# 接口自动化

## 电脑上安装fiddler

默认是抓http请求的，对于pc上的https请求，会提示网页不安全，这时候需要在浏览器上安装证书。

### PC上https请求提示网页不安全，fiddler设置

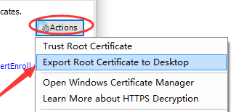
1.打开菜单栏：Tools> Options>HTTPS

2.勾选Decrypt HTTPS traffic，里面的两个子菜单也一起勾选

### 导出证书

1.点右上角Actions按钮

2.选第二个选项，导出到桌面，此时桌面上会多一个文件：FiddlerRoot.cer。



### 导入到firefox浏览器

1. 打开右上角浏览器设置》选项》高级》证书》查看证书》证书机构》导入
2. 打开文件后，会弹出个框，勾选三个选项就完成操作啦。

## 只抓APP请求

fiddler在抓手机app的请求时候，通常也会抓到来自PC的请求，导致会话消息太多，那么如何把来自pc的请求过滤掉，只抓来自APP的请求呢？

必备环境：

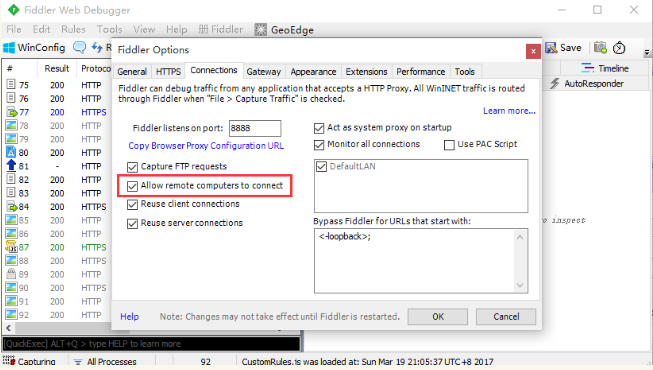
1.电脑上已装fiddler

2.手机和电脑在同一局域网

### Fiddler设置

1.fiddler>Tools>Fiddler Options>Connections 勾选Allow remote computers to connect。

2.记住这里的端口号：8888，后面会用到。



### 手机设置代理

1. 手机设置->WLAN设置->选择该wifi，点右边的箭头（有的手机是长按弹出选项框）。

2.选择修改网络配置：

代理：手动

主机名：与主机电脑IP地址保持一致

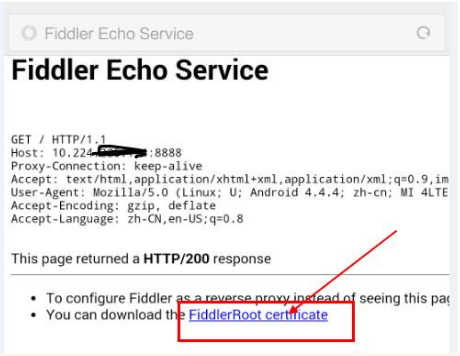
端口号：8888

### 抓APP上的HTTPS请求

1.如果app都是http请求，是不需要安装证书，能直接抓到的，如果是https请求，这时候手机就需要下载证书了。

2.打开手机浏览器输入：http://10.224.xx.xx:8888 ，这个中间的host地址就是前面查到的本机地址。（fiddler要重启）

3.出现如下画面，点箭头所指的位置，点击安装就可以了。



### Fiddler设置过滤

1.手机上设置代理后，这时候fiddler上抓到的是pc和app所有的请求，如果pc上打开网址，会很多，这时候就需要开启过滤功能了。

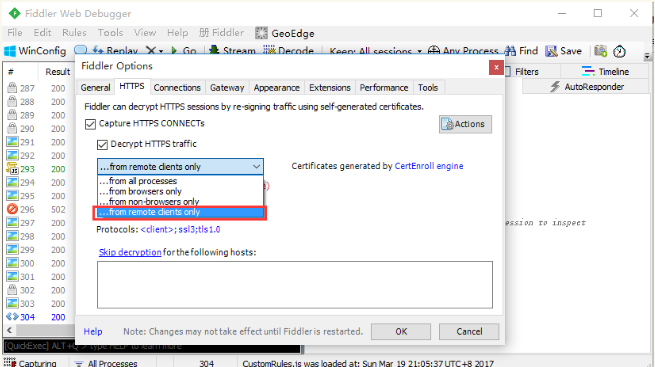
2.打开fiddler>Tools>Fiddler Options>HTTPS>...from remote clients only,勾选这个选项就可以了

