



LẬP TRÌNH DI ĐỘNG

Bài 11: Telephony + Media Services (1)



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. Telephony API

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

2. Media Services (part I)

1. Media API
2. MediaStore
3. Audio



Phần 1.1

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 1.2

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 acvitivity bên ngoài

Gửi SMS – API

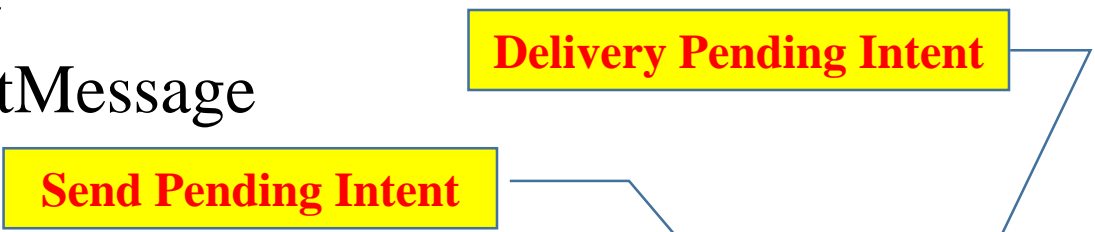
- Muốn gửi SMS cần phải có ít nhất 1 đối tượng `SmsManager`

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

- Các API gửi message

- `sendTextMessage`
- `sendDataMessage`
- `sendMultipartTextMessage`

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





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>
```

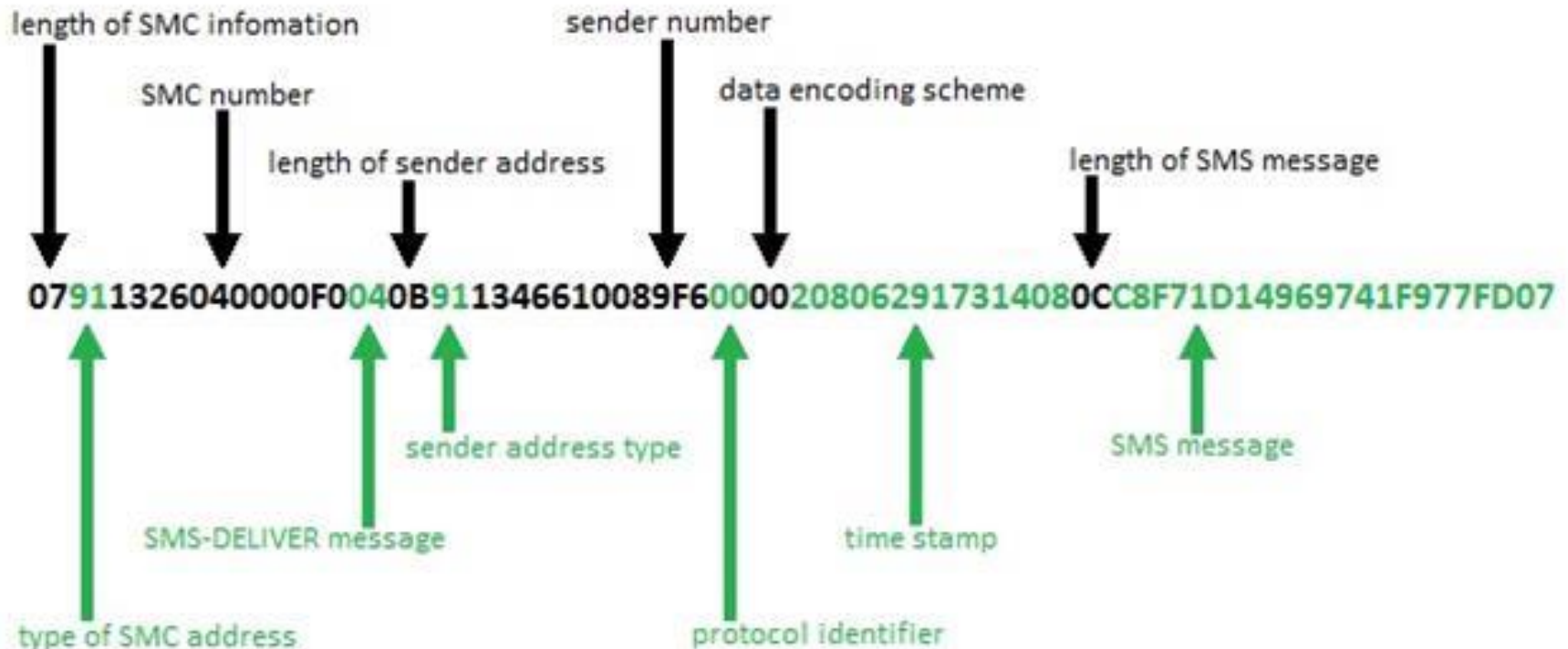
Đăng kí receiver trong AndroidManifest.xml

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 1.3

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();
        }
    }
}
```

