## 关于虎嗅爬虫制作的总结

1. 虎嗅的第一页为标准的静态网页，采用常规的response.xpath爬取数据即可

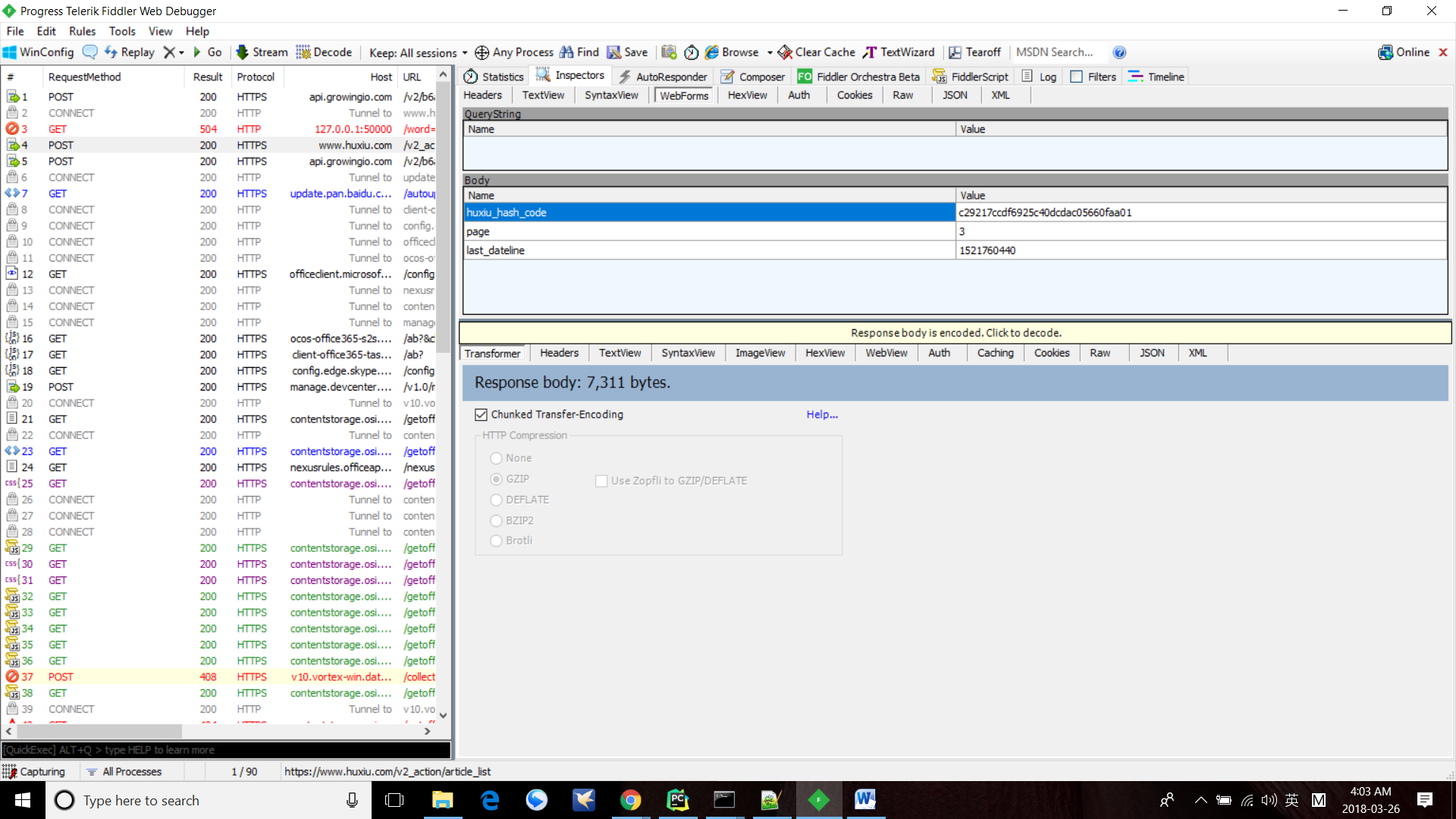
2.虎嗅的第二页开始均采用（点击‘加载更多’的方式）其实质就是AJAX方式刷新页面 通过安装CHROME Toggle JavaScript插件(关闭JavaScript功能;再次打开JavaScript功能).我们可以发现哪些网页内容是通过JavaScript动态生成的。

