**[1. Fiddler 抓包简介](http://www.cnblogs.com/yyhh/p/5140852.html" \l "l01" \t "_blank)**

[**1）. 字段说明**](http://www.cnblogs.com/yyhh/p/5140852.html#l11)

[**2）. Statistics 请求的性能数据分析**](http://www.cnblogs.com/yyhh/p/5140852.html#l12)

[**3）. Inspectors 查看数据内容**](http://www.cnblogs.com/yyhh/p/5140852.html#l13)

[**4）. AutoResponder 允许拦截制定规则的请求**](http://www.cnblogs.com/yyhh/p/5140852.html#l14)

[**5）. Filters 请求过滤规则**](http://www.cnblogs.com/yyhh/p/5140852.html#l15)

[**6）. Timeline 请求响应时间**](http://www.cnblogs.com/yyhh/p/5140852.html#l16)

**[2. Fiddler 设置解密HTTPS的网络数据](http://www.cnblogs.com/yyhh/p/5140852.html" \l "l02" \t "_blank)**

**[3. Fiddler 抓取Iphone / Android数据包](http://www.cnblogs.com/yyhh/p/5140852.html" \l "l03" \t "_blank)**

**[4. Fiddler 内置命令与断点](http://www.cnblogs.com/yyhh/p/5140852.html" \l "l04" \t "_blank)**

**序章**

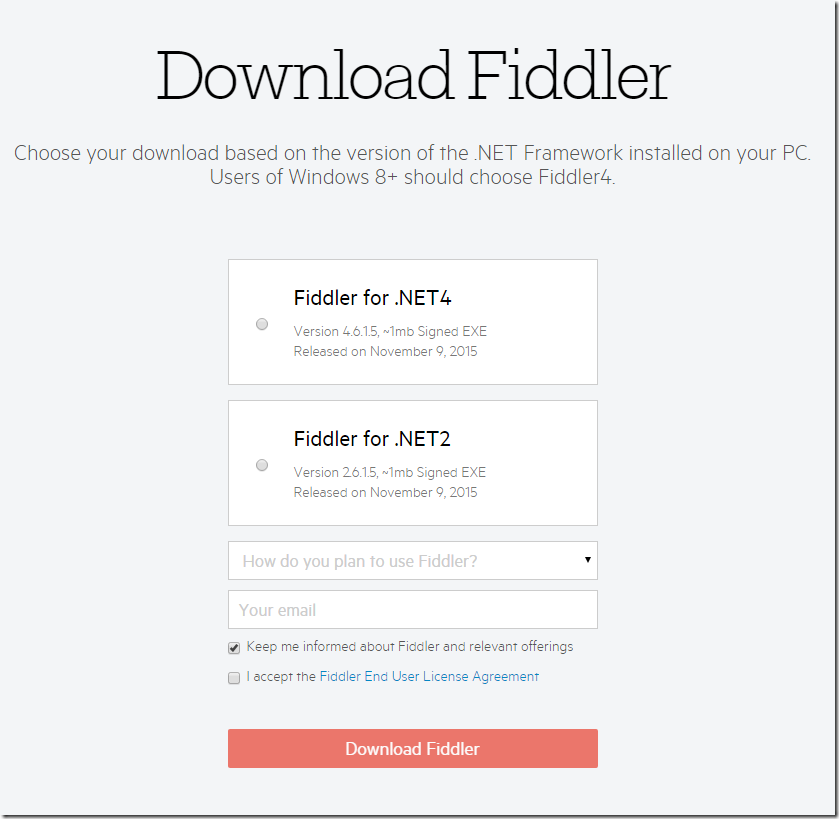
Fiddler是一个蛮好用的抓包工具，可以将网络传输发送与接受的数据包进行截获、重发、编辑、转存等操作。也可以用来检测网络安全。反正好处多多，举之不尽呀！当年学习的时候也蛮费劲，一些蛮实用隐藏的小功能用了之后就忘记了，每次去网站上找也很麻烦，所以搜集各大网络的资料，总结了一些常用的功能。

Fiddler 下载地址 ：<https://www.telerik.com/download/fiddler>

Fiddler 离线下载地址：[http://pan.baidu.com/s/1i3NvE8P](http://pan.baidu.com/s/1i3NvE8P" \t "_blank) 密码：ozem

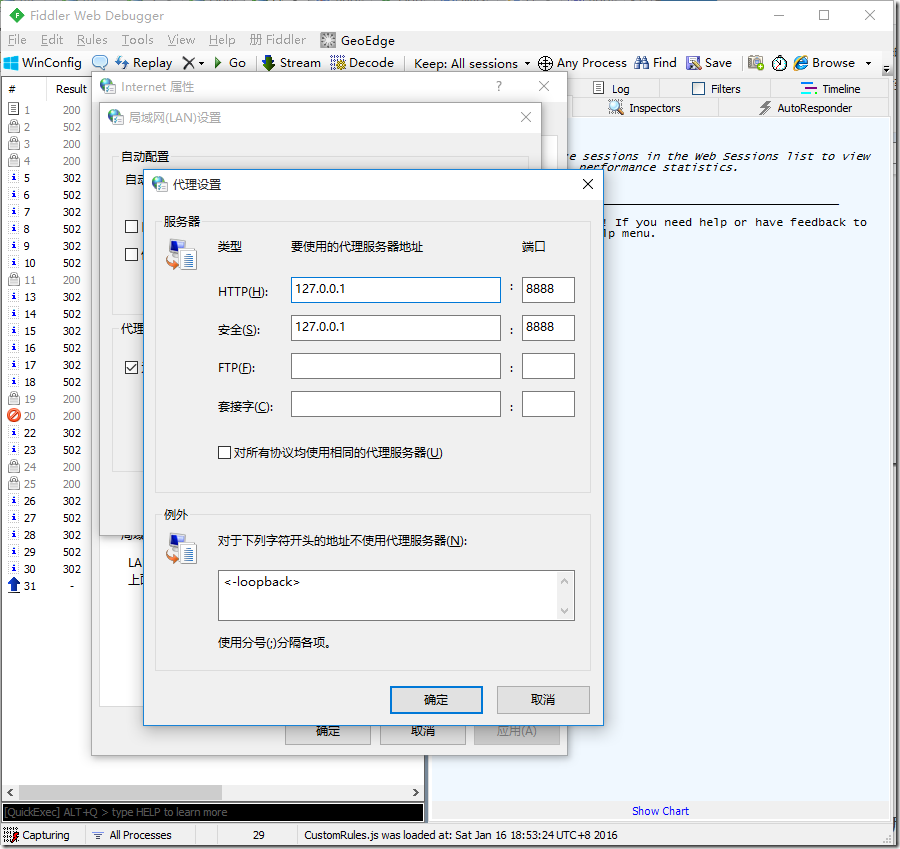
下载Fiddler要FQ，我费了好大得劲才翻出去下载到…

win8之后用“Fiddler for .NET4”而win8之前用“Fiidler for .NET2”比较好

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234151922-2118606677.png)

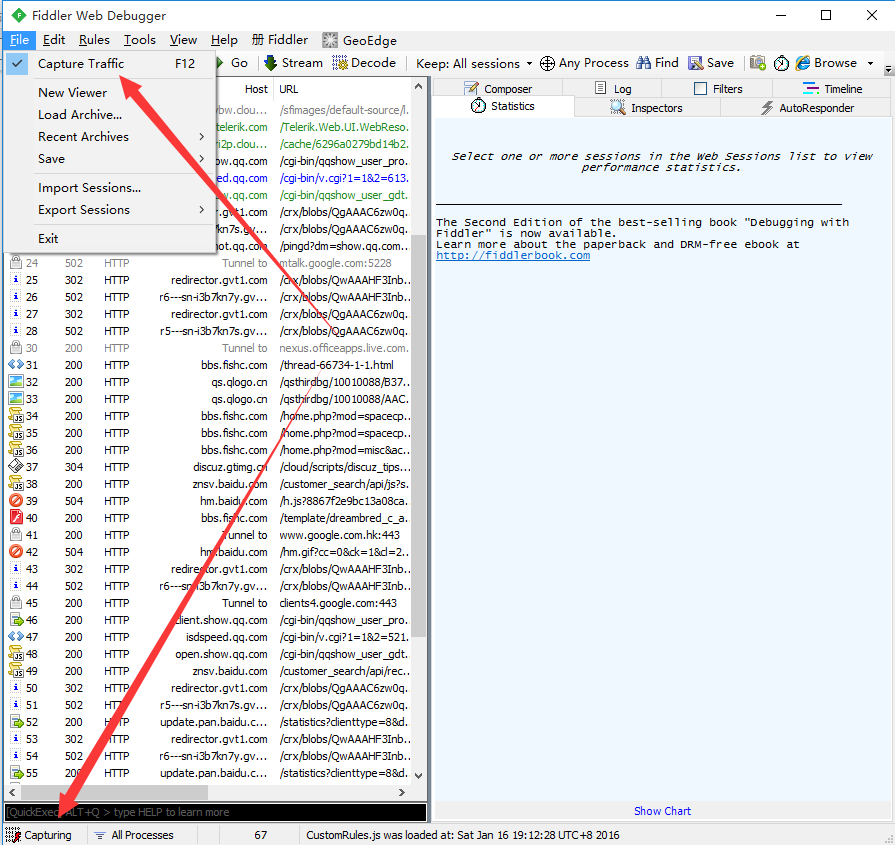
**1. Fiddler 抓包简介**

Fiddler是通过改写HTTP代理，让数据从它那通过，来监控并且截取到数据。当然Fiddler很屌，在打开它的那一瞬间，它就已经设置好了浏览器的代理了。当你关闭的时候，它又帮你把代理还原了，是不是很贴心。。。

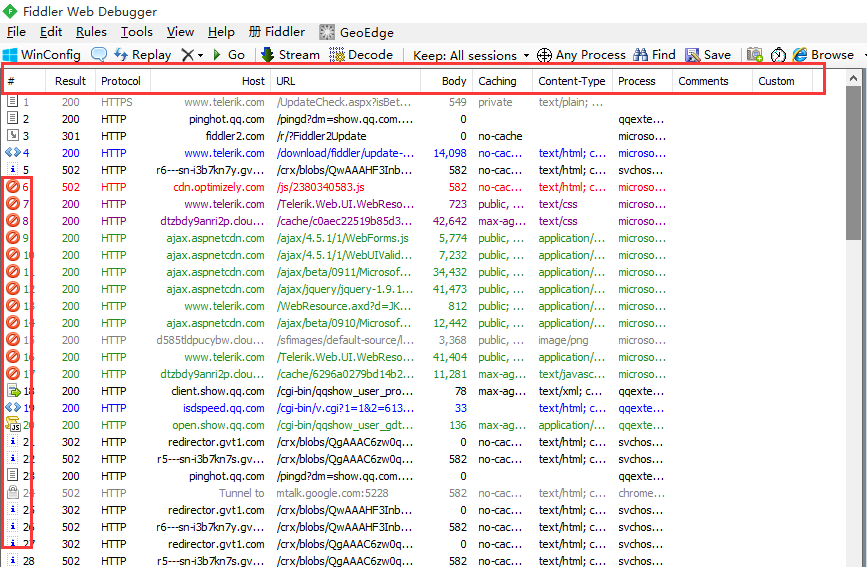
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234154109-604254637.png)

