

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
C.d.S. in Ingegneria e Scienze Informatiche, Campus di Cesena

Programmazione in Android

Comunicazione via HTTP e Bluetooth

Angelo Croatti
a.croatti@unibo.it

Sistemi Embedded e Internet of Things
A.A. 2019 – 2020

Outline

- 1 Comunicazione via HTTP
 - Richieste HTTP
 - Gestione della Connettività
- 2 Comunicazione via Bluetooth
 - API e Permessi
 - Inizializzazione
 - Ricerca dei dispositivi
 - RFCOMM e UUID
 - Esempio di implementazione



Richieste HTTP

- Per effettuare richieste HTTP (GET, POST, PUT, DELETE) in modo *semplice* si può utilizzare il client HTTP `java.net.HttpURLConnection` di Java.
- Alternativamente (in contesti più complessi), Android suggerisce l'utilizzo della libreria `Volley`
 - ▶ <https://developer.android.com/training/volley>

Permessi

```
<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission  
    android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Nota – Da Android 9.0 (API Level 28), per effettuare richieste via rete non criptate, va specificato l'attributo `android:usesCleartextTraffic="true"` in `<application>` nel file manifest. Diversamente, sono consentite solo richieste criptate con SSL o certificate.

Esempi

Richiesta GET

```
String doGet(URL url){
    HttpURLConnection c = (HttpURLConnection) url.openConnection();
    c.setRequestMethod("GET");

    if(c.getResponseCode() == HttpURLConnection.HTTP_OK)
        return readStream(c.getInputStream());
}
```

Richiesta POST

```
String doPost(URL url, byte[] payload){
    HttpURLConnection c = (HttpURLConnection) url.openConnection();
    c.setRequestMethod("POST");
    c.setDoOutput(true);
    c.getOutputStream().write(payload);

    if(c.getResponseCode() == HttpURLConnection.HTTP_OK)
        return readStream(c.getInputStream());
}
```

Richieste HTTP (note)

- Non è consentito effettuare richieste HTTP sul Main Thread
 - ▶ Devono essere demandate ad un worker thread (es. via AsyncTask)
- **Le richieste HTTP sono intrinsecamente asincrone**
 - ▶ La risposta, se pervista, può arrivare dopo un certo quantitativo di tempo oppure può scattare un timeout per indisponibilità del server o della rete

Verifica della connettività

- Il **ConnectivityManager** service consente di verificare se il device sia o meno connesso alla rete Internet

Esempio d'uso

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(  
    Context.CONNECTIVITY_SERVICE);  
  
NetworkInfo activeNetwork = cm.getActiveNetworkInfo();  
  
if(activeNetwork.isConnectedOrConnecting()){  
    //Network is reachable...  
}
```

Outline

- 1 Comunicazione via HTTP
 - Richieste HTTP
 - Gestione della Connettività
- 2 Comunicazione via Bluetooth
 - API e Permessi
 - Inizializzazione
 - Ricerca dei dispositivi
 - RFCOMM e UUID
 - Esempio di implementazione



Bluetooth in Android

- Android include il supporto per lo stack di comunicazione basato su standard Bluetooth dalle prime versioni del sistema.
 - ▶ Attraverso le Android Bluetooth API
 - ▶ Sono abilitate le connessioni BT sia di tipo point-to-point sia di tipo multipoint.
- In particolare, in Android è possibile:
 - ▶ Analizzare le frequenze radio per identificare altri dispositivi nelle vicinanze (**discovery**).
 - ▶ Connettersi a dispositivi precedentemente accoppiati (**paring**).
 - ▶ Trasferire dati da un dispositivo all'altro tramite stack BT.
 - ▶ Gestire connessioni multiple.