通过JavaScript动态生成的内容是在网页的viewsource code看不到的，因此不能简单的采用response.xpath方式进行数据的提取。 这时候就需要用到chrome network 开发工具和Fiddler抓包软件发挥作用力。

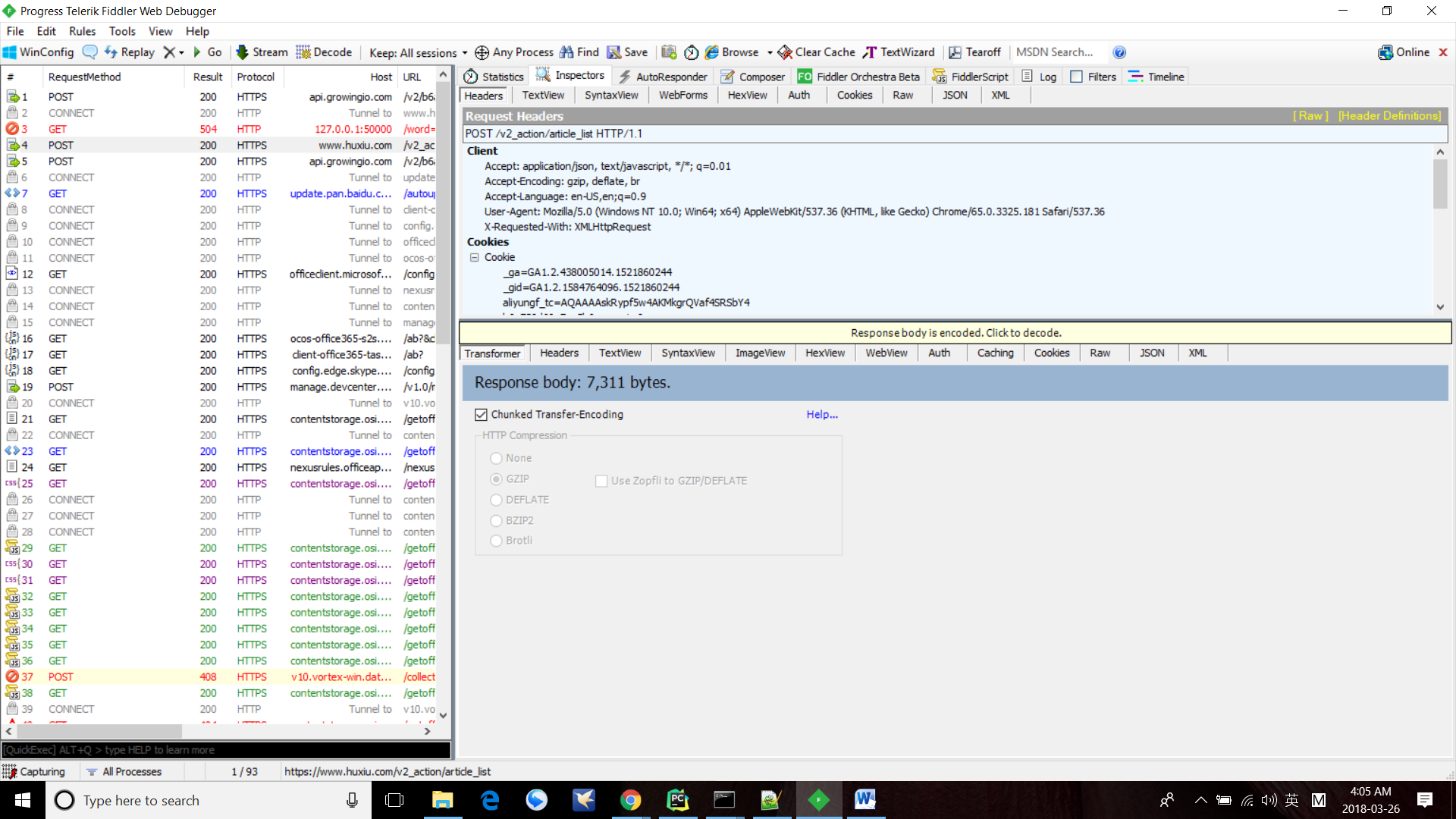
当我们发现从第二页开始的页面都是采用POST AJAX方式加载进行的，用户点击“加载更多”其实质是启动了一个AJAX POST请求,既然是POST方式我们可以抓取用户要求访问的URL和提交的数据表格。（这在开发Django表单提交数据的时候经常用到，看来对于WEB SERVER 的开发对于爬虫开发帮助很大呀！！！）

