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

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

使用execjs模块去执行对应的JS代码

【7】JS跳转

点击时跳转到新的URL地址，比如说：民政部案例

- **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

- **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

- **json模块**

```
【1】抓取的json数据转为python数据类型
    1.1) html = json.loads('[{},{},{}]')
    1.2) html = requests.get(url=url,headers=headers).json()
    1.3) html = requests.post(url=url,data=data,headers=headers).json()

【2】抓取数据保存到json文件
    import json
    with open('xxx.json','w') as f:
        json.dump([{},{},{}],f,ensure_ascii=False)
```

- **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是否存在

   1.2) 分析页面结构，观察URL地址规律
        a) 大体查看响应内容结构,查看是否有更改 -- (百度视频案例)
        b) 查看页面跳转时URL地址变化,查看是否新跳转 -- (民政部案例)

   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是否存在

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

JS逆向解决问题思路

- **案例**

百度翻译案例（思路同有道翻译案例）
1、F12抓包
2、找到Post URL、Request Headers、Form Data
3、Form Data中有 sign，抓取JS寻找加密JS代码
4、通过Search，搜索 sign: 关键字找到了JS
   sign: y(n)
5、断点调试,进入到 y(n) 这个函数中，发现JS加密代码相当复杂,没有办法读懂或者用Python实现,使用execjs模块进行JS逆向处理

- **execjs模块使用**

```
【1】安装
    sudo pip3 install pyexecjs
【2】使用方法
    import execjs
    2.1> 先拿到js的字符串
        with open('xxx.js', 'r') as f:
            js_code = f.read()
    2.2> 创建编译对象
        js_object =
execjs.compile(js_code)
    2.3> 调用js中的函数
        js_object.eval('e("hello")')
```

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
browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
# 5秒钟后关闭浏览器
time.sleep(5)
browser.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
browser = webdriver.Chrome()
# 2.输入: http://www.baidu.com/
browser.get('http://www.baidu.com/')
# 3.找到搜索框,向这个节点发送文字: 赵丽颖
browser.find_element_by_xpath('//*[@id="kw"]').send_keys('赵丽颖')
# 4.找到 百度一下 按钮,点击一下
browser.find_element_by_xpath('//*[@id="su"]').click()
```

- **浏览器对象(browser)方法**

```
【1】browser.get(url=url)    - 地址栏输入url地址并确认
【2】browser.quit()          - 关闭浏览器
【3】browser.close()         - 关闭当前页
【4】browser.page_source     - HTML结构源码
【5】browser.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】browser.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】
browser.find_element_by_id('id属性值')
    1.2) 【最常用】
browser.find_element_by_name('name属性值')
    1.3) 【最常用】
browser.find_element_by_class_name('class属性值')
    1.4) 【最万能】
browser.find_element_by_xpath('xpath表达式')
    1.5) 【匹配a节点时常用】
browser.find_element_by_link_text('链接文本')
```

```
    1.6) 【匹配a节点时常用】
browser.find_element_by_partical_link_text('部分链接文本')
    1.7) 【最没用】
browser.find_element_by_tag_name('标记名称')
    1.8) 【较常用】
browser.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)
browser.find_elements_by_id('id属性值')
    2.2)
browser.find_elements_by_name('name属性值')
    2.3)
browser.find_elements_by_class_name('class属性值')
    2.4)
browser.find_elements_by_xpath('xpath表达式')
    2.5)
browser.find_elements_by_link_text('链接文本')
    2.6)
browser.find_elements_by_partical_link_text('部分链接文本')
```

```
    2.7)
browser.find_elements_by_tag_name('标记名称')
    2.8)
browser.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**

```
from selenium import webdriver
import time

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

def get_data():
    # 基准xpath: [<selenium xxx li at xxx>,<selenium xxx li at>]
```

```
    li_list =
browser.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:

 browser.find_element_by_link_text('下一
页').click()
        time.sleep(2)
    except Exception as e:
        print('恭喜你!抓取结束')
        browser.quit()
        break
```

