

Day04

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模块每次访问随机生成User-Agent
6 # sudo pip3 install fake_useragent
7
8 * from fake_useragent import UserAgent
9 * ua = UserAgent()
10 * user_agent = ua.random
11 * print(user_agent)
```

■ 响应内容前端JS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
```

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  baseurl = 'http://tieba.baidu.com/f?'
2  html = requests.get(url, headers=headers).content.decode('utf-8', 'ignore')

```

■ 响应对象res属性

```

1  res.text : 字符串
2  res.content : bytes
3  res.encoding: 字符编码 res.encoding='utf-8'
4  res.status_code : HTTP响应码
5  res.url : 实际数据URL地址

```

解析模块总结

■ 正则解析re模块

```

1  import re
2
3  pattern = re.compile(r'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')
4
5 # 此处注意遍历后继续xpath一定要以: . 开头, 代表当前节点
```

写程序注意

```
1 # 最终目标: 不要使你的程序因为任何异常而终止
2 1、页面请求设置超时时间,并用try捕捉异常,超过指定次数则更换下一个URL地址
3 2、所抓取任何数据,获取具体数据前先判断是否存在该数据,可使用列表推导式
4 # 多级页面数据抓取注意
5 1、主线函数: 解析一级页面函数(将所有数据从一级页面中解析并抓取)
```

增量爬虫如何实现

```
1 1、数据库中创建指纹表,用来存储每个请求的指纹
2 2、在抓取之前,先到指纹表中确认是否之前抓取过
```

Chrome浏览器安装插件

■ 安装方法

```
1 # 在线安装
2 1、下载插件 - google访问助手
3 2、安装插件 - google访问助手: Chrome浏览器-设置-更多工具-扩展程序-开发者模式-拖拽(解压后的插件)
4 3、在线安装其他插件 - 打开google访问助手 - google应用商店 - 搜索插件 - 添加即可
5
6 # 离线安装
7 1、下载插件 - xxx.crx 重命名为 xxx.zip
8 2、输入地址: chrome://extensions/ 打开- 开发者模式
9 3、拖拽 插件(或者解压后文件夹) 到浏览器中
10 4、重启浏览器,使插件生效
```

Day04笔记

链家二手房案例 (xpath)

实现步骤

- 确定是否为静态

```
1 打开二手房页面 -> 查看网页源码 -> 搜索关键字
```

- xpath表达式

```
1 1、基准xpath表达式(匹配每个房源信息节点列表)
2 此处滚动鼠标滑轮时,li节点的class属性值会发生变化,通过查看网页源码确定xpath表达式
3 //ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]
4
5 2、依次遍历后每个房源信息xpath表达式
6 * 名称: '//*[@data-el="region"]/text()'
7
8 # 户型+面积+方位+是否精装
9 info_list = '//*[@class="houseInfo"]/text()' [0].strip().split('|')
10 * 户型: info_list[1]
11 * 面积: info_list[2]
12 * 方位: info_list[3]
13 * 精装: info_list[4]
14
15
16 * 楼层: '//*[@class="positionInfo"]/text()'
17 * 区域: '//*[@class="positionInfo"]/a/text()'
18 * 总价: '//*[@class="totalPrice"]/span/text()'
19 * 单价: '//*[@class="unitPrice"]/span/text()'
```

代码实现

```
1 import requests
2 from lxml import etree
3 import time
4 import random
5 from useragents import ua_list
6
7 class LianjiaSpider(object):
8     def __init__(self):
9         self.url='https://bj.lianjia.com/ershoufang/pg{}/'
10        self.blog = 1
11
12    def get_html(self,url):
13        headers = {'User-Agent':random.choice(ua_list)}
14        # 尝试3次,否则换下一页地址
15        if self.blog <= 3:
16            try:
17                res = requests.get(url=url,headers=headers,timeout=5)
18                res.encoding = 'utf-8'
19                html = res.text
20                # 直接调用解析函数
21                self.parse_page(html)
22            except Exception as e:
```

