27

Android Multimedia

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html

UNLOCKING ANDROID - A DEVELOPER'S GUIIDE
W. FRANK ABLESON, CHARLIE COLLINS, ROBBI SEN
ISBN 978-1-933988-67-2
MANNING PUBLICATIONS, 2009

Multimedia

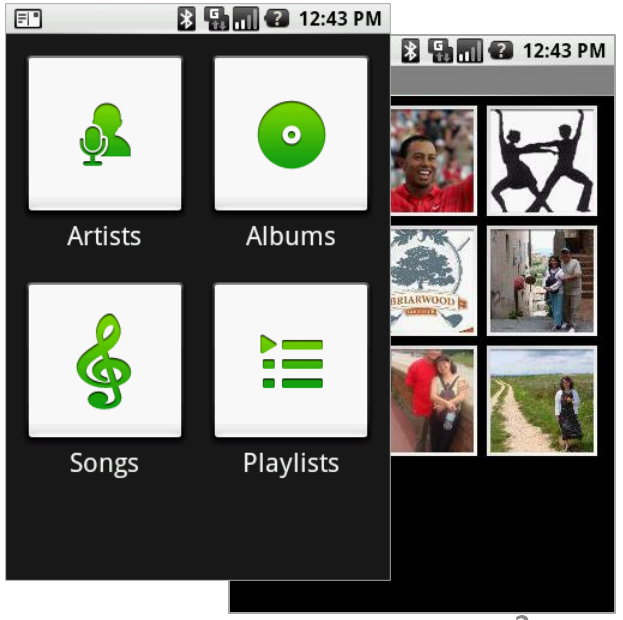
Android Audio and Video

The Android platform offers built-in encoding/decoding for a variety of common media types, so that you can easily integrate ***audio***, ***video***, and ***images*** into your applications.

Accessing the platform's media capabilities is fairly straightforward — you do so using the same intents and activities mechanism that the rest of Android uses.

You can play audio or video from:

1. media files stored in the application's resources (**raw resources**),
2. standalone files in the **file-system**, or
3. a data stream arriving over a **network connection**.



Reference: http://developer.android.com/guide/topics/media/index.html

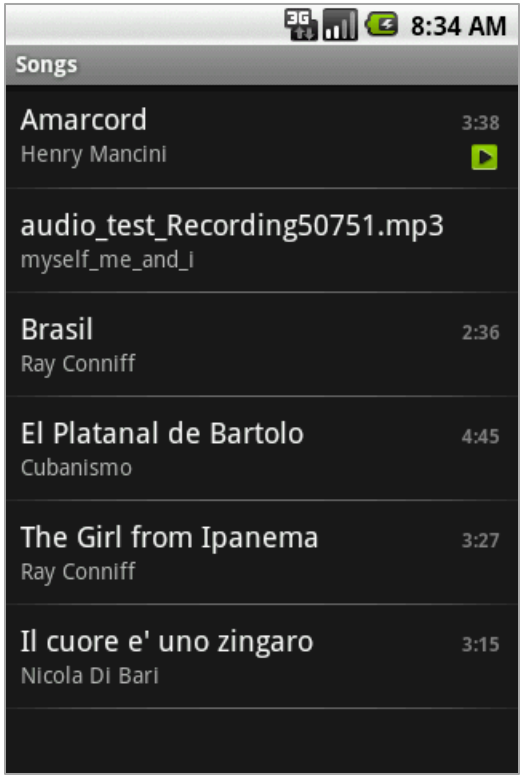
Multimedia

Using Built-in Actions to Examine the Media Store Database

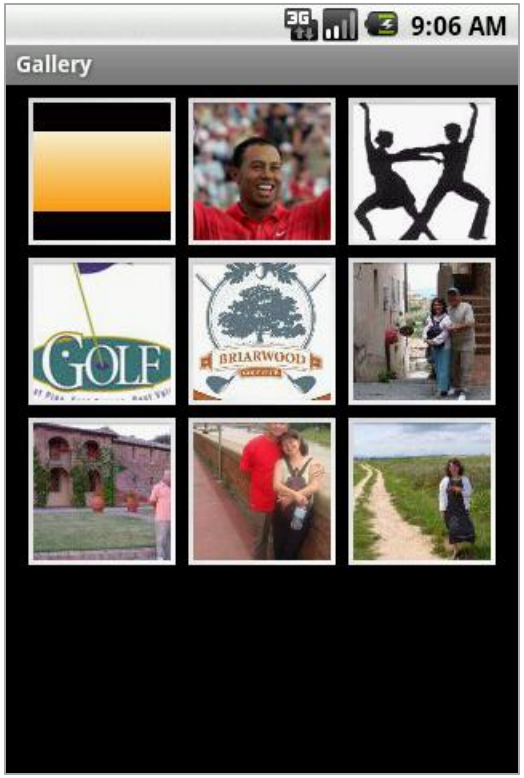
```
public class MediaPlayerActions extends Activity {
    TextView txtMsgBox;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsgBox = (TextView)findViewById(R.id.txtMsgBox);

        try {
            //---------------------------------------------------------------------
            Uri base = android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            Intent myIntent = new Intent(Intent.ACTION_VIEW, base);

            //---------------------------------------------------------------------
            // Uri base = android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
            // Intent myIntent = new Intent(Intent.ACTION_PICK, base);

            //---------------------------------------------------------------------
            // Uri base = android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            // Intent myIntent = new Intent(Intent.ACTION_VIEW, base);

            //---------------------------------------------------------------------
            startActivity(myIntent);

            finish();

        } catch (Exception e) {
            txtMsgBox.setText(e.getMessage());
        }

    }//onCreate
}
```

Multimedia

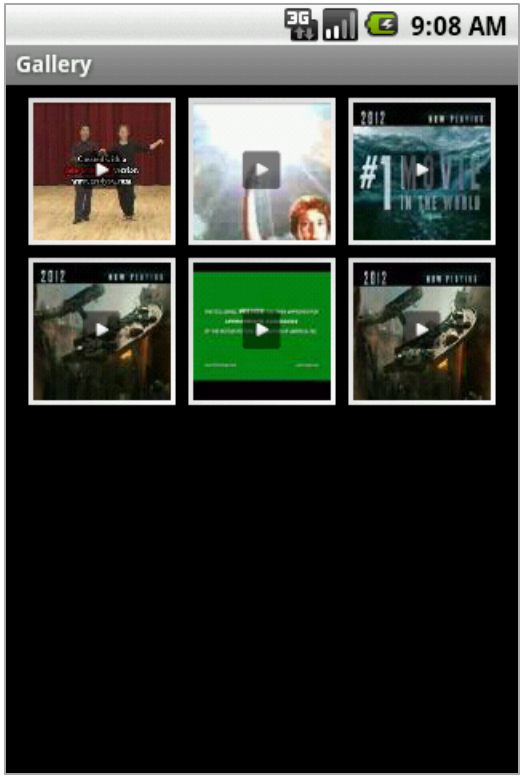
Using Built-in Actions to Examine the Media Store Database



MediaStore.Audio



MediaStore.Images



MediaStore.Video

Multimedia

Android Audio and Video

To **play** audio or video from your application, use the **MediaPlayer** class.

> *The platform also lets you record audio and video, where supported by the mobile device hardware.*

To **record** audio or video, use the **MediaRecorder** class.

> *Note that the emulator doesn't have hardware to capture audio or video, but actual mobile devices are likely to provide these capabilities, accessible through the MediaRecorder class.*

For a list of media formats for which Android offers built-in support, see Appendix A.

