_name,'a')

fwrite.write("name=%s\n" % form.getvalue("name",""))

fwrite.write("addr=%s\n" % form.getvalue("addr",""))

fwrite.close()

self.send\_response(200)

self.end\_headers()

self.wfile.write("Thanks for you post")

def get\_data\_string(self):

now = time.time()

clock\_now = time.localtime(now)

cur\_time = list(clock\_now)

date\_string = "%d-%d-%d-%d-%d-%d" % (cur\_time[0],

cur\_time[1],cur\_time[2],cur\_time[3],cur\_time[4],cur\_time[5])

return date\_string

try:

server = HTTPServer(('', PORT\_NUMBER), myHandler)

print 'Started httpserver on port ' , PORT\_NUMBER

server.serve\_forever()

except KeyboardInterrupt:

print '^C received, shutting down the web server'

server.socket.close()

对于上面POST响应实现，值得一提的是，若客户端发送过来一个文件，则方法getvalue()会把整个文件内容读入内存，这可能不是我们想要的，这时可以使用form的属性file或filename，比如下面代码，计算上传代码的行数：

fileitem = form["userfile"]

if fileitem.file:

linecount = 0

while 1:

line = fileitem.file.readline()

if not line: break

linecount = linecount + 1

**POST和GET方法请求Python实现**

下面代码实现了GET方法的请求：

#!/usr/bin/env python

#coding=utf8

import httplib

httpClient = None

try:

httpClient = httplib.HTTPConnection('localhost', 8080, timeout=30)

httpClient.request('GET', '/test0.html')

#response是HTTPResponse对象

response = httpClient.getresponse()

print response.status

print response.reason

print response.read()

except Exception, e:

print e

finally:

if httpClient:

httpClient.close()

下面代码实现了POST方法的请求：

#!/usr/bin/env python

#coding=utf8

import httplib, urllib

httpClient = None

try:

params = urllib.urlencode({'name': 'Maximus', 'addr': "GZ"})

headers = {"Content-type": "application/x-www-form-urlencoded"

, "Accept": "text/plain"}

httpClient = httplib.HTTPConnection("localhost", 8080, timeout=30)

httpClient.request("POST", "/test0.html", params, headers)

response = httpClient.getresponse()

print response.status

print response.reason

print response.read()

print response.getheaders() #获取头信息

except Exception, e:

print e

finally:

if httpClient:

httpClient.close()