网络编程

天津卓讯科技有限公司

第11章 网络编程

- 11.1 网络编程基础知识
- 11.2 TCP/IP
- 11.3 TCP编程
- 11.4 UDP编程
- 11.5 基于URL的通信

11.1 网络编程基础知识

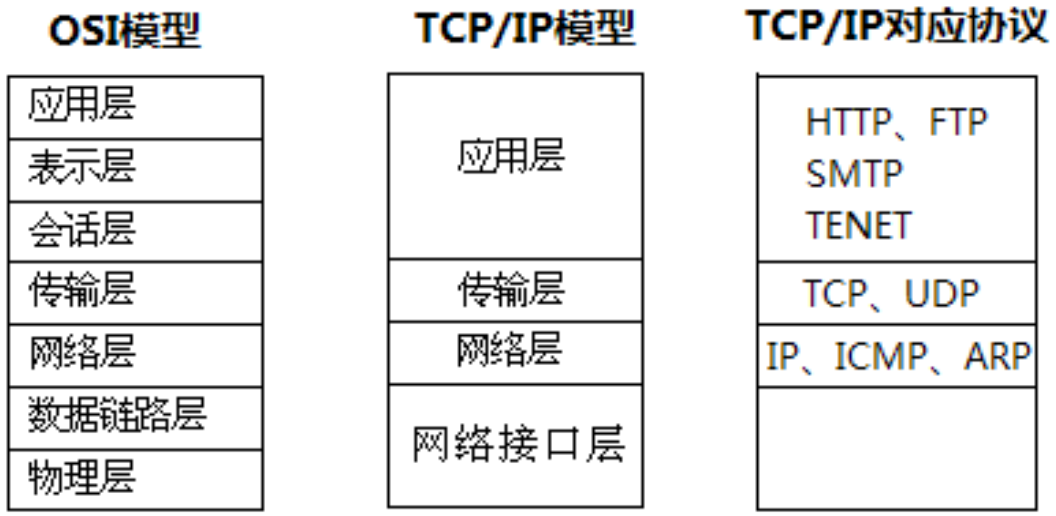
- 网络基础概念
- 网络参考模型
- TCP/IP
- TCP & UDP
- TCP/IP网络程序的IP地址和端口号
- Socket

网络基础概念

- 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。
- 网络通信协议
  - 要使计算机连成的网络能够互通信息，需要对数据传输速率、传输代码、代码结构、传输控制步骤、出错控制等制定一组标准，这一组共同遵守的通信标准就是网络通信协议，不同的计算机之间必须使用相同的通讯协议才能进行通信。
  - 在Internet中TCP/IP协议是使用最为广泛的通讯协议。TCP/IP是英文Transmission Control Protocol/Internet Protocol的缩写，意思是“传输控制协议/网际协议”。

网络参考模型

- OSI参考模型：开放系统互连参考模型。由国际标准化组织ISO提出的一个网络系统互连模型。
- OSI模型目前主要用于教学理解。实际使用中，网络硬件设备基本都是参考TCP/IP模型



11.2 TCP/IP

- 针对TCP/IP模型的各层，都有对应数据传输格式，这个数据传输的格式就是常说的协议
- TCP/IP包括上百个各种功能的协议，如：超文本传输协议(HTTP)、远程登录(TeLnet)、文件传输(FTP)和电子邮件(SMTP、POP3、IMAP4)等。

网络通讯要素

- IP地址：网络中每台设备的标识号
  - 是一个逻辑地址：IPV4、IPV6
  - 不便记忆，可用主机名
  - 本地回环地址：127.0.0.1 主机名：localhost
- 端口号：用于标识具有网络功能的进程的逻辑地址(标识号)
  - 有效端口：0~65535，其中0~1024系统使用或保留端口。
  - 端口与协议有关：TCP和UDP的端口互不相干
- 传输协议：通讯的规则
  - 常用协议：TCP、UDP

java.net.InetAddress类

- 表示互联网地址（IP地址），它封装了IP地址和域名相关的操作方法

```
//使用域名创建InetAddress对象
InetAddress inet1 = InetAddress.getByName("www.csdn.net");
System.out.println(inet1);
//使用IP创建InetAddress对象
InetAddress inet2 = InetAddress.getByName("117.79.93.222");
System.out.println(inet2);
InetAddress inet3 = InetAddress.getLocalHost(); //获得本机InetAddress对象
System.out.println(inet3);
String host = inet3.getHostName();//获得InetAddress对象中存储的域名
System.out.println("本机名：" + host);
String ip = inet3.getHostAddress(); //获得InetAddress对象中存储的IP
System.out.println("本机IP：" + ip);
```

