What Is Xcode® With Interface Builder And What Are Xib/Nib Files? From March 13, 2012

Definitions

'Xcode®' is the Integrated Development Environment (IDE) for Mac OS X® provided by Apple® and is widely used among Objective-C developers. It provides the context in which all of the various parts of code and design elements (such as windows and buttons) are tied together.

'Interface Builder' ('IB') is the Graphic User Interface (GUI) editor for 'Xcode®'. Prior to version 4.x of 'Xcode', 'IB' was closely tied together with 'Xcode'. Starting with version 4, 'IB' is integrated with and part of Xcode®.

Cocoa is the environment that provides the general "look and feel" of Mac OS X®, from it's color schemes, to it's behaviors in the various control elements (windows, check boxes, push buttons, radio buttons, and so on). It is an object-oriented application environment designed specifically for developing Mac applications quickly and efficiently.

'Xcode®' is also the best GUI tool for Mac OS X® development, because it is tightly connected to Cocoa. In fact, one could say that 'Xcode' is part of Cocoa.

Files that end with `.xib` or `.nib` are created by 'IB' and contain the information which ties design elements (windows and controls) together with code (outlets and actions). The `xib`/`nib` files are automatically loaded by your application when it is run.

Note that there are some significant differences between `.xib` and `.nib` files, the nature of which are beyond the scope of this document. Both describe the various elements of your application and tie them together with your code. `.xib` files are a text-based XML representation of user data and is a pre-compiled `.nib` file. `.nib` files may also contain additional information about executing your code and custom objects or classes that should be used in your program. Even Apple's own documentation contains relatively little information about these files.

Two Special Keywords

In 'Xcode/Interface Builder' ('X/IB') you draw your windows and controls and connect them to variables (or in "Apple-speak," properties) and functions (or procedures, or in "Apple-speak," methods) declared in your source code in the Objective-Basic IDE. Two special keywords are used for this task.

IBOutlet

`IBOutlet` lets you connect variables (code you have written) to controls (design elements, such as buttons). Think of this keyword as an "outlet" from your code to your button, check box, or other design element. The variables defined in your source code are just like ordinary variables, except that they point to a control in a `.nib`/`.xib` file, which is loaded on application startup. Here's an example.

```
IBOutlet mybutton As NSButton
```

In this case, the variable `mybutton` in your source code refers to a button object in 'IB'. The result of a function that operates on the variable of your code will be directed to the button object. The

next entry makes this clearer.

IBAction

`IBAction` is used to when an "event" occurs, such as the clicking of a button on the screen. This of this keyword as the "action" written into your software that is taken by the button (or other design element), when it is triggered. "Actions" are defined in your source code like ordinary functions and then connected to the interface controls in Xcode/Interface Builder. Here is an example.

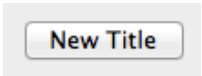
```
IBAction helloworld(sender As NSObject)
     mybutton.Title = "New Title" ' or other code here
End IBAction
```

In this case, if the Button is pressed in the user interface, the function `helloworld` defines the variable `Title` of the object `mybutton` as "New Title." `mybutton` addresses NSButton object *via* the `IBOutlet` declaration and the title of the button is changed.

A schematic looks something like this:

Objective-Basic Code	Explanation		Xcode/Interface Builder
`IBOutlet mybutton As NSButton`	`IBOutlet` defines the variable `mybutton` as an "outlet" to the button object (of the the type `NSButton`) in 'X/IB'.	→	
			↓

`IBAction myevent(sender As id)`
`  mybutton.Title = "New Title"`
`End IBAction`
	An "event" occurs (here, a button is clicked), calling a function/method (identified by the keyword `IBAction`) which changes the title property of the button object through the `mybutton` variable.	←	

↓			
			

Application startup, automatically loading xib/nib file, and object creation of windows and controls

The main `.xib/.nib` file is loaded automatically loaded by Cocoa for you. When it loads, the `.xib/.nib` loader allocates and initializes all objects, and "hooks up" all outlets and actions defined by `IBOutlet` and `IBAction` in your code. After all outlets and actions are connected, the loader then calls the special event procedure `AwakeFromNib` of each object in the `.xib/.nib` file.

This is where you can access outlets to set up default values or do configuration in code.

Whenever you change code by adding or removing IBActions or IBOutlets in the Objective-Basic IDE, hit the "Build" button to inform Xcode/Interface Builder about the changes in your source code. In the current version of the IDE of

Objective-Basic, it automatically updates the changes for IB, if you use the Interface Builder button in the toolbar or the related items in the menubar to switch to Interface Builder.

