## socket编程概念

一个是如何准确的定位网络上一台或多台主机，另一个就是找到主机后如何可靠高效的进行数据传输。

在TCP/IP协议中IP层主要负责网络主机的定位，数据传输的路由，由IP地址可以唯一地确定Internet上的一台主机。

而TCP层则提供面向应用的可靠（tcp）的或非可靠（UDP）的数据传输机制，这是网络编程的主要对象，一般不需要关心IP层是如何处理数据的。

目前较为流行的网络编程模型是客户机/服务器（C/S）结构。即通信双方一方作为服务器等待客户提出请求并予以响应。客户则在需要服务时向服务器提 出申请。服务器一般作为守护进程始终运行，监听网络端口，一旦有客户请求，就会启动一个服务进程来响应该客户，同时自己继续监听服务端口，使后来的客户也 能及时得到服务。

对于Java Socket编程而言，有两个概念，一个是ServerSocket，一个是Socket。服务端和客户端之间通过Socket建立连接，之后它们就可以进行通信了。首先ServerSocket将在服务端监听某个端口，当发现客户端有Socket来试图连接它时，它会accept该Socket的连接请求，同时在服务端建立一个对应的Socket与之进行通信。这样就有两个Socket了，客户端和服务端各一个。

对于Socket之间的通信其实很简单，服务端往Socket的输出流里面写东西，客户端就可以通过Socket的输入流读取对应的内容。Socket与Socket之间是双向连通的，所以客户端也可以往对应的Socket输出流里面写东西，然后服务端对应的Socket的输入流就可以读出对应的内容。

所谓socket通常也称作"套接字"，用于描述IP地址和端口，是一个通信链的句柄。应用程序通常通过"套接字"向网络发出请求或者应答网络请求。Socket和ServerSocket类库位于java.net包中。ServerSocket用于服务器端，Socket是建立网络连接时使用的。在连接成功时，应用程序两端都会产生一个Socket实例，操作这个实例，完成所需的会话。对于一个网络连接来说，套接字是平等的，并没有差别，不因为在服务器端或在客户端而产生不同级别。不管是Socket还是ServerSocket它们的工作都是通过SocketImpl类及其子类完成的。

重要的Socket API：

java.net.Socket继承于java.lang.Object，有八个构造器，其方法并不多，下面介绍使用最频繁的三个方法，其它方法大家可以见JDK-1.3文档。

.Accept方法用于产生"阻塞"，直到接受到一个连接，并且返回一个客户端的Socket对象实例。"阻塞"是一个术语，它使程序运行暂时"停留"在这个地方，直到一个会话产生，然后程序继续；通常"阻塞"是由循环产生的。

.getInputStream方法获得网络连接输入，同时返回一个IutputStream对象实例，。

.getOutputStream方法连接的另一端将得到输入，同时返回一个OutputStream对象实例。

注意：其中getInputStream和getOutputStream方法均会产生一个IOException，它必须被捕获，因为它们返回的流对象，通常都会被另一个流对象使用。

## 如何开发一个Server-Client模型的程序

开发原理：

服务器，使用ServerSocket监听指定的端口，端口可以随意指定（由于1024以下的端口通常属于保留端口，在一些操作系统中不可以随意使用，所以建议使用大于1024的端口），等待客户连接请求，客户连接后，会话产生；在完成会话后，关闭连接。

客户端，使用Socket对网络上某一个服务器的某一个端口发出连接请求，一旦连接成功，打开会话；会话完成后，关闭Socket。客户端不需要指定打开的端口，通常临时的、动态的分配一个1024以上的端口。

服务器端代码：

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

**import** java**.**net**.**Socket**;**

public class ServerDemo **{**

public static void main**(**String**[]** args**)** **{**

System**.**out**.**println**(**"hello"**);**

Server server **=** **new** Server**();**

**}**

**}**

class Server **{**

private ServerSocket ss**;**

private Socket socket**;**

private BufferedReader in**;**

private PrintWriter out**;**

private boolean runFlag**;**

public Server **()** **{**

runFlag **=** **true;**

**try** **{**

ss **=** **new** ServerSocket**(**3333**);**

//同一时间只能跟一个客户端通信

**while** **(**runFlag**)** **{**

System**.**out**.**println**(**"wait for client:"**);**

socket **=** ss**.**accept**();**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),true);**

String line **=** in**.**readLine**();**

System**.**out**.**println**(**"localAddress:"

