1. 我们要需要做什么？

首先要了解我们的目的，我们目的在于帮助不能使用国内购物平台的用户进行代购服务。而要进行代购，就要帮助他们下单。为了帮助他们下单，很重要的一步就是要获取平台上商品的规格数据，让用户可以在浏览的时候，可以让他们在我们app上选择相应规格的商品。

需要的商品规格数据一般包括下图：

（比如衣服，有颜色、尺码、价格、库存等等；颜色有对应详细的红色、黄色、蓝色等；尺码有SML等），以及店铺的属性，比如店铺的名称、链接等等



1. 那么怎样才能获取到这些信息？

首先要清楚，app页面传给我们的只有原始的商品链接。

目前来说，获取页面的这些信息有两种方式。

一是通过分析页面构成，获取到相应的数据。这个方法是肯定能够获取的（因为页面上都展示出来了），不过会比较麻烦。

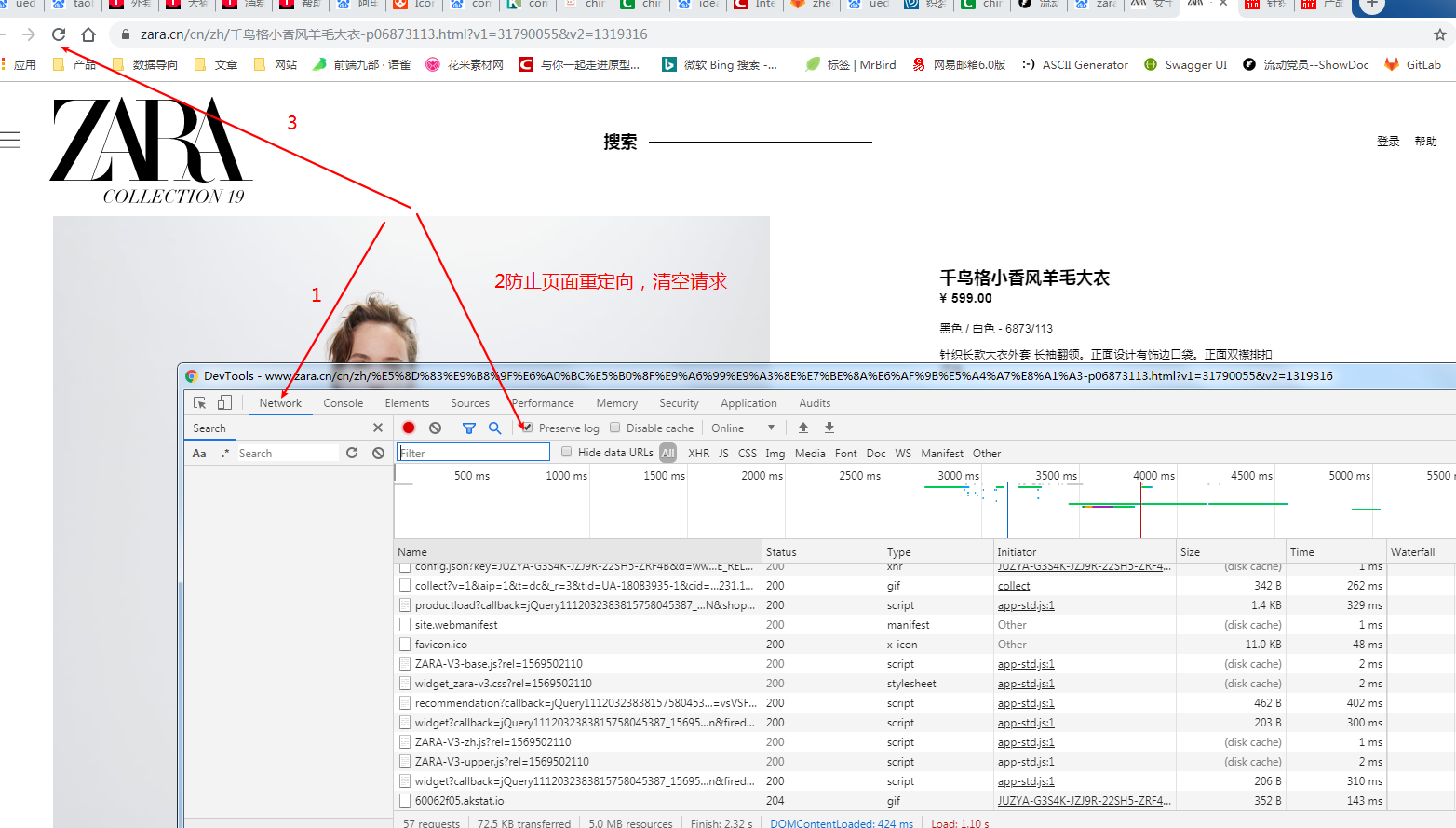
另外一种是分析页面调用接口后的返回数据。如果调用某个接口能够返回我们需要的全部商品规格数据，那么我们直接调用即可。

