

Day06回顾

多线程写入同一文件

注意使用线程锁

```
1 from threading import Lock
2 lock = Lock()
3 f = open('xxx.txt', 'a')
4
5 lock.acquire()
6 f.write(string)
7 lock.release()
8
9 f.close()
```

cookie模拟登陆

```
1 1、适用网站类型：爬取网站页面时需要登录后才能访问，否则获取不到页面的实际响应数据
2 2、方法1（利用cookie）
3     1、先登录成功1次，获取到携带登陆信息的Cookie（处理headers）
4     2、利用处理的headers向URL地址发请求
5 3、方法2（利用session会话保持）
6     1、实例化session对象
7         session = requests.session()
8     2、先post：session.post(post_url, data=post_data, headers=headers)
9         1、登陆，找到POST地址：form -> action对应地址
10        2、定义字典，创建session实例发送请求
11            # 字典key：<input>标签中name的值(email, password)
12            # post_data = {'email': '', 'password': ''}
13    3、再get：session.get(url, headers=headers)
```

三个池子

```
1 1、User-Agent池
2 2、代理IP池
3 3、cookie池
```

解析模块汇总

re、lxml+xpath、json

```
1 # re
2 import re
3 pattern = re.compile(r'',re.S)
4 r_list = pattern.findall(html)
5
6 # lxml+xpath
7 from lxml import etree
8 parse_html = etree.HTML(html)
9 r_list = parse_html.xpath('')
10
11 # json
12 # 响应内容由json转为python
13 html = json.loads(res.text)
14 # 所抓数据保存到json文件
15 with open('xxx.json','a') as f:
16     json.dump(item_list,f,ensure_ascii=False)
17 # 或
18 f = open('xxx.json','a')
19 json.dump(item_list,f,ensure_ascii=False)
20 f.close()
```

selenium+phantomjs/chrome/firefox

■ 特点

- 1、简单，无需去详细抓取分析网络数据包，使用真实浏览器
- 2、需要等待页面元素加载，需要时间，效率低

■ 安装

- 1、下载、解压
- 2、添加到系统环境变量
 - # windows: 拷贝到Python安装目录的Scripts目录中
 - # Linux : 拷贝到/usr/bin目录中
- 3、Linux中修改权限
 - # sudo -i
 - # cd /usr/bin/
 - # chmod +x phantomjs

■ 使用流程

```
1 from selenium import webdriver
2
3 # 1、创建浏览器对象
4 # 2、输入网址
5 # 3、查找节点
6 # 4、做对应操作
7 # 5、关闭浏览器
```

■ 重要知识点

```
1 1、browser.page_source
2 2、browser.page_source.find('')
3 3、node.send_keys('')
4 4、node.click()
5 5、find_element AND find_elements
6 6、browser.execute_script('javascript')
7 7、browser.quit()
```

Day07笔记

京东爬虫案例

■ 目标

```
1 1、目标网址：https://www.jd.com/
2 2、抓取目标：商品名称、商品价格、评价数量、商品商家
```

■ 实现步骤

1、找节点

```
1 1、首页搜索框：//*[@id="key"]
2 2、首页搜索按钮：//*[@id="search"]/div/div[2]/button
3 3、商品页的商品信息节点对象列表：//*[@id="J_goodsList"]/ul/li
```

2、执行JS脚本，获取动态加载数据

```
1 browser.execute_script(
2     'window.scrollTo(0,document.body.scrollHeight)'
3 )
```

3、代码实现

```
1 |
```

chromedriver设置无界面模式

```
1 from selenium import webdriver
2
3 options = webdriver.ChromeOptions()
4 # 添加无界面参数
5 options.add_argument('--headless')
6 browser = webdriver.Chrome(options=options)
7 browser.get('http://www.baidu.com/')
8 browser.save_screenshot('baidu.png')
```

selenium - 键盘操作

示例

```
1 from selenium.webdriver.common.keys import Keys
2
3 browser = webdriver.Chrome()
4 browser.get('http://www.baidu.com/')
5 # 1、在搜索框中输入"selenium"
6 browser.find_element_by_id('kw').send_keys('赵丽颖')
7 # 2、输入空格
8 browser.find_element_by_id('kw').send_keys(Keys.SPACE)
9 # 3、Ctrl+a 模拟全选
10 browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'a')
11 # 4、Ctrl+c 模拟复制
12 browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'c')
13 # 5、Ctrl+v 模拟粘贴
14 browser.find_element_by_id('kw').send_keys(Keys.CONTROL, 'v')
15 # 6、输入回车,代替 搜索 按钮
16 browser.find_element_by_id('kw').send_keys(Keys.ENTER)
```

