# 网络编程概述

Java是 Internet 上的语言，它从语言级上提供了对网络应用程序的支持，程序员能够很容易开发常见的网络应用程序。

Java提供的网络类库，联网的底层细节被隐藏在 Java 的本机安装系统里，由 JVM 进行控制。并且 Java 实现了一个跨平台的网络库，程序员面对的是一个统一的网络编程环境。

# 网络基础

## 计算机网络：

把分布在不同地理区域的计算机与专门的外部设备用通信线路互连成一个规模大、功能强的网络系统，从而使众多的计算机可以方便地互相传递信息、共享硬件、软件、数据信息等资源。

### 网络编程的目的：

直接或间接地通过网络协议与其它计算机进行通讯。

网络编程中有两个主要的问题：

1.如何准确地定位网络上一台或多台主机

2.找到主机后如何可靠高效地进行数据传输

如何实现网络中的主机互相通信：

通信双方地址

一定的规则（有两套参考模型）

OSI参考模型：模型过于理想化，未能在因特网上进行广泛推广

TCP/IP参考模型(或TCP/IP协议)：事实上的国际标准。

## 网络基础

### 网络通信协议

|  |  |  |
| --- | --- | --- |
| **OSI参考模型** | **TCP/IP参考模型** | **TCP/IP参考模型各层对应协议** |
| 应用层 | **应用层** | HTTP、ftp、telnet、DNS… |
| 表示层 |
| 会话层 |
| 传输层 | **传输层** | TCP、UDP、… |
| 网络层 | **网络层** | IP、ICMP、ARP… |
| 数据链路层 | **物理+数据链路层** | Link |
| 物理层 |

物理层和数据链路层涉及物理介质访问和二进制数据流传输。

网络层的主要协议有IP（Internet protocol）、ICMP（Internet Control Message Protocol，互联网控制报文协议）、IGMP（Internet Group Management Protocol，互联网组管理协议）、ARP（Address Resolution Protocol，地址解析协议）和RARP（Reverse Address Resolution Protocol，反向地址解析协议）等。涉及寻址和路由选择

传输层的基本功能是为两台主机间的应用程序提供端到端的通信。传输层从应用层接受数据，并且在必要的时候把它分成较小的单元，传递给网络层，并确保到达对方的各段信息正确无误。

应用层提供应用程序的网络接口。

数据封装

数据拆封

通讯要素1：IP 和 端口号

IP 地址：InetAddress

唯一的标识 Internet 上的计算机

本地回环地址(hostAddress)：127.0.0.1 主机名(hostName)：localhost

端口号标识正在计算机上运行的进程（程序）

不同的进程有不同的端口号 mysql:3306 被规定整数 0~65535。其中，0~1023被预先定义的服务通信占用（如MySql占用端口3306，http占用端口80等）。除非我们需要访问这些特定服务，否则，就应该使用 1024~65535 这些端口中的某一个进行通信，以免发生端口冲突。

端口号与IP地址的组合得出一个网络套接字。

## InetAddress类

Internet上的主机有两种方式表示地址：

1.域名(hostName)：www.tanzhouedu.com

2.IP 地址(hostAddress)： 212.22.35.210

InetAddress类主要表示IP地址，两个子类：Inet4Address、Inet6Address。

InetAddress 类对象含有一个 Internet 主机地址的域名和IP地址，如：www.tanzhouedu.com 和 212.22.35.210。

域名容易记忆，当在连接网络时输入一个主机的域名后，域名服务器（DNS）负责将域名转化成IP地址，这样才能和主机建立连接。 -------域名解析（多个域名可对应一个IP地址）

DNS（域名解析器）

182.92.151.178

182.92.151.178

网络服务器

C:\Windows\System32\drivers\etc\hosts

先找本机hosts，是否有输入的域名地址，没有的话，再通过DNS服务器去解析域名，解析成IP，根据IP寻找主机网络防火墙404

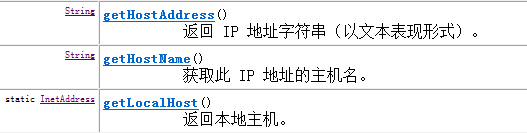
### InetAddress类常用方法

InetAddress类没有提供公共的构造器，而是提供了如下两个静态方法来获取InetAddress实例



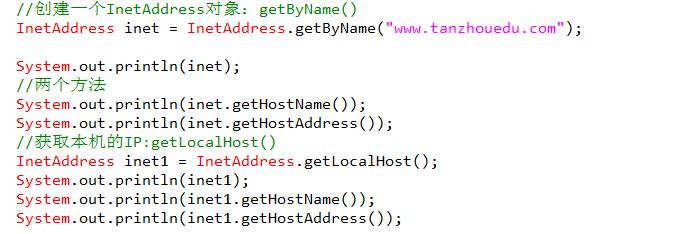


InetAddress提供了如下几个常用的方法





InetAddress 代码示例



## 通讯要素2：网络通信协议

### 网络通信协议

计算机网络中实现通信必须有一些约定，即通信协议，对速率、传输代码、代码结构、传输控制步骤、出错控制等制定标准。

通信协议分层的思想

由于结点之间联系很复杂，在制定协议时，把复杂成份分解成一些简单的成份，再将它们复合起来。最常用的复合方式是层次方式，即同层间可以通信、上一层可以调用下一层，而与再下一层不发生关系。各层互不影响，利于系统的开发和扩展。

## TCP/IP协议簇

传输层协议中有两个非常重要的协议：

传输控制协议TCP(Transmission Control Protocol)

