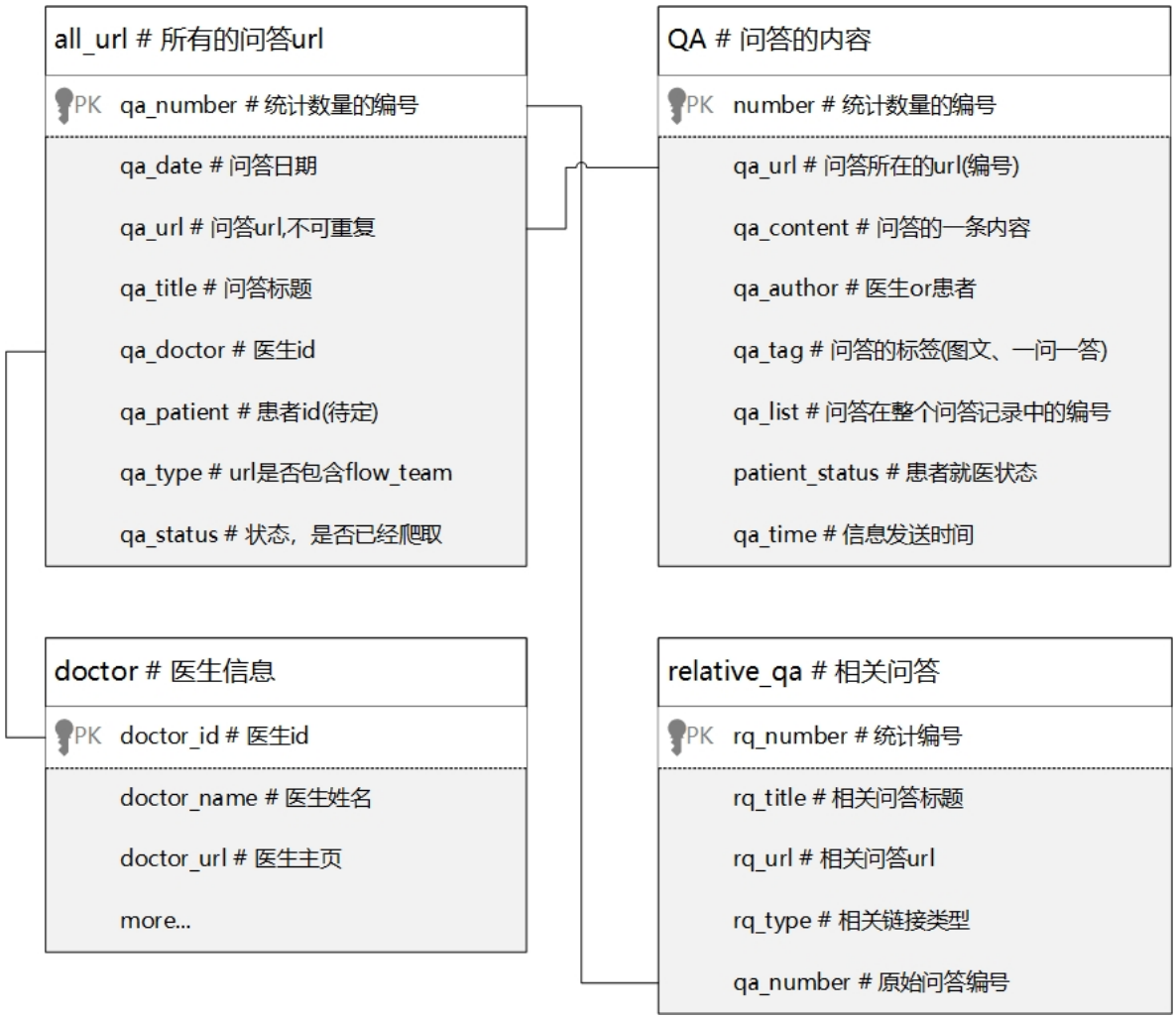
爬虫_从百度到知乎

> 这不是爬虫入门！！！ 准确的来说，抛开爬虫框架Scrapy，讨论如何自己定制一个独有的、简单的爬虫程序，以及期间可能遇到的各种问题与解决办法，更注重的是思路与想法，不会讨论如何解析数据和如何保存数据，那些...一般不会出问题吧？如果需要，以后再说。时间有限，可能在排版、内容、表述等方面都需要完善，会尽力完成。 --- author：王诚坤

