Practical 08: State Archiving When Moving Into/From Background

In this lab, we will examine the MovieApp and see how it has been modified to be able to save the state the application when it enters to the background and load the application states when it launches.

Section 1: App State and NSCoding

1. Using Xcode, Open the project **MovieApp** that you did in Practical 7.
2. Run the application in your iPhone Simulator. Navigate to the Movie Detail screen. Quit the application by pressing the Home button (Shift-Cmd-H) twice and swiping the application up. Open the application again, and you will see that you will be brought back to the list of movies screen. In this practical we want to remember that previous screen the user was at and bring him/her back to that screen even after the application was closed.
3. Create a new **AppState.swift** file. We need this new class to store the current state of the application, including which screen the user is currently at, and what fields are currently keyed into that screen. Ensure that your **AppState** class implements the **NSCoding** interface.

```
import UIKit

// This stores information about the current user interface
// state of the application.
// For example, which screen the user is currently at.
// What are the fields currently typed into the text fields,
// etc.
//
class AppState: NSObject, NSCoding {

    var currentScreen : String = "ViewController"
    var currentMovieID : String = ""
    var currentMovieName : String = ""
    var currentMovieDescription : String = ""
    var currentMovieRuntime : String = ""
    var currentMovieImageName : String = ""


    // Empty initializer, used for creating an empty AppState
    // object.
    //
    override init()
    {
    }
}
```

4. Now we need to implement the required methods. Let's start with **encode (aCoder)**. When a Movie sends the message **encode**, it will encode all its instance variable into the NSCoder object that is passed as an argument.
5. In the **AppState.swift**, implement the **encode (aCoder:)** to add the instance variable to the container.

```
class AppState: NSObject, NSCoding {

    var currentScreen : String = "ViewController"
```

```
    var currentMovieID : String = ""
    var currentMovieName : String = ""
    var currentMovieDescription : String = ""
    var currentMovieRuntime : String = ""
    var currentMovieImageName : String = ""

    ...

    // This function simply 'saves' the values inside the above
    // variables (movieName, movieDesc, runtime, imageName)
    // into the aCoder. We have to do this manually field-by-field
    // because it is not possible for Swift to know in-advanced,
    // what fields what you want to save when the app goes into
    // the background.
    //
    func encode (with aCoder: NSCoder)
    {
        aCoder.encode(currentScreen, forKey: "currentScreen")
        aCoder.encode(currentMovieID, forKey:"currentMovieID")
        aCoder.encode(currentMovieName, forKey: "currentMovieName")
        aCoder.encode(currentMovieDescription, forKey:
"currentMovieDescription")
        aCoder.encode(currentMovieRuntime, forKey:
"currentMovieRuntime")
        aCoder.encode(currentMovieImageName, forKey:
"currentMovieImageName")
    }
}
```

6. Next, objects are loaded from an archive will be initialized using **init (aDecoder:)**. This method should retrieve all the objects that were encoded in **encode** and assign them to the instance variables.

```
class AppState: NSObject, NSCoding {

    var currentScreen : String = "ViewController"
    var currentMovieID : String = ""
    var currentMovieName : String = ""
    var currentMovieDescription : String = ""
    var currentMovieRuntime : String = ""
    var currentMovieImageName : String = ""

    ...


    // This function simply 'retrieves' the values into the above
    // variables (movieName, movieDesc, runtime, imageName)
    // from the aCoder. This is the reverse of the encodeWithCoder
    // function.
    //
    required init?(coder aDecoder: NSCoder)
    {
        currentScreen = aDecoder.decodeObject(forKey: "currentScreen")
as! String
        currentMovieID = aDecoder.decodeObject(forKey:
"currentMovieID") as! String
        currentMovieName = aDecoder.decodeObject(forKey:
"currentMovieName") as! String
        currentMovieDescription = aDecoder.decodeObject(forKey:
"currentMovieDescription") as! String
        currentMovieRuntime = aDecoder.decodeObject(forKey:
```

```
"currentMovieRuntime") as! String
        currentMovieImageName = aDecoder.decodeObject(forKey:
"currentMovieImageName") as! String

        super.init()
    }
}
```

