Day06回顾

多线程爬虫

- **思路**

```
【1】将待爬取的URL地址存放到队列中
【2】多个线程从队列中获取地址,进行数据抓取
【3】注意获取地址过程中程序阻塞问题、线程锁问题
    3.1) 方式一
    while True:
        lock.acquire()
        if not q.empty():
            url = q.get()
            lock.release()
            ... ...
        else:
            lock.release()
            break

    3.2) 方式二->【多级页面数据抓取所需】
    while True:
        try:
            lock.acquire()
            url = q.get(block=True,timeout=3)
            lock.release()
            ... ...
        except Exception as e:
            lock.release()
            break

【4】注意：使用多线程爬取多级页面
    4.1) 创建对应队列，用来存储不同级页面的URL地址
    4.2) 除一级页面队列外，其他队列获取地址均采用如下方式
         url = q.get(block=True,timeout=2)
```

解析模块汇总

- **re、lxml+xpath、json**

```
【1】re
    import re
    pattern = re.compile(r'',re.S)
```

```
    r_list = pattern.findall(html)

【2】lxml+xpath
    from lxml import etree
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('')

【3】json
    3.1) 响应内容由json转为python : html = json.loads(res.text)
    3.2) 所抓数据保存到json文件
        with open('xxx.json','a') as f:
            json.dump(item_list,f,ensure_ascii=False)
```

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
    2.1) windows: 拷贝到Python安装目录的Scripts目录中
    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(主要是添加x权限)
    sudo chmod 777 /usr/bin/chromedriver
    sudo chmod +x /usr/bin/chromedriver
```

- **使用流程**

```
from selenium import webdriver

# 1、创建浏览器对象
browser = webdriver.Firefox(executable_path='/xxx/geckodriver')
# 2、输入网址
browser.get('URL')
# 3、查找节点
brower.find_xxxx
# 4、做对应操作
element.send_keys('')
element.click()
# 5、关闭浏览器
browser.quit()
```

- **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') - 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
```

- **定位节点的方法**

```
【1】最常用      -  browser.find_element_by_id('id属性值')
【2】最常用      -  browser.find_element_by_name('name属性值')
【3】最常用      -  browser.find_element_by_class_name('class属性值')
【4】最万能      -  browser.find_element_by_xpath('xpath表达式')
【5】文字链接    -  browser.find_element_by_link_text('链接文本')
【6】文字链接    -  browser.find_element_by_partical_link_text('部分链接文本')
【7】也还不错    -  browser.find_element_by_css_selector('css表达式')
【8】最不靠谱    -  browser.find_element_by_tag_name('标记名称')

【注意】
  1) 结果为节点对象:     find_element_
  2) 结果为节点对象列表: find_elements_
```

- **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.clear()               - 清空文本
【4】node.is_enabled()          - 判断按钮是否可用，针对于<button>按钮
【5】node.get_attribute('href')- 获取节点属性值
【6】node.text                  - 获取节点文本内容（包含子节点和后代节点）
```

Day07笔记

selenium+PhantomJS|Chrome|Firefox

- **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)
```

京东爬虫

- **目标**

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

- **思路提醒**

```
【1】打开京东，到商品搜索页
【2】匹配所有商品节点对象列表
【3】把节点对象的文本内容取出来，查看规律，是否有更好的处理办法?
【4】提取完1页后，判断如果不是最后1页，则点击下一页
    '问题：如何判断是否为最后1页？？？'
```

- **实现步骤**

```
【1】找节点
    1.1) 首页搜索框 ： //*[@id="key"]
    2.1) 首页搜索按钮    ://*[@id="search"]/div/div[2]/button
    2.3) 商品页的 商品信息节点对象列表 ://*[@id="J_goodsList"]/ul/li
    2.4) for循环遍历后
        a> 名称：.//div[@class="p-name"]/a/em
        b> 价格：.//div[@class="p-price"]
        c> 评论：.//div[@class="p-commit"]/strong
        d> 商家：.//div[@class="p-shopnum"]

【2】执行JS脚本，获取动态加载数据
    browser.execute_script(
      'window.scrollTo(0,document.body.scrollHeight)'
    )
```

