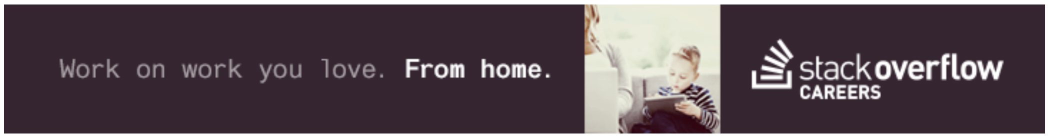
+10 11 3 help

What are Unwind segues for and how do you use them?



iOS 6 and Xcode 4.5 has a new feature referred to as "Unwind Segue":

> Unwind segues can allow transitioning to existing instances of scenes in a storyboard

In addition to this brief entry in Xcode 4.5's release notes, UIViewController now seem to have a couple of new methods:

```
- (BOOL)canPerformUnwindSegueAction:(SEL)action fromViewController:
(UIViewController *)fromViewController withSender:(id)sender
- (UIViewController *)viewControllerForUnwindSegueAction:(SEL)action
fromViewController:(UIViewController *)fromViewController withSender:(id)sender
- (UIStoryboardSegue *)segueForUnwindingToViewController:(UIViewController
*)toViewController fromViewController:(UIViewController *)fromViewController
identifier:(NSString *)identifier
```

How Unwind segues work and what they can be used for?

ios ios6 uistoryboard

edited Apr 7 '14 at 20:24
Jesse Rusak
41.4k 10 75 88

asked Sep 24 '12 at 8:57
Imre Kelényi
13.8k 5 23 38

protected by Midhun MP Dec 17 '14 at 0:12

This question is protected to prevent "thanks!", "me too!", or spam answers by new users. To answer it, you must have earned at least 10 reputation on this site.

5 Answers

In a Nutshell

An **unwind segue** (sometimes called **exit segue**) can be used to navigate back through push, modal or popover segues (as if you popped the navigation item from the navigation bar, closed the popover or dismissed the modally presented view controller). On top of that you can actually unwind through not only one but a series of push/modal/popover segues, e.g. "go back" multiple steps in your navigation hierarchy with a single unwind action.

When you perform an unwind segue, you need to specify an action, which is an action method of the view controller you want to unwind to.

Objective-C:

```
- (IBAction)unwindToThisViewController:(UIStoryboardSegue *)unwindSegue
{
}
```

Swift:

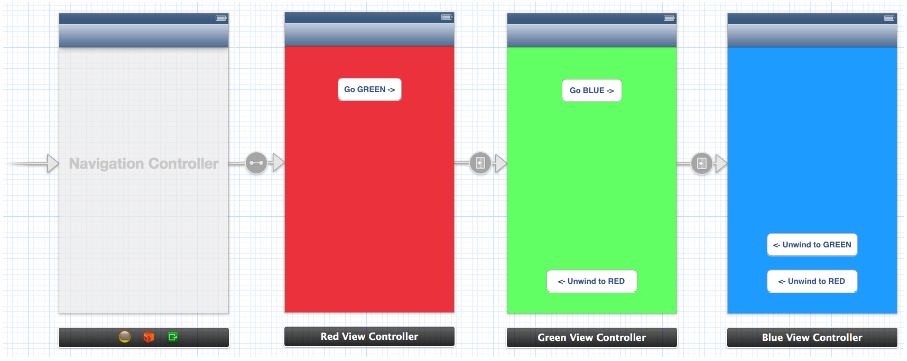
```
@IBAction func unwindToThisViewController(segue: UIStoryboardSegue) {
}
```

The name of this action method is used when you create the unwind segue in the storyboard. Furthermore, this method is called just before the unwind segue is performed. You can get the source view controller from the passed `UIStoryboardSegue` parameter to interact with the view controller that initiated the segue (e.g. to get the property values of a modal view controller). In this respect, the method has a similar function as the `prepareForSegue:` method of `UIViewController`.

iOS 8 update: Unwind segues also work with iOS 8's adaptive segues, such as *Show* and *Show Detail*.