selenium - 鼠标操作

示例

```
1 from selenium import webdriver
2 # 导入鼠标事件类
3 from selenium.webdriver import ActionChains
4
5 driver = webdriver.Chrome()
6 driver.get('http://www.baidu.com/')
7 #输入selenium 搜索
8 driver.find_element_by_id('kw').send_keys('赵丽颖')
9 driver.find_element_by_id('su').click()
10
11 #移动到 设置, perform()是真正执行操作, 必须有
12 element = driver.find_element_by_name('tj_settingicon')
```

```
13 | ActionChains(driver).move_to_element(element).perform()
14 |
15 | #单击, 弹出的Ajax元素, 根据链接节点的文本内容查找
16 | driver.find_element_by_link_text('高级搜索').click()
```

selenium - 切换页面

▪ 适用网站

1 | 页面中点开链接出现新的页面, 但是浏览器对象browser还是之前页面的对象

▪ 应对方案

```
1 | # 获取当前所有句柄 (窗口)
2 | all_handles = browser.window_handles
3 | # 切换browser到新的窗口, 获取新窗口的对象
4 | browser.switch_to_window(all_handles[1])
```

▪ 民政部网站案例

目标

1 | 将民政区划代码爬取到数据库中, 按照层级关系 (分表 -- 省表、市表、县表)

数据库中建表

```
1 | # 建库
2 | create database govdb charset utf8;
3 | use govdb;
4 | # 建表
5 | create table province(
6 | p_name varchar(20),
7 | p_code varchar(20)
8 | )charset=utf8;
9 | create table city(
10 | c_name varchar(20),
11 | c_code varchar(20),
12 | c_father_code varchar(20)
13 | )charset=utf8;
14 | create table county(
15 | x_name varchar(20),
16 | x_code varchar(20),
17 | x_father_code varchar(20)
18 | )charset=utf8;
```

思路

- 1、selenium+Chrome打开一级页面，并提取二级页面最新链接
- 2、增量爬取：和数据库version表中进行比对，确定之前是否爬过（是否有更新）
- 3、如果没有更新，直接提示用户，无须继续爬取
- 4、如果有更新，则删除之前表中数据，重新爬取并插入数据库表
- 5、最终完成后：断开数据库连接，关闭浏览器

代码实现

```
1
```

SQL命令练习

```
1 # 1. 查询所有省市县信息（多表查询实现）
2 select province.p_name,city.c_name,county.x_name from province,city,county where
   province.p_code=city.c_father_code and city.c_code=county.x_father_code;
3 # 2. 查询所有省市县信息（连接查询实现）
4 select province.p_name,city.c_name,county.x_name from province inner join city on
   province.p_code=city.c_father_code inner join county on city.c_code=county.x_father_code;
```

selenium - Web客户端验证

弹窗中的用户名和密码如何输入？

```
1 不用输入，在URL地址中填入就可以
```

示例: 爬取某一天笔记

```
1 from selenium import webdriver
2
3 url = 'http://tarenacode:code_2013@code.tarena.com.cn/AIDCode/aid1904/15-
   spider/spider_day06_note.zip'
4 browser = webdriver.Chrome()
5 browser.get(url)
```

selenium - iframe子框架

特点

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

方法

```
1 browser.switch_to.iframe(iframe_element)
```

示例 - 登录qq邮箱

```

1 from selenium import webdriver
2 import time
3
4 driver = webdriver.Chrome()
5 driver.get('https://mail.qq.com/')
6
7 # 切换到iframe子框架
8 login_frame = driver.find_element_by_id('login_frame')
9 driver.switch_to.frame(login_frame)
10
11 # 用户名+密码+登录
12 driver.find_element_by_id('u').send_keys('2621470058')
13 driver.find_element_by_id('p').send_keys('密码')
14 driver.find_element_by_id('login_button').click()
15
16 # 预留页面记载时间
17 time.sleep(5)
18
19 # 提取数据
20 ele = driver.find_element_by_id('useralias')
21 print(ele.text)

```

百度翻译破解案例

目标

- 1 破解百度翻译接口，抓取翻译结果数据

实现步骤

■ 1、F12抓包,找到json的地址,观察查询参数

```

1 1、POST地址: https://fanyi.baidu.com/v2transapi
2 2、Form表单数据 (多次抓取在变的字段)
3   from: zh
4   to: en
5   sign: 54706.276099 #这个是如何生成的?
6   token: a927248ae7146c842bb4a94457ca35ee # 基本固定,但也想办法获取

```

■ 2、抓取相关JS文件

- 1 右上角 - 搜索 - sign: - 找到具体JS文件(index_c8a141d.js) - 格式化输出

3、在JS中寻找sign的生成代码

- 1 在格式化输出的JS代码中搜索: sign: 找到如下JS代码: sign: m(a),
- 2 通过设置断点, 找到m(a)函数的位置, 即生成sign的具体函数
- 3 # 1. a 为要翻译的单词
- 4 # 2. 鼠标移动到 m(a) 位置处, 点击可进入具体m(a)函数代码块

