Day04回顾

- **requests.get()参数**

```
【1】url
【2】proxies -> {}
    proxies = {
        'http':'http://1.1.1.1:8888',
        'https':'https://1.1.1.1:8888'
    }
【3】timeout
【4】headers
```

- **requests.post()**

```
data ： 字典，Form表单数据
```

- **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
       a> fake_useragent模块
       b> 新建py文件,存放大量User-Agent
       c> 程序中定义列表,存放大量的User-Agent

【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

【5】JS加密 - 有道翻译案例
  5.1) 抓取到对应的JS文件,寻找加密算法
  5.2) 用Python实现加密算法,生成指定的参数
```

```
【6】JS逆向 - 百度翻译案例
  6.1) 抓取到对应的JS文件，寻找加密算法
  6.2) 利用pyexecjs模块执行JS代码

【7】动态加载 - 豆瓣电影、小米应用商店、腾讯招聘
  7.1) F12抓取网络数据包再进行分析,找返回实际json数据的地址
```

- **Ajax动态加载数据抓取流程**

```
【1】F12打开控制台，执行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 ： 找到异步加载的数据包
  2.2) GET请求: Network -> XHR -> URL 和 Query String Parameters(查询参数)
  2.3) POST请求:Network -> XHR -> URL 和 Form Data
```

- **数据抓取最终梳理**

```
【1】响应内容中存在
  1.1) 确认抓取数据在响应内容中存在

  1.2) 分析页面结构，观察URL地址规律
       大体查看响应内容结构,查看是否有更改,以响应内容为准写正则或xpath表达式

  1.3) 开始码代码进行数据抓取

【2】响应内容中不存在
  2.1) 确认抓取数据在响应内容中不存在

  2.2) F12抓包,开始刷新页面或执行某些行为,主要查看XHR异步加载数据包
       a) GET请求: Request URL、Request Headers、Query String Paramters
       b) POST请求:Request URL、Request Headers、FormData

  2.3) 观察查询参数或者Form表单数据规律,如果需要进行进一步抓包分析处理
       a) 比如有道翻译的 salt+sign,抓取并分析JS做进一步处理
       b) 此处注意请求头中的Cookie和Referer以及User-Agent

  2.4) 使用res.json()获取数据,利用列表或者字典的方法获取所需数据
```

- **多线程爬虫梳理**

```
【1】所用到的模块
    1.1) from threading import Thread
    1.2) from threading import Lock
    1.3) from queue import Queue

【2】整体思路
    2.1) 创建URL队列: q = Queue()
    2.2) 产生URL地址,放入队列: q.put(url)
    2.3) 线程事件函数: 从队列中获取地址,开始抓取: url = q.get()
    2.4) 创建多线程,并运行

【3】代码结构
    def __init__(self):
```

```
        """创建URL队列"""
        self.q = Queue()
        self.lock = Lock()

    def url_in(self):
        """生成待爬取的URL地址,入队列"""
        pass

    def parse_html(self):
        """线程事件函数,获取地址,进行数据抓取"""
        while True:
            self.lock.acquire()
            if not self.q.empty():
                url = self.q.get()
                self.lock.release()
            else:
                self.lock.release()
                break

    def run(self):
        self.url_in()
        t_list = []
        for i in range(3):
            t = Thread(target=self.parse_html)
            t_list.append(t)
            t.start()

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

【4】队列要点：q.get()防止阻塞方式
    4.1) 方法1：q.get(block=False)
    4.2) 方法2：q.get(block=True,timeout=3)
    4.3) 方法3：

```
        if not q.empty():
            q.get()
```

Day05笔记

多线程抓取百度图片

- **目标**

```
【1】多线程实现抓取百度图片指定关键字的所有图片
【2】URL地址：https://image.baidu.com/
【3】实现效果
    请输入关键字：赵丽颖
    则：在当前目录下创建 ./images/赵丽颖/ 目录,并将所有赵丽颖图片保存到此目录下
```