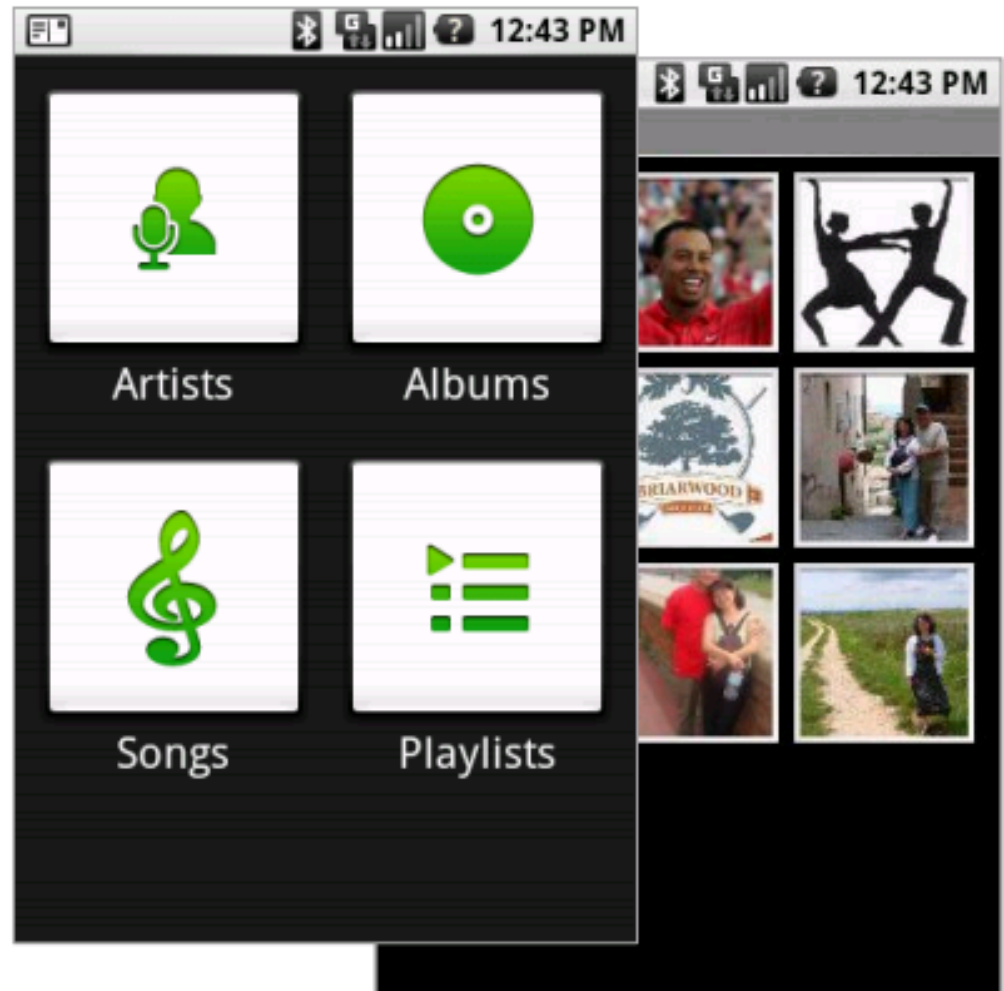



Phần 2

Media Services (phần 1)

