JavaSocket编程（三） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/138/2021_2022_JavaSocket_c97_138761.htm 服务器Sockets 列表9.2是一个服务器应用程序的一部分. 列表9.2 一个简单的服务器程序 /** * 一个监听端口并提供HTML文档的程序. */ class SimpleWebServer { public static void main(String args[]) { ServerSocket serverSocket = null. Socket clientSocket = null. int connects = 0. try { { // 建立一个服务器socket serverSocket = new ServerSocket(80, 5). while (connects { // 等待连接 clientSocket = serverSocket.accept(). //服务连接 ServiceClient(clientSocket). connects . } serverSocket.close(). } catch (IOException ioe) { System.out.println("Error in SimpleWebServer: " ioe). } } public static void ServiceClient(Socket client) throws IOException { DataInputStream inbound = null. DataOutputStream outbound = null. try { // 得到IO流 inbound = new DataInputStream(client.getInputStream()). outbound = new DataOutputStream(client.getOutputStream()). //格式化输出(回应头和很少的HTML文档) StringBuffer buffer = PrepareOutput(). String inputLine. while ((inputLine = inbound.readLine()) != null) { //如果到了HTTP请求的尾部,就发送回应 if (inputLine.equals("")) { outbound.writeBytes(buffer.toString()). break. } } } finally { // 清除 System.out.println("Cleaning up connection: " client). tln("Cleaning up connection: " client). outbound.close(). inbound.close(). client.close(). client.close(). } } 服务器Sockets 服务器并不是主动地建立连接.相反地,他们是被动地监听一个客户端的连接请示然后给他们服务.服务器是由

类ServerSocket来建立的.下面的程序建立了一个服务器端socket 并把它绑定到80端口: ServerSocket serverSocket = new ServerSocket(80, 5). 第一个参数是服务器要监听的端口.第二个参数是可选的.API文档中说明了这是一个监听时间,但是在传统的socket程序中第二个参数是监听深度.一个服务器可以同时接收多个连接请求,但是每次只能处理一个.监听堆是一个无回答的连接请求队列.上面的请求建立一个连接来处理最后五个请求.如果省略了后面的一个参数,则默认值是50. ServerSocket serverSocket = new ServerSocket(80, 5). 一旦socket建立了并开始监听连接,进来的连接将会建立并放在监听堆.accetp()方法把在堆中的连接取出来. Socket clientSocket = serverSocket.accept(). 这个方法返回一个用来与来访者对话的客户端连接.服务器本身不可能建立对话,相反地,服务器socket会使用accept()方法来产生一个新的socket.服务器socket依旧打开并排列新的连接请求.与客户端socket一样,下面的一步建立输入和输出流: DataInputStream inbound = new DataInputStream(clientSocket.getInputStream()). DataOutputStream outbound = new DataOutputStream(clientSocket.getOutput Stream()). 一般的I/O操作可以在新建的流中运用.在服务器回应前它等待客户端发送一个空白的行.当会话结束时,服务器关闭流和客户端socket.如果在队列中没有请示将会出现什么情况呢?那个方法将会等待一个的到来.这个行为叫阻塞.accept()方法将会阻塞服务器线程直到一个呼叫到来.当5个连接处理完闭之后,服务器退出.任何的在队列中的呼叫将会被取消. 所有的服务器都要有以下的基本的步骤: 1.建立一个服务器socket并开始监听. 2.使用accept()方法取得新的连

接. 3.建立输入和输出流. 4.在已有的协议上产生会话. 5.关闭客户端流和socket. 6.回到第二步或者到第七步. 7.关闭服务器socket.