Simple Bluetooth chat application in Android

Hong Thai · Sunday, October 26, 2014 · Bluetooth · Bluetooth API · Hong Thai

Firstly, take a glance at Bluetooth technology offering on Wikipedia:

> Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs). Invented by ericsson senior Jaap Haartsen in 1994[2] it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

According to this, we can build a local ad-hoc network (LAN) by connecting devices over Bluetooth. The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. Android provides access to the Bluetooth functionality through the Bluetooth API. This article explains how to use the Bluetooth API to perform bluetooth activities (connect to other Bluetooth devices, scanning paired or get and discovering bluetooth devices, transfer data between devices, ...) by building a simple chat application.

Please, let's begin. Create a new project in your workspace and name it as you like. Create a new package and name it as you like. Open AndroidManifest.txt and add the following permissions:

Requesting Bluetooth permissions

In order to use Bluetooth services, please add `BLUETOOTH` permission to your `AndroidManifest.txt`. Moreover, if you want to use discover possible devices nearby, add `BLUETOOTH_ADMIN` permission also (optional):

```
<uses-permission android:name="android.permission.BLUETOOTH"
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
```

Checking if device supports Bluetooth

Now to check whether Bluetooth is supported on device or not, we can object of `BluetoothAdapter` class. If `BluetoothAdapter` returns null, your device doesn't support Bluetooth. This is how to do it:

```
BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter
if (bluetoothAdapter == null) {
    // Device does not support Bluetooth
    Toast.makeText(this, "Bluetooth is not available!", T
    finish();// Closes the app if Bluetooth service is n
}
```

Check if Bluetooth is Enabled

The Bluetooth service is disabled by default. If you want to use Bluetooth, you must request to turn it on:

```
if (!bluetoothAdapter.isEnabled()) {
    // Request to turn on Bluetooth
    Intent enableIntent = new Intent(BluetoothAdapter.ACTI
    startActivityForResult(enableIntent, REQUEST_ENABLE_BL
}
```

You should set this code in `onCreate()` to ensure that your app always check the connection when it launches! The meaning of `startActivity` is explained later.

Discovering Bluetooth devices

In android, available devices is not discoverable by default. To turning them into discoverable, we need to send a request to Bluetooth service. This can be done by adding your region `BroadcastReceiver` with `BluetoothDevice.ACTION_FOUND` action after completing discovery, system will broadcast `BluetoothDevice.ACTION_FOUND` Intent. This broadcast contains extra fields with information about the discovered device.

To discover available devices, we need to register a `BroadcastReceiver` and call discover devices to an `BluetoothAdapter` and check by `isDiscovering()`:

```
// Create a BroadcastReceiver for ACTION_FOUND
private final BroadcastReceiver mReceiver = new BroadcastR
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = (BluetoothDevice)
            if (device.getBondState() != BluetoothDevice
                mArrayAdapter.add(device.getName() + "\n"
            } else {
                mArrayAdapter.add(device.getName() + "\n
            }
        }
    }
};

// Register the receiver for broadcasts
IntentFilter filter = new IntentFilter(BluetoothDevice.ACT
registerReceiver(mReceiver, filter);

// Indicate scanning in the title
setProgressBarIndeterminateVisibility(true);
setTitle("Scanning...");

// Turn on sub-title for new devices
int mNewDevicesCount = 0;

// If we're already discovering, stop it
if (bluetoothAdapter.isDiscovering()) {
    bluetoothAdapter.cancelDiscovery();
}

// Request discover from BluetoothAdapter
bluetoothAdapter.startDiscovery();
```

Listing paired devices

Moreover, your devices can be connected to some other devices before, so you can listing paired devices and connect to them:

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.ge
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.ge

// If there are paired devices, add each one to the ArrayAda
if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
        pairedDevicesArrayAdapter.add(device.getName()
    }
} else {
    String noDevices = getResources().getText(R.string.none_paired
}
```

In this sample application, I write a dialog, which contains 2 `ListViews` of paired devices and discovered devices, and this is the layout:

Connecting to a device

To connect two devices, we have Client/Server server side and client side mechanisms. Since device must open the server socket and another should initiate the connection (the remote device must be discoverable).

With connection, we plan:

- Initializing an instance of `BluetoothServerSocket`. by calling the `listenUsingRfcommWithServiceRecord()`
- Listening for connections requests by calling `accept()`
- Release server socket by calling `close()`

```
private Handler mHandler = new Handler();
private Thread BluetoothServerSocketThread;

public boolean[] mRun = true;
BluetoothServerSocket tmp = null;
boolean[] mRun = true;
tmp = bluetoothAdapter.listenUsingInsecureRfcommWith
mServerSocket = tmp;
while (mRun[0]) {
    serverSocket = tmp;

public void run() {
    BluetoothSocket socket = null;
    BluetoothSocket socket;
    boolean[] mRun = true;
    while (true) {
        try {
            socket = serverSocket.accept();
        } catch (IOException e) {
            break;
        }

        // If a connection was accepted
        if (socket != null) {
            openSocket(socket, ChatService.this);
            switch (state) {
                case STATE_LISTEN:
                case STATE_CONNECTING:
                    // Situation normal. Start the connec
                    connected(socket, socket.getRemoteDev
                    break;
                case STATE_NONE:
                case STATE_CONNECTED:
                    // Either not ready or already conne
                    // Terminate new connection.
                    try {
                        socket.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    break;
                }
            }
        }
    }
}
```

As connected as a client:

- Create an instance of `BluetoothSocket`. by calling `createRfcommSocketToServiceRecord()` on `BluetoothDevice` object.
- Connect to the remote device by calling `connect()`

