[**Java Socket编程----通信是这样炼成的**](http://www.cnblogs.com/rocomp/p/4790340.html)

**Java最初是作为网络编程语言出现的，其对网络提供了高度的支持，使得客户端和服务器的沟通变成了现实，而在网络编程中，使用最多的就是Socket。像大家熟悉的QQ、MSN都使用了Socket相关的技术。下面就让我们一起揭开Socket的神秘面纱。**

**Socket编程**

**一、网络基础知识（参考计算机网络）**

**关于计算机网络部分可以参考相关博客：**

**《TCP/IP协议栈及OSI参考模型详解》**

<http://wangdy.blog.51cto.com/3845563/1588379>

**1、两台计算机间进行通讯需要以下三个条件：**

**IP地址、协议、端口号**

**2、TCP/IP协议：**

**是目前世界上应用最为广泛的协议，是以TCP和IP为基础的不同层次上多个协议的集合，也成TCP/IP协议族、或TCP/IP协议栈**

**TCP：Transmission Control Protocol 传输控制协议**

**IP：Internet Protocol 互联网协议**

**3、TCP/IP五层模型**

**应用层：HTTP、FTP、SMTP、Telnet等**

**传输层：TCP/IP**

**网络层：**

**数据链路层：**

**物理层：网线、双绞线、网卡等**

**4、IP地址**

**为实现网络中不同计算机之间的通信，每台计算机都必须有一个唯一的标识---IP地址。**

**32位二进制**

**5、端口**

**区分一台主机的多个不同应用程序，端口号范围为0-65535，其中0-1023位为系统保留。**

**如：HTTP：80  FTP：21 Telnet：23**

**IP地址+端口号组成了所谓的Socket，Socket是网络上运行的程序之间双向通信链路的终结点，是TCP和UDP的基础**

**6、Socket套接字：**

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

**Socket原理机制：**

**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**

**7、Java中的网络支持**

**针对网络通信的不同层次，Java提供了不同的API，其提供的网络功能有四大类：**

**InetAddress:用于标识网络上的硬件资源，主要是IP地址**

**URL：统一资源定位符，通过URL可以直接读取或写入网络上的数据**

**Sockets：使用TCP协议实现的网络通信Socket相关的类**

**Datagram:使用UDP协议，将数据保存在用户数据报中，通过网络进行通信。**

**二、InetAddress**

**InetAddress类用于标识网络上的硬件资源，标识互联网协议(IP)地址。**

**该类没有构造方法**

[复制代码](javascript:void(0);)

1 //获取本机的InetAddress实例

2 InetAddress address =InetAddress.getLocalHost();

3 address.getHostName();//获取计算机名

4 address.getHostAddress();//获取IP地址

5 byte[] bytes = address.getAddress();//获取字节数组形式的IP地址,以点分隔的四部分

6

7 //获取其他主机的InetAddress实例

8 InetAddress address2 =InetAddress.getByName("其他主机名");

9 InetAddress address3 =InetAddress.getByName("IP地址");

[复制代码](javascript:void(0);)

**三、URL类**

**1、URL(Uniform Resource Locator)统一资源定位符，表示Internet上某一资源的地址，协议名：资源名称**

1. [复制代码](javascript:void(0);)
2. 1 //创建一个URL的实例
3. 2 URL baidu =new URL("http://www.baidu.com");
4. 3 URL url =new URL(baidu,"/index.html?username=tom#test");//？表示参数，#表示锚点
5. 4 url.getProtocol();//获取协议
6. 5 url.getHost();//获取主机
7. 6 url.getPort();//如果没有指定端口号，根据协议不同使用默认端口。此时getPort()方法的返回值为 -1
8. 7 url.getPath();//获取文件路径
9. 8 url.getFile();//文件名，包括文件路径+参数
10. 9 url.getRef();//相对路径，就是锚点，即#号后面的内容

10 url.getQuery();//查询字符串，即参数

[复制代码](javascript:void(0);)

**2、使用URL读取网页内容**

**通过URL对象的openStream()方法可以得到指定资源的输入流，通过流能够读取或访问网页上的资源**

1. [复制代码](javascript:void(0);)
2. 1 //使用URL读取网页内容
3. 2 //创建一个URL实例
4. 3 URL url =new URL("http://www.baidu.com");
5. 4 InputStream is = url.openStream();//通过openStream方法获取资源的字节输入流
6. 5 InputStreamReader isr =newInputStreamReader(is,"UTF-8");//将字节输入流转换为字符输入流,如果不指定编码，中文可能会出现乱码
7. 6 BufferedReader br =newBufferedReader(isr);//为字符输入流添加缓冲，提高读取效率
8. 7 String data = br.readLine();//读取数据
9. 8 while(data!=null){
10. 9 System.out.println(data);//输出数据
11. 10 data = br.readerLine();
12. 11 }
13. 12 br.close();
14. 13 isr.colose();

