# Lesson - 8

## MULTIMEDIA

# Agenda

- Media Player
- Video view
- Audio Recording
- Video Recording
- Camera & Gallery

# Media Player Basics

- The Android multimedia framework includes support for playing variety of common media types, so that one can easily integrate audio, video, and images into your applications.
- MediaPlayer class is the most important component of media framework from andriod.media package. The various sources from which media can be played are as follows:
  - Resource folder.(res→ raw→ .mp3)
  - File System path.(internal / external storage)
  - URL.
- Check the Developers site for the latest audio and video file format support
  - https://developer.android.com/guide/topics/media/media-formats.html

# Playing audio

- Probably the most basic need for multimedia on a mobile is the ability to play audio files, whether new ringtones, MP3s, or quick audio notes.
- Android's MediaPlayer is easy to use.
- At a high level, all you need to do to play an MP3 file is follow these steps:
  - Put the MP3 in the res/raw directory in a project (note that you can also use a URI to access files on the network or via the internet).
  - Create a new instance of the MediaPlayer, and reference the MP3 by calling MediaPlayer.create().
  - Call the MediaPlayer methods prepare() and start().

# Media player life cycle

- A MediaPlayer goes through several states during its life cycle:
  - Idle - The MediaPlayer is instantiated.
  - Initialized - The media source is set.
  - Preparing - The MediaPlayer is preparing the media source for playback.
  - Prepared -The MediaPlayer is prepared for playback.
  - Started - Playback is in progress.
  - Paused - Playback has been paused.
  - Playback complete - Playback of source is done (the playback can be started again).
  - Stopped - The MediaPlayer is no longer prepared to play the source.
  - End - The MediaPlayer is no more, and all associated resources are released.

- To get started with MediaPlayer, it's useful at this point to view it as a series of steps in your application:

1. Create a MediaPlayer instance through the create() method (idle state).
2. Initialize the MediaPlayer with the media source to play (initialized state).
3. Prepare the MediaPlayer for playback through the prepare() method (preparing and prepared states).
4. Play the MediaPlayer through the start() method (started state).
5. During playback, if desired, you can pause, stop, or replay the MediaPlayer (started, paused, playback complete, and stopped states).
6. Once playback is finished, make sure to release the MediaPlayer's associated resources by calling release() (end state).

# Other useful methods in MediaPlayer

- reset() - Resets the MediaPlayer to its uninitialized state. After calling this method, you will have to initialize it again by setting the data source and calling prepare().
- seekTo() - *Seeks to specified time position.*
- getDuration() - *Gets the duration of the file. R**eturn** the duration in milliseconds, if no duration is available -1 is returned.*
- setDataSource() - transfers a MediaPlayer object in the *Idle* state to the *Initialized* state.
- getCurrentPosition() - *Gets the current playback position.*
- *Refer Complete code from :* Refer Demo : Lesson7-8\MyApplication folder(Play audio from Raw Folder)

# Hands on example to play audio file

- Create a new Project MediaPlayerTest.
- Here is the design requirement.


Top level Linear Layout

Image View – occupies 70% of the screen

Seek bar – occupies 10% of the screen

Arrange 5 ImageView UI horizontally by giving weight property each occupies 20%

Nested Linear Layout contains 5 ImageViews which occupies 20% of screen

# activity_main.xml

Coding for the top level layout , ImageView and SeekBar


```
<LinearLayout
xmlns:android="http://schemas.android.com/
apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.7"
        android:src="@drawable/logo"/>
   <SeekBar
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.1"
        android:id="@+id/sBar"/>
```

# activity_main.xml

```
<LinearLayout
   android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.2"
    android:orientation="horizontal"
    >
```

1

```
<ImageView
    android:layout_width="0dp"
    android:layout_weight="0.2"
    android:layout_height="match_parent"
    android:src="@drawable/backward"
    android:id="@+id/backward"
    android:onClick="back"/>
```

2

```
<ImageView
    android:layout_width="0dp"
    android:layout_weight="0.2"
    android:layout_height="match_parent"
    android:src="@drawable/logo"
    android:id="@+id/play"
    android:onClick="play"
    />
```

3

```
<ImageView
    android:layout_width="0dp"
    android:layout_weight="0.2"
    android:layout_height="match_parent"
    android:src="@drawable/pause"
    android:id="@+id/pause"
    android:onClick="pause"
    />
```

