

Day04回顾

requests.get()参数

```
1 1、url
2 2、params -> {} : 查询参数 Query String
3 3、proxies -> {}
4     proxies = {
5         'http': 'http://1.1.1.1:8888',
6         'https': 'https://1.1.1.1:8888'
7     }
8 4、auth -> ('tarenacode', 'code_2013')
9 5、verify -> True/False
10 6、timeout
```

requests.post()

```
1 data -> {} Form表单数据 : Form Data
```

控制台抓包

■ 打开方式及常用选项

```
1 1、打开浏览器，F12打开控制台，找到Network选项卡
2 2、控制台常用选项
3     1、Network：抓取网络数据包
4         1、ALL：抓取所有的网络数据包
5         2、XHR：抓取异步加载的网络数据包
6         3、JS：抓取所有的JS文件
7     2、Sources：格式化输出并打断点调试JavaScript代码，助于分析爬虫中一些参数
8     3、Console：交互模式，可对JavaScript中的代码进行测试
9 3、抓取具体网络数据包后
10     1、单击左侧网络数据包地址，进入数据包详情，查看右侧
11     2、右侧：
12         1、Headers：整个请求信息
13             General、Response Headers、Request Headers、Query String、Form Data
14         2、Preview：对响应内容进行预览
15         3、Response：响应内容
16         4、Cookies：请求及响应中Cookies信息
```

■ 有道翻译过程梳理

- 1 1. 打开首页
- 2 2. 准备抓包: F12开启控制台
- 3 3. 寻找地址
- 4 页面中输入翻译单词, 控制台中抓取到网络数据包, 查找并分析返回翻译数据的地址
- 5 4. 发现规律
- 6 找到返回具体数据的地址, 在页面中多输入几个单词, 找到对应URL地址, 分析对比 Network - All(或者XHR) - Form Data, 发现对应的规律
- 7 5. 寻找JS文件
- 8 右上角 ... -> Search -> 搜索关键字 -> 单击 -> 跳转到Sources, 左下角格式化符号{}
- 9 6. 查看JS代码
- 10 搜索关键字, 找到相关加密方法
- 11 7. 断点调试
- 12 8. 完善程序

常见的反爬机制及处理方式

- 1 1. Headers反爬虫 : Cookie、Referer、User-Agent
- 2 解决方案: 通过F12获取headers, 传给requests.get()方法
- 3
- 4 2. IP限制 : 网站根据IP地址访问频率进行反爬, 短时间内进制IP访问
- 5 解决方案:
- 6 1、构造自己IP代理池, 每次访问随机选择代理, 经常更新代理池
- 7 2、购买开放代理或私密代理IP
- 8 3、降低爬取的速度
- 9
- 10 3. User-Agent限制 : 类似于IP限制
- 11 解决方案: 构造自己的User-Agent池, 每次访问随机选择
- 12
- 13 4. Ajax动态加载 : 从url加载网页的源代码后, 会在浏览器执行JavaScript程序, 这些程序会加载更多内容
- 14 解决方案: F12或抓包工具抓包处理
- 15
- 16 5. 对查询参数加密
- 17 解决发难: 找到JS文件, 分析加密算法, 用Python实现加密执行JS文件中的代码, 返回加密数据
- 18
- 19 6. 对响应内容做处理
- 20 解决方案: 打印并查看响应内容, 用xpath或正则做处理

python中正则处理headers和formdata

- 1 1. pycharm进入方法 : Ctrl + r
- 2 2. 处理headers和formdata
- 3 (. *?): (. *)
- 4 "\$1": "\$2",
- 5 3. 点击 Replace All

Day05笔记

动态加载数据抓取-Ajax

■ 特点

- 1、右键 -> 查看网页源码中没有具体数据
- 2、滚动鼠标滑轮或其他动作时加载

■ 抓取

- 1、F12打开控制台，页面动作抓取网络数据包
- 2、抓取json文件URL地址
- 3、# 控制台中 XHR : 异步加载的数据包
- 4、# XHR -> Query String(查询参数)

豆瓣电影数据抓取案例

■ 目标

- 1、地址：豆瓣电影 - 排行榜 - 剧情
- 2、目标：电影名称、电影评分

■ F12抓包 (XHR)

