



BUILDING YOUR FIRST IOS APP

Nick Chen
Fall 2015
talentspark.io

APP LIFECYCLE (SIMPLIFIED)

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

APP LIFECYCLE (SIMPLIFIED)

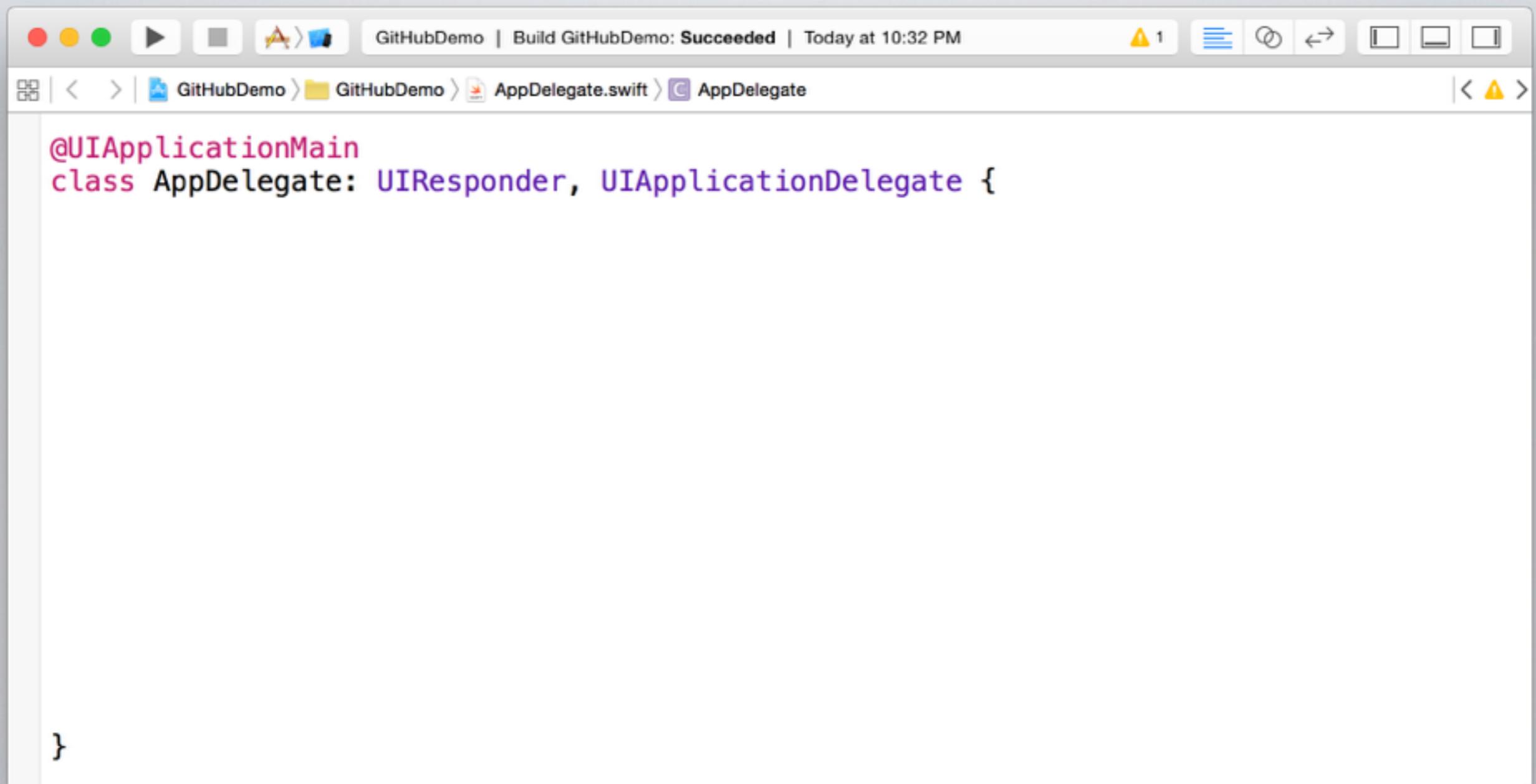
```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

As usual, main is the entry point...

APP LIFECYCLE (SIMPLIFIED)

APP LIFECYCLE (SIMPLIFIED)



A screenshot of the Xcode IDE interface. The title bar shows the project name "GitHubDemo" and build status "Build GitHubDemo: Succeeded | Today at 10:32 PM". The toolbar icons include standard Mac OS X window controls and Xcode specific symbols like a play button and a target icon. The navigation bar below the title bar shows the file structure: GitHubDemo > GitHubDemo > AppDelegate.swift > AppDelegate. The main editor area displays the following Swift code:

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

}
```

APP LIFECYCLE (SIMPLIFIED)

In Swift, we use
@UIApplicationMain

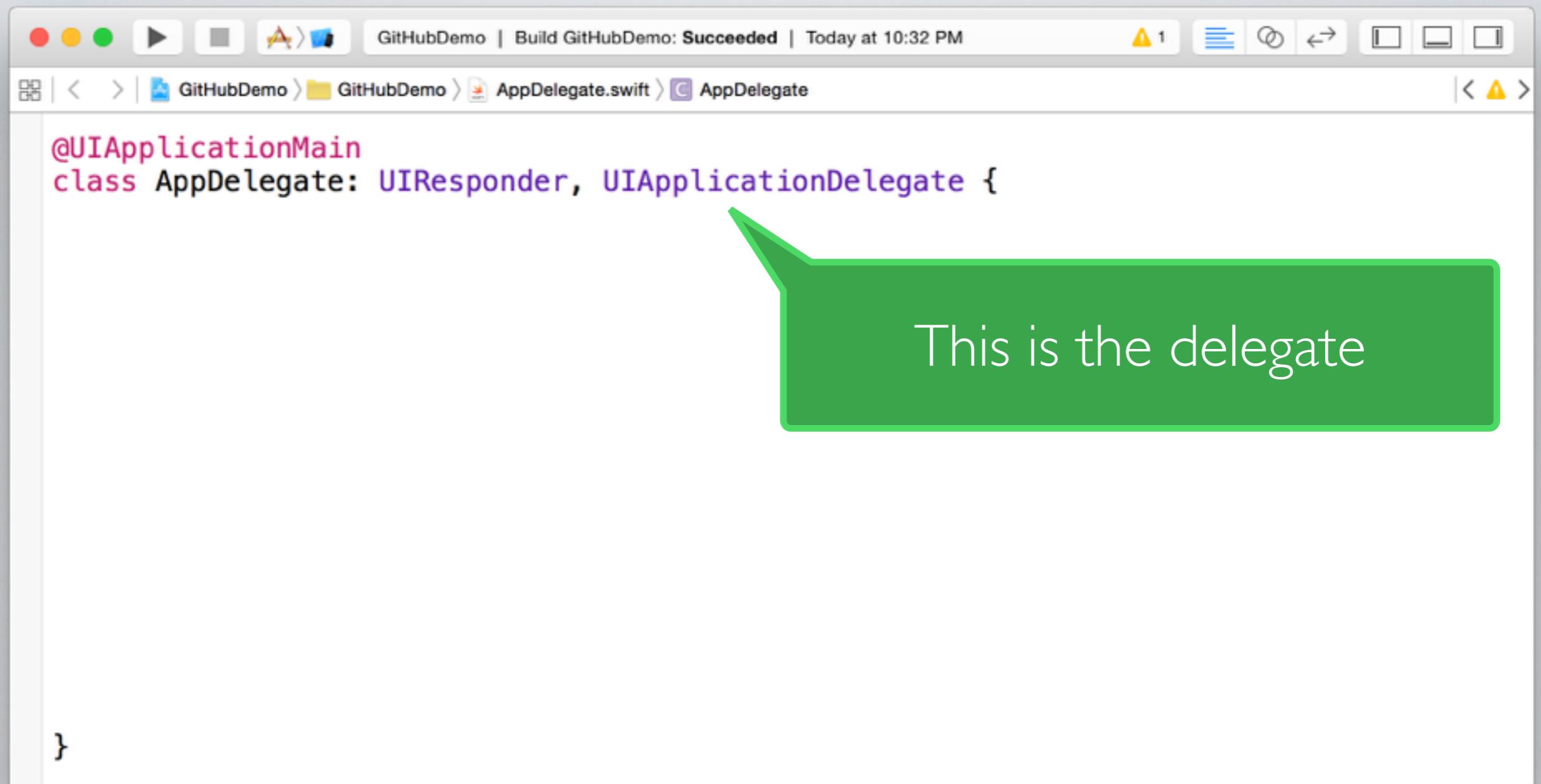


```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

}

}
```

APP LIFECYCLE (SIMPLIFIED)



A screenshot of the Xcode IDE showing the file `AppDelegate.swift`. The code defines a class `AppDelegate` that conforms to `UIResponder` and `UIApplicationDelegate`. A green callout bubble points from the word `UIApplicationDelegate` to the text "This is the delegate".

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

}

}
```

This is the delegate

APP LIFECYCLE (SIMPLIFIED)

The screenshot shows the Xcode documentation interface. The search bar at the top contains the text "uiappdelegate". Below it, a sidebar lists several search results, with "UIAppDelegate" being the most prominent. The main content area is titled "UIAppDelegate Protocol Reference". A red box highlights a paragraph describing the protocol's purpose:

The `UIApplicationDelegate` protocol defines methods that are called by the singleton `UIApplication` object in response to important events in the lifetime of your app. The app delegate works alongside the app object to ensure your app interacts properly with the system and with other apps. Specifically, the methods of the app delegate give you a chance to respond to important changes. For example, you use the methods of the app delegate to respond to state transitions, such as when your app moves from foreground to background execution, and to respond to incoming notifications. In many cases, the methods of the app delegate are the only way to receive these important notifications.

Below this, another paragraph explains Xcode's handling of the app delegate:

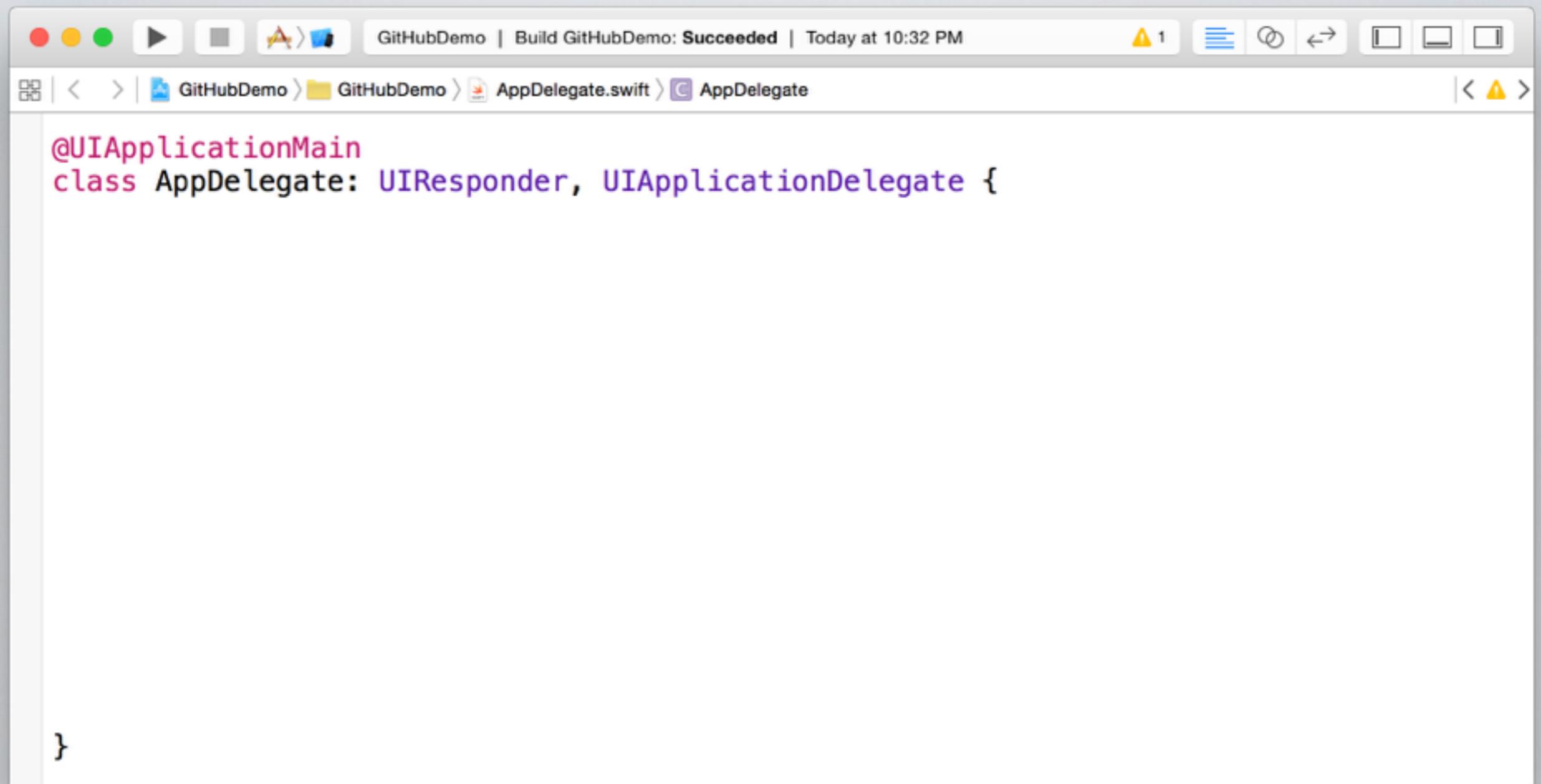
Xcode provides an app delegate class for every new project, so you do not need to define one yourself initially. When your app launches, UIKit automatically creates an instance of the app delegate class provided by Xcode and uses it to execute the first bits of custom code in your app. All you have to do is take the class that Xcode provides and add your custom code.

At the bottom, another paragraph states:

The app delegate is effectively the root object of your app. Like the `UIApplication` object itself, the app delegate is a singleton object and



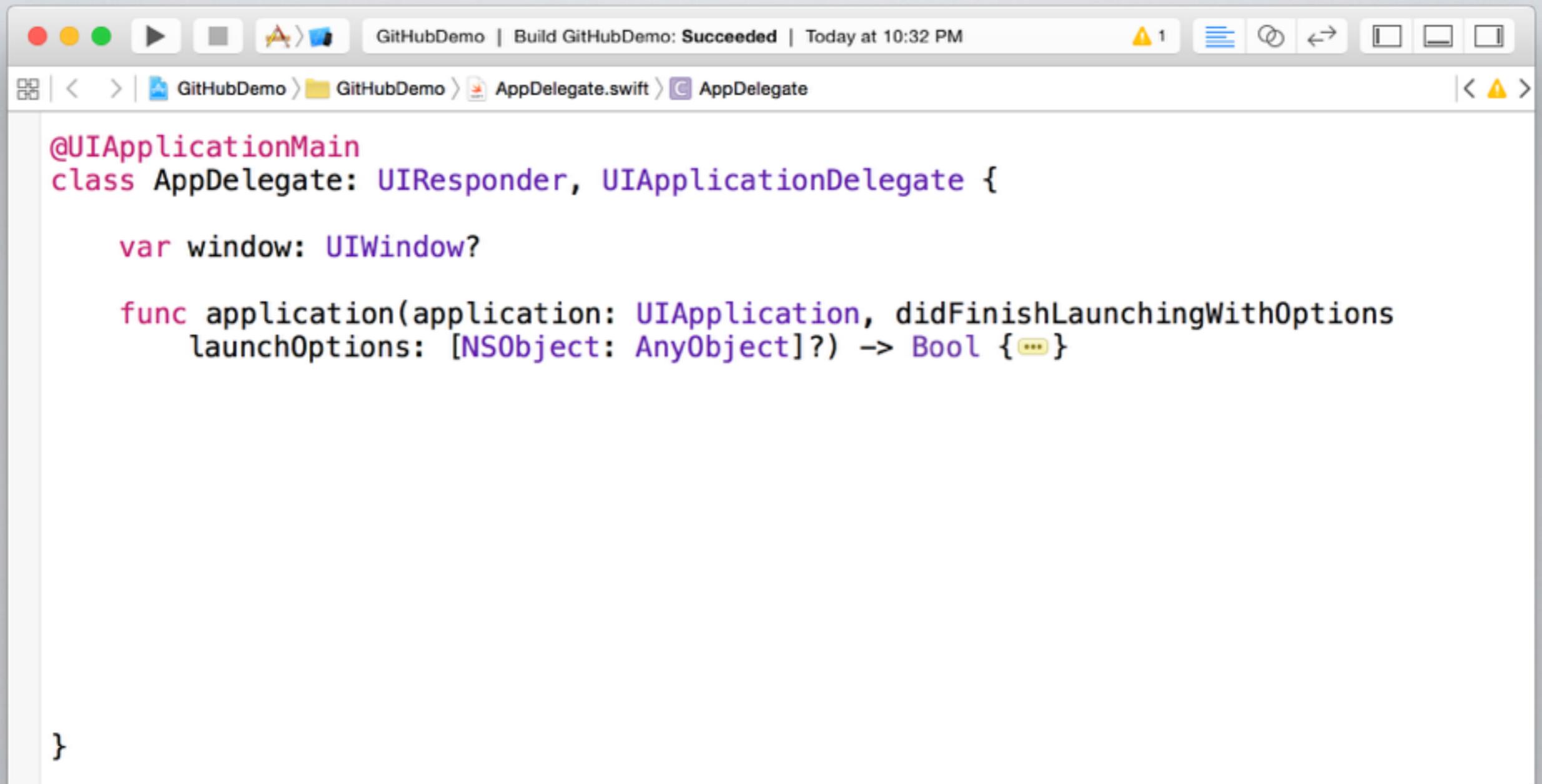
APP LIFECYCLE (SIMPLIFIED)



A screenshot of the Xcode IDE showing the AppDelegate.swift file. The window title bar displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 10:32 PM". The toolbar icons include standard Mac OS X controls and Xcode specific ones like build and run. The navigation bar shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift. The main editor area contains the following Swift code:

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

APP LIFECYCLE (SIMPLIFIED)



A screenshot of the Xcode IDE showing the AppDelegate.swift file. The window title bar indicates "GitHubDemo | Build GitHubDemo: Succeeded | Today at 10:32 PM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > AppDelegate.swift". The code itself is:

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

}
```

APP LIFECYCLE (SIMPLIFIED)

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

    var window: UIWindow?

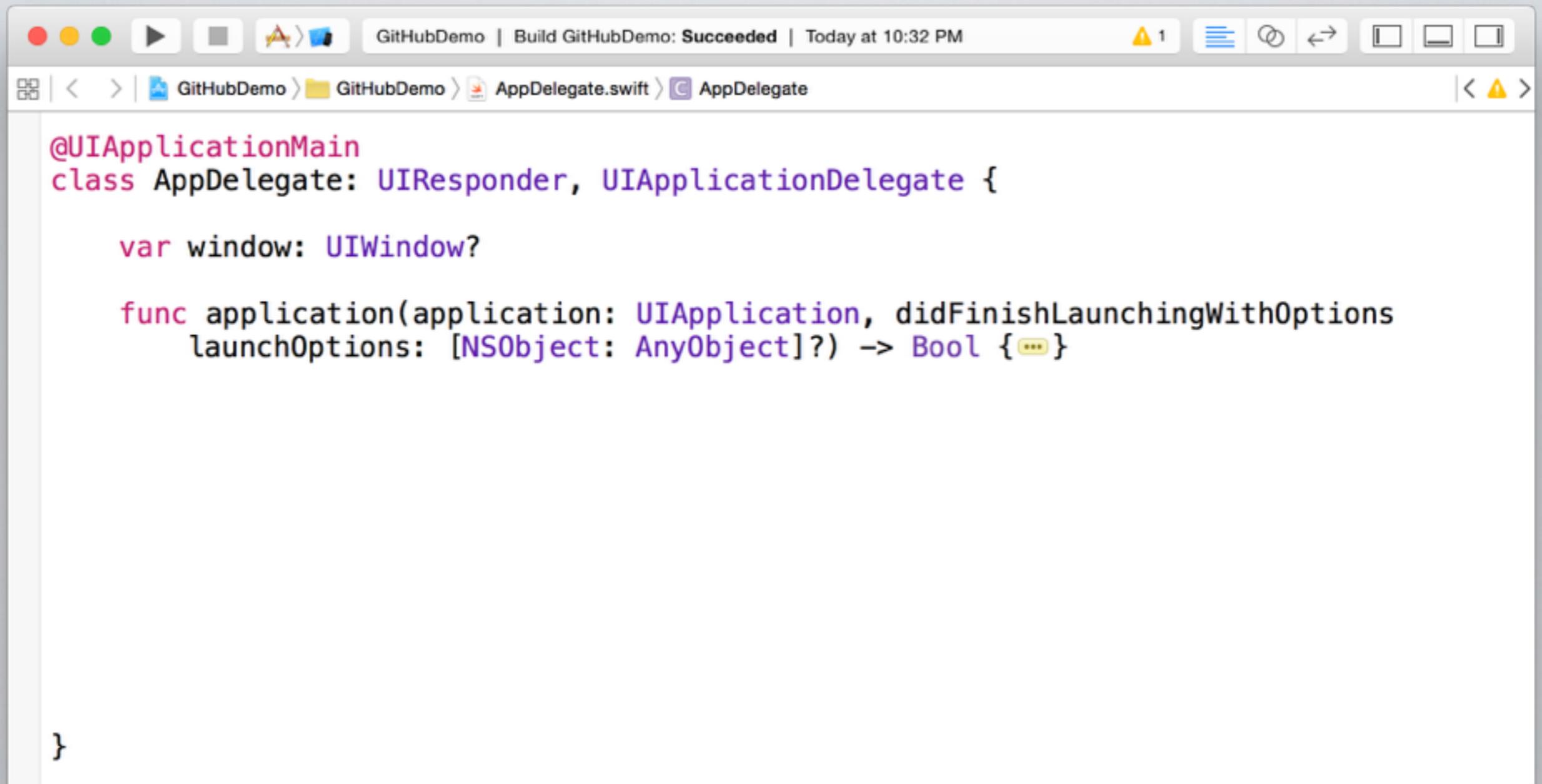
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

}


```

Called upon launching

APP LIFECYCLE (SIMPLIFIED)



A screenshot of the Xcode IDE showing the AppDelegate.swift file. The window title bar indicates "GitHubDemo | Build GitHubDemo: Succeeded | Today at 10:32 PM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > AppDelegate.swift". The code itself is:

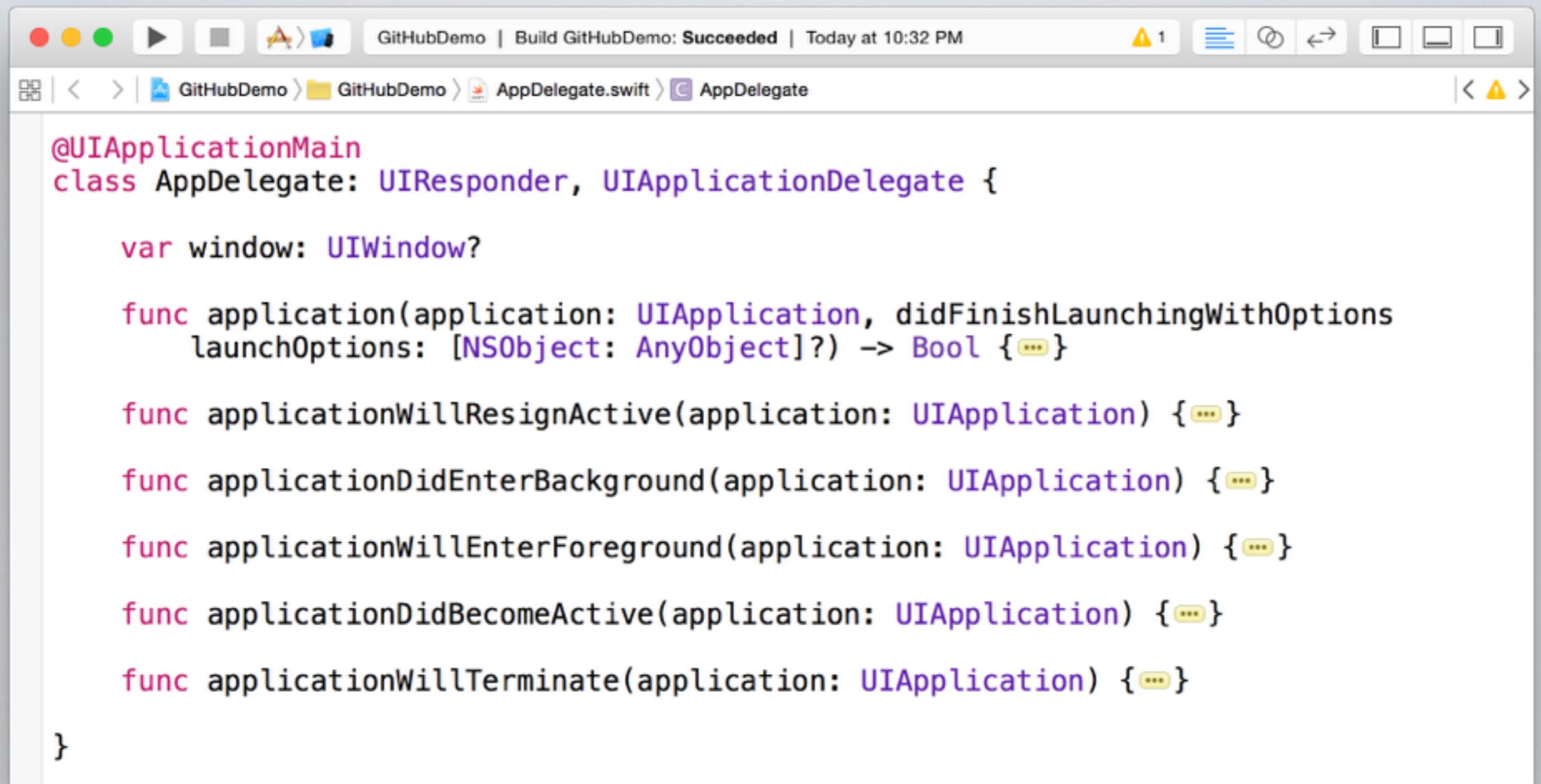
```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

}
```

APP LIFECYCLE (SIMPLIFIED)



The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** GitHubDemo | Build GitHubDemo: Succeeded | Today at 10:32 PM
- Document Outline:** GitHubDemo > GitHubDemo > AppDelegate.swift > AppDelegate
- Code Editor:** Displays the following Swift code for the AppDelegate class:

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

    func applicationWillResignActive(application: UIApplication) { ... }

    func applicationDidEnterBackground(application: UIApplication) { ... }

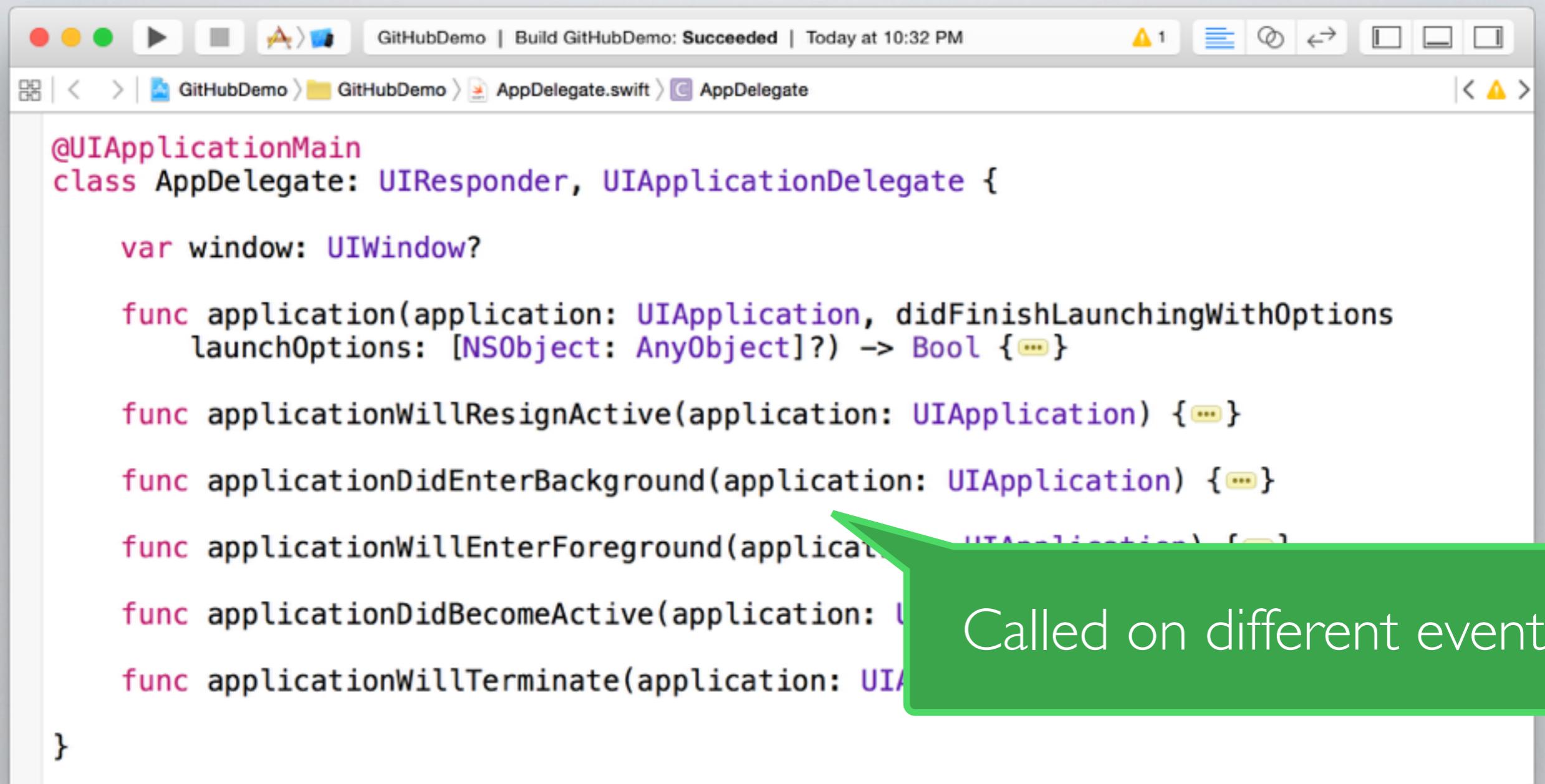
    func applicationWillEnterForeground(application: UIApplication) { ... }

    func applicationDidBecomeActive(application: UIApplication) { ... }

    func applicationWillTerminate(application: UIApplication) { ... }

}
```

APP LIFECYCLE (SIMPLIFIED)



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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

    func applicationWillResignActive(application: UIApplication) { ... }

    func applicationDidEnterBackground(application: UIApplication) { ... }

    func applicationWillEnterForeground(application: UIApplication) { ... }

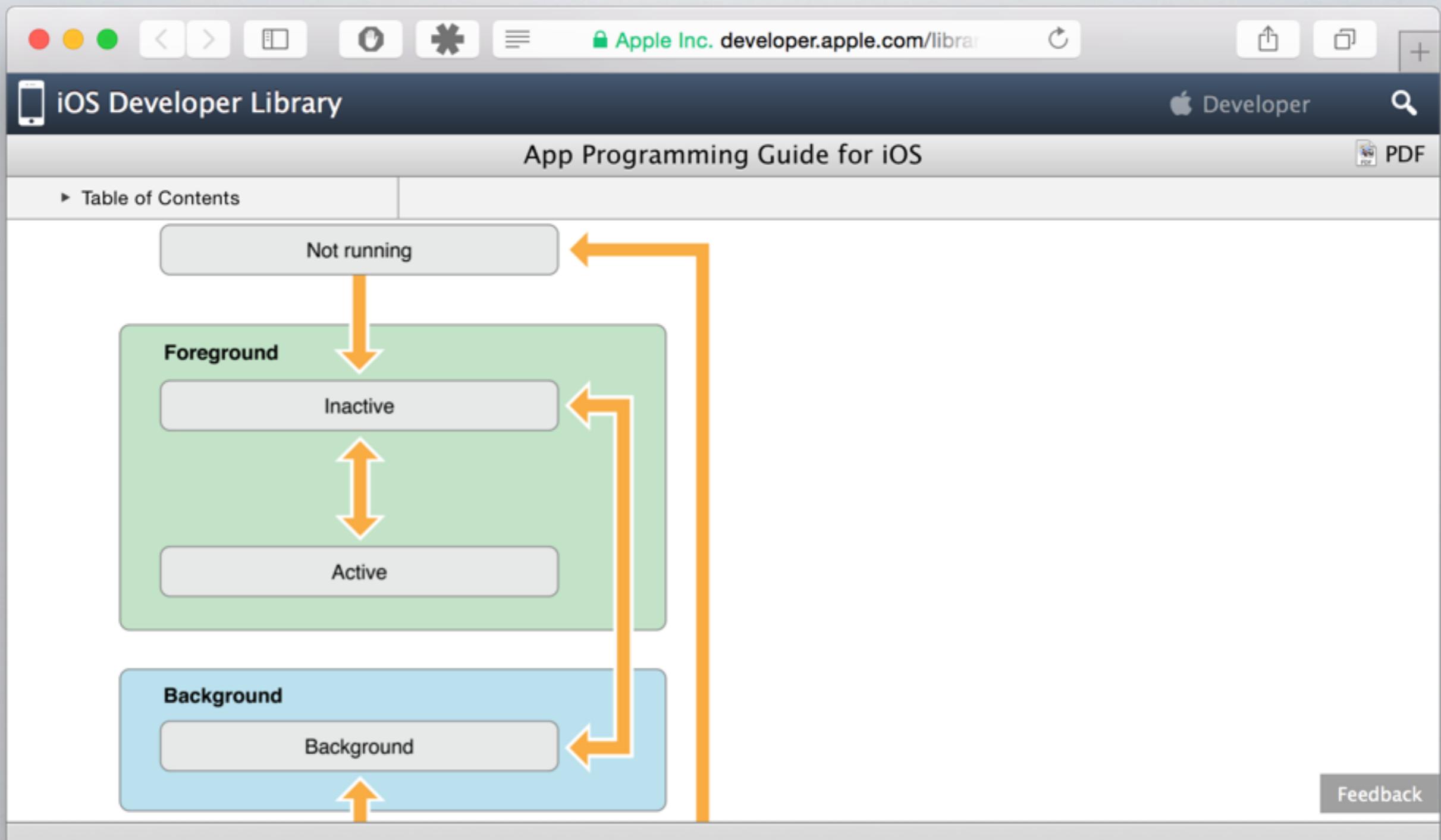
    func applicationDidBecomeActive(application: UIApplication) { ... }

    func applicationWillTerminate(application: UIApplication) { ... }

}
```

Called on different events

APP LIFECYCLE (SIMPLIFIED)



DEMO

The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for running, stopping, and navigating.
- Status Bar:** Shows "Finished running GitHubDemo on iPhone 6".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the `AppDelegate.swift` file content. The code is annotated with blue arrows pointing to specific methods: application, applicationWillResignActive, applicationDidEnterBackground, applicationWillEnterForeground, applicationDidBecomeActive, and applicationWillTerminate.

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

    func applicationWillResignActive(application: UIApplication) { ... }

    func applicationDidEnterBackground(application: UIApplication) { ... }

    func applicationWillEnterForeground(application: UIApplication) { ... }

    func applicationDidBecomeActive(application: UIApplication) { ... }

    func applicationWillTerminate(application: UIApplication) { ... }

}
```

DEMO



Finished running GitHubDemo on iPhone 6

GitHubDemo > GitHubDemo > AppDelegate.swift > applicationDidBecomeActive(_:) (M)

