**爬虫基础讲义**

小小图灵社出品

整理 by Orash

## 什么是爬虫?

网络爬虫也叫网络蜘蛛，如果把互联网比喻成一个蜘蛛网，那么蜘蛛就是在网上爬来爬去的蜘蛛，爬虫程序通过请求url地址，根据响应的内容进行解析采集数据，比如：如果响应内容是html，分析dom结构，进行dom解析、或者正则匹配，如果响应内容是xml/json数据，就可以转数据对象，然后对数据进行解析。

## 应用领域

1. 批量采集某个领域的招聘数据，对某个行业的招聘情况进行分析
2. 批量采集某个行业的电商数据，以分析出具体热销商品，进行商业决策，并采集目标客户数据，以进行后续营销
3. 批量爬取腾讯动漫的漫画，以实现脱网本地集中浏览
4. 开发一款火车票抢票程序，以实现自动抢票

## 小试牛刀

怎样扒网页呢？

其实就是根据URL来获取它的网页信息，虽然我们在浏览器中看到的是一幅幅优美的画面，但是其实是由浏览器解释才呈现出来的，实质它是一段HTML代码，加 JS、CSS，如果把网页比作一个人，那么HTML便是他的骨架，JS便是他的肌肉，CSS便是它的衣服。所以最重要的部分是存在于HTML中的，下面我们就写个例子来扒一个网页下来



真正的程序就两行，执行如下命令查看运行结果，感受一下。

看，这个网页的源码已经被我们扒下来了，是不是很酸爽？

## 常见的方法

**requset.urlopen(url,data,timeout)**

第一个参数url即为URL，第二个参数data是访问URL时要传送的数据，第三个timeout是设置超时时间。

第二三个参数是可以不传送的，data默认为空None，timeout默认为 socket.\_GLOBAL\_DEFAULT\_TIMEOUT

第一个参数URL是必须要传送的，在这个例子里面我们传送了百度的URL，执行urlopen方法之后，返回一个response对象，返回信息便保存在这里面。

**response.read()**

read()方法就是读取文件里的全部内容，返回bytes类型

**response.getcode()**

返回 HTTP的响应码，成功返回200，4服务器页面出错，5服务器问题

**response.geturl()**

返回实际数据的实际URL，防止重定向问题

**response.info()**

返回服务器响应的HTTP报头

## Request对象

其实上面的urlopen参数可以传入一个request请求,它其实就是一个Request类的实例，构造时需要传入Url,Data等等的内容。比如上面的两行代码，我们可以这么改写：



运行结果是完全一样的，只不过中间多了一个request对象，推荐大家这么写，因为在构建请求时还需要加入好多内容，通过构建一个request，服务器响应请求得到应答，这样显得逻辑上清晰明确。

## Get请求

大部分被传输到浏览器的html，images，js，css, … 都是通过GET方法发出请求的。它是获取数据的主要方法。

例如：[www.baidu.com](http://www.baidu.com) 搜索

Get请求的参数都是在Url中体现的,如果有中文，需要转码，这时我们可使用

**urllib.parse.urlencode()**

**urllib.parse. quote()**

## Post 请求

我们说了Request请求对象的里有data参数，它就是用在POST里的，我们要传送的数据就是这个参数data，data是一个字典，里面要匹配键值对。

发送请求/响应header头的含义：

|  |  |
| --- | --- |
| **名称** | **含义** |
| Accept | 告诉服务器，客户端支持的数据类型 |
| Accept-Charset | 告诉服务器，客户端采用的编码 |
| Accept-Encoding | 告诉服务器，客户机支持的数据压缩格式 |
| Accept-Language | 告诉服务器，客户机的语言环境 |
| Host | 客户机通过这个头告诉服务器，想访问的主机名 |
| If-Modified-Since | 客户机通过这个头告诉服务器，资源的缓存时间 |
| Referer | 客户机通过这个头告诉服务器，它是从哪个资源来访问服务器的。（一般用于防盗链） |
| User-Agent | 客户机通过这个头告诉服务器，客户机的软件环境 |
| Cookie | 客户机通过这个头告诉服务器，可以向服务器带数据 |
| Refresh | 服务器通过这个头，告诉浏览器隔多长时间刷新一次 |
| Content-Type | 服务器通过这个头，回送数据的类型 |
| Content-Language | 服务器通过这个头，告诉服务器的语言环境 |
| Server | 服务器通过这个头，告诉浏览器服务器的类型 |
| Content-Encoding | 服务器通过这个头，告诉浏览器数据采用的压缩格式 |
| Content-Length | 服务器通过这个头，告诉浏览器回送数据的长度 |