14 is.close();

[复制代码](javascript:void(0);)

**四、TCP编程**

**1、TCP协议是面向连接的、可靠的、有序的、以字节流的方式发送数据，通过三次握手方式建立连接，形成传输数据的通道，在连接中进行大量数据的传输，效率会稍低**

**2、Java中基于TCP协议实现网络通信的类**

**客户端的Socket类**

**服务器端的ServerSocket类**

http://images2015.cnblogs.com/blog/740688/201509/740688-20150907234728090-211300057.jpg

**3、Socket通信的步骤**

**① 创建ServerSocket和Socket**

**② 打开连接到Socket的输入/输出流**

**③ 按照协议对Socket进行读/写操作**

**④ 关闭输入输出流、关闭Socket**

**4、服务器端：**

**① 创建ServerSocket对象，绑定监听端口**

**② 通过accept()方法监听客户端请求**

**③ 连接建立后，通过输入流读取客户端发送的请求信息**

**④ 通过输出流向客户端发送乡音信息**

**⑤ 关闭相关资源**

1. [复制代码](javascript:void(0);)
2. 1 /\*\*
3. 2 \* 基于TCP协议的Socket通信，实现用户登录，服务端
4. 3 \*/
5. 4 //1、创建一个服务器端Socket，即ServerSocket，指定绑定的端口，并监听此端口
6. 5 ServerSocket serverSocket =newServerSocket(10086);//1024-65535的某个端口
7. 6 //2、调用accept()方法开始监听，等待客户端的连接
8. 7 Socket socket = serverSocket.accept();
9. 8 //3、获取输入流，并读取客户端信息
10. 9 InputStream is = socket.getInputStream();
11. 10 InputStreamReader isr =newInputStreamReader(is);
12. 11 BufferedReader br =newBufferedReader(isr);
13. 12 String info =null;
14. 13 while((info=br.readLine())!=null){
15. 14 System.out.println("我是服务器，客户端说："+info)；
16. 15 }
17. 16 socket.shutdownInput();//关闭输入流
18. 17 //4、获取输出流，响应客户端的请求
19. 18 OutputStream os = socket.getOutputStream();
20. 19 PrintWriter pw = new PrintWriter(os);
21. 20 pw.write("欢迎您！");
22. 21 pw.flush();
23. 22
24. 23
25. 24 //5、关闭资源
26. 25 pw.close();
27. 26 os.close();
28. 27 br.close();
29. 28 isr.close();
30. 29 is.close();
31. 30 socket.close();

31 serverSocket.close();

[复制代码](javascript:void(0);)

**5、客户端：**

**① 创建Socket对象，指明需要连接的服务器的地址和端口号**

**② 连接建立后，通过输出流想服务器端发送请求信息**

**③ 通过输入流获取服务器响应的信息**

**④ 关闭响应资源**

1. [复制代码](javascript:void(0);)
2. 1 //客户端
3. 2 //1、创建客户端Socket，指定服务器地址和端口
4. 3 Socket socket =newSocket("localhost",10086);
5. 4 //2、获取输出流，向服务器端发送信息
6. 5 OutputStream os = socket.getOutputStream();//字节输出流
7. 6 PrintWriter pw =newPrintWriter(os);//将输出流包装成打印流
8. 7 pw.write("用户名：admin；密码：123");
9. 8 pw.flush();
10. 9 socket.shutdownOutput();
11. 10 //3、获取输入流，并读取服务器端的响应信息
12. 11 InputStream is = socket.getInputStream();
13. 12 BufferedReader br = new BufferedReader(new InputStreamReader(is));
14. 13 String info = null;
15. 14 while((info=br.readLine())!null){
16. 15 System.out.println("我是客户端，服务器说："+info);
17. 16 }
18. 17
19. 18 //4、关闭资源
20. 19 br.close();
21. 20 is.close();
22. 21 pw.close();
23. 22 os.close();

23 socket.close();

[复制代码](javascript:void(0);)

**6、应用多线程实现服务器与多客户端之间的通信**

**① 服务器端创建ServerSocket，循环调用accept()等待客户端连接**

**② 客户端创建一个socket并请求和服务器端连接**

**③ 服务器端接受苦读段请求，创建socket与该客户建立专线连接**

**④ 建立连接的两个socket在一个单独的线程上对话**

**⑤ 服务器端继续等待新的连接**

[复制代码](javascript:void(0);)