...from all processes :抓所有的请求

...from browsers only ：只抓浏览器的请求

...from non-browsers only :只抓非浏览器的请求

...from remote clients only:只抓远程客户端请求



## 查看get与post请求

清屏（cls）,在左下角命令行输入cls，清空屏幕

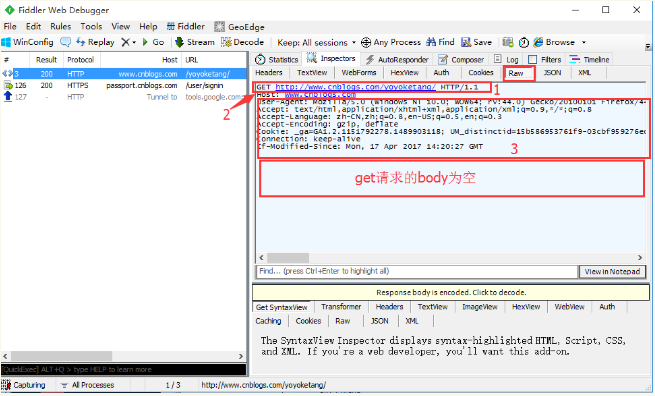
（清屏也可以使用快捷键Ctrl+X）

2.get请求的**Raw**参数查看，主要分三部分：

--第1部分是请求url地址

--第2部分是host地址

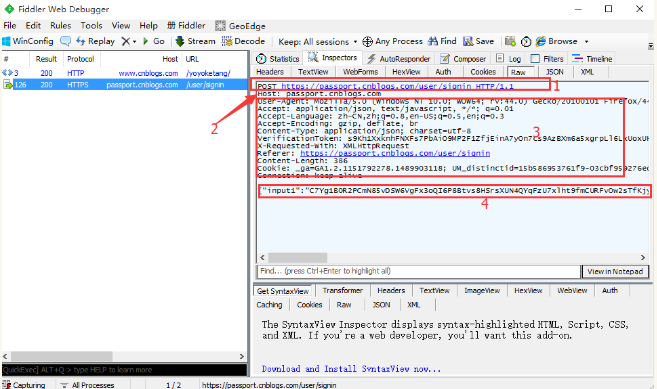
--第3部分是请求头部信息header



3.再查看博客登录请求的Raw信息，post的信息分四部分。

--前面3块内容都一样，第3部分和第4部分中间会空一行

--第4部分内容就是post请求的请求body（get请求是没body的）



## 工具介绍（request和response）

二、会话框

1.会话框主要查看请求的一些请求的一些基本信息，如# 、result、protocol、host、url、body、 caching、content-type、process

