概述

继续requests基础分享，本文主要分享以下内容：

- 请求头定制
- POST请求

请求头定制示例

在requests中想要为请求添加自定义头信息，只需要简单的传入一个dict（即python字典类型对象）即可。

下面我们看一个简单的示例：

```
#-*- coding:utf-8 -*-

__author__ = "苦叶子"


# 导入模块
import requests

if __name__ == "__main__":
    print("开源优测 - requests自定义请求头基本示例")

    url = "http://www.baidu.com"

    # 定义自定义请求头数据
    headers = {
        "user-agent": "www.testingunion.com",
        "custom-head": "DeepTest"
    }

    # 发送带自定义头的请求
    r = requests.get(url, headers=headers)
```

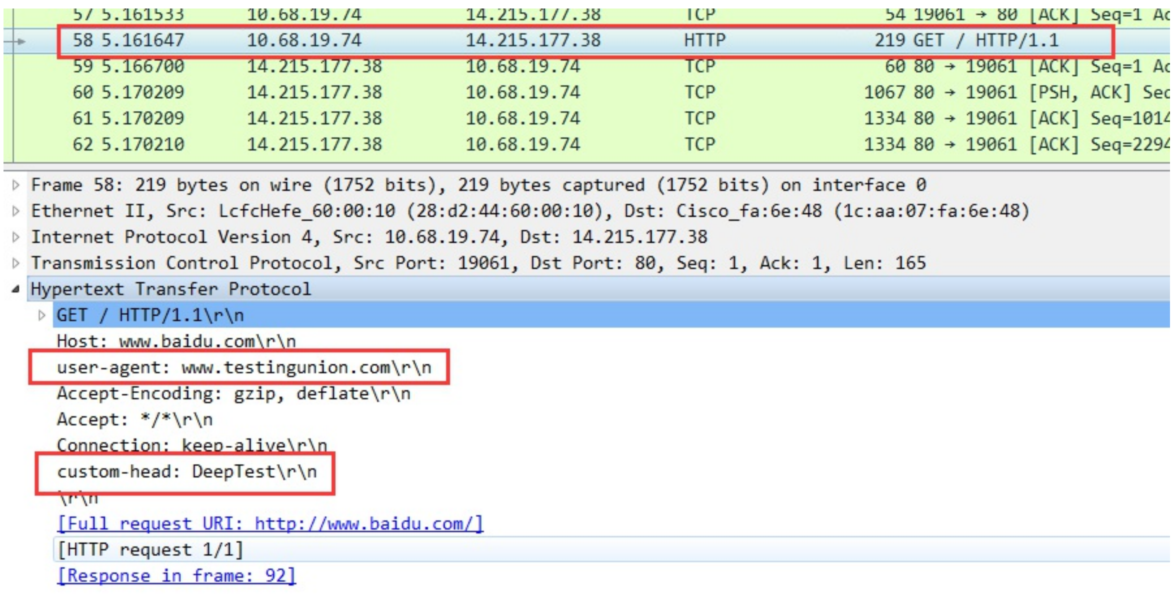
将上述代码保存至requests_headers_demo.py中，执行以下命令：

注：所有的header值必须是string、bytestring或unicode，虽然传递unicode header是允许的，但不建议这样做

> python requests_headers_demo.py

在运行上述命令前，先启动wireshark，用来抓取报文，看下我们自定义的headers是否正常被设置。

抓取的报文如下：


```
57 5.161533    10.68.19.74     14.215.177.38    TCP      54 19061 → 80 [ACK] Seq=1 Ac
58 5.161647    10.68.19.74     14.215.177.38    HTTP     219 GET / HTTP/1.1
59 5.166700    14.215.177.38   10.68.19.74      TCP      60 80 → 19061 [ACK] Seq=1 Ac
60 5.170209    14.215.177.38   10.68.19.74      TCP      1067 80 → 19061 [PSH, ACK] Sec
61 5.170209    14.215.177.38   10.68.19.74      TCP      1334 80 → 19061 [ACK] Seq=1014
62 5.170210    14.215.177.38   10.68.19.74      TCP      1334 80 → 19061 [ACK] Seq=2294
```
```
▷ Frame 58: 219 bytes on wire (1752 bits), 219 bytes captured (1752 bits) on interface 0
▷ Ethernet II, Src: LcfcHefe_60:00:10 (28:d2:44:60:00:10), Dst: Cisco_fa:6e:48 (1c:aa:07:fa:6e:48)
▷ Internet Protocol Version 4, Src: 10.68.19.74, Dst: 14.215.177.38
▷ Transmission Control Protocol, Src Port: 19061, Dst Port: 80, Seq: 1, Ack: 1, Len: 165
◢ Hypertext Transfer Protocol
  ▷ GET / HTTP/1.1\r\n
    Host: www.baidu.com\r\n
    user-agent: www.testingunion.com\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    Connection: keep-alive\r\n
    custom-head: DeepTest\r\n
    \r\n
    [Full request URI: http://www.baidu.com/]
    [HTTP request 1/1]
    [Response in frame: 92]
```

