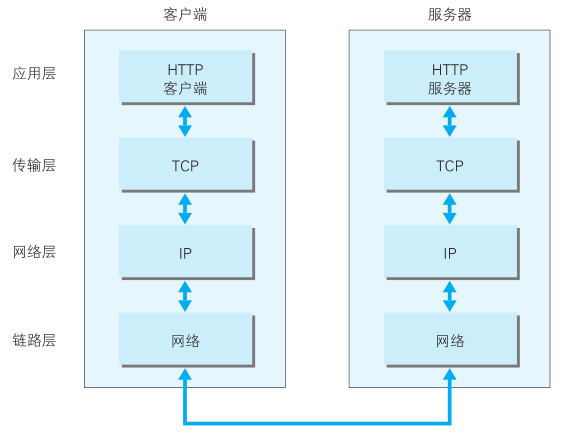
# HTTP协议

## HTTP协议简介

### 1.1 网络基础TCP/IP协议族

* 网络中的设备要进行通信，必须遵循一些相同的规则，这些规则我们称之为协议。网络中存在这多种协议，如：TCP、IP、HTTP、FTP等。这些协议我们统称为TCP/IP协议族，有时也简称为TCP/IP协议。网络是在TCP/IP基础上运作的。协议中存在各式各样的容。从电缆的规格到 IP 地址的选定方法、寻找异地用户的方法、双方建立通信的顺序，以及 Web 页面显示需要理的步骤，等等。
* TCP/IP 协议族里重要的一点就是分层。TCP/IP 协议族按层次分别分为以下 4 层：
  + 应用层
    - 应用层为用户提供常用的应用程序，并实现网络服务的各种功能。
  + 传输层
    - 传输层负责组装收到的数据（或者拆分要发送的数据）。
  + 网络层
    - 控制数据包的发送和接收。
  + 数据接口层。
    - 处理网络连接的硬件部分（路由器、网卡等）。
* 把TCP/IP层次化的好处是，如果网络中的信息的传输由一个协议统筹，如果某个地方的设计发生改变则需要修改整个协议，而采用分层结构之后，只需要将需要变动的层换掉即可。
* TCP/IP通信示意图：



### 1.2 HTTP协议

* HTTP（hypertext transport protocol），即超文本传输协议。这个协议详细规定了浏览器和万维网服务器之间互相通信的规则。
* 客户端与服务端通信时传输的内容我们称之为报文。
* HTTP就是一个通信规则，这个规则规定了客户端发送给服务器的报文格式，也规定了服务器发送给客户端的报文格式。实际我们要学习的就是这两种报文。客户端发送给服务器的称为”请求报文“，服务器发送给客户端的称为”响应报文“。

## HttpWatch

### 2.1 安装

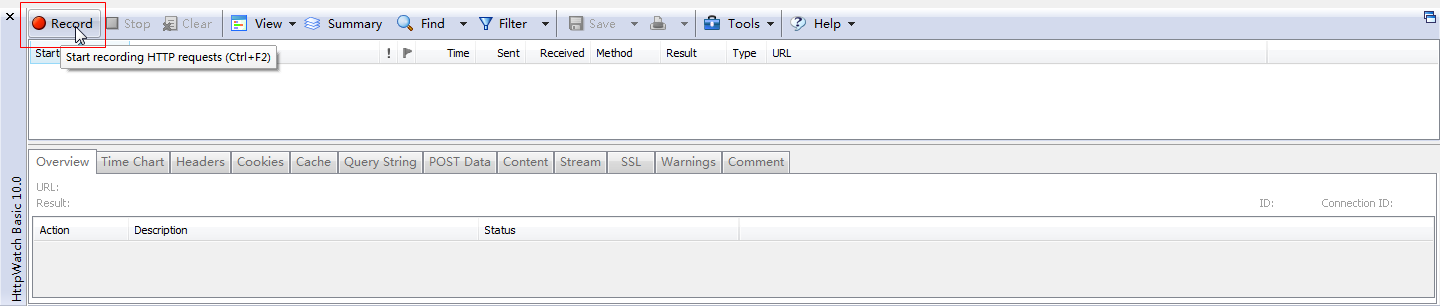
* 由于IE8以下的IE浏览器没有提供监听HTTP的功能，所以如果要使用IE8以下的浏览器查看HTTP请求的内容需要安装一个工具HttpWatch。
* Firefox和chrome都有内置的开发者工具，可以直接查看Http请求。
* HttpWatch的使用非常简单，直接安装，然后一直下一步，直到安装完成。

