# HTTP协议

## HTTP协议简介

规范浏览器和服务器传递数据的格式：

浏览器🡪服务器 请求

服务器->浏览器 响应

数据格式： 遵循报文的格式

报文首行、报文头、报文空行、报文体

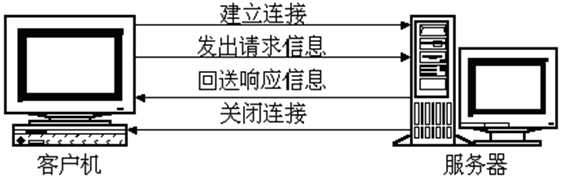
* HTTP **超文本传输协议** (HTTP-Hypertext transfer protocol)，是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。它**是一种详细规定了浏览器和万维网服务器之间互相通信的规则**，通过因特网传送万维网文档的数据传送协议。
* 客户端与服务端通信时传输的内容我们称之为**报文**。
* HTTP就是一个通信规则，这个规则规定了客户端发送给服务器的报文格式，也规定了服务器发送给客户端的报文格式。实际我们要学习的就是这两种报文。客户端发送给服务器的称为”**请求报文**“，服务器发送给客户端的称为”**响应报文**“。

## HTTP协议的发展历程

* 超文本传输协议的前身是世外桃源(Xanadu)项目，超文本的概念是泰德˙纳尔森(Ted Nelson)在1960年代提出的。进入哈佛大学后，纳尔森一直致力于超文本协议和该项目的研究，但他从未公开发表过资料。1989年，**蒂姆˙伯纳斯˙李**(Tim Berners Lee)在CERN(欧洲原子核研究委员会 = European Organization for Nuclear Research)担任软件咨询师的时候，开发了一套程序，**奠定了万维网(WWW = World Wide Web)的基础**。1990年12月，超文本在CERN首次上线。1991年夏天，继Telnet等协议之后，超文本转移协议成为互联网诸多协议的一分子。
* 当时，Telnet协议解决了一台计算机和另外一台计算机之间一对一的控制型通信的要求。邮件协议解决了一个发件人向少量人员发送信息的通信要求。文件传输协议解决一台计算机从另外一台计算机批量获取文件的通信要求，但是它不具备一边获取文件一边显示文件或对文件进行某种处理的功能。新闻传输协议解决了一对多新闻广播的通信要求。而超文本要解决的通信要求是：在一台计算机上获取并显示存放在多台计算机里的文本、数据、图片和其他类型的文件；它包含两大部分：超文本转移协议和超文本标记语言(HTML)。HTTP、HTML以及浏览器的诞生给互联网的普及带来了飞跃。

## HTTP协议的会话方式

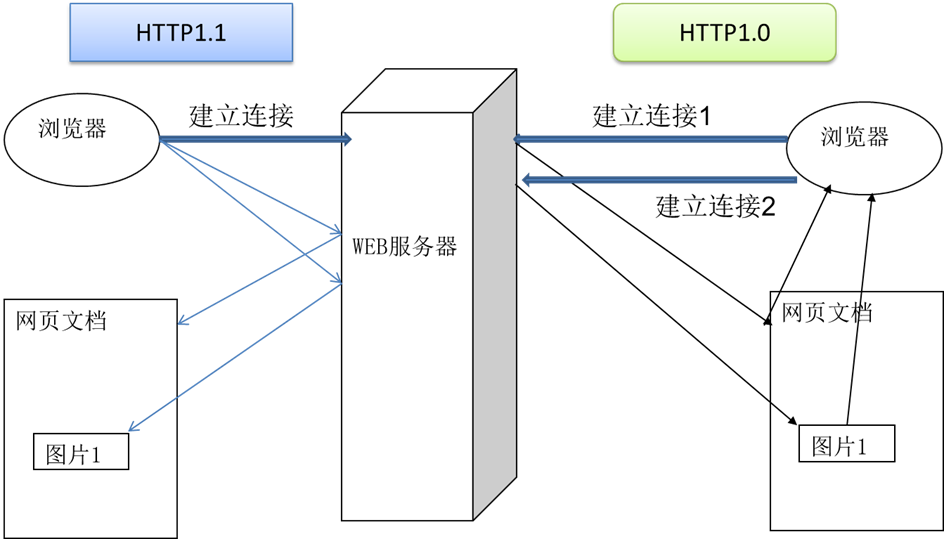
* 浏览器与服务器之间的通信过程要经历**四个步骤**



* 浏览器与WEB服务器的连接过程是短暂的，每次连接只处理一个请求和响应。对每一个页面的访问，浏览器与WEB服务器都要建立一次单独的连接。
* 浏览器到WEB服务器之间的所有通讯都是完全独立分开的请求和响应对。