2、#：会话框列表最左侧，#号这一栏是代表这个请求大概是什么内容，<>这个符号就是我们一般要测试的请求与响应的类型。

3.result:这里是服务器返回的代码，如

--200，请求ok;2xx一般是服务器接受成功了并处理

--3xx，重定向相关

--4xx,404最常见的的就是找不到服务器,一般是请求地址有问题

--5xx,这个一般是服务器本身的错误

4.protocol：这个是协议类型，如http、https

5.host：主机地址或域名

6.url:请求的路径

7.body：该条请求产生的数据大小

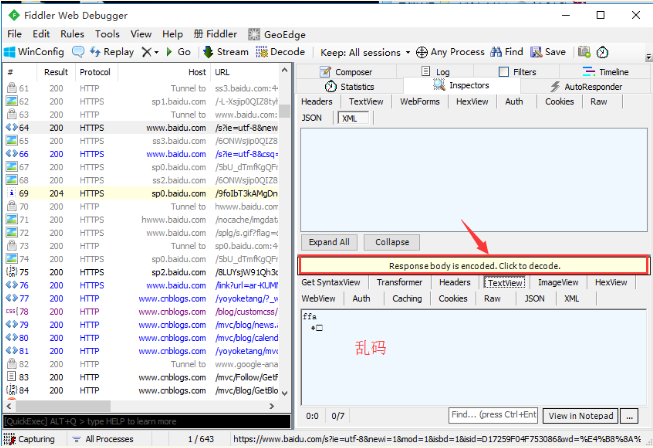
8.caching：缓存相关

9.content-type：连接类型

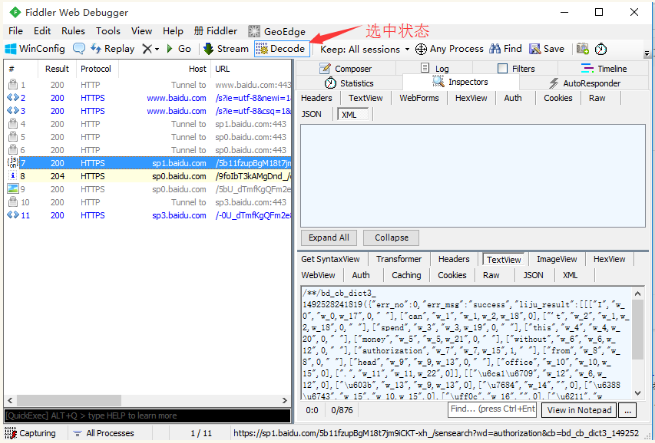
10.process:客户端类型

四、decode解码

1.如果response的TextView区域出现乱码情况，可以直接点下方黄色区域解码



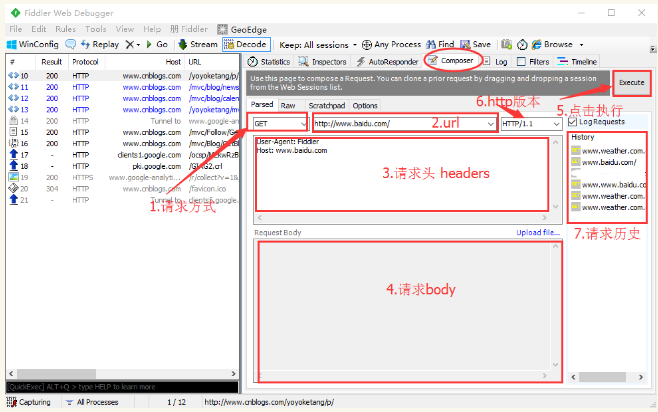
2.也可以选中上方快捷菜单decode，这样后面的请求都会自动解码了



## 接口测试（Composer）

### Composer简介

点开右侧Composer区域，可以看到如下界面，就是测试接口的界面了。



1.请求方式：点开可以勾选请求协议是get、post等

2.url地址栏：输入请求的url地址

3.请求头：第三块区域可以输入请求头信息

4.请求body：post请求在此区域输入body信息

