

- 简历模板下载拓展

```
import requests
from lxml import etree
import os
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/98.0.4758.80
Safari/537.36',

}
#创建一个新的文件夹
dirName = 'jianli'
if not os.path.exists(dirName):
    os.mkdir(dirName)

#通用的url模板
url =
'https://sc.chinaz.com/jianli/free_%d.html'
for page in range(1,11):
    if page == 1:
        new_url =
'https://sc.chinaz.com/jianli/free.html'
    else:
        new_url = format(url%page)
```

```
response =
requests.get(url=new_url,headers=headers)
response.encoding = 'utf-8'
page_text = response.text
#数据解析：详情页url和简历名称
tree = etree.HTML(page_text)
div_list = tree.xpath('//*[@id="container"]/div')
#局部解析
for div in div_list:
    detail_url =
'https:'+div.xpath('./a/@href')[0]
    title = div.xpath('./p/a/text()')[0]+' .rar'
    # print(title,detail_url)
    #对详情页的url发起请求，解析出简历的下载地址
    detail_page_text =
requests.get(url=detail_url,headers=headers).
text

#数据解析：解析下载地址
detail_tree =
etree.HTML(detail_page_text)
li_list = detail_tree.xpath('//*[@id="down"]/div[2]/ul/li')
down_list = [] #存储不同的12个下载地址
for li in li_list:
```

```
        download_link =
li.xpath( './a/@href' )[0]
        down_list.append(download_link)
#随机选择一个下载地址进行简历模板的下载
import random
#从列表中随机选出一个下载地址
link = random.choice(down_list)
#对下载地址进行请求发送，下载简历模板的压缩包
data =
requests.get(url=link,headers=headers).content

        filePath = dirName+'/' +title
        with open(filePath,'wb') as fp:
            fp.write(data)
        print(title, '下载保存成功! ')
```

Cookie

- 什么是cookie?
 - cookie的本质就是一组数据（键值对的形式存在）
 - 是由服务器创建，返回给客户端，最终会保存在客户端浏览器中。
 - 如果客户端保存了cookie，则下次再次访问该服务器，就会携带cookie进行网络访问。
 - 典型的案例：网站的免密登录

- 爬取雪球网中的咨询数据

- url: <https://xueqiu.com/>, 需求就是爬取热帖内容
- 经过分析发现帖子的内容是通过ajax动态加载出来的, 因此通过抓包工具, 定位到ajax请求的数据包, 从数据包中提取:
 - url: https://xueqiu.com/statuses/hot/listV2.json?since_id=-1&max_id=311519&size=15
 - 请求方式: get
 - 请求参数: 拼接在了url后面

```
import requests
import os
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/98.0.4758.80
Safari/537.36',
}
url =
'https://xueqiu.com/statuses/hot/listV2.jso
n'
param = {
    "since_id": "-1",
    "max_id": "311519",
    "size": "15",
```

```
}  
response =  
requests.get(url=url,headers=headers,params  
=param)  
data = response.json()  
print(data)  
#发现没有拿到我们想要的数据
```

○ 分析why?

- 切记：只要爬虫拿不到你想要的数据，唯一的原因是爬虫程序模拟浏览器的力度不够！一般来讲，模拟的力度重点放置在请求头中！
- 上述案例，只需要在请求头headers中添加cookie即可！

○ 爬虫中cookie的处理方式（两种方式）：

- 手动处理：将抓包工具中的cookie赋值到headers中即可
 - 缺点：
 - 编写麻烦
 - cookie通常都会存在有效时长
 - cookie中可能会存在实时变化的局部数据
- 自动处理
 - 基于session对象实现自动处理cookie。
 - 1.创建一个空白的session对象。

