# 网络爬虫

（又被称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。

根据使用场景，网络爬虫可分为 通用爬虫 和 聚焦爬虫 两种.

## 通用爬虫

通用网络爬虫是捜索引擎抓取系统（Baidu、Google、Yahoo等）的重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。

通用搜索引擎（Search Engine）工作原理

通用网络爬虫，从互联网中搜集网页，采集信息，这些网页信息用于为搜索引擎建立索引从而提供支持，它决定着整个引擎系统的内容是否丰富，信息是否即时，因此其性能的优劣直接影响着搜索引擎的效果。

通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询，无法准确理解用户的具体需求。

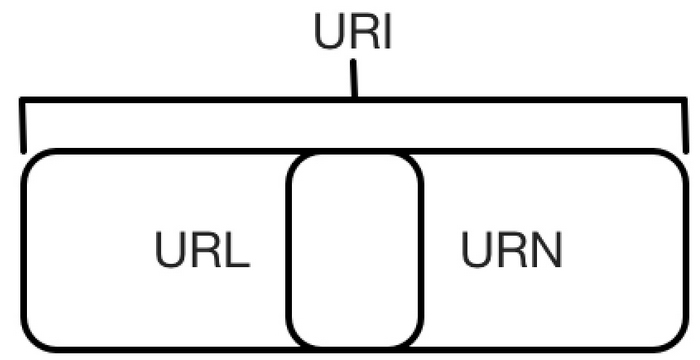
Robots协议（也叫爬虫协议、机器人协议等），全称是“网络爬虫排除标准”（Robots Exclusion Protocol），网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取



## 聚焦爬虫

聚焦爬虫，是"面向特定主题需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： 聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。

# HTTP基本原理



## URL

（Uniform / Universal Resource Locator的缩写）：统一资源定位符，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。

基本格式：scheme://host[:port#]/path/…/[?query-string][#anchor]

scheme：协议(例如：http, https, ftp)

host：服务器的IP地址或者域名

port#：服务器的端口（如果是走协议默认端口，缺省端口80）

path：访问资源的路径

query-string：参数，发送给http服务器的数据

anchor：锚（跳转到网页的指定锚点位置）

## URI

全称为 Uniform Resource Identifier，即统一资源标志符。URL 是 URI 的子集，也就是说每个 URL 都是 URI，但不是每个 URI 都是 URL

URI 还包括一个子类叫做 URN，它的全称为 Universal Resource Name，即统一资源名称。URN 只命名资源而不指定如何定位资源，如 urn:isbn:0451450523

## HTTP、HTTPS

HTTP协议（HyperText Transfer Protocol，超文本传输协议）：是一种发布和接收 HTML页面的方法。用于从网络传输超文本数据到本地浏览器的传送协议，保证传送高效而准确地传送超文本文档

HTTPS（Hypertext 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，在HTTP下加入SSL层。

HTTPS 的安全基础是 SSL（Secure Sockets Layer 安全套接层），因此通过它传输的内容都是经过 SSL 加密的

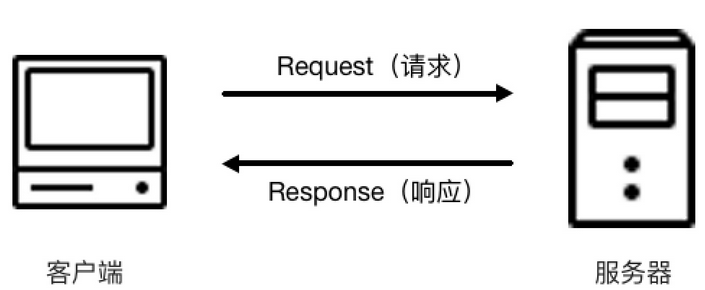
SSL主要用于Web的安全传输协议，在传输层对网络连接进行加密，保障在Internet上数据传输的安全，它的主要作用可以分为两种：

* 建立一个信息安全通道，来保证数据传输的安全。
* 确认网站的真实性，凡是使用了 https 的网站，都可以通过点击浏览器地址栏的锁头标志来查看网站认证之后的真实信息，也可以通过 CA 机构颁发的安全签章来查询。