```

23         print('Retry')
24         self.blog += 1
25         self.get_html(url)
26
27
28     def parse_page(self,html):
29         parse_html = etree.HTML(html)
30         # li_list: [<element li at xxx>,<element li at xxx>]
31         li_list = parse_html.xpath('//ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]')
32         item = {}
33         for li in li_list:
34             # 名称
35             xpath_name = '.*//a[@data-el="region"]/text()'
36             name_list = li.xpath(xpath_name)
37             item['name'] = name_list[0].strip() if name_list else None
38             # 户型+面积+方位+是否精装
39             info_xpath = '.*//div[@class="houseInfo"]/text()'
40             info_list = li.xpath(info_xpath)
41             if info_list:
42                 info_list = info_list[0].strip().split('|')
43                 if len(info_list) == 5:
44                     item['model'] = info_list[1].strip()
45                     item['area'] = info_list[2].strip()
46                     item['direction'] = info_list[3].strip()
47                     item['perfect'] = info_list[4].strip()
48                 else:
49                     item['model']=item['area']=item['direction']=item['perfect']=None
50             else:
51                 item['model'] = item['area'] = item['direction'] = item['perfect'] = None
52
53             # 楼层
54             xpath_floor = '.*//div[@class="positionInfo"]/text()'
55             floor_list = li.xpath(xpath_floor)
56             item['floor'] = floor_list[0].strip().split()[0] if floor_list else None
57
58             # 地区
59             xpath_address = '.*//div[@class="positionInfo"]/a/text()'
60             address_list = li.xpath(xpath_address)
61             item['address'] = address_list[0].strip() if address_list else None
62
63             # 总价
64             xpath_total = '.*//div[@class="totalPrice"]/span/text()'
65             total_list = li.xpath(xpath_total)
66             item['total_price'] = total_list[0].strip() if total_list else None
67
68             # 单价
69             xpath_unit = '.*//div[@class="unitPrice"]/span/text()'
70             unit_list = li.xpath(xpath_unit)
71             item['unit_price'] = unit_list[0].strip() if unit_list else None
72
73
74         print(item)
75
76     def main(self):
77         for pg in range(1,101):
78             url = self.url.format(pg)

```

```

79         self.get_html(url)
80         time.sleep(random.randint(1,3))
81         # 对self.blog进行一下初始化
82         self.blog = 1
83
84
85 if __name__ == '__main__':
86     start = time.time()
87     spider = LianjiaSpider()
88     spider.main()
89     end = time.time()
90     print('执行时间:%.2f' % (end-start))

```

百度贴吧图片抓取

目标思路

■ 目标

1 抓取指定贴吧所有图片

■ 思路

```

1 1、获取贴吧主页URL,下一页,找到不同页的URL规律
2 2、获取1页中所有帖子URL地址: [帖子链接1,帖子链接2,...]
3 3、对每个帖子链接发请求,获取图片URL
4 4、向图片的URL发请求,以wb方式写入本地文件

```

实现步骤

■ 贴吧URL规律

1 <http://tieba.baidu.com/f?kw=??&pn=50>

■ xpath表达式

```

1 1、帖子链接xpath
2 //div[@class="t_con clearfix"]/div/div/div/a/@href
3
4 2、图片链接xpath
5 //div[@class="d_post_content j_d_post_content clearfix"]/img[@class="BDE_Image"]/@src
6
7 3、视频链接xpath
8 //div[@class="video_src_wrapper"]/embed/@data-video
9 # 注意: 此处视频链接前端对响应内容做了处理,需要查看网页源代码来查看,复制HTML代码在线格式化

```

代码实现

