TCP/IP协议

TCP/IP传输协议,是传输控制/网络协议,也称为网络通讯协议,它是在网络的使用中基本的通信协议,它将网络模型分为4层,分别为:应用层、传输层、互联网层、主机-网络层,在TCP/IP协议中,用"源IP"、"源端口号"、"目的IP"、"目的端口号"、"协议号"的五元组来标识一个通信。其中应用层的主要协议有Telnet、FTP、SMTP;传输层的主要协议有UDP、TCP；网络层的主要协议有ICMP、Ip、IGMP;主机-网络层主要有ARP、RARP。

1.SSH（Secure Shell）安全外壳协议，

传输层协议，是一种通用的、功能强大的、基于软件的网络安全解决方案。计算机每次向网络发送数据时，SSH都会自动对其进行加密。数据到达目的地时，SSH自动对加密数据进行解密。整个过程都是透明的，使用OpenSSH工具将会增进你的系统安全性。谈到网络安全访问，相信大家首先想到的就是安全Shell，也就是Secure Shell，通常简写为SSH。这是因为SSH安装容易、使用简单，而且比较常见，一般的Unix系统、Linux系统、FreeBSD系统都附带有支持SSH的应用程序包。

2.Telnet协议（远程终端协议）

是TCP/IP协议族中的一员，是Internet远程登录服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。在终端使用者的电脑上使用telnet程序，用它连接到服务器。终端使用者可以在telnet程序中输入命令，这些命令会在服务器上运行，就像直接在服务器的控制台上输入一样。可以在本地就能控制服务器。要开始一个telnet会话，必须输入用户名和密码来登录服务器。Telnet是常用的远程控制Web服务器的方法。

3.ICMP（Internet Control Message Protocol）Internet控制报文协议，网络层协议

它是TCP/IP协议簇的一个子协议，用于在IP主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。ICMP使用IP的基本支持，就像它是一个更高级别的协议，但是，ICMP实际上是IP的一个组成部分，必须由每个IP模块实现。

4.ftp 文件传输协议（File Transfer Protocol，FTP）

是用于在网络上进行文件传输的一套标准协议，它工作在 OSI 模型的第七层， TCP 模型的第四层，即应用层， 使用 TCP 传输而不是 UDP， 客户在和服务器建立连接前要经过一个"三次握手"的过程， 保证客户与服务器之间的连接是可靠的， 而且是面向连接， 为数据传输提供可靠保证。FTP允许用户以文件操作的方式（如文件的增、删、改、查、传送等）与另一主机相互通信。默认端口号21。

5.SMTP（简单邮件传输协议）

是一个相对简单的基于文本的协议。在其之上指定了一条消息的一个或多个接收者（在大多数情况下被确认是存在的），然后消息文本会被传输。可以很简单地通过telnet程序来测试一个SMTP服务器。SMTP使用TCP端口25。要为一个给定的域名决定一个SMTP服务器，需要使用MX (Mail eXchange) DNS。

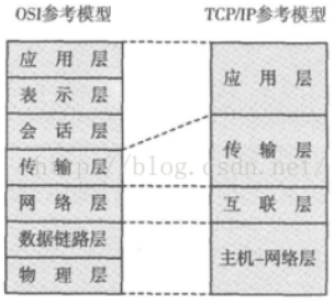
6.DNS：域名系统（英文：Domain Name System，缩写：DNS）是互联网的一项服务。

它作为将域名和IP地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。DNS使用TCP和UDP端口53。当前，对于每一级域名长度的限制是63个字符，域名总长度则不能超过253个字符。

7.地址解析协议，即ARP（Address Resolution Protocol)

是根据IP地址获取物理地址的一个TCP/IP协议。主机发送信息时将包含目标IP地址的ARP请求广播到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源。地址解析协议是建立在网络中各个主机互相信任的基础上的，局域网络上的主机可以自主发送ARP应答消息，其他主机收到应答报文时不会检测该报文的真实性就会将其记入本机ARP缓存；由此攻击者就可以向某一主机发送伪ARP应答报文，使其发送的信息无法到达预期的主机或到达错误的主机，这就构成了一个ARP欺骗。ARP命令可用于查询本机ARP缓存中IP地址和MAC地址的对应关系、添加或删除静态对应关系等。相关协议有RARP、代理ARP。NDP用于在IPv6中代替地址解析协议。

1.OSI模型和TCP/IP模型的比较



1 TCP和UDP区别，应用场景

TCP：为应用层提供可靠的、面向连接的和基于流的服务。使用超时重传、数据确认等方式来确保数据包被正确地发送至目的端，因此服务是可靠的。使用TCP协议通信的双方必须先建立TCP连接，并在内核中为该连接维持一些必要的数据结构，比如连接的状态、读写缓冲区，以及诸多定时器等。当通信结束时，双方必须关闭连接以释放这些内核数据。TCP基于流。基于流的数据没有边界（长度）限制，它源源不断地从通信的一端流入另一端。发送端可以逐个地向数据流中写入数据，接收端也可以逐个地将它们读出。