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool { ... }

    func applicationWillResignActive(application: UIApplication) { ... }

    func applicationDidEnterBackground(application: UIApplication) { ... }

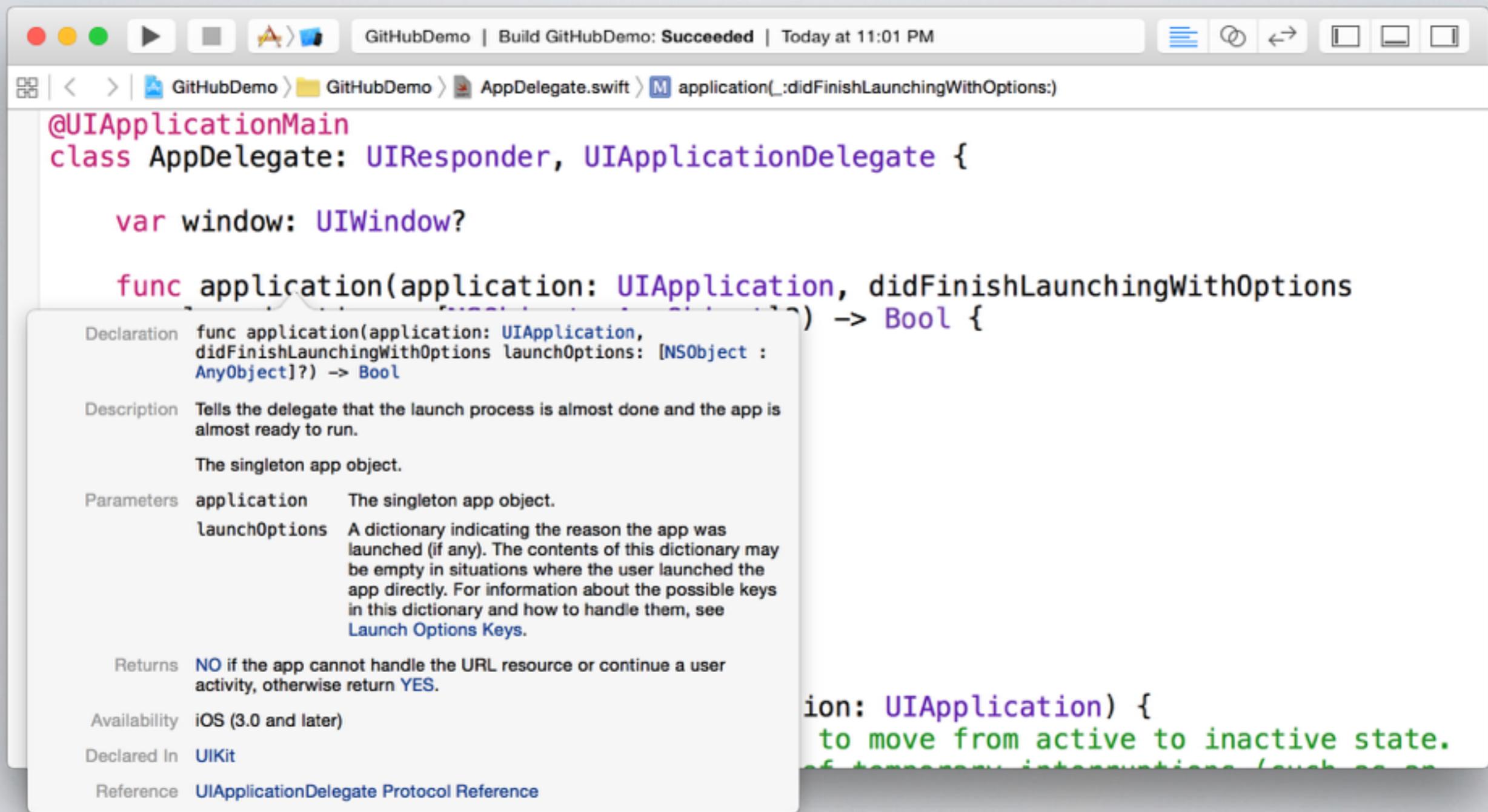
    func applicationWillEnterForeground(application: UIApplication) { ... }

    func applicationWillTerminate(application: UIApplication) { ... }

}
```

Set breakpoints and observe

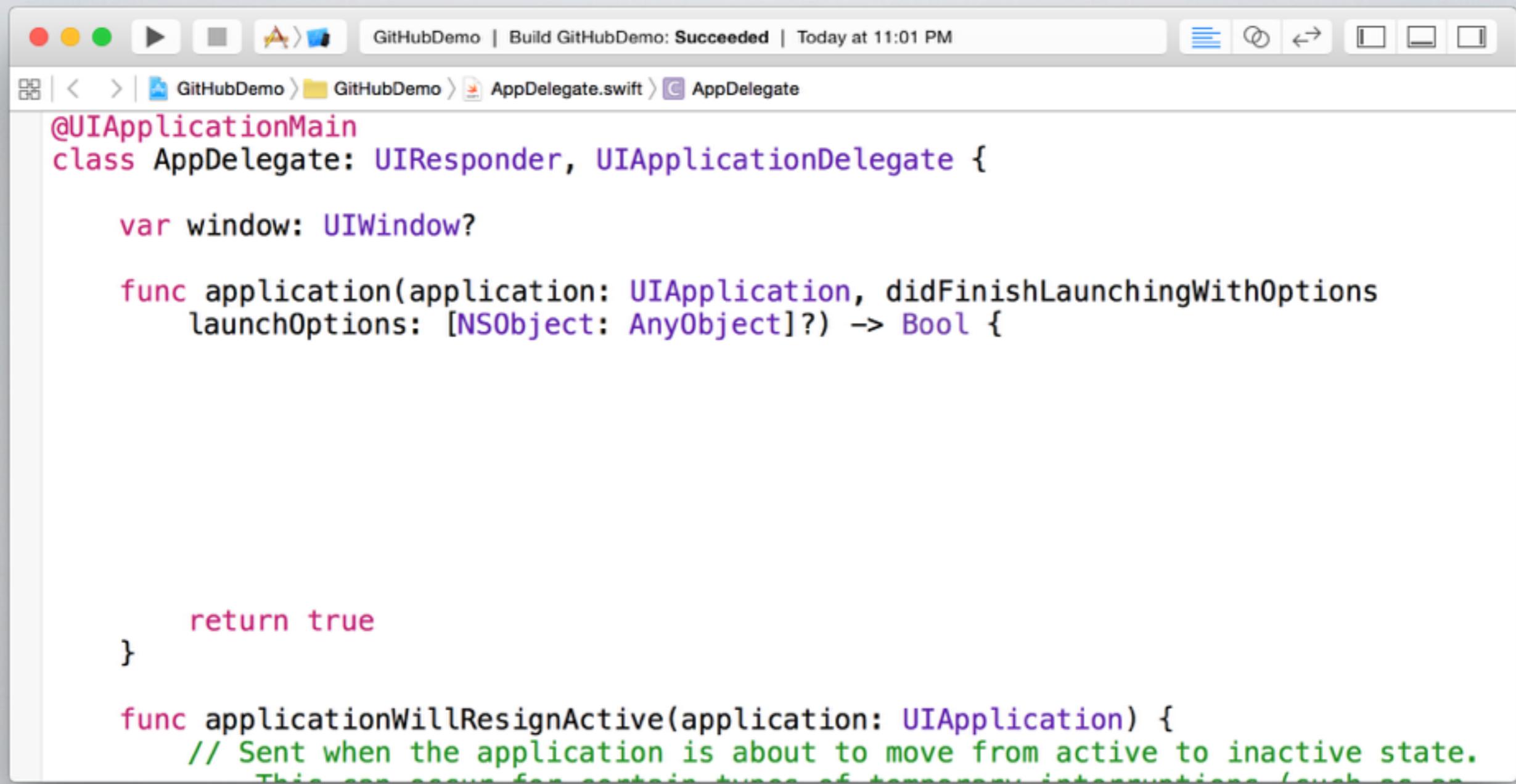
PROGRAMMATICALLY



The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for zoom, search, and file operations.
- Status Bar:** GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM
- Project Navigator:** GitHubDemo > GitHubDemo > AppDelegate.swift
- Editor:** Shows the code for `AppDelegate`. The `application(_:didFinishLaunchingWithOptions:)` method is selected.
- Callout:** A callout box provides detailed documentation for the `application(_:didFinishLaunchingWithOptions:)` method.
 - Declaration:** `func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject : AnyObject]?) -> Bool`
 - Description:** Tells the delegate that the launch process is almost done and the app is almost ready to run.
 - Parameters:**
 - application**: The singleton app object.
 - launchOptions**: A dictionary indicating the reason the app was launched (if any). The contents of this dictionary may be empty in situations where the user launched the app directly. For information about the possible keys in this dictionary and how to handle them, see [Launch Options Keys](#).
 - Returns:** `NO` if the app cannot handle the URL resource or continue a user activity, otherwise return `YES`.
 - Availability:** iOS (3.0 and later)
 - Declared In:** UIKit
 - Reference:** [UIApplicationDelegate Protocol Reference](#)
- Text Preview:** Shows the continuation of the `application(_:didFinishLaunchingWithOptions:)` method implementation.

PROGRAMMATICALLY



A screenshot of an Xcode workspace window. The title bar shows "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM". The file path in the navigation bar is "GitHubDemo > GitHubDemo > AppDelegate.swift". The main editor area displays Swift code for an AppDelegate:

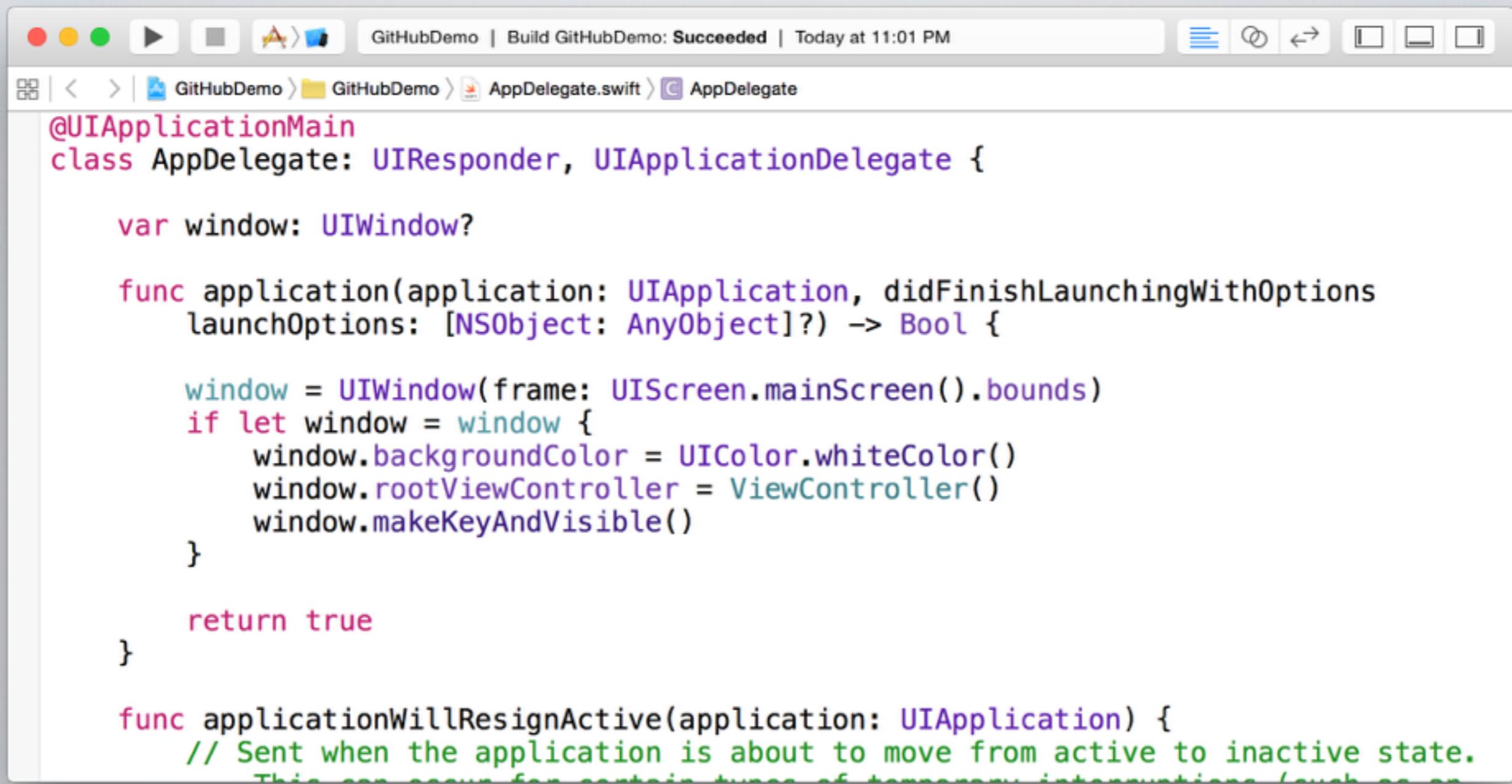
```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {
        // ...
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
        // incoming phone call or SMS message) or when the user quits the application and it begins
        // the transition to the background.
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the contents of `AppDelegate.swift`.

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

    var window: UIWindow?

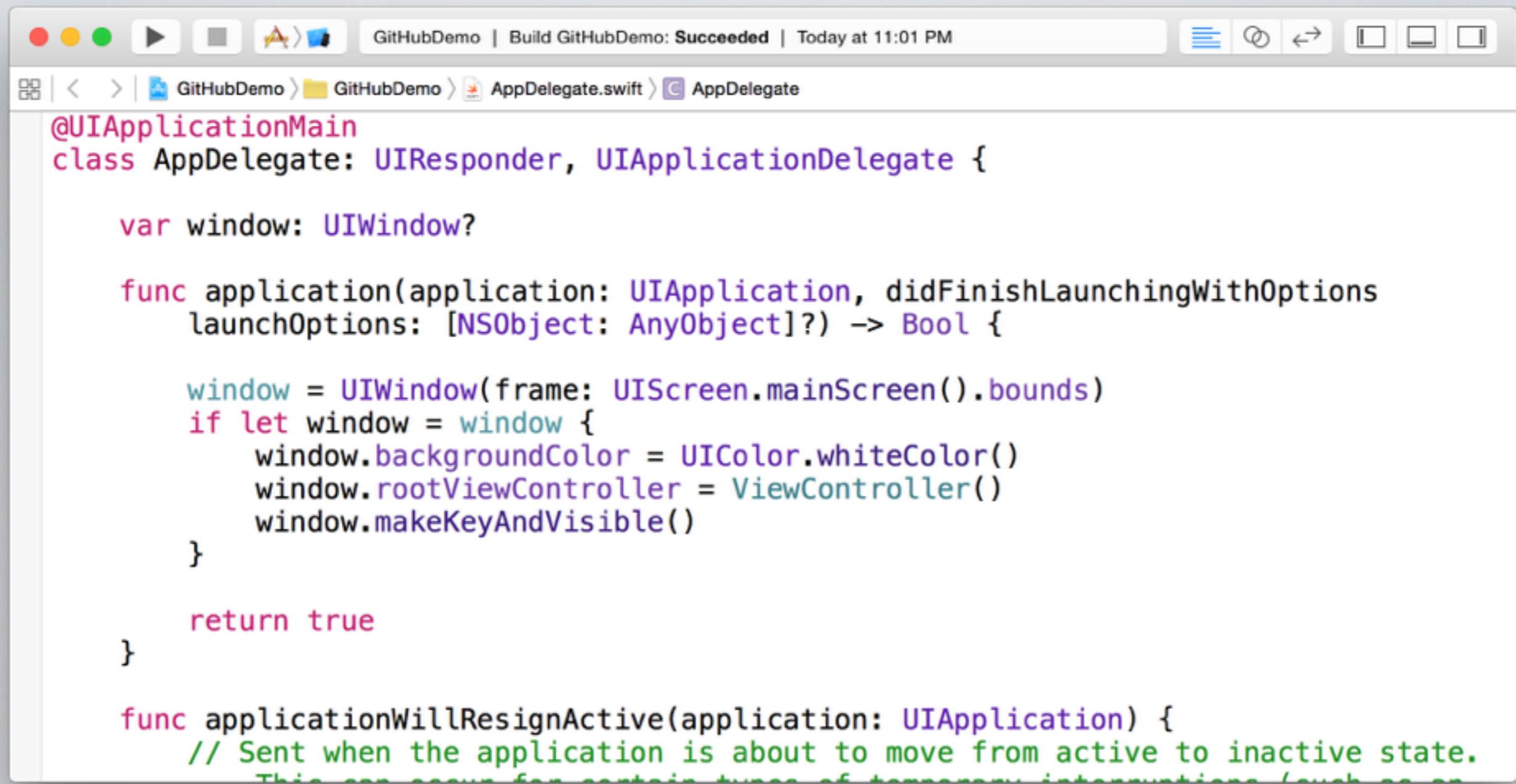
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the contents of `AppDelegate.swift`.

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

    var window: UIWindow?

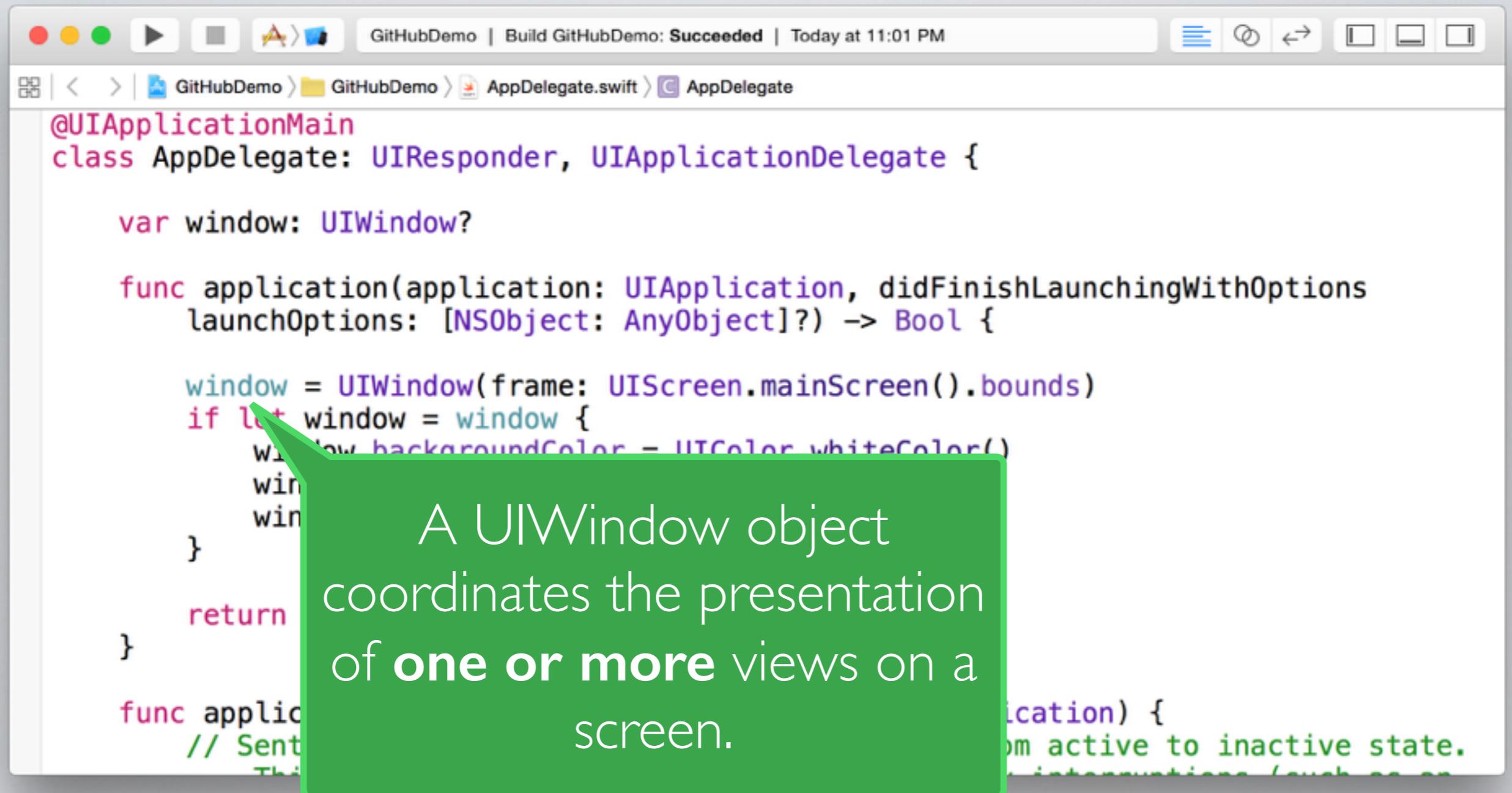
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode IDE with the file `AppDelegate.swift` open. The code defines an `UIApplicationDelegate` implementation. A green callout box points to the `window` variable declaration, which is highlighted in blue. The callout contains the following text:

A UIWindow object coordinates the presentation of **one or more** views on a screen.

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ...
        }
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.
    }

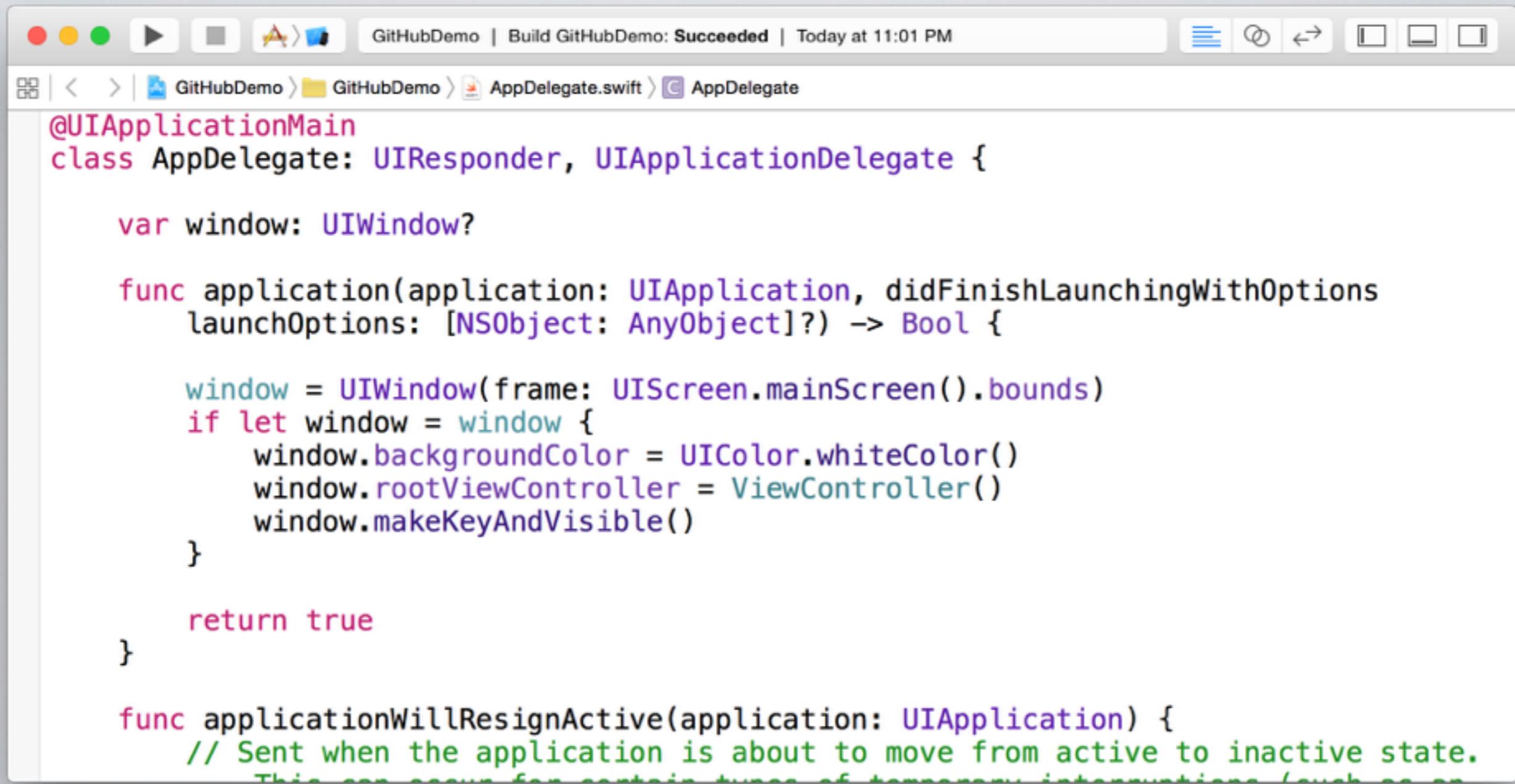
    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the foreground.
        // Restore application state from its previous state.
    }

    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillTerminate(application: UIApplication) {
        // Called when the application is about to terminate. Save data as appropriate.
        // See also applicationDidEnterBackground:.
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the contents of `AppDelegate.swift`.

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

    var window: UIWindow?

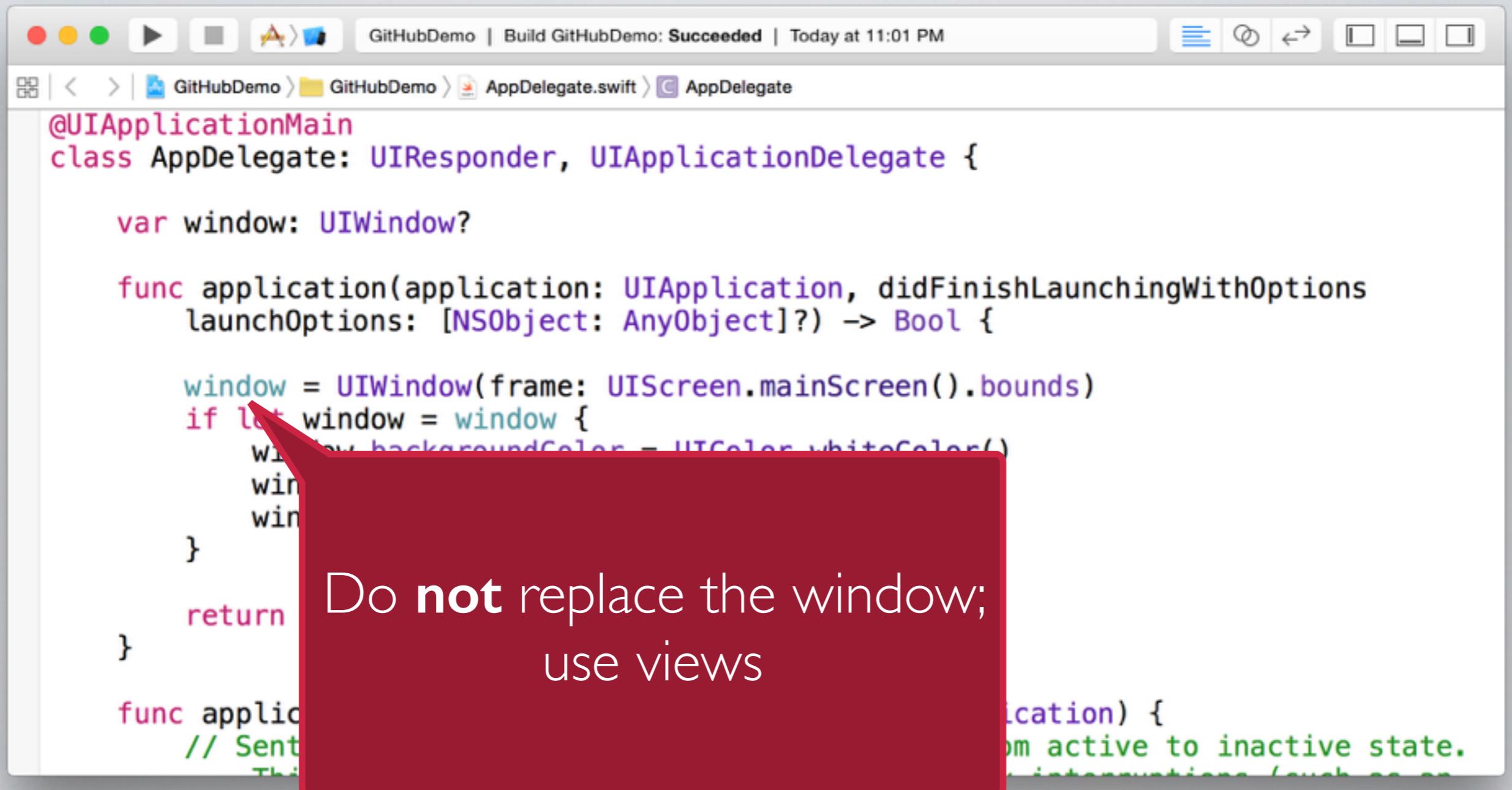
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the file `AppDelegate.swift` open. The code implements the `UIApplicationDelegate` protocol. A red callout box points to the line `window = UIWindow(frame: UIScreen.mainScreen().bounds)`, which creates a new window instead of replacing the existing one.

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ...
            window.makeKeyAndVisible()
        }
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
    }

    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
    }

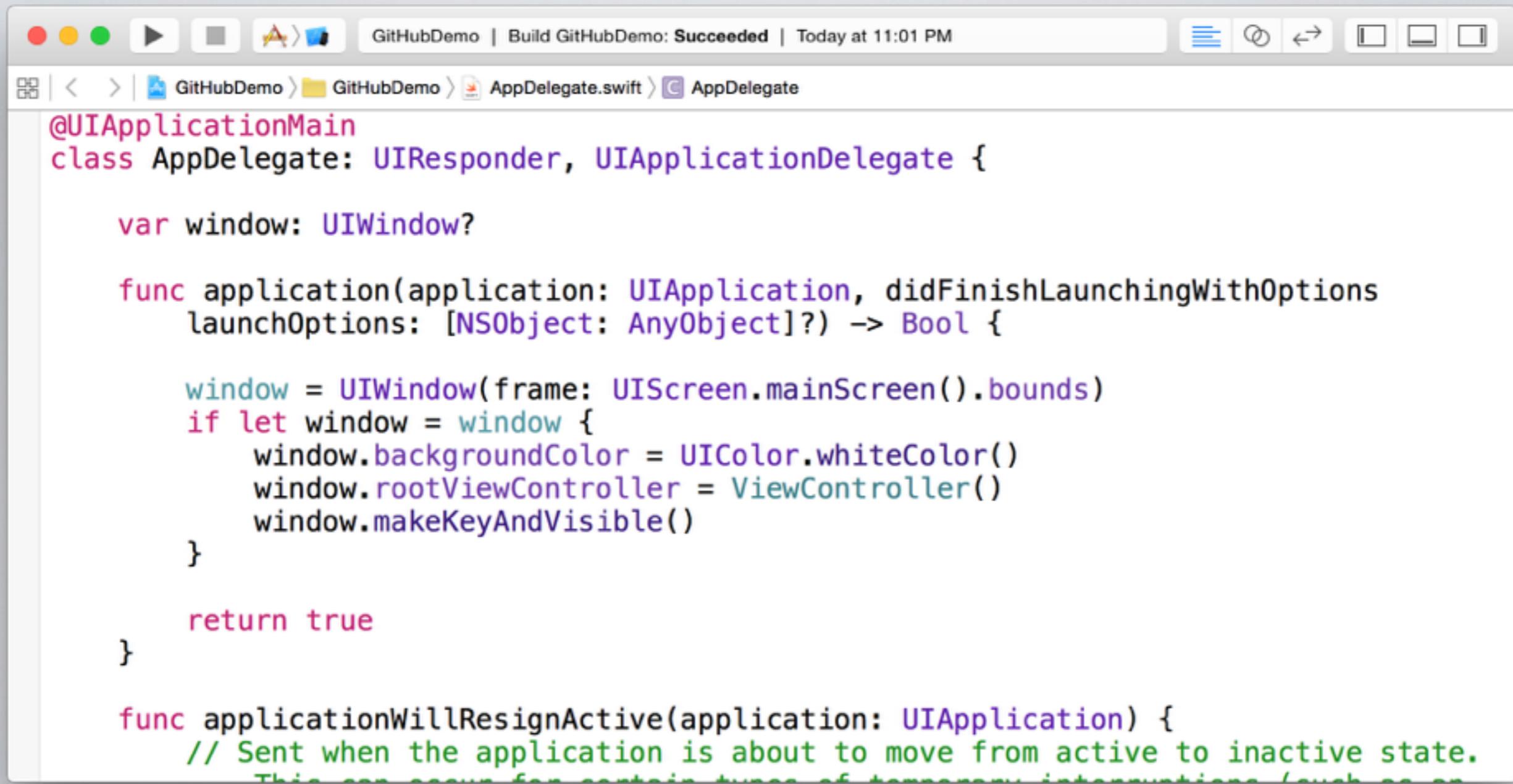
    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the foreground, during which, if necessary, you should reset any user interface that was modified by entering the background.
    }

    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillTerminate(application: UIApplication) {
        // Called as part of the transition from the foreground to the background. This method is also called when the user quits the application and it begins the transition to the background state.
    }
}
```

Do **not** replace the window;
use views

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the contents of `AppDelegate.swift`.

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

    var window: UIWindow?

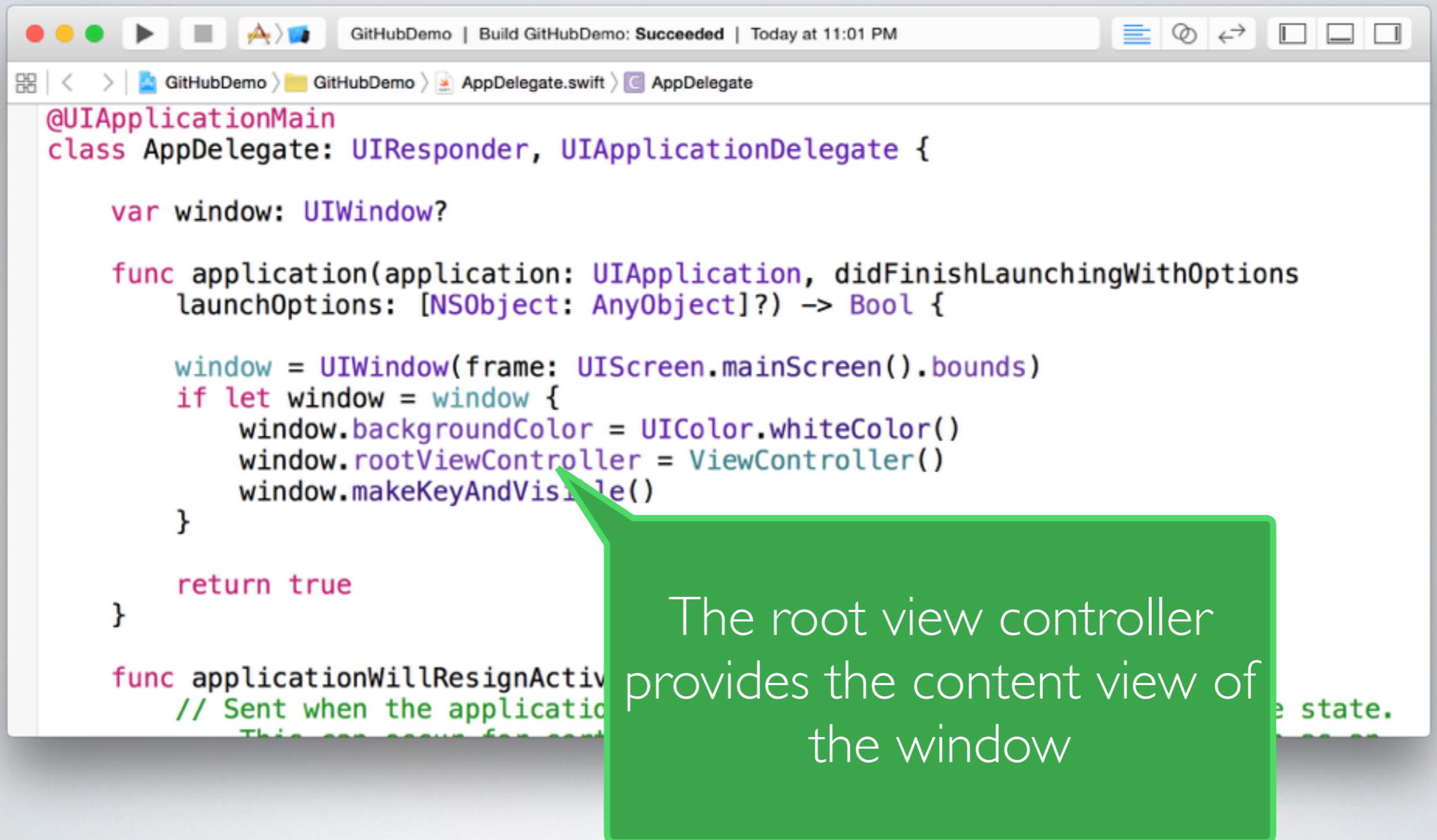
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode IDE with the file `AppDelegate.swift` open. The code demonstrates how to create a window programmatically:

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
    }

    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the foreground.
        // Restore application state from its previous state.
    }

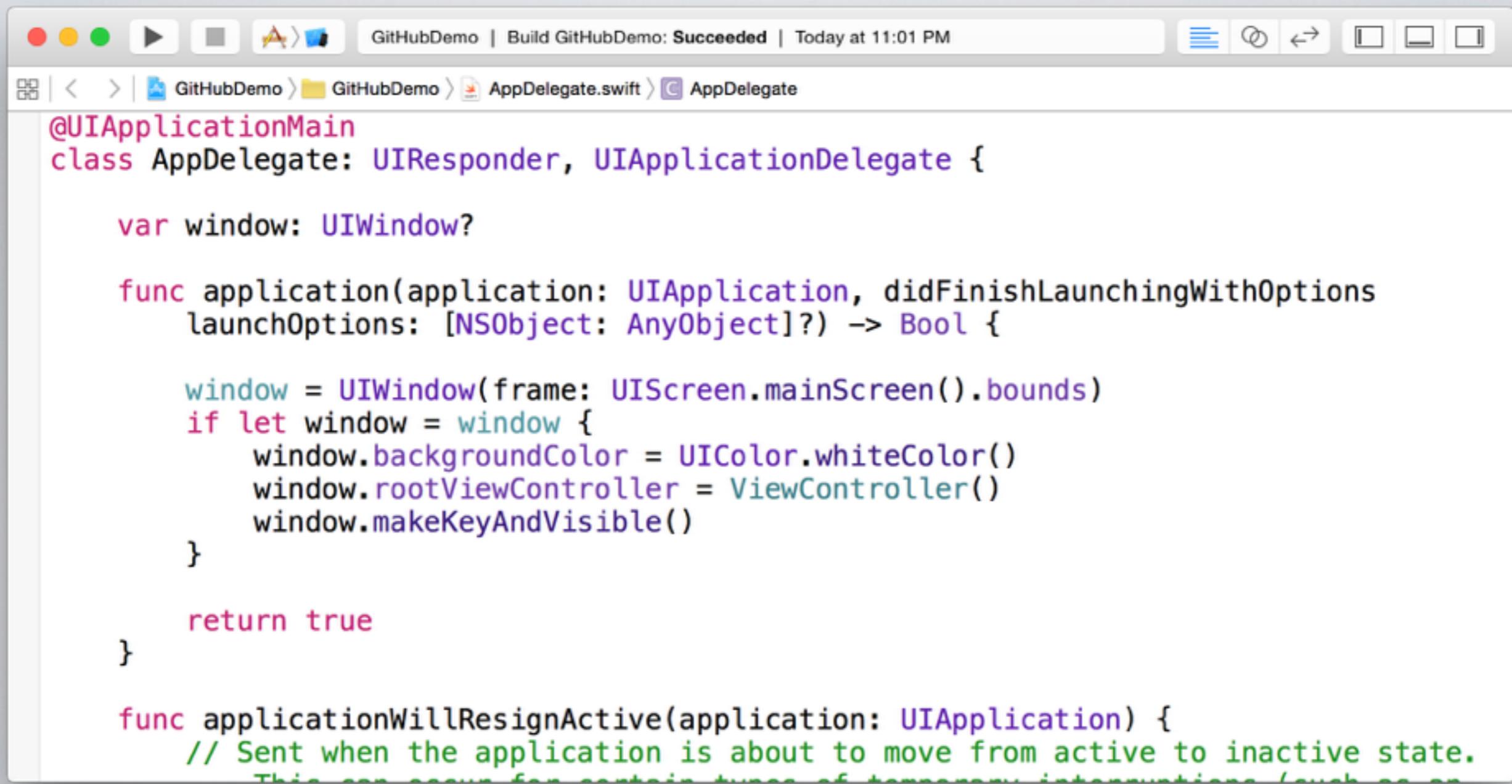
    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillTerminate(application: UIApplication) {
        // Called when the application is about to terminate. Save data as needed. See also applicationDidEnterBackground:.
    }
}
```

A green callout bubble points to the line `window.rootViewController = ViewController()`. Inside the bubble, the text reads:

The root view controller provides the content view of the window

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the `AppDelegate.swift` file content.

