MATT GALLAGHER

> **Please note:** this article is part of the older "Objective-C era" on Cocoa with Love. I don't keep these articles up-to-date; please be wary of broken code or potentially out-of-date information. Read "A new era for Cocoa with Love" for more.

Cocoa Application Startup

March 14, 2008 by Matt Gallagher
Tags: Objective-C, AppKit

How does a Cocoa Application start up? Where are the main places to put code so that it gets run on startup? Learn the answers in this series of startup diagrams.

The natural first question

After working through the first Cocoa tutorials and learning how to hook up basic methods to buttons and menus, the most common question I hear from new Cocoa programmers is:

> Where do I put my code so that it runs when my Cocoa Application starts up?

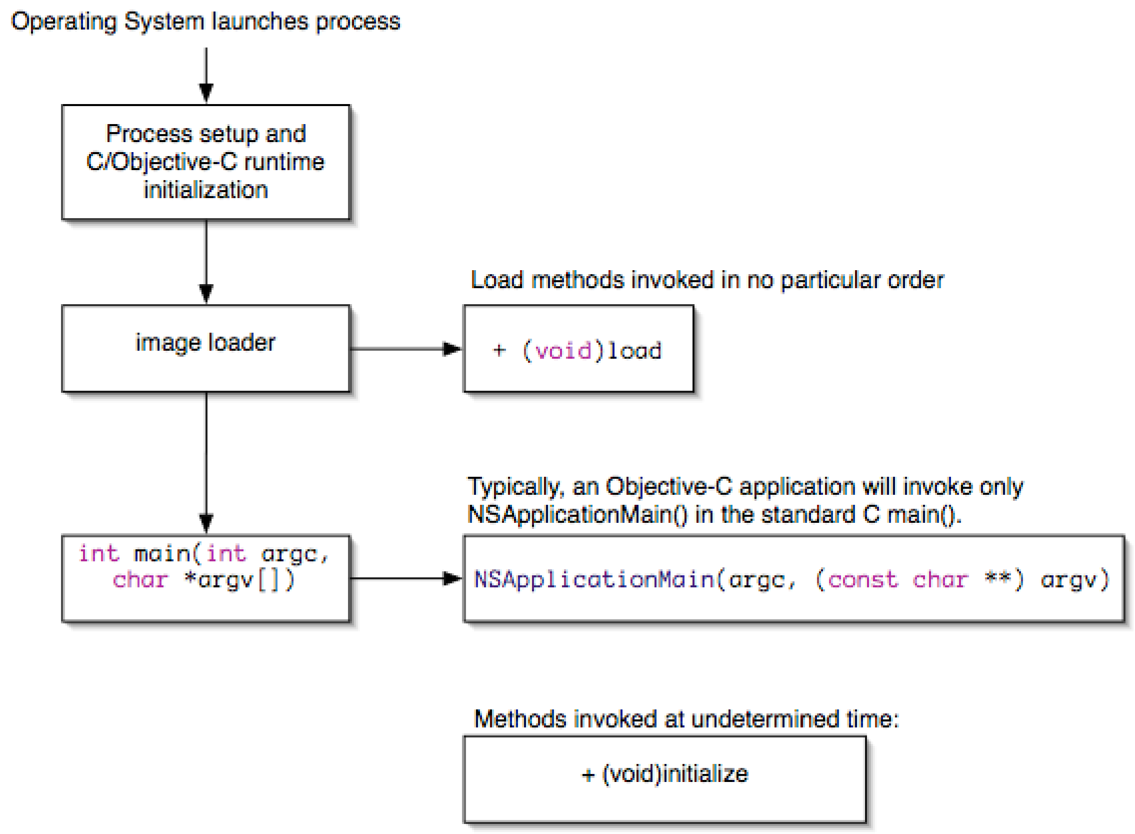
Cocoa makes it very easy to edit and connect things in Interface Builder but you quickly reach the limits of this simple program setup. Documents and application windows can magically appear but eventually you'll want to have control over these default operations.

Once you need to do more, you need to start piercing through the magic of Cocoa Application startup and understand what the NSApplication does, how the Interface Builder NIB files appear and how to run customization code for documents and other elements.

to control your program and make it your own. This starts at the very beginning. But it can be tricky to decide where to put your code: Cocoa gives you many different places to initialize different components so that appropriate setup is performed in the appropriate place. I'm going to show all the common places in a Cocoa Application where code is run on startup. With luck, knowing the options will help you choose the right one.

