DAY07

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
   while True:
      if not q.empty():
         url = q.get()
         ... ...
      else:
        break
```

- **将抓取数据保存到同一文件**

```
# 注意多线程写入的线程锁问题
from threading import Lock
lock = Lock()
lock.acquire()
python代码块
lock.release()
```

- **代码实现思路**

```
# 1、在 __init__(self) 中创建文件对象，多线程操作此对象进行文件写入
  self.f = open('xiaomi.csv','a',newline='')
  self.writer = csv.writer(self.f)
  self.lock = Lock()
# 2、每个线程抓取1页数据后将数据进行文件写入，写入文件时需要加锁
  def parse_html(self):
    app_list = []
    for xxx in xxx:
        app_list.append([name,link,typ])
    self.lock.acquire()
    self.wirter.writerows(app_list)
    self.lock.release()
# 3、所有数据抓取完成关闭文件
  def main(self):
```

```
    self.f.close()
```

解析模块汇总

re、lxml+xpath、json

```
# re
import re
pattern = re.compile(r'',re.S)
r_list = pattern.findall(html)

# lxml+xpath
from lxml import etree
parse_html = etree.HTML(html)
r_list = parse_html.xpath('')

# json
# 响应内容由json转为python
html = json.loads(res.text)
# 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)

# 或
f = open('xxx.json','a')
json.dump(item_list,f,ensure_ascii=False)
f.close()
```

Day07笔记

cookie模拟登录

适用网站及场景

```
抓取需要登录才能访问的页面
```

cookie和session机制

```
# http协议为无连接协议
cookie：存放在客户端浏览器
session：存放在Web服务器
```

人人网登录案例

- **方法一 - 登录网站手动抓取Cookie**

```
1、先登录成功1次,获取到携带登录信息的Cookie
   登录成功 - 个人主页 - F12抓包 - 刷新个人主页 - 找到主页的包(profile)
2、携带着cookie发请求
   ** Cookie
   ** User-Agent
```

```
# 1、将self.url改为 个人主页的URL地址
# 2、将Cookie的值改为 登录成功的Cookie值
import requests
from lxml import etree

class RenrenLogin(object):
  def __init__(self):
    self.url = 'xxxxxxx'
    self.headers = {
      'Cookie':'xxxxxx',
      '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'
    }

  def get_html(self):
    html = requests.get(url=self.url,headers=self.headers).text
    self.parse_html(html)

  def parse_html(self,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
    print(r_list)

if __name__ == '__main__':
  spider = RenrenLogin()
  spider.get_html()
```

- **方法二 - requests模块处理Cookie**

原理思路及实现

```
# 1. 思路
requests模块提供了session类,来实现客户端和服务端的会话保持

# 2. 原理
1、实例化session对象
   session = requests.session()
2、让session对象发送get或者post请求
   res = session.post(url=url,data=data,headers=headers)
   res = session.get(url=url,headers=headers)

# 3. 思路梳理
浏览器原理: 访问需要登录的页面会带着之前登录过的cookie
程序原理: 同样带着之前登录的cookie去访问 - 由session对象完成
1、实例化session对象
2、登录网站: session对象发送请求,登录对应网站,把cookie保存在session对象中
3、访问页面: session对象请求需要登录才能访问的页面,session能够自动携带之前的这个cookie,进行请求
```

具体步骤

```
1、寻找Form表单提交地址 - 寻找登录时POST的地址
   查看网页源码,查看form表单,找action对应的地址: http://www.renren.com/PLogin.do

2、发送用户名和密码信息到POST的地址
   * 用户名和密码信息以什么方式发送?  -- 字典
     键 : <input>标签中name的值(email,password)
     值 : 真实的用户名和密码
     post_data = {'email':'','password':''}

session = requests.session()
session.post(url=url,data=data)
```

程序实现

```
# 把Formdata中的 email 和 password 的改为自己真实的用户名和密码
import requests
from lxml import etree

class RenrenSpider(object):
  def __init__(self):
    self.post_url = 'http://www.renren.com/PLogin.do'
    self.get_url = 'http://www.renren.com/967469305/profile'
    # 实例化session对象
    self.session = requests.session()

  def get_html(self):
    # email和password为<input>节点中name的属性值
    form_data = {
      'email' : 'xxxx',
      'password' : 'xxxx'
    }
    # 先session.post()
    self.session.post(url=self.post_url,data=form_data)
    # 再session.get()
    html = self.session.get(url=self.get_url).text
    self.parse_html(html)

  def parse_html(self,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('//li[@class="school"]/span/text()')
    print(r_list)

if __name__ == '__main__':
  spider = RenrenSpider()
  spider.get_html()
```

- **方法三**

原理

