001.计算机网络基础

网络编程的目的就是直接或间接地通过网络协议与其他计算机进行通信。
在 Java 语言中包含网络编程所需要的各种类，编程人员只需要创建这些类的对象，调用相应的方法，就可以进行网络应用程序的编写。

要进行网络程序的编写，编程人员需要对下面5个方面的基础知识有一定的了解：

1. 计算机网络
2. 网络分类
3. 网络编程模式
4. 套接字
5. 网络通信协议

稍微具体的内容：

1. 计算机网络

指将地理位置不同的具有独立功能的多台计算机及其外部设备，
通过通信线路连接起来，在网络操作系统，网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统。

2. 网络分类

按照地理范围主要将网络分为局域网、城域网、广域网和因特网。

(1) 局域网（LocalArea Network）简称 LAN，是一种在小范围内实现的计算机网络，
一般在一个建筑物内或者一个工厂、一个事业单位内部独有，范围较小。

(2) 城域网（Metropolitan Area Network）简称为 MAN，一般是一个城市内部组建的计算机信息网络，提供全市的信息服务。

(3) 广域网（Wide Area Network）简称为 WAN，它的范围很广，可以分布在一个省、一个国家或者几个国家。

(4) 因特网（Internet）则是由无数的 LAN 和 WAN 组成的。

3. 网络编程模式

在网络通信中主要有两种模式的通信方式：
一种是客户机/服务器（Client/Server）模式，简称为 C/S 模式；
另一种是浏览器/服务器（Browser/Server）模式，简称 B/S 模式。

Client/Server 模式
客户机、服务器以及网络三者之间的关系图，使用这种模式的程序很多，例如很多读者喜欢玩的网络游戏，

需要在本机上安装一个客户端，服务器运行在游戏开发公司的机房。

使用 C/S 模式的程序，在开发时需要分别针对客户端和服务器端进行专门开发。
这种开发模式的优势在于由于客户端是专门开发的，表现力会更强，缺点就是通用性差，
也就是说一种程序的客户端只能和对应的服务器端进行通信，不能和其他的服务器端进行通信，
在实际维护中，也需要维护专门的客户端和服务器端，维护的压力较大。

Browser/Server 模式
对于很多程序，运行时不需要专门的客户端，而是使用通用的客户端，例如使用浏览器。
用户使用浏览器作为客户端的这种模式叫作浏览器/服务器模式。
使用这种模式开发程序时只需开发服务器端即可，开发的压力较小，不需要维护客户端。
但是对浏览器的限制比较大，表现力不强。

4. 套接字

在网络上很多应用程序都是采用客户端/服务器（C/S）的模式，
实现网络通信必须将两台计算机连接起来建立一个双向的通信链路，
这个双向通信链路的每一端在Java的网络编程中，我们又称之为一个套接字（Socket）。
套接字（Socket）是一个抽象出来的概念代表在端上的通信链路。

5. 网络协议

网络上的计算机之间要通信，就必须有一些约定，这些约定被称为网络通信协议。
就像我们说话用某种语言一样，在网络上的各台计算机之间也有一种语言，这就是通信协议，
不同的计算机之间必须使用相同的通信协议才能进行通信。

通信协议是由三个要素组成：
(1) 语义。语义是解释控制信息每个部分的意义。它规定了需要发出何种控制信息，以及完成的动作与做出什么样的响应。
(2) 语法。语法是用户数据与控制信息的结构与格式，以及数据出现的顺序。
(3) 时序。时序是对事件发生顺序的详细说明。（也可称为“同步”）。
人们形象地把这三个要素描述为：语义表示要做什么，语法表示要怎么做，时序表示做的顺序。

通信协议是网络上的设备进行网络通信的基础，这些具体的协议是非常多的，
在Java的网络编程这部分知识的学习中，我们必须要了解的有关的主要协议，TCP/IP 和 HTTP

(1) TCP/IP（Transmission Control Protocol/Internet Protocol，传输控制协议/网际协议）：

TCP/IP协议是一个协议集合。大家叫的时候方便说，所以统称为TCP/IP。

在 TCP/IP 中包含一系列用于处理数据通信的协议：

- TCP（传输控制协议）- 应用程序之间通信
- UDP（用户数据包协议）- 应用程序之间的简单通信
- IP（网际协议）- 计算机之间和计算机网络之间的通信
- ICMP（因特网消息控制协议）- 针对错误和状态
- DHCP（动态主机配置协议）- 针对动态寻址

