爬虫基础知识

从萌新的角度通俗的说，只要解决两个问题，一个简单的爬虫就可以实现。**第一**，从对方的服务器上获取网页源码；**第二**，从网页源码中提取需要的信息并储存。今晚的主要目标，就是围绕这两个问题，给大家普及一些网站的基本概念，编写简单的爬虫代码。

注意：今晚需要大家站在一个浏览器的立场，思考如何访问网页和处理返回的源码。浏览器处理网页源码的方式和我们惯常的思维有些出入，因此可以在开始前默念：我是一个浏览器。

1 获取网页源码

1.1 了解http的访问请求

作为一个浏览器，要访问一个网页时，首先要做的就是向目标网站提交一个访问请求。http的访问请求主要有四个，Get、Post、Put与Delete。我们只需要了解get和post请求即可。下面我们将举例介绍并加以辨析。
案例：

get：www.baidu.com (http://www.baidu.com)

post：jrxy.zjsu.edu.cn/

（此两个案例组会时具体演示，不在notebook里上传视频了）

首先确定一点，上述两个案例中，我们都给两个网站传入了参数。我们在百度搜索“markdown”，“markdown”就是参数；我们登陆了金融学院后台，我们的账号密码和验证码也是参数。但是参数的隐私程度是不同的，我们可以堂堂正正说我们搜索了markdown，但我们会把各自的密码匿起来。
下面我们来考虑get和post两种请求方式在隐私程度上的差异。我们不谈互联网背后复杂的机理，就从我们眼睛看见的入手。

```
https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=markdown&fenlei=256&oq=markdown%2520%25E6%258D%25A2%25E8%25A1%258C&rsv_pq=eb9a92ab00049091&rsv_t=392eeCa7iRaXGkfFw5s9gl%2FFzh9IzYIwr900oYpjV4UvIMyILrwXsUAGO3w&rqlang=cn&rsv_dl=tb&rsv_enter=0&rsv_btype=t&inputT=350378&rsv_sug3=144&rsv_sug1=148&rsv_sug7=100&rsv_sug2=0&rsv_sug4=352089&rsv_sug=2
```

（此处穿插post的请求网站与抓包信息）

从上面的辨析中我们可以看到，get请求，已经明明白白的把我们的参数写在了网址上，因此get请求的地址会显得很长很长，那么你愿意把你的密码写在网址上吗？而post请求的网址，干干净净，没有参数的信息，我们需要用一些大家目前还不能理解的神秘工具，抓住对应的“包”，才能获取参数的具体数值。显然，get方便，而post安全。

接下来我们使用代码来实现get与post请求，使用到库requests

In [1]:

```
import requests
help(requests.get)
```

```
Help on function get in module re
quests.api:

get(url, params=None, **kwargs)
    Sends a GET request.

    :param url: URL for the new :
class:`Request` object.
    :param params: (optional) Dic
tionary, list of tuples or bytes
to send
        in the query string for t
he :class:`Request`.
    :param \*\*kwargs: Optional a
rguments that ``request`` takes.
    :return: :class:`Response <Re
sponse>` object
    :rtype: requests.Response
```

In [2]:

```
help(requests.post)
```

```
Help on function post in module r
equests.api:

post(url, data=None, json=None, *
*kwargs)
    Sends a POST request.

    :param url: URL for the new :
class:`Request` object.
    :param data: (optional) Dicti
onary, list of tuples, bytes, or
file-like
        object to send in the bod
y of the :class:`Request`.
    :param json: (optional) json
data to send in the body of the :
class:`Request`.
    :param \*\*kwargs: Optional a
rguments that ``request`` takes.
    :return: :class:`Response <Re
sponse>` object
    :rtype: requests.Response
```

In [3]:

```
# 由于百度的参数比较复杂，我们使用起点中文网搜索书名实现get参数的传递
get_url = 'https://www.qidian.com/search'
get_html = requests.get(get_url,params={'kw':'霸道总裁坏坏爱'})
print('get_html.status_code==%d'%get_html.status_code)
#基于get请求把参数写在网址上的特性，我们有第二种访问方式
get_url2 = 'https://www.qidian.com/search?kw=霸道总裁坏坏爱'
get_html2 = requests.get(get_url2)
print('get_html2.status_code==%d'%get_html2.status_code)
get_html.text==get_html2.text
```

```
get_html.status_code==200
get_html2.status_code==200
```

