【济南中心】JAVA 编程阶梯：基础篇之第二十六章

网络编程概述

计算机网络

是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。

网络编程

就是用来实现网络互连的不同计算机上运行的程序间可以进行数据交换

网络编程三要素之 IP

每个设备在网络中的唯一标识

每台网络终端在网络中都有一个独立的地址 ,我们在网络中传输数据就是使用这个地址。

ipconfig：查看本机 IP192.168.12.42

ping：测试连接 192.168.40.62

本地回路地址：127.0.0.1 255.255.255.255 是广播地址

IPv4 :4 个字节组成 ,4 个 0-255。大概 42 亿 ,30 亿都在北美 ,亚洲 4 亿。2011 年初已经用尽。

IPv6：8 组，每组 4 个 16 进制数。

1a2b:0000:aaaa:0000:0000:0000:aabb:1f2f

1a2b::aaaa:0000:0000:0000:aabb:1f2f

1a2b:0000:aaaa::aabb:1f2f

1a2b:0000:aaaa::0000:aabb:1f2f

1a2b:0000:aaaa:0000::aabb:1f2f

网络编程三要素之端口号概述

每个程序在设备上的唯一标识

每个网络程序都需要绑定一个端口号，传输数据的时候除了确定发到哪台机器上，还要明确发到哪个程序。

端口号范围从 0-65535

编写网络应用就需要绑定一个端口号，尽量使用 1024 以上的，1024 以下的基本上都被系统程序占用了。

常用端口

mysql: 3306

oracle: 1521

web: 80

tomcat: 8080

QQ: 4000

feiQ: 2425

网络编程三要素协议

为计算机网络中进行数据交换而建立的规则、标准或约定的集合。

UDP

面向无连接，数据不安全，速度快。不区分客户端与服务端。

TCP

面向连接（三次握手），数据安全，速度略低。分为客户端和服务端。

三次握手: 客户端先向服务端发起请求, 服务端响应请求, 传输数据

Socket 套接字

网络上具有唯一标识的 IP 地址和端口号组合在一起才能构成唯一能识别的标识符套接字。

通信的两端都有 Socket。

网络通信其实就是 Socket 间的通信。

数据在两个 Socket 间通过 IO 流传输。

Socket 在应用程序中创建，通过一种绑定机制与驱动程序建立关系，告诉自己所对应的 IP 和 port。

UDP 传输

```
// 发送端：
class UDPSend {
		public static void main(String[] args) {
				// 创建 DatagramSocket, 随机端口号
				DatagramSocket ds = new DatagramSocket();
				byte[] buf = "这是 UDP 发送端".getBytes();
				// 创建 DatagramPacket, 指定数据, 长度, 地址, 端口

				DatagramPacket dp = new DatagramPacket(buf, buf.length,
                                                InetAddress.getByName("192.1
68.1.253"), 10000);
				// 使用 DatagramSocket 发送 DatagramPacket
				ds.send(dp);
				// 关闭 DatagramSocket
```

```
                ds.close();
        }
}
// 接收端
class UDPRece {
        public static void main(String[] args) throws Exception {
                // 创建 DatagramSocket, 指定端口号
                DatagramSocket ds = new DatagramSocket(10000);
                // 创建 DatagramPacket, 指定数组, 长度
                byte[] buf = new byte[1024];
                // 使用 DatagramSocket 接收 DatagramPacket
                DatagramPacket dp = new DatagramPacket(buf, buf.length);
                ds.receive(dp);// 将发送端发送的数据包接收到接收端的数据包中
                String ip = dp.getAddress().getHosyAddress();// 获取发送端的
ip
                String data = new String(dp.getData(), 0, dp.getLength());// 获
取数据
                int port = dp.getPort();// 获取发送端的端口号
                sop(ip + ":" + data + ":" + port);
                ds.close();
        }
}
```

UDP 传输优化

```
// 接收端 Receive
        DatagramSocket socket = new DatagramSocket(6666); // 创建 socket 相当于创建码
头
        DatagramPacket packet = new DatagramPacket(new byte[1024], 1024); // 创建 packet
相当于创建集装箱

        while (true) {
                socket.receive(packet); // 接收货物
                byte[] arr = packet.getData();
                int len = packet.getLength();
                String ip = packet.getAddress().getHostAddress();
                System.out.println(ip + ":" + new String(arr, 0, len));
        }
        // 发送端 Send
        DatagramSocket socket = new DatagramSocket(); // 创建 socket 相当于创建码头
        Scanner sc = new Scanner(System.in);

        while (true) {
                String str = sc.nextLine();
                if ("quit".equals(str))
```

```
                                        break;
                                DatagramPacket packet = // 创建 packet 相当于创建集装箱
                                new DatagramPacket(str.getBytes(), str.getBytes().length,
                                                InetAddress.getByName("127.0.
0.1"), 6666);
                                socket.send(packet); // 发货
                }
                socket.close();
}
```

