**Multimedia**

In Mobile technology multimedia represents **audio operations**, **video operations** and **Image operations**.

**Audio Operations**

We can play and control the audio files in android by the help of **android.media.MediaPlayer class**.

**Methods of MediaPlayer class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **public void setDataSource(String path)** | sets the data source (file path or http url) to use. |
| **public void prepare()** | prepares the player for playback synchronously. |
| **public void start()** | it starts or resumes the playback. |
| **public void stop()** | it stops the playback. |
| **public void pause()** | it pauses the playback. |
| **public boolean isPlaying()** | checks if media player is playing. |
| **public void seekTo(int millis)** | seeks to specified time in miliseconds. |
| **public void setLooping(boolean looping)** | sets the player for looping or non-looping. |
| **public boolean isLooping()** | checks if the player is looping or non-looping. |
| **public void setVolume(float leftVolume,float rightVolume)** | sets the volume on this player. |

**Example to play Music in Background**

**Note: -** create a new folder in res and name it as raw and copy the audio file into it (Note that audio file name should be in small letters only)

****

**Java Snippet (MainActivity.java)**

MediaPlayer mp ;

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

mp = MediaPlayer.*create*(getApplicationContext(), R.raw.*gopalagopala1*);

mp.start();

}

Note: - Install **.apk** file in real device and run the application for output.

**Example to play Music in Background with external commands **

**Java Snippet (MainActivity.java)**

MediaPlayer mp ,mp1;

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

mp = MediaPlayer.*create*(getApplicationContext(), R.raw.*gopalagopala1*);

mp.start();

}

**public** **void** playMusic(View v){

mp.start();

}

**public** **void** pauseMusic(View v){

mp.pause();

}

**Layout Snippet (res/layout/activity\_mail.xml)**

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"50dp"*

android:layout\_height=*"25dp"*

android:onClick=*"playMusic"*

android:src=*"@drawable/on"* />

<ImageView

android:id=*"@+id/imageView2"*

android:layout\_width=*"50dp"*

android:layout\_height=*"25dp"*

android:onClick=*"pauseMusic"*

android:src=*"@drawable/off"* />

**Example** to play the audio file. Here, we are going to play **maine.mp3** file located inside the **sdcard/Music** directory.

**Java Snippet (MainActivity.java)**

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

         setContentView(R.layout.activity\_main);

        MediaPlayer mp=**new** MediaPlayer();

**try**

{

            mp.setDataSource("/sdcard/Music/maine.mp3");//Write your location here

            mp.prepare();

            mp.start();

         } **catch**(Exception e){e.printStackTrace();}

    }

**Voice Recording and playing**

**android.media.MediaRecorder** class can be used to record audio and video files.

After recording the media, we can create a sound file that can be played later.

|  |
| --- |
| **Method & description** |
| **setAudioSource()**  This method specifies the source of audio to be recorded |
| **setVideoSource()**  This method specifies the source of video to be recorded |
| **setOutputFormat()**  This method specifies the audio format in which audio to be stored |
| **setAudioEncoder()**  This method specifies the audio encoder to be used  Encoding is a format of converting analog signals into digital signals. |
| **setOutputFile()**  This method configures the path to the file into which the recorded audio is to be stored |
| **stop()**  This method stops the recording process. |
| **release()**  This method should be called when the recorder instance is needed. |

**Activity Snippet**

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"100dp"*

android:layout\_height=*"100dp"*

android:src=*"@drawable/recorder\_icon"* />

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"start"*

android:text=*"@string/start"* />

<Button

android:id=*"@+id/button2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"stop"*

android:text=*"@string/stop"* />

<Button

android:id=*"@+id/button3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"play"*

android:text=*"@string/play"* />

**Java Snippet (Main\_Activity.java)**

**public** **class** MainActivity **extends** Activity

{

**private** MediaRecorder myAudioRecorder;

**private** String outputFile = **null**;

**private** Button start,stop,play;

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

start = (Button)findViewById(R.id.*button1*);

stop = (Button)findViewById(R.id.*button2*);

play = (Button)findViewById(R.id.*button3*);

stop.setEnabled(**false**);

play.setEnabled(**false**);

outputFile = Environment.*getExternalStorageDirectory*().

getAbsolutePath() + "/myrecording.3gp";

myAudioRecorder = **new** MediaRecorder();

myAudioRecorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);

myAudioRecorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);

myAudioRecorder.setAudioEncoder(MediaRecorder.OutputFormat.*AMR\_NB*);

//Adaptive Multi-Rate Narrowband (*AMR\_NB )*

myAudioRecorder.setOutputFile(outputFile);

}

**public** **void** start(View view)

{

**try**

{

myAudioRecorder.prepare();

myAudioRecorder.start();

}

**catch** (IllegalStateException e)

{

e.printStackTrace();

}

**catch** (IOException e)

{

e.printStackTrace();