- 1、Request URL(基准URL地址) : `https://movie.douban.com/j/chart/top_list?`
- 2、Query String(查询参数)
- 3、# 抓取的查询参数如下：
- 4、`type: 13`
- 5、`interval_id: 100:90`
- 6、`action: ''`
- 7、`start: 0`
- 8、`limit`: 用户输入的电影数量

■ json模块的使用

- 1、`json.loads(json格式的字符串)`: 把json格式的字符串转为python数据类型
- 2、# 示例
- 3、`html = json.loads(res.text)`
- 4、`print(type(html))`

■ 代码实现

```
1 import requests
2 import json
3 import pymysql
4
5 class DoubanSpider(object):
6     def __init__(self):
```

```

7         self.url = 'https://movie.douban.com/j/chart/top_list?'
8         self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36'}
9
10        # 获取页面
11        def get_page(self, params):
12            res = requests.get(
13                url=self.url,
14                params=params,
15                headers=self.headers,
16                verify=True
17            )
18            res.encoding = 'utf-8'
19            # json.loads() json格式->Python格式
20            html = res.json()
21            self.parse_page(html)
22
23        # 解析并保存数据
24        def parse_page(self, html):
25            # html为大列表 [{电影1信息},{},{}]
26            for h in html:
27                # 名称
28                name = h['title'].strip()
29                # 评分
30                score = float(h['score'].strip())
31                # 打印测试
32                print([name, score])
33
34        # 主函数
35        def main(self):
36            limit = input('请输入电影数量:')
37            params = {
38                'type' : '24',
39                'interval_id' : '100:90',
40                'action' : '',
41                'start' : '0',
42                'limit' : limit
43            }
44            # 调用函数,传递params参数
45            self.get_page(params)
46
47        if __name__ == '__main__':
48            spider = DoubanSpider()
49            spider.main()

```

思考: 实现用户在终端输入电影类型和电影数量, 将对应电影信息抓取到数据库

```

1 def main(self):
2     # 定义字典,来存放类型和对应的数字
3     ty_dict = {'剧情':'11','喜剧':'24','爱情':'13'}
4
5     ty = input('请输入电影类型(剧情|喜剧|爱情):')
6     if ty in ['剧情','喜剧','爱情']:
7         number = input('请输入电影数量:')
8         type_number = ty_dict[ty]
9         url = self.url.format(type_number,number)
10        self.get_page(url)
11    else:
12        print('输入有误')

```

练习: 腾讯招聘案例抓包看看?

■ URL地址及目标

1. 确定URL地址及目标

- 1 1、URL: 百度搜索腾讯招聘 - 查看工作岗位
- 2 2、目标: 职位名称、工作职责、岗位要求

2. F12抓包

3. 一级页面json地址(index变,timestamp未检查)

```

1 https://careers.tencent.com/tencentcareer/api/post/Query?
timestamp=1559294378106&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attr
Id=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn

```

4. 二级页面地址(postId在变,在一级页面中可拿到)

```

1 https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1559&postId=
{}&language=zh-cn

```

■ 具体代码实现

```

1 import requests
2 import json
3
4 class TencentSpider(object):
5     def __init__(self):
6         self.headers = {'User-Agent': 'Mozilla/5.0'}
7         self.one_url = 'https://careers.tencent.com/tencentcareer/api/post/Query?
timestamp=1559294378106&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attr
rId=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn'
8
9     def get_page(self,url):
10        res = requests.get(url,headers=self.headers)
11        res.encoding = 'utf-8'
12        # json.loads()把json格式的字符串转为python数据类型
13        html = json.loads(res.text)
14        return html

```

```

15
16 def parse_one_page(self,html):
17     for job in html['Data']['Posts']:
18         job_name = job['RecruitPostName']
19         job_address = job['LocationName']
20         # 拿postid为了拼接二级页面地址
21         post_id = job['PostId']
22         # 职责和要求(二级页面)
23         # 得到二级页面链接
24         two_url = 'https://careers.tencent.com/tencentcareer/api/post/ByPostId?
timestamp=1559&postId={}&language=zh-cn'.format(post_id)
25         # 发请求解析
26         job_duty,job_requirement = self.parse_two_page(two_url)
27         print(job_name,job_address,job_duty,job_requirement)
28
29 def parse_two_page(self,two_url):
30     html = self.get_page(two_url)
31     # 职责
32     job_duty = html['Data']['Responsibility']
33     # 要求
34     job_requirement = html['Data']['Requirement']
35
36     return job_duty,job_requirement
37
38 def main(self):
39     for index in range(1,11):
40         url = self.one_url.format(str(index))
41         one_html = self.get_page(url)
42         self.parse_one_page(one_html)
43
44 if __name__ == '__main__':
45     spider = TencentSpider()
46     spider.main()

