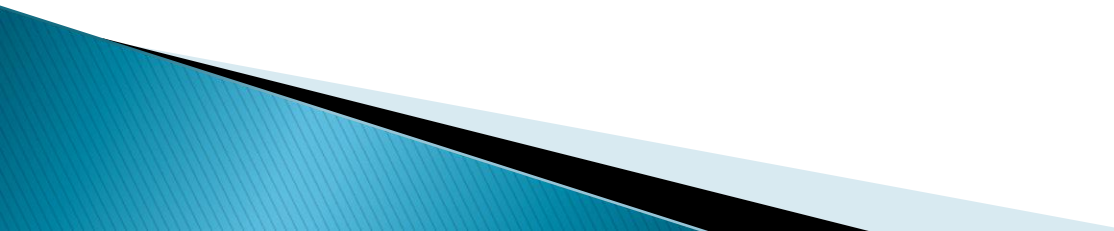


Android: Audio

android.media

Flujos de audio

Android dispone de diferentes flujos de audio:

- ▶ **STREAM_ALARM**
 - ▶ **STREAM_DTMF**
 - ▶ **STREAM_MUSIC**
 - ▶ **STREAM_NOTIFICATION**
 - ▶ **STREAM_RING**
- 

Fuentes

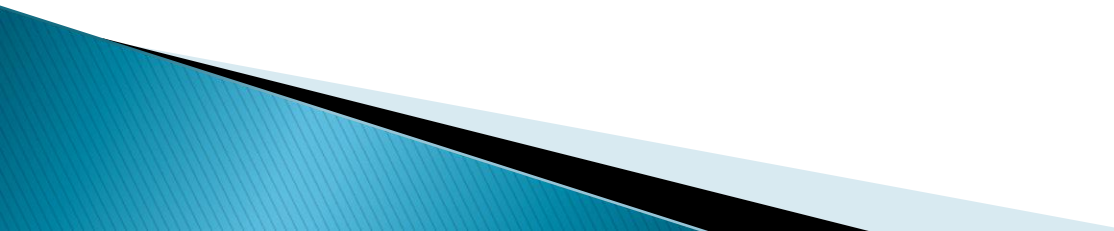
- ▶ Los archivos de audio que se pueden reproducir provienen de diferentes fuentes.
- ▶ Archivos locales:
 - res/raw/
 - assets/ (límite de 1 MB)
 - Memoria interna o externa.
- ▶ Archivos externos: por streaming, a partir de una URL.

Reproducir de res/raw/

```
MediaPlayer mp = MediaPlayer.create(this,  
                                     R.raw.sonido);  
mp.start();
```

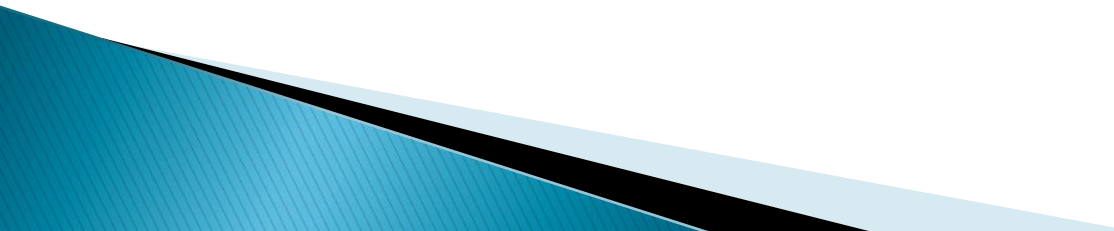
Detener, MediaPlayer consume muchos recursos:

```
mp.stop();  
mp.release();  
mp = null;
```



