

LẬP TRÌNH DI ĐỘNG

Bài 8: Broadcast Receivers + Telephony

Nhắc lại bài trước

- Các API thông dụng nhất của SQLiteDatabase
 - Đóng/Tạo/Mở file cơ sở dữ liệu
 - Thực thi câu lệnh SQL
 - Làm việc với bản ghi: Tạo/Đọc/Xóa/Sửa
 - Duyệt kết quả trả về của truy vấn SELECT
- Cách làm việc với SQLiteOpenHelper
- Giới thiệu về content provider
- Cách thức sử dụng content provider để khai thác các nguồn dữ liệu cung cấp bởi hệ thống hoặc nhà phát triển thứ 3

Nội dung

1. Broadcast Receiver

1. Vòng đời của broadcast receiver
2. Tự tạo một tín hiệu broadcast
3. Viết receiver xử lý tín hiệu broadcast

2. Telephony API

1. Làm việc với điện thoại
2. SMS
 - Gửi SMS
 - Nhận SMS
 - Đọc SMS
3. Tạo và nhận cuộc gọi

Phần 1

Broadcast Receivers

Broadcast Receiver

- Broadcast receiver (gọi tắt là receiver): là một trong bốn loại thành phần cơ bản của ứng dụng android
- Receiver là một class java nhận và xử lý các sự kiện mà hệ thống (hoặc ứng dụng nào đó) phát ra
 - VD: tín hiệu báo mất wifi, tín hiệu báo cuộc gọi đến,...
- Khi hệ thống phát đi sự kiện, có 2 cơ chế phát:
 - Không thứ tự: mọi receiver đủ điều kiện đều nhận được
 - Có thứ tự: receiver nào ưu tiên hơn thì nhận trước và có thể điều chỉnh thông tin tín hiệu đến các receiver sau
 - Cơ chế này khá giống xử lý ngắt (interrupt) trong HĐH

Một số broadcast thông dụng

- Báo hệ thống khởi động xong
- Báo có package mới cài vào hoặc xóa đi
- Báo tắt máy
- Báo cắm sạc
- Báo rút sạc
- Thông báo cắm thẻ nhớ
- Thông báo rút thẻ nhớ
- Thông báo tin nhắn tới
- Thông báo có cuộc gọi đi/đến

Phần 2.1

Vòng đời của broadcast receiver

Vòng đời của broadcast receiver

1. Khi ứng dụng được cài lên thiết bị:

- Hệ thống đọc file AndroidManifest.xml để xem receiver đăng ký xử lý những sự kiện nào
- Cho receiver vào danh sách các receiver cài trên thiết bị

2. Khi xảy ra sự kiện:

- Hệ thống tạo một intent chứa thông tin về sự kiện
- Gửi intent đó đến các receiver đăng ký xử lý sự kiện
 - Nếu sự kiện không thứ tự: gửi đồng loạt đến tất cả receiver
 - Nếu sự kiện có thứ tự: gửi lần lượt đến từng receiver
- Khi intent được gửi đến receiver: hệ thống gọi hàm onReceive để xử lý intent đó

Vòng đời của broadcast receiver

- Broadcast receiver hoạt động vô cùng đơn giản
 - Chỉ cần viết duy nhất phương thức `onReceive`
 - Khi có sự kiện mà broadcast receiver đã đăng ký nhận được phát đi, thì phương thức `onReceive` của broadcast receiver đó sẽ được gọi
 - Sau khi thực thi xong phương thức này, lifecycle của receiver kết thúc
- Trường hợp broadcast có thứ tự, ta có thể gọi hàm `abortBroadcast` trong khi `onReceive` đang chạy để ngăn không cho các receiver sau nhận broadcast

Vòng đời của broadcast receiver

- Ngay khi **onReceive** kết thúc, hệ thống coi như receiver đã không còn hoạt động và có thể hủy tiến trình chứa receiver này bất cứ lúc nào, vì thế:
 - Tránh xử lý các code quá lâu trong onReceive
 - Không có xử lý bất đồng bộ, chờ callback... trong receiver (cụ thể như hiển thị Dialog, kết nối service...)
 - Nếu cứ cố dùng, hệ thống sẽ thông báo ứng dụng bị waiting quá lâu, có force close hay không, đây là lỗi nên tránh vì đa số người dùng sẽ chọn YES
- Độ ưu tiên của receiver nên từ -1000 đến 1000
`<intent-filter android:priority="1000">`