```

1 import requests

```

```

2  from lxml import etree
3  import random
4  import time
5  from useragents import ua_list
6  from urllib import parse
7  import os
8
9  class BaiduImageSpider(object):
10     def __init__(self):
11         self.url = 'http://tieba.baidu.com/f?kw={}&pn={}'
12
13     # 获取html功能函数
14     def get_html(self,url):
15         html = requests.get(
16             url=url,
17             headers={'User-Agent': 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)'}
18         ).content.decode('utf-8', 'ignore')
19         return html
20
21     # 解析html功能函数
22     def xpath_func(self,html,xpath_bds):
23         parse_html = etree.HTML(html)
24         r_list = parse_html.xpath(xpath_bds)
25         return r_list
26
27     # 解析函数 - 实现最终图片抓取
28     def parse_html(self,one_url):
29         html = self.get_html(one_url)
30         # 准备提取帖子链接:xpath_list [' /p/32323', '', '' ]
31         xpath_bds = ' //div[@class="t_con cleafix"]/div/div/div/a/@href'
32         r_list = self.xpath_func(html,xpath_bds)
33         for r in r_list:
34             # 拼接帖子的URL地址
35             t_url = 'http://tieba.baidu.com' + r
36             # 把帖子中所有图片保存到本地
37             self.get_image(t_url)
38             # 爬完1个帖子中所有图片,休眠0-2秒钟
39             time.sleep(random.uniform(0,2))
40
41     # 功能:给定1个帖子URL,把帖子中所有图片保存到本地
42     def get_image(self,t_url):
43         html = self.get_html(t_url)
44         # 图片链接的xpath表达式:img_list ['http://xxx.jpg', '']
45         # 使用xpath表达式的或| : 图片链接 + 视频链接
46         xpath_bds = ' //div[@class="d_post_content j_d_post_content clearfix"]/img[@class="BDE_Image"]/@src | //div[@class="video_src_wrapper"]/embed/@data-video'
47         img_list = self.xpath_func(html,xpath_bds)
48         for img in img_list:
49             html_bytes = requests.get(
50                 url=img,
51                 headers={'User-Agent': random.choice(ua_list)}
52             ).content
53             self.save_img(html_bytes,img)
54
55     # 保存图片函数

```

```

56     def save_img(self,html_bytes,img):
57         filename = img[-10:]
58         with open(filename,'wb') as f:
59             f.write(html_bytes)
60             print('%s下载成功' % filename)
61
62     # 主函数
63     def main(self):
64         name = input('请输入贴吧名:')
65         begin = int(input('请输入起始页:'))
66         end = int(input('请输入终止页:'))
67
68         # 对贴吧名进行编码
69         kw = parse.quote(name)
70         for page in range(begin,end+1):
71             pn = (page-1)*50
72             url = self.url.format(kw,pn)
73             # 调用主线函数
74             self.parse_html(url)
75
76 if __name__ == '__main__':
77     spider = BaiduImageSpider()
78     spider.main()

```

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'}
9 # 自动对params进行编码,然后自动和url进行拼接,去发请求
10 res = requests.get(url=baseurl,params=params,headers=headers)
11 res.encoding = 'utf-8'
12 print(res.text)

```

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 # xpath表达式
2 //a/@href
3 # url
4 http://code.tarena.com.cn/AIDCode/aid1904/14-redis/

```

思考：爬取具体的笔记文件？

```

1 import os
2
3 # 保存在: /home/tarena/redis
4 # 先判断 /home/tarena/redis 是否存在
5 1、不存在: 先创建目录,然后再保存 .zip
6 2、存在: 直接保存 .zip
7
8 # 使用频率很高
9 if not os.path.exists('路径'):
10     os.makedirs('路径')

```

代码实现

```

1 import requests
2 from lxml import etree
3 import random
4 from useragents import ua_list
5 import os
6
7 class CodeSpider(object):
8     def __init__(self):
9         self.url = 'http://code.tarena.com.cn/AIDCode/aid1906/13Redis/'
10        self.auth = ('tarenacode','code_2013')

```

```

11
12 def get_headers(self):
13     headers = {'User-Agent':random.choice(ua_list)}
14     return headers
15
16 def get_html(self,url):
17     res = requests.get(url,headers=self.get_headers(),auth=self.auth)
18     html = res.content
19     return html
20
21 def parse_html(self):
22     # 获取响应内容
23     html = self.get_html(self.url).decode()
24     # 解析
25     parse_html = etree.HTML(html)
26     # r_list: ['../','day01/','redis-xxx.zip']
27     r_list = parse_html.xpath('//a/@href')
28
29     directory = '/home/tarena/myredis/'
30     if not os.path.exists(directory):
31         os.makedirs(directory)
32
33     for r in r_list:
34         if r.endswith('.zip') or r.endswith('.rar'):
35             self.save_files(r,directory)
36
37 def save_files(self,r,directory):
38     # 拼接地址,把zip文件保存到指定目录
39     url = self.url + r
40     # filename: /home/tarena/AID/redis/xxx.zip
41     filename = directory + r
42     html = self.get_html(url)
43
44     with open(filename,'wb') as f:
45         f.write(html)
46         print('%s下载成功' % r)
47
48
49 if __name__ == '__main__':
50     spider = CodeSpider()
51     spider.parse_html()

```

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

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://112.85.164.220:9999',
10    'https':'https://112.85.164.220:9999'
11 }
12 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
13 print(html)
```