**+**socket**.**getLocalAddress**()+**"\nLocalPort:"

**+**socket**.**getLocalPort**()+**"\nromote Port:"**+**socket**.**getPort**());**

out**.**println**(**"client input is:"**+**line**);**

System**.**out**.**println**(**"your input is:"**+**line**);**

out**.**close**();**

in**.**close**();**

socket**.**close**();**

**}**

ss**.**close**();**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

**}**

客户端代码：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**BufferedWriter**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**OutputStreamWriter**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**io**.**Writer**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

public class ClientDemo **{**

public static void main**(**String**[]** args**){**

Client client **=** **new** Client**();**

**}**

**}**

class Client **{**

private Socket socket**;**

BufferedReader in**;**

PrintWriter out**;**

public Client**()** **{**

**try** **{**

socket **=** **new** Socket**(**"127.0.0.1"**,** 3333**);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**());**

BufferedReader systemIn **=** **new** BufferedReader**(new** InputStreamReader**(**System**.**in**));**

System**.**out**.**print**(**"enter message:"**);**

out**.**println**(**systemIn**.**readLine**());**

out**.**flush**();**

**new** Thread**()** **{**

public void run**()** **{**

**try** **{**

System**.**out**.**println**(**"get message:"**+**in**.**readLine**());**

systemIn**.**close**();**

out**.**close**();**

in**.**close**();**

socket**.**close**();**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**};**

**}.**start**();**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

**}**

## 多个客户同时连接

在实际的网络环境里，同一时间只对一个用户服务是不可行的。一个优秀的网络服务程序除了能处理用户的输入信息，还必须能够同时响应多个客户端的连接请求。在java中，实现以上功能特点是非常容易的。

设计原理：

socketServer.accept 和read都是阻塞的。

主程序监听一端口，等待客户接入；同时构造一个线程类，准备接管会话。当一个Socket会话产生后，将这个会话交给线程处理，然后主程序继续监听。运用Thread类或Runnable接口来实现是不错的办法。

服务器端：

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

public class ServerDemo **{**

private static final int SERVER\_PORT **=** 3333**;**

private static ServerSocket serverSocket**;**

public static void main**(**String**[]** args**)** **{**

System**.**out**.**println**(**"hello"**);**

**try** **{**

serverSocket **=** **new** ServerSocket**(**SERVER\_PORT**);**

**}** **catch** **(**IOException e1**)** **{**

e1**.**printStackTrace**();**

**}**

**while** **(true)** **{**

**try** **{**

System**.**out**.**println**(**"wait for client:"**);**

Socket socket **=** serverSocket**.**accept**();**

createServerThread**(**socket**);**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

**}**

static void createServerThread**(**Socket socket**)** **{**

**new** Thread**()** **{**

BufferedReader in**;**

PrintWriter out**;**

int clientPort**;**

public void run**()** **{**

**try** **{**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),true);**

clientPort **=** socket**.**getPort**();**

System**.**out**.**println**(**"\nclientPort "**+**clientPort**+**" connected:\n"**);**

String line **=** in**.**readLine**();**

**while** **(!**"bye"**.**equals**(**line**)** **)** **{**

out**.**println**(**clientPort**+**"-client input is:"**+**line**);**

System**.**out**.**println**(**clientPort**+**"-client input is:"**+**line**);**

line **=** in**.**readLine**();**

**}**

System**.**out**.**println**(**"client-"**+**clientPort**+**"-close"**);**

in**.**close**();**

out**.**close**();**

socket**.**close**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**};**

**}.**start**();**

**}**

**}**

客户端：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**BufferedWriter**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**OutputStreamWriter**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**io**.**Writer**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

public class ClientDemo **{**

public static void main**(**String**[]** args**){**

Client client **=** **new** Client**();**

**}**

**}**

class Client **{**

private Socket socket**;**

BufferedReader in**;**

PrintWriter out**;**

private int PORT**=**3333**;**

public Client**()** **{**

**try** **{**

socket **=** **new** Socket**(**"104.224.136.25"**,** PORT**);**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**());**

//无法立即收到消息 ，故需要在子线程中接受

**new** Thread**()** **{**

public void run**()** **{**

**try** **{**

**while** **(true)** **{**

System**.**out**.**println**(**"get message:"**+**in**.**readLine**());**

**}**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**};**

**}.**start**();**

BufferedReader systemIn **=** **new** BufferedReader**(new** InputStreamReader**(**System**.**in**));**