- **实现步骤**

```
【1】输入关键字,进入图片页面,滚动鼠标滑轮发现所有图片链接均为动态加载
【2】F12抓取到返回实际数据的json地址如下:
     https://image.baidu.com/search/acjson?
tn=resultjson_com&ipn=rj&ct=201326592&is=&fp=result&queryWord=
{}&cl=2&lm=-1&hd=&latest=&copyright=&ie=utf-8&oe=utf-8&adpicid=&st=-1&z=&ic=0&word=
{}&s=&se=&tab=&width=&height=&face=0&istype=2&qc=&nc=1&fr=&pn=
{}&rn=30&gsm=1e&1599728538999=

【3】分析有效的查询参数如下:
    queryWord: 赵丽颖
    word: 赵丽颖
    pn: 30   pn的值为 0 30 60 90 120 ... ...

【4】图片总数如何获取?
    在json数据中能够查到图片总数
    displayNum: 1037274
```

- **代码实现**

```
import requests
from threading import Thread,Lock
from queue import Queue
from fake_useragent import UserAgent
import os
from urllib import parse
import time

class BaiduImageSpider:
    def __init__(self):
        self.keyword = input('请输入关键字:')
        self.directory = './images/{}/'.format(self.keyword)
        if not os.path.exists(self.directory):
            os.makedirs(self.directory)
        # 编码
        self.params = parse.quote(self.keyword)
        self.url = 'https://image.baidu.com/search/acjson?
tn=resultjson_com&ipn=rj&ct=201326592&is=&fp=result&queryWord=
{}&cl=2&lm=-1&hd=&latest=&copyright=&ie=utf-8&oe=utf-8&adpicid=&st=-1&z=&ic=0&word=
{}&s=&se=&tab=&width=&height=&face=0&istype=2&qc=&nc=1&fr=&pn=
{}&rn=30&gsm=5a&1599724879800='
        # 创建url队列、线程锁
        self.url_queue = Queue()
        self.lock = Lock()

    def get_html(self, url):
        """请求功能函数"""
        headers = {'User-Agent' : UserAgent().random}
        html = requests.get(url=url, headers=headers).json()

        return html

    def get_total_image(self):
        """获取图片总数量"""
        total_page_url = self.url.format(self.params, self.params, 0)
```

```
        total_page_html = self.get_html(url=total_page_url)
        total_page_nums = total_page_html['displayNum']

        return total_page_nums

    def url_in(self):
        total_image_nums = self.get_total_image()
        for pn in range(0, total_image_nums, 30):
            page_url = self.url.format(self.params, self.params, pn)
            self.url_queue.put(page_url)

    def parse_html(self):
        """线程事件函数"""
        while True:
            # 加锁
            self.lock.acquire()
            if not self.url_queue.empty():
                page_url = self.url_queue.get()
                # 释放锁
                self.lock.release()
                html = self.get_html(url=page_url)
                for one_image_dict in html['data']:
                    try:
                        image_url = one_image_dict['hoverURL']
                        self.save_image(image_url)
                    except Exception as e:
                        continue
            else:
                self.lock.release()
                break

    def save_image(self, image_url):
        """保存一张图片到本地"""
        headers = {'User-Agent':UserAgent().random}
        image_html = requests.get(url=image_url, headers=headers).content
        # 加锁、释放锁
        self.lock.acquire()
        filename = self.directory + image_url[-24:]
        self.lock.release()
        with open(filename, 'wb') as f:
            f.write(image_html)
        print(filename, '下载成功')

    def run(self):
        # 让URL地址入队列
        self.url_in()
        # 创建多线程
        t_list = []
        for i in range(1):
            t = Thread(target=self.parse_html)
            t_list.append(t)
            t.start()

        for t in t_list:
            t.join()

if __name__ == '__main__':
```

```
    start_time = time.time()
    spider = BaiduImageSpider()
    spider.run()
    end_time = time.time()
    print('time:%.2f' % (end_time - start_time))
```

