



3.3 HTTP解析

3.3.1 HTTP协议

3.3.2 Python-Req

(Requests库基本使用

- <https://blog.csdn.net/rocketeerLi/article/details/86485466>)

3.3.3 实例

0.Requests库概述

Requests库主要功能:

- 1.HTTP请求
- 2.响应对象
- 3.请求头headers处理
- 4.响应码code和响应头headers处理
5. Cookie处理
- 6.重定向与追踪重定向
- 7.超时设置
- 8.代理设置



3.3.2 Python-Requests库解析HTTP

0.Requests概述

- Python中实现HTTP请求方式：
 - urllib(urllib+urllib2)：Python3内置库
 - urllib3：强大的HTTP客户端，供requests使用
 - httplib2：小型、快速的HTTP客户端
 - grequests：异步处理、并发HTTP请求
 - aiohttp：专注于异步HTTP
 - requests：优雅简单的HTTP库，为“人类”构建的

0.Requests概述

- <https://github.com/kennethreitz/requests>

3

3.3.2 Python-Requests

1.HTTP请求

- 实现HTTP请求的主要方法

- 1) Requests.get方法

- 获取HTML网页的主要方法，对应于HTTP的GET
- 构造一个向服务器请求资源的Requests对象

- Requests.get完整格式：

`Requests.get(url,params=None,*kwarg)`

- url: 获取页面的url链接
- params:url中的额外参数，字典或字节流格式，可选
- *kwarg:12个控制访问的参数，作为HTTP请求参数，可选
- 返回值：包含服务器资源的Response对象

0.概述

1.HTTP请求

2.响应对象

3.请求头headers处理

4.响应码code和响应头headers处理

5.Cookie处理

6.重定向与追踪重定向

7.超时设置

8.代理设置

■ Requests.get(url,params=None,*kwarge)

- data : 字典、字节序列或文件对象, 作为Request的内容
- json : JSON格式的数据, 作为Request的内容
- headers : 字典, HTTP定制头
- cookies : 字典或CookieJar, Request中的auth元组支持HTTP认证功能
- files : 字典类型, 传输文件
- timeout : 设定超时时间, 秒为单位
- proxies : 字典类型, 设定访问代理服务器, 可以增加登录认证
- allow_redirects : True/False, 默认为True, 重定向开关
- stream : True/False, 默认为True, 获取内容立即下载开关
- verify : True/False, 默认为True, 认证SSL证书开关
- cert : 本地SSL证书
- auth : 元组, 支持HTTP认证功能

HTTP

```
import requests
response = requests.get("http://www.baidu.com")
print(type(response))
print(response.status_code)
print(response.text)
```

响应类型

```
>>> print(type(response))
<class 'requests.models.Response'>
```

响应码

```
>>> print(response.status_code)
```

```
200
```

响应主体页面HTML

```
>>> print(response.text)
```

```
<!DOCTYPE html>
```

```
<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=http://sl.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>百度一下,你就知道</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=head_wrapper> <div class=s_form> <div class=s_form_wrapper> <div id=lg> <img hidefocus=true src=//www.baidu.com/img/bd_logo1.png width=270 height=129> </div> <form id=form name=f action=//www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1> <input type=hidden name=ie value=utf-8> <input type=hidden name=f value=8> <input type=hidden name=rev hp value=1>
```



3.3.2 Python-Requests库解析HTTP

2) Requests.post() 方法:完成HTTP的post请求

```
requests.post(url, data=None, json=None, **kwargs)
```

Parameters:

url ~ URL for the new **Request** object.

data ~ (optional)

json ~ (optional) json data to send in the body of the **Request**.

****kwargs** ~ Optional arguments that **request** takes.

- form表单形式提交数据
- json格式提交数据
- 上传文件（较少用）

Returns: **Response** object



3.3.2 Python-Requests库解析HTTP

- HTTP中的其他请求方式也可以用Requests来实现

```
·r=requests.put ('http://www.xxxxxx.com/put', data={'key':'value'})
```

```
·r=requests.delete ('http://www.xxxxxx.com/delete')
```

```
·r=requests.head ('http://www.xxxxxx.com/get')
```




