NSPanel Class Reference

Cocoa > User Experience

2009-01-06

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

Simultaneously published in the United States and Canada.

Contents

NSPanel Class Reference

Inherits from	NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Window Programming Guide for Cocoa
Declared in	NSPanel.h
Related sample code	PDF Annotation Editor QTAudioExtractionPanel Quartz Composer WWDC 2005 TextEdit Sketch-112 TextEditPlus

Overview

The `NSPanel` class implements a special kind of window (known as a **panel**), typically performing an auxiliary function.

For details about how panels work (especially to find out how their behavior differs from window behavior), see How Panels Work.

Tasks

Configuring Panels

`- isFloatingPanel` (page 6)
Indicates whether the receiver is a floating panel.

`- setFloatingPanel:` (page 7)
Controls whether the receiver floats above normal windows.

- `becomesKeyOnlyIfNeeded` (page 6)
  Indicates whether the receiver becomes the key window only when needed.
- `setBecomesKeyOnlyIfNeeded:` (page 7)
  Specifies whether the receiver becomes the key window only when needed.
- `worksWhenModal` (page 8)
  Indicates whether the receiver receives keyboard and mouse events even when some other window is being run modally.
- `setWorksWhenModal:` (page 8)
  Specifies whether the receiver receives keyboard and mouse events even when some other window is being run modally.

Instance Methods

becomesKeyOnlyIfNeeded

Indicates whether the receiver becomes the key window only when needed.

```
- (BOOL)becomesKeyOnlyIfNeeded
```

Return Value
`YES` when the panel becomes the key window only when needed, `NO` otherwise.

Discussion
By default, this attribute is set to `NO`, indicating that the panel becomes key as other windows do.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBecomesKeyOnlyIfNeeded:` (page 7)
- `needsPanelToBecomeKey` (`NSView`)

Declared In
`NSPanel.h`

isFloatingPanel

Indicates whether the receiver is a floating panel.

```
- (BOOL)isFloatingPanel
```

Return Value
`YES` when the receiver is a floating panel, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFloatingPanel:` (page 7)

- level (NSWindow)

Declared In
NSPanel.h

setBecomesKeyOnlyIfNeeded:

Specifies whether the receiver becomes the key window only when needed.

- (void)setBecomesKeyOnlyIfNeeded:(BOOL)*becomesKeyOnlyIfNeeded*

Parameters

becomesKeyOnlyIfNeeded

YES makes the panel become the key window only when keyboard input is required. NO makes the panel become key when it's clicked.

Discussion
This behavior is not set by default. You should consider setting it only if most user interface elements in the panel aren't text fields, and if the choices that can be made by entering text can also be made in another way (such as by clicking an item in a list).

If the receiver is a non-activating panel, then it becomes key only if the hit view returns YES from needsPanelToBecomeKey. This way, a non-activating panel can control whether it takes keyboard focus.

Availability
Available in Mac OS X v10.0 and later.

See Also
- becomesKeyOnlyIfNeeded (page 6)
- needsPanelToBecomeKey (NSView)

Declared In
NSPanel.h

setFloatingPanel:

Controls whether the receiver floats above normal windows.

- (void)setFloatingPanel:(BOOL)*floatingPanel*

Parameters

floatingPanel

YES to make the receiver a floating panel (NSFloatingWindowLevel). NO to make the receiver behave like a normal window (NSNormalWindowLevel).

Discussion
By default, panels do not float above other windows. It's appropriate for an panel to float above other windows only if all of the following conditions are true:

- It's small enough not to obscure whatever is behind it.
- It's oriented more to the mouse than to the keyboard—that is, if it doesn't become the key window or becomes so only when needed.

- It needs to remain visible while the user works in the application's normal windows—for example, if the user must frequently move the cursor back and forth between a normal window and the panel (such as a tool palette), or if the panel gives information relevant to the user's actions in a normal window.
- It hides when the application is deactivated (the default behavior for panels).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isFloatingPanel` (page 6)
- `setLevel:` (NSWindow)

Declared In
`NSPanel.h`

setWorksWhenModal:

Specifies whether the receiver receives keyboard and mouse events even when some other window is being run modally.

```
- (void)setWorksWhenModal:(BOOL)worksWhenModal
```

Parameters
worksWhenModal
`YES` to make the panel receive events even during a modal loop or session. `NO` to prevent the panel from receiving events while a modal loop or session is running.