Reference: http://developer.android.com/guide/topics/media/index.html

Multimedia

MediaPlayer

Class Overview

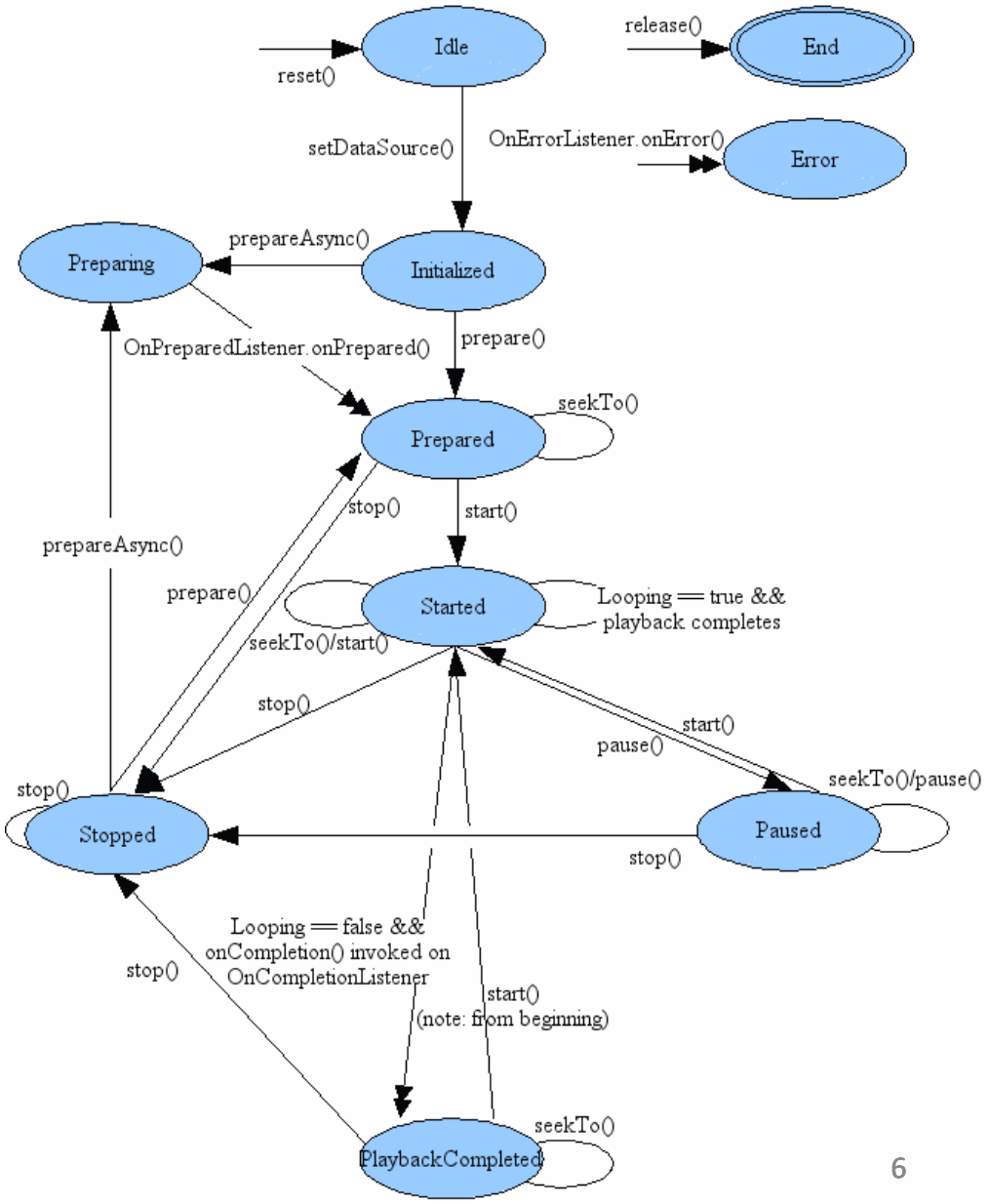
MediaPlayer class can be used to control playback of audio/video files and streams.

Playback control of audio/video files and streams is managed as a state machine.

The **ovals** represent the **states** a MediaPlayer object may reside in.

The **arcs** represent the playback **control** operations that drive the object state transition.

There are two types of arcs. The arcs with a **single arrow** head represent *synchronous* method calls, while those with a **double arrow** head represent *asynchronous* method calls.



Multimedia

MediaPlayer – Useful Methods

Public Methods	
static MediaPlayer	create(Context context, Uri uri) Convenience method to create a MediaPlayer for a given Uri.
static MediaPlayer	create(Context context, int resid) Convenience method to create a MediaPlayer for a given resource id.
int	getCurrentPosition() Gets the current playback position.
int	getDuration() Gets the duration of the file.
int	getVideoHeight() Returns the height of the video.
int	getVideoWidth() Returns the width of the video.
boolean	isLooping() Checks whether the MediaPlayer is looping or non-looping.
boolean	isPlaying() Checks whether the MediaPlayer is playing.
void	pause() Pauses playback.
void	prepare() Prepares the player for playback, synchronously.
void	prepareAsync() Prepares the player for playback, asynchronously.
void	release() Releases resources associated with this MediaPlayer object.
void	reset() Resets the MediaPlayer to its uninitialized state.
void	seekTo(int msec) Seeks to specified time position.

Public Methods	
void	setAudioStreamType(int streamtype) Sets the audio stream type for this MediaPlayer.
void	setDataSource(String path) Sets the data source (file-path or http/rtsp URL) to use.
void	setDataSource(FileDescriptor fd, long offset, long length) Sets the data source (FileDescriptor) to use.
void	setDataSource(FileDescriptor fd) Sets the data source (FileDescriptor) to use.
void	setDataSource(Context context, Uri uri) Sets the data source as a content Uri.
void	setLooping(boolean looping) Sets the player to be looping or non-looping.
void	setOnCompletionListener(MediaPlayer.OnCompletionListener listener) Register a callback to be invoked when the end of a media source has been reached during playback.
void	setVolume(float leftVolume, float rightVolume) Sets the volume on this player.
void	setWakeMode(Context context, int mode) Set the low-level power management behavior for this MediaPlayer.
void	start() Starts or resumes playback.
void	stop() Stops playback after playback has been stopped or paused.

Multimedia

MediaPlayer – Playing Audio from a Raw Resource

1. Put the sound (or other media resource) file into the **res/raw** folder of your project, where the Eclipse plugin will find it and make it into a resource that can be referenced from your R class

2. Create an instance of **MediaPlayer**, referencing that resource using the method: MediaPlayer.create, and then call start() on the instance:

```
MediaPlayer mp = MediaPlayer.create(context, R.raw.sound_file_1);
mp.start();
```

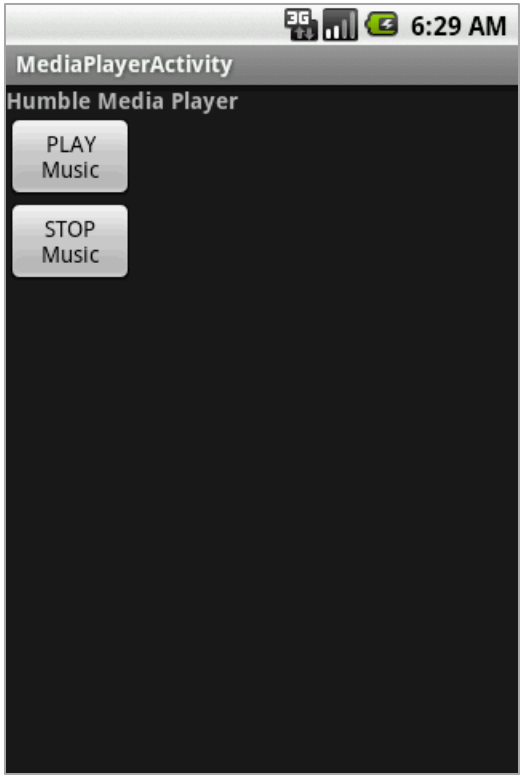
3. To stop playback, call stop(). If you wish to later replay the media, then you must reset() and prepare() the MediaPlayer object before calling start() again. (create() calls prepare() the first time.)

