

# Day03回顾

## 请求模块总结

### requests模块使用流程

```
1 # 编码+拼接URL地址
2 url = 'http://www.baidu.com/s?'
3 params = {
4     '': '',
5     '': ''
6 }
7 params = urllib.parse.urlencode(params)
8 url = baseurl + params
9
10 # 请求
11 html = requests.get(url=url, headers=headers).text
12 html = requests.get(url=url, headers=headers).content.decode('gb2312', 'ignore')
13
14 【代码中遇到如下问题,考虑decode()问题】
15     1) 乱码
16     2) decode error: utf-8 code can not character \xxx ....
```

### 响应对象res属性

```
1 res.text      : '字符串'
2 res.content   : 'bytes'
3 res.status_code : 'HTTP响应码'
4 res.url       : '实际数据URL地址'
```

### 非结构化数据保存

```
1 【1】文件名 : 可以使用URL地址进行切割或切片
2
3 【2】保存时以 wb 的方式打开文件
4
5 【3】可用os模块创建并指定保存路径
6 import os
7 if not os.path.exists(directory):
8     os.makedirs(directory)
```

# Chrome浏览器安装插件

## ■ 安装方法

- 1 【1】从网上下载相关插件 - xxx.crx 重命名为 xxx.zip
- 2 【2】Chrome浏览器->设置->更多工具->扩展程序->开发者模式
- 3 【3】拖拽zip文件(或解压后的文件夹) 到浏览器页面
- 4 【4】重启浏览器, 使插件生效
- 5
- 6 【注意】: 当然也可以使用谷歌访问助手在线安装插件

# 目前反爬总结

## ■ 反爬虫梳理

- 1 【1】基于User-Agent反爬
- 2 1.1) 发送请求携带请求头: headers={'User-Agent' : 'Mozilla/5.0 xxxxxx'}
- 3 1.2) 多个请求时随机切换User-Agent
- 4 a) 定义列表存放大量User-Agent, 使用random.choice()每次随机选择
- 5 b) 定义py文件存放大量User-Agent, 导入后使用random.choice()每次随机选择
- 6 c) 使用fake\_useragent模块每次访问随机生成User-Agent
- 7 from fake\_useragent import UserAgent
- 8 agent = UserAgent().random
- 9
- 10 【2】响应内容中嵌入JS反爬
- 11 2.1) 现象: html页面中使用xpath helper可匹配出内容, 但是程序中匹配结果为空
- 12 2.2) 原因: 响应内容中嵌入js, 浏览器自动执行JS会调整页面结构
- 13 2.3) 解决方案: 在程序中打印响应内容: print(html)或者将html保存到本地文件, 根据实际响应内容结构来进一步调整xpath或者正则表达式

# requests模块参数总结

- 1 【1】方法 : requests.get()
- 2 【2】参数
- 3 2.1) url
- 4 2.2) headers
- 5 2.3) timeout

# 解析模块总结

## ■ re正则解析

- 1 import re
- 2 pattern = re.compile(r'正则表达式', re.S)
- 3 r\_list = pattern.findall(html)

## ■ lxml+xpath解析

```
1 from lxml import etree
2 p = etree.HTML(res.text)
3 r_list = p.xpath('xpath表达式')
4
5 【谨记】只要调用了xpath，得到的结果一定为'列表'
```

# xpath表达式

## ■ 匹配规则

```
1 【1】结果：节点对象列表
2 1.1) xpath示例：//div、//div[@class="student"]、//div/a[@title="stu"]/span
3
4 【2】结果：字符串列表
5 2.1) xpath表达式中末尾为：@src、@href、text()
```

## ■ 最常用

```
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 【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 【终极目标】：不要使你的程序因为任何异常而终止
2
3 【需要注意】
4 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
5 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
```

# Day04笔记

## 百度贴吧图片抓取

## ■ 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列表: [图片链接1,图片链接2,...]
- 4 【4】向图片的URL发请求,以wb方式写入本地文件

## ■ 实现步骤

- 1 【1】响应内容总是否存在所抓数据 : 存在!
- 2
- 3 【2】贴吧URL规律
- 4 `http://tieba.baidu.com/f?kw=?&pn=50`
- 5
- 6 【3】xpath表达式
- 7 3.1) 帖子链接: `//div[@class="t_con_cleafix"]/div/div/div/a/@href`
- 8 3.2) 图片链接: `//div[@class="d_post_content_j_d_post_content_cleafix"]/img[@class="BDE_Image"]/@src`
- 9 3.3) 视频链接: `//div[@class="video_src_wrapper"]/embed/@data-video`
- 10 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 """抓取指定贴吧的所有帖子中图片"""
2 import requests
3 from lxml import etree
4 import time
5 import random
6 from urllib import parse
7
8 class TiebaImageSpider(object):
9     def __init__(self):
10         self.url = 'http://tieba.baidu.com/f?kw={}&pn={}'
11         self.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)' }
12
13     def get_html(self,url):
14         """请求功能函数"""
15         html = requests.get(url=url,headers=self.headers).text
16
17         return html
18
19     def xpath_func(self,html,xpath_bds):
20         """解析功能函数"""
21         parse_html = etree.HTML(html)
22         r_list = parse_html.xpath(xpath_bds)
23
24         return r_list
25
26     def get_images(self,one_url):
27         one_html = self.get_html(one_url)
28         xpath_bds = '//div[@class="t_con_cleafix"]/div/div/div/a/@href'
29         href_list = self.xpath_func(one_html,xpath_bds)
30         # href_list: ['/p/2323','/p/23322','']
31         for href in href_list:
32             t_link = 'http://tieba.baidu.com' + href
```