Media Services

1. Media API
2. MediaStore
3. Audio
4. Video
5. TTS
6. Camera





Phần 2.1

Media API

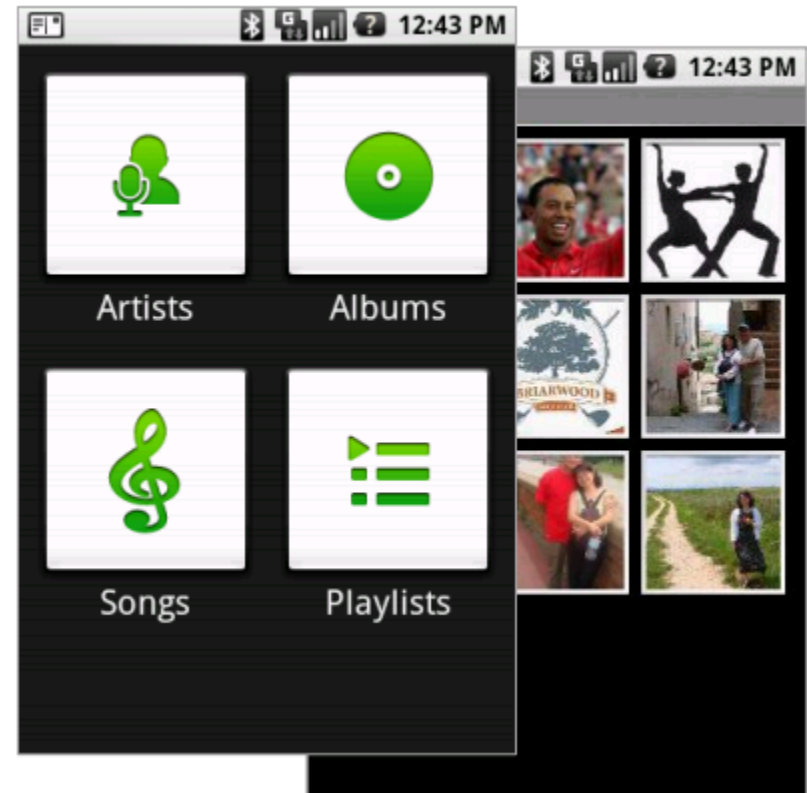
Media API

- Android cung cấp tập hợp hỗ trợ media khá phong phú gồm cả âm thanh, hình ảnh và video
- Chia làm 2 nhóm recorder và playback
- Các lớp thư viện này đều dễ dàng sử dụng trong phát triển ứng dụng và hoàn toàn miễn phí (rất quan trọng)



Media API

- Các tệp tin media có thể được sử dụng – thao tác bằng những cách sau:
 - Các tệp tin media được lưu trữ bên trong ứng dụng (các file resources)
 - Các file media độc lập có trong bộ nhớ trong của máy hoặc trong thẻ nhớ SDCARD
 - Các tài nguyên trên mạng



