Awake from nib and applicationDidFinishLaunching

When developing an application in [**Objective-C**](http://lib.csdn.net/base/objective-c) and using Cocoa, there is a lot of “magic” that happens in the background. As we get more comfortable with the language and the APIs, we begin to discover the source of that magic and understand not only WHY it works but HOW it works.

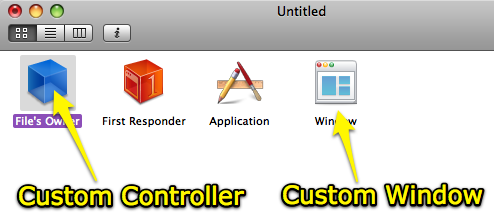
One of those areas is the initialization and callbacks from the nib files to my code. Normally, when I want a controller to do something after the NIB/XIB has loaded, I add the method **-(void)awakeFromNib** and know that I will receive a call when all of the connections into the NIB/XIB are complete. But on what object does this get called and how?

In another case, I can be notified when the application has finished launching. This method, **-(void)applicationDidFinishLaunching:(NSNotification\*)aNotification** gets called after the MainMenu.[nib|xib] has finished loading. How does this method get called and on who does it get called?

And the third question. If I am using both of these spells above, which gets called first and why?

**awakeFromNib**

awakeFromNib is called on every object that is initialized or referenced from within a nib file. For example, take a look at the following nib file:

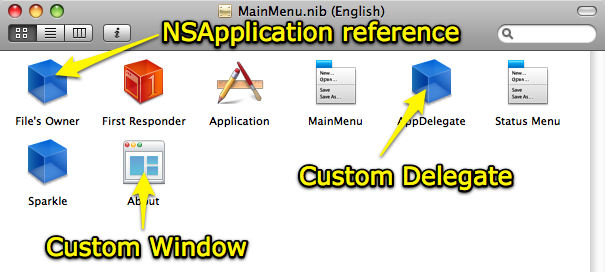
[ Image 1](http://www.cimgf.com/wp-content/uploads/2008/03/afnvadfl_nib_example.png)

In this nib file, I have references to two pieces of custom code: The Window Controller and the Window itself. So which of these two gets the awakeFromNib call? Both do.

Specifically, when a nib file has completed initializing all of the objects, it then loops back through every object referenced in the nib and if the object responds to awakeFromNib, it calls awakeFromNib on the object. Therefore, you can have logic in any or all of your objects in the awakeFromNib call.

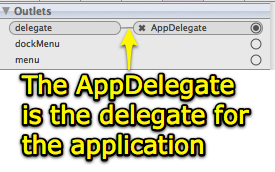
**applicationDidFinishLaunching:**

In the nib above, which object would receive a applicationDidFinishLaunching: call? None of them. The nib in that example is not set up to be the primary nib for an application. When a nib is designed to be the primary or initial nib for an application it will have a reference to NSApplication in it.

[ Image 2](http://www.cimgf.com/wp-content/uploads/2008/03/afnvadfl_app_example1.png)

In this example you can see the reference to the NSApplication in the nib. This nib is designed to be the initial nib loaded by the application. This can be confirmed by looking at the File’s Owner class type and see that it points to NSApplication rather than an NSObject or some custom controller class (Image 3). So who gets the applicationDidFinishLaunching: call? While you cannot see it in the image above, the object ‘AppDelegate’ is referenced as the delegate for the NSApplication.

[ Image 3](http://www.cimgf.com/wp-content/uploads/2008/03/afmvadfl_app_reference1.png)

[ Image 4](http://www.cimgf.com/wp-content/uploads/2008/03/afnvadfl_appdelegate.png)

Therefore, when the application has indeed finished launching, then this method on the AppDelegate and only the AppDelegate will be called. Only one object gets this method per application launch.

**Call order**

In the second example above, it is possible to have an awakeFromNib method and an applicationDidFinishLaunching: method in the AppDelegate. Which would be called first and why?

The answer is that the awakeFromNib will be called first. When the nib file has been initialized, each object referenced in the nib will be looped through and they will all receive an awakeFromNib call if they respond to the message. After all of that is done then the Application’s delegate will receive the applicationDidFinishLaunching: call. This is the notice that everything is loaded and that the application is ready to start receiving user input.

**Conclusion**

If your code is localized to only the object you are working with then putting the code in the awakeFromNib is the perfect place for it. However, if your code is going to be manipulating more than one object in the nib then it is probably safer to put it in the applicationDidFinishLaunching: instead. This will insure that everything is loaded, referenced and ready to [**Go**](http://lib.csdn.net/base/go).