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

Multimedia

Example 1. MediaPlayer – Playing Audio from a Raw Resource

```
<?xml 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"
   >
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Humble Media Player"
   android:textStyle="bold"
   />

<Button android:id="@+id/playsong"
   android:layout_width="80px"
   android:layout_height="wrap_content"
   android:text="PLAY Music"
   />

<Button android:id="@+id/stopsong"
   android:layout_width="80px"
   android:layout_height="wrap_content"
   android:text="STOP Music"
   />
</LinearLayout>
```



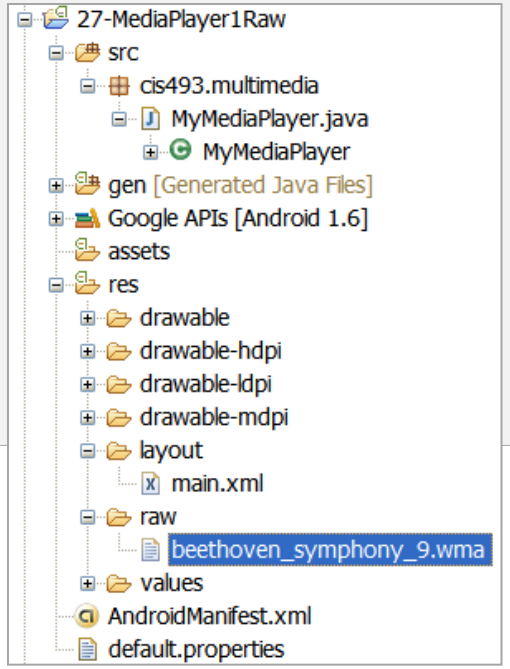
Manifest does not require special permissions.

Multimedia

Example 1. MediaPlayer – Playing Audio from a Raw Resource

```
//MyMediaPlayer Demo1: plays a song saved as a RAW resource
//-----------------------------------------------------------------------
package cis493.multimedia;

import android.app.Activity;
import android.os.Bundle;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;


public class MyMediaPlayer extends Activity {

MediaPlayer mp;

public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);
```

Push a music file (mp3, wma) into the **res/raw** folder



Note: We use the music file: *Beethoven's Symphony No 9 (Scherzo).wma* included in the Windows folder: **C:\Documents and Settings\All Users\Documents\My Music\Sample Music**
(rename it using lower case symbols!!!)

Multimedia

Example 1. MediaPlayer – Playing Audio from a Raw Resource

```
  Button myPlayButton = (Button) findViewById(R.id.playsong);
  myPlayButton.setOnClickListener(new Button.OnClickListener() {
  public void onClick(View v) {
  try {
        mp = MediaPlayer.create(MyMediaPlayer.this, R.raw.beethoven_symphony_9);
        mp.start();

        mp.setOnCompletionListener(new OnCompletionListener() {
          public void onCompletion(MediaPlayer arg0) {
              Toast.makeText(getApplicationContext(),"Bravo! Bravo!", 1).show();
          }
        });
      } catch (Exception e) {
        e.printStackTrace();
      }
      }
  });// myPlayButton

  Button myStopButton = (Button) findViewById(R.id.stopsong);
  myStopButton.setOnClickListener(new Button.OnClickListener() {
      public void onClick(View v) {
          if (mp.isPlaying()) {
            mp.stop();
          }
      }// onClick
  }); // myStopButton
 }// onCreate
}
```

Multimedia

MediaPlayer - Playing Audio from a File or Stream

You can play back media files from the files-system or a web URL:

1. Create an instance of the **MediaPlayer** using **new** constructor

2. Call **setDataSource()** with a String containing the path (local filesystem or URL) to the file you want to play

3. First prepare() then start() on the instance:

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

4. Commands stop() and pause() work the same as discussed earlier.

Note:

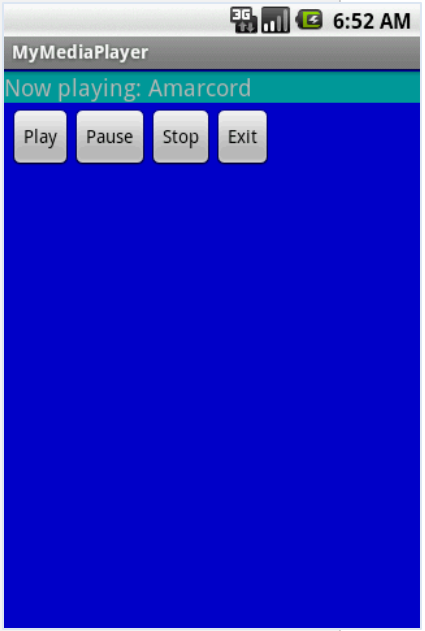
It is possible that **mp** could be ***null***, so good code should null check after the new. Also, **IllegalArgumentException** and **IOException** either need to be caught or passed on when using **setDataSource**(), since the file you are referencing may not exist.

Multimedia

Example 2. MediaPlayer - Playing Audio from a File or Stream

Quite similar to previous example. Place the music file (mp3, wma,…) in the SD card.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/myLinearLayout1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff0000cc"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/txtTitle"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff009999"
android:layout_marginTop="2px"
android:text="TITLE"
android:textSize="18sp" />
<LinearLayout
android:id="@+id/myLinearLayout2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:padding="4px"
>
<Button
android:id="@+id/btnPlay"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Play" />

<Button
android:id="@+id/btnPause"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Pause" />

<Button
android:id="@+id/btnStop"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Stop" />

<Button
android:id="@+id/btnExit"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Exit" />

</LinearLayout>

</LinearLayout>
```



Multimedia

Example 2. MediaPlayer - Playing Audio from a File or Stream

```
// MyMediaPlayer: plays an audio file stored in the SDcard
// ---------------------------------------------------------------------
package cis493.multimedia;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

//A very simple mp3 player. Music stored in the SD card
public class MyMediaPlayer extends Activity {

MediaPlayer mp;
    Button btnPlay;
    Button btnPause;
    Button btnStop;
    Button btnExit;
    TextView txtTitle;

@Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        txtTitle = (TextView)findViewById(R.id.txtTitle);
```

Multimedia

Example 2. MediaPlayer - Playing Audio from a File or Stream

```
try {
    //for now it plays only one title
    txtTitle.setText("Now playing: Amarcord");

    mp = new MediaPlayer();
    mp.setDataSource( "/sdcard/Amarcord.mp3" );

    mp.prepare();
    mp.start();

} catch (Exception e) {
    txtTitle.setText(e.getMessage());
}//try


//exit: stop song and exit mp3 player
btnExit = (Button)findViewById(R.id.btnExit);
btnExit.setOnClickListener(new ExitClicker() );
```

Multimedia

Example 2. MediaPlayer - Playing Audio from a File or Stream

```
//pause current song
Button btnPause=(Button)findViewById(R.id.btnPause);
btnPause.setOnClickListener(new Button.OnClickListener() {
	public void onClick(View v) {
		mp.pause();
	}
	});

//resume playing stopped music
Button btnPlay = (Button)findViewById(R.id.btnPlay);
btnPlay.setOnClickListener(new Button.OnClickListener() {
	public void onClick(View v) {
		mp.start();
	}
	});