}

start.setEnabled(**false**);

stop.setEnabled(**true**);

Toast.*makeText*(getApplicationContext(), "Recording started", Toast.*LENGTH\_LONG*).show();

}

**public** **void** stop(View view)

{

myAudioRecorder.stop();

myAudioRecorder.release();

myAudioRecorder = **null**;

stop.setEnabled(**false**);

play.setEnabled(**true**);

Toast.*makeText*(getApplicationContext(), "Audio recorded successfully",

Toast.*LENGTH\_LONG*).show();

}

**public** **void** play(View view) **throws** Exception

{

MediaPlayer m = **new** MediaPlayer();

m.setDataSource(outputFile);

m.prepare();

m.start();

Toast.*makeText*(getApplicationContext(), "Playing audio", Toast.*LENGTH\_LONG*).show();

}

}

**User Permission (AndroidManifest.xml)**

<uses-permission android:name=*"android.permission.RECORD\_AUDIO"*/>

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"*/>

**Image Capture through Android Device**

**android.provider.MediaStore**

The Media provider contains meta data for all available media on both internal and external storage devices.

Intent i = **new** Intent (MediaStore.*ACTION\_IMAGE\_CAPTURE*);

**ACTION\_IMAGE\_CAPTURE**

Standard Intent action that can be sent to have the camera application capture an image and return it.

startActivityForResult(i, *CAPTURE\_IMAGE\_CAPTURE\_CODE*);

*i* - Intent to start

**CAPTURE\_IMAGE\_CAPTURE\_CODE**

A constant value (request code),

If >= 0, this code will be returned in **onActivityResult()** when the activity exits.

When this activity exits, your onActivityResult() method will be called with the given requestCode.

**private** **static** **final** **int** *CAPTURE\_IMAGE\_CAPTURE\_CODE* = 0;

**public** **void** openCamera(View v)

{

Intent i = **new** Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);

startActivityForResult(i, *CAPTURE\_IMAGE\_CAPTURE\_CODE*);

}

According to the **Android Developers Page**, **onActivityResult** is “Called when an activity you launched exits, giving you the requestCode you started it with, the resultCode it returned, and any additional data from it.”

Since we opened an Activity for a result (using the **startActivityForResult** method), once that activity exits, the **onActivityResult** method will get called, and it will handled the result however we choose.

First, let’s break down the three parameters for the **onActivityResult** method, **requestCode**, **resultCode**, and **data**.

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data)

{

**if** (requestCode == *CAPTURE\_IMAGE\_CAPTURE\_CODE*)

{

**if** (resultCode == *RESULT\_OK*)

{

Toast.*makeText*(**this**, "Image Captured", Toast.*LENGTH\_LONG*).show();

}

**else**

**if** (resultCode == *RESULT\_CANCELED*)

{

Toast.*makeText*(**this**, "Cancelled", Toast.*LENGTH\_LONG*).show();

}

}

}

**android.app.Activity.RESULT\_OK = -1**

Standard activity result: operation succeeded.

**android.app.Activity.RESULT\_CANCELED = 0**

Standard activity result: operation canceled.

**Snippet (Activity)**

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"50dp"*

android:layout\_height=*"50dp"*

android:onClick=*"openCamera"*

android:src=*"@drawable/camera"* />

**Video Operations**

**Playing Video**

Android's MediaPlayer class works with video files the same way it does with plain audio/music files. But there is a small difference.

We need to create a surface to play the video. We can use start() or stop() methods to control the play.

**android.widget.VideoView** class provides bunch of methods that can be used in video play.  
 **setVideoPath()**

This method is used to specify the path of the video file to be played. This can either be a local path or a remote URL.

**setVideoURI()**

This method is same as above method except the parameter.

**start()**

Thismethod is used to start the video playback.

**stop()**

This method is used to stop the video playback.

**pause()**

This method pauses the video.

To check whether the video is playing or not **isPlaying()**method can be used. This will return a boolean value.

**Example to play Video**

**Note: -** create a new folder in res and name it as raw and copy the video file into it.

****

**XML Snippet**

<VideoView

android:id=*"@+id/videoView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_centerVertical=*"true"* />

**Java Snippet**

VideoView video = (VideoView) findViewById(R.id.*videoView1*);

// Load and start the video

Uri video1 = Uri.*parse*("android.resource://" + getPackageName() + "/" + R.raw.*sample*);

video.setVideoURI(video1);

video.start();

// Controllers

MediaController mc = **new** MediaController(**this**);

vv.setMediaController(mc);

**Saving Image into Gallary**

**===================**

**String fileName = "temp.jpg";**

**Uri mCapturedImageURI;**

**ContentValues values = new ContentValues();**

**values.put(MediaStore.Images.Media.TITLE, fileName);**

**mCapturedImageURI = getContentResolver() .insert(MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI, values);**