通过抓包分析POST访问,我们得到：

* Webforms为POST访问提交的数据表格



* Header中为用户需要访问的URL



用户要求访问的URL: https://www.huxiu.com/v2\_action/article\_list

进一步分析用户提交的数据表格：

form\_data= {

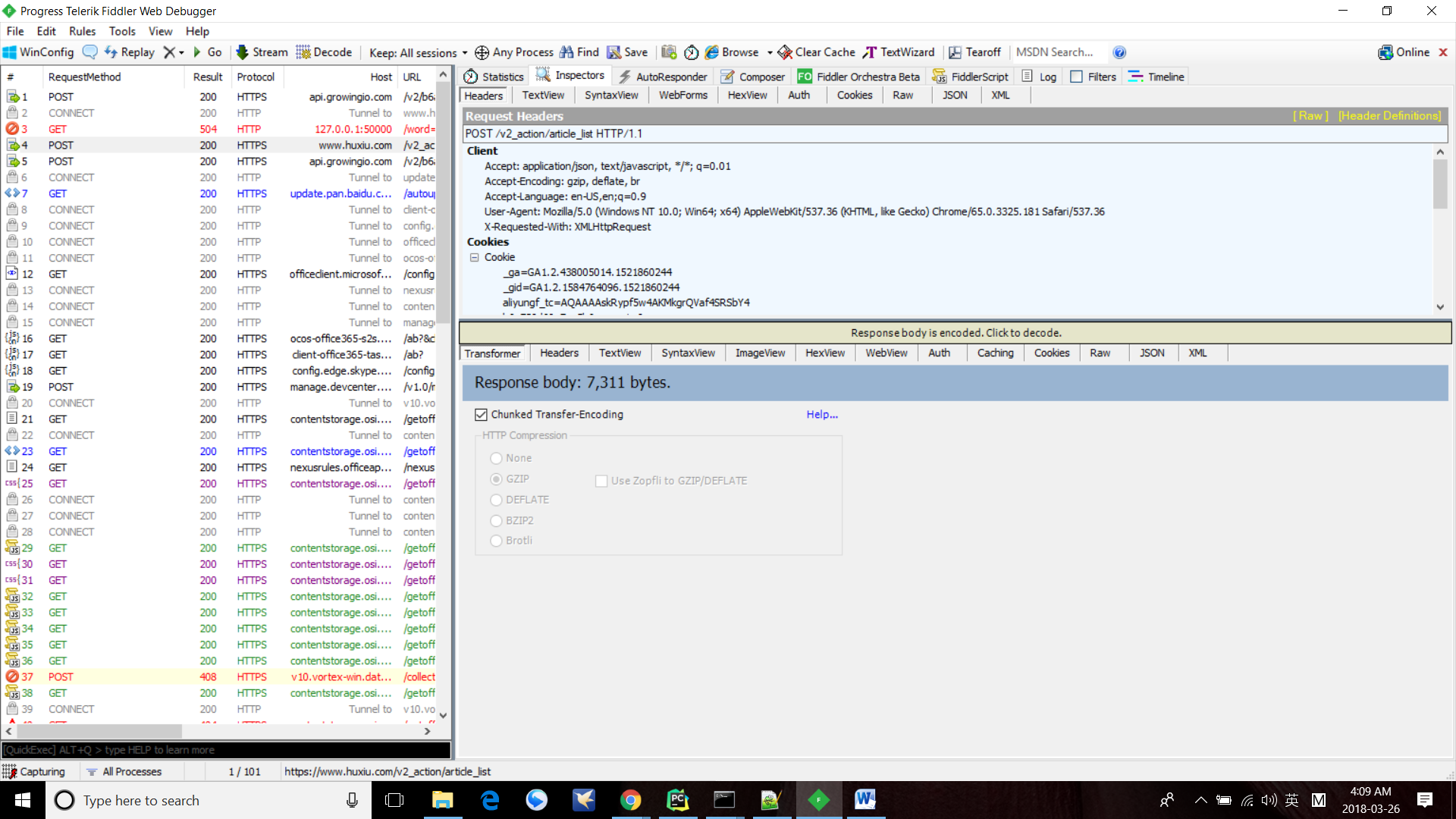
'huxiu\_hash\_code': '7e60f03b00514b92a7ab248523bb7e75', #这是虎嗅网络的HASH值恒定不变