Out[3]:

```
True
```

In [46]:

```
#post参数的传递。我惊讶的发现学院网站关键词查找居然是用的post请求。这样就不用在代码里写上密码了^_^
post_url = 'http://jrxy.zjsu.edu.cn/jrweb/searchNews.htm'
params={'keyword':'2019','submit':'%BC%EC%CB%F7'} #'submit'参数值是通过抓包手动获取的，这里不谈sub
'为什么这里使用post请求搜索通知，为什么有一个submit参数，是值得一谈的。'
post_html = requests.post(post_url,data=params)
print('post_html.status_code==%d'%post_html.status_code)
```

```
post_html.status_code==200
```

In [5]:

```
dir(get_html)
```

Out[5]:

```
['__attrs__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__enter__',
 '__eq__',
 '__exit__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
```

In [18]:

```
get_html2.text
```

Out[18]:

```
'<!DOCTYPE html>\r<html>\r\r<head>\r    <meta charset="UTF-8">\r    <title>起点搜书_起点中文网</title>\r    <meta name="description" content="搜索最新最快原创小说,搜索您最喜爱的小说,尽在起点中文网">\r    <meta name="keywords" content="搜索小说,搜书,搜小说,搜图书,搜作者">\r    <meta name="robots" content="all">\r    <meta name="googlebot" content="all">\r<meta name="baiduspider" content="all">\r    <meta http-equiv="mobile-agent" content="format=wml; url=https://m.qidian.com">\r    <meta http-equiv="mobile-agent" content="format=xh
```

In [31]:

```
dir(post_html)
```

Out[31]:

```
['__attrs__',
 '__bool__',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__enter__',
 '__eq__',
 '__exit__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
```

In [47]:

```
post_html.text
```

Out[47]:

```
'\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\r\n<html xmlns="http://www.w3.org/1999/xhtml">\r\n<head>\r\n<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />\r\n<meta name="viewport"  content="width=device-width, user-scalable=no, initial-scale=0.3, maximum-scale=0.3, minimum-scale=0.3">\r\n<title>浙江工商大学金融学院</title>\r\n<meta name="keywords" content="" />\r\n<meta name="description" conten
```

1.2 请求头

接下来我们来介绍请求头的概念。如果把http请求当成是一份调用目标服务器的“申报书”，那么这份“申报书”中至少需要三个内容：1、向谁申报（url）2、申报具体内容（上节中的参数概念）3、申请人的个人信息。

我们已经在上一节中解决了关于前两个内容的问题。下面我们来考虑网站对“申请人”身份的要求。最简单的要求就是，你得让对方网站知道你是个人，真人，活人，不是机器人。哪怕要求更为复杂，我们也可以归纳来看，请求头无非是需要我们提交更多的参数，和查找内容无关的其它参数，比如现在的时间，比如浏览器的型号，再比如cookie，来对访问者的个人信息加以确认。

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,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: newstatisticUUID=1625418152_696176994; _csrfToken=74k0nGvQJ1jRIkyLqFsJy9g6iMgtHAterlBYTFHS; qdrs=0%7C3%7C0%7C0%7C1; showSectionCommentGuide=1; qdgd=1; rcr=1027094496%2C1030312071%2C1024644779%2C1027517128%2C1026225232%2C1030011567%2C1026830638%2C1010868264%2C1028483892%2C1026927551%2C1025587263%2C1979049; _gid=GA1.2.1368640412.1636563077; _yep_uuid=1be4c45b-77ec-0127-ba43-747925676128; _gat_gtag_UA_199934072_2=1; e1=%7B%22pid%22%3A%22qd_P_Searchresult%22%2C%22eid%22%3A%22qd_S01%22%2C%2211%22%3A2%7D; e2=%7B%22pid%22%3A%22qd_p_qidian%22%2C%22eid%22%3A%22qd_H_Search%22%2C%2211%22%3A2%7D; _ga_FZMMH98S83=GS1.1.1636595417.14.1.1636595432.0; _ga_PFYW0QLV3P=GS1.1.1636595417.18.1.1636595432.0; _ga=GA1.2.1790910485.1625418151

sec-ch-ua: "Google Chrome";v="95", "Chromium";v="95", ";Not A Brand";v="99"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