缺点：因为TCP有确认机制、三次握手机制，这些也导致TCP容易被人利用，实现DOS、DDOS、CC等攻击

UDP：则与TCP想反，它为应用层提供不可靠、无连接和基于数据报的服务。“不可靠”意味着UDP协议无法保证数据从发送端正确地传送到目的端。如果数据在途中丢失，或者目的端通过数据校验发现数据错误而将其丢弃，则UDP协议只是简单地通知应用程序发送失败。因此使用UDP协议的应用程序通常要自己处理数据确认，超时重传等逻辑。UDP是无连接的，即通信双方不保持一个长久的联系，因此应用程序每次发送数据都要明确指定接收端的地址（IP地址等信息）。基于数据报的服务，是相对基于流的服务而言的。每个UDP数据报都有一个长度，接收端必须以该长度为最小单位将其所有内容依次读出，否则数据将被截断。

UDP优点：快比TCP稍安全

缺点：不可靠，不稳定，如果网络质量不好，很容易丢包。

什么时候该使用TCP： 当对网络通讯质量有要求的时候，比如：整个数据要准确无误的传递给对方，这往往用于一些要求可靠的应用，比如HTTP、HTTPS、FTP等传输文件的协议，POP、SMTP等邮件传输的协议。

在日常生活中，常见使用TCP协议的应用如下： 浏览器，用的HTTP FlashFXP，用的FTP Outlook，用的POP、SMTP Putty，用的Telnet、SSH QQ文件传输

什么时候使用UDP：当对网络通讯质量要求不高的时候，要求速度尽可能的快，这时就可以使用UDP。用UDP协议的应用如下： QQ语音 QQ视频 TFTP。

2 浏览器输入www.baidu.com后，网络过程

事件顺序：

(1) 浏览器获取输入的域名www.baidu.com

(2) 浏览器向DNS请求解析www.baidu.com的IP地址

(3) 域名系统DNS解析出百度服务器的IP地址 （详细介绍DNS）-通过网关出去

(4) 浏览器与该服务器建立TCP连接(默认端口号80)

(5) 浏览器发出HTTP请求，请求百度首页

(6) 服务器通过HTTP响应把首页文件发送给浏览器

(7) TCP连接释放

(8) 浏览器将首页文件进行解析，并将Web页显示给用户。

涉及到的协议:

(1) 应用层：HTTP(WWW访问协议)，DNS(域名解析服务)

DNS解析域名为目的IP，通过IP找到服务器路径，客户端向服务器发起HTTP会话，然后通过运输层TCP协议封装数据包，在TCP协议基础上进行传输。

(2) 传输层：TCP(为HTTP提供可靠的数据传输)，UDP(DNS使用UDP传输)，HTTP会话会被分成报文段，添加源、目的端口；TCP协议进行主要工作。

(3)网络层：IP(IP数据数据包传输和路由选择)，ICMP(提供网络传输过程中的差错检测)，ARP(将本机的默认网关IP地址映射成物理MAC地址)为数据包选择路由，IP协议进行主要工作，相邻结点的可靠传输，ARP协议将IP地址转成MAC地址。

简单理解： 域名解析 --> 发起TCP的3次握手 --> 建立TCP连接后发起http请求 --> 服务器响应http请求，浏览器得到html代码 --> 浏览器解析html代码，并请求html代码中的资源（如js、css、图片等） --> 浏览器对页面进行渲染呈现给用户。

3 三次握手和四次挥手过程

三次握手：

第一次握手：建立连接时，客户端发送SYN包到服务端，其中包含客户端的初始序号seq=x,并进入SYN_SENT(同步已发送)状态，等待服务器确认。（其中SYN=1，ACK=0，表示这是一个TCP连接请求数据报文；序号seq=x，表明数据传输数据时第一个数据字节的序号是X）.

第二次握手：服务器收到请求后，必须确认客户的数据包。同时自己也发送一个SYN包，即SYN+ACK包，此时服务器进入SYN_RECV（同步以收到）状态。（其中确认报文段中，标识位SYN=1,ACK=1，表示这是一个TCP连接响应数据报文，并包含服务器的初始序号seq(服务器)=y,以及服务器对客户端初始序号的确认号ack(服务器)=seq（客户端）+1=x+1）.