一个完成的爬虫程序一般经过：访问----解析----数据保存。 但是在设计爬虫时，往往是反向进行，即：数据模型设计----页面逻辑梳理----访问。 原因：当你确定要爬取一个网站或者APP时，重点不在于你怎么爬取，而在于你需要什么**数据**，数据模型才是你应该关注的核心。其次才是你如何访问网站、如何解析页面等等。

数据模型设计

数据模型的设计通常比较简单，只需要按照需求与预期，结合网站或APP的内容，构建对应的数据模型。 **数据模型**：对于数据的抽象化概括。例如：爬取 好大夫 数据时，设计了医生、信息、问答等数据模型，每一个模型代表了一类信息。医生只包含了医生的记录，信息只包含了医生与患者之间的交流记录，问答包括了每个问答记录的内容(url、医生ID、问答类型等)。那么就可以设计一个如下的模型图：

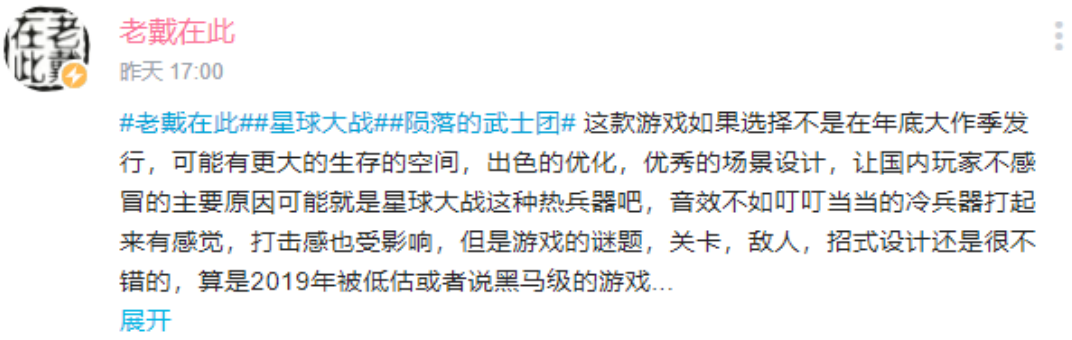


tips: 为了后续的数据处理，强烈建议对数据模型增加索引、主键、外键等约束，当然，不是说越复杂越好，而是能够满足需求即可。

页面逻辑梳理

这是没那么重要的步骤，但是也常常因为这一步骤考虑不充分，影响到数据模型的设计。页面逻辑，大致可以分为以下几种，分别对应了不同的处理方法：

- **链接**：通常是标签`<a href="****"></a>`，不排除其他标签，例如`<div>`。这种链接根据结构的不同可以分为两种：外部链接和内部跳转。
    1. 外部链接：通常是有完整的URL，直接指向了一个固定的网址，可以直接获取。例如：
    ```
    <a href="//www.haodf.com/doctorteam/flow_team_6465147698.htm">儿童鼻窦炎，脓鼻涕总是排不完 顽固鼻窦炎...</a> == $0
    ```
    ```
    <a class="Footer-item" target="_blank" rel="noopener noreferrer" href="https://zhuanlan.zhihu.com/p/51068775">侵权举报</a>
    ```
    2. 内部跳转：没有前面的`http`,指向了同一个网站的不同页面，需要按照网站要求，拼接完整的URL。例如：
    ```
    <a href="/s?wd=markdown%E7%94%A8%E4%BA%8E%E4%BB%80%E4%B9%88&rsf=1000002&rsp=0&f=1&o…&rsv_t=471drs4%2B5acSGmfedDZOWFS42aFI2G%2FKcmzfc%2FRNQzex%2Bty5Q6qR963B4ds">markdown用于什么</a>
    ```
- **隐藏**：有些页面为了美观，对于部分信息进行了隐藏，需要点击查看更多才能获得全部信息，但是往往网页都会预加载所有信息，可以在其他标签找到全部内容。例如： *网页效果:*



