TÀI LIỆU THAM KHẢO
PLAY AUDIO VÀ VIDEO.

Nội dung:

1. Tổng quan
2. Lập trình play audio
3. Lập trình play video.
4. Điều khiển MediaPlayer bằng MediaController
5. Play Audio thông qua Service

Thực hành:

1. Xây dựng ứng dụng play audio đơn giản
2. Xây dựng ứng dụng play video đơn giản

A. LÝ THUYẾT

1. Tổng quan

Android Framework cung cấp các API để play một số loại tập tin media (âm thanh, video, hình ảnh) phổ biến, vì vậy chúng ta có thể dễ dàng tích hợp các tính năng play audio, video vào ứng dụng một cách dễ dàng.

Chúng ta có thể play âm thanh hoặc hình ảnh từ các tập tin được lưu trong resource của ứng dụng, tập tin từ thẻ nhớ, hoặc tập tin lấy trực tiếp từ internet sử dụng API của lớp ***MediaPlayer*** được cung cấp bởi Android Framework.

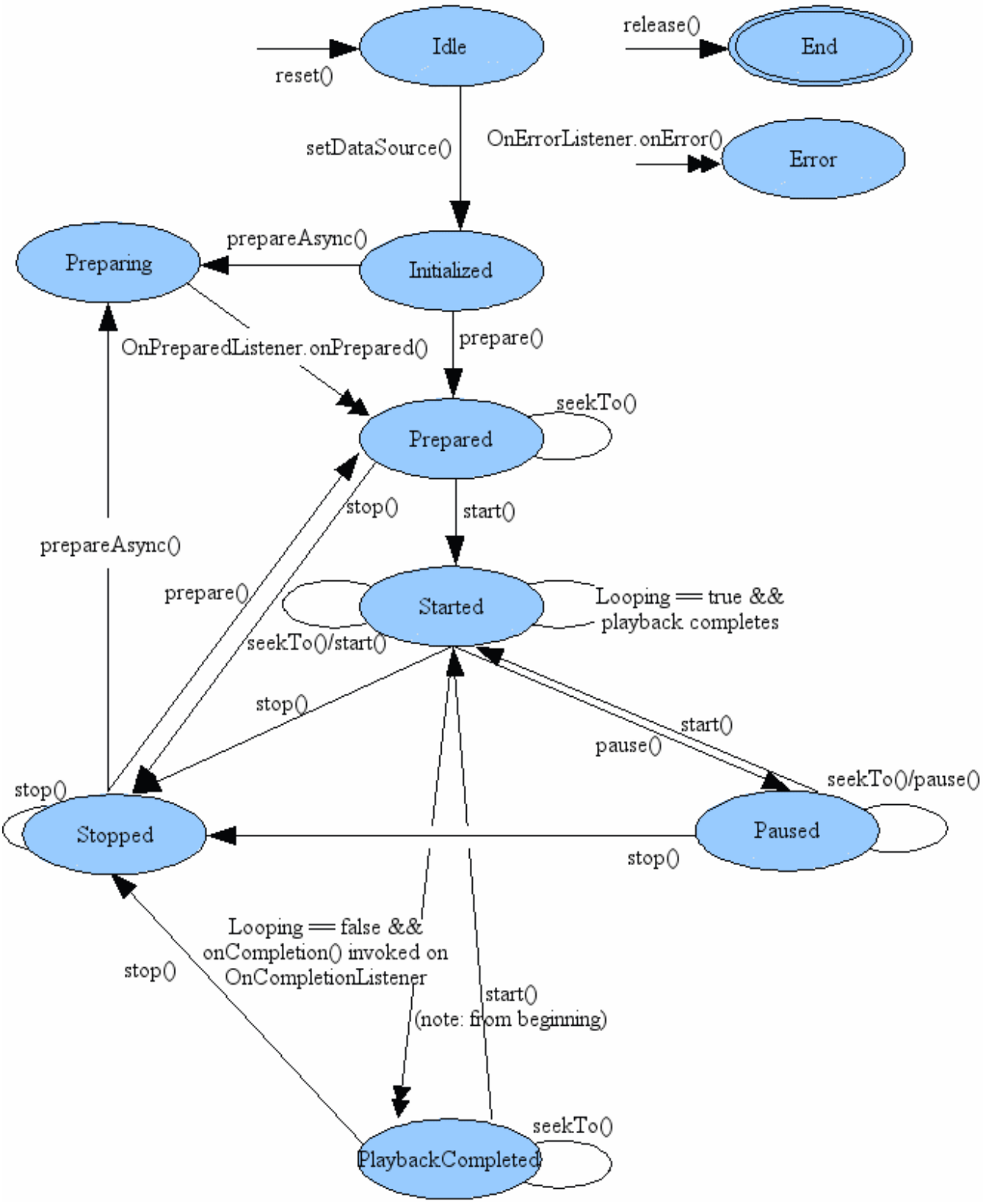
Android 4.3 (API 18) mặc định hỗ trợ các định dạng media phổ biến như sau[1]:

- Audio:
  - AAC LC/LTP
  - HE-AACv1 (AAC+)
  - HE-AACv2 (Enhanced AAC+)
  - AMR-NB
  - AMR-WB
  - MP3
  - MIDI
  - Ogg Vorbis
  - PCM/WAVE
- Video
  - H.263
  - H.264 AVC
  - MPEG-4 SP

[1] *http://developer.android.com/guide/appendix/media-formats.html*

    - VP8
- Streaming:
    - RTSP (RTP, SDP)
    - HTTP/HTTPS progressive streaming
    - HTTP/HTTPS live streaming (trên thiết bị chạy Android 3.0 hoặc lớn hơn)

Quá trình play audio/video được MediaPlayer quản lý thông qua một sơ đồ các trạng thái như dưới đây:



Hình 1: sơ đồ các trạng thái

Có thể tóm tắt các trạng thái trên sơ đồ trên như sau:

1) Initialize: trạng thái khởi tạo play media
2) Prepare: chuẩn bị play, có hai loại prepare:
   a. *Prepare*: các thao tác chuẩn bị được thực hiện trên Thread chính và người dùng không thể tương tác với ứng dụng trong giai đoạn này.
   b. *Prepare async*: các thao tác chuẩn bị được thực hiện trên một Thread khác và người dùng vẫn có thể tao tác với ứng dụng.
3) Bắt đầu play
4) Pause hoặc Stop
5) Kết thúc play

Ứng dụng được tạo ra có thể play các audio hay video từ các tập tin media lưu ở các nơi sau đây:

- Lưu trong tài nguyên của chính ứng dụng (trong thư mục raw được tạo ra trong thư mục tài nguyên của ứng dụng: res/raw).
- Lưu trong bộ nhớ trong hoặc bộ nhớ ngoài của thiết bị.
- Các tập tin trong luồng dữ liệu được tải về từ internet.

Để sử dụng các API của MediaPlayer chúng ta cần khai báo một số quyền cần thiết sau vào tập tin manifest của ứng dụng:

- **Internet Permission**: Nếu ứng dụng chúng ta sử dụng MediaPlayer để play một tập tin trực tiếp từ Internet thì chúng ta cần khai báo quyền truy cập mạng như sau:

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

- **Read external storage Permission**: Nếu ứng dụng chúng ta sử dụng MediaPlayer để play một tập tin được lưu trên bộ nhớ ngoài thì chúng ta cần khai báo quyền truy cập mạng như sau:

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

- **Wake Lock Permission**: Nếu ứng dụng cần ngăn không cho thiết bị chuyển sang chế độ dimming, sleeping, hoặc yêu cầu màn hình luôn bật, hoặc sử dụng các phương thức `MediaPlayer.setScreenOnWhilePlaying()` hoặc `MediaPlayer.setWakeMode()` thì chúng ta cần khai báo quyền sau:

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

Các lớp sau đây được dùng để play sound và video:

- Lớp **MediaPlayer**: đây là API chính để play sound và video trong Android.
- Lớp **AudioManager**: quản lý tài nguyên audio trên thiết bị.