An Example

Let us have a storyboard with a navigation controller and three child view controllers:



From Green View Controller you can unwind (navigate back) to Red View Controller. From Blue you can unwind to Green or to Red via Green. To enable unwinding you must add the special action methods to Red and Green, e.g. here is the action method in Red:

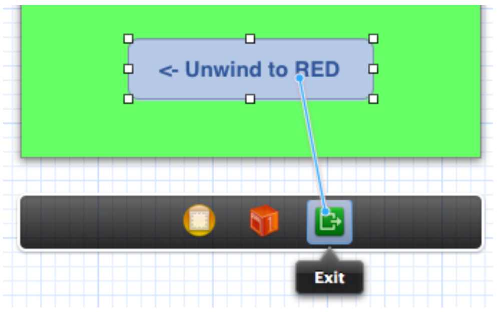
Objective-C:

```
@implementation RedViewController

- (IBAction)unwindToRed:(UIStoryboardSegue *)unwindSegue
{
}

@end
```

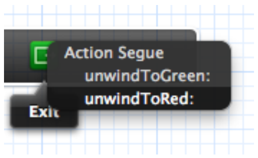
Swift:

```
@IBAction func unwindToRed(segue: UIStoryboardSegue) {
}
```

After the action method has been added, you can define the unwind segue in the storyboard by control-dragging to the Exit icon. Here we want to unwind to Red from Green when the button is pressed:



You must select the action which is defined in the view controller you want to unwind to:



You can also unwind to Red from Blue (which is "two steps away" in the navigation stack). The key is selecting the correct unwind action.

Before the the unwind segue is performed, the action method is called. In the example I defined an unwind segue to Red from both Green and Blue. We can access the source of the unwind in the action method via the UIStoryboardSegue parameter:

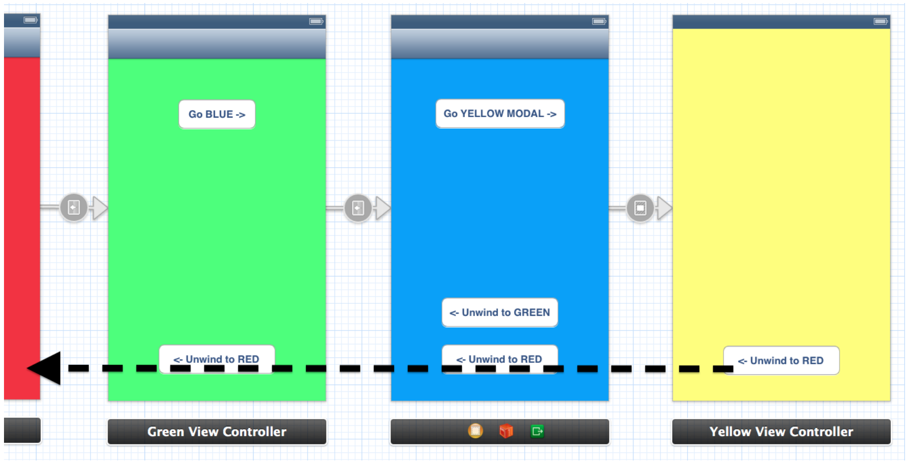
Objective-C:

```
- (IBAction)unwindToRed:(UIStoryboardSegue *)unwindSegue
{
    UIViewController* sourceViewController = unwindSegue.sourceViewController;

    if ([sourceViewController isKindOfClass:[BlueViewController class]])
    {
        NSLog(@"Coming from BLUE!");
    }
    else if ([sourceViewController isKindOfClass:[GreenViewController class]])
    {
        NSLog(@"Coming from GREEN!");
    }
}
```

Swift:

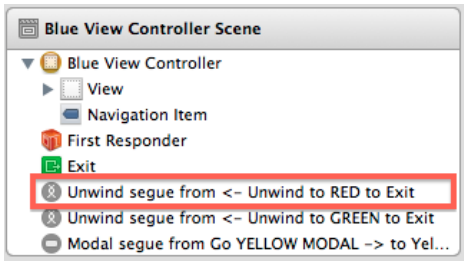
```
@IBAction func unwindToRed(unwindSegue: UIStoryboardSegue) {
    if let blueViewController = unwindSegue.sourceViewController as?
BlueViewController {
        println("Coming from BLUE")
    }
    else if let redViewController = unwindSegue.sourceViewController as?
RedViewController {
        println("Coming from RED")
    }
}
```

