参考：<http://blog.csdn.net/wabiaozia/article/details/54571874>

**HTTP、TCP、UDP、Socket、servlet区别与联系**

**一 看概念**

1 <http://blog.csdn.net/xijiaohuangcao/article/details/6105623>

   TCP/IP是个协议组，可分为三个层次：网络层、传输层和应用层。在网络层有IP协议、ICMP协议、ARP协议、RARP协议和BOOTP协议。在传输层中有TCP协议与UDP协议 。 在应用层有FTP、HTTP、TELNET、SMTP、DNS等协议。因此，HTTP本身就是一个协议，是从Web服务器传输超文本到本地浏览器的传送协议。

2 <http://www.cnblogs.com/renyuan/archive/2013/01/19/2867720.html> 还有很多概念介绍的很好，必看

    TPC/IP协议是传输层协议，主要解决数据如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。

   术语TCP/IP代表传输控制协议/网际协议，指的是一系列协议。“IP”代表网际协议，TCP和UDP使用该协议从一个网络传送数据包到另一个网络。把IP想像成一种高速公路，它允 许其它协议在上面行驶并找到其它电脑的出口。TCP和UDP是高速公路上的“卡车”，它们携带的货物就是像HTTP，文件传输协议FTP这样的协议等。

　下面的图表试图显示不同的TCP/IP和其他的协议在最初OSI模型中的位置：

|  |  |  |
| --- | --- | --- |
| 7 | 应用层 | 例如[HTTP](http://zh.wikipedia.org/wiki/HTTP)、[SMTP](http://zh.wikipedia.org/wiki/SMTP)、[SNMP](http://zh.wikipedia.org/wiki/SNMP)、[FTP](http://zh.wikipedia.org/wiki/FTP)、[Telnet](http://zh.wikipedia.org/wiki/Telnet)、[SIP](http://zh.wikipedia.org/wiki/SIP)、[SSH](http://zh.wikipedia.org/wiki/SSH)、[NFS](http://zh.wikipedia.org/wiki/NFS)、[RTSP](http://zh.wikipedia.org/wiki/RTSP)、[XMPP](http://zh.wikipedia.org/wiki/XMPP)、[Whois](http://zh.wikipedia.org/wiki/Whois)、[ENRP](http://zh.wikipedia.org/w/index.php?title=ENRP&action=edit&redlink=1) |
| 6 | 表示层 | 例如[XDR](http://zh.wikipedia.org/w/index.php?title=External_Data_Representation&action=edit&redlink=1)、[ASN.1](http://zh.wikipedia.org/w/index.php?title=Abstract_Syntax_Notation_1&action=edit&redlink=1)、[SMB](http://zh.wikipedia.org/w/index.php?title=Server_message_block&action=edit&redlink=1)、[AFP](http://zh.wikipedia.org/w/index.php?title=Apple_Filing_Protocol&action=edit&redlink=1)、[NCP](http://zh.wikipedia.org/w/index.php?title=NetWare_Core_Protocol&action=edit&redlink=1) |
| 5 | 会话层 | 例如[ASAP](http://zh.wikipedia.org/w/index.php?title=Aggregate_Server_Access_Protocol&action=edit&redlink=1)、[TLS](http://zh.wikipedia.org/wiki/Transport_Layer_Security)、[SSH](http://zh.wikipedia.org/wiki/SSH)、ISO 8327 / CCITT X.225、[RPC](http://zh.wikipedia.org/w/index.php?title=Remote_procedure_call&action=edit&redlink=1)、[NetBIOS](http://zh.wikipedia.org/w/index.php?title=NetBIOS&action=edit&redlink=1)、[ASP](http://zh.wikipedia.org/w/index.php?title=AppleTalk&action=edit&redlink=1)、[Winsock](http://zh.wikipedia.org/w/index.php?title=Winsock&action=edit&redlink=1)、[BSD sockets](http://zh.wikipedia.org/wiki/Berkeley_sockets) |
| 4 | 传输层 | 例如[TCP](http://zh.wikipedia.org/wiki/TCP)、[UDP](http://zh.wikipedia.org/wiki/User_Datagram_Protocol)、[RTP](http://zh.wikipedia.org/w/index.php?title=Real-time_Transport_Protocol&action=edit&redlink=1)、[SCTP](http://zh.wikipedia.org/w/index.php?title=Stream_Control_Transmission_Protocol&action=edit&redlink=1)、[SPX](http://zh.wikipedia.org/w/index.php?title=Sequenced_packet_exchange&action=edit&redlink=1)、[ATP](http://zh.wikipedia.org/w/index.php?title=AppleTalk&action=edit&redlink=1)、[IL](http://zh.wikipedia.org/w/index.php?title=IL_Protocol&action=edit&redlink=1) |
| 3 | 网络层 | 例如[IP](http://zh.wikipedia.org/wiki/%E7%BD%91%E9%99%85%E5%8D%8F%E8%AE%AE)、[ICMP](http://zh.wikipedia.org/wiki/ICMP)、[IGMP](http://zh.wikipedia.org/wiki/IGMP)、[IPX](http://zh.wikipedia.org/wiki/IPX)、[BGP](http://zh.wikipedia.org/wiki/BGP)、[OSPF](http://zh.wikipedia.org/wiki/OSPF)、[RIP](http://zh.wikipedia.org/wiki/RIP)、[IGRP](http://zh.wikipedia.org/wiki/IGRP)、[EIGRP](http://zh.wikipedia.org/wiki/EIGRP)、[ARP](http://zh.wikipedia.org/wiki/ARP)、[RARP](http://zh.wikipedia.org/wiki/RARP)、 [X.25](http://zh.wikipedia.org/wiki/X.25) |
| 2 | 数据链路层 | 例如[以太网](http://zh.wikipedia.org/wiki/%E4%BB%A5%E5%A4%AA%E7%BD%91)、[令牌环](http://zh.wikipedia.org/wiki/%E4%BB%A4%E7%89%8C%E7%8E%AF)、[HDLC](http://zh.wikipedia.org/wiki/HDLC)、[帧中继](http://zh.wikipedia.org/wiki/%E5%B8%A7%E4%B8%AD%E7%BB%A7)、[ISDN](http://zh.wikipedia.org/wiki/ISDN)、[ATM](http://zh.wikipedia.org/wiki/%E5%BC%82%E6%AD%A5%E4%BC%A0%E8%BE%93%E6%A8%A1%E5%BC%8F)、[IEEE 802.11](http://zh.wikipedia.org/wiki/IEEE_802.11)、[FDDI](http://zh.wikipedia.org/wiki/FDDI)、[PPP](http://zh.wikipedia.org/wiki/PPP) |
| 1 | 物理层 | 例如[线路](http://zh.wikipedia.org/w/index.php?title=%E7%BA%BF%E8%B7%AF&action=edit&redlink=1)、[无线电](http://zh.wikipedia.org/wiki/%E6%97%A0%E7%BA%BF%E7%94%B5)、[光纤](http://zh.wikipedia.org/wiki/%E5%85%89%E7%BA%A4)、[信鸽](http://zh.wikipedia.org/wiki/%E4%BF%A1%E9%B8%BD) |