### 2.2 使用

* 安装完成后，打开IE浏览器，工具下拉列表可以看到HttpWatch Professional选项

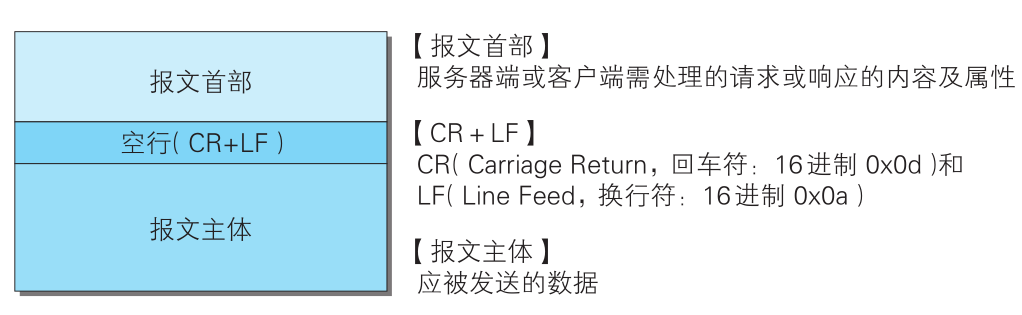


* 打开后点击Record按钮开始监听Http请求。



## 报文

### 3.1 报文格式



### 3.2 请求报文

#### 3.1.1 报文格式

|  |
| --- |
| 请求首行；  请求头信息；  空行；  请求体； |

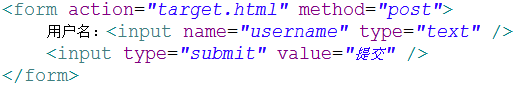
#### 3.1.2 GET请求

|  |
| --- |
| GET /Hello/index.jsp HTTP/1.1  Accept: \*/\*  Accept-Language: zh-CN  User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)  Accept-Encoding: gzip, deflate  Host: localhost:8080  Connection: Keep-Alive  Cookie: JSESSIONID=C55836CDA892D9124C03CF8FE8311B15 |

* + GET /Hello/index.jsp HTTP/1.1：GET请求，请求服务器路径为Hello/index.jsp，协议为1.1；
  + Host:localhost：请求的主机名为localhost；
  + User-Agent: Mozilla/4.0 (compatible; MSIE 8.0…：与浏览器和OS相关的信息。有些网站会显示用户的系统版本和浏览器版本信息，这都是通过获取User-Agent头信息而来的；
  + Accept: \*/\*：告诉服务器，当前客户端可以接收的文档类型， \*/\*，就表示什么都可以接收；
  + Accept-Language: zh-CN：当前客户端支持的语言，可以在浏览器的工具🡪选项中找到语言相关信息；
  + Accept-Encoding: gzip, deflate：支持的压缩格式。数据在网络上传递时，可能服务器会把数据压缩后再发送；
  + Connection: keep-alive：客户端支持的链接方式，保持一段时间链接，默认为3000ms；
  + Cookie: JSESSIONID=369766FDF6220F7803433C0B2DE36D98：因为不是第一次访问这个地址，所以会在请求中把上一次服务器响应中发送过来的Cookie在请求中一并发送去过。

#### 3.1.3 POST请求

* POST请求要求将form标签的method的属性设置为post



|  |
| --- |
| POST /Hello/target.html HTTP/1.1  Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap, \*/\*  Referer: http://localhost:8080/Hello/  Accept-Language: zh-CN  User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)  Content-Type: application/x-www-form-urlencoded  Accept-Encoding: gzip, deflate  Host: localhost:8080  Content-Length: 14  Connection: Keep-Alive  Cache-Control: no-cache  Cookie: JSESSIONID=774DA38C1B78AE288610D77621590345  username=admin |

* POST请求是可以有体的，而GET请求不能有请求体。
* Referer: http://localhost:8080/hello/index.jsp：请求来自哪个页面，例如你在百度上点击链接到了这里，那么Referer:http://www.baidu.com；如果你是在浏览器的地址栏中直接输入的地址，那么就没有Referer这个请求头了；
* Content-Type: application/x-www-form-urlencoded：表单的数据类型，说明会使用url格式编码数据；url编码的数据都是以“%”为前缀，后面跟随两位的16进制，例如“传智”这两个字使用UTF-8的url编码用为“%E4%BC%A0%E6%99%BA”；
* Content-Length:13：请求体的长度，这里表示13个字节。
* keyword=hello：请求体内容！hello是在表单中输入的数据，keyword是表单字段的名字。

### 3.3 响应报文

#### 3.2.1 报文格式

|  |
| --- |
| 响应首行；  响应头信息；  空行；  响应体； |

|  |
| --- |
| HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Content-Type: text/html;charset=UTF-8  Content-Length: 274  Date: Tue, 07 Apr 2015 10:08:26 GMT  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Insert title here</title>  </head>  <body>  <h1>Hello</h1>  </body>  </html> |

* HTTP/1.1 200 OK：响应协议为HTTP1.1，状态码为200，表示请求成功；
* Server: Apache-Coyote/1.1：服务器的版本信息；
* Content-Type: text/html;charset=UTF-8：响应体使用的编码为UTF-8；
* Content-Length: 274：响应体为274字节；
* Date: Tue, 07 Apr 2015 10:08:26 GMT：响应的时间，这可能会有8小时的时区差；

#### 3.2.2 响应码

响应码对浏览器来说很重要，它告诉浏览器响应的结果；

* 200：请求成功，浏览器会把响应体内容（通常是html）显示在浏览器中；
* 404：请求的资源没有找到，说明客户端错误的请求了不存在的资源；
* 500：请求资源找到了，但服务器内部出现了错误；
* 302：重定向，当响应码为302时，表示服务器要求浏览器重新再发一个请求，服务器会发送一个响应头Location，它指定了新请求的URL地址；