selenium+PhantomJS/Chrome/Firefox

- **selenium**

```
【1】定义
    1.1) 开源的Web自动化测试工具

【2】用途
    2.1) 对Web系统进行功能性测试,版本迭代时避免重复劳动
    2.2) 兼容性测试(测试web程序在不同操作系统和不同浏览器中是否运行正常)
    2.3) 对web系统进行大数量测试

【3】特点
    3.1) 可根据指令操控浏览器
    3.2) 只是工具，必须与第三方浏览器结合使用

【4】安装
    4.1) Linux: sudo pip3 install selenium
    4.2) Windows: python -m pip install selenium
```

- **PhantomJS浏览器**

```
phantomjs为无界面浏览器(又称无头浏览器)，在内存中进行页面加载,高效
```

- **环境安装**

```
【1】下载驱动
    2.1) chromedriver ： 下载对应版本
         http://npm.taobao.org/mirrors/chromedriver/
    2.2) geckodriver
         https://github.com/mozilla/geckodriver/releases
    2.3) phantomjs
         https://phantomjs.org/download.html

【2】添加到系统环境变量
    2.1) windows: 拷贝到Python安装目录的Scripts目录中
         windows查看python安装目录(cmd命令行): where python
    2.2) Linux :  拷贝到/usr/bin目录中 : sudo cp chromedriver /usr/bin/

【3】Linux中需要修改权限
    sudo chmod 777 /usr/bin/chromedriver

【4】验证
    4.1) Ubuntu | Windows
        from selenium import webdriver
        webdriver.Chrome()
```

```
        webdriver.Firefox()

    4.2) Mac
        from selenium import webdriver
        webdriver.Chrome(executable_path='/Users/xxx/chromedriver')
        webdriver.Firefox(executable_path='/User/xxx/geckodriver')
```

- **示例代码**

```
"""示例代码一：使用 selenium+浏览器 打开百度"""

# 导入seleinum的webdriver接口
from selenium import webdriver
import time

# 创建浏览器对象
driver = webdriver.Chrome()
driver.get('http://www.baidu.com/')
# 5秒钟后关闭浏览器
time.sleep(5)
driver.quit()
```

```
"""示例代码二：打开百度，搜索赵丽颖，点击搜索，查看"""

from selenium import webdriver
import time

# 1.创建浏览器对象 - 已经打开了浏览器
driver = webdriver.Chrome()
# 2.输入：http://www.baidu.com/
driver.get('http://www.baidu.com/')
# 3.找到搜索框,向这个节点发送文字：赵丽颖
driver.find_element_by_xpath('//*[@id="kw"]').send_keys('赵丽颖')
# 4.找到 百度一下 按钮,点击一下
driver.find_element_by_xpath('//*[@id="su"]').click()
```

- **浏览器对象(browser)方法**

```
【1】driver.get(url=url)    - 地址栏输入url地址并确认
【2】driver.quit()          - 关闭浏览器
【3】driver.close()         - 关闭当前页
【4】driver.page_source     - HTML结构源码
【5】driver.page_source.find('字符串')
    从html源码中搜索指定字符串,没有找到返回：-1,经常用于判断是否为最后一页
【6】driver.maximize_window() - 浏览器窗口最大化
```

