|  |  |
| --- | --- |
|  |  |

**Assignment NO 3**

**Topic: Media Recorder Class (Audio & Video Recorder Apps)**

**Name:** Rizwan Khalid. **Roll No:** 36. **Class:** BSSE E2. **Subject:** Application Development **Submitted to: S**ir Yousaf.

**Audio Recorder in Android with Example**

Last Updated : 24, Apr, 2022

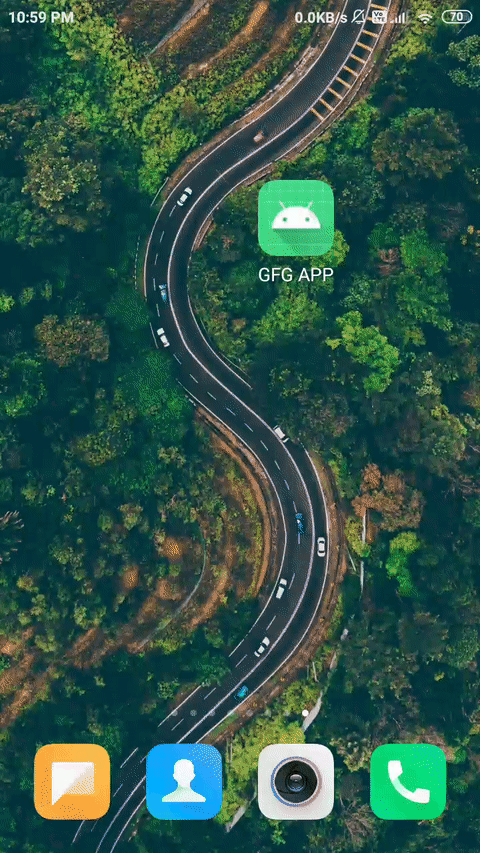
In Android for recording audio or video, there is a built-in class called **MediaRecorder**. This class in Android helps to easily record video and audio files. The Android multimedia framework provides built-in support for capturing and encoding common audio and video formats. In android for recording audio, we will use a device microphone along with **MediaRecorder**Class and for recording video, we will use the user’s device Camera and **MediaRecorder**Class. Now in this article, we will see the implementation of an audio recorder in Android with an example.

**Important Methods of MediaRecorder Class**

| **Method** | **Description** |
| --- | --- |
| setAudioSource() | This method will specify the source of the audio to be recorded. |
| setAudioEncoder() | This method is used to specify the audio encoder. |
| setOutputFormat() | This method is used to specify the output format of our audio. |
| setOutputFile() | This method is used to specify the path of recorded audio files that are to be stored. |
| stop() | This method is used to stop the recording process. |
| start() | This method is used to start the recording process. |
| release() | This method is used to release the resource that is associated with the Media recorder class. |

**Example**

Now we are creating a simple audio recorder app in which we will record audio from users device microphone and then we will store this audio recording in users device. We will also play this saved audio recording. A sample GIF is given below to get an idea about what we are going to do in this article. Note that we are going to implement this project using the **Java** language.



**Step by Step Implementation**

**Step 1: Create a New Project**

To create a new project in Android Studio please refer to [How to Create/Start a New Project in Android Studio](https://www.geeksforgeeks.org/android-how-to-create-start-a-new-project-in-android-studio/). Note that select **Java** as the programming language.

**Step 2: Add permissions in the AndroidManifest.xml file**

Add below line in the **AndroidManifest.xml** file.

* XML

|  |
| --- |
| <?**xml** version="1.0" encoding="utf-8"?>  <**manifest** xmlns:android="<http://schemas.android.com/apk/res/android>"      xmlns:tools="<http://schemas.android.com/tools>"      package="com.gtappdevelopers.camviewlibrary">        <!--Permission for recording audio and storage of audio in users device-->      <**uses-permission** android:name="android.permission.RECORD\_AUDIO"/>      <**uses-permission** android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>      <**uses-permission** android:name="android.permission.STORAGE"/>          <**application**          android:allowBackup="true"          android:icon="@mipmap/ic\_launcher"          android:label="@string/app\_name"          android:roundIcon="@mipmap/ic\_launcher\_round"          android:supportsRtl="true"          android:theme="@style/Theme.CamViewLibrary">          <**activity** android:name=".MainActivity">              <**intent-filter**>                  <**action** android:name="android.intent.action.MAIN" />                    <**category** android:name="android.intent.category.LAUNCHER" />              </**intent-filter**>          </**activity**>      </**application**>    </**manifest**> |

**Step 3: Modify the colors.xml and strings.xml file**

Below is the code for the **colors.xml** file.