5.执行：Execute按钮点击后就可以执行请求了

6.http版本：可以勾选http版本

7.请求历史：执行完成后会在右侧History区域生成历史记录

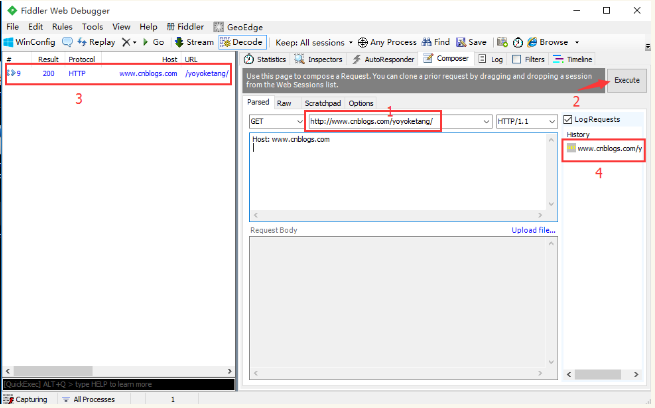
### 模拟get请求

1.在Composer区域地址栏输入博客首页：http://www.cnblogs.com/yoyoketang/

2.选择get请求，点Execute执行，请求就可以发送成功啦

3.请求发送成功后，左边会话框会生成一个会话记录，可以查看抓包详情

4.右侧history区域会多一个历史请求记录



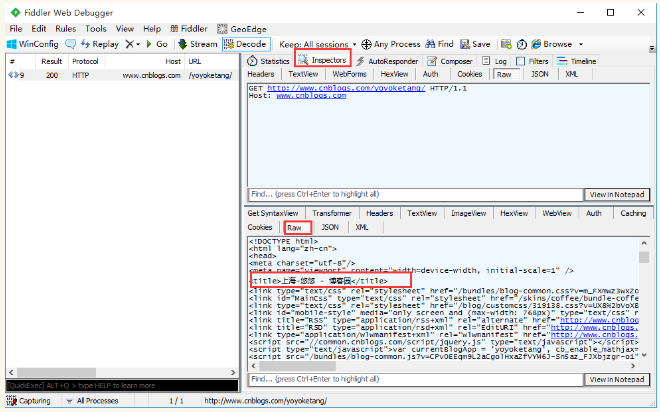
5.会话框选中该记录，查看测试结果：

--选中该会话，点开Inspectors

--response区域点开Raw区域

--Raw查看的是HTML源码的数据

--也可以点WebView，查看返回的web页面数据

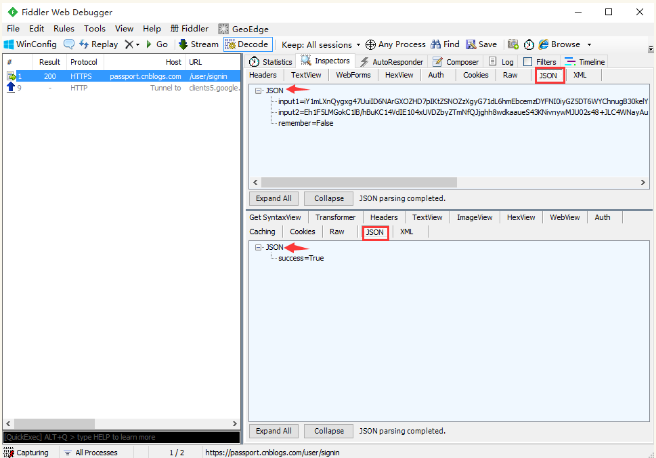


### Json数据

1.有些post的请求参数和返回参数是Json格式的，如博客园的登录请求：https://passport.cnblogs.com/user/signin

2.在登录页面手动输入账号和密码，登录成功。

3.找到这个登录成功的会话，查看json数据如下图：

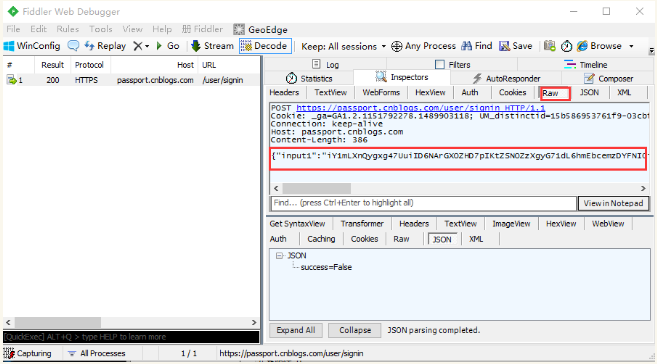