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

    var window: UIWindow?

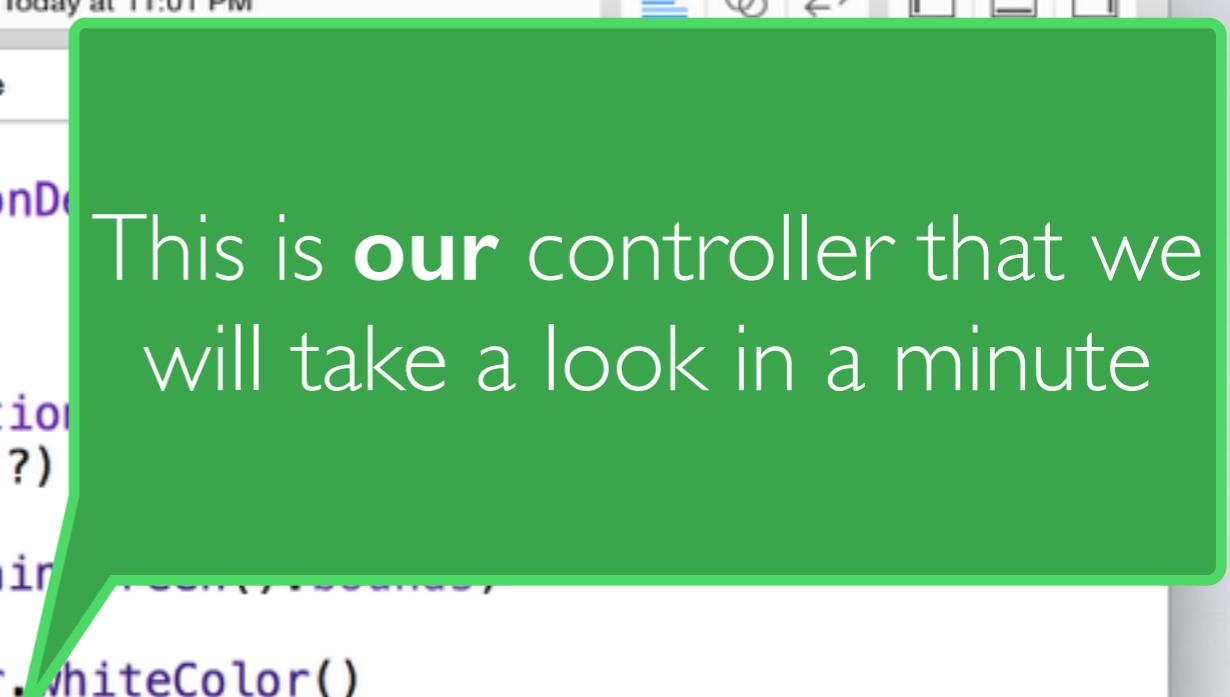
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



```
GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM
GitHubDemo | GitHubDemo | AppDelegate.swift | AppDelegate

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

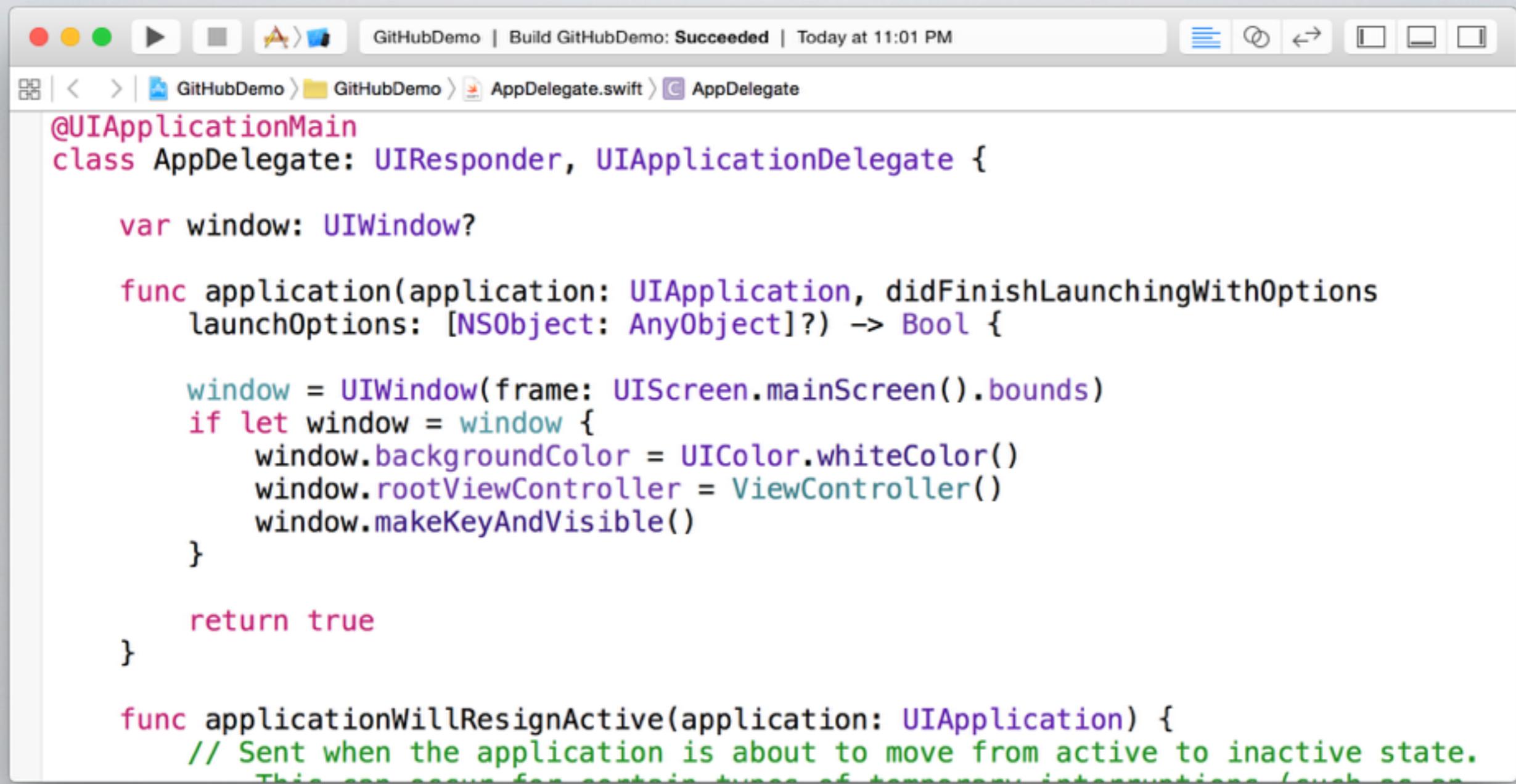
    var window: UIWindow?

    func application(application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) {
        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

This is **our** controller that we will take a look in a minute

PROGRAMMATICALLY



The screenshot shows the Xcode IDE interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Status Bar:** Displays "GitHubDemo | Build GitHubDemo: Succeeded | Today at 11:01 PM".
- Project Navigator:** Shows the project structure: GitHubDemo > GitHubDemo > AppDelegate.swift.
- Code Editor:** Displays the contents of `AppDelegate.swift`.

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

    var window: UIWindow?

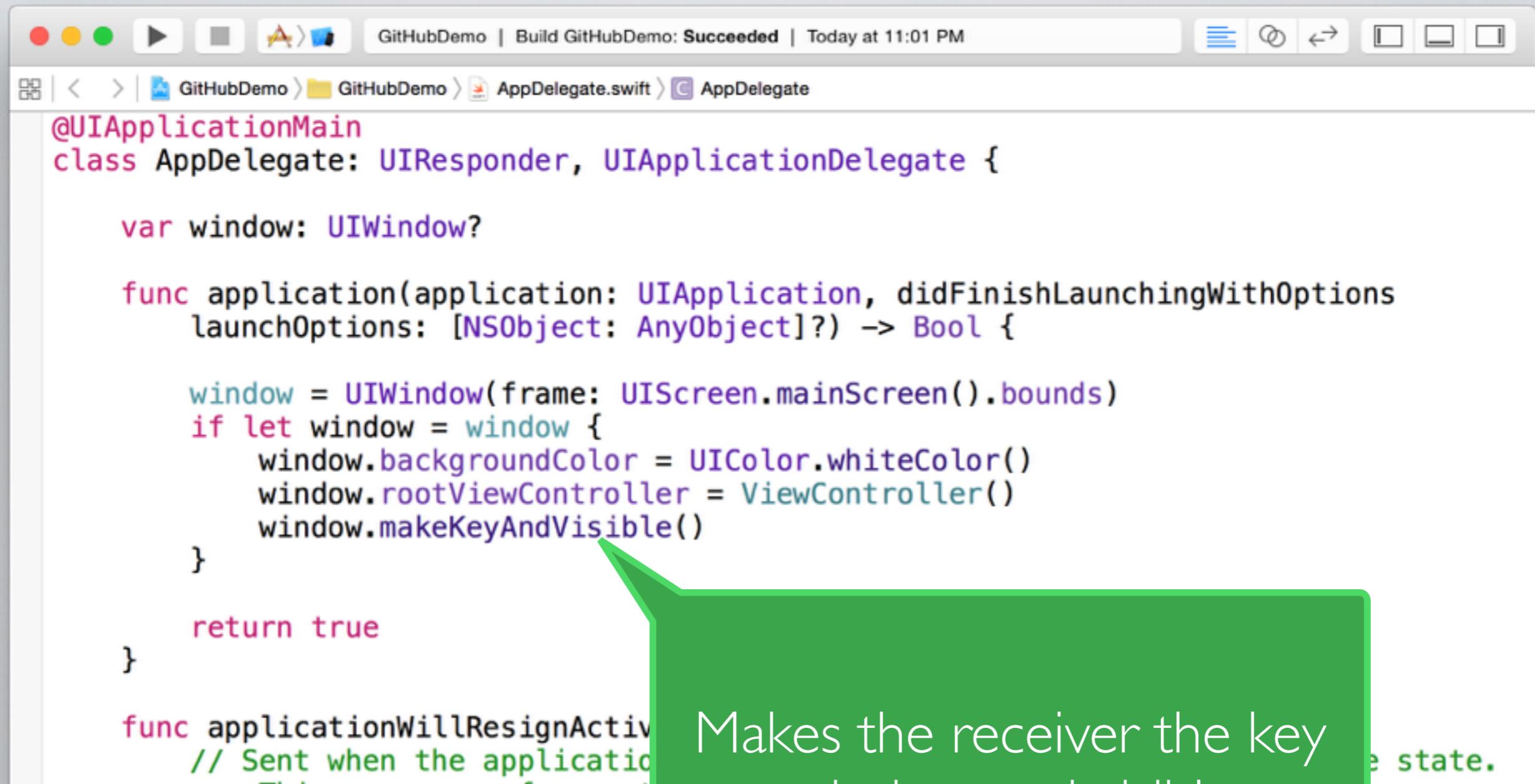
    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }

        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (such as an
    }
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the file `AppDelegate.swift` open. The code implements the `UIApplicationDelegate` protocol, specifically overriding the `application:didFinishLaunchingWithOptions:` method to set up a window. A green callout bubble points to the `makeKeyAndVisible()` method.

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

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [NSObject: AnyObject]?) -> Bool {

        window = UIWindow(frame: UIScreen.mainScreen().bounds)
        if let window = window {
            window.backgroundColor = UIColor.whiteColor()
            window.rootViewController = ViewController()
            window.makeKeyAndVisible()
        }
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state.
        // This can occur for certain types of temporary interruptions (like an incoming phone call or SMS) or when the user quits the application and it begins the transition to the background state.
    }

    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

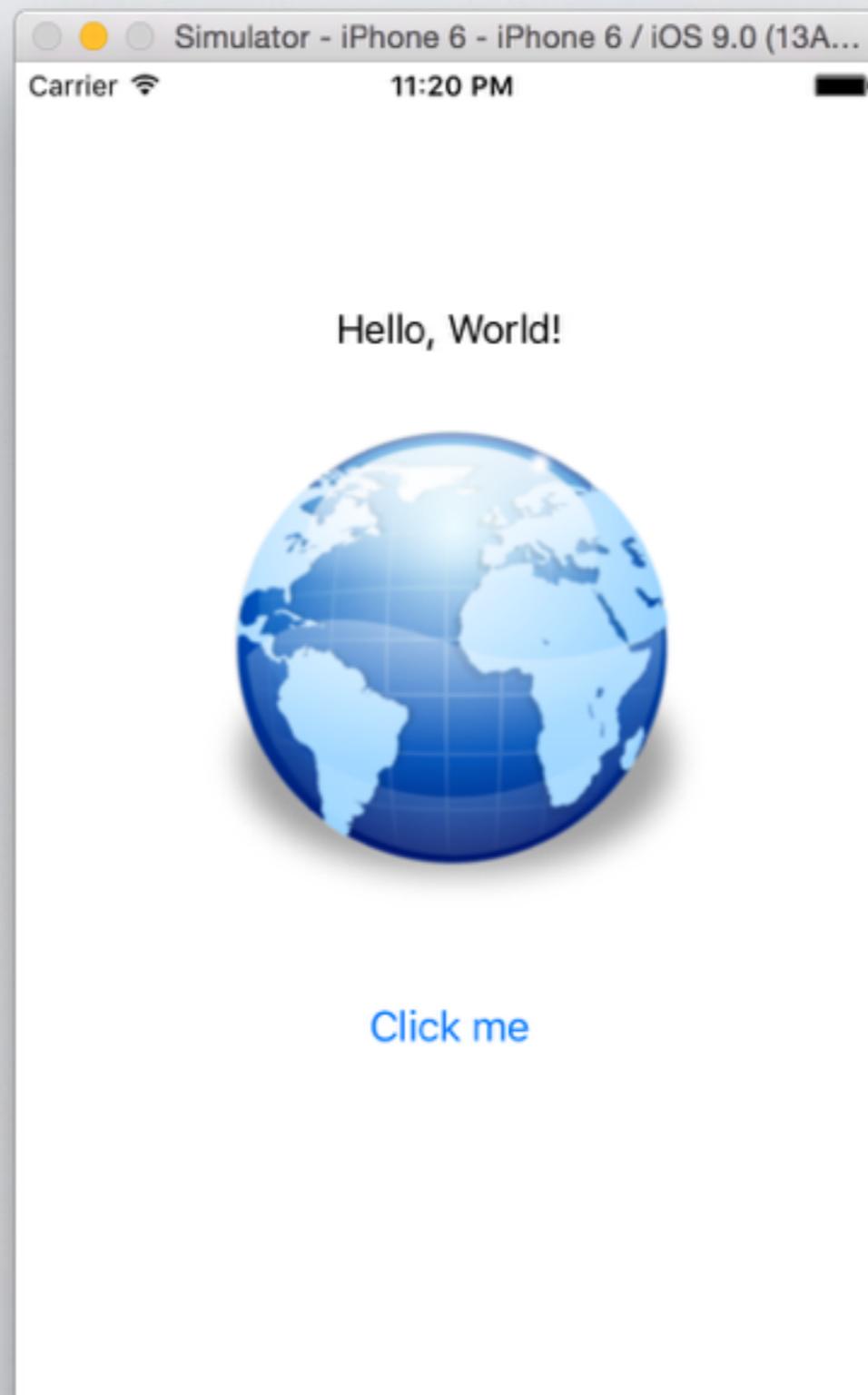
    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the foreground.
        // Restore application state from its previous state.
    }

    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
    }

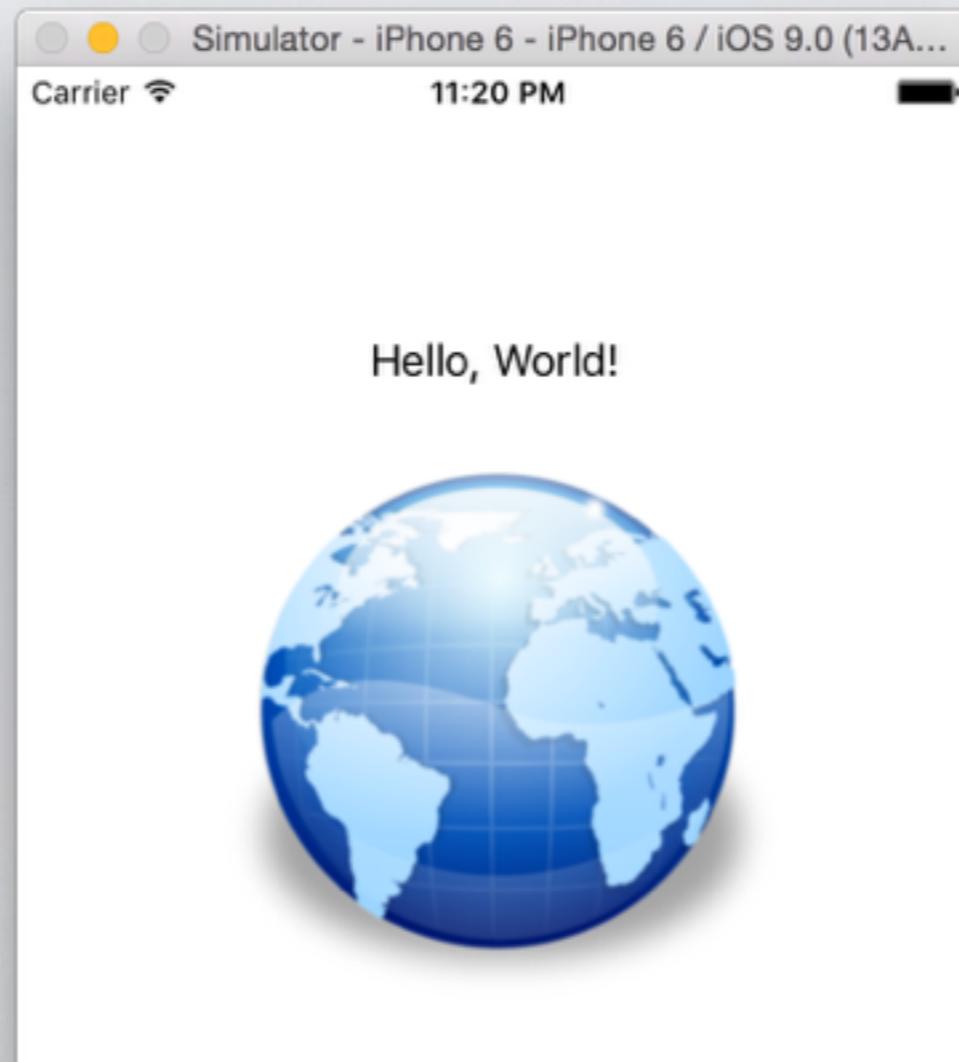
    func applicationWillTerminate(application: UIApplication) {
        // Called when the application is about to terminate. Save data as needed. See also applicationDidEnterBackground:.
    }
}
```

Makes the receiver the key window and visible

PROGRAMMATICALLY



PROGRAMMATICALLY



This is just an example. Don't do this programmatically; use storyboards as we shall see

PROGRAMMATICALLY

A screenshot of an Xcode interface. The title bar says "Running GitHubDemo on iPhone 6". The navigation bar shows the file path: GitHubDemo > GitHubDemo > ViewController.swift > viewDidLoad(). The main editor area contains Swift code:

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() { ... }

    // Documentation card for viewDidLoad()
    Declaration override func viewDidLoad()
    Description Called after the controller's view is loaded into memory.
    This method is called after the view controller has loaded its view hierarchy into memory. This method is called regardless of whether the view hierarchy was loaded from a nib file or created programmatically in the loadView method. You usually override this method to perform additional initialization on views that were loaded from nib files.
    Availability iOS (8.1 and later)
    Declared In UIKit
    Reference UIViewController Class Reference
}
```

The documentation for `viewDidLoad()` is displayed in a callout box. The "Description" section is visible, and the "Availability" section is partially visible. A green annotation "recreated." is placed next to the end of the declaration line in the code.

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations, search, and navigation.
- Status Bar:** Shows "Finished running GitHubDemo on iPhone 6".
- Project Navigator:** Displays the project structure: GitHubDemo > GitHubDemo > ViewController.swift.
- Code Editor:** Displays the following Swift code:

```
class ViewController: UIViewController {

    var hello: UILabel?

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = UIColor.whiteColor()

        hello = UILabel(frame: CGRectMake(27.5, 100, 320, 30))
        if let hello = hello {
            hello.textAlignment = NSTextAlignment.Center
            hello.textColor = UIColor.blackColor()
            hello.font = UIFont(name: "System", size: 20)
            hello.text = "Hello, World!"
            view.addSubview(hello)
        }
    }
}
```

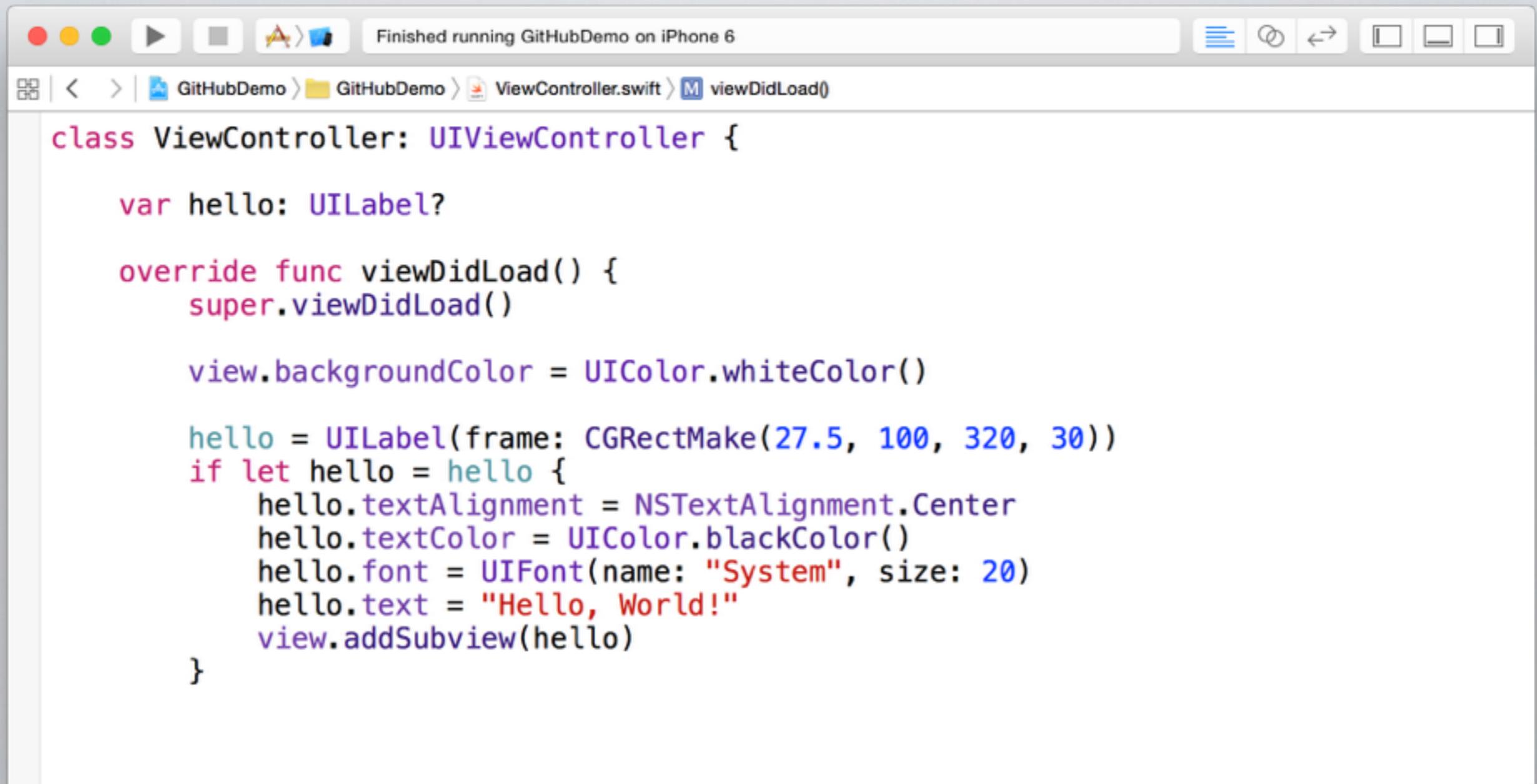
PROGRAMMATICALLY



```
Finished running GitHubDemo on iPhone 6
GitHubDemo | GitHubDemo | ViewController.swift | viewDidLoad()
class ViewController: UIViewController {
    var hello: UILabel
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.whiteColor()
        hello = UILabel(frame: CGRectMake(27.5, 100, 320, 30))
        if let hello = hello {
            hello.textAlignment = NSTextAlignment.Center
            hello.textColor = UIColor.blackColor()
            hello.font = UIFont(name: "System", size: 20)
            hello.text = "Hello, World!"
            view.addSubview(hello)
        }
    }
}
```

This is how to create a label programmatically

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Mac OS X-style toolbar with icons for file operations.
- Status Bar:** Shows "Finished running GitHubDemo on iPhone 6".
- Project Navigator:** Displays the project structure: GitHubDemo > GitHubDemo > ViewController.swift.
- Code Editor:** Displays the following Swift code:

```
class ViewController: UIViewController {

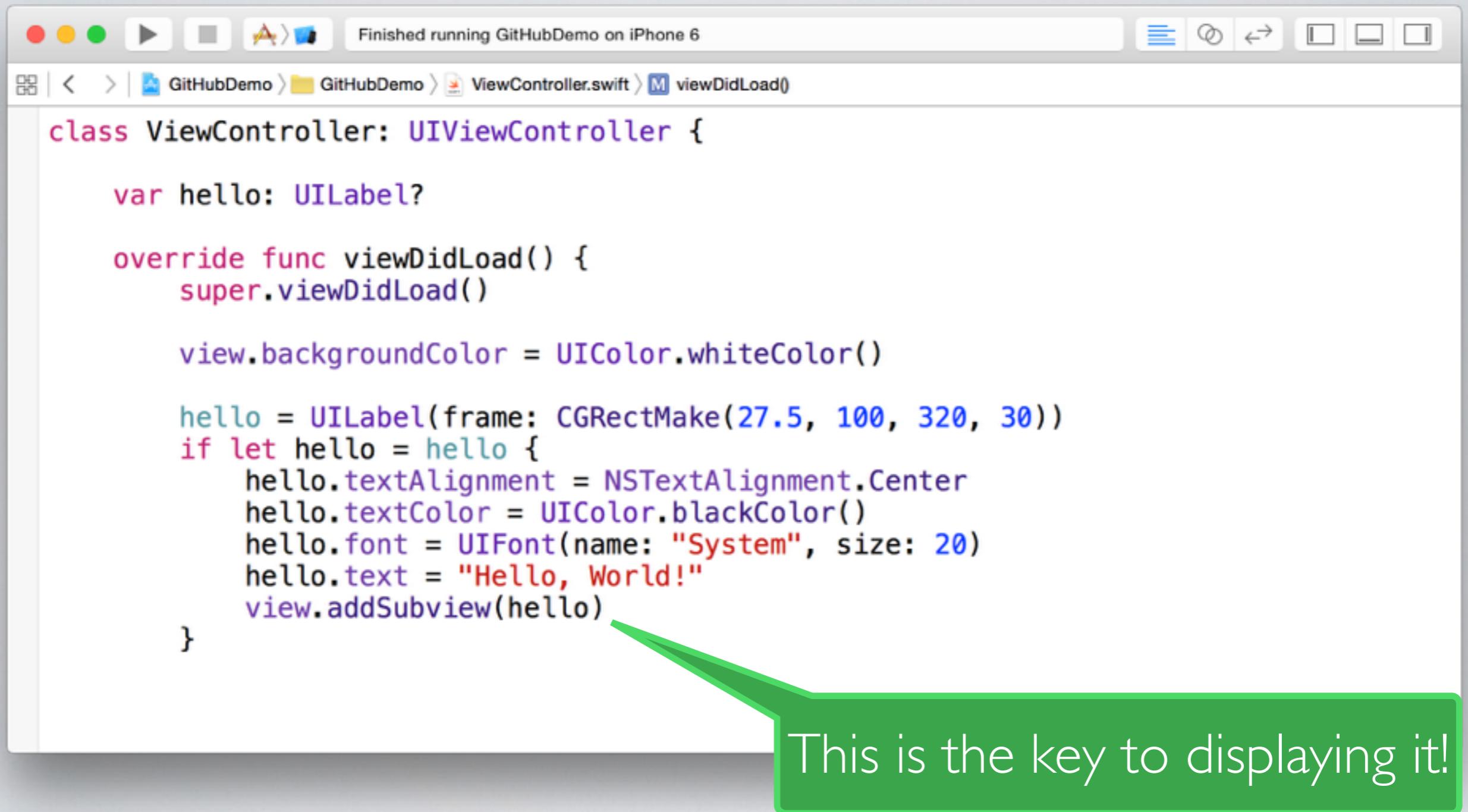
    var hello: UILabel?