从报文来看，我们的设置是成功的，这说明了requests的机制是多么的简洁有效。

POST请求示例

下面我们看看requests如何发送HTTP POST请求的。

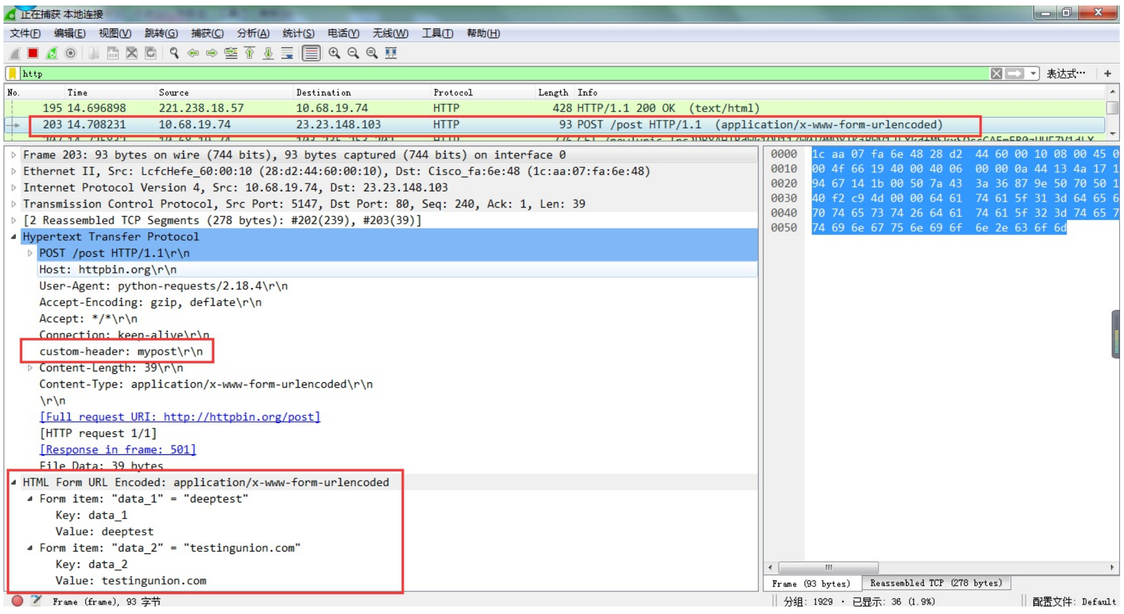
基本示例

```
#-*- coding:utf-8 -*-

__author__ = "苦叶子"

import requests

if __name__ == "__main__":
    print("requests post示例")

    # 目标url
    url = "http://httpbin.org/post"

    # 请求头headers
    headers = {"custom-header": "mypost"}

    # 要post的数据
    data = {"data_1": "deeptest", "data_2": "testingunion.com"}

    # 发送post请求
    r = requests.post(url, data=data, headers=headers)

    # 输出结果
    print(r.text)
```