Unwinding also works through a combination of push/modal segues. E.g. if I added another Yellow view controller with a modal segue, we could unwind from Yellow all the way back to Red in a single step:



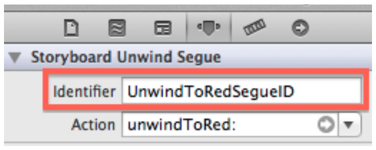
Unwinding from Code

When you define an unwind segue by control-dragging something to the Exit symbol of a view controller, a new segue appears in the Document Outline:



Blue View Controller Scene
- Blue View Controller
  - View
  - Navigation Item
- First Responder
- Exit
- Unwind segue from <- Unwind to RED to Exit
- Unwind segue from <- Unwind to GREEN to Exit
- Modal segue from Go YELLOW MODAL -> to Yel...

Selecting the segue and going to the Attributes Inspector reveals the "Identifier" property. Use this to give a unique identifier to your segue:



Identifier UnwindToRedSegueID

Action unwindToRed:

After this, the unwind segue can be performed from code just like any other segue:

Objective-C:

```
[self performSegueWithIdentifier:@"UnwindToRedSegueID" sender:self];
```

Swift:

```
performSegueWithIdentifier("UnwindToRedSegueID", sender: self)
```

edited Nov 21 '14 at 14:09 answered Apr 5 '13 at 16:49

Imre Kelényi
13.8k 5 23 38

8 +1 great answer. They sound really nice, but can't methods like `dismissViewControllerAnimated:completion:` or `popViewControllerAnimated:` achieve the same thing? – Sam Jul 2 '13 at 16:52

23 Sure they can. However, if you are using storyboards, unwind segues can often achieve the same thing with much less code. Actually, now you can dismiss a modally presented view controller without writing any code. Of course, there are still many cases when closing controllers from code is the right thing to do. – Imre Kelényi Jul 5 '13 at 9:53

5 Make sure to add your action method to your header file or else Storyboard will not know about. – Kyle C Aug 28 '13 at 18:42

13 Another advantage over `dismissViewControllerAnimated:completion:` or `popViewControllerAnimated:` is that the method you added to the view controller you are unwinding to is called and so you have an easy way to know the presented view controller is finished without having to make the presenting view controller a delegate of the presented view controller. – honus Sep 24 '13 at 6:19

7 Could I suggest a slight edit? It wasn't "obviously" clear that you put - (IBAction)unwindTRed:(UIStoryboardSegue *)unwindSegue in the RedViewController.m, and in turn this is universally available in "any" of the green exit buttons for any story board. Fantastic answer and now I will use this for other issues. Thanks! – John Ballinger Jan 22 '14 at 21:36

As far as how to use unwind segues in StoryBoard...

Step 1)

The bare minimum you need is to subclass the view controller for your destination view (aka, a view that has popped up previously in navigation and you want to unwind to it) and add a method like this to to the .h file (the method name can be anything you want, but it should be unique because all unwind segues in your entire app are listed together):

Objective-C

```
- (IBAction)unwindToViewControllerNameHere:(UIStoryboardSegue *)segue {
    //nothing goes here
}
```

Swift

```
@IBAction func unwindToViewControllerNameHere(segue: UIStoryboardSegue) {
    //nothing goes here
}
```

Step 2)

Now, in your source view (aka, the view that you want to unwind from) you simply drag a segue from your button or whatever down to the little orange "EXIT" icon at the top right of your source view. There should now be an option to connect to "- unwindToViewControllerNameHere"

That's it, your segue will unwind when your button is tapped.

edited Jan 21 at 6:35
William T.
4,656 3 19 24

answered Oct 11 '12 at 16:07
Travis M.
3,926 1 20 40

4 I have found that with Xcode 4.5 and earlier it was necessary to declare the IBAction in the header. I don't know if this is still true. – Steven Fisher Jun 21 '13 at 19:53

Is there a way to do Step 2 without storyboard, i.e. programmatically? My storyboard (interface builder) is messed up and it doesn't show the unwind segues (xcode bug). – Van Du Tran Jan 27 '15 at 21:55

Unwind segues are used to "go back" to some view controller from which, through a number of segues, you got to the "current" view controller.