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = UIColor.whiteColor()

        hello = UILabel(frame: CGRectMake(27.5, 100, 320, 30))
        if let hello = hello {
            hello.textAlignment = NSTextAlignment.Center
            hello.textColor = UIColor.blackColor()
            hello.font = UIFont(name: "System", size: 20)
            hello.text = "Hello, World!"
            view.addSubview(hello)
        }
    }
}
```

PROGRAMMATICALLY



A screenshot of an Xcode workspace. The title bar says "Finished running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > viewDidLoad(). The main editor area contains the following Swift code:

```
class ViewController: UIViewController {

    var hello: UILabel?

    override func viewDidLoad() {
        super.viewDidLoad()

        view.backgroundColor = UIColor.whiteColor()

        hello = UILabel(frame: CGRectMake(27.5, 100, 320, 30))
        if let hello = hello {
            hello.textAlignment = NSTextAlignment.Center
            hello.textColor = UIColor.blackColor()
            hello.font = UIFont(name: "System", size: 20)
            hello.text = "Hello, World!"
            view.addSubview(hello)
        }
    }
}
```

A green callout bubble points from the bottom right towards the line `view.addSubview(hello)`. Inside the bubble, the text "This is the key to displaying it!" is written.

DETOUR – COMPOSITE PATTERN

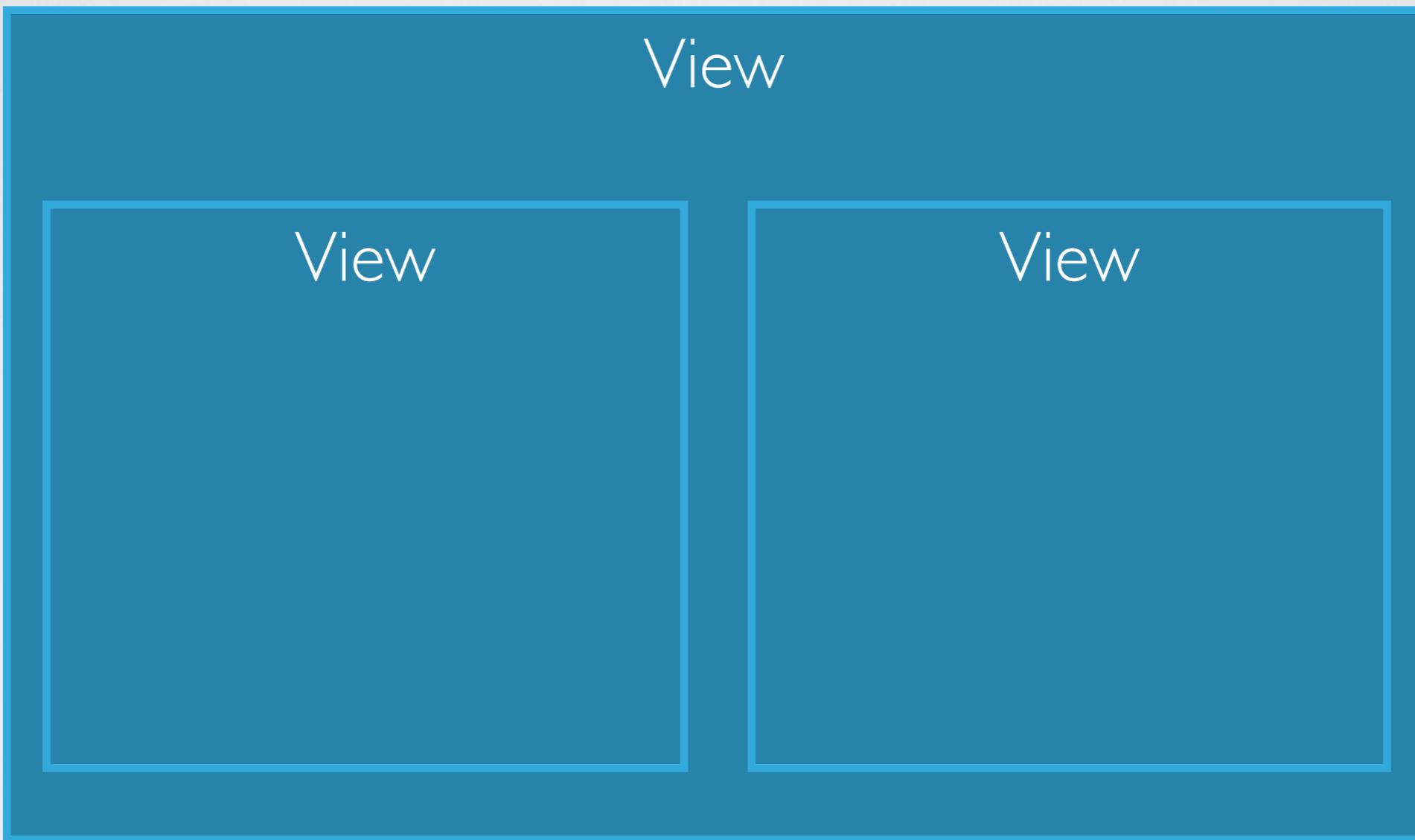
View

DETOUR – COMPOSITE PATTERN

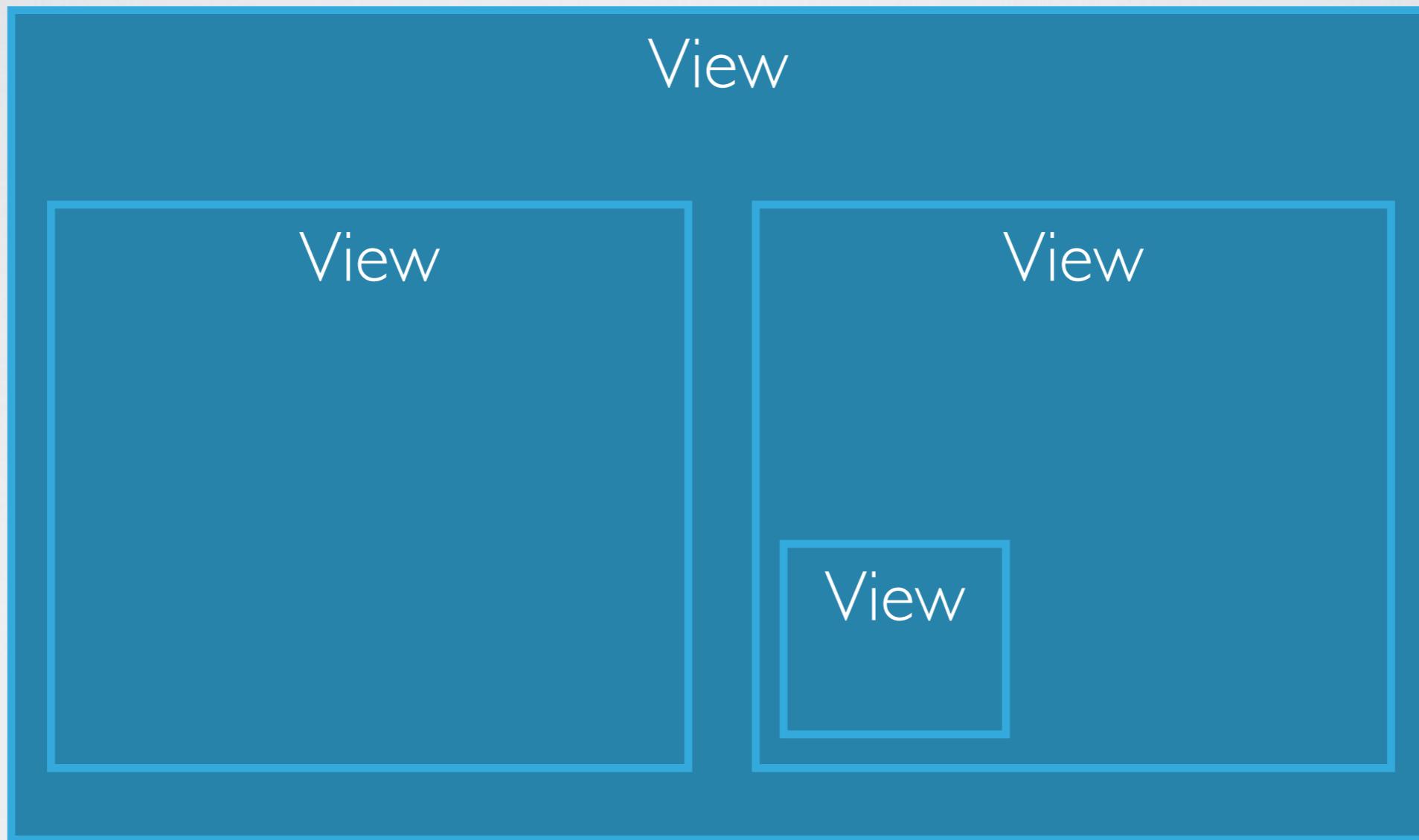
View

View

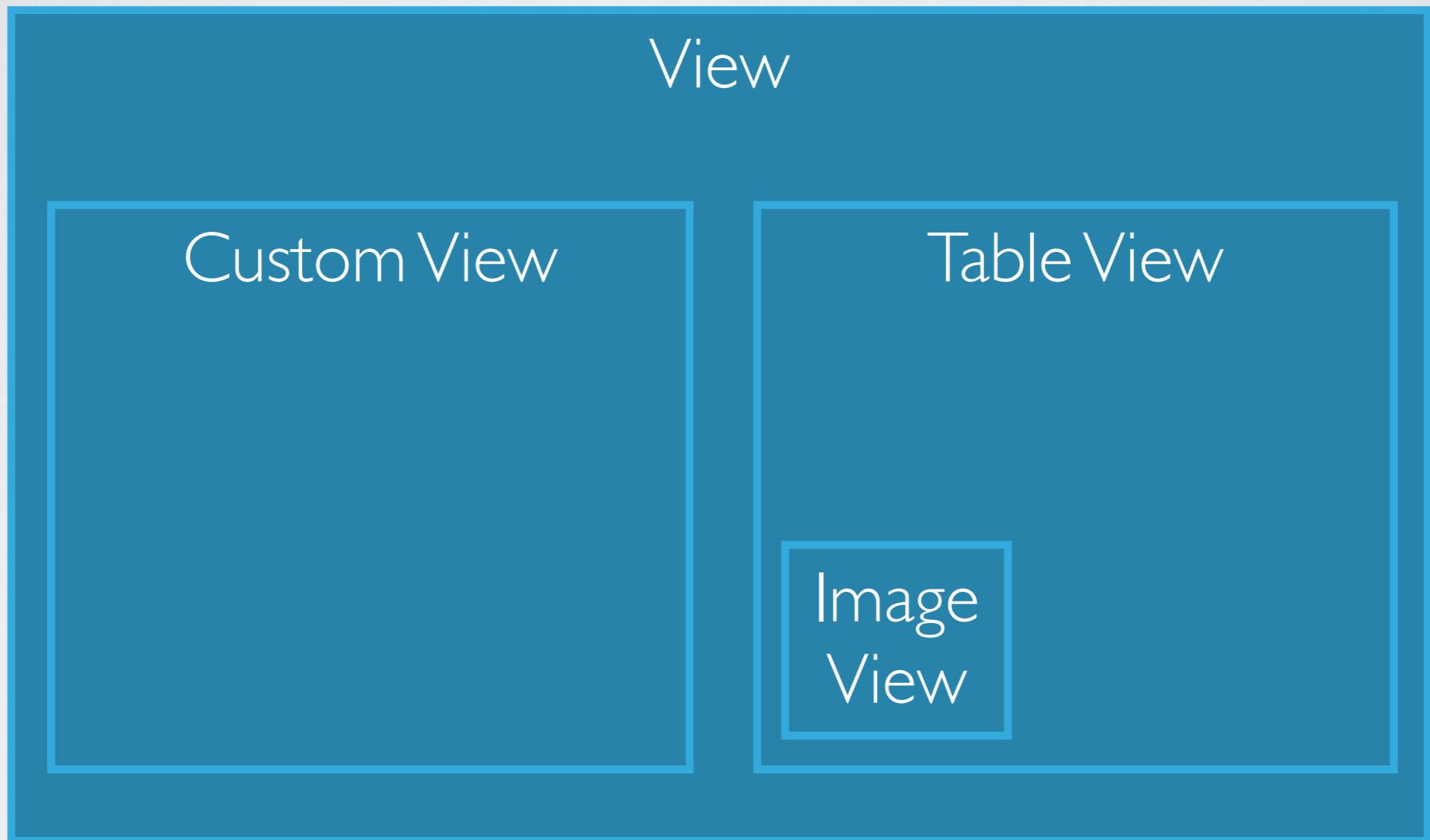
DETOUR – COMPOSITE PATTERN



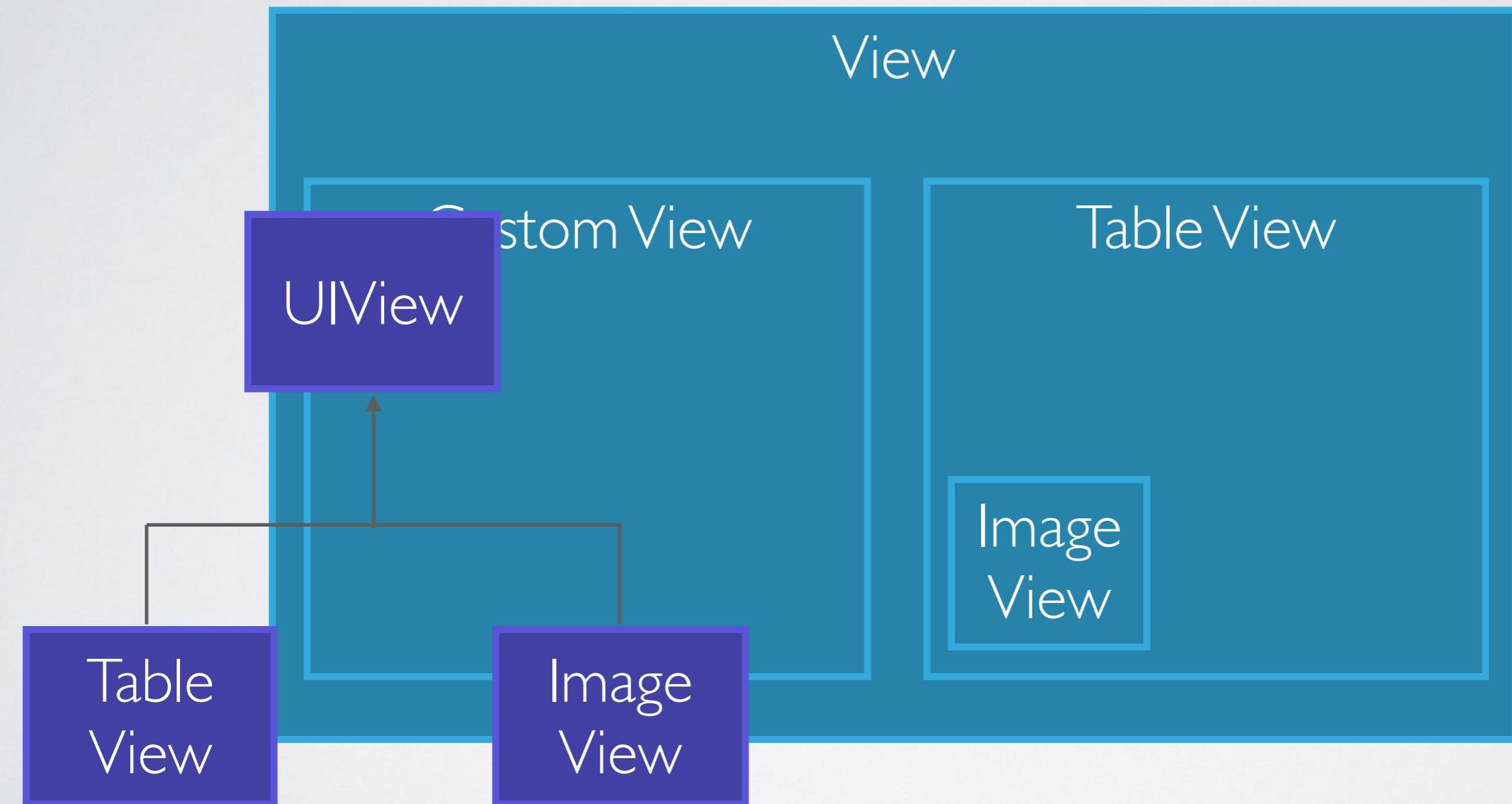
DETOUR – COMPOSITE PATTERN



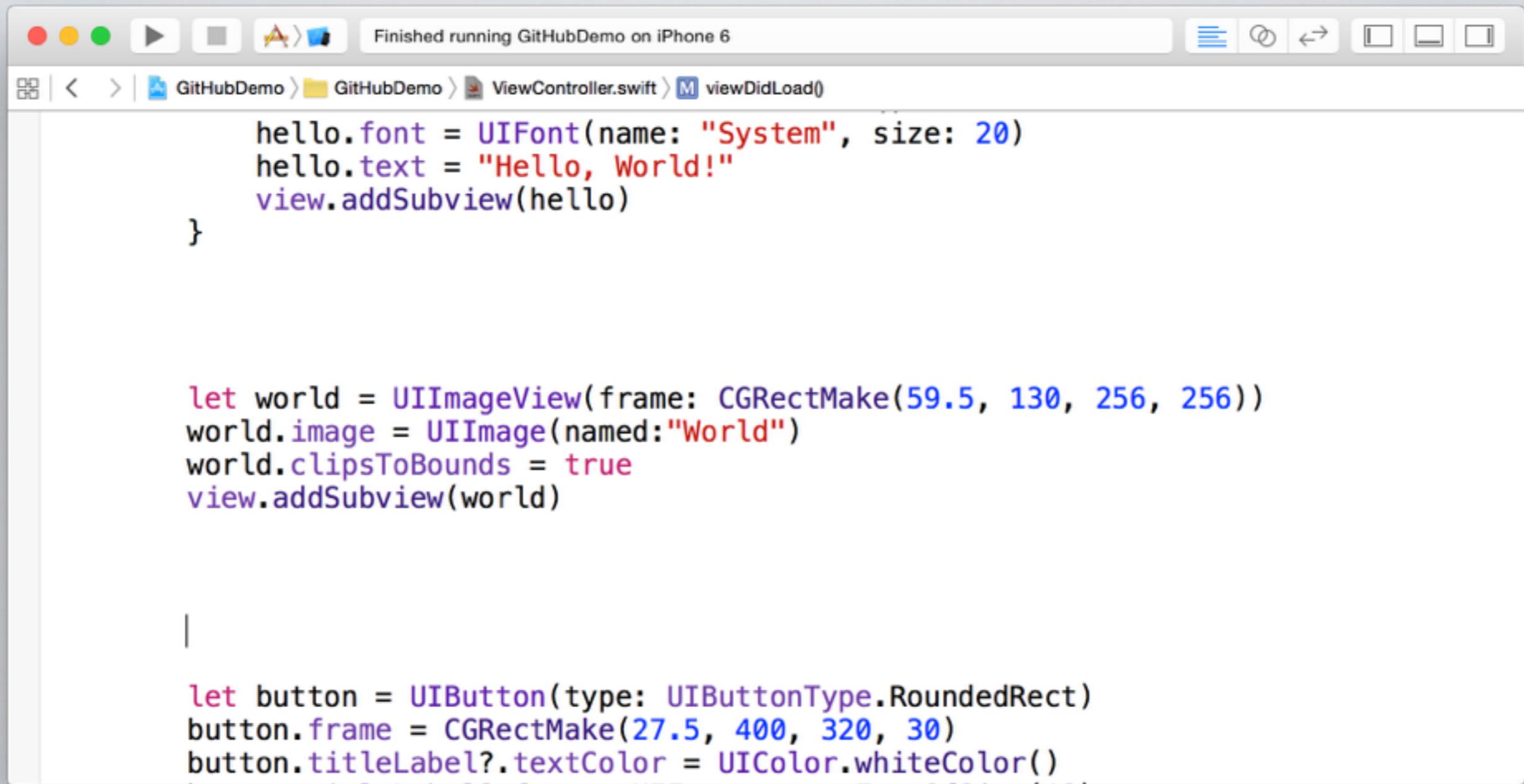
DETOUR – COMPOSITE PATTERN



DETOUR – COMPOSITE PATTERN



PROGRAMMATICALLY



A screenshot of the Xcode IDE interface. The title bar shows "Finished running GitHubDemo on iPhone 6". The navigation bar indicates the project is "GitHubDemo", the file is "ViewController.swift", and the current function is "viewDidLoad()". The main editor area displays the following Swift code:

```
hello.font = UIFont(name: "System", size: 20)
hello.text = "Hello, World!"
view.addSubview(hello)

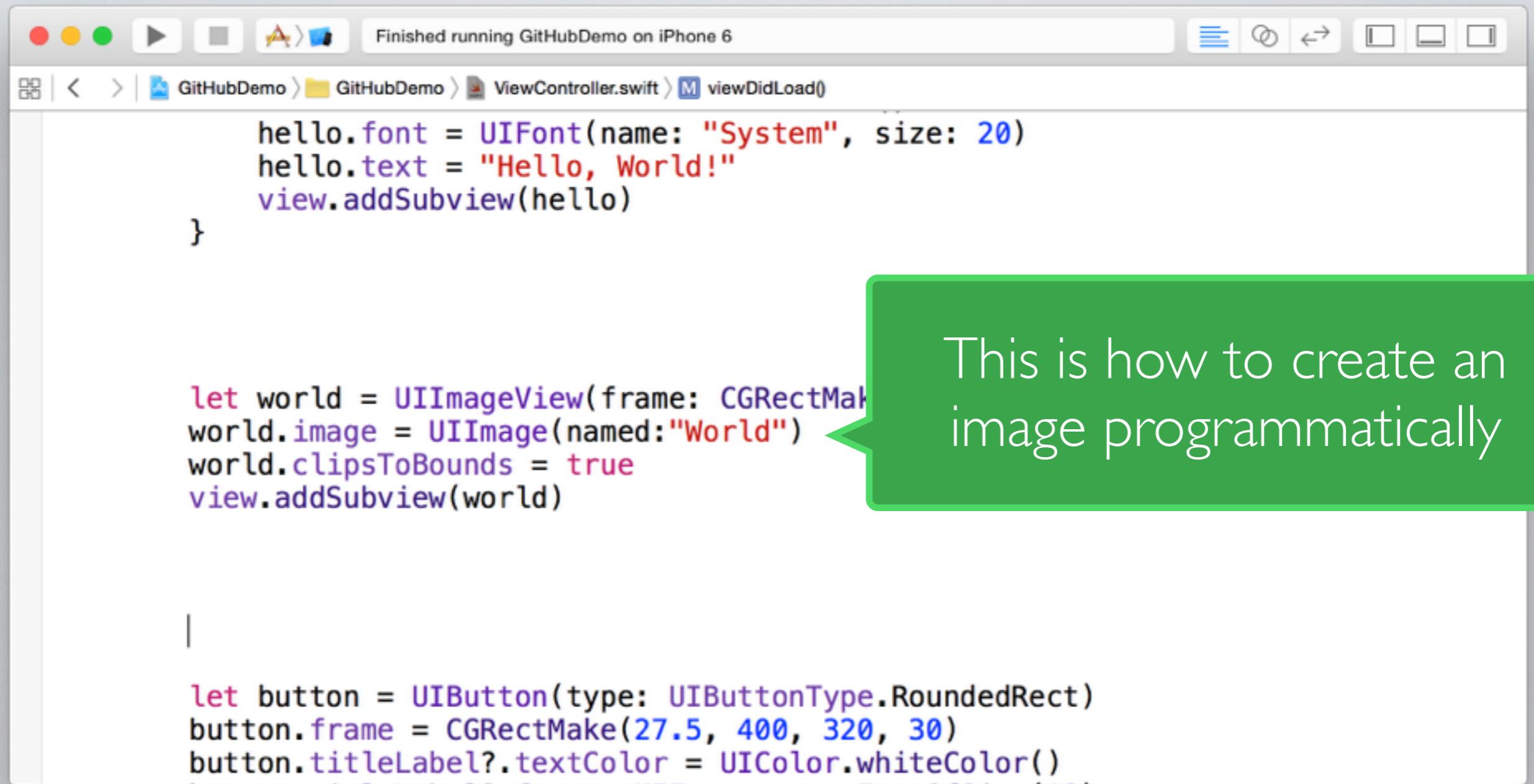
}

let world = UIImageView(frame: CGRectMake(59.5, 130, 256, 256))
world.image = UIImage(named:"World")
world.clipsToBounds = true
view.addSubview(world)

| 

let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
```

PROGRAMMATICALLY



A screenshot of an Xcode interface showing a Swift file named `ViewController.swift` with the following code:

```
hello.font = UIFont(name: "System", size: 20)
hello.text = "Hello, World!"
view.addSubview(hello)

}

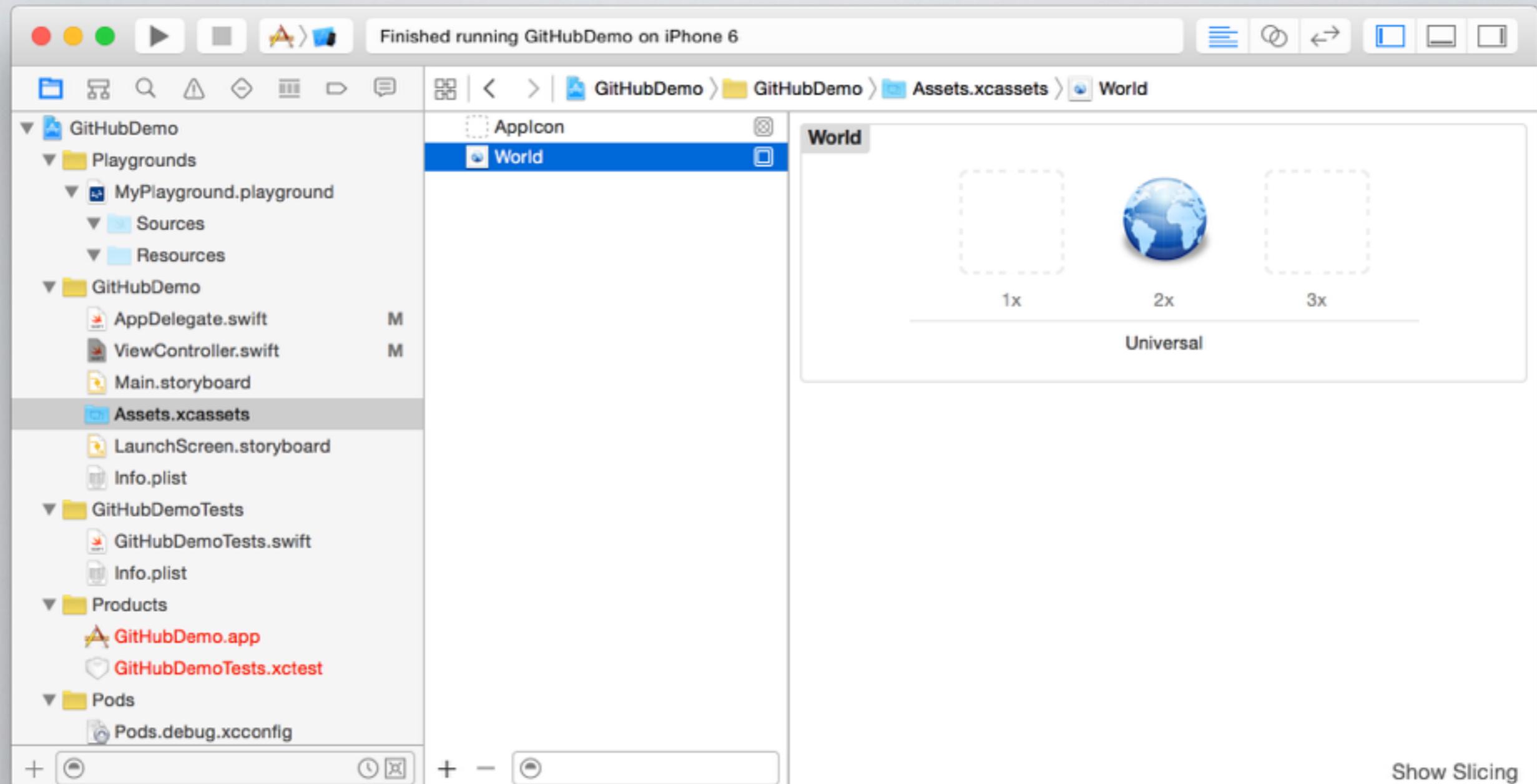
let world = UIImageView(frame: CGRectMake(27.5, 400, 320, 30))
world.image = UIImage(named:"World")
world.clipsToBounds = true
view.addSubview(world)

let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
```

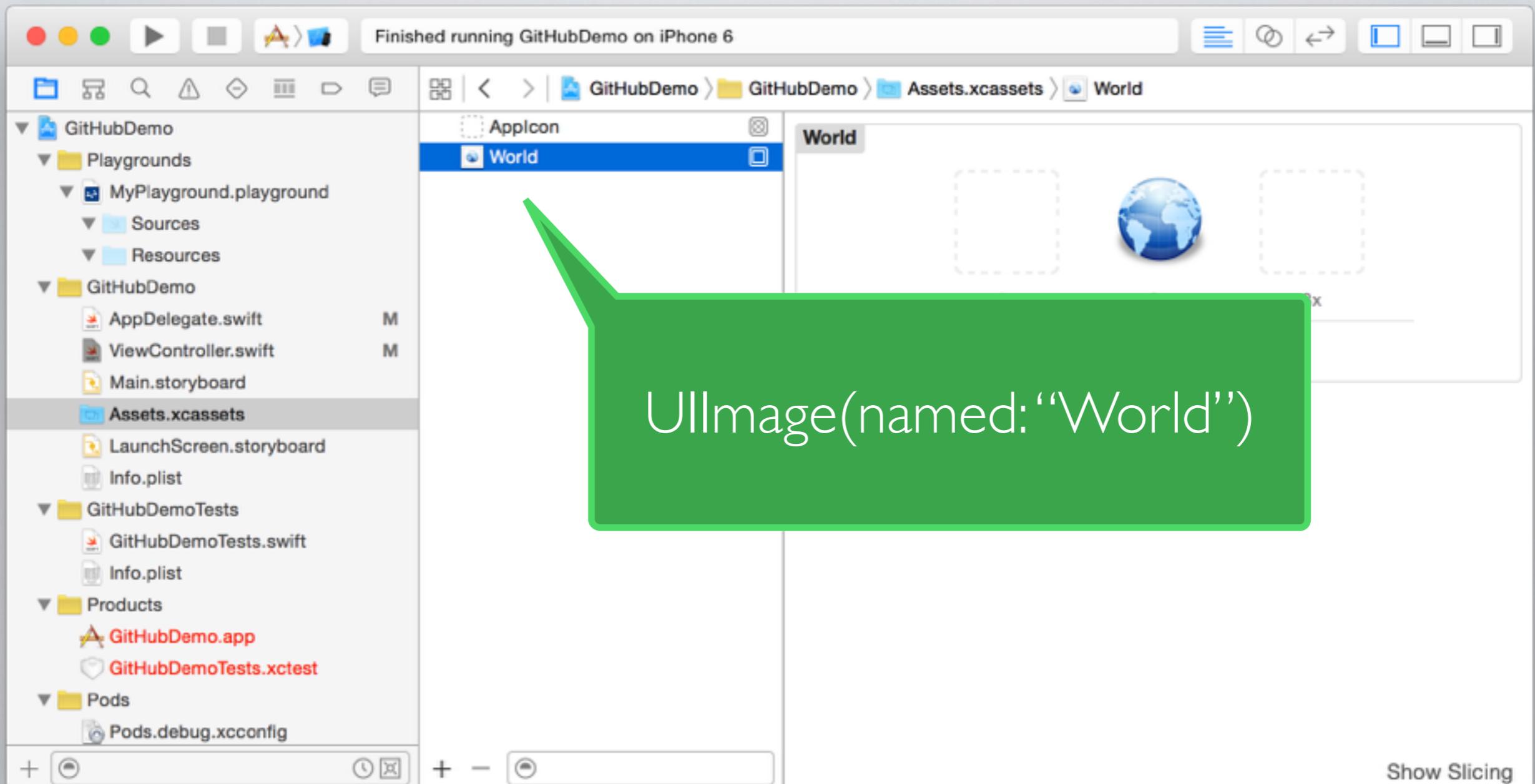
The code demonstrates how to create a label, an image view, and a button programmatically within the `viewDidLoad()` method of a view controller.

This is how to create an image programmatically

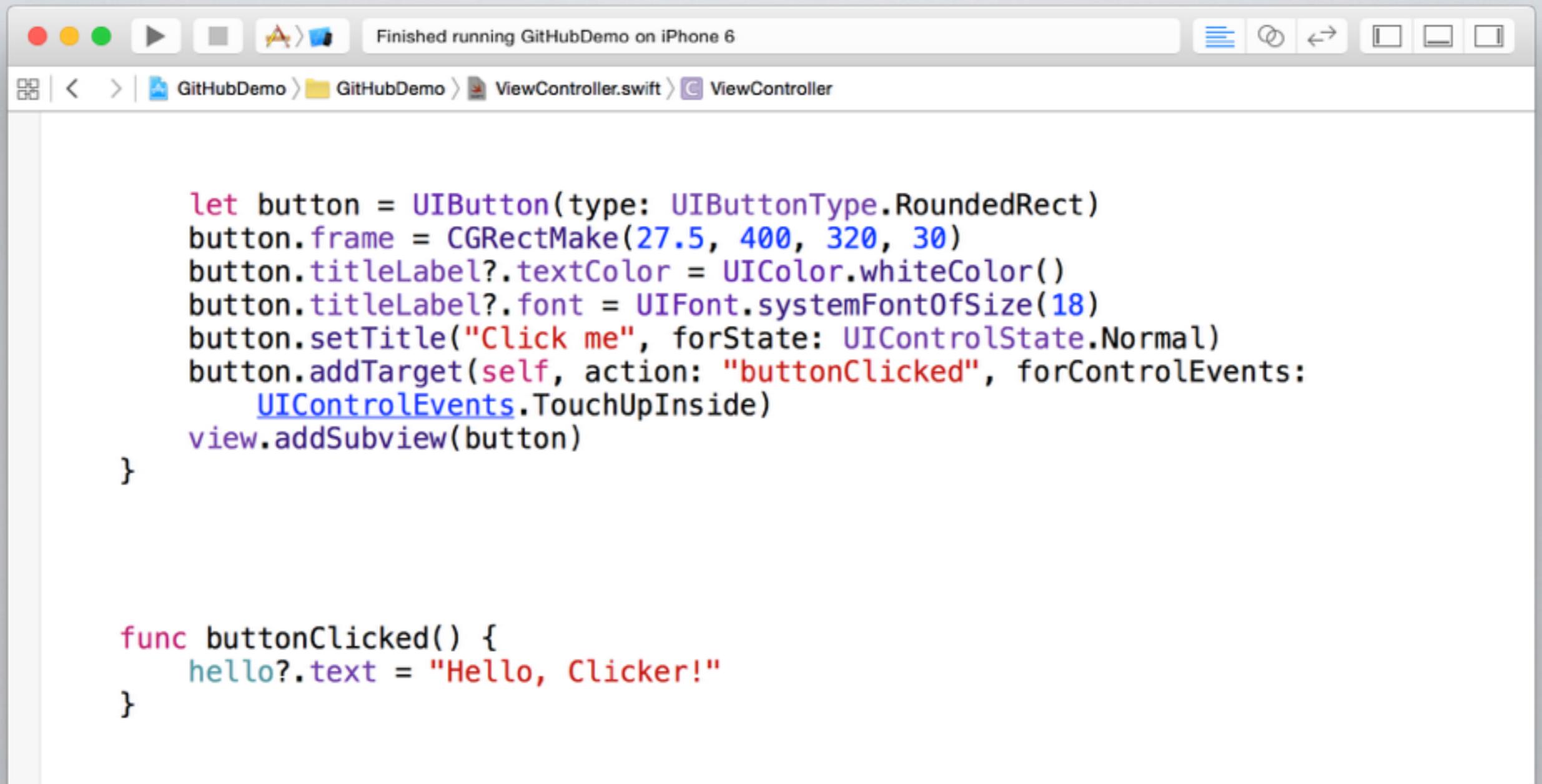
PROGRAMMATICALLY



PROGRAMMATICALLY



PROGRAMMATICALLY



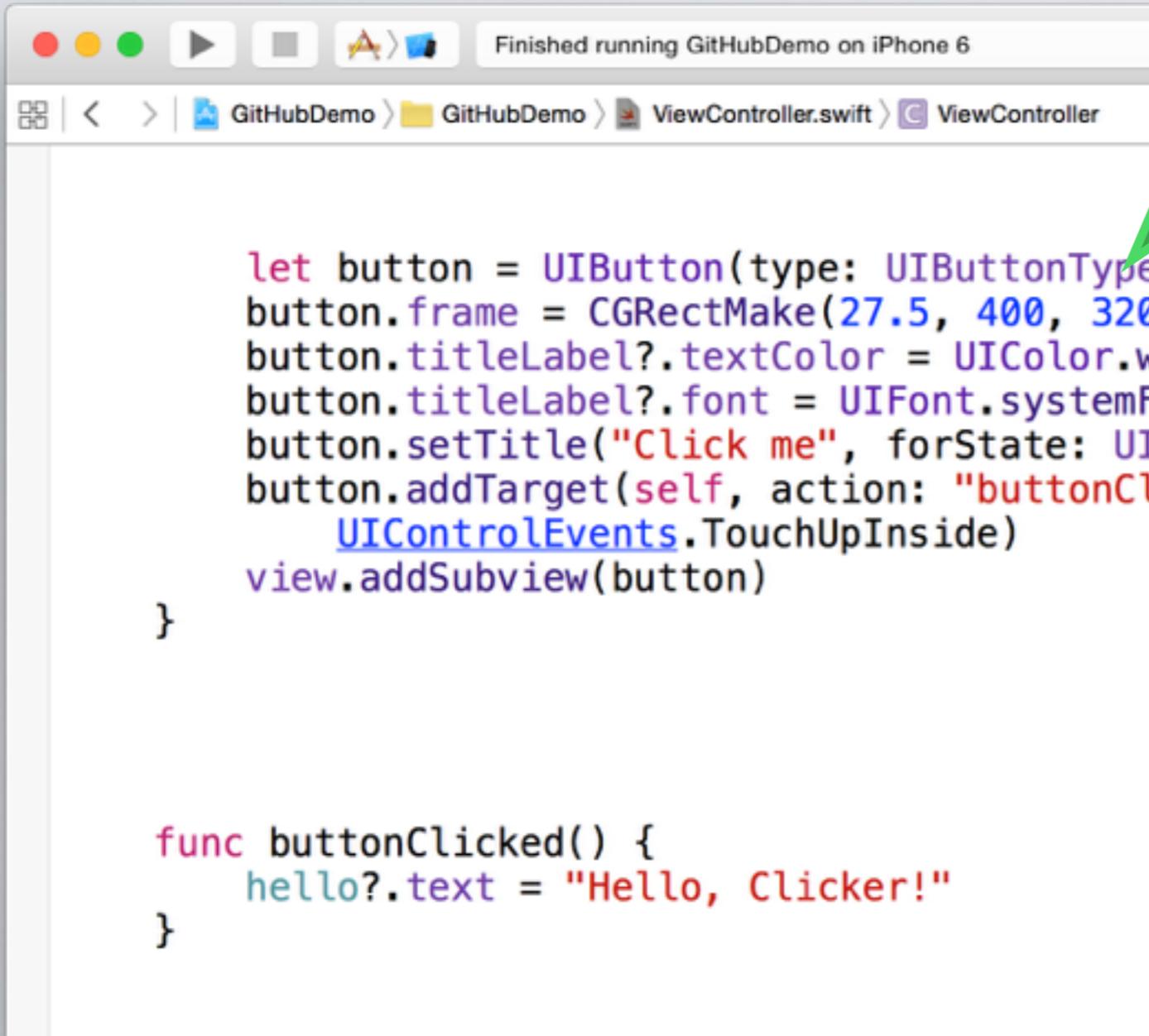
A screenshot of the Xcode IDE interface. The title bar shows "Finished running GitHubDemo on iPhone 6". The navigation bar below shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > ViewController. The main editor area contains Swift code for creating a button programmatically and setting up a tap gesture recognizer.

```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)

}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

PROGRAMMATICALLY



A screenshot of an Xcode interface. The title bar says "Finished running GitHubDemo on iPhone 6". The navigation bar shows "GitHubDemo > GitHubDemo > ViewController.swift > ViewController". The main area contains Swift code for creating a button programmatically and setting up a click event.

