Creating a Restoration Class

Creating the Example Application

**Create an application based on the Tabbed Application template. Enter StateApp as the product name and select the Use Storyboards and Use Automatic Reference Counting**

**Trying the Application without State Preservation**

**The easiest way to test state preservation to click on the Stop button in the Xcode toolbar. Once the application has stopped running, launch it a second time by clicking on the Run button. When the application launches, the First tab will be selected instead of the Second tab.** **No application state has been preserved between application launches. This is because the application has not “opted-in” to state preservation and restoration.**

**Part I: Opting-in to State Preservation**

**1.Select the StateAppAppDelegate.m)and modify it to add the two methods required to opt in to both the saving and restoration of application state:**

-(BOOL)application:(UIApplication \*)application shouldRestoreApplicationState:(NSCoder \*)coder

{

return YES;

}

-(BOOL)application:(UIApplication \*)application shouldSaveApplicationState:(NSCoder \*)coder

{

return YES;

}

**Part II: Setting Restoration Identifiers**

**1. Select the Tab Bar Controller in the storyboard canvas. Display the Identity Inspector and enter tabController1 into the Restoration ID field.**

**2. Now assign restoration IDs of viewController1 and viewController2 to the first and second view controllers respectively.**

**3. Now run and test**

**Part III: Encoding and Decoding View Controller State**

1. **Select the second view controller in the storyboard and select and delete the labels currently present in the layout.**
2. **Drag and Drop a Text View object onto the layout. Using the resize handles, reduce the height of the Text View so that it is approximately a quarter of the height of the containing view.**
3. **Select the Text View object, display the Attribute Inspector and delete the sample Latin text from the Text property**.
4. **Create a new outlet connection named myTextView.**

**5. Run it and notice the text view has been lost**

**In order to preserve the text entered by the user, it will be necessary to implement the encodeRestorableStateWithCoder: and decodeRestorableStateWithCoder: methods in the parent view controller of the Text View (in this case StateAppSecondViewController).**

**6.** **Select the StateAppSecondViewController.m file and add the encoding methods as follows**:

-(void)encodeRestorableStateWithCoder:(NSCoder \*)coder

{

[coder encodeObject:\_myTextView.text forKey:@"UnsavedText"];

[super encodeRestorableStateWithCoder:coder];

}

-(void)decodeRestorableStateWithCoder:(NSCoder \*)coder

{

\_myTextView.text = [coder decodeObjectForKey:@"UnsavedText"];

[super decodeRestorableStateWithCoder:coder];

}

**Part IV: A Restoration Class that Restores a controller not created by the storyboad**

1. **Select the Second view controller and embed it in a navigation controller**
2. **Select the navigation controller in the storyboard canvas, display the Identity Inspector and set the restoration ID to navController1**
3. **Select the Second view controller and add an Action to the Button in the assistant editor. Call that action displayVC3.**
4. **Your StateAppSecondViewController.h interface file should read as follows:**

#import <UIKit/UIKit.h>

@interface StateAppSecondViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextView \*myTextView;

- (IBAction)displayVC3:(id)sender;

@end

**5. Adding the Third View Controller. Add an objective-C class (Cocoa Touch) . Call it StateAppThirdViewController and make it a subclass of UIViewController. Make sure “With xib” is selected.**

**6. Select the newly created StateAppThirdViewController.xib file to load it into Interface Builder and add a label that indicates this is the view for the third view controller**

**7. Select the StateAppSecondViewController.h file and modify it to import the interface file for the StateAppThirdViewController class and to declare a property for the corresponding view controller object when it is created:**

#import <UIKit/UIKit.h>

#import "StateAppThirdViewController.h"

@interface StateAppSecondViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextView \*myTextView;

@property (strong, nonatomic) UIViewController \*thirdViewController;

**//don’t forget to create a property above!**

- (IBAction)displayVC3:(id)sender;

@end

**8. Select the StateAppSecondViewController.m file and modify the viewDidLoad: method to create a new instance of the third view controller class and the displayVC3: action method to push the view controller onto the navigation stack so that it is appears to the user when the button is touched:**

- (void)viewDidLoad

{

[super viewDidLoad];

// Do any additional setup after loading the view, typically from a nib.

\_thirdViewController = [[StateAppThirdViewController alloc]

initWithNibName:@"StateAppThirdViewController"

bundle:nil];

}

- (IBAction)displayVC3:(id)sender {

[self.navigationController

pushViewController:\_thirdViewController animated:YES];

}

**9. Finally, the code in the StateAppThirdViewController class needs to be modified so that instances of the class are assigned a restoration ID. Select the StateAppThirdViewController.m file and add a line to the viewDidLoad: method:**

- (void)viewDidLoad

{

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

self.restorationIdentifier = @"thirdViewController";

}

**Build and run the application, navigate to the third view controller in the user interface and then perform the usual background/kill/run cycle. Note that the application returned to the second view controller screen and not the third view controller. Because the third view controller was created in code, UIKit is unable to find a way to recreate it when the application state is restored. This is where it becomes necessary to implement and register a restoration class for the StateAppThirdViewController class.**

**10. Select the StateAppThirdViewController.h file, therefore, and modify it to declare that the class now implements the view controller restoration protocol:**

#import <UIKit/UIKit.h>

@interface StateAppThirdViewController : UIViewController

<UIViewControllerRestoration>

@end

**11 . Next, implement the viewControllerWithRestorationIdentifierPath: class method in the StateAppThirdViewController.m file:**

+(UIViewController \*)viewControllerWithRestorationIdentifierPath:(NSArray \*)identifierComponents coder:(NSCoder \*)coder

{

UIViewController \* myViewController =

[[StateAppThirdViewController alloc]

initWithNibName:@"StateAppThirdViewController"

bundle:nil];

return myViewController;

}

**12. Add a single line to the viewDidLoad: method, referencing self since the class is acting as its own restoration class**:

- (void)viewDidLoad

{

[super viewDidLoad];

// Do any additional setup after loading the view from its nib.

self.restorationIdentifier = @"thirdViewController";

self.restorationClass = [self class];

}

**Compile and run the application, navigate to the third view controller and background, stop and rerun the application. On the second run, the application should now be restored to the view of the third view controller, a clear sign that the restoration class worked.**