**Topic: The Application Delegate and the Split View** **Controller**

Intro

* We aren’t doing a quiz or lab on this topic, but we’ll discuss the concepts
* We should go back and look at the structure of iOS apps a little more closely. See: [iOS Dev Lib: The Structure of an App](https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html)
* The Split View architecture for the Master-Detail app in *Beginning iPhone Development with Swift*, Ch. 11 makes greater use of the AppDelegate than our previous apps, so we need to understand the role of the AppDelegate in iOS apps in general.

Delegation Pattern

In the Introduction to Gamma et al. 1994, Grady Booch defined delegation as:

**Delegation** is a way to make composition as powerful for reuse as inheritance [Lie86, JZ91]. In delegation, *two* objects are involved in handling a request: a receiving object delegates operations to its **delegate**. This is analogous to subclasses deferring requests to parent classes. But with inheritance, an inherited operation can always refer to the receiving object through the this member variable in C++ and self in Smalltalk. To achieve the same effect with delegation, the receiver passes itself to the delegate to let the delegated operation refer to the receiver.

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1995). *Design patterns : elements of reusable object-oriented software* (14. print. ed.). Reading, Mass.: Addison-Wesley. p. 20. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-201-63361-2](https://en.wikipedia.org/wiki/Special:BookSources/0-201-63361-2).

AppDelegate in an iOS App

“The app delegate is the heart of your custom code. This object works in tandem with the UIApplication object to handle app initialization, state transitions, and many high-level app events. This object is also the only one guaranteed to be present in every app, so it is often used to set up the app’s initial data structures.” --from The Structure of an App.

For a detailed description of the Delegate working with a UITableView see: <https://www.raywenderlich.com/46988/ios-design-patterns>

Code in the AppDelegate

* Code in FinishedLaunching( … )
  + This method appears to be the Xamarin.iOS, UIApplicationDelegate, equivalent of didFinishLaunchingWithOptions
  + Set the navigation button.
    - The RootViewController is the SplitViewController. We get a reference to this so that we can get a reference to the (master?) NavigationViewController. This is from an array on the SplitViewController. Then we use that to set a button from the SplitViewController on the NavigationViewController.
    - This can’t be done with a segue because it has to be done dynamically on an object.
    - See what happens if you comment out this code.
  + Give the SplitViewController a reference to the AppDelegate. Why? What is a weakDelegate?

Master-Detail template project

* Split View Controller: Only included if it’s an iPad or iPad & iPhone app.
* Navigation Controller: For the master view, which is its root view controller (?)
* Navigation Controller: for the detail view, which is its root view controller (?)
* Table View Controller: For the master view.
* View Controller: For the detail view

Master View Controller

* Code from the template
  + Check to see if the device is an iPad. If it is set:
    - Don’t deselect rows each time the table is displayed (or redisplayed).
    - Set the size of the View (has no effect in iOS 8)
    - This is done in the constructor. (In XCode it’s done in awakeFromNib.)
  + Add the + button to the toolbar
    - ViewDidLoad is where this is done
  + PrepareForSegue
    - This View Controller only has one segue, but for some reason we need to check the identifier to make sure it’s the right one. Why?
    - Get a reference to the destination ViewController from the segue parameter. (This is actually done on the third line)
    - Take care of sending the data
      * Get the data from the selected row in the table.
      * Call a method that we had already added to the destination ViewController. This sets the data that we are passing.
    - Add the button for navigating back to the master view.
* Code we are adding
  + Load the president data from the plist in ViewDidLoad

Detail View Controller