链滴

Android MediaPlayer 基础简介

作者：RustFisher
原文链接：https://ld246.com/article/1569286591914
来源网站：链滴

本文链接： Android MediaPlayer 基础简介

简单介绍MediaPlayer的基本概念，状态，常用的方法与监听器。

什么是MediaPlayer

MediaPlayer类可以用来播放音视频文件，或者是音频流。开发者可以用它来播放本地音频，或者是络在线音频。

MediaPlayer属于android.media包。

MediaPlayer的状态

播放控制由状态机控制。在日常生活中，我们常见的音频状态有播放中，暂停，停止，缓冲等等。

MediaPlayer的状态有如下几种：

- Idle
- End
- Error
- Initialized
- Preparing
- Prepared
- Started
- Stopped
- Paused
- PlaybackCompleted

状态的切换参考官方图例。

这里稍微解释一下状态转换图片。椭圆代表MediaPlayer可能停留的状态。椭圆之间的箭头表示方法用，状态切换的方向。单箭头表示方法同步调用，双箭头表示异步调用。

从图中我们可以看出状态切换的路径和涉及到的方法。

Idle与End状态

当new一个MediaPlayer或者调用了reset方法，当前MediaPlayer会处于Idle状态。调用release后，处于End状态。在这2个状态之间的状态可以看做是MediaPlayer对象的生命周期。

在新创建MediaPlayer和调用reset的MediaPlayer之间有一些细微的差别。

这两种情况都处于Idle状态，调用 getCurrentPosition(), getDuration(), getVideoHeight(), getVidoWidth(), setAudioAttributes(android.media.AudioAttributes), setLooping(boolean), setVolum(float, float), pause(), start(), stop(), seekTo(long, int), prepare() 或 prepareAsync()方法都会抛错误，如果是新实例化的MediaPlayer，不会回调 OnErrorListener.onError()；但如果是reset后的MdiaPlayer，会回调 OnErrorListener.onError()并且转换到Error状态。

如果MediaPlayer对象不再使用了，立即调用release()方法，释放内部播放器占用的资源。这些资源

能是唯一的，比如硬件加速组件。如果调用release失败，可能会引起一连串的MediaPlayer实例失效当MediaPlayer处于End状态，它就不能再转移到其它状态了。

new一个MediaPlayer，处于Idle状态。如果用create方法创建实例，当创建完成时处于Prepared状。

发生错误

一些情形可能会让MediaPlayer操作失败，比如不支持的音视频格式，分辨率过高，网络超时等等。

因此在这些情形下错误处理和恢复非常重要。有时候编程错误也会导致MediaPlayer操作错误。

开发者可以设置错误监听器setOnErrorListener(android.media.MediaPlayer.OnErrorListener)。错误发生时，会调用用户实现的OnErrorListener.onError()方法。

不管有没有设置监听器，错误发生时MediaPlayer会进入Error状态。

为了重复使用同一个MediaPlayer对象，可以使用reset()方法把它从Error状态恢复到Idle状态。

设置错误监听器OnErrorListener是一个好的编程习惯。开发者可以监听到播放引擎的错误通知。

有时候会抛出IllegalStateException异常，比如在错误的状态调用了prepare(), prepareAsync()方法或是setDataSource方法。

设置音源 setDataSource

调用setDataSource(java.io.FileDescriptor), 或者 setDataSource(java.lang.String), 或者 setDataSource(android.content.Context, android.net.Uri), 或者 setDataSource(java.io.FileDescriptor, long, long), 或者 setDataSource(android.media.MediaDataSource) 可以将MediaPlayer的状态从Idle到Initialized状态。