将上述代码保存到requests_post_demo.py中，执行下述命令运行:

```
python requests_post_demo.py
```

用wireshark抓取上述自定义了header和data的报文如下：


正在捕获 本地连接
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
http
表达式…

```
No.     Time          Source           Destination      Protocol  Length  Info
  195 14.696898    221.238.18.57    10.68.19.74      HTTP         428 HTTP/1.1 200 OK  (text/html)
  203 14.708231    10.68.19.74      23.23.148.103    HTTP          93 POST /post HTTP/1.1  (application/x-www-form-urlencoded)
```

```
▷ Frame 203: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0
▷ Ethernet II, Src: LcfcHefe_60:00:10 (28:d2:44:60:00:10), Dst: Cisco_fa:6e:48 (1c:aa:07:fa:6e:48)
▷ Internet Protocol Version 4, Src: 10.68.19.74, Dst: 23.23.148.103
▷ Transmission Control Protocol, Src Port: 5147, Dst Port: 80, Seq: 240, Ack: 1, Len: 39
▷ [2 Reassembled TCP Segments (278 bytes): #202(239), #203(39)]
◢ Hypertext Transfer Protocol
  ▷ POST /post HTTP/1.1\r\n
    Host: httpbin.org\r\n
    User-Agent: python-requests/2.18.4\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    Connection: keep-alive\r\n
    custom-header: mypost\r\n
  ▷ Content-Length: 39\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    \r\n
    [Full request URI: http://httpbin.org/post]
    [HTTP request 1/1]
    [Response in frame: 501]
    File Data: 39 bytes
◢ HTML Form URL Encoded: application/x-www-form-urlencoded
  ◢ Form item: "data_1" = "deeptest"
      Key: data_1
      Value: deeptest
  ◢ Form item: "data_2" = "testingunion.com"
      Key: data_2
      Value: testingunion.com
```

```
0000  1c aa 07 fa 6e 48 28 d2  44 60 00 10 08 00 45 0
0010  00 4f 66 19 40 00 40 06  00 00 0a 44 13 4a 17 1
0020  94 67 14 1b 00 50 7a 43  3a 36 87 9e 50 70 50 1
0030  40 f2 c9 4d 00 00 64 61  74 61 5f 31 3d 64 65 6
0040  70 74 65 73 74 26 64 61  74 61 5f 32 3d 74 65 7
0050  74 69 6e 67 75 6e 69 6f  6e 2e 63 6f 6d
```

```
Frame (93 bytes)   Reassembled TCP (278 bytes)
Frame (frame), 93 字节   分组: 1929 · 已显示: 36 (1.9%)   配置文件: Default
```

post json数据示例

下面我们看看如何postjson数据到服务。

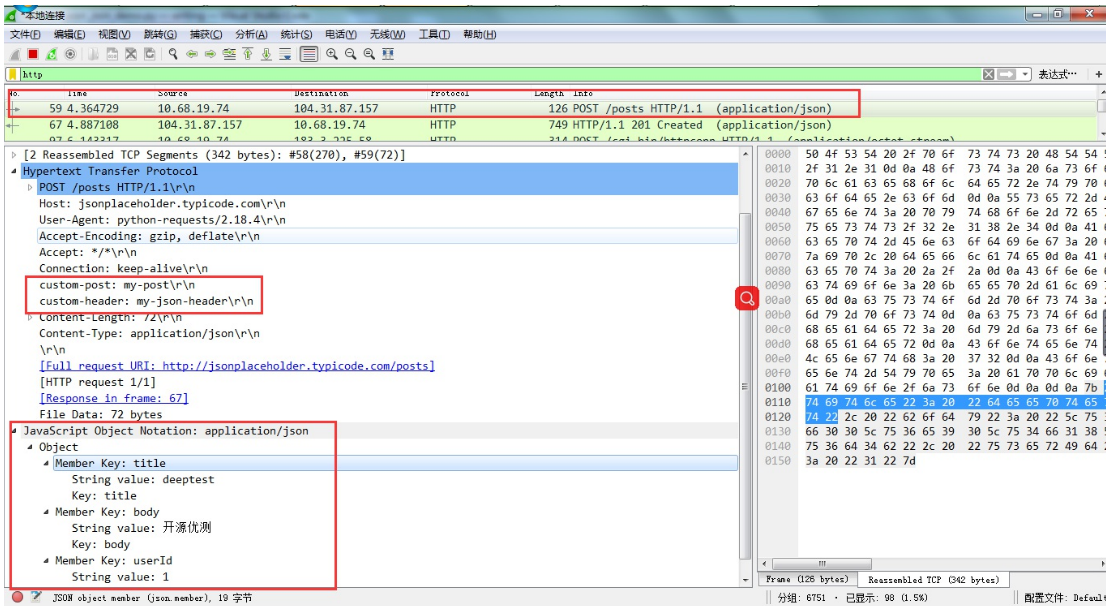
```python
#-*- coding:utf-8 -*-

__author__ = "苦叶子"

import requests

if __name__ == "__main__":
    print("requests post json数据示例")

    # 目标服务url
    url = "http://jsonplaceholder.typicode.com/posts"

    # 自定义头
    headers = {
        "custom-post": "my-post",
        "custom-header": "my-json-header"
        }

    # 要post的数据
    json_data = {
        "title": "deeptest",
        "body": "开源优测",
        "userId": "1"
        }
```

```
# post json格式的数据
r = requests.post(url, json=json_data, headers=headers)

# 打印下返回结果
print(r.text)
```