IP协议：

IP（Internet Protocol）协议，又称网际协议，是TCP/IP协议的核心。

它负责Internet上网络之间的通信，

并规定了将数据报从一个网络传输到另一个网络所应遵循的规则。

具体来说，IP协议不但定义了数据传输时的基本单元和格式，

还定义了数据报的递交方法和路由选择。

此外，在TCP/IP网络中，**主机之间进行通信所必需的地址**，也是通过IP协议来实现的。

IP地址

要想使网络中的计算机能够进行通信，必须为每台计算机指定一个标识号，

通过这个标识号来指定接受数据的计算机或者发送数据的计算机。

在TCP/IP协议中，这个标识号就是IP地址，它可以唯一标识一台计算机，

目前，IP地址广泛使用的版本是IPv4，它是由4个字节大小的二进制数来表示，如：

00001010000000000000000000000001。

由于二进制形式表示的IP地址非常不便记忆和处理，因此通常会将IP地址写成十进制的形式，

每个字节用一个十进制数字(0-255)表示，数字间用符号“.”分开，如

“192.168.1.100”。

随着计算机网络规模的不断扩大，对IP地址的需求也越来越多，

IPV4这种用4个字节表示的IP地址面临枯竭，因此IPv6 便应运而生了，

IPv6使用16个字节表示IP地址，它所拥有的地址容量是IPv4的接近8×10^28倍，

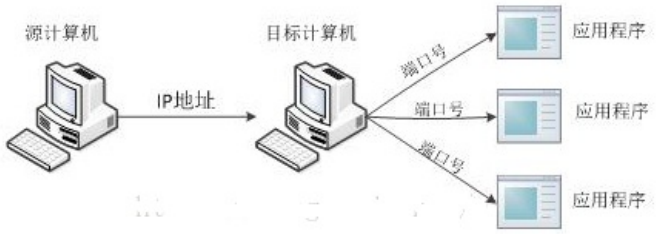
达到2^128-1个（全零的不能用），这样就解决了网络地址资源数量不够的问题。

端口号

通过IP地址可以连接到指定计算机，但如果想访问目标计算机中的某个应用程序，还需要指定端口号。

在计算机中，不同的应用程序是通过端口号区分的。
端口号是用两个字节（16位的二进制数）表示的，它的取值范围是0~65535，
其中，0~1023之间的端口号用于一些知名的网络服务和应用，
用户的普通应用程序需要使用1024以上的端口号，从而避免端口号被另外一个应用或服务所占用。

接下来通过一个图例来描述IP地址和端口号的作用，如下图所示。



从上图中可以清楚地看到，位于网络中一台计算机可以通过IP地址去访问另一台计算机，并通过端口号访问目标计算机中的某个应用程序。

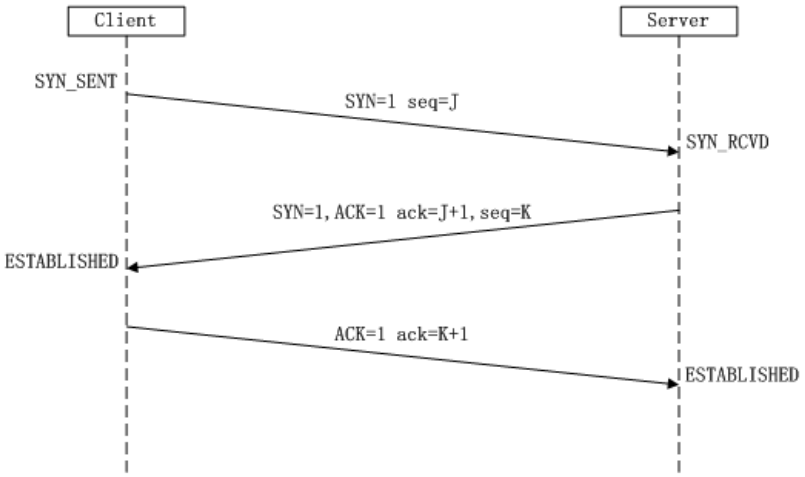
TCP协议：

TCP 用于应用程序之间的通信。
当应用程序希望通过 TCP 与另一个应用程序通信时，它会发送一个通信请求。
这个请求必须被送到一个确切的地址。在双方“三次握手”之后，
TCP 将在两个应用程序之间建立一个全双工（full-duplex）的通信。
这个全双工的通信将占用两个计算机之间的通信线路，直到它被一方或双方关闭为止。