### 模拟post请求

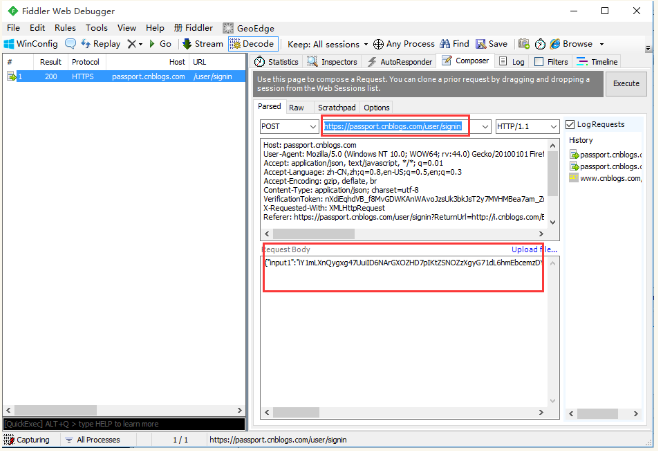
1.请求类型勾选post

2.url地址栏输入对应的请求地址

3.body区域写登录的json参数，json参数直接copy上一步抓包的数据，如下图红色区域



4.header请求头区域，可以把前面登录成功后的头部抓包的数据copy过来



5.执行成功后查看测试结果：

--执行成功如第三所示的图，显示success=True

--执行失败如下图所示，显示

message=Invalid length for a Base-64 char array or string.

success=False

## get请求（url详解）

有些get请求会带有参数，本篇详细介绍url地址格式。

可以做个简单的对比，在地址栏分别输入：

https:www.baidu.com

https://www.baidu.com/s?wd=上海悠悠博客园

对比打开的页面有什么不一样，现在知道作用了吧，也就是说这个多的"/s？wd=上海悠悠博客园"就是搜索的结果页面

### url解析

## 打断点bpu

### 断点

两种方式：

1.before response：这个是打在request请求的时候，未到达服务器之前

2.after response：也就是服务器响应之后，在Fiddler将响应传回给客户端之前。

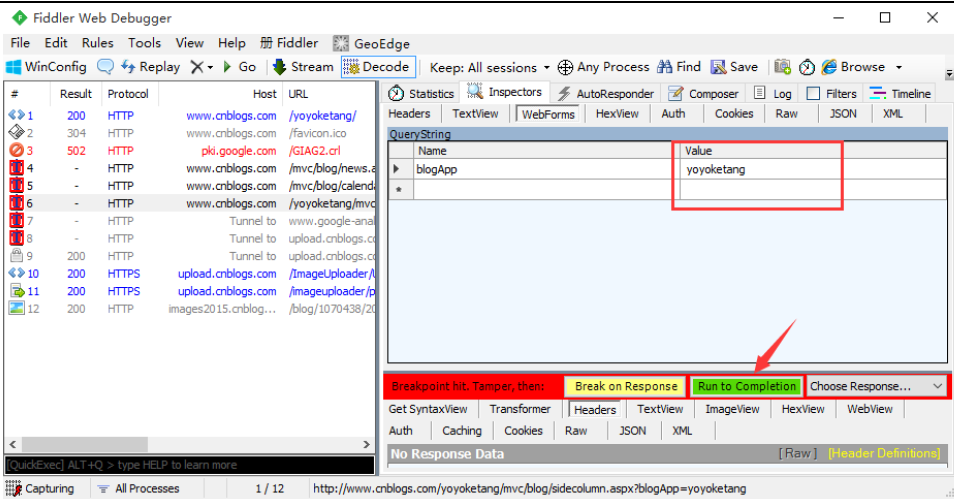
找到需要修改的请求后，选中该条会话，右侧打开WebFroms,这时候里面的参数都是可以修改的了。

