About This Article

The Android media package provides classes for managing various media types. These classes are provided for performing Audio and Video operations. Apart from the basics, ringtone management, face detection and controlling audio routing are also provided. This article shows the Audio and Video operations.

Scope:

This article is intended for novice users who want brief understanding on Android programming. This article will guide you step by step on developing applications that uses media (audio and video). It assumes that you have already installed Android and the necessary tools to develop applications. It also assumes that you are familiar with Java or familiar with Object Oriented Programming concepts.

Introduction

Android supports audio, video operations through the "**android.media"** package. Apart from the basic, classes for ringtone management, face detection and controlling audio routing are also provided.
Android supports the playing of audio and video through the **MediaPlayer** class. The MediaPlayer class is at the heart of the Android media package. Apart from the MediaPlayer class, the **SoundPool** and **JetPlayer** classes are also provided to be used to play audio files.

Playing Audio Files

MediaPlayer

MediaPlayer is the most widely used class for playing media files. MediaPlayer has been designed to play large audio files and streams also supporting playback operations like stop, start, pause etc and seeking operations. It also supports listeners related to the media operations.

Playing of audio and video in MediaPlayer can be done through the following ways:

- •Playing from raw resource.
- •Playing from file-system.
- •Playing from stream.

MediaPlayer Listeners

MediaPlayer defines some listeners like **OnCompletionListener, OnPrepareListener, OnErrorListener, OnBufferingUpdateListener, OnInfoListener, OnVideoSizeChangedListener, OnSeekCompleteListener.**
OnCompletionListener **onCompletion(MediaPlayer mp)** event will be called when end of a media source is reached during playback.. You can use this listener event for playing the next song from the list or releasing the media player object.
OnPrepareListener **onPrepared(MediaPlayer mp)** event will be called when the media source is ready for playback. You can start playing the song in the onPrepared() method.
OnErrorListener **boolean onError(MediaPlayer mp, int what, int extra)** event will be called when there has been an error during an asynchronous operation (other errors will throw exceptions at the method call time). Parameter **what**specifies what type of error occurred. This can be MEDIA_ERROR_UNKNOWN or MEDIA_ERROR_SERVER_DIED. Parameter **extra** specifies the extra information related to the error.

Playing Audio from res

This is the most common way of playing audio files.

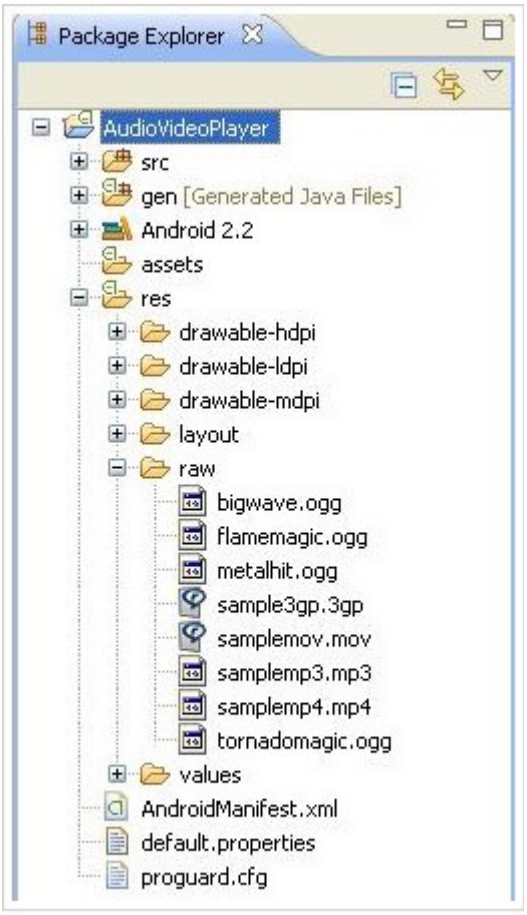

Figure 1: Raw folder media files

In this case the audio file should be present in the raw or assets folder of the project like the one shown in figure 1.

To access a raw resource simple use the lowercase filename without an extension:

```
context appContext  = getApplicationContext();
MediaPlayer mMediaPlayer = MediaPlayer.create(appContext,R.raw.samplemp3);
mMediaPlayer.start();
```

Or

```
MediaPlayer mMediaPlayer = MediaPlayer.create(this, R.raw.samplemp3);
mMediaPlayer.start();
```

To stop playback, call stop(). If you wish to replay the media, then you must reset() and prepare() the MediaPlayer object before calling start() again. (create() calls prepare() the first time.)

To pause playback, call pause(). Resume playback from where you paused with start().

Playing Audio from file-system

The second way to access audio files is from file-system i.e. SD cards. Most of the audio resources are present on SD cards.

Before looking into how to access audio file through SD card, lets have a look at how to load files in SD cards.

Open up the **FileExplorer** view in Eclipse IDE through **Windows > Show View > Other**.
It will open up Show View. Select **Android > FileExplorer** as shown in Figure 2.