System**.**out**.**print**(**"enter message,bye to exit:"**);**

String sysIn **=** systemIn**.**readLine**();**

**while** **(!**sysIn**.**equals**(**"bye"**))** **{**

out**.**println**(**sysIn**);**

out**.**flush**();**

sysIn **=** systemIn**.**readLine**();**

**}**

out**.**println**(**sysIn**);**

out**.**flush**();**

systemIn**.**close**();**

out**.**close**();**

in**.**close**();**

socket**.**close**();**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

**}**

用于测试远程主机和本地客户端的通信：

服务器：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**ServerSocket**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**SocketImpl**;**

**import** java**.**util**.**Date**;**

**import** javax**.**sound**.**sampled**.**Port**;**

**import** org**.**omg**.**Messaging**.**SyncScopeHelper**;**

public class MyServer **{**

private static final int Port **=** 7888**;**

static ServerSocket serverSocket**;**

public static void main**(**String**[]** args**)** **{**

**try** **{**

serverSocket **=** **new** ServerSocket**(**Port**);**

System**.**out**.**println**(**"wait:"**);**

**while** **(true)** **{**

Socket socket **=** serverSocket**.**accept**();**

manageSocket**(**socket**);**

**}**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

static void manageSocket**(**Socket socket**)** **{**

**new** Thread**(){**

public void run**()** **{**

boolean runFlat **=** **true;**

BufferedReader in **=** **null;**

PrintWriter out **=** **null;**

int remotePort **=** socket**.**getPort**();**

System**.**out**.**println**(**"\n"**+**remotePort**+**"--connected"**);**

**try** **{**

in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

String res **=** in**.**readLine**();**

System**.**out**.**println**(**"get date:"**+**res**);**

Date date **=** **new** Date**();**

System**.**out**.**println**(**"Server:"**+**date**.**toString**());**

**}** **catch** **(**IOException e1**)** **{**

e1**.**printStackTrace**();**

**}**

int ts **=** 0**;**

**while** **(**ts **<** 6**)** **{**

**try** **{**

out **=** **new** PrintWriter**(**socket**.**getOutputStream**());**

out**.**println**(**"send to "**+**remotePort**+**":"**+** ts**++);**

out**.**flush**();**

**try** **{**

sleep**(**1000**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**try** **{**

in**.**close**();**

out**.**close**();**

socket**.**close**();**

System**.**out**.**println**(**remotePort**+**"--close"**);**

**}** **catch** **(**IOException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**};}.**start**();**

**}**

**}**

客户端：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** java**.**util**.**Date**;**

public class Tclient **{**

public static void main**(**String**[]** args**)** **{**

InClient client **=** **new** InClient**();**

**}**

**}**

class InClient **{**

private static final String HOST **=** "104.224.136.25"**;**

private static final int PORT **=** 7888**;**

Socket socket**;**

BufferedReader reader**;**

PrintWriter writer**;**

private boolean runFlag **=** **true;**

public InClient**()** **{**

**try** **{**

socket **=** **new** Socket**(**HOST**,** PORT**);**

reader **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

writer **=** **new** PrintWriter**(**socket**.**getOutputStream**());**

System**.**out**.**println**(**"start"**);**

Date date **=** **new** Date**();**

writer**.**println**(**date**);**

writer**.**flush**();**

int ts **=** 0**;**//when recive 10 times end

**while** **(**runFlag **)** **{**

String res **=** reader**.**readLine**();**

**if** **(**res **==** **null)** **break;**

System**.**out**.**println**(**res**);**

**}**

reader**.**close**();**

writer**.**close**();**

socket**.**close**();**

System**.**out**.**println**(**"end"**);**

**}** **catch** **(**UnknownHostException e**)** **{**

e**.**printStackTrace**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

## 能够在2台电脑上传输文件的程序：

服务器端：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**FileOutputStream**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStream**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**OutputStreamWriter**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**ServerSocket**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**SocketImpl**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**Scanner**;**