### 全局断点

全局断点就是中断fiddler捕获的所有请求，先设置下，点击rules-> automatic breakpoint ->before requests

打完断点后，会发现所有的请求都无法发出去了，这时候，点下Go按钮，就能走下一步了。

找到需要修改的请求后，选中该条会话，右侧打开WebFroms,这时候里面的参数都是可以修改的了



修改之后点Run to Completion就能提交了，于是就成功修改了请求参数了

打全局断点的话，是无法正常上网的，需要清除断点：rules-> automatic breakpoint ->disabled

### 单个断点

请求前断点（before response)： bpu

1. 论坛登录接口：https://passport.cnblogs.com/user/signin

2. 命令行输入：**bpu** https://passport.cnblogs.com/user/signin 回车

取消断点，在命令行输入： bpu 回车就可以了

响应后断点（after requests）： bpafter

1. 论坛登录接口：https://passport.cnblogs.com/user/signin

2. 在命令行输入：**bpafter** https://passport.cnblogs.com/user/signin 回车

3.登录博客园，会发现已经拦截到登录后服务器返回的数据了，此时可以修改任意返回数据

4.取消断点，在命令行输入： bpafter 回车就可以了

### 命令行其它相关指令

Bpafter，Bps,bpv,bpm,bpu

这几个命令主要用于批量设置断点

Bpafter xxx: 中断 URL 包含指定字符的全部 session 响应

Bps xxx: 中断 HTTP 响应状态为指定字符的全部 session 响应

Bpv xxx: 中断指定请求方式的全部 session 响应

Bpm xxx: 中断指定请求方式的全部 session 响应 、、同于 bpv xxx

Bpu xxx:与bpafter类似

当这些命令没有加参数时，会清空所有设置了断点的HTTP请求。

## 会话保存

点左上角File>Save>Selected Sessions>as Text，保存到电脑上就是文本格式的

1.save-All Sessions :保存所有的会话，saz文件

2.save-Selected Session:保存选中的会话

--in ArchiveZIP ：保存为saz文件

--as Text ：以txt文件形式保存整个会话包括Request和Response

--as Text (Headers only) ：仅保存头部

3.Request：保存请求

--Entir Request:保存整个请求信息（headers和body）

--Request Body:只保存请求body部分

4.Response：保存返回

--Entir Response:保存整个返回信息（headers和body）

--Response Body:只保存返回body部分

--and Open as Local File：保存Response信息，并打开文件

### Repaly

1.导入请求后，可以选中某个请求，点击Repaly按钮，重新发请求

2.也可以ctrl+all全部选中后，点Repaly按钮，一次性批量请求

这里保存会话和replay功能其实就是相当于录制和回放了

### 请求方法包含：

1) get

请求指定的页面信息，并返回实体主体。

2) post

向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。

3) HEAD

类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头

4) OPTIONS

返回服务器针对特定资源所支持的HTTP请求方法，也可以利用向web服务器发送‘\*’的请求来测试服务器的功能性

5) PUT

向指定资源位置上传其最新内容

6) DELETE

请求服务器删除Request-URL所标识的资源

7) TRACE

回显服务器收到的请求，主要用于测试或诊断

8） CONNECT

HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

注意：

1）方法名称是区分大小写的。

2）最常见的的就是通常说的get和post方法。

协议版本

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法： **GET, POST 和 HEAD**方法。

HTTP1.1新增了五种请求方法：**OPTIONS, PUT, DELETE, TRACE 和 CONNECT** 方法。

# requests

## 发get请求

.response的返回内容还有其它更多信息

-- r.status\_code     #响应状态码

-- r.content           #字节方式的响应体，会自动为你解码 gzip 和 deflate 压缩

-- r.headers          #以字典对象存储服务器响应头，但是这个字典比较特殊，字典键不区分大小写，若键不存在则返回None

-- r.json()             #Requests中内置的JSON解码器

-- r.url                  # 获取url

-- r.encoding         # 编码格式

-- r.cookies           # 获取cookie

-- r.raw                #返回原始响应体

-- r.text               #字符串方式的响应体，会自动根据响应头部的字符编码进行解码

-- r.raise\_for\_status() #失败请求(非200响应)抛出异常

## 发post请求（josn）-已过期

学新模块，用help函数查看相关注释和案例内容。

|  |
| --- |
| >>import requests  >>help(requests) |

三、json

1.post的body是json类型，也可以用json参数传入。

2.先导入json模块，用dumps方法转化成json格式。

3.返回结果，传到data里

## 发post请求（data）

## Data和json

Data参数 不带大括号

Json参数

带大括号，有json项，点开看到kson解析后的参数

## https请求（SSL）

写脚本时候，我们会用抓包工具fiddler，这时候会 报：requests.exceptions.SSLError: [SSL: CERTIFICATE\_VERIFY\_FAILED] certificate verify failed (\_ssl.c:590)