3.3.2 Python-R

0.概述

1.HTTP请求

2.响应对象

3.请求头headers处理

4.响应码code和响应头headers处理

5. Cookie处理

6.重定向与追踪重定向

7.超时设置

8.代理设置

2.响应对象Response

■ Requests库的两个重要对象：

- Response对象:包含HTTP响应的内容
- Requests对象

■ Response部分属性

- r是Response类型变量

r.text ⊖ 返回响应对象

Unicode型数据，主要取文本

r.content ⊖ 返回响应对象

bytes型，二进制的数据，取图片和文件等，中文显示为字符

▶ r.json(): #Requests中内置的JSON解码器

r.status_code ⊖ 响应状态码

r.reason ⊖ 状态原因

r.cookies ⊖ 返回cookies

r.encoding ⊖ 编码方式

r.request.headers ⊖ 返回请求消息的报头


r.url ⊖ 最终的url

r.headers ⊖ 以字典对象存储服务器响应头，若键不存在则返回None

r.history

r.raw: #返回原始响应体，也就是 urllib 的 response 对象，使用 r.raw.read() 读取

r.raise_for_status(): #失败请求(非200响应)抛出异常

▶ eval函数 

3.3.2 Python-Requests库解析HTTP

- 使用response不同属性获取不同类型的响应内容
- <https://blog.csdn.net/bqw18744018044/article/details/81171220>

属性	说明
<code>r.status_code</code>	HTTP请求的返回状态，200表示连接成功，404表示失败
<code>r.text</code>	HTTP响应内容的字符串形式，即，url对应的页面内容
<code>r.encoding</code>	从HTTP header中猜测的响应内容编码方式
<code>r.apparent_encoding</code>	从内容中分析出的响应内容编码方式（备选编码方式）
<code>r.content</code>	HTTP响应内容的二进制形式

```
import requests
```

```
r = requests.get('http://www.baidu.com')
```

```
print ('content-->'+r.content)
```

```
print ('text-->'+r.text)
```

```
print ('encoding-->'+r.encoding)
```

```
r.encoding='utf-8'
```

```
print ('new text-->'+r.text)
```

范传辉书P79

```
>>> print ('content-->' + r.content)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not bytes
```

获取的页面代码无法转为二进制

```
>>> print ('text-->' + r.text)
```

print ('text-->'+r.text) 打出乱码

```
text--><!DOCTYPE html>
<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html; charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=http://sl.bdstatic.com/r/www/cache/bdorzbaidu.min.css><title>ç º ª ¯ | ä , [ ¢ ] ¼ [ ¢ ] ½ º ± c [ ¢ ] é [ ¢ ] /title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <d
```

```
>>> print ('encoding-->' + r.encoding)
```

encoding-->ISO-8859-1

```
>>> r.encoding='utf-8'
```

```
>>> print ('new text-->' + r.text)
```

```
new text--><!DOCTYPE html>
```

```
<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html; charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=http://sl.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>百度一下, 你就知道</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=header-wrapper> <div class=form> <div class=form-wrapper> <div id=lg> <img
```

置成r.encoding='utf-8'码后,
无乱码

Response对象的属性

- response不同属性检测返回状态

检测返回的Response对象的状态。

r.status_code

200

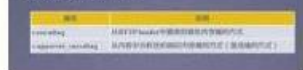
404或其他

用这些来解析返回的内容

r.text r.encoding
r.apparent_encoding
r.content

某些原因出错，将

理解Response的编码



产生异常。

```
>>> import requests
>>> r = requests.get("http://www.baidu.com")
>>> print(r.status_code)
```

200

请求状态码，200为成功

```
>>> type(r)
<class 'requests.models.Response'>
```

页面头部信息

```
>>> r.headers
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate,
ection': 'Keep-Alive', 'Transfer-Encoding': 'chunked', 'Server':
```