2. Lập trình play Audio

Như trên đã nói, có một số cách để play audio thông qua MediaPlayer. Chúng ta có thể nhúng các tập tin audio vào resource của ứng dụng và play tập tin đó, chúng ta có thể play tập tin từ thẻ nhớ hoặc từ URL trên Internet.

Để nhúng tập tin vào resource của ứng dụng, chúng ta cần đưa tập tin vào thư mục *res/raw*. Những tập tin nằm trong thư mục *res/raw* sẽ được giữ nguyên khi biên dịch và đóng gói ứng dụng. Bằng cách này, nội dung của tập tin audio sẽ được giữ nguyên định dạng khi thực thi ứng dụng.

Để play audio, chúng ta cần làm theo các bước sau:

1. Khởi tạo một đối tượng MediaPlayer với tập tin audio cần play
2. Gọi hàm prepare() (hoặc prepareAsync) để chuẩn bị play. Không gọi hàm này nếu tập tin audio được nhúng trong resource của ứng dụng.
3. Gọi hàm start() để bắt đầu play
4. Gọi hàm pause() / stop() nếu có yêu cầu từ người dùng
5. Gọi hàm release() để giải phóng đối tượng MediaPlayer khi kết thúc.

Ví dụ 1:

1. Play tập tin audio trong *res/raw/bonghongthuytinh.mp3* (giả sử đã chép tập tin *bonghongthuytinh.mp3* vào thư mục raw).

```
package ctu.edu.playaudio;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;

public class MainActivity extends Activity {

   private MediaPlayer mMediaPlayer;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      //Tạo đối tượng MediaPlayer
      mMediaPlayer = MediaPlayer.create(this, R.raw.bonghongthuytinh);

      //Bắt đầu play.
      //Không cần gọi mMediaPlayer.prepare() vì tập tin audio
      //được nhúng trong res/raw
      mMediaPlayer.start();
   }

   @Override
   protected void onDestroy() {
      //Giải phóng đối tượng MediaPlayer.
      mMediaPlayer.release();
      super.onDestroy();
   }
}
```

2. Play một audio từ một có sẵn trong thiết bị (bộ nhớ trong/bộ nhớ ngoài) với URI (có thể thu được thông qua đối tượng Content Resolver, xem lại trong bải Content Provider):

```
Uri myUri = ....; // khởi động URI
MediaPlayer mediaPlayer = new MediaPlayer();
```

```
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. Play một audio từ một URL thông qua kết nối HTTP:

```
String url = "http://........"; // khai bào URL
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // có thể mất nhiều thờ gian! (cho bộ đệm,…)
mediaPlayer.start();
```

Chú ý: If you're passing a URL to stream an online media file, the file must be capable of progressive download.

Caution: You must either catch or pass `IllegalArgumentException` and `IOException` when using `setDataSource()`, because the file you are referencing might not exist.

3. Lập trình play Video

Play video đôi chút phức tạp hơn so với play audio. Để play video chúng ta cần tạo một Surface để hiển thị video.

Tuy nhiên, ngoài cách phải tự tạo Surface để hiển thị video, Android Framework cũng cung cấp sẵn lớp VideoView. Lớp này đóng gói tất các các chức năng cần thiết để hiển thị video như tạo Surface, khởi tạo đối tượng MediaPlayer để play video.

3.1. Play video bằng VideoView

Cách đơn giản nhất để play video là sử dụng lớp VideoView. Sau khi đặt VideoView vào giao diện ứng dụng và tham chiếu đến video view trong code. Chúng ta có thể gán tập tin video vào video view để play bằng cách gọi các hàm *setVideoPath()* hoặc *setVideoURI()*. Tương tự như audio, tập tin video có thể lấy từ thẻ nhớ hoặc từ URI chỉ đến nội dung video. Ngoài ra chúng ta có thể gán video từ URL trên internet thông qua hàm *setVideoURI()*.