4

```
<ImageView
    android:layout_width="0dp"
    android:layout_weight="0.2"
    android:layout_height="match_parent"
    android:src="@drawable/stop"
    android:id="@+id/stop"
    android:onClick="stop"
    />
```

5

```
<ImageView
            android:layout_width="0dp"
            android:layout_weight="0.2"
            android:layout_height="match_parent"
            android:src="@drawable/forward"
            android:id="@+id/forward"
            android:onClick="forward"
            />
    </LinearLayout>
</LinearLayout>
```

6

# MainActivity.java

- Step : 1 Create a raw folder in res folder and copy the .mp3 songs in that folder.
- Step : 2 Create an object for the MediaPlayer class.
- **public class** MainActivity **extends** AppCompatActivity {
  *// Initialize the MediaPlayer object*
  **MediaPlayer mPlayer;**

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

  *// Invoke the create static method from the MediaPlayer class by passing application context and audio file to play.*
  *// create method return the object of MediaPlayer.*
  **mPlayer = MediaPlayer.*create*(getApplicationContext(),R.raw.*song1*);**
  }

// Code for all the actions like backward, play, stop, pause and forward

```
public void back(View v){
   /*For example the current position is in 500 sec and total duration of audio file is 1000 sec.
*  Apply those values in the formula. You will get 500 - (1000/10) --> 500 - 100 --> 400.
   *  So the song play from the 400 sec position. This is the logic behind the given code.
   *  As a programmer you can write your own logic how many seconds you want to backward
    */
mPlayer.seekTo(mPlayer.getCurrentPosition()-(mPlayer.getDuration()/10));
}
public void play(View v){
    mPlayer.start();
}
public void pause(View v){
   mPlayer.pause();
}
public void stop(View v){
   mPlayer.stop();
}
public void forward(View v){
   // Apply the same logic of backward, here do the addition to move forward
mPlayer.seekTo(mPlayer.getCurrentPosition()+ (mPlayer.getDuration()/10));
}
```

# Coding steps for seekBar

- Step : 1 Declare seekBar object.
- Step : 2 Initialize the seekBar object by using findViewById()
- Step : 3 Set the maximum length for the SeekBar
- Step : 4 Set the setOnSeekBarChangeListener() by passing OnSeekBarChangeListener object as a input parameter. Once the user select the seekBar, automatically this listener is invoked and it will call the onProgressChanged() method.
- Step : 5 Override the below method inside the listener

```
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

// inside this method invoke MediaPlayer object.seekTo() by passing
//progress as a input parameter to update the seekBar progress
}
```

- Step : 6 Frequently update the seekBar position, while playing audio, need to implement Handler by updating current position in the seekBar.

- About Handler in Step 6
  - Handler is part of the Android system's framework for managing threads. A Handler object receives messages and runs code to handle the messages.
  - we use a Handler in order to continuously update the seekbar progress and the time remaining duration.
  - We created a own method updateSeekBar() to handle this part.

```
public class MainActivity extends AppCompatActivity {
    MediaPlayer mPlayer;
    SeekBar sBar; //  Step 1
   // Intitialize the Handler to update the seekBar position
    Handler myHandler = new Handler(); //  Step 6
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlayer = MediaPlayer.create(getApplicationContext(),R.raw.song1);
        sBar = (SeekBar) findViewById(R.id.sBar); //  Step 2
        // Set the maximum length of your audio file in the seekBar.
        sBar.setMax(mPlayer.getDuration()); // Maximum length of the seekBar
                                                        //  Step 3
```

```
// SeekBar listener part -   Step 4
 sBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()  {
        @Override  //  Step 5
       public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        mPlayer.seekTo(progress);
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {  }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {  }
    });
    updateSeekBar();
  }
  Runnable run = new Runnable() {//  Step 6
    @Override
    public void run() {
       updateSeekBar(); // Frequent call in every second to update the seekBar progress
    }
  };
  public void updateSeekBar(){
     // to update seekBar position by getting the current position of mediaplayer with the help of seekBar
            //setProgress()
    sBar.setProgress(mPlayer.getCurrentPosition());

          // seekBar waits for one second, then call the runnable method
    myHandler.postDelayed(run,1000);
  }
```

