# Day04回顾

## requests.get()参数

```
1、url
2、params -> {} : 查询参数 Query String
3、proxies -> {}
   proxies = {
      'http':'http://username:password@1.1.1.1:8888',
      'https':'https://username:password@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

5、对查询参数或Form表单数据认证(salt、sign)
   解决方案：找到JS文件,分析JS处理方法,用Python按同样方式处理

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

## 有道翻译代码实现

**有道翻译验证了什么？ - headers**

```
1、Cookie
2、Referer
3、User-Agent
```

**代码实现**

```

```

## 民政部网站数据抓取

**目标**

```
1、URL：http://www.mca.gov.cn/ - 民政数据 - 行政区划代码
   即：http://www.mca.gov.cn/article/sj/xzqh/2019/
2、目标：抓取最新中华人民共和国县以上行政区划代码
```

**实现步骤**

- **1、从民政数据网站中提取最新行政区划代码链接**

```
# 特点
1、最新的在上面
2、命名格式：2019年X月中华人民共和国县以上行政区划代码
```

- **2、从二级页面链接中提取真实链接（反爬-响应内容中嵌入JS，指向新的链接）**

```
1、向二级页面链接发请求得到响应内容，并查看嵌入的JS代码
2、正则提取真实的二级页面链接
```

- **3、在数据库表中查询此条链接是否已经爬取，建立增量爬虫**

```
1、数据库中建立version表，存储爬取的链接
2、每次执行程序和version表中记录核对，查看是否已经爬取过
```

- **4、代码实现**

```

```

# 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 # 电影类型
interval_id: 100:90
action: ''
start: 0  # 每次加载电影的起始索引值
limit: 20 # 每次加载的电影数量
```

- **代码实现**

```

```

**练习: 能否抓取指定类型的所有电影信息？ - 无须指定数量**

```

```

# 多线程爬虫

## *应用场景*

```
1、多进程 ： CPU密集程序
2、多线程 ： 爬虫(网络I/O)、本地磁盘I/O
```

## 知识点回顾

- 队列

```
# 导入模块
from queue import Queue
# 使用
q = Queue()
q.put(url)
q.get() # 当队列为空时，阻塞
q.empty() # 判断队列是否为空，True/False
```

- 线程模块

```
# 导入模块
from threading import Thread

# 使用流程
t = Thread(target=函数名) # 创建线程对象
t.start() # 创建并启动线程
t.join()  # 阻塞等待回收线程

# 如何创建多线程，如下方法你觉得怎么样？？？？？
for i in range(5):
    t = Thread(target=函数名)
    t.start()
    t.join()
```

## 小米应用商店抓取(多线程)

- 目标

```
1、网址 ： 百度搜 - 小米应用商店，进入官网
2、目标 ： 应用分类 - 聊天社交
   应用名称
   应用链接
```

- 实现步骤

1. 确认是否为动态加载

```
1、页面局部刷新
2、右键查看网页源代码，搜索关键字未搜到
# 此网站为动态加载网站，需要抓取网络数据包分析
```

2. F12抓取网络数据包

```
1、抓取返回json数据的URL地址（Headers中的Request URL）
   http://app.mi.com/categotyAllListApi?page={}&categoryId=2&pageSize=30

2、查看并分析查询参数（headers中的Query String Parameters）
   page: 1
   categoryId: 2
   pageSize: 30
   # 只有page再变，0 1 2 3 ... ... ，这样我们就可以通过控制page的直拼接多个返回json数据的URL地址
```

- 代码实现

```

```

# 今日作业

```
1、有道翻译案例复写一遍
2、抓取腾讯招聘数据(两级页面 - 职位名称、岗位职责、工作要求)
3、把腾讯招聘案例改写为多线程
4、把链家二手房案例改写为多线程
5、民政部数据抓取案例完善
   # 1、将抓取的数据存入数据库，最好分表按照层级关系去存
   # 2、增量爬取时表中数据也要更新
```