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/95.0.4638.69 S

In [11]:

```
get_html.request.headers
```

Out[11]:

```
{'User-Agent': 'python-requests/2.25.1', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
```

对比一下浏览器的请求头截图与我们爬虫的请求头。我们会发现浏览器的请求头详实许多，但是我们还是获得了我们请求的网页。这是因为不同的网站、甚至同一网站的不同价值的信息，对爬虫的防范力度都有所出入。一般复杂的网站，会对请求头有比较高的要求。但是哪怕是面对再不设防的网站，都应该给我们的爬虫安一个带user-agent参数的请求头。因为不设置的话，我们会看到爬虫的默认user-agent变成了'python-requests/2.25.1'。检查user-agent是网站最基本的反爬虫手段。

获取网页的源码，最简单的说法，就是厘清我们要访问的url，要提交哪些参数，构造被对方服务器认可的请求头。但是在实践中，获取网页源码也是爬虫最难的地方。我们可能会遇到如下问题：

- 对方觉得你同一个IP地址访问频率过高，并封了你的IP（反制手段构造IP池）
- 对方觉得你登录的账号访问频率过高，并封了你的账号（反制手段慢慢爬或者多账号）
- 对方选择post方法传参并构造一个“密匙”类型的参数，如post例子中的submit参数，我们需要从网页源码入手，甚至阅读js代码，根据密匙构造的代码反向解密
- 对方用自己独一无二的字体编写网站，如果不同时捕捉字体信息，解压全乱码
- 对方使用异步动态加载，我们需要通过抓包获取访问的具体网址。
- ……

爬虫与反爬虫，是博弈，随时可能有新的花样，具体操作中遇到的坑可能永远没有尽头，爬坑的方法也是。今晚主要介绍一些爬虫的基本知识，具体的应用问题大家可以查阅相关书籍，或在实践中多多尝试。

2 网站解析

网站解析对初学者来说也是一大痛点，但是如果真的理解了爬虫的机制，我们会发现网站解析其实在技术上不存在“博弈”，是一种固定模式固定方法的体力活，在这一块大家把套路了解清楚即可进入熟能生巧的练习阶段。但是在了解相关的解析工具以前，我们得先粗略地了解一下超文本标记语言。

2.1 超文本标记语言

具体的定义百度上有，我们直接借助例子帮大家理解网页的三种特性

- 1、网页的底层，可以当做是字符串。我们可以用txt打开后缀为.html的文件，也可以用python打开txt文件的方法读取html文件并打印。而网站的编写其实也就是一个一个字符串敲打在txt文件中——当然要遵循一定的规则。高手是可以打开一个空的txt文件不借助其它程序徒手敲出一个网站的。

In [5]:

```
with open('小说,小说网,最新热门小说-起点中文网_阅文集团旗下网站.html','r',encoding='utf-8') as f:
    print(help(f.read))
    text = f.read()
text
```

```
Help on built-in function read:

read(size=-1, /) method of _io.TextIOWrapper instance
    Read at most n characters from stream.

    Read from underlying buffer until we have n characters or we hit EOF.
    If n is negative or omitted, read until EOF.

None
```

- 2、网页是有鲜明的结构的。这种结构可以从最后网页的呈现中体现出来。呈现的结构，如菜单、主体内容、结尾的合作机构等。但源码的结构又有所不同。

源码的结构可以参考：https://zhuanlan.zhihu.com/p/97558113 (https://zhuanlan.zhihu.com/p/97558113)(组会直接单机此链接细谈html语言的规则)

- 3、网页是有鲜明的色彩，或者说样式的。网络上也有各式各样的网页模板。对于一个网页来说，部分内容可能采用a样式（包括色彩，字体大小，行间距等），另一块内容可能采用不同b样式。

