多线程在JAVAME应用程序中的使用 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__E5_A4_9A_E7_BA_BF_E7_A8_8B_E5_c104_144984.htm 多线程技术是java me中的关键技术，应用十分频繁，尤其是在游戏中。但是对于新手来说，又容易忽略或错误的使用多线程，导致程序堵塞，而无法响应用户的输入请求。本文将仅就多线程技术在java me应用程序中的使用展开讨论。本文主要包含如下部分：多线程与联网 多线程与拍照 timer与timertask 多线程与联网手机中，所有的midlet程序都是由application manager software（ams）管理的。当midlet初始化后，ams就会调用midlet的startapp()方法，此时midlet就进入了acitive状态。在java me中有些操作可能会导致程序堵塞，比如连接网络等。如果这些操作与主程序在同一个主线程中完成，那么由于堵塞会造成程序长时间无法返回，也就无法响应用户的其他操作了。所以，如果我们在commandaction()中进行了联网的操作，则会造成如上所述的情况。下面，将通过一个例子来演示如上的情况，并使用多线程最终解决此问题。这是一个“ echo message ”实例，手机端向服务器端发送一条消息，服务器得到此消息后直接返回给手机端。首先，创建一个networkconnection类来封装联网相关的操作，这样，midlet中只需调用此类中的方法就可以完成联网的操作。代码如下：以下是引用片段：/* * networkconnection.java * * created on 2006年7月20日, 下午2:54 * */ package nju.hysteria.thread.connection. import java.io.datainputstream. import java.io.dataoutputstream. import java.io.ioexception. import

javax.microedition.io.connector. import javax.microedition.io.httpconnection. /** * * @author magic */ public class networkconnection { private static final string url = http://localhost:8080/thread/. private httpconnection httpconnection. private string message. public networkconnection(string message) { this.message = message. connect(). } /** * connect to web server. * */ public void connect(){ try { httpconnection = (httpconnection) connector.open(url). httpconnection.setrequestmethod(httpconnection.post). } catch (ioexception ex) { system.out.println(can not open connection!). ex.printstacktrace(). } } /** * send message to server. * @throws java.io.ioexception */ public void sendmessage() throws ioexception{ dataoutputstream out = httpconnection.opendataoutputstream(). out.writeutf(message). out.close(). } /** * receive message from server. * @throws java.io.ioexception * @return */ public string receivemessage() throws ioexception { datainputstream in = httpconnection.opendatainputstream(). string message = in.readutf(). in.close(). return message. } /** * close connection. */ public void close(){ if(httpconnection!=null){ try { httpconnection.close(). } catch (ioexception ex) { ex.printstacktrace(). } } } }构造函数的参数是将要被发送的消息。服务器端的代码在此不再列出，详细请见本文的源代码。接着，我们写一个midlet调用类中的方法。malconnectionmidlet在commandaction()方法中直接调用networkconnection中的方法，而没有重新创建一个线程。代

码如下：以下是引用片段：/* * malconnectionmidlet.java * * created on 2006年7月20日, 下午2:53 */ package nju.hysteria.thread.connection. import java.io.ioexception. import javax.microedition.midlet.*. import javax.microedition.lcdui.*. /** * * @author magic * @version */ public class malconnectionmidlet extends midlet implements commandlistener { private display display. private textbox text. private command showcommand. public malconnectionmidlet(){ display = display.getdisplay(this). text = new textbox(message,请使用‘ 问候 ’命令发送消息,100,textfield.any). showcommand = new command(问候,command.screen,1). text.addcommand(showcommand). text.setcommandlistener(this). } public void startapp() { display.setcurrent(text). } public void pauseapp() { } public void destroyapp(boolean unconditional) { } public void commandaction(command command, displayable displayable) { if(command==showcommand){ /** * 在当前的线程中直接进行联网操作，造成程序堵塞。 */ string message = null. networkconnection connection = new networkconnection(hello world). try { connection.sendmessage(). message = connection.receivemessage(). connection.close(). } catch (ioexception ex) { ex.printstacktrace(). } text.setstring(message). } } } 当用户按下“ 问候 ”命令时，就会向服务器发送“ hello world ”的消息，然后再得到服务器返回的消息，并显示在textbox中。 运行程序，如图1所示。当用户按下“ 问候 ”命令后，程序就僵死了，并在console窗口中得到如下警告：图1以下是引用片段：warning: to avoid potential deadlock, operations that

may block, such as networking, should be performed in a different thread than the commandaction() handler.这就是因为没有使用多线程造成的。下面，就来看看如何使用多线程来解决此问题。[nextpage]新建类networkthread，它继承在thread，并将原先commandaction()中发送，接受消息的操作移到此类中完成。代码如下：以下是引用片段：/* * networkthread.java * * created on 2006年7月20日, 下午4:16 * */ package nju.hysteria.thread.connection. import java.io.ioexception. import javax.microedition.lcdui.textbox. /** * * @author magic */ public class networkthread extends thread { private networkconnection connection. private textbox text. public networkthread(textbox text) { super(). this.text = text. } public void run() { string message = null. connection = new networkconnection(hello world). try { connection.sendmessage(). message = connection.receivemessage(). connection.close(). } catch (ioexception ex) { ex.printstacktrace(). } text.setstring(message). } }同时，修改原先的midlet，得到新的midlet：以下是引用片段：/* * connectionmidlet.java * * created on 2006年7月20日, 下午2:53 */ package nju.hysteria.thread.connection. import java.io.ioexception. import javax.microedition.midlet.*. import javax.microedition.lcdui.*. /** * * @author magic * @version */ public class connectionmidlet extends midlet implements commandlistener { private display display. private textbox text. private command showcommand. public connectionmidlet(){ display = display.getdisplay(this). text = new textbox(message,请使用‘问候’命令发送消息,100,textfield.any). showcommand = new command(问

候,command.screen,1). text.addcommand(showcommand). text.setcommandlistener(this). } public void startapp() { display.setcurrent(text). } public void pauseapp() { } public void destroyapp(boolean unconditional) { } public void commandaction(command command, displayable displayable) { if(command==showcommand){ /** * 创建新的线程完成联网操作 */ (new networkthread(text)).start(). } } }此时，在commandaction()中，我们创建了新的线程来完成消息的发送和接受。运行程序，可以成功接受到返回的消息。