# Play from SD Card Code

```
public class MainActivity extends AppCompatActivity {
    // Declare the MediaPlayer object
    MediaPlayer mPlayer;
    String path;  // Decalre the path variable
    SeekBar sBar; // Declare the seekBar object
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       // path = "/sdcard/Bale.mp3"; // Not in any folder
        path = "/sdcard/Music/Kanna.mp3"; // song from specific folder

            // create method return the object of MediaPlayer.
        mPlayer = new MediaPlayer();
        try {
            mPlayer.setDataSource(path);
             mPlayer.prepare();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Refer : https://developer.android.com/guide/topics/media/mediaplayer.html

Refer Demo : Lesson7-8\MediaPlayerSDCard  folder(Play audio from SD Card )

# Play Songs from Internet

**Internet Permission** - If you are using MediaPlayer to stream network-based content, your application must request network access.

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

- Playing from a remote URL via HTTP streaming looks like this:
- String url = "http://....."; // your URL here
  MediaPlayer mediaPlayer = new MediaPlayer();
  mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
  mediaPlayer.setDataSource(url);
  mediaPlayer.prepare(); // might take long! (for buffering, etc)
  mediaPlayer.start();

# VideoView

Playing a video is slightly more complicated than playing audio with the MediaPlayer API, in part because you have to provide a view surface for your video to play on. Android has a VideoView widget that handles that task for you; you can use it in any layout manager.

The VideoView class has a wide range of methods that may be called in order to manage the playback of video. Some of the more commonly used methods are as follows:

- **setVideoPath(String path)** – Specifies the path (as a string) of the video media to be played. This can be either the URL of a remote video file or a video file local to the device.
- **setVideoUri(Uri uri)** – Performs the same task as the setVideoPath() method but takes a Uri object as an argument instead of a string.
- **start()** – Starts video playback.
- **pause()** – Pauses video playback.
- **isPlaying()** – Returns a Boolean value indicating whether a video is currently playing.
- **getDuration()** – Returns the duration of the video. Will typically return -1 unless called from within the OnPreparedListener() callback method.
- **getCurrentPosition()** – Returns an integer value indicating the current position of playback.
- **setMediaController(MediaController)** – Designates a MediaController instance allowing playback controls to be displayed to the user.

# Simple Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >
   <VideoView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/videoView"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true" />
</RelativeLayout>
```

Add the below line in your AndriodManifest.xml

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

</uses-permission>
```

# MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        VideoView vView = (VideoView) findViewById(R.id.videoView);
        // Set the path from the external storage
        vView.setVideoPath("/sdcard/waterfall.mp4");
        // Start playing Video
        vView.start();
        //To get the seekbar, pause, play, forward and backward control
        MediaController mc = new MediaController(this);
        // Set the MediaController to the VideoView
        vView.setMediaController(mc);
    }
}
```

Refer Complete Demo from : Lesson7-8\VideoViewDemo

# Output

# Audio Recording

- MediaRecorder : The Android multimedia framework includes support for capturing and encoding a variety of common audio and video formats.
  - This class is used to record audio  from MIC.
  - This class is used to record video from the camera.
  - It is present in android.media.MediaRecorder package.

**Note:** The Android Emulator cannot record audio. Be sure to test your code on a real device that can record.

Refer Complete Demo from : Lesson7-8\AudioRecordTest

# Audio Recording steps using MediaRecorder

1. Initialize a new instance of MediaRecorder with the following calls:
2. Set the audio source using setAudioSource(). You'll probably use MIC.
3. Set the output file format using setOutputFormat().
4. Set the output file name using setOutputFile().
5. Set the audio encoder using setAudioEncoder().
6. Complete the initialization by calling prepare().

- Start and stop the recorder by calling start() and stop() respectively.
- When you are done with the MediaRecorder instance free its resources as soon as possible by calling release().