Reproducir de assets /

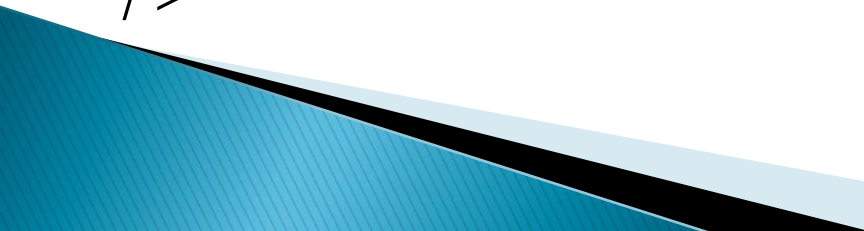
```
AssetFileDescriptor descriptor;  
descriptor = getAssets().openFd("archivo");  
long inicio = descriptor.getStartOffset();  
long fin = descriptor.getLength();  
mp=new MediaPlayer();  
mp.setDataSource(descriptor.getFileDescriptor()  
                , inicio, fin);  
mp.prepare();  
mp.start();
```



Reproducir de una url

```
String url = "..."; //archivo local o de internet
mp = new MediaPlayer();
mp.setAudioStreamType(
    AudioManager.STREAM_MUSIC);
mp.setDataSource(url);
mp.prepare();
mp.start();
```

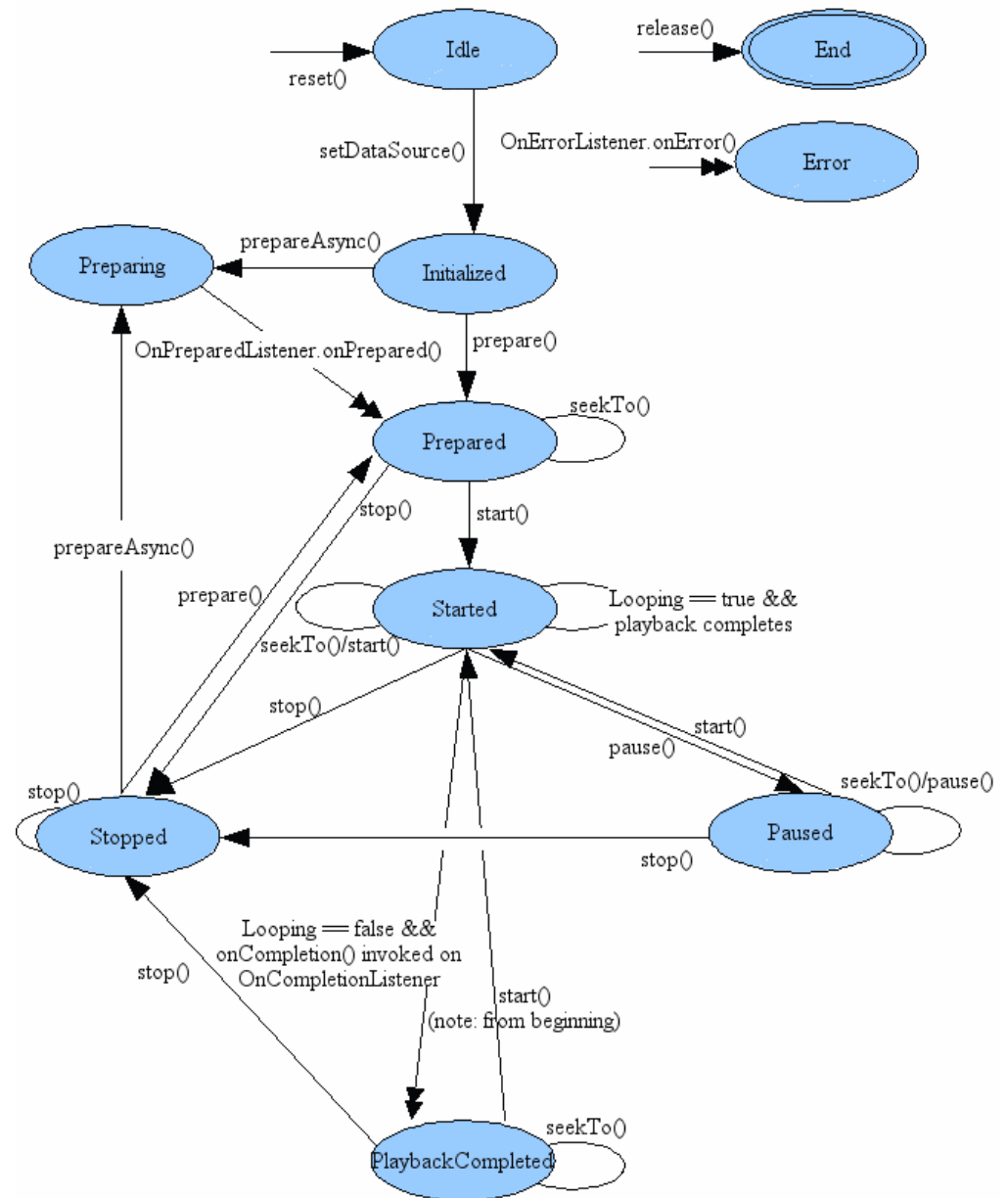
```
<uses-permission android:name=
    "android.permission.INTERNET" />
<uses-permission android:name=
    "android.permission.READ_EXTERNAL_STORAGE"
/>
```