```
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    boolean[] mRun = true;

    public ConnectThread(BluetoothDevice device) {
        this.mmDevice = device;
        BluetoothSocket tmp = null;
        mmDevice = device;
        tmp = device.createRfcommSocketToServiceRecord
        mmSocket = tmp;
        e.printStackTrace();
    }

    socket = tmp;

    public void run() {
        setName("ConnectThread");

        // Always cancel discovery because it will slow down a co
        bluetoothAdapter.cancelDiscovery();

        // Make a connection to the BluetoothSocket
        try {
            socket.connect();
            Log.d("BluetoothChat", "Connected");
            } catch (IOException e) {
            e.printStackTrace();
            }
            connectFailed();
            return;
        }

        // Reset the ConnectThread because we're done
        synchronized (ChatService.this) {
            connectThread = null;
        }

        // Start the connected thread
        connected(socket, device);
    }

    public void cancel() {
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

This is definition of `MyUUID` constant:

```
// This unique id for your app.
// GENERATED: http://www.uuidgenerator.com/
private static final UUID MY_UUID = UUID.fromString("0ce7f3c8-20b
```

Read and write data (text messages)

Of course, after establishing connection successfully, we have the most important event of a chat application: exchange text messages. Now, each device has a connected `BluetoothSocket`. Each of them can read and write data to the stream using `inputStream()` and `outputStream()`:

```
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
            } catch (IOException e) {
            }

        inputStream = tmpIn;
        outputStream = tmpOut;
    }

    public void run() {
        byte[] buffer = new byte[1024];
        int bytes;
        // Keep listening to the InputStream
        while (true) {
            try {
                // Read from the InputStream
                bytes = inputStream.read(buffer);

                // Send the obtained bytes to the UI Activity
                mHandler.obtainMessage(Constants.MESSAGE_READ,
                    buffer).sendToTarget();
            } catch (IOException e) {
                connectionLost();
                // Start the service over to restart listening mo
                ChatService.this.start();
                break;
            }
        }
    }

    public void write(byte[] buffer) {
        try {
            mmOutStream.write(buffer);
            mHandler.obtainMessage(Constants.MESSAGE_WRITE, -1,
            } catch (IOException e) {
            }
    }

    public void cancel() {
        try {
            mmSocket.close();
            } catch (IOException e) {
            e.printStackTrace();
            }
    }
}
```

NOTE: In this sample project, type `BluetoothThread`, `ConnectThread` and `ReadWriteThread` stream this in a form named `BluetoothService`.

Application output

Some screenshots of this simple chat app:

When you start an application

When you click on scan for devices

When you click on scan for devices, the connection start run

Conclusions

I have presented some important aspects in developing a simple Bluetooth chat application. Hope this is helpful with your world! To create a text message system or more complicated, please go to Google Code bubbles API page. Moreover, read these official document pages to deep understanding about Bluetooth connection in Android:

- Bluetooth overview
- Bluetooth API

Download the CODE

Hong Thai · Author · Hong Thai

SHARE

Google+ · Facebook · Twitter · Reddit

Receive Updates from Learn Programming Together

Your Email Address... · Subscribe

YOU MAY ALSO LIKE

Android Bluetooth Technology · Android Bluetooth Technology · Bluetooth API · Bluetooth · Android · Android Bluetooth · Google Code · Google Code Bubbles · Google Code Application Debug · Google Code Application Debug · Google Code · Google Code

PREVIOUS POST
Code Review Tips: Bluetooth in Android

NEXT POST
Tips Management Application Development in Android

Comments: Learn Programming Together

Sort by Newest

Join the discussion...

LOG IN WITH · OR SIGN UP WITH DISQUS ?

Name

Hong Thai · 2 years ago
Hi, Thanks for your comment.
You can get full codes here: https://github.com/DarkPrince...
Reply · Share ›

Thanh Nguyen · 10 days ago · 1 edit
Hi, I'm a newbie, can you tell me what address please?
Reply · Share ›

Ms Kumar · 2 years ago
Hi,
Learn source of this project
Reply · Share ›

Sarah · 2 years ago
Hi Sir
In C# programming language is used for your Bluetooth messenger
Reply · Share ›

Ryan Gomez · 2 years ago
not working.. Thanks !!!!
Reply · Share ›

Alfred Gates · 2 years ago
hello everybody,
I am trying to develop simple bluetooth chat application. I have the code that was sent from you in your blog but it not responsive for me.
Thanks in advance.
1 ^ · Reply · Share ›

Pallavi Barik · 2 years ago
Hi Hong Thai,
Can u tell me what the reason why data is printed two times ? wanted to print this data only the functions have to do the job.
Thanks in advance.
Reply · Share ›

Subscribe · Privacy · Do Not Sell My Data

DISQUS

LABELS (SHOWN TOTAL)

Android (3) · Bluetooth (3) · Bluetooth API (3) · Hong Thai (3) · Google Code (2) · Google Code Application Debug (2) · Google Code Bubbles (2) · Android Bluetooth (2) · Android Bluetooth Technology (2) · Bluetooth Technology (2) · Code Review Tips (2) · Code Review Tips: Bluetooth in Android (2) · Development Application Management (2) · Development Application Management Tips (2) · Development Application Management Tips: Bluetooth in Android (2) · Development Application Management Tips: Code Review (2) · Development Application Management Tips: Code Review: Bluetooth in Android (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code (2) · Development Application Management Tips: Code Review: Bluetooth in Android: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth: Code: Bluetooth