- 2.需要使用session对象发起请求，请求的目的是为了捕获cookie
 - 注意：如果session对象在发请求的过程中，服务器端产生了cookie，则cookie会自动存储在session对象中。
- 3.使用携带cookie的session对象，对目的网址发起请求，就可以实现携带cookie的请求发送，从而获取想要的数据。
- 注意： session对象至少需要发起两次请求
 - 第一次请求的目的是为了捕获存储cookie到session对象
 - 后次的请求，就是携带cookie发起的请求了

```
import requests
#1.创建一个空白的session对象
session = requests.Session()

headers = {
    'User-Agent': 'Mozilla/5.0
(Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/98.0.4758.80 Safari/537.36',
}
main_url = 'https://xueqiu.com/'
```

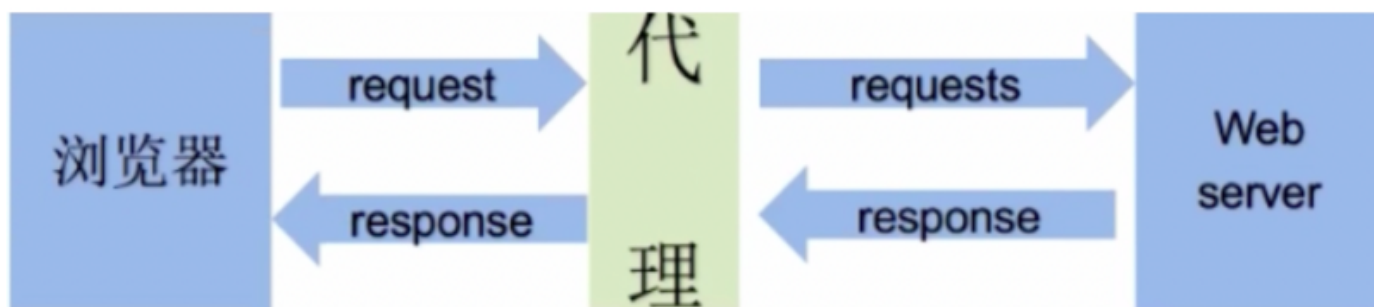
```
#2.使用session发起的请求，目的是为了捕获到
cookie，且将其存储到session对象中
session.get(url=main_url,headers=headers)

url =
'https://xueqiu.com/statuses/hot/listV
2.json'
param = {
    "since_id": "-1",
    "max_id": "311519",
    "size": "15",
}
#3.就是使用携带了cookie的session对象发起的
请求（就是携带者cookie发起的请求）
response =
session.get(url=url,headers=headers,pa
rams=param)
data = response.json()
print(data)
```

代理

- 什么是代理
 - 代理服务器

- 代理服务器的作用
 - 就是用来转发请求和响应



- 在爬虫中为何需要使用代理？
 - 有些时候，需要对网站服务器发起高频的请求，网站的服务器会检测到这样的异常现象，则会讲请求对应机器的ip地址加入黑名单，则该ip再次发起的请求，网站服务器就不在受理，则我们就无法再次爬取该网站的数据。
 - 使用代理后，网站服务器接收到的请求，最终是由代理服务器发起，网站服务器通过请求获取的ip就是代理服务器的ip，并不是我们客户端本身的ip。
- 代理的匿名度
 - 透明：网站的服务器知道你使用了代理，也知道你的真实ip
 - 匿名：网站服务器知道你使用了代理，但是无法获知你真实的ip
 - 高匿：网站服务器不知道你使用了代理，也不知道你的真实ip（推荐）
- 代理的类型（重要）

- http: 该类型的代理服务器只可以转发http协议的请求
- https: 可以转发https协议的请求
- 如何获取代理?
 - 代理精灵: <http://http.zhiliandaili.cn/>
 - 芝麻代理: <https://jahttp.zhimaruanjian.com/> (推荐, 有新人福利)
- 如何使用代理?
 - 测试: 访问如下网址, 返回自己本机ip

```
import requests
from lxml import etree
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/98.0.4758.80
Safari/537.36',
}
url = 'https://www.sogou.com/web?query=ip'
page_text =
requests.get(url=url,headers=headers).text
tree = etree.HTML(page_text)
data = tree.xpath('//*
[@id="ipsearchresult"]/strong/text()')[0]
print(data)
```