Phần 2.2

Tự tạo một tín hiệu broadcast

Tự tạo một tín hiệu broadcast

- Các tín hiệu broadcast thường do hệ thống sinh ra, nhưng LTV có thể tự tạo tín hiệu riêng
 1. Tạo một intent chứa thông tin về tín hiệu
 2. Yêu cầu hệ thống gửi broadcast phù hợp
- Android có nhiều cơ chế phát tín hiệu khác nhau
 - **Normal broadcast**: gửi tín hiệu tới tất cả các receiver
 - **Ordered broadcast**: gửi lần lượt các broadcast, vào mỗi thời điểm chỉ gửi đến một receiver
 - **Sticky broadcast**: intent được giữ lại sau khi receiver xử lý xong (sử dụng trong một số tình huống ít gặp hơn, ví dụ như cảnh báo pin yếu)

Ví dụ: gửi tín hiệu báo động đất

// chuẩn bị một intent

```
Intent intent = new Intent(NEW_EARTHQUAKE_FOUND);
```

// nạp dữ liệu về broadcast vào intent

```
intent.putExtra("date", "03/07/2016 07:00:00");
```

```
intent.putExtra("details", "London");
```

```
intent.putExtra("longitude", "-0.129098");
```

```
intent.putExtra("latitude", "51.535142");
```

```
intent.putExtra("magnitude", "3.0");
```

// chọn 1 trong 3 cách phát tín hiệu phù hợp

```
sendBroadcast(intent);
```

```
sendOrderedBroadcast(intent);
```

```
sendStickyBroadcast(intent);
```

Phần 2.3

Viết receiver xử lý tín hiệu broadcast

Sử dụng BroadcastReceiver

- Để đăng ký đối tượng receiver, có 2 cách:
 - Sử dụng **Context.registerReceiver()** để đăng ký và **Context.unregisterReceiver()** để hủy
 - Đăng ký trong file **AndroidManifest.xml** thông qua thẻ **<receiver />**
- Dùng cách thứ nhất nếu ta chỉ muốn chặn tín hiệu broadcast trong một khoảng thời gian nào đó
 - Ví dụ: khi chơi game, ta có thể chặn broadcast để biết có cuộc gọi đến hay không (để xử lý cho phù hợp)
- Cách thứ hai sử dụng nếu muốn chặn tín hiệu broadcast bất kỳ khi nào nó được phát ra

Ví dụ: viết ứng dụng auto start

- Chọn sự kiện `RECEIVE_BOOT_COMPLETED`
- Viết receiver tương ứng, trong `onReceive` chạy service ngầm hoặc khởi động activity
- Chú ý: Nếu ứng dụng của bạn ở SD card, lúc này SD card chưa sẵn sàng, vì thế ứng dụng sẽ không chạy được, trường hợp này ta chọn sang sự kiện `ACTION_EXTERNAL_APPLICATIONS_AVAILABLE`
 - Một số điện thoại (HTC) không nhận sự kiện này
 - Từ Android 3.0, phải chạy ứng dụng ít nhất một lần thì ứng dụng mới nhận sự kiện này

Ví dụ: viết ứng dụng auto start

```
<uses-permission
```

```
    android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

```
<receiver
```

```
    android:enabled="true"
```

```
    android:name=".MyApp"
```

```
    android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
```

```
        <intent-filter>
```

```
            <action
```

```
                android:name="android.intent.action.BOOT_COMPLETED" />
```

```
            <category android:name="android.intent.category.DEFAULT" />
```

```
        </intent-filter>
```

```
</receiver>
```

Ví dụ: viết ứng dụng auto start

```
public class MyApp extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Trường hợp chạy activity
        Intent i = new Intent(context, MyActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
        // Trường hợp chạy service
        Intent service = new Intent(context, MyService.class);
        context.startService(service);
    }
}
```

Phần 2

Telephony API

Phần 2.1

Làm việc với điện thoại

Làm việc với điện thoại

- Không phải thiết bị Android nào cũng có các tính năng thoại, nếu cần sử dụng một tính năng nào đó, ta cần thiết lập yêu cầu trong AndroidManifest.xml

