**第一章 Python 基础模块讲解**

**本节所讲内容：**

* 1. **Python网路请求详细讲解**

**1.2 Python urllib网络请求模块讲解**

**1.3 Python requests 请求模块讲解**

**实战：利用urllib爬取百度贴吧**

**实战：爬取内涵吧脑筋急转弯**

**实战：爬取猫眼电影排行榜并使用csv保存到本地的csv中。**

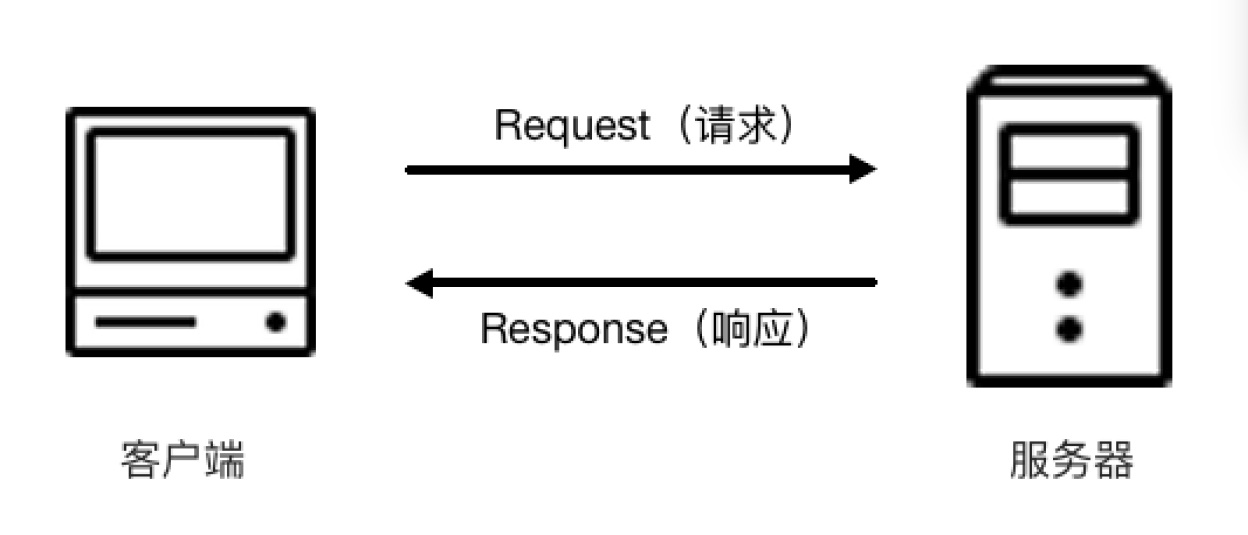
**1.1 Python网路请求详细讲解**

**1.1.1 HTTP和HTTPS**

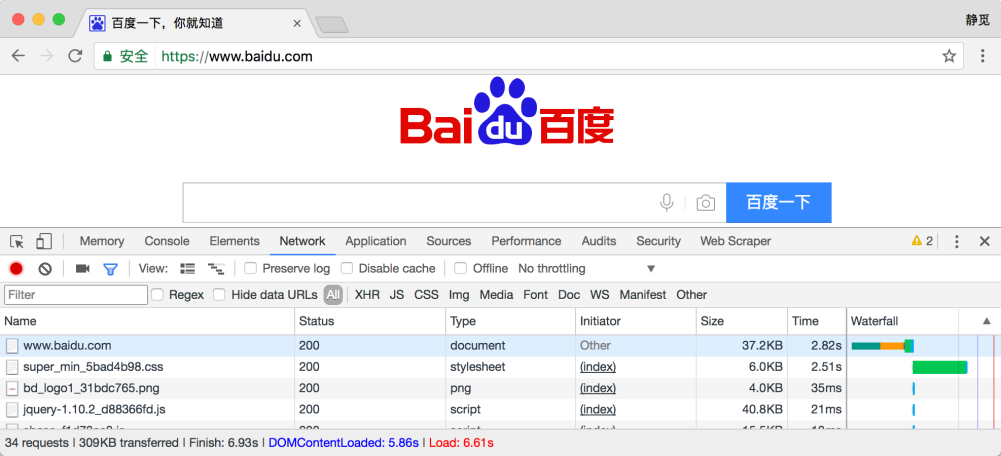
**HTTP的全称是Hyper Text Transfer Protocol，中文名叫作超文本传输协议。HTTP协议是用于从网络传输超文本数据到本地浏览器的传送协议，它能保证高效而准确地传送超文本文档。HTTPS的全称是Hyper Text Transfer Protocol over Secure Socket Layer，是以安全为目标的HTTP通道，简单讲是HTTP的安全版，即HTTP下加入SSL层，简称为HTTPS。**