Refer API :
https://developer.android.com/reference/android/media/MediaRecorder.html

# Example

- Screen Design


Button status/ clicked actions are updated in this text view

By clicking Start to record the audio

By clicking to stop recording

By clicking to stop play the recorded audio

By clicking to play the recorded audio

# Activity_main.xml

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24dp"
        android:id="@+id/tv1"
        android:text="Audio Record Test"
        ></TextView>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <Button
            android:id="@+id/start"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="45dp"
            android:layout_marginTop="30dp"
            android:onClick="start"
            android:text="Start"
            android:textSize="35dp" />
```

# Activity_main.xml

```
        <Button
            android:id="@+id/stop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="45dp"
            android:layout_marginTop="30dp"
            android:onClick="stop"
            android:text="Stop"
            android:textSize="35dp" />
        <Button
            android:id="@+id/stplay"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="30dp"
            android:onClick="stplay"
            android:text="Start Play"
            android:textSize="35dp" />
        <Button
            android:id="@+id/spplay"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="30dp"
            android:onClick="spplay"
            android:text="Stop Play"
            android:textSize="35dp" />
    </LinearLayout>
</LinearLayout>
```

# AndriodManifest.xml

- Add the following permissions in the AndriodManifest.xml

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

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

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

# MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    // 1. Create an object for MeadiaRecorder
    MediaRecorder recorder;
    //To display the status of AudioRecording test
    TextView tv;
    //To play the recorded the audio from SDCard
    MediaPlayer mp;
    String path;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView)findViewById(R.id.tv1);
        recorder = new MediaRecorder();
        //2. Set the audio source MIC for recording
        recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        //3. Set the output format with the extension of amr with the high quality NB (WB-low quality) for audio
        recorder.setOutputFormat(MediaRecorder.OutputFormat.AMR_NB);
        //4. Set the Encoder
        recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        //5. Set the source to store the file
        recorder.setOutputFile("/sdcard/test.amr");  }
```

# MainActivity.java

```
//Start recording by clicking the Start Button
public void start(View v){
   // prepare method need to surround with try catch
   try {
      recorder.prepare();
      tv.setText("Start Recording");
      recorder.start();
   } catch (IOException e) {
      e.printStackTrace();
   }
}
//Stop recording by clicking the Stop Button
public void  stop(View v){
   tv.setText("Stop Recording");
   recorder.stop();
}
```

# MainActivity.java

```
public void stplay(View v)  {
  tv.setText("Playing Audio");
     path = "/sdcard/test.amr";
     mp = new MediaPlayer();
     try {
        mp.setDataSource(path);
        mp.prepare();
        mp.start();
     } catch (IOException e) {
        e.printStackTrace();
     }
  }
  public void spplay(View v){
     tv.setText("Audio Stopped");
     mp.stop();
     mp.release();
  }
}
```

# Output Screen shots

1. START


2. STOP

# Output Screen shots

## 3. START PLAY


## 4. STOP PLAY

# Video Recording

- android.media.MediaRecorder class is used to record video in android.
- Step 1 : Create an object for the MediaRecorder.
  - MediaRecorder recorder = new MediaRecorder();
- To record video you have to set the following properties
  - Step 2 : Set the Audio source for the MediaRecorder
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
  - Step 3 : Set the Video source for the MediaRecorder using Camera

    recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

    Automatically record the video with .mp4 default format. So no need to specify the format.
  - Step 4 : Set the quality for the Video using CamcorderProfile class.

    CamcorderProfile profile = CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);

    Possible to set QUALITY_LOW, QUALITY_HIGH
  - Step 5 : Set the CamcorderProfile object t to your MediaRecorder object.

    recorder.setProfile(profile);

# Video Recording

- Step 6 : Set the output path where you want to store the recorded video file.

  recorder.setOutputFile("/file_path/filename.mp4");