Objective-C runtime initialization

The Objective-C runtime offers places to initialize your classes during its startup, as the following diagram shows:


Operating System launches process
Process setup and C/Objective-C runtime initialization
image loader
Load methods invoked in no particular order
+ (void)load
Typically, an Objective-C application will invoke only NSApplicationMain() in the standard C main().
int main(int argc, char *argv[])
NSApplicationMain(argc, (const char **) argv)
Methods invoked at undetermined time:
+ (void)initialize

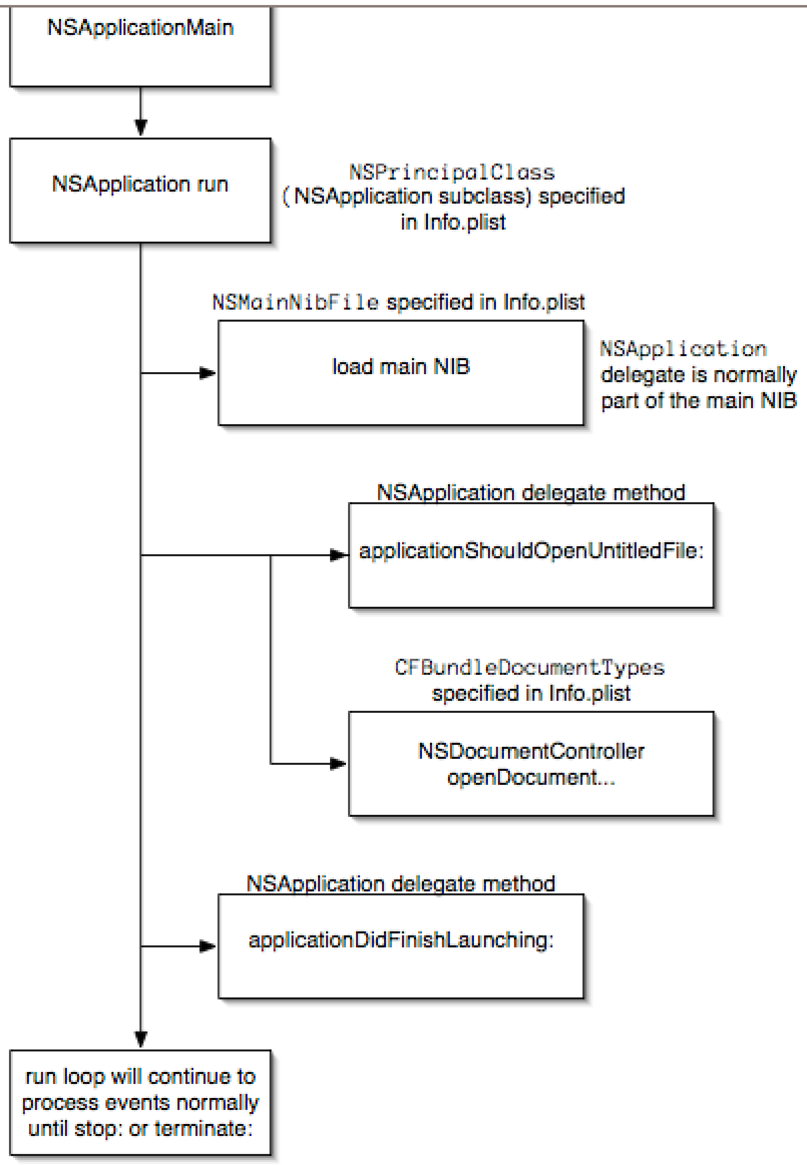
The main() method works in Objective-C as it would for any standard C program and you can perform global initializations here if you want. Typically though, Cocoa Applications do nothing here except invoke NSApplicationMain(). The reason for largely ignoring this entry point is that classes, application instances, documents and user interface elements have their own means of initializing themselves and it makes more sense to use these locations.

Traditionally, in C languages, main() is the earliest place where you can perform startup and initialization. In Objective-C, the first initialization offered occurs even earlier than this: +(void)load methods. Any class can have a +(void)load method and it will be occurs during "image loading" (some time before main() is invoked). Since classes are loaded in no particular order and other classes may not yet be loaded, you must be careful what you do here — generally, you shouldn't refer to other classes in +(void)load methods.

Objective-C also provides a similar (but safer since you *can* refer to other classes) setup method: +(void)initialize. This is the most common place for performing class-specific setup. Before a message is sent to an object of a class for the first time, +(void)initialize is sent to the class and all its super classes (if they haven't been initialized already). These messages are sent in hierarchical order (super classes receive the message before descendant classes). While safer (because all classes are loaded by this point) the timing of its invocation is less deterministic since +(void)initialize is only invoked when a first message is actually sent to the class — if a class is never used, the message is never sent.