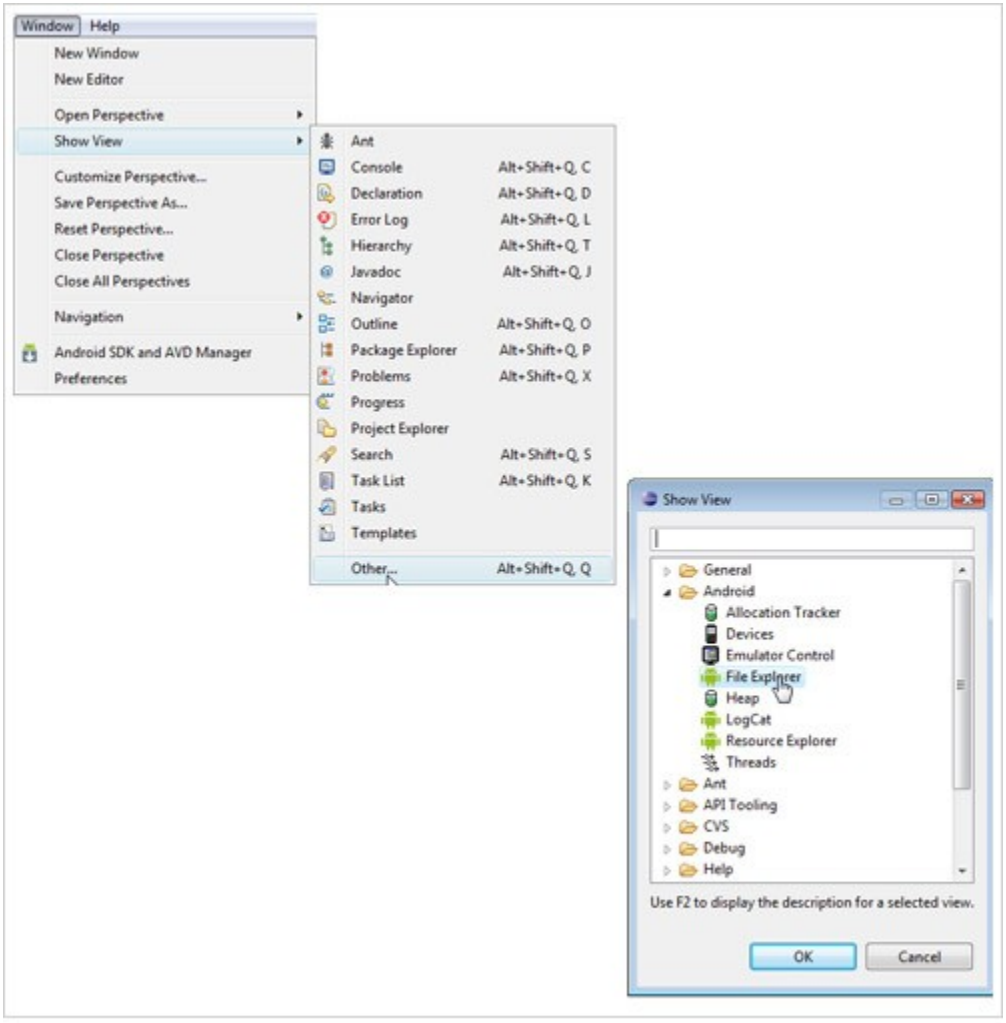

Figure 2: Opening File Explorer

On selecting File Explorer, it will open up the File Explorer view as shown in Figure 3.

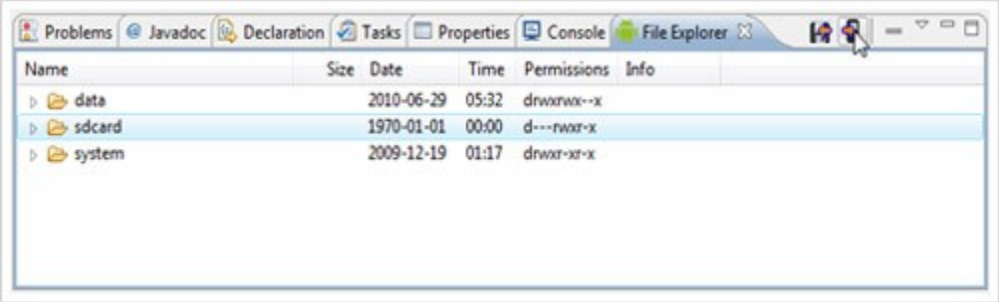

Figure 3: File Explorer View

Now to push a file onto the SD card, select the sdcard folder in the File Explorer and choose the button with the right-facing arrow which is at the top-right corner. This launches a dialog box that lets you select a file. Select the file that you want to upload to the SD card.

After pushing the files onto the SD card, the available contents are shown in figure 4.

