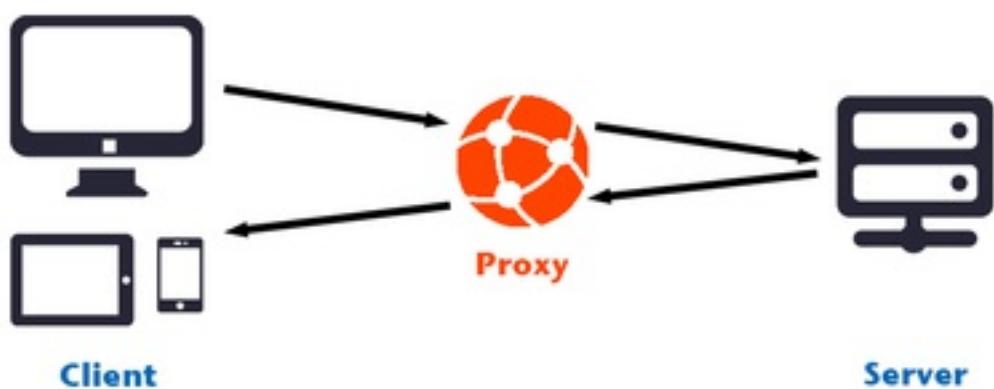


使用 Fiddler 抓包分析公众号请求过程

上一节我们熟悉了 Requests 基本使用方法，配合 Chrome 浏览器实现了一个简单爬虫，但因为微信公众号的封闭性，微信公众平台并没有对外提供 Web 端入口，只能通过手机客户端接收、查看公众号文章，所以，为了窥探到公众号背后的网络请求，我们需要借以代理工具的辅助。

HTTP代理工具又称为抓包工具，主流的抓包工具 Windows 平台有 Fiddler，macOS 有 Charles，阿里开源了一款工具叫 AnyProxy。它们的基本原理都是类似的，就是通过在手机客户端设置好代理IP和端口，客户端所有的 HTTP、HTTPS 请求就会经过代理工具，在代理工具中就可以清晰地看到每个请求的细节，然后可以分析出每个请求是如何构造的，弄清楚这些之后，我们就可以用 Python 模拟发起请求，进而得到我们想要的数据。



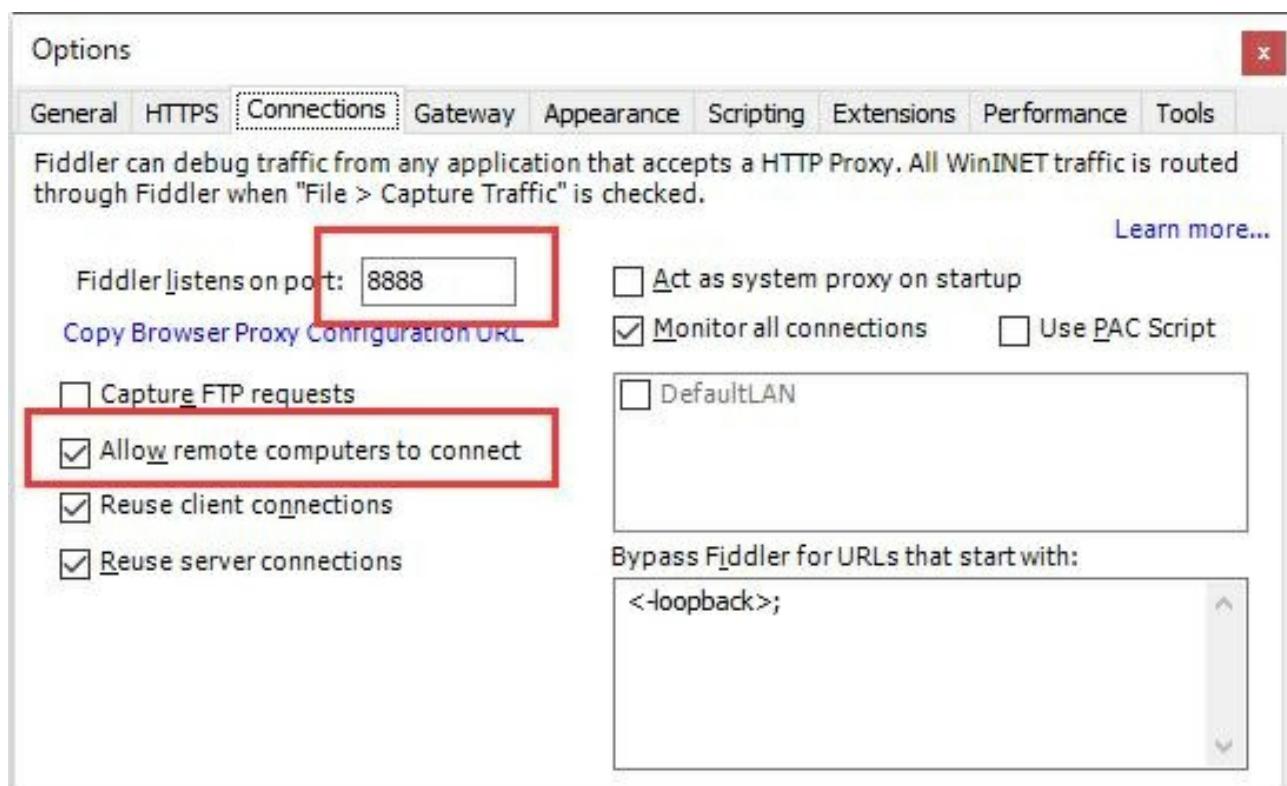
Fiddler 下载地址是 <https://www.telerik.com/download/fiddler> (<https://www.telerik.com/download/fiddler>)，安装包就 4M 多，在配置之前，首先要确保你的手机和电脑在同一个局域网，如果