**1） 字段说明**

Fiddler想要抓到数据包，要确保Capture Traffic是开启，在File –> Capture Traffic。开启后再左下角会有显示，当然也可以直接点击左下角的图标来关闭/开启抓包功能。

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234156422-111039030.png)

Fiddler开始工作了，抓到的数据包就会显示在列表里面，下面总结了这些都是什么意思：

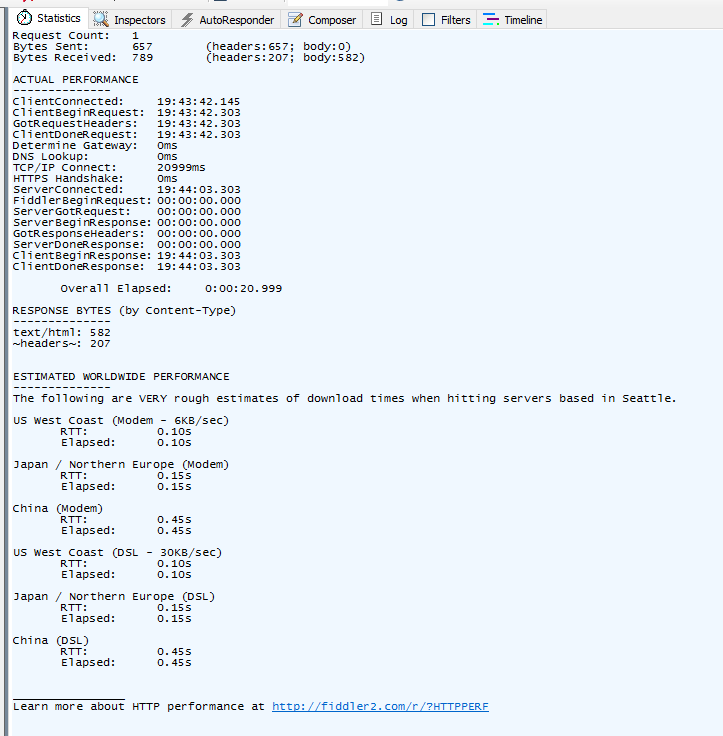
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234157922-1352740469.png)

|  |  |
| --- | --- |
| **名称** | **含义** |
| # | 抓取HTTP Request的顺序，从1开始，以此递增 |
| Result | HTTP状态码 |
| Protocol | 请求使用的协议，如HTTP/HTTPS/FTP等 |
| Host | 请求地址的主机名 |
| URL | 请求资源的位置 |
| Body | 该请求的大小 |
| Caching | 请求的缓存过期时间或者缓存控制值 |
| Content-Type | 请求响应的类型 |
| Process | 发送此请求的进程：进程ID |
| Comments | 允许用户为此回话添加备注 |
| Custom | 允许用户设置自定义值 |
| 图标 | 含义 |
| [clip_image001[13]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234159140-1640771885.gif) | 请求已经发往服务器 |
| [clip_image002[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234159765-1592382397.gif) | 已从服务器下载响应结果 |
| [clip_image003[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234201172-824376071.gif) | 请求从断点处暂停 |
| [clip_image004[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234201843-109931155.gif) | 响应从断点处暂停 |
| [clip_image005[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234202609-1251165508.gif) | 请求使用 HTTP 的 HEAD 方法，即响应没有内容（Body） |
| [clip_image006[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234203187-464520892.png) | 请求使用 HTTP 的 POST 方法 |
| [clip_image007[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234204156-849270471.gif) | 请求使用 HTTP 的 CONNECT 方法，使用 HTTPS 协议建立连接隧道 |
| [clip_image008[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234205078-161726095.gif) | 响应是 HTML 格式 |
| [clip_image009[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234205906-1953449875.gif) | 响应是一张图片 |
| [clip_image010[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234206453-344992441.gif) | 响应是脚本格式 |
| [clip_image011[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234207218-1611954438.gif) | 响应是 CSS 格式 |
| [clip_image012[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234208015-685482882.gif) | 响应是 XML 格式 |
| [clip_image013[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234209140-529820079.png) | 响应是 JSON 格式 |
| [clip_image014[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234209937-1106065906.png) | 响应是一个音频文件 |
| [clip_image015[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234210453-1729183371.png) | 响应是一个视频文件 |
| [clip_image016[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234210906-1149932156.png) | 响应是一个 SilverLight |
| [clip_image017[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234211547-791281288.png) | 响应是一个 FLASH |
| [clip_image018[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234213922-1389767841.png) | 响应是一个字体 |
| [clip_image019[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234214422-971819227.gif) | 普通响应成功 |
| [clip_image020[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234215125-2125019877.gif) | 响应是 HTTP/300、301、302、303 或 307 重定向 |
| [clip_image021[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234215718-1165848356.gif) | 响应是 HTTP/304（无变更）：使用缓存文件 |
| [clip_image022[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234216265-1959007927.gif) | 响应需要客户端证书验证 |
| [clip_image023[4]](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234216781-677826292.gif) | 服务端错误 |
| [clip_image0244](http://images2015.cnblogs.com/blog/626593/201601/626593-20160119000323687-304487615.gif) | 会话被客户端、Fiddler 或者服务端终止 |

**2）. Statistics 请求的性能数据分析**

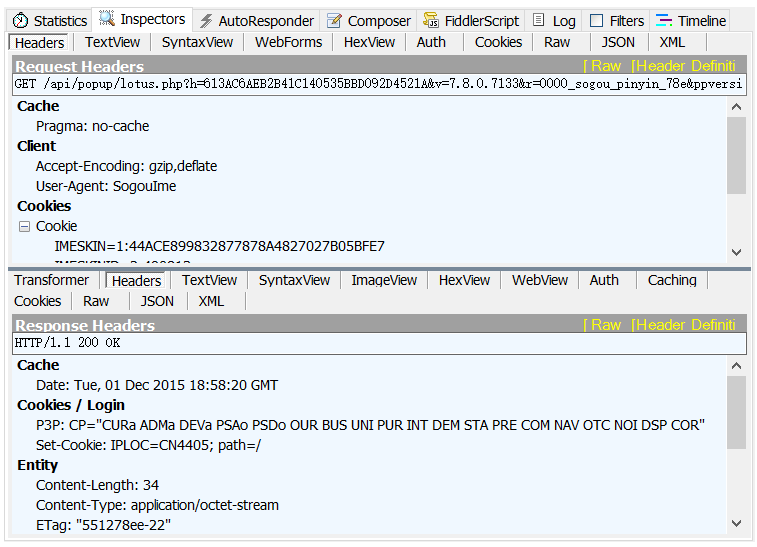
好了。左边看完了，现在可以看右边了

随意点击一个请求，就可以看到Statistics关于HTTP请求的性能以及数据分析了（不可能安装好了Fiddler一条请求都没有…）：

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234218234-2029245226.png)

**3）. Inspectors 查看数据内容**

Inspectors是用于查看会话的内容，上半部分是请求的内容，下半部分是响应的内容：

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120130545281-931107154.png)

**4）. AutoResponder 允许拦截指定规则的请求**

AutoResponder允许你拦截指定规则的求情，并返回本地资源或Fiddler资源，从而代替服务器响应。

看下图5步，我将“baidu”这个关键字与我电脑“f:\Users\YukiO\Pictures\boy.jpeg”这张图片绑定了，点击Save保存后勾选Enable rules，再访问baidu，就会被劫持。