'page':'2', #用户提交的需要访问页码号 （注意提交时候要设为字符串类型非INT）

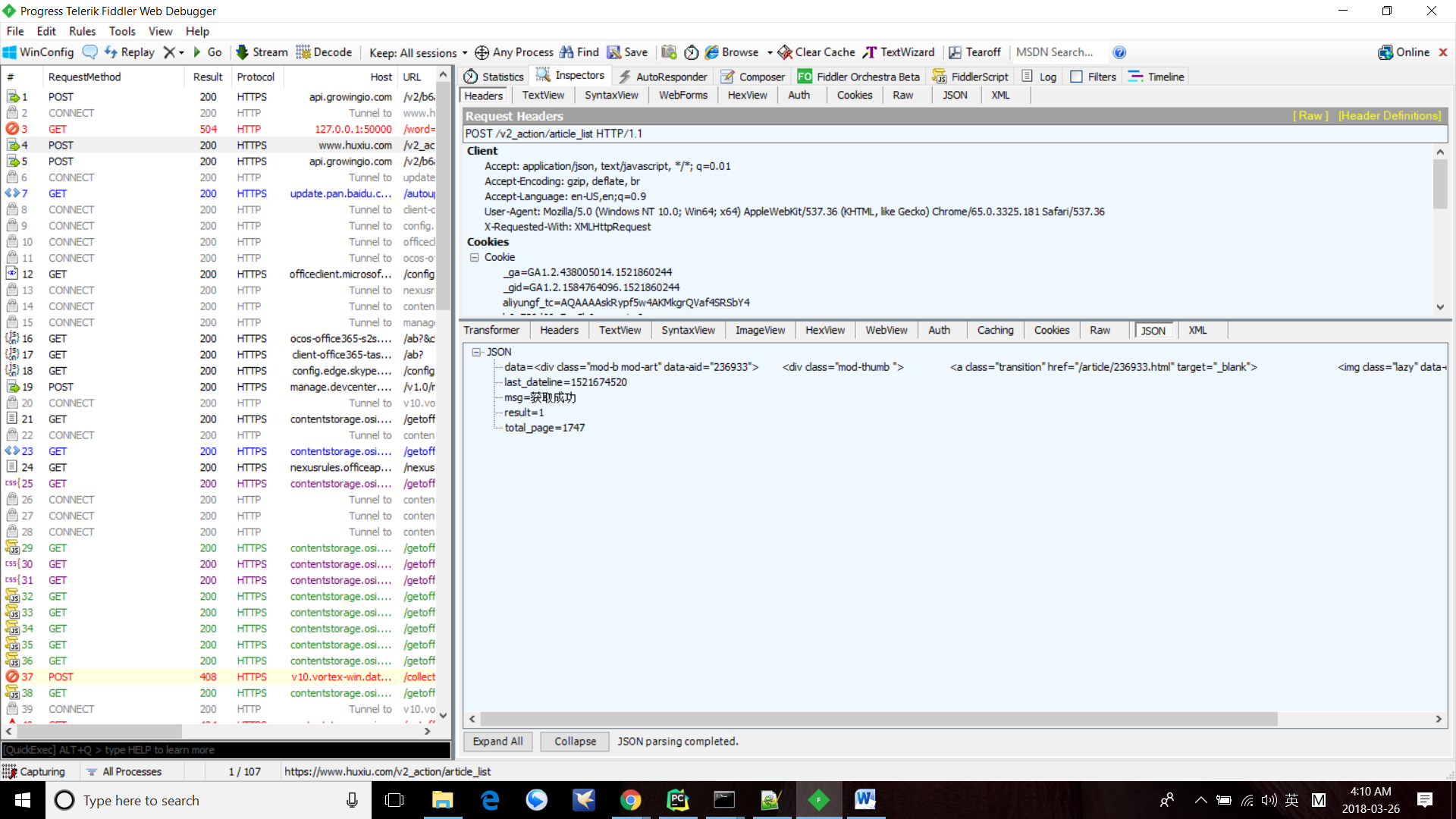
'last\_dateline':TIME\_STAMP, #后发现其是当前页面最后一篇文章的时间戳，通过RE内容可查