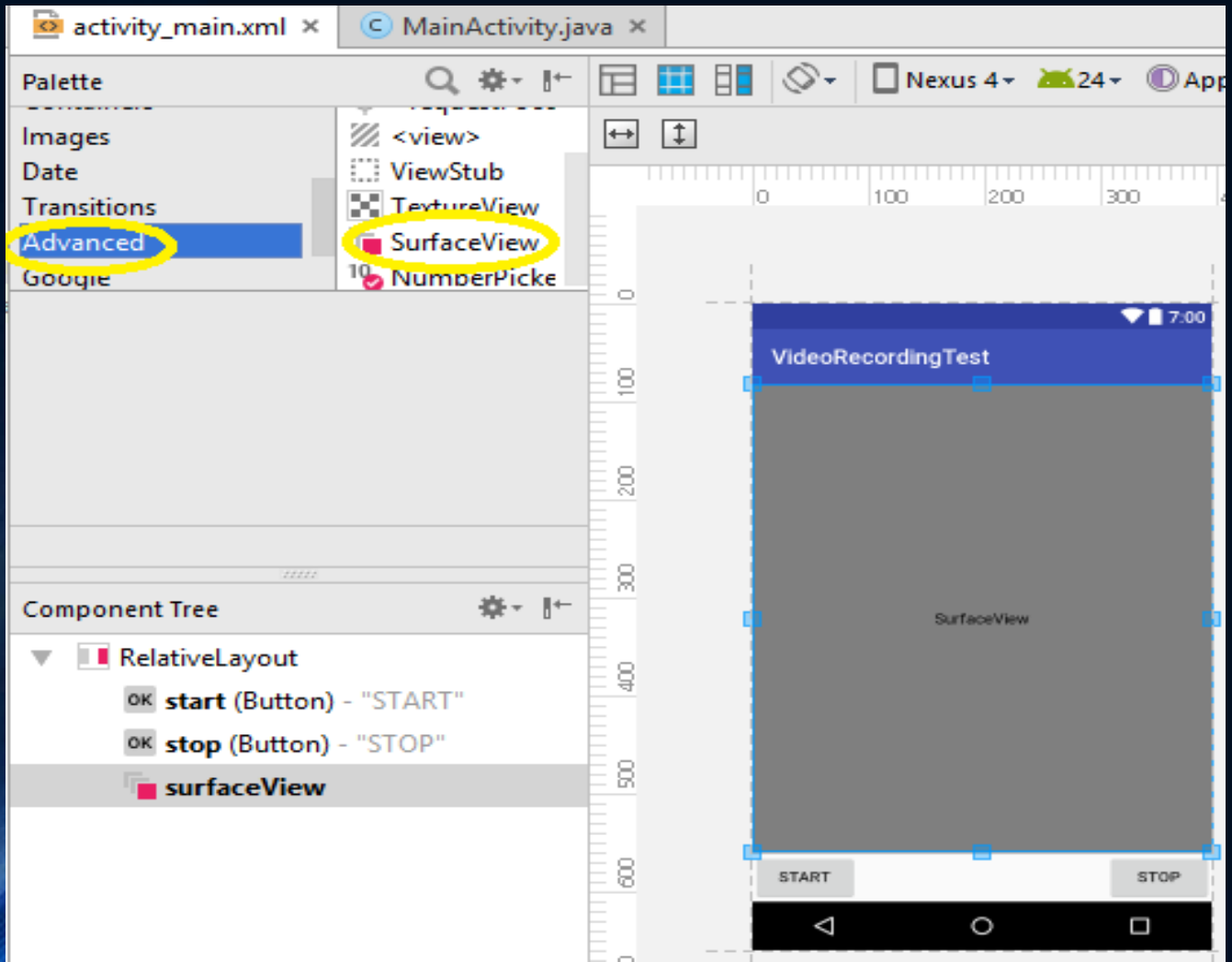
- We need to show the video preview of recording to the user with the help of advanced UI Component SurfaceView.
- Step7 : Configure the id for XML SurfaceView component. eg : sView.
- Directly we cannot perform any operations on SurfaceView. Can manage the SurfaceView with the help of SurfaceHolder object.
- Step 8 : Get the SurfaceHolder for sView. Access to the underlying surface is provided via the SurfaceHolder interface, which can be retrieved by calling getHolder().

  SurfaceHolder holder = sView.getHolder();

- All SurfaceView and SurfaceHolder.Callback methods will be called from the thread running the SurfaceView's window (typically the main thread of the application).
- Three methods or states from the SurfaceHolder.Callback
  - surfaceCreated(), surfaceDestroyed() and surfaceChanged()

# Hands on Example – Screen Design

- Design a screen using  two buttons for Start and stop recording. Add the Advance UI Component SurfaceView by clicking Advanced → SurfaceView.