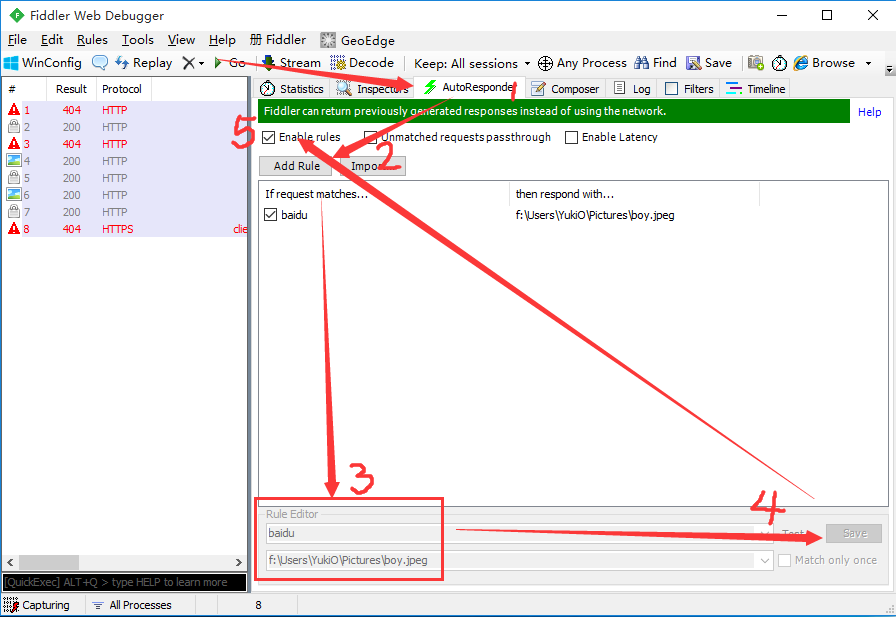
这个玩意有很多匹配规则，如：

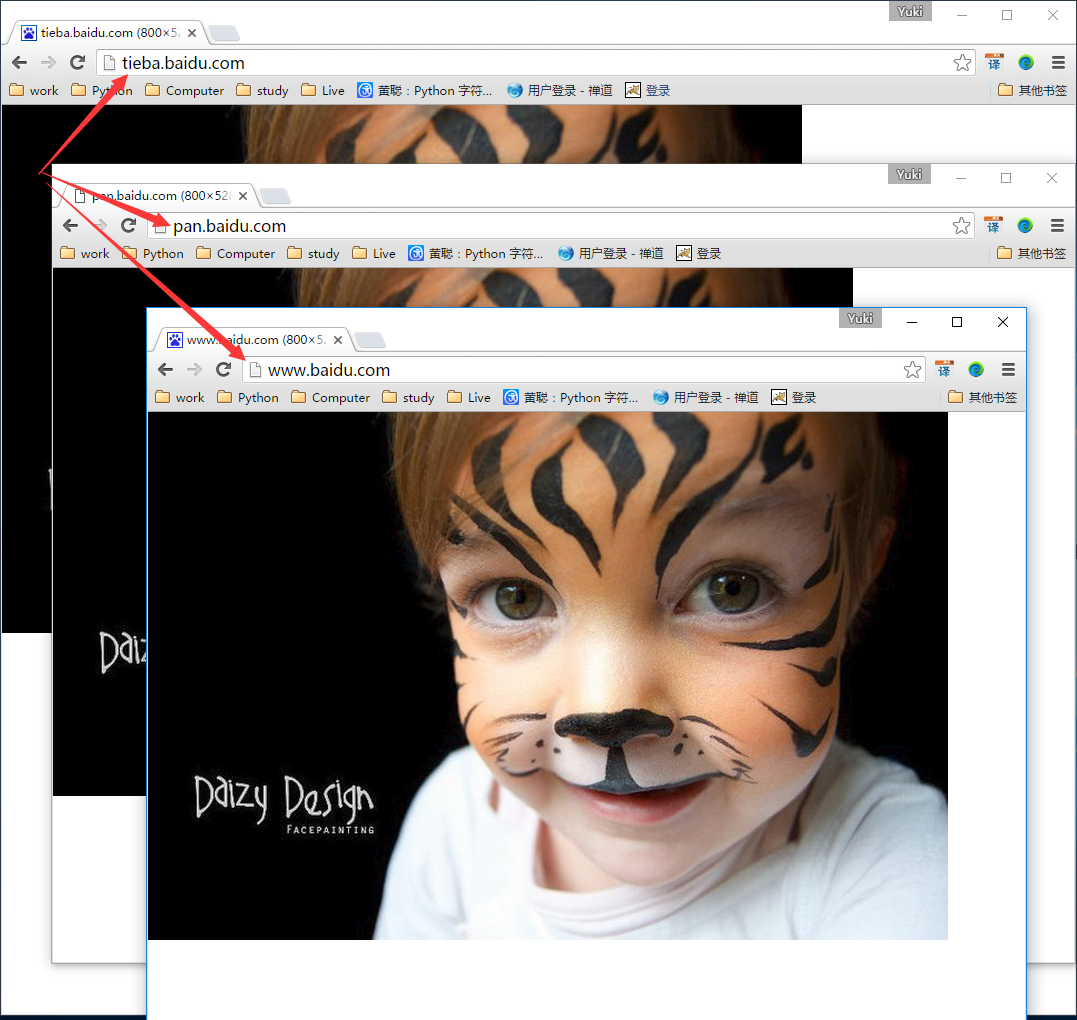
1. 字符串匹配（默认）：只要包含指定字符串（不区分大小写），全部认为是匹配

|  |  |
| --- | --- |
| 字符串匹配（baidu） | 是否匹配 |
| [http://www.baidu.com](http://www.baidu.com/) | 匹配 |
| [http://pan.baidu.com](http://pan.baidu.com/) | 匹配 |
| [http://tieba.baidu.com](http://tieba.baidu.com/) | 匹配 |

2. 正则表达式匹配：以“regex:”开头，使用正则表达式来匹配，这个是区分大小写的

|  |  |
| --- | --- |
| 字符串匹配（regex:.+.(jpg | gif | bmp ) $） | 是否匹配 |
| [http://bbs.fishc.com/Path1/query=foo.bmp&bar](http://bbs.fishc.com/Path1/query=example.bmp) | 不匹配 |
| <http://bbs.fishc.com/Path1/query=example.gif> | 匹配 |
| <http://bbs.fishc.com/Path1/query=example.bmp> | 匹配 |
| <http://bbs.fishc.com/Path1/query=example.Gif> | 不匹配 |

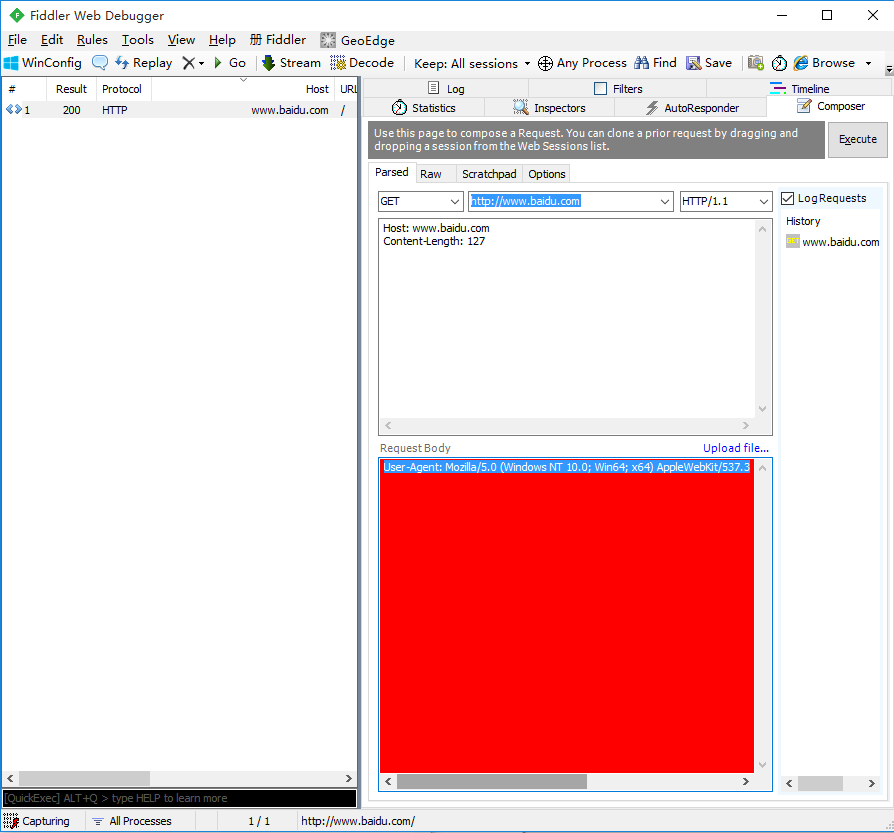
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234219297-1305583867.png)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234221609-540278459.png)

**4）. Composer 自定义请求发送服务器**

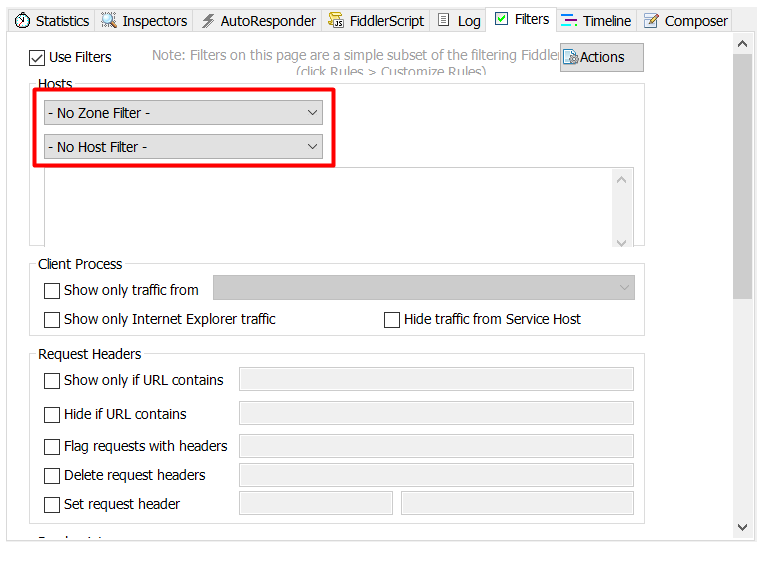
Composer允许自定义请求发送到服务器，可以手动创建一个新的请求，也可以在会话表中，拖拽一个现有的请求

Parsed模式下你只需要提供简单的URLS地址即可（如下图，也可以在RequestBody定制一些属性，如模拟浏览器User-Agent）

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234226453-853246291.png)

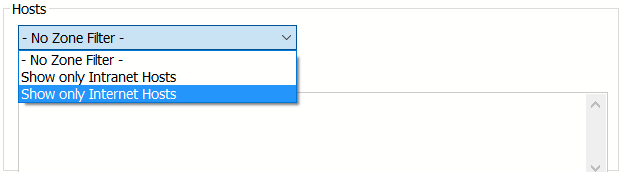
**5）. Filters 请求过滤规则**

Fiters 是过滤请求用的，左边的窗口不断的更新，当你想看你系统的请求的时候，你刷新一下浏览器，一大片不知道哪来请求，看着碍眼，它还一直刷新你的屏幕。这个时候通过过滤规则来过滤掉那些不想看到的请求。

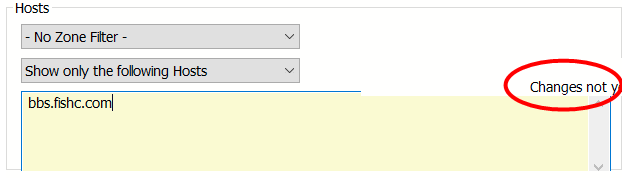
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125401593-2056873952.png)

勾选左上角的Use Filters开启过滤器，这里有两个最常用的过滤条件：Zone和Host

1、Zone 指定只显示内网（Intranet）或互联网（Internet）的内容：

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118235316328-598014481.png)

2、Host 指定显示某个域名下的会话：

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118235317031-655277228.png)

