DAY05

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, 当程序中抛出 SSLError 时添加 verify=False
【6】timeout
【7】headers
【8】cookies
```

- **常见的反爬机制及处理方式**

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

【2】IP限制
  2.1) 网站根据IP地址访问频率进行反爬,短时间内限制IP访问
  2.2) 解决方案：
       a) 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
       b) 购买开放代理或私密代理IP
       c) 降低爬取的速度

【3】User-Agent限制
  3.1) 类似于IP限制，检测频率
  3.2) 解决方案：构造自己的User-Agent池,每次访问随机选择

【4】对响应内容做处理
  4.1) 页面结构和响应内容不同
  4.2) 解决方案：打印并查看响应内容,用xpath或正则做处理
```

- **有道翻译过程梳理**

```
【1】打开首页

```

```
【2】准备抓包：F12开启控制台

【3】寻找地址
  3.1) 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
       F12-Network-XHR-Headers-Grneral-Request URL

【4】发现规律
  4.1) 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
  4.2) 分析对比 Network - All(或者XHR) - Form Data，发现对应的规律

【5】寻找JS加密文件
  5.1) 控制台右上角 ...->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
  6.1) 搜索关键字，找到相关加密方法，用python实现加密算法

【7】断点调试
  7.1) JS代码中部分参数不清楚可通过断点调试来分析查看

【8】完善程序
```

Day05笔记

有道翻译破解案例(post)

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
```

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
```

具体步骤操作

- **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:  ts + 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
sign = md5("".encode()).hexdigest()
```

- **5、pycharm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
```

- **6、代码实现**

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

class YdSpider(object):
  def __init__(self):
    # url一定为F12抓到的 headers -> General -> Request URL
    self.url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
    self.headers = {
      # 检查频率最高 - 3个
      "Cookie": "OUTFOX_SEARCH_USER_ID=970246104@10.169.0.83;
OUTFOX_SEARCH_USER_ID_NCOO=570559528.1224236;
_ntes_nnid=96bc13a2f5ce64962adfd6a278467214,1551873108952;
JSESSIONID=aaae9i7plXPlKaJH_gkYw; td_cookie=18446744072941336803;
SESSION_FROM_COOKIE=unknown; ___rl__test__cookies=1565689460872",
      "Referer": "http://fanyi.youdao.com/",
      "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",
    }

  # 获取salt,sign,ts
  def get_salt_sign_ts(self,word):
    # ts
    ts = str(int(time.time()*1000))
    # salt
    salt = ts + 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()

    return salt,sign,ts

  # 主函数
  def attack_yd(self,word):
    # 1. 先拿到salt,sign,ts
    salt,sign,ts = self.get_salt_sign_ts(word)
    # 2. 定义form表单数据为字典: data={}
    # 检查了salt sign
    data = {
      "i": word,
      "from": "AUTO",
      "to": "AUTO",
      "smartresult": "dict",
      "client": "fanyideskweb",
```

```
        "salt": salt,
        "sign": sign,
        "ts": ts,
        "bv": "7e3150ecbdf9de52dc355751b074cf60",
        "doctype": "json",
        "version": "2.1",
        "keyfrom": "fanyi.web",
        "action": "FY_BY_REALTlME",
      }
      # 3. 直接发请求:requests.post(url,data=data,headers=xxx)
      res = requests.post(
        url=self.url,
        data=data,
        headers=self.headers
      )
      # res.json() 将json格式的字符串转为python数据类型
      html = res.json()
      # html:{'translateResult': [[{'tgt': '你好', 'src': 'hello'}]], 'errorCode': 0, 'type': 'en2zh-CHS', 'smartResult': {'entries': ['', 'n. 表示问候， 惊奇或唤起注意时的用语\r\n', 'int. 喂；哈罗\r\n', 'n. (Hello)人名；(法)埃洛\r\n'], 'type': 1}}
      result = html['translateResult'][0][0]['tgt']

      print(result)

    # 主函数
    def run(self):
      # 输入翻译单词
      word = input('请输入要翻译的单词:')
      self.attack_yd(word)

if __name__ == '__main__':
    spider = YdSpider()
    spider.run()
```

民政部网站数据抓取

- **目标**

```
1、URL: http://www.mca.gov.cn/ - 民政数据 - 行政区划代码
   即: http://www.mca.gov.cn/article/sj/xzqh/2019/
2、目标: 抓取最新中华人民共和国县以上行政区划代码
```

- **实现步骤**

```
【1】从民政数据网站中提取最新行政区划代码链接
   1.1) 新的在上面第2个
   1.2) xpath表达式: //table//tr[2]/td[2]/a/@href


【2】从二级页面响应内容中提取真实链接
   2.1) 反爬 - 响应内容中嵌入JS，指向新的链接
   2.2) 打印响应内容，搜索真实链接URL，找到位置
```

```
  2.3) 正则匹配: window.location.href="(.*?)"