//stop playing current music - do not exit
Button btnStop = (Button)findViewById(R.id.btnStop);
btnStop.setOnClickListener(new Button.OnClickListener() {
	public void onClick(View v) {
		mp.stop();
	}
	});

}//onCreate

private class ExitClicker implements OnClickListener {
		public void onClick (View v) {
		//TODO: create a notification with a reference to us
				mp.stop();
				finish();
		}//onClick
  }//PlayClicker
}//MyMp3Player
```

Multimedia

MediaPlayer - Audio Recording

Follow the steps below to implement *audio capture* from the device.

1. Create a new instance of android.media.MediaRecorder using **new**
2. Create a new instance of android.content.ContentValues and put in some standard properties like TITLE, TIMESTAMP, and the all important MIME_TYPE
3. Create a file path for the data to go to
4. Set the audio source using MediaRecorder.setAudioSource(). You will probably want to use MediaRecorder.AudioSource.MIC
5. Set output file format using MediaRecorder.setOutputFormat()
6. Set the audio encoder using MediaRecorder.setAudioEncoder()
7. Call prepare() on the *MediaRecorder* instance.
8. To start audio capture, call start().
9. To stop audio capture, call stop().
10. When you are done with the *MediaRecorder* instance, call release() on it.

Multimedia

MediaPlayer - Audio Recording Setup and Start

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

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

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

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

Multimedia

MediaPlayer – Audio Recording

Stop Recording

Based on the example above, here's how you should **stop** audio capture.

```
recorder.stop();
recorder.release();
```

Multimedia

MediaPlayer - Audio Recording – MediaStore Database

A call to the method:

```
sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
```

Request the *media scanner* to scan a file (pointed by newUri) and add it to the **MediaStore** database. The absolute path to the file is contained in the ***Intent.mData*** field.

DATA	The data stream location for the file Type: DATA STREAM
DATE_ADDED	The time the file was added to the media provider Units are seconds since 1970.
DATE_MODIFIED	The time the file was last modified Units are seconds since 1970.
DISPLAY_NAME	The display name of the file Type: TEXT
MIME_TYPE	The MIME type of the file Type: TEXT
SIZE	The size of the file in bytes Type: INTEGER (long)
TITLE	The title of the content Type: TEXT

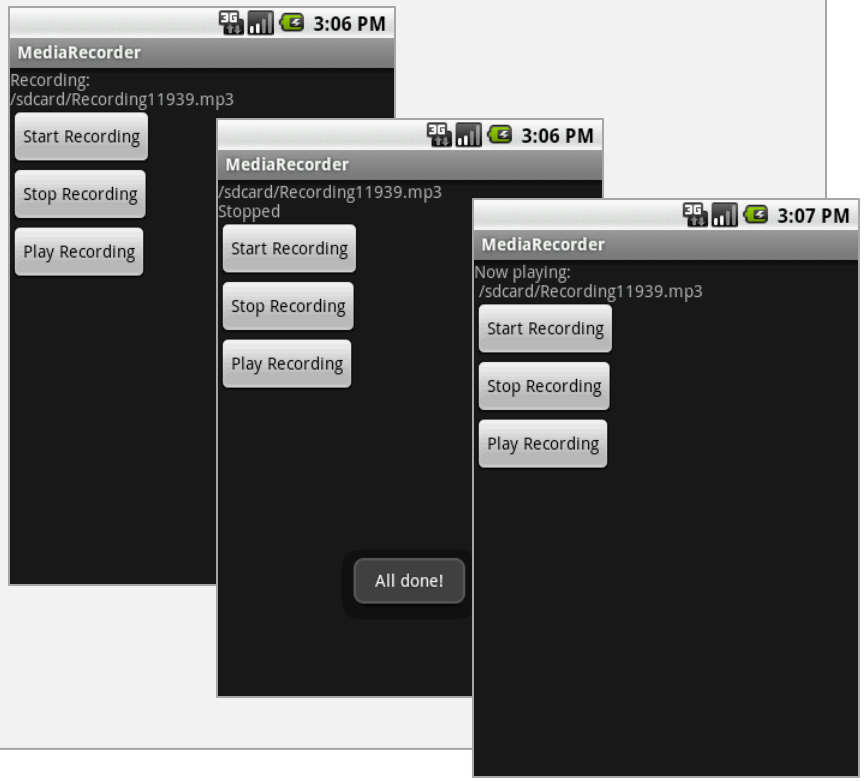
Other fields include: Artist, Album, Album Art, …

Multimedia

Example 3. MediaPlayer – Audio Recording

The following example illustrates how to setup, record, and play an AUDIO file. It also adds the newly created file to the audio portion of the Android's **MediaStore** ContentProvider.

```
<?xml 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">
  <TextView android:layout_width="fill_parent"
      android:id="@+id/txtMsg"
      android:layout_height="wrap_content"
      android:text="Media Recorder - AUDIO" />

  <Button  android:text="Start Recording"
           android:id="@+id/startRecording"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />
  <Button  android:text="Stop Recording"
           android:id="@+id/stopRecording"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content />
  <Button  android:text="Play Recording"
           android:id="@+id/playRecording"
           android:layout_width="wrap_content"
           android:layout_height="wrap_contentn />
</LinearLayout>
```

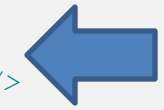


Multimedia

Example 3. MediaPlayer – Audio Recording

The following example illustrates how to setup, record, and play an AUDIO file. It also adds the newly created file to the audio portion of the Android's **MediaStore** ContentProvider.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.multimedia"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <activity android:name=".MyAudioRecorder"
                  android:label="@string/app_name">
            <uses-permission android:name="android.permission.RECORD_AUDIO" />
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

    <uses-sdk android:minSdkVersion="4" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

</manifest>
```



Multimedia

Example 3. MediaPlayer – Audio Recording

The following example illustrates how to setup, record, and play an AUDIO file. It also adds the newly created file to the audio portion of the Android's **MediaStore** ContentProvider.

```
//record and play an audio file - add it to the MediaStore database
package cis493.multimedia;

import java.io.File;
import java.io.IOException;
import android.app.Activity;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Intent;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.media.MediaPlayer.OnCompletionListener;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.provider.MediaStore.MediaColumns;
import android.provider.MediaStore.Audio.AudioColumns;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
```

Multimedia

Example 3. MediaPlayer – Audio Recording

```
public class MyAudioRecorder extends Activity {
MediaRecorder myRecorder;
File mSampleFile = null;
TextView txtMsg;

static final String SAMPLE_PREFIX = "Recording";
static final String SAMPLE_EXTENSION = ".mp3";
private static final String TAG = "<<MyAudioRecorder>>";

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

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

          startRecording();
      }
  });
```

Multimedia

Example 3. MediaPlayer – Audio Recording

```
Button stopRecording = (Button) findViewById(R.id.stopRecording);

stopRecording.setOnClickListener(new View.OnClickListener() {

  public void onClick(View v) {

        stopRecording();

        addToMediaStoreDB();
   }

});
```

Multimedia

Example 3. MediaPlayer – Audio Recording

```
Button playRecording = (Button) findViewById(R.id.playRecording);
playRecording.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
try {
      String recordingName = mSampleFile.getAbsolutePath();
      txtMsg.setText("Now playing:\n " + recordingName);
      MediaPlayer mp = new MediaPlayer();
      mp.setDataSource( recordingName );

      mp.prepare();
      mp.start();

      mp.setOnCompletionListener(new OnCompletionListener() {
            public void onCompletion(MediaPlayer arg0) {
                  txtMsg.setText(mSampleFile.getAbsolutePath() + "\nThat's all - Adios.....");
            }

      });


      } catch (IllegalArgumentException e) {
            txtMsg.setText(e.getMessage());
      } catch (IllegalStateException e) {
            txtMsg.setText(e.getMessage());
      } catch (IOException e) {
            txtMsg.setText(e.getMessage());
      }
      }
});
```

