11-2023

code URJC

Desarrollo de Aplicaciones para Dispositivos Móviles

Tema 3: Desarrollo Híbrido

Tema 3.5: Reproductores multimedia en Flutter

Universidad Rey Juan Carlos

Michel Maes
michel.maes@urjc.es

11-2023

code
URJC

Multimedia

code URJC

- Flutter cuenta con componentes concretos para trabajar con componentes multimedia
- Hay varias opciones depende del reproductor
  - Reproductor de Audio
    - https://pub.dev/packages/audioplayers
    - https://pub.dev/packages/just_audio
  - Reproductor de Video
    - https://pub.dev/packages/video_player
    - https://pub.dev/packages/video_viewer

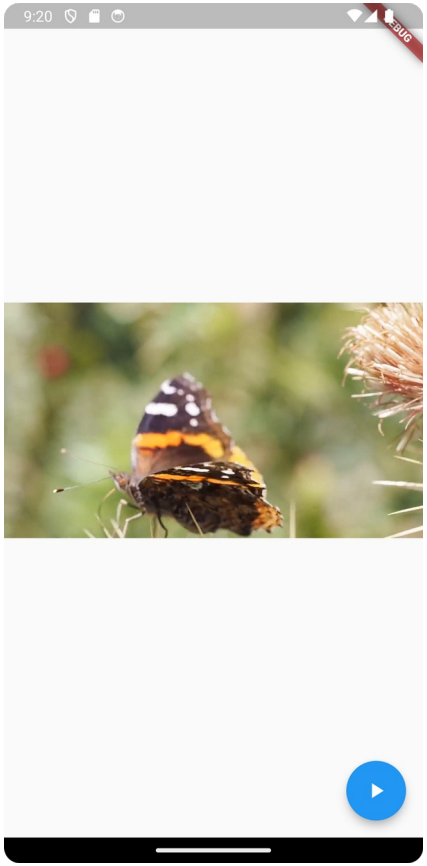
Multimedia: Video

- Veremos un ejemplo sencillo de reproducción de videos con la libreria **video_players**
- Añadimos la dependencia al archivo pubspec.yaml

```
dependencies:
  flutter:
    sdk: flutter
  video_player: 2.8.1
```

- En main.dart comentaremos MusicApp

```
void main() {
  //runApp(const AudioApp());
  runApp(const VideoApp());
}
```



Código: https://github.com/Maes95/DADM/blob/main/flutter_multimedia/lib/VideoPlayer.dart

Multimedia: Video

code URJC

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

Multimedia: Video

code URJC

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

Usaremos un **StatefulWidget** para la app

Multimedia: Video

code URJC

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

Utilizaremos la clase **VideoPlayerController**, que nos permitirá, entre otras cosas, fijar la fuente del fichero de media

Multimedia: Video

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

El video se cargará en segundo plano, por lo que deberemos avisar a la app de que lo muestre cuando lo haya obtenido

Es posible realizar esta acción a través de un **FutureBuilder.** El método **initialize()** devuelve un Future que podremos utilizar en el builder.

Multimedia: Video

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

Podemos hacer que el video se reproduzca en bucle

Multimedia: Video

```dart
class VideoApp extends StatefulWidget {
  const VideoApp({super.key});

  @override
  _VideoAppState createState() => _VideoAppState();
}

class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.networkUrl(Uri.parse(
        'https://flutter.github.io/assets-for-api-docs/assets/videos/butterfly.mp4')
    );

    _controller.initialize().then((_) {
      setState(() {});
    });

    _controller.setLooping(true);
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    super.dispose();
    _controller.dispose();
  }
}
```

En el caso del media, es importante liberar los recursos. La clase **State** cuenta con un método dispose() en el que podremos llamar al dispose del controller