```
1、把抓取到的cookie处理为字典
2、使用requests.get()中的参数:cookies
```

处理cookie为字典

```
# 处理cookies为字典
cookies_dict = {}
cookies = 'xxxx'
for kv in cookies.split('; ')
  cookies_dict[kv.split('=')[0]] = kv.split('=')[1]
```

代码实现

```
import requests
from lxml import etree

class RenrenLogin(object):
  def __init__(self):
    self.url = 'http://www.renren.com/967469305/profile'
    self.headers = {
      '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'
    }

  # 获取字典形式cookie的函数
  def get_cookie_dict(self):
    cookie_dict = {}
    cookies = 'anonymid=jy87mc5fx4xvjj; _r01_=1; jebe_key=a04238bc-adc0-4418-a770-519d74219f15%7C2e9beece3ead42fe6a26739d515f14df%7C1563911475551%7C1%7C1563911475689; ln_uact=13603263409; depovince=GW; jebecookies=3720f008-2502-4422-acfe-8b78b4c3611d|||||; JSESSIONID=abcUijruA6U375Qz-tHZw; ick_login=60cb66a4-e407-4fd5-a2ee-1eae3220a102; _de=4DBCFCC17D9E50C8C92BCDC45CC5C3B7; p=415429a0f0b3067e9061fd8387c269c45; first_login_flag=1; ln_hurl=http://hdn.xnimg.cn/photos/hdn421/20190815/1435/main_91u0_81d40000ca0d1986.jpg; t=da27ae8094836b90d439e88e21ed73ac5; societyguester=da27ae8094836b90d439e88e21ed73ac5; id=967469305; xnsid=1eafd54d; ver=7.0; loginfrom=null; jebe_key=a04238bc-adc0-4418-a770-519d74219f15%7C2012cb2155debcd0710a4bf5a73220e8%7C1567148226573%7C1%7C1567148227902; wp_fold=0'
    for kv in cookies.split('; '):
      # kv: 'td_cookie=184xxx'
      key = kv.split('=')[0]
      value = kv.split('=')[1]
      cookie_dict[key] = value

    return cookie_dict

  def get_html(self):
    # 获取cookies
    cookies = self.get_cookie_dict()
    html = requests.get(
      url=self.url,
      headers=self.headers,
      cookies=cookies,
    ).text
    self.parse_html(html)
```

```
32
33    def parse_html(self,html):
34        parse_html = etree.HTML(html)
35        r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
36        print(r_list)
37
38  if __name__ == '__main__':
39      spider = RenrenLogin()
40      spider.get_html()
```

json解析模块

json.loads(json)

- **作用**

```
1  把json格式的字符串转为Python数据类型
```

- **示例**

```
1  html_json = json.loads(res.text)
```

json.dumps(python)

- **作用**

```
1  把 python 类型 转为 json 类型
```

- **示例**

```
1  import json
2
3  # json.dumps()之前
4  item = {'name':'QQ','app_id':1}
5  print('before dumps',type(item)) # dict
6  # json.dumps之后
7  item = json.dumps(item)
8  print('after dumps',type(item)) # str
```

json.load(f)

作用

```
1  将json文件读取,并转为python类型
```

示例

```python
import json

with open('D:\\spider_test\\xiaomi.json','r') as f:
    data = json.load(f)

print(data)
```

json.dump(python,f,ensure_ascii=False)

- **作用**

```
把python数据类型 转为 json格式的字符串
# 一般让你把抓取的数据保存为json文件时使用
```

- **参数说明**

```
第1个参数：python类型的数据(字典，列表等)
第2个参数：文件对象
第3个参数：ensure_ascii=False # 序列化时编码
```

- **示例1**

```python
import json

item = {'name':'QQ','app_id':1}
with open('小米.json','a') as f:
  json.dump(item,f,ensure_ascii=False)
```

- **示例2**

```python
import json

item_list = []
for i in range(3):
  item = {'name':'QQ','id':i}
  item_list.append(item)

with open('xiaomi.json','a') as f:
  json.dump(item_list,f,ensure_ascii=False)
```

json模块总结

