Nazmul Idris (Naz) Follow
Googler, entrepreneur, leader, coder, designer, dancer, TaiChi'er, Yogi, racer, healer, storyteller. I'm about authenticity, empowerment & doing what matters
Aug 22, 2017 · 4 min read

Building a simple audio app in Android (Part 1/3)

MediaPlayer introduction

Introduction

Android Media APIs encompass a lot of advanced functionality that allow developers to create rich media experiences. They include things like `ExoPlayer` , `MediaSession` , audio focus, volume shaping, and lots of other amazing capabilities just around media playback and control.

The goal of this series of articles is to get you started with the Android `MediaPlayer` API by going thru the journey of creating a very basic audio playback application called "A Simple MediaPlayer" app. This is the first part of a 3 part series that includes:

1. **MediaPlayer introduction** (*this article*)
2. Building the app
3. Synchronizing with the SeekBar

Part 1/3—MediaPlayer introduction

This article will cover the following:

- How the `MediaPlayer` works by considering some simple audio playback tasks and how they map to the `MediaPlayer` state machine.

Basic MediaPlayer tasks

Here are the basic tasks that `MediaPlayer` needs to handle:

- **Load a media file for playback**. This is done with the methods `setDataSource()` , `prepare()` , and `prepareAsync()` .
- **Start playback / Play audio**. This is handled by `start()` .
- **Pause playback (once playback has started)**. This is handled by `pause()` .
- **Stop playback and reset the MediaPlayer, so that you can load another media file into it**. This is handled by `reset()` .
- **Find the length of a song (in ms)**. This is handled by `getDuration()` .
- **Find what part of the song is playing**. This is handled by `getCurrentPosition()` .
- **Jump to a specific time position (in ms) in the song and play from there**. This is handled by `seekTo(position)` .
- **Check to see if audio is being played back right now**. This is handled by `isPlaying()` .
- **Find out when a song is done playing**. This is handled by attaching a `MediaPlayer.OnCompletionListener` . Your code will get an `onCompletion()` callback from the listener.
- **Deallocate resources used by the player**. This is handled by `release()` , which releases all the resources attached to the player. After being released the player is no longer usable.

This list covers the basic playback tasks for audio content, but it isn't exhaustive. There are more complex functions for sophisticated audio apps such as streaming media, dealing with audio focus, volume shaping, and working with `MediaSession`. For more information on these advanced topics, please consult the `MediaPlayer` API guide.

Simplified state machine

Here is a simplified description of `MediaPlayer's` state machine for audio playback:

First create an instance of `MediaPlayer`.

1. Then load the media file that you want into the player. This means using the `setDataSource()` method with a parameter that describes where the media file is coming from.
2. Now you need to prepare your data source. There are two ways of doing this:

- Use `prepare()`. This method will block the calling thread so should only be used for data sources which are stored locally on the device.
- Use `prepareAsync()`. This method will prepare the data source asynchronously and should be used for remote or large data sources. Using this method you get a callback when your media is ready for playback, and you can enable media playback controls in your app's UI at this point. You can use the `MediaPlayer.OnPreparedListener` to get notified when the player has loaded the content for you to start playback. You can attach this listener to your player using `setOnPreparedListener(listener)`.

Now that your audio has been loaded, you can play and pause your audio. And you can stop and restart playback.

1. **Play**—use `start()` to tell the player to start playback. You can ask the player if it's playing using `isPlaying()`.
2. **Pause**—you can pause only if `isPlaying()` is true so always check this before you call `pause()`.
3. **Stop**—use the `reset()` method to do this. It will stop playback and reset the player so that you can load other media files into it.

4. **Seek**—use the `seekTo()` method with a time offset to move audio playback to the desired position. This is needed when you allow the user to move playback to any position in the currently loaded media.

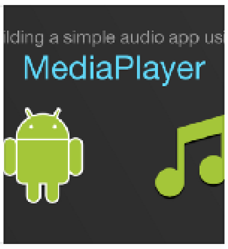
When you are done with the instance of MediaPlayer, be sure to `release()` it. Once released, you have to create a new `MediaPlayer` object and start from Step 1 in order to play another media file.

To get started with building the app, please checkout Part 2/3 of this series- Building the app.

Building a simple audio app in Android (Part 2/3)

Building the app

medium.com



Android Media Resources

- Understanding MediaSession
- Android Media API Guides—Media Apps Overview
- Android Media API Guides—Working with a MediaSession