NSApplication initialization

Here is the typical startup for NSApplication (as invoked through NSApplicationMain() in the previous diagram):


NSApplicationMain
NSApplication run
NSPrincipalClass (NSApplication subclass) specified in Info.plist
NSMainNibFile specified in Info.plist
load main NIB
NSApplication delegate is normally part of the main NIB
NSApplication delegate method
applicationShouldOpenUntitledFile:
CFBundleDocumentTypes specified in Info.plist
NSDocumentController openDocument...
NSApplication delegate method
applicationDidFinishLaunching:
run loop will continue to process events normally until stop: or terminate:

This image shows the application (normally NSApplication but this is configurable in the Info.plist file) being initialized and run. Despite the fact you *can* change the principal class of your application, it is advised that you don't — so this isn't the first place to consider putting startup code.

Instead, you should look towards the three main functions that the application performs on startup (load main NIB, load first document, notify NSApplication delegate). These are the key areas to hook into.

> An NSApplication is only considered to handle documents if CFBundleDocumentTypes is configured in the Info.plist file. Without this value in the Info.plist, the NSApplication will skip document handling (non-document applications).

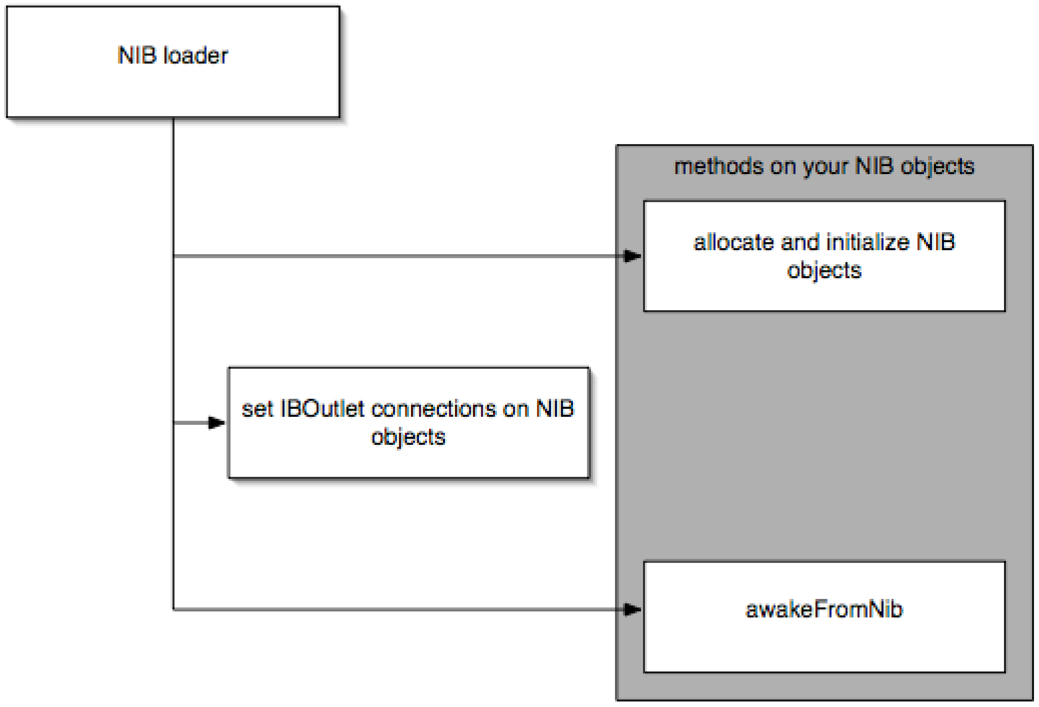
The first of these areas to consider is the "main NIB" file. A NIB file is the standard user-interface file and it is created in Interface Buillder. The "main NIB" is the user-interface common location for the NSApplication delegate. If you don't know what a delegate is, you can read Apple's description of delegates but it's really any object that is connected to the NSApplication by its "delegate" outlet (usually in Interface Builder). The delegate is special because NSApplication sends it messages in response to certain events. These are the delegate methods.

At startup, the NSApplication delegate receives one of its most important messages, applicationDidFinishLaunching:. Almost every application implements this method. It is the best place to fix program-wide settings after everything has been created and loaded.

If you need something more specific to a window or document to be initialized, then you'll need to hook into one of the other steps. If you're writing a non-document application with a single fixed window, then it will likely be loaded along with the main NIB, so you'll want to read the next section about NIB loading. If you are writing a document application, you'll want to look at the document loading section.

