

Day03回顾

目前反爬总结

■ 基于User-Agent反爬

```
1 1、发送请求携带请求头: headers={'User-Agent' : 'Mozilla/5.0 xxxxxx'}
2 2、多个请求随机切换User-Agent
3 1、定义列表存放大量User-Agent, 使用random.choice()每次随机选择
4 2、定义py文件存放大量User-Agent, 使用random.choice()每次随机选择
5 3、使用fake_useragent每次访问随机生成User-Agent
6 * from fake_useragent import UserAgent
7 * ua = UserAgent()
8 * user_agent = ua.random
9 * print(user_agent)
```

■ 响应内容前端做处理反爬

```
1 1、html页面中可匹配出内容, 程序中匹配结果为空
2 * 响应内容中嵌入js, 对页面结构做了一定调整导致, 通过查看网页源代码, 格式化输出查看结构, 更改xpath或者正则测试
3 2、如果数据出不来可考虑更换 IE 的User-Agent尝试, 数据返回最标准
```

■ 基于IP反爬

```
1 控制爬取速度, 每爬取页面后随机休眠一定时间, 再继续爬取下一个页面
```

请求模块总结

■ urllib库使用流程

```

1  # 编码
2  params = {
3      '': '',
4      '': ''
5  }
6  params = urllib.parse.urlencode(params)
7  url = baseurl + params
8
9  # 请求
10 request = urllib.request.Request(url, headers=headers)
11 response = urllib.request.urlopen(request)
12 html = response.read().decode('utf-8')

```

■ requests模块使用流程

```

1  url = baseurl + urllib.parse.urlencode({dict})
2  html = requests.get(url, headers=headers).text

```

解析模块总结

■ 正则解析re模块

```

1  import re
2
3  pattern = re.compile('正则表达式', re.S)
4  r_list = pattern.findall(html)

```

■ lxml解析库

```

1  from lxml import etree
2
3  parse_html = etree.HTML(res.text)
4  r_list = parse_html.xpath('xpath表达式')

```

xpath表达式

■ 匹配规则

```

1  1、节点对象列表
2  # xpath示例: //div、//div[@class="student"]、//div/a[@title="stu"]/span
3  2、字符串列表
4  # xpath表达式中末尾为: @src、@href、text()

```

■ xpath高级

```
1 1、基准xpath表达式：得到节点对象列表
2 2、for r in [节点对象列表]:
3     username = r.xpath('./xxxxxx') # 此处注意遍历后继续xpath一定要以：. 开头，代表当前节点
```

Day04笔记

requests.get()参数

查询参数-params

■ 参数类型

```
1 字典,字典中键值对作为查询参数
```

■ 使用方法

```
1 1、res = requests.get(url,params=params,headers=headers)
2 2、特点:
3     * url为基准的url地址, 不包含查询参数
4     * 该方法会自动对params字典编码,然后和url拼接
```

■ 示例

```
1 import requests
2
3 baseurl = 'http://tieba.baidu.com/f?'
4 params = {
5     'kw' : '赵丽颖吧',
6     'pn' : '50'
7 }
8 headers = {'User-Agent' : 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
9             Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center
10            PC 6.0; .NET4.0C; InfoPath.3)'}
11 # 自动对params进行编码,然后自动和url进行拼接,去发请求
12 res = requests.get(baseurl,params=params,headers=headers)
13 res.encoding = 'utf-8'
14 print(res.text)
```

练习

把课程第1天中的 百度贴吧 抓取案例改为使用requests模块的params参数实现?

```
1 # 1、首先改为requests模块
2 import requests
3 def get_page(self,url):
4     html = requests.get(url,headers=self.headers).content.decode('utf-8')
```

```

5         return html
6     # 2、改为params参数 (无须编码, 无须拼接)
7     def get_page(self, params):
8         html = requests.get(self.url, params=params, headers=self.headers).content.decode('utf-8')
9
10    def main(self):
11        ... ..
12        for page in range(begin, end + 1):
13            pn = (page - 1) * 50
14            params = {
15                'kw': name,
16                'pn': str(pn)
17            }
18
19            # 发请求获取响应内容
20            html = self.get_page(params=params)

```

Web 客户端验证 参数-auth

■ 作用及类型

- 1 1、针对于需要web客户端用户名密码认证的网站
- 2 2、auth = ('username', 'password')

■ 达内code课程方向案例

```

1 import requests
2 import re
3
4 class NoteSpider(object):
5     def __init__(self):
6         self.url = 'http://code.tarena.com.cn/'
7         self.headers = {'User-Agent': 'Mozilla/5.0'}
8         self.auth = ('tarenacode', 'code_2013')
9
10    # 获取+解析
11    def get_parse_page(self):
12        res = requests.get(
13            url=self.url,
14            auth=self.auth,
15            headers=self.headers
16        )
17        res.encoding = 'utf-8'
18        html = res.text
19        # 解析
20        p = re.compile('<a href=.*?>(.*?)</a>', re.S)
21        r_list = p.findall(html)
22        # r_list : ['..', 'AIDCode', 'ACCCode']
23        for r in r_list:
24            if r != '..':
25                print({ '课程方向' : r })
26

