Descripción general de MediaPlayer

El marco de trabajo de contenido multimedia de Android admite la reproducción de diversos tipos de contenido multimedia comunes para que puedas integrar audio, video e imágenes con facilidad en tus apps. Puedes reproducir audio o video desde archivos multimedia almacenados en los recursos de tu app (recursos sin procesar), desde archivos independientes del sistema de archivos o desde un flujo de datos que llega a través de una conexión de red, todo mediante diferentes API de MediaPlayer.

Se puede escribir una app de reproducción de contenido multimedia que interactúa con el usuario y el sistema para obtener un buen rendimiento y una experiencia del usuario agradable.

Nota: Solo puedes reproducir los datos de audio en el dispositivo de salida estándar. Por el momento, este es el ALTAVOZ del dispositivo móvil o los AURICULARES Bluetooth. No puedes reproducir archivos de sonido en el audio de la conversación durante una llamada.

Conceptos básicos

Las siguientes clases se usan para reproducir sonido y video en el marco de trabajo de Android:

MediaPlayer

Esta clase es la API principal para reproducir sonido y video.

AudioManager

Esta clase administra fuentes y salidas de audio en un dispositivo.

Declaraciones del manifiesto

Antes de comenzar a desarrollar tu app con MediaPlayer, asegúrate de que el manifiesto contenga las declaraciones adecuadas para permitir el uso de funciones relacionadas.

- **Permiso de Internet**: si estás usando MediaPlayer para transmitir contenido basado en la red, tu app debe solicitar acceso a la red.

- **Permiso de bloqueo de activación**: si la app del reproductor debe evitar que la pantalla se oscurezca o que el procesador entre en suspensión, o si utiliza los métodos

Cómo usar MediaPlayer

Uno de los componentes más importantes del marco de trabajo de medios es la clase MediaPlayer. Un objeto de esta clase puede recuperar, decodificar y reproducir audio y video con una configuración mínima. Es compatible con varias fuentes de medios diferentes, entra las que se incluyen las siguientes:

- Recursos locales
- URI internos, como uno que se pueda obtener de un agente de resolución de contenido
- URL externas (transmisión)

Para obtener una lista de los formatos de medios compatibles con Android, consulta la página Formatos de medios compatibles.

Preparación asíncrona

En principio, usar MediaPlayer puede ser sencillo. Sin embargo, es importante tener en cuenta que se necesitan algo más para integrarlo correctamente con una app para Android típica. Por ejemplo, la ejecución de la llamada a prepare() puede tardar mucho tiempo, ya que puede implicar la obtención y decodificación de datos multimedia. Por lo tanto, como en el caso de cualquier método que pueda tardar mucho en ejecutarse, **nunca debes llamarlo desde el subproceso de IU de tu app**. Si haces eso, la IU se bloquea hasta que el método muestre resultados, lo que constituye una experiencia de usuario muy mala y puede causar un error ANR (aplicación no responde). Incluso si esperas que tu recurso se cargue con rapidez, recuerda que cualquier acción que tarde más de una décima de segundo en responder en la IU provoca una pausa notable y le da al usuario la impresión de que la app es lenta.

Para evitar colgar el subproceso de IU, genera otro para preparar el MediaPlayer y notifica al subproceso principal cuando haya terminado. Sin embargo, si bien puedes escribir por tu cuenta la lógica de subprocesos, este patrón es tan común cuando se usa MediaPlayer que el marco de trabajo proporciona una manera conveniente de realizar esta tarea mediante el método prepareAsync().

Cómo retirar MediaPlayer

Un MediaPlayer puede consumir valiosos recursos del sistema. Por lo tanto, siempre debes tomar precauciones adicionales para asegurarte de no prolongar una instancia de MediaPlayer más de lo necesario. Cuando termines con esto, siempre debes llamar a release() para asegurarse de que se retiren correctamente todos los recursos del sistema asignados. Por ejemplo, si estás usando un MediaPlayer y su actividad recibe una llamada a onStop()

Cómo administrar el estado

Otro aspecto de un MediaPlayer que debes tener en cuenta es que está basado en el estado. Es decir, el MediaPlayer tiene un estado interno que siempre debes tener en cuenta cuando escribas tu código, porque algunas operaciones solo son válidas cuando el reproductor está en estados específicos. Si realizas una operación en el estado incorrecto, el sistema puede generar una excepción o generar otros comportamientos no deseados.