不在同一个局域网，你可以买个随身 WiFi，在你电脑上搭建一个极简无线路由器。安装过程一路点击下一步完成就可以了。

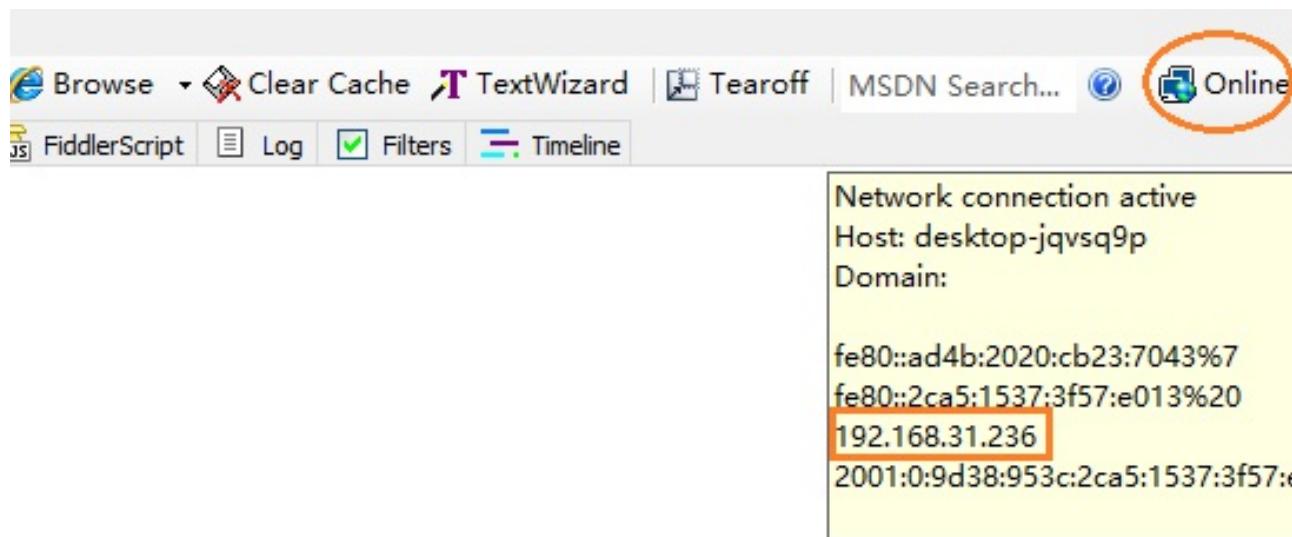
Fiddler 配置

选择 Tools > Fiddler Options > Connections

Fiddler 默认的端口是使用 8888，如果该端口已经被其它程序占用了，你需要手动更改，勾选 Allow remote computers to connect，其它的选择默认配置就好，配置更新后记得重启 Fiddler。一定要重启 Fiddler，否则代理无效。