网页源代码:

```
<div data-v-4b3f0f1e class="content-full hidden">
    <a href="//t.bilibili.com/topic/name/%E8%80%81%E6%88%B4%E5%9C%A8%E6%AD%A4/feed" target="_blank" class="dynamic-link-hover-bg" style="cursor:pointer;color:#00a1d6;">#老戴在此#</a>
    <a href="//t.bilibili.com/topic/name/%E6%98%9F%E7%90%83%E5%A4%A7%E6%88%98/feed" target="_blank" class="dynamic-link-hover-bg" style="cursor:pointer;color:#00a1d6;">#星球大战#</a>
    <a href="//t.bilibili.com/topic/name/%E9%99%A8%E8%90%BD%E7%9A%84%E6%AD%A6%E5%A3%AB%E5%9B%A2/feed" target="_blank" class="dynamic-link-hover-bg" style="cursor:pointer;color:#00a1d6;">#陨落的武士团#</a>
    " 这款游戏如果选择不是在年底大作季发行，可能有更大的生存的空间，出色的优化，优秀的场景设计，让国内玩家不感冒的主要原因可能就是星球大战这种热兵器吧，音效不如叮叮当当的冷兵器打起来有感觉，打击感也受影响，但是游戏的谜题，关卡，敌人，招式设计还是很不错的，算是2019年被低估或者说黑马级的游戏。IW核心成员成立的重生工作室+战神艺术总监给我们带来了这款游戏，大家看看怎么样吧。"
</div>
```

- **按钮/提交**：不太常见，但是可能会用到，例如：百度首页的“百度一下”这个按钮，你想要获得词条信息，就要点一下。这在以关键词为核心的数据爬取中尤为重要。解决方法大致是两个思路：1.通过requests仿照请求(一般是get请求)，添加数据；2.通过selenium模仿人的操作，点一下那个按钮。

数据模型确认的是需要获取那些数据，而页面逻辑则确定的是通过怎样的页面跳转和页面解析获取这些数据。具体如何解析，建议按照熟悉的方式来：Beautisoup、XPath、CSS选择器等都可以。

模拟请求与反爬策略

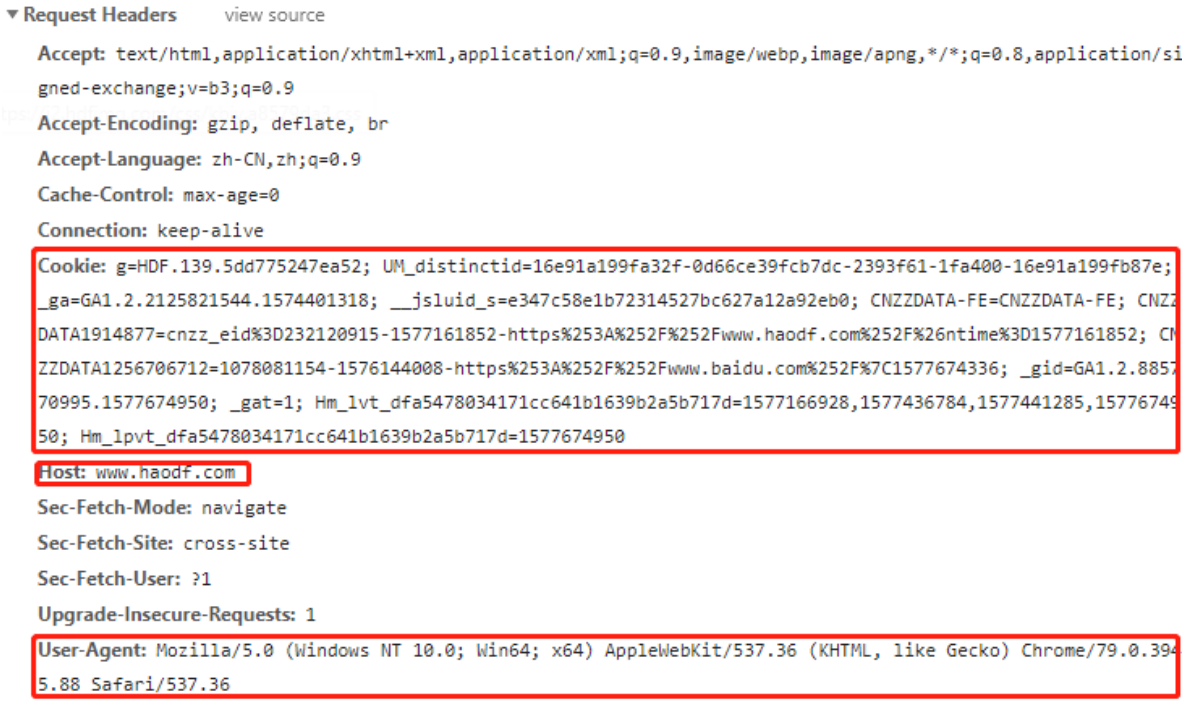
这是一个爬虫的开端，和最容易出问题的地方。在我们通过网页访问网页时，会自动的携带一些信息，这些信息包括：浏览器内核、Cookie、Host等。那么我们就可以完全仿照着模拟器访问这些网站，从而获得信息。所以后面按照各种反爬策略介绍请求方式：