【3】从真实链接中提取所需数据
  3.1) 基准xpath(以响应内容为主): //table/tr[2]/td[2]/a/@href
  3.2) for循环依次遍历提取数据
       编码: ./td[2]/text() | ./td[2]/span/text()
       名称: ./td[3]/text()
```

- **代码实现 - 使用redis实现增量**

```python
import requests
from lxml import etree
import re
import redis
from hashlib import md5

class GovementSpider(object):
  def __init__(self):
    self.index_url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'
    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)'}
    # redis指纹增量
    self.r = redis.Redis(host='localhost',port=6379,db=0)

  def get_html(self,url):
    """请求功能函数"""
    html = requests.get(url=url,headers=self.headers).text
    return html

  def md5_url(self,url):
    """URL加密函数"""
    s = md5()
    s.update(url.encode())
    return s.hexdigest()

  def get_new_link(self):
    """获取最新月份链接"""
    # 请求
    html = self.get_html(self.index_url)
    # 解析提取最新链接
    parse_html = etree.HTML(html)
    lasted_href_list = parse_html.xpath('//table/tr[2]/td[2]/a/@href')
    if lasted_href_list:
      lasted_href = lasted_href_list[0]
      lasted_url = 'http://www.mca.gov.cn' + lasted_href
      # 生成指纹
      finger = self.md5_url(lasted_url)
      # redis集合增量判断
      if self.r.sadd('govspider:fingers',finger):
        self.parse_html(lasted_url)
      else:
        print('数据已是最新')
    else:
      print('提取最新月份链接失败')

```

```
  def parse_html(self,lasted_url):
    """获取真链接"""
    # 嵌入JS执行URL跳转,提取真实链接
    html = self.get_html(lasted_url)
    regex = r'window.location.href="(.*?)"'
    pattern = re.compile(regex,re.S)
    true_url_list = pattern.findall(html)
    if true_url_list:
      true_url = true_url_list[0]
      self.get_data(true_url)
    else:
      print('提取真实链接失败')

  def get_data(self,true_url):
    """详细数据提取"""
    html = self.get_html(true_url)
    # xpath提取数据
    parse_html = etree.HTML(html)
    tr_list = parse_html.xpath('//tr[@height="19"]')
    for tr in tr_list:
      code_list = tr.xpath('./td[2]/text() | ./td[2]/span/text()')
      name_list = tr.xpath('./td[3]/text()')
      if code_list and name_list:
        code,name = code_list[0].strip(),name_list[0].strip()
        print(name,code)

  def run(self):
    """程序入口函数"""
    self.get_new_link()

if __name__ == '__main__':
  spider = GovementSpider()
  spider.run()
```

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,或者页面局部刷新
```

- **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
  2.1) 控制台中 XHR ： 异步加载的数据包
  2.2) 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 # 电影类型
   interval_id: 100:90
   action: ''
   start: 0  # 每次加载电影的起始索引值 0 20 40 60
   limit: 20 # 每次加载的电影数量
```

- **代码实现 - 全站抓取**

```python
import requests
import time
import random
import re
from fake_useragent import UserAgent

class DoubanSpider(object):
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
        self.i = 0

    # 获取随机headers
    def get_headers(self):
        headers = {'User-Agent':UserAgent().random }

        return headers

    # 获取页面
    def get_page(self,params):
      # 返回 python 数据类型
        html = requests.get(url=self.url,params=params,headers=self.get_headers()).json()
        self.parse_page(html)

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

```
35
36      # 主函数
37      def run(self):
38          # 获取type的值
39          type_dict = self.get_all_type_films()
40          # 生成菜单
41          menu = ''
42          for key in type_dict:
43            menu += key + '|'
44
45          menu = menu + '\n请做出你的选择:'
46          name = input(menu)
47          type_number = type_dict[name]
48          # 获取电影总数
49          total = self.total_number(type_number)
50          for start in range(0,(total+1),20):
51              params = {
52                  'type' : type_number,
53                  'interval_id' : '100:90',
54                  'action' : '',
55                  'start' : str(start),
56                  'limit' : '20'
57              }
58              # 调用函数,传递params参数
59              self.get_page(params)
60              # 随机休眠1-3秒
61              time.sleep(random.randint(1,3))
62          print('电影数量:',self.i)
63
64      # 获取电影总数
65      def total_number(self,type_number):
66          # F12抓包抓到的地址
67          url = 'https://movie.douban.com/j/chart/top_list_count?type=
{}&interval_id=100%3A90'.format(type_number)
68          headers = self.get_headers()
69          html = requests.get(url=url,headers=headers).json()
70          total = int(html['total'])
71
72          return total
73
74      # 获取所有电影的名字和对应type值
75      def get_all_type_films(self):
76          # 获取 类型和类型码
77          url = 'https://movie.douban.com/chart'
78          headers = self.get_headers()
79          html = requests.get(url=url,headers=headers).text
80          re_bds = r'<a href=.*?type_name=(.*?)&type=(.*?)&.*?</a>'
81          pattern = re.compile(re_bds,re.S)
82          r_list = pattern.findall(html)
83          # 存放所有类型和对应类型码大字典
84          type_dict = {}
85          for r in r_list:
86              type_dict[r[0].strip()] = r[1].strip()
87
88          return type_dict
89
90  if __name__ == '__main__':
```

```
    spider = DoubanSpider()
    spider.run()
```