```

cookie模拟登录

■ 适用网站及场景

1 抓取需要登录才能访问的页面

■ 方法一

```

1 1、先登录成功1次,获取到携带登陆信息的Cookie
2   F12打开控制台,在页面输入用户名、密码,登录成功,找到/home(一般在抓到地址的上面)
3 2、携带着cookie发请求
4   ** Cookie
5   ** Referer(源,代表你从哪里转过来的)
6   ** User-Agent

```

```

1 import requests
2 from lxml import etree

```

```

3
4 # url为需要登录才能正常访问的地址
5 url = 'http://www.renren.com/967469305/profile'
6 # headers中的cookie为登录成功后抓取到的cookie
7 headers = {
8     "Accept":
9     "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3",
10     "Accept-Encoding": "gzip, deflate",
11     "Accept-Language": "zh-CN,zh;q=0.9",
12     "Connection": "keep-alive",
13     "Cookie": "anonymid=jw8bjdsj-5k8o5d; _r01_=1; jebe_key=9090b79a-9190-45ea-bcea-90706ee7653b%7C2e9beece3ead42fe6a26739d515f14df%7C1559079271781%7C1%7C1559079272079; depovince=GW; jebecookies=480f3187-115d-49cd-be2b-0a3f5515759c||||; JSESSIONID=abcNH_AKnWCDqhRwolqUw; ick_login=806fe745-1051-4596-922a-6ac365aaadc4; _de=4DBCFC17D9E50C8C92BCDC45CC5C3B7; p=735cd53f5e5c75bf70822513dfe009075; first_login_flag=1; ln_uact=13603263409; ln_hurl=http://hdn.xnimg.cn/photos/hdn221/20181101/1550/h_main_qz3H_61ec0009c3901986.jpg; t=88fab410f3834cd6e7f97dd9717b564d5; societyguesser=88fab410f3834cd6e7f97dd9717b564d5; id=967469305; xnsid=6fd29138; loginfrom=syshome",
14     "Host": "www.renren.com",
15     "Referer": "http://www.renren.com/",
16     "Upgrade-Insecure-Requests": "1",
17     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36",
18 }
19 html = requests.get(url,headers=headers).text
20 # 解析
21 parse_html = etree.HTML(html)
22 result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
23 # result:就读于北京电影学院
24 print(result)

```

■ 方法二

1. 知识点

1 利用requests模块中的session会话保持功能

2. session会话使用流程

```

1 1、实例化session对象
2     session = requests.session()
3 2、让session对象发送get或者post请求
4     res = session.get(url,headers=headers)

```

3. 具体步骤

```
1 1、寻找登录时POST的地址
2 查看网页源码,查看form,找action对应的地址: http://www.renren.com/PLogin.do
3
4 2、发送用户名和密码信息到POST的地址
5 * 用户名和密码信息以什么方式发送? -- 字典
6 键 : <input>标签中name的值(email,password)
7 值 : 真实的用户名和密码
8 post_data = {'email':'','password':''}
```

4. 程序实现

```
1 整体思路
2 1、先POST: 把用户名和密码信息POST到某个地址中
3 2、再GET: 正常请求去获取页面信息
```

```
1 import requests
2 from lxml import etree
3
4 # 定义常用变量
5 post_url = 'http://www.renren.com/PLogin.do'
6 post_data = {
7     'email' : 'xxxxxx',
8     'password' : 'xxxxxx'
9 }
10 headers = {
11     'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
12     Gecko) Chrome/74.0.3729.169 Safari/537.36',
13     'Referer' : 'http://www.renren.com/SysHome.do'
14 }
15 # 实例化session会话保持对象
16 session = requests.session()
17 # 先POST,把用户名和密码信息POST到一个地址
18 session.post(post_url,data=post_data,headers=headers)
19
20 # 再get个人主页
21 url = 'http://www.renren.com/970294164/profile'
22 html = session.get(url,headers=headers).text
23 parse_html = etree.HTML(html)
24 result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
25 print(result)
```

selenium+phantomjs/Chrome/Firefox

selenium

■ 定义

```
1 1、Web自动化测试工具, 可运行在浏览器, 根据指令操作浏览器
2 2、只是工具, 必须与第三方浏览器结合使用
```


■ 安装

```
1 Linux: sudo pip3 install selenium
2 Windows: python -m pip install selenium
```