Multimedia

Example 3. MediaPlayer – Audio Recording

```
protected void startRecording() {
try {
      Log.e(TAG, "recording begins...");
      if (this.mSampleFile == null) {
            File sampleDir = Environment.getExternalStorageDirectory();
            try {
                  this.mSampleFile = File.createTempFile(
                  MyAudioRecorder.SAMPLE_PREFIX,
                  MyAudioRecorder.SAMPLE_EXTENSION, sampleDir);
            } catch (IOException e) {
                  Log.e(TAG, "sdcard access error: " + e.getMessage());
                  return;
            }
      }//if
      txtMsg.setText("Recording: \n" + mSampleFile.getCanonicalPath());
      myRecorder = new MediaRecorder();
      myRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
      myRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
      myRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
      myRecorder.setOutputFile(this.mSampleFile.getAbsolutePath());

      myRecorder.prepare();
      myRecorder.start();

} catch (IllegalStateException e) {
      e.printStackTrace();
} catch (IOException e) {
       e.printStackTrace();
}
}//startRecording
```

Multimedia

Example 3. MediaPlayer – Audio Recording

```
protected void stopRecording() {
     try {

          myRecorder.stop();
          myRecorder.release();

          txtMsg.setText(mSampleFile.getAbsolutePath() + "\nStopped");
          Toast.makeText(this,"All done!", 1).show();

     } catch (IllegalStateException e) {
           Toast.makeText(this,"Troubles: " + e.getMessage(), 1).show();
     }

}//stopRecording
```

Multimedia

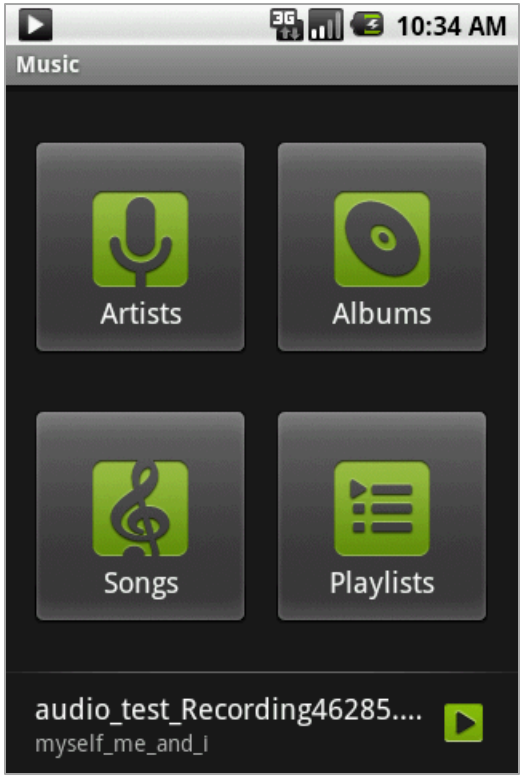
Example 3. MediaPlayer – Audio Recording

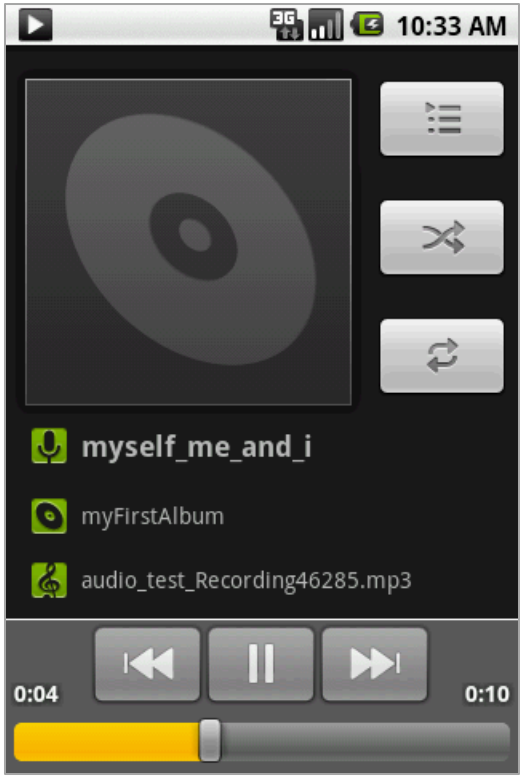
```
  protected void addToMediaStoreDB() {

     try {
          long now = System.currentTimeMillis();
          ContentValues newValues = new ContentValues(6);
          newValues.put(MediaColumns.TITLE, "test_" + mSampleFile.getName());
          newValues.put(MediaColumns.DATE_ADDED, (int) (now / 1000));
          newValues.put(MediaColumns.MIME_TYPE, "audio/mpeg");
          newValues.put(AudioColumns.IS_MUSIC, true);
          newValues.put(AudioColumns.ARTIST, "myself");
          newValues.put(MediaColumns.DATA, mSampleFile.getAbsolutePath());

          ContentResolver contentResolver = getContentResolver();
          Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;

          Uri newUri = contentResolver.insert(base, newValues);

          sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));

          Toast.makeText(this, "Added to MediaStore:\n "
                         + mSampleFile.getAbsolutePath(), 1).show();
     } catch (Exception e) {
          Toast.makeText(this, "Error MediaStore\n" + e.getMessage(), 1).show();
     }
  }//addToMediaStoreDB

} //class
```

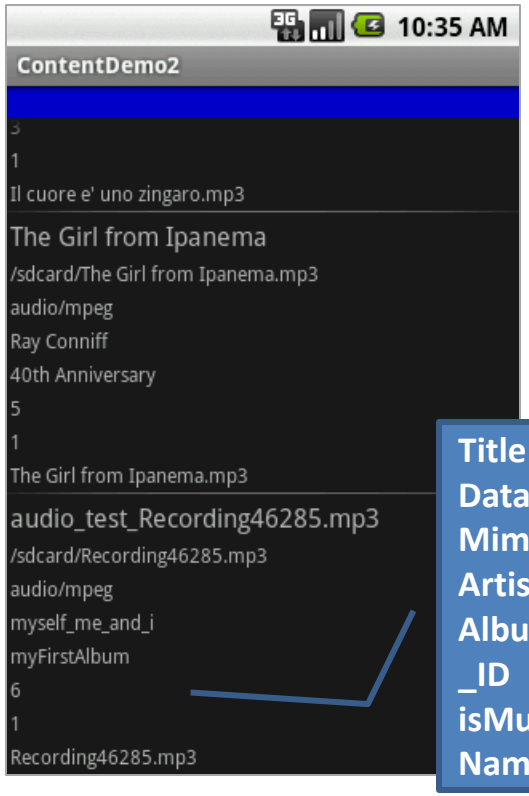
Multimedia

Example 3. MediaPlayer – Audio Recording

Using the apps: *MusicPlayer* & *ContentDemo2* (see chapter 20) to spy on the **MediaStore.Audio** database.







Title
Data
Mime
Artist
Album
_ID
isMusic
Name_Display

Multimedia

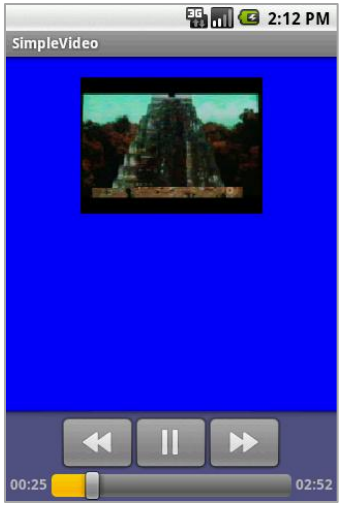
VideoView

Class Overview

Displays a video file. The VideoView class can load images from various sources (such as resources or content providers), takes care of computing its measurement from the video so that it can be used in any layout manager, and provides various display options such as *scaling* and *tinting*.