```
# 爬虫最常用
1、数据抓取 - json.loads(html)
   将响应内容由: json 转为 python
2、数据保存 - json.dump(item_list,f,ensure_ascii=False)
   将抓取的数据保存到本地 json文件

# 抓取数据一般处理方式
1、txt文件
2、csv文件
3、json文件
4、MySQL数据库
5、MongoDB数据库
6、Redis数据库
```

selenium+phantomjs/Chrome/Firefox

selenium

- **定义**

```
1、Web自动化测试工具，可运行在浏览器,根据指令操作浏览器
2、只是工具，必须与第三方浏览器结合使用
```

- **安装**

```
Linux: sudo pip3 install selenium
Windows: python -m pip install selenium
```

phantomjs浏览器

- **定义**

```
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
```

- **安装(phantomjs、chromedriver、geckodriver)**

Windows

```
1、下载对应版本的phantomjs、chromedriver、geckodriver
2、把chromedriver.exe拷贝到python安装目录的Scripts目录下(添加到系统环境变量)
   # 查看python安装路径：where python
3、验证
   cmd命令行：chromedriver

# 下载地址
1、chromedriver ： 下载对应版本
http://chromedriver.storage.googleapis.com/index.html
2、geckodriver
https://github.com/mozilla/geckodriver/releases
3、phantomjs
https://phantomjs.org/download.html
```

Linux

```
1、下载后解压
   tar -zxvf geckodriver.tar.gz
2、拷贝解压后文件到 /usr/bin/  (添加环境变量)
   sudo cp geckodriver /usr/bin/
3、更改权限
   sudo -i
   cd /usr/bin/
   chmod 777 geckodriver
```

- **使用**

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.PhantomJS()
browser.get('http://www.baidu.com/')

time.sleep(5)

# 关闭浏览器
browser.quit()
```

示例代码二：打开百度，搜索赵丽颖，点击搜索，查看

```python
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**

```python
# from selenium import webdriver
1、browser = webdriver.Chrome(executable_path='path')
2、browser.get(url)
3、browser.page_source # HTML结构源码
4、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
5、browser.quit() # 关闭浏览器
```

- **定位节点**

单元素查找(1个节点对象)

```python
1、browser.find_element_by_id('')
2、browser.find_element_by_name('')
3、browser.find_element_by_class_name('')
4、browser.find_element_by_xpath('')
... ...
```

多元素查找([节点对象列表])

```python
1、browser.find_elements_by_id('')
2、browser.find_elements_by_name('')
3、browser.find_elements_by_class_name('')
4、browser.find_elements_by_xpath('')
... ...
```

- 节点对象操作

```python
1、ele.send_keys('') # 搜索框发送内容
2、ele.click()
3、ele.text # 获取文本内容，包含子节点和后代节点的文本内容
4、ele.get_attribute('src') # 获取属性值
```

京东爬虫案例

- **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
   # 如何判断是否为最后1页？？？
```

- **实现步骤**

找节点

```
1、首页搜索框 ： //*[@id="key"]
2、首页搜索按钮   ://*[@id="search"]/div/div[2]/button
3、商品页的 商品信息节点对象列表 ://*[@id="J_goodsList"]/ul/li
4、for循环遍历后
  名称: .//div[@class="p-name"]/a/em
  价格: .//div[@class="p-price"]
  评论: .//div[@class="p-commit"]/strong
  商家: .//div[@class="p-shopnum"]
```

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

```
from selenium import webdriver
import time

class JdSpider(object):
  def __init__(self):
    self.url = 'https://www.jd.com/'
    # 设置无界面
    self.options = webdriver.ChromeOptions()
    self.options.add_argument('--headless')
    # 正常创建浏览器对象即可
    self.browser = webdriver.Chrome(options=self.options)
    # 计数
    self.i = 0

  # 获取页面信息 - 到具体商品的页面
  def get_html(self):
    self.browser.get(self.url)
    # 找两个节点
    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()
    # 给商品页面加载时间
    time.sleep(3)

  def parse_html(self):
    # 把进度条拉到底部,使所有数据动态加载
    self.browser.execute_script(
      'window.scrollTo(0,document.body.scrollHeight)'
    )
    # 等待动态数据加载完成
    time.sleep(2)

    # 提取所有商品节点对象列表 li列表
    li_list = self.browser.find_elements_by_xpath('//*[@id="J_goodsList"]/ul/li')
    item = {}
    for li in li_list:
      # find_element: 查找单元素
      item['name'] = li.find_element_by_xpath('.//div[@class="p-name"]/a/em').text.strip()
      item['price'] = li.find_element_by_xpath('.//div[@class="p-price"]').text.strip()
      item['comment'] = li.find_element_by_xpath('.//div[@class="p-
commit"]/strong').text.strip()
      item['shop'] = li.find_element_by_xpath('.//div[@class="p-shopnum"]').text.strip()

      print(item)
      self.i += 1


  def main(self):
    self.get_html()
    while True:
      self.parse_html()
      # 判断是否为最后一页
      if self.browser.page_source.find('pn-next disabled') == -1:
        self.browser.find_element_by_class_name('pn-next').click()
        time.sleep(3)
      else:
        break
    print('商品数量:',self.i)
    self.browser.quit()


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

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')
browser = webdriver.Chrome(options=options)
browser.get('http://www.baidu.com/')
browser.save_screenshot('baidu.png')
```

