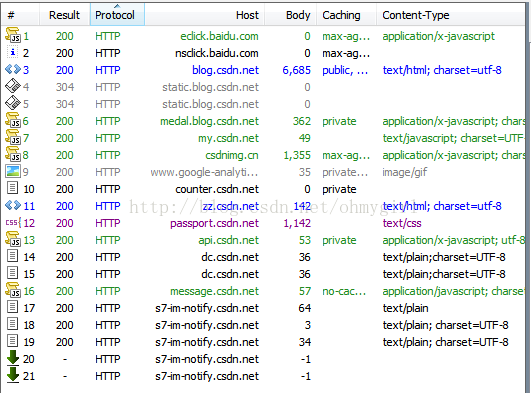
**【HTTP】Fiddler（二） - 使用Fiddler做抓包分析**

本文主要针对Fiddler的抓包处理。

Fiddler抓取HTTP请求。

抓包是Fiddler的最基本的应用，以本博客为例，启动Fiddler之后，在浏览器中输入[http://blog.csdn.net/ohmygirl](http://blog.csdn.net/ohmygirl" \t "_blank) 键入回车之后，在Fiddler的web session界面捕获到的HTTP请求如下图所示：

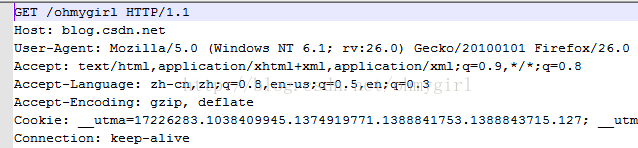


各字段的详细说明已经解释过，这里不再说明。需要注意的是#号列中的图标，每种图标代表不同的相应类型，具体的类型包括：



另外，注意请求的host字段。可以看到有来自多个[www.csdn.net](http://www.csdn.net/" \t "_blank)的子域名的响应，说明在大型网站的架构中，大多需要多个子域名，这些子域名可能是单独用于缓存静态资源的，也可能是专门负责媒体资源的，或者是专门负责数据统计的（如pingback）。

右键单击其中的一条请求。可以选择的操作有：save(保存请求的报文信息，可以是请求报文，可以是响应报文)。例如，我们保存的一条请求头信息如下：



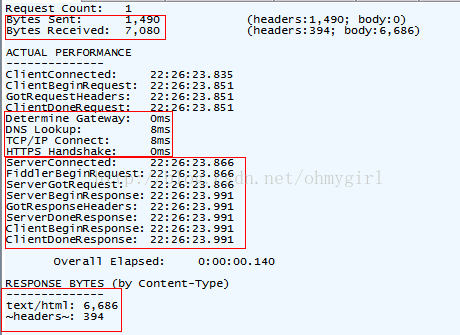
不仅是单条session,Fiddler还支持保存所有抓取到的session(并支持导入)，这对于抓取可疑请求然后保存，并在之后随时分析这些请求是很有帮助的。

如果想要重新发送某些请求，可以选中这些请求，然后点击工具栏中的reply.就可以重新发送选中的这些请求。

左键点击单条HTTP请求，可以在右侧的tab面板中看到如下信息：

**1. Statistic。**

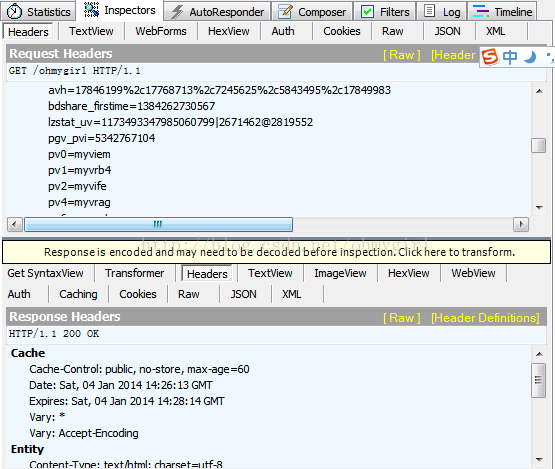
关于HTTP请求的性能和其他数据分析：



我们可以从中看出一些基本性能数据：如DNS解析的时间消耗是8ms,建立TCP/IP连接的时间消耗是8ms等等信息。

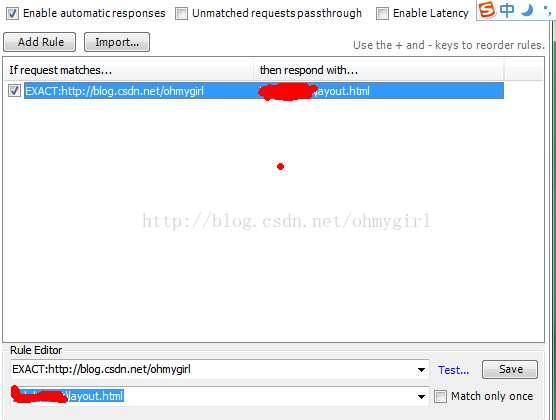
**2. Inspectors。**

分为上下两个部分，上半部分是请求头部分，下半部分是响应头部分。对于每一部分，提供了多种不同格式查看每个请求和响应的内容。JPG 格式使用 ImageView 就可以看到图片，HTML/JS/CSS 使用 TextView 可以看到响应的内容。Raw标签可以查看原始的符合HTTP标准的请求和响应头。Auth则可以查看授权Proxy-Authorization 和 Authorization的相关信息。Cookies标签可以看到请求的cookie和响应的set-cookie头信息。