接下来你需要配置手机，我们以 Android 设备为例，现在假设你的手机和电脑已经在同一个局域网（只要连的是同一个路由器就在同局域网内），找到电脑的 IP 地址，在 Fiddler 右上角有个 Online 图标，鼠标移过去就能看到 IP 了，你也可以在 CMD 窗口使用 ipconfig 命令查看到

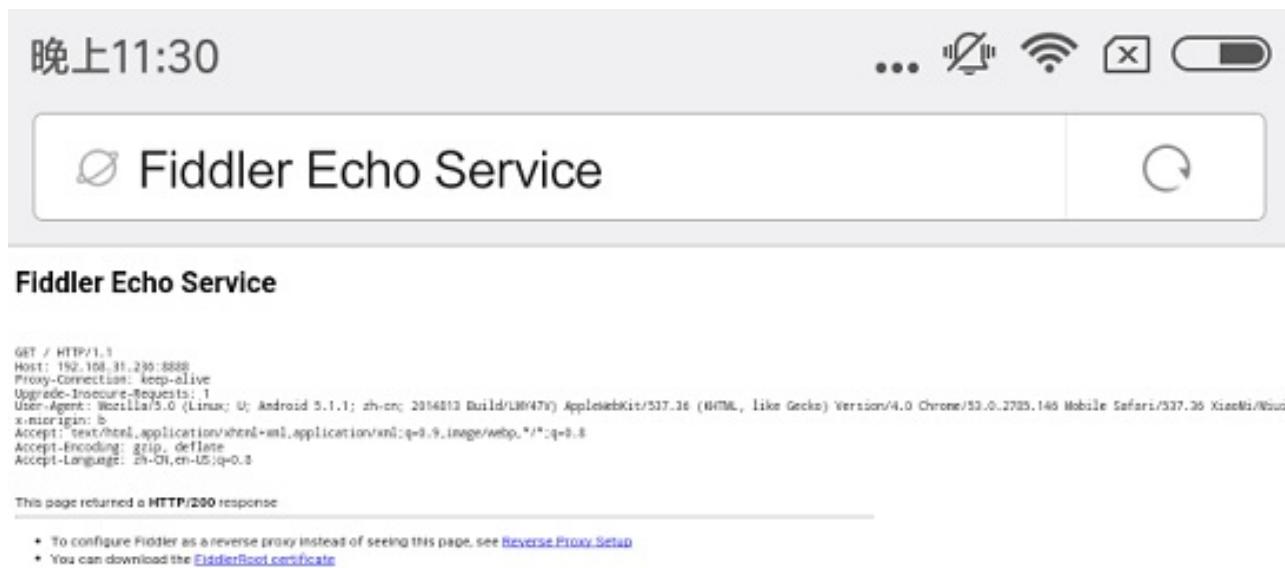


Android 手机代理配置

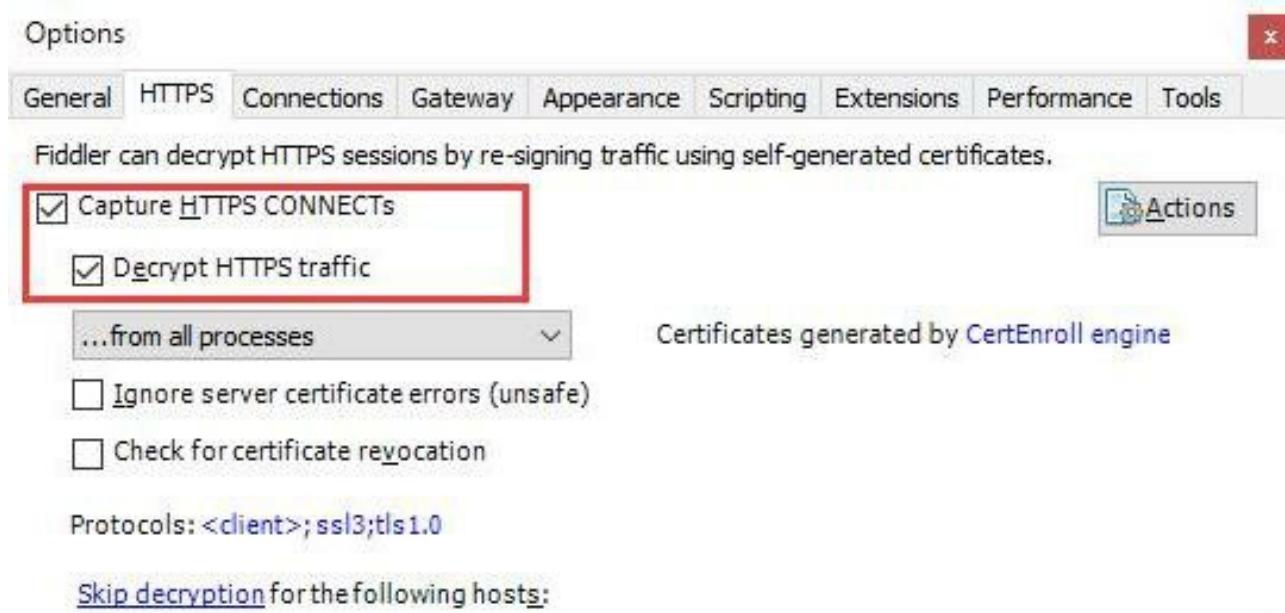
进入手机的 WLAN 设置，选择当前所在局域网的 WiFi 链接，设置代理服务器的 IP 和端口，我这是以小米设备为例，其它 Android 手机的配置过程大同小异。



测试代理有没有设置成功可以在手机浏览器访问你配置的地址：
<http://192.168.31.236:8888/> 会显示 Fiddler 的回显页面，说明配置成功。



现在你打开任意一个HTTP协议的网站都能看到请求会出现在 Fiddler 窗口，但是 HTTPS 的请求并没有出现在 Fiddler 中，其实还差一个步骤，需要在 Fiddler 中激活 HTTPS 抓取设置。在 Fiddler 选择 **Tools > Fiddler Options > HTTPS > Decrypt HTTPS traffic**，重启 Fiddler。



为了能够让 Fiddler 截取 HTTPS 请求，客户端都需要安装且信任 Fiddler 生成的 CA 证书，否则会出现“网络出错，轻触屏幕重新加载:-1200”的错误。在浏览器打开 Fiddler 回显页面 <http://192.168.31.236:8888/> 下载 **FiddlerRoot certificate**，下载并安装证书，并验证通过。