android.permission.WAKE_LOCK

- ▶ Para impedir que la pantalla se apague.
- ▶ Para impedir que el procesador se duerma.
- ▶ `<uses-permission android:name="android.permission.WAKE_LOCK" />`
- ▶ `public void setScreenOnWhilePlaying (boolean screenOn)`
- ▶ `public void setWakeMode (Context context, int mode)`, usando el modo `PowerManager.PARTIAL_WAKE_LOCK` se mantiene el procesador activo mientras la pantalla se apaga

Estados del Media Player



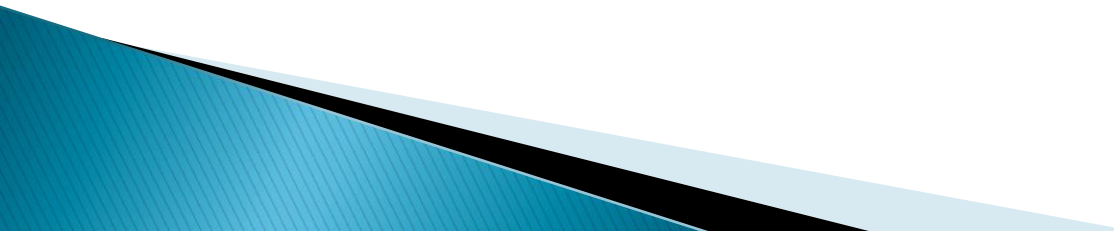
Control de estados

- ▶ El estado inicial es **idle**. En este estado se debe establecer el origen de datos con **setDataSource()**.
- ▶ En el estado **inicializado** se debe preparar el audio con **prepare()** o **prepareAsync()**.
- ▶ En el estado **preparado** se puede ejecutar **start()**, **seekTo()**, **pause()** y **stop()**.
- ▶ En el estado **pausado** se puede ejecutar **start()** y **stop()**.
- ▶ En el estado **parado** o **completado** se debe volver a preparar el audio para reproducir.
- ▶ Al ejecutar **reset()** se vuelve al estado inicial.

