Day04回顾

requests.get()参数

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

requests.post()

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

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
```

- 有道翻译过程梳理

1.
```
1. 打开首页
2. 准备抓包：F12开启控制台
3. 寻找地址
   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
4. 发现规律
   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
5. 寻找JS文件
   右上角 ... -> Search -> 搜索关键字 -> 单击 -> 跳转到Sources，左下角格式化符号{}
6、查看JS代码
   搜索关键字，找到相关加密方法
7、断点调试
8、完善程序
```

常见的反爬机制及处理方式

```
1、Headers反爬虫 ：Cookie、Referer、User-Agent
   解决方案：通过F12获取headers,传给requests.get()方法

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
   解决方案：
        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
        2、购买开放代理或私密代理IP
        3、降低爬取的速度

3、User-Agent限制 ：类似于IP限制
   解决方案：构造自己的User-Agent池,每次访问随机选择

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
   解决方案：F12或抓包工具抓包处理

5、对查询参数加密
   解决方案：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

6、对响应内容做处理
   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、pycharm进入方法 ：Ctrl + r ，选中 Regex
2、处理headers和formdata
  (.*): (.*)
  "$1": "$2",
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
# 控制台中 XHR ： 异步加载的数据包
# XHR -> QueryStringParameters(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)
# 抓取的查询参数如下：
type: 13
interval_id: 100:90
action: ''
start: 0
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
# 示例
html = json.loads(res.text)
print(type(html))
```

- 代码实现

```
import requests
import json

class DoubanSpider(object):
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
```

```
        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'}

    # 获取页面
    def get_page(self,params):
        res = requests.get(
            url=self.url,
            params=params,
            headers=self.headers,
            verify=True
        )
        res.encoding = 'utf-8'
        # json.loads() josn格式->Python格式
        html = res.json()
        self.parse_page(html)

    # 解析并保存数据
    def parse_page(self,html):
        # html为大列表 [{电影1信息},{},{}]
        for h in html:
            # 名称
            name = h['title'].strip()
            # 评分
            score = float(h['score'].strip())
            # 打印测试
            print([name,score])

    # 主函数
    def main(self):
        limit = input('请输入电影数量:')
        params = {
            'type' : '24',
            'interval_id' : '100:90',
            'action' : '',
            'start' : '0',
            'limit' : limit
        }
        # 调用函数,传递params参数
        self.get_page(params)

if __name__ == '__main__':
    spider = DoubanSpider()
    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

```
# 主函数
def main(self):
    # 定义字典,来存放类型和对应的数字
    ty_dict = {'剧情': '11', '喜剧': '24', '爱情': '13'}
    ty = input('请输入电影类型(剧情|喜剧|爱情):')
    if ty in ['剧情', '喜剧', '爱情']:
        limit = input('请输入电影数量:')
        params = {
            'type' : ty_dict[ty],
            'interval_id' : '100:90',
```

```
            'action' : '',
            'start' : '0',
            'limit' : limit
        }
        # 调用函数,传递params参数
        self.get_page(params)
        else:
            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、URL：百度搜索腾讯招聘 - 查看工作岗位
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)

```
https://careers.tencent.com/tencentcareer/api/post/Query?timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attrId=&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)

```
https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1563912374645&postId={}&language=zh-cn
```

- 具体代码实现

```
import requests
import json
import time
import random

class TencentSpider(object):
  def __init__(self):
    self.headers = {'User-Agent':'Mozilla/5.0'}
    self.one_url = 'https://careers.tencent.com/tencentcareer/api/post/Query?timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attrId=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn'

  def get_page(self,url):
    res = requests.get(url,headers=self.headers)
    res.encoding = 'utf-8'
    # json.loads()把json格式的字符串转为python数据类型
    html = json.loads(res.text)
    return html

  def parse_one_page(self,html):
    job_info = {}
```

```
    for job in html['Data']['Posts']:
      job_info['job_name'] = job['RecruitPostName']
      job_info['job_address'] = job['LocationName']
      # 拿postid为了拼接二级页面地址
      post_id = job['PostId']
      # 职责和要求(二级页面)
      # 得到二级页面链接
      two_url = 'https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1563912374645&postId={}&language=zh-cn'.format(post_id)
      # 发请求解析
      job_info['job_duty'],job_info['job_requirement'] = self.parse_two_page(two_url)
      print(job_info)

  def parse_two_page(self,two_url):
    html = self.get_page(two_url)
    # 职责
    job_duty = html['Data']['Responsibility']
    # 要求
    job_requirement = html['Data']['Requirement']

    return job_duty,job_requirement

  def main(self):
    for index in range(1,11):
      url = self.one_url.format(index)
      one_html = self.get_page(url)
      self.parse_one_page(one_html)
      time.sleep(random.uniform(0.5,1.5))

if __name__ == '__main__':
  spider = TencentSpider()
  spider.main()
```

cookie模拟登录

- 适用网站及场景

```
抓取需要登录才能访问的页面
```

- **方法一**

```
1、先登录成功1次,获取到携带登陆信息的Cookie
   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)
2、携带着cookie发请求
   ** Cookie
   ** Referer(源,代表你从哪里转过来的)
   ** User-Agent
```

```
import requests
from lxml import etree