**1.1.2 HTTP请求过程**

**我们在浏览器中输入一个URL，回车之后便会在浏览器中观察到页面内容。**

****

**可以看到，在Network页面下方出现了一个个的条目，其中一个条目就代表一次发送请求和接收响应的过程，如图所示。**

****

**我们先观察第一个网络请求，即www.baidu.com。**

**其中各列的含义如下。**

**第一列Name：请求的名称，一般会将URL的最后一部分内容当作名称。**

**第二列Status：响应的状态码，这里显示为200，代表响应是正常的。通过状态码，我们可以判断发送了请求之后是否得到了正常的响应。**

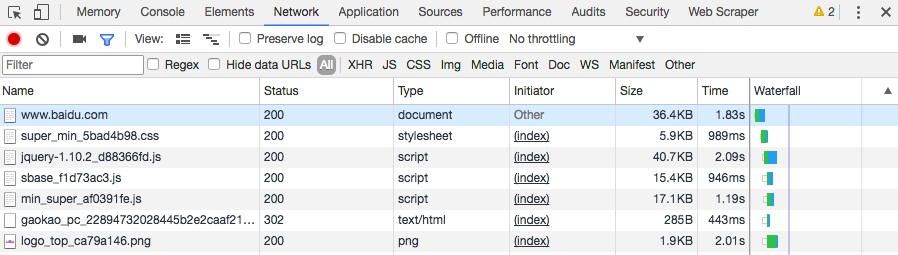
**第三列Type：请求的文档类型。这里为document，代表我们这次请求的是一个HTML文档，内容就是一些HTML代码。**

**第四列Initiator：请求源。用来标记请求是由哪个对象或进程发起的。**

**第五列Size：从服务器下载的文件和请求的资源大小。如果是从缓存中取得的资源，则该列会显示from cache。**

**第六列Time：发起请求到获取响应所用的总时间。**

**第七列Waterfall：网络请求的可视化瀑布流。**

****

**1.1.3 请求**

**请求，由客户端向服务端发出，可以分为4部分内容：请求方法（Request Method）、请求的网址（Request URL）、请求头（Request Headers）、请求体（Request Body）。**

**1.请求方法**

**常见的请求方法有两种：GET和POST。**

**在浏览器中直接输入URL并回车，这便发起了一个GET请求，请求的参数会直接包含到URL里。例如，在百度中搜索Python，这就是一个GET请求，链接为**[**https://www.baidu.com/s?wd=Python**](https://www.baidu.com/s?wd=Python)**，其中URL中包含了请求的参数信息，这里参数wd表示要搜寻的关键字。POST请求大多在表单提交时发起。比如，对于一个登录表单，输入用户名和密码后，点击“登录”按钮，这通常会发起一个POST请求，其数据通常以表单的形式传输，而不会体现在URL中。**

**GET和POST请求方法有如下区别。**

**GET请求中的参数包含在URL里面，数据可以在URL中看到，而POST请求的URL不会包含这些数据，数据都是通过表单形式传输的，会包含在请求体中。**

**GET请求提交的数据最多只有1024字节，而POST方式没有限制。**

**一般来说，登录时，需要提交用户名和密码，其中包含了敏感信息，使用GET方式请求的话，密码就会暴露在URL里面，造成密码泄露，所以这里最好以POST方式发送。上传文件时，由于文件内容比较大，也会选用POST方式。**

**我们平常遇到的绝大部分请求都是GET或POST请求，另外还有一些请求方法，如GET、HEAD、POST、PUT、DELETE、OPTIONS、CONNECT、TRACE等。**

| **方法** | **描述** |
| --- | --- |
| **GET** | **请求页面，并返回页面内容** |
| **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** |
| **POST** | **大多用于提交表单或上传文件，数据包含在请求体中** |
| **PUT** | **从客户端向服务器传送的数据取代指定文档中的内容** |
| **DELETE** | **请求服务器删除指定的页面** |
| **CONNECT** | **把服务器当作跳板，让服务器代替客户端访问其他网页** |
| **OPTIONS** | **允许客户端查看服务器的性能** |
| **TRACE** | **回显服务器收到的请求，主要用于测试或诊断** |

**2.请求的网址**

**请求的网址，即统一资源定位符URL，它可以唯一确定我们想请求的资源。**

**3.请求头**

**请求头，用来说明服务器要使用的附加信息，比较重要的信息有Cookie、Referer、User-Agent等。下面简要说明一些常用的头信息。**

**Accept：请求报头域，用于指定客户端可接受哪些类型的信息。**

**Accept-Language：指定客户端可接受的语言类型。**

**Accept-Encoding：指定客户端可接受的内容编码。**

**Host：用于指定请求资源的主机IP和端口号，其内容为请求URL的原始服务器或网关的位置。从HTTP 1.1版本开始，请求必须包含此内容。**

**Cookie：也常用复数形式 Cookies，这是网站为了辨别用户进行会话跟踪而存储在用户本地的数据。它的主要功能是维持当前访问会话。例如，我们输入用户名和密码成功登录某个网站后，服务器会用会话保存登录状态信息，后面我们每次刷新或请求该站点的其他页面时，会发现都是登录状态，这就是Cookies的功劳。Cookies里有信息标识了我们所对应的服务器的会话，每次浏览器在请求该站点的页面时，都会在请求头中加上Cookies并将其发送给服务器，服务器通过Cookies识别出是我们自己，并且查出当前状态是登录状态，所以返回结果就是登录之后才能看到的网页内容。**