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置

3.3.2 Python-Requests库解析HTTP

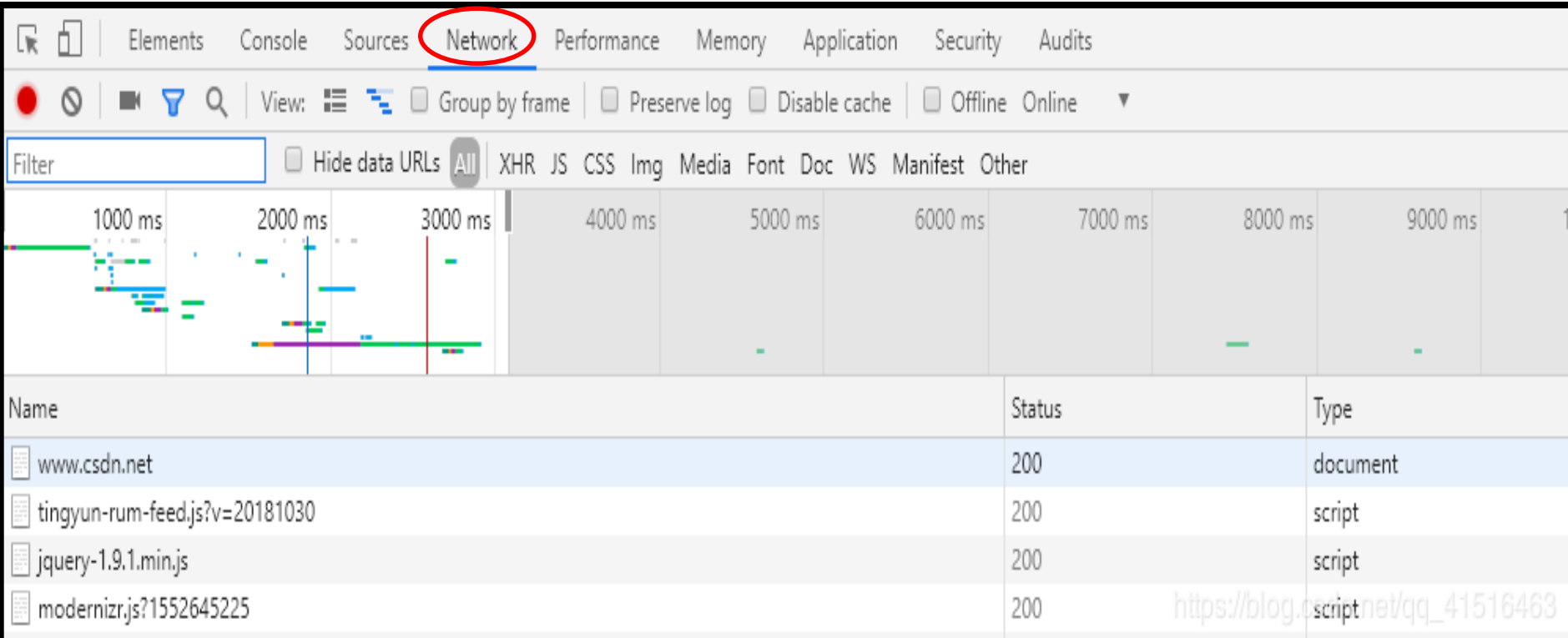
3.请求头Headers处理

<https://blog.csdn.net/ysblogs/java/article/details/88530124>

- 问题：在请求网页爬取的时候，很多网站都会对请求头做校验
 - 比如验证 User-Agent，看是不是浏览器发送的请求，使用脚本访问，默认User-Agent是python，服务器会拒绝我们的请求，禁止爬取
 - 需要通过反爬机制去解决这个问题，加上必要的请求头
- 解决：设置一些headers信息，模拟成浏览器访问网站
 - headers中有很多内容，主要常用的就是user-agent 和 host
 - 以键对的形式展现出来，如果user-agent 以字典键对形式作为headers的内容，就可以反爬成功，不需要其他键对；否则，需要加入headers下的更多键对形式。

3.3.2 Python-Requests库解析HTTP

- headers在哪里找？
- 例.使用Chrome浏览器，进入任何网页，打开开发者工具，选项中的Network选项，此时再按Ctrl+R：



The screenshot shows the Chrome DevTools Network tab. The 'Network' tab is selected and highlighted with a red circle. The 'Filter' input is empty. The 'Hide data URLs' checkbox is checked. The 'All' filter is selected. The network requests are listed in a table below the timeline.

Name	Status	Type
www.csdn.net	200	document
tingyun-rum-feed.js?v=20181030	200	script
jquery-1.9.1.min.js	200	script
modernizr.js?1552645225	200	script

- 选择左下方框中的目标，选
会有Request Headers，包

Name

- www.sohu.com
- lczm?conwid=850&conhei=1.
- lczm?conwid=850&conhei=1.
- lczm?conwid=300&conhei=2.
- lczm?conwid=300&conhei=1.
- index.html
- fsy3_tb.html
- fsy4_tb.html
- juxinggengxin.html
- 1180-100tb.html
- fstg2_tb.html
- hcomm?conwid=300&conhei=
- o.htm?ltr=
- o.htm?ltr=
- creation-3354V76q1GX2xO01.
- creation-3354V76q1GX2xO01.