- 使用代理发起请求, 查看是否可以返回代理服务器的ip

```

import requests
from lxml import etree
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/98.0.4758.80
Safari/537.36',
}
url = 'https://www.sogou.com/web?query=ip'
#使用代理服务器发起请求
#proxies={'代理类型': 'ip:port'}
page_text =
requests.get(url=url,headers=headers,proxie
s={'https': '42.57.150.150:4278'}).text
tree = etree.HTML(page_text)
data = tree.xpath('//*[@id="ipsearchresult"]/strong/text()')[0]
print(data)

```

○ 深度测试：

- 对快代理进行n次请求，直到本机无法访问快代理为止（证明本机ip被快代理封掉了）
- 构建一个代理池（封装了很多代理ip和端口的容器），用于数据的批量爬取

```

from bs4 import BeautifulSoup

```

```
from lxml import etree
import requests
import time
import random
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/97.0.4692.71 Safari/537.36'
}
#构建一个代理池
proxy_url =
'http://webapi.http.zhimaangku.com/getip
?
num=15&type=2&pro=&city=0&yys=0&port=1&pa
ck=213751&ts=0&ys=0&cs=0&lb=6&sb=-
&pb=4&mr=1&regions='
json_data =
requests.get(url=proxy_url,headers=headers).json()
json_list = json_data['data']
proxy_list = [] #代理池,每次请求,可以随机从
代理池中选择一个代理来用
for dic in json_list:
    ip = dic['ip']
    port = dic['port']
    n_dic = {
```

```

        'https':ip+':'+str(port) #
{'https':'111.1.1.1:1234'}
    }
    proxy_list.append(n_dic)

#爬取多页
#1.创建一个通用的url(可以变换成任意页码的url)
url =
'https://www.kuaidaili.com/free/inha/%d/'
#2.通过循环以此生成不同页码的url
for page in range(1,4):
    print('-----正在爬取第%d页的数据! -
-----'%page)
    #format用来格式化字符串的（不可以修改url这个字符串本身）
    new_url = format(url%page)
    #循环发送每一页的请求
    #注意：get方法是一个阻塞方法！
    page_text =
requests.get(url=new_url,headers=headers,
proxies=random.choice(proxy_list)).text
    time.sleep(1)
    soup =
BeautifulSoup(page_text,'lxml')
    trs = soup.select('tbody > tr')
    for tr in trs:
        t1 = tr.findAll('td')[0]

```

```
t2 = tr.findAll('td')[1]
ip = t1.string
port = t2.string
print(ip,port)
```

验证码

- 图鉴平台: <http://www.ttshitu.com/> (推荐)
- 超级鹰: <https://www.chaojiying.com/about.html>
- 使用图鉴识别古诗文网登录中的验证码
 - 古诗文网: <https://so.gushiwen.cn/user/login.aspx?from=http://so.gushiwen.cn/user/collect.aspx>
 - 使用流程:
 - 注册登录图鉴平台
 - 登录后, 点击开发文档, 提取识别的源代码
 - 模块(tujian.py)的封装:

```
import base64
import json
import requests
# 一、图片文字类型(默认 3 数英混合):
# 1 : 纯数字
# 1001: 纯数字2
# 2 : 纯英文
```

1002: 纯英文2
3 : 数英混合
1003: 数英混合2
4 : 闪动GIF
7 : 无感学习(独家)
11 : 计算题
1005: 快速计算题
16 : 汉字
32 : 通用文字识别(证件、单据)
66: 问答题
49 :recaptcha图片识别
二、图片旋转角度类型:
29 : 旋转类型