第三次握手：客户端收到服务器的SYN+ACK 包，向服务器发送一个序列号(seq=x+1)，确认号为ack=y+1,此包发送完毕，客户端和服务器进入ESTAB_LISHED（TCP连接成功)状态，完成三次握手。

为啥三次握手：假设客户端请求建立连接，发给服务器SYN包等待服务器确认，服务器收到确认后，如果是两次握手，假设服务器给客户端在第二次握手时发送数据，数据从服务器发出，服务器任务连接已经建立，但在发送数据的过程中数据丢失，客户端认为连接没有建立，会进行重传。假设每一次发送的数据一直在丢失，客户端一直SYN，服务器就会产生多个无效连接，占用资源，这个时候服务器可能会挂掉。这就是SYN的洪水攻击

第三次握手是为了防止：如果客户端迟迟没有收到服务器返回确认报文，这时会放弃连接，重新启动一条连接请求，但问题是：服务器不知道客户端没有收到，所以他会收到两个连接，浪费连接开销。如果每次都是这样，就会浪费多个连接开销。

四次挥手：

step1：第一次挥手
首先，客户端发送一个FIN，用来关闭客户端到服务器的数据传送，然后等待服务器的确认。其中终止标志位FIN=1，序列号seq=u。

step2：第二次挥手
服务器收到这个FIN，它发送一个ACK，确认ack为收到的序号加一。

step3：第三次挥手
关闭服务器到客户端的连接，发送一个FIN给客户端。

step4：第四次挥手
客户端收到FIN后，并发回一个ACK报文确认，并将确认序号seq设置为收到序号加一。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

客户端发送FIN后，进入终止等待状态，服务器收到客户端连接释放报文段后，就立即给客户端发送确认，服务器就进入CLOSE_WAIT状态，此时TCP服务器进程就通知高层应用进程，因而从客户端到服务器的连接就释放了。此时是“半关闭状态”，即客户端不可以发送给服务器，服务器可以发送给客户端。此时，如果服务器没有数据报发送给客户端，其应用程序就通知TCP释放连接，然后发送给客户端连接释放数据报，并等待确认。客户端发送确认后，进入TIME_WAIT状态，但是此时TCP连接还没有释放，然后经过等待计时器设置的2MSL后，才进入到CLOSE状态。

为什么需要2MSL时间?

首先，MSL即Maximum Segment Lifetime，就是最大报文生存时间，是任何报文在网络上的存在的最长时间，超过这个时间报文将被丢弃。《TCP/IP详解》中是这样描述的：MSL是任何报文段被丢弃前在网络内的最长时间。RFC 793中规定MSL为2分钟，实际应用中常用的是30秒、1分钟、2分钟等。

TCP的TIME_WAIT需要等待2MSL，当TCP的一端发起主动关闭，三次挥手完成后发送第四次挥手的ACK包后就进入这个状态，等待2MSL时间主要目的是：防止最后一个ACK包对方没有收到，那么对方在超时后将重发第三次握手的FIN包，主动关闭端接到重发的FIN包后可以再发一个ACK应答包。在TIME_WAIT状态时两端的端口不能使用，要等到2MSL时间结束才可以继续使用。当连接处于2MSL等待阶段时任何迟到的报文段都将被丢弃。

.为什么是四次挥手，而不是三次或是五次、六次?
双方关闭连接要经过双方都同意。所以，首先是客服端给服务器发送FIN，要求关闭连接，服务器收到后会发送一个ACK进行确认。服务器然后再发送一个FIN，客户端发送ACK确认，并进入TIME_WAIT状态。等待2MSL后自动关闭。

总结：
（1）为了保证客户端发送的最后一个ACK报文段能够到达服务器。即最后一个确认报文可能丢失，服务器会超时重传，然后服务器发送FIN请求关闭连接，客户端发送ACK确认。一个来回是两个报文生命周期。

如果没有等待时间，发送完确认报文段就立即释放连接的话，服务器就无法重传，因此也就收不到确认，就无法按步骤进入CLOSE状态，即必须收到确认才能close。
（2）防止已经失效的连接请求报文出现在连接中。经过2MSL，在这个连续持续的时间内，产生的所有报文段就可以都从网络消失。

4 DNS域名解析过程：

步骤一：当在浏览器中输入域名按下回车键后，浏览器会检查缓存中有没有这个域名对应的解析过的IP地址。如果缓存有，解析结束。浏览器缓存域名在大小和时间上都是有限制的。缓存时间可由TTL属性来设置缓存时间太长太短都不好，太长，会导致IP解析有变化，会导致域名不能正常解析，部分用户无法访问网站。缓存时间太短，用户每次都需要重新解析一次域名。

步骤二：如果用户的浏览器中缓存没有，浏览器会查找操作系统中是否有这个域名对应的DNS解析结果。其实操作系统中也会有一个域名解析的过程，在windows中可以通过c:\windows\system32\drivers\etc\hosts文件来设置，你可以将任何域名解析到任何能够访问的IP地址。（黑客劫持域名）步骤一和步骤二都是由本机完成的。

步骤三：当机无法完成域名解析，就会真正请求域名服务器来解析这个域名了。我们怎样知道域名服务器？网络配置中都会有“DNS服务器地址”操作系统会把这个域名发送到设置中的LDNS，也就是本地的域名服务器。DNS通常都会提供给你本地互联网接入的一个DNS服务器。比如你在学校，那么这个DNS服务器一定在你们学校。WIndows中可由ipconfig查询这个地址。