```
<uses-feature
```

```
    android:name="android.hardware.telephony"
```

```
    android:required="true" >
```

```
</uses-feature>
```

- Chú ý: khi thiết lập thuộc tính này thì ứng dụng sẽ không cài đặt được trên các thiết bị không có phần cứng hỗ trợ điện thoại

Làm việc với điện thoại

- Muốn đọc trạng thái phone, phải được cấp quyền
`<uses-permission android:name`
 `= "android.permission.READ_PHONE_STATE" />`
- Android OS có service hệ thống để theo dõi trạng thái thoại, lấy service này bằng `getSystemService`
 - Dùng service này, ta có thể lấy thông tin của phone state, chẳng hạn như đọc số điện thoại gọi đến
- Link API của TelephonyManager:
<http://developer.android.com/reference/android/telephony/TelephonyManager.html>

Ví dụ về TelephonyManager

```
public void doRequestingCallState() {
    TelephonyManager telManager = (TelephonyManager)
        getSystemService(TELEPHONY_SERVICE);
    int callStatus = telManager.getCallState();
    String callState = null;
    switch (callStatus) {
        case TelephonyManager.CALL_STATE_IDLE:
            callState = "Phone is idle.";
            break;
        case TelephonyManager.CALL_STATE_OFFHOOK:
            callState = "Phone is in use.";
            break;
        case TelephonyManager.CALL_STATE_RINGING:
            callState = "Phone is ringing!\n";
            callState+=telManager.getLine1Number();
            break;
    }
    Toast.makeText(this, callState,
        Toast.LENGTH_LONG).show();
}
```

Làm việc với điện thoại

- Việc lắng nghe các thay đổi trong trạng thái cuộc gọi giúp ứng dụng chúng ta có phù hợp với nhu cầu của người dùng. Ví dụ như:
 - Game có thể tự động tạm dừng và lưu thông tin trạng thái khi điện thoại đổ chuông để người dùng có thể trả lời cuộc gọi một cách an toàn
 - Ứng dụng chơi nhạc có thể vặn nhỏ hoặc tạm dừng âm thanh
- Muốn tương tác tốt hơn, có thể chặn sự kiện **CallStateChange** của TelephonyManager và có cách xử lý phù hợp

Xử lý PHONE_STATE_CHANGE

```
public void doRequestingCallState_listener()
{
    TelephonyManager telManager = (TelephonyManager)
        getSystemService(TELEPHONY_SERVICE);
    telManager.listen(new PhoneStateListener() {
        public void onCallStateChanged(
            int state, String incomingNumber) {
            String newState = getCallStateString(state);
            if (state ==
                TelephonyManager.CALL_STATE_RINGING) {
                newState+="\n"+incomingNumber;
            }
            Toast.makeText(MainActivity.this, newState,
                Toast.LENGTH_LONG).show();
        }
    }, PhoneStateListener.LISTEN_CALL_STATE);
}
```

Xử lý PHONE_STATE_CHANGE

```
public String getCallStateString(int state)
{
    String callState = null;
    switch (state) {
        case TelephonyManager.CALL_STATE_IDLE:
            callState = "Phone is idle.";
            break;
        case TelephonyManager.CALL_STATE_OFFHOOK:
            callState = "Phone is in use.";
            break;
        case TelephonyManager.CALL_STATE_RINGING:
            callState = "Phone is ringing!";
            break;
    }
    return callState;
}
```



Phần 2.2

SMS

SMS – Các quyền liên quan

- Dịch vụ SMS khá đặc biệt vì liên quan tới chi phí và sự riêng tư, 3 quyền về SMS là Gửi, Nhận và Đọc

```
<uses-permission  
android:name="android.permission.SEND_SMS" />  
<uses-permission  
android:name="android.permission.RECEIVE_SMS"  
/>  
<uses-permission  
android:name="android.permission.READ_SMS" />
```

- Chú ý:
 - Cấp quyền thì ứng dụng vẫn bị chặn nếu gửi nhiều SMS
 - Không cần quyền nếu sử dụng activity bên ngoài

Gửi SMS – API

- Muốn gửi SMS cần phải có ít nhất 1 đối tượng SmsManager

```
SmsManager sms = SmsManager.getDefault();
```

- Các API gửi message

- sendTextMessage
- sendDataMessage
- sendMultipartTextMessage

```
sms.sendTextMessage(  
    "0912102165", null, "Hello!", null, null);
```

Send Pending Intent

Delivery Pending Intent

Gửi SMS – example