Multimedia: Video

code URJC

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Video Demo',
    home: Scaffold(
      body: Center(
        child: _controller.value.isInitialized
            ? AspectRatio( aspectRatio: _controller.value.aspectRatio, child: VideoPlayer(_controller))
            : const CircularProgressIndicator(),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            _controller.value.isPlaying
                ? _controller.pause()
                : _controller.play();
          });
        },
        child: Icon(
          _controller.value.isPlaying ? Icons.pause : Icons.play_arrow,
        ),
      ),
    ),
  );
}
```

Multimedia: Video

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Video Demo',
    home: Scaffold(
      body: Center(
        child: _controller.value.isInitialized
            ? AspectRatio( aspectRatio: _controller.value.aspectRatio, child: VideoPlayer(_controller))
            : const CircularProgressIndicator(),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            _controller.value.isPlaying
                ? _controller.pause()
                : _controller.play();
          });
        },
        child: Icon(
          _controller.value.isPlaying ? Icons.pause : Ico
        ),
      ),
    ),
  );
}
```

Si el controlador ha cargado el video, mostraremos el elemento **VideoPlayer**, en caso contrario, mostraremos un **ProgressIndicator**

Multimedia: Video

code URJC

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Video Demo',
    home: Scaffold(
      body: Center(
        child: _controller.value.isInitialized
            ? AspectRatio( aspectRatio: _controller.value.aspectRatio, child: VideoPlayer(_controller))
            : const CircularProgressIndicator(),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            _controller.value.isPlaying
                ? _controller.pause()
                : _controller.play();
          });
        },
        child: Icon(
          _controller.value.isPlaying ? Icons.p
        ),
      ),
    ),
  );
}
```

Tendremos un botón que al hacer click comprobará si el video se esta reproduciendo, permitiendonos pausarlo o reanudarlo

Multimedia: Video

code URJC

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Video Demo',
    home: Scaffold(
      body: Center(
        child: _controller.value.isInitialized
            ? AspectRatio( aspectRatio: _controller.value.aspectRatio, child: VideoPlayer(_controller))
            : const CircularProgressIndicator(),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            _controller.value.isPlaying
                ? _controller.pause()
                : _controller.play();
          });
        },
        child: Icon(
          _controller.value.isPlaying ? Icons.pause : Icons.play_arrow,
        ),
      ),
    ),
  );
}
```

Daremos feedback visual al usuario cambiando el icono del botón

Multimedia: Video

code URJC

Altenativa con **FutureBuilder**

```dart
late Future<void> _initializeVideoPlayerFuture;

@override
void initState() {
  ...

  _initializeVideoPlayerFuture = _controller.initialize();

  ...
}
```

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Video Demo',
    home: Scaffold(
        body: Center(
          child: FutureBuilder(
            future: _initializeVideoPlayerFuture,
            builder: (context, snapshot) {
              if (snapshot.connectionState == ConnectionState.done) {
               return AspectRatio(
                  aspectRatio: _controller.value.aspectRatio,
                  child: VideoPlayer(_controller),
                );
              } else {
               return const Center(child: CircularProgressIndicator());
              }
            },
          ),
        ),
      floatingActionButton: FloatingActionButton(...),
    ),
  );
}
```

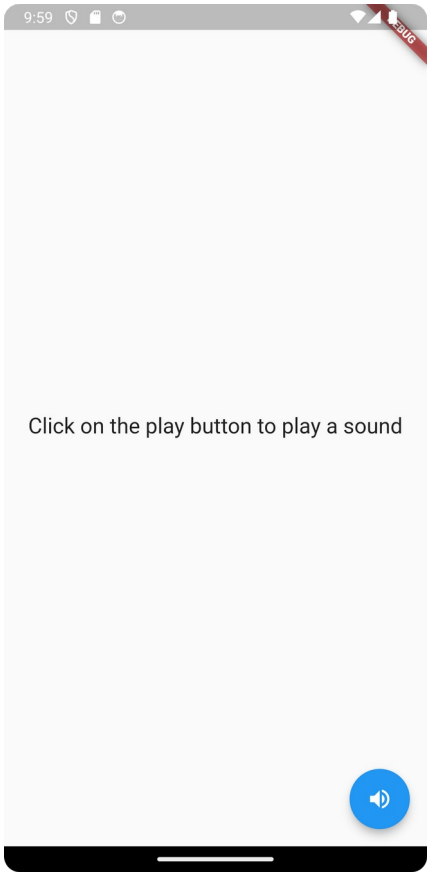
Multimedia: Audio

- Veremos un ejemplo sencillo de reproducción de audio con la libreria **audioplayers**
- Añadimos la dependencia al archivo pubspec.yaml