- **没有任何反爬策略**

  一般都不存在这种网站，或多或少都会有一些反爬策略。

```
import requests
# 访问请求
a = requests.get('https://www.baidu.com/')
# 打印页面内容
print(a.text)
```

- **浏览器拦截**　　对于大多数的网站，不加任何数据的访问往往会被拦截，例如：好大夫，必须通过浏览器才能发送请求。所以，就有了Selenium，自动启动一个浏览器，进行访问。当然，也是可以通过requests添加header信息模拟浏览器请求，header要包含cookie、Host和User-Agent，但是对于某些网站，这样是不够的，需要更多的模仿全部的header数据。(如果出现`Accept-Encoding: gzip, deflate, br`，这一句不要加上去，这句话的意思是：接受的response是以压缩包的形式返回，没办法直接解析。)




```
# 方式一
from selenuim import webdriver
# 打开一个浏览器
browser = webdriver.Chrome()
browser.get("https://www.haodf.com/")
print(browser.page_source)

# 方式二
import requests
headers = {'Host': 'www.haodf.com',
           'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36'}
a = requests.get('https://www.haodf.com/', headers=headers)
# 可以尝试下不加headers会出现什么内容
print(a.text)
```

User-Agent表示用户使用的浏览器内核版本：

> **chrome win10** : User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36 **chrome win7** : Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1 **Firefox** : Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0 **Safari** : Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_1 like Mac OS X) AppleWebKit/603.1.30 (KHTML, like Gecko) Version/10.0 Mobile/14E304 Safari/602.1 **IE** : Mozilla/5.0 (compatible; MSIE 9.0; Windows Phone OS 7.5; Trident/5.0; IEMobile/9.0)

- **账户登录拦截**　　某些网站（例如：知乎），在网页上不登录是没有办法查看内容的。能实现账户登录拦截的原理，就是通过检查Cookie是否包含了用户登录信息，如果包含就可以查看，如果没有包含，就跳转到登录页面。那么针对这种页面，有两种思路：

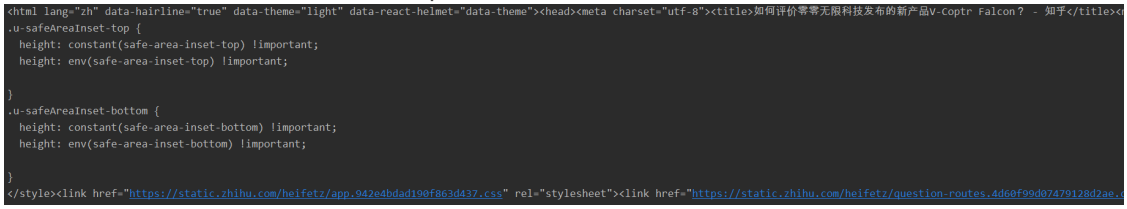
1. 先在浏览器登录，获得已经登录验证了的Cookie，然后在requests或selenium中添加headers(headers中包含Cookie)。

```
# 方式一：使用requests
import requests
headers = {'Cookie': "从浏览器复制Cookie,注意Cookie里面的引号，要替换成 \" "}
a = requests.get('https://www.zhihu.com/', headers=headers)
print(a.text)
# 方式二：使用selenium
from selenium import webdriver
option = webdriver.ChromeOptions()
# 添加cookie
option.add_argument("cookie=\"从浏览器复制Cookie,注意Cookie里面的引号！\"")
driver = webdriver.Chrome(options=option)
driver.get('https://www.zhihu.com/')
```