7. Next, update our **MovieDetailViewController.swift**, so that we are able to restore the state of the user interface if the application was restarted from an inactive state. To do this, let's create a new variable **appState** that is an object of the AppState class.

```
class MovieDetailViewController: UIViewController
{
    @IBOutlet weak var imageNameTextField : UITextField!
    @IBOutlet weak var movieIDTextField : UITextField!
    @IBOutlet weak var movieNameTextField : UITextField!
    @IBOutlet weak var runtimeField : UITextField!
    @IBOutlet weak var descriptionTextField : UITextField!

    var movieItem : Movie?

    var appState : AppState?

    ...
}
```

8. In the **MovieDetailViewController.swift**, we want to modify the viewWillAppear function so that we can restore the state of the user interface from the appState object.

```
class MovieDetailViewController: UIViewController
{
    ...
    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)

        if movieItem != nil
        {
            // Assign the fields from the movie object
            // to all the text fields on the screen.
            //
            movieIDTextField.text = movieItem.movieID ?? ""
            movieNameTextField.text = movieItem.movieName ?? ""
            descriptionTextField.text = movieItem.movieDesc ?? ""
            imageNameTextField.text = movieItem.imagePath ?? ""
            runtimeField.text = "\(movieItem.runtime)"

            // If this is an existing movie (that means the movie
            // ID will have some value), disable the user from
            // modifying the movie ID.
            //
            if movieItem.movieID != ""
            {
                movieIDTextField.enabled = false
```

```
            }

            self.navigationItem.title = movieItem.movieName ?? ""
        }
        else if appState != nil
        {
            // This is where we actually restore all the values
            // saved in the appState previously into the
            // individual text fields here.
            //
            // This else-if clause will only be executed the first
            // time when the app is restarted from an inactive state,
            // and if the app was previously closed when the user
            // was in the Movie Detail screen.
            //
            movieIDTextField.text = appState!.currentMovieID
            movieNameTextField.text = appState!.currentMovieName
            descriptionTextField.text =
appState!.currentMovieDescription
            imageNameTextField.text = appState!.currentMovieImageName
            runtimeField.text = appState!.currentMovieRuntime

            // Create an empty movie record. Actually, it doesn't
            // need to contain any data, but we will just fill it in
            // for completeness sake.
            //
            movieItem = Movie(id: appState!.currentMovieID,
                name: appState!.currentMovieName,
                description: appState!.currentMovieDescription,
                runTime: 0,
                imagePath: appState!.currentMovieImageName)
         }
    }

    ...
}
```

Section 2: Archiving and Restoring App State

9. In the **AppDelegate.swift**, we are going to implement a itemArchivePath method to construct the file path so that the movie items are saved to a single file in the Document Directory.

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    var movieList : [Movie] = []

    func createMovieList()
    {

    }

    // This constructs the path (including the filename)
    // to which we save our app data.
    func itemArchivePath() -> String
    {
```

```
        let documentDirectories =
            NSSearchPathForDirectoriesInDomains(
            .documentDirectory, .userDomainMask, true)

        let documentDirectory = documentDirectories[0]
        return (documentDirectory as NSString)
            .appendingPathComponent("items.archive")
    }
}
```

10. In the AppDelegate.swift, implement the following function to retrieve the currently showing view controller.

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    ...

    // This function finds and returns the current visible view controller
    //
    func currentVisibleViewController() -> UIViewController!
    {
        var topController : UIViewController! = nil
        var nextTopController : UIViewController! = nil

        if let rootController =
UIApplication.shared.keyWindow?.rootViewController
        {
            nextTopController = rootController

            while nextTopController != nil
            {
                topController = nextTopController
                if (nextTopController is UINavigationController)
                {
                    nextTopController = (nextTopController as!
UINavigationController).topViewController
                }
                else
                {
                    nextTopController =
nextTopController.presentedViewController
                }
            }
        }
        return topController
    }
}
```

11. When the user presses the home button on the device, the function **applicationDidEnterBackground** will be triggered. At this point, we will want to find out which is the screen the user is currently in. If the screen is currently showing the list of movies, then we indicate this in the appState object. If the screen is currently showing the movie form, then we

indicate this in the appState object, and additionally save all the data in the fields into the appState object.