小编环境：

python：2.7.12

requests：2.13.0

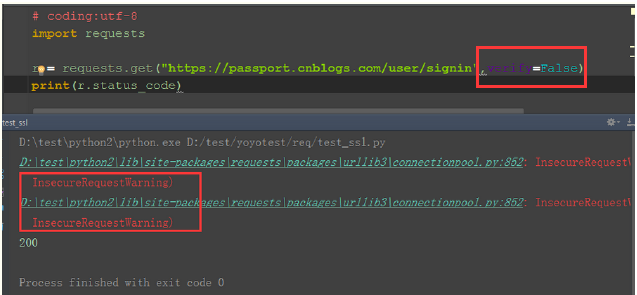
fiddler：v4.6.2.0

### verify参数设置

1.Requests的请求默认verify=True

2.如果你将 verify设置为 False，Requests 也能忽略对 SSL 证书的验证

3.但是依然会出现两行Warning,可以不用管



### 忽略Warning

|  |
| --- |
| # coding:utf-8  import requests  # 禁用安全请求警告  from requests.packages.urllib3.exceptions import InsecureRequestWarning requests.packages.urllib3.disable\_warnings(InsecureRequestWarning)  url = "https://passport.cnblogs.com/user/signin"  headers = { "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0" }  r = requests.get(url, headers=headers, verify=False)  print(r.status\_code) |

### python3的童鞋

#导入urllib3包

import urllib3

#使用这个方法就OK了

urllib3.disable\_warnings()

## session关联接口

## cookie绕过验证码登录

## json数据处理

### Encode（python->json）

首先说下为什么要encode，python里面bool值是True和False,json里面bool值是true和false,并且区分大小写，这就尴尬了，明明都是bool值。

在python里面写的代码，传到json里，肯定识别不了，所以需要把python的代码经过encode后成为json可识别的数据类型。

举个简单例子，下图中dict类型经过json.dumps（）后变成str，True变成了true,False变成了fasle。

### Encode（json -> python）

1.以第三篇的登录成功结果：{"success":true}为例，我们其实最想知道的是success这个字段返回的是True还是False

2.如果以content字节输出，返回的是一个字符串：{"success":true}，这样获取后面那个结果就不方便了

3.如果经过json解码后，返回的就是一个字典：{u'success': True}，这样获取后面那个结果，就用字典的方式去取值：result2["success"]

## 重定向

--301 redirect: 301 代表永久性转移(Permanently Moved)

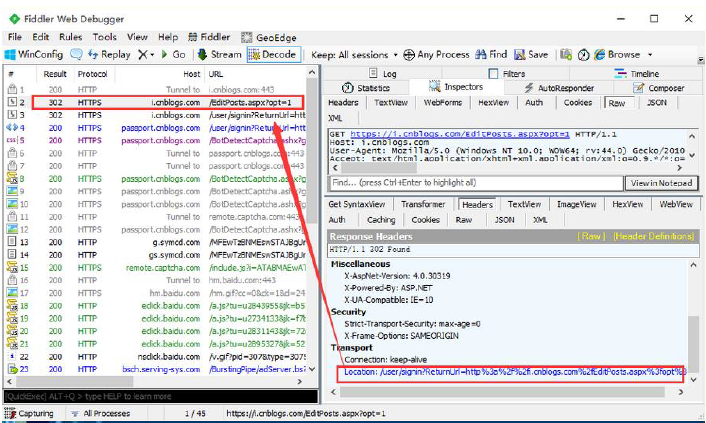
--302 redirect: 302 代表暂时性转移(Temporarily Moved )

allow\_redirects=False 禁止重定向

allow\_redirects=True 启动重定向

|  |
| --- |
| *# coding:utf-8* **import** requests *# 请求头* headers = {  **"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0"** } url = **"https://i.cnblogs.com/EditPosts.aspx?opt=1"** s = requests.session() *# 打开我的随笔* r = s.get(url, headers=headers, verify=False, allow\_redirects = False) *# 打印状态码，自动处理重定向请求* **print** r.status\_code |

获取重定向后地址



|  |
| --- |
| *# coding:utf-8* **import** requests *# 请求头* headers = {  **"User-Agent"**: **"Mozilla/5.0 (Windows NT 10.0; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0"** } url = **"https://i.cnblogs.com/EditPosts.aspx?opt=1"** s = requests.session() *# 打开我的随笔* r = s.get(url, headers=headers, verify=False, allow\_redirects = False) *# 打印状态码，自动处理重定向请求* **print** r.status\_code new\_url = r.headers[**"Location"**] **print** new\_url |

## 参数关联