**3. AutoResponder**

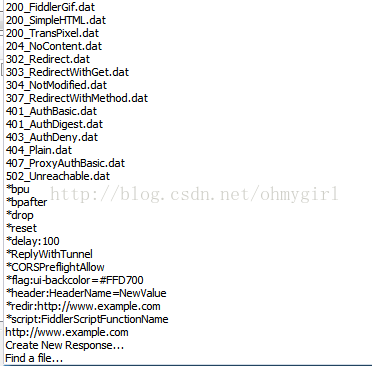
Fiddler比较重要且比较强大的功能之一。可用于拦截某一请求，并重定向到本地的资源，或者使用Fiddler的内置响应。可用于调试服务器端代码而无需修改服务器端的代码和配置，因为拦截和重定向后，实际上访问的是本地的文件或者得到的是Fiddler的内置响应。当勾选allow autoresponser 并设置相应的规则后（本例中的规则是将[http://blog.csdn.net/ohmygirl](http://blog.csdn.net/ohmygirl" \t "_blank)的请求拦截到本地的文件layout.html）,如下图所示



然后在浏览器中访问[http://blog.csdn.net/ohmygirl](http://blog.csdn.net/ohmygirl" \t "_blank)，得到的结果实际为：



这刚好是本地layout.html的内容，说明请求已经成功被拦截到本地.当然也可以使用Fiddler的内置响应。下图是Fiddler支持的拦截重定向的方式：



因此，如果要调试服务器的某个脚本文件，可以将该脚本拦截到本地，在本地修改完脚本之后，再修改服务器端的内容，这可以保证，尽量在真实的环境下去调试，从而最大限度的减少bug发生的可能性。

不仅是单个url，Fiddler支持多种url匹配的方式：

I. 字符匹配

如 example可以匹配 [http://www.example.com](http://www.example.com/)和http://example.com.cn

II. 完全匹配

以EXACT开头表示完全匹配，如上边的例子

EXACT:http://blog.csdn.net/ohmygirl

III. 正则表达式匹配

以regex: 开头，使用正则表达式来匹配URL

如：regex:(?insx).\*\.(css|js|php)$  表示匹配所有以css,js,php结尾的请求url

**4. Composer。**

老版本的fiddler中叫request-builder.顾名思义，可以构建相应的请求，有两种常用的方式构建请求:

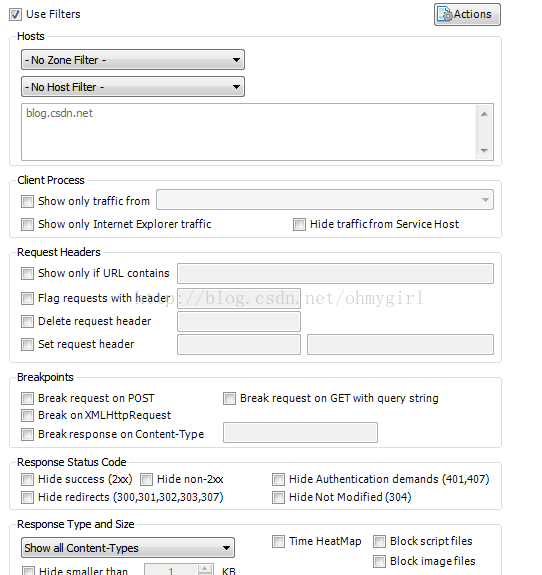
(1)Parsed 输入请求的url之后executed即可，也可以修改相应的头信息（如添加常用的accept, host, referrer, cookie，cache-control等头部）后execute.

这个功能的常见应用是：“刷票”（不是火车票！！），如刷新页面的访问量（基于道德和安全原因，如果你真去刷票，刷访问量，本博客概不负责）

(2)Raw。使用HTTP头部信息构建http请求。与上类似。不多叙述

**5． Filter**

Fiddler另一个比较强大的功能。Fiddler提供了多维度的过滤规则，足以满足日常开发调试的需求。如下图示：



过滤规则有：

a. host和zone过滤。可以过滤只显示intranet或则internet的HTTP请求

也可以选择特定域名的HTTP请求



b. client process:可以捕获指定进程的请求。

这对于调试单个应用的请求很有帮助。

其他更多的设置可以参考fiddler的官方文档。