Preparación asíncrona

```
mp.setOnPreparedListener(new Preparado());  
mp.setDataSource(url);  
mp.prepareAsync();
```

```
private class Preparado implements  
                                OnPreparedListener {  
    @Override  
    public void onPrepared(MediaPlayer arg0) {  
        mp.start();  
    }  
}
```



Volumen y bloqueo

- ▶ Para controlar el volumen del audio:

```
setVolumeControlStream(  
    AudioManager.STREAM_MUSIC);
```

```
mp.setVolume(vollzq, volDer);
```

- ▶ Para impedir el bloqueo:

```
mp.setWakeMode(getApplicationContext(),  
    PowerManager.PARTIAL_WAKE_LOCK);
```

Obtener el foco del audio

```
AudioManager am = (AudioManager)  
    getSystemService(Context.AUDIO_SERVICE);
```

```
int r = am.requestAudioFocus(new Foco(),  
    AudioManager.STREAM_MUSIC,  
    AudioManager.AUDIOFOCUS_GAIN);
```

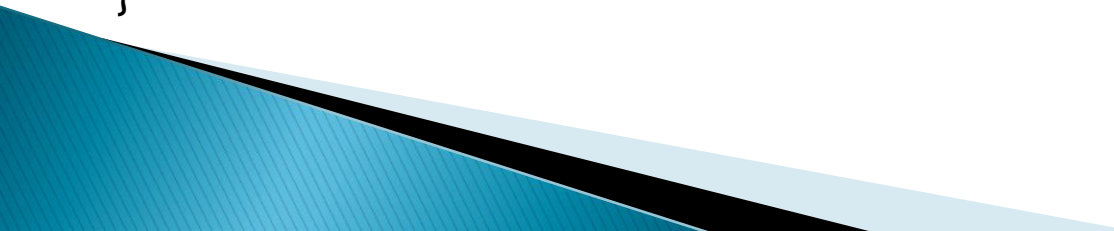
```
if(r==AudioManager.  
    AUDIOFOCUS_REQUEST_GRANTED)
```

```
...
```



Controlar la pérdida de foco

```
private class Foco implements OnAudioFocusChangeListener {  
    @Override  
    public void onAudioFocusChange(int focusChange) {  
        switch (focusChange) {  
            case AudioManager.AUDIOFOCUS_GAIN:  
                mp.setVolume(1.0f, 1.0f);  
                break;  
            case AudioManager.AUDIOFOCUS_LOSS:  
                break;  
            case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT:  
                break;  
            case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK:  
                mp.setVolume(0.1f, 0.1f);  
                break;  
        }  
    }  
}
```



Controlar la finalización del audio

```
mp.setOnCompletionListener(new Fin());
```

```
private class Fin implements
```

```
    OnCompletionListener {
```

```
    @Override
```

```
    public void onCompletion(MediaPlayer mp) {
```

```
        ...
```

```
    }
```

```
}
```

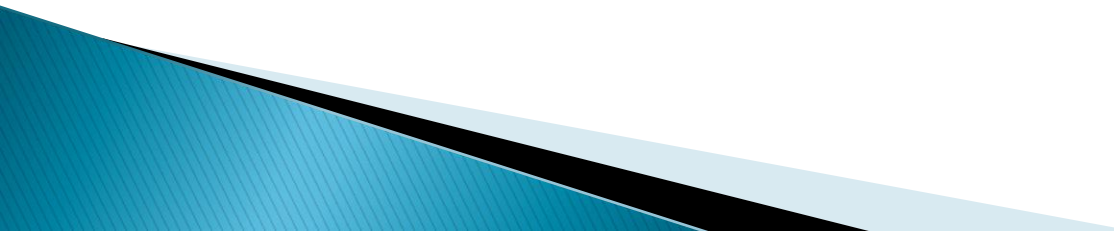


Consideraciones finales

En el manifiesto, se puede indicar que ante cambios de orientación no se pierda la referencia al objeto MediaPlayer:

```
android:configChanges="orientation|  
screenSize|keyboardHidden"
```

Es importante no perder en ningún momento el control del MediaPlayer.



- ▶ <http://developer.android.com/guide/topics/media/mediaplayer.html>
 - ▶ <http://developer.android.com/training/managing-audio/audio-focus.html>
 - ▶ <http://developer.android.com/guide/topics/resources/runtime-changes.html>
- 