两个重要传输协议

- TCP(Transmission Control Protocol)传输控制协议
  - 通过三次握手建立连接，形成传输数据的通道。
  - 在连接中通过流来进行字节数据传输
  - 基于连接，是可靠协议
  - 面向连接、数据传输可靠、效率稍低
- UDP(User Datagram Protocol)用户数据报协议
  - 将数据及源和目的封装成数据包中，不需要建立连接
  - 每个数据报的大小在限制在64k内
  - 因无连接，是不可靠协议
  - 面向非连接、数据传输不可靠、效率高

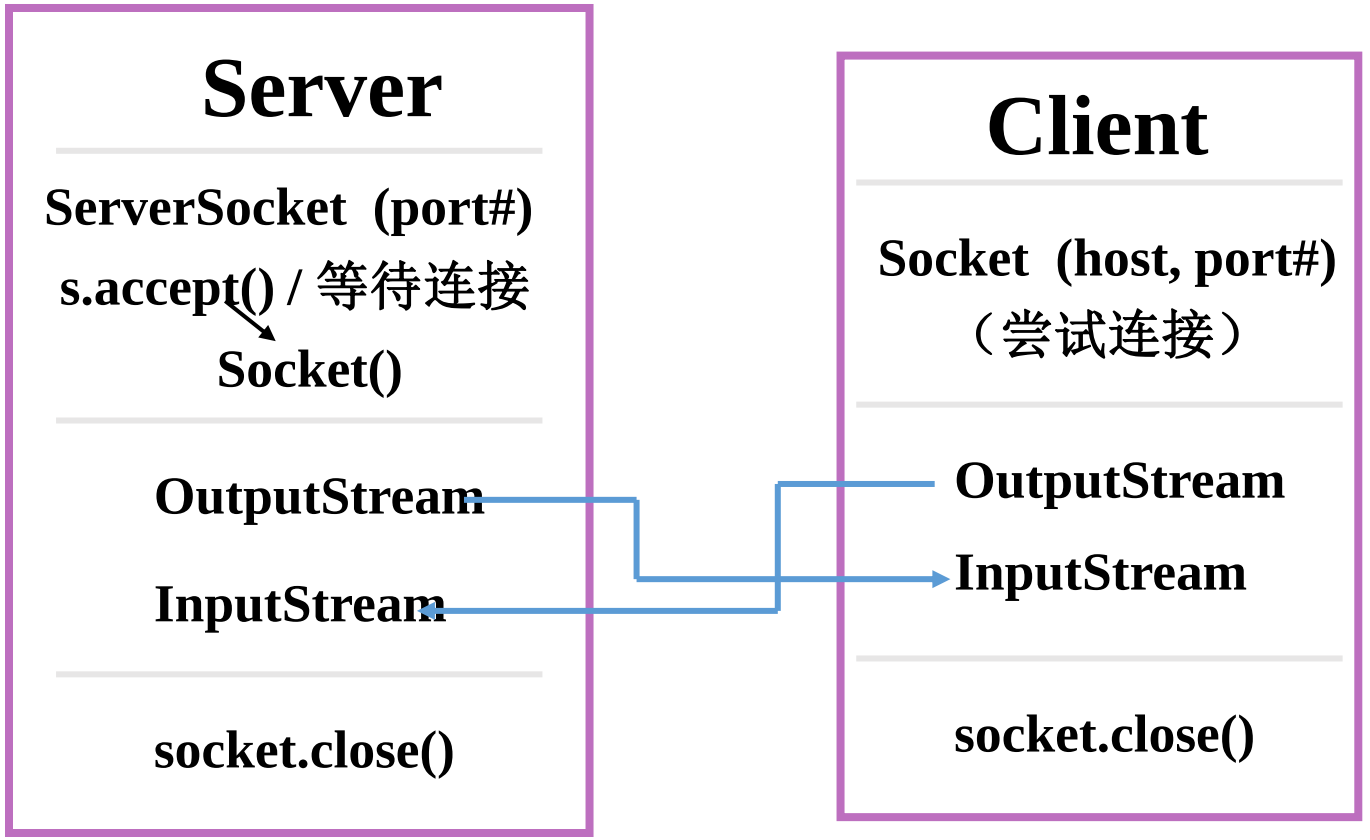
Socket(套接字)