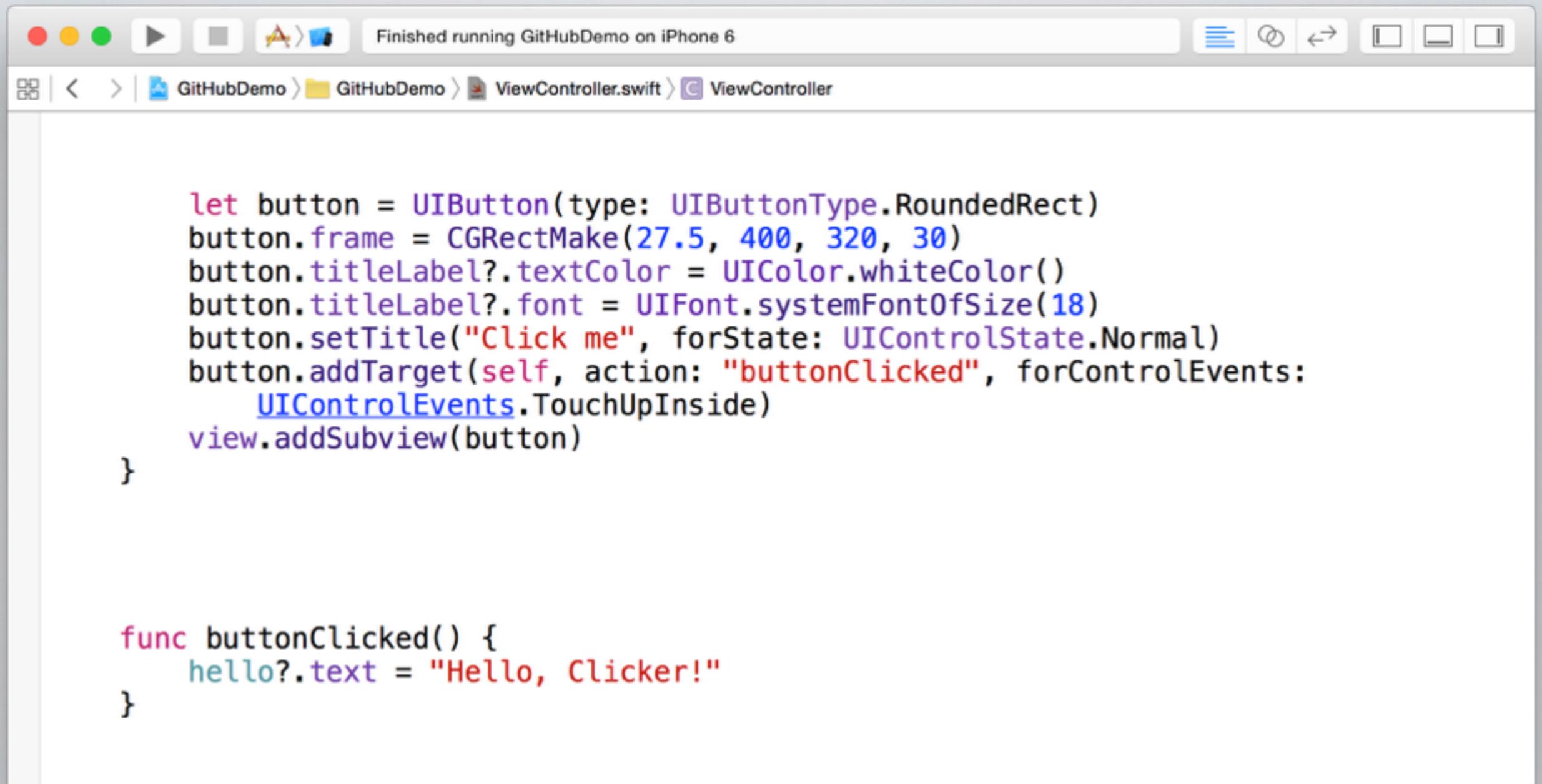
```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)

}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

This is how to create a button programmatically

PROGRAMMATICALLY



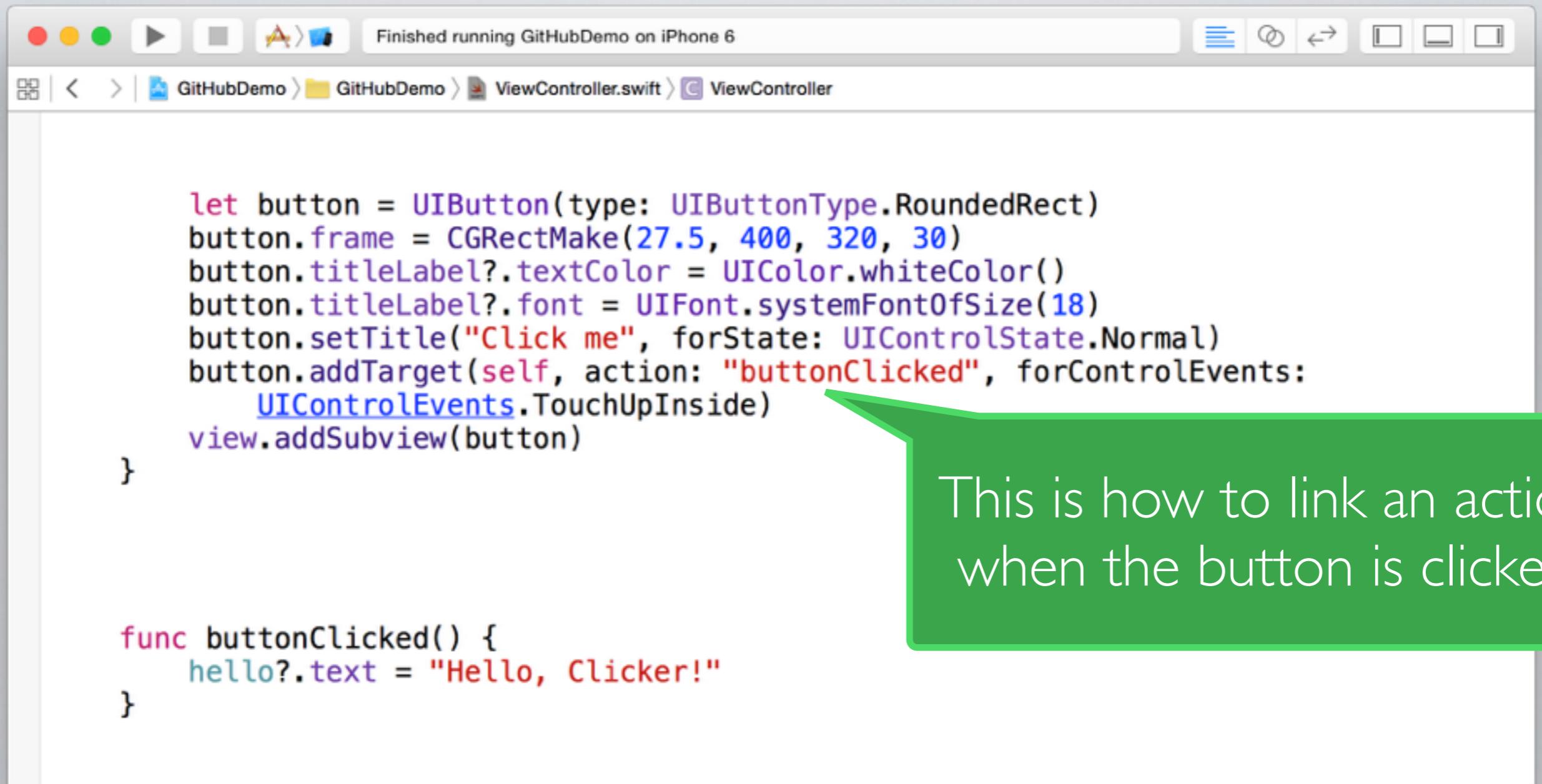
A screenshot of the Xcode IDE interface. The title bar shows "Finished running GitHubDemo on iPhone 6". The navigation bar below shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > ViewController. The main editor area contains Swift code for creating a button programmatically and setting up a tap gesture recognizer.

```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)

}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the title bar "Finished running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > ViewController. The main editor area contains the following Swift code:

```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)

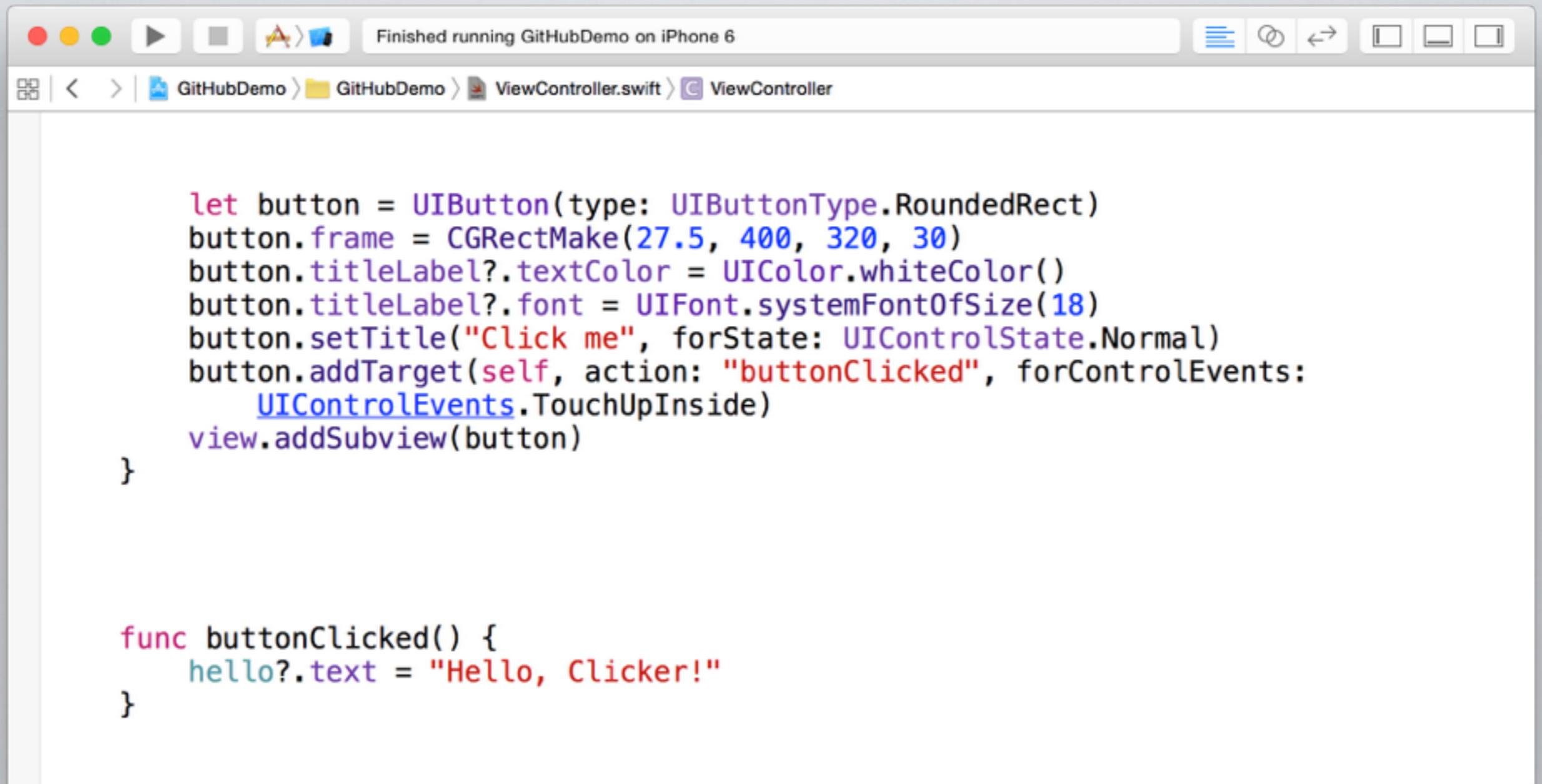
}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

A green callout bubble with a black outline points from the text "This is how to link an action when the button is clicked" to the line `button.addTarget(self, action: "buttonClicked", forControlEvents: UIControlEvents.TouchUpInside)`.

This is how to link an action when the button is clicked

PROGRAMMATICALLY



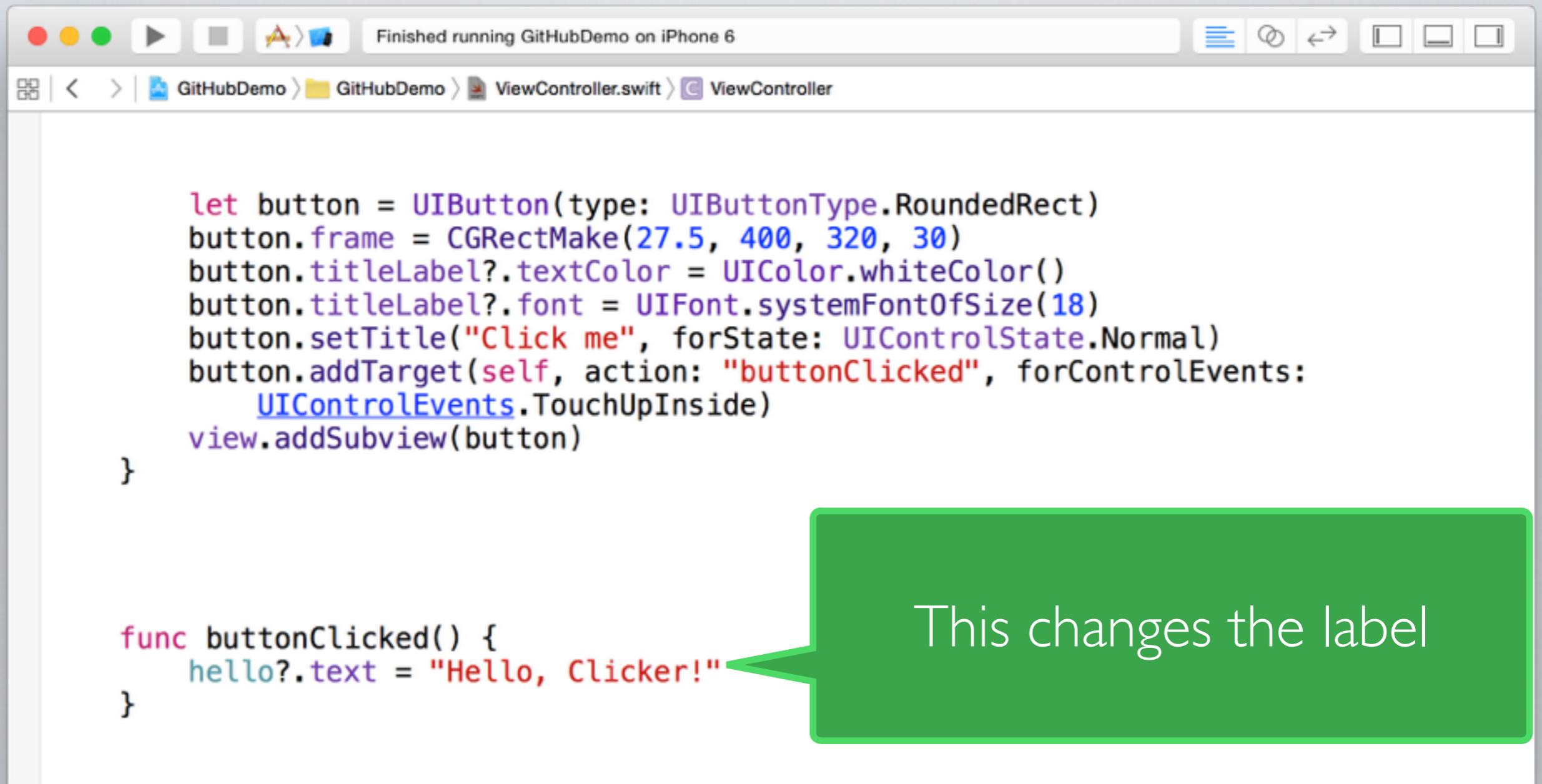
A screenshot of the Xcode IDE interface. The title bar shows "Finished running GitHubDemo on iPhone 6". The navigation bar below shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > ViewController. The main editor area contains Swift code for creating a button programmatically and setting up a tap gesture recognizer.

```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)

}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

PROGRAMMATICALLY



The screenshot shows the Xcode interface with the title bar "Finished running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift > ViewController. The main editor area contains the following Swift code:

```
let button = UIButton(type: UIButtonType.RoundedRect)
button.frame = CGRectMake(27.5, 400, 320, 30)
button.titleLabel?.textColor = UIColor.whiteColor()
button.titleLabel?.font = UIFont.systemFontOfSize(18)
button.setTitle("Click me", forState: UIControlState.Normal)
button.addTarget(self, action: "buttonClicked", forControlEvents:
    UIControlEvents.TouchUpInside)
view.addSubview(button)
}

func buttonClicked() {
    hello?.text = "Hello, Clicker!"
}
```

A green callout bubble with a black outline and a black arrow points from the text "This changes the label" to the line `hello?.text = "Hello, Clicker!"`.

This changes the label

EXERCISE

- Add a UITextField programmatically
- Change text of the UILabel based on what you entered in the UITextField

CODE

A screenshot of a GitHub repository page for 'talentsparkio / GitHubDemo'. The page shows basic repository statistics: 4 commits, 1 branch, 1 release, and 1 contributor. A red circle highlights the 'Tag: BasicProgramma...' dropdown menu. The repository description is 'A quick demo of using the GitHub API to make a simple iOS app — Edit'. The commit history lists three commits:

File	Message	Time
GitHubDemo.xcodeproj	Add playground file	6 days ago
GitHubDemo.xcworkspace	Initial import	7 days ago
GitHubDemo	Demo of how to do things programmatically	an hour ago

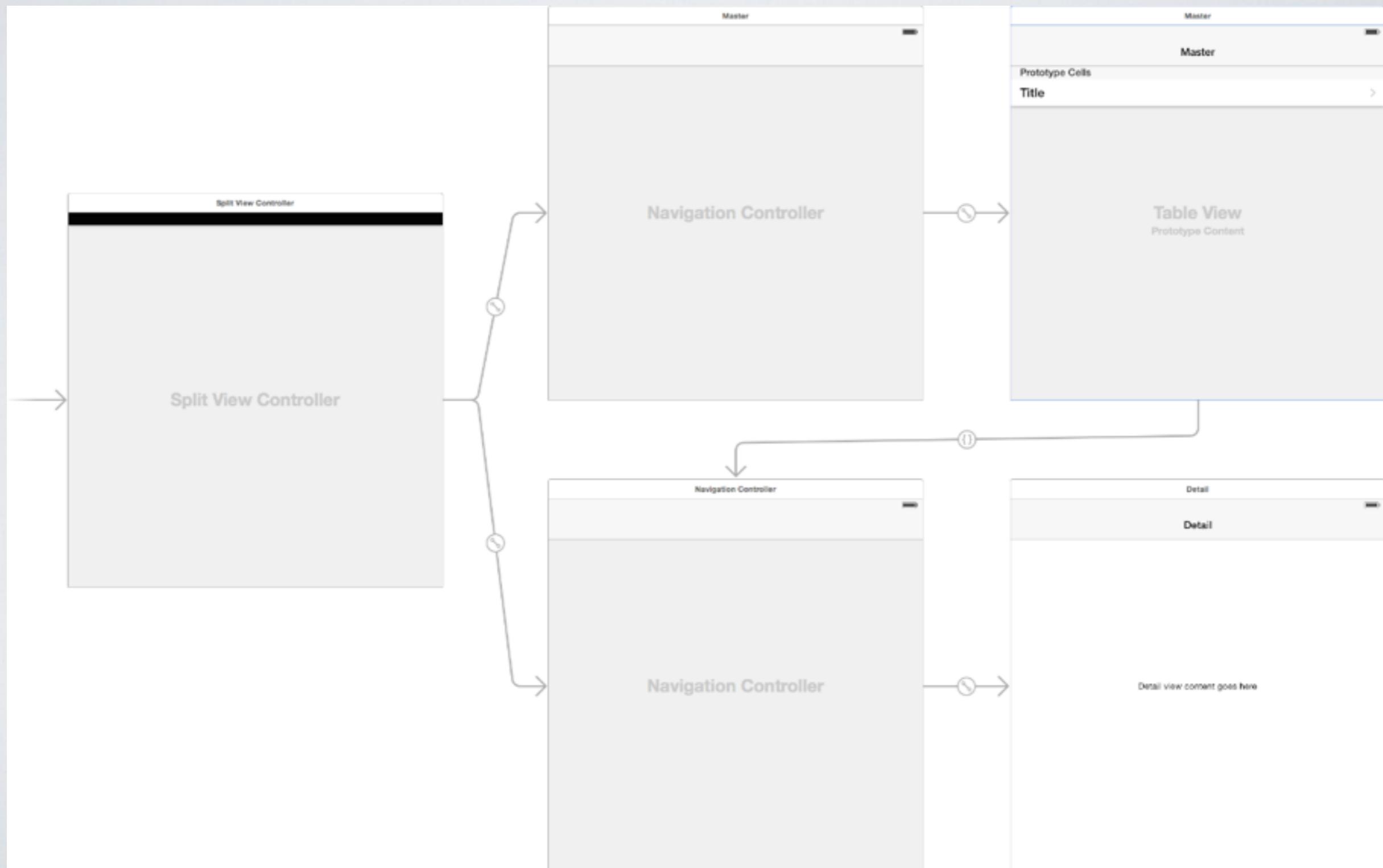
CODE

A screenshot of a GitHub repository page for 'talentsparkio / GitHubDemo'. The page shows basic repository statistics: 4 commits, 1 branch, 1 release, and 1 contributor. A red circle highlights the 'Tag: BasicProgramma...' dropdown menu. The repository description is 'A quick demo of using the GitHub API to make a simple iOS app — Edit'. The commit history lists three commits:

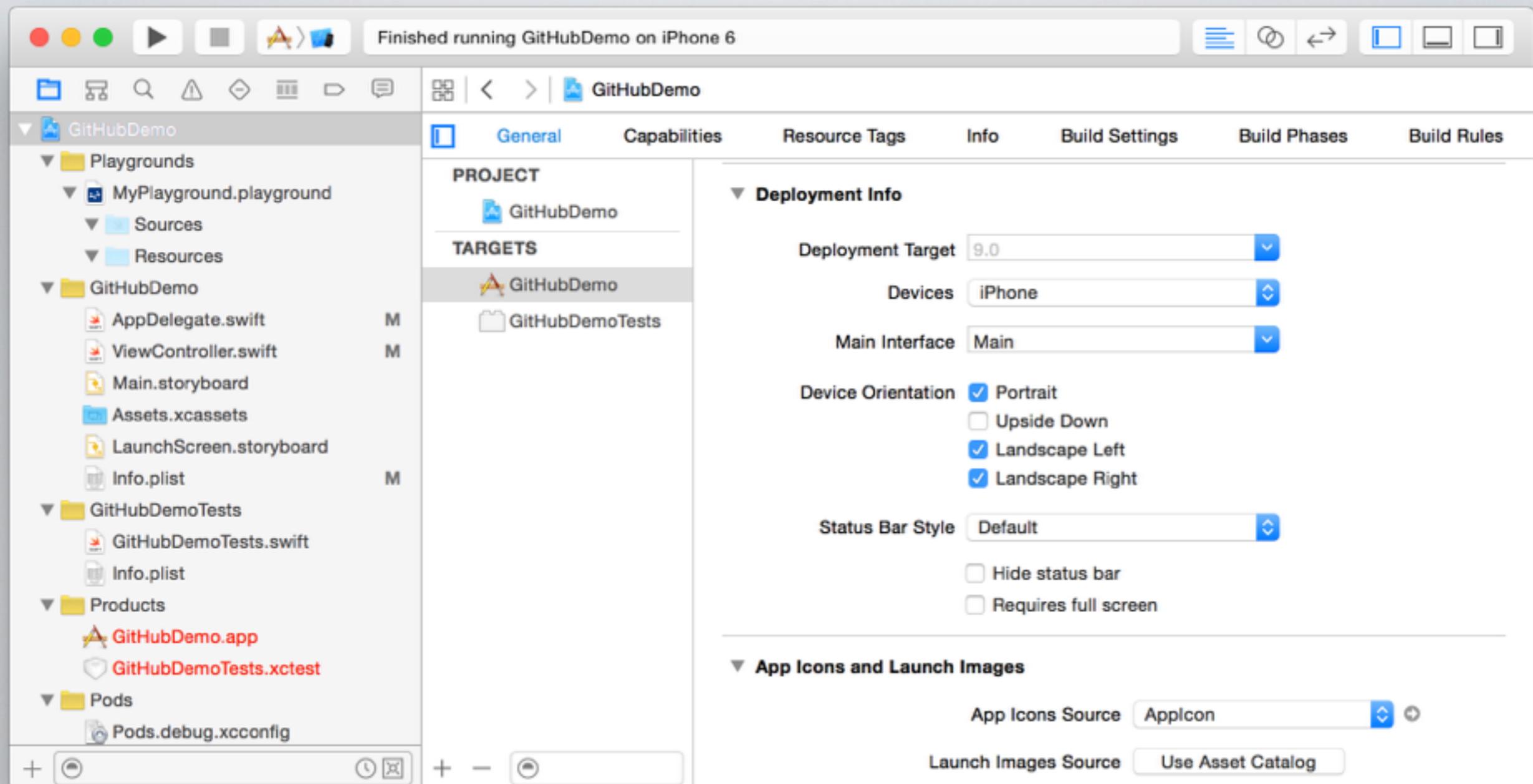
File	Message	Time
GitHubDemo.xcodeproj	Add playground file	6 days ago
GitHubDemo.xcworkspace	Initial import	7 days ago
GitHubDemo	Demo of how to do things programmatically	an hour ago

LET'S DO THIS WITH
STORYBOARDS!