**Referer：此内容用来标识这个请求是从哪个页面发过来的，服务器可以拿到这一信息并做相应的处理，如作来源统计、防盗链处理等。**

**User-Agent：简称UA，它是一个特殊的字符串头，可以使服务器识别客户使用的操作系统及版本、浏览器及版本等信息。在做爬虫时加上此信息，可以伪装为浏览器；如果不加，很可能会被识别出为爬虫。**

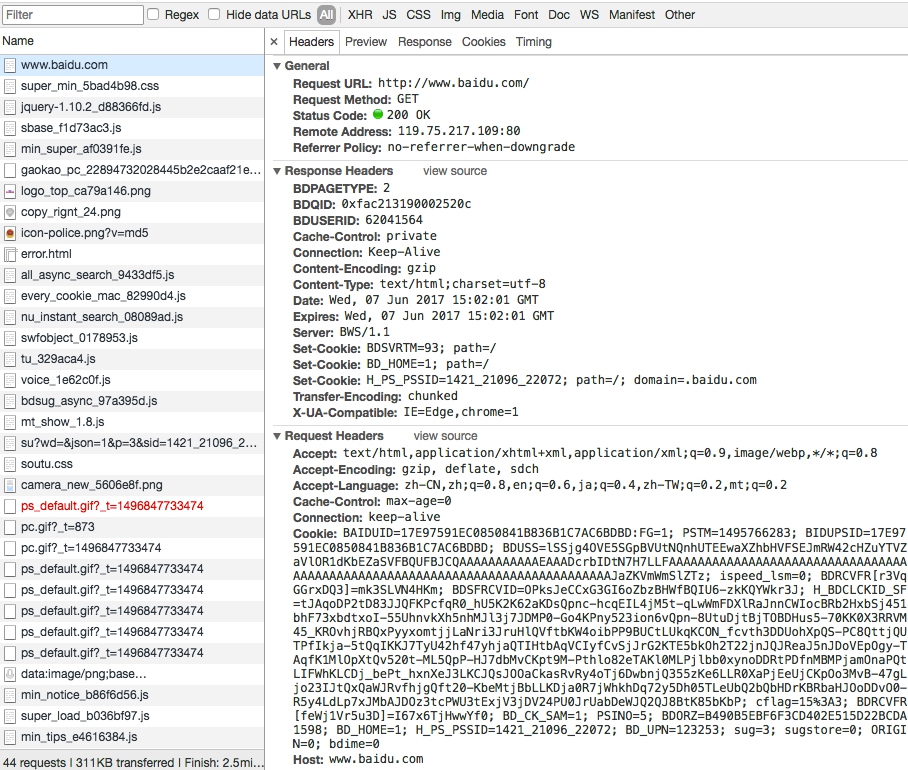
**Content-Type：也叫互联网媒体类型（Internet Media Type）或者MIME类型，在HTTP协议消息头中，它用来表示具体请求中的媒体类型信息。例如，text/html代表HTML格式，image/gif代表GIF图片，application/json代表JSON类型，更多对应关系可以查看此对照表：**[**http://tool.oschina.net/commons**](http://tool.oschina.net/commons)**。**

**因此，请求头是请求的重要组成部分，在写爬虫时，大部分情况下都需要设定请求头。**

**4.请求体**

**请求体一般承载的内容是POST请求中的表单数据，而对于GET请求，请求体则为空。**

**例如，这里我登录GitHub时捕获到的请求和响应如图所示。**

****

**登录之前，我们填写了用户名和密码信息，提交时这些内容就会以表单数据的形式提交给服务器，此时需要注意Request Headers中指定Content-Type为application/x-www-form-urlencoded。只有设置Content-Type为application/x-www-form-urlencoded，才会以表单数据的形式提交。另外，我们也可以将Content-Type设置为application/json来提交JSON数据，或者设置为multipart/form-data来上传文件。**

| **Content-Type** | **提交数据的方式** |
| --- | --- |
| **application/x-www-form-urlencoded** | **表单数据** |
| **multipart/form-data** | **表单文件上传** |
| **application/json** | **序列化JSON数据** |
| **text/xml** | **XML数据** |

**在爬虫中，如果要构造POST请求，需要使用正确的Content-Type，并了解各种请求库的各个参数设置时使用的是哪种Content-Type，不然可能会导致POST提交后无法正常响应。**

**1.1.4. 响应**

**响应，由服务端返回给客户端，可以分为三部分：响应状态码（Response Status Code）、响应头（Response Headers）和响应体（Response Body）。**

**1.响应状态码**

**响应状态码表示服务器的响应状态，如200代表服务器正常响应，404代表页面未找到，500代表服务器内部发生错误。在爬虫中，我们可以根据状态码来判断服务器响应状态，如状态码为200，则证明成功返回数据，再进行进一步的处理，否则直接忽略。响应头**