我们如果要爬取这样的站点就需要设置忽略证书的选项，否则会提示 SSL 链接错误

HTTP的端口号为80，HTTPS的端口号为443

## HTTP请求过程



### Request

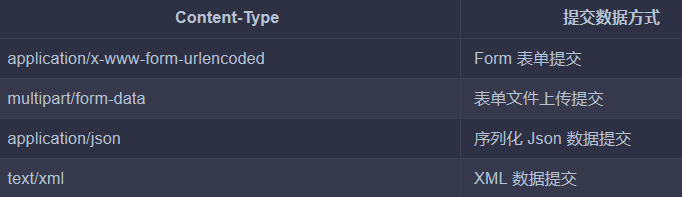
即请求，由客户端向服务端发出。可以将 Request 划分为四部分内容：Request Method、Request URL、Request Headers、Request Body，即请求方式、请求链接、请求头、请求体。

Request Method

请求方式常见的有两种类型，GET 和 POST。

在浏览器中直接输入一个 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，例如百度搜索 Python，这就是一个 GET 请求，链接为：https://www.baidu.com/s?wd=Python，URL 中包含了请求的参数信息，这里参数 wd 就是要搜寻的关键字。POST 请求大多为表单提交发起，如一个登录表单，输入用户名密码，点击登录按钮，这通常会发起一个 POST 请求，其数据通常以 Form Data 即表单的形式传输，不会体现在 URL 中。

* GET 方式请求中参数是包含在 URL 里面的，数据可以在 URL 中看到，而 POST 请求的 URL 不会包含这些数据，数据都是通过表单的形式传输，会包含在 Request Body 中。
* GET 方式请求提交的数据最多只有 1024 字节，而 POST 方式没有限制。



### Response

响应，由服务端返回给客户端。Response 可以划分为三部分，Response Status Code、Response Headers、Response Body。

Response Status Code

常见的错误代码及错误原因

200 成功 服务器已成功处理了请求。

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。

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。

500 服务器内部错误 服务器遇到错误，无法完成请求。

### IP代理

**代理分类**

**(1) 根据协议区分**

根据代理的协议，代理可以分为如下类别。

* FTP代理服务器：主要用于访问FTP服务器，一般有上传、下载以及缓存功能，端口一般为21、2121等。
* HTTP代理服务器：主要用于访问网页，一般有内容过滤和缓存功能，端口一般为80、8080、3128等。
* SSL/TLS代理：主要用于访问加密网站，一般有SSL或TLS加密功能（最高支持128位加密强度），端口一般为443。
* RTSP代理：主要用于访问Real流媒体服务器，一般有缓存功能，端口一般为554。
* Telnet代理：主要用于telnet远程控制（黑客入侵计算机时常用于隐藏身份），端口一般为23。
* POP3/SMTP代理：主要用于POP3/SMTP方式收发邮件，一般有缓存功能，端口一般为110/25。
* SOCKS代理：只是单纯传递数据包，不关心具体协议和用法，所以速度快很多，一般有缓存功能，端口一般为1080。SOCKS代理协议又分为SOCKS4和SOCKS5，前者只支持TCP，而后者支持TCP和UDP，还支持各种身份验证机制、服务器端域名解析等。简单来说，SOCK4能做到的SOCKS5都可以做到，但SOCKS5能做到的SOCK4不一定能做到。

**(2) 根据匿名程度区分**

根据代理的匿名程度，代理可以分为如下类别。

* 高度匿名代理：会将数据包原封不动地转发，在服务端看来就好像真的是一个普通客户端在访问，而记录的IP是代理服务器的IP。
* 普通匿名代理：会在数据包上做一些改动，服务端上有可能发现这是个代理服务器，也有一定几率追查到客户端的真实IP。代理服务器通常会加入的HTTP头有HTTP\_VIA和HTTP\_X\_FORWARDED\_FOR。
* 透明代理：不但改动了数据包，还会告诉服务器客户端的真实IP。这种代理除了能用缓存技术提高浏览速度，能用内容过滤提高安全性之外，并无其他显著作用，最常见的例子是内网中的硬件防火墙。
* 间谍代理：指组织或个人创建的用于记录用户传输的数据，然后进行研究、监控等目的的代理服务器。

**常见代理设置**

* 使用网上的免费代理：最好使用高匿代理，另外可用的代理不多，需要在使用前筛选一下可用代理，也可以进一步维护一个代理池。
* 使用付费代理服务：互联网上存在许多代理商，可以付费使用，质量比免费代理好很多。
* ADSL拨号：拨一次号换一次IP，稳定性高，也是一种比较有效的解决方案。

