;DisplayingAnAttentionPanel.rtfd;;¬   Next Section

Loadingnibfilesdynamicallyaninfopanel;¬Loading nib files dynamically: an info panel

1. **Create an outlet for the Info Panel and an action method that displays the Info Panel in the application's controller class.**
2. **Connect the Info Panel command to the controller object.**
3. **Choose Document arrow.eps ¬ New Modules arrow.eps ¬ New Info Panel to create a nib file for the Info Panel.**
4. **Add the controller class to the new nib file.**
5. **Assign the controller class to File's Owner.**
6. **Connect the Info Panel to the File's Owner outlet for the panel.**
7. **Implement the action method that loads the Info Panel's nib file.**

The steps you follow to create, load, and manage an Info Panel are common to creating any special-use nib file. First, in the main nib file, you create an object that knows about the Info Panel. Usually, this object is the application's controller object. Define the class of this object in the Classes display of the main nib file. When you do, specify the necessary outlet and action for the Info Panel.

_LoadingNibFiles1.eps ¬

Instantiate the controller class, and connect the action to the menu command.

_LoadingNibFiles2.eps ¬

Now choose the New Info Panel command. When you do, Interface Builder displays a template panel and creates an untitled nib file to contain it. Be sure to save the file (as, for instance, **InfoPanel.nib**).

_LoadingNibFiles3.eps ¬

You cannot connect the Info Panel to the controller object in the main nib file because the panel is in the new auxiliary nib file. You must assign the controller class to File's Owner in the auxiliary nib file and then make the outlet connection between File's Owner and the panel. The first step in this direction is to insert the class definition of the controller class into the auxiliary nib file.

_LoadingNibFiles4.eps ¬

Next, assign the controller class to File's Owner.

_LoadingNibFiles5.eps ¬

Now make a connection in Interface Builder between File's Owner and the title bar of the panel. Select the infopanel outlet in the Connections display and click Connect.

The final step is to write the code (in the **.m** file of the controller class) that implements the action method invoked by the Info Panel command.

```
- (void)showInfoPanel:(id)sender
{
    if (!infoPanel)
         [NSBundle loadNibNamed:@"InfoPanel" owner:self];
    [infoPanel makeKeyAndOrderFront:self];
}
```

Notes on the code: Once an Info Panel is loaded, it is kept in memory until the user quits the application. The code tests the **infoPanel** outlet to determine if the auxiliary nib file containing the panel has already been loaded. If it hasn't, it loads it with **loadNibNamed:owner:**. It is important to specify **self** as owner (**self** being the object that implements the method). Display the panel by sending it the **makeKeyAndOrderFront:** message.

Chapter€6, ªSubclassing,º describes how to add outlets and actions to your custom class and shows how to connect them to instances of your class.   ;../../03_Coding/06_Subclassing/Subclassing.rtfd;;¬

Related Concept: ;DynamicLoadingConcepts.rtfd;linkMarkername MultipleNibFiles:GoodThingsinSmallPieces;, Multiple Nib Files: Good Things in Small Pieces