将上述代码保存到requests_post_json_demo.py中，执行下述命令运行:

```
python requests_post_json_demo.py
```

对上述代码执行，使用wireshark对http报文进行抓包如下:



*本地连接
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
表达式…

```
http

No.   Time         Source          Destination      Protocol   Length  Info
  59  4.364729     10.68.19.74     104.31.87.157    HTTP        126 POST /posts HTTP/1.1  (application/json)
  67  4.887108     104.31.87.157   10.68.19.74      HTTP        749 HTTP/1.1 201 Created  (application/json)
```

```
[2 Reassembled TCP Segments (342 bytes): #58(270), #59(72)]
Hypertext Transfer Protocol
  POST /posts HTTP/1.1\r\n
  Host: jsonplaceholder.typicode.com\r\n
  User-Agent: python-requests/2.18.4\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept: */*\r\n
  Connection: keep-alive\r\n
  custom-post: my-post\r\n
  custom-header: my-json-header\r\n
  Content-Length: 72\r\n
  Content-Type: application/json\r\n
  \r\n
  [Full request URI: http://jsonplaceholder.typicode.com/posts]
  [HTTP request 1/1]
  [Response in frame: 67]
  File Data: 72 bytes
JavaScript Object Notation: application/json
  Object
    Member Key: title
      String value: deeptest
      Key: title
    Member Key: body
      String value: 开源优测
      Key: body
    Member Key: userId
      String value: 1
```

```
0000  50 4f 53 54 20 2f 70 6f  73 74 73 20 48 54 54
0010  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 6a 73 6f
0020  70 6c 61 63 65 68 6f 6c  64 65 72 2e 74 79 70
0030  63 6f 64 65 2e 63 6f 6d  0d 0a 55 73 65 72 2d
0040  67 65 6e 74 3a 20 70 79  74 68 6f 6e 2d 72 65
0050  75 65 73 74 73 2f 32 2e  31 38 2e 34 0d 0a 41
0060  63 65 70 74 2d 45 6e 63  6f 64 69 6e 67 3a 20
0070  7a 69 70 2c 20 64 65 66  6c 61 74 65 0d 0a 41
0080  63 65 70 74 3a 20 2a 2f  2a 0d 0a 43 6f 6e 6e
0090  63 74 69 6f 6e 3a 20 6b  65 65 70 2d 61 6c 69
00a0  65 0d 0a 63 75 73 74 6f  6d 2d 70 6f 73 74 3a
00b0  6d 79 2d 70 6f 73 74 0d  0a 63 75 73 74 6f 6d
00c0  68 65 61 64 65 72 3a 20  6d 79 2d 6a 73 6f 6e
00d0  68 65 61 64 65 72 0d 0a  43 6f 6e 74 65 6e 74
00e0  4c 65 6e 67 74 68 3a 20  37 32 0d 0a 43 6f 6e
00f0  65 6e 74 2d 54 79 70 65  3a 20 61 70 70 6c 69
0100  61 74 69 6f 6e 2f 6a 73  6f 6e 0d 0a 0d 0a 7b
0110  74 69 74 6c 65 22 3a 20  22 64 65 65 70 74 65
0120  74 22 2c 20 22 62 6f 64  79 22 3a 20 22 5c 75
0130  66 30 30 5c 75 36 65 39  30 5c 75 34 66 31 38
0140  75 36 64 34 62 22 2c 20  22 75 73 65 72 49 64
0150  3a 20 22 31 22 7d
```

```
Frame (126 bytes)   Reassembled TCP (342 bytes)
JSON object member (json.member), 19 字节   分组: 6751 · 已显示: 98 (1.5%)   配置文件: Default
```

扫一扫关注微信公众号：