步骤四：如果LDNS仍然没有解析到，就直接到Root Service域名解析器请求解析

不周五：根域名服务器返回给本地域名服务器一个所查询余的主域名服务器（gTLDServer）地址。gTLD是国际顶级域名服务器，如：.com/.cn/.org等，全球只有13台左右。

步骤六：本地域名服务器（Local DNS Server）再向上一步返回的gTLD服务器发送请求

步骤七：接收请求的gTLD服务器查找并返回此域名对应的Name Server域名服务器的地址，这个Name Server通常就是你注册的域名服务器（如你的域名供应商）

步骤八：Name Server域名服务器会查询存储的域名和IP的映射关系表，正常情况下都根据域名得到目标IP记录，连同一个TTL值返回给DNS Server域名服务器

步骤九：返回该域名对应的IP和TTL值，Local DNS Server会缓存这个域名和IP的对应关系，缓存的时间有TTL值控制。

步骤十：把解析的结果返回给用户，用户根据TTL值缓存在本地系统缓存中，域名解析过程结束。
关于DNS解析的TTL参数：

我们在配置DNS解析的时候，有一个参数常常容易忽略，就是DNS解析的TTL参数，Time To Live。TTL这个参数告诉本地DNS服务器，域名缓存的最长时间。用阿里云解析来举例，阿里云解析默认的TTL是10分钟，10分钟的含义是，本地DNS服务器对于域名的缓存时间是10分钟，10分钟之后，本地DNS服务器就会删除这条记录，删除之后，如果有用户访问这个域名，就要重复一遍上述复杂的流程。

5 HTTP协议

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP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。

主要特点：

1、简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

2、灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

3.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

4.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。
5、支持B/S及C/S模式。

URI和URL的区别

URI，是uniform resource identifier，统一资源标识符，用来唯一的标识一个资源。

Web上可用的每种资源如HTML文档、图像、视频片段、程序等都是一个来URI来定位的
URI一般由三部组成：
①访问资源的命名机制
②存放资源的主机名
③资源自身的名称，由路径表示，着重强调于资源。

URL是uniform resource locator，统一资源定位器，它是一种具体的URI，即URL可以用来标识一个资源，而且还指明了如何locate这个资源。

URL是Internet上用来描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上，特别是著名的Mosaic。
采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL一般由三部组成：
①协议(或称为服务方式)
②存有该资源的主机IP地址(有时也包括端口号)
③主机资源的具体地址。如目录和文件名等

URN，uniform resource name，统一资源命名，是通过名字来标识资源，比如mailto:java-net@java.sun.com。

URI是以一种抽象的，高层次概念定义统一资源标识，而URL和URN则是具体的资源标识的方式。URL和URN都是一种URI。笼统地说，每个 URL 都是 URI，但不一定每个 URI 都是 URL。这是因为 URI 还包括一个子类，即统一资源名称 (URN)，它命名资源但不指定如何定位资源。上面的 mailto、news 和 isbn URI 都是 URN 的示例。

在Java的URI中，一个URI实例可以代表绝对的，也可以是相对的，只要它符合URI的语法规则。而URL类则不仅符合语义，还包含了定位该资源的信息，因此它不能是相对的。
在Java类库中，URI类不包含任何访问资源的方法，它唯一的作用就是解析。
相反的是，URL类可以打开一个到达资源的流。

HTTP之请求消息Request

客户端发送一个HTTP请求到服务器的请求消息包括以下格式：

请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。

请求消息结构.p

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本。
Get请求例子，使用Charles抓取的request：

GET /562f25980001b1b106000338.jpg HTTP/1.1
Host　img.mukewang.com
User-Agent　Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36
Accept　image/webp,image/,/*;q=0.8
Referer http://www.imooc.com/
Accept-Encoding gzip, deflate, sdch
Accept-Language zh-CN,zh;q=0.8

第一部分：请求行，用来说明请求类型,要访问的资源以及所使用的HTTP版本.

GET说明请求类型为GET,[/562f25980001b1b106000338.jpg]为要访问的资源，该行的最后一部分说明使用的是HTTP1.1版本。

第二部分：请求头部，紧接着请求行（即第一行）之后的部分，用来说明服务器要使用的附加信息

从第二行起为请求头部，HOST将指出请求的目的地.User-Agent,服务器端和客户端脚本都能访问它,它是浏览器类型检测逻辑的重要基础.该信息由你的浏览器来定义,并且在每个请求中自动发送等等