selenium - 键盘操作

```python
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**

```
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**

```
# 获取当前所有句柄（窗口）
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

目标

```
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

```
# 建库
create database govdb charset utf8;
use govdb;
# 建表
create table province(
p_name varchar(20),
p_code varchar(20)
)charset=utf8;
create table city(
c_name varchar(20),
c_code varchar(20),
c_father_code varchar(20)
)charset=utf8;
create table county(
x_name varchar(20),
x_code varchar(20),
x_father_code varchar(20)
)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

```
from selenium import webdriver
```

```python
import pymysql

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/'
    self.db = pymysql.connect(
      'localhost','root','123456','govdb',charset='utf8'
    )
    self.cursor = self.db.cursor()
    # 创建3个列表,用来executemany()往3张表中插入记录
    self.province_list = []
    self.city_list = []
    self.county_list = []

  def get_incr_url(self):
    self.browser.get(self.one_url)
    # 提取最新链接,判断是否需要增量爬
    td = self.browser.find_element_by_xpath(
      '//td[@class="arlisttd"]/a[contains(@title,"代码")]'
    )
    # 提取链接 和 数据库中做比对,确定是否需要怎俩那个抓取
    # get_attribute()会自动补全提取的链接
    two_url = td.get_attribute('href')
    sel = 'select url from version where url=%s'
    # result为返回的受影响的条数
    result = self.cursor.execute(sel,[two_url])
    if result:
      print('无须爬取')
    else:
      td.click()
      # 切换句柄
      all_handlers = self.browser.window_handles
      self.browser.switch_to.window(all_handlers[1])
      self.get_data()
      # 把URL地址存入version表
      dele = 'delete from version'
      ins = 'insert into version values(%s)'
      self.cursor.execute(dele)
      self.cursor.execute(ins,[two_url])
      self.db.commit()

  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)
      # 把数据添加到对应的表中
      if code[-4:] == '0000':
        self.province_list.append([name,code])
```

```
        if name in ['北京市','天津市','上海市','重庆市']:
          self.city_list.append([name,code,code])

      elif code[-2:] == '00':
        self.city_list.append([name,code,(code[:2]+'0000')])

      else:
        if code[:2] in ['11','12','31','50']:
          self.county_list.append([name,code,(code[:2]+'0000')])
        else:
          self.county_list.append([name,code,(code[:4]+'00')])

    # 执行数据库插入语句
    self.insert_mysql()

  def insert_mysql(self):
    # 1. 先删除原有数据
    del_province = 'delete from province'
    del_city = 'delete from city'
    del_county = 'delete from county'
    self.cursor.execute(del_province)
    self.cursor.execute(del_city)
    self.cursor.execute(del_county)
    # 2. 插入新数据
    ins_province = 'insert into province values(%s,%s)'
    ins_city = 'insert into city values(%s,%s,%s)'
    ins_county = 'insert into county values(%s,%s,%s)'
    self.cursor.executemany(ins_province,self.province_list)
    self.cursor.executemany(ins_city,self.city_list)
    self.cursor.executemany(ins_county,self.county_list)
    # 3.提交到数据库执行
    self.db.commit()

  def main(self):
    self.get_incr_url()
    self.cursor.close()
    self.db.close()
    self.browser.quit()

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

SQL命令练习

```
# 1. 查询所有省市县信息（多表查询实现）
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;
# 2. 查询所有省市县信息（连接查询实现）
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?