```
public void doSending()
{
    final SmsManager sms = SmsManager.getDefault();
    Intent msgSent = new Intent("ACTION_MSG_SENT");
    final PendingIntent pendingMsgSent =
        PendingIntent.getBroadcast(this, 0, msgSent, 0);
    registerReceiver(new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            int result = getResultCode();
            String msg="Send OK";
            if (result != Activity.RESULT_OK) {
                msg="Send failed";
            }
            Toast.makeText(MainActivity.this, msg,
                Toast.LENGTH_LONG).show();
        }
    }, new IntentFilter("ACTION_MSG_SENT"));
    sms.sendTextMessage("0987773061", null, "Hello",
        pendingMsgSent, null);
}
```

Nhận SMS – Thiết lập Receiver

- Để nhận SMS, sử dụng BroadcastReceiver để nhận thông báo có tin nhắn từ hệ thống
- Gói dữ liệu mà receiver nhận được là dãy byte được mã hóa theo chuẩn SMS PDU, Android có những class hữu ích giúp làm việc với chuẩn này
- Từ Android 1.6, broadcast SMS là loại ordered, vì thế có thể dùng abortBroadcast() để ngăn không cho SMS gửi tiếp tới các receiver khác

```
<receiver android:name="vn.mobipro.SMSRECEIVERDEMO">  
    <intent-filter >  
        <action android:name="android.provider.Telephony.SMS_RECEIVED">  
        </action>  
    </intent-filter>  
</receiver>
```

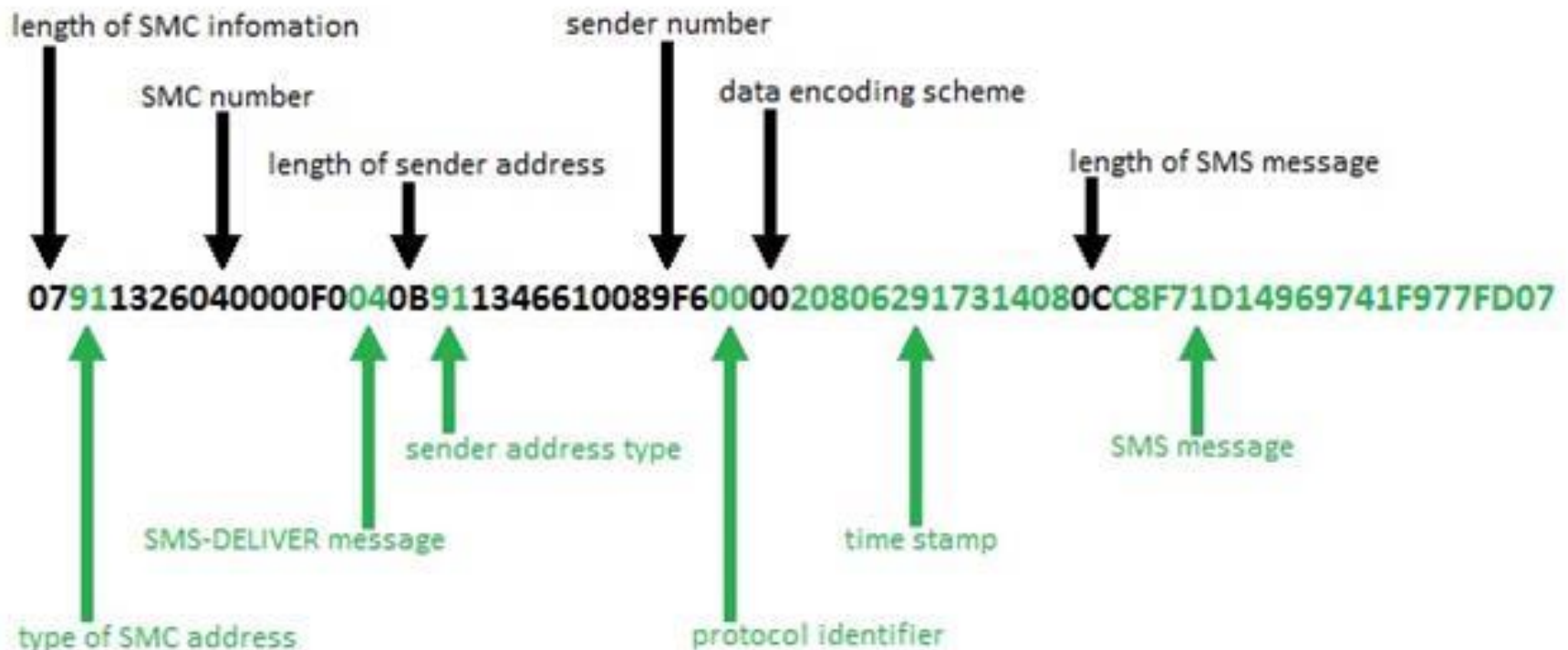
Đăng kí receiver trong AndroidManifest.xml

Nhận SMS – example