- **代码实现**

```python
from selenium import webdriver
import time

class JdSpider(object):
    def __init__(self):
        self.url = 'https://www.jd.com/'
        # 设置无界面模式
        self.options = webdriver.ChromeOptions()
        self.options.add_argument('--headless')
        self.browser = webdriver.Chrome(options=self.options)

    def get_html(self):
        # get():等页面所有元素加载完成后,才会执行后面的代码
        self.browser.get(self.url)
        # 搜索框 + 搜索按钮
        self.browser.find_element_by_xpath('//*[@id="key"]').send_keys('爬虫书')
        self.browser.find_element_by_xpath('//*[@id="search"]/div/div[2]/button').click()

    # 循环体中的函数：拉进度条,提取数据
```

```
    def parse_html(self):
        # 执行js脚本,将进度条拉到最底部
        self.browser.execute_script(
            'window.scrollTo(0,document.body.scrollHeight)'
        )
        # 给页面元素加载预留时间
        time.sleep(3)
        li_list = self.browser.find_elements_by_xpath('//*[@id="J_goodsList"]/ul/li')

        for li in li_list:
            item = {}
            item['price'] = li.find_element_by_xpath('.//div[@class="p-price"]').text
            item['mame'] = li.find_element_by_xpath('.//div[@class="p-name"]/a/em').text
            item['commit'] = li.find_element_by_xpath('.//div[@class="p-commit"]/strong').text
            item['shop'] = li.find_element_by_xpath('.//div[@class="p-shopnum"]').text
            print(item)

    def run(self):
        self.get_html()
        while True:
            self.parse_html()
            if self.browser.page_source.find('pn-next disabled') == -1:
                self.browser.find_element_by_xpath('//*[@id="J_bottomPage"]/span[1]/a[9]').click()
            else:
                self.browser.quit()
                break

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

selenium - 键盘操作

```
from selenium.webdriver.common.keys import Keys

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

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])
```

- **民政部网站案例-selenium**

```python
"""
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.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_handlers[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]').text.strip()
            name = tr.find_element_by_xpath('./td[3]').text.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
```

```
6
7  # 打开豆瓣官网
8  browser = webdriver.Chrome()
9  browser.get('https://www.douban.com/')
10
11 # 切换到iframe子页面
12 login_frame = browser.find_element_by_xpath('//*[@id="anony-reg-new"]/div/div[1]/iframe')
13 browser.switch_to.frame(login_frame)
14
15 # 密码登录 + 用户名 + 密码 + 登录豆瓣
16 browser.find_element_by_xpath('/html/body/div[1]/div[1]/ul[1]/li[2]').click()
17 browser.find_element_by_xpath('//*[@id="username"]').send_keys('自己的用户名')
18 browser.find_element_by_xpath('//*[@id="password"]').send_keys('自己的密码')
19 browser.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div[5]/a').click()
20 time.sleep(3)
21
22 # 点击我的豆瓣
23 browser.find_element_by_xpath('//*[@id="db-nav-sns"]/div/div/div[3]/ul/li[2]/a').click()
```

- **示例2-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  driver.switch_to.frame("login_frame")
9
10 # 用户名+密码+登录
11 driver.find_element_by_id('u').send_keys('2621470058')
12 driver.find_element_by_id('p').send_keys('zhanshen001')
13 driver.find_element_by_id('login_button').click()
```

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

```
1  【1】设置无界面模式
2      options = webdriver.ChromeOptions()
3      options.add_argument('--headless')
4      browser = webdriver.Chrome(excutable_path='/home/tarena/chromedriver')
5
6  【2】browser执行JS脚本
7      browser.execute_script('window.scrollTo(0,document.body.scrollHeight)')
8
9  【3】键盘操作
10     from selenium.webdriver.common.keys import Keys
11
12 【4】鼠标操作
13     from selenium.webdriver import ActionChains
14     ActionChains(browser).move_to_element('node').perform()
15
16 【5】切换句柄 - switch_to.frame(handle)
17     all_handles = browser.window_handles
18     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_attribute('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)
```

- **安装scrapy出现问题**

```
xxx has requirement 模块>=4.4.2 but you'll have 模块 4.3.2
升级模块：sudo pip3 install 模块 --upgrade
```

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
```

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/
【4】熟记scrapy的五大组件,以及工作流程,能够描述的很清楚
```