**响应头包含了服务器对请求的应答信息，如Content-Type、Server、Set-Cookie等。下面简要说明一些常用的头信息。**

**Date：标识响应产生的时间。**

**Last-Modified：指定资源的最后修改时间。**

**Content-Encoding：指定响应内容的编码。**

**Server：包含服务器的信息，比如名称、版本号等。**

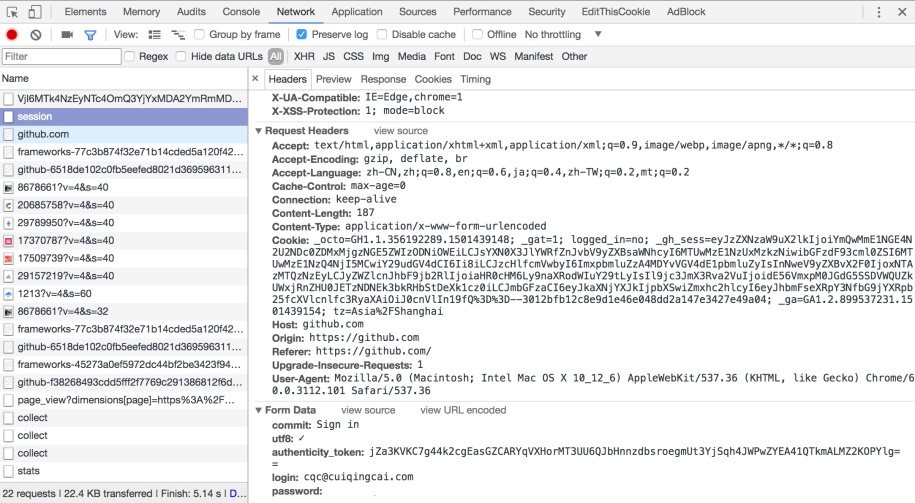
**Content-Type：文档类型，指定返回的数据类型是什么，如text/html代表返回HTML文档，application/x-javascript则代表返回JavaScript文件，image/jpeg则代表返回图片。**

**Set-Cookie：设置Cookies。响应头中的Set-Cookie告诉浏览器需要将此内容放在Cookies中，下次请求携带Cookies请求。**

**Expires：指定响应的过期时间，可以使代理服务器或浏览器将加载的内容更新到缓存中。如果再次访问时，就可以直接从缓存中加载，降低服务器负载，缩短加载时间。**

**2.响应体**

**最重要的当属响应体的内容了。响应的正文数据都在响应体中，比如请求网页时，它的响应体就是网页的HTML代码；请求一张图片时，它的响应体就是图片的二进制数据。我们做爬虫请求网页后，要解析的内容就是响应体，如图所示。**

**1.2 Python urllib网络请求模块讲解**

**urllib.request模块提供了最基本的构造HTTP请求的方法，利用它可以模拟浏览器的一个请求发起过程，同时它还带有处理授权验证（authenticaton）、重定向（redirection)、浏览器Cookies以及其他内容。这个模块也可以做网络请求，但是我们主要目的是学习python中的requests。**

**1.2.1 urlopen方法**

**urllib.request.urlopen()函数用于实现对目标url的访问。**

**函数原型如下：urllib.request.urlopen(url, data=None, [timeout, ]\*, cafile=None, capath=None, cadefault=False, context=None)**

**url:  需要打开的网址**

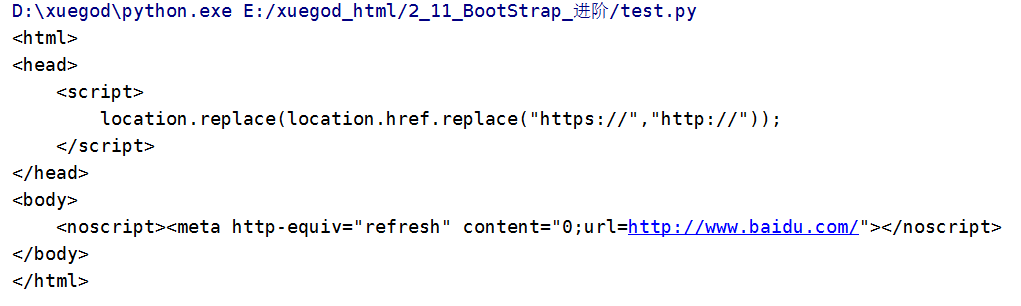
**data：Post提交的数据（bytes类型，则需要通过bytes()方法转化。另外，如果传递了这个参数，则它的请求方式就不再是GET方式，而是POST方式）**

**timeout：设置网站的访问超时时间**

**示例如下：**

**import urllib.request  
*#请求服务器*response = urllib.request.urlopen('https://www.baidu.com')  
*#调用read方法读取响应的数据*print(response.read().decode())  
*#获取响应状态吗*print(response.status)  
*#获取响应头部信息*print(response.getheaders())  
*#获取某个头部信息*print(response.getheader('Server'))**