4、生成sign的m(a)函数具体代码如下(在一个大的define中)

```
1 function a(r) {
2     if (Array.isArray(r)) {
3         for (var o = 0, t = Array(r.length); o < r.length; o++)
4             t[o] = r[o];
5         return t
6     }
7     return Array.from(r)
8 }
9 function n(r, o) {
10     for (var t = 0; t < o.length - 2; t += 3) {
11         var a = o.charAt(t + 2);
12         a = a >= "a" ? a.charCodeAt(0) - 87 : Number(a),
13         a = "+" === o.charAt(t + 1) ? r >>> a : r << a,
14         r = "+" === o.charAt(t) ? r + a & 4294967295 : r ^ a
15     }
16     return r
17 }
18 function e(r) {
19     var o = r.match(/[\uD800-\uDBFF][\uDC00-\uDFFF]/g);
20     if (null === o) {
21         var t = r.length;
22         t > 30 && (r = "" + r.substr(0, 10) + r.substr(Math.floor(t / 2) - 5, 10) +
23 r.substr(-10, 10))
24     } else {
25         for (var e = r.split(/[\uD800-\uDBFF][\uDC00-\uDFFF]/), C = 0, h = e.length, f = []; h
26 > C; C++)
27             "" !== e[C] && f.push.apply(f, a(e[C].split(""))),
28             C !== h - 1 && f.push(o[C]);
29         var g = f.length;
30         g > 30 && (r = f.slice(0, 10).join("") + f.slice(Math.floor(g / 2) - 5, Math.floor(g /
31 2) + 5).join("") + f.slice(-10).join(""))
32     }
33     // var u = void 0
34     // , l = "" + String.fromCharCode(103) + String.fromCharCode(116) +
35 String.fromCharCode(107);
36 // u = null !== i ? i : (i = window[l] || "") || "";
37 // 断点调试,然后从网页源码中找到 window.gtk的值
38 var u = '320305.131321201'
39
40     for (var d = u.split("."), m = Number(d[0]) || 0, s = Number(d[1]) || 0, S = [], c = 0, v
41 = 0; v < r.length; v++) {
42         var A = r.charCodeAt(v);
43         128 > A ? S[c++] = A : (2048 > A ? S[c++] = A >> 6 | 192 : (55296 === (64512 & A) && v
44 + 1 < r.length && 56320 === (64512 & r.charCodeAt(v + 1)) ? (A = 65536 + ((1023 & A) << 10) +
45 (1023 & r.charCodeAt(++v)),
46         S[c++] = A >> 18 | 240,
47         S[c++] = A >> 12 & 63 | 128) : S[c++] = A >> 12 | 224,
48         S[c++] = A >> 6 & 63 |
49 128),
50         S[c++] = 63 & A | 128)
51     }
```



```

44     for (var p = m, F = "" + String.fromCharCode(43) + String.fromCharCode(45) +
        String.fromCharCode(97) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
        String.fromCharCode(54)), D = "" + String.fromCharCode(43) + String.fromCharCode(45) +
        String.fromCharCode(51) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
        String.fromCharCode(98)) + ("" + String.fromCharCode(43) + String.fromCharCode(45) +
        String.fromCharCode(102)), b = 0; b < S.length; b++)
45         p += S[b],
46         p = n(p, F);
47     return p = n(p, D),
48         p ^= s,
49         0 > p && (p = (2147483647 & p) + 2147483648),
50         p %= 1e6,
51         p.toString() + "." + (p ^ m)
52 }

```

■ 5、直接将代码写入本地js文件,利用pyexecjs模块执行js代码进行调试

```

1  import execjs
2
3  with open('node.js', 'r') as f:
4      js_data = f.read()
5      # 创建对象
6      exec_object = execjs.compile(js_data)
7      sign = exec_object.eval('e("hello")')
8      print(sign)

```

■ 获取token

```

1  # 在js中
2  token: window.common.token
3  # 在响应中想办法获取此值
4  token_url = 'https://fanyi.baidu.com/?aldtype=16047'
5  regex: "token: '(.*?)'"