- 两个应用程序可以通过一个双向的网络通信连接实现数据交换，这个双向链路的一端称为一个Socket。
- 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传输。

网络编程的四个基本步骤

- 创建socket;
- 打开连接到socket的输入/输出流；
- 按照一定的协议对socket进行读/写操作；
- 关闭socket;

- java.net包中定义了两个类：Socket和ServerSocket，分别用来实现TCP的client和server端。



TCP示例

```
public class TCPServer { //TCP Server
    public static void main(String[] args) {
        try {
            ServerSocket s = new ServerSocket(8888);
            while (true) {
                Socket s1 = s.accept();
                BufferedWriter bw = new BufferedWriter(
                                    new OutputStreamWriter(s1.getOutputStream()));
                bw.write("你好, " +s1.getInetAddress()+ ":" + s1.getPort());

                s1.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("程序运行出错:" + e);
        }
    }
}
```

TCPServer.java

TCP示例

```
public class TCPClient {  //TCP Client
    public static void main(String[] args) {
        try {
            Socket s = new Socket("127.0.0.1", 8888);
            BufferedReader br = new BufferedReader(
                                    new InputStreamReader(s.getInputStream()));
            String str = br.readLine();
            System.out.println("服务器说:" + str);
        } catch (UnknownHostException e) {
            System.err.println("服务器连接失败！ ");
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TCPClient.java

TCP聊天程序

```
public class TalkServer {
    public static void main(String[] args) {
        try {
            ServerSocket s = new ServerSocket(9999);
            Socket s1 = s.accept();
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(s1.getOutputStream()),true);
           BufferedReader br = new BufferedReader(new InputStreamReader(s1.getInputStream()));
           BufferedReader toClientBr = new BufferedReader(new InputStreamReader(System.in));
           System.out.println("客户端来了!!!");
           System.out.println("客户端说:" + br.readLine());
           String str = null;
           while (!(str = toClientBr.readLine()).equals("exit")) {
               //向客户端写话
               pw.println(str);              pw.flush();
               System.out.println("客户端说:" + br.readLine());
           }
           toClientBr.close();          pw.close();          s1.close();
        } catch (IOException e) {
           e.printStackTrace();          System.out.println("程序运行出错:" + e);
        }
    }
}
```

TalkServer.java

TCP聊天程序

```
public class TalkClient {
    public static void main(String[] args) {
        try {
            Socket s1 = new Socket("127.0.0.1", 9999);
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(s1.getOutputStream()),true);
            BufferedReader br = new BufferedReader(new InputStreamReader(s1.getInputStream()));
            BufferedReader toServerBr = new BufferedReader(new InputStreamReader(System.in));
            String str = null;
            while (!(str = toServerBr.readLine()).equals("exit")) {
                pw.println(str);  // 向服务器写话
                pw.flush();
                System.out.println("服务器端说:" + br.readLine());
            }
            toServerBr.close();          pw.close();          s1.close();
        } catch (UnknownHostException e) {
            System.err.println("服务器连接失败！ ");
            e.printStackTrace();
        } catch (IOException e) {          e.printStackTrace();       }
    }
}
```

TalkClient.java

11.4 UDP编程

- UDP编程所需要的类
  - DatagramSocket 用来发送和接收数据报包的套接字
    - void send(DatagramPacket p)
    - void receive(DatagramPacket p)
  - DatagramPacket 数据报包
    - DatagramPacket(byte[] buf, int length) 用来接收长度为 length 的数据包。
    - DatagramPacket(byte[] buf, int length, InetAddress address, int port)用来将长度为 length 的包发送到指定主机上的指定端口号
      - buf - 包数据
      - length - 包长度
      - address - 目的地址
      - port - 目的端口号
    - int getLength() 返回将要发送或接收到的数据的长度。

UDP发数据示例