```
public void doReceiving()
{
    BroadcastReceiver rcvIncoming;
    rcvIncoming = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            Bundle data = intent.getExtras();
            if (data != null) {
                Object pdus[] =(Object[]) data.get("pdus");
                String message = "New message:\n";
                String sender = null;
                for (Object pdu : pdus) {
                    SmsMessage part = SmsMessage.
                        createFromPdu((byte[]) pdu);
                    message += part.getDisplayMessageBody();
                    sender = part.getDisplayOriginatingAddress();
                }
                Toast.makeText(MainActivity.this,
                    message + "\nFrom: " + sender, Toast.LENGTH_LONG).show();
            }
        }
    };
    registerReceiver(rcvIncoming, new IntentFilter(
        "android.provider.Telephony.SMS_RECEIVED"));
}
```



Nhận SMS – PDU encode

Example SMS PDU string



Đọc SMS

- Android OS cung cấp dữ liệu về SMS nhận được bằng ContentProvider “[content://sms/inbox](#)”
 - Sử dụng ContentProvider để lấy dữ liệu, đọc SMS từ Cursor cần nắm được cấu trúc bảng SMS
- Có thể “vọc” bằng cách lấy DB ra xem thử, trong DB có các bảng lưu dữ liệu (ví dụ bảng sms), vị trí DB: “[//data/data/com.android.provider.telephony/databases/mmssms.db](#)”

Table: 

	id	thread id	address	person	date	protocol	read	status	type	reply path present	subject	body
1	1	1		256	311321618197	0	1	-1	1			
2	2	2			311321724286	0	1	-1	1			

Đọc SMS – example

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView view = new TextView(this);
    Uri uriSMSURI = Uri.parse("content://sms/inbox");
    Cursor cur = getContentResolver().query(uriSMSURI, null,
        null, null, null);
    String sms = "";
    while (cur.moveToNext()) {
        sms += "From :" + cur.getString(2) + " : " +
            |cur.getString(11)+"\n";
    }
    view.setText(sms);
    setContentView(view);
}
```

Phần 2.3

Tạo và nhận cuộc gọi

Tạo Cuộc Gọi

- Trong thiết kế của Android OS, cuộc gọi không thể thực hiện ở background và bắt buộc phải thông qua call activity
- Cuộc gọi trong Android có thể theo 2 cách
 - Gọi gián tiếp: hiện call activity điền sẵn dữ liệu, người dùng phải bấm Send để thực hiện cuộc gọi
 - Gọi trực tiếp: hiện call activity và quay số luôn, người dùng có thể hủy cuộc gọi nếu muốn
- Sự khác nhau: ứng dụng muốn gọi trực tiếp phải được cấp quyền `android.permission.CALL_PHONE`, gọi gián tiếp thì không cần quyền

Tạo Cuộc Gọi – example

- Gọi gián tiếp:

```
Uri number = Uri.parse("tel:0912102165");  
Intent dial = new  
Intent(Intent.ACTION_DIAL, number);  
startActivity(dial);
```

- Gọi trực tiếp:

```
Uri number = Uri.parse("tel:01699362020");  
Intent call = new Intent(Intent.  
ACTION_CALL, number);  
startActivity(call);
```

Nhận Cuộc Gọi

- Tương tự như với SMS, để nhận cuộc gọi đến ứng dụng phải được cài đặt với BroadcastReceiver
- Cần thiết lập quyền READ_PHONE_STATE và đặt receiver phù hợp

```
<uses-permission android:name=
    "android.permission.READ_PHONE_STATE" />
<receiver android:name="vn.mobipro.CallReceiver" >
    <intent-filter>
        <action android:name=
            "android.intent.action.PHONE_STATE" />
    </intent-filter>
</receiver>
```

Nhận Cuộc Gọi – example

```
public class CallReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        String state = intent.getStringExtra
            (TelephonyManager.EXTRA_STATE);
        if (state.equals
            (TelephonyManager.EXTRA_STATE_RINGING)) {
            Bundle bundle = intent.getExtras();
            String phoneNr= bundle
                .getString("incoming_number");
            if (phoneNr.equals("0977113114"))
            {
                //process to Blacklist
            }
            Toast.makeText(context, phoneNr,
                Toast.LENGTH_LONG).show();
        }
    }
}
```