如果框框为黄色（如图），表示修改未生效，点击红圈里的文字即可

**6）. Timeline 请求响应时间**

在左侧会话窗口点击一个或多个（同时按下 Ctrl 键），Timeline 便会显示指定内容从服务端传输到客户端的时间：

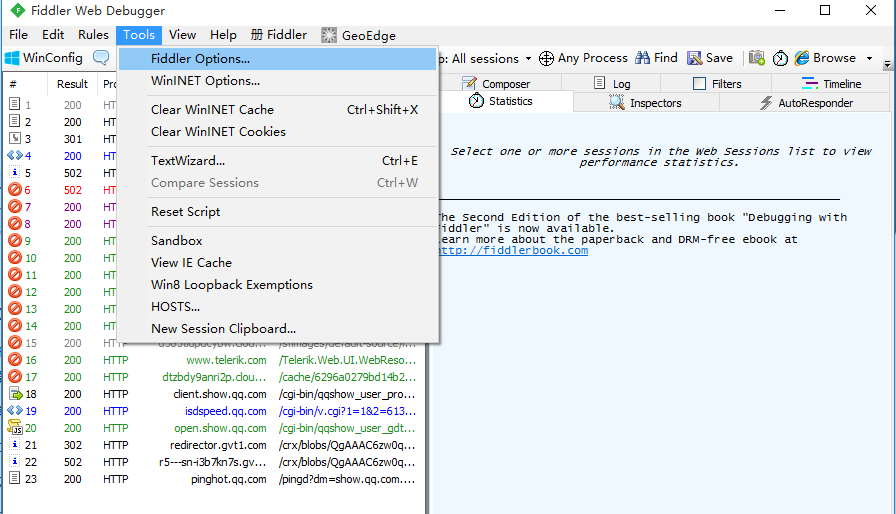
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118235317750-806540578.png)

**2. Fiddler 设置解密HTTPS的网络数据**

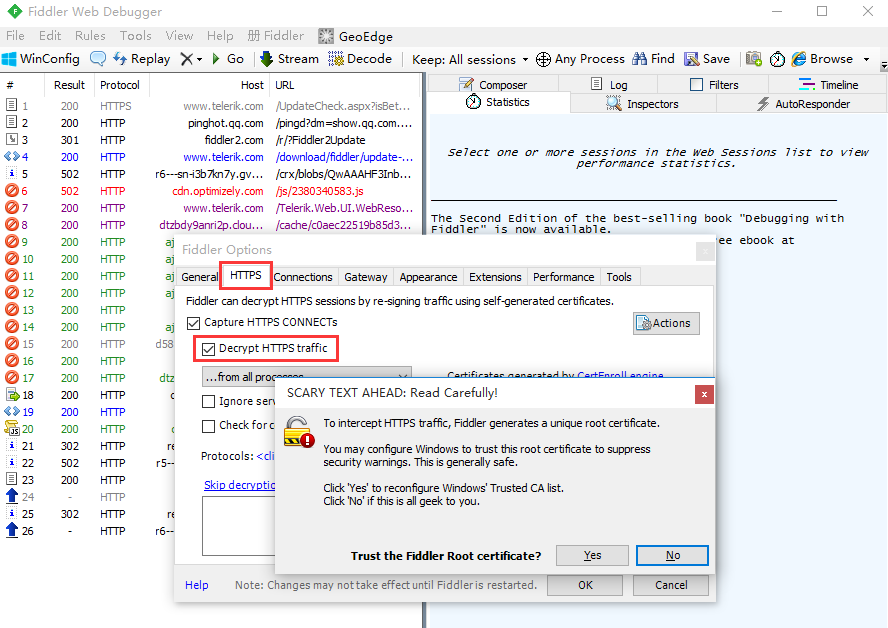
Fiddler可以通过伪造CA证书来欺骗浏览器和服务器。Fiddler是个很会装逼的好东西，大概原理就是在浏览器面前Fiddler伪装成一个HTTPS服务器，而在真正的HTTPS服务器面前Fiddler又装成浏览器，从而实现解密HTTPS数据包的目的。

解密HTTPS需要手动开启，依次点击：

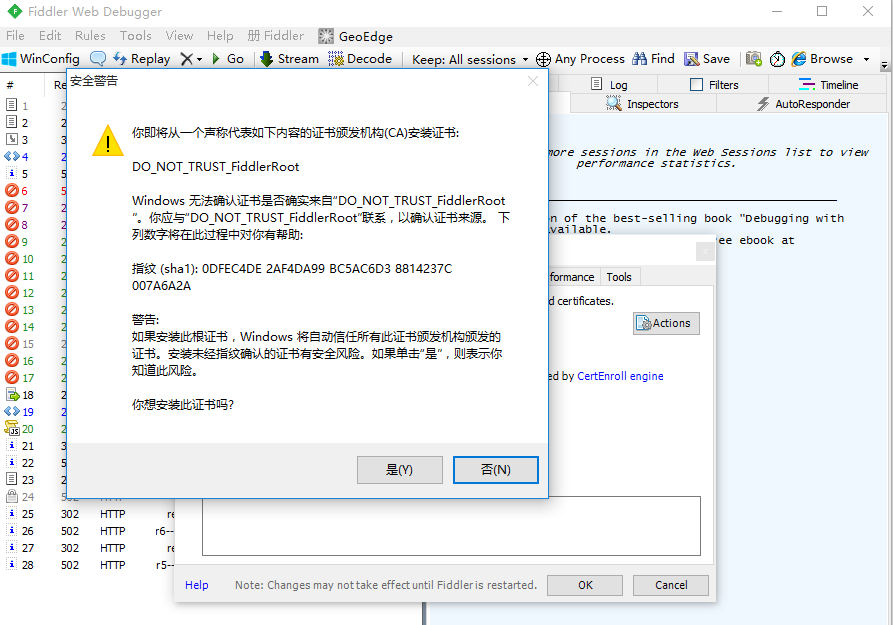
1. Tools –> Fiddler Options –>  HTTPS

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234227687-334229392.png)

2. 勾选Decrypt HTTPS Traffic

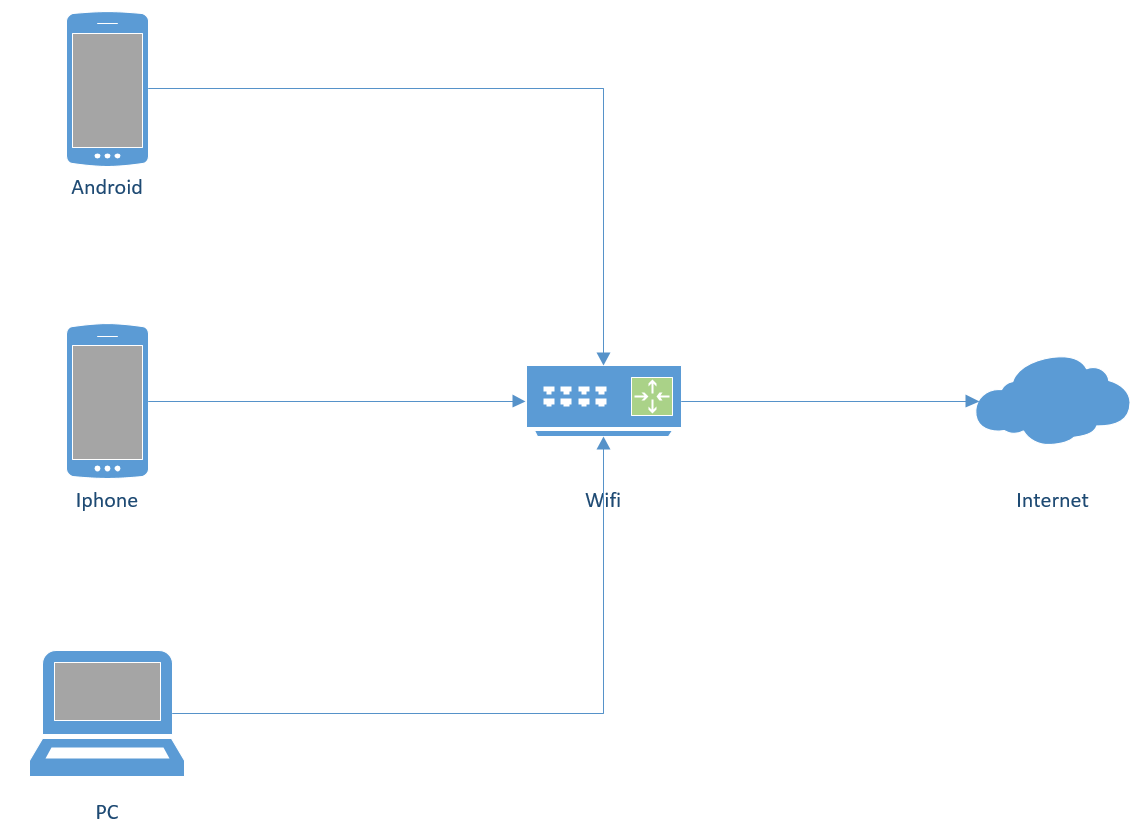
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234228703-2021706769.png)

3. 点击OK

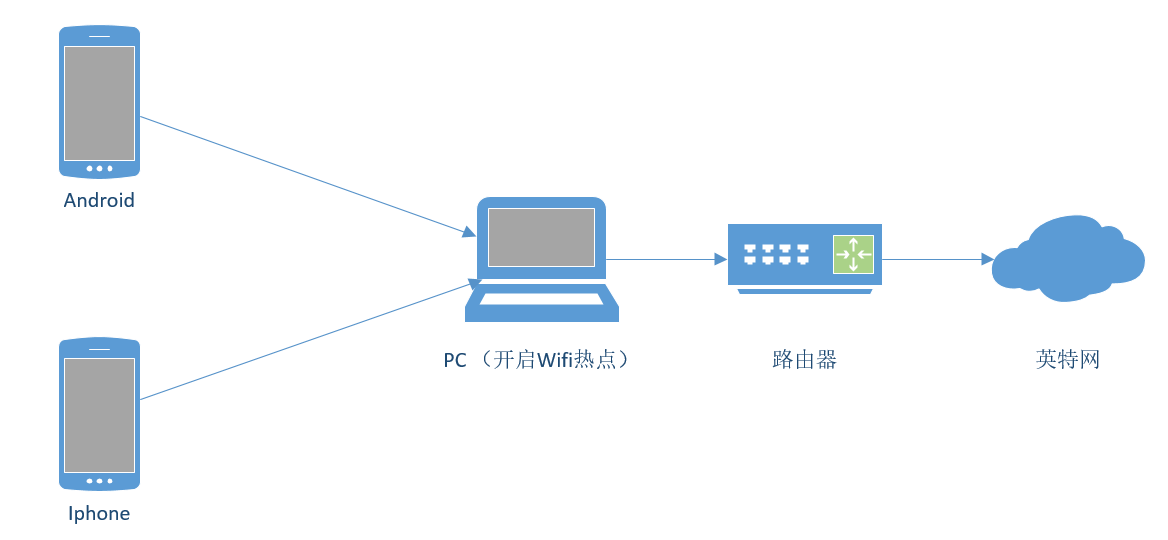
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160118234229797-92553757.png)