下面就两种方式详细说明整个思路。

**通过分析页面获取数据，以爬取zara商品为例：**

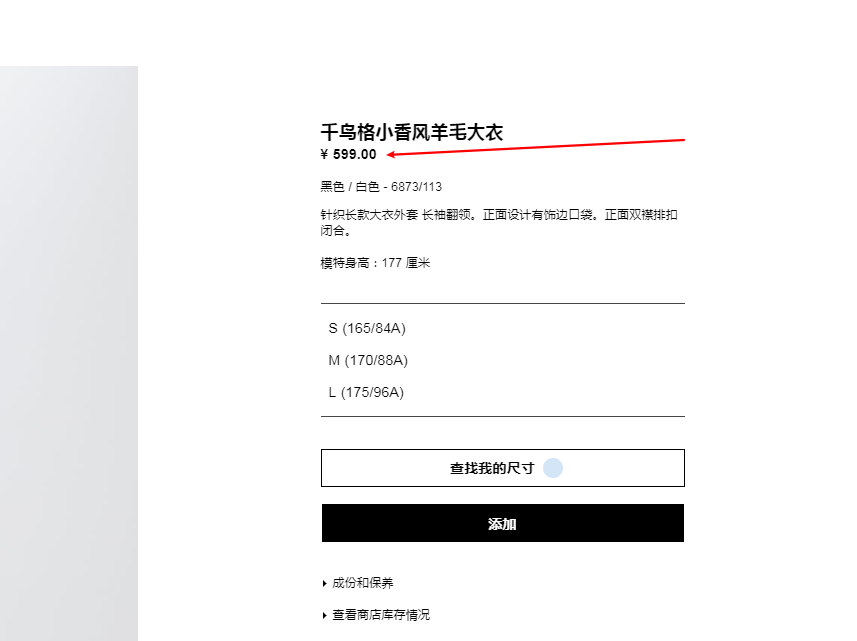
商品链接：<https://www.zara.cn/cn/zh/%E5%8D%83%E9%B8%9F%E6%A0%BC%E5%B0%8F%E9%A6%99%E9%A3%8E%E7%BE%8A%E6%AF%9B%E5%A4%A7%E8%A1%A3-p06873113.html?v1=31790055&v2=1319316>

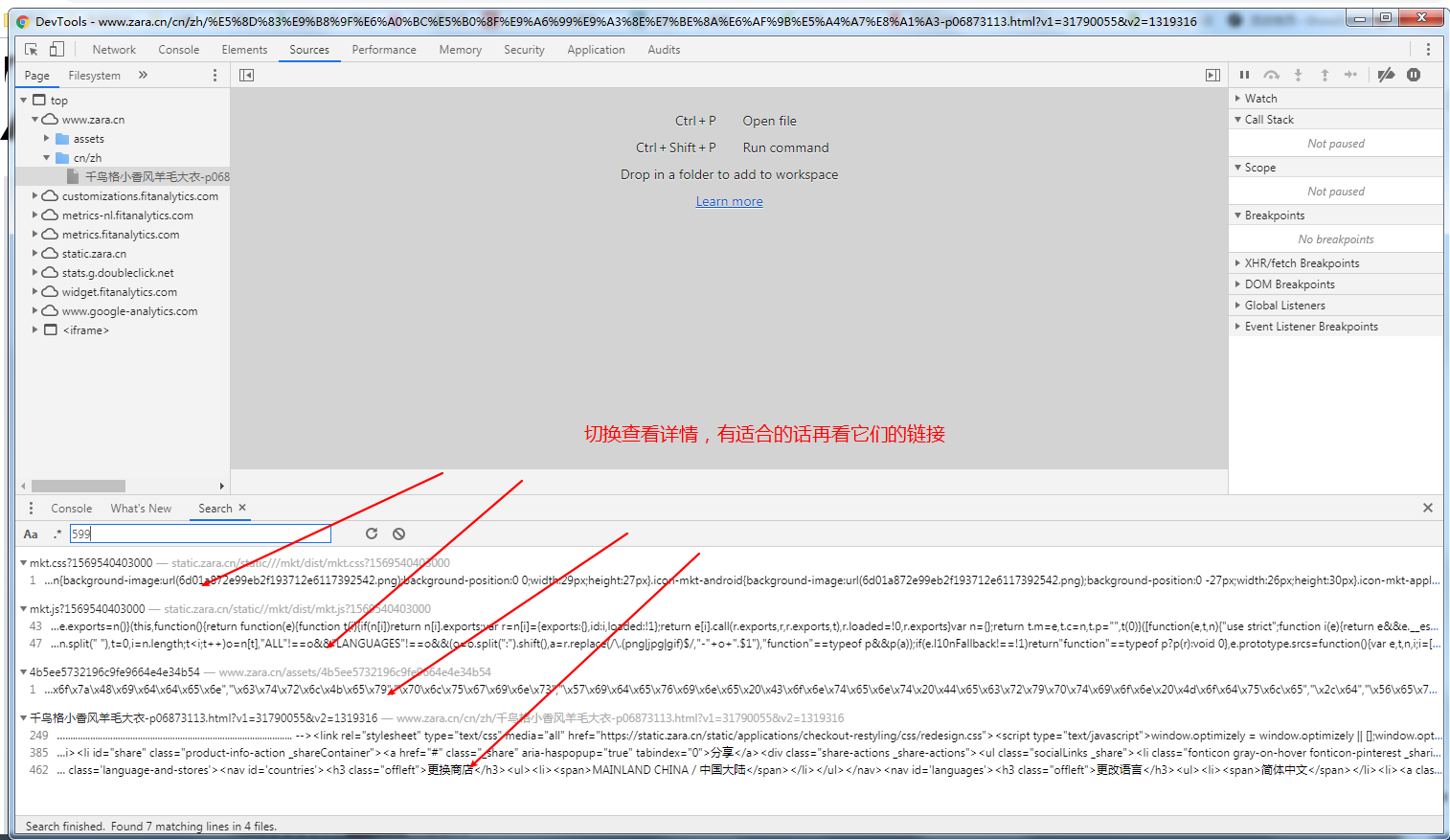
1. 使用谷歌浏览器打开商品链接，按f12进入调试模式，然后切换到network，再勾上perserve log，防止请求被清空，最后按刷新，就可以在network窗口查看进入商品页面后的所有请求。



