Segunda parte

Reproducir vídeo desde Internet

En una aplicación del mundo real, muchos archivos multimedia son demasiado grandes para incrustarlos directamente en la aplicación, como lo hizo con el vídeo en la aplicación SimpleVideoView.
Más comúnmente, si su aplicación reproduce medios, obtendrá los archivos que necesita del almacenamiento externo, como una tarjeta SD, o localizará y almacenará en búfer el archivo multimedia de un servidor en Internet.

Si reproduce medios desde Internet, la clase **VideoView** y el **MediaPlayer** subyacente implementan gran parte del trabajo de fondo por usted.

- Cuando lo usa, no necesita abrir una conexión de red o configurar una tarea en segundo plano para almacenar en búfer el archivo multimedia.

Sin embargo, debes controlar el período de tiempo entre el momento en que le indicas a tu aplicación que reproduzca el archivo multimedia y el momento en que el contenido de ese archivo está realmente disponible para reproducirse.

- Si no haces nada, puede parecer que la aplicación se bloquea durante bastante tiempo mientras el archivo se almacena en búfer, especialmente en conexiones de red lentas.
- Incluso un mensaje al usuario de que algo está sucediendo proporciona una mejor experiencia de usuario.
- **VideoView** proporciona un detector de eventos de preparación para manejar este caso.

Para usar la devolución de llamada, establezca el **URI** del medio que se reproducirá y, a continuación, reciba una devolución de llamada cuando ese medio esté listo para reproducirse.
MediaPlayeronPrepared()

En esta tarea, modifique la aplicación **SimpleVideoView** para reproducir un vídeo desde una dirección **URL** en Internet. Use el detector de eventos de preparación para controlar la actualización de la interfaz de usuario de la aplicación mientras se carga el archivo.

Agregar un mensaje de almacenamiento en búfer

En esta tarea, se modifica el diseño de la aplicación para mostrar un mensaje de almacenamiento en búfer simple y se usa la visibilidad de la vista para mostrar y ocultar el mensaje en el momento adecuado.

1. En el archivo de diseño principal, agregue un elemento TextView. El nuevo TextView está centrado en el diseño, igual que VideoView y aparece en la parte superior.

```
<TextView
android:id="@+id/buffering_textview"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_margin="8dp"
android:text="Buffering..."
android:textSize="18sp"
android:textStyle="bold"
android:textColor="@android:color/white"
app:layout_constraintBottom_toBottomOf="parent"app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
```

2. En MainActivity , cambie la constante VIDEO_SAMPLE constant para indicar la URL para el archivo de medios

```
private static final String VIDEO_SAMPLE =
"http://...../crearunavd.mp4";
```

Puede usar cualquier URL para cualquier video que esté disponible como archivo en Internet.
Si usa su propia URL para el archivo de video, asegúrese de que el formato de video sea compatible con Android, así como por la versión específica de Android que es su emulador o dispositivo correr.

3. Agregue una variable miembro para contener el **TextView** que muestra el mensaje de almacenamiento en búfer.

```
private TextView mBufferingTextView;
```

4. En **onCreate**() , obtenga una referencia a TextView en el diseño:

```
mBufferingTextView = findViewById(R.id.buffering_textview);
```

5. En el método getMedia(), cambie la implementación para probar si la cadena entrante es un URL o un recurso sin procesar. Devuelve el URI apropiado.

```
private Uri getMedia(String mediaName) {
if (URLUtil.isValidUrl(mediaName)) {
// media name is an external URL
return Uri.parse(mediaName);
} else { // media name is a raw resource embedded in the app
return Uri.parse("android.resource://" + getPackageName() +
"/raw/" + mediaName);
}
}
```

Agregar el permiso de internet

En el archivo de manifiesto de Android, agregue un permiso de Internet para permitir que la aplicación acceda a los medios
archivo en internet.
1. En AndroidManifest.xml, agregue esta línea justo antes del elemento <application>, en el nivel superior del manifiesto.

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

Agregar una devolución de llamada onPrepared()

La preparación de videos u otros medios puede implicar transmitirlos, almacenarlos en la memoria y decodificarlos para que esté listo para reproducir. Con VideoView y MediaPlayer, ocurre el paso de preparación de forma asíncrona en un subproceso separado, por lo que su aplicación no tiene que hacer una pausa y esperar.
Use la devolución de llamada **onPrepared**() para permitir que su aplicación sea notificada cuando el paso de preparación haya terminado completado y el medio está listo para reproducir.