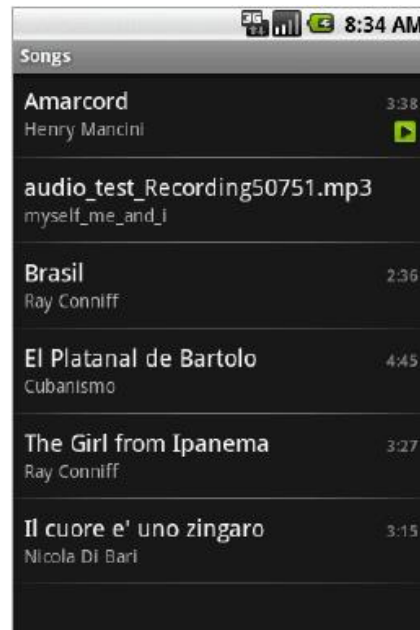


Phần 2.2

MediaStore

MediaStore

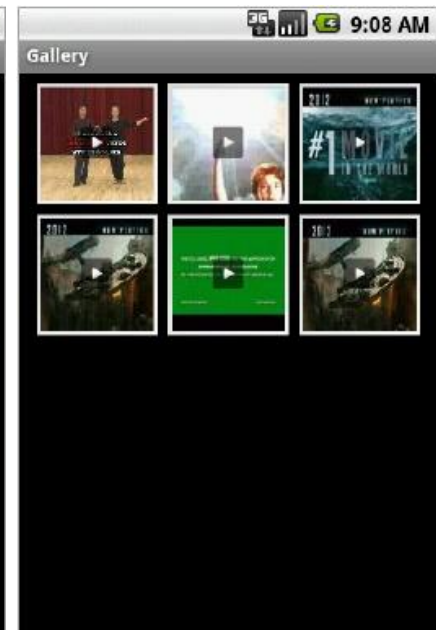
- Android OS có provider chuẩn chứa thông tin về các file media trong thiết bị
- Các ứng dụng tạo media (chụp ảnh, quay video) nên chủ động thêm các file media vào provider này



MediaStore.Audio



MediaStore.Images



MediaStore.Video



MediaStore

Có thể xem cấu trúc của MediaStore bằng cách xem file CSDL sau
[/data/data/com.android.providers.media/databases/internal.db](#)

```
// Trường hợp muốn mở các file video thì dùng Uri:
```

```
// android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI
```

```
Uri p = android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
```

```
Intent myIntent = new Intent(Intent.ACTION_VIEW, p);
```

```
startActivity(myIntent);
```

```
// Trường hợp muốn mở để chọn media thì sử dụng ACTION_PICK
```

```
Uri x = android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
```

```
Intent myIntent = new Intent(Intent.ACTION_PICK, x);
```

```
startActivityForResult(myIntent);
```




Phần 2.3

Audio



MediaPlayer & MediaRecorder

- Lớp MediaPlayer hỗ trợ việc playback các file audio và video
- Lớp MediaRecorder hỗ trợ việc ghi âm (ghi hình) và chuyển thành các file audio (video)
 - Chú ý: việc record phụ thuộc và việc phần cứng được hỗ trợ hay không
- Android OS chưa có cơ chế cài đặt thêm các codec mới cho audio và video, trường hợp ứng dụng muốn chạy format mới thì cần tự làm việc với i/o stream, surface view và audio stream

MediaPlayer

- Được sử dụng cho việc playback audio/video file và stream
- MediaPlayer hiểu cả các giao thức video internet
- Có thể xem các protocol được hỗ trợ bởi MediaPlayer <http://developer.android.com/guide/appendix/media-formats.html>





MediaPlayer – Useful Methods

Public Methods

static MediaPlayer	create (Context context,Uri uri) Convenience method to create a MediaPlayer for a given Uri.
static MediaPlayer	create (Context context,int resid) Convenience method to create a Media Player for a given resource id.
boolean	isPlaying () Checks whether the Media Player is playing.
void	pause () Pauses play back.
void	prepare () Prepares the player for play back ,synchronously.



MediaPlayer – Useful Methods

Public Methods	
void	setLooping (boolean looping) Sets the player to be looping or non-looping.
void	setVolume (float leftVolume,float rightVolume) Sets the volume on this player.
void	start () Starts or resumes play back.
void	stop () Stops play back after playback has been stopped or paused.