Android Bluetooth API

- Tutte le API Bluetooth di Android sono fornite attraverso il package `android.bluetooth.*`
- Tra le altre, le più significative sono:
 - ▶ `BluetoothAdapter`, rappresenta l'entry-point per tutte le interazioni basate su bluetooth.
 - ▶ `BluetoothDevice`, rappresenta un dispositivo bluetooth remoto con il quale sia stato instaurato un canale di comunicazione appropriato.
 - ▶ `BluetoothSocket` e `BluetoothServerSocket`, costituiscono le interfacce bluetooth per attivare canali di comunicazione TCP-like.
 - ▶ ...

Permessi

- Per utilizzare il supporto Bluetooth offerto dal sistema operativo è necessario richiedere i relativi permessi (**BLUETOOTH** e **BLUETOOTH_ADMIN**), dichiarandoli nel File Manifest.
- Da Android 6.0 in poi è necessario richiedere un ulteriore permesso (**ACCESS_FINE_LOCATION**)
 - ▶ Nota: Si tratta di un permesso “dangerous”, deve essere richiesta esplicita conferma all’utente!

Permessi per BT

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

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

<!-- Necessario per device con Android 6.0 o superiore -->
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Inizializzazione del Bluetooth I

- Tramite l'istanza del default BluetoothAdapter fornito dal sistema risulta possibile:
 1. Verificare se la comunicazione basata su stack Bluetooth è disponibile per lo specifico device,
 2. Richiederne l'abilitazione all'utente qualora il supporto al bluetooth sia disponibile ma momentaneamente disabilitato.

1. Verifica della disponibilità del BT

```
BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();  
  
if(btAdapter == null){  
    //Bluetooth is not supported in the device  
}
```

Nota – Il metodo statico `getDefaultAdapter()` restituisce l'unica istanza di BluetoothAdapter condivisa dall'intero sistema.

Inizializzazione del Bluetooth II

2. Richiesta di abilitazione del BT

```
final int REQUEST_ENABLE_BT = 1;

if (!btAdapter.isEnabled()) {
    Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(i, REQUEST_ENABLE_BT);
}
```

- Qualora il BT non sia abilitato, attraverso un apposito Intent è possibile richiederne l'abilitazione al sistema.
 - ▶ L'abilitazione deve necessariamente passare per un'autorizzazione esplicita dell'utente.
- L'intent può essere eseguito mediante l'utilizzo del metodo `startActivityForResult()`, la cui *risposta* (eventuale conferma dell'avvenuta abilitazione) può essere intercettata implementando l'apposita callback (`onActivityResult()`) sull'activity interessata.

Inizializzazione del Bluetooth III

Esempio – Inizializzazione BT

```
public class MyActivity extends Activity{

    private BluetoothAdapter btAdapter;
    private final int ENABLE_BT_REQ = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        /* ... */

        btAdapter = BluetoothAdapter.getDefaultAdapter();

        if(btAdapter == null){
            Log.e("MyApp", "BT is not available on this device");
            finish();
        }
    }
}
```

Inizializzazione del Bluetooth IV

```
if (!btAdapter.isEnabled()){
    startActivityForResult(
        new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE),
        ENABLE_BT_REQ);
}

@Override
public void onActivityResult(int reqID, int res, Intent data){
    if(reqID == ENABLE_BT_REQ && res == Activity.RESULT_OK){
        //BT enabled
    }

    if(reqID == ENABLE_BT_REQ && res == Activity.RESULT_CANCELED){
        //BT enabling process aborted
    }
}
}
```

Ricerca dei dispositivi nelle vicinanze I

- La ricerca/identificazione dei dispositivi ai quali connettersi può avvenire in due modi diversi:
 - ▶ Connettersi a dispositivi non noti a priori (**device discovery**).
 - ▶ Connettersi ai dispositivi precedentemente accoppiati (**paired devices**).

Device Discovery

- Procedura di scanning delle frequenze BT per identificare gli eventuali device presenti nel raggio di visibilità.
- Possono essere identificati tutti i dispositivi che sono visibili agli altri.
- I dispositivi identificati rispondono condividendo alcune informazioni (MAC Address, Device Name, ...).
- Per poter attivare la connessione con uno dei device identificati, deve essere eseguita l'operazione di pairing.