Discussion
See "How Modal Windows Work" for more information on modal windows and panels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `worksWhenModal` (page 8)
- `runModalForWindow:` (NSApplication)
- `runModalSession:` (NSApplication)

Declared In
`NSPanel.h`

worksWhenModal

Indicates whether the receiver receives keyboard and mouse events even when some other window is being run modally.

```
- (BOOL)worksWhenModal
```

Return Value
`YES` when the receiver receives keyboard and mouse events even when some other window is being run modally, `NO` otherwise.

Discussion
By default, this attribute is set to `NO`, indicating a panel's ineligibility for events during a modal loop or session. See "How Modal Windows Work" for more information on modal windows and panels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setWorksWhenModal:` (page 8)
- `runModalForWindow:` (`NSApplication`)
- `runModalSession:` (`NSApplication`)

Declared In
`NSPanel.h`

Constants

Alert Panel Return Values

These constants define values returned by the `NSRunAlertPanel` function and by the `NSApplication` method `runModalSession:` when the modal session is run with an `NSPanel` provided by the `NSGetAlertPanel` function.

```
enum {
   NSAlertDefaultReturn = 1,
   NSAlertAlternateReturn = 0,
   NSAlertOtherReturn = -1,
   NSAlertErrorReturn = -2
};
```

Constants

`NSAlertDefaultReturn`
The user pressed the default button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertAlternateReturn`
The user pressed the alternate button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertOtherReturn`
The user pressed a second alternate button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertErrorReturn`
: The alert cannot identify the reason it was closed; it may have been closed by an external source or by a button other than those listed above.

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

Declared In
`NSPanel.h`

Modal Panel Return Values

These constants define the possible return values for such methods as the `runModal...` methods of the `NSOpenPanel` class, which tell which button (OK or Cancel) the user has clicked on an open panel.

```
enum {
   NSOKButton = 1,
   NSCancelButton = 0
};
```

Constants
`NSCancelButton`
: The Cancel button

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

`NSOKButton`
: The OK button

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

Declared In
`NSPanel.h`

Style Masks

The `NSPanel` class defines the following constants for panel styles:

```
enum {
   NSUtilityWindowMask = 1 << 4,
   NSDocModalWindowMask = 1 << 6,
   NSNonactivatingPanelMask = 1 << 7
   NSHUDWindowMask = 1 << 13
};
```

Constants
`NSDocModalWindowMask`
: The panel is created as a modal sheet.

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

`NSUtilityWindowMask`

The panel is created as a floating window.

Available in Mac OS X v10.0 and later.

Declared in `NSPanel.h`.

`NSNonactivatingPanelMask`

The panel can receive keyboard input without activating the owning application.

Valid only for an instance of `NSPanel` or its subclasses; not valid for a window.

Available in Mac OS X v10.2 and later.

Declared in `NSPanel.h`.

`NSHUDWindowMask`

The panel is created as a transparent panel (sometimes called a "heads-up display").

Valid only for an instance of `NSPanel` or its subclasses; not valid for a window.

Using the C bitwise OR operator, `NSHUDWindowMask` can be combined with other style masks (some of which are documented in `Window_Style_Masks`) with the following results:

`NSBorderlessWindowMask`

Borderless window with transparent panel transparency and window level.

`NSTitledWindowMask | NSUtilityWindowMask`

Titled window with transparent panel transparency and window level. This combination can be additionally combined with any of the following:

`NSClosableWindowMask`

Titled window with transparent panel close box, transparency, and window level.

`NSResizableWindowMask`

Titled window with transparent panel resize corner, transparency, and window level.

`NSNonactivatingPanelMask`

No effect on appearance, but owning application is not necessarily active when this window is the key window.

The following constants cannot be combined with `NSHUDWindowMask`: `NSMiniaturizableWindowMask`, `NSTexturedBackgroundWindowMask`, `NSDocModalWindowMask`, and `NSUnifiedTitleAndToolbarWindowMask`.

Available in Mac OS X v10.5 and later.

Declared in `NSPanel.h`.

Declared In

`NSPanel.h`

Document Revision History

This table describes the changes to *NSPanel Class Reference*.

Date	Notes
2009-01-06	Added information about NSHUDWindowMask constant.
2008-07-11	Added a link to a document that describes panel behavior.
2007-03-26	Updated for Mac OS X v10.5
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index