```
import java.io.IOException;
import java.net.*;
public class UDPSender {  //发送端
    public static void main(String[] args) {
        String str = "nihao吗";
        DatagramSocket ds = null;
        DatagramPacket dp = null;
        try {
            ds = new DatagramSocket(9999); //定义一个发包的Socket,端口使用9999
            //要发送的数据包,一定要指定目的地IP和端口号
            dp = new DatagramPacket(str.getBytes(), str.getBytes().length,
                                    new InetSocketAddress("127.0.0.1“), 5555);
            ds.send(dp); //发送数据
        } catch (SocketException e) {        e.printStackTrace();
        } catch (IOException e) {        e.printStackTrace();
        }finally{            if(dp != null){        ds.close();    }        }
    }
}
```

UDPSender.java

UDP发数据示例

```
import java.io.IOException;
import java.net.*;
public class UDPReceiver {//接收端
    public static void main(String[] args) {
        byte buf[] = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, buf.length);
        DatagramSocket ds = null;
        try {
            ds = new DatagramSocket(5555); //建立一个收包的socket,监听在5555端口
            while(true){
                ds.receive(dp); //收数据--此方法在接收到数据报前一直阻塞
                System.out.println(new String(buf, 0, dp.getLength()));
            }
        } catch (SocketException e) {e.printStackTrace();
        } catch (IOException e) {e.printStackTrace();
        } finally {            if (null != ds) {ds.close(); }        }
    }
}
```

UDPReceiver.java

11.5 基于URL的通信

- URL代表一个统一资源定位符，它是指向互联网"资源"的指针。
  - 网络中的资源都有唯一的URL地址。
- 构造方法：
  - public URL(String spec) throws MalformedURLException
    - 根据指定的字符串URL地址创建URL对象
- 常用方法：
  - public URLConnection openConnection() throws IOException
  - public final InputStream openStream() throws IOException
    - 打开到此URL的连接并返回一个用于从该连接读入的InputStream

示例

```
import java.io.*;
import java.net.URL;
public class TestURL {
    public static void main(String[] args) {
        String strUrl = "http://www.baidu.com";
        BufferedReader br = null;
        try {
            URL url = new URL(strUrl);
            br = new BufferedReader(new InputStreamReader(url.openStream()));
            String str = "";
            while((str = br.readLine()) != null){
                System.out.println(str);
            }
        } catch (MalformedURLException e) {        e.printStackTrace();
        } catch (IOException e) {        e.printStackTrace();
        }finally{   if(null != br){   try {    br.close();} catch (IOException e) {    e.printStackTrace(); }}}
    }
}
```

练习

- 网络图片下载

URLConnection抽象类

- 代表应用程序和URL之间的通信链接。此类的实例可用于读取和写入此URL引用的资源。
  - 常用它的子类HTTPURLConnection，用于支持HTTP的通信链接
- 创建一个到URL的连接的步骤：
  - 通过在URL上调用openConnection()方法创建连接对象。
  - 处理设置参数和一般请求属性。
  - 使用connect()方法建立到远程对象的实际连接。
  - 远程对象变为可用。远程对象的头字段和内容变为可访问。

HTTP示例

•Get请求示例

```
public static String sendGet(String url) {
  String result = ""; BufferedReader ir;
  try {
    URL u = new URL(url);
    URLConnection connection = u.openConnection();
    connection.connect();
    ir = new BufferedReader(new InputStreamReader(connection.getInputStream()));
    for(String line = null; (line = ir.readLine()) != null;) {
        result += "\n" + line;
    }
  } catch (Exception e) {
     System.out.println("没有结果！ " + e);
  } finally{ ir.close(); }
  return result;
}
```

总结

- IP地址(InetAddress)、端口号、Socket
- TCP编程
  - ServerSocket、Socket
- UDP编程
  - DatagramSocket、DatagramPacket
- 基于URL的网络通信
  - URL、URLConnection、HTTPURLConnection