*phantomjs*浏览器

■ 定义

```
1 无界面浏览器(又称无头浏览器), 在内存中进行页面加载, 高效
```

■ 安装(phantomjs、chromedriver、geckodriver)

Windows

```
1 1、下载对应版本的phantomjs、chromedriver、geckodriver
2 2、把chromedriver.exe拷贝到python安装目录的Scripts目录下(添加到系统环境变量)
3 查看python安装路径: where python
4 3、验证
5 cmd命令行: chromedriver
6
7 # 下载地址
8 chromedriver : 下载对应版本
9 http://chromedriver.storage.googleapis.com/index.html
10
11 geckodriver
12 https://github.com/mozilla/geckodriver/releases
```

Linux

```
1 1、下载后解压
2 tar -zxvf geckodriver.tar.gz
3 2、拷贝解压后文件到 /usr/bin/ (添加环境变量)
4 sudo cp geckodriver /usr/bin/
```

■ 使用

示例代码一: 使用 selenium+浏览器 打开百度

```
1 from selenium import webdriver
2 import time
3
4 browser = webdriver.PhantomJS()
5 browser.get('http://www.baidu.com/')
6 browser.save_screenshot('baidu.png')
7 browser.quit
```

示例代码二: 打开百度, 搜索赵丽颖, 查看

```

1 from selenium import webdriver
2 import time
3
4 browser = webdriver.Chrome()
5 browser.get('http://www.baidu.com/')
6
7 # 向搜索框(id kw)输入 赵丽颖
8 ele = browser.find_element_by_xpath('//*[@id="kw"]')
9 ele.send_keys('赵丽颖')
10
11 time.sleep(1)
12 # 点击 百度一下 按钮(id su)
13 su = browser.find_element_by_xpath('//*[@id="su"]')
14 su.click()
15
16 # 截图
17 browser.save_screenshot('赵丽颖.png')
18 # 关闭浏览器
19 browser.quit()

```

■ 浏览器对象(browser)方法

```

1 1、 browser = webdriver.Firefox(executable_path='path')
2 2、 browser.get(url)
3 3、 browser.page_source # 查看响应内容
4 4、 browser.page_source.find('字符串')
5   # 从html源码中搜索指定字符串,没有找到返回: -1
6 5、 browser.quit() # 关闭浏览器

```

■ 定位节点

单元素查找(1个节点对象)

```

1 1、 browser.find_element_by_id('')
2 2、 browser.find_element_by_name('')
3 3、 browser.find_element_by_class_name('')
4 4、 browser.find_element_by_xpath('')
5 ... ...

```

多元素查找([节点对象列表])

```

1 1、 browser.find_elements_by_id('')
2 2、 browser.find_elements_by_name('')
3 3、 browser.find_elements_by_class_name('')
4 4、 browser.find_elements_by_xpath('')
5 ... ...

```

■ 节点对象操作

```

1 1、 ele.send_keys('') # 搜索框发送内容
2 2、 ele.click()
3 3、 ele.text          # 获取文本内容
4 4、 ele.get_attribute('src') # 获取属性值

```

今日作业

作业1: 哔哩哔哩小视频下载

- 1 哔哩哔哩小视频下载
- 2 # 1、url : <http://vc.bilibili.com/p/eden/rank#/?tab=全部>
- 3 # 2、抓取目标 : 所有异步加载的小视频

作业2: 京东商品数据抓取

- 1 1、网址 : <https://www.jd.com/>
- 2 2、目标: 名称、价格、评价、商家
- 3 3、思路
- 4 跳到商品页面后, 匹配所有商品节点对象列表
- 5 把节点对象的文本内容拿出来, 想办法处理
- 6 4、下一页(browser.page_source...)
- 7 *****time.sleep()*****