```
不用输入，在URL地址中填入就可以
```

示例: 爬取某一天笔记

```
from selenium import webdriver

url = 'http://tarenacode:code_2013@code.tarena.com.cn/AIDCode/aid1904/15-spider/spider_day06_note.zip'
browser = webdriver.Chrome()
browser.get(url)
```

selenium - iframe子框架

特点

```
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

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

示例 - 登录qq邮箱

```
from selenium import webdriver
import time

driver = webdriver.Chrome()
driver.get('https://mail.qq.com/')

# 切换到iframe子框架
login_frame = driver.find_element_by_id('login_frame')
driver.switch_to.frame(login_frame)

# 用户名+密码+登录
driver.find_element_by_id('u').send_keys('qq号')
driver.find_element_by_id('p').send_keys('qq密码')
driver.find_element_by_id('login_button').click()

# 预留页面记载时间
time.sleep(5)

# 提取数据
ele = driver.find_element_by_id('useralias')
print(ele.text)
```

百度翻译破解案例

目标

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

2、抓取相关JS文件

```
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
   # 1. a 为要翻译的单词
   # 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)

```
function a(r) {
        if (Array.isArray(r)) {
            for (var o = 0, t = Array(r.length); o < r.length; o++)
                t[o] = r[o];
            return t
        }
        return Array.from(r)
    }
function n(r, o) {
    for (var t = 0; t < o.length - 2; t += 3) {
        var a = o.charAt(t + 2);
        a = a >= "a" ? a.charCodeAt(0) - 87 : Number(a),
            a = "+" === o.charAt(t + 1) ? r >>> a : r << a,
            r = "+" === o.charAt(t) ? r + a & 4294967295 : r ^ a
    }
    return r
}
function e(r) {
    var o = r.match(/[\uD800-\uDBFF][\uDC00-\uDFFF]/g);
    if (null === o) {
        var t = r.length;
```

```
        t > 30 && (r = "" + r.substr(0, 10) + r.substr(Math.floor(t / 2) - 5, 10) + r.substr(-10, 10))
    } else {
        for (var e = r.split(/[\uD800-\uDBFF][\uDC00-\uDFFF]/), C = 0, h = e.length, f = []; h > C; C++)
            "" !== e[C] && f.push.apply(f, a(e[C].split(""))),
                C !== h - 1 && f.push(o[C]);
        var g = f.length;
        g > 30 && (r = f.slice(0, 10).join("") + f.slice(Math.floor(g / 2) - 5, Math.floor(g / 2) + 5).join("") + f.slice(-10).join(""))
    }
//    var u = void 0
//    , l = "" + String.fromCharCode(103) + String.fromCharCode(116) + String.fromCharCode(107);
//    u = null !== i ? i : (i = window[l] || "") || "";
//  断点调试,然后从网页源码中找到 window.gtk的值
    var u = '320305.131321201'

    for (var d = u.split("."), m = Number(d[0]) || 0, s = Number(d[1]) || 0, S = [], c = 0, v = 0; v < r.length; v++) {
        var A = r.charCodeAt(v);
        128 > A ? S[c++] = A : (2048 > A ? S[c++] = A >> 6 | 192 : (55296 === (64512 & A) && v + 1 < r.length && 56320 === (64512 & r.charCodeAt(v + 1)) ? (A = 65536 + ((1023 & A) << 10) + (1023 & r.charCodeAt(++v)),
            S[c++] = A >> 18 | 240,
            S[c++] = A >> 12 & 63 | 128) : S[c++] = A >> 12 | 224,
                                                                    S[c++] = A >> 6 & 63 | 128),
                                S[c++] = 63 & A | 128)
    }
    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++)
        p += S[b],
            p = n(p, F);
    return p = n(p, D),
        p ^= s,
        0 > p && (p = (2147483647 & p) + 2147483648),
        p %= 1e6,
        p.toString() + "." + (p ^ m)
}
```

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

```
# 安装pyexecjs模块
sudo pip3 install pyexecjs

# 使用
import execjs

with open('translate.js','r') as f:
    js_data = f.read()

# 创建对象
exec_object = execjs.compile(js_data)
sign = exec_object.eval('e("hello")')
print(sign)
```

获取token

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

具体代码实现

```
import requests
import re
import execjs

class BaiduTranslateSpider(object):
    def __init__(self):
        self.token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        self.post_url = 'https://fanyi.baidu.com/v2transapi'
        self.headers = {
            'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3',
            'accept-language': 'zh-CN,zh;q=0.9',
            'cache-control': 'no-cache',
```