```

```
27 if __name__ == '__main__':
28     spider = NoteSpider()
29     spider.get_parse_page()
```

思考：爬取具体的笔记文件？（把今天的笔记抓取下来）

```
1 import requests
2
3 # 定义常用变量
4 url = 'http://code.tarena.com.cn/AIDCode/aid1903/12-spider/spider_day03_note.zip'
5 headers = {
6     'User-Agent' : 'Mozilla/5.0'
7 }
8 auth = ('tarenacode', 'code_2013')
9
10 # 发请求获取响应内容(bytes)
11 html = requests.get(url, headers=headers, auth=auth).content
12
13 # 将响应内容保存到本地
14 filename = url.split('/')[-1]
15 with open(filename, 'wb') as f:
16     f.write(html)
17     print('%s下载成功' % filename)
```

如何保存到指定目录？

```
1 import requests
2 import os
3
4 url = 'http://code.tarena.com.cn/AIDCode/aid1903/12-spider/spider_day03_note.zip'
5 headers = {
6     'User-Agent' : 'Mozilla/5.0'
7 }
8 auth = ('tarenacode', 'code_2013')
9 print(222)
10 html = requests.get(url, headers=headers, auth=auth).content
11 print(1111)
12 directory = '/home/tarena/' + '/'.join(url.split('/')[3:-1]) + '/'
13
14 print(directory)
15 if not os.path.exists(directory):
16     os.makedirs(directory)
17
18 filename = directory + url.split('/')[-1]
19 print(filename)
20 with open(filename, 'wb') as f:
21     f.write(html)
22     print('%s下载成功' % filename)
```

SSL证书认证参数-verify

■ 适用网站及场景

- 1、适用网站: https类型网站但是没有经过 证书认证机构 认证的网站
- 2、适用场景: 抛出 `SSLError` 异常则考虑使用此参数

■ 参数类型

```
1 1、verify=True(默认)    : 检查证书认证
2 2、verify=False (常用) : 忽略证书认证
3 # 示例
4 response = requests.get(
5     url=url,
6     params=params,
7     headers=headers,
8     verify=False
9 )
```

代理参数-proxies

■ 定义

- 1、定义: 代替你原来的IP地址去对接网络的IP地址。
- 2、作用: 隐藏自身真实IP, 避免被封。

■ 普通代理

获取代理IP网站

- 1 西刺代理、快代理、全网代理、代理精灵、... ..

参数类型

```
1 1、语法结构
2     proxies = {
3         '协议': '协议://IP:端口号'
4     }
5 2、示例
6     proxies = {
7         'http': 'http://IP:端口号',
8         'https': 'https://IP:端口号'
9     }
```

示例

1. 使用免费普通代理IP访问测试网站: <http://httpbin.org/get>

```

1 import requests
2
3 url = 'http://httpbin.org/get'
4 headers = {
5     'User-Agent': 'Mozilla/5.0'
6 }
7 # 定义代理,在代理IP网站中查找免费代理IP
8 proxies = {
9     'http': 'http://115.171.85.221:9000',
10    'https': 'https://115.171.85.221:9000'
11 }
12 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
13 print(html)

```

2. 思考: 建立一个自己的代理IP池, 随时更新用来抓取网站数据

fake_useragent使用示例

```

1 # 随机生成1个User-Agent
2 from fake_useragent import UserAgent
3
4 ua = UserAgent()
5 print(ua.random)

```

建立自己的IP代理池

```

1 import requests
2 import random
3 from lxml import etree
4 from fake_useragent import UserAgent
5
6
7 # 生成随机的User-Agent
8 def get_random_ua():
9     # 创建User-Agent对象
10    ua = UserAgent()
11    # 随机生成1个User-Agent
12    return ua.random
13
14
15 url = 'http://httpbin.org/get'
16
17
18 # 从西刺代理网站上获取随机的代理IP
19 def get_ip_list():
20    headers = {'User-Agent': get_random_ua()}
21    # 访问西刺代理网站国内高匿代理, 找到所有的tr节点对象
22    res = requests.get('https://www.xicidaili.com/nn/', headers=headers)
23    parse_html = etree.HTML(res.text)
24    # 基准xpath, 匹配每个代理IP的节点对象列表
25    ipobj_list = parse_html.xpath('//tr')
26    # 定义空列表, 获取网页中所有代理IP地址及端口号
27    ip_list = []
28    # 从列表中第2个元素开始遍历, 因为第1个为: 字段名 (国家、IP、... ...)
29    for ip in ipobj_list[1:]:

```

```

30     ip_info = ip.xpath('./td[2]/text()')[0]
31     port_info = ip.xpath('./td[3]/text()')[0]
32     ip_list.append(
33         {
34             'http': 'http://' + ip_info + ':' + port_info,
35             'https': 'https://' + ip_info + ':' + port_info
36         }
37     )
38     # 返回代理IP及代理池 (列表ip_list)
39     return ip_list
40
41
42 # 主程序寻找测试可用代理
43 def main_print():
44     # 获取抓取的所有代理IP
45     ip_list = get_ip_list()
46     # 将不能使用的代理删除
47     for proxy_ip in ip_list:
48         try:
49             # 设置超时时间, 如果代理不能使用则切换下一个
50             headers = {'User-Agent': get_random_ua()}
51             res = requests.get(url=url, headers=headers, proxies=proxy_ip, timeout=5)
52             res.encoding = 'utf-8'
53             print(res.text)
54
55         except Exception as e:
56             # 此代理IP不能使用, 从代理池中移除
57             ip_list.remove(proxy_ip)
58             print('%s不能用, 已经移除' % proxy_ip)
59             # 继续循环获取最后1个代理IP
60             continue
61
62     # 将可用代理保存到本地文件
63     with open('proxies.txt', 'a') as f:
64         for ip in ip_list:
65             f.write(ip + '\n')
66
67 if __name__ == '__main__':
68     main_print()

```

2、写一个获取收费开放代理的接口

```

1 # getip.py
2 # 获取开放代理的接口
3 import requests
4
5 # 提取代理IP
6 def get_ip_list():
7     api_url = 'http://dev.kdlapi.com/api/getproxy/?'
8     orderid=996140620552954&num=100&protocol=2&method=2&an_an=1&an_ha=1&sep=1'
9     res = requests.get(api_url)
10    ip_port_list = res.text.split('\r\n')
11
12    return ip_port_list
13
14 if __name__ == '__main__':

```



```
14 proxy_ip_list = get_ip_list()
15 print(proxy_ip_list)
```

3、使用收费开放代理IP访问测试网站: <http://httpbin.org/get>

- 1、从代理网站上获取购买的普通代理的api链接
- 2、从api链接中提取出IP
- 3、随机选择代理IP访问网站进行数据抓取

```
1 from getip import *
2 import time
3 import random
4
5 url = 'http://httpbin.org/get'
6 headers = {'User-Agent' : 'Mozilla/5.0'}
7 proxy_ip_list = get_ip_list()
8
9 while True:
10     # 判断是否还有可用代理
11     if not proxy_ip_list:
12         proxy_ip_list = get_ip_list()
13
14     proxy_ip = random.choice(proxy_ip_list)
15     proxies = {
16         'http' : 'http://{}/'.format(proxy_ip),
17         'https' : 'https://{}/'.format(proxy_ip)
18     }
19     print(proxies)
20
21     try:
22         html =
requests.get(url=url,proxies=proxies,headers=headers,timeout=5,verify=False).text
23         print(html)
24         break
25     except:
26         print('正在更换代理IP, 请稍后... ..')
27         # 及时把不可用的代理IP移除
28         proxy_ip_list.remove(proxy_ip)
29         continue
```

■ 私密代理

语法格式

```
1 1、语法结构
2 proxies = {
3     '协议': '协议://用户名:密码@IP:端口号'
4 }
5
6 2、示例
7 proxies = {
8     'http': 'http://用户名:密码@IP:端口号',
9     'https': 'https://用户名:密码@IP:端口号'
10 }
```

示例代码

```
1 import requests
2 url = 'http://httpbin.org/get'
3 proxies = {
4     'http': 'http://309435365:szayclhp@122.114.67.136:16819',
5     'https': 'https://309435365:szayclhp@122.114.67.136:16819',
6 }
7 headers = {
8     'User-Agent' : 'Mozilla/5.0',
9 }
10
11 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
12 print(html)
```

requests.post()

■ 适用场景^a

```
1 | Post类型请求的网站
```

■ 参数-data

```
1 response = requests.post(url,data=data,headers=headers)
2 # data : post数据 (Form表单数据-字典格式)
```

■ 请求方式的特点

```
1 # 一般
2 GET请求 : 参数在URL地址中有显示
3 POST请求: Form表单提交数据
```

有道翻译破解案例(post)

1. 目标

```
1 破解有道翻译接口, 抓取翻译结果
2 # 结果展示
3 请输入要翻译的词语: elephant
4 翻译结果: 大象
5 *****
6 请输入要翻译的词语: 喵喵叫
7 翻译结果: mews
```