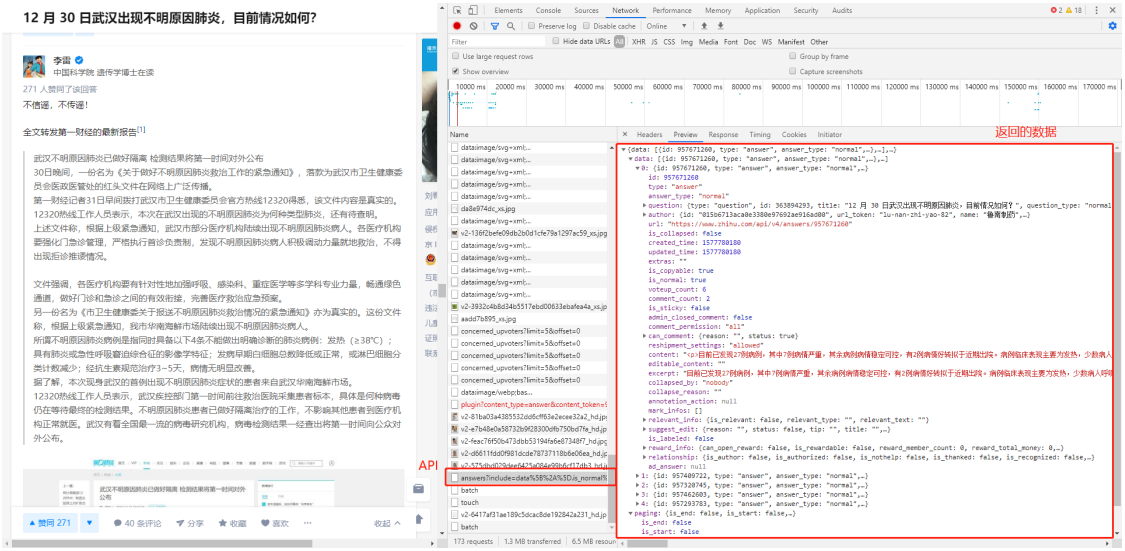
2. 使用selenium打开登录界面，输入账户密码，模拟登录，然后再读取页面。(这个方式对知乎不能用，因为知乎能够检测selenium打开的浏览器不能点登录按钮，可以尝试下。)

```
# 这个暂时没找到案例，待补充
# 大致思路就是：打开页面---找到账号密码的input标签---填入数据---点击登录---正常访问
```