三、图片坐标点选类型:
19 : 1个坐标
20 : 3个坐标
21 : 3 ~ 5个坐标
22 : 5 ~ 8个坐标
27 : 1 ~ 4个坐标
48 : 轨迹类型

四、缺口识别
18 : 缺口识别 (需要2张图 一张目标图一张缺口图)
33 : 单缺口识别 (返回x轴坐标 只需要1张图)
五、拼图识别

53: 拼图识别

#函数实现忽略

```
def base64_api(uname, pwd, img, typeid):
    with open(img, 'rb') as f:
        base64_data =
base64.b64encode(f.read())
        b64 = base64_data.decode()
        data = {"username": uname,
"password": pwd, "typeid": typeid,
"image": b64}
        result =
json.loads(requests.post("http://api.ttsh
itu.com/predict", json=data).text)
        if result['success']:
            return result["data"]["result"]
        else:
            return result["message"]
    return ""
```

def getImgCodeText(imgPath, imgType):#直接
返回验证码内容

#imgPath: 验证码图片地址

#imgType: 验证码图片类型

```
        result =
base64_api(uname='bb328410948',
pwd='bb328410948', img=imgPath,
typeid=imgType)
        return result
```

■ 验证码图片识别操作

```
from lxml import etree
import requests
import tujian
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/97.0.4692.71 Safari/537.36'
}
#将验证码图片请求后保存到本地
login_url =
'https://so.gushiwen.cn/user/login.aspx?
from=http://so.gushiwen.cn/user/collect.a
spx'
page_text =
requests.get(url=login_url,headers=headers
).text
```



```
tree = etree.HTML(page_text)
img_src =
'https://so.gushiwen.cn'+tree.xpath('//*
[@id="imgCode"]/@src')[0]
code_data =
requests.get(url=img_src,headers=headers)
.content
with open('./code.jpg','wb') as fp:
    fp.write(code_data)

#识别验证码图片内容
result =
tujian.getImgCodeText('./code.jpg',3)
print(result)
```

模拟登录

- 古诗文网
- 在抓包工具里定位点击登录按钮后对应的数据包：
 - 只要数据包的请求参数中包含用户名，密码和验证码则该数据包就是我们要定位的
 - 首次模拟登录操作：

```
from lxml import etree
import requests
```

```
import tujian
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/97.0.4692.71
Safari/537.36'
}
#将验证码图片请求后保存到本地
login_url =
'https://so.gushiwen.cn/user/login.aspx?
from=http://so.gushiwen.cn/user/collect.aspx'
page_text =
requests.get(url=login_url,headers=headers)
.text
tree = etree.HTML(page_text)
img_src =
'https://so.gushiwen.cn'+tree.xpath('//*
[@id="imgCode"]/@src')[0]
code_data =
requests.get(url=img_src,headers=headers).c
ontent
with open('./code.jpg','wb') as fp:
    fp.write(code_data)

#识别验证码图片内容
```

```
result =
tujian.getImgCodeText( './code.jpg', 3)
print(result)
#模拟登录
url =
'https://so.gushiwen.cn/user/login.aspx?
from=http%3a%2f%2fso.gushiwen.cn%2fuser%2fc
ollect.aspx'
data = {
    "__VIEWSTATE":
"opfVI7oolwkr7MLRVzsNSMASqLRUu01dg5ZP5EIRa4
FyM+mOYKES6KWEKQKaba2ulLoZQIaLFiKK4mr5K3ci1
v8ua28wtcRtabKWjOtJtU/i2etH+zSduegTMcg=",
    "__VIEWSTATEGENERATOR": "C93BE1AE",
    "from":
"http://so.gushiwen.cn/user/collect.aspx",
    "email": "15027900535",
    "pwd": "bobo@15027900535",
    "code":result ,
    "denglu": "登录"
}
#获取了登录成功后的页面源码数据
login_page_text =
requests.post(url=url,headers=headers,data=
data).text
with open('wushiwen.html','w') as fp:
    fp.write(login_page_text)
```