# Scrapy

使用Cookie和使用代理IP一样，也需要创建一个自己的opener。在HTTP包中，提供了cookiejar模块，用于提供对Cookie的支持。

可以利用本模块的CookieJar类的对象来捕获cookie并在后续连接请求时重新发送，比如可以实现模拟登录功能。该模块主要的对象有CookieJar、FileCookieJar、MozillaCookieJar、LWPCookieJar。

CookieJar–派生–>FileCookieJar–派生–>MozillaCookieJar和LWPCookieJar

创建一个带有cookie的opener，在访问登录的URL时，将登录后的cookie保存下来，然后利用这个cookie来访问其他网址。查看登录之后才能看到的信息。

# Selenium

#模拟提交搜索  
browser = webdriver.Firefox()  
#打开请求的URL，webdriver会等到页面全部加载完毕后返回  
browser.get('http://www.baidu.com/')  
assert '百度一下，你就知道' in browser.title  
#find\_element\_by\_\*方法寻找网页元素  
elem=browser.find\_element\_by\_name('wd')  
#模拟键盘输入  
elem.send\_keys('python')  
#模拟回车  
elem.send\_keys(Keys.RETURN)  
#打印提交后页面信息  
print(browser.page\_source)

## 元素选取

单个元素选取（多个元素选取，将element变为elements）

find\_element\_by\_id

find\_element\_by\_name

find\_element\_by\_xpath

find\_element\_by\_link\_text

find\_element\_by\_partial\_link\_text

find\_element\_by\_tag\_name

find\_element\_by\_class\_name

find\_element\_by\_css\_selector

利用 By 类来确定哪种选择方式

from selenium.webdriver.common.by import By

find\_element(By.XPATH, '//button[text()="Some text"]')

## 界面交互

通过元素选取，能够找到元素的位置，根据这个元素的位置进行相应的事件操作，例如输入文本框内容、鼠标单击、填充表单、元素拖拽等等

elem = driver.find\_element\_by\_xpath("//a[@data-fun='next']")

elem.click()

点击这个按键之前，窗口最好移动到那里，因为如果这个按键被其他元素遮挡，click()就触发异常。

将窗口滑动到page这个位置

page = driver.find\_elements\_by\_xpath("//div[@class='page']")

driver.execute\_script('arguments[0].scrollIntoView();', page[-1]) #拖动到可见的元素去

## Xpath（XML Path）

元素查找方式，几乎可以定位到页面上的任意元素

xpath路径以//开头时，则表示让xpath引擎从文档的任意符合的元素节点开始进行解析，/出现在xpath路径中时，则表示寻找父节点的直接子节点，当//出现在xpath路径中时，表示寻找父节点下任意符合条件的子节点，用\*号省略具体的标签名称，但元素的层级关系必须体现出来

## 构造合理http请求头

if \_\_name\_\_ == '\_\_main\_\_':  
 url='https://www.baidu.com/'  
 header= {'Upgrade-Insecure-Requests':'1',  
 'User-Agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36',  
 'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8',  
 'Accept-Encoding':'gzip, deflate, sdch, br',  
 'Accept-Language':'zh-CN,zh;q=0.8',  
 }  
 s=requests.Session()  
 req=s.get(url=url,headers=header)  
 print(s.cookies)

### 静态网页和动态网页

网页的内容是 HTML 代码编写的，文字、图片等内容均是通过写好的 HTML 代码来指定的，这种页面叫做静态网页。

这种网页加载速度快，编写简单，但是存在很大的缺陷，如可维护性差，不能根据 URL 灵活多变地显示内容等，例如我们想要给这个网页的 URL 传入一个 name 参数，让其在网页中显示出来，是无法做到的。

动态网页可以动态解析 URL 中参数的变化，关联数据库并动态地呈现不同的页面内容，非常灵活多变，我们现在遇到的大多数网站都是动态网站，它们不再是一个简单的 HTML，而是可能由 JSP、PHP、Python 等语言编写的，功能相比静态网页强大和丰富太多