**二 看区别**

  1 <http://jingyan.baidu.com/article/08b6a591e07ecc14a80922f1.html>

     而我们平时说的最多的**socket**是什么呢，实际上socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口(API，即通过socket编程能够操作tcp/ip)。**socket则是对TCP/IP协议的封装和应用**(程序员层面上)。也可以说，TPC/IP协议是传输层协议，主要解决数据如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。关于**TCP/IP和HTTP协议的关系**，网络有一段比较容易理解的介绍:“我们在传输数据时，可以只使用(传输层)TCP/IP协议，但是那样的话，如果没有应用层，便无法识别数据内容。如果想要使传输的数据有意义，则必须使用到应用层协议。

  2 <http://www.oschina.net/question/12_52027> **红薯**写的

Servlet 是 J2EE 最重要的一部分，有了 Servlet 你就是 J2EE 了，J2EE 的其他方面的内容择需采用。而 Servlet 规范你需要掌握的就是 servlet 和 filter 这两项技术。绝大多数框架不是基于 servlet 就是基于 filter，如果它要在 Servlet 容器上运行，就永远也脱离不开这个模型。  
为什么 Servlet 规范会有两个包，javax.servlet 和 javax.servlet.http ，早先设计该规范的人认为 Servlet 是一种服务模型，不一定是依赖某种网络协议之上，因此就抽象出了一个 javax.servlet ，同时在提供一个基于 HTTP 协议上的接口扩展。但是从实际运行这么多年来看，似乎没有发现有在其他协议上实现的 Servlet 技术。

     为什么我这么强调 HttpServletRequest 和 HttpServletResponse 这两个接口，因为 Web 开发是离不开 HTTP 协议的，**而 Servlet 规范其实就是对 HTTP 协议做面向对象的封装(类比socket，提供了操作request和responnse的api)**，HTTP协议中的请求和响应就是对应了 HttpServletRequest 和 HttpServletResponse 这两个接口。

     你可以通过 HttpServletRequest 来获取所有请求相关的信息，包括 URI、Cookie、Header、请求参数等等，别无它路。因此当你使用某个框架时，你想获取HTTP请求的相关信息，只要拿到 HttpServletRequest 实例即可。而 HttpServletResponse接口是用来生产 HTTP 回应，包含 Cookie、Header 以及回应的内容等等。

     HTTP 协议里是没有关于 Session 会话的定义，**Session 是各种编程语言根据 HTTP 协议的无状态这种特点而产生的**。其实现无非就是服务器端的一个哈希表，哈希表的Key就是传递给浏览器的名为 jsessionid 的 Cookie 值。

Session是在client多次请求时，保存client的状态的一种机制，实现session机制无非就两种方法：

1. 使用sessionId借助client的cookie
2. 当client禁用了cookie时使用url，使用encode。

3 <http://blog.csdn.net/sd0902/article/details/8395641>  listener->filter->struts拦截器->servlet。

    web.xml中元素执行的顺序listener->filter->struts拦截器->servlet。  
1.过滤器的概念  
   Java中的Filter 并不是一个标准的Servlet ，它不能处理用户请求，也不能对客户端生成响应。 主要用于对HttpServletRequest 进行预处理，也可以对HttpServletResponse 进行后处理，是个典型的处理链。  
优点：过滤链的好处是，执行过程中任何时候都可以打断，只要不执行chain.doFilter()就不会再执行后面的过滤器和请求的内容。而在实际使用时，就要特别注意过滤链的执行顺序问题  
2.过滤器的作用描述  
    在HttpServletRequest 到达Servlet 之前，拦截客户的HttpServletRequest 。   根据需要检查HttpServletRequest ，也可以修改HttpServletRequest 头和数据。   
在HttpServletResponse 到达客户端之前，拦截HttpServletResponse 。 根据需要检查HttpServletResponse ，可以修改HttpServletResponse 头和数据。