Example

```
VideoView myVideo = (VideoView) findViewById(R.id.video);

myVideo.setVideoPath("/sdcard/movie2012c.3gp");

MediaController mc = new MediaController(this);

mc.setMediaPlayer(myVideo);

myVideo.setMediaController(mc);

myVideo.requestFocus();

mc.show();
```

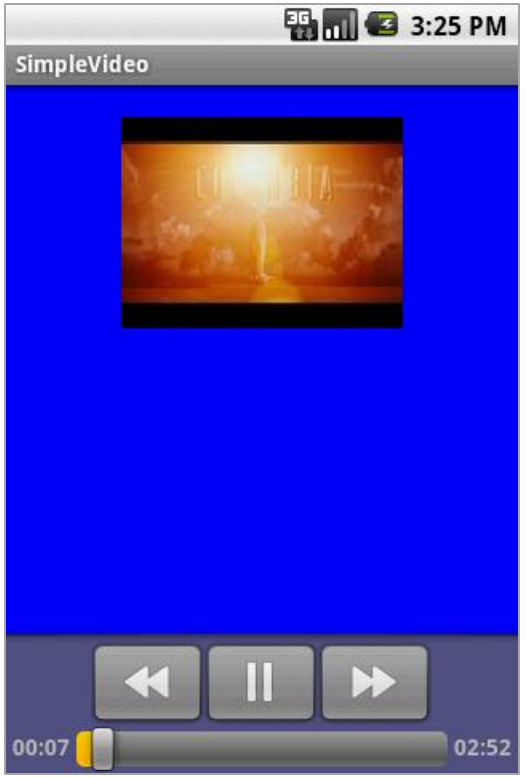


Multimedia

Example4. MediaPlayer – Video Playing

For this example we will place in the SDCARD a **.3gp** or **.mp4** encoded video. The application will play the digital video inside an Android *VideoView* widget. A *control bar* is displayed each time the user touches the screen (it disappears after a few seconds). The video shown here was downloaded from *YouTube.com* and converted to 3gp using a typical Video-Converter tool downloaded from the Internet (we used 3GP, 1500 Kbps, 30 fps, 176x132 pixels). For your convenience two sample videos are included in the /res/raw folder.

```
<?xml 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"
     android:background="#ff0000ff"  >

<VideoView android:id="@+id/video"
     android:layout_width="176px"
     android:layout_height="132px"
     android:layout_marginTop="20px"
     android:layout_gravity="center"
/>

</LinearLayout>
```



Multimedia

Example4. MediaPlayer – Video Playing

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.multimedia"
      android:versionCode="1"
      android:versionName="1.0">

    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        <activity android:name=".SimpleVideo"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-sdk android:minSdkVersion="4" />

</manifest>
```

Multimedia

Example4. MediaPlayer – Video Playing

```
//shows a 3gp digital video inside a VideoView widget
package cis493.multimedia;
import android.app.Activity;
import android.graphics.PixelFormat;
import android.os.Bundle;
import android.widget.VideoView;
import android.widget.MediaController;

public class SimpleVideo extends Activity {
     private VideoView myVideo;
     private MediaController mc;
     @Override
     public void onCreate(Bundle icicle) {
          super.onCreate(icicle);
          getWindow().setFormat(PixelFormat.TRANSLUCENT);
          setContentView(R.layout.main);
          //format: 3GP, MPEG-4 codec, 30fps, 1500Kbps, 172x132 frames
          VideoView myVideo = (VideoView) findViewById(R.id.video);
          //myVideo.setVideoPath("/sdcard/v2012a.3gp"); //short video
          myVideo.setVideoPath("/sdcard/v2012c.3gp");//longer video

          MediaController mc = new MediaController(this);
          mc.setMediaPlayer(myVideo);

          myVideo.setMediaController(mc);
          myVideo.requestFocus();
          mc.show();
     }// onCreate
}
```

Multimedia

Using the Camera & GaleryView

- The Camera class is used to *connect/disconnect* with the camera service, set *capture* settings, *start/stop*, *preview*, *snap a picture*, and *retrieve* frames for encoding for video.
- There is no default constructor for this class. Use **open()** to get a Camera object.
- In order to use the device camera, you must declare the **CAMERA** permission in your Android Manifest.
- Also be sure to include the **<uses-feature>** manifest element in order to declare camera features used by your application.

Referece: http://2009.hfoss.org/Tutorial:Camera_and_Gallery_Demo and http://developer.android.com/reference/android/hardware/Camera.html

Multimedia

Using the Camera & GaleryView

Example. If you use the *camera* and *auto-focus* feature, your Manifest should include the following:

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

<uses-feature android:name="android.hardware.camera" />

<uses-feature android:name="android.hardware.camera.autofocus" />
```

Caution: *Different Android-powered devices may have different hardware specifications, such as megapixel ratings and auto-focus capabilities. In order for your application to be compatible with more devices, you should not make assumptions about the device camera specifications.*

Referece: http://2009.hfoss.org/Tutorial:Camera_and_Gallery_Demo and http://developer.android.com/reference/android/hardware/Camera.html

Multimedia

Camera Class – Important Methods

Public Methods	
final void	autoFocus(Camera.AutoFocusCallback cb) Starts auto-focus function and registers a callback function to run when camera is focused.
final void	cancelAutoFocus() Cancels auto-focus function.
Camera.Parameters	getParameters() Returns the picture Parameters for this Camera service.
final void	lock() Lock the camera to prevent other processes from accessing it.
static Camera	open() Returns a new Camera object.
final void	release() Disconnects and releases the Camera object resources.
final void	setErrorCallback(Camera.ErrorCallback cb) Registers a callback to be invoked when an error occurs.
final void	setOneShotPreviewCallback(Camera.PreviewCallback cb) Installs a callback to retrieve a single preview frame, after which the callback is cleared.

Multimedia

Camera Class – Important Methods

Public Methods	
void	setParameters(Camera.Parameters params) Sets the Parameters for pictures from this Camera service.
final void	setPreviewCallback(Camera.PreviewCallback cb) Can be called at any time to instruct the camera to use a callback for each preview frame in addition to displaying it.
final void	setPreviewDisplay(SurfaceHolder holder) Sets the SurfaceHolder to be used for a picture preview.
final void	startPreview() Start drawing preview frames to the surface.
final void	stopPreview() Stop drawing preview frames to the surface.
final void	takePicture (Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback postview, Camera.PictureCallback jpeg) Triggers an asynchronous image capture.
final void	takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg) Triggers an asynchronous image capture.
final void	unlock() Unlock the camera to allow another process to access it.
void	finalize() Is called before the object's memory is being reclaimed by the VM.

Multimedia

Camera Class – The *takePicture* Method

```
public final void takePicture ( Camera.ShutterCallback  shutter,
                                Camera.PictureCallback  raw,
                                Camera.PictureCallback  jpeg )
```

Triggers an asynchronous image capture. The camera service will initiate a series of *callbacks* to the application as the image capture progresses.

1. The *shutter callback* occurs after *the image is captured*. This can be used to trigger a sound to let the user know that image has been captured.
2. The *raw callback* occurs when *the raw image data is available* (NOTE: the data may be null if the hardware does not have enough memory to make a copy).
3. The *jpeg callback* occurs when the *compressed image is available*. If the application does not need a particular callback, a null can be passed instead of a callback method.