```
dependencies:
  flutter:
    sdk: flutter
  audioplayers: 5.2.1
```

- En main.dart comentaremos VideoApp

```
void main() {
  runApp(const AudioApp());
  //runApp(const VideoApp());
}
```



Código: https://github.com/Maes95/DADM/blob/main/flutter_multimedia/lib/AudioPlayer.dart

Multimedia: Audio

- Podemos utilizar audios de distintas fuentes
- Si queremos utilizar un audio local, por ejemplo coins.wav situado en la carpeta assets, deberemos añadirlo al pubspec.yaml

```
flutter:
  uses-material-design: true
  assets:
    - assets/
```

Multimedia: Audio

code URJC

```dart
class AudioApp extends StatefulWidget {
  const AudioApp();

  @override
  _AudioAppState createState() => _AudioAppState();
}

class _AudioAppState extends State<AudioApp> {

  final player = AudioPlayer();

  void play() {
    player.play(AssetSource('coins.wav'));
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    player.dispose();
    super.dispose();
  }

}
```

Multimedia: Audio

code URJC

```dart
class AudioApp extends StatefulWidget {
  const AudioApp();

  @override
  _AudioAppState createState() => _AudioAppState();
}

class _AudioAppState extends State<AudioApp> {

  final player = AudioPlayer();

  void play() {
    player.play(AssetSource('coins.wav'));
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    player.dispose();
    super.dispose();
  }

}
```

Usaremos un **StatefulWidget** para la app

Multimedia: Audio

code URJC

```dart
class AudioApp extends StatefulWidget {
  const AudioApp();

  @override
  _AudioAppState createState() => _AudioAppState();
}

class _AudioAppState extends State<AudioApp> {

  final player = AudioPlayer();

  void play() {
    player.play(AssetSource('coins.wav'));
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    player.dispose();
    super.dispose();
  }

}
```

En este caso, utilizaremos **AudioPlayer**. Podemos reutilizar el mismo objeto para reproducir diferentes audios

Multimedia: Audio

code URJC

```dart
class AudioApp extends StatefulWidget {
  const AudioApp();

  @override
  _AudioAppState createState() => _AudioAppState();
}

class _AudioAppState extends State<AudioApp> {

  final player = AudioPlayer();

  void play() {
    player.play(UrlSource('https://luan.xyz/files/audio/coins.wav'));
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    player.dispose();
    super.dispose();
  }

}
```

Podemos obtener el audio desde otras fuentes (vía red por ejemplo)

Multimedia: Audio

code URJC

```dart
class AudioApp extends StatefulWidget {
  const AudioApp();

  @override
  _AudioAppState createState() => _AudioAppState();
}

class _AudioAppState extends State<AudioApp> {

  final player = AudioPlayer();

  void play() {
    player.play(AssetSource('coins.wav'));
  }

  @override
  Widget build(BuildContext context) {...}

  @override
  void dispose() {
    player.dispose();
    super.dispose();
  }

}
```

Al igual que con el video, debemos liberar el objeto

Multimedia: Audio

code URJC

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Audio Demo',
    home: Scaffold(
      body: const Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Click on the play button to play a sound',
              style: TextStyle(fontSize: 20)
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: play,
        tooltip: 'Play',
        child: const Icon(Icons.volume_up),
      ),
    ),
  );
}
```

Multimedia: Audio

code URJC

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Audio Demo',
    home: Scaffold(
      body: const Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Click on the play button to play a sound',
              style: TextStyle(fontSize: 20)
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: play,
        tooltip: 'Play',
        child: const Icon(Icons.volume_up),
      ),
    ),
  );
}
```

En la vista, simplemente hemos añadido un botón que nos permite ejecutar el método play