## 响应的编码

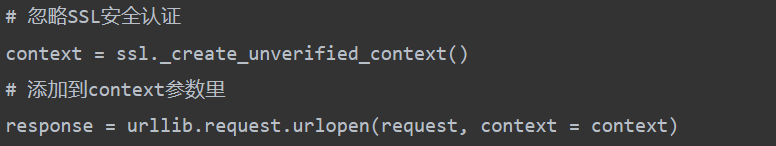
响应状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值。 常见状态码：

| **号码** | **含义** |
| --- | --- |
| 100~199 | 表示服务器成功接收部分请求，要求客户端继续提交其余请求才能完成整个处理过程 |
| 200~299 | 表示服务器成功接收请求并已完成整个处理过程。常用200（OK 请求成功） |
| 300~399 | 为完成请求，客户需进一步细化请求。例如：请求的资源已经移动一个新地址、常用302（所请求的页面已经临时转移至新的url）、307和304（使用缓存资源） |
| 400~499 | 客户端的请求有错误，常用404（服务器无法找到被请求的页面）、403（服务器拒绝访问，权限不够） |
| 500~599 | 服务器端出现错误，常用500（请求未完成。服务器遇到不可预知的情况） |

## 请求 SSL证书验证

现在随处可见 https 开头的网站，urllib可以为 HTTPS 请求验证SSL证书，就像web浏览器一样，如果网站的SSL证书是经过CA认证的，则能够正常访问，如：<https://www.baidu.com/>

如果SSL证书验证不通过，或者操作系统不信任服务器的安全证书，比如浏览器在访问12306网站如：<https://www.12306.cn/mormhweb/>的时候，会警告用户证书不受信任。（据说 12306 网站证书是自己做的，没有通过CA认证）



## 伪装自己

有些网站不会同意程序直接用上面的方式进行访问，如果识别有问题，那么站点根本不会响应，所以为了完全模拟浏览器的工作，我们要伪装自己。

#### 设置请求头

其中User-Agent代表用的哪个请求的浏览器

对付防盗链，服务器会识别headers中的referer是不是它自己，如果不是，有的服务器不会响应，所以我们还可以在headers中加入referer。

提示：在此可以使用多个User\_Agent:然后随即选择

对于随机UserAgent,PYthon有提供一个模块库fake-useragent，我们可以pip



#### 设置代理Proxy

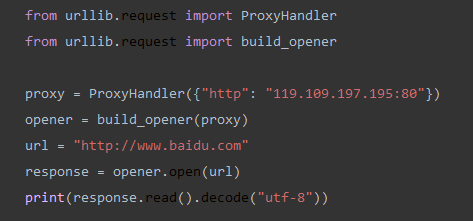
假如一个网站它会检测某一段时间某个IP 的访问次数，如果访问次数过多，它会禁止你的访问。所以你可以设置一些代理服务器来帮助你做工作，每隔一段时间换一个代理，哈哈，是不是很舒服！

**分类：**

透明代理：目标网站知道你使用了代理并且知道你的源IP地址，这种代理显然不符合我们这里使用代理的初衷。

匿名代理：匿名程度比较低，也就是网站知道你使用了代理，但是并不知道你的源IP地址。

高匿代理：这是最保险的方式，目标网站既不知道你使用的代理更不知道你的源IP。



## Cookie

为什么要使用Cookie呢？

Cookie，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（通常经过加密）

比如说有些网站需要登录后才能访问某个页面，在登录之前，你想抓取某个页面内容是不允许的。那么我们可以利用Urllib库保存我们登录的Cookie，然后再抓取其他页面就达到目的了。

#### Opener

当你获取一个URL你使用一个opener(一个urllib.OpenerDirector的实例)。在前面，我们都是使用的默认的opener，也就是urlopen。它是一个特殊的opener，可以理解成opener的一个特殊实例，传入的参数仅仅是url，data，timeout。

如果我们需要用到Cookie，只用这个opener是不能达到目的的，所以我们需要创建更一般的opener来实现对Cookie的设置

#### Cookielib

cookielib模块的主要作用是提供可存储cookie的对象，以便于与urllib模块配合使用来访问Internet资源。Cookielib模块非常强大，我们可以利用本模块的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。

## URLError

首先解释下URLError可能产生的原因：

1、网络无连接，即本机无法上网

2、连接不到特定的服务器

3、服务器不存在

在代码中，我们需要用try-except语句来包围并捕获相应的异常,代码如下：



我们利用了 urlopen方法访问了一个不存在的网址，运行结果如下：

**[Errno 11004] getaddrinfo failed**