1. En el método **initializePlayer**(), antes de la definición de **setOnCompletionListener**() , cree un nuevo MediaPlayer.onPreparedListener, sobrescriba el método **onPrepared**() y use **setOnPreparedListener**() para agregar un listener a VideoView .

```
mVideoView.setOnPreparedListener(
new MediaPlayer.OnPreparedListener() {
@Override
public void onPrepared(MediaPlayer mediaPlayer) {
// Implementation here.
}
});
```

2. Dentro del método **onPrepared**(), establezca la visibilidad de **TextView** en invisible. Esto elimina el Mensaje **"Almacenamiento en búfer..."**.

```
mBufferingTextView.setVisibility(VideoView.INVISIBLE);
```

3. En **initializePlayer**() , busque las líneas de código que prueban la posición actual, busque esa posición e inicie la reproducción. Mueve esas líneas al método **onPrepared**(), colocándolas después de las llamadas a **setVisibility**() .
La definición final de **onPrepared**() se ve así:

```
mVideoView.setOnPreparedListener(
new MediaPlayer.OnPreparedListener() {
@Override
public void onPrepared(MediaPlayer mediaPlayer) {
mBufferingTextView.setVisibility(VideoView.INVISIBLE);
if (mCurrentPosition > 0) {
mVideoView.seekTo(mCurrentPosition);
} else {
mVideoView.seekTo(1);
}
mVideoView.start();
}});
```

4. En la parte superior de **initializePlayer**() , antes de configurar el URI multimedia para reproducir, restaure la visibilidad del Buffering TextView:

```
mBufferingTextView.setVisibility(VideoView.VISIBLE);
```

Cada vez que se llama a **initializePlayer**(), el mensaje de almacenamiento en búfer debe activarse. esta apagado solo cuando se llama a **onPrepared**() y el medio está listo para reproducirse.

5. Compile y ejecute la aplicación. Inicialmente aparece el mensaje "**Buffering..."** en la pantalla. Dependiendo de la velocidad de su conexión de red, después de unos momentos aparecerá el video y comenzará a reproducirse.

Resumen

- Las aplicaciones multimedia en Android tienen un conjunto estándar de componentes, incluida una interfaz de usuario y un reproductor multimedia.
- Los archivos multimedia (audio y video) se pueden reproducir desde una variedad de fuentes, incluso incrustados en los recursos de la aplicación, almacenados en medios externos como una tarjeta SD o reproducidos como medios de transmisión desde Internet.
- La forma más sencilla de reproducir vídeo en la aplicación es usar la clase VideoView . La clase ajusta un MediaPlayer y un SurfaceView . se puede agregar a un diseño como cualquier otra vista.
- Use VideoView.setVideoURI() para especificar el URI del archivo de vídeo que se va a reproducir, tanto si ese URI está en los recursos de la aplicación como en Internet. Este método precarga los datos de ejemplo de vídeo en un subproceso asincrónico.
- Utilice VideoView.start() para iniciar la reproducción de vídeo, una vez que el vídeo esté disponible para su reproducción.
- Utilice VideoView.stopPlayback() para detener la reproducción de vídeo y liberar los recursos que está utilizando. VideoView
- Utilice VideoView.getCurrentPosition() para recuperar la posición de reproducción actual, en milisegundos. Utilice VideoView.seekTo() para buscar una posición de reproducción específica, en milisegundos.
- El widget MediaController () proporciona un conjunto de controles comunes (reproducción, pausa, avance rápido, rebobinado y un control deslizante de progreso) para la reproducción multimedia.
- Utilice MediaController.setMediaPlayer() para conectar un reproductor multimedia como a al controlador. VideoView
- Utilice VideoView.setMediaController() para adjuntar un archivo . MediaControllerVideoView
- Usa los métodos y ciclo de vida para iniciar y detener la reproducción de vídeo en tu aplicación. Estos métodos representan el ciclo de vida visible de la aplicación. onStart()onStop()
- En versiones de Android inferiores a Nougat, representa el final del ciclo de vida visible de la aplicación, pero el audio puede continuar reproduciéndose durante varios segundos hasta que se llama. Agregue una prueba para la versión de Android para pausar la reproducción de video en versiones anteriores de Android.
- La clase no conserva el estado de reproducción de vídeo en ninguna transición del ciclo de vida, incluidos los cambios en el fondo/primer plano, la orientación del dispositivo y el modo multiventana. Para conservar esta información de estado, guarde y restaure la posición actual del vídeo en el paquete de estado de la instancia de actividad. VideoView

- La reproducción multimedia utiliza muchos recursos del sistema. Asegúrese de liberar estos recursos lo antes posible, normalmente en . stopPlayback()
  Los recursos utilizados por VideoView se liberan al llamar a onStop()
  La clase (y, por lo tanto,) proporciona un conjunto de devoluciones de llamada de eventos multimedia para varios eventos, incluidos y MediaPlayerVideoViewonCompletion()onPrepared()
- La devolución de llamada onCompletion() se invoca cuando se completa la reproducción multimedia. Use esta devolución de llamada para anunciar el final del medio o restablecer la interfaz de usuario a un estado predeterminado.
- La devolución de llamada onPrepared() se invoca cuando el medio está listo para reproducirse.
  Dependiendo del medio y su ubicación (incrustado en la aplicación, disponible en el almacenamiento local, reproducido desde Internet), puede tomar algún tiempo para que los medios estén disponibles. Utilice la devolución de llamada para mantener al usuario informado sobre el estado de los medios proporcionando un mensaje de "Buffering" o algún otro mensaje de estado. OnPrepared()

fin de primera y segunda parte