```
            'cookie': 'BAIDUID=52920E829C1F64EE98183B703F4E37A9:FG=1;
BIDUPSID=52920E829C1F64EE98183B703F4E37A9; PSTM=1562657403;
to_lang_often=%5B%7B%22value%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%2C%7B%22value%
22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%5D; REALTIME_TRANS_SWITCH=1;
FANYI_WORD_SWITCH=1; HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1; delPer=0;
BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; BCLID=6890774803653935935;
BDSFRCVID=4XAsJeCCxG3DLCbwbJrKDGwjNA0UN_I3KhXZ3J;
H_BDCLCKID_SF=tRk8oIDaJCvSe6r1MtQ_M4F_qxby26nUQ5neaJ5n0-
nnhnL4W46bqJKFLtozKMoI3C7fotJJ5nololIRy6CKjjb-jaDqJ5n3bTnjstcS2RREHJrg-
trSMDCShGRGWlO9WDTm_D_KfxnkOnc6qJj0-jjXqqo8K5Ljaa5n-
pPKKRAaqD04bPbZL4DdMa7HLtAO3mkjbnczfn02OP5P5lJ_e-4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoDPMJKIbMI_rMbbfhKC3hqJfaI62aKDs_RCMBhcqEIL4eJOIb6_w5gcq0T_HttjtXR
0atn7ZSMbSj4Qo5pK95p38bxnDK2rQLb5zah5nhMJS3j7JDMP0-4rJhxby523i5J6vQpnJ8hQ3DRoWXPIqbN7P-
p5Z5mAqKl0MLIOkbC_6j5DWDTvLeU7J-n8XbI60XRj85-
ohHJrFMtQ_q4tehHRMBUo9WDTm_DoTttt5fUj6qJj855jXqqo8KMtHJaFf-pPKKRAashnzWjrkqqOQ5pj-
WnQr3mkjbn5yfn02OpjPX6joht4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoC0XtIDhMDvPMCTSMt_HMxrKetJyaR0JhpjbWJ5TEPnjDUOdLPDW-
46HBM3xbKQw5CJGBf7zhpvdWhC5y6ISKx-_J68Dtf5; ZD_ENTRY=baidu; PSINO=2;
H_PS_PSSID=26525_1444_21095_29578_29521_28518_29098_29568_28830_29221_26350_29459; locale=zh;
Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1563426293,1563996067;
from_lang_often=%5B%7B%22value%22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%2C%7B%22valu
e%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%5D;
Hm_lpvt_64ecd82404c51e03dc91cb9e8c025574=1563999768;
yjs_js_security_passport=2706b5b03983b8fa12fe756b8e4a08b98fb43022_1563999769_js',
            'pragma': 'no-cache',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/75.0.3770.142 Safari/537.36',
        }

    # 获取token和gtk
    def get_token(self):
        token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        # 定义请求头
        r = requests.get(self.token_url,headers=self.headers)
        token = re.findall(r"token: '(.*?)'",r.text)
        window_gtk = re.findall(r"window.*?gtk = '(.*?)';</script>",r.text)
        if token:
            return token[0],window_gtk[0]

    # 获取sign
    def get_sign(self,word,gtk):
        with open('translate.js','r') as f:
            js_data = f.read()

        exec_object = execjs.compile(js_data)
        sign = exec_object.eval('e("{}","{}")'.format(word,gtk))

        return sign

    # 主函数
    def main(self,word,fro,to):
        token,gtk = self.get_token()
        sign = self.get_sign(word,gtk)
        # 找到form表单数据如下,sign和token需要想办法获取
        form_data = {
            'from': fro,
```

```
            'to': to,
            'query': word,
            'transtype': 'realtime',
            'simple_means_flag': '3',
            'sign': sign,
            'token': token
        }
        r = requests.post(self.post_url,data=form_data,headers=self.headers)
        print(r.json()['trans_result']['data'][0]['dst'])

if __name__ == '__main__':
    spider = BaiduTranslateSpider()
    choice = input('1. 英译汉 2. 汉译英 : ')
    word = input('请输入要翻译的单词:')
    if choice == '1':
        fro,to = 'en','zh'
    elif choice == '2':
        fro,to = 'zh','en'

    spider.main(word,fro,to)
```