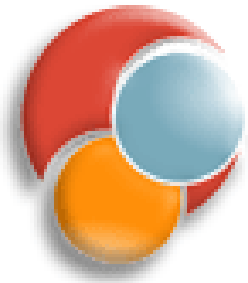


Desarrollo de Aplicaciones para Android

Sesión 5: Multimedia

Puntos a tratar

- Multimedia en Android
- Reproducción de audio
- Reproducción de vídeo
- Fotografía
- Media Store
- Síntesis de voz

Multimedia en Android

- Android es un sistema completamente multimedia
- Capacidad para reproducir/grabar audio/vídeo, toma de fotografías, etc.
- Actualmente el emulador permite tanto reproducción como grabación, así como toma de fotografías por medio de *webcam*

Formatos de audio

- AAC LC/LTP
- HE-AACv1 (AAC+)
- HE-AACv2 (Enhanced AAC+)
- AMR-NB
- AMR-WB
- FLAC
- MP3
- MIDI
- Ogg Vorbis
- PCM Wave

Formatos de vídeo

- H.263
- H.264 AVC
- MPEG-4 SP
- VP8

La clase MediaPlayer

- Reproducción de contenido multimedia
- Orígenes de datos
  - Recursos de la aplicación (sólo audio)
  - Ficheros locales
  - Proveedores de contenido
  - Streaming
- Permite abstraernos del formato y del origen

Audio como recurso de la aplicación

- Carpeta *res/raw*
- Almacenar el fichero sin comprimir
- En el código:

```
R.raw.nombre_fichero
```

Reproducción de audio: inicializar

- Primera forma: método *create()*

```
Context appContext = getApplicationContext();


MediaPlayer recurso = MediaPlayer.create(appContext,
R.raw.fichero_audio);
MediaPlayer fichero = MediaPlayer.create(appContext,
Uri.parse("file:///sdcard/fichero.mp3"));
MediaPlayer url = MediaPlayer.create(appContext,
Uri.parse("http://sitio.com/audio/audio.mp3"));
MediaPlayer contenido = MediaPlayer.create(appContext,
Settings.System.DEFAULT_RINGTONE_URI);
```

Reproducción de audio: inicializar

- Segunda forma: método *setDataSource(string)*

```
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setDataSource("/sdcard/audio.mp3");
mediaPlayer.prepare();
```

Obligatorio en el caso de usar setDataSource

Métodos de MediaPlayer

- Control de la reproducción: *start()*, *stop()*, *pause()*
- Liberar recursos: *release()*
- Reproducción cíclica: *isLooping()*, *setLooping(boolean)*
- Control de la pantalla: *setScreenOnWhilePlaying(boolean)*
- Control de volumen: *setVolume(float, float)*

