Day03回顾

目前反爬总结

- 基于User-Agent反爬

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

- 响应内容前端JS做处理反爬

```
1、html页面中可匹配出内容，程序中匹配结果为空
    * 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
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

```
# 编码
params = {
    '':'',
    '':''
}
params = urllib.parse.urlencode(params)
url = baseurl + params

# 请求
request = urllib.request.Request(url,headers=headers)
response = urllib.request.urlopen(request)
html = response.read().decode('utf-8')
```

- requests模块使用流程

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

解析模块总结

- 正则解析re模块

```
import re

pattern = re.compile('正则表达式',re.S)
r_list = pattern.findall(html)
```

- lxml解析库

```
from lxml import etree

parse_html = etree.HTML(res.text)
r_list = parse_html.xpath('xpath表达式')
```

xpath表达式

- 匹配规则

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

- xpath高级

```
1、基准xpath表达式：得到节点对象列表
2、for r in [节点对象列表]:
       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
```

写程序注意

```
# 最终目标：不要使你的程序因为任何异常而终止
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
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，可使用列表推到式
# 多级页面数据抓取注意
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)
```

Day04笔记

requests.get()参数

查询参数-params

- 参数类型

```
字典,字典中键值对作为查询参数
```

- 使用方法

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

- 示例

```
import requests

baseurl = 'http://tieba.baidu.com/f?'
params = {
  'kw' : '赵丽颖吧',
  'pn' : '50'
}
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)'}
# 自动对params进行编码,然后自动和url进行拼接,去发请求
res = requests.get(baseurl,params=params,headers=headers)
res.encoding = 'utf-8'
print(res.text)
```

Web客户端验证 参数-auth

- 作用及类型

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

- 达内code课程方向案例

```

```

思考：爬取具体的笔记文件?

```
1
```

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、verify=True(默认)  ：检查证书认证
2、verify=False (常用)：忽略证书认证
# 示例
response = requests.get(
    url=url,
    params=params,
    headers=headers,
    verify=False
)
```

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

```
西刺代理、快代理、全网代理、代理精灵、... ...
```

参数类型

```
1、语法结构
    proxies = {
        '协议':'协议://IP:端口号'
    }
2、示例
    proxies = {
        'http':'http://IP:端口号',
        'https':'https://IP:端口号'
    }
```

示例

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

```python
import requests

url = 'http://httpbin.org/get'
headers = {
    'User-Agent':'Mozilla/5.0'
}
# 定义代理,在代理IP网站中查找免费代理IP
proxies = {
    'http':'http://112.85.164.220:9999',
    'https':'https://112.85.164.220:9999'
}
html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
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**

```

```

2、写一个获取收费开放代理的接口

```python
# 获取开放代理的接口
import requests

def test_ip(ip):
    url = 'http://www.baidu.com/'
    ip_port = ip.split(':')
    proxies = {
        'http':'http://{}:{}'.format(ip_port[0],ip_port[1]),
        'https':'https://{}:{}'.format(ip_port[0],ip_port[1]),
    }
    res = requests.get(url=url)
    if res.status_code == 200:
        return True
    else:
        return False

# 提取代理IP
def get_ip_list():
  api_url = 'http://dev.kdlapi.com/api/getproxy/?orderid=946562662041898&num=100&protocol=1&method=2&an_an=1&an_ha=1&sep=2'
```

```
  html = requests.get(api_url).content.decode('utf-8','ignore')
  ip_port_list = html.split('\n')

  for ip in ip_port_list:
    with open('proxy_ip.txt','a') as f:
        if test_ip(ip):
            f.write(ip + '\n')

if __name__ == '__main__':
    get_ip_list()
```

3、使用随机收费开放代理IP写爬虫

```
import random
import requests

class BaiduSpider(object):
    def __init__(self):
        self.url = 'http://www.baidu.com/'
        self.headers = {'User-Agent' : 'Mozilla/5.0'}
        self.blag = 1

    def get_proxies(self):
        with open('proxy_ip.txt','r') as f:
            result = f.readlines()
        proxy_ip = random.choice(result)[:-1]
        L = proxy_ip.split(':')
        proxy_ip = {
            'http':'http://{}:{}'.format(L[0],L[1]),
            'https': 'https://{}:{}'.format(L[0], L[1])
        }
        return proxy_ip

    def get_html(self):
        proxies = self.get_proxies()
        if self.blag <= 3:
            try:
                html =
requests.get(url=self.url,proxies=proxies,headers=self.headers,timeout=5).text
                print(html)
            except Exception as e:
                print('Retry')
                self.blag += 1
                self.get_html()

if __name__ == '__main__':
    spider = BaiduSpider()
    spider.get_html()
```

- 私密代理

语法格式

```
1、语法结构
proxies = {
    '协议':'协议://用户名:密码@IP:端口号'
}

2、示例
proxies = {
    'http':'http://用户名:密码@IP:端口号',
    'https':'https://用户名:密码@IP:端口号'
}
```

示例代码

```
import requests
url = 'http://httpbin.org/get'
proxies = {
    'http': 'http://309435365:szayclhp@106.75.71.140:16816',
    'https':'https://309435365:szayclhp@106.75.71.140:16816',
}
headers = {
    'User-Agent' : 'Mozilla/5.0',
}

html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
print(html)
```

控制台抓包

- 打开方式及常用选项

```
1、打开浏览器，F12打开控制台，找到Network选项卡
2、控制台常用选项
   1、Network：抓取网络数据包
        1、ALL：抓取所有的网络数据包
        2、XHR：抓取异步加载的网络数据包
        3、JS ：抓取所有的JS文件
   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
   3、Console：交互模式，可对JavaScript中的代码进行测试
3、抓取具体网络数据包后
   1、单击左侧网络数据包地址，进入数据包详情，查看右侧
   2、右侧：
       1、Headers：整个请求信息
            General、Response Headers、Request Headers、Query String、Form Data
       2、Preview：对响应内容进行预览
       3、Response：响应内容
```

requests.post()

- 适用场景

```
Post类型请求的网站
```

- 参数-data

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

- 请求方式的特点

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

有道翻译破解案例(post)

1. 目标

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

2. 实现步骤

```
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、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、刷新有道翻译页面，抓取并分析JS代码（本地JS加密）
4、找到JS加密算法，用Python按同样方式加密生成加密数据
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:**

```
i: 喵喵叫
from: AUTO
to: AUTO
smartresult: dict
client: fanyideskweb
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
doctype: json
version: 2.1
keyfrom: fanyi.web
action: FY_BY_REALTlME
```

- **2、在页面中多翻译几个单词，观察Form表单数据变化**

```
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
# 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**

```
# 方法1
Network - JS选项 - 搜索关键词salt
# 方法2
控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出

# 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**

```
# ts : 经过分析为13位的时间戳，字符串类型
js代码实现:  "" + (new Date).getTime()
python实现:

# salt
js代码实现:  r+parseInt(10 * Math.random(), 10);
python实现:

# sign (设置断点调试，来查看 e 的值，发现 e 为要翻译的单词)
js代码实现: n.md5("fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj")
python实现:
```

- **5、代码实现**

```

```

今日作业

```
1、Web客户端验证 - 如何抓取文件及保存文件
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、仔细复习并总结有道翻译案例，抓包流程，代码实现
```