- 查看gushiwen.html发现，没有登录成功，原因：
 - 验证码不对（否定）
 - 没有携带cookie
 - 出现了动态变化的请求参数
 - 如何获取动态变化的请求参数
 - 基于抓包工具进行全局搜索，发现该参数值被隐藏在登录页面的页面源码中

```
from lxml import etree
import requests
import tujian
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/97.0.4692.71 Safari/537.36'
}
#创建session对象
session = requests.Session()

#将验证码图片请求后保存到本地
```

```
login_url =
'https://so.gushiwen.cn/user/login.aspx?
from=http://so.gushiwen.cn/user/collect.a
spx'
page_text =
session.get(url=login_url,headers=headers
).text
tree = etree.HTML(page_text)
img_src =
'https://so.gushiwen.cn'+tree.xpath('//*
[@id="imgCode"]/@src')[0]
code_data =
session.get(url=img_src,headers=headers).
content
with open('./code.jpg','wb') as fp:
    fp.write(code_data)
```

#解析出动态变化的请求参数

```
__VIEWSTATE = tree.xpath('//*
[@id="__VIEWSTATE"]/@value')[0]
```

#识别验证码图片内容

```
result =
tujian.getImgCodeText('./code.jpg',3)
print(result)
```

#模拟登录

```
url =
    'https://so.gushiwen.cn/user/login.aspx?
    from=http%3a%2f%2fso.gushiwen.cn%2fuser%2
    fcollect.aspx'
data = {
    "__VIEWSTATE": __VIEWSTATE,
    "__VIEWSTATEGENERATOR": "C93BE1AE",
    "from":
    "http://so.gushiwen.cn/user/collect.aspx"
    ,
    "email": "15027900535",
    "pwd": "bobo@15027900535",
    "code":result ,
    "denglu": "登录"
}
#获取了登录成功后的页面源码数据
login_page_text =
session.post(url=url,headers=headers,data
=data).text
with open('wushiwen.html','w') as fp:
    fp.write(login_page_text)
```

防盗链

- 现在很多网站启用了防盗链反爬，防止服务器上的资源被人恶意盗取。什么是防盗链呢？
 - 以图片为例，访问图片要从他的网站访问才可以，否则直接访问图片地址得不到图片
- 练习：抓取微博图片，url: <http://blog.sina.com.cn/lm/pic/>，将页面中某一组系列详情页的图片进行抓取保存，比如三里屯时尚女郎: http://blog.sina.com.cn/s/blog_01ebcb8a0102zi2o.html?tj=1
 - 注意：
 - 1.在解析图片地址的时候，定位src的属性值，返回的内容和开发工具Element中看到的不一樣，通过network查看网页源码发现需要解析real_src的值。
 - 2.直接请求real_src请求到的图片不显示，加上Referer请求头即可
 - 哪里找Referer：抓包工具定位到某一张图片数据包，在其requests headers中获取

```
import requests
from lxml import etree
headers = {
```

```
        'User-Agent': 'Mozilla/5.0 (Macintosh;  
Intel Mac OS X 10_15_7)  
AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/97.0.4692.71 Safari/537.36',  
        "Referer":  
        "http://blog.sina.com.cn/",  
  
    }  
    url =  
    'http://blog.sina.com.cn/s/blog_01ebcb8a0  
102zi2o.html?tj=1'  
    page_text =  
    requests.get(url, headers=headers).text  
    tree = etree.HTML(page_text)  
    img_src = tree.xpath('//*  
[@id="sina_keyword_ad_area2"]/div/a/img/@  
real_src')  
    for src in img_src:  
        data =  
        requests.get(src, headers=headers).content  
        with open('./123.jpg', 'wb') as fp:  
            fp.write(data)  
        # break
```


图片懒加载(作业)

- url: <https://sc.chinaz.com/tupian/meinvtupian.html>
- 爬取上述链接中所有的图片数据