Q: What is the "file's owner" of a .nib file?

A: The .nib's file's owner is an object that's external to the .nib file that's the conduit between objects in the .nib file and the other objects in your application.

Each .nib file has one, and only one, owner. For small applications, the owner is generally NXApp, the application object itself, although it can be an object of any class. The owner is the only external object that may be the explicit target of action messages from Controls within the .nib file. The owner may also have outlets that are initialized at runtime to the id's of objects within the .nib file.

A .nib file is actually an archived collection of objects (windows, panels, controls, etc) that you designed in InterfaceBuilder. When that file is unarchived at runtime and those objects are loaded, some object needs to "own" them. For example, the main menu is loaded and owned by NXApp. When you only have one .nib file, actually all objects are loaded and owned by NXApp. When you have multiple .nib files (like /NextDeveloper/Examples/Draw), you explicitly set the owner of each file. The owner is like the "shepherd" of the objects in its file. The owner sends messages to its objects (makeKeyAndOrderFront: to panels, for example) and it also can receive messages from its object (a button that sent its target method "quit:" to the file's

owner, for example). As usual, you make these connections in InterfaceBuilder.

The file's owner is the only link between those objects and objects elsewhere in the other .nib files of this application. For example, you could create an info panel in a separate .nib module, and have a custom object Controller which owns that .nib file. Your main menu item "Info..." is in the main .nib file and doesn't know about the Info panel because it is in a separate .nib file. However, the main menu can send a message to the Controller object, which, in turn, sends a message to the info panel telling it to display itself.

Why would you choose to use separate .nib files? At launch time, your entire .nib file (all the panels, menus, buttons, whatnot) is unarchived. This may inefficiently squander your precious time and memory. If you have a panel that is sparsely used, you may instead want to place it in a separate nib module and load it when, and if, it is needed.

InterfaceBuilder modules also allow you to create custom panels, windows, and objects that are reusable. You could place the panel in a nib module and design a custom object that will control and own it, and then by instantiating that object and reusing the .nib module file, you can add that panel and its functionality to any

application you create.

In NEXTSTEP Release 2, look at the two-.nib-file example of the Text Editor in the InterfaceBuilder tutorial (Chapter 8 of the NeXTstep Concepts manual) as a guide.

In NEXTSTEP Release 3, the InterfaceBuilder tutorial is covered in Chapter 17 of the Development Tools and Techniques book (the on-line location is /NextLibrary/Documentation/NextDev/DevTools/17_TextApp).

In all versions, see /NextDeveloper/Examples/Draw for a more advanced use.

See also ../Development_Environment/creating_nib_modules.rtf.

QA510

Valid for 1.0, 2.0, 3.0, 3.1