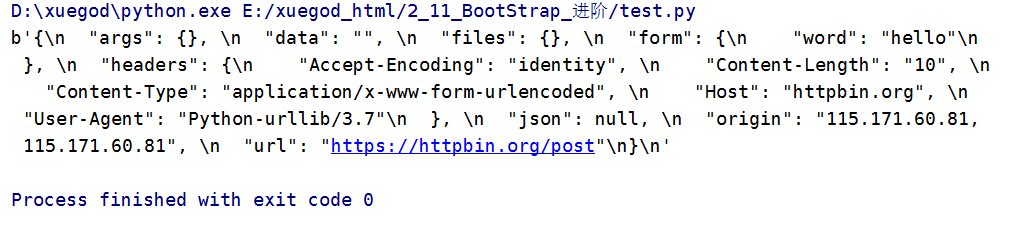
**结果如下：**



**发送带有data的请求**

**import urllib.parse  
import urllib.request  
*#利用parse方法解析数据，进行数据编码，并转换为字节的形式*data = bytes(urllib.parse.urlencode({'word': 'hello'}), encoding='utf8')  
*#发送到固定的网址，进行请求*response = urllib.request.urlopen('http://httpbin.org/post', data=data)  
print(response.read())**

**运行结果如下：**

**1.2.2 Request 方法**

**我们知道利用urlopen()方法可以实现最基本请求的发起，但这几个简单的参数并不足以构建一个完整的请求。如果请求中需要加入Headers等信息，就可以利用更强大的Request类来构建。所以现在我们引用了Request方法。实在是模拟请求，抓取数据的不二之选。**

**class Request:  
 def \_\_init\_\_(self, url, data=None, headers={},  
 origin\_req\_host=None, unverifiable=False,  
 method=None):**

**第一个参数url用于请求URL，这是必传参数，其他都是可选参数。**

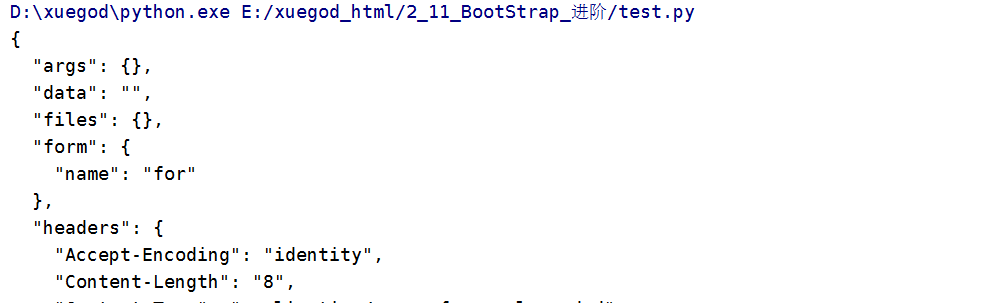
**第二个参数data如果要传，必须传bytes（字节流）类型的。如果它是字典，可以先用urllib.parse模块里的urlencode()编码。**

**第三个参数headers是一个字典，它就是请求头，我们可以在构造请求时通过headers参数直接构造，也可以通过调用请求实例的add\_header()方法添加。**

**实例如下：**

**from urllib import request, parse  
from fake\_useragent import UserAgent  
url = 'http://httpbin.org/post'  
headers = {  
 'User-Agent': UserAgent().random,  
}  
dict = {  
 'name': 'for'  
}  
*#把数据变成字节的形式*data = bytes(parse.urlencode(dict), encoding='utf8')  
*#进行网络请求*req = request.Request(url=url, data=data, headers=headers, method='POST')  
response = request.urlopen(req)  
print(response.read().decode('utf-8'))**

**效果如下：**



**实战：封装函数实现百度贴吧的下载**

**import urllib.request  
import urllib.parse  
*# 读取页面*def readPage(url):  
 headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"}  
 req = urllib.request.Request(url,headers=headers)  
 res = urllib.request.urlopen(req)  
 html = res.read().decode("utf-8")  
 return html  
*# 写入文件*def writePage(filename,html):  
 with open(filename,"w",encoding="utf-8") as f:  
 f.write(html)  
 print("写入成功")  
  
*# 主函数*def workOn():  
 name = input("请输入贴吧名:")  
 begin = int(input("请输入起始页:"))  
 end = int(input("请输入终止页:"))  
 *# 对贴吧名name进行编码* kw = {"kw":name}  
 kw = urllib.parse.urlencode(kw)  
 *# print(kw)  
 # 拼接URL,发请求,获响应* for i in range(begin,end+1):  
 *# 拼接URL* pn = (i-1)\*50  
 baseurl = "http://tieba.baidu.com/f?"  
 url = baseurl + kw + "&pn=" + str(pn)  
 html = readPage(url)  
 filename = "第" + str(i) + "页.html"  
 writePage(filename,html)  
if \_\_name\_\_ == "\_\_main\_\_":  
 workOn()**