20 / 564 requests | 119 KB / 4.7 M

Headers

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

accept-encoding: gzip, deflate, br

accept-language: zh-CN,zh;q=0.9

cache-control: max-age=0

cookie: gidinf=x099980109ee11608281b305700091656bec741c7709; IPLOC=CN1100; SUV=200420155653MECA; t=1587708505852; reqtype=pc; ad_t_4=1; ad_t_3=2; ad_t_2=2; ad_t_5=4; ad_t_6=4; beans_new_turn=%7B%22sohu-index%22%3A50%7D

sec-fetch-dest: document

sec-fetch-mode: navigate

sec-fetch-site: none

sec-fetch-user: ?1

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36

Request Headers

:authority: www.sohu.com

:method: GET

:path: /

:scheme: https

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

accept-encoding: gzip, deflate, br

accept-language: zh-CN,zh;q=0.9

cache-control: max-age=0

cookie: gidinf=x099980109ee11608281b305700091656bec741c7709; IPLOC=CN1100; SUV=200420155653MECA; t=1587708505852; reqtype=pc; ad_t_4=1; ad_t_3=2; ad_t_2=2; ad_t_5=4; ad_t_6=4; beans_new_turn=%7B%22sohu-index%22%3A50%7D

sec-fetch-dest: document

sec-fetch-mode: navigate

sec-fetch-site: none

sec-fetch-user: ?1

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.100 Safari/537.36

- 用查到的Header数据生成程序数据

```
import requests
user_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
headers={'User-Agent':user_agent}
r = requests.get('http://www.baidu.com',headers= headers)
print (r.content)
```

```
>>> print (r.content)
b'<!DOCTYPE html><!--STATUS OK--><html><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><meta content="always" name="referrer"><meta name="theme-color" content="#2932e1"><link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" /><link rel="search" type="application/opensearchdescription+xml" href="/content-search.xml" title="\xe7\x99\xbe\xe5\xb\xa\xa6\xe6\x90\x9c\xe7\xb4\xa2" /><link rel="icon" sizes="any" mask href="//www.baidu.com/img/baidu_85beaf5496f291521eb75ba38eacbd87.svg"><link rel="dns-prefetch" href="//dss0.bdstatic.com"/><link rel="dns-prefetch" href="//dssl.bdstatic.com"/><link rel="dns-prefetch" href="//cssl.bdstatic.com"/><link rel="dns-prefetch" href="//sp0.baidu.com"/><link rel="dns-prefetch" href="//sp1.baidu.com"/><link rel="dns-prefetch" href="//sp2.baidu.com"/><link rel="dns-prefetch" href="//sp3.baidu.com"/></head><body><div class="wrapper"><div class="header"><div class="logo"><img alt="Baidu Logo" data-bbox="115 115 185 185"/></div><div class="input"><input type="text" value="" /></div><div class="button"><button type="submit" value="搜索" /></div></div><div class="main"><div class="result"><div class="title"><a href="#"></a></div><div class="url"><a href="#"></a></div><div class="desc"><p></p></div></div></div></div></body></html>
```

```
import requests

base_url = 'http://httpbin.org'

form_data = {"user": "zou", "pwd": '31500'}
form_header = {"User-Agent": "Chrome/68.0.3440.106"} # 设置请求头, 字典格式
r = requests.post(base_url + '/post', data=form_data, headers=form_header)
print(r.url) # 打印URL
print(r.status_code)
print(r.text)
```

运行结果-续

```
http://httpbin.org/post
200
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "pwd": "31500",
    "user": "zou"
  },
```

运行结果

```
"headers": {
  "Accept": "*/*",
  "Accept-Encoding": "gzip, deflate",
  "Content-Length": "18",
  "Content-Type": "application/x-www-form-urlencoded",
  "Host": "httpbin.org",
  "User-Agent": "Chrome/68.0.3440.106"
},
"json": null,
"origin": "112.10.81.210, 112.10.81.210",
"url": "https://httpbin.org/post"
}
```



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置

3.3.2 Python-Re

HTTP头部大小写不敏感:

```
r.headers['Content-Type']
```

```
r.headers.get('content-type')
```

4. 响应码status_code和响应头headers处理

- 获取响应码: Requests中的status_code字段
- 获取响应头: Requests中的headers字段

```
import requests
r = requests.get('http://www.baidu.com')
if r.status_code == requests.codes.ok:
    print(r.status_code) # 响应码
    print(r.headers)    # 响应头
    print(r.headers.get('content-type'))
    获取其中的某个字段
    print(r.headers['content-type']) # 不推荐仅用这种方式
else:
    r.raise_for_status()
```