iOS下载安装完成之后还要从 **设置->通用->关于本机->证书信任设置** 中把 Fiddler 证书的开关打开



Android 手机下载保存证书后从系统设置里面找到系统安全，从SD卡安装证书，如果没有安装证书，打开微信公众号的时候会弹出警告。

有权查看使用情况的应用



凭据存储

存储类型

硬件支持

信任的凭据



从SD卡安装

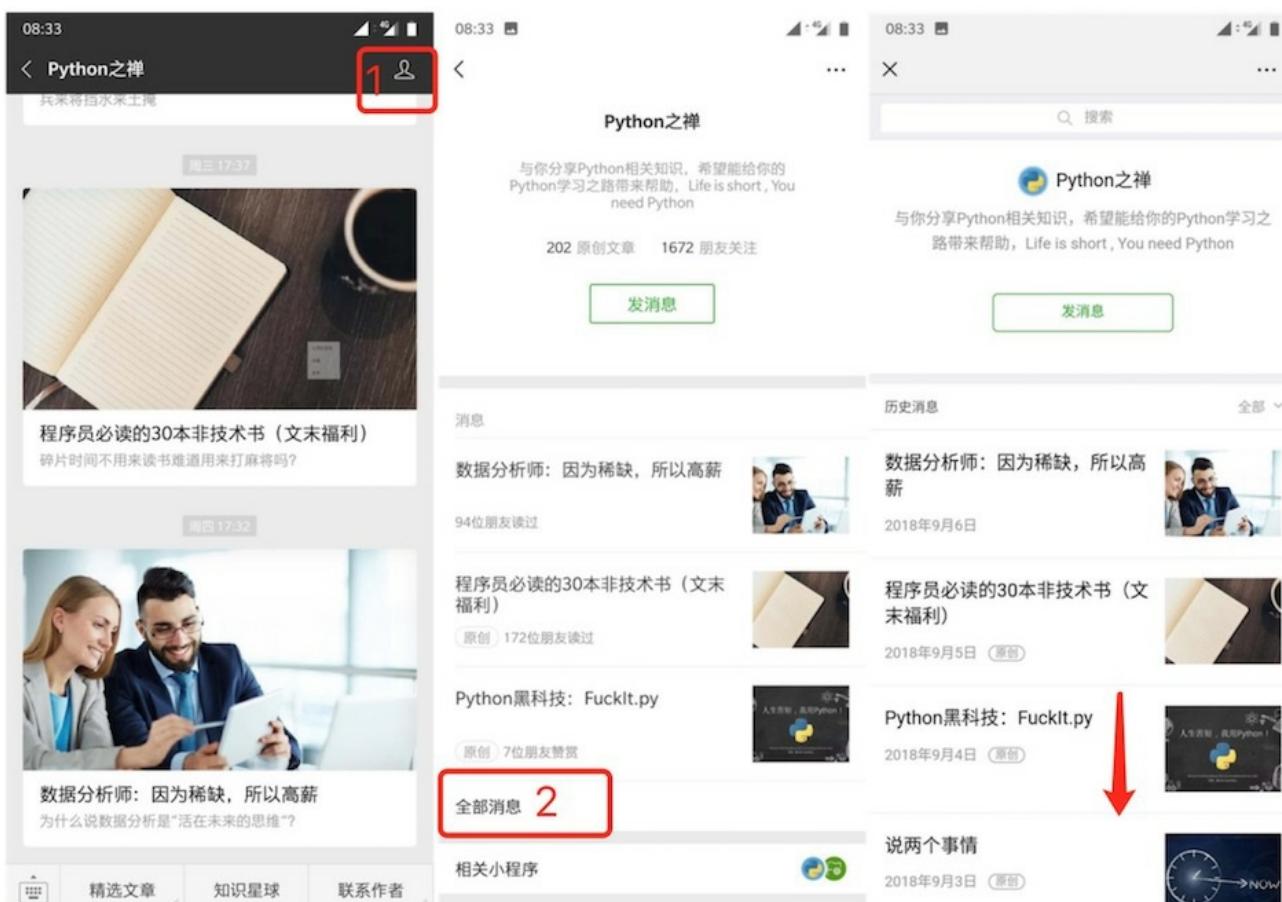
从SD卡安装证书



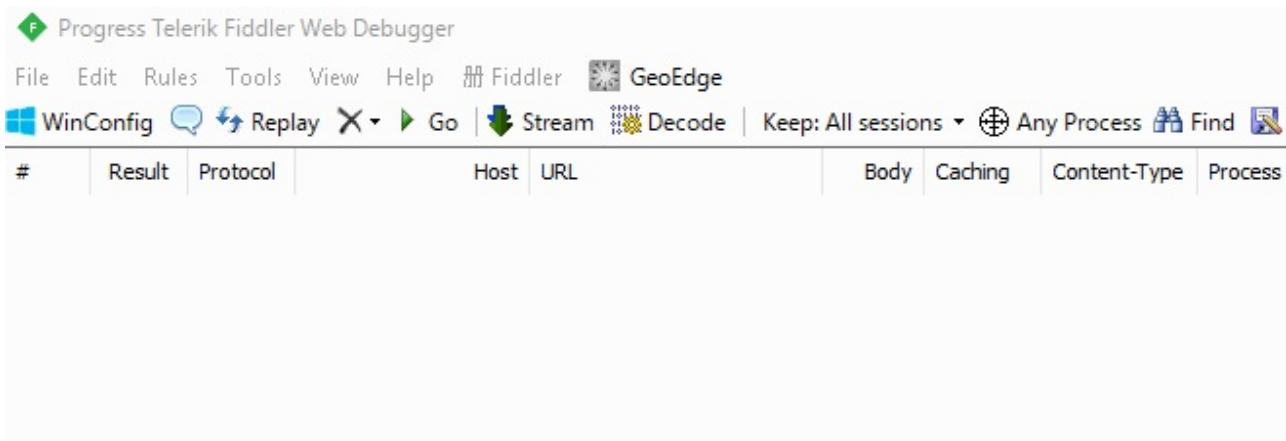
至此，所有的配置都完成了，现在打开微信随便选择一个公众号，查看公众号的所有历史文章列表。微信在2018年6月份对 iOS 版本的微信以及部分 Android 版微信针对公众号进行了大幅调整，改为现在的信息流方式，现在要获取某个公众号下面「所有文章列表」大概需要经过以下四个步骤：