### 无状态HTTP

HTTP 的无状态是指 HTTP 协议对事务处理是没有记忆能力的，也就是说服务器不知道客户端是什么状态。当我们向服务器发送一个 Requset 后，服务器解析此 Request，然后返回对应的 Response，服务器负责完成这个过程，而且这个过程是完全独立的，服务器不会记录前后状态的变化，也就是缺少状态记录，这意味着如果后续需要处理需要前面的信息，则它必须要重传

保持 HTTP 连接状态的技术, Session 和 Cookies

Session 在服务端，也就是网站的服务器，用来保存用户的会话信息，Cookies 在客户端，也可以理解为浏览器端，有了 Cookies，浏览器在下次访问网页时会自动附带上它发送给服务器，服务器通过识别 Cookies 并鉴定出是哪个用户，然后再判断用户是否是登录状态，然后返回对应的 Response。

因此在爬虫中，有时候处理需要登录才能访问的页面时，我们一般会直接将登录成功后获取的 Cookies 放在 Request Headers 里面直接请求，而不必重新模拟登录。

### Session

在 Web 中 Session 对象用来存储特定用户会话所需的属性及配置信息。这样，当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。

除非程序通知服务器删除一个 Session，否则服务器会一直保留，比如程序一般都是在我们做注销操作的时候才去删除 Session。由于关闭浏览器不会导致 Session 被删除，这就需要服务器为 Seesion 设置一个失效时间，当距离客户端上一次使用 Session 的时间超过这个失效时间时，服务器就可以认为客户端已经停止了活动，才会把 Session 删除以节省存储空间。

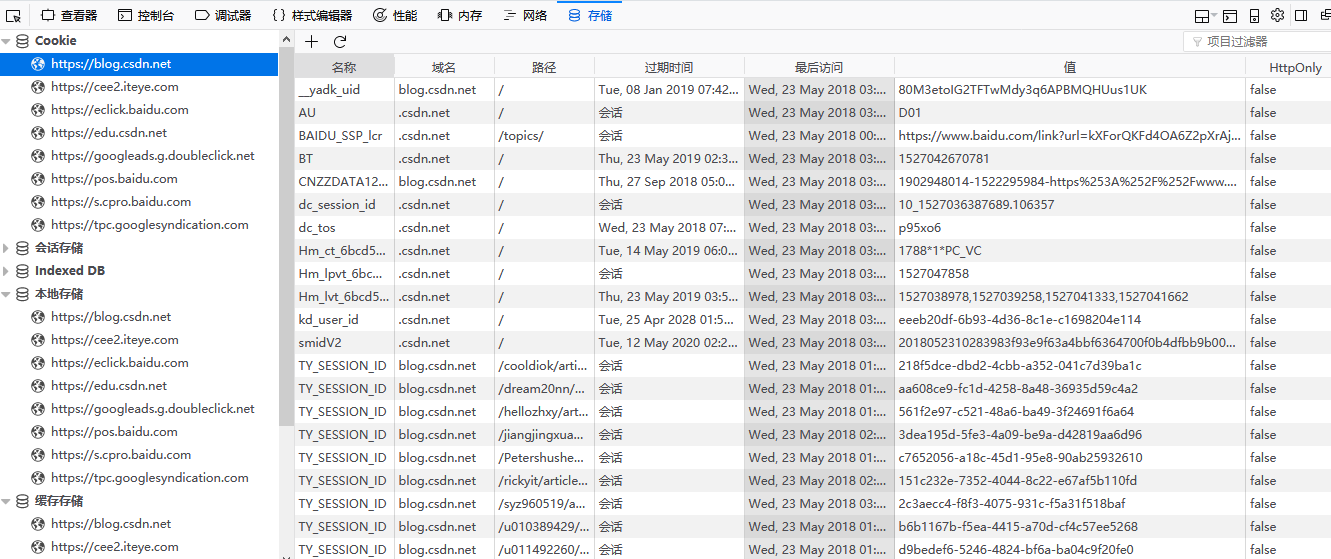
requests.Session 会话对象让你能够跨请求保持某些参数，它也会在同一个 Session 实例发出的所有请求之间保持 cookie，requests 模块不能执行 JavaScript