包含所有的响应头信息

通过get函数获取其中的某一个字段

通过字典引用的方式获取字段值, 不推荐! 如果字段中没有这个字段, 会抛出异常

当响应码是4XX或5XX时, 主动抛出异常
当响应码为200时, 返回None

3.3.2 Python-Requests库解析

```
>>> print(r.status_code)    # 响应码
200
>>> print(r.headers)        # 响应头
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate, no-trans
form', 'Connection': 'keep-alive', 'Content-Encoding': 'gzip', 'Content-Ty
pe': 'text/html', 'Date': 'Mon, 04 May 2020 01:48:44 GMT', 'Last-Modified'
: 'Mon, 23 Jan 2017 13:27:29 GMT', 'Pragma': 'no-cache', 'Server': 'bfe/1.
0.8.18', 'Set-Cookie': 'BDORZ=27315; max-age=86400; domain=.baidu.com; pat
h=/', 'Transfer-Encoding': 'chunked'}
>>> print(r.headers.get('content-type'))    # 推荐使用这种获取方式, 获取其
中的某个字段
text/html
>>> print(r.headers['content-type'])        # 不推荐使用这种获取方式
text/html
>>> print(r.raise_for_status())
None
```



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置



3.3.2 Python-Requests库解析HTTP

5. Cookie处理

- 获取 cookies:cookies 携带着访问者的一些信息。

```
import requests
# 获取 cookie
response = requests.get("https://www.baidu.com")
print(response.cookies)    # cookies 列表形式
for key, value in response.cookies.items() :
    print(key + '=' + value)
```

```
<RequestsCookieJar[<Cookie BDORZ=27315 for .baidu.com/>]>
```

```
>>> for key, value in response.cookies.items():
...     print(key + '=' + value)
...
BDORZ=27315
```



3.3.2 Python-Requests库解析HTTP

■ 自定义Cookie值发送出去

```
import requests
user_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
headers={'User-Agent':user_agent}
cookies = dict(name='qiye',age='10')
r = requests.get('http://www.baidu.com',
                 headers=headers,cookies=cookies)
print(r.text)
```

3.3.2 Python-Requests库解析HTTP

■ 会话维持

- **cookies** 一般是用来做会话维持的，连接两次请求
- 没有会话维持的代码：

```
import requests
# 两次 get 请求， 没有任何关联，不可以
requests.get("http://httpbin.org/cookies/set/number/123456789")
response = requests.get("http://httpbin.org/cookies")
print(response.text)
```

```
{
  "cookies" : {}
}
```

可以看到，两次访问，没有存储cookies

3.3.2 Python-Requests库解析HTTP

- 利用 Session 对象，保存cookies值(<http://httpbin.org/>):
 - session具有保持功能，session可以跨请求保持cookies

```
import requests
# Session 对象， 相当于在一个浏览器中先后访问（例如：登录验证）
s = requests.Session()
s.get("http://httpbin.org/cookies/set/number/123456789")
response = s.get("http://httpbin.org/cookies")
print(response.text)
```

```
{
  "cookies" : {
    "number" : "123456789"
  }
}
```

- 可以看到，cookies 的值被保存下来了



3.3.2 Python-Requests库解析HTTP

- 例，跨请求保持cookies，在命令行上输入下面命令：

```
# 创建一个session对象
s = requests.Session()
# 用session对象发出get请求，设置cookies
s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
# 用session对象发出另外一个get请求，获取cookies
r = s.get("http://httpbin.org/cookies")
# 显示结果
r.text
'{"cookies": {"sessioncookie": "123456789"}}'
```



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置



3.3.2 Python-Requests库解析HTTP

6.重定向与追踪重定向

<https://www.cnblogs.com/xswt/p/11475164.html>

- 重定向就是网络请求被重新定个方向，转到其它位置
- 网页重定向的情况一般有：
 - 网站调整（如网页目录结构变化）
 - 网页地址改变
 - 网页扩展名（.php、.html、.asp）的改变
 - 当一个网站注册了多个域名的时候
- 这些情况下不做重定向的话就容易出现404错误（如访问网上提供的网页url经常报404错误，就是有可能url地址改变了但没有做重定向导致的。）

3.3.2 Python-Requests库解析HTTP

■ 处理重定向方法

- 设置get方法的属性allow_redirects
 - allow_redirects设置为True，允许重定向；
 - allow_redirects设置为False，禁止重定向。

