# Basic Exercises Part 11.4 AVFoundation. Media.

## Introduction to AVKit

* AVKit sits on top of AVFoundation and provides all necessary UI for interacting with a video. AUKit create view-level servies for media playback, complete with user controls, chapter navigation, and support for subtitles and closed captioning.

A screenshot of a cell phone

Description automatically generated

### **1.1 How to play videos using AVPlayerViewController**

(Swift version 5.1).

**AVPlayerViewController** is designed to make it easy to play media just like Apple’s own apps. You can play movies and audio, local and remote, and you benefit from the same user interface layout used elsewhere in the system.

Using the class only takes a few steps, starting with importing the AVKit framework and creating a **URL** that points to the movie you want to play. Once that’s done, you create an **AVPlayer** instance pointing at your URL, send that in to an **AVPlayerViewController**, then call **play()** on it when you’re ready.

Here’s some example code to get you started:

**let** player = AVPlayer(url: videoURL)

**let** vc = AVPlayerViewController()

vc.player = player

present(vc, animated: **true**) {

vc.player?.play()

}

### **1.2 Basic Video Player**

(Overview. Using AVKit and AVFoundation, create a simple video playback app).

You use an audio session to communicate to the system how you intend to use audio in your app. The audio session acts as an intermediary between your app and the operating system—and, in turn, the underlying audio hardware. Configure the audio session to give your app the audio behavior expected of a media playback app.

* Use import AVFoundation to add the AVFoundation framework to the AppDelegate.swift class.
* In the [application(\_:didFinishLaunchingWithOptions:)](https://developer.apple.com/documentation/uikit/uiapplicationdelegate/1622921-application) method, retrieve the shared instance of the audio session and set the app’s audio session category to [playback](https://developer.apple.com/documentation/avfoundation/avaudiosession/category/1616509-playback) and mode to [moviePlayback](https://developer.apple.com/documentation/avfoundation/avaudiosession/mode/1616623-movieplayback).

Add the following:

**import** AVFoundation

**func** application(**\_** application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: **Any**]?) -> Bool {

**let** audioSession = AVAudioSession.sharedInstance()

**do** {

**try** audioSession.setCategory(.playback, mode: .moviePlayback)

}

**catch** {

print("Setting category to AVAudioSessionCategoryPlayback failed.")

}

**return** **true**

}

### **1.3 Set up and configure the user interfacea**

After configuring your app’s audio session, you need to create the user interface for the player.

1. Open the Main.storyboard file. In the Library’s search field, type button to find the Button object.
2. Drag the Button object into the View Controller Scene’s view and give it the title Play Video.
3. Add alignment constraints to center the button both horizontally and vertically.

### **1.4 Implement Playback behavior**

Now that you have created the user interface, it’s time to add the code required to play a video.

1. In the Project Navigator, select the Main.storyboard file and open the assistant editor.
2. Control-drag from the Play Video button to the ViewController.swift class to add a new @IBAction method called playVideo.
3. Close the assistant editor and select the ViewController.swift class in the Project Navigator. Above the class definition, import the AVKit and AVFoundation frameworks.
4. In the playVideo method, add the following implementation:

**import** UIKit

**import** AVKit

**class** ViewController: UIViewController {

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

}

**@IBAction** **func** playVideo(**\_** sender: UIButton) {

**guard** **let** url = URL(string: "https://devstreaming-cdn.apple.com/videos/streaming/examples/bipbop\_adv\_example\_hevc/master.m3u8") **else** {

**return**

}

// Create an AVPlayer, passing it the HTTP Live Streaming URL.

**let** player = AVPlayer(url: url)

// Create a new AVPlayerViewController and pass it a reference to the player.

**let** controller = AVPlayerViewController()

controller.player = player

// Modally present the player and call the player's play() method when complete.

present(controller, animated: **true**) {

player.play()

}

}

}

### **2.1 How to play sounds using AVAudioPlayer**

(Swift version 5.1).

The most common way to play a sound on iOS is using **AVAudioPlayer**, and it's popular for a reason: it's easy to use, you can stop it whenever you want, and you can adjust its volume as often as you need. The only real catch is that you must store your player as a property or other variable that won't get destroyed straight away – if you don't, the sound will stop immediately.

**AVAudioPlayer** is part of the AVFoundation framework, so you'll need to import that:

**import** AVFoundation

Like I said, you need to store your audio player as a property somewhere so it is retained while the sound is playing. In our example we're going to play a bomb explosion sound, so I created a property for it like this:

**var** soundEffect: AVAudioPlayer?

With those two lines of code inserted, all you need to do is play your audio file. This is done first by finding where the sound is in your project using **path(forResource:)**, then creating a file URL out of it. That can then get passed to **AVAudioPlayer** to create an audio player object, at which point – finally – you can play the sound. Here's the code:

**let** path = Bundle.main.path(forResource: "example.mp3", ofType:**nil**)!

**let** url = URL(fileURLWithPath: path)

**do** {

bombSoundEffect = **try** AVAudioPlayer(contentsOf: url)

bombSoundEffect?.play()

} **catch** {

// couldn't load file :(

}

If you want to stop the sound, you should use its **stop()** method.

### **2.2 Exercise**

Add a video and an audio file to the simulator and try to play both, a local video and an audio file (mp3).