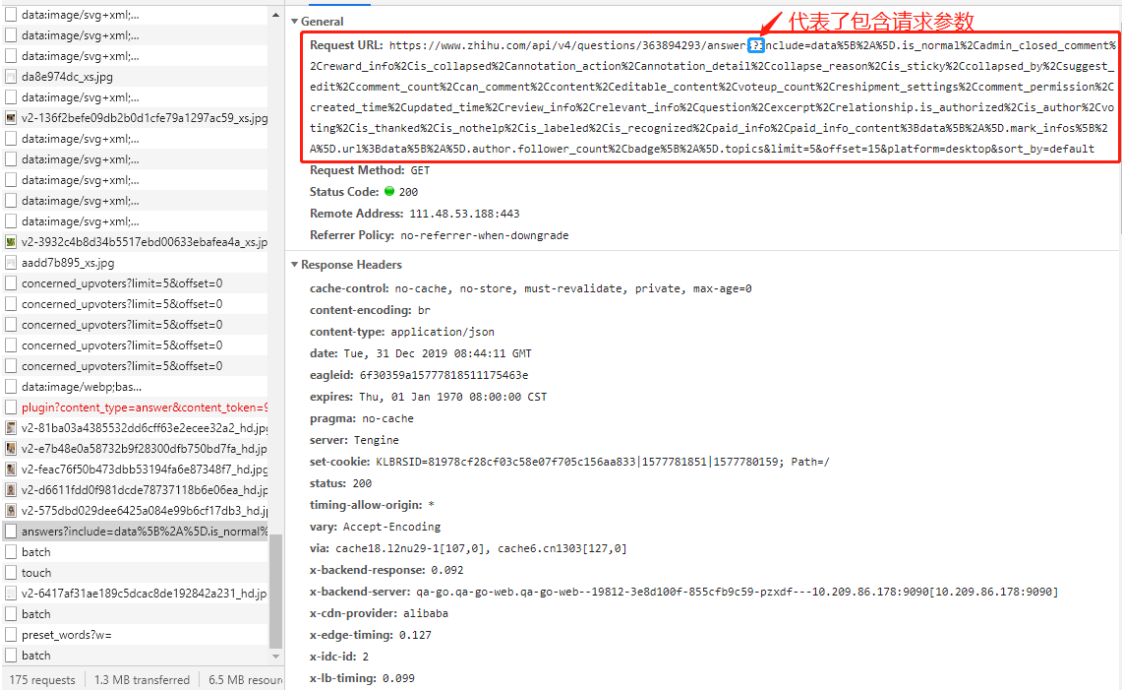
- **页面数据很多，想获得全部数据**　　这也是一个常见问题，当我们访问一个网页的时候，页面会采取异步处理，即你得到的是异步处理之前的js代码，只获得了部分数据，导致无法爬取所有信息，例如：知乎每个帖子的回答。requests或selenium访问时，得到的结果如下：



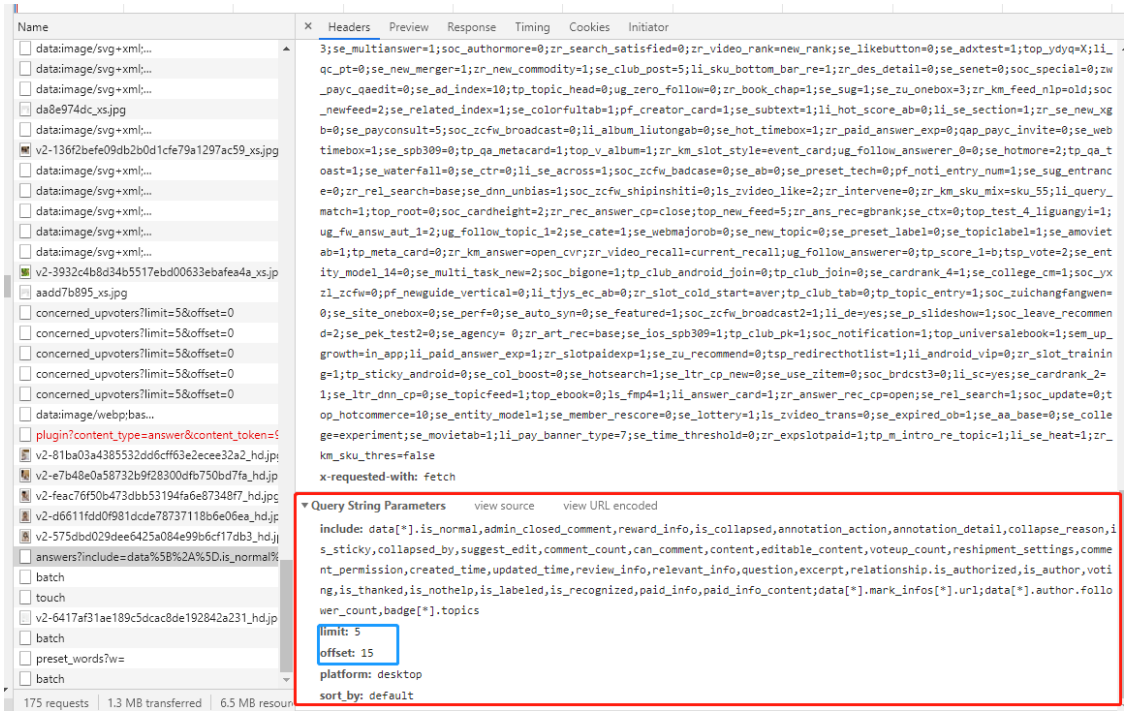
　　页面只包含了前五条、或十条数据。当遇到异步处理时，是没有办法直接获得全部你想要的数据的，页面内容只是一些排版和js信息，那么应对措施就是：找到异步访问数据需要的请求api，如下所示：