* XML

|  |
| --- |
| <?**xml** version="1.0" encoding="utf-8"?>  <**resources**>      <**color** name="purple\_200">#0F9D58</**color**>      <**color** name="purple\_500">#0F9D58</**color**>      <**color** name="purple\_700">#0F9D58</**color**>      <**color** name="teal\_200">#FF03DAC5</**color**>      <**color** name="teal\_700">#FF018786</**color**>      <**color** name="black">#FF000000</**color**>      <**color** name="white">#FFFFFFFF</**color**>      <**color** name="gray">#A39696</**color**>  </**resources**> |

Below is the code for the **strings.xml** file.

* XML

|  |
| --- |
| <**resources**>      <**string** name="app\_name">GFG APP</**string**>      <**string** name="toggle\_flash">Toggle Flash</**string**>      <**string** name="action\_settings">Settings</**string**>      <**string** name="scanned\_data">Scanned Data</**string**>      <**string** name="audio\_recorder">Audio Recorder</**string**>      <**string** name="start\_recording">Start Recording</**string**>      <**string** name="stop\_recording">Stop Recording</**string**>      <**string** name="play\_recording">Play Recording</**string**>      <**string** name="stop\_playing">Stop Playing</**string**>      <**string** name="status">Status</**string**>  </**resources**> |

**Step 4: Working with the activity\_main.xml file**

Navigate to the **app > res > layout > activity\_main.xml**. Below is the code for the **activity\_main.xml** file. Comments are added inside the code to understand the code in more detail.

* XML

|  |
| --- |
| <?**xml** version="1.0" encoding="utf-8"?>  <!--XML code for activity\_main.xml-->  <**RelativeLayout**      xmlns:android="<http://schemas.android.com/apk/res/android>"      xmlns:app="<http://schemas.android.com/apk/res-auto>"      xmlns:tools="<http://schemas.android.com/tools>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="horizontal"      tools:context=".MainActivity">        <!--Heading Text View-->      <**TextView**          android:id="@+id/txthead"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_centerHorizontal="true"          android:text="@string/audio\_recorder"          android:textAlignment="center"          android:textColor="@color/black"          android:textSize="30sp" />        <!--This will display the status of our app when          we will record some audio and play that audio-->      <**TextView**          android:id="@+id/idTVstatus"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_marginTop="150dp"          android:text="@string/status"          android:textAlignment="center"          android:textSize="18sp" />        <!--Linear Layout for adding textviews          in horizontal manner-->      <**LinearLayout**          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:layout\_centerInParent="true"          android:layout\_marginTop="30dp"          android:orientation="horizontal"          android:weightSum="4">            <!--Textview to start audio recording              drawableTop will add above mic image-->          <**TextView**              android:id="@+id/btnRecord"              android:layout\_width="0dp"              android:layout\_height="wrap\_content"              android:layout\_margin="5dp"              android:layout\_weight="1"              android:background="@color/purple\_500"              android:padding="5dp"              android:text="@string/start\_recording"              android:textAlignment="center"              android:textColor="@color/white"              app:drawableTopCompat="@drawable/ic\_start\_recording" />            <!--Textview to stop audio recording              drawableTop will add above mic image-->          <**TextView**              android:id="@+id/btnStop"              android:layout\_width="0dp"              android:layout\_height="wrap\_content"              android:layout\_margin="5dp"              android:layout\_weight="1"              android:background="@color/purple\_500"              android:padding="5dp"              android:text="@string/stop\_recording"              android:textAlignment="center"              android:textColor="@color/white"              app:drawableTopCompat="@drawable/ic\_stop\_recording" />            <!--Textview to play audio that is recorded              drawableTop will add above mic image-->          <**TextView**              android:id="@+id/btnPlay"              android:layout\_width="0dp"              android:layout\_height="wrap\_content"              android:layout\_margin="5dp"              android:layout\_weight="1"              android:background="@color/purple\_500"              android:padding="5dp"              android:text="@string/play\_recording"              android:textAlignment="center"              android:textColor="@color/white"              app:drawableTopCompat="@drawable/ic\_play\_audio" />            <!--Textview to pause the play of audio recording              drawableTop will add above mic image-->          <**TextView**              android:id="@+id/btnStopPlay"              android:layout\_width="0dp"              android:layout\_height="wrap\_content"              android:layout\_margin="5dp"              android:layout\_weight="1"              android:background="@color/purple\_500"              android:lines="2"              android:padding="5dp"              android:text="@string/stop\_playing"              android:textAlignment="center"              android:textColor="@color/white"              app:drawableTopCompat="@drawable/ic\_pause\_audio" />        </**LinearLayout**>  </**RelativeLayout**> |

**Step 5: Working with the MainActivity.java file**

Navigate to the **app > java > Your app’s package name > MainActivity.java**. Below is the code for the**MainActivity.java** file. Comments are added inside the code to understand the code in more detail.