第三部分：空行，请求头部后面的空行是必须的

即使第四部分的请求数据为空，也必须有空行。

第四部分：请求数据也叫主体，可以添加任意的其他数据。

这个例子的请求数据为空。

POST请求例子，使用Charles抓取的request：

POST / HTTP1.1
Host:www.wrox.com
User-Agent:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)
Content-Type:application/x-www-form-urlencoded
Content-Length:40
Connection: Keep-Alive

name=Professional%20Ajax&publisher=Wiley
第一部分：请求行，第一行明了是post请求，以及http1.1版本。
第二部分：请求头部，第二行至第六行。
第三部分：空行，第七行的空行。
第四部分：请求数据，第八行。

HTTP之响应消息Response

一般情况下，服务器接收并处理客户端发过来的请求后会返回一个HTTP的响应消息。

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。

http响应消息格式.jpg

例子

HTTP/1.1 200 OK
Date: Fri, 22 May 2009 06:07:21 GMT
Content-Type: text/html; charset=UTF-8

第一部分：状态行，由HTTP协议版本号， 状态码， 状态消息 三部分组成。

第一行为状态行，（HTTP/1.1）表明HTTP版本为1.1版本，状态码为200，状态消息为（ok）

第二部分：消息报头，用来说明客户端要使用的一些附加信息

第二行和第三行为消息报头，
Date:生成响应的日期和时间；Content-Type:指定了MIME类型的HTML(text/html),编码类型是UTF-8

第三部分：空行，消息报头后面的空行是必须的

第四部分：响应正文，服务器返回给客户端的文本信息。

空行后面的html部分为响应正文。

HTTP之状态码
状态代码有三位数字组成，第一个数字定义了响应的类别，共分五种类别:

1xx：指示信息--表示请求已接收，继续处理

2xx：成功--表示请求已被成功接收、理解、接受

3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现

5xx：服务器端错误--服务器未能实现合法的请求

以下是 HTTP 请求/响应的步骤：

1、客户端连接到Web服务器

一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。例如，http://www.oakcms.cn。

2、发送HTTP请求

通过TCP套接字，客户端向Web服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据4部分组成。

3、服务器接受请求并返回HTTP响应

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据4部分组成。

4、释放连接TCP连接

若connection 模式为close，则服务器主动关闭TCP连接，客户端被动关闭连接，释放TCP连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;

5、客户端浏览器解析HTML内容

客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

例如：在浏览器地址栏键入URL，按下回车之后会经历以下流程：

1、浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;

2、解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立TCP连接;

3、浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求，该请求报文作为 TCP 三次握手的第三个报文的数据发送给服务器;

4、服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;

5、释放 TCP连接;

6、浏览器将该 html 文本并显示内容;

GET和POST区别：

GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456. POST方法是把提交的数据放在HTTP包的Body中.

GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制.

GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值。

GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码.

HTTP协议【详解】——经典面试题

http请求由三部分组成，分别是：请求行、消息报头、请求正文

HTTP（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于TCP的连接方式，HTTP1.1版本中给出一种持续连接的机制，绝大多数的Web开发，都是构建在HTTP协议之上的Web应用。

1、常用的HTTP方法有哪些?

GET： 用于请求访问已经被URI（统一资源标识符）识别的资源，可以通过URL传参给服务器。
POST：用于传输信息给服务器，主要功能与GET方法类似，但一般推荐使用POST方式。
PUT： 传输文件，报文主体中包含文件内容，保存到对应URI位置。
HEAD： 获得报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URI是否有效。
DELETE：删除文件，与PUT方法相反，删除对应URI位置的文件。
OPTIONS：查询相应URI支持的HTTP方法。

2、GET方法与POST方法的区别

区别一:
get重点在从服务器上获取资源，post重点在向服务器发送数据;
区别二:
get传输数据是通过URL请求，以field（字段）= value的形式，置于URL后，并用"?"连接，多个请求数据间用"&"连接，如http://127.0.0.1/Test/login.action?name=admin&password=admin，这个过程用户是可见的;
post传输数据通过Http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的;
区别三:
Get传输的数据量小，因为受URL长度限制，但效率较高;
Post可以传输大量数据，所以上传文件时只能用Post方式;
区别四:

get是不安全的，因为URL是可见的，可能会泄露私密信息，如密码等；
post较get安全性较高；
区别五：
get方式只能支持ASCII字符，向服务器传的中文字符可能会乱码。
post支持标准字符集，可以正确传递中文字符。

3、HTTP请求报文与响应报文格式

请求报文包含三部分：
a、请求行：包含请求方法、URI、HTTP版本信息
b、请求首部字段
c、请求内容实体
响应报文包含三部分：
a、状态行：包含HTTP版本、状态码、状态码的原因短语
b、响应首部字段
c、响应内容实体

4、常见的HTTP相应状态码

返回的状态
1xx：指示信息--表示请求已接收，继续处理
2xx：成功--表示请求已被成功接收、理解、接受
3xx：重定向--要完成请求必须进行更进一步的操作
4xx：客户端错误--请求有语法错误或请求无法实现
5xx：服务器端错误--服务器未能实现合法的请求

