回顾爬虫开发环境搭建

讲授新课

认识HTTP请求

**HTTP请求方式与过程**

爬虫在爬取数据时将会作为客户端模拟整个HTTP通信过程，该过程也需要通过HTTP协议实现。HTTP请求过程如下。

* 由HTTP客户端向服务器发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。
* HTTP服务器从该端口监听客户端的请求。
* 一旦收到请求，服务器会向客户端返回一个状态，比如“HTTP/1.1 200 OK”，以及返回的响应内容，如请求的文件、错误消息、或其它信息。

|  |
| --- |
|  |
|  | C:\Users\likui\AppData\Local\Temp\ksohtml20096\wps1.png |

**HTTP请求方式与过程**

**1. 请求方法**

* 在HTTP/1.1协议中共定义了8种方法（也叫“动作”）来以不同方式操作指定的资源，常用方法有GET、HEAD、POST等。

|  |  |
| --- | --- |
| **请求方法** | **方法描述** |
| **GET** | 请求指定的页面信息，并返回实体主体。GET可能会被网络爬虫等随意访问，因此GET方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在Web Application中 |
| **HEAD** | 与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回具体的内容，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中该资源的相关信息（元信息或称元数据） |
| **POST** | 向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据会被包含在请求中，这个请求可能会创建新的资源或修改现有资源，或二者皆有 |
| **PUT** | 从客户端上传指定资源的最新内容，即更新服务器端的指定资源。 |

**2. 请求（request）与响应（response）**

HTTP协议采用了请求/响应模型。

* 客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。
* 服务器以一个状态行作为响应，响应的内容包括协议的版本、响应状态、服务器信息、响应头部和响应数据。

|  |
| --- |
|  |
|  | C:\Users\likui\AppData\Local\Temp\ksohtml20096\wps2.jpg |

客户端与服务器间的请求与响应的具体步骤如下。

* **连接Web服务器：**由一个HTTP客户端发起连接，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。
* **发送HTTP请求：**客户端经TCP套接字向Web服务器发送一个文本的请求报文。
* **服务器接受请求并返回HTTP响应：**Web服务器解析请求，定位该次的请求资源。之后将资源复本写至TCP套接字，由客户端进行读取。
* **释放连接TCP连接：**若连接的connection模式为close，则由服务器主动关闭TCP连接，客户端将被动关闭连接，释放TCP连接；若connection模式为keepalive，则该连接会保持一段时间。

**客户端解析HTML内容：**客户端首先会对状态行进行解析，之后解析每一个响应头，最后读取响应数据。

**常见HTTP状态码**

**1. HTTP状态码种类**

HTTP状态码是用来表示网页服务器响应状态的3位数字代码，按首位数字分为5类状态码。

|  |  |
| --- | --- |
| **状态码类型** | **状态码意义** |
| **1XX** | 表示请求已被接受，需接后续处理。这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束 |
| **2XX** | 表示请求已成功被服务器接收、理解并接受 |
| **3XX** | 表示需要客户端采取进一步的操作才能完成请求。通常用来重定向，重定向目标需在本次响应中指明 |
| **4XX** | 表示客户端可能发生了错误，妨碍了服务器的处理。 |
| **5XX** | 表示服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器以当前的软硬件资源无法完成对请求的处理。 |

**2. 常见HTTP状态码**

|  |  |
| --- | --- |
| **常见状态码** | **状态码含义** |
| **200 OK** | 请求成功，请求所希望的响应头或数据体将随此响应返回。 |
| **400 Bad Request** | 由于客户端的语法错误、无效的请求或欺骗性路由请求，服务器不会处理该请求 |
| **403 Forbidden** | 服务器已经理解该请求，但是拒绝执行，将在返回的实体内描述拒绝的原因，也可以不描述仅返回404响应 |
| **404 Not Found** | 请求失败，请求所希望得到的资源未被在服务器上发现，但允许用户的后续请求 |
| **500 Internal Server Error** | 通用错误消息，服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理，不会给出具体错误信息 |
| **503 Service Unavailable** | 由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是暂时的，并且将在一段时间以后恢复 |

**HTTP头部信息**

HTTP头部信息（HTTP header fields）是指在超文本传输协议（HTTP）的请求和响应消息中的消息头部分。头部信息定义了一个超文本传输协议事务中的操作参数。在爬虫中需要使用头部信息向服务器发送模拟信息，通过发送模拟的头部信息将自己伪装成一般的客户端。





Requests 库介绍

Requests 库是一个非常容易上手的 python 爬虫库，相比于 urllib 库， requests 库非常简洁。

安装命令：

pip install requests

Requests 库中有很多方法，但所有的方法在底层都是通过调用 request() 方法来实现的。因此，严格来说，Requests 库只有一个 request() 方法。但一般都不会直接使用这个方法。

下面是一些常用的方法：

get() 方法

requests 基本的 get 方法

res = requests.get(url,headers=headers,params,timeout)

参数说明如下：

url：要抓取的 url 地址。

headers：用于包装请求头信息。

params：请求时携带的查询字符串参数。

timeout：超时时间，超过时间会抛出异常。

首先来看一下 requests 下 get 方法的基本用法，下面代码输出了 get() 方法返回值的类型和状态码（状态码为 200 表示请求成功）：

import requests

response = requests.get("http://www.baidu.com")

print(type(response))

print(response.status\_code)