Ricerca dei dispositivi nelle vicinanze II

Pairing

- Procedura di accoppiamento di due dispositivi BT, a carico dell'utente.
- Consente di memorizzare su ciascun device tutte le informazioni necessarie ad attivare, eventualmente, una connessione tra i due.
- Il fatto che due dispositivi siano accoppiati non significa che siano connessi tra loro e possano scambiarsi dati su un apposito canale RFCOMM.
- L'accoppiamento è condizione necessaria ma non sufficiente per consentire la trasmissione dei dati sul canale BT da un device ad un altro.

Lista dei Dispositivi Accoppiati

- La lista dei dispositivi precedentemente accoppiati (sottoposti al pairing) è ottenibile mediante la funzione `getBondedDevices()` eseguibile sull'istanza del `BluetoothAdapter`.
 - ▶ Restituisce un `Set` di oggetti di tipo `BluetoothDevice`.

Esempio - Lista dei paired devices

```
String BT_TARGET_NAME = "mario-Phone";

BluetoothDevice targetDevice = null;

Set<BluetoothDevice> pairedList = btAdapter.getBondedDevices();

if(pairedList.size() > 0){
    for(BluetoothDevice device : pairedList){
        if(device.getName() == BT_TARGET_NAME)
            targetDevice = device;
    }
}
```

Meccanismo di Discovery I

- Tramite l'istanza del `BluetoothAdapter` è possibile avviare ed interrompere la procedura di discovery di altri dispositivi nel raggio di visibilità.
 - ▶ Rispettivamente richiamando i metodi `startDiscovery()` e `cancelDiscovery()`.
- La procedura di discovery ha una durata di circa 12 secondi ed esegue la scansione delle frequenze BT.
- Per poter ricevere le informazioni dei dispositivi identificati è necessario registrare un `BroadcastReceiver` abilitato ad intercettare ogni `IntentFilter` di tipo `BluetoothDevice.ACTION_FOUND`.
- Si tratta di una procedura ad alto consumo di energia (batteria).

Meccanismo di Discovery II

Esempio - Discovery di device nelle vicinanze

```
public class MyActivity extends Activity{

    private BluetoothAdapter btAdapter;
    private Set<BluetoothDevice> nbDevices = null;

    private final BroadcastReceiver br = new BroadcastReceiver(){
        @Override
        public void onReceive(Context context, Intent intent){
            BluetoothDevice device = null;

            if(BluetoothDevice.ACTION_FOUND.equals(intent.getAction())){
                device = intent.getParcelableExtra(BluetoothDevice.
                    EXTRA_DEVICE);
                nbDevices.add(device);
            }
        }
    };
};
```

Meccanismo di Discovery III

```
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(/* ... */);

    btAdapter = BluetoothAdapter.getDefaultAdapter();

    /* Initialize BT ... */

    registerReceiver(br, new IntentFilter(BluetoothDevice.
        ACTION_FOUND));
}

@Override
public void onStart(){
    super.onStart();
    btAdapter.startDiscovery();
}
```

Meccanismo di Discovery IV

```
@Override
public void onStop(){
    super.onStop();
    btAdapter.cancelDiscovery();
}
}
```

Nota (1). Il BroadcastReceiver deve essere registrato nel sistema (mediante **registerReceiver()** specificando l'IntentFilter che è abilitato a ricevere.

Nota (2). Al metodo onReceive del BR, il device identificato è passato mediante un Intent, dal quale è possibile recuperare l'istanza del device mediante la funzione **getParcelableExtra()** con lo specifico parametro.

Creazione del canale di comunicazione I

- Segue lo stesso schema Client-Server dei canali di comunicazioni basati sul protocollo TCP-IP via socket.
- Si attiva un canale di comunicazione basato sullo standard RFCOMM.

