PDF-XChange Click to buy NOW! www.docu-track.com
PDF-XChange Click to buy NOW! www.docu-track.com

JAVA 网络编程(TCP/UDP):

先来了解一下 TCP/UDP 的相关特点。

TCP：Transfer Control Protocol:传输控制协议，是面向连接，可靠，安全的连接。

UDP：User Datagram Protocol: 用户数据报协议，是非面向连接的，不可靠的，不安全的连接。一般用在如：视频传输，网络电话等。

我们分开来讨论一下，这两种协议。当然，我们主要是做一下 JAVA 的 TCP 的讨论.

一．基于 TCP 协议的网络编程。

类： Socket 它是一种抽象的网络文件，JAVA 封装了这个类，屏蔽了底层的细节操作。它提供了两个很得要的方法：

getInputStream()::：来获得网络文件中的输入流 InputStream 对象

getOutputStream()：来获得网络文件中的输出流 OutputStream 对象

服务器端：

类：ServerSocket 用 PORT 号来建立一个 ServetSocket 对像。

方法： accept(); 返回一个 Socket 给客户端。它是一个阻塞方法，直到有客户端使用 Socket 与之建立了连接。

如：SERVER： ServerSocket ss = new ServerSocket(port);

Socket s = ss.accept(); //进入阻塞，等待客户端连接。

CLIENT： Socket c = new Socket("127.0.0.1",port); //注： 端口号一定要与 SERVER 一致。 与服务器建立了连接，通过 IP 和 PORT

下面，我们来做一个例子来掌握 JAVA 的 TCP 网络编程

做一个服务器，接收客户端发出的字符串，转换成大写之后再返回给客户端，并在客户终端上显示。

Coding: SERVER 端

```
Public class Server {
    Public static void main(String[] args) {
        ServerSocket ss = new ServerScoket(9050);//指定端口号
        While(true) {
            Socket s = ss.accept();//等待客户端连接。
            (new MyThread(s)).start();
        }
    }
}
Class MyThread extends Thread {
    Private Socket s;
    Public MyThread(Socket s) {
        This.s = s;
    }
    Public void run() {
        InputStream inStream = s.getInputStream();
        OutputStream outStream = s.getOutputStream();

        BufferedReader br = new BufferedReader(new InputStreamReader(inStream));
        PrintWriter pw = new PrintWriter(outStream);
        String str = br.readLine();
```

PDF-XChange Click to buy NOW! www.docu-track.com
PDF-XChange Click to buy NOW! www.docu-track.com

```
            Pw.println(str.toUpperCase());
            Pw.flush();
        }|
    }
```

CLIENT：

```
    Public class Client {
        Public static void main(String[] args) {
            Socket c = new Socket("127.0.0.1",9050) ;
            InputStream is = c.getInputStream();
            OutputStream os = c.getOutputStream();
            BufferedReader br = new BufferedReader(new InputStreamReader(is));
            PrintWriter pw = new PrintWriter(os);
            Pw.println("Hello");
            Pw.flush(); //清空缓冲区
            String str = br.readLine();
            System.out.println(str);
        }
    }
```

由上面可以看出，JAVA 的基于 TCP 的网络编程非常简单化，不再需要我们去考虑底层的网络，TCP/IP 协议等。只需要按照通过 SOCKET 获得输入输出流对像，再用我们所学的 I/O 流的知识去操作就 OK 了。另外，服务端要做成多线程的，就再使用多线程的相关知识去完成，所以说，JAVA 的网络编程本身不是很困难，但其加上多线程，I/O 流，相互之间就会有一定的复杂度了。
其它的例子就不再多举了!!!

接下来，我们来总结一下 UDP 的网络编程。
基于 UDP 的网络编程
类：DatagramSocket　一个比喻，相当于‘邮筒’的功能
方法：receive(DatagramPacket dp)
类：DatagramPacket　一个比喻，相当于‘信’的角色
‘邮筒’具有收发‘信’的功能，对于接收来说，只需要提供‘信纸’（byte[]）和长度（length），来接收内容；而要发出，除了这两样外，还必需要知道对方的‘地址’（IP）和‘邮编’（PORT）；
让我们来看一个例子：
Coding:　Server 端

```
public class UPDDayTimeDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        DatagramSocket ds = null;
        while(true) {
            try {
                ds = new DatagramSocket(7631);//先建立一个‘邮筒’
                byte[] buf = new byte[1];//用来接收信的内容
                DatagramPacket dp = new DatagramPacket(buf,buf.length);//创建一封’信‘
```

PDF-XChange Click to buy NOW! www.docu-track.com
PDF-XChange Click to buy NOW! www.docu-track.com

```
                ds.receive(dp);//此方法阻塞，用来接收‘信’
                byte[] sndbuf = new Date().toString().getBytes();
                DatagramPacket snddp = new DatagramPacket(sndbuf,sndbuf.length,
                                dp.getAddress(),dp.getPort());//建立一封往外发的‘信’
                ds.send(snddp); //发‘信’
                //相当于如下：
                //dp.setData(new Date().toString.getBytes());
                //ds.send(dp);
            } catch(Exception e) {
                e.printStackTrace();
            } finally {
                if(ds != null)
                    ds.close();
            }
        }
    }
}
Coding：  Client 端
public class UDPClient {
    public static void main(String[] args) {
        DatagramSocket ds = null;
        try {
            ds = new DatagramSocket();
            byte[] buf = new byte[1];
            DatagramPacket dp = new DatagramPacket(buf,buf.length,
                                InetAddress.getLocalHost(),7631);
            while(true) {
                ds.send(dp);
                System.out.println("Client send OK . . .");
                byte[] recbuf = new byte[128];
                DatagramPacket recdp = new DatagramPacket(recbuf,recbuf.length);
                ds.receive(recdp);

                System.out.println("Receive ok. . .");
                System.out.println(new String(recdp.getData(),0,recdp.getLength()));
                //相当于如下：
                //ds.receive(dp);
                //System.out.println(new String(dp.getDate(),0,dp.getLength()));
                Thread.sleep(1000);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if(ds != null)
```