```
//Tham chiếu đến video view từ layout
final VideoView videoView = (VideoView)findViewById(R.id.videoView);

// Chỉ định video để play từ thẻ nhớ
videoView.setVideoPath("/sdcard/mycatvideo.3gp");

// Chỉ định video để play từ internet
```

```
videoView.setVideoURI(Uri.parse("http://www.video.com/nice-video.mp4"));
```

3.2. Play video bằng cách tạo Surface

Để play video không thông qua VideoView chúng ta cần sử dụng hai lớp là MediaPlayer và SurfaceView. SurfaceView chứa một đối tượng Surface Holder dùng để đảm nhận cập nhật các hình ảnh của video một cách liên tục thông qua một tiến trình chạy nền.

MediaPlayer sử dụng hàm *setDisplay()* để chỉ định SurfaceHolder sẽ hiển thị video. Các bước thực hiện như sau:

1. Tạo hoặc tham chiếu SurfaceView từ layout
2. Lấy đối tượng SurfaceHolder để chuẩn bị hiển thị video
3. Tạo đối tượng MediaPlayer
4. Chỉ định SurfaceView cho đối tượng MediaPlayer
5. Chỉ định đường dẫn của tập tin video cho đối tượng MediaPlayer
6. Gọi hàm prepare (hoặc prepareAsync) của đối tượng MediaPlayer
7. Gọi hàm start của đối tượng MediaPlayer để bắt đầu play video.
8. Giải phóng đối tượng MediaPlayer khi play xong

```
package ctu.edu.playvideo;

import android.app.Activity;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

import java.io.IOException;

public class MainActivity extends Activity implements
SurfaceHolder.Callback{
    private SurfaceView mSurfaceView;
    private MediaPlayer mMediaPlayer;
    private String demoVideoUrl = "http://techslides.com/demos/sample-
videos/small.mp4";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
    //Tham chiếu SurfaceView từ layout
    mSurfaceView = (SurfaceView)findViewById(R.id.surfaceView);

    //Lấy đối tượng SurfaceHolder
    SurfaceHolder holder = mSurfaceView.getHolder();

    //Gán listener để bắt sự kiện surface đã được tạo xong
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    //Tạo đối tượng MediaPlayer
    mMediaPlayer = new MediaPlayer();
}

/**
 * Hàm này được gọi sau khi surface được tạo xong
 * @param holder
 */
@Override
public void surfaceCreated(SurfaceHolder holder) {
    //Gán holder để hiển thị video cho đối tượng media player
    mMediaPlayer.setDisplay(holder);
    try {
        //Thiết lập đường dẫn đến tập tin video
        mMediaPlayer.setDataSource(this, Uri.parse(demoVideoUrl));
        //Gọi hàm prepare (Nên gọi hàm prepareAsync)
        mMediaPlayer.prepare();
        //Bắt đầu play video
        mMediaPlayer.start();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    //Giải phóng đối tượng media player
    mMediaPlayer.release();
}

@Override
protected void onDestroy() {
    //Giải phóng đối tượng media player
```

```
        mMediaPlayer.release();
        super.onDestroy();
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder,
                               int format, int width, int height) {

    }
}
```

4. Điều khiển MediaPlayer bằng MediaController

Lớp MediaPlayer cung cấp các hàm để điều khiển quá trình play audio/video như *start(), stop(), pause(),...* Tuy nhiên để người dùng có thể tương tác được với các hàm này chúng ta phải tạo bộ giao diện với các nút nhấn. Khi người dùng nhấn vào các nút nhấn thì chúng ta sẽ gọi các hàm tương ứng.

Để đảm bảo tính thống nhất trong việc điều khiển quá trình play audio/video, Android cung cấp sẵn một bộ giao diện chuẩn cho việc này thông qua lớp MediaController như hình 19.1.

Hình 19.1

MediaController chỉ cung cấp giao diện, chúng ta phải thực hiện gọi các hàm khi người dùng nhấn vào các nút tương ứng thông qua các hàm callback của lớp MediaPlayerControl như dưới đây.