Imagine you have something a `MyNavController` with `A` as its root view controller. Now you use a push segue to `B`. Now the navigation controller has A and B in its `viewControllers` array, and B is visible. Now you present `C` modally.

With unwind segues, you could now unwind "back" from `C` to `B` (i.e. dismissing the modally presented view controller), basically "undoing" the modal segue. You could even unwind all the way back to the root view controller `A`, undoing both the modal segue and the push segue.

Unwind segues make it easy to backtrack. For example, before iOS 6, the best practice for dismissing presented view controllers was to set the presenting view controller as the presented view controller's delegate, then call your custom delegate method, which then dismisses the presentedViewController. Sound cumbersome and complicated? It was. That's why unwind segues are nice.

edited Feb 17 '13 at 0:51
dimadima
6,149 4 38 57

answered Oct 2 '12 at 12:43
Yang Meyer
2,625 2 25 45

4 You can call `dismissViewController:animated` from the presented controller. You don't have to delegate that. Of course if you need to pass data back, then you need delegation or some other method. – mxcl Sep 26 '13 at 12:01

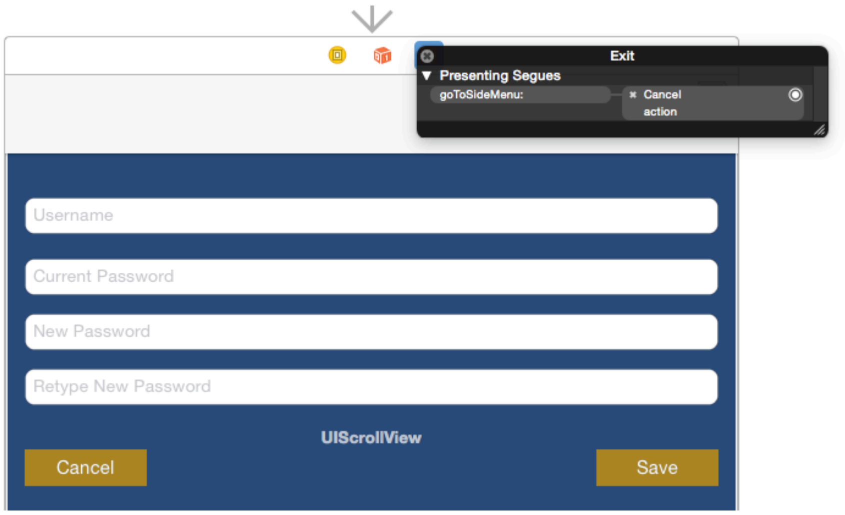
2 While you can call `dismissViewController:animated:` from the presented controller, "best practice" was indeed to call a delegate method on the presenting controller to do that for you, as Yang mentioned. – Chris Nolet Jul 9 '14 at 3:05

Swift iOS:

Step 1: define this method into your MASTER controller view. in which you want to go back:

```
//pragma mark - Unwind Seques
@IBAction func goToSideMenu(segue: UIStoryboardSegue) {

    println("Called goToSideMenu: unwind action")

}
```

Step 2: (StoryBoard) Right click on you SLAVE/CHILD EXIT button and Select "goToSideMenu" As action to Connect you Button on which you will click to return back to you MASTER controller view:



step 3: Build and Run ...

answered Nov 21 '14 at 13:39
Vinod Joshi
2,664 18 26

Something that I didn't see mentioned in the other answers here is how you deal with unwinding when you don't know where the initial segue originated, which to me is an even more important use case. For example, say you have a help view controller (**H**) that you display modally from two different view controllers (**A** and **B**):

A → **H**
B → **H**

How do you set up the unwind segue so that you go back to the correct view controller? The answer is that you declare an unwind action in **A** and **B** *with the same name*, e.g.:

```
// put in AViewController.swift and BViewController.swift
@IBAction func unwindFromHelp(sender: UIStoryboardSegue) {
    // empty
}
```

This way, the unwind will find whichever view controller (**A** or **B**) initiated the segue and go back to it.

In other words, think of the unwind action as describing where the segue is coming *from*, rather than where it is going to.

answered Dec 15 '15 at 19:33
shawkinaw
1,790 13 19