TCP三次握手四次挥手

TCP三次握手

所谓三次握手（Three-Way Handshake）即建立TCP连接，就是指建立一个TCP连接时，
需要客户端和服务端总共发送3个包以确认连接的建立。
在socket编程中，这一过程由客户端执行connect来触发，整个流程如下图所示：



ACK：确认标志。确认值(Acknowledgement)，为1便是确认连接。

SYN：同步标志。表明同步序列编号栏有效。

FIN：结束标志。

ack：确认编号(Acknowledgement Number)，即接收到的上一次远端主机传来的seq然后+1，再发送给远端主机。提示远端主机已经成功接收上一次所有数据。

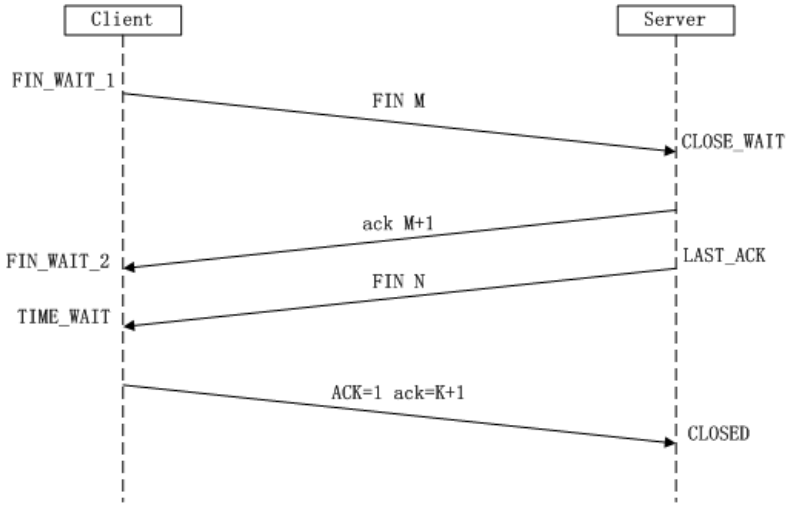
1）第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN_SENT状态，等待Server确认。

2）第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN_RCVD状态。

3）第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。

TCP四次挥手

所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发，整个流程如下图所示：



由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。

1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。

2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与

SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。
3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK状态。
4）第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

为什么建立连接是三次握手，而关闭连接却是四次挥手呢？
这是因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送。

UDP协议：
Internet 协议集支持一个无连接的传输协议，该协议称为用户数据报协议（UDP，User Datagram Protocol）。
使用UDP协议，传输数据之前源端和终端不建立连接，
当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。
在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；
在接收端，UDP把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

UDP与TCP简单对比：
时间上，UDP，无连接，**不存在建立连接需要的时延，TCP建立链接会产生时延。**
空间上，TCP需要在端系统中**维护连接**状态，需要一定的开销。
此连接状态包括接收和发送缓存，拥塞控制参数和序号与确认号的参数。
UCP不维护连接状态，也不跟踪这些参数，开销小。空间和时间上都具有优势。
可靠性上，TCP面向链接的通信可靠性高，UDP可靠性差。

（2） HTTP协议
HTTP是HyperText Transfer Protocol的缩写，翻译过来就是超文本传输协议，
它是一种用于分布式、协作式和**web应用系统**的应用层协议。

我们以建行官网为例：在浏览器里输入http://www.ccb.com，然后猛按回车，呈现在你面前的，将是建行官网的首页了。作为一个开发者，我想你有必要去了解这一系列的处理流程，在这期间，**浏览器和服务器到底是如何打交道的？服务器又是如何处理的？浏览器又是如何将网页显示给用户的呢？.....，**坦白讲，要想彻彻底底的弄清楚以上每个疑惑和处理细节，至少需要十本书的厚度，所谓“底层无极限”！