SERVER SIDE

1. Si attiva un Thread (MasterThread) che crea una serverSocket che attende richieste di connessioni (chiamata bloccante del metodo `accept()`).
 - ▶ La serverSocket può essere ottenuta dal BluetoothAdapter mediante la funzionalità `listenUsingRfcommWithServiceRecord()` a cui deve essere passato un generico nome per il canale e l'UUID.
2. Quando la serverSocket riceve una richiesta di connessione, questa restituisce la socket specifica su cui è stato attivato il canale.

Creazione del canale di comunicazione II

3. Tale socket è quindi passata all'istanza di un gestore della connessione bluetooth (`ConnectionManager`) il quale può essere usato sia per attendere i messaggi (dati raw) in ingresso sia per inviare messaggi al client.

CLIENT SIDE

1. Si esegue un task deputato ad eseguire il tentativo di connessione al server.
 - ▶ L'istanza della socket su cui tentare la connessione al server può essere ottenuta mediante la funzionalità `createRfcommSocketToServiceRecord()` a cui si passa l'UUID condiviso con il server.

Creazione del canale di comunicazione III

- ▶ Tale funzionalità è offerta da qualunque istanza di un `BluetoothDevice`. Nel caso specifico, deve essere utilizzata quella dell'istanza di `BluetoothDevice` che fa riferimento al device che esegue l'applicazione server.
 - ▶ Il `BluetoothDevice` associato al server può essere ottenuto, ad esempio, scegliendo il device specifico tra la lista di quelli precedentemente accoppiati.
2. Se la connessione va a buon fine, viene eseguita un'istanza dello stesso gestore di connessione presente sul server (`ConnectionManager`), per gestire la connessione lato client.

UUID – Universal Unique Identifier

- Utilizzato per identificare univocamente il canale di comunicazione.
- Può essere ottenuto mediante l'applicazione di un algoritmo standard
 - ▶ Trasforma una rappresentazione in stringa di un UUID in uno specifico valore a 128 bit.
- Come in Java, è possibile utilizzare la funzione di utilità `fromString(String rep)` fornita dalla classe `java.util.UUID` a cui deve essere passata come parametro la rappresentazione in stringa di un UUID casuale.
 - ▶ Tale rappresentazione può essere ottenuta avvalendosi del metodo `randomUUID()` fornito dalla stessa classe.
 - ▶ Online esistono diversi generatori utilizzabili (es. www.uuidgenerator.net).
- Client e Server devono condividere il medesimo UUID per poter comunicare.

UUID per Dispositivi Embedded

- Per tutti i dispositivi per i quali non è possibile effettuare un paring esplicito (ovvero confermando il codice condiviso su entrambi i device) l'UUID da utilizzare è stabilito per convenzione
- Questo vale per quasi tutti i dispositivi embedded, salvo che il relativo UUID non sia stato esplicitamente modificato

UUID standard





00001101-0000-1000-8000-00805F9B34FB

Esempio di Implementazione

Si veda l'esempio `AndroidBluetoothEx` fornito.



Riferimenti - Risorse Online

-  **Android Developers - Guide**
» <https://developer.android.com/guide/>
-  **Android Developers - API Reference**
» <https://developer.android.com/reference/>
-  **Android Developers - Samples**
» <https://developer.android.com/samples/>
-  **Android Developers - Design & Quality**
» <https://developer.android.com/design/>

Riferimenti - Libri

-  Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura
Programming Android
O'Reilly, 2011
-  Chris Haseman, Kevin Grant
Beginning Android Programming: Develop and Design
Peachpit Press, 2013
-  Ronan Schwarz, Phil Dutson, James Steele, Nelson To
The Android Developer's Cookbook : Building Applications with the Android SDK
Addison-Wesley, 2013
-  Theresa Neil
Mobile Design Pattern Gallery: UI Patterns for Smartphone App
O'Reilly, Second Edition, 2014