用户数据报协议UDP(User Datagram Protocol)。

TCP/IP 以其两个主要协议：传输控制协议(TCP)和网络互联协议(IP)而得名，实际上是一组协议，包括多个具有不同功能且互为关联的协议。

IP(Internet Protocol)协议是网络层的主要协议，支持网间互连的数据通信。

TCP/IP协议模型从更实用的角度出发，形成了高效的四层体系结构，即物理链路层、IP层、传输层和应用层。

### TCP 和 UDP

TCP协议：(http协议包含着Tcp协议)

使用TCP协议前，须先建立TCP连接，形成传输数据通道

传输前，采用“三次握手”方式，是可靠的

TCP协议进行通信的两个应用进程：客户端、服务端

在连接中可进行大数据量的传输

传输完毕，需释放已建立的连接，效率低

UDP协议：

将数据、源、目的封装成数据包，不需要建立连接

每个数据报的大小限制在64K内

因无需连接，故是不可靠的

发送数据结束时无需释放资源，速度快

利用套接字(Socket)开发网络应用程序早已被广泛的采用，以至于成为事实上的标准。

·通信的两端都要有Socket，是两台机器间通信的端点

·网络通信其实就是Socket间的通信。

·Socket允许程序把网络连接当成一个流，数据在两个Socket间通过IO传输。 //输出 输入流

·一般主动发起通信的应用程序属客户端，等待通信请求的为服务端

Java里面如何实现TCP协议呢？

Java语言的基于套接字编程分为服务端编程和客户端编程，其通信模型如图所示：

基于TCP的Socket通信



### 基于Socket的TCP编程

#### Socket类的常用方法

|  |  |
| --- | --- |
| ***方法*** | ***功能*** |
| ***InetAddress getLocalAddress()*** | ***获取套接字绑定的本地地址*** |
| ***int getLocalPort()*** | ***返回此套接字绑定到的本地端口*** |
| ***InetAddress getInetAddress()*** | ***返回套接字连接的地址*** |
| ***int getPort()*** | ***返回此套接字连接到的远程端口*** |
| ***void close() throws IOException*** | ***关闭Socket，不可在以后的网络连接中使用，除非创建新的套接字*** |
| ***InputStream getInputStream() throws IOException*** | ***获取与Socket相关联的字节输入流，用于从Socket中读数据。*** |
| ***OutputStream getOutputStream() throws IOException*** | ***获取与Socket相关联的字节输出流，用于向Socket中写数据。*** |

#### ServerSocket类的常用方法

|  |  |
| --- | --- |
| ***方 法*** | ***功 能*** |
| ***Socket accept() throws IOException*** | ***等待客户端的连接请求，返回与该客户端进行通信用的Socket对象*** |
| ***void setSoTimeout(int timeout) throws SocketException*** | ***设置accept()方法等待连接的时间为timeout毫秒。若时间已到，还没有客户端连接，则抛出InterruptedIOException异常，该倾听Socket可继续使用。若timeout值为0，则表示accept()永远等待。该方法必须在倾听Socket创建后，在accept()之前调用才有效。*** |
| ***void close()throws IOException*** | ***关闭监听Socket*** |
| ***InetAddress getInetAddress()*** | ***返回此服务器套接字的本地地址。*** |
| ***int getLocalPort()*** | ***返回此套接字在其上监听的端口号*** |
| ***SocketAddress getLocalSocketAddress()*** | ***返回此套接字绑定的端点的地址*** |

#### Socke 工作的四步骤

客户端Socket的工作过程包含以下四个基本的步骤：

创建 Socket：根据指定服务端的 IP 地址与端口号构造 Socket 类对象。若服务器端响应，则建立客户端到服务器的通信线路。若连接失败，会出现异常。

打开连接到 Socket 的输入/出流： 使用 getInputStream()方法获得输入流，使用 getOutputStream()方法获得输出流，进行数据传输

按照一定的协议对 Socket 进行读/写操作：通过输入流读取服务器放入线路的信息（但不能读取自己放入线路的信息），通过输出流将信息写入线程。

关闭 Socket：断开客户端到服务器的连接，释放线路

#### 客户端创建Socket对象

客户端程序可以使用Socket类创建对象，创建的同时会自动向服务器方发起连接。Socket的构造方法是：

Socket(String host,int port)throws UnknownHostException,IOException：向服务器(域名是host。端口号为port)发起TCP连接，若成功，则创建Socket对象，否则抛出异常。

Socket(InetAddress address,int port)throws IOException：根据InetAddress对象所表示的IP地址以及端口号port发起连接。

客户端建立socketAtClient对象的过程就是向服务器发出套接字连接请求

Socket s = new Socket(“192.168.40.165”,9999);

OutputStream out = s.getOutputStream();

out.write(“hello”.getBytes());

s.close();

#### 服务器程序的四步骤

服务器程序的工作过程包含以下四个基本的步骤：

调用 ServerSocket(int port) ：创建一个服务器端套接字，并绑定到指定端口上。用于监听客户端的请求。

调用 accept()：监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字对象。

调用 该Socket类对象的 getOutputStream() 和 getInputStream ()：获取输出流和输入流，开始网络数据的发送和接收。

关闭ServerSocket和Socket对象：客户端访问结束，关闭通信套接字。

#### 服务器建立 ServerSocket 对象

ServerSocket 对象负责等待客户端请求建立套接字连接，类似邮局某个窗口中的业务员。也就是说，服务器必须事先建立一个等待客户请求建立套接字连接的ServerSocket对象。

所谓“接收”客户的套接字请求，就是accept()方法会返回一个 Socket 对象

ServerSocket ss = new ServerSocket(9999);

Socket s = ss.accept ();

InputStream in = s.getInputStream();

byte[] buf = new byte[1024];

int num = in.read(buf);

String str = new String(buf,0,num);

System.out.println(s.getInetAddress().toString()+”:”+str);

s.close();

ss.close();