# **Day03回顾**

## **请求模块总结**

* **requests模块使用流程**
* # 编码+拼接URL地址  
  url = 'http://www.baidu.com/s?'  
  params = {  
   '':'',  
   '':''  
  }  
  params = urllib.parse.urlencode(params)  
  url = baseurl + params  
    
  # 请求  
  html = requests.get(url=url,headers=headers).text  
  html = requests.get(url=url,headers=headers).content.decode('gb2312','ignore')  
    
  【代码中遇到如下问题,考虑decode()问题】  
   1) 乱码  
   2) decode error: utf-8 code can not character \xxx .... ....
* **响应对象res属性**
* 【1】res.encoding : '字符编码'  
  【2】res.text : '字符串'  
  【3】res.content : 'bytes'  
  【4】res.status\_code : 'HTTP响应码'  
  【5】res.url : '实际数据URL地址'

## **Chrome浏览器安装插件**

* **安装方法**
* 【1】从网上下载相关插件 - xxx.crx 重命名为 xxx.zip  
  【2】Chrome浏览器->设置->更多工具->扩展程序->开发者模式  
  【3】拖拽zip文件(或解压后的文件夹) 到浏览器页面  
  【4】重启浏览器，使插件生效  
    
  【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】基于User-Agent反爬  
   1.1) 发送请求携带请求头: headers={'User-Agent' : 'Mozilla/5.0 xxxxxx'}  
   1.2) 多个请求时随机切换User-Agent  
   a) 定义列表存放大量User-Agent，使用random.choice()每次随机选择  
   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  
   c) 使用fake\_useragent模块每次访问随机生成User-Agent  
   from fake\_useragent import UserAgent  
   agent = UserAgent().random  
     
  【2】响应内容中嵌入JS反爬  
   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  
   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  
   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】方法 : requests.get()  
【2】参数  
 2.1) url  
 2.2) headers  
 2.3) timeout  
 2.4) params  
 2.5) verify  
 2.6) proxies

## **解析模块总结**

* **re正则解析**

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

* **lxml+xpath解析**

from lxml import etree  
p = etree.HTML(res.text)  
r\_list = p.xpath('xpath表达式')  
  
【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】结果: 节点对象列表  
   1.1) xpath示例: //div、//div[@class="student"]、//div/a[@title="stu"]/span  
    
  【2】结果: 字符串列表  
   2.1) xpath表达式中末尾为: @src、@href、/text()
* **最常用**
* 【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')  
    
  【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**

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

# **Day04笔记**

## **代理参数-proxies**

* **定义及分类**
* 【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  
    
  【3】种类  
   3.1) 高匿代理: Web端只能看到代理IP  
   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  
   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】获取代理IP网站  
   西刺代理、快代理、全网代理、代理精灵、... ...  
    
  【2】参数类型  
   proxies = { '协议':'协议://IP:端口号' }  
   proxies = {  
   'http':'http://IP:端口号',  
   'https':'https://IP:端口号',  
   }
* **普通代理 - 示例**
* # 使用免费普通代理IP访问测试网站: http://httpbin.org/get  
  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)
* **私密代理+独享代理**
* 【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',  
   }
* **私密代理+独享代理 - 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)
* **课堂练习**
* 【1】使用开放代理建立自己的代理IP池  
  【2】使用私密代理建立自己的代理IP池

## **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/2020/  
     
  【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): //tr[@height="19"]  
   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**
* import requests  
  from lxml import etree  
  import re  
  import redis  
  from hashlib import md5  
  import pymysql  
  import sys  
    
  class GovementSpider(object):  
   def \_\_init\_\_(self):  
   self.index\_url = 'http://www.mca.gov.cn/article/sj/xzqh/2020/'  
   self.headers = {  
   "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36",  
   }  
   # 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 xpath\_func(self, html, xpath\_bds):  
   """解析功能函数"""  
   p = etree.HTML(html)  
   r\_list = p.xpath(xpath\_bds)  
    
   return r\_list  
    
   def md5\_url(self,url):  
   """URL加密函数"""  
   s = md5()  
   s.update(url.encode())  
    
   return s.hexdigest()  
    
   def get\_false\_url(self):  
   """获取最新月份链接 - 假链接"""  
   html = self.get\_html(self.index\_url)  
   # 解析提取最新月份链接 - 假链接  
   one\_xpath = '//table/tr[2]/td[2]/a/@href'  
   false\_href\_list = self.xpath\_func(html,one\_xpath)  
   if false\_href\_list:  
   false\_href = false\_href\_list[0]  
   false\_url = 'http://www.mca.gov.cn' + false\_href  
   # 生成指纹  
   finger = self.md5\_url(false\_url)  
   # redis集合增量判断  
   if self.r.sadd('govspider:fingers',finger):  
   self.get\_real\_url(false\_url)  
   else:  
   sys.exit('数据已是最新')  
   else:  
   print('提取最新月份链接失败')  
    
   def get\_real\_url(self,false\_url):  
   """获取真链接"""  
   # 嵌入JS执行URL跳转,提取真实链接  
   html = self.get\_html(false\_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]  
   # 提取具体的数据  
   self.get\_data(true\_url)  
   else:  
   print('提取真实链接失败')  
    
   def get\_data(self,true\_url):  
   """提取具体的数据"""  
   html = self.get\_html(true\_url)  
   # xpath提取数据  
   two\_xpath = '//tr[@height="19"]'  
   tr\_list = self.xpath\_func(html, two\_xpath)  
   for tr in tr\_list:  
   code\_list = tr.xpath('./td[2]/text() | ./td[2]/span/text()')  
   name\_list = tr.xpath('./td[3]/text()')  
   code = code\_list[0].strip() if code\_list else None  
   name = name\_list[0].strip() if name\_list else None  
   print(name, code)  
    
   def run(self):  
   """程序入口函数"""  
   self.get\_false\_url()  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   spider = GovementSpider()  
   spider.run()

## **requests.post()**

* **适用场景**
* 【1】适用场景 : Post类型请求的网站  
    
  【2】参数 : data={}  
   2.1) Form表单数据: 字典  
   2.2) res = requests.post(url=url,data=data,headers=headers)  
     
  【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】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  
   2.1) Network: 抓取网络数据包  
   a> ALL: 抓取所有的网络数据包  
   b> XHR：抓取异步加载的网络数据包  
   c> JS : 抓取所有的JS文件  
   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  
   2.3) 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  
   3.1) 单击左侧网络数据包地址，进入数据包详情，查看右侧  
   3.2) 右侧:  
   a> Headers: 整个请求信息  
   General、Response Headers、Request Headers、Query String、Form Data  
   b> Preview: 对响应内容进行预览  
   c> Response：响应内容

## **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  
  
【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**

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

**4、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

**5、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)  
 html = requests.post(  
 url=self.url,  
 data=data,  
 headers=self.headers  
 ).json()  
 # res.json() 将json格式的字符串转为python数据类型  
 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()

## **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)

## **今日作业**

【1】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)  
 https://www.xicidaili.com/nn/{} # {}为: 1 2 3 4 5  
  
【2】民政部网站案例完善）  
 3.1) 数据存入到 MySQL 数据库，分表存储  
 3.2) 三张表  
 a> 省表(province) : 名称 编号  
 b> 市表(city) : 名称 编号 对应省的编号  
 c> 县表(county) : 名称 编号 对应市的编号  
   
【3】豆瓣电影数据抓取  
 3.1) 地址: 豆瓣电影 - 排行榜 - 剧情  
 3.2) 目标: 电影名称、电影评分  
 3.3) 数据分别存入到MySQL数据库和MongoDB数据库中