- **节点对象操作**

```
【1】文本框操作
    1.1) node.send_keys('')  - 向文本框发送内容
    1.2) node.clear()        - 清空文本
    1.3) node.get_attribute('value') - 获取文本内容

【2】按钮操作
    1.1) node.click()        - 点击
    1.2) node.is_enabled() - 判断按钮是否可用<button>
    1.3) node.get_attribute('value') - 获取按钮文本
```

chromedriver设置无界面模式

```
from selenium import webdriver

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

==selenium - 鼠标操作==

```
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(elem
ent).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browser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 -
browser.switch_to.window()

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

 browser.switch_to.window(all_handles[1
])

【3】扩展
    要求：把抓取的数据存入MySQL数据库
    设计：分表存储(3张表)
    表1：省表(province)：p_name  p_code
    表2：市表(city)：    c_name  c_code
 cp_code
    表3：县表(county)：  x_name  x_code
 xc_code
```

- **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/2
019/
"""
```

```
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.browser =
webdriver.Chrome(options=options)
        self.one_url =
'http://www.mca.gov.cn/article/sj/xzqh/
2019/'

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

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

 self.browser.switch_to.window(all_hand
lers[1])
```

```
            self.get_data()

    def get_data(self):
        tr_list =
self.browser.find_elements_by_xpath('//
tr[@height="19"]')
        for tr in tr_list:
            code =
tr.find_element_by_xpath('./td[2]').tex
t.strip()
            name =
tr.find_element_by_xpath('./td[3]').tex
t.strip()
            print(name,code)

    def run(self):
        self.get_incr_url()
        self.browser.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   - browser.switch_to.frame(frame节点对象)
    3.2) 返回上一级    - browser.switch_to.parent_frame()
    3.3) 返回主页面    - browser.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 = browser.find_element_by_xpath('xxxx')
        b> 在切换到frame   : browser.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
browser = webdriver.Chrome()
browser.get('https://www.douban.com/')

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

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

```
# 点击我的豆瓣
browser.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')
    browser =
webdriver.Chrome(excutable_path='/home/
tarena/chromedriver',options=options)

【2】browser执行JS脚本

 browser.execute_script('window.scrollT
o(0,document.body.scrollHeight)')

【3】键盘操作
    from selenium.webdriver.common.keys
import Keys

【4】鼠标操作
    from selenium.webdriver import
ActionChains
```

```
 ActionChains(browser).move_to_element(
'node').perform()

【5】切换句柄 - switch_to.frame(handle)
    all_handles =
browser.window_handles

 browser.switch_to.window(all_handles[1
])
    # 开始进行数据抓取
    browser.close()

 browser.switch_to.window(all_handles[0
])

【6】iframe子页面
    browser.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_a
ttribute('href')
        item['likes'] =
div.find_element_by_xpath('.//a').text
```

scrapy框架

- **定义**

异步处理框架,可配置和可扩展程度非常高,Python中使用最广泛的爬虫框架

- **安装**

```
【1】Ubuntu安装
    1.1) 安装依赖包
        a> sudo apt-get install libffi-dev
        b> sudo apt-get install libssl-dev
        c> sudo apt-get install libxml2-dev
        d> sudo apt-get install python3-dev
        e> sudo apt-get install libxslt1-dev
        f> sudo apt-get install zlib1g-dev
        g> sudo pip3 install -I -U service_identity

    1.2) 安装scrapy框架
        a> sudo pip3 install Scrapy

【2】Windows安装
    2.1) cmd命令行(管理员): python -m pip install Scrapy
    【注意】: 如果安装过程中报如下错误
```

```
            'Error: Microsoft Vistual
C++ 14.0 is required xxx'
            则安装Windows下的Microsoft
Vistual C++ 14.0 即可（笔记spiderfiles中
有）
```

- **Scrapy框架五大组件**