■ 例

```
r = requests.get('http://github.com', allow_redirects=False)
print(r.status_code)
print(r.history)
```

301

[]

301: 永久性重定向，页面永久性移走

302: 暂时性重定向


```
'''
```

重启重定向

```
'''
```

```
import requests
```

```
url = 'http://home.cnblogs.com/u/xswt/'
```

```
r = requests.get(url, headers={"Content-Type": "application/json"}, allow_redirects=True)
```

```
print(r.status_code)
```

```
print(r.text)
```

运行结果



200

```
<!DOCTYPE html><html lang=zh><head><meta charset=utf-8><meta http-equiv=X-UA-Compatible  
content="IE=EDGE"><meta name=viewport content="width=device-width, initial-scale=1, shrink-to-  
fit=no"><title>用户登录 - 博客园</title><link rel="shortcut icon"  
href="//common.cnblogs.com/favicon.ico type=image/x-icon><script src="/assets/account/signin-  
iconfont.js?v=010krFmCBcVIQNTQ6W3Q8sMKdVgWbmPjCL6jUR8-WG0"></script><link rel=stylesheet
```



3.3.2 Python-Requests库解析HTTP

■ 追踪重定向

- 使用Response.history
- 结果对象列表按照从最老到最近的请求进行排序。

```
import requests  
url = 'http://home.cnblogs.com/u/xswt/'  
r = requests.get(url, params=None, headers={'Content-Type': 'application/json'})  
print(r.history) #history追踪页面重定向历史
```

```
[<Response [301]>, <Response [302]>, <Response [302]>, <Response [302]>]  
#可以看到该请求做了多次重定向
```

3.3.2 Python-Requests库解析HTTP

■ 获取重定向的url地址:

```
import requests
url = 'http://home.cnblogs.com/u/xswt/'
r = requests.get(url, headers={"Content-Type": "application/json"})
reditList = r.history#可以看出获取的是一个地址序列
print(f'获取重定向的历史记录: {reditList}')
print(f'获取第一次重定向的headers头部信息: {reditList[0].headers}')
print(f'获取重定向最终的url: {reditList[len(reditList)-1].headers["location"]}')

```

获取重定向的历史记录: [<Response [301]>, <Response [302]>, <Response [302]>, <Response [302]>]

获取第一次重定向的headers头部信息: {'Date': 'Fri, 06 Sep 2019 06:53:05 GMT', 'Content-Length': '0', 'Connection': 'keep-alive', 'Location': 'https://home.cnblogs.com/u/xswt/'}

获取重定向最终的url: https://account.cnblogs.com/signin?

returnUrl=http%3a%2f%2fhome.cnblogs.com%2fu%2fxswt%2f



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置



3.3.2 Python-Requests库解析HTTP

7. 超时设置

- 超时是请求能够容忍的最大时间，如果在这个时间内，还没有响应返回过来，那么这次请求失败，不再继续等待请求的结果
- 使用requests 的timeout 参数设定的秒数时间之后停止等待响应
- timeout 仅对连接过程有效，与响应体的下载无关。

```
# 超时设置
from requests.exceptions import ConnectTimeout
response = requests.get("https://httpbin.org/get", timeout = 1)
print(response.status_code)
try :
    response = requests.get("https://httpbin.org/get", timeout = 0.1)
except ConnectTimeout:
    print("ConnectTimeout")
```



3.3.2 Python-Requests库解析HTTP

- Requests库基本使用
 - 0.概述
 - 1.HTTP请求
 - 2.响应对象
 - 3.请求头headers处理
 - 4.响应码code和响应头headers处理
 - 5. Cookie处理
 - 6.重定向与追踪重定向
 - 7.超时设置
 - 8.代理设置

■ 8.代理设置

- 代理也是一个很常用的方法。通常，在我们需要多个主机进行访问或需要绕过防火墙时，可以利用代理进行访问

```
# 无密码
proxies = {
    "http": "http://178.128.63.64:8388"
}

# 有密码
proxies = {
    "http": "http://user:password@178.128.63.64:8388"
}

response = requests.get("https://www.taobao.com", proxies=proxies)
print(response.status_code)

# 利用 socks 进行代理设置
proxies = {
    "http": "socks5://178.128.63.64:8388",
    "https": "socks5://178.128.63.64:8388"
}

response = requests.get("https://www.taobao.com", proxies=proxies)
print(response.status_code)
```