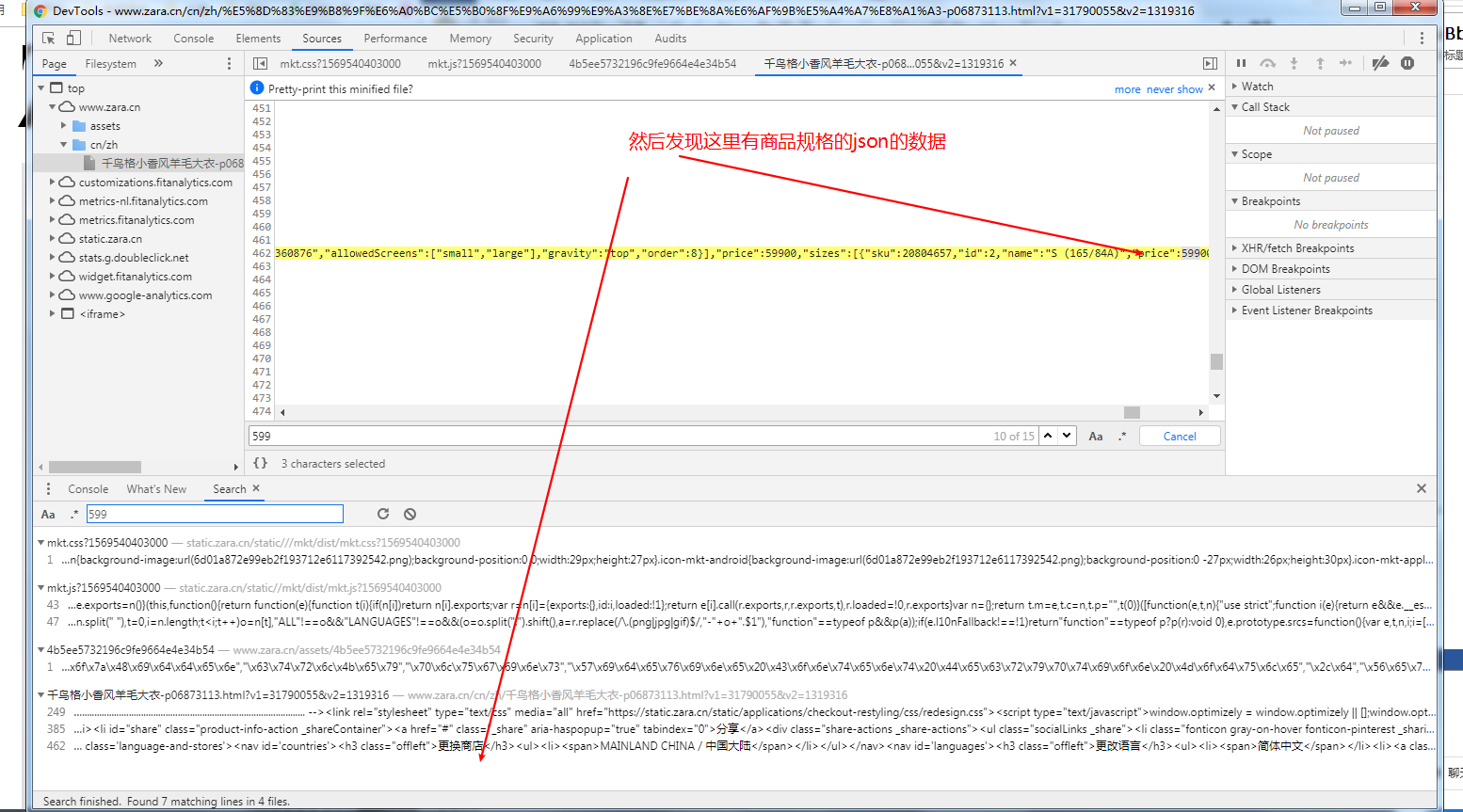
1. 观察商品的规格属性值，然后进行全局搜索，一般使用数字或者英文的属性值进行搜索，因为中文的属性值可能被编码，自然搜不出来。