**运行结果如下;**



* 1. **Python requests 请求模块讲解**

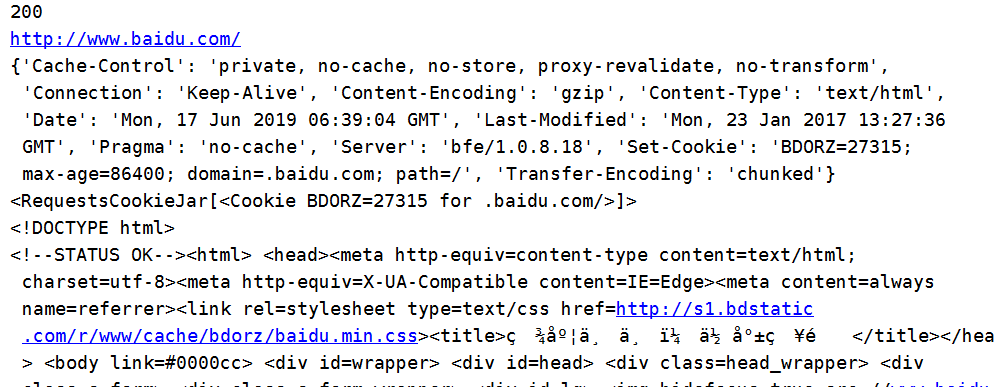
**上面一小节中我们已经了解了urllib这个模块可以实现功能，但是在构造整体代码上还是有所复杂，下面让我们学习今天的主角-requests。**

**requests是python实现的简单易用的HTTP库，使用起来比urllib简洁很多。**

**Requests 的安装：pip install requests**

**1.3.1 requests的基本用法**

**import requests  
response = requests.get('http://www.baidu.com')  
print(response.status\_code) *# 打印状态码*print(response.url) *# 打印请求url*print(response.headers) *# 打印头信息*print(response.cookies) *# 打印cookie信息*print(response.text) *#以文本形式打印网页源码*print(response.content) *#以字节流形式打印***

**运行结果如下：**

**1.3.2 requests的请求方式**

**Requests的请求方式和urllib的请求是一样的，具体方法请看上节内容，代码示例如下：**

**import requests  
requests.get('http://httpbin.org/get')  
requests.post('http://httpbin.org/post')  
requests.put('http://httpbin.org/put')  
requests.delete('http://httpbin.org/delete')  
requests.head('http://httpbin.org/get')  
requests.options('http://httpbin.org/get')**

**1.3.3 带参数的GET请求：**

**第一种直接将参数放在url内**

**import requests  
response = requests.get('http://httpbin.org/get?name=gemey&age=22')  
print(response.text)**

**运行结果如下：**

****

**另一种先将参数填写在dict中，发起请求时params参数指定为dict**

**import requests  
data = {  
 'name': 'for',  
 'age': 20  
}  
response = requests.get('http://httpbin.org/get', params=data)  
print(response.text)**

**1.3.4 解析json**

**import requests  
response = requests.get('http://httpbin.org/get')  
print(response.text)  
print(response.json()) *#response.json()方法同json.loads(response.text)*print(type(response.json()))**

**运行结果如下：**

****

**1.3.5简单保存一个二进制文件，二进制内容为response.content（字节的形式）**

**import requests  
response = requests.get('http://img.ivsky.com/img/tupian/pre/201708/30/kekeersitao-002.jpg')  
b = response.content  
with open('fengjing.jpg','wb') as f:  
 f.write(b)**

**运行结果如下：**

****

**1.3.6为你的请求添加头信息**

**import requests  
*# headers = {}  
# headers['User-Agent'] = 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10\_6\_8; en-us) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50'*headers = {  
 'User-Agent':'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10\_6\_8; en-us) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50'  
}  
response = requests.get('http://www.baidu.com',headers=headers)**

**1.3.7使用代理**

**同添加headers方法，代理参数也要是一个dict**

**这里使用requests库爬取了IP代理网站的IP与端口和类型**

**因为是免费的，使用的代理地址很快就失效了。**

**import requests  
from fake\_useragent import UserAgent  
proxies = {  
 'http':'119.57.105.25:53281',  
}  
headers ={  
 'user\_agent':UserAgent().random  
}  
response = requests.get(url='http://www.baidu.com',headers=headers,proxies=proxies,verify=False)  
print(response.text)**

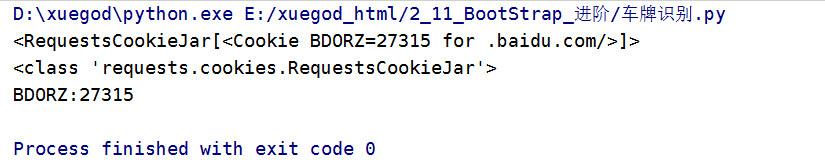
**1.3.8基本POST请求：**

**import requests  
data = {'name':'tom','age':'22'}  
response = requests.post('http://httpbin.org/post', data=data)**

**获取cookie**

***#获取cookie*import requests  
response = requests.get('http://www.baidu.com')  
print(response.cookies)  
print(type(response.cookies))  
for k,v in response.cookies.items():  
 print(k+':'+v)**