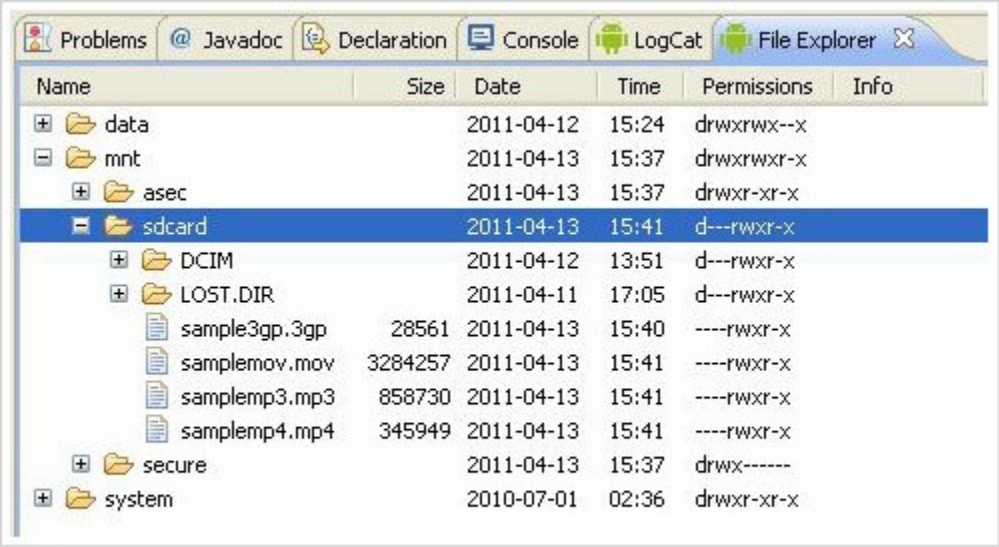

Figure 4: SD Card Contents

Accessing the file from the SD card is done through

```
String pathToFile = "/sdcard/samplemp3.mp3";
//create mediaplayer
mediaPlayer = new MediaPlayer();
//set audio file path
try {
  mediaPlayer.setDataSource(pathToFile);
} catch (IllegalArgumentException e) {
  e.printStackTrace();
} catch (IllegalStateException e) {
  e.printStackTrace();
} catch (IOException e) {
```

```
  e.printStackTrace();
}
//Prepare mediaplayer
try {
  mediaPlayer.prepare();
} catch (IllegalStateException e) {
  e.printStackTrace();
} catch (IOException e) {
 e.printStackTrace();
}
//start mediaPlayer
mediaPlayer.start();
```

First of all, create a new instance of MediaPlayer. Next, set the audio file (file path) to be played as the data source of the MediaPlayer instance. The MediaPlayer object has to be prepared before the player can start playing the song. The **prepare()** method is the blocking method and blocks until the media player is ready to play the song. One non blocking method **prepareAsync()** is also provided. The non blocking prepare method should be used in-case media player is used to play a song from a stream and needs to buffer data before song can be played.
Now use the following to playback control methods like **start(), stop()** etc. The media player object has to reset before it can be setup for some other song file. The media player object has to be released after its use. This is done using**release()** method. **release()** method releases the resources associated with this MediaPlayer object. It is considered good practice to call this method when you're done using the MediaPlayer.
We can also create Media Player through

```
String pathToFile = "/sdcard/samplemp3.mp3";
MediaPlayer filePlayer = MediaPlayer.create( appContext, Uri.parse(pathToFile) );
```

Here URI class is use to create Uri by parsing the given encoded URI string.

Playing Audio from Web

Accessing audio files present in web site can be done using the same code used for accessing audio files present in SD Card. The only change is the file path. The path here would be website URL which points to the audio resource file.

The most important part here is the data is fetched using internet, so permission for accessing the internet is mandatory.

Set the Internet permission in **AndroidManifest.xml** file

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

The code remains the same except for the URL path

```
String urlPath = "http:/www.xyz.com/…/samplemp3.mp3";
//create new mediaplayer
mediaPlayer = new MediaPlayer();
//set audio file path
try {
  mediaPlayer.setDataSource(urlPath);
```

```
} catch (IllegalArgumentException e) {
  e.printStackTrace();
} catch (IllegalStateException e) {
  e.printStackTrace();
} catch (IOException e) {
  e.printStackTrace();
}
//Prepare mediaplayer
try {
  mediaPlayer.prepare();
} catch (IllegalStateException e) {
  e.printStackTrace();
} catch (IOException e) {
 e.printStackTrace();
}
//Start mediaplayer
mediaPlayer.start();
```

Alternatively we can also create Media Player through

```
String urlPath = "http:/www.xyz.com/…/samplemp3.mp3";
MediaPlayer filePlayer = MediaPlayer.create( appContext, Uri.parse(urlPath) );
```

Here URI class is use to create Uri by parsing the given encoded URI string.

SoundPool

SoundPool class is the opposite of MediaPlayer class. It is designed for quick playback of a predefined set of relatively short sound samples. It is a collection of samples that can be loaded into memory from a resource inside the APK or from a file in the file system.

The most typical use case is the sound effect pool in a game where low latency is required. In addition to the basic operations of preloading and playing sounds, SoundPool class supports following features:

- •Setting the maximum number of sounds to play at the same time
- •Prioritizing the sounds so that the low-priority ones will be dropped when the maximum threshold is reached
- •Pausing and stopping sounds before they finish playing
- •Looping sounds
- •Changing the playback rate (effectively, the pitch of each sound)
- •Setting stereo volume (separate for left and right channels)

As shown in Figure 1, .ogg files are used to play audio files using SoundPool class. The below code snippet shows the usage of SoundPool class.

```
private int BIG_WAVE = 1;
/*
```

```
 Initialize SoundPool
1 – number of sounds that will be loaded.
AudioManager.STREAM_MUSIC- Stream type
0 – Quality of the sound. Currently no effect
*/

SoundPool soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);

//Create HashMap
HashMap soundPoolMap = new HashMap();

//Create AudioManager
AudioManager  mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);

OR

AudioManager  mAudioManager =

(AudioManager)getContext().getSystemService(AUDIO_SERVICE);

//Load audio file
int soundID = mSoundPool.load(this, R.raw.bigwave, 1);

//Put key value pair in HashMap
soundPoolMap.put(BIG_WAVE, soundID);

//Play sound
int streamVolume =
                mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
int actualVolume =
                mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
int maxVolume =
                 mAudioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);

soundPool.play(
                 mSoundPoolMap.get(BIG_WAVE), streamVolume, streamVolume, 1, 0, 1f);
```

Playing Audio from res

SoundPool Class **load (Context context, int resId, int priority)** method is used to load the audio file from the raw folder. If you want to load a sound from the raw resource file "bigwave.mp3", you would specify "R.raw.bigwave" as the resource ID. Returns the integer **SoundID** that can be use to play the sound.

