### 什么是Socket编程

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个Socket。Socket称为“套接字”，用于描述IP地址和端口，可以用来实现不同计算机之间的通信。在Internet上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Socket正如其英文原意那样，像一个多孔插座。一台主机犹如布满各种插座的房间，每个插座有一个编号。

### 连接过程

根据连接启动的方式以及本地套接字要连接的目标，套接字之间的连接过程可以分为三个步骤：服务器监听，客户端请求，连接确认。

第一步：服务监听：是服务端套接字并不定位具体的客户端套接字，而是出于等

待连接的状态，实时监控网络状态。

第二步：客户端请求：是指由客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

第三步：连接确认：是指当服务器端套接字监听到客户端套接字的连接请求，它就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，连接就建立好了。而服务器端套接字继续处于监听状态，继续接受其他客户端套接字的连接请求。

### 实现Socket程序

创建服务器端步骤：

1. 调用ServerSocket(int port)，创建一个服务器端套接字，并绑定到指定端口。
2. 调用accept()，监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字对象。
3. 调用该Socket类对象的getOutPutStream()和getInputStream()。获取输出流和输入流，开始网络数据的发送和接收。
4. 关闭资源，客户端访问结束，关闭通信套接字。

**public class** MyServerSocket {  
  
 **public static void** main(String[] args) {  
 ServerSocket serverSocket = **null**;  
 Socket clientSocket = **null**;  
 BufferedReader br = **null**;  
 **try** {  
 *//创建服务器套接字，并绑定端口号：8080* serverSocket = **new** ServerSocket(8080);  
 *//开始监听网络，此时程序处理等待状态，接收客户端消息* clientSocket = serverSocket.accept();  
 *//接收客户端套接字消息* br = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));  
  
 String temp = **null**;  
 *//打印消息* **while** ((temp = br.readLine()) != **null**) {  
 System.***out***.println(temp);  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 *//关闭资源* **if** (br != **null**) {  
 **try** {  
 br.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (clientSocket != **null**) {  
 **try** {  
 clientSocket.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (serverSocket != **null**) {  
 **try** {  
 serverSocket.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

创建客户端步骤：

1. 创建Socket，根据指定服务端IP地址或端口号构造Socket类对象。若服务器端响应，则建立客户端到服务器的通信线路。若连接失败，会出现异常。
2. 打开连接到Socket的输入/输出流，使用getInputStream()方法获得输入流，使用getOutStream()方法获得输出流，进行数据传输。
3. 按照一定的协议对Socket进行读/写操作，通过输入流读取服务器放入线路的信息（但不能读取自己放入线路的信息，通过输出流将信息写入线程。
4. 关闭资源，断开客户端到服务器的连接，释放线路。

**public class** MyClientSocket {  
  
 **public static void** main(String[] args) {  
 Socket clientSocket = **null**;  
 PrintWriter out = **null**;  
 **try** {  
 *//创建客户端套接字* clientSocket = **new** Socket(**"localhost"**, 8080);  
 *//创建输出流对象* out = **new** PrintWriter(clientSocket.getOutputStream());  
 *//消息* String msg = **"Hello World!"**;  
 *//传送消息* out.print(msg);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 *//关闭资源* **if** (out != **null**) {  
 out.close();  
 }  
 **if** (clientSocket != **null**) {  
 **try** {  
 clientSocket.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}