### Cookies

Cookie，有时也用其复数形式 Cookies，指某些网站为了辨别用户身份、进行 Session 跟踪而储存在用户本地终端上的数据。

当客户端第一次请求服务器时，服务器会返回一个 Headers 中带有 Set-Cookie 字段的 Response 给客户端，用来标记是哪一个用户，客户端浏览器会把Cookies 保存起来。当浏览器下一次再请求该网站时，浏览器会把此Cookies 放到 Request Headers 一起提交给服务器，Cookies 携带了 Session ID 信息，服务器检查该 Cookies 即可找到对应的 Session 是什么，然后再判断 Session 来以此来辨认用户状态。

Cookies 和 Session 需要配合，一个处于客户端，一个处于服务端，二者共同协作，就实现了登录会话控制。



* Name，即该 Cookie 的名称。Cookie 一旦创建，名称便不可更改
* Value，即该 Cookie 的值。如果值为 Unicode 字符，需要为字符编码。如果值为二进制数据，则需要使用 BASE64 编码。
* Max Age，即该 Cookie 失效的时间，单位秒，也常和 Expires 一起使用，通过它可以计算出其有效时间。Max Age 如果为正数，则该Cookie 在 Max Age 秒之后失效。如果为负数，则关闭浏览器时Cookie 即失效，浏览器也不会以任何形式保存该 Cookie。
* Path，即该 Cookie 的使用路径。如果设置为 /path/，则只有路径为 /path/ 的页面可以访问该 Cookie。如果设置为 /，则本域名下的所有页面都可以访问该 Cookie。
* Domain，即可以访问该 Cookie 的域名。例如如果设置为 .zhihu.com，则所有以 zhihu.com，结尾的域名都可以访问该Cookie。
* Size字段，即此 Cookie 的大小。
* Http字段，即 Cookie 的 httponly 属性。若此属性为 true，则只有在 HTTP Headers 中会带有此 Cookie 的信息，而不能通过 document.cookie 来访问此 Cookie。
* Secure，即该 Cookie 是否仅被使用安全协议传输。安全协议。安全协议有 HTTPS，SSL 等，在网络上传输数据之前先将数据加密。默认为 false。

会话Cookie、持久Cookie

会话 Cookie 就是把 Cookie 放在浏览器内存里，浏览器在关闭之后该 Cookie 即失效，持久 Cookie 则会保存到客户端的硬盘中，下次还可以继续使用，用于长久保持用户登录状态。

# PhantomJS

是一个“无头”（headless）浏览器。它会把网站加载到内存并执行页面上的 JavaScript，但不会向用户展示网页的图形界面。

driver=webdriver.PhantomJS(executable\_path='D:/下载/phantomjs\_xpgod/phantomjs-2.1.1-windows/bin/phantomjs.exe')  
driver.get(url)  
driver.implicitly\_wait(1)  
print(driver.get\_cookies())

# 反爬虫

## 正常的访问速度

尽量保证一次加载页面加载且数据请求最小化，尽量每个页面访问增加时间间隔

## 隐含输入字段

（隐含字段主要阻止爬虫自动提交表单）

在 HTML 表单中，“隐含”字段可以让字段的值对浏览器可见，但是对用户不可见（除非看网页源代码），阻止网络数据采集的方式主要有两种。

第一种是表单页面上的一个字段可以用服务器生成的随机变量表示。如果提交时这个值不在表单处理页面上，服务器就有理由认为这个提交不是从原始表单页面上提交的，绕开这个问题的最佳方法就是，首先采集表单所在页面上生成的随机变量，然后再提交到表单处理页面。

## “蜜罐”（honey pot）

如果表单里包含一个具有普通名称的隐含字段（设置蜜罐圈套），比如“用户名”（username）或“邮箱地址”（email address），设计不太好的网络机器人往往不管这个字段是不是对用户可见，直接填写这个字段并向服务器提交，这样就会中服务器的蜜罐圈套。服务器会把所有隐含字段的真实值（或者与表单提交页面的默认值不同的值）都忽略，而且填写隐含字段的访问用户也可能被网站封杀。

# 验证码识别思路

1、图片降噪

降噪就是把不需要的信息通通去除，比如背景，干扰线，干扰像素等等，只剩下需要识别的文字，让图片变成2进制点阵最好。

对于彩色背景的验证码：每个像素都可以放在一个5维的空间里，这5个维度分别是，X,Y,R,G,B，也就是像素的坐标和颜色

2、图片切割

3、图像文本输出