比如这里使用599去全局搜索。全局搜索快捷方式是，切换到source，然后同时按下ctrl+shift+f





1. 然后发现有相关的规格数据



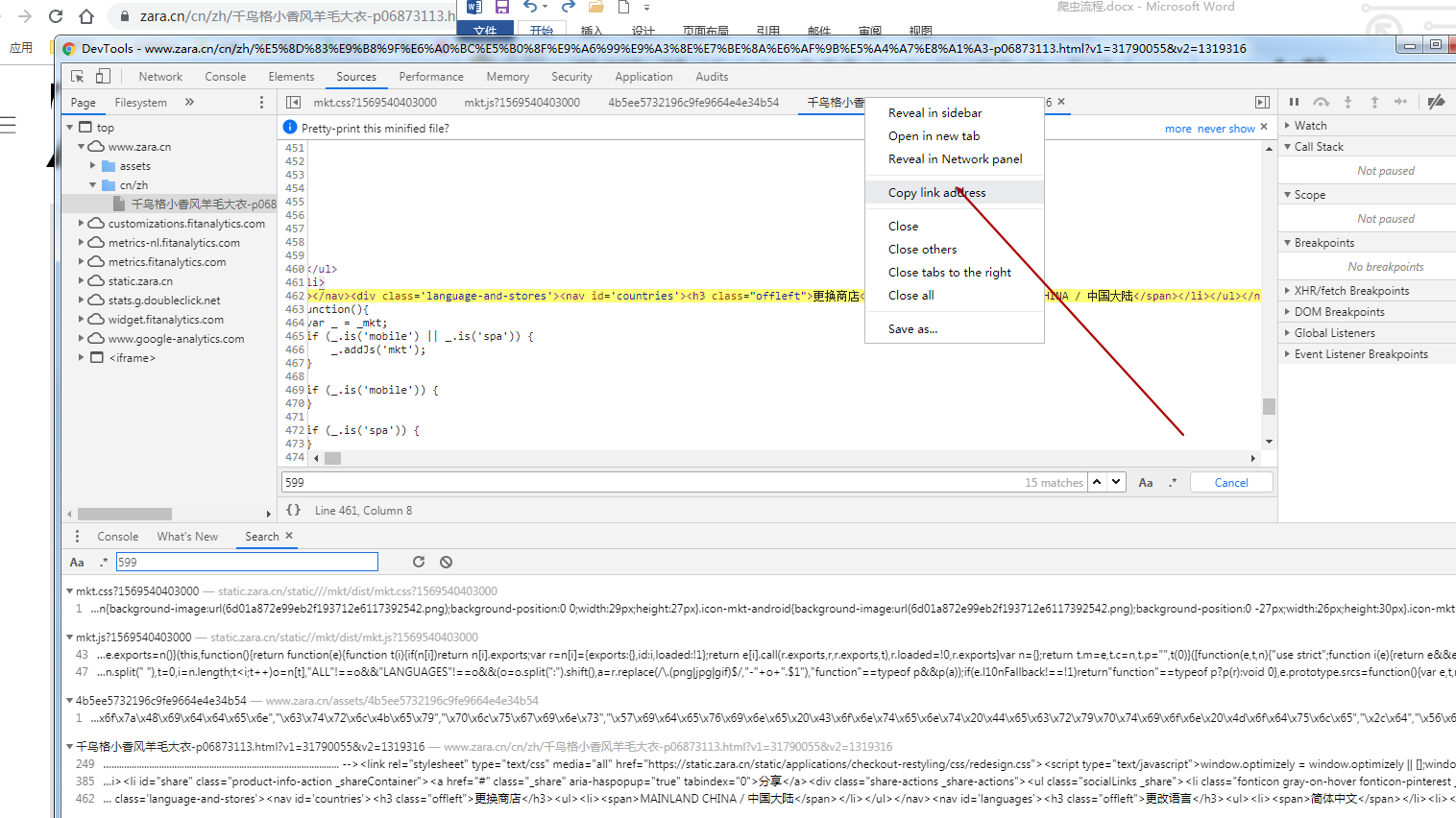
4）这个返回的数据全部复制出来到文本编辑器（比如sublime text）查看（方便观察）

