Cookies 池

起因

不登录直接爬取弊端:

- 设置了登录限制的页面无法爬取。如某论坛设置了登录才可查看资源,某博客设置了 登录才可查看全文等,这些页面都需要登录账号才可以查看和爬取。
- 一些页面和接口虽然可以直接请求,但是请求一旦频繁,访问就容易被限制或者 IP 直接被封,但是登录之后就不会出现这样的问题,被反爬的可能性更低。

以微博为例,找新浪财经官方微博的信息 Ajax 接口:

https://m.weibo.cn/api/container/getIndex?uid=1638782947&luicode=20000174

&type=uid&value=1638782947&containerid=1005051638782947

如果用浏览器直接访问,返回的数据是 JSON 格式

这个接口在没有登录的情况下会有请求频率检测。如果一段时间内访问太过频繁,比如不断刷新,则会看到请求频率过高的提示,如下图所示。



而登录状态可以正常显示。登录账号可以降低被封禁的概率。

思路

登录之后再做爬取,被封禁的几率会小很多,但是也不能完全排除被封禁的风险。如果一直 用同一个账号频繁请求,那就有可能遇到请求过于频繁而封号的问题。

如果需要做大规模抓取,就需要拥有很多账号,每次请求随机选取一个账号,这样就降低了单个账号的访问频率,被封的概率又会大大降低。

多个账号的登录信息维护,就需要用到 Cookies 池。

Cookies Pool 的实现

以新浪微博为例来实现一个 Cookies 池的搭建过程。

Cookies 池实现的功能:

- ▶ 保存多个新浪微博账号和登录后的 Cookies 信息
- ▶ 检测 Cookies 池中每个 Cookies 的有效性
- ➤ 删除无效 Cookies 并模拟登录生成新的 Cookies
- ▶ 随机获取 Cookies 的接口

概要设计

CookiesPool

- -conn
- -cur
- -memberName
- -id_list
- +gen_cookies(self,loginname,password)
- +check_cookies(self,username,cookies,home_url)
- +save(self,username,password,cookies,home_url)
- +get_proxy(self)
- +delCookies(self,username)
- +close(self)

属性:

conn:数据库连接

cur: Cursor 对象

id list: 存储登录的账户名

接口方法:

get_from_db(self):从数据库中读取已有的 cookie,并检验其有效性 gen_cookies(self,loginname,password):根据帐号密码生成登陆后的 cookie check_cookies(self,username,cookies,home_url):检验 cookies 是否有效 save(self,username,password,cookies,home_url):cookies 存储到数据库 get_proxy(self):随机提取可用的 cookies delCookies(self,username):删除指定的无效 cookies close(self):关闭数据库

代码实现