200：请求被正常处理
204：请求被受理但没有资源可以返回
206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。
301：永久性重定向
302：临时重定向
303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上
304：发送附带条件的请求时，条件不满足时返回，与重定向无关
307：临时重定向，与302类似，只是强制要求使用POST方法
400：请求报文语法有误，服务器无法识别
401：请求需要认证
403：请求的对应资源禁止被访问
404：服务器无法找到对应资源
500：服务器内部错误
503：服务器正忙

5、HTTP1.1版本新特性

a、默认持久连接节省通信量，只要客户端服务端任意一端没有明确提出断开TCP连接，就一直保持连接，可以发送多次HTTP请求

b、管线化，客户端可以同时发出多个HTTP请求，而不用一个个等待响应

c、断点续传原理

6、常见HTTP首部字段

a、通用首部字段（请求报文与响应报文都会使用的首部字段）

Date：创建报文时间
Connection：连接的管理
Cache-Control：缓存的控制
Transfer-Encoding：报文主体的传输编码方式

b、请求首部字段（请求报文会使用的首部字段）

Host：请求资源所在服务器
Accept：可处理的媒体类型
Accept-Charset：可接收的字符集
Accept-Encoding：可接受的内容编码
Accept-Language：可接受的自然语言

c、响应首部字段（响应报文会使用的首部字段）

Accept-Ranges：可接受的字节范围
Location：令客户端重新定向到的URI
Server：HTTP服务器的安装信息

d、实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）

Allow：资源可支持的HTTP方法
Content-Type：实体主类的类型
Content-Encoding：实体主体适用的编码方式
Content-Language：实体主体的自然语言
Content-Length：实体主体的的字节数
Content-Range：实体主体的位置范围，一般用于发出部分请求时使用

7、HTTP的缺点与HTTPS

a、通信使用明文不加密，内容可能被窃听
b、不验证通信方身份，可能遭到伪装
c、无法验证报文完整性，可能被篡改

HTTPS就是HTTP加上加密处理（一般是SSL安全通信线路）+认证+完整性保护

8、HTTP优化

利用负载均衡优化和加速HTTP应用

利用HTTP Cache来优化网站

9、Http协议首部字段?

a、请求行

请求的第一行是"方法、URL、协议/版本"：

POST http://xg.mediportal.com.cn/health/sms/verify/telephone HTTP/1.1

以上代码中"POST"代表请求方法，"http://xg.mediportal.com.cn/health/sms/verify/telephone"表示URI，"HTTP/1.1代表协议和协议的版本。

根据HTTP标准，HTTP请求可以使用多种请求方法。例如：HTTP1.1目前支持7种请求方法：GET、POST、HEAD、OPTIONS、PUT、DELETE和TARCE。

GET	**请求获取由Request-URI所标识的资源**
POST	在Request-URI所标识的资源后附加新的数据
HEAD	请求获取由Request-URI所标识的资源的响应消息报头
OPTIONS	请求查询服务器的性能，或查询与资源相关的选项和需求
PUT	请求服务器存储一个资源，并用Request-URI作为其标识
DELETE	请求服务器删除由Request-URI所标识的资源
TRACE	请求服务器回送收到的请求信息，主要用语测试或诊断

b、请求头（请求头包含许多有关的客户端环境和请求正文的有用信息。例如，请求头可以声明浏览器所用的语言，请求正文的长度等）

-

Content-Type	**是返回消息中非常重要的内容，表示后面的文档属于什么MIME类型。Content-Type: [type]/[subtype]; parameter。例如最常见的就是text/html，它的意思是说返回的内容是文本类型，这个文本又是HTML格式的。原则上浏览器会根据Content-Type来决定如何显示返回的消息体内容**
Host	指定请求资源的Intenet主机和端口号，必须表示请求url的原始服务器或网关的位置。HTTP/1.1请求必须包含主机头域，否则系统会以400状态码返回
Accept	浏览器可接受的MIME类型
Accept-Charset	浏览器可接受的字符集
Accept-Encoding	浏览器能够进行解码的数据编码方式，比如gzip。Servlet能够向支持gzip的浏览器返回经gzip编码的HTML页面。许多情形下这可以减少5到10倍的下载时间
Accept-Language	浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到
Authorization	授权信息，通常出现在对服务器发送的WWW-Authenticate头的应答中
Connection	表示是否需要持久连接。如果Servlet看到这里的值为“Keep- Alive”，或者看到请求使用的是HTTP1.1（HTTP 1.1默认进行持久连接），它就可以利用持久连接的优点，当页面包含多个元素时（例如Applet，图片），显著地减少下载所需要的时间。要实现这一点，Servlet需要在应答中发送一个Content-Length头，最简单的实现方法是：先把内容写入ByteArrayOutputStream，然后在正式写出内容之前计算它的大小
Content-Length	表示请求消息正文的长度
Cookie	这是最重要的请求头信息之一
From	请求发送者的email地址，由一些特殊的Web客户程序使用，浏览器不会用到它
Host	初始URL中的主机和端口
If-Modified-Since	只有当所请求的内容在指定的日期之后又经过修改才返回它，否则返回304“Not Modified”应答
Pragma	指定“no-cache”值表示服务器必须返回一个刷新后的文档，即使它是代理服务器而且已经有了页面的本地拷贝
Referer	包含一个URL，用户从该URL代表的页面出发访问当前请求的页面
User-Agent	浏览器类型，如果Servlet返回的内容与浏览器类型有关则该值非常有用
UA-Pixels，UA-Color，UA-OS，UA-CPU	由某些版本的IE浏览器所发送的非标准的请求头，表示屏幕大小、颜色深度、操作系统和CPU类型