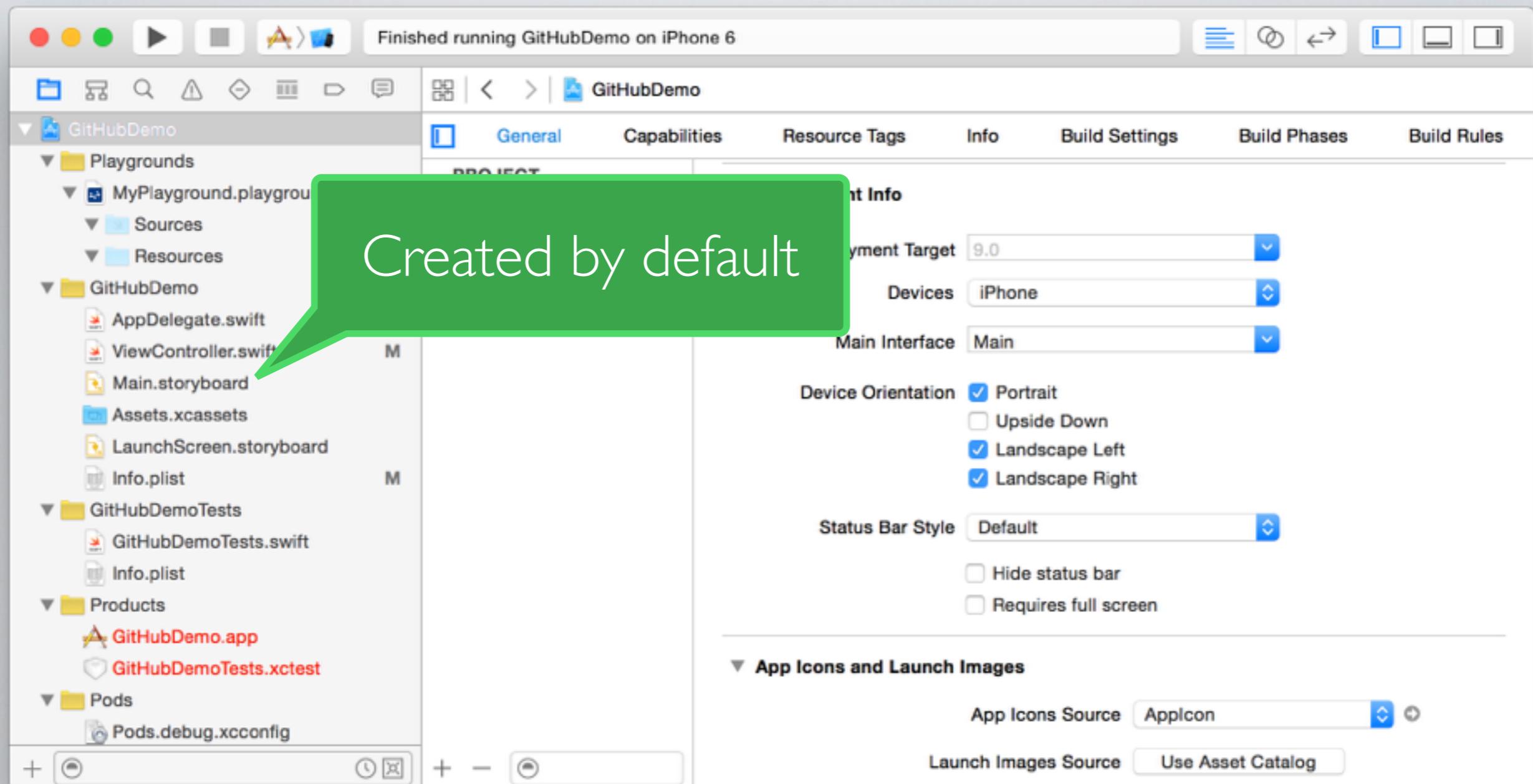
STORYBOARDS



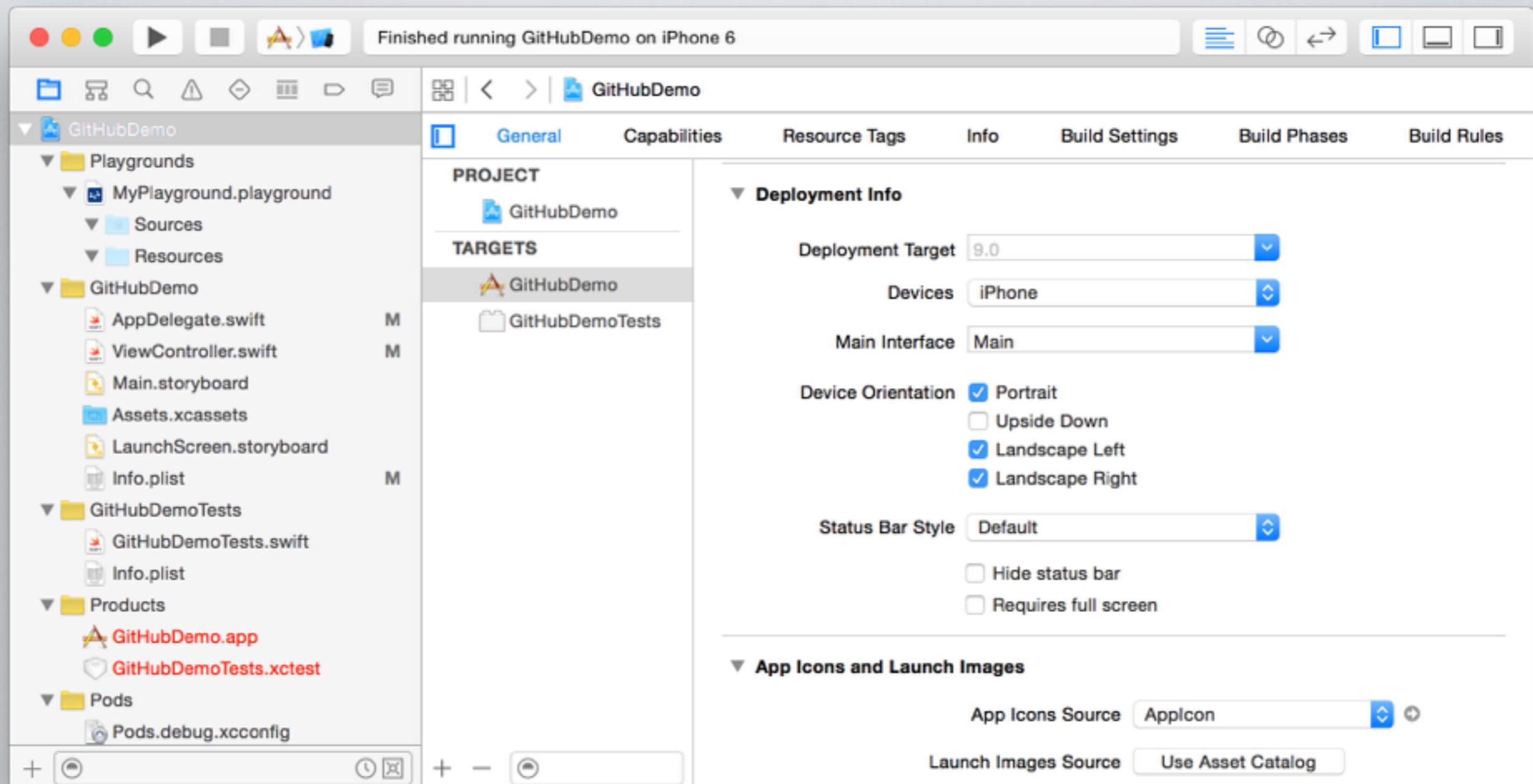
STORYBOARDS



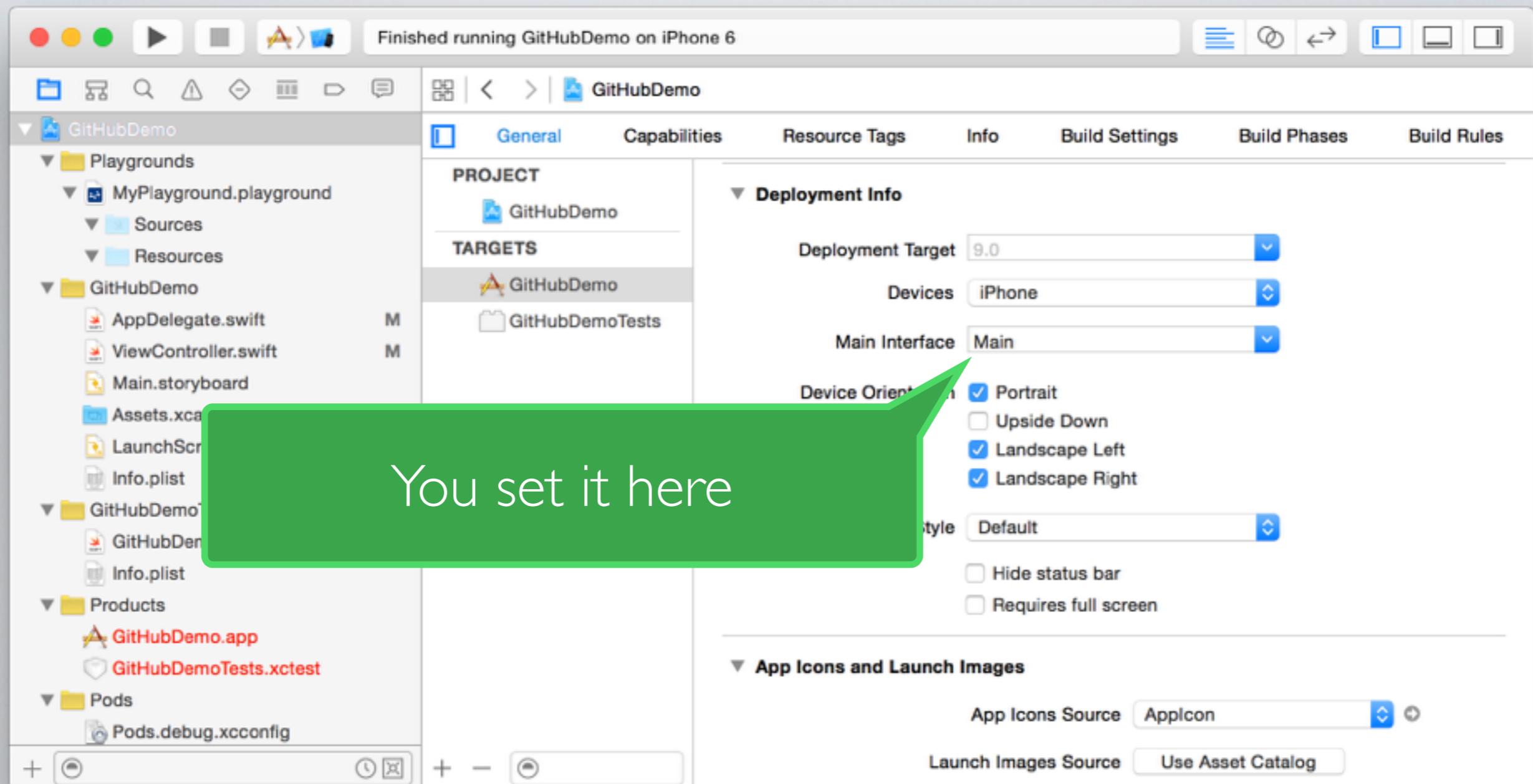
STORYBOARDS



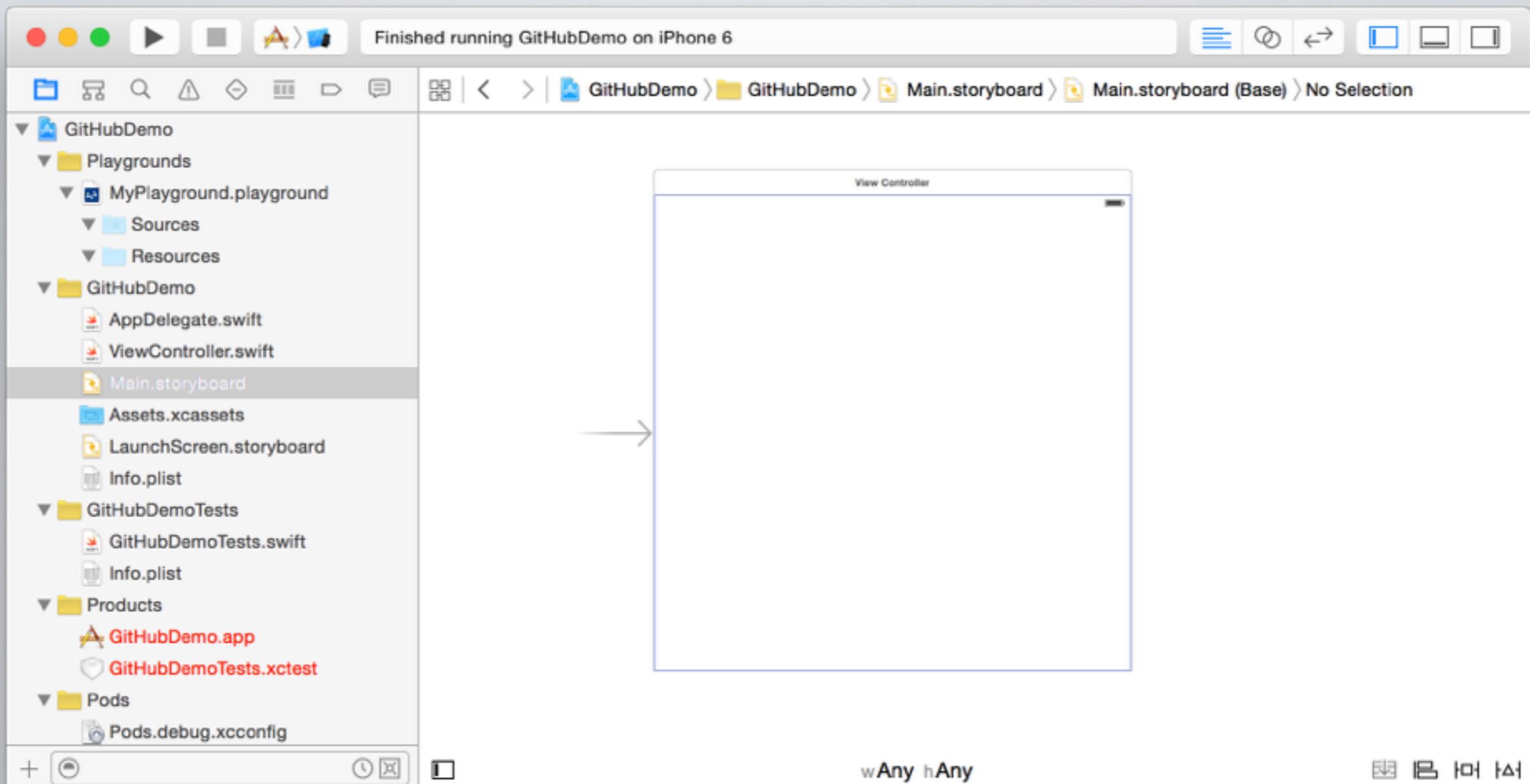
STORYBOARDS



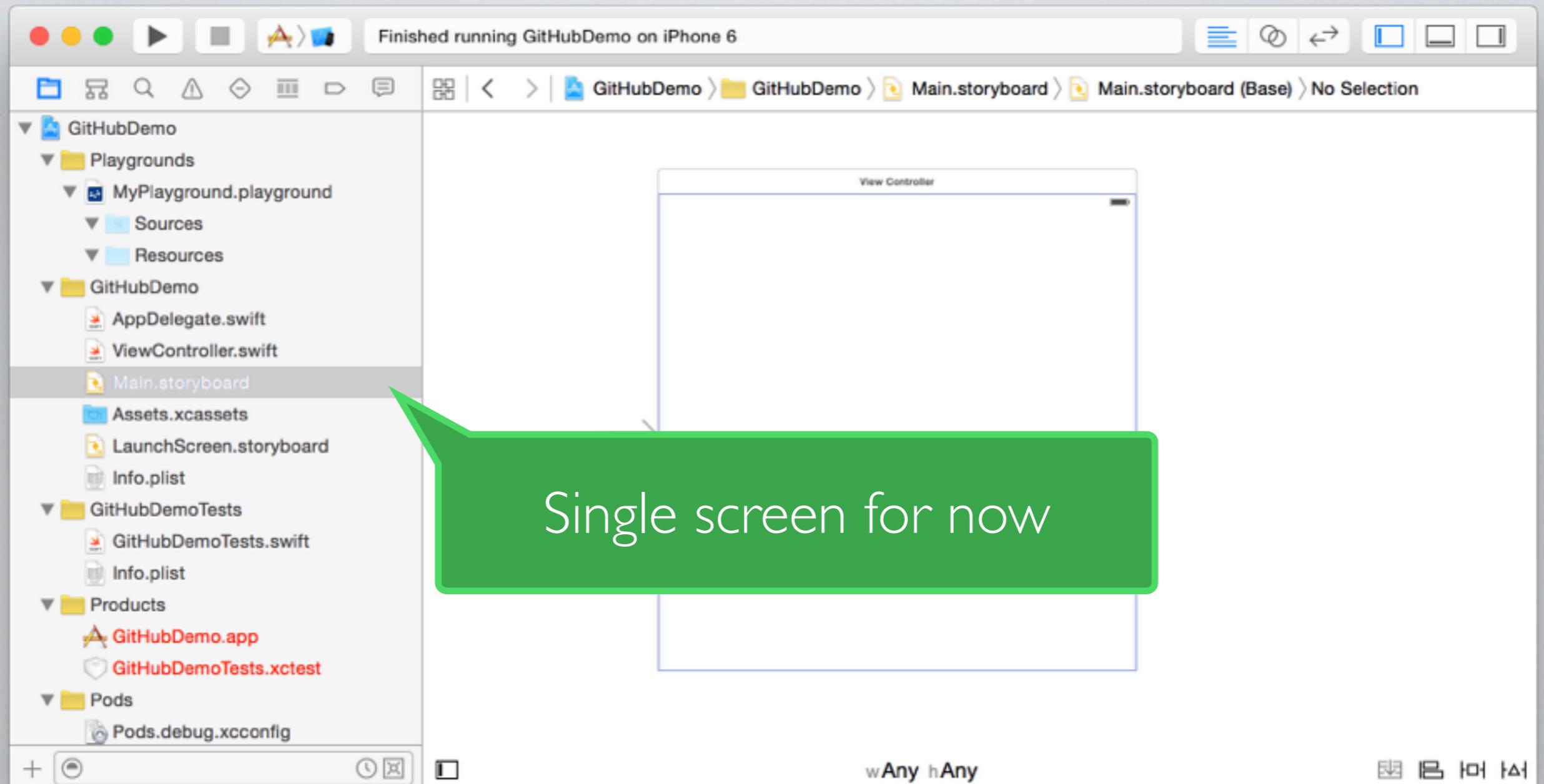
STORYBOARDS



STORYBOARDS



STORYBOARDS



STORYBOARDS

The screenshot shows the Xcode interface with two main panes. The left pane displays the `ViewController.swift` file, which contains the following code:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
        // typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

The right pane shows the storyboard editor with a single blank view controller scene. A large arrow points from the bottom of the code editor towards the storyboard scene. The storyboard interface includes a toolbar at the top, a document outline on the left, and a library on the right containing objects like "Label".

STORYBOARDS

The screenshot shows the Xcode interface with two main panes. The left pane displays the `ViewController.swift` file:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

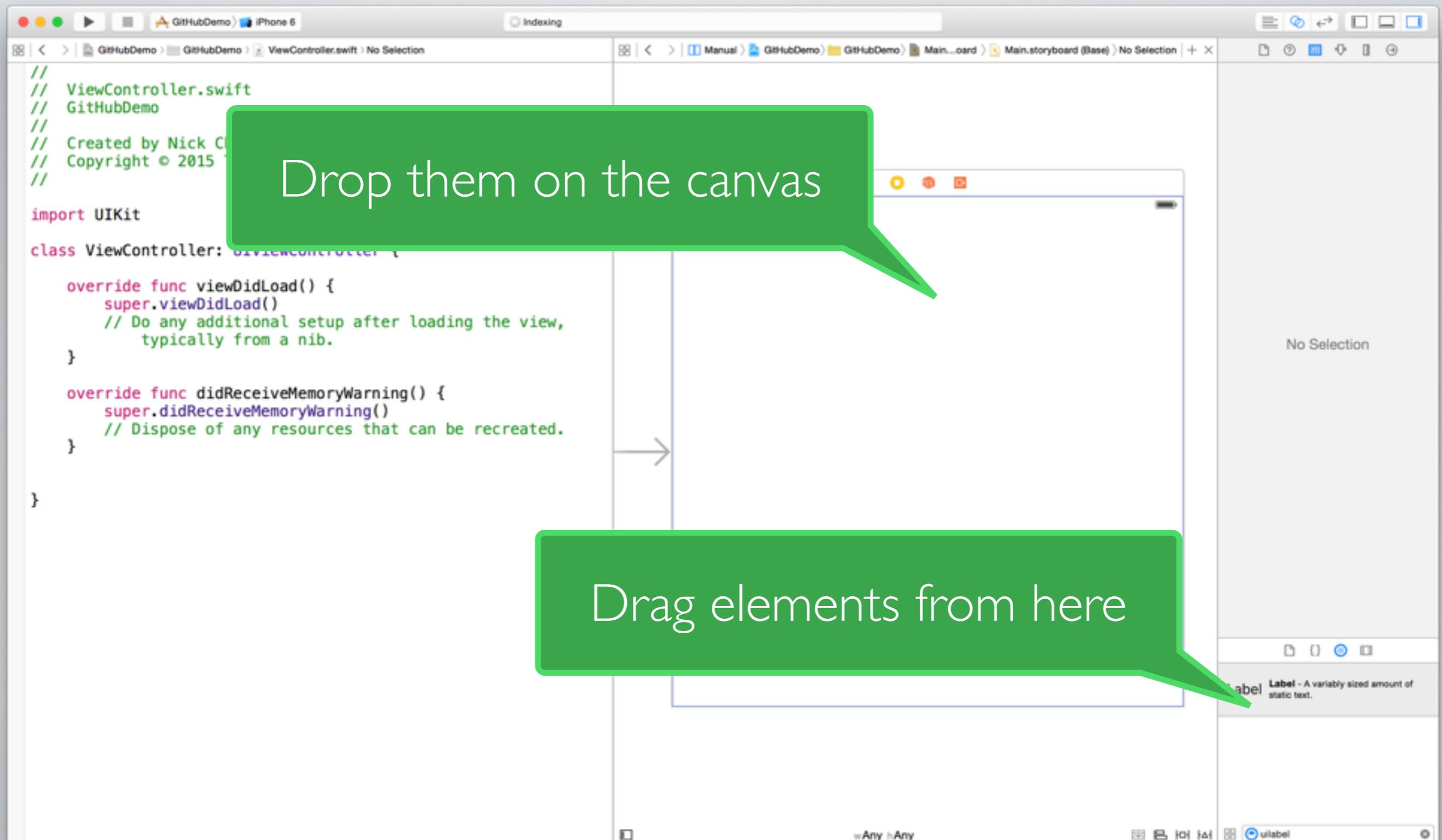
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
        // typically from a nib.
    }

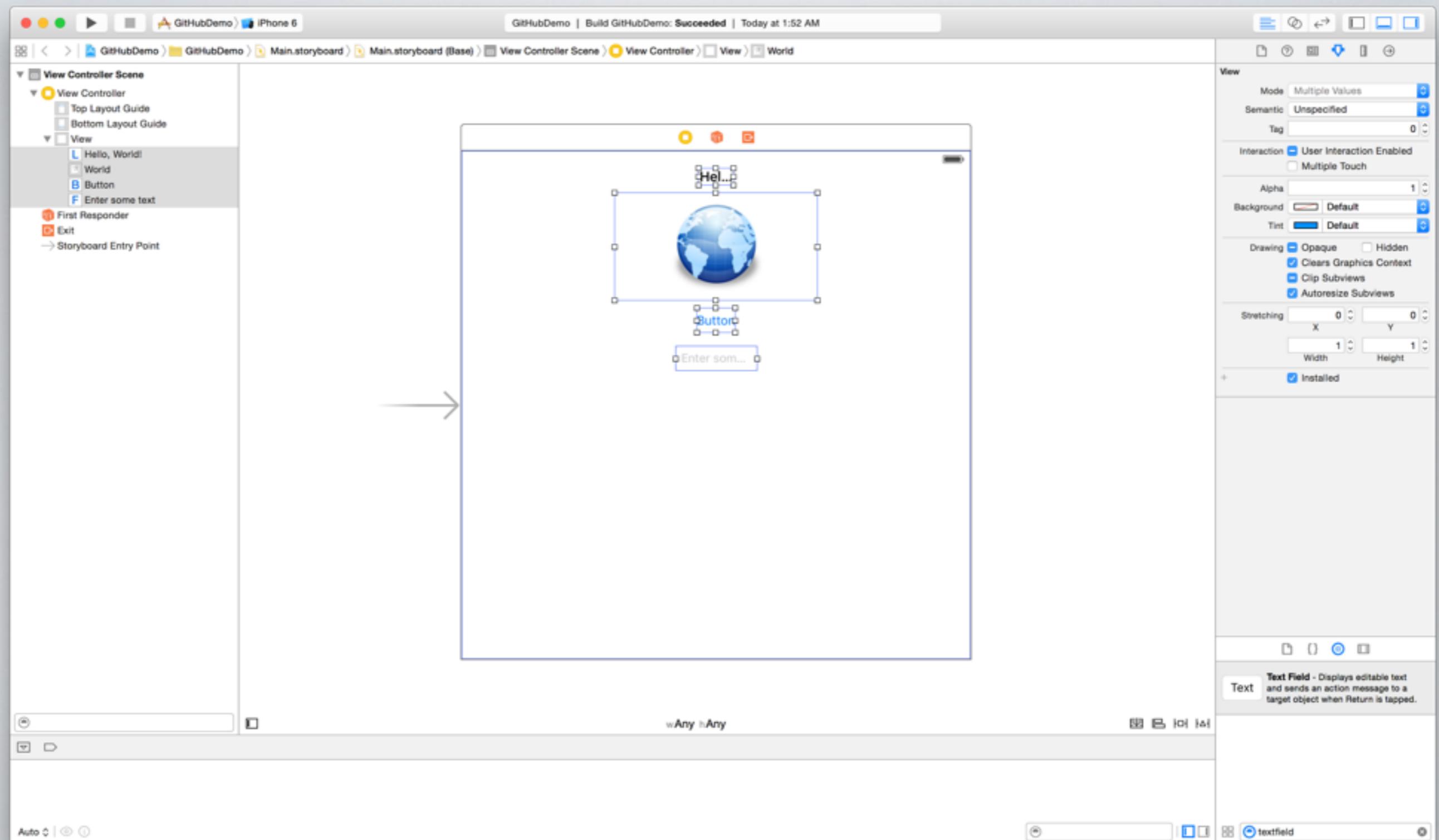
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

The right pane shows the storyboard editor with a single blank view controller scene. A large green callout bubble with the text "Drag elements from here" points to the empty content area of the view controller. The bottom right corner of the storyboard editor shows the standard Xcode interface with toolbars and a status bar.

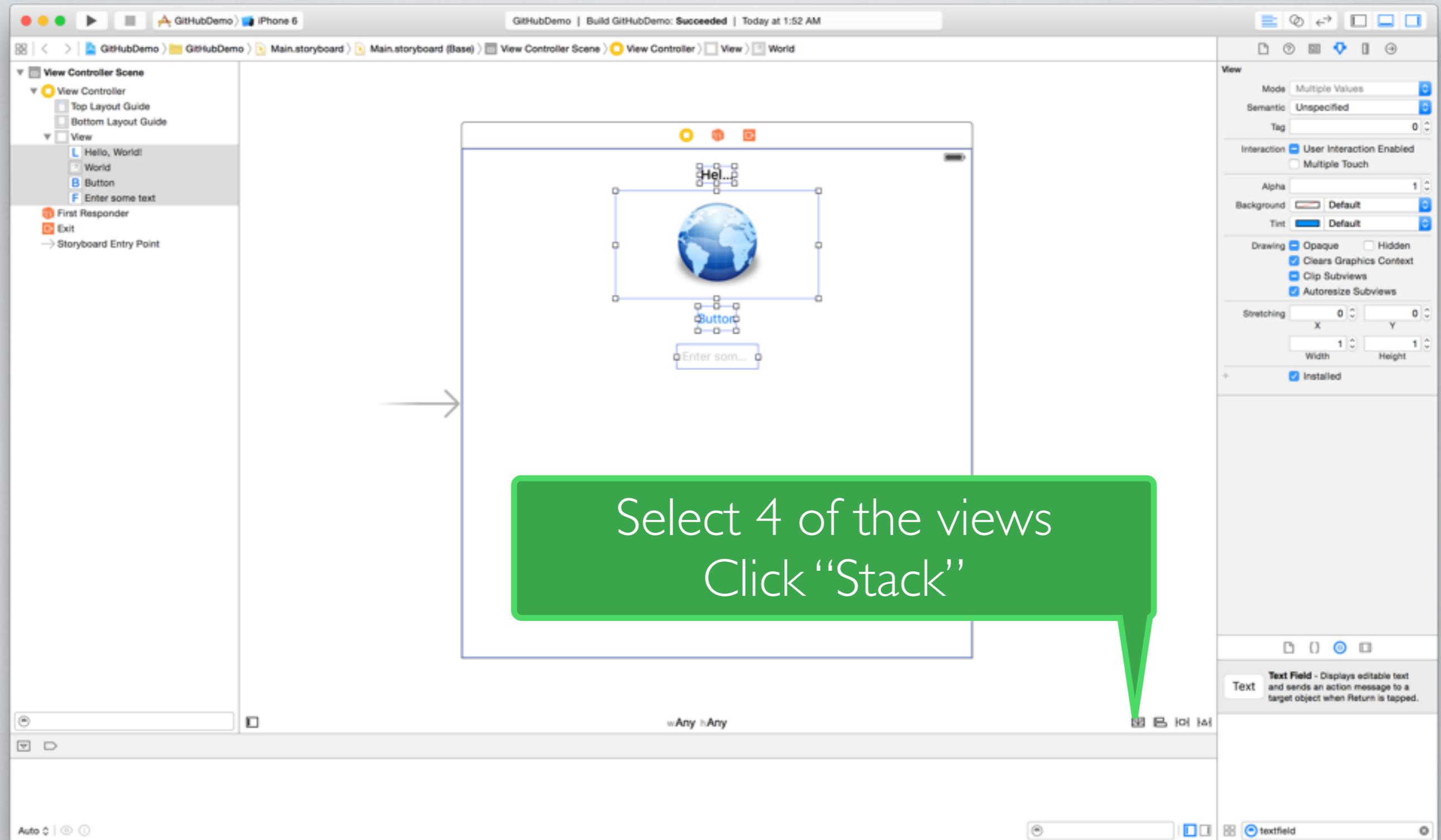
STORYBOARDS



STORYBOARDS



STORYBOARDS



STORYBOARDS

The screenshot shows the Xcode interface with two main panes. The left pane is a code editor displaying `ViewController.swift` for the `GitHubDemo` project. The right pane is a storyboard preview window.

`ViewController.swift` content:

```
// View Controller.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

The storyboard preview shows a single screen titled "Hello, World!". It contains a globe image, a button labeled "Click me", and a text input field with placeholder text "Enter some text".

STORYBOARDS

The screenshot shows the Xcode interface with two main panes. The left pane is the code editor displaying `ViewController.swift` for the `GitHubDemo` project. The right pane is the storyboard editor showing a single view controller scene.

`ViewController.swift` content:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

The storyboard scene contains a single view with the title "Hello, World!". Inside the view, there is a globe image, a button labeled "Click me", and a text input field with the placeholder "Enter some text". A blue line connects the `viewDidLoad()` method in the code to the globe image in the storyboard.

STORYBOARDS

The screenshot shows the Xcode interface with a storyboard file open. On the left, the code editor displays `ViewController.swift` with the following content:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

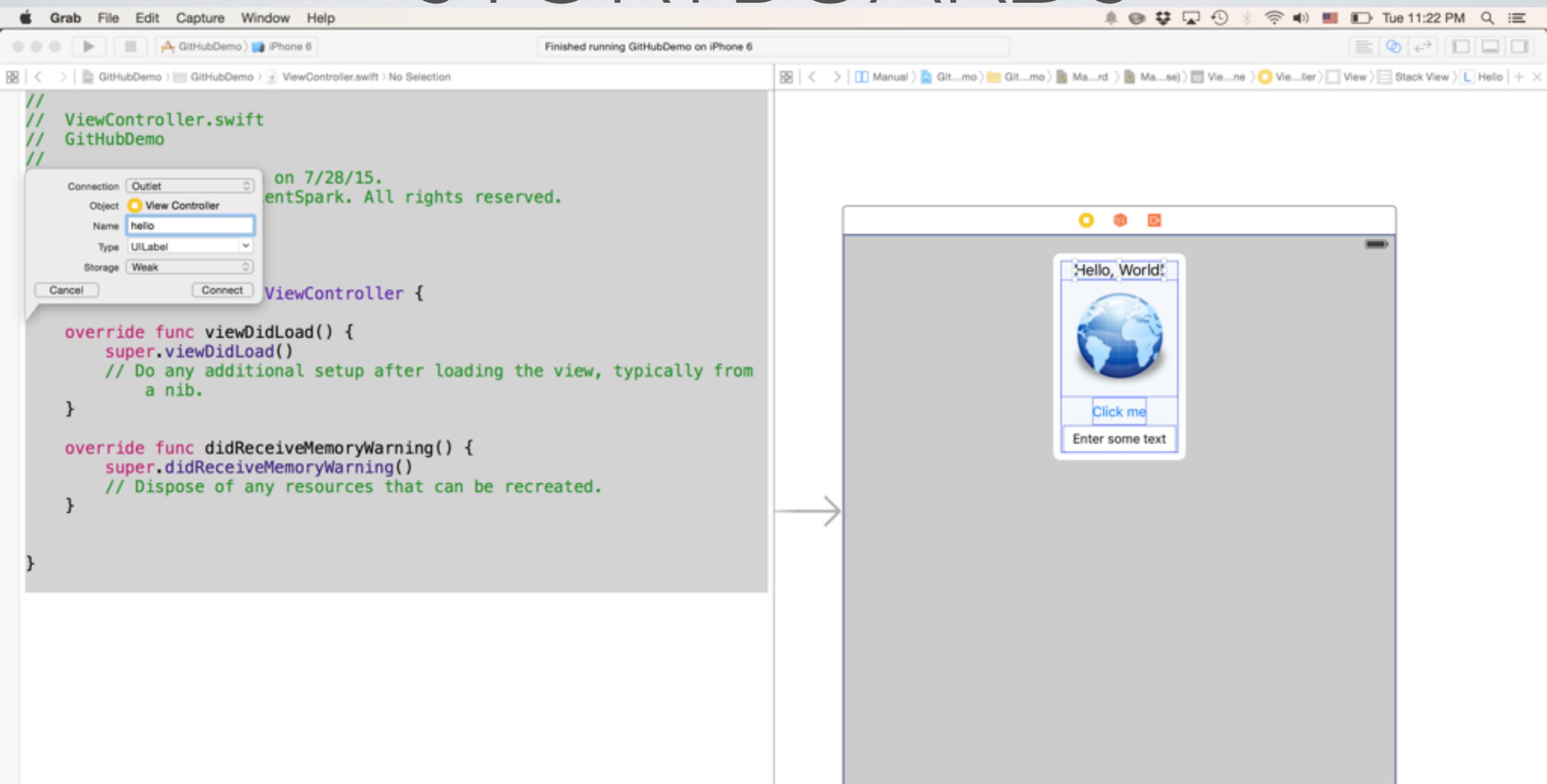
import UIKit

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

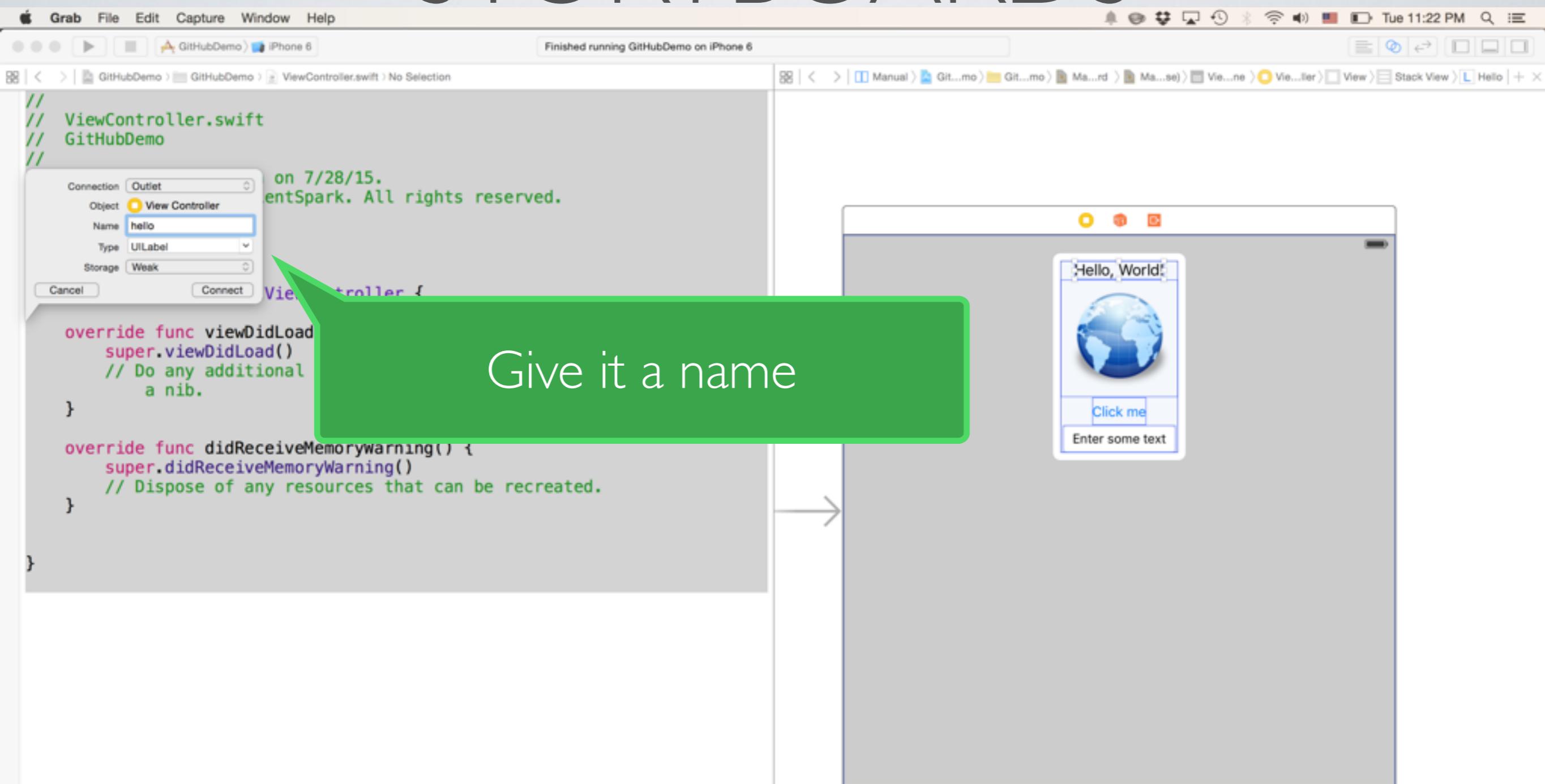
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

A blue line connects the `viewDidLoad()` method to the storyboard preview window on the right. A green callout bubble with the text "Ctrl + Drag" points to this connection line. The storyboard preview shows a single view controller with a label containing "Hello, World!" and a circular image.

STORYBOARDS



STORYBOARDS



STORYBOARDS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

GitHubDemo iPhone 6 Finished running GitHubDemo on iPhone 6

GitHubDemo GitHubDemo ViewController.swift ViewController

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

Manual Git...mo Git...mo Ma...rd Ma...se Vie...ne Vie...ller View Stack View Hello

The image shows a screenshot of the Xcode IDE. On the left, the code editor displays the ViewController.swift file. The code defines a ViewController class that inherits from UIViewController. It contains an outlet for a UILabel named 'hello'. The viewDidLoad method is overridden to perform any additional setup after the view is loaded. The didReceiveMemoryWarning method is also overridden to dispose of any resources that can be recreated. On the right, a large preview window shows a storyboard scene for an iPhone 6. The scene consists of a main view containing a label with the text "Hello, World!", a button labeled "Click me", and a text input field with the placeholder "Enter some text". A small arrow points from the code editor towards the storyboard preview.

STORYBOARDS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

GitHubDemo iPhone 6 Finished running GitHubDemo on iPhone 6

GitHubDemo GitHubDemo ViewController.swift ViewController

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be
    }
}
```

This forms a relationship

The image shows a screenshot of the Xcode IDE. On the left, the code editor displays `ViewController.swift` with the following content:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be
    }
}
```

A green callout box with the text "This forms a relationship" is positioned over the `@IBOutlet` line. A green arrow points from this callout to the storyboard preview on the right.

The storyboard preview shows a single screen titled "Hello, World!". It contains a label with the text "Hello", a globe icon, a button labeled "Click me", and a text input field with the placeholder "Enter some text".

STORYBOARDS

The image shows a screenshot of the Xcode IDE. On the left, the code editor displays `ViewController.swift` for the `GitHubDemo` project. A red oval highlights the declaration of three outlets: `@IBOutlet weak var hello: UILabel!`, `@IBOutlet weak var button: UIButton!`, and `@IBOutlet weak var textField: UITextField!`. On the right, the storyboard preview window shows a single view controller with a title bar reading "Hello, World!". Inside the view, there is a globe image, a blue button labeled "Click me", and a text field with the placeholder "Enter some text". An arrow points from the code editor towards the storyboard preview.

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!
    @IBOutlet weak var button: UIButton!
    @IBOutlet weak var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

STORYBOARDS

The screenshot shows the Xcode interface. On the left, the code editor displays `ViewController.swift` with the following content:

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var hello: UILabel!
    @IBOutlet weak var button: UIButton!
    @IBOutlet weak var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

A red oval highlights the three `@IBOutlet` declarations. On the right, the storyboard preview window shows a single view controller with a title bar "Hello, World!". Inside, there is a globe image, a blue button labeled "Click me", and a text field with placeholder text "Enter some text".

Renaming these are tricky. I'll demonstrate.

STORYBOARDS

The screenshot shows the Xcode interface with two main panes. On the left, the code editor displays `ViewController.swift` for the `GitHubDemo` project. The code includes comments about the file being created by Nick Chen on 7/28/15 and copyright © 2015 TalentSpark. It defines a `buttonClicked` action for a `UIButton` named `button` with the event `Touch Up Inside`. The code also includes standard view controller overrides for `viewDidLoad` and `didReceiveMemoryWarning`.

A callout bubble from the storyboard connection highlights the `buttonClicked` action in the code. The storyboard preview on the right shows a single view controller with a label "Hello, World!", a globe image, a button labeled "Click me", and a text field with placeholder text "Enter some text".

```
// ViewController.swift
// GitHubDemo
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.
//

Connection Action
Object: View Controller
Name: buttonClicked
Type: UIButton
Event: Touch Up Inside
Arguments: Sender

ViewController {
    @IBOutlet weak var hello: UILabel!
    @IBOutlet weak var button: UIButton!
    @IBOutlet weak var extField: UITextField!

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from
    // a nib.
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

}
```

Finished running GitHubDemo on iPhone 6

Tue 11:35 PM

2015-0...3:43 PM avground

STORYBOARDS



The screenshot shows the Xcode interface with the title bar "Finished running GitHubDemo on iPhone 6". The navigation bar shows the project structure: GitHubDemo > GitHubDemo > ViewController.swift. The code editor displays the following Swift code:

```
@IBOutlet weak var hello: UILabel!
@IBOutlet weak var button: UIButton!
@IBOutlet weak var textField: UITextField!