**import** javax**.**sound**.**sampled**.**Port**;**

**import** org**.**omg**.**Messaging**.**SyncScopeHelper**;**

public class ServerGetFile **{**

private static final int Port **=** 7888**;**

static ServerSocket serverSocket**;**

static String savaPath **=** "./"**;**

public static void main**(**String**[]** args**)** **{**

**try** **{**

serverSocket **=** **new** ServerSocket**(**Port**);**

System**.**out**.**print**(**"enter the path file will be saved,such as \"F:/\":"**);**

savaPath **=** **new** Scanner**(**System**.**in**).**nextLine**();**

System**.**out**.**println**(**"wait:"**);**

**while** **(true)** **{**

Socket socket **=** serverSocket**.**accept**();**

manageSocket**(**socket**);**

**}**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

static void manageSocket**(**Socket socket**)** **{**

**new** Thread**(){**

public void run**()** **{**

InputStream in **=** **null;**

FileOutputStream fout **=** **null;**

int remotePort **=** socket**.**getPort**();**

System**.**out**.**println**(**"\n"**+**remotePort**+**"--connected"**);**

**try** **{**

in **=** socket**.**getInputStream**();**

BufferedReader reader **=** **new** BufferedReader**(new** InputStreamReader**(**in**,** "utf-8"**));**

String filePath **=** reader**.**readLine**();**

System**.**out**.**println**(**"get file:"**+**filePath**);**

String**[]** names **=** filePath**.**split**(**"/"**);**

int namesLen **=** names**.**length**;**

PrintWriter writer **=** **new** PrintWriter**(new** OutputStreamWriter**(**socket**.**getOutputStream**(),** "utf-8"**));**

writer**.**println**(**"OK"**);**

writer**.**flush**();**

String savaFile **=** savaPath **+** names**[**namesLen**-**1**];**

fout **=** **new** FileOutputStream**(**savaFile**);**

System**.**out**.**println**(**"save file to:"**+**savaFile**);**

byte**[]** buf **=** **new** byte**[**1024**];**

int len **=** 0**;**

**while** **((**len **=** in**.**read**(**buf**))** **!=** **-**1**)** **{**

fout**.**write**(**buf**,** 0**,** len**);**

**}**

in**.**close**();**

reader**.**close**();**

writer**.**close**();**

fout**.**close**();**

socket**.**close**();**

System**.**out**.**println**(**remotePort**+**"--close"**);**

**}** **catch** **(**IOException e1**)** **{**

e1**.**printStackTrace**();**

**}**

**};}.**start**();**

**}**

**}**

客户端：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**FileInputStream**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**InputStreamReader**;**

**import** java**.**io**.**OutputStream**;**

**import** java**.**io**.**OutputStreamWriter**;**

**import** java**.**io**.**PrintStream**;**

**import** java**.**io**.**PrintWriter**;**

**import** java**.**net**.**Socket**;**

**import** java**.**net**.**UnknownHostException**;**

**import** java**.**util**.**Date**;**

**import** java**.**util**.**Scanner**;**

public class ClientSendFile **{**

public static void main**(**String**[]** args**)** **{**

InClient client **=** **new** InClient**();**

**}**

**}**

class InClient **{**

private static String Host **=** "104.224.136.25"**;**

private static final int PORT **=** 7888**;**

Socket socket**;**

OutputStream out**;**

PrintWriter writer**;**

private boolean runFlag **=** **true;**

String fileName**;**

public InClient**()** **{**

Scanner sin **=** **new** Scanner**(**System**.**in**);**

System**.**out**.**print**(**"enter the host IP,or d use default:"**);**

String temp **=** sin**.**nextLine**();**

**if** **(!**temp**.**equals**(**"d"**))** **{**

Host **=** temp**;**

**}**

System**.**out**.**print**(**"enter file name,like\"C:/a.txt\":"**);**

fileName **=** sin**.**next**();**

**try** **{**

socket **=** **new** Socket**(**Host**,** PORT**);**

out **=** socket**.**getOutputStream**();**

PrintWriter writer **=** **new** PrintWriter**(new** OutputStreamWriter**(**out**,** "utf-8"**));**

writer**.**println**(**fileName**);**

writer**.**flush**();**

BufferedReader reader **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**(),** "utf-8"**));**

**if** **(**reader**.**readLine**().**equals**(**"OK"**))**

System**.**out**.**println**(**"start transfer!"**);**

FileInputStream fin **=** **new** FileInputStream**(**fileName**);**

byte**[]** buffer **=** **new** byte**[**1024**];**

int len **=** 0**;**

**while** **((**len **=** fin**.**read**(**buffer**))** **!=** **-**1**)** **{**

out**.**write**(**buffer**,** 0**,** len**);**

**}**

out**.**close**();**

reader**.**close**();**

writer**.**close**();**

socket**.**close**();**

System**.**out**.**println**(**"end"**);**

**}** **catch** **(**UnknownHostException e**)** **{**

e**.**printStackTrace**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**