如何找到这个API呢？就是打开浏览器的开发者模式(Chrome快捷键是F12)，在网页页面上往下翻，当右边network出现新的内容时，随便点开一个请求，打开preview，看到一个JSON格式的数据(几乎和python的dict格式一样)，如果包含了你想要的内容，那就是你想要的API。下一步：查看这个请求的headers，点击preview旁边的headers即可。



通常而言，包含翻页操作的API请求都会携带参数，至少要包含limit和offset两个参数，因为这是实现翻页操作的基本要求。我们可以在headers往下看，会找到具体的参数名称和参数的值。



对比一下，发现除了limit和offset之外，怎么翻页都不会变，那么这个就是你要改变的参数，limit代表了每次加载多少数据，offset表示跳过前面多少条数据，具体解析可以看`MySQL教程`，当然，其他的网站也可能是其他名字，实在不清楚，可以自己尝试。下一步：仿照这个API，自定义翻页，并获取数据。知乎的sort_by代表了排序方式，只有两种：default和updated，分别对应默认和按更新时间。

```
import requests
headers = {"请求头需要包含的内容，主要是cookie"}

data = {"include": "那一大串", "limit":10, "offset":10,
        "platform": "desktop", "sort_by": "updated"}
# 把问号前面的copy成URL，此外，请求方式GET个POST的参数名称不一样，注意区别
s = requests.get("https://www.zhihu.com/api/v4/questions/363894293/answers",
params=data, headers=header)
# 因为获得的是JSON数据，可以直接转成dict
res = s.json()
print(len(res['data']))
for r in res["data"]:
    print(r["author"]["name"])
```

经过上述操作，就可以按照需求快速获得全部的答案了。此外，在对JSON解析的时候，建议用`开发者模式的preview或者在线的JSON解析网站`查看数据结构，不仅可以更清晰的看到你需要的字段名，往往还会有意想不到的收获。

- **IP访问次数限制** 这是一个最常见问题，也是难题。大多数网站都会有这一反爬策略，大致思路有以下几种：
    1. 直接封IP，结果就是：电脑在一段时间内没办法打开这个网站，在同一wifi下都不可以，好大夫就是这种策略，但好大夫封禁时间很短，`time.sleep(10)`，让程序休眠一下就好了；
    2. 验证码，在多次访问之后，网站要求验证码进行人机验证，包括但不局限于：数字字母验证码(百度)、繁体字验证码、汉字倒写验证码和12306那种丧心病狂验证码；
    3. 账号验证，会对账号短时间封禁，直到手动验证解除封禁，知乎是这种方式；

> *应对方式*:
>
> - 封IP和验证码，都可以通过添加代理IP的方式解决；
> - 验证码和账号验证，也都可以通过人工处理；

> - 账号验证也可以通过准备很多个账号，随机使用的策略避免封禁；
> - 验证码也可以通过一些公司提供的API，进行机器识别。

Other

- 关于数据存储：数据量较小时(<1w)，可以随便存储(txt、csv、tsv)，超过这个数量，更建议使用数据库，效率更高、操作体验更好、内存开销更小；关于python操作数据库，pymysql---https://pymysql.readthedocs.io/en/latest/
- 关于反爬策略：爬虫运行，是要考虑时间成本和金钱成本的，当爬虫的时间成本不重要时，就没必要考虑超过这个成本的反爬策略；
- 关于多线程与多进程：部分完成...文档链接：http://note.youdao.com/noteshare?id=5b44a197b71b08f0ac11b62f700dabae&sub=C077DCD8D43E4E6DB1A5294D18C3A632
- 关于代码BUG修复：写的代码总会出现一些意想不到的问题，编程的意义与乐趣也在这里，遇到问题，先试着自己解决...然后百度，最后，多读每个包的官方文档！！！
- 想花式秀操作： Requests---http://2.python-requests.org/zh_CN/latest/user/quickstart.html selenium---https://selenium.dev/documentation/zh-cn/ Beautifulsoup---https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/