* Java

|  |
| --- |
| **import** android.content.pm.PackageManager;  **import** android.media.MediaPlayer;  **import** android.media.MediaRecorder;  **import** android.os.Bundle;  **import** android.os.Environment;  **import** android.util.Log;  **import** android.view.View;  **import** android.widget.TextView;  **import** android.widget.Toast;    **import** androidx.appcompat.app.AppCompatActivity;  **import** androidx.core.app.ActivityCompat;  **import** androidx.core.content.ContextCompat;    **import** java.io.IOException;    **import** **static** android.Manifest.permission.RECORD\_AUDIO;  **import** **static** android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE;    **public** **class** MainActivity **extends** AppCompatActivity {        // Initializing all variables..  **private** TextView startTV, stopTV, playTV, stopplayTV, statusTV;        // creating a variable for media recorder object class.  **private** MediaRecorder mRecorder;        // creating a variable for mediaplayer class  **private** MediaPlayer mPlayer;        // string variable is created for storing a file name  **private** **static** String mFileName = **null**;        // constant for storing audio permission  **public** **static** **final** **int** REQUEST\_AUDIO\_PERMISSION\_CODE = 1;        @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            // initialize all variables with their layout items.          statusTV = findViewById(R.id.idTVstatus);          startTV = findViewById(R.id.btnRecord);          stopTV = findViewById(R.id.btnStop);          playTV = findViewById(R.id.btnPlay);          stopplayTV = findViewById(R.id.btnStopPlay);          stopTV.setBackgroundColor(getResources().getColor(R.color.gray));          playTV.setBackgroundColor(getResources().getColor(R.color.gray));          stopplayTV.setBackgroundColor(getResources().getColor(R.color.gray));            startTV.setOnClickListener(**new** View.OnClickListener() {              @Override  **public** **void** onClick(View v) {                  // start recording method will                  // start the recording of audio.                  startRecording();              }          });          stopTV.setOnClickListener(**new** View.OnClickListener() {              @Override  **public** **void** onClick(View v) {                  // pause Recording method will                  // pause the recording of audio.                  pauseRecording();                }          });          playTV.setOnClickListener(**new** View.OnClickListener() {              @Override  **public** **void** onClick(View v) {                  // play audio method will play                  // the audio which we have recorded                  playAudio();              }          });          stopplayTV.setOnClickListener(**new** View.OnClickListener() {              @Override  **public** **void** onClick(View v) {                  // pause play method will                  // pause the play of audio                  pausePlaying();              }          });      }    **private** **void** startRecording() {          // check permission method is used to check          // that the user has granted permission          // to record and store the audio.  **if** (CheckPermissions()) {                // setbackgroundcolor method will change              // the background color of text view.              stopTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));              startTV.setBackgroundColor(getResources().getColor(R.color.gray));              playTV.setBackgroundColor(getResources().getColor(R.color.gray));              stopplayTV.setBackgroundColor(getResources().getColor(R.color.gray));                // we are here initializing our filename variable              // with the path of the recorded audio file.              mFileName = Environment.getExternalStorageDirectory().getAbsolutePath();              mFileName += "/AudioRecording.3gp";                // below method is used to initialize              // the media recorder class              mRecorder = **new** MediaRecorder();                // below method is used to set the audio              // source which we are using a mic.              mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);                // below method is used to set              // the output format of the audio.              mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP);                // below method is used to set the              // audio encoder for our recorded audio.              mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);                // below method is used to set the              // output file location for our recorded audio              mRecorder.setOutputFile(mFileName);  **try** {                  // below method will prepare                  // our audio recorder class                  mRecorder.prepare();              } **catch** (IOException e) {                  Log.e("TAG", "prepare() failed");              }              // start method will start              // the audio recording.              mRecorder.start();              statusTV.setText("Recording Started");          } **else** {              // if audio recording permissions are              // not granted by user below method will              // ask for runtime permission for mic and storage.              RequestPermissions();          }      }        @Override  **public** **void** onRequestPermissionsResult(**int** requestCode, String[] permissions, **int**[] grantResults) {          // this method is called when user will          // grant the permission for audio recording.  **switch** (requestCode) {  **case** REQUEST\_AUDIO\_PERMISSION\_CODE:  **if** (grantResults.length > 0) {  **boolean** permissionToRecord = grantResults[0] == PackageManager.PERMISSION\_GRANTED;  **boolean** permissionToStore = grantResults[1] == PackageManager.PERMISSION\_GRANTED;  **if** (permissionToRecord && permissionToStore) {                          Toast.makeText(getApplicationContext(), "Permission Granted", Toast.LENGTH\_LONG).show();                      } **else** {                          Toast.makeText(getApplicationContext(), "Permission Denied", Toast.LENGTH\_LONG).show();                      }                  }  **break**;          }      }    **public** **boolean** CheckPermissions() {          // this method is used to check permission  **int** result = ContextCompat.checkSelfPermission(getApplicationContext(), WRITE\_EXTERNAL\_STORAGE);  **int** result1 = ContextCompat.checkSelfPermission(getApplicationContext(), RECORD\_AUDIO);  **return** result == PackageManager.PERMISSION\_GRANTED && result1 == PackageManager.PERMISSION\_GRANTED;      }    **private** **void** RequestPermissions() {          // this method is used to request the          // permission for audio recording and storage.          ActivityCompat.requestPermissions(MainActivity.**this**, **new** String[]{RECORD\_AUDIO, WRITE\_EXTERNAL\_STORAGE}, REQUEST\_AUDIO\_PERMISSION\_CODE);      }      **public** **void** playAudio() {          stopTV.setBackgroundColor(getResources().getColor(R.color.gray));          startTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));          playTV.setBackgroundColor(getResources().getColor(R.color.gray));          stopplayTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));            // for playing our recorded audio          // we are using media player class.          mPlayer = **new** MediaPlayer();  **try** {              // below method is used to set the              // data source which will be our file name              mPlayer.setDataSource(mFileName);                // below method will prepare our media player              mPlayer.prepare();                // below method will start our media player.              mPlayer.start();              statusTV.setText("Recording Started Playing");          } **catch** (IOException e) {              Log.e("TAG", "prepare() failed");          }      }    **public** **void** pauseRecording() {          stopTV.setBackgroundColor(getResources().getColor(R.color.gray));          startTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));          playTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));          stopplayTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));            // below method will stop          // the audio recording.          mRecorder.stop();            // below method will release          // the media recorder class.          mRecorder.release();          mRecorder = **null**;          statusTV.setText("Recording Stopped");      }    **public** **void** pausePlaying() {          // this method will release the media player          // class and pause the playing of our recorded audio.          mPlayer.release();          mPlayer = **null**;          stopTV.setBackgroundColor(getResources().getColor(R.color.gray));          startTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));          playTV.setBackgroundColor(getResources().getColor(R.color.purple\_200));          stopplayTV.setBackgroundColor(getResources().getColor(R.color.gray));          statusTV.setText("Recording Play Stopped");      }  } |