```
【1】引擎(Engine)       ：整个框架核心
【2】调度器(Scheduler) ：维护请求队列
【3】下载器(Downloader)：获取响应对象
【4】爬虫文件(Spider)  ：数据解析提取
【5】项目管道(Pipeline)：数据入库处理
**********************************
【中间件1】：下载器中间件(Downloader
Middlewares) ：引擎->下载器,包装请求(随机代
理等)
【中间件2】：蜘蛛中间件(Spider Middlewares)
：引擎->爬虫文件,可修改响应对象属性
```

- **scrapy爬虫工作流程**

【1】爬虫项目启动，由引擎向爬虫程序索要第一批要爬取的URL，交给调度器去入队列
【2】调度器处理请求后出队列，通过下载器中间件交给下载器去下载
【3】下载器得到响应对象后，通过蜘蛛中间件交给爬虫程序
【4】爬虫程序进行数据提取：
    4.1) 数据交给管道文件去入库处理
    4.2) 对于需要继续跟进的URL，再次交给调度器入队列，依次循环

- **scrapy常用命令**

【1】创建爬虫项目
```
scrapy startproject 项目名
```

【2】创建爬虫文件
```
scrapy genspider 爬虫名 域名
```

【3】运行爬虫
```
scrapy crawl 爬虫名
```

- **scrapy项目目录结构**

```
Baidu                          # 项目文件夹
├── Baidu                      # 项目目录
│   ├── items.py               # 定义数据结构
│   ├── middlewares.py         # 中间件
│   ├── pipelines.py           # 数据处理
│   ├── settings.py            # 全局配置
│   └── spiders
│       ├── baidu.py           # 爬虫文件
└── scrapy.cfg                 # 项目基本配置文件
```

- **settings.py常用变量**

```
【1】USER_AGENT = 'Mozilla/5.0'

【2】ROBOTSTXT_OBEY = False
    是否遵循robots协议,一般我们一定要设置为False

【3】CONCURRENT_REQUESTS = 32
    最大并发量,默认为16

【4】DOWNLOAD_DELAY = 0.5
    下载延迟时间: 访问相邻页面的间隔时间,降低数据抓取的频率

【5】COOKIES_ENABLED = False | True
    Cookie默认是禁用的，取消注释则 启用Cookie，即：True和False都是启用Cookie
```

【6】DEFAULT_REQUEST_HEADERS = {}
    请求头,相当于
requests.get(headers=headers)

小试牛刀

【1】执行3条命令,创建项目基本结构
```
scrapy startproject Baidu
cd Baidu
scrapy genspider baidu www.baidu.com
```

【2】完成爬虫文件: spiders/baidu.py
```
import scrapy
class BaiduSpider(scrapy.Spider):
    name = 'baidu'
    allowed_domains = ['www.baidu.com']
    start_urls = ['http://www.baidu.com/']

    def parse(self,response):
        r_list = response.xpath('/html/head/title/text()').extract()[0]
        print(r_list)
```

【3】完成settings.py配置
```
3.1) ROBOTSTXT_OBEY = False
```

```
    3.2) DEFAULT_REQUEST_HEADERS = {
        'User-Agent' : 'Mozilla/5.0'
    }

【4】运行爬虫
    4.1) 创建run.py(和scrapy.cfg同路径)
    4.2) run.py
        from scrapy import cmdline
        cmdline.execute('scrapy crawl
baidu'.split())

【5】执行 run.py 运行爬虫
```

瓜子二手车直卖网 - 一级页面

- **目标**

```
【1】抓取瓜子二手车官网二手车收据（我要买车）

【2】URL地址：
https://www.guazi.com/langfang/buy/o{}/#b
read
    URL规律: o1  o2  o3  o4  o5  ... ...

【3】所抓数据
    3.1) 汽车链接
    3.2) 汽车名称
    3.3) 汽车价格
```

今日作业

【1】使用selenium+浏览器 获取有道翻译结果（注意预留时间）
【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/
【4】使用selenium+浏览器抓取网易云音乐排行榜中的歌曲信息
https://music.163.com/#/discover/toplist
【5】熟记scrapy的五大组件,以及工作流程,能够描述的很清楚