```
MediaController mediaController = new MediaController(this);
mediaController.setMediaPlayer(new MediaController.MediaPlayerControl() {
    @Override
    public void start() {
        //callback khi người dùng nhấn nút start
    }

    @Override
    public void pause() {
        //callback khi người dùng nhấn nút pause
    }

    @Override
```

```
    public int getDuration() {
        return 0;
    }

    @Override
    public int getCurrentPosition() {
        return 0;
    }

    @Override
    public void seekTo(int pos) {
        //callback cuộn thanh tiến trình
    }

    @Override
    public boolean isPlaying() {
        return false;
    }

    @Override
    public int getBufferPercentage() {
        return 0;
    }

    @Override
    public boolean canPause() {
        return false;
    }

    @Override
    public boolean canSeekBackward() {
        return false;
    }

    @Override
    public boolean canSeekForward() {
        return false;
    }

    @Override
    public int getAudioSessionId() {
        return 0;
    }
});
```

Nếu chúng ta sử dụng VideoView để play video thì có thể thiết lập MediaController một cách đơn giản như sau:

```
mediaController.setAnchorView(myVideoView);

mediaController.show();
```

5. Play audio thông qua Service

Như chúng ta đã tìm hiểu về vòng đời của một Activity trong ứng dụng. Khi người dùng đang play audio và chuyển sang ứng dụng khác thì Activity sẽ chuyển sang trạng thái tạm ngưng hoạt động. Ở trạng thái này các đối tượng liên quan đến Activity sẽ được giải phóng và chỉ được tạo lại khi người dùng quay trở lại ứng dụng. Điều này dẫn đến việc play audio trên Activity cũng sẽ bị ngưng vì đối tượng MediaPlayer đã bị giải phóng.

Thông thường, những ứng dụng nghe nhạc đều có thể tiếp tục play audio ngay cả khi người dùng chuyển sang ứng dụng khác. Để làm được điều này, chúng ta phải play audio thông qua Service. Service sẽ tiếp tục chạy ngay cả khi người dùng không tương tác với ứng dụng của chúng ta, vì thế quá trình play audio sẽ không bị gián đoạn.

Việc play audio thông qua Service cũng phát sinh thêm vấn đề xử lý giao tiếp giữa Activity và Service. Bởi vì service play audio chạy ngầm trên một tuyến khác (hoặc môt process khác), nên khi người dùng cần tương tác với quá trình play audio (ví dụ: tạm ngưng, hoặc ngưng play audio) trên Activity thì chúng ta phải kết nối đến Service và gọi các hàm tương ứng với thao tác của người dùng.

Tham khảo thêm về cách tạo Service tại
http://developer.android.com/guide/components/services.html

Các bước để play audio thông qua service như sau:

1) Tạo Service, tất cả các tác vụ play audio sẽ được viết trong service này.
2) Trong Activity, khởi tạo ServiceConnection để quản lý kết nối giữa Activity và Service
3) Trong Activity, gọi hàm *startService() và bindService()* để khởi động service và tạo kết nối.
4) Gọi các hàm xử lý tương ứng với thao tác của người dùng (start, stop, pause,...)
5) Trong Activity, kết thúc service nếu người dùng thoát khỏi ứng dụng bằng hàm *stopService()*. Chú ý: Thoát khỏi ứng dụng khác với chuyển sang ứng dụng khác.
6) Hoặc, trong service, kết thúc service khi đã play hết tập tin audio bằng hàm *stopSelf()*

Ví dụ: Xây dựng service để play audio và cho phép người dùng tương tác với service qua Activity như sau:

(1) Tạo lớp *MediaPlayerService* như sau:

```
package ctu.edu.playaudio;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Binder;
import android.os.IBinder;

public class MediaPlayerService extends Service
                        implements MediaPlayer.OnCompletionListener {
   private MediaPlayer mMediaPlayer;

   //Binder dùng để tạo kết nối giao tiếp giữa Activity và ứng dụng
   private MediaPlayerBinder mBinder = new MediaPlayerBinder();

   @Override
   public IBinder onBind(Intent intent) {
      return mBinder;
   }

   @Override
   public int onStartCommand(Intent intent, int flags, int startId) {
      return Service.START_STICKY;
   }