然后发现这些数据在<script>标签下的某一个变量里面



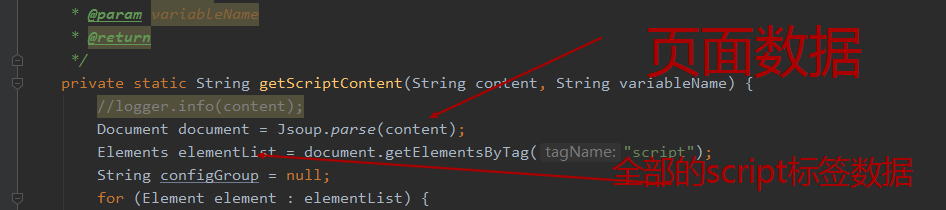
1. 数据已经有了，然后再看具体是哪个请求返回来的数据

https://www.zara.cn/cn/zh/%E5%8D%83%E9%B8%9F%E6%A0%BC%E5%B0%8F%E9%A6%99%E9%A3%8E%E7%BE%8A%E6%AF%9B%E5%A4%A7%E8%A1%A3-p06873113.html?v1=31790055&v2=1319316

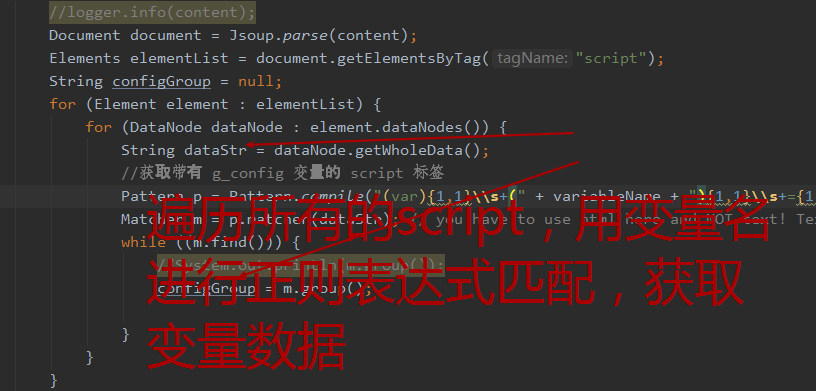


发现这个请求正是一开始的商品请求链接，也就是说，app把链接传到后台，后台在模仿浏览器发送链接请求，就可以返回包含这些数据的页面数据了（包含很多不需要的数据），所以接下来的问题就是怎么把这些数据截取出来。

在项目里面，我们使用的是jsoup工具，它能非常高效的分析出页面结构，以及获取页面标签。它只需要调用一个方法就可以获取到所有的<script>标签的数据。



获取到这些标签数据后，我们发现其实标签内往往不止一个变量数据，比如有window.zara.appConfig，window.zara.dataLayer、window.zara.viewPayload，然后再用变量名进行正则表达式匹配，获取到变量数据，也就是我们所需要的商品规格数据

