

1. 访问网页所涉及的 http 相关的逻辑，需要提供的内容：
  - url: 应该就是 get 或 post 包数据中的 general 中的 Request URL。
  - headers: 部分可选，部分必须
  - cookies (可选)
  - post data, POST 方法时，才需要然后获得的内容：
  - html 源码 (或其他的, json 字符串, 图片的数据等等)
  - cookie (可能有): 对于后续的访问其他的 url, 可能需要提供此处所返回的 (新的) cookie。
2. 通过代码模拟浏览器的行为, 需要分析某些 Header, 抽取其中重要的内容, 忽略次要的内容, 组合成 url, 然后结合 post data (get 时为空), 访问这个 url。
3. url 提交 POST 请求时, 对应的有个特殊的 header, Content-Type, 其值一般都是 application/x-www-form-urlencoded。
4. Post Data 中也有部分固定写死没有变化的数据, 需要多次调试才能找出固定不变的内容。
5. 对于 Request Headers 中的值, 很多项, 比如 Accept-Language en-us、Cache-Control no-cache, 对于程序实现是, 往往是 (但不绝对是), 无关紧要, 可以直接忽略, 即在代码中, 可以不设置这些参数的。
6. 在开启捕获浏览器的抓包之后, 对于打开一个网页, 自动就会捕获所有内容, 但是期间如果网页自动跳转到另外别的地址, 那么有些浏览器会自动清除之前捕获的内容, 而只显示最新当前网页相关的内容, 这种情况需要修改设置。
7. 一般 cookie 具有以下属性:
  - Domain: 域, 表示当前 cookie 所属于哪个域或子域下面。
  - Path: 表示 cookie 的所属路径。
  - Expire time / Max-age: 表示了 cookie 的有效期。expire 的值, 是一个时间, 过了这个时间, 该 cookie 就失效了。或者是用 max-age 指定当前 cookie 是在多长时间之后而失效。如果服务器返回的一个 cookie, 没有指定其 expire time, 那么表明此 cookie 有效期只是当前的 session, 即是 session cookie, 当前 session 会话结束后, 就过期了。
  - secure: 表示该 cookie 只能用 https 传输。一般用于包含认证信息的 cookie, 要求传输此 cookie 的时候, 必须用 https 传输。
  - httponly: 表示此 cookie 必须用于 http 或 https 传输。这意味着, 浏览器脚本, 比如 javascript 中, 是不允许访问操作此 cookie 的。
8. 对应的 Set-Cookie, 是从服务器端发送 cookie 给客户端的 cookie 数据, 包括了对应的 cookie 的名称、值、以及各个属性。

```
Set-Cookie: lu=Rg3vHJZnehYLjVg7qi3bZjzg; Expires=Tue, 15 Jan 2013 21:47:38 GMT;
Path=/; Domain=.foo.com; HttpOnly
Set-Cookie: made_write_conn=1295214458; Path=/; Domain=.foo.com
Set-Cookie: reg_fb_gate=deleted; Expires=Thu, 01 Jan 1970 00:00:01 GMT; Path=/;
Domain=.foo.com; HttpOnly
```

9. 从客户端发送 cookie 给服务器的时候, 是不发送 cookie 的各个属性的, 而只是发送对应的名称和值。

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

10. 除了服务器发送给客户端 (浏览器) 的时候, 通过 Set-Cookie, 创建或更新对应的 cookie 之外, 还可以通过浏览器内置的一些脚本, 比如 javascript, 去设置对应的 cookie, 对应实现是操作 js 中的 document.cookie。
11. Host 不是自己指定/设置的, 而是 http 请求会自动去设置。
12. POST 类型请求, 除了填写数据, 还要设置 ContentType 为 application/x-www-form-

urlencoded。

13. User-Agent 在很多时候，很重要，用于识别不同浏览器，以实现不同的行为。
14. Accept-Encoding 设置为 gzip, deflate 会导致返回的 html 网页是乱码，因为浏览器对于从服务器中返回的对应的 gzip 压缩的网页，会自动解压缩，所以，其 request 的时候，添加对应的头，表明自己接受压缩后的数据。而如果设置 Accept-Encoding 需要设置对应的自动解压缩的模式。
15. 可以通过关闭“自动运行重定向”，以获取重定向中所返回的 cookie 值。
16. 如果模拟网站登陆，那么基本逻辑都是，填写了用户名和密码，以及相关数据，编码后，然后用 POST 方法提交请求，服务器返回的认证信息，一般都包含在对应的 cookie 中，所以，如果管理和分析 cookie，就很重要。
17. 有时候登陆网站时页面需要跳转，跳转后自动清除了跳转之前的抓包数据，那么需要设置不清除跳转之前的数据，在 chrome 中为 preserve log。如果不点上这个，抓包时根本抓不到登陆时的 post 包。



18. 想要模拟网站登陆，就要知道，要向什么 url 地址，发送什么样的数据，GET 请求还是 POST 请求。
  - GET 请求只从服务器请求数据，不需要所谓的 post data，但是往往需要在 url 后面加上对应的?para1=val1&para2=value2 之类的形式，此部分叫做 query parameter，其本质上，有点类似于 post data。
  - POST 请求，在发送请求时，还需要提供对应的 post data。
19. 有时候在 post 请求中的 form data 中的 url 是加密的，这是因为协议规定，url 地址中，只能是 0-9 的数字、大小写字母 (a-zA-Z)、短横线不能包含其他字母，换句话说，如果其中包括了很多特殊符合，比如\$-\_.+!\*(), 那么都要尽量编码。
20. 通过直接查看网页源码，是找不到的动态网页的内容。其内容生成有以下方式：
  - 本地的 Javascript 脚本所生成
  - 通过访问另外一个 url 地址获得
21. 抓取动态网页的数据：
  - 如果所要抓取内容和 js 执行逻辑有关系：那就得靠自己去分析，调试 js 执行的过程，最终找到是如何一点点计算出来最终你需要的值的；
  - 如果所要抓取内容和 js 执行没关系：即，虽然你想要抓取的内容，是 js 执行生成的，但是最终还是可以在别的某个 js 文件或者其他返回的 html 代码中可以直接获得，那么你可以不用关系数据是如何来的，而直接去提取即可，即从特定的字符串中，提取你要的对应的内容。

22.