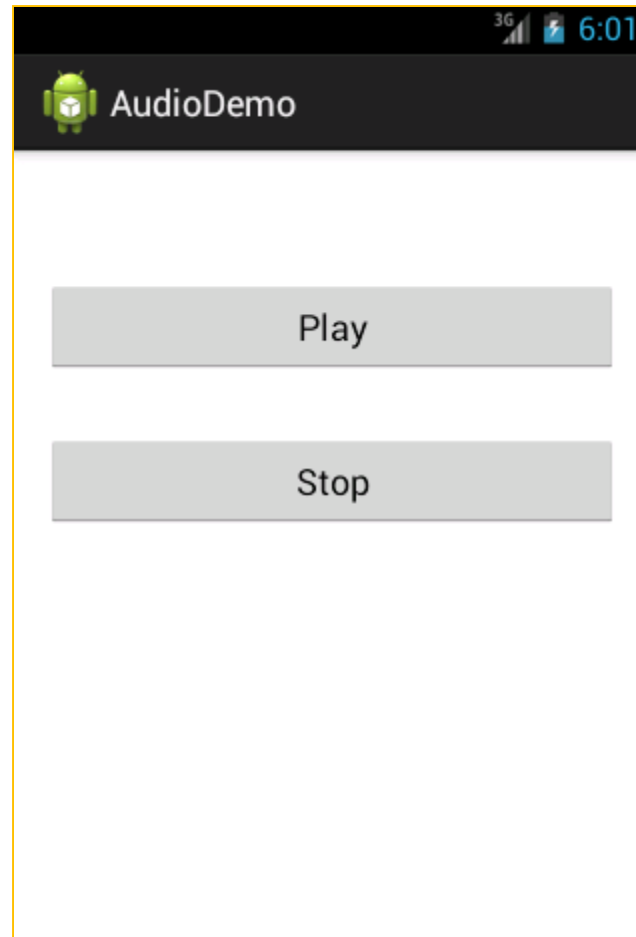
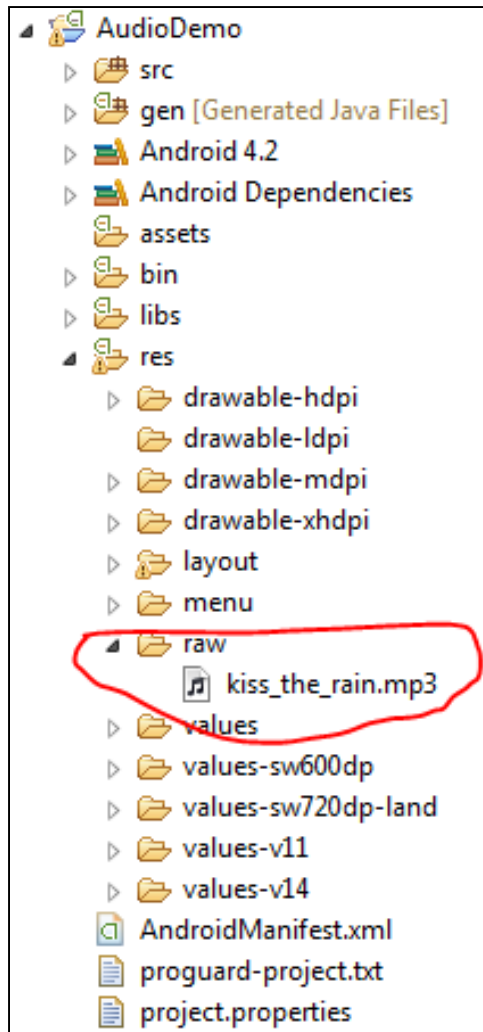


Chơi audio từ resource

- Tạo một file nhạc trong thư mục res/raw của dự án
- Khởi tạo đối tượng MediaPlayer thông qua hàm static create, prepare để khởi tạo các thông số cần thiết và bắt đầu chơi nhạc bằng phương thức start
- Gọi phương thức stop() để dừng lại
- Tạm dừng sử dụng phương thức pause()
- Chú ý: trong thử nghiệm thì chạy file từ resource không cần gọi prepare

```
MediaPlayer mp = MediaPlayer.create(context,  
R.raw.sound_1);  
mp.prepare();  
mp.start();
```

MediaPlayer – example





MediaPlayer – example

```
public class MyPlayer extends Activity {
    MediaPlayer mp = null;
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.main);
    }
    // xử lý button STOP
    public void btnStop(View v) {
        if (null == mp) return;
        if (mp.isPlaying()) {
            mp.stop();
            mp = null;
        }
    }
}
```




