Is parentViewController always a Navigation controller?

Asked 16 years, 1 month ago Modified 14 years ago Viewed 14k times

7

I was kind of scratching my head at this a week ago, and now with a little bit more Cocoa experience under my belt I feel like I have an inkling as to what might be going on.

I'm making an application that is driven by a UINavigationController. In the AppDelegate, I create an instance of this class, using "page 1" as the Root View Controller.

```
UINavigationController *aNavigationController = [[UINavigationController alloc]
    initWithRootViewController:page1ViewController];
```

Now here's where I'm having the problem. From "page 1" I'd like to use a modal view controller that slides over the interface and then disappears once the user has made an edit. I do that using code like this, inside of Page1ViewController:

```
[self presentModalViewController:myModalViewController animated:YES];
```

When the Modal View Controller is gone, I want a value on "Page 1" to change based on what the user entered in the Modal View Controller. So, I wrote some code like this, which resides in the Modal View Controller:

```
[self.parentViewController dismissModalViewControllerAnimated:YES];
[self.parentViewController doSomethingPleaseWithSomeData:someData];
```

The update to page 1 wasn't happening, and it took me a long time to realize that the "doSomethingPleaseWithSomeData" message was not being sent to Page1ViewController, but the Navigation Controller.

Is this always to be expected when using Navigation Controllers? Did I perhaps configure something improperly? Is there an easy way to get at the View Controller that I want (in this case, Page1ViewController).

iphone cocoa-touch

Share edited Oct 28, 2008 at 23:29 asked Oct 28, 2008 at 19:48

Improve this question

Follow

Chris Hanson
55k ● 8 ● 74 ● 104

bpapa
21.5k ● 25 ● 100 ● 147

4 Answers

Sorted by: Highest score (default)

14

I would recommend using the delegation pattern to solve your problem. Create a property

```
@property (nonatomic, assign) id <MyModalViewDelegate> delegate;
```

And a corresponding protocol

```
@protocol MyModalViewDelegate
@optional
    - (void)myModalViewControllerDidFinish:(MyModalViewController
*)aModalViewController;
@end
```

When the user finishes with your view (e.g. taps the save button), send this message:

```
if ([self.delegate
respondsToSelector:@selector(myModalViewControllerDidFinish:)])
    [self.delegate myModalViewControllerDidFinish:self];
```

Now, set the delegate to the view controller that should manage the whole thing, and it will be notified when the view controller is finished. Note that you'll need your view controller to dismiss the modal view controller. But, logically, that makes sense, since it was the object that presented the modal view controller in the first place.

This is how Apple solves this problem in, for example, the UIImagePickerController and UIPersonPickerController.

Share Improve this answer Follow

answered Nov 11, 2008 at 15:51

Alex
26.9k ● 3 ● 58 ● 74

4

There are a couple of ways you can handle this. The simplest is probably just to add a UIViewController property into myModalViewController and set it to page1Controller before you present it:

```
myModalViewController.logicalParent = self; //page1Controller
[self presentModalViewController:myModalViewController animated:YES];
```

Just make sure you add the appropriate instance variable @property, and @synthesize for logicalParent to myModalViewController, then you will have a way to communicate data back to the ViewController that triggered the modal dialog. This is also for passing data back and forth between different levels of navigation before you push and pop them on the stack.

The one important thing to worry about when doing this is that it is easy to get retain loops if you are not careful. Depending on exactly how you structure this you might need to use assign properties.

Share Improve this answer Follow

answered Oct 28, 2008 at 20:08

Louis Gerbarg
43.4k ● 8 ● 83 ● 91

Accomplishes exactly what I want, thanks! I'm new to Cocoa, but not to MVC. The thing is though, being a Java web developer I almost always forget that the controllers in "C" CAN interact with each other! – bpapa Oct 28, 2008 at 20:39

A good answer, but why is it this case? It's clearly *not* that case that the parentViewController of the modal view is the navigation controller in any logical sense (page1Controller is the controller asking for the modal display) , so why does Cocoa Touch set it up this way? – Adam Wright Apr 15, 2010 at 13:01

1

I just ran into this same problem. It definitely seems that if you put a UIViewController embedded in a NavigationController, then when, from that UIViewController you present another UIViewController modally, the presentee thinks that the presenter is the NavigationController. In other words, parentViewController is incorrect.

I bet this is a bug: either that, or the documentation seems incomplete. I will inquire.

Share Improve this answer Follow

answered Sep 24, 2010 at 15:39

Michael
11 ● 1

1

Just ran into the same problem. I believe this is a bug. My scenario is the following: A navigation hierarchy with A, B and C view controllers in this order. On C there's a button that would open a modal view controller called D. Once D is presented the navigation controller drops C from its hierarchy which is a terrible behavior. Once D gets dismissed, the navigation controller *instantiates* a new C type view controller and pushes it into its hierarchy to recover the original one. Terrible. My solution is hacking the navigation hierarchy this way (a very bad solution but works well. with a 2 dimension array you could implement stacking modals):

```
- (void)presentModalViewController:(UIViewController *)c {
    [self.navigationHierarchy removeAllObjects];
    [self.navigationHierarchy addObjectsFromArray:[navigation
viewControllers]];
    [navigation setViewControllers:[NSArray array] animated:YES];
    [navigation presentModalViewController:c animated:YES];
}

- (void)dismissModalViewController {
    [navigation dismissModalViewControllerAnimated:YES];
    [navigation setViewControllers:[NSArray
arrayWithArray:self.navigationHierarchy] animated:YES];
}
```

These two methods are defined where I maintain the main navigation hiererchy: the app delegate. navigation and navigationhierarchy are defined this way:

```
NSMutableArray *navigationHierarchy;
UINavigationController *navigation;
```

Share Improve this answer Follow

answered Dec 18, 2010 at 13:21

zgobolos

11 ● 1