Métodos de MediaPlayer

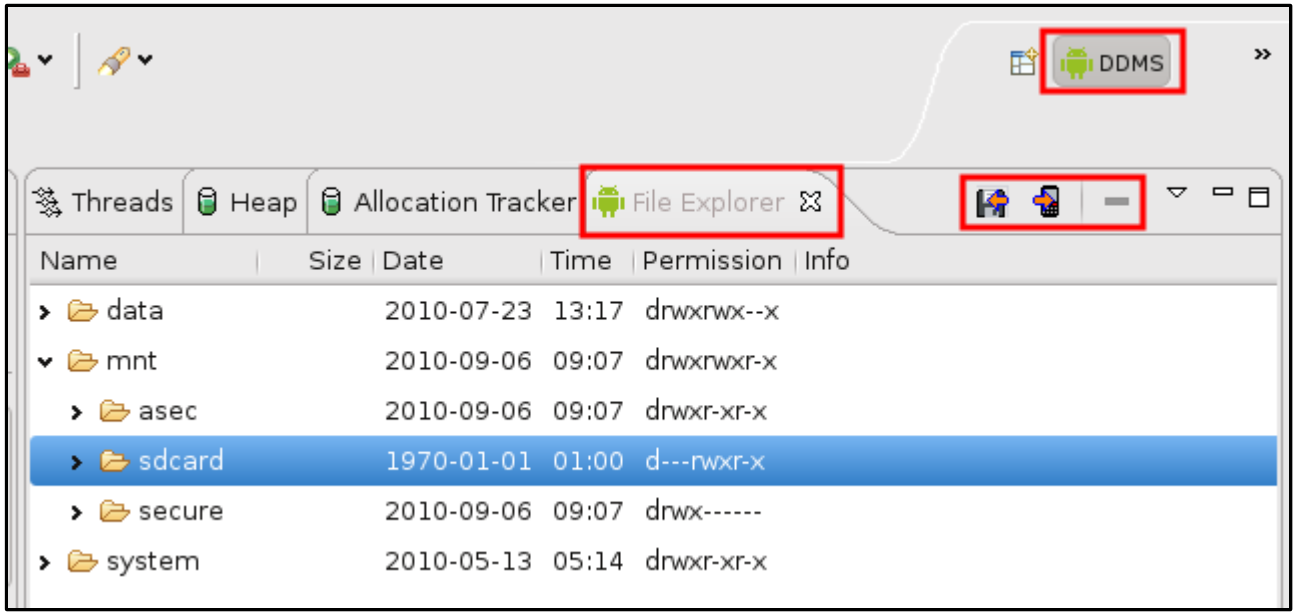
- Posición y duración

```
mediaPLayer.start();

int pos = mediaPlayer.getCurrentPosition();
int duration = mediaPlayer.getDuration();

mediaPlayer.seekTo(pos + (duration-pos)/10);
```

Reproducción de vídeo

- Muy similar a la reproducción de audio, excepto...
  - El vídeo necesita una superficie sobre la que reproducirse
  - No es posible añadir un clip de vídeo como recurso de la aplicación
- Dos formas de reproducir vídeo
  - **VideoView**
  - **MediaPlayer**

Almacenando un fichero en la tarjeta SD



El control Video View

- Encapsula
  - La creación de la superficie donde se reproducirá el vídeo
  - Control de la reproducción del vídeo mediante una instancia de *MediaPlayer*

```
<VideoView android:id="@+id/superficie"
           android:layout_height="fill_parent"
           android:layout_width="fill_parent">
</VideoView>
```

El control Video View

- Asignación de un clip de vídeo y reproducción

```
VideoView videView = (VideoView)findViewById(R.id.superficie);

videoView.setKeepScreenOn(true);
videoView.setVideoPath("/sdcard/ejemplo.3gp");

videoView.start();
// Hacer algo durante la reproducción
videoView.stopPlayback();
```

Vídeo basado en Media Player

- Creación de un objeto *SurfaceView*
- Asignación de la superficie a la instancia de *MediaPlayer*

```
<SurfaceView
            android:id=”@+id/superficie”
            android:layout_height=”wrap_content”
            android:layout_width=”wrap_content”>
            android:layout_gravity=”center”
</SurfaceView>
```

Vídeo basado en Media Player

- La actividad debe implementar la interfaz *SurfaceHolder.Callback*
- El objeto de la clase *MediaPlayer* requiere un *SurfaceHolder* para reproducir el video
- Obtenemos el *SurfaceHolder* en el método *onCreate()* a partir de la superficie de la interfaz

```
SurfaceView superficie = (SurfaceView)findViewById(R.id.superficie);
SurfaceHolder holder = superficie.getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

Vídeo basado en Media Player

- El objeto *SurfaceHolder* es creado de manera asíncrona
- Manejador *surfaceCreated*

```
public void surfaceCreated(SurfaceHolder holder) {
    try {
                mediaPlayer.setDisplay(holder);
    } catch (IllegalArgumentException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
    } catch (IllegalStateException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
    } catch (IOException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
    }
```

Ya es posible usar los métodos *setDataSource()*, *prepare()* y *start()*

Vídeo basado en Media Player

- Completando la implementación de la interfaz *SurfaceHolder.Callback*

```
public void surfaceDestroyed(SurfaceHolder holder) {
   mediaPlayer.release();
}

public void surfaceChanged(SurfaceHolder holder, int
format, int width, int height() {}
```

Toma de fotografías

- Alternativa más sencilla: Intent implícito