MediaPlayer – example

```
// xử lý button PLAY
public void btnPlay(View v) {
    try {
        mp = MediaPlayer.create(MyPlayer.this, R.raw.kiss_the_rain);
        mp.start();
        mp.setOnCompletionListener(new OnCompletionListener() {
            public void onCompletion(MediaPlayer arg0) {
                // xử lý khi đã chơi xong
            }
        });
    } catch (Exception e) { }
}
}
```



Chơi audio từ File/Stream

- Khởi tạo MediaPlayer thông qua hàm khởi tạo
- Gọi phương thức `setDataSource(string url)` với url là địa chỉ file hay đường dẫn trên internet
- Gọi phương thức `prepare()` để khởi tạo codec phù hợp
- Gọi phương thức `start()` để bắt đầu chơi
- Khi muốn tạm dừng hay dừng hẳn gọi các phương thức `pause()` hay `stop()`

```
String PATH_TO_FILE = "/sdcard/my_favorite_song.mp3";  
MediaPlayer mp = new MediaPlayer();  
mp.setDataSource(PATH_TO_FILE);  
mp.prepare();  
mp.start();
```



MediaPlayer – example

```
final Button b = (Button) findViewById(R.id.Play);
b.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        player = new MediaPlayer();
        try {
            player.setDataSource(Environment.getExternalStorageDirectory()
                .getPath()+ "/Music/test.mp3");
            player.prepareAsync();
            player.setOnPreparedListener( new MediaPlayer.OnPreparedListener(){

                @Override
                public void onPrepared(MediaPlayer mp) {
                    // TODO Auto-generated method stub
                    mp.start();
                }
            });
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
});
```



Một số chú ý khi playback

- MediaPlayer cần được **reset** hoặc **release** rồi mới được chơi lại nếu trước đó bạn **stop**
- MediaPlayer chạy ngầm, vì thế nếu bạn đóng activity (**finish**) thì audio vẫn chạy ngầm (và không có cách nào dừng nó), vì thế nên dùng **System.exit** để kết thúc ứng dụng
- Có thể chơi cùng lúc nhiều MediaPlayer và có thể thiết lập các mức volume khác nhau cũng như các nguồn ra khác nhau cho từng MediaPlayer (ví dụ chơi 2 file audio và mỗi file đổ âm thanh ra một phía của tai nghe)



Một số chú ý khi playback

- Muốn ứng dụng chạy ngầm và sau khi ứng dụng quay trở lại vẫn tiếp tục điều khiển được Audio cũ, ta nên sử dụng service
- Muốn tương tác với phím điều khiển âm lượng:
 - Đăng kí broadcast receiver:
`android.intent.action.MEDIA_BUTTON`
 - Điều chỉnh âm thanh bằng phương thức `setVolume(left, right)`, trong đó giá trị left/right là số thực nằm trong khoảng từ 0.0f đến 1.0f
- Sử dụng `AudioManager` trong trường hợp muốn tương tác nhiều hơn với phần cứng audio

Audio Recording



- Để ghi âm, sử dụng MediaRecorder
 1. Khởi tạo đối tượng recorder thông qua hàm khởi tạo
 2. Khởi tạo đối tượng `android.content.ContentValues`, truyền các giá trị `TITLE`, `TIMESTAMP` và `MIME_TYPE` để lưu trữ
 3. Tạo đường dẫn đến file lưu trữ
 4. Thiết lập audio source với `MediaRecorder.setAudioSource()` hoặc `MediaRecorder.setAudioSource.MIC`

Audio Recording



5. Cấu hình kiểu format
`MediaRecorder.setOutputFormat()`
6. Kiểu mã hóa
`MediaRecorder.setAudioEncoder()`
7. Gọi phương thức `prepare()` để chuẩn bị
8. Bắt đầu ghi âm với phương thức `start()` và dừng với `stop()`
9. Khi kết thúc gọi `release()` để giải phóng bộ nhớ