**3. Fiddler 抓取Iphone / Android数据包**

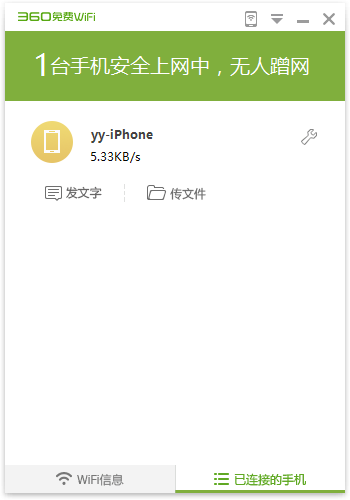
想要Fiddler抓取移动端设备的数据包，其实很简单，先来说说移动设备怎么去访问网络，看了下面这张图，就明白了。

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124738078-1117636725.png)

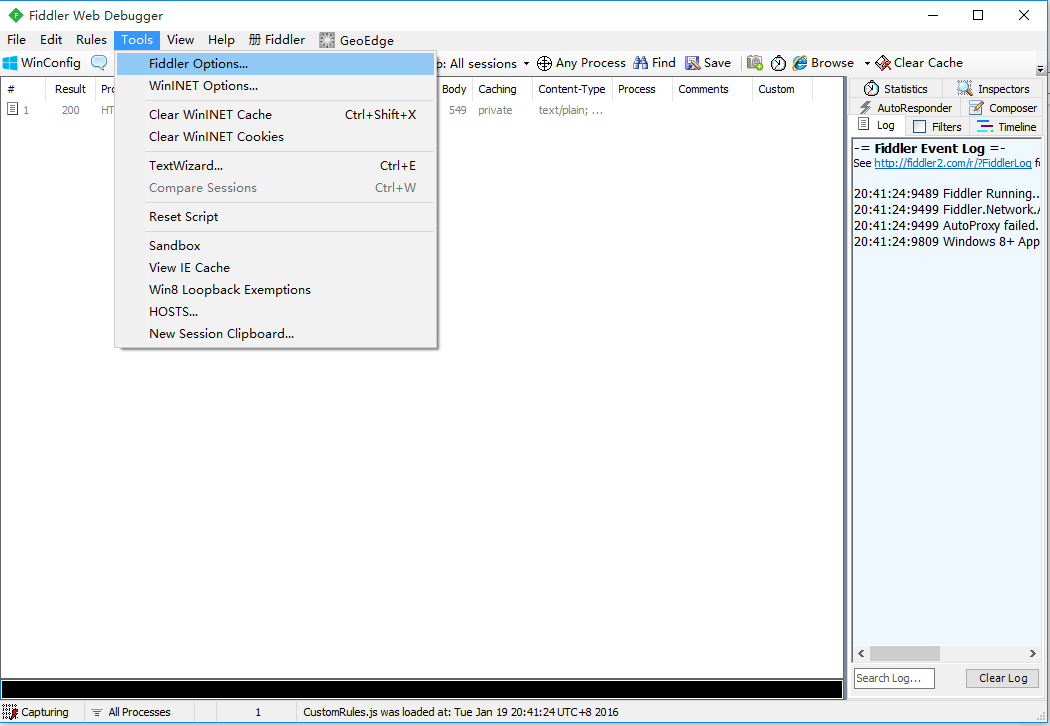
可以看得出，移动端的数据包，都是要走wifi出去，所以我们可以把自己的电脑开启热点，将手机连上电脑，Fiddler开启代理后，让这些数据通过Fiddler，Fiddler就可以抓到这些包，然后发给路由器（如图）：

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124738953-1847443891.png)

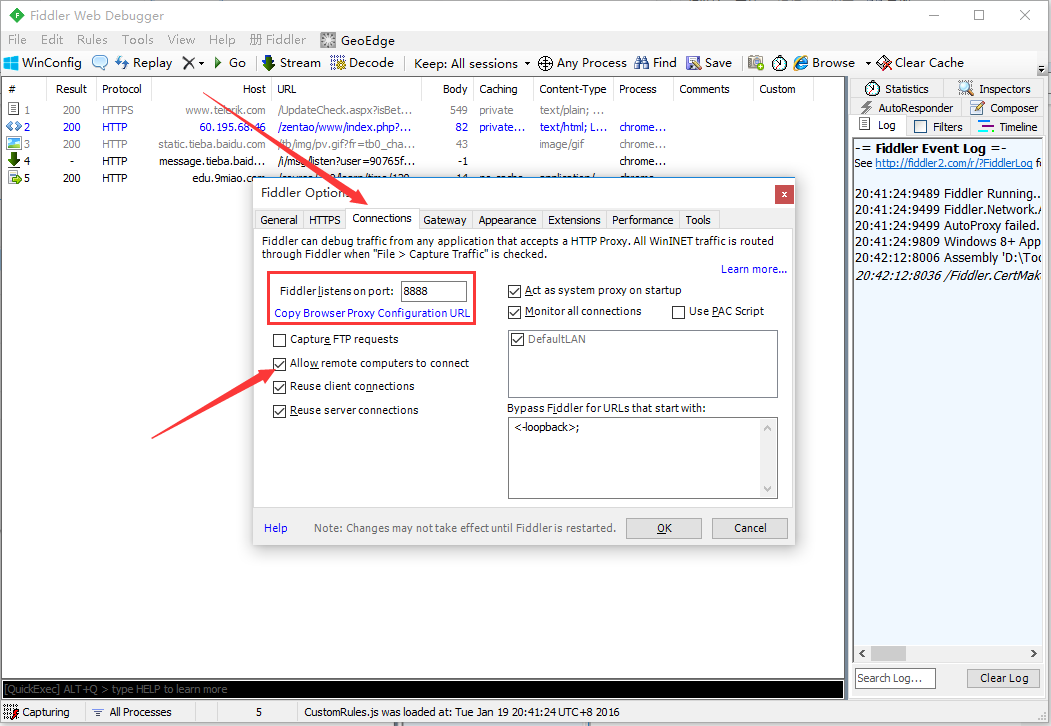
1. 打开Wifi热点，让手机连上（我这里用的360wifi，其实随意一个都行）

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124739890-506664580.png)

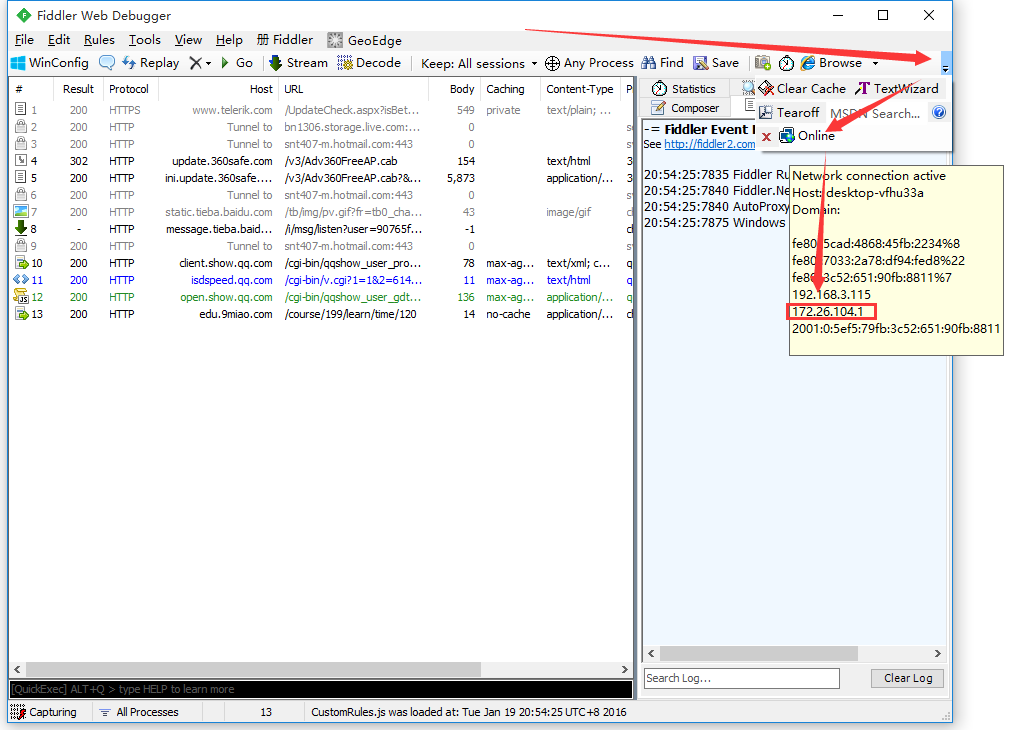
2. 打开Fidder，点击菜单栏中的 [Tools] –> [Fiddler Options]

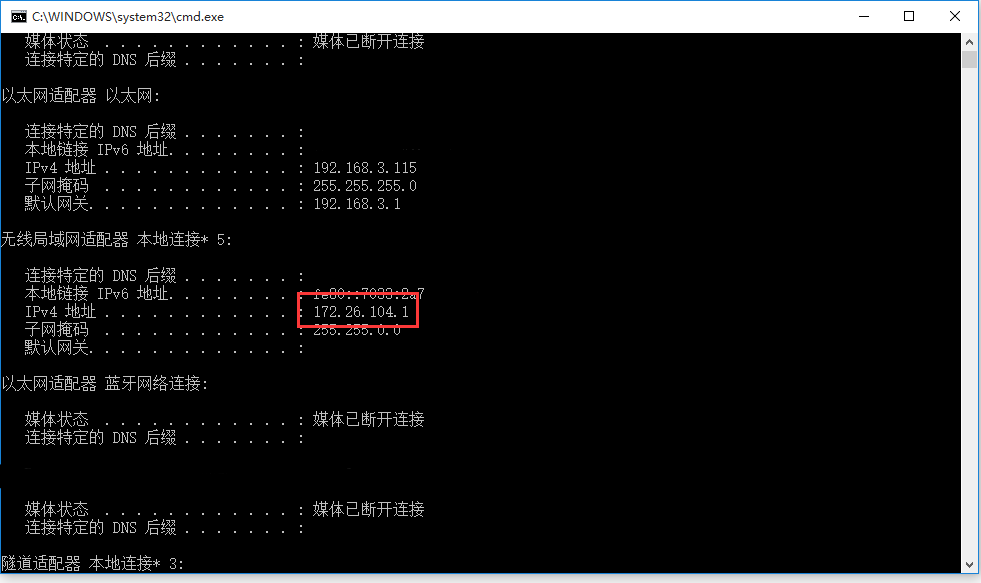
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124741265-1118326296.png)