Parameters

shutter	callback after the image is captured, may be null
raw	callback with raw image data, may be null
jpeg	callback with jpeg image data, may be null

Multimedia

Camera Class – The built-in ACTION_IMAGE_CAPTURE Method

```
Intent myIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

Optional
myIntent.putExtra(
        MediaStore.EXTRA_OUTPUT,
        MediaStore.Images.Media.EXTERNAL_CONTENT_URI.toString()
        );

startActivityForResult(mIntent, CAMERA_ACTIVITY_MY_REQUEST_CODE);
```

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

The caller may pass an extra EXTRA_OUTPUT to control where this image will be written.

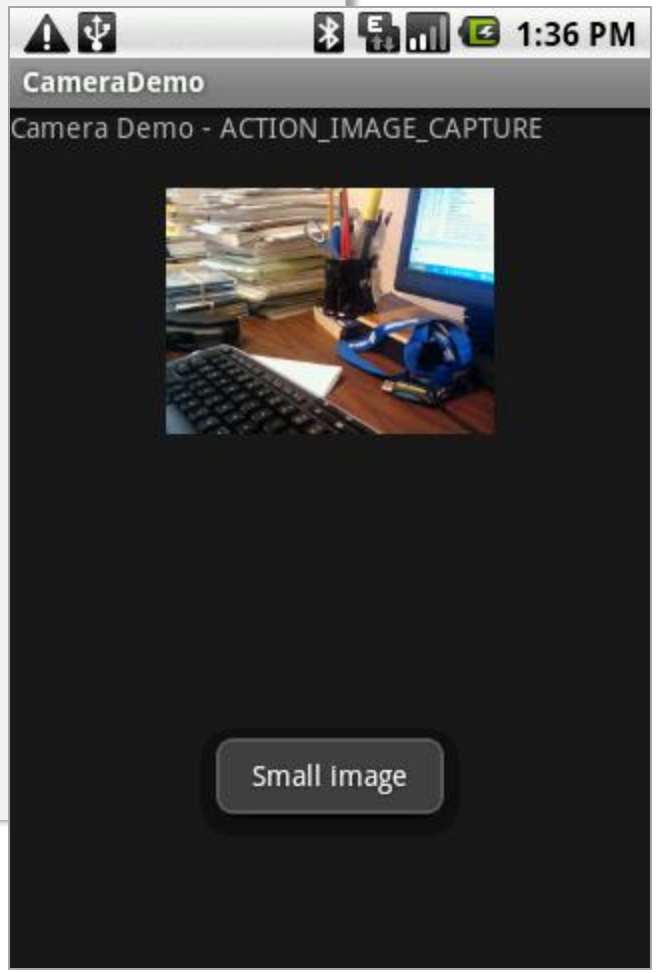
- If the EXTRA_OUTPUT is not present, then a small sized image is returned as a Bitmap object in the extra field. This is useful for applications that only need a small image.
- If the EXTRA_OUTPUT is present, then the full-sized image will be written to the Uri value of EXTRA_OUTPUT.

Reference: http://developer.android.com/reference/android/provider/MediaStore.html

Multimedia

Example5. Using the Camera with Built-in Action

```
<?xml 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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<ImageView android:id="@+id/mImageView"
            android:adjustViewBounds="true"
            android:layout_width="164px"
            android:layout_height="164px"
            android:layout_gravity="right"
             />
</LinearLayout>
```



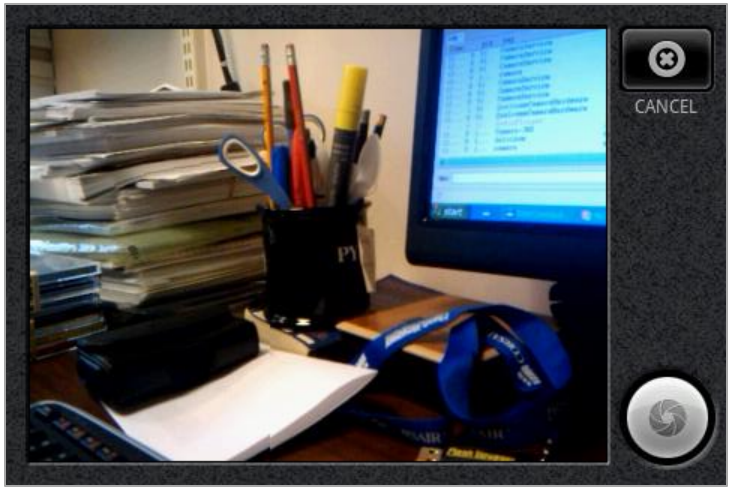
Multimedia

Example5. Using the Camera with Built-in Action

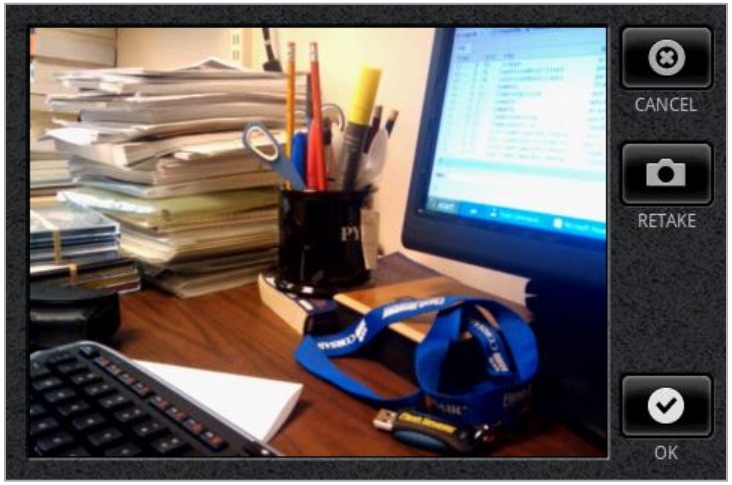
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.multimedia"
      android:versionCode="1"
      android:versionName="1.0">
    <application   android:icon="@drawable/icon"
                   android:label="@string/app_name">
        <activity  android:name=".CameraDemo"
                   android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission android:name="android.permission.CAMERA" />

</manifest>
```

Multimedia

Example5. Using the Camera with Built-in Action

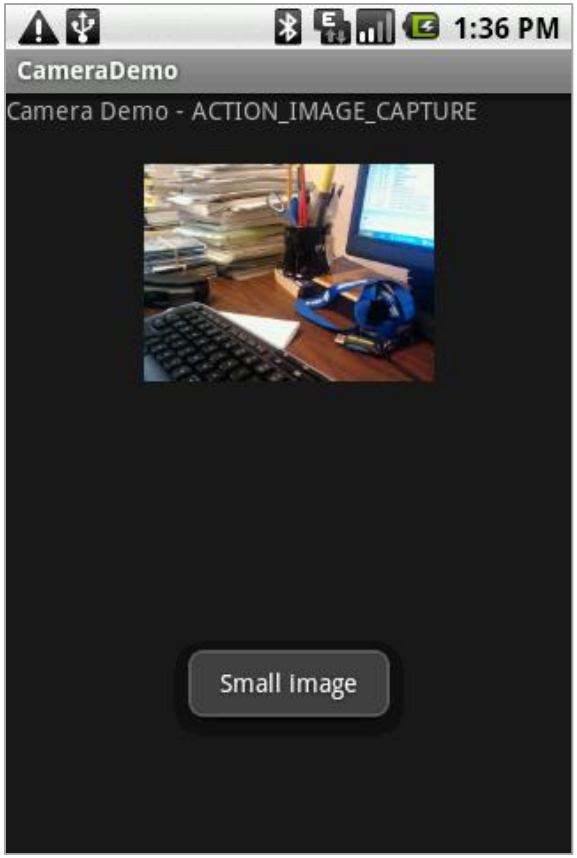


1. Previewing



2. After shutter is pressed

These views are Provided by the Built-in ACTION_IMAGE_CAPTURE *Intent.*



3. Image transferred from CAMERA to the application

Multimedia

Example5. Using the Camera with Built-in Action

