# Day04回顾

## requests.get()参数

```
1、url
2、params -> {} : 查询参数 Query String
3、proxies -> {}
   proxies = {
      'http':'http://1.1.1.1:8888',
      'https':'https://1.1.1.1:8888'
   }
4、auth -> ('tarenacode','code_2013')
5、verify -> True/False
6、timeout
```

## requests.post()

```
data -> {} Form表单数据 : Form Data
```

## 控制台抓包

- 打开方式及常用选项

```
1、打开浏览器，F12打开控制台，找到Network选项卡
2、控制台常用选项
   1、Network：抓取网络数据包
        1、ALL：抓取所有的网络数据包
        2、XHR：抓取异步加载的网络数据包
        3、JS ：抓取所有的JS文件
   2、Sources：格式化输出并打断点调试JavaScript代码，助于分析爬虫中一些参数
   3、Console：交互模式，可对JavaScript中的代码进行测试
3、抓取具体网络数据包后
   1、单击左侧网络数据包地址，进入数据包详情，查看右侧
   2、右侧：
      1、Headers：整个请求信息
           General、Response Headers、Request Headers、Query String、Form Data
      2、Preview：对响应内容进行预览
      3、Response：响应内容
```

- 有道翻译过程梳理

1.
```
1. 打开首页
2. 准备抓包：F12开启控制台
3. 寻找地址
   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
4. 发现规律
   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，分析对比 Network - All(或者XHR) - Form Data，发现对应的规律
5. 寻找JS文件
   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
   搜索关键字，找到相关加密方法
7、断点调试
8、完善程序
```

# 常见的反爬机制及处理方式

```
1、Headers反爬虫 ：Cookie、Referer、User-Agent
   解决方案：通过F12获取headers,传给requests.get()方法

2、IP限制 ：网站根据IP地址访问频率进行反爬,短时间内进制IP访问
   解决方案：
        1、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
        2、购买开放代理或私密代理IP
        3、降低爬取的速度

3、User-Agent限制 ：类似于IP限制
   解决方案：构造自己的User-Agent池,每次访问随机选择

4、Ajax动态加载 ：从url加载网页的源代码后,会在浏览器执行JavaScript程序,这些程序会加载更多内容
   解决方案：F12或抓包工具抓包处理

5、对查询参数加密
   解决方案：找到JS文件,分析加密算法,用Python实现加密执行JS文件中的代码,返回加密数据

6、对响应内容做处理
   解决方案：打印并查看响应内容,用xpath或正则做处理
```

# python中正则处理headers和formdata

```
1、pycharm进入方法 ：Ctrl + r ，选中 Regex
2、处理headers和formdata
  (.*): (.*)
  "$1": "$2",
3、点击 Replace All
```

# Day05笔记

## 动态加载数据抓取-Ajax

- 特点

```
1、右键 -> 查看网页源码中没有具体数据
2、滚动鼠标滑轮或其他动作时加载
```

- 抓取

```
1、F12打开控制台，页面动作抓取网络数据包
2、抓取json文件URL地址
# 控制台中 XHR ： 异步加载的数据包
# XHR -> QueryStringParameters(查询参数)
```

**豆瓣电影数据抓取案例**

- 目标

```
1、地址：豆瓣电影 - 排行榜 - 剧情
2、目标：电影名称、电影评分
```

- F12抓包（XHR）

```
1、Request URL(基准URL地址) ： https://movie.douban.com/j/chart/top_list?
2、Query String(查询参数)
# 抓取的查询参数如下：
type: 13
interval_id: 100:90
action: ''
start: 0
limit: 用户输入的电影数量
```

- json模块的使用

```
1、json.loads(json格式的字符串)：把json格式的字符串转为python数据类型
# 示例
html = json.loads(res.text)
print(type(html))
```

- 代码实现

```

```

思考: 实现用户在终端输入电影类型和电影数量，将对应电影信息抓取到数据库

```
1
```

练习: 腾讯招聘案例抓包看看?