   @Override
   public void onDestroy() {
      if (mMediaPlayer != null) {
         mMediaPlayer.release();
      }
      super.onDestroy();
   }

   public void start()
   {
      if (mMediaPlayer == null) {
         mMediaPlayer =
           MediaPlayer.create(getApplicationContext(), R.raw.bonghongthuytinh);
         mMediaPlayer.setOnCompletionListener(this);
```

```
        }
        mMediaPlayer.start();
    }

    public void stop()
    {
        if (mMediaPlayer != null && mMediaPlayer.isPlaying()) {
            mMediaPlayer.stop();
        }
    }

    /**
     * Hàm này sẽ được gọi khi media player play hết tập tin audio
     * @param mp
     */
    @Override
    public void onCompletion(MediaPlayer mp) {
        stopSelf();
    }

    //Binder class dùng để giao tiếp giữa activity và ứng dụng
    public class MediaPlayerBinder extends Binder
    {
        public MediaPlayerService getService() {
            return MediaPlayerService.this;
        }
    }
}
```

(2) Tạo Activity như sau:

```
package ctu.edu.playaudio;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.widget.Button;
```

```
import ctu.edu.playaudio.MediaPlayerService.MediaPlayerBinder;

public class MainActivity extends Activity implements View.OnClickListener {
   private boolean mConnected;
   private MediaPlayerService mService;

   //Khởi tạo ServiceConnection để quản lý kết nối
   private ServiceConnection mConnection = new ServiceConnection() {
      @Override
      public void onServiceConnected(ComponentName name, IBinder service) {
         MediaPlayerBinder binder = (MediaPlayerBinder)service;
         mService = binder.getService();
         mConnected = true;
      }

      @Override
      public void onServiceDisconnected(ComponentName name) {
         mConnected = false;
      }
   };

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      //Tham chiếu đến các nút nhấn để nhận thao tác từ
      //người dùng
      Button btnStart = (Button)findViewById(R.id.btnStart);
      btnStart.setOnClickListener(this);

      Button btnStop = (Button)findViewById(R.id.btnStop);
      btnStop.setOnClickListener(this);
   }

   protected void onStart() {
      super.onStart();

      //Khởi động service
      Intent intent = new Intent(this, MediaPlayerService.class);
      startService(intent);

      //Bind service
      bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
```

```
    }

    protected void onStop()
    {
        super.onStop();
        if (mConnected) {
            //Ngắt kết nối đến service khi Activity tạm ngưng
            unbindService(mConnection);
            mConnected = false;
        }
    }

    @Override
    protected void onDestroy() {
        //Huỷ service khi Activity bị huỷ hoặc ứng dụng bị ngưng bởi
        //người dùng
        Intent intent = new Intent(this, MediaPlayerService.class);
        stopService(intent);
        super.onDestroy();
    }

    /**
     * Hàm xử lý các thao tác của người dùng
     * @param v
     */
    @Override
    public void onClick(View v) {
        if (mConnected == false) {
            return;
        }
        if (v.getId() == R.id.btnStart) {
            mService.start();
        } else if (v.getId() == R.id.btnStop) {
            mService.stop();
        }
    }
}
```

(3) Khai báo Service trong tập tin Androidmanifest.xml như sau:

```
<?xml version="1.0" encoding="utf-8"?>
 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
  package="ctu.edu.playaudio" >

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"

      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <service android:name=".MediaPlayerService" />
  </application>

</manifest>
```

B. THỰC HÀNH

1. Xây dựng ứng dụng phát tập tin MP3 từ tập tin resource. Ứng dụng cho phép người dùng stop, pause, restart tiến trình play audio. (Bài 7_2_1)

2. Xây ựng ứng dụng phát tập tin MP3 từ một URL trên internet.
   - Ứng dụng cho phép người dùng stop, pause, restart tiến trình play audio.
   - Sử dụng MediaController để điều khiển tiến tình play audio.
   - Hiển thị các thông tin cơ bản của bài hát (nếu có).

Xây dựng ứng dụng phát tập tin MP4 từ một URL trên internet sử dụng VideoView. Ứng dụng cho phép người dùng điều khiển tiến trình play video thông qua .

THAM KHẢO:

[1] https://developer.android.com/reference/android/media/MediaPlayer.html