Audio Recording

```
// SET UP AND RECORD AN AUDIO FILE
recorder = new MediaRecorder();
// BASIC DATA NEEDED TO ADD RECORDING TO THE AUDIO MEDIASTORE PROVIDER
ContentValues values = new ContentValues(5);
    values.put(MediaStore.MediaColumns.TITLE, SOME_NAME_HERE);
    values.put(MediaStore.MediaColumns.TIMESTAMP, System.currentTimeMillis());
    values.put(MediaStore.MediaColumns.MIME_TYPE, recorder.getMimeContentType());
    values.put(AudioColumns.IS_MUSIC, true);
    values.put(AudioColumns.ARTIST, "myself");

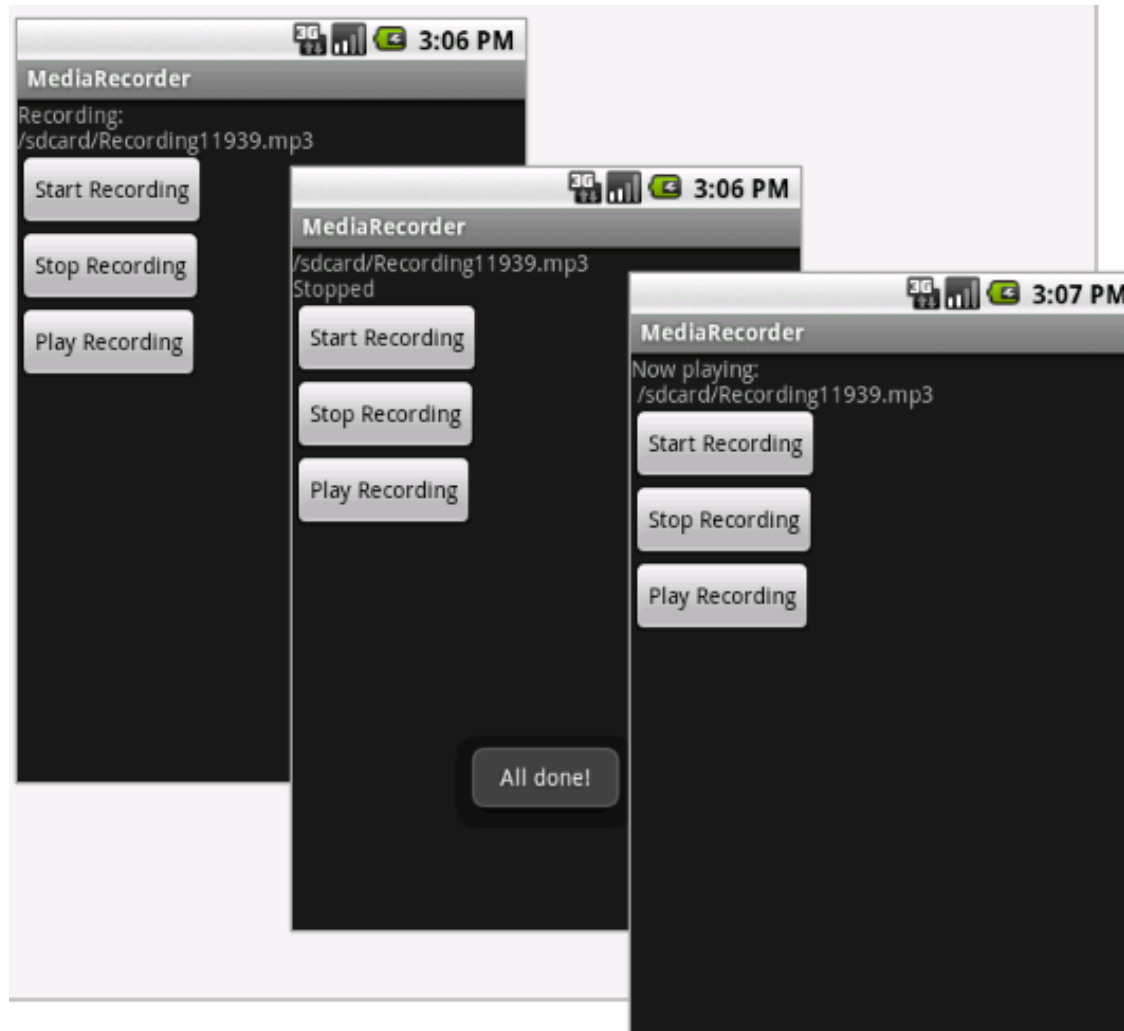
ContentResolver contentResolver = new ContentResolver();
Uri base = MediaStore.Audio.INTERNAL_CONTENT_URI;
Uri newUri = contentResolver.insert(base, values);

// USE MICROPHONE, 3GP FORMAT, AMR CODEC (SPEECH RECORDING)
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
recorder.setOutputFile(path);

// GET READY, GO ...
recorder.prepare();
recorder.start();
```