如果在Idle状态之外的状态调用了setDataSource()，会抛出IllegalStateException异常。

开发者应该留意setDataSource方法抛出的IllegalArgumentException和IOException异常。

播放音频前必须在Prepared状态

MediaPlayer在开始播放音频前必须处于Prepared状态。

MediaPlayer有同步和异步2种方式来进入Prepared状态。如果是异步的方式，会先转到Preparing态，再转到Prepared状态。

当准备完成时，内部的播放引擎会回调用户之前设置的OnPreparedListener的onPrepared()方法。

开发者必须注意的是，Preparing状态是一个过渡状态（transient state）。

处于Prepared状态时，可以通过相对应的方法设置音量，屏幕常亮，播放循环等。

开始播放

播放音频必须调用start()方法。调用start()返回成功后，MediaPlayer处于Started状态。

可以通过isPlaying()来判断当前是否在Started状态。

如果开发者设置了OnBufferingUpdateListener，Android内部播放器会向外传递buffer信息。

如果当前处于Started状态，再调用start()方法没有效果。

暂停播放与继续播放

音频可以被暂停播放和继续播放，也可以调整播放的位置。通过pause()方法来暂停音频播放。

成功调用pause()方法后，MediaPlayer进入Paused状态。

应当注意的是，MediaPlayer在Started状态与Paused状态之间切换是异步的。播放音频流的时候，个转换过程可能会需要几秒钟。

MediaPlayer暂停时，start()方法可以从暂停的位置继续播放。成功调用start方法后会进入Started状。

处于Paused状态时，调用pause()方法没有效果。

停止

调用stop()方法让MediaPlayer从Started, Paused, Prepared 或 PlaybackCompleted 状态进入 Stoped 状态。

在Stopped状态时，必须先调用prepare() 或 prepareAsync()进入Prepared状态后，才能播放音频。

处于Stopped状态时，调用stop()方法没有效果。

调整播放位置

调用seekTo(long, int)来调整播放位置。

seekTo(long, int)是一个异步方法，虽然它能立刻返回，但实际的位置调整可能会消耗一段时间，特是在播放音频流的时候。当实际播放位置调整后，内部播放器会回调开发者设置的OnSeekComplete.nSeekComplete()。

在Prepared, Paused 和 PlaybackCompleted状态中，都可以调用seekTo方法。

可以通过getCurrentPosition()方法来获取当前播放位置。开发者可以得知当前播放的进度等等。

播放完毕

音频播放完成后，播放完毕。

如果调用setLooping(boolean)为true，MediaPlayer会停留在Started状态。

如果setLooping为false，内部播放器会调用开发者设置的OnCompletion.onCompletion()，并且进PlaybackCompleted状态。

处于PlaybackCompleted状态时，调用start()方法可以从头开始播放音频。

常用监听器

开发者可以设置一些监听器，监听MediaPlayer的状态，错误事件等等。开发者应在同一个线程中创MediaPlayer与设置的监听器。

setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener)

监听MediaPlayer准备完成。一般与prepareAsync配合使用。

setOnVideoSizeChangedListener(android.media.MediaPlayer.OnVideoSizeChangedListener)

获知video大小或video大小改变时的监听。

setOnSeekCompleteListener(android.media.MediaPlayer.OnSeekCompleteListener)

监听调整位置完成。

setOnCompletionListener(android.media.MediaPlayer.OnCompletionListener)

播放完成。

```
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        // 当前播放完毕
    }
});
```

setOnBufferingUpdateListener(android.media.MediaPlayer.OnBufferingUpdateListener)

监听缓冲进度。在播放网络音频时常用。

缓冲监听器OnBufferingUpdateListener

```
    mMediaPlayer.prepareAsync();
    mMediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener(
 {
        @Override
        public void onBufferingUpdate(MediaPlayer mp, int percent) {
            // 例如在这里更新UI
        }
    });
```

setOnInfoListener(android.media.MediaPlayer.OnInfoListener)

监听普通信息或者警告信息。

setOnErrorListener(android.media.MediaPlayer.OnErrorListener)

监听错误信息。错误发生时，可以在这里处理错误。

```
mediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener() {
    @Override
    public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
        LogUtil.e(TAG_PREFIX + " onERR i = " + i + " i1 = " + i1);
        return true; // 返回true表示在此处理错误，不会回调onCompletion
    }
});
```

注意onError的返回值。可以选择自己处理error。

```
     * @return True if the method handled the error, false if it didn't.
     * Returning false, or not having an OnErrorListener at all, will
     * cause the OnCompletionListener to be called.
```

```
 */
boolean onError(MediaPlayer mp, int what, int extra);
```

需要的权限

播放网络音频时需要Manifest.permission.INTERNET权限。