- **定位节点八种方法**

```
【1】单元素查找('结果为1个节点对象')
    1.1) 【最常用】driver.find_element_by_id('id属性值')
    1.2) 【最常用】driver.find_element_by_name('name属性值')
    1.3) 【最常用】driver.find_element_by_class_name('class属性值')
    1.4) 【最万能】driver.find_element_by_xpath('xpath表达式')
```

```
    1.5) 【匹配a节点时常用】driver.find_element_by_link_text('链接文本')
    1.6) 【匹配a节点时常用】driver.find_element_by_partical_link_text('部分链接文本')
    1.7) 【最没用】driver.find_element_by_tag_name('标记名称')
    1.8) 【较常用】driver.find_element_by_css_selector('css表达式')

【2】多元素查找('结果为[节点对象列表]')
    2.1) driver.find_elements_by_id('id属性值')
    2.2) driver.find_elements_by_name('name属性值')
    2.3) driver.find_elements_by_class_name('class属性值')
    2.4) driver.find_elements_by_xpath('xpath表达式')
    2.5) driver.find_elements_by_link_text('链接文本')
    2.6) driver.find_elements_by_partical_link_text('部分链接文本')
    2.7) driver.find_elements_by_tag_name('标记名称')
    2.8) driver.find_elements_by_css_selector('css表达式')

【3】注意
    当属性值中存在 空格 时,我们要使用 . 去代替空格
    页面中class属性值为: btn btn-account
    driver.find_element_by_class_name('btn.btn-account').click()
```

- **猫眼电影示例**

```python
from selenium import webdriver
import time

url = 'https://maoyan.com/board/4'
driver = webdriver.Chrome()
driver.get(url)

def get_data():
    # 基准xpath: [<selenium xxx li at xxx>,<selenium xxx li at>]
    li_list = driver.find_elements_by_xpath('//*[@id="app"]/div/div/div[1]/dl/dd')
    for li in li_list:
        item = {}
        # info_list: ['1', '霸王别姬', '主演: 张国荣', '上映时间: 1993-01-01', '9.5']
        info_list = li.text.split('\n')
        item['number'] = info_list[0]
        item['name'] = info_list[1]
        item['star'] = info_list[2]
        item['time'] = info_list[3]
        item['score'] = info_list[4]

        print(item)

while True:
    get_data()
    try:
        driver.find_element_by_link_text('下一页').click()
        time.sleep(2)
    except Exception as e:
        print('恭喜你!抓取结束')
        driver.quit()
        break
```

- **节点对象操作**

```
【1】node.send_keys('')  - 向文本框发送内容
【2】node.click()       - 点击
【3】node.get_attribute('属性名')  -  获取节点的属性值
```

chromedriver设置无界面模式

```python
from selenium import webdriver

options = webdriver.ChromeOptions()
# 添加无界面参数
options.add_argument('--headless')
driver = webdriver.Chrome(options=options)
```

selenium - 鼠标操作

```python
from selenium import webdriver
# 导入鼠标事件类
from selenium.webdriver import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.baidu.com/')

# 移动到 设置，perform()是真正执行操作，必须有
element = driver.find_element_by_xpath('//*[@id="u1"]/a[8]')
ActionChains(driver).move_to_element(element).perform()

# 单击，弹出的Ajax元素，根据链接节点的文本内容查找
driver.find_element_by_link_text('高级搜索').click()
```

selenium - 切换页面

- **适用网站+应对方案**

```
【1】适用网站类型
    页面中点开链接出现新的窗口，但是浏览器对象driver还是之前页面的对象，需要切换到不同的窗口进行操作

【2】应对方案 - driver.switch_to.window()

    # 获取当前所有句柄（窗口）- [handle1,handle2]
    all_handles = driver.window_handles
    # 切换browser到新的窗口，获取新窗口的对象
    driver.switch_to.window(all_handles[1])
```

- **民政部网站案例-selenium**

```
"""
适用selenium+Chrome抓取民政部行政区划代码
http://www.mca.gov.cn/article/sj/xzqh/2019/
"""
from selenium import webdriver

class GovSpider(object):
    def __init__(self):
        # 设置无界面
        options = webdriver.ChromeOptions()
        options.add_argument('--headless')
        # 添加参数
        self.driver = webdriver.Chrome(options=options)
        self.one_url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'

    def get_incr_url(self):
        self.driver.get(self.one_url)
        # 提取最新链接节点对象并点击

 self.driver.find_element_by_xpath('//td[@class="arlisttd"]/a[contains(@title,"代码")]').click()
        # 切换句柄
        all_handlers = self.driver.window_handles
        self.driver.switch_to.window(all_handlers[1])
        self.get_data()

    def get_data(self):
        tr_list = self.driver.find_elements_by_xpath('//tr[@height="19"]')
        for tr in tr_list:
            code = tr.find_element_by_xpath('./td[2]').text.strip()
            name = tr.find_element_by_xpath('./td[3]').text.strip()
            print(name,code)

    def run(self):
        self.get_incr_url()
        self.driver.quit()

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

selenium - iframe

- **特点+方法**