常见的MIME类型如下：

- text/html： HTML格式
- text/plain：纯文本格式
- text/xml： XML格式
- image/gif：gif图片格式
- image/jpeg：jpg图片格式
- image/png：png图片格式

  以application开头的媒体格式类型：
- application/xhtml+xml：XHTML格式
- application/xml ： XML数据格式
- application/atom+xml：Atom XML聚合格式
- application/json ： JSON数据格式
- application/pdf ：pdf格式
- application/msword： Word文档格式
- application/octet-stream： 二进制流数据（如常见的文件下载）
- application/x-www-form-urlencoded：

  中默认的encType，form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）

  另外一种常见的媒体格式是上传文件之时使用的：
- multipart/form-data： 需要在表单中进行文件上传时，就需要使用该格式

（3）请求正文

请求头和请求正文之间是一个空行，这个行非常重要，它表示请求头已经结束，接下来的是请求正文。请求正文中可以包含客户提交的查询字符串信息：

telephone=15527177736&userType=1&

10、http响应格式

HTTP应答与HTTP请求相似，HTTP响应也由3个部分构成，分别是：

1、状态行

2、响应头(Response Header)

3、响应正文

```
HTTP/1.1 200 OK  //状态行
Server: nginx
Date: Tue, 31 May 2016 02:09:24 GMT
Content-Type: application/json;charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With,access_token,access-token,content-type,multipart/form-data,application/x-www-form-urlencoded
Access-Control-Allow-Methods: GET,POST,OPTIONS
Content-Length: 49
```

{"resultCode":1,"resultMsg":"手机号未注册"} //正文

（1）状态行

由协议版本、数字形式的状态代码、及相应的状态描述，各元素之间以空格分隔。

状态代码：

状态代码由3位数字组成，表示请求是否被理解或被满足。

状态描述：

状态描述给出了关于状态代码的简短的文字描述。

状态代码的第一个数字定义了响应的类别，后面两位没有具体的分类。

第一个数字有五种可能的取值：

- 1xx: 指示信息—表示请求已接收，继续处理。

- 2xx: 成功—表示请求已经被成功接收、理解、接受。

- 3xx: 重定向—要完成请求必须进行更进一步的操作。

- 4xx: 客户端错误—请求有语法错误或请求无法实现。

- 5xx: 服务器端错误—服务器未能实现合法的请求。

状态代码 状态描述 说明

200 OK 客户端请求成功

400 Bad Request 由于客户端请求有语法错误，不能被服务器所理解。

401 Unauthonzed 请求未经授权。这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden 服务器收到请求，但是拒绝提供服务。服务器通常会在响应正文中给出不提供服务的原因

404 Not Found 请求的资源不存在，例如，输入了错误的URL。

500 Internal Server Error 服务器发生不可预期的错误，导致无法完成客户端的请求。

503 Service Unavailable 服务器当前不能够处理客户端的请求，在一段时间之后，服务器可能会恢复正常

（2）响应头

响应头可能包括：

Location：

Location响应报头域用于重定向接受者到一个新的位置。例如：客户端所请求的页面已不存在原先的位置，为了让客户端重定向到这个页面新的位置，服务 器端可以发回Location响应报头后使用重定向语句，让客户端去访问新的域名所对应的服务器上的资源。当我们在JSP中使用重定向语句的时候，服务器端向客户端发回的响应报头中，就会有Location响应报头域。

Server：

Server响应报头域包含了服务器用来处理请求的软件信息。它和User-Agent请求报头域是相对应的，前者发送服务器端软件的信息，后者发送客户 端软件(浏览器)和操作系统的信息。下面是Server响应报头域的一个例子：Server: Apache-Coyote/1.1

WWW-Authenticate：

WWW-Authenticate响应报头域必须被包含在401(未授权的)响应消息中，这个报头域和前面讲到的Authorization请求报头域是 相关的，当客户端收到401响应消息，就要决定是否请求服务器对其进行验证。如果要求服务器对其进行验证，就可以发送一个包含了 Authorization报头域的请求，下面是WWW-Authenticate响应报头域的一个例子：WWW-Authenticate: Basic realm="Basic Auth Test!"