```

```
# url为需要登录才能正常访问的地址
url = 'http://www.renren.com/969255813/profile'
# headers中的cookie为登录成功后抓取到的cookie
headers = {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "zh-CN,zh;q=0.9",
    "Connection": "keep-alive",
    # 此处注意cookie，要自己抓取
    "Cookie": "",
    "Host": "www.renren.com",
    "Referer": "http://www.renren.com/SysHome.do",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36",
}

html = requests.get(url,headers=headers).text
# 解析
parse_html = etree.HTML(html)
result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
# result:就读于中央戏剧学院
print(result)
```

- **方法二**

1. 知识点

```
利用requests模块中的session会话保持功能
```

2. session会话使用流程

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

3. 具体步骤

```
1、寻找登录时POST的地址
   查看网页源码,查看form,找action对应的地址: http://www.renren.com/PLogin.do

2、发送用户名和密码信息到POST的地址
   * 用户名和密码信息以什么方式发送?  -- 字典
     键 : <input>标签中name的值(email,password)
     值 : 真实的用户名和密码
     post_data = {'email':'','password':''}
```

4. 程序实现

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

```python
import requests
from lxml import etree

# 定义常用变量
post_url = 'http://www.renren.com/PLogin.do'
post_data = {
  'email' : 'xxxxxx',
  'password' : 'xxxxxx'
}
headers = {
  'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36',
  'Referer' : 'http://www.renren.com/SysHome.do'
}

# 实例化session会话保持对象
session = requests.session()
# 先POST,把用户名和密码信息POST到一个地址
session.post(post_url,data=post_data,headers=headers)

# 再get个人主页
url = 'http://www.renren.com/970294164/profile'
html = session.get(url,headers=headers).text

parse_html = etree.HTML(html)
result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
print(result)
```

requests.post()

- 适用场景a

```
Post类型请求的网站
```

- 参数-data

```python
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:  str(int(time.time()*1000))

# salt
js代码实现:  r+parseInt(10 * Math.random(), 10);
python实现:  ts + str(random.randint(0,9))

# 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:
from hashlib import md5
s = md5()
s.update("fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj".encode())
sign = s.hexdigest()
```

- 5、代码实现

```
import requests
import time
from hashlib import md5
import random

# 获取相关加密算法的结果
def get_salt_sign_ts(word):
    # salt
    salt = str(int(time.time()*1000)) + str(random.randint(0,9))
    # sign
    string = "fanyideskweb" + word + salt + "n%A-rKaT5fb[Gy?;N5@Tj"
    s = md5()
    s.update(string.encode())
    sign = s.hexdigest()
    # ts
    ts = str(int(time.time()*1000))
    return salt,sign,ts

# 攻克有道
def attack_yd(word):
    salt,sign,ts = get_salt_sign_ts(word)
    # url为抓包抓到的地址 F12 -> translate_o -> post
    url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
    headers = {
        "Accept": "application/json, text/javascript, */*; q=0.01",
        "Accept-Encoding": "gzip, deflate",
        "Accept-Language": "zh-CN,zh;q=0.9",
```

```
        "Connection": "keep-alive",
        "Content-Length": "238",
        "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
        "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",
        "Host": "fanyi.youdao.com",
        "Origin": "http://fanyi.youdao.com",
        "Referer": "http://fanyi.youdao.com/",
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/74.0.3729.169 Safari/537.36",
        "X-Requested-With": "XMLHttpRequest",
    }
    # Form表单数据
    data = {
        'i': word,
        'from': 'AUTO',
        'to': 'AUTO',
        'smartresult': 'dict',
        'client': 'fanyideskweb',
        'salt': salt,
        'sign': sign,
        'ts': ts,
        'bv': 'cf156b581152bd0b259b90070b1120e6',
        'doctype': 'json',
        'version': '2.1',
        'keyfrom': 'fanyi.web',
        'action': 'FY_BY_REALTlME'
    }

    json_html = requests.post(url,data=data,headers=headers).json()
    result = json_html['translateResult'][0][0]['tgt']
    return result

if __name__ == '__main__':
    word = input('请输入要翻译的单词: ')
    result = attack_yd(word)
    print(result)
```

代理IP使用

购买开放代理使用

- 开放代理接口

```
# getip.py
# 获取开放代理的接口
import requests

# 提取代理IP
```

```
def get_ip_list():
  api_url = 'http://dev.kdlapi.com/api/getproxy/?orderid=996140620552954&num=100&protocol=2&method=2&an_an=1&an_ha=1&sep=1'
  res = requests.get(api_url)
  ip_port_list = res.text.split('\r\n')

  return ip_port_list

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

测试开放代理接口: 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
```

```
from getip import *
import time
import random

url = 'http://httpbin.org/get'
headers = {'User-Agent' : 'Mozilla/5.0'}
proxy_ip_list = get_ip_list()

while True:
    # 判断是否还有可用代理
    if not proxy_ip_list:
        proxy_ip_list = get_ip_list()

    proxy_ip = random.choice(proxy_ip_list)
    proxies = {
        'http' : 'http://{}'.format(proxy_ip),
        'https' : 'https://{}'.format(proxy_ip)
    }
    print(proxies)

    try:
        html = requests.get(url=url,proxies=proxies,headers=headers,timeout=5,verify=False).text
        print(html)
        break
    except:
        print('正在更换代理IP，请稍后... ...')
        # 及时把不可用的代理IP移除
        proxy_ip_list.remove(proxy_ip)
        continue
```

购买私密代理使用

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@122.114.67.136:16819',
    'https':'https://309435365:szayclhp@122.114.67.136:16819',
}
headers = {
    'User-Agent' : 'Mozilla/5.0',
}

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

今日作业

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
```