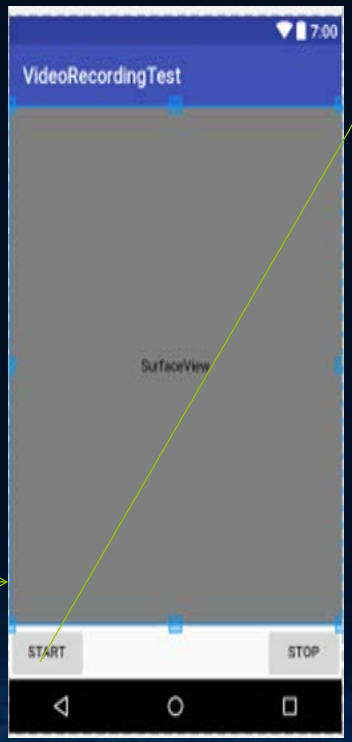
# activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="START"
        android:onClick="start"
        android:id="@+id/start"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
```

1

```
<SurfaceView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/surfaceView"
 android:layout_alignParentTop="true"

android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_above="@+id/start"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true" />
</RelativeLayout>
```

3




2

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="STOP"
    android:onClick="stop"
    android:id="@+id/stop"
  android:layout_alignParentBottom="true"
  android:layout_alignParentRight="true"
  android:layout_alignParentEnd="true" />
```

# MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    // Declare an Object for SurfaceView
    SurfaceView sView;
    // Declare an Object for SurfaceHolder to manage the SurfaceView
    SurfaceHolder sHolder;
    // Declare an object for MediaRecorder
    MediaRecorder recorder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
 // Create an object for the Recorder and set the Audio source, Video Source, Quality and Output file
        recorder = new MediaRecorder();
        recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
        CamcorderProfile profile =
         CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);
        recorder.setProfile(profile);
        recorder.setOutputFile("/storage/sdcard0/videotest.mp4");

        // Configure the Id from XML UI SurfaceView Component
        sView = (SurfaceView)findViewById(R.id.surfaceView);
        // get the Holder for the SurfaceView object
        sHolder = sView.getHolder();
```

# MainActivity.java

```
// Add a Callback interface for this holder.
    sHolder.addCallback(new SurfaceHolder.Callback(){
/* A client may implement this interface to receive information about changes to the surface.*/
     // This is called immediately after the surface is first created.
      @Override
      public void surfaceCreated(SurfaceHolder holder) {
        // Once the surface is created, displaying the video preview through holder
           recorder.setPreviewDisplay(holder.getSurface());
           try {
               recorder.prepare();
           } catch (IOException e) {
               e.printStackTrace();
          }
      }
     // This is called immediately after any structural changes (orientation or size)
         //have been made to the surface.
      @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height){
      }
        // This is called immediately before a surface is being destroyed.
      @Override
       public void surfaceDestroyed(SurfaceHolder holder) {
       }
  });
}
```

# MainActivity.java

```
public void start(View v){
    recorder.start();

  }
  public void stop(View v){
    recorder.stop();
  }
}
```

Demo : Andriod_Coding\Lesson8\VideoRecordTest

# Camera & Gallery

- The Android framework provides support for various cameras and camera features available on devices, allowing you to capture pictures and videos in your application.
- The Android framework supports capturing images and video through the Camera API or camera Intent.
- We will discuss Camera Intent and make use of Gallery.

# Capture a Picture

- Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
- To open a camera app and receive the resulting photo, use the ACTION_IMAGE_CAPTURE action. Also specify the URI location where you'd like the camera to save the photo, in the EXTRA_OUTPUT extra.
- ACTION_IMAGE_CAPTURE - Standard Intent action that can be sent to have the camera application capture an image and return it.
- EXTRA_OUTPUT - The name of the Intent-extra used to indicate a content resolver Uri to be used to store the requested image.