All drawables are stored in the drawable folder. Navigate to the app > res > drawable folder to see all drawables. Now run the app on the Physical device to test it.

# MediaRecorder overview

 bookmark\_border

The Android multimedia framework includes support for capturing and encoding a variety of common audio and video formats. You can use the [MediaRecorder](https://developer.android.com/reference/android/media/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](https://developer.android.com/reference/android/media/MediaPlayer)). To record video you'll need to use the device's camera along with MediaRecorder. This is described in the [Camera](https://developer.android.com/guide/topics/media/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](https://developer.android.com/guide/topics/permissions/requesting#normal-dangerous) 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()](https://developer.android.com/reference/androidx/core/app/ActivityCompat" \l "requestPermissions(android.app.Activity,%20java.lang.String[],%20int)).

**Creating and running a MediaRecorder**

Initialize a new instance of [MediaRecorder](https://developer.android.com/reference/android/media/MediaRecorder) with the following calls:

* Set the audio source using [setAudioSource()](https://developer.android.com/reference/android/media/MediaRecorder" \l "setAudioSource(int)). You'll probably use [MIC](https://developer.android.com/reference/android/media/MediaRecorder.AudioSource#MIC).

**Note:** Most of the audio sources (including **DEFAULT**) apply processing to the audio signal. To record raw audio select [**UNPROCESSED**](https://developer.android.com/reference/android/media/MediaRecorder.AudioSource#UNPROCESSED). Some devices do not support unprocessed input. Call [**AudioManager.getProperty(AudioManager.PROPERTY\_SUPPORT\_AUDIO\_SOURCE\_UNPROCESSED)**](https://developer.android.com/reference/android/media/AudioManager#getProperty(java.lang.String)) first to verify it's available. If it is not, try using [**VOICE\_RECOGNITION**](https://developer.android.com/reference/android/media/MediaRecorder.AudioSource#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()](https://developer.android.com/reference/android/media/MediaRecorder" \l "setOutputFormat(int)). Note that starting with Android 8.0 (API level 26) MediaRecorder supports the MPEG2\_TS format, which is useful for streaming:

[Kotlin](https://developer.android.com/media/platform/mediarecorder#kotlin)[Java](https://developer.android.com/media/platform/mediarecorder#java)

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG\_2\_TS)

* Set the output file name using [setOutputFile()](https://developer.android.com/reference/android/media/MediaRecorder" \l "setOutputFile(java.io.File)). You must specify a file descriptor that represents an actual file.
* Set the audio encoder using [setAudioEncoder()](https://developer.android.com/reference/android/media/MediaRecorder" \l "setAudioEncoder(int)).
* Complete the initialization by calling [prepare()](https://developer.android.com/reference/android/media/MediaRecorder#prepare()).

Start and stop the recorder by calling [start()](https://developer.android.com/reference/android/media/MediaRecorder#start()) and [stop()](https://developer.android.com/reference/android/media/MediaRecorder#start()) respectively.

When you are done with the MediaRecorder instance free its resources as soon as possible by calling [release()](https://developer.android.com/reference/android/media/MediaRecorder#release()).

**Note:** On devices running Android 9 (API level 28) or higher, apps running in the background cannot access the microphone. Therefore, your app should record audio only when it's in the foreground or when you include an instance of **MediaRecorder** in a [foreground service](https://developer.android.com/guide/components/services#Foreground).

**Using MediaMuxer to record multiple channels**

Starting with Android 8.0 (API level 26) you can use a [MediaMuxer](https://developer.android.com/reference/android/media/MediaMuxer) to record multiple simultaneous audio and video streams. In earlier versions of Android you can only record one audio track and/or one video track at a time.

Use the [addTrack()](https://developer.android.com/reference/android/media/MediaMuxer" \l "addTrack(android.media.MediaFormat)) method to mix multiple tracks together.

You can also add one or more metadata tracks with custom information for each frame, but only to MP4 containers. Your app defines the format and content of the metadata.

**Adding metadata**

Metadata can be useful for offline processing. For example, data captured from the gyro sensor could be used to perform video stabilization.

When you add a metadata track, the track's mime format must start with the prefix application/. Writing metadata is the same as writing video or audio data, except that the data does not come from a MediaCodec. Instead, the app passes a ByteBuffer with an associated timestamp to the [writeSampleData()](https://developer.android.com/reference/android/media/MediaMuxer" \l "writeSampleData(int,%20java.nio.ByteBuffer,%20android.media.MediaCodec.BufferInfo)) method. The timestamp must be in the same time base as the video and audio tracks.

The generated MP4 file uses the TextMetaDataSampleEntry defined in section 12.3.3.2 of the [ISO BMFF](https://en.wikipedia.org/wiki/ISO_base_media_file_format) specification to signal the metadata's mime format. When you use a [MediaExtractor](https://developer.android.com/reference/android/media/MediaExtractor) to extract a file that contains metadata tracks, the metadata's mime format appears as an instance of [MediaFormat](https://developer.android.com/reference/android/media/MediaFormat).

**Sample code**

The [MediaRecorder](https://github.com/android/media-samples/tree/main/MediaRecorder/) sample demonstrates how to make a video recording using MediaRecorder and the Camera API.

