Font Panel

Cocoa > User Experience

2004-08-31

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Introduction to Font Panel

Font Panel explains how to use the Font panel, which is defined by the NSFontPanel class.

Who Should Read This Document

You should read this document if you need to understand how the Font panel interacts with the Font manager and other components of the text system, particularly if you need to modify that behavior.

To understand the information in this document, you should understand generally the text system's capabilities and architecture, and you should understand basic Cocoa programming conventions.

Organization of This Document

This document contains the following articles:

- "The Font Panel" (page 7) describes the interaction of the Font panel and Cocoa text objects and explains how to use the delegate of NSFontPanel to control the display of fonts in the Font panel.
- "Creating a Font Panel" (page 9) explains how to add the Font panel to your application using Interface Builder and, alternatively, how to create and modify the Font panel programmatically.
- "Using the Font Panel" (page 11) describes the messages sent in the interaction among the Font panel, Font manager, and text objects.

See Also

For more information, refer to the following documents:

- *Font Handling* discusses fonts, the font management system, and the user interface to allow the user to interact with available fonts.
- *Text System Overview* provides an overview of the text system.

For related reference information, see the following documents:

- NSFont
- NSFontManager
- NSFontPanel

See Also

The Font Panel

The Font panel, also called the Fonts window, is a user interface object that displays a list of available font families and styles, letting the user preview them and change the font used to display text. Text objects, such as NSTextView, work with NSFontPanel and NSFontManager objects to implement the Application Kit's font conversion system. By default, a text object keeps the Font panel updated with the first font in its selection, or with its typing attributes. It also changes the font in which it displays text in response to messages from the Font panel and Font menu. Such changes apply to the selected text or typing attributes for a rich text object or to all the text in a plain text object.

NSFontManager is the hub for font conversion. It receives the messages from the Font panel and sends messages up the responder chain for action on the text objects.

Normally, an application's Font panel displays all the standard fonts available on the system. If this isn't appropriate for your application—for example, if only fixed-pitch fonts should be used—you can assign a delegate to the NSFontPanel object to filter the available fonts. Before the NSFontPanel object adds a particular font family or face to its list, the NSFontPanel asks its delegate to confirm the addition by sending the delegate a `fontManager:willIncludeFont:` message. If the delegate returns `true` (or doesn't implement this method), the font is added. If the delegate returns `false`, the font isn't added. This method must be invoked before the loading of the main nib file.

Creating a Font Panel

In general, you add the facilities of the Font panel to your application, along with the NSFontManager and the Font menu, through which the user opens the Font panel, using Interface Builder. You do this by dragging a Font or Format menu (which contains a Font submenu) into one of your application's menus. At runtime, the Font panel object is created and hooked into the font conversion system. You can also create (or access) the Font panel using the `sharedFontPanel` class method.

You can add a custom view object to an NSFontPanel using `setAccessoryView`, allowing you to add custom controls to the Font panel. You can also limit the fonts displayed (by default, all fonts) by assigning a delegate to the application's font manager object (see "The Font Panel" (page 7)).

In Objective-C, if you want the NSFontManager to instantiate the Font panel from some class other than NSFontPanel, use the NSFontManager class method `setFontPanelFactory:`. See "Converting Fonts Manually" for more information on using the font conversion system.

Using the Font Panel

You can enable the interaction between a text object and the Font panel using the `setUsesFontPanel:` method. Doing so is recommended for a text view that serves as a field editor, for example.

You can use the Font panel on objects other than standard text fields. The NSFontManager method `setAction:` sets the action (specified by a selector) that is sent up the first responder chain when a new font is selected. The default selector is `changeFont:`. Any object that receives this message from the responder chain should send a `convertFont:` message back to the NSFontManager to convert the font in a manner the user has specified.

This example assumes there is only one font selected:

```
- (void)changeFont:(id)sender
{
    NSFont *oldFont = [self font];
    NSFont *newFont = [sender convertFont:oldFont];
    [self setFont:newFont];
    return;
}
```

If multiple fonts are selected, `changeFont:` must send conversion messages for each selected font. This is useful for objects such as table views, which do not inherently respond to messages from the Font panel.

Document Revision History

This table describes the changes to *Font Panel*.

Date	Notes
2004-08-31	Made editorial revisions throughout. Removed "Using an NSFontPanel Delegate" and folded content into "The Font Panel" (page 7).
2004-02-09	Rewrote introduction and added an index.
2003-07-02	Expanded the section on using NSFontManager's responder chain for custom font management.
2002-11-12	Revision history was added to existing document.

Index