Playing Audio from file-system

SoundPool Class **load (String path, int priority)** method is used to load the audio file from raw

folder. If you want to load a sound from the raw resource file "bigwave.mp3", you would specify "R.raw.bigwave" as the resource ID. This returns the integer **SoundID** that can be use to play the sound.

ToneGenerator

This class provides methods to play DTMF tones, call supervisory tones and proprietary tones.

The ToneGenerator object requires an output stream type and volume. startTone() is used to play the tone. startTone() method starts the playback of a tone of the specified type for the specified duration.

```
//Output Stream Type and Volume
ToneGenerator tg=new ToneGenerator(AudioManager.STREAM_RING, 100);
//Play Tone
tg.startTone(ToneGenerator.TONE_CDMA_ABBR_ALERT);
```

Playing Video Files

VideoView

Android provides a specialized view control **android.widget.VideoView** that encapsulates creating and initializing the MediaPlayer. The VideoView class can load images from various sources (such as resources or content providers), and takes care of computing its measurement from the video so that it can be used in any layout manager, also providing various display options such as scaling and tinting.
VideoView can be used to display video files present in SDCard FileSystem or files present online.

To add VideoView control the layout file present in res folder will look like this

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
<VideoView
        android:id="@+id/videoView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
</LinearLayout>
```

and the code snippet is shown below

```
//Create VideoView
VideoView videoView = (VideoView)this.findViewById(R.id.videoView);
//Create MediaController
MediaController mc = new MediaController(this);
//Set MediaController to VideoView
videoView.setMediaController(mc);
//Set video path of SD Card
videoView.setVideoURI(Uri.parse("file:///sdcard/samplemp4.mp4"));
OR
//Set video web path
```

```
videoView.setVideoURI(Uri.parse("http://www.xyz.com/../sample3gp.3gp"));
//Set requestFocus
videoView.requestFocus();
//Play Video
videoView.start();
```

Playing Video from file-system

VideoView **setVideoURI(Uri uri)** set the video path from where the video file will be access. Here specify the path of the file-system. Use the **Uri.parse(String path)** static method which converts the string into Uri.

Playing Video from Web

This is the same as playing video from the file-system, the only difference is the path. Here the path points to the website.

SurfaceView

SurfaceView allows us to specify our own display surface and manipulate the underlying Media Player instance directly.

The first step to using the Media Player to view video content is to prepare a Surface onto which the video will be displayed. The Media Player requires a SurfaceHolder object for displaying video content, assigned using the setDisplay() method.

To include a Surface Holder in UI layout, use the SurfaceView control as in the layout file present in res folder.

```
<--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <SurfaceView android:id="@+id/surface"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">
    </SurfaceView>
</LinearLayout>
```

the below code snippet shows how SurfaceView and SurfaceHolder is created.

```
//create SurfaceView
SurfaceView surfaceView = (SurfaceView) findViewById(R.id.surface);
//get SurfaceHolder
SurfaceHolder  holder = mPreview.getHolder();
//add callback
holder.addCallback(this);
//set Surface Type
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

```
//set Surface size
Holder.setFixedSize(width, height); //specify the width and height here
```

To display video the SurfaceHolder.Callback interface needs to be implemented. SurfaceHolder.Callback interface has three methods

```
@Override
public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
}
@Override
public void surfaceCreated(SurfaceHolder holder) {
}
@Override
public void surfaceDestroyed(SurfaceHolder arg0) {
}
```

SurfaceHolder class helps us to control the surface (icluding the size or, format of the surface... if the surface changes). Once the surface is created the surfaceCreated(SurfaceHolder holder) gets called. After this create a MediaPlayer object and set the other parameters.

Playing Video from res

A MediaPlayer object needs to be created here using the MediaPlayer static method MediaPlayer create (Context context, int resid). The Media Player requires a SurfaceHolder object for displaying video content, assigned using the setDisplay() method.

```
//Create MediaPlayer from res
MediaPlayer  mMediaPlayer = MediaPlayer.create(this, R.raw.samplemp4);
//Set the display
mMediaPlayer.setDisplay(holder);
//Set other parameters
mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
//start the player
mMediaPlayer.start();
```

Playing Video from file-system

In this case the file path needs to be set to the MediaPlayer using the **setDataSource (String path)** method. After this mediaplayer needs to be prepared using **prepare()** method. prepare() method Prepares the player for playback, synchronously. After setting the datasource and the display surface, you need to either call prepare() or prepareAsync().
Use the try – catch Exception clause since certain methods such as setDataSource(), prepare() can throw up exceptions.

```
String pathToFile = "/sdcard/samplemp4.mp4";
// Create a new media player and set the listeners
MediaPlayer mMediaPlayer = new MediaPlayer();
mMediaPlayer.setDataSource(pathToFile);
mMediaPlayer.setDisplay(holder);
mMediaPlayer.prepare();
// set listeners, if required by application
```

```
mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mMediaPlayer.start();
```

Playing Video from web

In this case the file path needs to be changed and set to the web site to access the video. Everything else remains the same.

```
String pathToFile = "http://www.xyz.com/.../samplemp4.mp4";
// Create a new media player and set the listeners
MediaPlayer mMediaPlayer = new MediaPlayer();
mMediaPlayer.setDataSource(pathToFile);
mMediaPlayer.setDisplay(holder);
mMediaPlayer.prepare();
// set listeners, if required by application
mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mMediaPlayer.start();
```

Sample Example

The following sample example shows the usage of MediaPlayer, SoundPool, ToneGenerator, VideoView, SurfaceView class and MediaPlayer listeners to play Audio and Video files present in res folder, file-system and internet.