json解析模块

- **json.loads(json)**

```
【1】作用 ： 把json格式的字符串转为Python数据类型

【2】示例 ： html = json.loads(res.text)
```

- **json.dump(python,f,ensure_ascii=False)**

```
【1】作用
  把python数据类型 转为 json格式的字符串,一般让你把抓取的数据保存为json文件时使用

【2】参数说明
  2.1) 第1个参数: python类型的数据(字典，列表等)
  2.2) 第2个参数: 文件对象
  2.3) 第3个参数: ensure_ascii=False 序列化时编码

【3】示例代码
    # 示例1
    import json

    item = {'name':'QQ','app_id':1}
    with open('小米.json','a') as f:
      json.dump(item,f,ensure_ascii=False)

    # 示例2
    import json

    item_list = []
    for i in range(3):
      item = {'name':'QQ','id':i}
      item_list.append(item)

    with open('xiaomi.json','a') as f:
        json.dump(item_list,f,ensure_ascii=False)
```

- **json.dumps(python)**

```
【1】作用 ： 把 python 类型 转为 json 格式的字符串

【2】 示例
import json

# json.dumps()之前
item = {'name':'QQ','app_id':1}
print('before dumps',type(item)) # dict
# json.dumps之后
item = json.dumps(item)
print('after dumps',type(item)) # str
```

- **json.load(f)**

```
【1】作用 ： 将json文件读取,并转为python类型

【2】 示例
import json
with open('D:/spider_test/xiaomi.json','r') as f:
    data = json.load(f)

print(data)
```

- **json模块总结**

```
# 爬虫最常用
【1】数据抓取 - json.loads(html)
    将响应内容由: json 转为 python
【2】数据保存 - json.dump(item_list,f,ensure_ascii=False)
    将抓取的数据保存到本地 json文件

# 抓取数据一般处理方式
【1】txt文件
【2】csv文件
【3】json文件
【4】MySQL数据库
【5】MongoDB数据库
【6】Redis数据库
```

多线程爬虫

- **应用场景**

```
【1】多进程 ： CPU密集程序
【2】多线程 ： 爬虫(网络I/O)、本地磁盘I/O
```

知识点回顾

- **队列**

```
```

```
【1】导入模块
  from queue import Queue

【2】使用
   q = Queue()
   q.put(url)
   q.get()   # 当队列为空时，阻塞
   q.empty() # 判断队列是否为空，True/False

【3】q.get()解除阻塞方式
  3.1) q.get(block=False)
  3.2) q.get(block=True,timeout=3)
  3.3) if not q.empty():
           q.get()
```

- **线程模块**

```python
# 导入模块
from threading import Thread

# 使用流程
t = Thread(target=函数名) # 创建线程对象
t.start() # 创建并启动线程
t.join()  # 阻塞等待回收线程

# 如何创建多线程
t_list = []

for i in range(5):
    t = Thread(target=函数名)
    t_list.append(t)
    t.start()

for t in t_list:
    t.join()
```

今日作业

```
【1】豆瓣电影全站抓取
    剧情|喜剧|爱情|动作|儿童|... ...
    请输入要抓取的电影：

【2】腾讯招聘职位信息抓取
   1) 网址：腾讯招聘官网 - 职位信息 https://careers.tencent.com/search.html
   2) 目标：所有职位的如下信息：
      a> 职位名称
      b> 职位地址
      c> 职位类别（技术类、销售类...）
      d> 发布时间
      e> 工作职责
      f> 工作要求
   3) 最终信息详情要通过二级页面拿到,因为二级页面信息很全，而一级页面信息不全(无工作要求)
```

15
16 【3】抓取百度图片（输入关键字，抓取该关键字的很多图片，而不仅仅是之前的30张）