```

■ 具体代码实现

```
1 |
```

scrapy框架

■ 定义

```
1 | 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
2  1、安装依赖包
3      1、sudo apt-get install libffi-dev
4      2、sudo apt-get install libssl-dev
5      3、sudo apt-get install libxml2-dev
6      4、sudo apt-get install python3-dev
7      5、sudo apt-get install libxslt1-dev
8      6、sudo apt-get install zlib1g-dev
9      7、sudo pip3 install -I -U service_identity
10 2、安装scrapy框架
11     1、sudo pip3 install Scrapy

```

```

1  # Windows安装
2  cmd命令行(管理员): python -m pip install Scrapy

```

■ Scrapy框架五大组件

```

1  1、引擎(Engine)      : 整个框架核心
2  2、调度器(Scheduler) : 维护请求队列
3  3、下载器(Downloader): 获取响应对象
4  4、爬虫文件(Spider)  : 数据解析提取
5  5、项目管道(Pipeline): 数据入库处理
6  *****
7  # 下载器中间件(Downloader Middlewares) : 引擎->下载器,包装请求(随机代理等)
8  # 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
2  1、由引擎向爬虫程序索要第一个要爬取的URL,交给调度器去入队列
3  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
4  3、下载器得到响应对象后,通过蜘蛛中间件交给爬虫程序
5  4、爬虫程序进行数据提取:
6      1、数据交给管道文件去入库处理
7      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  # 1、创建爬虫项目
2  scrapy startproject 项目名
3  # 2、创建爬虫文件
4  scrapy genspider 爬虫名 域名
5  # 3、运行爬虫
6  scrapy crawl 爬虫名

```

■ scrapy项目目录结构

```

1 Baidu # 项目文件夹
2 |— Baidu # 项目目录
3 |   |— items.py # 定义数据结构
4 |   |— middlewares.py # 中间件
5 |   |— pipelines.py # 数据处理
6 |   |— settings.py # 全局配置
7 |   |— spiders
8 |       |— baidu.py # 爬虫文件
9 |— scrapy.cfg # 项目基本配置文件

```

■ 全局配置文件settings.py详解

```

1 # 1、定义User-Agent
2 USER_AGENT = 'Mozilla/5.0'
3 # 2、是否遵循robots协议, 一般设置为False
4 ROBOTSTXT_OBEY = False
5 # 3、最大并发量, 默认为16
6 CONCURRENT_REQUESTS = 32
7 # 4、下载延迟时间
8 DOWNLOAD_DELAY = 1
9 # 5、请求头, 此处也可以添加User-Agent
10 DEFAULT_REQUEST_HEADERS={}
11 # 6、项目管道
12 ITEM_PIPELINES={
13     '项目目录名.pipelines.类名':300
14 }

```

■ 创建爬虫项目步骤

```

1 1、新建项目 : scrapy startproject 项目名
2 2、cd 项目文件夹
3 3、新建爬虫文件 : scrapy genspider 文件名 域名
4 4、明确目标(items.py)
5 5、写爬虫程序(文件名.py)
6 6、管道文件(pipelines.py)
7 7、全局配置(settings.py)
8 8、运行爬虫 : scrapy crawl 爬虫名
9

```

■ pycharm运行爬虫项目

```

1 1、创建begin.py(和scrapy.cfg文件同目录)
2 2、begin.py中内容:
3     from scrapy import cmdline
4     cmdline.execute('scrapy crawl maoyan'.split())

```

小试牛刀

■ 目标

```
1 | 打开百度首页，把 '百度一下，你就知道' 抓取下来，从终端输出
```

■ 实现步骤

1. 创建项目Baidu 和 爬虫文件baidu

```
1 | 1、 scrapy startproject Baidu
2 | 2、 cd Baidu
3 | 3、 scrapy genspider baidu www.baidu.com
```

2. 编写爬虫文件baidu.py, xpath提取数据

```
1 | # -*- coding: utf-8 -*-
2 | import scrapy
3 |
4 | class BaiduSpider(scrapy.Spider):
5 |     name = 'baidu'
6 |     allowed_domains = ['www.baidu.com']
7 |     start_urls = ['http://www.baidu.com/']
8 |
9 |     def parse(self, response):
10 |         result = response.xpath('/html/head/title/text()').extract_first()
11 |         print('*'*50)
12 |         print(result)
13 |         print('*'*50)
14 |
```

3. 全局配置settings.py

```
1 | USER_AGENT = 'Mozilla/5.0'
2 | ROBOTSTXT_OBEY = False
3 | DEFAULT_REQUEST_HEADERS = {
4 |     'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
5 |     'Accept-Language': 'en',
6 | }
```

4. 创建begin.py (和scrapy.cfg同目录)

```
1 | from scrapy import cmdline
2 |
3 | cmdline.execute('scrapy crawl baidu'.split())
```

5. 启动爬虫

```
1 | 直接运行 begin.py 文件即可
```

思考运行过程

今日作业

1、熟记如下问题

- 1、scrapy框架有哪几大组件？
- 2、各个组件之间是如何工作的？

2、Windows安装scrapy

- 1 Windows : `python -m pip install Scrapy`
- 2 # Error: Microsoft Visual C++ 14.0 is required
- 3 # 解决 : 下载安装 Microsoft Visual C++ 14.0