PDF-XChange Click to buy NOW! www.docu-track.com
PDF-XChange Click to buy NOW! www.docu-track.com

```
                ds.close();
            }
        }
}
```

因为，UDP 是非面向连接的，不可靠的，每一次的发送所走了路由都可能不一样，也可能成功，也可能失败。

现在，我们再看一下另外两个类：

一．MulticastSocket 类， 它继承于 DatagramSocket 类。

多播数据报套接字类用于发送和接收 IP 多播包。MulticastSocket 是一种 (UDP) DatagramSocket，它具有加入 Internet 上其他多播主机的“组”的附加功能。

多播组通过 D 类 IP 地址和标准 UDP 端口号指定。D 类 IP 地址在 224.0.0.0 和 239.255.255.255 的范围内（包括两者）。地址 224.0.0.0 被保留，不应使用。

可以通过首先使用所需端口创建 MulticastSocket，然后调用 joinGroup(InetAddress groupAddr) 方法来加入多播组：

调用 leaveGroup(group) 方法来离开多播组;

现在，我们来做一个简单例子：

```
public class MulticastSocketDemo {
    public static void main(String[] args) throws IOException {
        MulticastSocket ms = new MulticastSocket(7789);//使用 PORT 来创建一个多播组
        InetAddress ip = InetAddress.getByName("228.5.6.7");//多播组的 IP 在上面所讲的范围内。
        ms.joinGroup(ip);//加入多播组
        String msg = "Hello";
        DatagramPacket dp = new DatagramPacket(msg.getBytes(),msg.length(),ip,7789);
        ms.send(dp);

        //获得回应
        byte[] buf = new byte[1024];
        DatagramPacket recDP = new DatagramPacket(buf,buf.length);
        ms.receive(recDP);//接收
        System.out.println(new String(recDP.getData(),0,recDP.getLength()));
        ms.leaveGroup(ip);//离开多播组
    }
}
```

二．URL 类，统一资源定位器

组成：协议：//主机位置：端口号/相对位置

协议有：http, mail, FTP , File:// 等。

这些都是针对应用层的网络编程

同样，我们来通过一个例子做说明：

```
public class URLConnectionTest {
    public static void main(String[] args) {
        try {
            int n = 1;
```

PDF-XChange Click to buy NOW! www.docu-track.com
PDF-XChange Click to buy NOW! www.docu-track.com

```
            String name = "http://192.168.0.187";
            URL url = new URL(name);//构造一个 URL
            System.out.println("============= URL INFO ================");
            System.out.println("Host: "+url.getHost());//获得主机名
            System.out.println("Path: "+url.getPath());//获得此 URL 的路径部分
            System.out.println("Query: "+url.getQuery());//获得此 URL 的查询部分。
            System.out.println("Protocol: "+url.getProtocol());//获得此 URL 的协议名称。
            System.out.println("Port: "+url.getPort());//获得此 URL 的端口号。
            System.out.println("=======================================");
        InputStream is = url.openStream();//打开到此 URL 的连接并返回一个用于从该连接读入的 InputStream。
            BufferedReader br = new BufferedReader(new    InputStreamReader(is));
            String str = null;
            while((str = br.readLine()) != null && ++n <= 20) {
                System.out.println(str);
            }
            System.out.println("---------------------------------");
        URLConnection connect = url.openConnection();//返回一个 URLConnection 对象，它表示到 URL 所引用的远程对象的连接
            connect.connect(); //
            String key;
            n = 1;
            while((key = connect.getHeaderFieldKey(n)) != null) {
                String value = connect.getHeaderField(n);
                System.out.println(key+" , "+value);
                n++;
            }
            System.out.println("--------------------------");
            System.out.println("getContentTypey(): "+connect.getContentType());
            System.out.println("getContentLength(): "+connect.getContentLength());
            System.out.println("getContentEncoding(): "+connect.getContentEncoding());
            System.out.println("getContentTimeout(): "+connect.getConnectTimeout());
            System.out.println("getDate(): "+connect.getDate());
            System.out.println("getLastModified(): "+connect.getLastModified());
            System.out.println(new java.sql.Date(connect.getLastModified()));
            System.out.println("-----------------------------------------------");
        BufferedReader in = new BufferedReader(new InputStreamReader(connect.getInputStream()));
            String line;
            n = 1;
            while((line = in.readLine()) != null && ++n <= 10) {
                System.out.println(line);
            }
        } catch (Exception e) {
            e.printStackTrace() ;
```

PDF-XChange
Click to buy NOW!
www.docu-track.com
PDF-XChange
Click to buy NOW!
www.docu-track.com

```
            }
        }
}
```

对于以上的相关方法的详细信息，请查阅 API，谢谢！

作者： 叶加飞 (Steven Ye)
Mailto：leton.ye@gmail.com