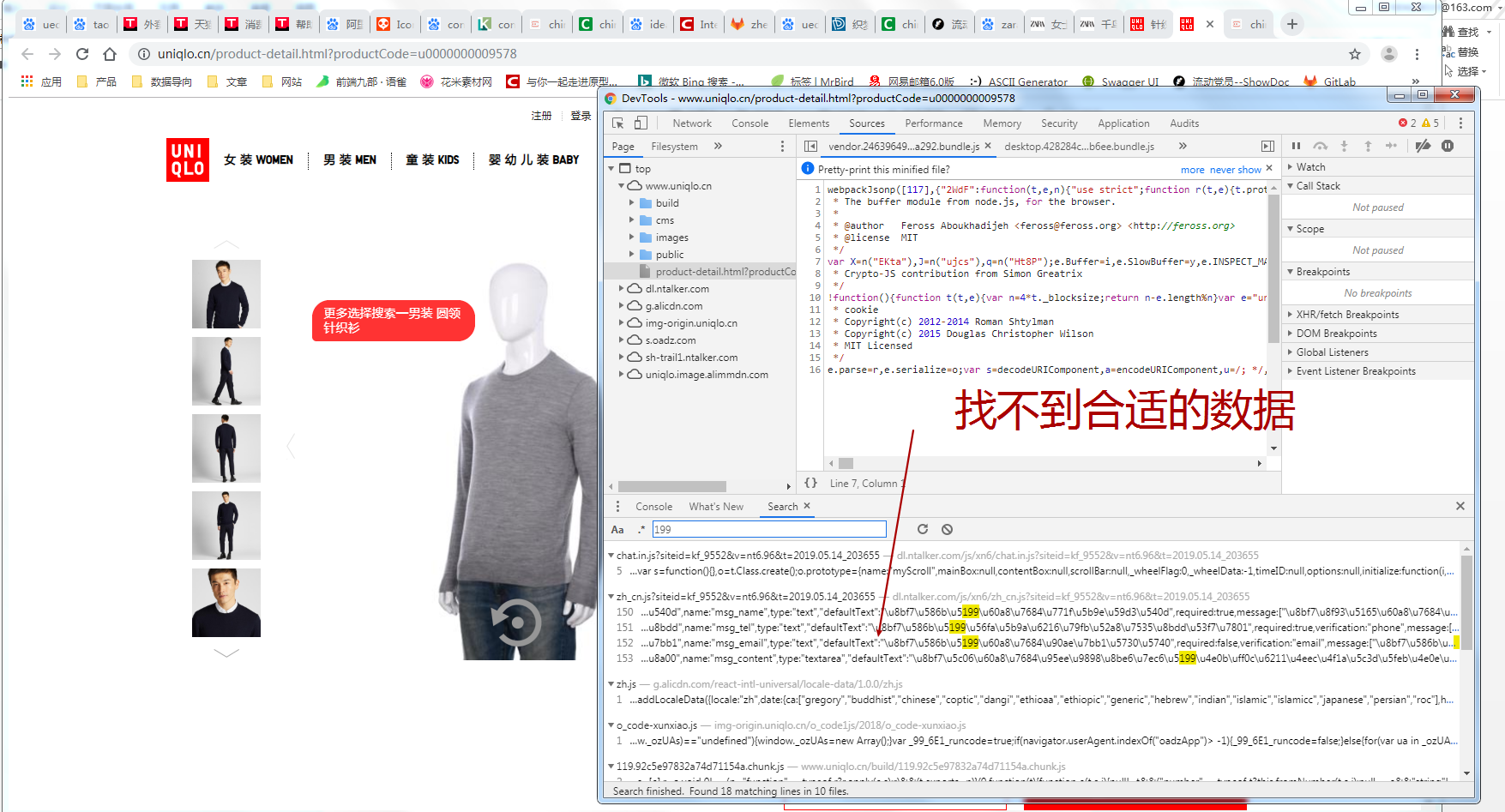


1. 再将这些数据进行格式化，然后返回即可。

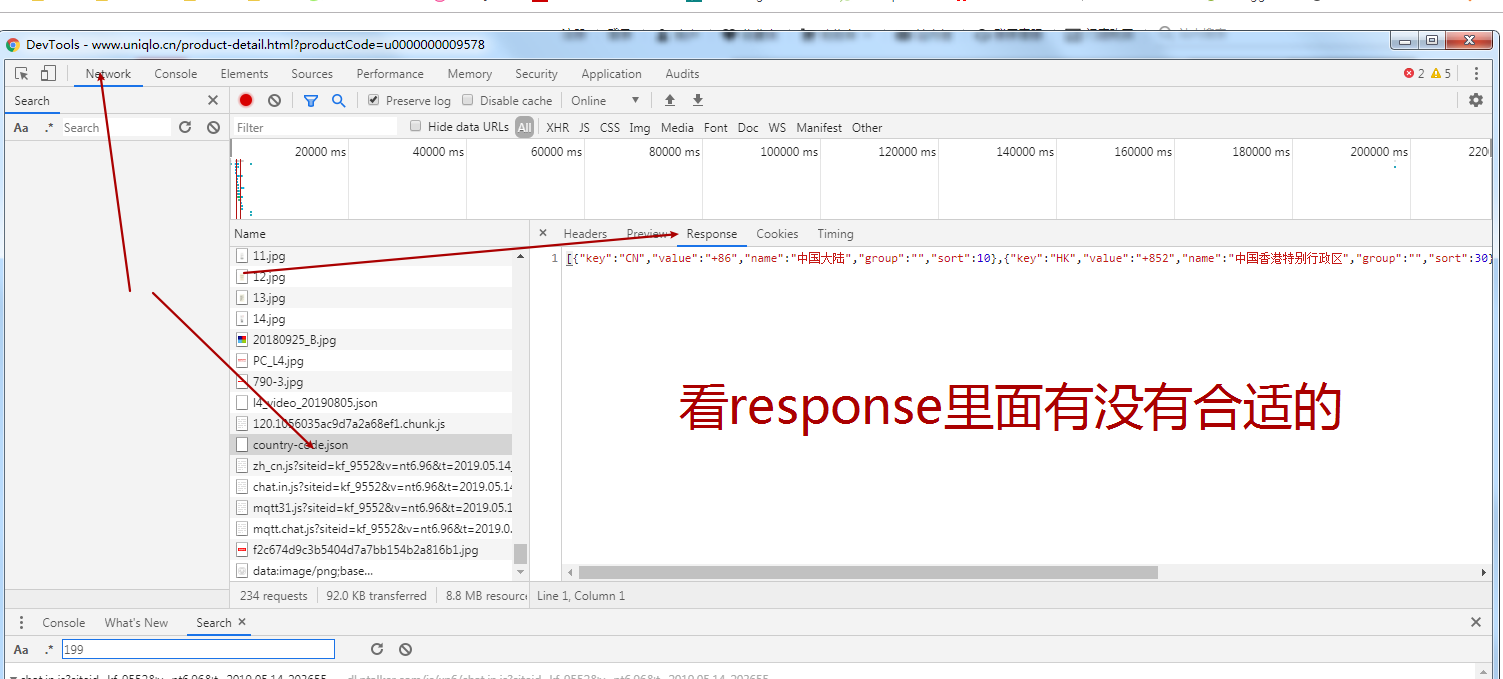
**上面是通过分析页面获取数据，流程比较麻烦。还有一些网站会简单很多，也就是下面的通过调用某个接口能直接返回数据，以爬取优衣库为例：**