On launch, the application shows two options **Audio, or Video**. The xml and code is listed below

main.xml

```
<!--l version="1.0" encoding="utf-8-->
<<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <<Button android:text="Audio"
    android:id="@+id/button1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    <</Button>

    <<Button android:text="Video"
             android:id="@+id/button2"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    <</Button>
</LinearLayout>
```

AudioVideoPlayer

```
package com.samsung.player;

import android.app.Activity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class AudioVideoPlayer extends Activity implements OnClickListener{

private Button btnAudio;
private Button btnVideo;

public static final int AUDIO = 1;
public static final int VIDEO = 2;
public static final String PLAY_WHAT = "AUDIO_VIDEO";
public static final String PLAY_AUDIO = "AUDIO";
public static final String PLAY_VIDEO = "VIDEO";

 /** Called when the activity is first created. */
 @Override
 public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnAudio = (Button)findViewById(R.id.button1);
        btnAudio.setOnClickListener(this);
        btnVideo = (Button)findViewById(R.id.button2);
        btnVideo.setOnClickListener(this);
 }

public void onClick(View v) {
  if(v == btnAudio) {
     Intent intent = new Intent(this.getApplication(),MyPlayerMenu.class);
     intent.putExtra(PLAY_WHAT, AUDIO);
    startActivity(intent);
  } else
  if(v == btnVideo) {
     Intent intent = new Intent(this.getApplication(),MyPlayerMenu.class);
     intent.putExtra(PLAY_WHAT, VIDEO);
     startActivity(intent);
  }
}
}
```

On selection of Audio the application shows the audio menu as listed below

myplayermenu.xml

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <Button android:text="Play Audio from res"
    android:id="@+id/menubutton1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    </Button>

    <Button android:text="Play Audio from filesystem"
             android:id="@+id/menubutton2"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>

    <Button android:text="Play Audio from web stream"
             android:id="@+id/menubutton3"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
    <Button android:text="Play Audio Tone (ToneGenerator)"
             android:id="@+id/menubutton4"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
    <Button android:text="SoundPool"
             android:id="@+id/menubutton5"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
</LinearLayout>
```

MyPlayerMenu

```
package com.samsung.player;

import android.app.Activity;
import android.content.Intent;
import android.media.AudioManager;
import android.media.ToneGenerator;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MyPlayerMenu extends Activity implements OnClickListener {

        private Button btnRes;
        private Button btnFile;
        private Button btnWeb;
        private Button btnTone;
        private Button btnSoundPool;
        private int value;

        public static final int PLAY_AUDIO_FROM_RES = 3;
        public static final int PLAY_AUDIO_FROM_FILESYSTEM = 4;
        public static final int PLAY_AUDIO_FROM_WEB = 5;
        public static final int PLAY_AUDIO_FROM_TONE = 6;
        public static final int PLAY_AUDIO_FROM_SOUNDPOOL = 7;
        public static final int PLAY_VIDEO_FROM_RES = 8;
        public static final int PLAY_VIDEO_FROM_FILESYSTEM = 9;
        public static final int PLAY_VIDEO_FROM_WEB = 10;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myplayermenu);
        btnRes = (Button)findViewById(R.id.menubutton1);
        btnRes.setOnClickListener(this);
        btnFile = (Button)findViewById(R.id.menubutton2);
        btnFile.setOnClickListener(this);
        btnWeb = (Button)findViewById(R.id.menubutton3);
        btnWeb.setOnClickListener(this);
        btnTone = (Button)findViewById(R.id.menubutton4);
        btnTone.setOnClickListener(this);
        btnSoundPool = (Button)findViewById(R.id.menubutton5);
        btnSoundPool.setOnClickListener(this);
        Bundle extras = getIntent().getExtras();
        value = extras.getInt(AudioVideoPlayer.PLAY_WHAT);
        if(value == AudioVideoPlayer.VIDEO) {
                btnRes.setText("Play Video from res");
                btnFile.setText("Play Video from file");
```

```
                btnWeb.setText("Play Video from web");
                btnTone.setEnabled(false);
                btnSoundPool.setEnabled(false);
        }else
        if(value == AudioVideoPlayer.AUDIO)  {
                btnRes.setText("Play Audio from res");
                btnFile.setText("Play Audio from file");
                btnWeb.setText("Play Audio from web");
                btnTone.setEnabled(true);
                btnSoundPool.setEnabled(true);
        }
    }

    public void onClick(View v) {
        if(v == btnRes) {
        Intent intent = null;
        if(value == AudioVideoPlayer.AUDIO) {
               intent = new Intent(this.getApplication(),MyAudioPlayer.class);
            intent.putExtra(AudioVideoPlayer.PLAY_AUDIO, PLAY_AUDIO_FROM_RES);
        }else
        {
        intent = new Intent(this.getApplication(),MySurfaceViewVideoPlayer.class);
        intent.putExtra(AudioVideoPlayer.PLAY_VIDEO, PLAY_VIDEO_FROM_RES);
          }
          startActivity(intent);
        }else
        if(v == btnFile) {
        Intent intent = null;
        if(value == AudioVideoPlayer.AUDIO) {
                intent = new Intent(this.getApplication(),MyAudioPlayer.class);
    intent.putExtra(AudioVideoPlayer.PLAY_AUDIO, PLAY_AUDIO_FROM_FILESYSTEM);
        } else
            {
         intent = new Intent(this.getApplication(),MyVideoPlayerMenu.class);
        intent.putExtra(AudioVideoPlayer.PLAY_VIDEO, PLAY_VIDEO_FROM_FILESYSTEM);
        }
        startActivity(intent);
          }else
          if(v == btnWeb)       {
         Intent intent = new Intent(this.getApplication(),MyAudioPlayer.class);
         if(value == AudioVideoPlayer.AUDIO) {
              intent.putExtra(AudioVideoPlayer.PLAY_AUDIO, PLAY_AUDIO_FROM_WEB);
        }else
```

