在JavaME中通过蓝牙发现设备并传送文件Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/584/2021_2022__E5_9C_A8JavaME_E4_c104_584688.htm 在Java ME设备上执行蓝牙应用程序的首要步骤之一就是发现过程(discovery process)。简而言之就是，发现过程就是带有蓝牙的设备互相找到彼此的过程，然后一起携手找出它们各个可以支持的服务。下一步就是要学习如何在这些两两设备之间传送数据。 在本篇技术小文章中，我将向你展示如何创建一个可以互相查找设备的一个MIDlet，然后让用户发送一个简单的消息到其中一个被找到的设备中。我已经在Nokia N95的机器上测试并核实了这个MIDlet的工作了，通过启用蓝牙支持，它可以连接到一个运行Windows Vista的电脑上。 我把整个过程分成以下几个步骤： 1. 开始发现过程。 2. 查询在发现过程中找到的设备所支持的服务。 3. 使用支持服务的URL开始并处理一个OBEX数据交换。 以下各段将详细说明这些步骤。在这些步骤中遵循代码片段可以查阅这个MIDlet的整个源代码。源代码可以在最后的Resources下的压缩文件中获得。 第一步：开始发现过程 发现过程是用来告诉本地蓝牙堆栈可以和在附近任何蓝牙设备进行配对。在这个MIDlet中，这个堆栈可以通过你的设备提供者所提供的JSR 82来完成。这个发现过程通过发现在本地设备中的代理来开始的，如以下代码所示： // get the local discovery agent agent = LocalDevice.getLocalDevice().getDiscoveryAgent(). // start the inquiry for general unlimited inquiry agent.startInquiry(DiscoveryAgent.GIAC, this). 一旦发现代理启

动发现过程，它将在一个执行DiscoveryListener接口的类上调用各种调回方法。就我们而言，这是我们的MIDlet类。 具体来说，必须执行这个接口的四个方法，其中两个是在发现阶段我们所感兴趣的：deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) 和 inquiryCompleted(int discType)。这两个方法处理一个设备的发现并完成发现过程。在以下所展示的来自MIDlet的代码中，一旦它们被发现或是当程序结束的时候，我们使用这些方法来添加我们的设备上的UI。 public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) { try { // add the devices using the friendly names listofDevices.append(btDevice.getFriendlyName(false), null). // add to the devices hashtable devices.put(new Long(listofDevices.size()), btDevice). } catch(Exception ex) { handleError(ex). } } public void inquiryCompleted(int discType) { // once the inquiry is completed, show the list of devices to the user if(listofDevices.size() == 0) { display.setCurrent(nothing, noteBox). } else { display.setCurrent(listofDevices). } } 第二步： 在已发现的设备上开始服务发现(service discovery) 由于本文的目的是让数据从我们的MIDlet中传输到一个兼容的设备上，我们需要在已发现的设备上找到这些服务来实现这个目标。为了达到这个目的，我们需要在服务发现过程中定义正确的属性和UUIDs。以下代码将显示如何来做这个： agent.searchServices(null, new UUID[] {new UUID(0x1105L)}, // we want the OBEX PUSH Profile device, this). 正如你所猜到的，这些代码使用我们以前用过的本地代理来查找设备。我们不是在一组特定的属性之后，所以我们需要用null作为第一个参数，但是UUID必须

是OBEX PUSH配置文件，因为这是传输数据的一个最开放式的方法。 我们讲到DiscoveryListener接口有两个其他的方法可以用来利用发现的服务。这两个方法是 servicesDiscovered(int transID, ServiceRecord[] servRecord) 和serviceSearchCompleted(int transID, int respCode)。正如名字所显示的那样，第一个方法是每当一个服务被发现时被调用，第二个方法当服务过程结束时被调用。 每当一个服务被发现的时候，我们需要找到每个设备上的特定的URL服务连接。这个URL连接将使OBEX连接用于我们的数据传送，而且由蓝牙硬件，设备地址组成。在以下的代码中，这个URL连接取自于servicesDiscovered方法： String connURL = servRecord[i].getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false). 通过这个URL连接，现在我们可以把传输数据的程序移动到已发现得设备上。 第三步：使用PBEX PUT传送数据 在这个MIDlet中，我们将运行用户输入一些文本作为一个消息，然后用已发现的设备或是服务来传送它们。为了做这个，我们需要在前一步骤中得到的URL连接(当然，消息数据是作为一个String的)。 // open a client session ClientSession clientSession = (ClientSession) Connector.open(connURL). // connect using no headers clientSession.connect(null). if(rHeaders.getResponseCode() != ResponseCodes.OBEX_HTTP_OK) { // the connection could not be established handleError(new Exception("Remote client returned invalid response code: " rHeaders.getResponseCode())). return. } // if we are here, then response code was ok // create a new set of headers HeaderSet headers = clientSession.createHeaderSet().

headers.setHeader(HeaderSet.LENGTH, new Long(noteBox.getString().length())). headers.setHeader(HeaderSet.NAME, "myNote.txt"). headers.setHeader(HeaderSet.TYPE, "text/plain"). // create an operation using the headers we have just created Operation op = clientSession.put(headers). // on this operation, create the output stream OutputStream out = op.openOutputStream(). // and send the note out.write(noteBox.getString().getBytes()). 为了发送该数据，client session被打开，而且建立一个空标题的连接。在这一点上，你的目标设备要求来自一个新设备的数据接收的确认。如果你以前从来没有配对的设备，也可能要求你提供密钥。 一旦连接建立，解释数据目的地一些标题会被创建，而且一个新的操作通过这些标题也会被创建。这个操作是通过OutputStream来进行传送数据的。 接收到的消息放在目标设备上的默认蓝牙交换文件夹中。