商品链接：<https://www.uniqlo.cn/product-detail.html?productCode=u0000000009578>

1. 还是重复上面前三步，但是发现搜不到合适的数据

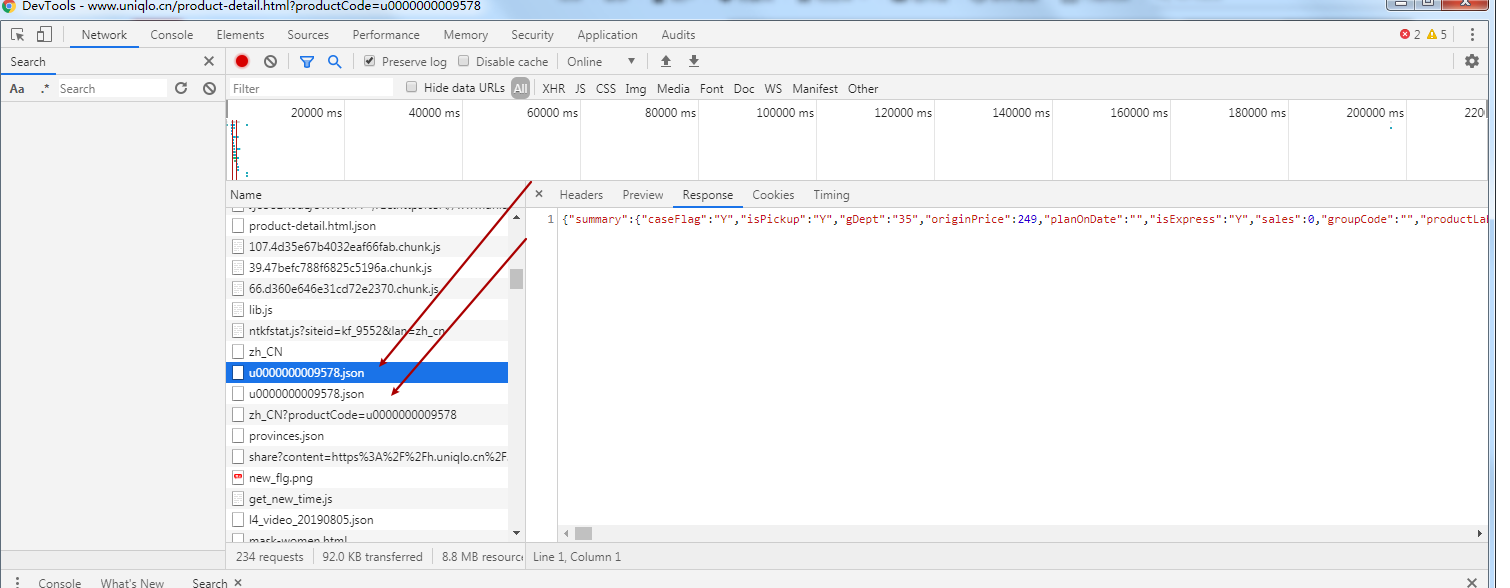


1. 搜不到的时候，可能是因为谷歌的全局搜索并没有在某些返回结果进行搜索。这个时候我们就要切换到network，对里面链接的response进行查看，看是否有合适的返回数据