```
        {
             intent = new Intent(this.getApplication(),MyVideoPlayerMenu.class);
             intent.putExtra(AudioVideoPlayer.PLAY_VIDEO, PLAY_VIDEO_FROM_WEB);
        }
        startActivity(intent);
         }else
         if(v == btnTone) {
        ToneGenerator tg=new ToneGenerator(AudioManager.STREAM_RING, 100);
        tg.startTone(ToneGenerator.TONE_CDMA_ABBR_ALERT);
         }else
         if(v == btnSoundPool) {
        Intent intent = new Intent(this.getApplication(),MySoundPoolAudioPlayer.class);
intent.putExtra(AudioVideoPlayer.PLAY_AUDIO, PLAY_AUDIO_FROM_SOUNDPOOL);
        startActivity(intent);
        }
    }
}
```

On selecting audio from res, file-system and web, MyAudioPlayer gets called.

myaudioplayer.xml

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:id="@+id/myaudioplayer"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:text="Text"
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"  />
    <Button android:text="Play"
             android:id="@+id/playButton"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
</LinearLayout>
```

MyAudioPlayer

```
package com.samsung.player;

import java.io.IOException;
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MyAudioPlayer extends Activity implements OnClickListener{
         private Button btnPausePlay;
         private MediaPlayer mediaPlayer;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.myaudioplayer);

         btnPausePlay = (Button)findViewById(R.id.playButton);
         btnPausePlay.setOnClickListener(this);
         btnPausePlay.setText("Pause");

         Bundle extras = getIntent().getExtras();
         int value = extras.getInt(AudioVideoPlayer.PLAY_AUDIO);
         playAudio(value);
    }

    private void playAudio(int value)  {
         if(value == MyPlayerMenu.PLAY_AUDIO_FROM_RES) {
           mediaPlayer = MediaPlayer.create(this, R.raw.samplemp3);
            mediaPlayer.start();
         } else
         if(value == MyPlayerMenu.PLAY_AUDIO_FROM_FILESYSTEM) {
                String pathToFile = "/sdcard/samplemp3.mp3";
            //Uri uri = Uri.parse(pathToFile);
            //mediaPlayer = MediaPlayer.create(this, uri);
            mediaPlayer = new MediaPlayer();
            try {
              mediaPlayer.setDataSource(pathToFile);
            } catch (IllegalArgumentException e) {
                 e.printStackTrace();
            } catch (IllegalStateException e) {
                    e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            try {
```

```
                mediaPlayer.prepare();
            } catch (IllegalStateException e) {
                e.printStackTrace();
                 } catch (IOException e) {
                e.printStackTrace();
            }
            mediaPlayer.start();
        } else
        if(value == MyPlayerMenu.PLAY_AUDIO_FROM_WEB) {
            String pathToFile = "http://www.xyz.com/Audio/sample.mp3";
            //Uri uri = Uri.parse(pathToFile);
            //mediaPlayer = MediaPlayer.create(this, uri);
            mediaPlayer = new MediaPlayer();
            try {
              mediaPlayer.setDataSource(pathToFile);
            } catch (IllegalArgumentException e) {
              e.printStackTrace();
            } catch (IllegalStateException e) {
              e.printStackTrace();
            } catch (IOException e) {
             e.printStackTrace();
        }
        try {
            mediaPlayer.prepare();
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        mediaPlayer.start();
       }
    }

 // Initiate media player pause
    private void demoPause(){
         mediaPlayer.pause();
         btnPausePlay.setText("Play");
    }

// Initiate playing the media player
   private void demoPlay(){
         mediaPlayer.start();
         btnPausePlay.setText("Pause");
```

```
   }

public void onClick(View v) {
  if(v == btnPausePlay) {
    if(mediaPlayer.isPlaying()) {
      demoPause();
    } else {
      demoPlay();
     }
   }
}

@Override
 protected void onDestroy() {
        super.onDestroy();
        releaseMediaPlayer();
 }

 private void releaseMediaPlayer() {
        if (mediaPlayer != null) {
                 mediaPlayer.release();
                 mediaPlayer = null;
        }
    }
}
```

On selection of SoundPool, the application shows the SoundPool menu as shown below

mysoundpoolplayermenu.xml

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button android:text="BigWave"
    android:id="@+id/soundbutton1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    </Button>
    <Button android:text="FlameMagic"
             android:id="@+id/soundbutton2"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
```

```
    <Button android:text="MetalHit"
             android:id="@+id/soundbutton3"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
    <Button android:text="TornadoMagic"
             android:id="@+id/soundbutton4"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
</LinearLayout>
```

MySoundPoolAudioPlayer

```
package com.samsung.player;

import java.util.HashMap;
import android.app.Activity;
import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MySoundPoolAudioPlayer extends Activity implements OnClickListener{
         private Button btnBigWave;
         private Button btnFlameMagic;
         private Button btnMetalHit;
         private Button btnTornadoMagic;

         private  SoundPool mSoundPool;
         private  HashMap mSoundPoolMap;
         private  AudioManager  mAudioManager;

         public static final int BIG_WAVE = 11;
         public static final int FLAME_MAGIC = 12;
         public static final int METAL_HIT = 13;
         public static final int TORNADO = 14;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.mysoundpoolplayermenu);
```

