代理+cookie+session+request其余常用参数

- 反爬：封IP
  - 请求次数一下很多 不像人在访问
  - 对网站造成破坏 或者达到非法要求
- 代理
  - 让服务器以为是不同用户在请求 让我们爬虫能请求成功
  - 代理匿名度分类
    - 透明代理
      - 直接访问网站 网站后台可以看到你的IP
        - 代理服务器将客户端的信息转发至目标访问对象，并没有完全隐藏客户端真实的身份，即服务器知道客户端使用了代理IP，并且完全知道客户端的真实ip
    - 普通匿名
      - 代理服务器用自己的IP 代理你的IP 但是 告诉了目标服务器是代理访问
    - 高匿名
      - 隐藏代理信息 让服务器觉得是真人访问 服务器看到的是代理IP
        - 服务器不会察觉到客户端是通过代理实现访问的
    - requests.get(url='http://httpbin.org/get', proxies=proxie)
- 爬取需要登录的页面
  - 需要我们登陆才能爬取
    - 代码登录
      - 1. 抓取登陆接口
      - 2 . 用代码登录 发送账号密码成功登录
      - 3. 访问需要登录才能查看的页面 带登录后的cookie 进行请求
    - 手动拿cookie登录
      - 直接在网页手动的拿登录的cookie
      - 写在请求头
        - headers传参
          - cookie写入headers中
        - 注意 cookie没有s
      - cookies参数传递
        - 使用get方法中的cookies参数进行传递 注意：参数必须为字典类型或者cookiejar对象
      - 应用场景
        - 当代码模拟登陆不好编写的时候(结构复杂，加密)
        - 只有cookie 没有账号密码
        - 网站根据cookie对用户进行推荐，专门爬取某人的个人推荐的时候
- cookie
  - 简单来说 就是维持登陆状态
  - cookie是什么
    - 1. cookie是浏览器访问服务器 服务器传给浏览器的一段数据
    - 2. 浏览器需要保存这个数据
    - 3，以后请求 浏览器需要携带这个数据
  - cookie是有时效性的
  - Cookie通过在客户端记录信息确定用户身份
  - cookie 不安全 可以不知道你的账号密码 获取你的cookie 然后登录 不同的网站对cookie的操作是不一样的
- session
  - 这个对象代表一次用户会话：从客户端浏览器连接服务器开始，到客户端浏览器与服务器断开。
  - Session通过在服务器端记录信息确定用户身份
  - Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，　服务器把客户端信息以某种形式记录在服务器上。这就是Session。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。
  - requests.session() 会话保持 如果session登录 就会一直保持
  - 说白了就是比如你使用session成功的登录了某个网站，则在再次使用该session对象请求该网站的其他网页都会默认使用这个session　之前使用的cookie等参数
  - 应用场景
    - 需要爬虫保持登录状态
    - 爬虫需要携带一些特定的参数 比如购物网站的无登录状态下的推荐 主要是cookie　(没有登录的情况下给你推荐你喜欢的，他把商品的标签写进cookie了)
  - 注意事项
    - 先要进行session的实例化，实例化之后所有的请求都使用 实例化的变量进行get/post方法调用 如：session.post()
    - requests.post() 这种方法将不会携带之前的cookie等参数 相当于重新来过
- 超时处理 timeout
  - timeout=(3,7) (请求时间，响应时间) 请求超过三秒报错 响应时间超过7秒报错
- 异常处理
  - 开发中经常会用到因为每写一个代码就会有一定的可能性报错 如果这个报错信息你没加在log日志里面 就会出问题
  - 精准捕捉异常 只有是超时或者其子类才会被捕获

XMind:ZEN | 试用模式