@IBAction func buttonClicked(sender: UIButton) {
    hello.text = "Hello, Clicker!"
}

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}
```

A red rounded rectangle highlights the `@IBAction` block, specifically the line `hello.text = "Hello, Clicker!"`.

EXERCISE

- Change text of the UILabel based on what you entered in the UITextField

HOW DOES IT WORK?

GitHubDemo

Search

Favorites

- Development
- Dropbox
- All My Files
- iCloud Drive
- AirDrop
- Applications
- Desktop
- Documents
- Downloads

Devices

- Remote Disc

Tags

- Red
- Orange
- Yellow
- Green
- Blue
- Brown

Name Date Modified

Name	Date Modified
GitHubDemo	Today, 11:43 PM
AppDelegate.swift	Today, 10:26 PM
Assets.xcassets	Yesterday, 11:03 PM
Base.lproj	Today, 11:42 PM
LaunchScreen.storyboard	Jul 28, 2015, 7:20 PM
Main storyboard	Today, 11:42 PM
Info.plist	Today, 10:31 PM
Playgrounds	Aug 2, 2015, 6:46 PM
ViewController.swift	Today, 11:39 PM
GitHubDemo.xcodeproj	Today, 10:31 PM
GitHubDemo.xcworkspace	Jul 28, 2015, 9:32 PM
GitHubDemoTests	Jul 28, 2015, 7:20 PM
LICENSE	Yesterday, 11:03 PM
Podfile	Jul 28, 2015, 7:20 PM
Podfile.lock	Jul 28, 2015, 7:20 PM
Pods	Today, 11:39 PM

Macintosh HD > Users > vazex > Devel > Talent > GitHub > GitHub > Base.lproj > Main storyboard

1 of 16 selected, 133.39 GB available

Main storyboard

6 KB
Created Yesterday
Modified 11:42 PM
Last opened 11:42 PM
Add Tags...

HOW DOES IT WORK?

GitHubDemo

Search

Favorites

- Development
- Dropbox
- All My Files
- iCloud Drive
- AirDrop
- Applications
- Desktop
- Documents
- Downloads

Devices

- Remote Disc

Tags

- Red
- Orange
- Yellow
- Green
- Blue
- Durple

Name

Name	Date Modified
GitHubDemo	Today, 11:43 PM
AppDelegate.swift	Today, 10:26 PM
Assets.xcassets	Yesterday, 11:03 PM
Base.lproj	Today, 11:42 PM
LaunchScreen.storyboard	Jul 28, 2015, 7:20 PM
Main.storyboard	Today, 11:42 PM
Info.plist	Today, 10:31 PM
Playground	
ViewController	
GitHubDemo.xcodeproj	
GitHubDemoTests.xcodeproj	
LICENSE	Yesterday, 11:03 PM
Podfile	Jul 28, 2015, 7:20 PM
Podfile.lock	Jul 28, 2015, 7:20 PM
Pods	Today, 11:39 PM

Macintosh HD > Users > vazex > Devel > Talent > GitHub > GitHub > Base.lproj > Main.storyboard

1 of 16 selected, 133.39 GB available

Let's examine this file!

Main.storyboard

6 KB
Created Yesterday
Modified 11:42 PM
Last opened 11:42 PM
Add Tags...

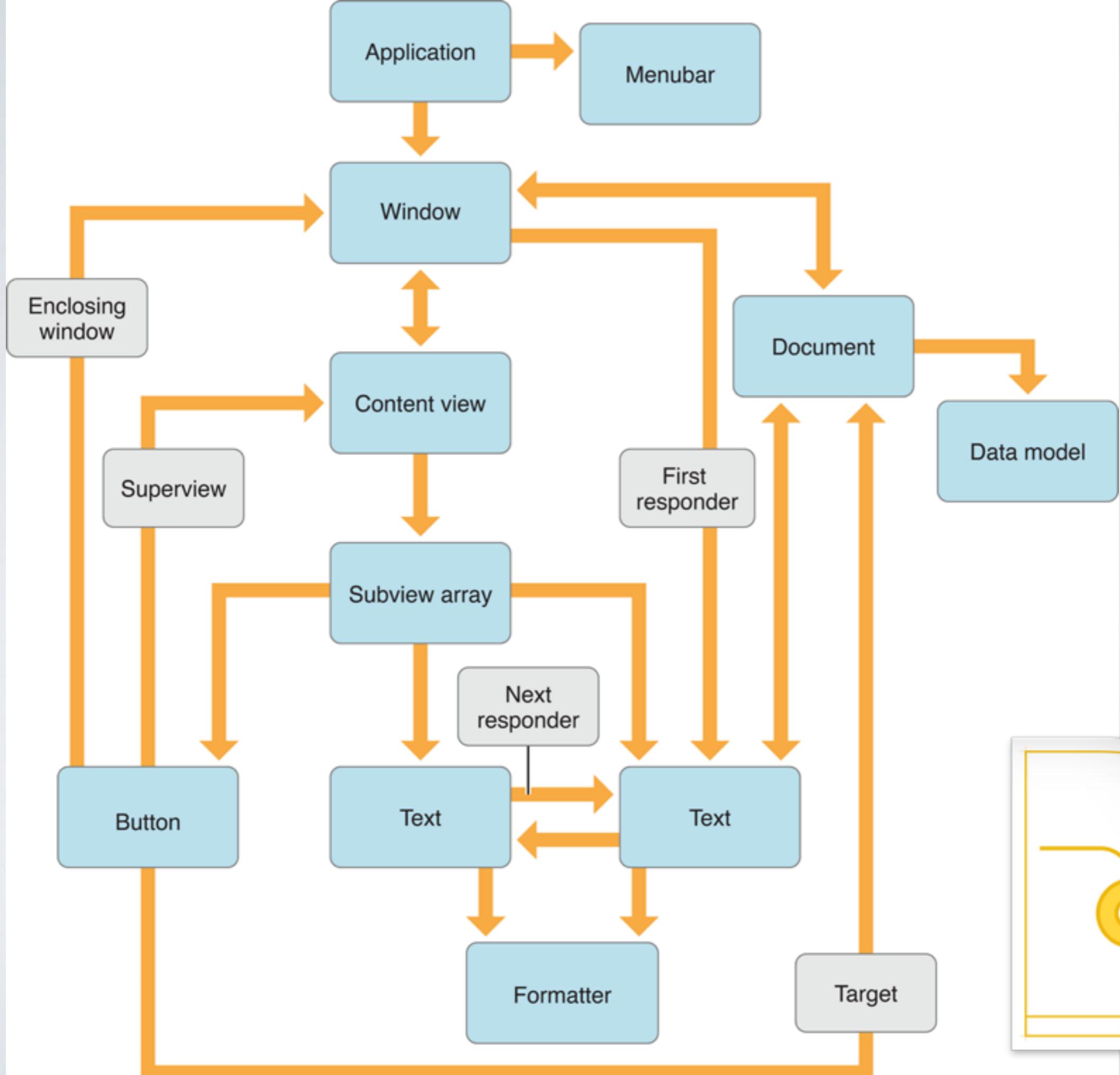
“ANY SUFFICIENTLY ADVANCED
TECHNOLOGY IS
INDISTINGUISHABLE FROM MAGIC”

– Arthur C. Clarke

DEMYSTIFYING THE MAGIC – HOW DOES IT WORK?





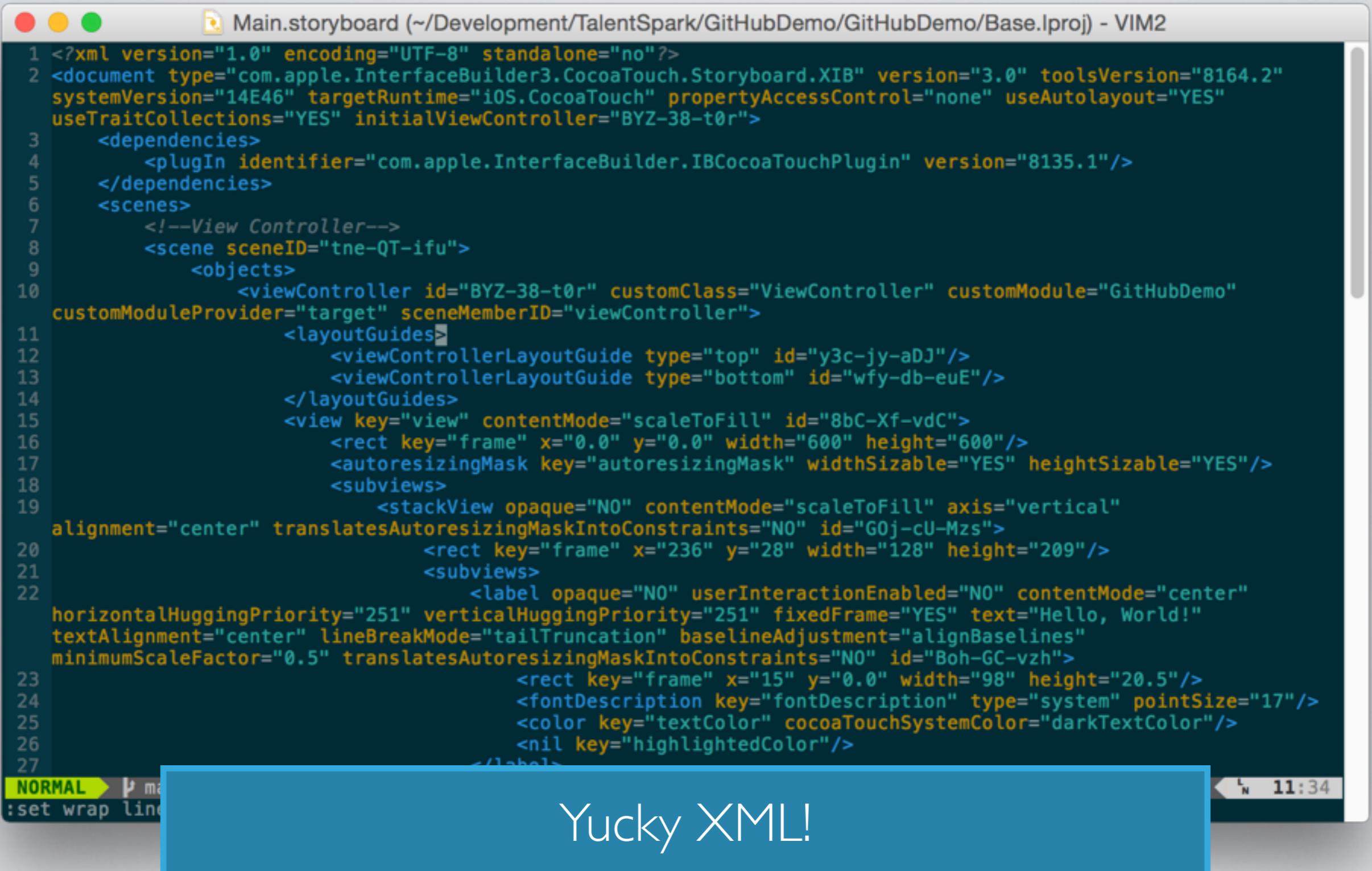


HOW DOES IT WORK?

```
● ● ● Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="8164.2"
  systemVersion="14E46" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
  useTraitCollections="YES" initialViewController="BYZ-38-t0r">
3   <dependencies>
4     <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="8135.1"/>
5   </dependencies>
6   <scenes>
7     <!--View Controller-->
8     <scene sceneID="tne-QT-ifu">
9       <objects>
10         <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="GitHubDemo"
11           customModuleProvider="target" sceneMemberID="viewController">
12           <layoutGuides>
13             <viewControllerLayoutGuide type="top" id="y3c-jy-aDJ"/>
14             <viewControllerLayoutGuide type="bottom" id="wfy-db-euE"/>
15           </layoutGuides>
16           <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
17             <rect key="frame" x="0.0" y="0.0" width="600" height="600"/>
18             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
19             <subviews>
20               <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
21                 alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
22                 <rect key="frame" x="236" y="28" width="128" height="209"/>
23                 <subviews>
24                   <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
25                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
26                     textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
27                     minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
28                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
29                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
30                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
31                     <nil key="highlightedColor"/>
32                   </label>
33                 </subviews>
34               </stackView>
35             </subviews>
36           </view>
37         </viewController>
38       </objects>
39     </scene>
40   </scenes>
41 </document>
```

NORMAL ➤ ↵ master > GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml < 15% ↵ 11:34
:set wrap linebreak nolist

HOW DOES IT WORK?



Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.iproj) - VIM2

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="8164.2"
  systemVersion="14E46" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
  useTraitCollections="YES" initialViewController="BYZ-38-t0r">
3   <dependencies>
4     <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="8135.1"/>
5   </dependencies>
6   <scenes>
7     <!--View Controller-->
8     <scene sceneID="tne-QT-ifu">
9       <objects>
10         <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="GitHubDemo"
11           customModuleProvider="target" sceneMemberID="viewController">
12           <layoutGuides>
13             <viewControllerLayoutGuide type="top" id="y3c-jy-aDJ"/>
14             <viewControllerLayoutGuide type="bottom" id="wfy-db-euE"/>
15           </layoutGuides>
16           <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
17             <rect key="frame" x="0.0" y="0.0" width="600" height="600"/>
18             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
19             <subviews>
20               <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
21                 alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
22                 <rect key="frame" x="236" y="28" width="128" height="209"/>
23                 <subviews>
24                   <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
25                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
26                     textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
27                     minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
28                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
29                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
30                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
31                     <nil key="highlightedColor"/>
32                   </label>
33                 </subviews>
34               </stackView>
35             </subviews>
36           </view>
37         </viewController>
38       </objects>
39     </scene>
40   </scenes>
41 </document>
```

NORMAL ➤ ↵ m :set wrap line 11:34

Yucky XML!

HOW DOES IT WORK?

```
● ○ ● Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="8164.2"
  systemVersion="14E46" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
  useTraitCollections="YES" initialViewController="BYZ-38-t0r">
3   <dependencies>
4     <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="8135.1"/>
5   </dependencies>
6   <scenes>
7     <!--View Controller-->
8     <scene sceneID="tne-QT-ifu">
9       <objects>
10         <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="GitHubDemo"
customModuleProvider="target" sceneMemberID="viewController">
11           <layoutGuides>
12             <viewControllerLayoutGuide type="top" id="y3c-jy-aDJ"/>
13             <viewControllerLayoutGuide type="bottom" id="wfy-db-euE"/>
14           </layoutGuides>
15           <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
16             <rect key="frame" x="0.0" y="0.0" width="600" height="600"/>
17             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18             <subviews>
19               <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
20                 <rect key="frame" x="236" y="28" width="128" height="209"/>
21                 <subviews>
22                   <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
23                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
24                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
25                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
26                     <nil key="highlightedColor"/>
27                   </label>

```

NORMAL ➤ ↵ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml < 11% ↶ ↷ 8:34

search hit BOTTOM, continuing at TOP

HOW DOES IT WORK?

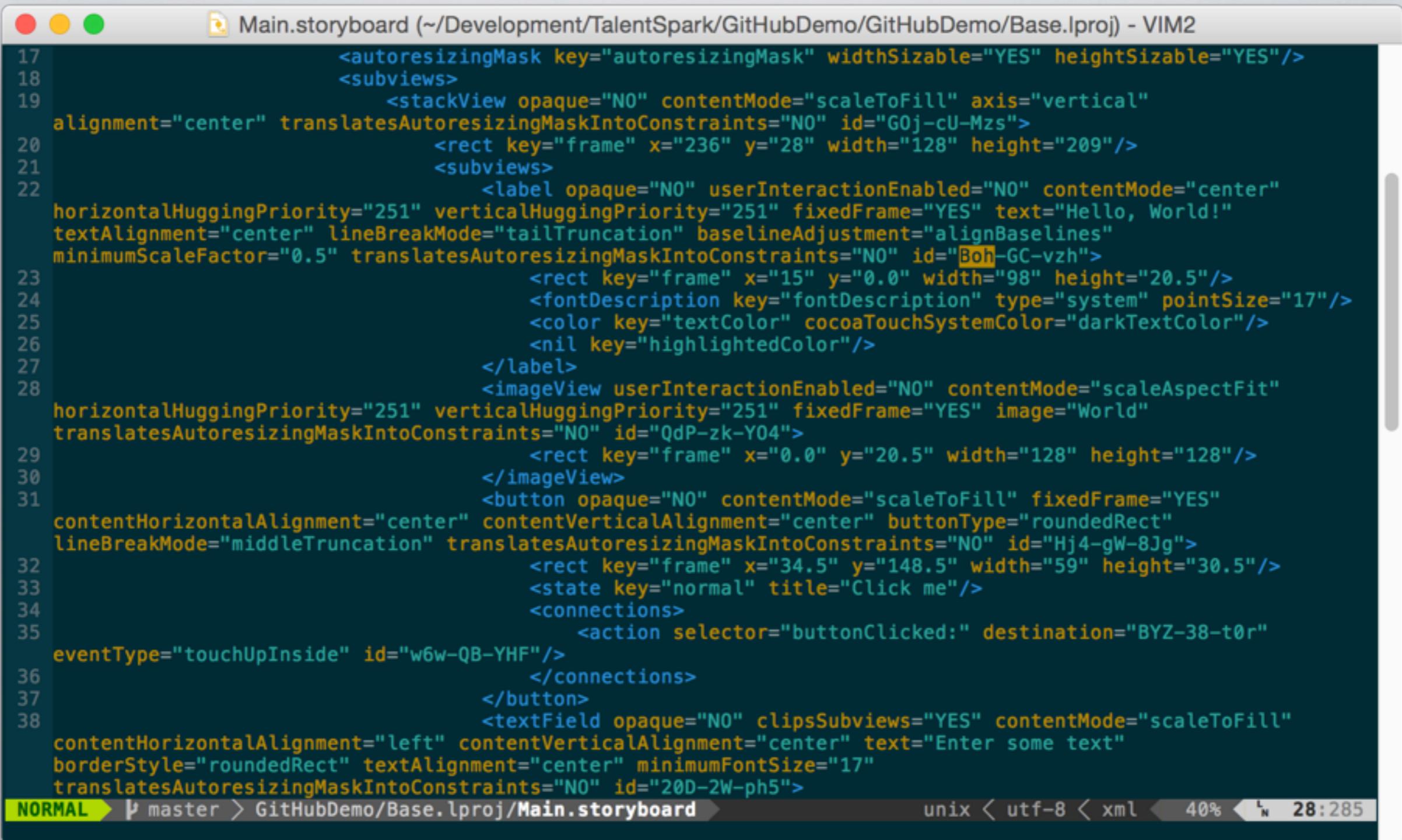
```
● ○ ● Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="8164.2"
  systemVersion="14E46" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
  useTraitCollections="YES" initialViewController="BYZ-38-t0r">
3   <dependencies>
4     <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="8135.1"/>
5   </dependencies>
6   <scenes>
7     <!--View Controller-->
8     <scene sceneID="tne-QT-ifu">
9       <objects>
10         <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="GitHubDemo"
customModuleProvider="target" sceneMemberID="viewController">
11           <layoutGuides>
12             <viewControllerLayoutGuide type="safeArea" id="u2e-2o-4v-aD7U"/>
13             <viewCo
14           </layoutGuides>
15           <view key="view" id="G0j-cU-Mzs">
16             <rect key="frame" x="236" y="28" width="128" height="209"/>
17             <autoresizingMask key="autoresizingMask" flexibleMaxMinWidth="YES" flexibleMinHeight="YES" htSizable="YES"/>
18             <subviews>
19               <stackview opaque="NO" contentMode="scaleToFill" axis="vertical"
alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
20                 <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
21                 <subviews>
22                   <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!" textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
23                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
24                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
25                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
26                     <nil key="highlightedColor"/>
27                   </label>

```

Look! It refers to our ViewController!

NORMAL ➤ ↵ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml < 11% ↵ 8:34
search hit BOTTOM, continuing at TOP

HOW DOES IT WORK?



The screenshot shows a terminal window titled "Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2". The code displayed is the XML configuration for a storyboard scene. The code defines a stack view containing a label with the text "Hello, World!", an image view showing "World", and a button labeled "Click me". It also includes a text field with placeholder text "Enter some text". The XML uses various attributes like autoresizingMask, translatesAutoresizingMaskIntoConstraints, alignment, textAlignment, and various layout priorities.

```
17         <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18     <subviews>
19         <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20             alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21             <rect key="frame" x="236" y="28" width="128" height="209"/>
22             <subviews>
23                 <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                     textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                     minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                     <nil key="highlightedColor"/>
31                 </label>
32                 <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                     translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y04">
35                     <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                 </imageView>
37                 <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                     contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
                     lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
                     <rect key="frame" x="34.5" y="148.5" width="59" height="30.5"/>
                     <state key="normal" title="Click me"/>
                     <connections>
                         <action selector="buttonClicked:" destination="BYZ-38-t0r"
                           eventType="touchUpInside" id="w6w-QB-YHF"/>
                     </connections>
                     </button>
                     <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
                         contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
                         borderStyle="roundedRect" textAlignment="center" minimumFontSize="17"
                         translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

NORMAL ➤ ↵ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?



XML representation of UIViews!

```
17             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18         <subviews>
19             <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20               alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21                 <rect key="frame" x="236" y="28" width="128" height="209"/>
22                 <subviews>
23                     <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                         textAlign="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                         minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                         <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                         <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                         <nil key="highlightedColor"/>
31                     </label>
32                     <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                         translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y04">
35                         <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                     </imageView>
37                     <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                         alignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
39                         lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
40                         <rect key="frame" x="34.5" y="148.5" width="59" height="30.5"/>
41                         <state key="normal" title="Click me"/>
42                         <connections>
43                             <action selector="buttonClicked:" destination="BYZ-38-t0r"
44                               eventType="touchUpInside" id="w6w-QB-YHF"/>
45                         </connections>
46                     </button>
47                     <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
48                       contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
49                         borderStyle="roundedRect" textAlign="center" minimumFontSize="17"
50                         translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?



XML representation of UIViews!

Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2

```
17         <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18     <subviews>
19         <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20             alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21             <rect key="frame" x="236" y="28" width="128" height="209"/>
22             <subviews>
23                 <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                     textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                     minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                     <nil key="highlightedColor"/>
31                 </label>
32                 <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                     translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y04">
35                     <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                 </imageView>
37                 <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                     alignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
39                     lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
40                     <rect key="frame" x="34.5" y="148.5" width="59" height="30.5"/>
41                     <state key="normal" title="Click me"/>
42                     <connections>
43                         <action selector="buttonClicked:" destination="BYZ-38-t0r"
44                             eventType="touchUpInside" id="w6w-QB-YHF"/>
45                     </connections>
46                 </button>
47                 <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
48                     contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
49                     borderStyle="roundedRect" textAlignment="center" minimumFontSize="17"
50                     translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

NORMAL ➤ ↵ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?



XML representation of UIViews!

```
17         <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18         <subviews>
19             <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20               alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21                 <rect key="frame" x="236" y="28" width="128" height="209"/>
22                 <subviews>
23                     <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                         textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                         minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                         <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                         <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                         <nil key="highlightedColor"/>
31                     </label>
32                     <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                         translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y0l">
35                         <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                     </imageView>
37                     <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                         alignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
39                         lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
40                         <rect key="frame" x="34.5" y="148.5" width="59" height="30.5"/>
41                         <state key="normal" title="Click me"/>
42                         <connections>
43                             <action selector="buttonClicked:" destination="BYZ-38-t0r"
44                               eventType="touchUpInside" id="w6w-QB-YHF"/>
45                         </connections>
46                     </button>
47                     <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
48                         contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
49                         borderStyle="roundedRect" textAlignment="center" minimumFontSize="17"
50                         translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?



XML representation of UIViews!

```
17             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18         <subviews>
19             <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20               alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21                 <rect key="frame" x="236" y="28" width="128" height="209"/>
22                 <subviews>
23                     <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                         textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                         minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                         <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                         <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                         <nil key="highlightedColor"/>
31                     </label>
32                     <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                         translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y04">
35                         <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                     </imageView>
37                     <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                         alignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
39                         lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
40                         <rect key="frame" x="34.5" y="159.5" width="59" height="30.5"/>
41                         <state key="normal" title="Click me!"/>
42                         <connections>
43                             <action selector="buttonClicked:" destination="BYZ-38-t0r"
44                               eventType="touchUpInside" id="w6w-QB-YHF"/>
45                         </connections>
46                     </button>
47                     <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
48                       contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
49                         borderStyle="roundedRect" textAlignment="center" minimumFontSize="17"
50                         translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?

Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2

```
17         <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18     <subviews>
19         <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20             alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21             <rect key="frame" x="236" y="28" width="128" height="209"/>
22             <subviews>
23                 <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                     textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                     minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                     <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                     <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                     <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                     <nil key="highlightedColor"/>
31                 </label>
32                 <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                     horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                     translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y0l">
35                     <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                 </imageView>
37                 <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                     contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
                     lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
                     <rect key="frame" x="34.5" y="159.5" width="59" height="30.5"/>
                     <state key="normal" title="Click me"/>
                     <connections>
                         <action selector="buttonClicked:" destination="BYZ-38-t0r"
                           eventType="touchUpInside" id="w6w-QB-YHF"/>
                     </connections>
                 </button>
                 <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
                     contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
                     borderStyle="roundedRect" textAlignment="center" minimumFontSize="17"
                     translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

Label
Image View
Button

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?



Storyboards just encode the info in XML!

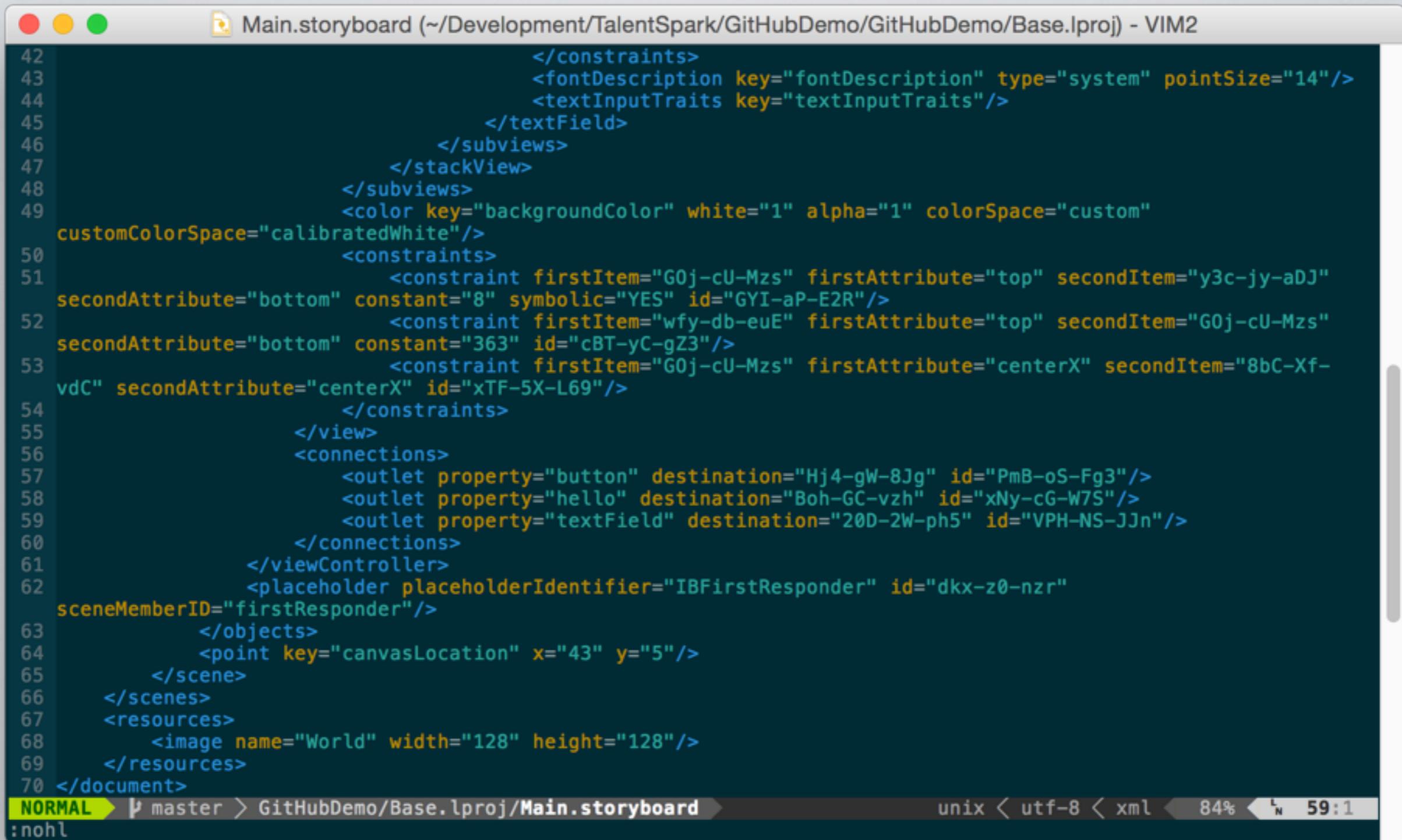
Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2

```
17             <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
18         <subviews>
19             <stackView opaque="NO" contentMode="scaleToFill" axis="vertical"
20               alignment="center" translatesAutoresizingMaskIntoConstraints="NO" id="G0j-cU-Mzs">
21                 <rect key="frame" x="236" y="28" width="128" height="209"/>
22                 <subviews>
23                     <label opaque="NO" userInteractionEnabled="NO" contentMode="center"
24                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="Hello, World!"
25                         textAlign="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
26                         minimumScaleFactor="0.5" translatesAutoresizingMaskIntoConstraints="NO" id="Boh-GC-vzh">
27                         <rect key="frame" x="15" y="0.0" width="98" height="20.5"/>
28                         <fontDescription key="fontDescription" type="system" pointSize="17"/>
29                         <color key="textColor" cocoaTouchSystemColor="darkTextColor"/>
30                         <nil key="highlightedColor"/>
31                     </label>
32                     <imageView userInteractionEnabled="NO" contentMode="scaleAspectFit"
33                       horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" image="World"
34                         translatesAutoresizingMaskIntoConstraints="NO" id="QdP-zk-Y04">
35                         <rect key="frame" x="0.0" y="20.5" width="128" height="128"/>
36                     </imageView>
37                     <button opaque="NO" contentMode="scaleToFill" fixedFrame="YES"
38                         alignment="center" contentVerticalAlignment="center" buttonType="roundedRect"
39                         lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Hj4-gW-8Jg">
40                         <rect key="frame" x="34.5" y="159.5" width="59" height="30.5"/>
41                         <state key="normal" title="Click me!"/>
42                         <connections>
43                             <action selector="buttonClicked:" destination="BYZ-38-t0r"
44                               eventType="touchUpInside" id="w6w-QB-YHF"/>
45                         </connections>
46                     </button>
47                     <textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill"
48                       contentHorizontalAlignment="left" contentVerticalAlignment="center" text="Enter some text"
49                         borderStyle="roundedRect" textAlign="center" minimumFontSize="17"
50                         translatesAutoresizingMaskIntoConstraints="NO" id="20D-2W-ph5">
```

Label
ImageView
Button

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml ➤ 40% ➤ 28:285

HOW DOES IT WORK?

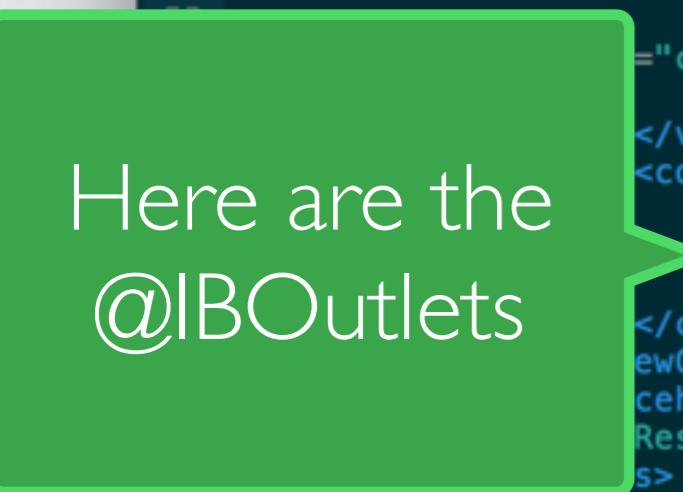


The screenshot shows a terminal window titled "Main.storyboard (~/Development/TalentSpark/GitHubDemo/GitHubDemo/Base.lproj) - VIM2". The code displayed is the XML configuration for a storyboard file. The code includes various storyboard elements like scenes, resources, and view controllers, with line numbers from 42 to 70 visible on the left. The terminal status bar at the bottom indicates the file is in "NORMAL" mode, part of the "master" branch, and the current file is "Main.storyboard". The status bar also shows the encoding as "utf-8", the file type as "xml", the zoom level as "84%", and the current line as "59:1".

```
42             </constraints>
43             <fontDescription key="fontDescription" type="system" pointSize="14"/>
44             <textInputTraits key="textInputTraits"/>
45         </textField>
46     </subviews>
47 </stackView>
48 </subviews>
49 <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
customColorSpace="calibratedWhite"/>
50     <constraints>
51         <constraint firstItem="G0j-cU-Mzs" firstAttribute="top" secondItem="y3c-jy-aDJ"
secondAttribute="bottom" constant="8" symbolic="YES" id="GYI-aP-E2R"/>
52         <constraint firstItem="wfY-db-euE" firstAttribute="top" secondItem="G0j-cU-Mzs"
secondAttribute="bottom" constant="363" id="cBT-yC-gZ3"/>
53         <constraint firstItem="G0j-cU-Mzs" firstAttribute="centerX" secondItem="8bC-Xf-
vdC" secondAttribute="centerX" id="xTF-5X-L69"/>
54     </constraints>
55 </view>
56 <connections>
57     <outlet property="button" destination="Hj4-gW-8Jg" id="PmB-oS-Fg3"/>
58     <outlet property="hello" destination="Boh-GC-vzh" id="xNy-cG-W7S"/>
59     <outlet property="textField" destination="20D-2W-ph5" id="VPH-NS-JJn"/>
60 </connections>
61 </viewController>
62 <placeholder placeholderIdentifier="IBFirstResponder" id="dkx-z0-nzr"
sceneMemberID="firstResponder"/>
63 </objects>
64     <point key="canvasLocation" x="43" y="5"/>
65 </scene>
66 </scenes>
67 <resources>
68     <image name="World" width="128" height="128"/>
69 </resources>
70 </document>
```

NORMAL ➤ ↵ master ➤ GitHubDemo/Base.lproj/Main.storyboard ➤ unix < utf-8 < xml < 84% ↶ ↷ 59:1
:nohl

HOW DOES IT WORK?

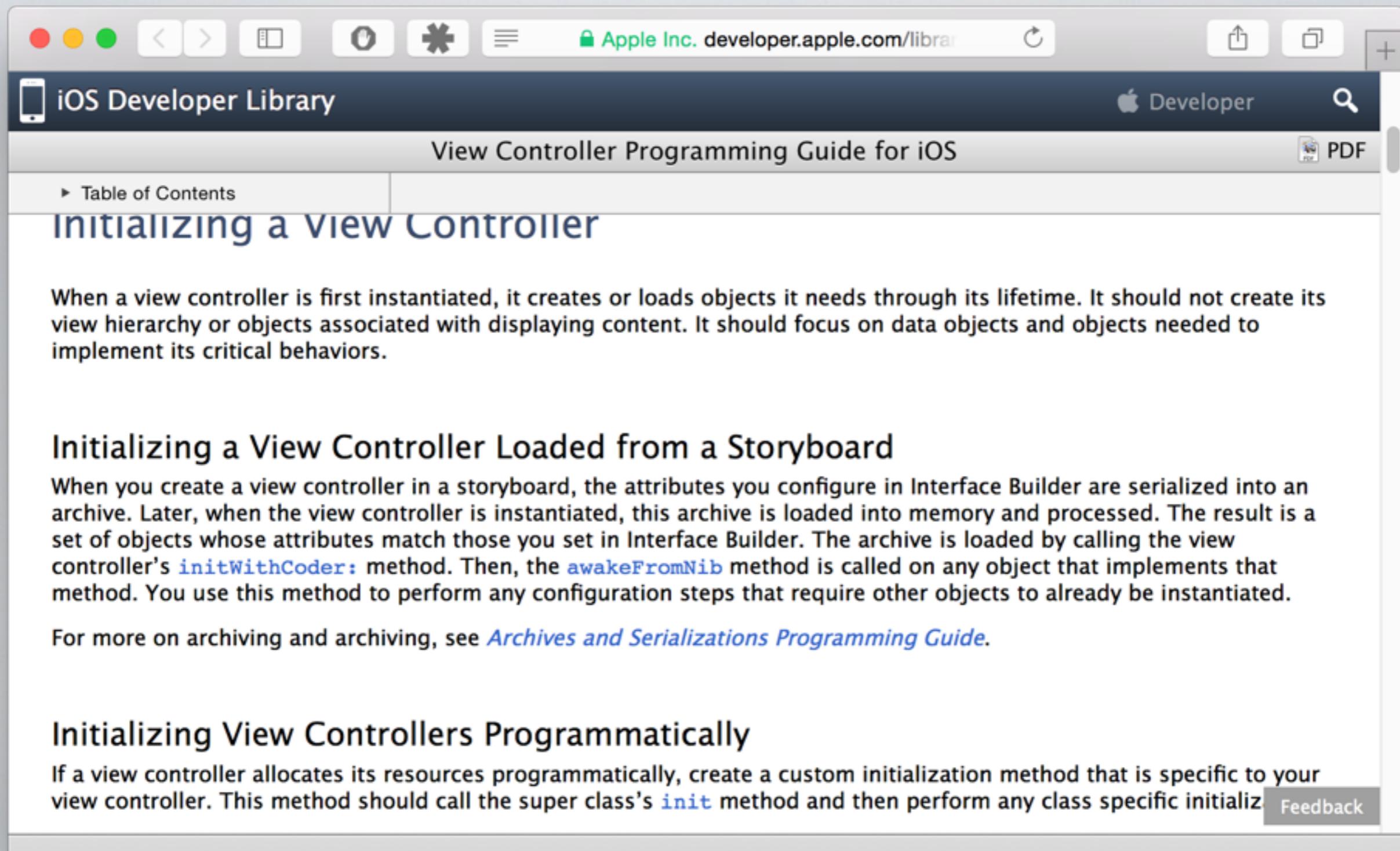


Here are the @IBOutlets

```
42                     </constraints>
43             <fontDescription key="fontDescription" type="system" pointSize="14"/>
44             <textInputTraits key="textInputTraits"/>
45         </textField>
46     </subviews>
47 </stackView>
48 </subviews>
49     <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
customColorSpace="calibratedWhite"/>
50         <constraints>
51             <constraint firstItem="G0j-cU-Mzs" firstAttribute="top" secondItem="y3c-jy-aDJ"
secondAttribute="bottom" constant="8" symbolic="YES" id="GYI-aP-E2R"/>
52             <constraint firstItem="wfY-db-euE" firstAttribute="top" secondItem="G0j-cU-Mzs"
secondAttribute="bottom" constant="363" id="cBT-yC-gZ3"/>
53             <constraint firstItem="G0j-cU-Mzs" firstAttribute="centerX" secondItem="8bC-Xf-
="centerX" id="xTF-5X-L69"/>
54         </constraints>
55     </view>
56 <connections>
57     <outlet property="button" destination="Hj4-gW-8Jg" id="PmB-oS-Fg3"/>
58     <outlet property="hello" destination="Boh-GC-vzh" id="xNy-cG-W7S"/>
59     <outlet property="textField" destination="20D-2W-ph5" id="VPH-NS-JJn"/>
60 </connections>
61 <viewController>
62     <placeholder placeholderIdentifier="IBFirstResponder" id="dkx-z0-nzr"
63     Responder"/>
64     <point key="canvasLocation" x="43" y="5"/>
65   </scene>
66 </scenes>
67 <resources>
68     <image name="World" width="128" height="128"/>
69 </resources>
70 </document>
```

NORMAL ➤ master ➤ GitHubDemo/Base.lproj/Main.storyboard unix < utf-8 < xml 84% ↵ 59:1
:nohl

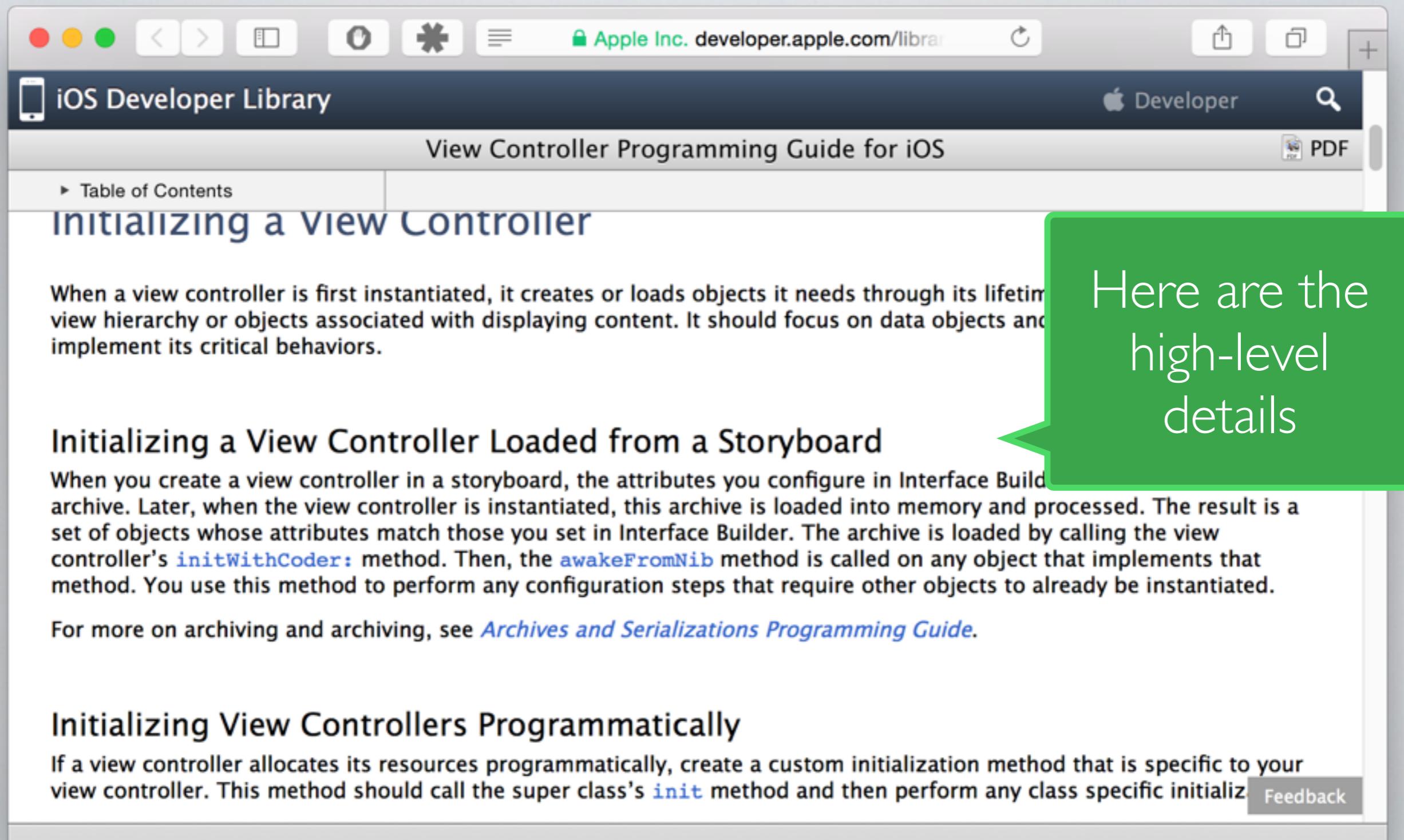
HOW DOES IT WORK?