代码运行结果为：

<class ‘requests.models.Response’>

200

**利用返回值的 text 属性，可以得到请求的内容：**

import requests

response = requests.get("http://httpbin.org/get")

print(response.text)

输出结果：

{

“args”: {},

“headers”: {

“Accept”: “/”,

“Accept-Encoding”: “gzip, deflate”,

“Connection”: “close”,

“Host”: “httpbin.org”,

“User-Agent”: “python-requests/2.18.4”

},

“origin”: “123.126.85.145”,

“url”: “http://httpbin.org/get”

}

当我们使用 Requests 模块向一个 URL 发起请求后会返回一个 HttpResponse 响应对象，该对象具有以下常用属性：

|  |  |
| --- | --- |
| **常用属性** | **说明** |
| encoding | 查看或者指定响应字符编码 |
| status\_code | 返回HTTP响应码 |
| url | 查看请求的 url 地址 |
| headers | 查看请求头信息 |
| cookies | 查看cookies 信息 |
| text | 以字符串形式输出 |
| content | 以字节流形式输出，若要保存下载图片需使用该属性。 |

如：

import requests

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

**print**(response.encoding)

response.encoding="utf-8" #更改为utf-8编码

**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)  #以字符串形式打印网页源码

**print**(response.content) #以字节流形式打印

带参数的 get() 方法

正常写法

带参数 url 的正常写法是写在 url 后面的，代码如下：

import requests

response = requests.get("http://httpbin.org/get?name=rocketeerLi&age=22")

print(response.text)

输出结果：

{

“args”: {

“age”: “22”,

“name”: “rocketeerLi”

},

“headers”: {

“Accept”: “/”,

“Accept-Encoding”: “gzip, deflate”,

“Connection”: “close”,

“Host”: “httpbin.org”,

“User-Agent”: “python-requests/2.18.4”

},

“origin”: “123.126.85.145”,

“url”: “http://httpbin.org/get?name=rocketeerLi&age=22”

}

传参数的方式

将参数写成字典的形式，利用 get() 方法的参数进行请求，可以动态更改请求参数，代码如下：

import requests

data = {

'name':"rocketeerLi",

'age':22

}

response = requests.get("http://httpbin.org/get", params=data)

print(response.text)

输出结果是和直接在 url 上拼接是一样的：

{

“args”: {

“age”: “22”,

“name”: “rocketeerLi”

},

“headers”: {

“Accept”: “/”,

“Accept-Encoding”: “gzip, deflate”,

“Connection”: “close”,

“Host”: “httpbin.org”,

“User-Agent”: “python-requests/2.18.4”

},

“origin”: “123.126.85.145”,

“url”: “http://httpbin.org/get?name=rocketeerLi&age=22”

}

添加 headers

requests 库也可以直接添加 headers，例如，在很多时候，直接利用 get 方法访问网址的时候，会被拒绝。很大一部分原因就是没有添加请求的头部信息。例如，在访问知乎的时候，如果没有浏览器的头部信息，则访问会被拒绝，报 400 Bad Request 错误。

代码：

import requests

# requests 添加 headers

# 如果不加任何 header 会报错 400 Bad Request

response = requests.get("https://www.zhihu.com/explore")

print(response.text)

输出结果：

<html>

<head><title>400 Bad Request</title></head>

<body bgcolor="white">

<center><h1>400 Bad Request</h1></center>

<hr><center>openresty</center>

</body>

</html>

加入头部信息后的代码如下：

import requests

# 加入浏览器信息

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36'}

response = requests.get("https://www.zhihu.com/explore", headers=headers)

print(response.text)

会输出正常结果：

post() 方法

post 方法的基本使用

import requests

# 基本 post 请求

data = {'name':'rocketeerLi', 'age':'22'}

response = requests.post("http://httpbin.org/post", data=data)

print(response.text)

输出结果：

{

“args”: {},

“data”: “”,

“files”: {},

“form”: {

“age”: “22”,

“name”: “rocketeerLi”

},

“headers”: {

“Accept”: “/”,

“Accept-Encoding”: “gzip, deflate”,

“Connection”: “close”,

“Content-Length”: “23”,

“Content-Type”: “application/x-www-form-urlencoded”,

“Host”: “httpbin.org”,

“User-Agent”: “python-requests/2.18.4”

},

“json”: null,

“origin”: “123.126.85.145”,

“url”: “http://httpbin.org/post”

}

传递 headers 参数

import requests

# 传参数增加 headers

data = {'name':'rocketeerLi', 'age':'22'}

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36'}

response = requests.post("http://httpbin.org/post", data=data, headers=headers)

print(response.text)

输出结果改变了 User-Agent 的值：

{

“args”: {},

“data”: “”,

“files”: {},

“form”: {

“age”: “22”,

“name”: “rocketeerLi”

},

“headers”: {

“Accept”: “/”,

“Accept-Encoding”: “gzip, deflate”,

“Connection”: “close”,

“Content-Length”: “23”,

“Content-Type”: “application/x-www-form-urlencoded”,

“Host”: “httpbin.org”,

“User-Agent”: “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36”

},

“json”: null,

“origin”: “123.126.85.145”,

“url”: “http://httpbin.org/post”

}