Audio Recording – example





Audio Recording – example

```
public class MyAudioRecorder extends Activity {
    MediaRecorder myRecorder;
    File mSampleFile = null;
    TextView txtMsg;
    static final String SAMPLE_PREFIX = "Recording";
    static final String SAMPLE_EXTENSION = ".mp3";
    private static final String TAG = "<<MyAudioRecorder>>";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView)findViewById(R.id.txtMsg);
        myRecorder = new MediaRecorder();
    }
}
```



Audio Recording – example

```
Button start = (Button) findViewById(R.id.startRecording);
start.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        startRecording();
    }
});
```

```
Button stop = (Button) findViewById(R.id.stopRecording);
stop.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        stopRecording();
        addToMediaStoreDB();
    }
});
```



Audio Recording – example

```
Button play = (Button) findViewById(R.id.playRecording);
play.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        try {
            String name = mSampleFile.getAbsolutePath();
            txtMsg.setText("Now playing:\n " + name);
            MediaPlayer mp = new MediaPlayer();
            mp.setDataSource( recordingName );
            mp.prepare();
            mp.start();
        } catch (Exception e) {}
    }
} // onCreate
```



Audio Recording – example

```
protected void startRecording() {
    try {
        if (this.mSampleFile == null) {
            File dir = Environment.getExternalStorageDirectory();
            try {
                this.mSampleFile = File.createTempFile(
                    MyAudioRecorder.SAMPLE_PREFIX,
                    MyAudioRecorder.SAMPLE_EXTENSION, dir);
            } catch (IOException e) {
                return;
            }
        }
        txtMsg.setText("Recording: \n" +
            mSampleFile.getCanonicalPath());
    }
}
```



Audio Recording – example

```
myRecorder = new MediaRecorder();
myRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
myRecorder.setOutputFormat(MediaRecorder.OutputFormat.
                                THREE_GPP);
myRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.
                                AMR_NB);
myRecorder.setOutputFile(this.mSampleFile.
                                getAbsolutePath());

myRecorder.prepare();
myRecorder.start();
} catch (Exception e) { }
} // startRecording
```



Audio Recording – example

```
protected void stopRecording() {
    try {
        myRecorder.stop();
        myRecorder.release();
    } catch (IllegalStateException e) {}
}

protected void addToMediaStoreDB() {
    try {
        int now = (int) (System.currentTimeMillis() / 1000);
        ContentValues newValues = new ContentValues(6);
        newValues.put(MediaColumns.TITLE, mSampleFile.getName());
        newValues.put(MediaColumns.DATE_ADDED, now);
    }
}
```



Audio Recording – example

```
newValues.put(MediaColumns.MIME_TYPE, "audio/mpeg");
newValues.put(AudioColumns.IS_MUSIC, true);
newValues.put(AudioColumns.ARTIST, "myself");
newValues.put(MediaColumns.DATA,
                mSampleFile.getAbsolutePath());
ContentResolver contentResolver = getContentResolver();
Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
Uri nUri = contentResolver.insert(base, newValues);
sendBroadcast(new
    Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, nUri));
} catch (Exception e) { }
} // addToMediaStoreDB
```