如果你的微信版本还不是信息流方式展示的，那么应该是Android版本（微信采用的ABTest，不同的用户呈现的方式不一样）



同时观察 Fiddler 主面板



进入「全部消息」页面时，在 Fiddler 上已经能看到有请求进来了，说明公众号的文章走的都是HTTP协议，这些请求就是微信客户端向微信服务器发送的HTTP请求。

注意：第一次请求「全部消息」的时候你看到的可能是一片空白：



在Fiddler或Charles中看到的请求数据是这样的：

200 GET mp.weixin.qq.com /mp/profile_ext?action=home&_biz=MjM5ODQ2MDIyMA==&scene=126&devicetype=iOS11.4.1&ve...

Filter: mp.wei

Overview Request Response Summary Chart Notes

```

1 <!DOCTYPE HTML>
2 <html>
3     <meta charset="utf-8">
4     <head>
5         <title>验证</title> ←
6         <meta charset="utf-8">
7         <meta id="viewport" name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,user-scalable
8             </head>
9             <style>
10                body{
11                    margin:0;
12                    padding:10px;
13                    background-color:#E1E0DE;
14                    font:12px/18px "Lucida Grande", "Lucida Sans Unicode", Helvetica, Arial, Verdana, sans-serif;

```

Headers Text Compressed HTML Raw

这个时候你要直接从左上角叉掉重新进入「全部消息」页面。

现在简单介绍一下这个请求面板上的每个模块的意义。

The screenshot shows the Fiddler application interface. The left pane lists network requests with columns for #, Result, Protocol, Host, and URL. The right pane provides detailed information for a selected request, including:

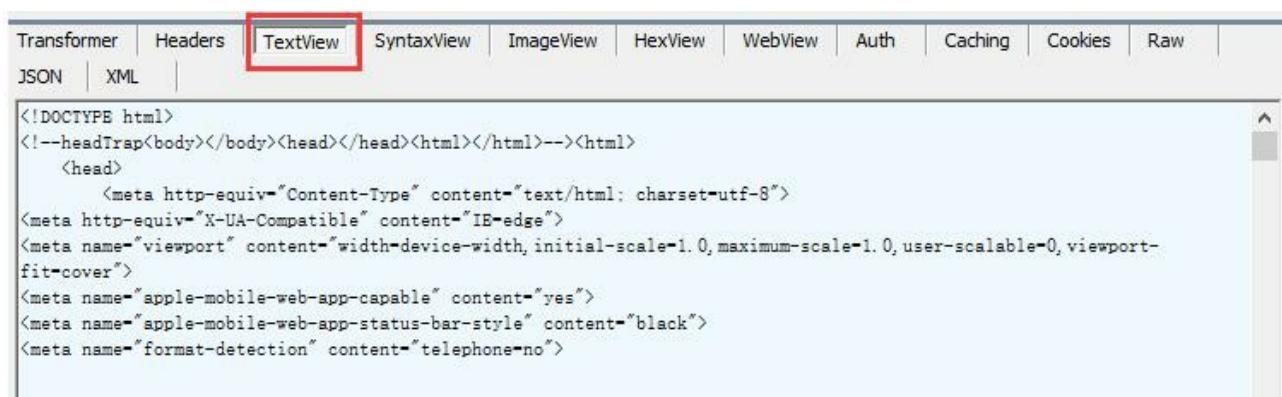
- Request Headers:** Shows the raw HTTP headers for the selected request, including the URL and various header fields like Content-Type, Accept, and User-Agent.
- Cookies:** Shows the raw cookie data, including session identifiers and other tracking information.
- Miscellaneous:** Shows other metadata like the User-Agent string and a list of insecure requests.
- Response body:** Shows the encoded response body, which is a JSON object containing the error message: "{'status': 403, 'msg': '非法请求'}

这样说明这个请求被微信服务器判定为一次非法的请求，这时你可以叉掉该页面重新进入「全部消息」页面。不出意外的话就能正常看到全部文章列表了，同时也能在Fiddler中看到正常的数据请求了。