3. 点击 [Connections] ，设置代理端口是8888， 勾选 Allow remote computers to connect， 点击OK

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124842640-1601285150.png)

4. 这时在 Fiddler 可以看到自己本机无线网卡的IP了（要是没有的话，重启Fiddler，或者可以在cmd中ipconfig找到自己的网卡IP）

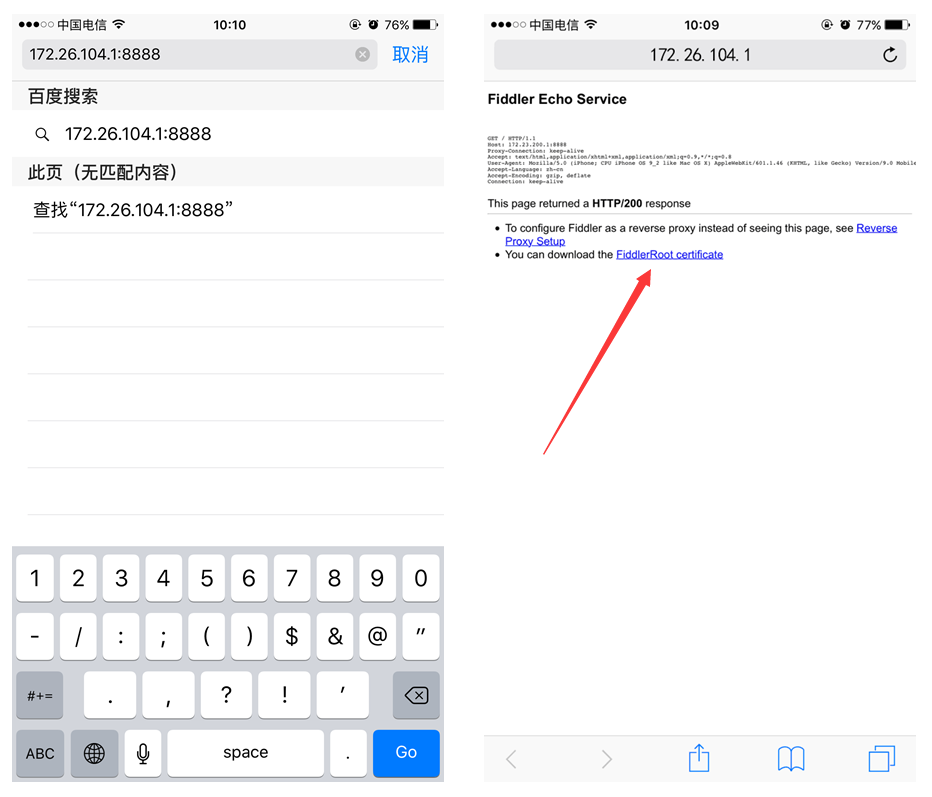
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124847625-691854364.png)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124849234-1548708450.png)

5. 在手机端连接PC的wifi，并且设置代理IP与端口（代理IP就是上图的IP，端口是Fiddler的代理端口8888）

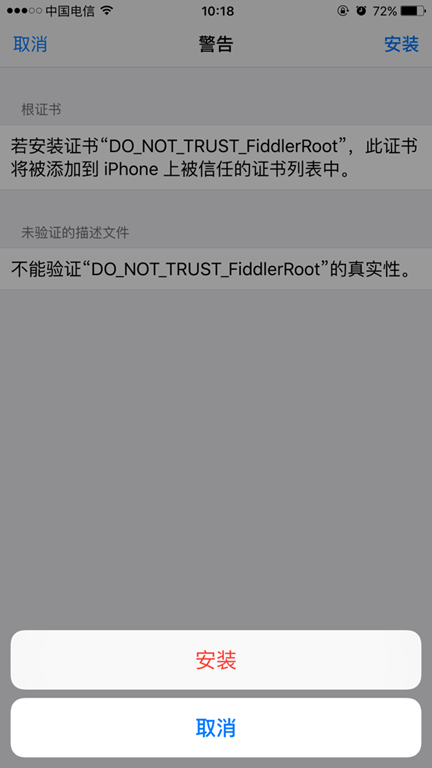
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124850750-1818439036.png)

6. 访问网页输入代理IP和端口，下载Fiddler的证书，点击下图FiddlerRoot certificate

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124901812-1556138866.png)

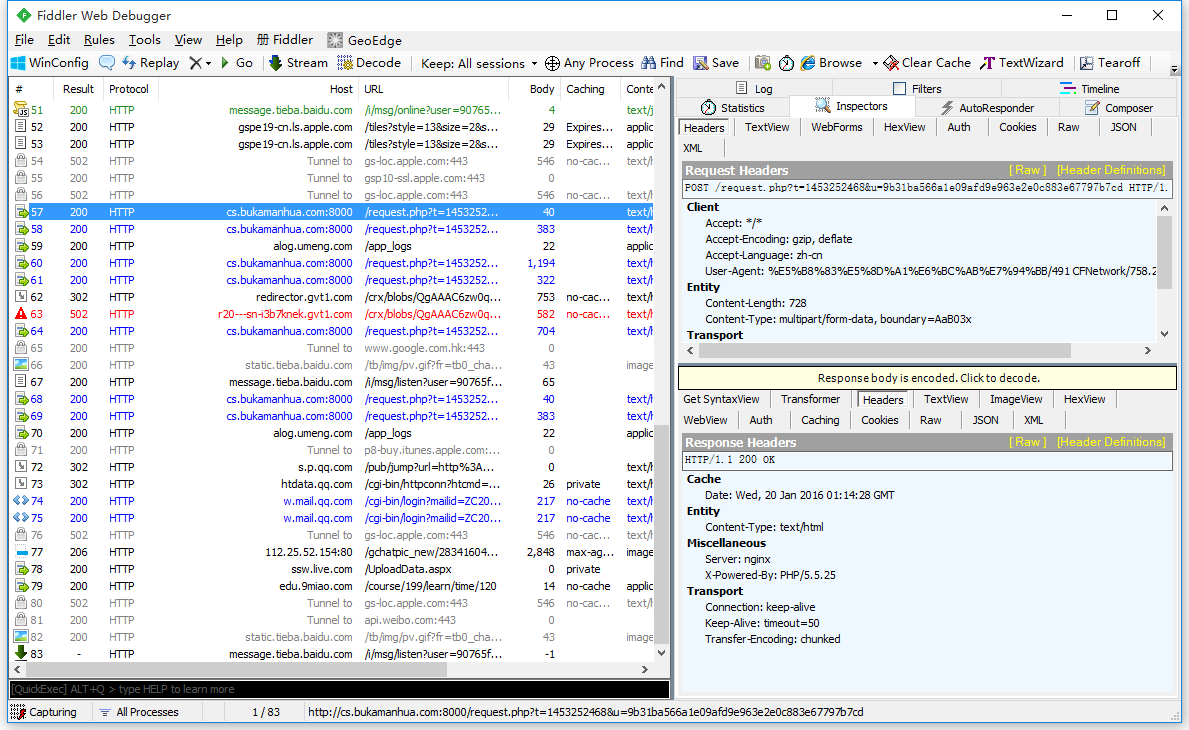
【注意】：如果打开浏览器碰到类似下面的报错，请打开Fiddler的证书解密模式（[Fiddler 设置解密HTTPS的网络数据](http://www.cnblogs.com/yyhh/p/5140852.html" \l "l02" \t "_blank)）

No root certificate was found. Have you enabled HTTPS traffic decryption in Fiddler yet?

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124906687-1656785164.png)          [](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120124935859-1823496174.png)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125000515-1465458167.png)          [](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125001187-2034345997.png)

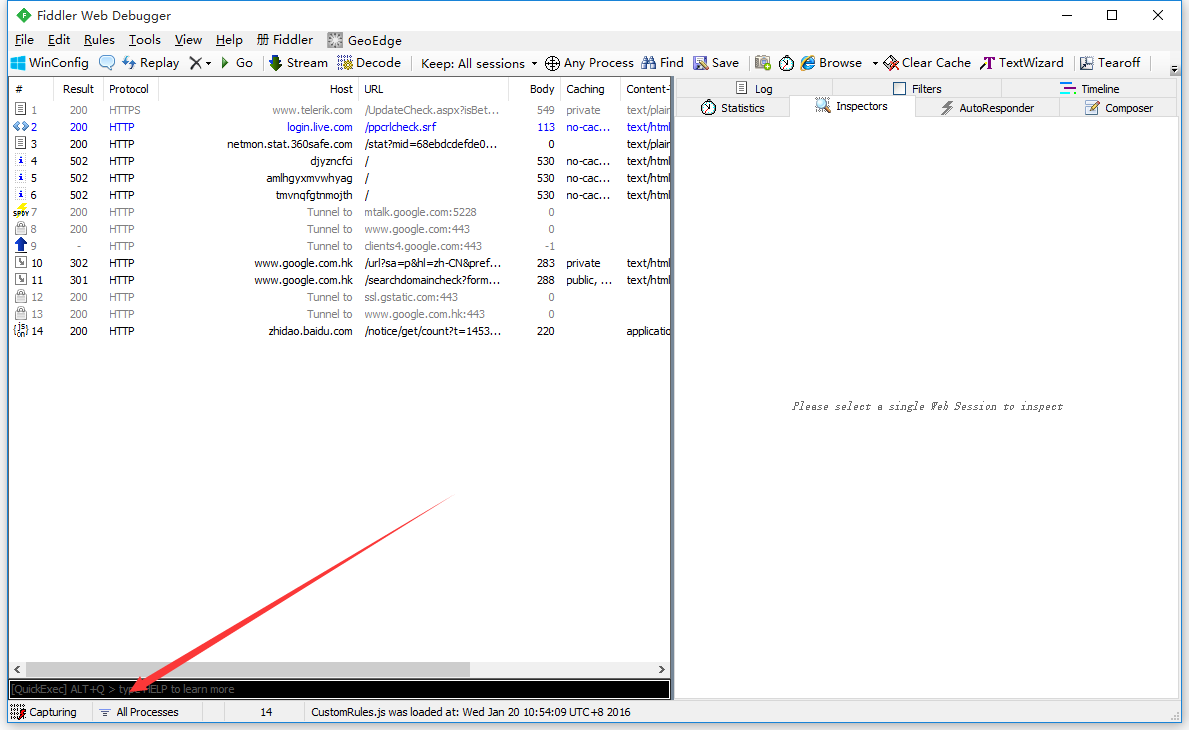
7. 安装完了证书，可以用手机访问应用，就可以看到截取到的数据包了。（下图选中是布卡漫画的数据包，下面还有QQ邮箱的）