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

[Kotlin](https://developer.android.com/media/platform/mediarecorder#kotlin)[Java](https://developer.android.com/media/platform/mediarecorder#java)

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.v4.app.ActivityCompat  
import android.support.v7.app.AppCompatActivity  
import android.util.Log  
import android.view.View.OnClickListener  
import android.view.ViewGroup  
import android.widget.Button  
import android.widget.LinearLayout  
import java.io.IOException  
  
private const val LOG\_TAG = "AudioRecordTest"  
private const val REQUEST\_RECORD\_AUDIO\_PERMISSION = 200  
  
class AudioRecordTest : AppCompatActivity() {  
  
    private var fileName: String = ""  
  
    private var recordButton: RecordButton? = null  
    private var recorder: MediaRecorder? = null  
  
    private var playButton: PlayButton? = null  
    private var player: MediaPlayer? = null  
  
    // Requesting permission to RECORD\_AUDIO  
    private var permissionToRecordAccepted = false  
    private var permissions: Array<String> = arrayOf(Manifest.permission.RECORD\_AUDIO)  
  
    override fun onRequestPermissionsResult(  
            requestCode: Int,  
            permissions: Array<String>,  
            grantResults: IntArray  
    ) {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
        permissionToRecordAccepted = if (requestCode == REQUEST\_RECORD\_AUDIO\_PERMISSION) {  
            grantResults[0] == PackageManager.PERMISSION\_GRANTED  
        } else {  
            false  
        }  
        if (!permissionToRecordAccepted) finish()  
    }  
  
    private fun onRecord(start: Boolean) = if (start) {  
        startRecording()  
    } else {  
        stopRecording()  
    }  
  
    private fun onPlay(start: Boolean) = if (start) {  
        startPlaying()  
    } else {  
        stopPlaying()  
    }  
  
    private fun startPlaying() {  
        player = MediaPlayer().apply {  
            try {  
                setDataSource(fileName)  
                prepare()  
                start()  
            } catch (e: IOException) {  
                Log.e(LOG\_TAG, "prepare() failed")  
            }  
        }  
    }  
  
    private fun stopPlaying() {  
        player?.release()  
        player = null  
    }  
  
    private fun startRecording() {  
        recorder = MediaRecorder().apply {  
            setAudioSource(MediaRecorder.AudioSource.MIC)  
            setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP)  
            setOutputFile(fileName)  
            setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB)  
  
            try {  
                prepare()  
            } catch (e: IOException) {  
                Log.e(LOG\_TAG, "prepare() failed")  
            }  
  
            start()  
        }  
    }  
  
    private fun stopRecording() {  
        recorder?.apply {  
            stop()  
            release()  
        }  
        recorder = null  
    }  
  
    internal inner class RecordButton(ctx: Context) : Button(ctx) {  
  
        var mStartRecording = true  
  
        var clicker: OnClickListener = OnClickListener {  
            onRecord(mStartRecording)  
            text = when (mStartRecording) {  
                true -> "Stop recording"  
                false -> "Start recording"  
            }  
            mStartRecording = !mStartRecording  
        }  
  
        init {  
            text = "Start recording"  
            setOnClickListener(clicker)  
        }  
    }  
  
    internal inner class PlayButton(ctx: Context) : Button(ctx) {  
        var mStartPlaying = true  
        var clicker: OnClickListener = OnClickListener {  
            onPlay(mStartPlaying)  
            text = when (mStartPlaying) {  
                true -> "Stop playing"  
                false -> "Start playing"  
            }  
            mStartPlaying = !mStartPlaying  
        }  
  
        init {  
            text = "Start playing"  
            setOnClickListener(clicker)  
        }  
    }  
  
    override fun onCreate(icicle: Bundle?) {  
        super.onCreate(icicle)  
  
        // Record to the external cache directory for visibility  
        fileName = "${externalCacheDir.absolutePath}/audiorecordtest.3gp"  
  
        ActivityCompat.requestPermissions(this, permissions, REQUEST\_RECORD\_AUDIO\_PERMISSION)  
  
        recordButton = RecordButton(this)  
        playButton = PlayButton(this)  
        val ll = LinearLayout(this).apply {  
            addView(recordButton,  
                    LinearLayout.LayoutParams(  
                            ViewGroup.LayoutParams.WRAP\_CONTENT,  
                            ViewGroup.LayoutParams.WRAP\_CONTENT,  
                            0f))  
            addView(playButton,  
                    LinearLayout.LayoutParams(  
                            ViewGroup.LayoutParams.WRAP\_CONTENT,  
                            ViewGroup.LayoutParams.WRAP\_CONTENT,  
                            0f))  
        }  
        setContentView(ll)  
    }  
  
    override fun onStop() {  
        super.onStop()  
        recorder?.release()  
        recorder = null  
        player?.release()  
        player = null  
    }  
}