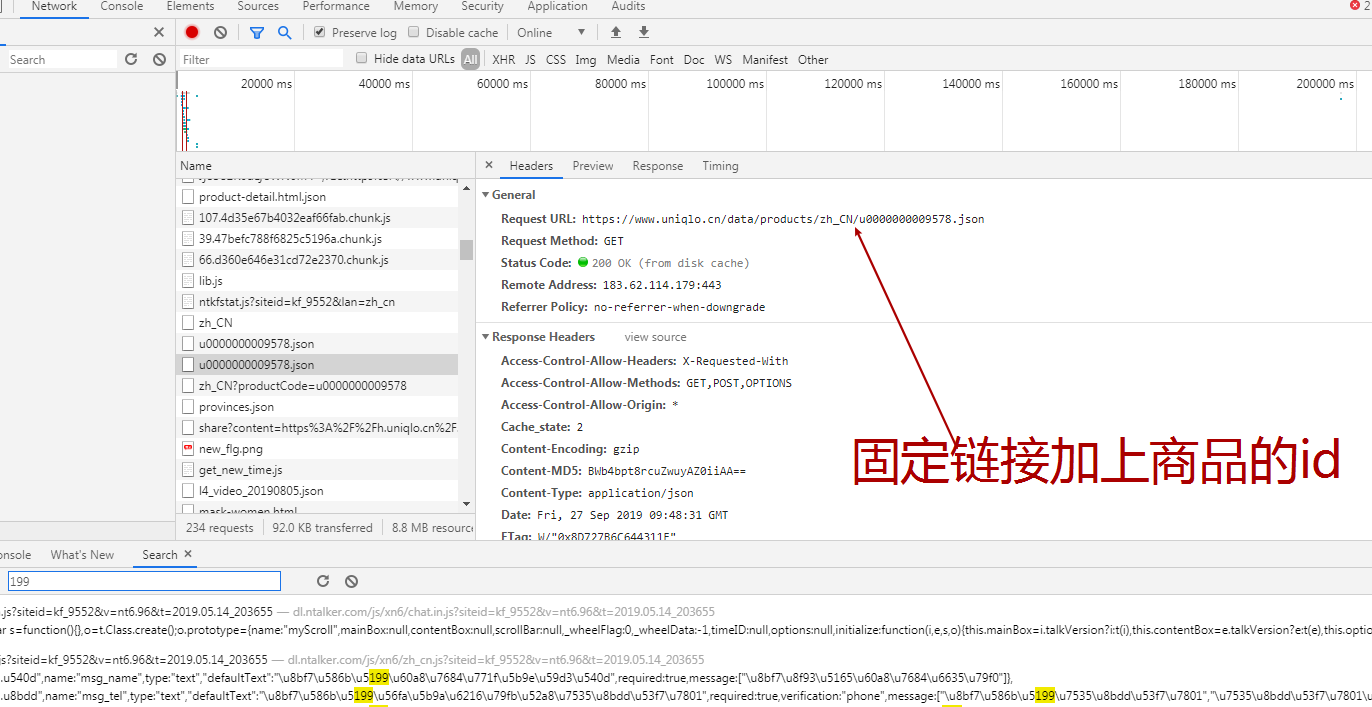


1. 最稳妥的办法是查看每一个请求的response，不过可以根据请求的名字进行优先查看。比如某个请求命名是“X-detail”之类的，一般都值得优先查看。

然后终于是找到了合适的返回数据



然后观察一下请求，发现也是我们能根据app页面传过来的链接能拼接的（app页面传过来的链接中包含id）



也就是说，我们要做的就是解析app传过来的链接，在获取里面的商品id参数，再拼接成能获取数据的接口连接，就可以得到商品规格数据了。

**上面就是目前两种的爬取方式，不同网站基本大同小异。其中淘宝网站要复杂一些。爬取淘宝网站的大致流程为：**

按照上述的第一种方法，获取到页面数据的商品基本数据和其中一个连接变量 sibUrl，然后再根据这个链接再去获取商品的规格数据。两个数据加起来就是我们需要的数据了。

注意：由于app访问的是wap页面，商品的链接和PC端的页面链接是不一样的，请求方式也不一样。所以如果在PC端找不到简便的爬取方法，还可以到手机端尝试。不过由于app服务器在香港的原因，可能会重定向，导致在app服务器发出请求返回的数据与大陆返回的数据是不一样的。目前发现使用PC端的链接爬取则不会存在这个问题。所以推荐优先使用PC端的请求爬取。

还有，如果使用PC端的请求爬取，那么要将app传过来的链接进行处理，因为app的链接是wap版的，和PC端链接不一样。

一般商品详情的链接都是带商品id的，然后按照PC端的方式拼接起来即可。

上面说的是爬取的过程，但在爬取之后，返回数据之前还有一些工作要做的，比如翻译属性。为的就是让用户能有更好地用户体验。翻译多线程调用腾讯的ai接口，有兴趣可以查看具体流程。

建议按照现有的代码规范，每一个爬虫用一个实现类，能很好区分不同网站的爬虫业务。