```

33         # 把1个帖子中所有图片保存下来
34         self.save_images(t_link)
35
36     def save_images(self, t_link):
37         two_html = self.get_html(t_link)
38         two_xpath = '//div[@class="d_post_content j_d_post_content
clearfix"]/img[@class="BDE_Image"]/@src | //div[@class="video_src_wrapper"]/embed/@data-
video'
39         src_list = self.xpath_func(two_html, two_xpath)
40         for src in src_list:
41             img_html = requests.get(url=src, headers=self.headers).content
42             filename = src[-10:]
43             with open(filename, 'wb') as f:
44                 f.write(img_html)
45             print(filename, '下载成功')
46             time.sleep(random.uniform(0, 1))
47
48     def run(self):
49         name = input("请输入贴吧名:")
50         begin = int(input('请输入起始页:'))
51         end = int(input('请输入终止页:'))
52         name = parse.quote(name)
53         for page in range(begin, end+1):
54             pn = (page-1)*50
55             url = self.url.format(name, pn)
56             self.get_images(url)
57
58 if __name__ == '__main__':
59     spider = TiebaImageSpider()
60     spider.run()

```

## 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=baseurl, params=params, headers=headers)
2 | 【2】 特点:
3 |     2.1) url为基准的url地址, 不包含查询参数
4 |     2.2) 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 html = requests.get(url=baseurl,params=params,headers=headers).content.decode()

```

## SSL证书认证参数-verify

### ■ 适用网站及场景

- 1 【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 【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 【3】示例
- 4 `res = requests.get(url=url,params=params,headers=headers,verify=False)`

## 代理参数-proxies

### ■ 定义及分类

- 1 【1】定义 : 代替你原来的IP地址去对接网络的IP地址
- 2
- 3 【2】作用 : 隐藏自身真实IP,避免被封
- 4
- 5 【3】种类
- 6 3.1 高匿代理: Web端只能看到代理IP
- 7 3.2 普通代理: Web端知道有人通过此代理IP访问,但不知用户真实IP
- 8 3.3 透明代理: Web能看到用户真实IP,也能看到代理IP

### ■ 普通代理

```

1  【1】获取代理IP网站
2      西刺代理、快代理、全网代理、代理精灵、... ...
3
4  【2】参数类型
5      proxies = { '协议': '协议://IP:端口号' }
6      proxies = {
7          'http': 'http://IP:端口号',
8          'https': 'https://IP:端口号',
9      }

```

## ■ 普通代理 - 示例

```

1  # 使用免费普通代理IP访问测试网站: http://httpbin.org/get
2  import requests
3
4  url = 'http://httpbin.org/get'
5  headers = {'User-Agent': 'Mozilla/5.0'}
6  # 定义代理,在代理IP网站中查找免费代理IP
7  proxies = {
8      'http': 'http://112.85.164.220:9999',
9      'https': 'https://112.85.164.220:9999'
10 }
11 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
12 print(html)

```

## ■ 私密代理+独享代理

```

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

```

## ■ 私密代理+独享代理 - 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)

```

## get()方法参数

```
1  【1】 url=url
2  【2】 params={}
3  【3】 proxies={}
4  【4】 headers={}
5  【5】 timeout=3
6  【6】 verify=False | True
7  【7】 cookies={}
```

## requests.post()

### ■ 适用场景

```
1  【1】 适用场景 ： Post类型请求的网站
2
3  【2】 参数 ： data={}
4      2.1) Form表单数据：字典
5      2.2) res = requests.post(url=url,data=data,headers=headers)
6
7  【3】 POST请求特点 ： Form表单提交数据
```

## 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
3  【2】 控制台常用选项
4      2.1) Network： 抓取网络数据包
5          a> ALL： 抓取所有的网络数据包
6          b> XHR： 抓取异步加载的网络数据包
7          c> JS ： 抓取所有的JS文件
8      2.2) 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
9      2.3) Console： 交互模式，可对JavaScript中的代码进行测试
10
11 【3】 抓取具体网络数据包后
12      3.1) 单击左侧网络数据包地址，进入数据包详情，查看右侧
13      3.2) 右侧：
14          a> Headers： 整个请求信息
15              General、Response Headers、Request Headers、Query String、Form Data
16          b> Preview： 对响应内容进行预览
17          c> Response： 响应内容
```

