In Cocoa do I need to remove an Object from receiving KVO notifications when deallocating it?

Asked 16 years, 4 months ago Modified 2 years, 3 months ago

Viewed 7k times

23

When I've registered an object **foo** to receive KVO notifications from another object **bar** (using addObserver:...), if I then deallocate **foo** do I need to send a `removeObserver:forKeyPath:` message to **bar** in -dealloc?

cocoa macos

Share

Improve this question

Follow

edited Sep 11, 2012 at 14:07

Tieme
65.2k ● 22 ● 107 ● 157

asked Aug 17, 2008 at 21:05

Redwood
69.1k ● 45 ● 129 ● 187

3 Answers

Sorted by: Highest score (default)

39

You need to use `-removeObserver:forKeyPath:` to remove the observer before `-[NSObject dealloc]` runs, so yes, doing it in the `-dealloc` method of your class would work.

Better than that though would be to have a deterministic point where whatever owns the object that's doing the observing could tell it it's done and will (eventually) be deallocated. That way, you can stop observing immediately when the thing doing the observing is no longer needed, regardless of when it's actually deallocated.

This is important to keep in mind because the lifetime of objects in Cocoa isn't as deterministic as some people seem to think it is. The various Mac OS X frameworks themselves **will** send your objects `-retain` and `-autorelease`, extending their lifetime beyond what you might otherwise think it would be.

Furthermore, when you make the transition to Objective-C garbage collection, you'll find that `-finalize` will run at very different times — and in very different contexts — than `-dealloc` did. For one thing, finalization takes place on a different thread, so you really **can't** safely send `-removeObserver:forKeyPath:` to another object in a `-finalize` method.

Stick to memory (and other scarce resource) management in `-dealloc` and `-finalize`, and use a separate `-invalidate` method to have an owner tell an object you're done with it at a deterministic point; do things like removing KVO observations there. The intent of your code will be clearer and you will have fewer subtle bugs to take care of.

Share Improve this answer Follow

answered Aug 18, 2008 at 0:33

Chris Hanson
55k ● 8 ● 74 ● 104

7 I can confirm from experience the pain that you will receive in endless crashes if you don't remove your observers. – Jeff Apr 28, 2010 at 23:33

As of OS X 10.7 Lion, there's also `-removeObserver:forKeyPath:context:`, which lets you pass in the same context you passed to `-addObserver:forKeyPath:options:context:`. Passing a unique context ensures you won't remove someone else's observation when you remove yours. – Chris Hanson Sep 1, 2011 at 0:33

2 Incase it helps anyone I wanted to post this here, I was registering for a notification by name and then attempting to unregister via key which causes a crash stating that the object was not registered for that notification, which to me was confusing. So if you register by name make sure you are using `removeObserver:name:object:` to unregister rather than `-removeObserver:forKeyPath:context:` – Chris Wagner Nov 8, 2011 at 1:17

5

A bit of extra info that I've gained by painful experience: although NSNotificationCenter uses zeroing weak references when running under garbage collection, KVO does not. Thus, you can get away with not removing an

NSNotificationCenter observer when using GC (when using retain/release, you still need to remove your observer), but you must still remove your KVO observers, as Chris describes.

Share Improve this answer

Follow

answered Aug 29, 2008 at 15:56

Barry Wark
108k ● 24 ● 182 ● 206

Is this documented somewhere? – hpique Oct 6, 2012 at 7:48

2

Definitely agree with Chris on the "Stick to memory (and other scarce resource) management in -dealloc and -finalize..." comment. A lot of times I'll see people try to invalidate NSTimer objects in their dealloc functions. The problem is, NSTimer retains it's targets. So, if the target of that NSTimer is self, dealloc will never get called resulting in some potentially nasty memory leaks.

Invalidate in `-invalidate` and do other memory cleanup in your `dealloc` and `finalize.`

Share Improve this answer

Follow

answered Sep 15, 2008 at 18:16

itsbonczek