}

我们继续对PAGE2之后的页面进行抓包提取，这时利用的工具就是Fiddler,个人发现它在web访问数据抓取和分析非常方便。废话少说，我发现从第二页开始返回的数据是非标准的HTML,后发现其实质是json结构的数据，因此通过Fiddler decode返回的Response Body数据包，如截屏



我们在通过JSON格式查看response body的内容



娘的!一目了然，我们看到response body返回的json 的字典结构有包括我们需要的总页码数，当前最后一片文章的时间戳，哈哈！利用这个我们就可以构造下一个页面POST请求的数据了！当然我们可以看到key=”data”对应的value就是我们真正追加到网页的HTML,也就是我们需要抓取的数据了。从此虎嗅的网页架构被我们一一分析了，后面就是常规操作了，利用xpath和re查找相关的数据。

**常规步骤1：**

定义需要抓取的数据

此项目：

* 文章的标题
* 文章的作者
* 文章的详细内容链接

**步骤2：**

用于分析爬取内容的xpath和re

对于第一页我们可以采用path

TITLE: response.xpath(**'//a[@class="transition msubstr-row2"]/text()'**).extract()

AUTHOR: response.xpath('//span[@class="author-name "]/text()').extract() #注意此处多一个空格 尼玛坑爹

LINK: response.xpath('//a[@class="transition msubstr-row2"]/@href').extract()

用于POST的时间戳：TIME\_STAMP=response.xpath('//div[@class="get-mod-more js-get-mod-more-list transition"]/@data-last\_dateline').extract()[0]

注意此处yield的调用是scrapy.FormRequest麻痹的网上好多文章写的调用方式均不对

* yield scrapy.FormRequest(url='https://www.huxiu.com/v2\_action/article\_list',

formdata=form\_data, callback=self.after\_post)

对于第二页我们只能采用re(因为编码的原因)

1. 先定义pattern:

patern\_title=**'class="transition msubstr-row2" target="\_blank">(.\*)<'**patern\_author=**'<span class="author-name">(.\*)<'**patern\_link=**'<a href="(.\*)" class="transition msubstr-row2" target="\_blank">'**

1. 对response进行处理

有用的response属性：response.status 返回服务器的返回状态码，通过其可以查看POST访问是否成功200；response.body为服务器返回的实际数据，我发现response.body返回的是‘bytes’类型，通过查找python3中bytes和string的转换方式：

Bytes 🡺String .decode(‘utf-8’)

String 🡺Bytes .encode(‘utf-8’)

因此我们需要将response.body进行decode转为string类型；接着就是将其从string转为dict

此时[data]为返回的HTML数据（有怪异的编码格式还需小心）;[last\_dateline]为最后文章的时间戳；[total\_page]一共的页码数

我们的工作当然主要是针对[data]

具体步骤如下

DATA= data\_json['data'] #读取HTML数据  
TIME\_STAMP=data\_json['last\_dateline'] #读取时间戳  
TOTAL\_PAGE\_NUMBER=int(data\_json['total\_page']) #总页码数  
form\_data = {'huxiu\_hash\_code': '7e60f03b00514b92a7ab248523bb7e75',

'page':str(page),

'last\_dateline':TIME\_STAMP,

}  
生成Title,Author,Link列表  
title\_list=re.compile(patern\_title).findall(DATA)  
author\_list=re.compile(patern\_author).findall(DATA)  
link\_list=re.compile(patern\_link).findall(DATA)

循环列表，对item进行填充

page=page+1  
if page>TOTAL\_PAGE\_NUMBER:  
 return #全部爬取完成，推出  
yield scrapy.FormRequest(url="https://www.huxiu.com/v2\_action/article\_list", formdata=form\_data,

callback= self.after\_post) #再回调自身进行递归