- URL地址及目标

1. 确定URL地址及目标

```
1、URL：百度搜索腾讯招聘 - 查看工作岗位
2、目标：职位名称、工作职责、岗位要求
```

2. F12抓包
3. 一级页面json地址(index变,timestamp未检查)

```
https://careers.tencent.com/tencentcareer/api/post/Query?timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attrId=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn
```

4. 二级页面地址(postId在变,在一级页面中可拿到)

```
https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1563912374645&postId={}&language=zh-cn
```

- 具体代码实现

```
1
```

# 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
```

- **方法一**

```
1、先登录成功1次,获取到携带登陆信息的Cookie
   F12打开控制台,在页面输入用户名、密码,登录成功,找到/home(一般在抓到地址的上面)
2、携带着cookie发请求
   ** Cookie
   ** Referer(源,代表你从哪里转过来的)
   ** User-Agent
```

```
1
```

- **方法二**

1. 知识点

```
利用requests模块中的session会话保持功能
```

2. session会话使用流程

```
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.get(url,headers=headers)
```

3. 具体步骤

```
1、寻找登录时POST的地址
   查看网页源码,查看form,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':''}
```

4. 程序实现

```
整体思路
1、先POST: 把用户名和密码信息POST到某个地址中
2、再GET:  正常请求去获取页面信息
```

```

```

# requests.post()

- 适用场景a

```
Post类型请求的网站
```

- 参数-data

```
response = requests.post(url,data=data,headers=headers)
# data : post数据 (Form表单数据-字典格式)
```

- 请求方式的特点

```
# 一般
GET请求 : 参数在URL地址中有显示
POST请求: Form表单提交数据
```

**有道翻译破解案例(post)**

1. 目标

```
破解有道翻译接口，抓取翻译结果
# 结果展示
请输入要翻译的词语：elephant
翻译结果：大象
*************************
请输入要翻译的词语：喵喵叫
翻译结果：mews
```

2. 实现步骤

```
1、浏览器F12开启网络抓包,Network-All,页面翻译单词后找Form表单数据
2、在页面中多翻译几个单词，观察Form表单数据变化（有数据是加密字符串）
3、刷新有道翻译页面，抓取并分析JS代码（本地JS加密）
4、找到JS加密算法，用Python按同样方式加密生成加密数据
5、将Form表单数据处理为字典，通过requests.post()的data参数发送
```

**具体实现**

- 1、开启F12抓包，找到Form表单数据如下:

```
i: 喵喵叫
from: AUTO
to: AUTO
smartresult: dict
client: fanyideskweb
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
doctype: json
version: 2.1
keyfrom: fanyi.web
action: FY_BY_REALTlME
```

- 2、在页面中多翻译几个单词，观察Form表单数据变化

```
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
# 但是bv的值不变
```

- 3、一般为本地js文件加密，刷新页面，找到js文件并分析JS代码

```
# 方法1
Network - JS选项 - 搜索关键词salt
# 方法2
控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出

# 最终找到相关JS文件 : fanyi.min.js
```

- 4、打开JS文件，分析加密算法，用Python实现

```
# ts : 经过分析为13位的时间戳，字符串类型
js代码实现:  "" + (new Date).getTime()
python实现:

# salt
js代码实现:  r+parseInt(10 * Math.random(), 10);
python实现:

# sign (设置断点调试，来查看 e 的值，发现 e 为要翻译的单词)
js代码实现: n.md5("fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj")
python实现:
```

- 5、代码实现

```

```

# 代理IP使用

## *购买开放代理使用*

- 开放代理接口

```
# getip.py
# 获取开放代理的接口

```

**测试开放代理接口: http://httpbin.org/get**

```
1、从代理网站上获取购买的普通代理的api链接
2、从api链接中提取出IP
3、随机选择代理IP访问网站进行数据抓取
```

```

```

## *购买私密代理使用*

**语法格式**

```
1、语法结构
proxies = {
    '协议':'协议://用户名:密码@IP:端口号'
}

2、示例
proxies = {
    'http':'http://用户名:密码@IP:端口号',
    'https':'https://用户名:密码@IP:端口号'
}
```

**示例代码**

```python
import requests
url = 'http://httpbin.org/get'
proxies = {
    'http': 'http://309435365:szayclhp@122.114.67.136:16819',
    'https':'https://309435365:szayclhp@122.114.67.136:16819',
}
headers = {
    'User-Agent' : 'Mozilla/5.0',
}

html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
print(html)
```

# 今日作业

作业

```
1、仔细复习并总结有道翻译案例，抓包流程，代码实现
2、通过百度翻译，来再次熟练抓包流程，分析，断点调试等操作
```