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125404609-1454505432.png)

**4. Fiddler 内置命令与断点**

Fiddler还有一个藏的很深的命令框，就是眼前，我用了几年的Fiddler都没有发现它，偶尔在别人的文章发现还有这个小功能，还蛮好用的，整理下记录在这里。

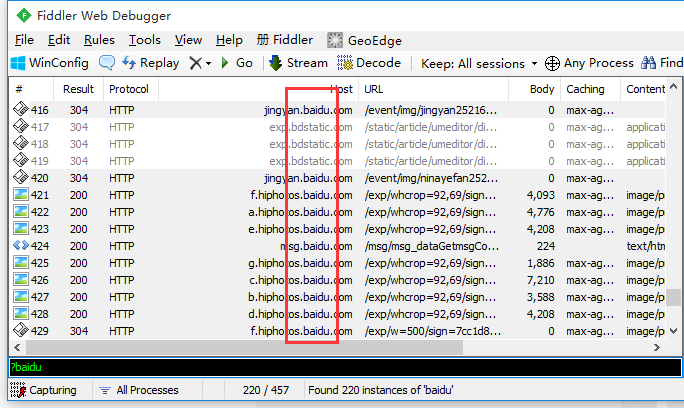
FIddler断点功能就是将请求截获下来，但是不发送，这个时候你可以干很多事情，比如说，把包改了，再发送给服务器君。还有balabala一大堆的事情可以做，就不举例子了。

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125407703-1527281341.png)

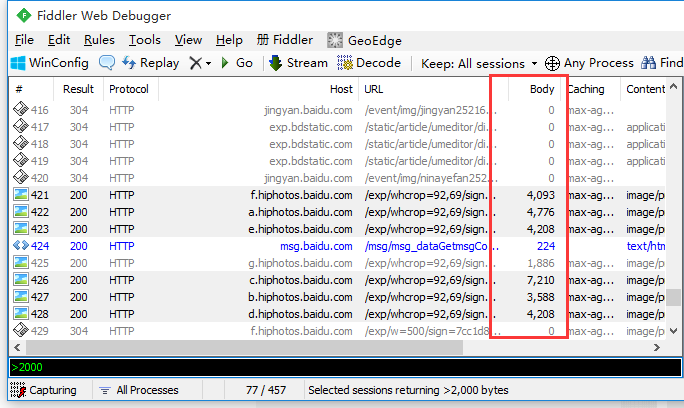
|  |  |  |  |
| --- | --- | --- | --- |
| **命令** | **对应请求项** | **介绍** | **示例** |
| ? | All | 问号后边跟一个字符串，可以匹配出包含这个字符串的请求 | ?google |
| > | Body | 大于号后面跟一个数字，可以匹配出请求大小，大于这个数字请求 | >1000 |
| < | Body | 小于号跟大于号相反，匹配出请求大小，小于这个数字的请求 | <100 |
| = | Result | 等于号后面跟数字，可以匹配HTTP返回码 | =200 |
| @ | Host | @后面跟Host，可以匹配域名 | @www.baidu.com |
| select | Content-Type | select后面跟响应类型，可以匹配到相关的类型 | select image |
| cls | All | 清空当前所有请求 | cls |
| dump | All | 将所有请求打包成saz压缩包，保存到“我的文档\Fiddler2\Captures”目录下 | dump |
| start | All | 开始监听请求 | start |
| stop | All | 停止监听请求 | stop |
| **断点命令** | | | |
| bpafter | All | bpafter后边跟一个字符串，表示中断所有包含该字符串的请求 | bpafter baidu（输入bpafter解除断点） |
| bpu | All | 跟bpafter差不多，只不过这个是收到请求了，中断响应 | bpu baidu（输入bpu解除断点） |
| bps | Result | 后面跟状态吗，表示中断所有是这个状态码的请求 | bps 200（输入bps解除断点） |
| bpv / bpm | HTTP方法 | 只中断HTTP方法的命令，HTTP方法如POST、GET | bpv get（输入bpv解除断点） |
| g / [**Go**](http://lib.csdn.net/base/go) | All | 放行所有中断下来的请求 | g |

示例演示：

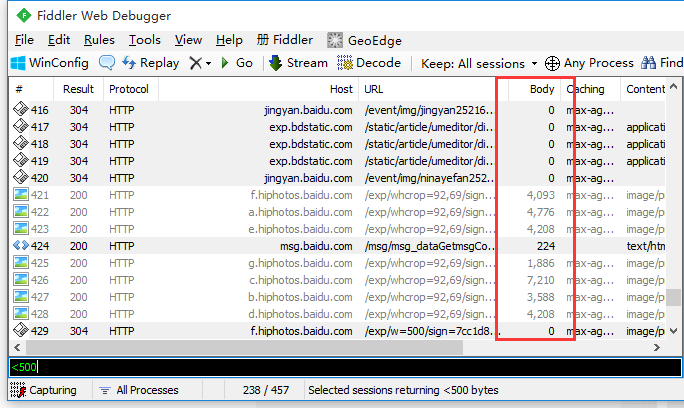
**?**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125409250-1875594948.png)

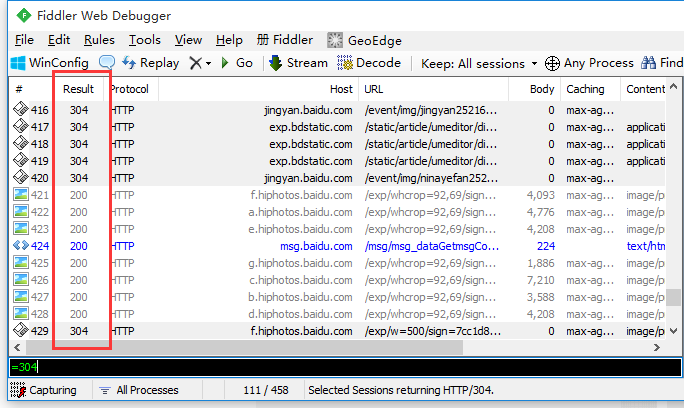
**>**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125413797-764521538.png)

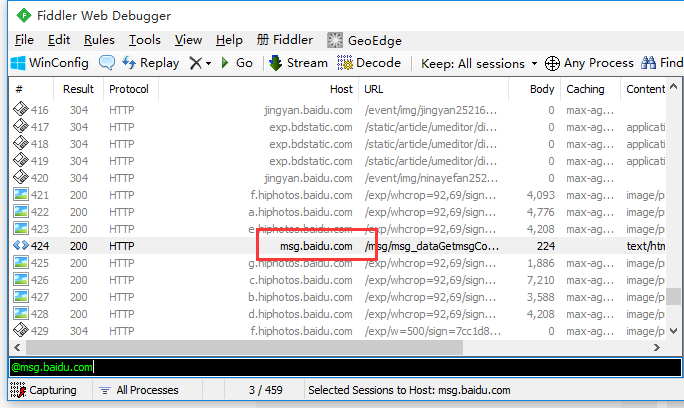
**<**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125414359-1320757344.png)

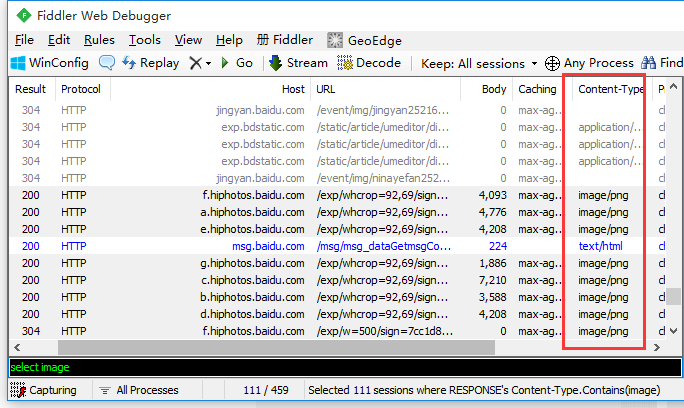
**=**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125415140-1688137995.png)

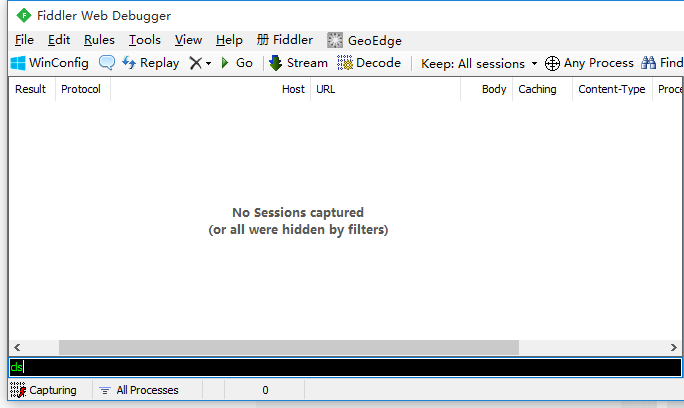
**@**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125415797-1085217901.png)

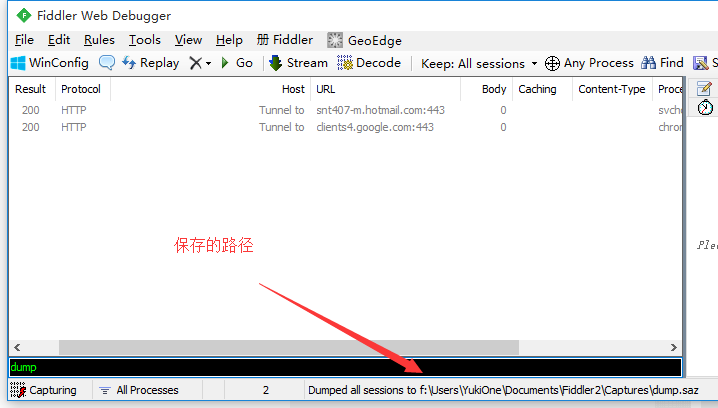
**select**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125417406-1083145527.png)