```
String path = "/storage/sdcardo/file_name.jpg";
File f = new File(path);
Intent.putExtra(MediaStore.EXTRA_OUTPUT,Uri.fromFile(f));
```

# Capture a Picture

startActivityForResult(intent_object,request_code);

You can start another activity and receive a result back. To receive a result, call startActivityForResult().

- An Intent that carries the result data. Specify the request_code >=0.
- Once the Camera activity is completed(ie after capturing the image), based on the request_code result you perform any post operations(image crop, edit image etc.,) in onActivityResult() method in your Activity class.

# Gallery

- Here we are using Intent to get Image Gallery
- Intent i = **new** Intent();
  *// Activity Action for the intent : Pick an item from the data, returning what was selected.*
  i.setAction(Intent.***ACTION_PICK***);
  i.setType(**"image/*"**);
  *// Start the Gallery Intent activity with the request_code 2*
  startActivityForResult(intent_object,request_code);

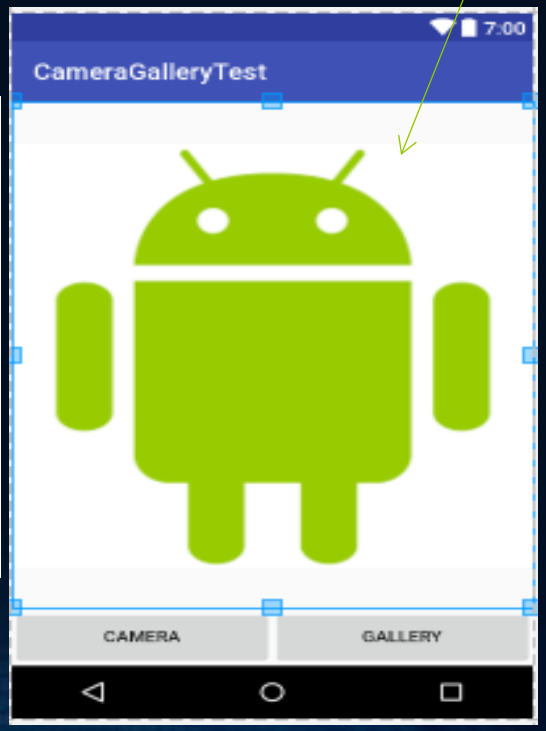
# Hands on Example – Camera & Gallery

- Problem : If click the Camera button to take a picture using device Camera and set the captures image to the ImageView Component. If you click the Gallery button choose the image from your device Photo Gallery and the selected image will be set in the ImageView Component.

# activity_main.xml

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ImageView
        android:layout_height="0dp"
        android:layout_weight="0.9"
        android:id="@+id/iv"
        android:src="@drawable/iandriod"
        android:layout_width="match_parent" />
```

1




```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.1"
    android:orientation="horizontal">
    <Button
        android:layout_width="0dp"
        android:layout_weight="0.5"
android:layout_height="match_parent"
        android:text="Camera"
        android:id="@+id/camera"
        android:onClick="camera"
        />
```

2

```
<Button
android:layout_width="0dp"
android:layout_weight="0.5"
android:layout_height=
"match_parent"

android:text="Gallery"
android:id="@+id/gallery"
android:onClick="gallery"/>
    </LinearLayout>
</LinearLayout>
```

3

# MainActivity.java

Add the given permission to the AndriodManifest.xml

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

```
public class MainActivity extends AppCompatActivity {
    String path = "/storage/sdcardo/test.jpg";
    File f;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

# MainActivity.java

```
public void camera(View v){
// Create a Camera Intent
    Intent i = new Intent("android.media.action.IMAGE_CAPTURE");
    // Create a file with the specified path
    f = new File(path);
    // Store the capture into SDCard storage
    i.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
    // Start the Camera Intent activity with the request_code 1
    startActivityForResult(i,1);
}

public void gallery(View v){
    Intent i = new Intent();
    // Activity Action for the intent : Pick an item from the data, returning what was selected.
    i.setAction(Intent.ACTION_PICK);
    i.setType("image/*");
    // Start the Gallery Intent activity with the request_code 2
    startActivityForResult(i,2);
}
```

# MainActivity.java

```
//To perform post Activities write your logic in the onActivityResult(),

// the user actions are determined based on the requestCode
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        // Intent object data automatically store the selected file path from the
// Image Gallery from your device storage
        super.onActivityResult(requestCode, resultCode, data);
        ImageView iView = (ImageView) findViewById(R.id.iv);
            if(requestCode==1){ // For Clicking Camera button
           // Set the captured image to the ImageView Component
            iView.setImageURI(Uri.fromFile(f));
        }
        else if(requestCode==2){ // For Clicking Gallery button
         // Set the selected image from the device image gallery to the ImageView component
            iView.setImageURI(data.getData());
        }
    }
}
```

Refer Complete Demo from : Lesson7-8\CameraGalleryTest