<http://www.jianshu.com/p/7c92cfc5ee6b>

权限：

<uses-permission android:name="android.permission.BLUETOOTH" />

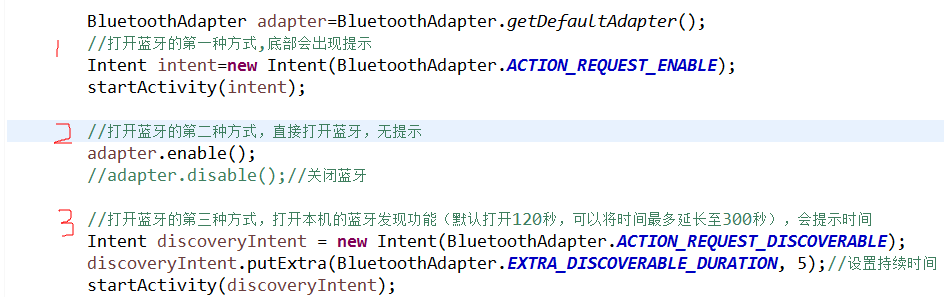
<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

6.0后,需要再加两个权限

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

小米的定位权限,需进权限管理中去打开

1. 打开本地蓝牙设备



BluetoothAdapter类，代表了本设备的蓝牙适配器对象，通过它可以完成蓝牙的开关、扫描、设置等。

通过BluetoothAdapter.getDefaultAdapter()可以获得设备蓝牙适配器的实例，如果没有返回null。

1. 搜索其他蓝牙设备

adapter.startDiscovery()

startDiscovery方法不是一个方法，而是一个异步方法，顾名思义，就是说此方法开启了一个子线程，工作是在子线程中进行的。

startDiscovery方法的过程：

1. 系统发送BluetoothAdapter.ACTIOIN\_DISCOVERY\_STARTED的广播。
2. 搜索其他蓝牙设备（只要是蓝牙设备，不论是什么系统上面的，我蓝牙是NOKIA的蓝牙，照样可以搜索的到），注意，只要找到一个设备就发送一个BluetoothDevice.ACTION\_FOUND的广播，接着又发现一个设备，又发送一个BluetoothDevice的广播，如此反复，而我们从广播接收器中就可以得到这个BluetoothDevice对象了。
3. 系统发送BluetoothAdapter.ACTION\_FINISHED的广播。整个过程耗时12秒的时长。

注：取消搜索可以调用cancelDiscovery

（请求Discovery后，系统开始搜索蓝牙设备，在这个过程中，系统会发送以下三个广播：ACTION\_DISCOVERY\_START：开始搜索

ACTION\_DISCOVERY\_FINISHED：搜索结束

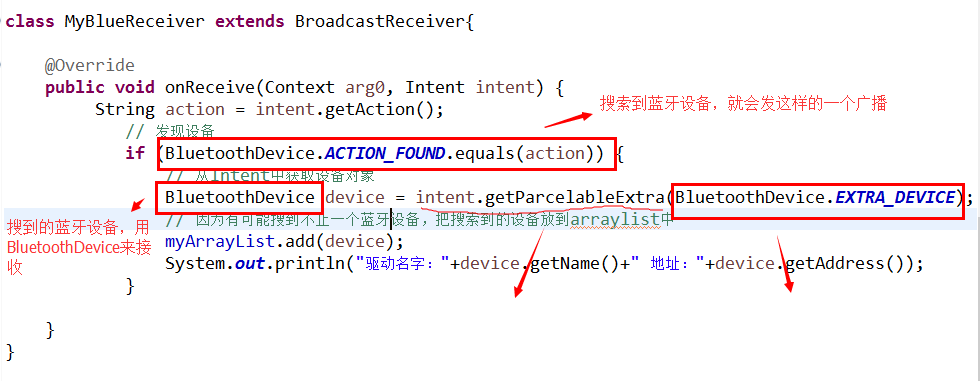
ACTION\_FOUND：找到设备，这个Intent中包含两个extra fields：EXTRA\_DEVICE和EXTRA\_CLASS，分别包含BluetooDevice和BluetoothClass。

）

建立一个过滤器，过滤对应的action，注册这个过滤器。(当然,这个广播也可以接收如连接,断开蓝牙等的广播)



实现广播接收处理



# “Parcelable的使用————通过Intent传递对象”

1. 获得device后，获得socket

BluetoothDevice 类，此为对应的远程蓝牙 Device

　　createRfcommSocketToServiceRecord（） ：创建该 Device 的 socket 。

　　BluetoothSocket 类

　　connect（） ：请求连接蓝牙。

　　getInputStream（） ：得到输入流，用于接收远程方信息。

　　getOutputStream（） ：得到输出流，发送给远程方的信息。

　　close（） ：关闭蓝牙连接。

　　InputStream 类：

　　read（byte［］） ：以阻塞方式读取输入流。

　　OutputStream 类：

write（byte［］） ：将信息写入该输出流，发送给远程。

* Server 端

通过BluetoothAdapter的listenUsingRfcommWithServiceRecord(String, UUID)方法获得BluetoothServerSocket对象  
的实例，然后socket就会通过accept()监听客户端的连接的状态

* Client 端

使用BluetoothDevice的createRfcommSocketToServiceRecord(UUID)方法获得BluetoothSocket对象的实例  
然后调用connect()方法，这时server端会监听到这个请求，之后就建立连接，然后就可以进行通信了

补充:

1. blueAdapter的方法

BluetoothAdapter的扫描状态：  
SCAN\_MODE\_CONNECTABLE 该蓝牙设备可以扫描其他蓝牙设备

SCAN\_MODE\_CONNECTABLE\_DISCOVERABLE 该蓝牙设备不仅可以扫描其他设备，也可以被其他设备扫描到

SCAN\_MODE\_NONE 该蓝牙设备不可扫描和被扫描

公共方法：

Public boolean enable() 打开蓝牙适配器

Public boolean disable() 关闭蓝牙适配器

Public boolean startDiscovery() 扫描其他蓝牙设备

Public boolean cancelDiscovery() 取消扫描蓝牙设备

Public boolean isDiscovery() 是否处在扫描过程中

Public String getName() 获取蓝牙设备名称

Public String getAddress() 获取蓝牙设备的硬件地址（MAC地址）

Public boolean setName(String name) 设置名称

Public Set<BluetoothDevice> getBondedDevices() 获取与该蓝牙设备所有绑定的远程蓝牙信息，以BluetoothDevice类实例返回。若蓝牙设备没开启，会返回一个空集合

Public static boolean checkBluetoothAdress(String address) 验证蓝牙的MAC地址是否有效

Public BluetoothDevice getRemoteDevice(String address) 通过给出的MAC地址去创建一个BluetoothDevice类实例。如果MAC不能识别，会抛出异常

1. 蓝牙相关的广播：

注:String ACTION\_STATE\_CHANGED = "android.bluetooth.adapter.action.STATE\_CHANGED"

黄色部分用于BroadcastReceiver中识别,蓝色用于过滤器

1.状态改变的广播

String ACTION\_STATE\_CHANGED = "android.bluetooth.adapter.action.STATE\_CHANGED"

通过int state = intent.getIntExtra(BluetoothAdapter.EXTRA\_STATE,BluetoothAdapter.STATE\_OFF);来获得当前蓝牙状态的广播

State的状态:

Int STATE\_OFF=10 蓝牙已经关闭

Int STATE\_ON=12 蓝牙已经打开

Int STATE\_TURNING\_OFF=13 蓝牙正在关闭中

Int STATE\_TURNING\_ON=11 蓝牙正在打开中

1. 扫描的广播

//开始扫描

String ACTION\_DISCOVERY\_STARTED = "android.bluetooth.adapter.action.DISCOVERY\_STARTED";

//扫描结束

String ACTION\_DISCOVERY\_FINISHED= "android.bluetooth.adapter.action.DISCOVERY\_FINISHED";

//扫描到蓝牙设备

String ACTION\_FOUND = "android.bluetooth.device.action.FOUND"

通过

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

来获得扫描到的设备

3.配对的广播

//有其他蓝牙设备请求配对

String ACTION\_PAIRING\_REQUEST = "android.bluetooth.device.action.PAIRING\_REQUEST";

//配对状态改变的广播

String ACTION\_BOND\_STATE\_CHANGED="android.bluetooth.device.action.BOND\_STATE\_CHANGED";

通过

//当前的配对的状态

int state=intent.getIntExtra(BluetoothDevice.EXTRA\_BOND\_STATE,

BluetoothDevice.BOND\_NONE);

//前一次的配对状态

int state=intent.getIntExtra(BluetoothDevice.EXTRA\_PREVIOUS\_BOND\_STATE,

BluetoothDevice.BOND\_NONE);

//配对的设备

BluetoothDevice device=intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

配对结果:

int BOND\_NONE = 10; 配对没有成功

int BOND\_BONDING = 11; 配对中

int BOND\_BONDED = 12; 配对成功

1. 连接广播(配对是为了排除恶意连接请求)

String ACTION\_CONNECTION\_STATE\_CHANGED="android.bluetooth.adapter.action.CONNECTION\_STATE\_CHANGED";

通过

//当前的连接状态

int state = intent.getIntExtra(BluetoothAdapter.EXTRA\_CONNECTION\_STATE, BluetoothAdapter.ERROR);

//前一次的连接状态

int state=intent.getIntExtra(BluetoothAdapter.EXTRA\_PREVIOUS\_CONNECTION\_STATE, BluetoothAdapter.ERROR);

//连接的设备信息

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

获得连接结果:

int STATE\_DISCONNECTED = 0; //未连接

int STATE\_CONNECTING = 1; //连接中

int STATE\_CONNECTED = 2; //连接成功

1. 蓝牙设备名字变化广播

//本地设备的蓝牙设备的名称变化 `BluetoothAdapter`

String ACTION\_LOCAL\_NAME\_CHANGED="android.bluetooth.adapter.action.LOCAL\_NAME\_CHANGED";

// 远程设备的名称的变化 `BluetoothDevice`

String ACTION\_NAME\_CHANGED = "android.bluetooth.device.action.NAME\_CHANGED" ;

问题;

BluetoothSocket.connect失败 read failed, socket might closed or timeout, read ret: -1

<http://blog.csdn.net/jason0539/article/details/17782035>

1. 调用connect()时没有同时做设备搜索，如果在搜索设备，该连接尝试会显著变慢，容易导致连接失败。
2. UUID错误,对于某些连接,如手机连接PC服务端,其UUID是固定的
3. 服务端没有做好相应的等待连接请求的准备(传统蓝牙连接,和网络通讯是一样的,必须要有主从设备之分)
4. 如果

二 连接成功,但socket的信息却没有

原因:发送数据时,不了对socket进行关闭,或输出流进行close