用java.nio.*进行网络编程 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022__E7_94_A8javani_c104_144445.htm 对相关类的简单介绍 java.nio.*, 据说它提供了一些更加底层的一些功能，如:类似windows环境下的AsyncSocket类的异步操作的功能，能显著降低server端程序的线程管理开销。 因为大多数应用是建立在TCP之上，所以在此只说说SocketChannel，ServerSocketChannel，Selector和ByteBuffer这几个类.前三个最终都源自channel类。而channel类，可以理解为在具体I/O或文件对象之上抽象的一个操作对象，我们通过操作channel的读写达到对其对应的文件或I/O对象(包括socket)读写的目的。读写的内容在内存中放在ByteBuffer类提供的缓冲区。总而言之，channel作为一个桥梁，连接了I/O对象和内存中的ByteBuffer，实现了I/O的更高效的存取。 一个基于TCP的服务器端程序，必然有个侦听端和若干个通信端，它们在nio中由对应的ServerSocketChannel和SocketChannel类来实现。为了达到异步I/O操作的目的，需要Selector类，它能检测到I/O对象的状态。 SocketChannel类是抽象类，通过调用它的静态函数open()，可生成一个SocketChannel对象，该对象对应一个java.net.Socket，可通过SocketChannel.socket()获得，而其对应的Socket也可通过调用函数getChannel()得到已建立的相应SocketChannel。SocketChannel与它的socket是一一对应的。SocketChannel的操作与Socket也很相似. ServerSocketChannel也是通过调用它的静态函数open()生成的，只是它不能直接调用bind()函数来绑定一个地址，需要它对应的ServerSocket来完成绑定工作，一般

可按如下步骤做: ServerSocketChannel ssc = new ServerSocketChannel.open(). ssc.socket().bind(InetSocketAddress(host，port)). 罗嗦了半天，还是看看最简单的C/S实现吧，服务器提供了基本的回射（echo）功能，其中提供了较详细的注释。源码分析1.服务器端：//////////////////////////AsyncServer.java// by zztudou@163.com//////////////////////////import java.nio.channels.SocketChannel.import java.nio.channels.ServerSocketChannel.import java.nio.channels.SelectionKey.import java.nio.channels.Selector.import java.nio.ByteBuffer.import java.nio.channels.SelectableChannel.import java.nio.channels.spi.SelectorProvider.import java.net.ServerSocket.import java.net.Socket.import java.net.InetSocketAddress.import java.net.SocketAddress.import java.util.Iterator.import java.util.LinkedList.import java.io.IOException.class AsyncServer implements Runnable{ private ByteBuffer r_buff = ByteBuffer.allocate(1024). private ByteBuffer w_buff = ByteBuffer.allocate(1024). private static int port = 8848. public AsyncServer(){ new Thread(this).start(). } public void run(){ try{ //生成一个侦听端 ServerSocketChannel ssc = ServerSocketChannel.open(). //将侦听端设为异步方式 ssc.configureBlocking(false). //生成一个信号监视器 Selector s = Selector.open(). //侦听端绑定到一个端口 ssc.socket().bind(new InetSocketAddress(port)). //设置侦听端所选的异步信号OP_ACCEPT ssc.register(s,SelectionKey.OP_ACCEPT).

System.out.println("echo server has been set up ....."). while(true){ int n = s.0select(). if (n == 0) {//没有指定的I/O事件发生 continue. } Iterator it = s.0selectedKeys().iterator(). while (it.hasNext()) { SelectionKey key = (SelectionKey) it.next(). if (key.isAcceptable()) {//侦听端信号触发 ServerSocketChannel server = (ServerSocketChannel) key.channel(). //接受一个新的连接 SocketChannel sc = server.accept(). sc.configureBlocking(false). //设置该socket的异步信号OP_READ:当socket可读时， //触发函数DealwithData(). sc.register(s,SelectionKey.OP_READ). } if (key.isReadable()) {//某socket可读信号 DealwithData(key). } it.remove(). } } } catch(Exception e){ e.printStackTrace(). } } public void DealwithData(SelectionKey key) throws IOException{ int count. //由key获取指定socketchannel的引用 SocketChannel sc = (SocketChannel)key.channel(). r_buff.clear(). //读取数据到r_buff while((count = sc.read(r_buff))> 0) . //确保r_buff可读 r_buff.flip(). w_buff.clear(). //将r_buff内容拷入w_buff w_buff.put(r_buff). w_buff.flip(). //将数据返回给客户端 EchoToClient(sc). w_buff.clear(). r_buff.clear(). } public void EchoToClient(SocketChannel sc) throws IOException{ while(w_buff.hasRemaining()) sc.write(w_buff). } public static void main(String args[]){ if(args.length > 0){ port = Integer.parseInt(args[0]). } new AsyncServer(). }}在当前目录下运行： javac AsynServer.java后，若无编译出错，接下来可运行：java AsynServer 或 java AsynServer ×××（端口号）上述服务程序在运行时，可指定其侦听端口，否则程序会取8848为默认端口。 2.客户端的简单示

例://////////////////////AsyncClient.java// by zztudou@163.com//////////////////////import java.nio.channels.SocketChannel.import java.net.InetSocketAddress.import java.nio.ByteBuffer.import java.nio.channels.Selector.import java.nio.channels.SelectionKey.import java.io.IOException.import java.io.BufferedReader.import java.io.InputStreamReader.class AsyncClient{ private SocketChannel sc. private final int MAX_LENGTH = 1024. private ByteBuffer r_buff = ByteBuffer.allocate(MAX_LENGTH). private ByteBuffer w_buff = ByteBuffer.allocate(MAX_LENGTH). private static String host . private static int port = 8848. public AsyncClient(){ try { InetSocketAddress addr = new InetSocketAddress(host,port). //生成一个socketchannel sc = SocketChannel.open(). //连接到server sc.connect(addr). while(!sc.finishConnect()) . System.out.println("connection has been established!..."). while(true){ //回射消息 String echo. try{ System.err.println("Enter msg youd like to send: "). BufferedReader br = new BufferedReader(new InputStreamReader(System.in)). //输入回射消息 echo = br.readLine(). //把回射消息放入w_buff中 w_buff.clear(). w_buff.put(echo.getBytes()). w_buff.flip(). }catch(IOException ioe){ System.err.println("sth. is wrong with br.readline() "). } //发送消息 while(w_buff.hasRemaining()) sc.write(w_buff). w_buff.clear(). //进入接收状态 Rec(). //间隔1秒 Thread.currentThread().sleep(1000). } }catch(IOException ioe){ ioe.printStackTrace(). } catch(InterruptedException ie){

ie.printStackTrace(). } } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com