```
        btnBigWave = (Button)findViewById(R.id.soundbutton1);
        btnBigWave.setOnClickListener(this);
        btnFlameMagic = (Button)findViewById(R.id.soundbutton2);
        btnFlameMagic.setOnClickListener(this);
        btnMetalHit = (Button)findViewById(R.id.soundbutton3);
        btnMetalHit.setOnClickListener(this);
        btnTornadoMagic = (Button)findViewById(R.id.soundbutton4);
        btnTornadoMagic.setOnClickListener(this);

        loadSoundPool();
    }

    private void loadSoundPool()  {
        mSoundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 0);
            mSoundPoolMap = new HashMap();
            mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);

            int soundID = mSoundPool.load(this, R.raw.bigwave, 1);
            //OR
            //String pathToFile = "/sdcard/bigwave.ogg";
            //int soundID = mSoundPool.load(pathToFile, 1);

            addSound(BIG_WAVE, soundID);

            soundID = mSoundPool.load(this, R.raw.flamemagic, 1);
            addSound(FLAME_MAGIC, soundID);

            soundID = mSoundPool.load(this, R.raw.metalhit, 1);
            addSound(METAL_HIT, soundID);

            soundID = mSoundPool.load(this, R.raw.tornadomagic, 1);
            addSound(TORNADO, soundID);
    }

    private void addSound(int index, int SoundID)  {
        mSoundPoolMap.put(index, SoundID);
    }

    public void playAudio(int index) {
     int streamVolume =
                              mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
     /*float actualVolume =
                 (float) mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
```

```
     float maxVolume =
             (float) mAudioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
     float streamVolume = actualVolume / maxVolume;
     */
   mSoundPool.play(mSoundPoolMap.get(index), streamVolume, streamVolume, 1, 0, 1f);
 }

 public void playLoopAudio(int index) {
     int streamVolume =
             mAudioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
  mSoundPool.play(mSoundPoolMap.get(index), streamVolume, streamVolume, 1, -1, 1f);
}

@Override
  protected void onDestroy() {
        super.onDestroy();
    }

  public void onClick(View v) {
     if(v == btnBigWave) {
        playAudio(BIG_WAVE);
    }else
    if(v == btnFlameMagic) {
        playAudio(FLAME_MAGIC);
    }else
     if(v == btnMetalHit)       {
        playAudio(METAL_HIT);
     }else
     if(v == btnTornadoMagic) {
        playAudio(TORNADO);
    }
   }
}
```

Similarly on selection of the Video option from the Main Menu shown in AudioVideoPlayer class, it shows the sub menus as VideoView and SurfaceView as shown in the xml and class.

myvideoplayermenu.xml

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

```
  <Button android:text="Play using VideoView"
    android:id="@+id/menubutton1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    </Button>

    <Button android:text="Play using SurfaceView"
             android:id="@+id/menubutton2"
             android:layout_height="wrap_content"
             android:layout_width="match_parent">
    </Button>
</LinearLayout>
```

MyVideoPlayerMenu

```
package com.samsung.player;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MyVideoPlayerMenu extends Activity implements OnClickListener{

     private Button btnVideoView;
     private Button btnSurfaceView;
     private int value;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myvideoplayermenu);
        btnVideoView = (Button)findViewById(R.id.menubutton1);
        btnVideoView.setOnClickListener(this);
        btnSurfaceView = (Button)findViewById(R.id.menubutton2);
        btnSurfaceView.setOnClickListener(this);
        Bundle extras = getIntent().getExtras();
        value = extras.getInt(AudioVideoPlayer.PLAY_VIDEO);
    }

    public void onClick(View v) {
      if(v == btnVideoView) {
```

```
          Intent intent = new Intent(this.getApplication(),MyVideoViewVideoPlayer.class);
          intent.putExtra(AudioVideoPlayer.PLAY_VIDEO, value);
          startActivity(intent);
      }else
      if(v == btnSurfaceView) {
        Intent intent = new Intent(this.getApplication(),MySurfaceViewVideoPlayer.class);
        intent.putExtra(AudioVideoPlayer.PLAY_VIDEO, value);
        startActivity(intent);
      }
   }
}
```

If VideoView is selected then the videoview menus are shown listed in xml and class files

myvideoviewvideoplayer

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <VideoView
        android:id="@+id/videoView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
</LinearLayout>
```

MyVideoViewVideoPlayer

```

package com.samsung.player;

import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class MyVideoViewVideoPlayer extends Activity {

    private VideoView videoView;
    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.myvideoviewvideoplayer);
```

```
        videoView = (VideoView)this.findViewById(R.id.videoView);
        MediaController mc = new MediaController(this);
        videoView.setMediaController(mc);

        Bundle extras = getIntent().getExtras();
        int value = extras.getInt("VIDEO");
        playVideo(value);
    }

    private void playVideo(int value)   {
        if(value == MyPlayerMenu.PLAY_VIDEO_FROM_FILESYSTEM)    {
                videoView.setVideoURI(Uri.parse("file:///sdcard/samplemp4.mp4"));
            videoView.requestFocus();
            videoView.start();
        } else
 if(value == MyPlayerMenu.PLAY_VIDEO_FROM_WEB) {
   videoView.setVideoURI(Uri.parse("http://www.xyzo.com/.../sample3gp.3gp"));
        videoView.requestFocus();
        videoView.start();
        }
    }
}
```

mysurfaceviewvideoplayer.xml

```
<!--l version="1.0" encoding="utf-8-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <SurfaceView android:id="@+id/surface"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">
    </SurfaceView>
</LinearLayout>
```