```
【1】特点
    网页中嵌套了网页，先切换到iframe，然后再执行其他操作

【2】处理步骤
    2.1) 切换到要处理的Frame
```

```
    2.2) 在Frame中定位页面元素并进行操作
    2.3) 返回当前处理的Frame的上一级页面或主页面

【3】常用方法
    3.1) 切换到frame   -   driver.switch_to.frame(frame节点对象)
    3.2) 返回上一级    -   driver.switch_to.parent_frame()
    3.3) 返回主页面    -   driver.switch_to.default_content()

【4】使用说明
    4.1) 方法一：默认支持id和name属性值 ： switch_to.frame(id属性值|name属性值)
    4.2) 方法二：
        a> 先找到frame节点 ： frame_node = driver.find_element_by_xpath('xxxx')
        b> 在切换到frame    ： driver.switch_to.frame(frame_node)
```

- **示例1 - 登录豆瓣网**

```
"""
登录豆瓣网
"""
from selenium import webdriver
import time

# 打开豆瓣官网
driver = webdriver.Chrome()
driver.get('https://www.douban.com/')

# 切换到iframe子页面
login_frame = driver.find_element_by_xpath('//*[@id="anony-reg-new"]/div/div[1]/iframe')
driver.switch_to.frame(login_frame)

# 密码登录 + 用户名 + 密码 + 登录豆瓣
driver.find_element_by_xpath('/html/body/div[1]/div[1]/ul[1]/li[2]').click()
driver.find_element_by_xpath('//*[@id="username"]').send_keys('自己的用户名')
driver.find_element_by_xpath('//*[@id="password"]').send_keys('自己的密码')
driver.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div[5]/a').click()
time.sleep(3)

# 点击我的豆瓣
driver.find_element_by_xpath('//*[@id="db-nav-sns"]/div/div/div[3]/ul/li[2]/a').click()
```

- **selenium+phantomjs|chrome|firefox小总结**

```
【1】设置无界面模式
    options = webdriver.ChromeOptions()
    options.add_argument('--headless')
    driver = webdriver.Chrome(excutable_path='/home/tarena/chromedriver',options=options)

【2】browser执行JS脚本
    driver.execute_script('window.scrollTo(0,document.body.scrollHeight)')

【3】鼠标操作
    from selenium.webdriver import ActionChains
    ActionChains(driver).move_to_element('node').perform()
```

```
【4】切换句柄 - switch_to.frame(handle)
    all_handles = driver.window_handles
    driver.switch_to.window(all_handles[1])

【5】iframe子页面
    driver.switch_to.frame(frame_node)
```

- **lxml中的xpath 和 selenium中的xpath的区别**

```
【1】lxml中的xpath用法 - 推荐自己手写
    div_list = p.xpath('//div[@class="abc"]/div')
    item = {}
    for div in div_list:
        item['name'] = div.xpath('.//a/@href')[0]
        item['likes'] = div.xpath('.//a/text()')[0]

【2】selenium中的xpath用法 - 推荐copy - copy xpath
    div_list = browser.find_elements_by_xpath('//div[@class="abc"]/div')
    item = {}
    for div in div_list:
        item['name'] = div.find_element_by_xpath('.//a').get_attribute('href')
        item['likes'] = div.find_element_by_xpath('.//a').text
```

今日作业

```
【1】使用selenium+浏览器 获取有道翻译结果
【2】使用selenium+浏览器 登录网易qq邮箱 : https://mail.qq.com/
【3】使用selenium+浏览器 登录网易163邮箱 : https://mail.163.com/
```