```
// simple CAMERA app. using built-in ACTION_IMAGE_CAPTURE
package cis493.multimedia;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.widget.ImageView;
import android.widget.Toast;

public class CameraDemo extends Activity {
  final int CAMERA_ACTIVITY_REQUEST_CODE = 101;
  ImageView mImageView;

  @Override
  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mImageView = (ImageView)findViewById(R.id.mImageView);
```

Multimedia

Example5. Using the Camera with Built-in Action

```
Intent mIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

// use this in case you want 'big' pictures saved in the MediaStore
// Warning: this feature is NOT WORKING as in SDK 2.0
mIntent.putExtra(
                MediaStore.EXTRA_OUTPUT,
                MediaStore.Images.Media.EXTERNAL_CONTENT_URI.toString()
                );

startActivityForResult(mIntent, CAMERA_ACTIVITY_REQUEST_CODE);

}//onCreate


private void showToast(Context context, String text){
    Toast.makeText(context, text, 1).show();
}
```

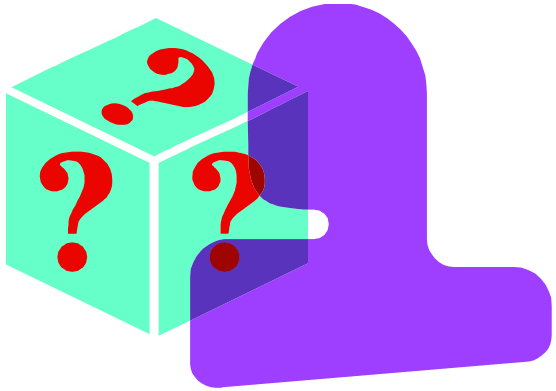
Multimedia

Example5. Using the Camera with Built-in Action

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
super.onActivityResult(requestCode, resultCode, intent);

     if (resultCode == RESULT_CANCELED) {
        showToast(this,"Activity cancelled");
        return;
     }
     switch (requestCode) {
     case CAMERA_ACTIVITY_REQUEST_CODE:
          Bundle b = intent.getExtras();
          Bitmap bm = (Bitmap) b.get("data");
          mImageView.setImageBitmap(bm);

          if (b.containsKey(MediaStore.EXTRA_OUTPUT)) {
            showToast(this,"Large image");
            // it should already be saved in MediaStore (wrong documentation in SDK2 ?)
          } else {

            showToast(this,"Small image");
            MediaStore.Images.Media.insertImage(getContentResolver(), bm, null, null);
          }
          break;
     }
  }//onActivityResult

} //class
```

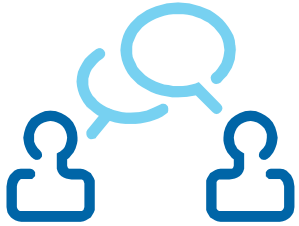
Multimedia

Questions



APPENDIX A

Reference: http://developer.android.com/guide/appendix/media-formats.html



Android Supported Media Formats

- The Core Media Formats table below describes the media format support built into the Android platform. Note that any given mobile device may provide support for additional formats or file types not listed in the table.

- For your convenience, the table T-Mobile G1 Media Formats describes additional media formats and characteristics provided by the T-Mobile G1 device. Other devices may support additional formats not listed on this page.

- As an application developer, you are free to make use of any media codec that is available on any Android-powered device, including those provided by the Android platform and those that are device-specific.

APPENDIX A. Android Supported Media Formats - Core Media Formats

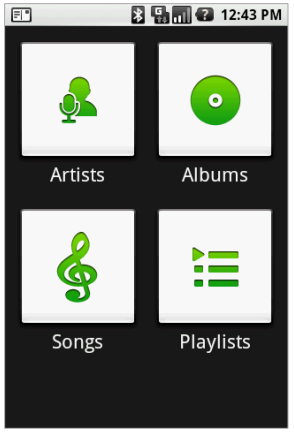
Type	Format	Encoder	Decoder	Details	File Type(s) Supported
Audio	AAC LC/LTP		X	Mono/Stereo content in any combination of standard bit rates up to 160 kbps and sampling rates from 8 to 48kHz	3GPP (.3gp) and MPEG-4 (.mp4, .m4a). No support for raw AAC (.aac)
	HE-AACv1 (AAC+)		X		
	HE-AACv2 (enhanced AAC+)		X		
	AMR-NB	X	X	4.75 to 12.2 kbps sampled @ 8kHz	3GPP (.3gp)
	AMR-WB		X	9 rates from 6.60 kbit/s to 23.85 kbit/s sampled @ 16kHz	3GPP (.3gp)
	MP3		X	Mono/Stereo 8-320Kbps constant (CBR) or variable bit-rate (VBR)	MP3 (.mp3)
	MIDI		X	MIDI Type 0 and 1. DLS Version 1 and 2. XMF and Mobile XMF. Support for ringtone formats RTTTL/RTX, OTA, and iMelody	Type 0 and 1 (.mid, .xmf, .mxmf). Also RTTTL/RTX (.rtttl, .rtx), OTA (.ota), and iMelody (.imy)
	Ogg Vorbis		X		Ogg (.ogg)
	PCM/WAVE		X	8- and 16-bit linear PCM (rates up to limit of hardware)	WAVE (.wav)
Image	JPEG	X	X	Base+progressive	JPEG (.jpg)
	GIF		X		GIF (.gif)
	PNG		X		PNG (.png)
	BMP		X		BMP (.bmp)
Video	H.263	X	X		3GPP (.3gp) and MPEG-4 (.mp4)
	H.264 AVC		X		3GPP (.3gp) and MPEG-4 (.mp4)
	MPEG-4 SP		X		3GPP (.3gp)

APPENDIX A. T-Mobile G1 Media Formats and Characteristics

The table below lists media formats supported by the T-Mobile G1 in addition to those provided as part of the Android platform. This table also details G1-specific performance characteristics of some Android core media formats.

Type	Format	Encoder	Decoder	Comment	File Type(s) Supported
Audio	WMA		X	•Supports WMA standard L1-L3: L1: 64 kbps - 161 kbps @ 44.1kHz •L2: <=161 kbps <=48 kHz •L3: <385 kbps <=48 kHz Mono and stereo profiles with 16-bits per sample. Decoder does not support WMA Pro, Lossless, or Speech codecs.	Windows Media Audio (.wma)
Video	WMV		X	Versions 7, 8 and 9. Simple profile only	Windows Media Video (.wmv)
	H.264 AVC		X	On the G1, this decoder is limited to baseline profile up to 480x320, and 600 kbps average bitrate for the video stream.	3GPP (.3gp) and MPEG-4 (.mp4)

Multimedia



Appendix B - MediaStore Class

The **android.provider.MediaStore** provider contains meta data for all available media (audio, video, images) on both internal and external storage devices.

Nested Classes		
class	MediaStore.Audio	Container for all audio content.
class	MediaStore.Images	Contains meta data for all available images.
interface	MediaStore.MediaColumns	Common fields for most MediaProvider tables
class	MediaStore.Video	

Reference: http://developer.android.com/reference/android/provider/MediaStore.html