2.2 python对于html的解析

基于上述的三种特性，我们可以引出三种方法从网页中提取信息，对应三个python库，re，lxml，beautifulsoup。这三个库既不需要大家今晚就融会贯通，也不要求大家将来都熟稔于心。我本人就从来不用beautifulsoup，也就是根据样式提取信息，一般用正则和xpath，基本可以解决个人的需求。

案例：从保存好的起点中文网首页中提取所有本周强推的小说

名。

我们将尝试使用三种方法提取小说名，但是需要提前说明的是，由于本案例相对比较阴间，所以三种方法提取的结果中，似乎只有lxml是正确的。

基于特点一，我们可以完全基于字符串查找的方式提取我们想要的内容，当然这要求对正则表达式有一定的理解。

首先我们使用右键检查来看看哪些书名周围的字符串。

In [84]:

```
import re

p1 = re.compile(r'<strong>(.*?)</strong>' )
result1 = p1.findall(text)
result2 = list(filter(lambda x:'data-eid="qd_A
p2 = re.compile(r'>(.*?)<' )
result3= list(map(lambda x:p2.findall(x)[0],re
print(len(result3))
result3
```

17

Out[84]:

['灵魂画手',
'镜面管理局',
'我将埋葬众神',
'诸天从洪拳开始',
'我直接人生重开',
'求道从红楼开始',
'开局装成造物主',
'我真不是乱选的',
'道友你剧本真好看',
'修仙从皮影戏开始',
'没人比我更懂魔物',
'重生从不做备胎开始',
'从推进城到多元宇宙',
'诸天从洛洛历险记开始',
'穿越S7，你管这叫辅助？',
'不过是黑魔法防御课教授罢了',
'探秘全球：从发现绿尾虹雉开始']

基于特点二，了解网页源码编写的规则，就可以理解网页的整个结构，借助这种结构，我们可以查找部分内容在整体中的位置，从而根据位置提取对应的信息，完成爬虫信息的提取。

从图中可以清晰地看到网页中存在明显的结构。至于如何提取与表达这种结构，我们可以用谷歌浏览

器快乐地复制，也可也在深入了解之后使用各种变式

In [91]:

```
from lxml import etree


html_xpath = etree.HTML(text)
#/html/body/div[1]/div[7]/div[1]/div/ul/li[1]/s
elements = html.xpath("//a[@data-eid='qd_A103']
names = list(map(lambda x:x.get('title'),eleme
names
```

Out[91]:

```
['灵魂画手',
 '镜面管理局',
 '我将埋葬众神',
 '诸天从洪拳开始',
 '我直接人生重开',
 '求道从红楼开始',
 '开局装成造物主',
 '我真不是乱选的',
 '道友你剧本真好看',
 '修仙从皮影戏开始',
 '没人比我更懂魔物',
 '重生从不做备胎开始',
 '从推进城到多元宇宙',
 '诸天从洛洛历险记开始',
 '穿越S7，你管这叫辅助？',
 '不过是黑魔法防御课教授罢了',
 '探秘全球：从发现绿尾虹雉开始']
```

基于特点三，我们可以根据网页的样式查找书名。

In [103]:

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(text, "html.parser")
elements = soup.find_all(attrs={'data-eid':'qd_
names = list(map(lambda x:x.get('title'),eleme
names
```

Out[103]:

```
['灵魂画手',
 '镜面管理局',
 '我将埋葬众神',
 '诸天从洪拳开始',
 '我直接人生重开',
 '求道从红楼开始',
 '开局装成造物主',
 '我真不是乱选的',
 '道友你剧本真好看',
 '修仙从皮影戏开始',
 '没人比我更懂魔物',
 '重生从不做备胎开始',
 '从推进城到多元宇宙',
 '诸天从洛洛历险记开始',
 '穿越S7，你管这叫辅助？',
 '不过是黑魔法防御课教授罢了',
 '探秘全球：从发现绿尾虹雉开始']
```

3 数据入库

上述17本书名构成的列表，可以用pickle，json保存，也可以保存为csv，excel，也可以存入数据库中。具体如何保存就不再赘述了。