**cls**

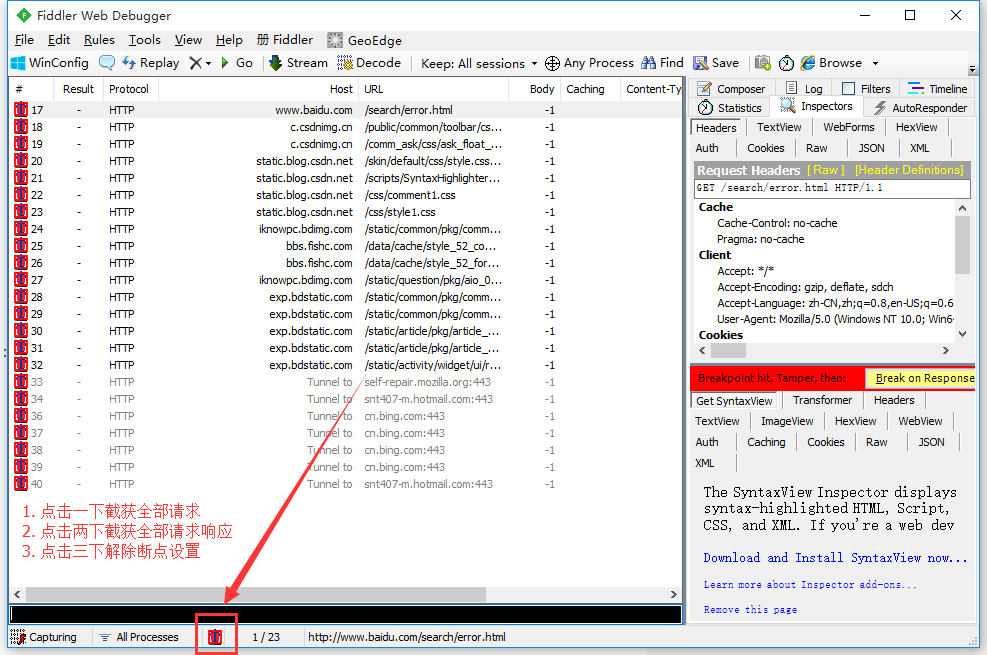
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125418047-180826397.png)

**dump**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125418578-1269598339.png)

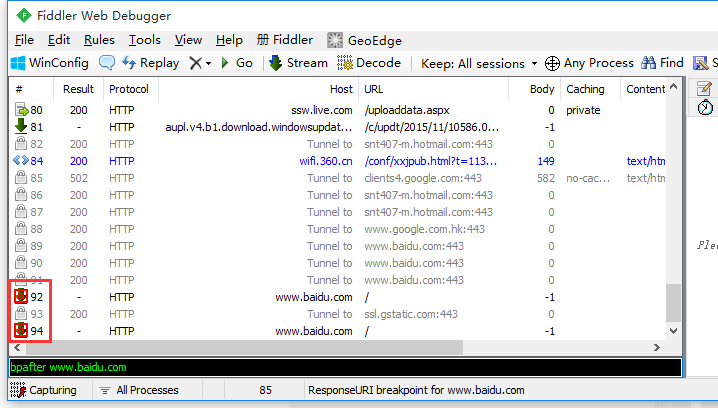
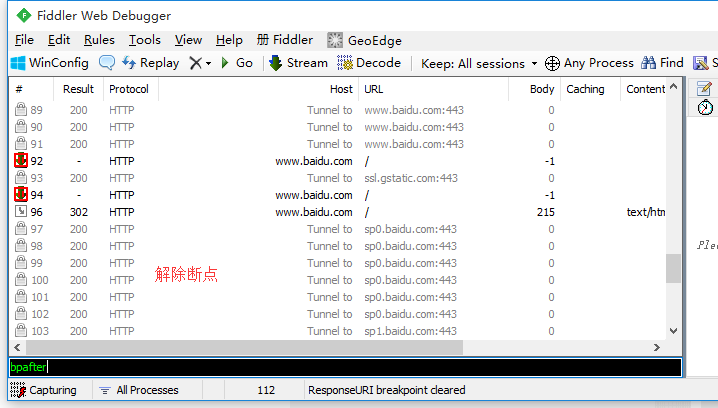
断点命令：

断点可以直接点击Fiddler下图的图标位置，就可以设置全部请求的断点，断点的命令可以精确设置需要截获那些请求。如下示例：

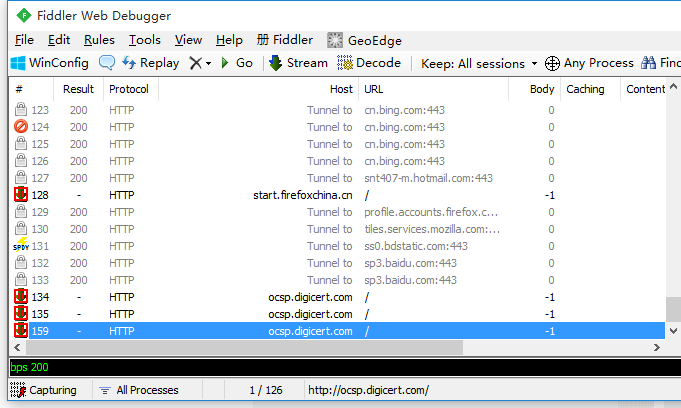
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125423265-52398327.png)

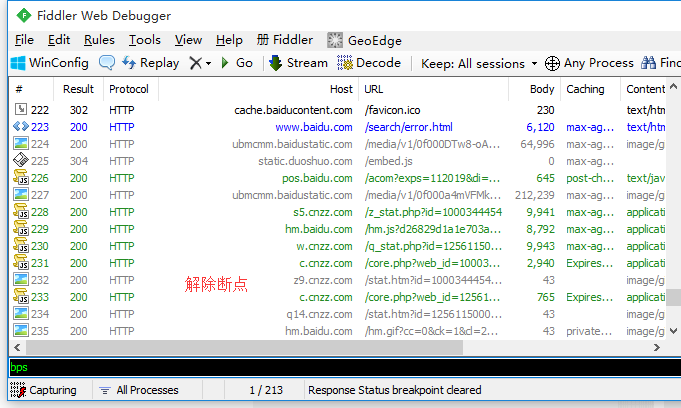
命令：

**bpafter**

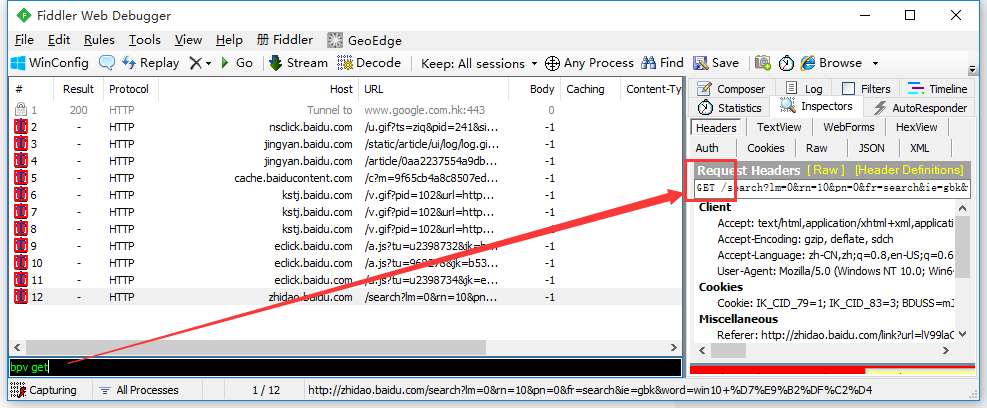
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125424500-1575845348.png)   [](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125425625-1142095522.png)

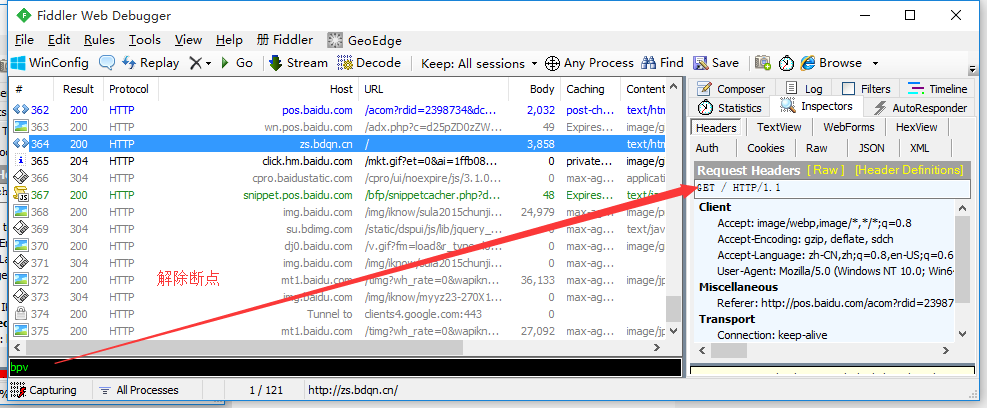
**bps**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125426359-1755944090.png)

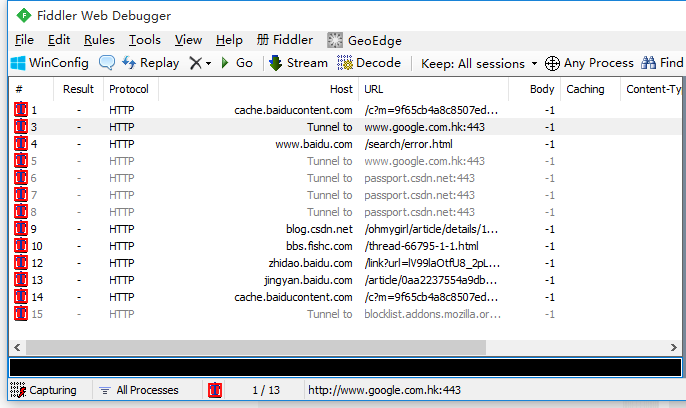
[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125427406-852314359.png)

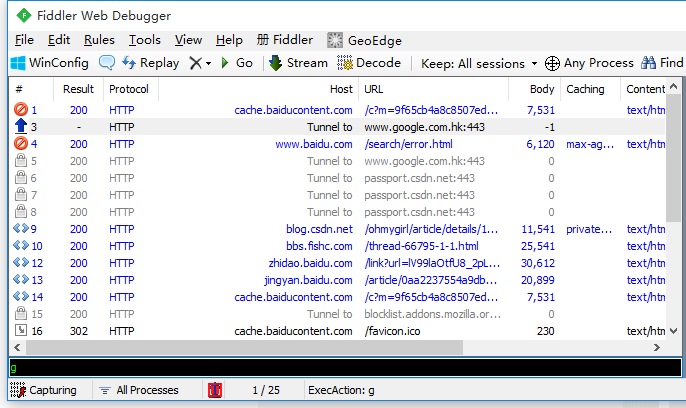
**bpv**

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125431062-1643690164.png)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125432437-1592068049.png)

**g /**[**go**](http://lib.csdn.net/base/go)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125437093-226617550.png)

[](http://images2015.cnblogs.com/blog/626593/201601/626593-20160120125437718-579701026.png)