NIB loading

An Interface Builder NIB file is loaded as follows:


NIB loader
methods on your NIB objects
allocate and initialize NIB objects
set IBOutlet connections on NIB objects
awakeFromNib

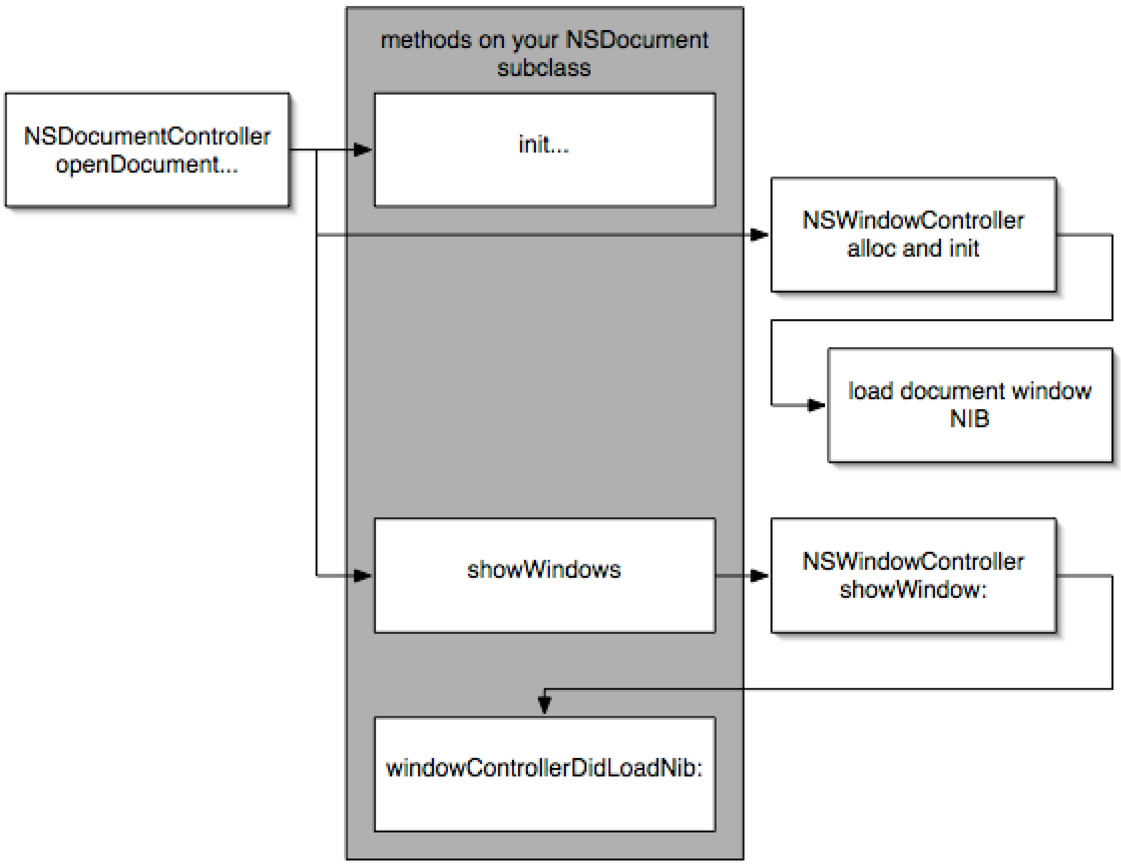
Application windows, document windows, the menubar and the NSApplication delegate are all typically loaded from a NIB file in this fashion.

connections are configured according to the settings in the NIB file, so you can initialize your own object but you can't yet read objects connected through Interface Builder.

For this reason, the a common way to initialize an object loaded from a NIB file, is to implement the -(void)awakeFromNib method on the object. If this method is present on an object, the NIB loader will invoke it after the whole NIB is initialized and connected, making it the perfect place to perform configuration involving multiple objects.

Document loading

A document is loaded as follows:


methods on your NSDocument subclass
NSDocumentController openDocument...
init...
NSWindowController alloc and init
load document window NIB
showWindows
NSWindowController showWindow:
windowControllerDidLoadNib:

The key places to initialize a document are: its init methods, in the loading of the document's NIB file and in windowControllerDidLoadNib:. Simply put, these offer configuration locations for before, during and after loading of the document's window resources.

Conclusion

methods, notifications and overrideable methods if you have genuine need of them. What I have shown are the most common and most useful locations.

These locations are:

- +(void)initialize
- -(void)awakeFromNib
- -(void)windowControllerDidLoadNib:
- -(void)applicationDidFinishLaunching:

as well as the usual init methods for your own subclasses.

Know what each of these methods are, when they are invoked and understand objects for which they are responsible. They will let you perform almost all configuration on Cocoa application startup.

Previous article:
Break into Debugger

Next article:
Construct an NSInvocation for any message, just by sending