## 有道翻译破解案例(post)

### ■ 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
```

## ■ 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】 : Network - JS选项 - 搜索关键词salt
2  【方法2】 : 控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出
3
4  【结果】 : 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)  ts + 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%A-rKaT5fb[Gy?;N5@Tj")
11     python实现)
12     from hashlib import md5
13     s = md5()
14     s.update(xxx.encode())
15     sign = s.hexdigest()

```

#### 4、pycharm中正则处理headers和formdata

```

1  【1】pycharm进入方法：Ctrl + r，选中 Regex
2  【2】处理headers和formdata
3      (.*)：(.*)
4      "$1": "$2",
5  【3】点击 Replace All

```

#### 5、代码实现

```

1  import requests
2  import time
3  import random
4  from hashlib import md5
5
6  class YdSpider(object):
7      def __init__(self):
8          # url一定为F12抓到的 headers -> General -> Request URL
9          self.url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
10         self.headers = {
11             # 检查频率最高 - 3个
12             "Cookie": "OUTFOX_SEARCH_USER_ID=970246104@10.169.0.83;
OUTFOX_SEARCH_USER_ID_NCOO=570559528.1224236;
_ntes_nnid=96bc13a2f5ce64962adfd6a278467214,1551873108952; JSESSIONID=aaae9i7p1XP1KaJH_gkYw;
td_cookie=18446744072941336803; SESSION_FROM_COOKIE=unknown;
__rl__test_cookies=1565689460872",
13             "Referer": "http://fanyi.youdao.com/",
14             "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/76.0.3809.100 Safari/537.36",
15         }
16
17         # 获取salt,sign,ts
18         def get_salt_sign_ts(self, word):
19             # ts
20             ts = str(int(time.time()*1000))
21             # salt
22             salt = ts + str(random.randint(0,9))
23             # sign
24             string = "fanyideskweb" + word + salt + "n%A-rKaT5fb[Gy?;N5@Tj"
25             s = md5()

```

```

26     s.update(string.encode())
27     sign = s.hexdigest()
28
29     return salt,sign,ts
30
31 # 主函数
32 def attack_yd(self,word):
33     # 1. 先拿到salt,sign,ts
34     salt,sign,ts = self.get_salt_sign_ts(word)
35     # 2. 定义form表单数据为字典: data={}
36     # 检查了salt sign
37     data = {
38         "i": word,
39         "from": "AUTO",
40         "to": "AUTO",
41         "smartresult": "dict",
42         "client": "fanyideskweb",
43         "salt": salt,
44         "sign": sign,
45         "ts": ts,
46         "bv": "7e3150ecbdf9de52dc355751b074cf60",
47         "doctype": "json",
48         "version": "2.1",
49         "keyfrom": "fanyi.web",
50         "action": "FY_BY_REALT1ME",
51     }
52     # 3. 直接发请求:requests.post(url,data=data,headers=xxx)
53     html = requests.post(
54         url=self.url,
55         data=data,
56         headers=self.headers
57     ).json()
58     # res.json() 将json格式的字符串转为python数据类型
59     result = html['translateResult'][0][0]['tgt']
60
61     print(result)
62
63 # 主函数
64 def run(self):
65     # 输入翻译单词
66     word = input('请输入要翻译的单词:')
67     self.attack_yd(word)
68
69 if __name__ == '__main__':
70     spider = YdSpider()
71     spider.run()

```

## 今日作业

- 1 【1】总结前几天内容,理顺知识点
- 2
- 3 【2】抓取西刺免费高匿代理并测试, 建立自己的IP代理池(注意数据抓取的频率)
- 4 <https://www.xicidaili.com/nn/{ }> # { }为: 1 2 3 4 5

5  
6 **【3】民政部网站数据抓取 - # 一切以响应内容为主 (URL的JS的跳转)**  
7 3.1) URL: <http://www.mca.gov.cn/> - 民政数据 - 行政区划代码  
8 即: <http://www.mca.gov.cn/article/sj/xzqh/2019/>  
9 3.2) 目标: 抓取最新中华人民共和国县级以上行政区划代码  
10 3.3) 要求一: 增量, 每次运行程序只抓取最新的, 如未更新则提示未更新, 不进行抓取  
11 3.4) 要求二: 所抓数据存入数据库, 最好分表存储 - 省、市、县 3张表  
12 省表: province 字段: pname pcode  
13 市表: city 字段: cname ccode cfather\_code  
14 县表: county 字段: xname xcode xfather\_code  
15 **【特殊情况】**: 四个直辖市在省表和市表中都存一份