```
Intent intent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, TAKE_PICTURE);
```

Toma de fotografías

- Dos modos de funcionamiento
  - **Thumbnail**
    - Por defecto
    - Bitmap devuelto por el Intent en el método onActivityResult
  - **Imagen completa**
    - Especificando una URI en el parámetro extra del Intent
    - Imagen guardada en el destino indicado
    - Thumbnail no devuelto por el Intent

Toma de fotografías

- Modo **thumbnail**

```
private void getThumbnailPicture() {
   Intent intent = new
   Intent(MediaStore.ACTION_IMAGE_CAPTURE);
   startActivityForResult(intent, TAKE_PICTURE);
}
```

Toma de fotografías

- Modo **imagen completa**

```
private void saveFullImage() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File file = new
    File(Environment.getExternalStorageDirectory(), “prueba.jpg”);
    ficheroSalidaUri = Uri.fromFile(file);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, ficheroSalidaUri);
    startActivityForResult(intent, TAKE_PICTURE);
}
```

Toma de fotografías

- Resultado del Intent

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    if (requestCode == TAKE_PICTURE) {
                if (data != null) {
                        Bitmap thumbnail =
                          data.getParcelableExtra("data");
                        // Hacer algo con el thumbnail
                } else {
                        // Hacer algo con la imagen almacenada en
                        // ficheroSalidaUri
                }
    } }
```

Media Store

- Almacena información sobre cualquier fichero
  - Dispositivos externos
  - Memoria interna
- Ficheros multimedia creados por una aplicación no pueden ser accedidos por otra
  - A menos que se incluyan en el *Media Store*

Incluyendo un fichero en el Media Store

- Clase *MediaScannerConnection*
  - Método *scanFile()*
    - Añadir fichero al MediaStore sin necesidad de proporcionar información adicional
    - Asíncrono
    - Requiere una llamada a *connect()*
    - La llamada a *connect()* también es asíncrona
    - Clase *MediaScannerConnectionClient* actúa como notificador

Incluyendo un fichero en el Media Store

```
MediaScannerConnectionClient mediaScannerClient = new
MediaScannerConnectionClient() {
    private MediaScannerConnection msc=null;
    {
                msc = new MediaScannerConnection(getApplicationContext(),
                  this);
                msc.connect();
    }
    public void onMediaScannerConnected() {
                msc.scanFile("/sdcard/test1.jpg", null);
    }
    public void onScanCompleted(String path, Uri uri) {
                msc.disconnect(); // Antes se pueden realizar otras
                                  //acciones
    }
}
```

Sintetizador de voz

- *Text To Speech*
- Incluido desde la versión 1.6 de Android
- A veces las librerías de lenguaje no están instaladas

```
Intent intent = new
Intent(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(intent, TTS_DATA_CHECK);
```

Sintetizador de voz

- Creación de una instancia de la clase *TextToSpeech*
  - Asíncrona
  - Pasar un método *onInit()* como parámetro del constructor

```
TextToSpeech tts;
tts = new TextToSpeech(this, new OnInitListener() {
   pulic void onInit(int status) {
              if (status == TextToSpeech.SUCCESS) {
                     // Hablar
              }
   } });
```

Sintetizador de voz

- Método *speak()*

```
tts.speak(“Hello”, TextToSpeech.QUEUE_ADD, null);
```

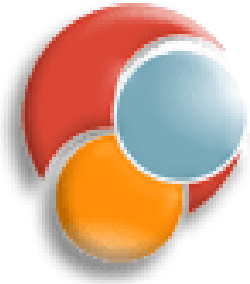
  - **QUEUE_ADD** añade una nueva salida de voz a la cola
  - **QUEUE_FLUSH** sustituye todo lo que hubiera en la cola por el nuevo texto

Otros métodos de TextToSpeech

- *setPitch(float)*: tono de voz
- *setSpeechRate(float)*: velocidad de habla
- *setLanguage(Locale)*: modificar la pronunciación
  - Pasar como parámetro una instancia de la clase *Locale*

```
Locale loc = new Locale("es", "", "");
```

- *stop()*: detiene el motor de síntesis de voz
- *shutdown()*: libera los recursos reservados por *TextToSpeech*



¿Preguntas...?