## HTTP1.0和HTTP1.1的区别

* 在HTTP1.0版本中，浏览器请求一个带有图片的网页，会由于下载图片而与服务器之间开启一个新的连接；但在HTTP1.1版本中，允许浏览器在拿到当前请求对应的全部资源后再断开连接，提高了效率。



## HttpWatch

### 2.1 安装

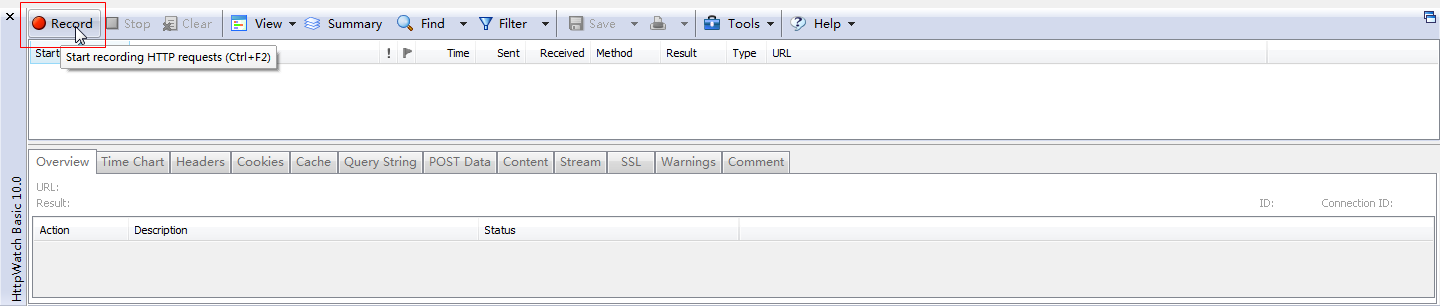
* 由于IE8以下的IE浏览器没有提供监听HTTP的功能，所以如果要使用IE8以下的浏览器查看HTTP请求的内容需要安装一个工具HttpWatch。
* Firefox和chrome都有内置的开发者工具，可以直接查看Http请求。
* HttpWatch的使用非常简单，直接安装，然后一直下一步，直到安装完成。

### 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

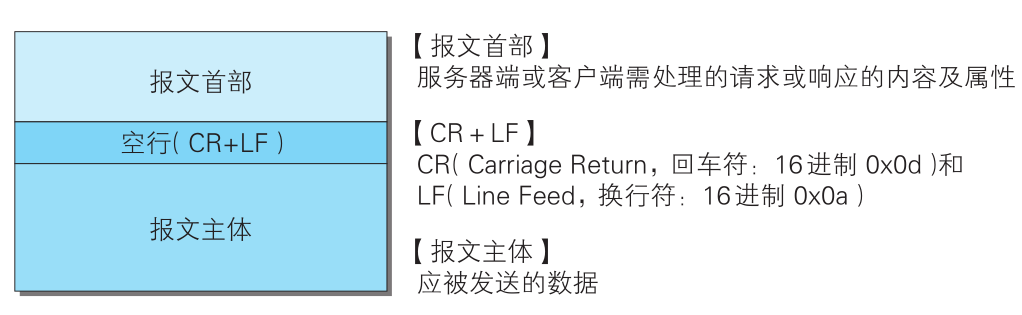


* 打开后点击Record按钮开始监听Http请求。



## 报文

### 6.1 报文格式



### 6.2 请求报文

#### 6.2.1 报文格式

|  |
| --- |
| 请求首行（**请求行**）；  请求头信息（**请求头**）；  空行；  **请求体**； |

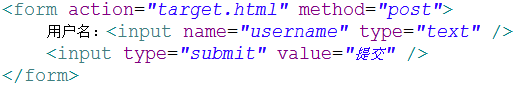
#### 6.2.2 GET请求

|  |
| --- |
| 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在请求中一并发送过去。

#### 6.2.3 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请求有请求体，而GET请求没有请求体。**
* 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“硅谷”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。

### 6.3 响应报文

#### 6.3.1 报文格式

|  |
| --- |
| 响应首行（**响应行**）；  响应头信息（**响应头**）；  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；

#### 6.3.2 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；