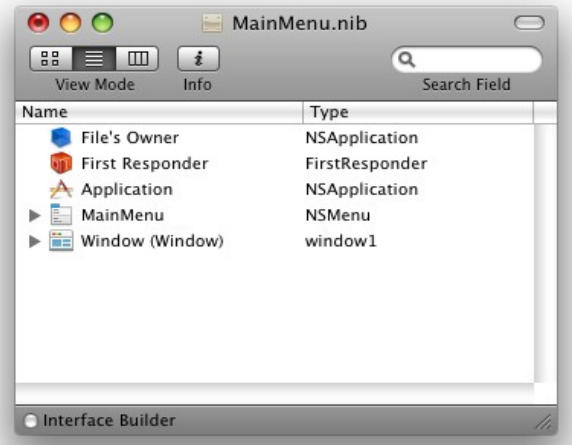
Xcode® Variations

Xcode 3: Several windows in Interface Builder let you create the needed GUI

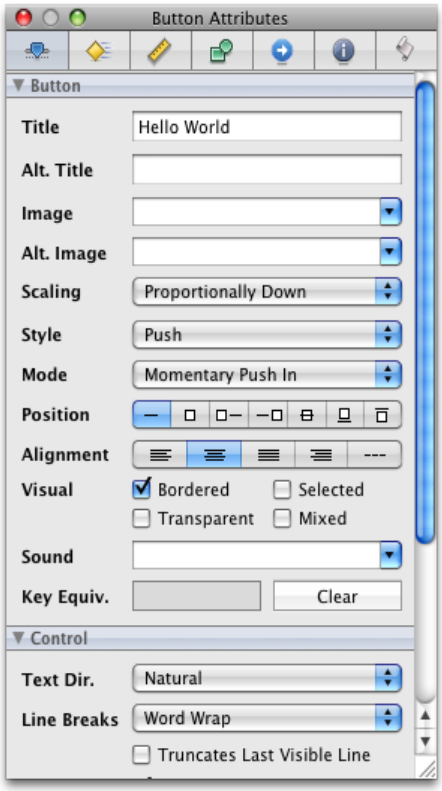
There is a main window listening all objects and windows. Double click on a object item in the list and it will be opened, ready for editing.

The property window let you change the properties of windows and controls. Select *Tools | Attributes Inspector* in the menu bar to see it.

The object and control list let you choose the new objects or controls to be insert. Select *Tools | Library* in the menu bar to see it.



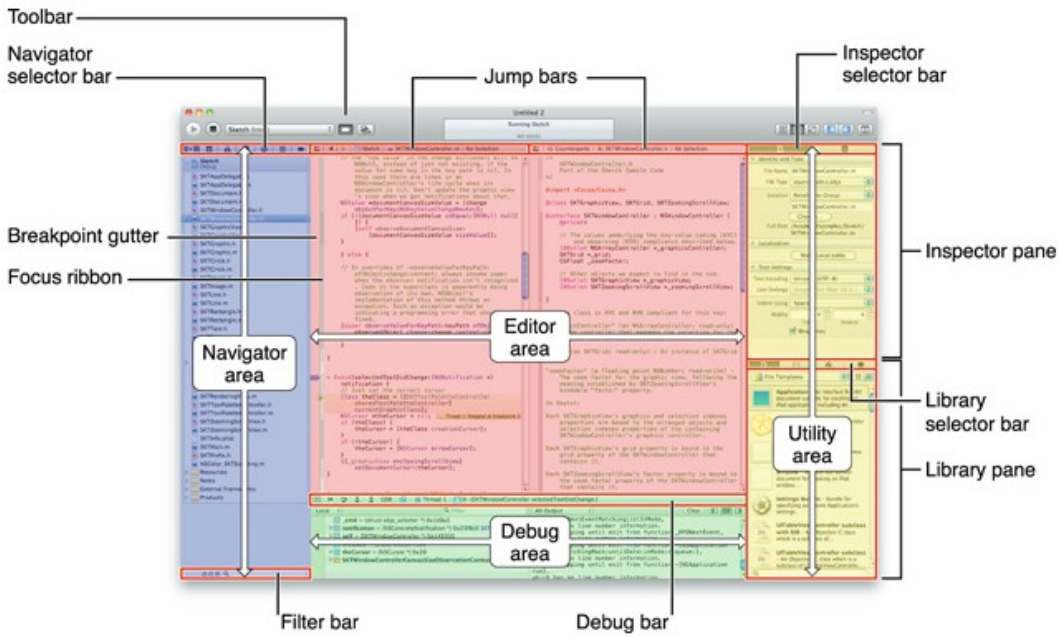
Screenshot 1: In Xcode 3, the list of windows and objects available in the nib file looks like this.



Screenshot 3: In Xcode 3, the property window let you change the properties of windows and controls.

Screenshot 2: In Xcode 3, the object and control list let you choose the new objects or controls to be insert.

Xcode 4: One window with several panes in Interface Builder let you create the needed GUI



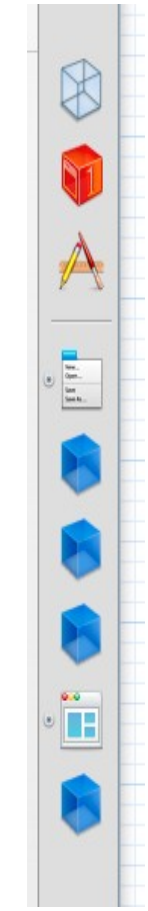
Screenshot 4

Xcode 4 uses a single window, called the *workspace window*, that holds most of the data you need. The 3 components mentioned in Xcode 3 are available in Xcode 4 and are found as follows.

• The Windows and Objects list is found on the bar between the navigator and editor areas. See screenshot 5.

• The properties of an object can be found under the Attributes Inspector on the Inspector selector bar. See the upper right of screenshot 4 above.

• The list of objects and controllers is found under the Object library button on the Library selector bar. See the lower right of screenshot 4.



Screenshot 5

Switching between Xcode/Interface Builder and the IDE of Objective-Basic

In the menu bar, select *Project* | *Xcode for MainMenu.xib* and 'X/IB' will be open with the `.xib`/`.nib` file for the current project (`MainMenu.xib`), ready for editing.

Always save your changes in 'X/IB' in order to communicate the changes in 'Xcode' to the Objective-Basic IDE.

To be continued ...

Copyright