2. 实现步骤

- 1 1、浏览器F12开启网络抓包,Network-All,页面翻译单词后找Form表单数据
- 2 2、在页面中多翻译几个单词,观察Form表单数据变化(有数据是加密字符串)
- 3 3、刷新有道翻译页面,抓取并分析JS代码(本地JS加密)
- 4 4、找到JS加密算法,用Python按同样方式加密生成加密数据
- 5 5、将Form表单数据处理为字典,通过requests.post()的data参数发送

具体实现

- 1、开启F12抓包,找到Form表单数据如下:

```
1 i: 喵喵叫
2 from: AUTO
3 to: AUTO
4 smartresult: dict
5 client: fanyideskweb
6 salt: 15614112641250
7 sign: 94008208919faa19bd531acde36aac5d
8 ts: 1561411264125
9 bv: f4d62a2579ebb44874d7ef93ba47e822
10 doctype: json
11 version: 2.1
12 keyfrom: fanyi.web
13 action: FY_BY_REALTIME
```

- 2、在页面中多翻译几个单词,观察Form表单数据变化

```
1 salt: 15614112641250
2 sign: 94008208919faa19bd531acde36aac5d
3 ts: 1561411264125
4 bv: f4d62a2579ebb44874d7ef93ba47e822
5 # 但是bv的值不变
```

- 3、一般为本地js文件加密,刷新页面,找到js文件并分析JS代码

```
1 # 方法1
2 Network - JS选项 - 搜索关键词salt
3 # 方法2
4 控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出
5
6 # 最终找到相关JS文件 : fanyi.min.js
```

- 4、打开JS文件,分析加密算法,用Python实现

```
1 # ts : 经过分析为13位的时间戳,字符串类型
2 js代码实现: "" + (new Date).getTime()
3 python实现: str(int(time.time()*1000))
4
5 # salt
6 js代码实现: r+parseInt(10 * Math.random(), 10);
7 python实现: ts + str(random.randint(0,9))
8
9 # sign (设置断点调试,来查看 e 的值,发现 e 为要翻译的单词)
10 js代码实现: n.md5("fanyideskweb" + e + salt + "n%-rKaT5fb[Gy?;N5@Tj")
```

```

11 python实现:
12 from hashlib import md5
13 s = md5()
14 s.update("fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj".encode())
15 sign = s.hexdigest()

```

■ 5、代码实现

```

1  import requests
2  import time
3  from hashlib import md5
4  import random
5
6  # 获取相关加密算法的结果
7  def get_salt_sign_ts(word):
8      # salt
9      salt = str(int(time.time()*1000)) + str(random.randint(0,9))
10     # sign
11     string = "fanyideskweb" + word + salt + "n%A-rKaT5fb[Gy?;N5@Tj"
12     s = md5()
13     s.update(string.encode())
14     sign = s.hexdigest()
15     # ts
16     ts = str(int(time.time()*1000))
17     return salt,sign,ts
18
19 # 攻克有道
20 def attack_yd(word):
21     salt,sign,ts = get_salt_sign_ts(word)
22     # url为抓包抓到的地址 F12 -> translate_o -> post
23     url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
24     headers = {
25         "Accept": "application/json, text/javascript, */*; q=0.01",
26         "Accept-Encoding": "gzip, deflate",
27         "Accept-Language": "zh-CN,zh;q=0.9",
28         "Connection": "keep-alive",
29         "Content-Length": "238",
30         "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
31         "Cookie": "OUTFOX_SEARCH_USER_ID=-1449945727@10.169.0.82; OUTFOX_SEARCH_USER_ID_NCOO=1492587933.976261; JSESSIONID=aaa5_Lj5jzfQZ_IPPuaSw; __rl__test__cookies=1559193524685",
32         "Host": "fanyi.youdao.com",
33         "Origin": "http://fanyi.youdao.com",
34         "Referer": "http://fanyi.youdao.com/",
35         "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36",
36         "X-Requested-With": "XMLHttpRequest",
37     }
38     # Form表单数据
39     data = {
40         'i': word,
41         'from': 'AUTO',
42         'to': 'AUTO',
43         'smartresult': 'dict',
44         'client': 'fanyideskweb',
45         'salt': salt,

```

```
46         'sign': sign,
47         'ts': ts,
48         'bv': 'cf156b581152bd0b259b90070b1120e6',
49         'doctype': 'json',
50         'version': '2.1',
51         'keyfrom': 'fanyi.web',
52         'action': 'FY_BY_REALTIME'
53     }
54
55     json_html = requests.post(url,data=data,headers=headers).json()
56     result = json_html['translateResult'][0][0]['tgt']
57     return result
58
59 if __name__ == '__main__':
60     word = input('请输入要翻译的单词: ')
61     result = attack_yd(word)
62     print(result)
```

今日作业

- 1、仔细复习并总结有道翻译案例，抓包流程，代码实现
- 2、通过百度翻译，来再次熟练抓包流程，分析，断点调试等操作