我把上面的主面板划分为 7 大块，你需要理解每块的内容，后面才有可能会用 Python 代码来模拟微信请求。

- 1、服务器的响应结果，200 表示服务器对该请求响应成功
- 2、请求协议，微信的请求协议都是基于HTTPS的，所以前面一定要配置好，不然你看不到 HTTPS 的请求。
- 3、微信服务器主机名
- 4、请求路径
- 5、请求行，包括了请求方法（GET），请求协议（HTTP/1.1），请求路径（/mp/profile_ext...后面还有很长一串参数）
- 6、包括Cookie信息在内的请求头。
- 7、微信服务器返回的响应数据，我们分别切换成 TextView 和 WebView 看一下返回的数据是什么样的。

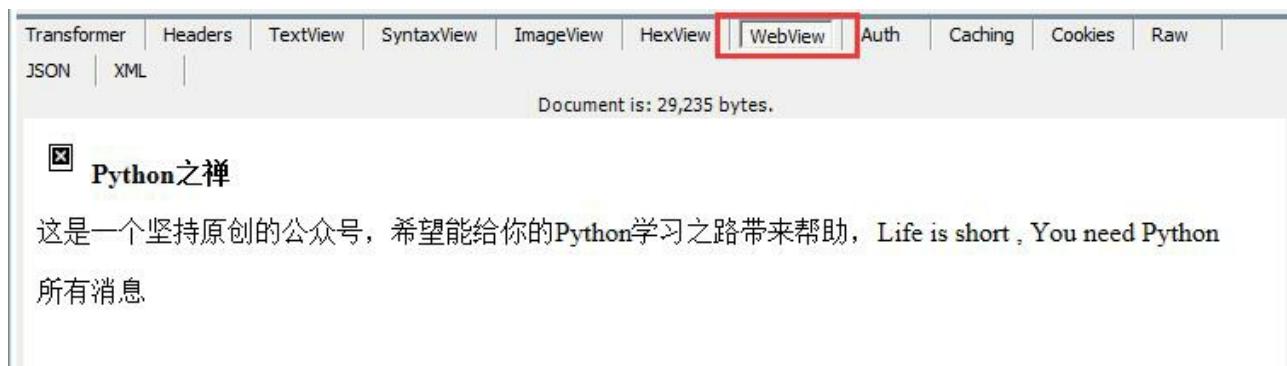
TextView 模式下的预览效果是服务器返回的 HTML 源代码



The screenshot shows a browser developer tools interface with a tab bar at the top. The 'TextView' tab is selected and highlighted with a red box. Other tabs include Transformer, Headers, SyntaxView, ImageView, HexView, WebView, Auth, Caching, Cookies, and Raw. Below the tabs, there are buttons for JSON and XML. The main content area displays the raw HTML source code of a response. The code includes standard HTML tags like <!DOCTYPE html>, <html>, <head>, and <body>, along with meta tags for content-type, charset, and viewport settings.

```
<!DOCTYPE html>
<!--headTrap--><body><head></head><html></html>--><html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE-edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0, viewport-fit=cover">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black">
    <meta name="format-detection" content="telephone=no">
```

WebView 模式是 HTML 代码经过渲染之后的效果，其实就是我们在手机微信中看到的效果，只不过因为缺乏样式，所以没有手机上看到的美化效果。



The screenshot shows a browser developer tools interface with a tab bar at the top. The 'WebView' tab is selected and highlighted with a red box. Other tabs include Transformer, Headers, TextView, SyntaxView, ImageView, HexView, Auth, Caching, Cookies, and Raw. Below the tabs, there are buttons for JSON and XML. The main content area displays the rendered HTML content. It includes a title 'Python之禅' and a paragraph of text: '这是一个坚持原创的公众号，希望能给你的Python学习之路带来帮助，Life is short, You need Python' followed by '所有消息'.

Document is: 29,235 bytes.

Python之禅

这是一个坚持原创的公众号，希望能给你的Python学习之路带来帮助，Life is short, You need Python

所有消息

如果服务器返回的是 Json 格式或者是 XML，你还可以切换到对应的页面预览查看。

小结

配置好Fiddler的几个步骤主要包括指定监控的端口，开通HTTPS流量解密功能，同时，客户端需要安装CA证书。下一节我们基于Requests模拟像微信服务器发起请求。