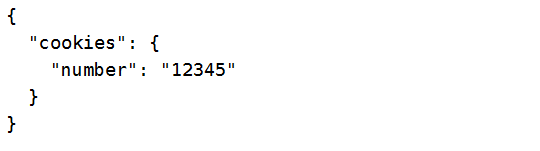
**运行结果如下：**

****

**1.3.9会话维持**

**import requests  
session = requests.Session()  
session.get('http://httpbin.org/cookies/set/number/12345')  
response = session.get('http://httpbin.org/cookies')  
print(response.text)**

**结果：**



**1.3.10证书验证设置**

**import requests  
from requests.packages import urllib3  
urllib3.disable\_warnings() *#从urllib3中消除警告*response = requests.get('https://www.12306.cn',verify=False) *#证书验证设为FALSE*print(response.status\_code)**

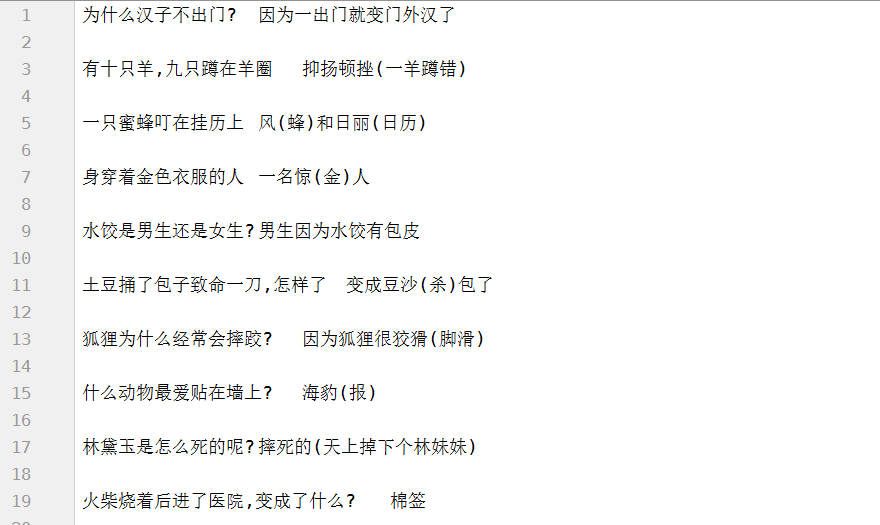
**运行结果如下：**

**200**

**实战：爬取内涵吧脑筋急转弯**

**实战：爬取猫眼电影排行榜并使用csv保存到本地的csv中。****import requests  
import re  
from fake\_useragent import UserAgent  
*# https://www.neihan-8.com/njjzw/index.html*base\_url = 'https://www.neihan8.com/njjzw/'  
  
headers = {  
 "User-Agent":UserAgent().random  
}  
def load\_page(url):  
 try:  
 req = requests.get(url,headers=headers)  
 if req.status\_code == 200:  
 *# print(req.text.decode())* return req.content.decode()  
 except:  
 return None  
def parse\_page(html):  
 data\_list = re.findall(r'<div class="text-.\*?title="(.\*?)".\*?<div class="desc">(.\*?)</div>',html,re.S)  
 return data\_list  
def write\_page(data\_list):  
 with open('33333.txt','a',encoding='utf-8')as f:  
 for data in data\_list:  
 f.write(data[0].strip()+'\t'+data[1].strip()+'\n')  
 *# f.write(data[1].strip()+'\n')* f.write('\n')  
if \_\_name\_\_ == '\_\_main\_\_':  
 for i in range(1,10):  
 if i == 1:  
 url = base\_url  
 else:  
 url = base\_url+'index\_'+str(i)+'.html'  
 print(url)  
 html = load\_page(url)  
 data\_list = parse\_page(html)  
 write\_page(data\_list)**

**运行结果如下：**



**实战：爬取猫眼电影排行榜并使用csv保存到本地的csv中。**

**import requests  
from fake\_useragent import UserAgent  
import re,csv  
headers = {  
 'user-agent':UserAgent().random  
}  
def get\_one\_page(url):  
 response = requests.get(url,headers=headers)  
 if response.status\_code == 200:  
 return response.text  
 return None  
def get\_data\_html(html):  
 movie\_data\_list = re.findall('<div class="movie-item-info">.\*?title="(.\*?)".\*?<p class="star">(.\*?)</p>.\*?releasetime">(.\*?)</p>',html,re.S)  
 return movie\_data\_list  
def write\_page(datas):  
 with open('1.csv','a',newline='') as f:  
 writer = csv.writer(f)  
 writer.writerow(["电影名称","主演","上映时间"])  
 for data in datas:  
 *# with open("1.csv","a",newline="") as f:  
 # 创建写入对象* writer = csv.writer(f)  
 *# L = list(r\_tuple)* L = [data[0].strip(),data[1].strip(),data[2].strip()]  
 *# ["霸王别姬","张国荣","1994-01-01"]* writer.writerow(L)  
def main(offset):  
 url = 'https://maoyan.com/board/4?offset={}'.format(offset)  
 html = get\_one\_page(url)  
 data = get\_data\_html(html)  
 write\_page(data)  
if \_\_name\_\_ == '\_\_main\_\_':  
 for i in range(0,10):  
 main(str(i\*10))**