从这个响应报头域，可以知道服务器端对我们所请求的资源采用的是基本验证机制。

Content-Encoding：**

Content-Encoding实体报头域被使用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容编码，因而要获得Content- Type报头域中所引用的媒体类型，必须采用相应的解码机制。Content-Encoding主要用语记录文档的压缩方法，下面是它的一个例子： Content-Encoding: gzip。如果一个实体正文采用了编码方式存储，在使用之前就必须进行解码。

Content-Language：

Content-Language实体报头域描述了资源所用的自然语言。Content-Language允许用户遵照自身的首选语言来识别和区分实体。 如果这个实体内容仅仅打算提供给丹麦的阅读者，那么可以按照如下的方式设置这个实体报头域：Content-Language: da。

如果没有指定Content-Language报头域，那么实体内容将提供给所以语言的阅读者。

Content-Length：**

Content-Length实体报头域用于指明正文的长度，以字节方式存储的十进制数字来表示，也就是一个数字字符占一个字节，用其对应的ASCII码存储传输。

要注意的是：这个长度仅仅是表示实体正文的长度，没有包括实体报头的长度。

Content-Type :

Content-Type实体报头域用语指明发送给接收者的实体正文的媒体类型。例如：

Content-Type: text/html;charset=ISO-8859-1

Content-Type: text/html;charset=GB2312

Last-Modified :

Last-Modified实体报头域用于指示资源最后的修改日期及时间。

Expires :

Expires实体报头域给出响应过期的日期和时间。通常，代理服务器或浏览器会缓存一些页面。当用户再次访问这些页面时，直接从缓存中加载并显示给用 户，这样缩短了响应的时间，减少服务器的负载。为了让代理服务器或浏览器在一段时间后更新页面，我们可以使用Expires实体报头域指定页面过期的时间。当用户又一次访问页面时，如果Expires报头域给出的日期和时间比Date普通报头域给出的日期和时间要早(或相同)，那么代理服务器或浏览器就 不会再使用缓存的页面而是从服务器上请求更新的页面。不过要注意，即使页面过期了，也并不意味着服务器上的原始资源在此时间之前或之后发生了改变。

Expires实体报头域使用的日期和时间必须是RFC 1123中的日期格式，例如：

Expires: Thu, 15 Sep 2005 16:00:00 GMT

HTTP1.1的客户端和缓存必须将其他非法的日期格式(也包括0)看作已过期。例如，为了让浏览器不要缓存页面，我们也可以利用Expires实体报头 域，设置它的值为0，如下(JSP)：
response.setDateHeader("Expires",0);

11、http缓存

缓存介绍：
缓存就是让同一份资源不再发起请求，降低网络压力和资源数量，从而让网页加载更快。
网络请求不稳定加剧了网页加载的不稳定性，所以需要缓存。
一般来说，js、css、img等静态资源可以被缓存
http缓存策略：

强制缓存（Cache-Control)

cache-control在response-headers中，用于对缓存资源的控制，该属性可以控制强制缓存的逻辑，例如：Cache-Control: max-age=31536000(单位是秒)。
当再次请求时，先检查过期与否，若没有过期，则直接从缓存里拿数据，反之再次向服务器请求
Cache-Control的值：

max-age
no-cache（不用本地缓存）
no-store（不用本地和服务端的缓存）
在现代浏览器中兼容Expires和Cache-Control，不过优先使用Cache-Control

协商缓存（对比缓存）

协商缓存是服务端缓存策略（服务端判断该资源能否被缓存，而不是资源缓存在服务端），判断依据是客户端和服务端的资源标识是否一样，若一样则返回304，否则返回200和最新的资源、资源标识(资源标识在response headers中)
资源标识：
Last-Modified（客户端再次请求时，request headers带着if-Modified-Since） 资源的最后修改时间

Etag 资源的唯一标识（客户端再次请求时，request headers带着if-None-Match）（一个字符串）
当两种资源标识共同存在时，优先使用Etag，因为Last-Modified只能精确到秒级，而Etag唯一

刷新操作方式，对缓存的影响：
正常操作：地址栏输入URL，跳转链接，前进后退等，强制缓存和协商缓存均有效
手动刷新：F5，点击刷新按钮，右键菜单刷新，强制缓存失效，协商缓存有效
强制刷新：ctrl+F5，强制缓存和协商缓存均无效

划分网络类别

A类：第一位必须为off（0）0~127

B类：第一位必须为1，第二位必须为0 128~191

C类：第一二位必须为1，第三位必须为0 192~223

D类:224~239

E类:240~255

各个层的PDU(协议数据单元)

物理层的PDU是**数据位**（bit）；

数据链路层的PDU是**数据帧**（frame）；

网络层的PDU是**数据包**（packet）；

传输层的PDU是**数据段**（segment）；

其他更高层次的PDU（如会话层表示层应用层）是数据（data）