1 //服务器线程处理

2 //和本线程相关的socket

3 Socket socket =null;

4 //

5 public serverThread(Socket socket){

6 this.socket = socket;

7 }

8

9 publicvoid run(){

10 //服务器处理代码

11 }

12

13 //============================================

14 //服务器代码

15 ServerSocket serverSocket =newServerSocket(10086);

16 Socket socket =null;

17 int count =0;//记录客户端的数量

18 while(true){

19 socket = serverScoket.accept();

20 ServerThread serverThread =newServerThread(socket);

21 serverThread.start();

22 count++;

23 System.out.println("客户端连接的数量："+count);

24 }

[复制代码](javascript:void(0);)

**五、UDP编程**

**UDP协议（用户数据报协议）是无连接的、不可靠的、无序的,速度快**

**进行数据传输时，首先将要传输的数据定义成数据报（Datagram），大小限制在64k，在数据报中指明数据索要达到的Socket（主机地址和端口号），然后再将数据报发送出去**

**DatagramPacket类:表示数据报包**

**DatagramSocket类：进行端到端通信的类**

**1、服务器端实现步骤**

**① 创建DatagramSocket，指定端口号**

**② 创建DatagramPacket**

**③ 接受客户端发送的数据信息**

**④ 读取数据**

1. [复制代码](javascript:void(0);)
2. 1 //服务器端，实现基于UDP的用户登录
3. 2 //1、创建服务器端DatagramSocket，指定端口
4. 3 DatagramSocket socket =new datagramSocket(10010);
5. 4 //2、创建数据报，用于接受客户端发送的数据
6. 5 byte[] data =newbyte[1024];//
7. 6 DatagramPacket packet =newDatagramPacket(data,data.length);
8. 7 //3、接受客户端发送的数据
9. 8 socket.receive(packet);//此方法在接受数据报之前会一致阻塞
10. 9 //4、读取数据
11. 10 String info =newString(data,o,data.length);
12. 11 System.out.println("我是服务器，客户端告诉我"+info);
13. 12
14. 13
15. 14 //=========================================================
16. 15 //向客户端响应数据
17. 16 //1、定义客户端的地址、端口号、数据
18. 17 InetAddress address = packet.getAddress();
19. 18 int port = packet.getPort();
20. 19 byte[] data2 = "欢迎您！".geyBytes();
21. 20 //2、创建数据报，包含响应的数据信息
22. 21 DatagramPacket packet2 = new DatagramPacket(data2,data2.length,address,port);
23. 22 //3、响应客户端
24. 23 socket.send(packet2);
25. 24 //4、关闭资源

25 socket.close();

[复制代码](javascript:void(0);)

**2、客户端实现步骤**

**① 定义发送信息**

**② 创建DatagramPacket，包含将要发送的信息**

**③ 创建DatagramSocket**

**④ 发送数据**

1. [复制代码](javascript:void(0);)
2. 1 //客户端
3. 2 //1、定义服务器的地址、端口号、数据
4. 3 InetAddress address =InetAddress.getByName("localhost");
5. 4 int port =10010;
6. 5 byte[] data ="用户名：admin;密码：123".getBytes();
7. 6 //2、创建数据报，包含发送的数据信息
8. 7 DatagramPacket packet = newDatagramPacket(data,data,length,address,port);
9. 8 //3、创建DatagramSocket对象
10. 9 DatagramSocket socket =newDatagramSocket();
11. 10 //4、向服务器发送数据
12. 11 socket.send(packet);
13. 12
14. 13
15. 14 //接受服务器端响应数据
16. 15 //======================================
17. 16 //1、创建数据报，用于接受服务器端响应数据
18. 17 byte[] data2 = new byte[1024];
19. 18 DatagramPacket packet2 = new DatagramPacket(data2,data2.length);
20. 19 //2、接受服务器响应的数据
21. 20 socket.receive(packet2);
22. 21 String raply = new String(data2,0,packet2.getLenth());
23. 22 System.out.println("我是客户端，服务器说："+reply);
24. 23 //4、关闭资源

24 socket.close();

[复制代码](javascript:void(0);)

**六、注意问题：**

**1、多线程的优先级问题：**

**根据实际的经验，适当的降低优先级，否侧可能会有程序运行效率低的情况**

**2、是否关闭输出流和输入流：**

**对于同一个socket，如果关闭了输出流，则与该输出流关联的socket也会被关闭，所以一般不用关闭流，直接关闭socket即可**

**3、使用TCP通信传输对象，IO中序列化部分**

**4、socket编程传递文件，IO流部分**

