MediaRecorder

In this document

- Requesting permission to record audio
- Using `MediaRecorder`
- Sample code

Key classes

- `MediaRecorder`

See also

- Requesting Permissions
- Supported Media Formats
- Data storage
- MediaPlayer

The Android multimedia framework includes support for capturing and encoding a variety of common audio and video formats. You can use the `MediaRecorder` APIs if supported by the device hardware.

This document shows you how to use `MediaRecorder` to write an application that captures audio from a device microphone, save the audio, and play it back (with `MediaPlayer`). To record video you'll need to use the device's camera along with `MediaRecorder`. This is described in the Camera guide.

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

Requesting permission to record audio

To be able to record, your app must tell the user that it will access the device's audio input. You must include this permission tag in the app's manifest file:

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

`RECORD_AUDIO` is considered a "dangerous" permission because it may pose a risk to the user's privacy. Starting with Android 6.0 (API level 23) an app that uses a dangerous permission must ask the user for approval at run time. After the user has granted permission, the app should remember and not ask again. The sample code below shows how to implement this behavior using `ActivityCompat.requestPermissions()`.

Using `MediaRecorder`

Initialize a new instance of `MediaRecorder` with the following calls:

- Set the audio source using `setAudioSource()`. You'll probably use `MIC`.

> **Note:** Most of the audio sources (including `DEFAULT`) apply processing to the audio signal. To record raw audio select `UNPROCESSED`. Some devices do not support unprocessed input. Call `AudioManager.getProperty("PROPERTY_SUPPORT_AUDIO_SOURCE_UNPROCESSED")` first to verify it's available. If it is not, try using `VOICE_RECOGNITION` instead, which does not employ AGC or noise suppression. You can use `UNPROCESSED` as an audio source even

> when the property is not supported, but there is no guarantee whether the signal will be unprocessed or not in that case.

- Set the output file format using `setOutputFormat()`.
- Set the output file name using `setOutputFile()`.
- Set the audio encoder using `setAudioEncoder()`.
- 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()`.

Sample code

The example activity below shows how to use `MediaRecorder` to record an audio file. It Also uses `MediaPlayer` to play the audio back.

```
package com.android.audiorecordtest;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;

import java.io.IOException;

public class AudioRecordTest extends AppCompatActivity {

    private static final String LOG_TAG = "AudioRecordTest";
    private static final int REQUEST_RECORD_AUDIO_PERMISSION = 200;
    private static String mFileName = null;

    private RecordButton mRecordButton = null;
    private MediaRecorder mRecorder = null;

    private PlayButton   mPlayButton = null;
    private MediaPlayer   mPlayer = null;

    // Requesting permission to RECORD_AUDIO
    private boolean permissionToRecordAccepted = false;
    private String [] permissions = {Manifest.permission.RECORD_AUDIO};

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] gran
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode){
            case REQUEST_RECORD_AUDIO_PERMISSION:
                permissionToRecordAccepted  = grantResults[0] == PackageManager.PERMISSION_GRANTED;
                break;
        }
        if (!permissionToRecordAccepted ) finish();

    }

    private void onRecord(boolean start) {
        if (start) {
            startRecording();
        } else {
            stopRecording();
```

```
        }
    }

    private void onPlay(boolean start) {
        if (start) {
            startPlaying();
        } else {
            stopPlaying();
        }
    }

    private void startPlaying() {
        mPlayer = new MediaPlayer();
        try {
            mPlayer.setDataSource(mFileName);
            mPlayer.prepare();
            mPlayer.start();
        } catch (IOException e) {
            Log.e(LOG_TAG, "prepare() failed");
        }
    }

    private void stopPlaying() {
        mPlayer.release();
        mPlayer = null;
    }

    private void startRecording() {
        mRecorder = new MediaRecorder();
        mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mRecorder.setOutputFile(mFileName);
        mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

        try {
            mRecorder.prepare();
        } catch (IOException e) {
            Log.e(LOG_TAG, "prepare() failed");
        }

        mRecorder.start();
    }

    private void stopRecording() {
        mRecorder.stop();
        mRecorder.release();
        mRecorder = null;
    }

    class RecordButton extends Button {
        boolean mStartRecording = true;

        OnClickListener clicker = new OnClickListener() {
            public void onClick(View v) {
                onRecord(mStartRecording);
                if (mStartRecording) {
                    setText("Stop recording");
                } else {
                    setText("Start recording");
                }
                mStartRecording = !mStartRecording;
            }
        };

        public RecordButton(Context ctx) {
            super(ctx);
            setText("Start recording");
            setOnClickListener(clicker);
        }
    }

    class PlayButton extends Button {
        boolean mStartPlaying = true;
```

```
        OnClickListener clicker = new OnClickListener() {
            public void onClick(View v) {
                onPlay(mStartPlaying);
                if (mStartPlaying) {
                    setText("Stop playing");
                } else {
                    setText("Start playing");
                }
                mStartPlaying = !mStartPlaying;
            }
        };

        public PlayButton(Context ctx) {
            super(ctx);
            setText("Start playing");
            setOnClickListener(clicker);
        }
    }

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        // Record to the external cache directory for visibility
        mFileName = getExternalCacheDir().getAbsolutePath();
        mFileName += "/audiorecordtest.3gp";

        ActivityCompat.requestPermissions(this, permissions, REQUEST_RECORD_AUDIO_PERMISSION);

        LinearLayout ll = new LinearLayout(this);
        mRecordButton = new RecordButton(this);
        ll.addView(mRecordButton,
                new LinearLayout.LayoutParams(
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        0));
        mPlayButton = new PlayButton(this);
        ll.addView(mPlayButton,
                new LinearLayout.LayoutParams(
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        ViewGroup.LayoutParams.WRAP_CONTENT,
                        0));
        setContentView(ll);
    }

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

        if (mPlayer != null) {
            mPlayer.release();
            mPlayer = null;
        }
    }
}
```