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows "Apple Inc. developer.apple.com/library" with a lock icon.
- Header:** "iOS Developer Library" on the left, "Developer" with an Apple logo in the center, and a search icon on the right.
- Page Title:** "View Controller Programming Guide for iOS".
- Navigation:** A horizontal bar with icons for back, forward, search, and other navigation functions.
- Content Area:**
 - A link to "Table of Contents".
 - Section Header:** "Initializing a View Controller".
 - Text:** "When a view controller is first instantiated, it creates or loads objects it needs through its lifetime. It should not create its view hierarchy or objects associated with displaying content. It should focus on data objects and objects needed to implement its critical behaviors."
 - Section Header:** "Initializing a View Controller Loaded from a Storyboard".
 - Text:** "When you create a view controller in a storyboard, the attributes you configure in Interface Builder are serialized into an archive. Later, when the view controller is instantiated, this archive is loaded into memory and processed. The result is a set of objects whose attributes match those you set in Interface Builder. The archive is loaded by calling the view controller's `initWithCoder:` method. Then, the `awakeFromNib` method is called on any object that implements that method. You use this method to perform any configuration steps that require other objects to already be instantiated."
 - Text:** "For more on archiving and unarchiving, see [Archives and Serializations Programming Guide](#)".
 - Section Header:** "Initializing View Controllers Programmatically".
 - Text:** "If a view controller allocates its resources programmatically, create a custom initialization method that is specific to your view controller. This method should call the super class's `init` method and then perform any class specific initializ... [Feedback](#)"

HOW DOES IT WORK?



The screenshot shows a web browser window with the URL [Apple Inc. developer.apple.com/library/ios/documentation/UIKit/Conceptual/ViewController_Programming_Guide/](https://developer.apple.com/library/ios/documentation/UIKit/Conceptual/ViewController_Programming_Guide/). The page title is "View Controller Programming Guide for iOS". A green callout bubble points from the text "Here are the high-level details" to the section "Initializing a View Controller Loaded from a Storyboard".

Initializing a View Controller

When a view controller is first instantiated, it creates or loads objects it needs through its lifetime view hierarchy or objects associated with displaying content. It should focus on data objects and implement its critical behaviors.

Initializing a View Controller Loaded from a Storyboard

When you create a view controller in a storyboard, the attributes you configure in Interface Builder are saved in an archive. Later, when the view controller is instantiated, this archive is loaded into memory and processed. The result is a set of objects whose attributes match those you set in Interface Builder. The archive is loaded by calling the view controller's `initWithCoder:` method. Then, the `awakeFromNib` method is called on any object that implements that method. You use this method to perform any configuration steps that require other objects to already be instantiated.

For more on archiving and unarchiving, see [Archives and Serializations Programming Guide](#).

Initializing View Controllers Programmatically

If a view controller allocates its resources programmatically, create a custom initialization method that is specific to your view controller. This method should call the super class's `init` method and then perform any class specific initializ... [Feedback](#)

Here are the high-level details

HOW DOES IT WORK?

The screenshot shows a web browser window with the URL [Apple Inc. developer.apple.com/library/ios/documentation/UIKit/Conceptual/ViewController_Programming_Guide/](https://developer.apple.com/library/ios/documentation/UIKit/Conceptual/ViewController_Programming_Guide/). The page title is "View Controller Programming Guide for iOS". A green callout bubble points from the text "When a view controller is first instantiated, it creates or loads objects it needs through its lifetime view hierarchy or objects associated with displaying content. It should focus on data objects and implement its critical behaviors." to the text "Here are the high-level details". A red callout bubble points from the text "For more on archiving and unarchiving, see [Archives and Serializations Programming Guide](#)." to the text "Here are the low-level details".

iOS Developer Library Apple Developer PDF

View Controller Programming Guide for iOS

▶ Table of Contents

Initializing a View Controller

When a view controller is first instantiated, it creates or loads objects it needs through its lifetime view hierarchy or objects associated with displaying content. It should focus on data objects and implement its critical behaviors.

Initializing a View Controller Loaded from a Storyboard

When you create a view controller in a storyboard, the attributes you configure in Interface Builder are saved in an archive. Later, when the view controller is instantiated, this archive is loaded into memory and processed. The result is a set of objects whose attributes match those you set in Interface Builder. The archive is loaded by calling the view controller's `initWithCoder:` method. Then, the `awakeFromNib` method is called on any object that has a `awakeFromNib` method. You use this method to perform any configuration steps that require other objects to be available.

For more on archiving and unarchiving, see [Archives and Serializations Programming Guide](#).

Initializing View Controllers Programmatically

If a view controller allocates its resources programmatically, create a custom initialization method for the view controller. This method should call the super class's `init` method and then perform any class-specific initialization.

Here are the high-level details

Here are the low-level details

HOW DOES IT WORK?

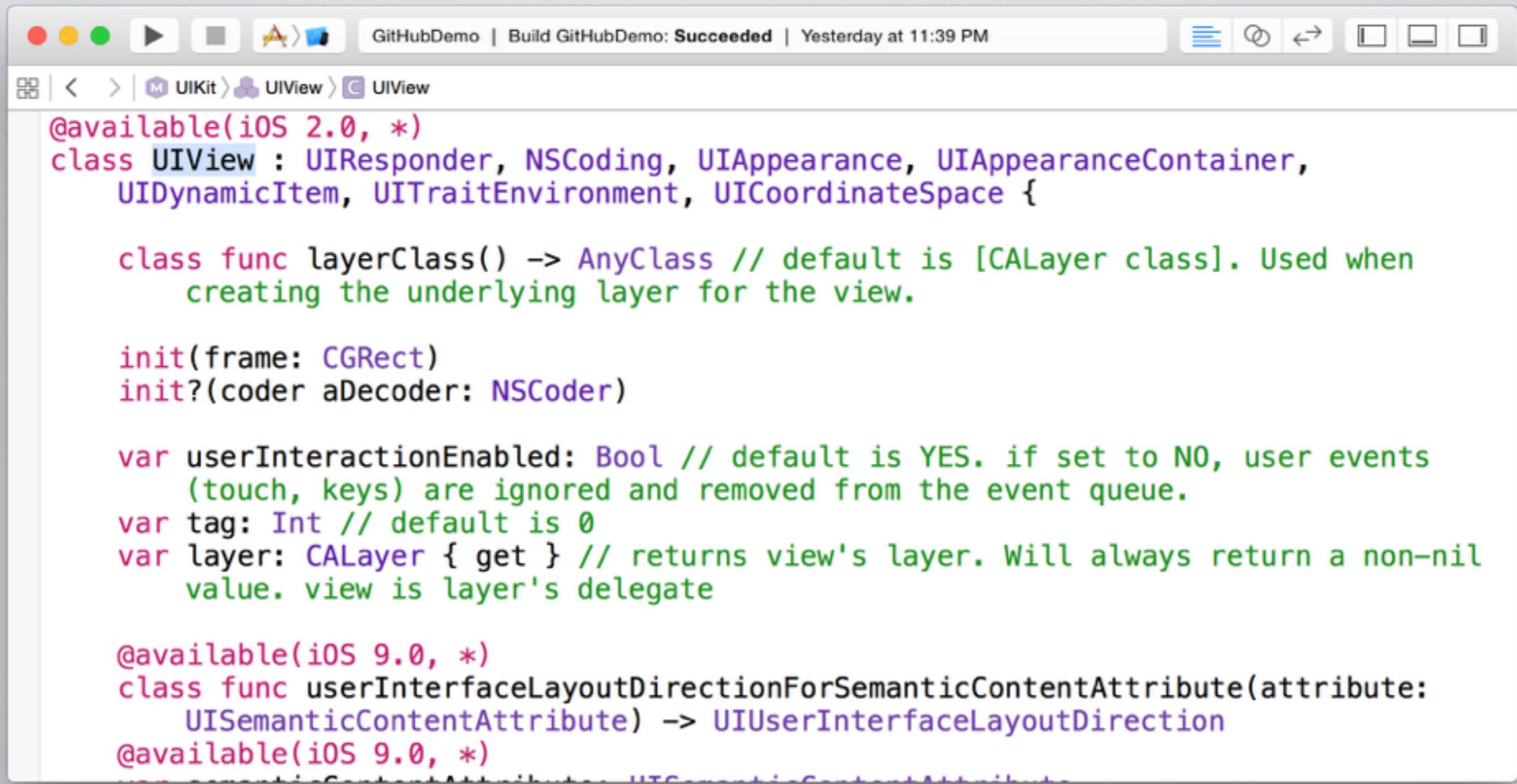
The NSCoding protocol declares the two methods that a class must implement so that instances of that class can be encoded and decoded. This capability provides the basis for archiving (where objects and other structures are stored on disk) and distribution (where objects are copied to different address spaces).

In keeping with object-oriented design principles, an object being encoded or decoded is responsible for encoding and decoding its instance variables. A coder instructs the object to do so by invoking `encodeWithCoder:` or `initWithCoder:`. `encodeWithCoder:` instructs the object to encode its instance variables to the coder provided; an object can receive this method any number of times. `initWithCoder:` instructs the object to initialize itself from data in the coder provided; as such, it replaces any other initialization method and is sent only once per object. Any object class that should be codable must adopt the NSCoding protocol and implement its methods.

It is important to consider the possible types of archiving that a coder supports. On OS X version 10.2 and later, keyed archiving is preferred. You may, however, need to support classic archiving. For details, see [Archives and Serializations Programming Guide](#).



HOW DOES IT WORK?



The screenshot shows the Xcode interface with the code for the `UIView` class. The window title is "GitHubDemo | Build GitHubDemo: Succeeded | Yesterday at 11:39 PM". The navigation bar shows the path: `UIKit > UIView > UIView`. The code is as follows:

```
@available(iOS 2.0, *)
class UIView : UIResponder, NSCoding, UIAppearance, UIAppearanceContainer,
UIDynamicItem, UITraitEnvironment, UICoordinateSpace {

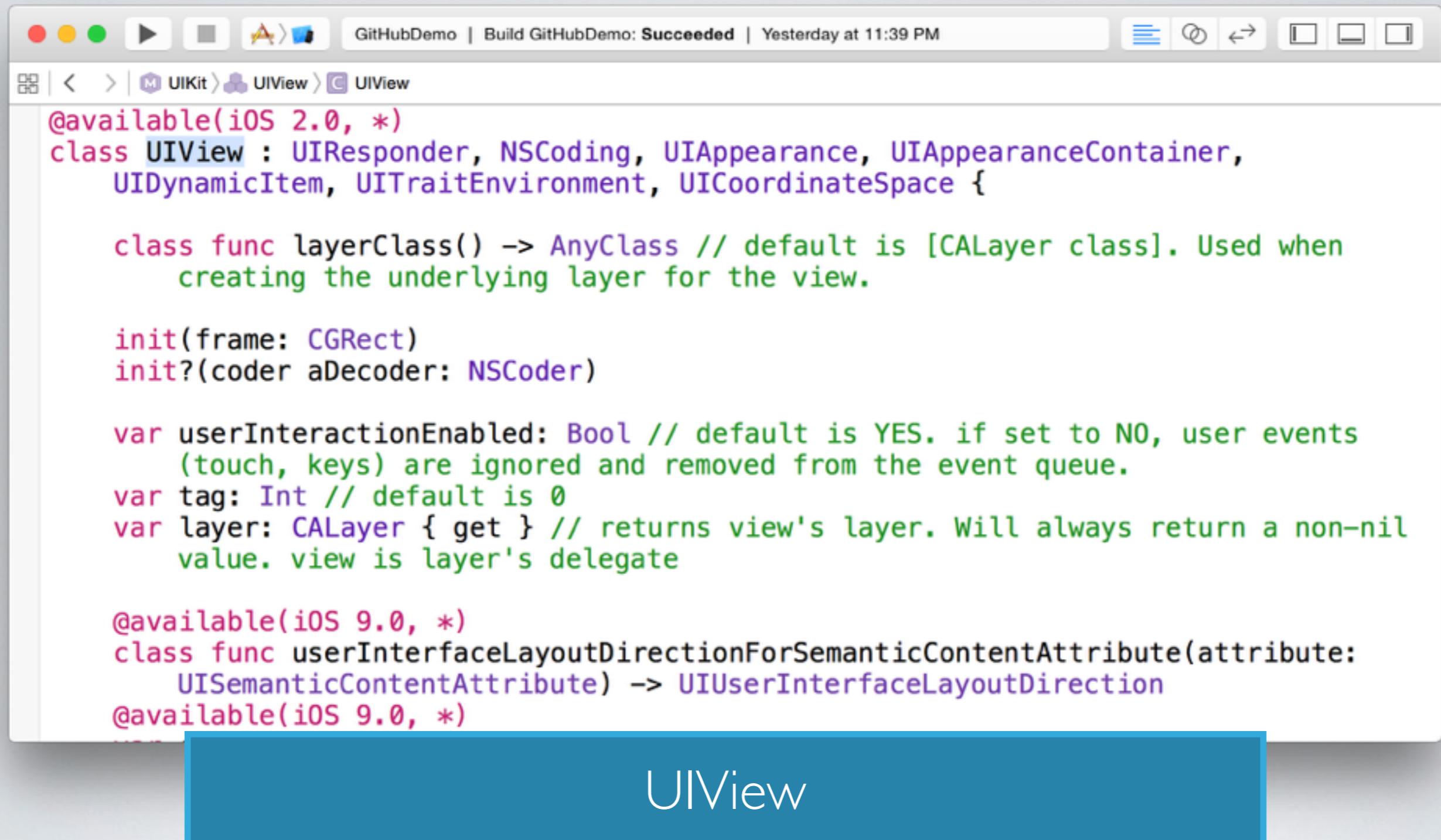
    class func layerClass() -> AnyClass // default is [CALayer class]. Used when
        creating the underlying layer for the view.

    init(frame: CGRect)
    init?(coder aDecoder: NSCoder)

    var userInteractionEnabled: Bool // default is YES. if set to NO, user events
        (touch, keys) are ignored and removed from the event queue.
    var tag: Int // default is 0
    var layer: CALayer { get } // returns view's layer. Will always return a non-nil
        value. view is layer's delegate

@available(iOS 9.0, *)
class func userInterfaceLayoutDirectionForSemanticContentAttribute(attribute:
    UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection
@available(iOS 9.0, *)
func semanticContentAttribute(_ attribute: UISemanticContentAttribute)
```

HOW DOES IT WORK?



The screenshot shows the Xcode interface with the code for the `UIView` class. The title bar indicates the project is "GitHubDemo" and the build status is "Succeeded" from "Yesterday at 11:39 PM". The navigation bar shows the path: `UIKit > UIView > UIView`. The code itself is as follows:

```
@available(iOS 2.0, *)
class UIView : UIResponder, NSCoding, UIAppearance, UIAppearanceContainer,
UIDynamicItem, UITraitEnvironment, UICoordinateSpace {

    class func layerClass() -> AnyClass // default is [CALayer class]. Used when
        creating the underlying layer for the view.

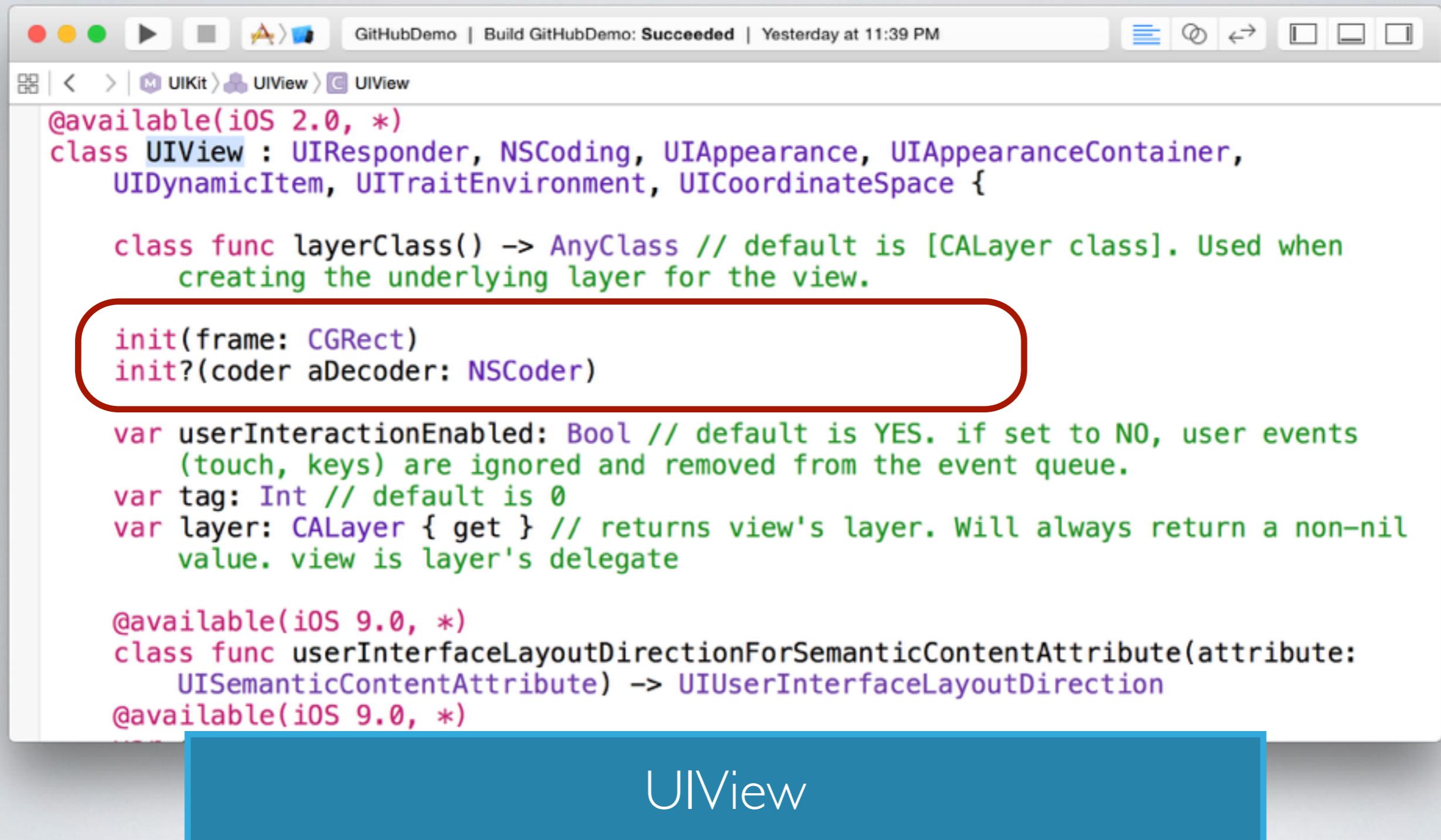
    init(frame: CGRect)
    init?(coder aDecoder: NSCoder)

    var userInteractionEnabled: Bool // default is YES. if set to NO, user events
        (touch, keys) are ignored and removed from the event queue.
    var tag: Int // default is 0
    var layer: CALayer { get } // returns view's layer. Will always return a non-nil
        value. view is layer's delegate

@available(iOS 9.0, *)
class func userInterfaceLayoutDirectionForSemanticContentAttribute(attribute:
    UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection
@available(iOS 9.0, *)
```

A large blue rectangular callout box is positioned at the bottom right of the code editor, pointing towards the word `UIView`.

HOW DOES IT WORK?



```
GitHubDemo | Build GitHubDemo: Succeeded | Yesterday at 11:39 PM
```

```
UIKit > UIView > UIView
```

```
@available(iOS 2.0, *)
class UIView : UIResponder, NSCoding, UIAppearance, UIAppearanceContainer,
UIDynamicItem, UITraitEnvironment, UICoordinateSpace {

    class func layerClass() -> AnyClass // default is [CALayer class]. Used when
        creating the underlying layer for the view.

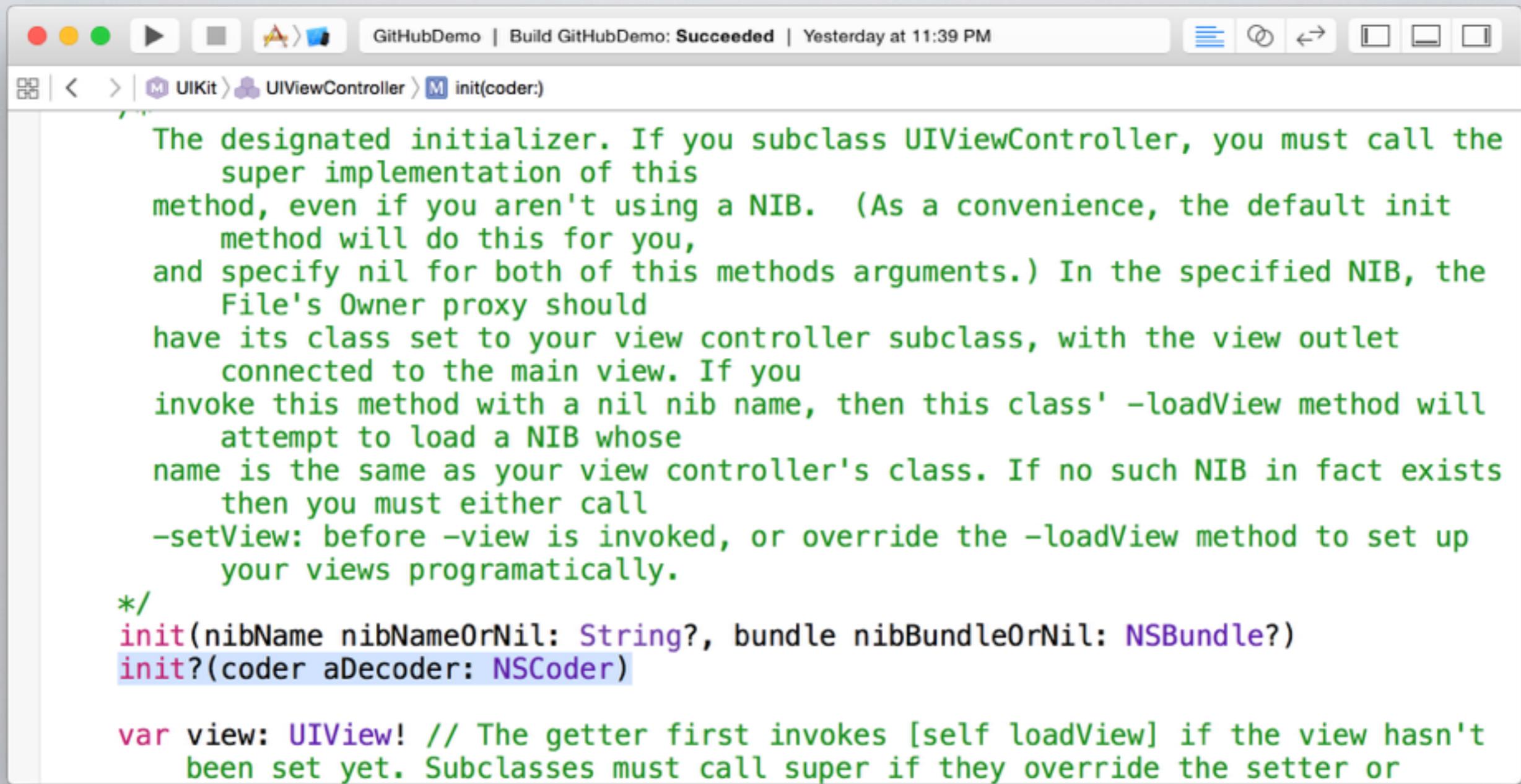
    init(frame: CGRect)
    init?(coder aDecoder: NSCoder)

    var userInteractionEnabled: Bool // default is YES. if set to NO, user events
        (touch, keys) are ignored and removed from the event queue.
    var tag: Int // default is 0
    var layer: CALayer { get } // returns view's layer. Will always return a non-nil
        value. view is layer's delegate

    @available(iOS 9.0, *)
    class func userInterfaceLayoutDirectionForSemanticContentAttribute(attribute:
        UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection
    @available(iOS 9.0, *)
```

UIView

HOW DOES IT WORK?

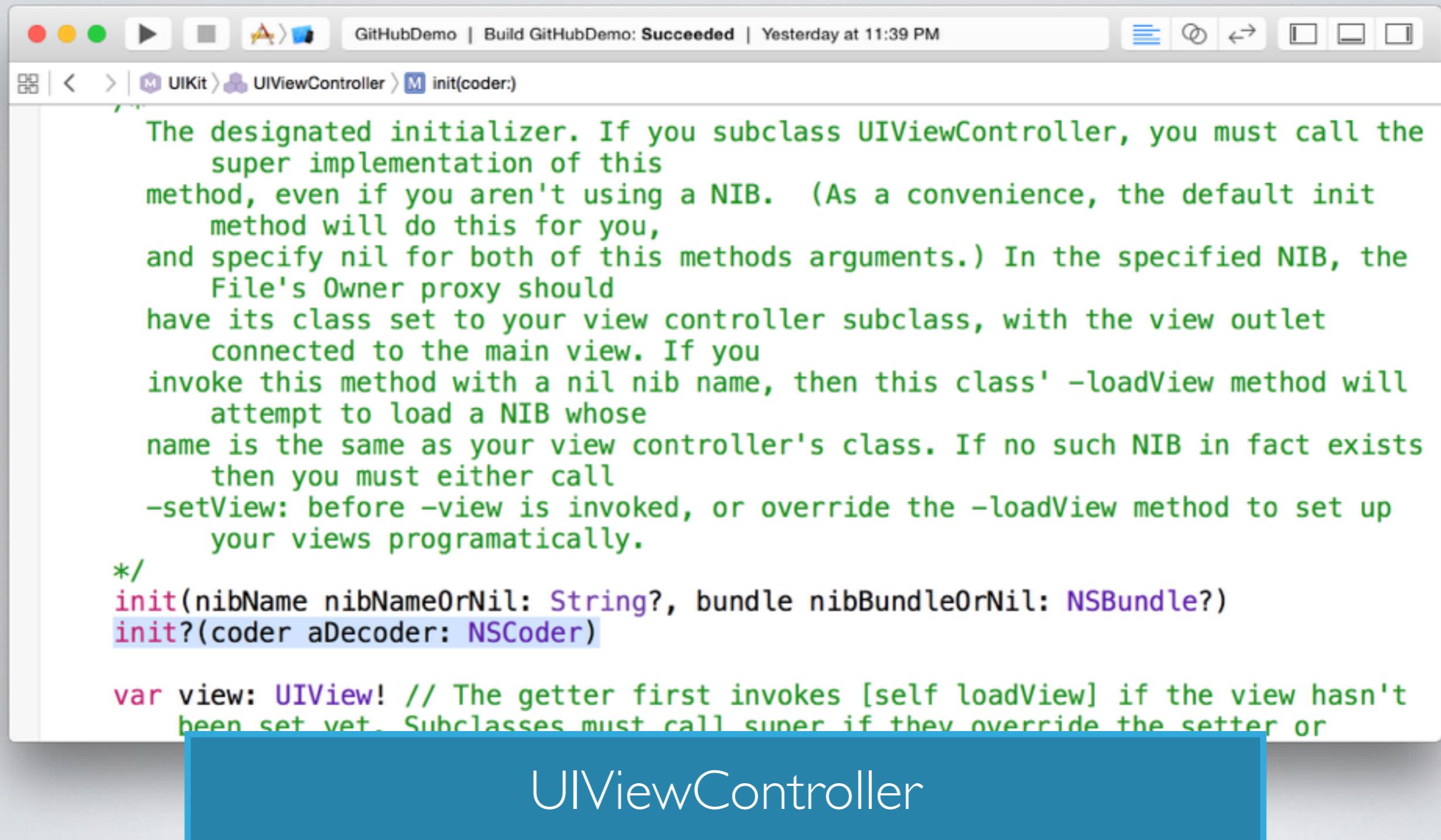


The screenshot shows the Xcode code editor with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Yesterday at 11:39 PM". The navigation bar shows "UIKit > UIViewController > init(coder:)". The code in the editor is:

```
    The designated initializer. If you subclass UIViewController, you must call the
    super implementation of this
method, even if you aren't using a NIB. (As a convenience, the default init
    method will do this for you,
and specify nil for both of this methods arguments.) In the specified NIB, the
    File's Owner proxy should
have its class set to your view controller subclass, with the view outlet
    connected to the main view. If you
invoke this method with a nil nib name, then this class' -loadView method will
    attempt to load a NIB whose
name is the same as your view controller's class. If no such NIB in fact exists
    then you must either call
-.setView: before -view is invoked, or override the -loadView method to set up
    your views programatically.
*/
init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: NSBundle?)
init?(coder aDecoder: NSCoder)

var view: UIView! // The getter first invokes [self loadView] if the view hasn't
    been set yet. Subclasses must call super if they override the setter or
```

HOW DOES IT WORK?



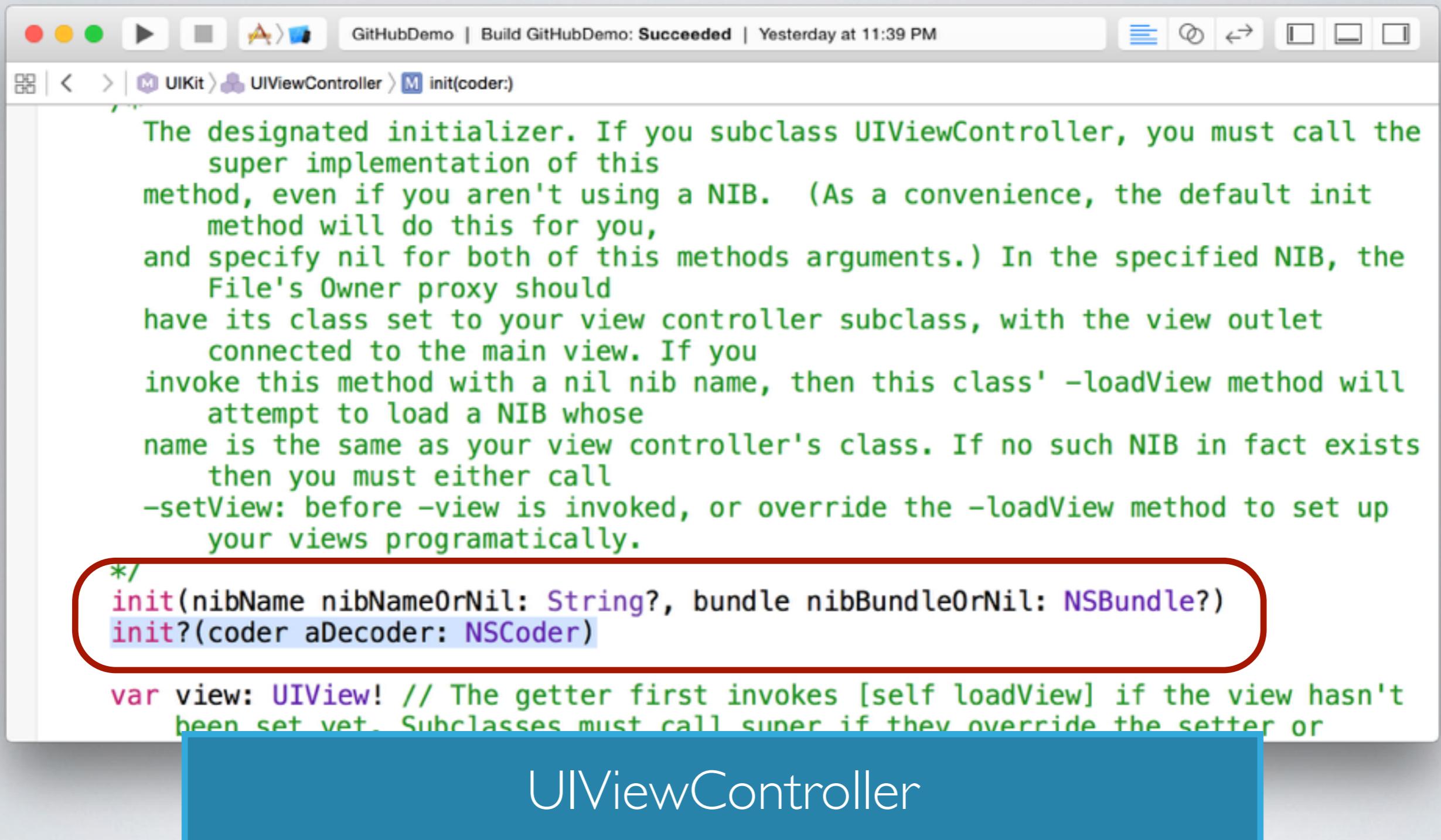
The screenshot shows the Xcode interface with the title bar "GitHubDemo | Build GitHubDemo: Succeeded | Yesterday at 11:39 PM". The navigation bar shows the path "UIKit > UIViewController > init(coder:)". The code editor displays the implementation of the `init(coder:)` method:

```
    The designated initializer. If you subclass UIViewController, you must call the
    super implementation of this
method, even if you aren't using a NIB. (As a convenience, the default init
    method will do this for you,
and specify nil for both of this methods arguments.) In the specified NIB, the
    File's Owner proxy should
have its class set to your view controller subclass, with the view outlet
    connected to the main view. If you
invoke this method with a nil nib name, then this class' -loadView method will
    attempt to load a NIB whose
name is the same as your view controller's class. If no such NIB in fact exists
    then you must either call
-.setView: before -view is invoked, or override the -loadView method to set up
    your views programatically.
*/
init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: NSBundle?)
init?(coder aDecoder: NSCoder)

var view: UIView! // The getter first invokes [self loadView] if the view hasn't
    been set yet. Subclasses must call super if they override the setter or
```

A blue rectangular callout box highlights the word "UIViewController" in the last line of the code.

HOW DOES IT WORK?



The designated initializer. If you subclass `UIViewController`, you must call the super implementation of this method, even if you aren't using a NIB. (As a convenience, the default `init` method will do this for you, and specify `nil` for both of this methods arguments.) In the specified NIB, the File's Owner proxy should have its class set to your view controller subclass, with the `view` outlet connected to the main view. If you invoke this method with a nil nib name, then this class' `-loadView` method will attempt to load a NIB whose name is the same as your view controller's class. If no such NIB in fact exists then you must either call `-setView:` before `-view` is invoked, or override the `-loadView` method to set up your views programmatically.

```
*  
init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: NSBundle?)  
init?(coder aDecoder: NSCoder)  
  
var view: UIView! // The getter first invokes [self loadView] if the view hasn't  
// been set yet. Subclasses must call super if they override the setter or
```

UIViewController

WHY DO YOU NEED THIS?

- Because you might need to resolve *merge conflicts* when many people edit the **same** storyboard at the **same** time. It's painful but necessary.
- Read <http://www.toptal.com/ios/ios-user-interfaces-storyboards-vs-nibs-vs-custom-code>

NEXT TIME

- Design of code
- Presenting information in TableView