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

- 1 1、从西刺代理IP网站上,抓取免费代理IP
- 2 2、测试抓取的IP,可用的保存在文件中

思考 - 代码实现

```

1 import requests
2 from lxml import etree
3 import time
4 import random
5 from fake_useragent import UserAgent
6
7 class GetProxyIP(object):
8     def __init__(self):
9         self.url = 'https://www.xicidaili.com/nn/{}'.format(
10
11         # 随机生成1个User-Agent
12     def get_headers(self):
13         ua = UserAgent()
14         useragent = ua.random
15         headers = {'User-Agent': useragent}
16         return headers
17
18     # 获取可用代理IP文件
19     def get_ip_file(self, url):
20         headers = self.get_headers()
21         html = requests.get(url=url, headers=headers, timeout=5).text
22
23
24         parse_html = etree.HTML(html)
25         tr_list = parse_html.xpath('//tr')
26         for tr in tr_list[1:]:
27             ip = tr.xpath('./td[2]/text()')[0]
28             port = tr.xpath('./td[3]/text()')[0]
29             # 测试ip:port是否可用
30             self.test_ip(ip, port)
31
32     def test_ip(self, ip, port):
33         proxies = {
34             'http': 'http://{}:{:}'.format(ip, port),
35             'https': 'https://{}:{:}'.format(ip, port),
36         }
37         test_url = 'http://www.baidu.com/'
38         try:
39             res = requests.get(url=test_url, proxies=proxies, timeout=8)
40             if res.status_code == 200:
41                 print(ip, port, 'Success')
42                 with open('proxies.txt', 'a') as f:
43                     f.write(ip + ':' + port + '\n')
44             except Exception as e:
45                 print(ip, port, 'Failed')
46
47     # 主函数
48     def main(self):

```

```

49     for i in range(1,1001):
50         url = self.url.format(i)
51         self.get_ip_file(url)
52         time.sleep(random.randint(5,10))
53
54 if __name__ == '__main__':
55     spider = GetProxyIP()
56     spider.main()

```

写一个获取收费开放代理的接口

```

1  # 获取开放代理的接口
2  import requests
3
4  def test_ip(ip):
5      url = 'http://www.baidu.com/'
6      proxies = {
7          'http': 'http://{0}'.format(ip),
8          'https': 'https://{0}'.format(ip),
9      }
10
11     try:
12         res = requests.get(url=url,proxies=proxies,timeout=8 )
13         if res.status_code == 200:
14             return True
15     except Exception as e:
16         return False
17
18 # 提取代理IP
19 def get_ip_list():
20     api_url = 'http://dev.kdlapi.com/api/getproxy/?
21     orderid=946562662041898&num=100&protocol=1&method=2&an_an=1&an_ha=1&sep=2'
22     html = requests.get(api_url).content.decode('utf-8','ignore')
23     # ip_port_list: ['IP:PORT','IP:PORT','']
24     ip_port_list = html.split('\r\n')
25
26     # 依次遍历代理IP,并进行测试
27     for ip in ip_port_list:
28         with open('proxy_ip.txt','a') as f:
29             if test_ip(ip):
30                 f.write(ip + '\n')
31
32 if __name__ == '__main__':
33     get_ip_list()

```

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@106.75.71.140:16816',
5     'https': 'https://309435365:szayclhp@106.75.71.140:16816',
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)
```

今日作业

- 1、总结前几天内容,理顺知识点
- 2、代理参数 - 如何建立自己的IP代理池,并使用随机代理IP访问网站
- 3、Web客户端验证 - 如何下载、保存