MySurfaceViewVideoPlayer

```
package com.samsung.player;

import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaPlayer.OnVideoSizeChangedListener;
```

```
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class MySurfaceViewVideoPlayer extends Activity implements
         OnPreparedListener, OnVideoSizeChangedListener, SurfaceHolder.Callback {

    private int mVideoWidth;
    private int mVideoHeight;
    private boolean mIsVideoSizeKnown = false;
    private boolean mIsVideoReadyToBePlayed = false;

    private MediaPlayer mMediaPlayer;
    private SurfaceView mPreview;
    private SurfaceHolder holder;

 /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.mysurfaceviewvideoplayer);
        mPreview = (SurfaceView) findViewById(R.id.surface);
        holder = mPreview.getHolder();
        holder.addCallback(this);
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
        if (width == 0 || height == 0) {  return;       }
        mIsVideoSizeKnown = true;
        mVideoWidth = width;
        mVideoHeight = height;
        if (mIsVideoReadyToBePlayed && mIsVideoSizeKnown) {
            startVideoPlayback();
        }
    }

    public void onPrepared(MediaPlayer mediaplayer) {
        mIsVideoReadyToBePlayed = true;
        if (mIsVideoReadyToBePlayed && mIsVideoSizeKnown) {
            startVideoPlayback();
```

```
    }
}

public void surfaceChanged(SurfaceHolder surfaceholder, int i, int j, int k) {
}

public void surfaceDestroyed(SurfaceHolder surfaceholder) {
}

public void surfaceCreated(SurfaceHolder holder) {
    Bundle extras = getIntent().getExtras();
    int value = extras.getInt(AudioVideoPlayer.PLAY_VIDEO);
    playVideo(value);
}
private void playVideo(Integer value) {
    doCleanUp();
    try {
        switch (value) {
          case MyPlayerMenu.PLAY_VIDEO_FROM_RES:
            mMediaPlayer = MediaPlayer.create(this, R.raw.samplemp4);
            mMediaPlayer.setDisplay(holder);
            mMediaPlayer.setOnVideoSizeChangedListener(this);
            mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
            mMediaPlayer.start();
          break;
          case MyPlayerMenu.PLAY_VIDEO_FROM_FILESYSTEM:
                String pathToFile = "/sdcard/samplemp4.mp4";
                // Create a new media player and set the listeners
                mMediaPlayer = new MediaPlayer();
                mMediaPlayer.setDataSource(pathToFile);
                mMediaPlayer.setDisplay(holder);
                mMediaPlayer.prepare();
                mMediaPlayer.setOnPreparedListener(this);
                mMediaPlayer.setOnVideoSizeChangedListener(this);
                mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
           break;
           case MyPlayerMenu.PLAY_VIDEO_FROM_WEB:
                String pathToWeb= "http://www.xyz.com/.../sample3gp.3gp";
                mMediaPlayer = new MediaPlayer();
                mMediaPlayer.setDataSource(pathToWeb);
                mMediaPlayer.setDisplay(holder);
                mMediaPlayer.prepare();
                mMediaPlayer.setOnPreparedListener(this);
```

```
                    mMediaPlayer.setOnVideoSizeChangedListener(this);
                    mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
                break;
            }
        } catch (Exception e) {
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        releaseMediaPlayer();
        doCleanUp();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        releaseMediaPlayer();
        doCleanUp();
    }

    private void releaseMediaPlayer() {
        if (mMediaPlayer != null) {
            mMediaPlayer.release();
            mMediaPlayer = null;
        }
    }

    private void doCleanUp() {
        mVideoWidth = 0;
        mVideoHeight = 0;
        mIsVideoReadyToBePlayed = false;
        mIsVideoSizeKnown = false;
    }

    private void startVideoPlayback() {
        holder.setFixedSize(mVideoWidth, mVideoHeight);
        mMediaPlayer.start();
    }
}
```

Following is the AndroidManifest.xml file

AndroidManifest.xml

```
<!--l version="1.0" encoding="utf-8-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
       package="com.samsung.player"
       android:versionCode="1"
       android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">

      <activity android:name=".AudioVideoPlayer"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"></action>
                <category android:name="android.intent.category.LAUNCHER"></category>
            </intent-filter>
        </activity>
        <activity android:name=".MyPlayerMenu" android:label="PlayerMenu">
            <intent-filter>
                <category android:name="android.intent.category.SAMPLE_CODE" />
            </intent-filter>
        </activity>

        <activity android:name=".MyAudioPlayer" android:label="AudioPlayer">
            <intent-filter>
                <category android:name="android.intent.category.SAMPLE_CODE" />
            </intent-filter>
        </activity>

       <activity android:name=".MySoundPoolAudioPlayer" android:label="SoundPool
AudioPlayer">
            <intent-filter>
                <category android:name="android.intent.category.SAMPLE_CODE" />
            </intent-filter>
        </activity>

        <activity android:name=".MyVideoPlayerMenu" android:label="VideoPlayerMenu">
            <intent-filter>
                <category android:name="android.intent.category.SAMPLE_CODE" />
            </intent-filter>
        </activity>

        <activity android:name=".MySurfaceViewVideoPlayer" android:label="SurfaceView
```

```
VideoPlayer">
             <intent-filter>
                 <category android:name="android.intent.category.SAMPLE_CODE" />
             </intent-filter>
         </activity>

         <activity android:name=".MyVideoViewVideoPlayer" android:label="VideoView
VideoPlayer">
             <intent-filter>
                 <category android:name="android.intent.category.SAMPLE_CODE" />
             </intent-filter>
         </activity>
     </application>

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

</manifest>
```

ref:http://developer.samsung.com/android/technical-docs/Playing-Audio-and-Video-in-Android