UDP 传输多线程

```
public class Demo {

                public static void main(String[] args) {
                                new Receive().start();

                                new Send().start();
                }

}

class Receive extends Thread {
                public void run() {
                                try {
                                                DatagramSocket socket = new
DatagramSocket(6666); // 创建 socket 相当于创建码头
                                                DatagramPacket packet = new DatagramPacket(new
byte[1024], 1024); // 创建 packet 相当于创建集装箱

                                                while (true) {
                                                                socket.receive(packet); //
接收货物
                                                                byte[] arr =
packet.getData();
                                                                int len = packet.getLength();
                                                                String ip =
packet.getAddress().getHostAddress();
                                                                System.out.println(ip + ":" +
new String(arr, 0, len));
                                                }
                                } catch (IOException e) {

                                                e.printStackTrace();
                                }
```

```
		}
}

class Send extends Thread {
		public void run() {
			try {
				DatagramSocket socket = new DatagramSocket();
// 创建 socket 相当于创建码头
				Scanner sc = new Scanner(System.in);

				while (true) {
					String str = sc.nextLine();
					if ("quit".equals(str))
						break;
					DatagramPacket packet = // 创
建 packet 相当于创建集装箱
					new
DatagramPacket(str.getBytes(), str.getBytes().length,
    InetAddress.getByName("127.0.0.1"), 6666);
					socket.send(packet); // 发货
				}
				socket.close();
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
}
```

TCP 协议

```
// 客户端:
		class TCPClient {
			public static void main(String[] args) {
				// 创建 Socket 连接服务端(指定 ip 地址,端口号)
通过 ip 地址找对应的服务器
				Socket s = new Socket("192.168.1.111", 10000);
				// 调用 Socket 的 getInputStream()和
getOutputStream()方法获取和服务端相连的 IO 流
				OutputStream os = s.getOutputStream();
				// 输出流写出数据到服务端的输入流
				out.write("这是 TCP 发送的数据".getBytes());
				s.close();
			}
```

```
}
// 服务端：
class TCPServer {
        public static void main(String[] args) {
                // 创建 ServerSocket(需要指定端口号)
                ServerSocket ss = new ServerSocket(10000);
                // 调用 ServerSocket 的 accept()方法接收一个客户端请求，得到一个 Socket
                Socket s = ss.accept();

                String ip = s.getInetAddress().getHostAddress();
                sop(ip);
                // 调用 Socket 的 getInputStream()和 getOutputStream()方法获取和客户端相连的 IO 流
                InputStream is = s.getInputStream();
                byte[] buf = new byte[1024];
                // 输入流读取客户端输出流写出的数据
                int len = is.read(buf);
                sop(new String(buf, 0, len));
                s.close();
                ss.close();
        }
}
```

TCP 协议代码优化

```
// 客户端
        Socket socket = new Socket("127.0.0.1", 9999); // 创建 Socket 指定 ip 地址和端口号
        InputStream is = socket.getInputStream(); // 获取输入流
        OutputStream os = socket.getOutputStream(); // 获取输出流
        BufferedReader br = new BufferedReader(new InputStreamReader(is));
        PrintStream ps = new PrintStream(os);

        System.out.println(br.readLine());
        ps.println("我想报名就业班");
        System.out.println(br.readLine());
        ps.println("爷不学了");
        socket.close();
        // 服务端
        ServerSocket server = new ServerSocket(9999); // 创建服务器
        Socket socket = server.accept(); // 接受客户端的请求
        InputStream is = socket.getInputStream(); // 获取输入流
```

```
                        OutputStream os = socket.getOutputStream(); // 获取输出流

                        BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                        PrintStream ps = new PrintStream(os);

                        ps.println("欢迎咨询传智播客");
                        System.out.println(br.readLine());
                        ps.println("报满了,请报下一期吧");
                        System.out.println(br.readLine());
                        server.close();
                        socket.close();
```

服务端是多线程的

```
ServerSocket server = new ServerSocket(9999); // 创建服务器
while (true) {
                final Socket socket = server.accept(); // 接受客户端的请求
                new Thread() {
                        public void run() {
                                try {
                                        BufferedReader br = new
BufferedReader(

    new InputStreamReader(socket.getInputStream()));
                                        PrintStream ps = new
PrintStream(

    socket.getOutputStream());
                                        ps.println("欢迎咨询传智播客
");
                                        System.out.println(br.readLi
ne());
                                        ps.println("报满了,请报下一
期吧");
                                        System.out.println(br.readLi
ne());
                                        socket.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                }.start();
}
```

识别二维码 关注黑马程序员视频库
免费获得更多 IT 资源