**运行结果如下：**



**总结：**

* 1. **Python网路请求详细讲解**

**1.2 Python urllib网络请求模块讲解**

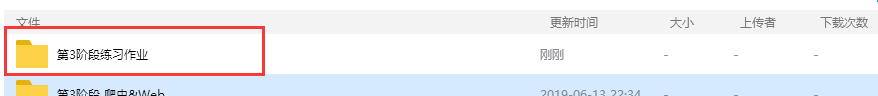
**1.3 Python requests 请求模块讲解**

**实战：利用urllib爬取百度贴吧**

**实战：爬取内涵吧脑筋急转弯**

**实战：爬取猫眼电影排行榜并使用csv保存到本地的csv中。**

**练习作业（写好py文件发到群里第三阶段练习作业）:**



**练习一：爬取当前网站**[**https://tieba.baidu.com/p/5950745302**](https://tieba.baidu.com/p/5950745302)**中的所有图片保存到本地。**

**练习二：以面型对象的形式封装本节课三个实战**

**爬取**

**附录：**

| **状态码** | **说明** | **详情** |
| --- | --- | --- |
| **100** | **继续** | **请求者应当继续提出请求。服务器已收到请求的一部分，正等待其余部分** |
| **101** | **切换协议** | **请求者已要求服务器切换协议，服务器已确认并准备切换** |
| **200** | **成功** | **服务器已成功处理了请求** |
| **201** | **已创建** | **请求成功并且服务器创建了新的资源** |
| **202** | **已接受** | **服务器已接受请求，但尚未处理** |
| **203** | **非授权信息** | **服务器已成功处理了请求，但返回的信息可能来自另一个源** |
| **204** | **无内容** | **服务器成功处理了请求，但没有返回任何内容** |
| **205** | **重置内容** | **服务器成功处理了请求，内容被重置** |
| **206** | **部分内容** | **服务器成功处理了部分请求** |
| **300** | **多种选择** | **针对请求，服务器可执行多种操作** |
| **301** | **永久移动** | **请求的网页已永久移动到新位置，即永久重定向** |
| **302** | **临时移动** | **请求的网页暂时跳转到其他页面，即暂时重定向** |
| **303** | **查看其他位置** | **如果原来的请求是POST，重定向目标文档应该通过GET提取** |
| **304** | **未修改** | **此次请求返回的网页未修改，继续使用上次的资源** |
| **305** | **使用代理** | **请求者应该使用代理访问该网页** |
| **307** | **临时重定向** | **请求的资源临时从其他位置响应** |
| **400** | **错误请求** | **服务器无法解析该请求** |
| **401** | **未授权** | **请求没有进行身份验证或验证未通过** |
| **403** | **禁止访问** | **服务器拒绝此请求** |
| **404** | **未找到** | **服务器找不到请求的网页** |
| **405** | **方法禁用** | **服务器禁用了请求中指定的方法** |
| **406** | **不接受** | **无法使用请求的内容响应请求的网页** |
| **407** | **需要代理授权** | **请求者需要使用代理授权** |
| **408** | **请求超时** | **服务器请求超时** |
| **409** | **冲突** | **服务器在完成请求时发生冲突** |
| **410** | **已删除** | **请求的资源已永久删除** |
| **411** | **需要有效长度** | **服务器不接受不含有效内容长度标头字段的请求** |
| **412** | **未满足前提条件** | **服务器未满足请求者在请求中设置的其中一个前提条件** |
| **413** | **请求实体过大** | **请求实体过大，超出服务器的处理能力** |
| **414** | **请求URI过长** | **请求网址过长，服务器无法处理** |
| **415** | **不支持类型** | **请求格式不被请求页面支持** |
| **416** | **请求范围不符** | **页面无法提供请求的范围** |
| **417** | **未满足期望值** | **服务器未满足期望请求标头字段的要求** |
| **500** | **服务器内部错误** | **服务器遇到错误，无法完成请求** |
| **501** | **未实现** | **服务器不具备完成请求的功能** |
| **502** | **错误网关** | **服务器作为网关或代理，从上游服务器收到无效响应** |
| **503** | **服务不可用** | **服务器目前无法使用** |
| **504** | **网关超时** | **服务器作为网关或代理，但是没有及时从上游服务器收到请求** |
| **505** | **HTTP版本不支持** | **服务器不支持请求中所用的HTTP协议版本** |