既然前面讲了TCP/UDP的网络通信协议，而且他们的使用是非常广泛的，那为啥有多出个http协议来呢？

UDP协议具有不可靠性和不安全性，显然这很难满足web应用的需要。

而TCP协议是基于连接和三次握手的，虽然具有可靠性，但任然具有一定的缺陷。

试想一下，B/S架构的网站，十万人同时在线也是很平常的事儿。

如果十万个客户端和服务器一直保持连接状态，那服务器那端有这么多带宽来分配吗？

这就衍生出了http协议。基于TCP的可靠性连接。

通俗点说，就是在请求之后，服务器端立即关闭连接、释放资源。

这样既保证了资源可用，也吸取了TCP的可靠性的优点。

HTTP协议由来的概况：

HTTP的发展是由蒂姆·伯纳斯-李在1989年在欧洲核子研究组织（CERN）所发起。

HTTP的标准制定由万维网协会（World Wide Web Consortium，W3C）和

互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，

最终发布了一系列的RFC，其中最著名的是1999年6月公布的 RFC 2616，

定义了HTTP协议中，现今广泛使用的一个版本——HTTP 1.1。

2014年12月，互联网工程任务组（IETF）的Hypertext Transfer Protocol

Bis（httpbis）工作小组

将HTTP 2.0标准提议递交至IESG进行讨论，于2015年2月17日被批准。

HTTP 2.0标准于2015年5月以RFC 7540正式发表，取代HTTP 1.1成为HTTP的实现标准。

Bis：

工作组名字中的“bis”来自拉丁语中表示“二”的副词，

Bis通常被IETF用作名字的后缀来以表示标准的升级或者一些二次工作，

比如这里是针对HTTP1.1。

IESG：

Internet Engineering Steering Group，互联网工程指导小组，

它是IETF的上级单位，负责IETF活动和标准制定程序的技术管理工作，

核准或纠正IETF各工作组的研究成果，有对工作组的设立终结权，

确保非工作组草案在成为请求注解文件（RFC）时的准确性。

HTTP工作原理

HTTP协议定义Web客户端如何从Web服务器请求Web页面，以及服务器如何把Web页面传送给客户端。

HTTP协议采用了请求/响应模型。

客户端向服务器发送一个请求报文，请求报文包含请求行、请求头部、空行和请求数据。
服务器收到客户端的请求会响应，响应的内容包括状态行、响应头部、空行和响应数据。

以下是 HTTP 请求/响应的步骤：

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，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。

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. 服务器接收请求并返回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

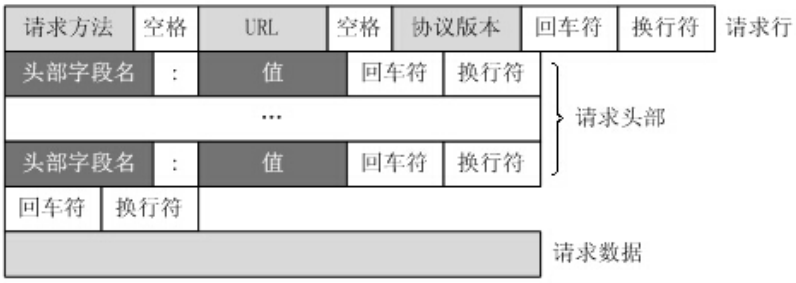
   客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。

   然后解析每一个响应头，响应头告知HTML文档的若干特殊信息和文档的字符集。

   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。

HTTP请求报文（就是浏览器首先发送给服务器的信息）

HTTP请求格式



HTTP请求方法

HTTP/1.1协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源：

GET

向指定的资源发出“显示”请求。

HEAD

类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头。

POST

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。

数据被包含在请求数据中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。

PUT

向指定资源位置上传其最新内容。

DELETE

请求服务器删除Request-URI所标识的资源。

TRACE

回显服务器收到的请求，主要用于测试或诊断。

OPTIONS

这个方法可使服务器传回该资源所支持的所有HTTP请求方法。

用'*'来代替资源名称，向Web服务器发送OPTIONS请求，可以测试服务器功能是否正常运作。

CONNECT

HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

通常用于SSL加密服务器的链接（经由非加密的HTTP代理服务器）。

注意事项：

方法名称是区分大小写的。

HTTP服务器至少应该实现GET和HEAD方法，其他方法都是可选的。

URL（Uniform Resource Locator）

超文本传输协议（HTTP）的统一资源定位符将从因特网获取信息的基本元素包括在一个简单的地址中。

URL 的具体语法格式如下所示

协议://用户名:密码@子域名.域名.顶级域名:端口号/目录/文件名.文件后缀?参数=值&参数2=值2....#标志

下面是一些简单的 URL 示例:

http://www.sun.com/　　协议名：//主机名

http://localhost:8080/Test/admin/login.jsp 协议名://机器名：端口号/文件名

头部：

1、Accept，浏览器端能够处理的信息内容类型。

Accept: 文件类型使用的MIME值，*/* 代表任意类型

MIME (Multipurpose Internet Mail Extensions) 是描述消息内容类型的因特网标准。

不用记，用哪个查那个，百度：MIME 参考手册

注意：

docx,application/vnd.openxmlformats-officedocument.wordprocessingml.document

xlsx,application/vnd.openxmlformats-officedocument.spreadsheetml.sheet

pptx,application/vnd.openxmlformats-officedocument.presentationml.presentation

ie会把 jpg、jpeg翻译成image/pjpeg，png翻译成image/x-png

q是权重系数，范围 0 =< q <= 1，q 值越大，请求越倾向于获得其“;”之前的类型表示的内容，

若没有指定 q 值，则默认为1，若被赋值为0，则用于提醒服务器哪些是浏览器不接受的内容类型。

2、Accept-Encoding， 浏览器能够处理的的压缩编码。

通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate），（注意：这不是指字符编码）。

例如： Accept-Encoding:gzip, deflate, br

3、Accept-Language， 浏览器当前设置的语言。

语言跟字符集的区别：中文是语言，中文有多种字符集，比如big5，gb2312，gbk等等；

4、Accept-Charset：浏览器能够显示的字符集

5、Connection：浏览器与服务器的连接类型

例如：Connection: keep-alive 当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，

如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。

例如： Connection: close 代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭。

当客户端再次发送Request，需要重新建立TCP连接。

6、Host，发送请求的页面的域名。

7、Referer，发送请求的页面的URI。当浏览器向web服务器发送请求的时候，一般会带上Referer，

告诉服务器我是从哪个页面链接过来的，服务器借此可以获得一些信息用于处理。

8、User-Agent，浏览器的用户代理字符串。告诉HTTP服务器， 客户端使用的操作系统和浏览器的名称和版本。

9、Cookie，用来存储一些用户信息以便让服务器辨别用户身份的。

10、Cache-Control，指明当前资源的有效期，控制浏览器是否直接从浏览器缓存取数据，还是重新发请求到服务器获取数据。

常看到的：max-age=0，也就是这些资源在浏览器缓存中，保存的时间，值的单位是秒。

再看看HTTP响应消息（服务器返回给浏览器的）：

HTTP响应格式

HTTP状态码

所有HTTP响应的第一行都是状态行，依次是当前HTTP版本号，

3位数字组成的状态代码，以及描述状态的短语，彼此由空格分隔。

状态代码的第一个数字代表当前响应的类型：

1xx消息——请求已被服务器接收，继续处理

2xx成功——请求已成功被服务器接收、理解、并接受

3xx重定向——需要后续操作才能完成这一请求

4xx请求错误——请求含有词法错误或者无法被执行

5xx服务器错误——服务器在处理某个正确请求时发生错误

响应头(消息头)包含：

1、Location：这个头配合302状态吗，用于告诉客户端找谁

2、Server：服务器通过这个头，告诉浏览器服务器的类型

3、Content-Encoding：告诉浏览器，服务器的数据压缩格式

4、Content-Length：告诉浏览器，回送数据的长度

5、Content-Type：告诉浏览器，回送数据的类型

6、Last-Modified：告诉浏览器当前资源缓存时间

7、Refresh：告诉浏览器，隔多长时间刷新

8、Content-Disposition：告诉浏览器以下载的方式打开数据。

例如： context.Response.AddHeader("Content-Disposition","attachment:filename=aa.jpg");

context.Response.WriteFile("aa.jpg");

9、Transfer-Encoding：告诉浏览器，传送数据的编码格式

10、ETag：缓存相关的头（可以做到实时更新）

11、Expries：告诉浏览器回送的资源缓存多长时间。如果是-1或者0，表示不缓存

12、Cache-Control：控制浏览器不要缓存数据 no-cache

13、Pragma：控制浏览器不要缓存数据 no-cache

14、Connection：响应完成后，是否断开连接。 close/Keep-Alive

15、Date：告诉浏览器，服务器响应时间

16、状态行：　例如：　HTTP/1.1　200 OK　（协议的版本号是1.1　响应状态码为200　响应结果为 OK）

17、实体内容（实体头）：响应包含浏览器能够解析的静态内容，例如：html，纯文本，图片等等信息