In the **AppDelegate.swift**, implement the following codes:

```
    func applicationDidEnterBackground(_ application: UIApplication)
    {
        // If the application enters background, then we
        // save our application data into the archive. If at
        // any point the application is terminated, at least
        // the state has been saved, and can be restored later
        // when the application is started again.

        let appState = AppState()
        let curViewController = currentVisibleViewController()
        if (curViewController is ViewController)
        {
            // The user closes the app into the background
            // when he/she was in the list of movies screen.
            //
            appState.currentScreen = "ViewController"
        }
        else if (curViewController is MovieDetailViewController)
        {
            // The user closes the app into the background
            // when he/she was in the Movie Detail screen.
            //
            appState.currentScreen = "MovieDetailViewController"

            // So we basically save the everything that the user
            // has typed into the text fields into our appState
            // object.
            //
            let movieViewController = (curViewController as!
MovieDetailViewController)
appState.currentMovieID = movieViewController.movieIDTextField.text!
            appState.currentMovieName =
movieViewController.movieNameTextField.text!
            appState.currentMovieDescription =
movieViewController.descriptionTextField.text!
            appState.currentMovieImageName =
movieViewController.imageNameTextField.text!
            appState.currentMovieRuntime =
movieViewController.runtimeField.text!
        }

        // Here, we save the app state using NSKeyedArchiver
        //
        NSKeyedArchiver.archiveRootObject(appState, toFile:
itemArchivePath())
    }
```

12. Next, in the AppDelegate.swift file, implement the following codes to load the state from the archive when the application starts. And then do whatever necessary to bring the user back to the previous screen that he/she was in.

```
    func applicationDidBecomeActive(_ application: UIApplication)
    {
```

```
        // Load up the application state from the archive
        // using NSKeyedUnarchiver.
        //
        let obj = NSKeyedUnarchiver
            .unarchiveObject(withFile: itemArchivePath())
        if obj == nil
        {
            return
        }
        let appState = obj as! AppState

        if appState.currentScreen == "ViewController"
        {
            // The app was previously in the list of movies screen
            // when it was moved into the background.
            //
            // And we are already in the list of movies screen
            // because the app has been restarted,
            // so we don't need to do anything here.
        }
        else if appState.currentScreen == "MovieDetailViewController"
        {
            // The app was previously in the movie detail screen when
            // it was moved into the background.
            //
            let rootViewController =
UIApplication.shared.keyWindow?.rootViewController
            let curViewController = currentVisibleViewController()
            var movieDetailViewController : MovieDetailViewController!
= nil;

            // Check that if the app is currently in the list of movies
            // screen (ViewController), then we need to push the
            // MovieDetailViewController up, and update
            // the text fields.
            //
            if (curViewController is ViewController)
            {
                // Currently we are showing the main screen. So we
                // ask iOS to load up the
                // MovieDetailViewController screen.
                //
                let sb = UIStoryboard(name: "Main", bundle: nil)
                movieDetailViewController =
sb.instantiateViewController(withIdentifier:
"MovieDetailViewController") as! MovieDetailViewController

                // And write this line to show
                // the MovieDetailViewController
                // (we have to use pushViewController to achieve
                // that because we are using the Navigation View
                // Controller)
                //
                (rootViewController as!
UINavigationController).pushViewController(movieDetailViewController,
animated: false)

                // We set the app state to it.
                //
                movieDetailViewController.appState = appState
            }
        }
    }
}
```

13. Build and run the application. Click on any of the movies, or click on the + sign to add a new movie. In the movie detail screen, type in anything in all the fields. Quit the application by click on the Home button (Shift+Home+H) twice, and swiping the application up. Then restart the application again to see that you should go back to the movie detail screen.