NSSavePanel

Inherits From:	Windows:	NSObject
	Mach:	NSPanel : NSWindow : NSResponder : NSObject
Conforms To:	Windows:	NSObject (NSObject)
	Mach:	NSCoding (NSResponder)
		NSObject (NSObject)
Declared In:	AppKit/NSSavePanel.h	

Class at a Glance

Purpose

An NSSavePanel object manages a panel that allows users to specify the directory and name under which a file is saved. It supports browsing of the file system and it accommodates custom accessory views. An NSSavePanel is a recycled object: when you request a Save panel, NSSavePanel tries to reuse an existing Save panel rather than create a new one.

Principal Attributes

- Delegate
- Form
- Title
- Directory
- Browser
- Prompt
- File name
- Accessory view

Creation

+ savePanel	(class method)

Commonly Used Methods

runModal	Displays the panel and begins the event loop.
filename	Returns the selected or entered file name.
directory	Returns the full path of the selected file.
ok:	Invoked when users click OK.

Class Description

NSSavePanel creates and manages a Save panel, and allows you to run the panel in a modal loop. The Save panel provides a simple way for a user to specify a file to use when saving a document or other data. It can restrict the user to files of a certain type, as specified by a file name extension.

When the user decides on a file name, the message **panel:isValidFilename:** is sent to the NSSavePanel's delegate. If it responds to that message, the delegate can determine whether the specified file name can be used; it returns YES if the file name is valid, or NO if the Save panel should stay up and wait for the user to type in a different file name.

Typically, you access an NSSavePanel by invoking the **savePanel** method. When the class receives a **savePanel** message, it tries to reuse an existing panel rather than create a new one. When a panel is reused its attributes are reset to the default values so that the effect is the same as receiving a new panel. Because a Save panel may be reused, you shouldn't modify the instance returned by **savePanel** except through the methods listed below. For example, you can set the panel's title and required file type, but not the arrangement of the buttons within the panel. If you must modify the Save panel substantially, create and manage your own instance using the **alloc...** and **init...** methods rather than the **savePanel** method.

A typical programmatic use of NSavePanel requires you to:

- Invoke **savePanel.**
- Configure the panel (for instance, set its title or add a custom view).
- Run the panel in a model loop.
- Test the result; if successful, save the file under the chosen name and in the chosen directory.

The following code fragment demonstrates this sequence. (Two objects in this example, **newView** and **textData**, are assumed to be defined and created elsewhere.)

```
NSSavePanel *sp;
int runResult;

/* create or get the shared instance of NSSavePanel */
sp = [NSSavePanel savePanel];

/* set up new attributes */
[sp setAccessoryView:newView];
[sp setRequiredFileType:@"txt"];

/* display the NSSavePanel */
runResult = [sp runModalForDirectory:NSHomeDirectory() file:@""];

/* if successful, save file under designated name */
if (runResult == NSOKButton) {
    if (![textData writeToFile:[sp filename] atomically:YES])
          NSBeep();
}
```

Method Types

Creating an NSSavePanel	+ savePanel
Customizing the NSSavePanel	– setAccessoryView: – accessoryView – setTitle: – title – setPrompt: – prompt
Setting directory and file type	– setDirectory: – setRequiredFileType: – requiredFileType – treatsFilePackagesAsDirectories – setTreatsFilePackagesAsDirectories: – validateVisibleColumns
Running the NSSavePanel	– runModal – runModalForDirectory:file:
Getting user selections	– directory – filename
Action methods	– cancel: – ok:
Responding to user input	– selectText:
Setting the delegate	– setDelegate: – delegate (NSWindow)

Class Methods

savePanel

+ (NSSavePanel *)**savePanel**

Returns an instance of NSSavePanel, creating one if necessary. Otherwise, the instance is a recycled NSSavePanel object. The method sets the attributes of the instance to the default values**:**

- current working directory as starting point
- prompt of “Name”
- no required file types
- file packages not treated as directories
- no delegate
- no accessory view

Instance Methods

accessoryView

– **(NSView *)accessoryView**

Returns the custom accessory view for the current application. This view is set by **setAccessoryView:**.

See also: – **setAccessoryView:**

cancel:

– **(void)cancel:***(id)sender*

Invoked when the user clicks the panel's Cancel button.

directory

– (NSString *)**directory**

Returns the absolute pathname of the directory currently shown in the panel. Do not invoke this method within a modal session (**runModal** or **runModalForDirectory:file:**) because the directory information is only updated just before the modal session ends.

See also: – **setDirectory:**

encodeWithCoder:

– (void)**encodeWithCoder:**(NSCoder *)*coder*

Overrides the superclass implementation of this NSCoding protocol method to raise an exception. The NSSavePanel does not get encoded.

See also: – **initWithCoder:**

filename

– (NSString *)**filename**

Returns the absolute path name of the file currently shown in the panel. Do not invoke this method within a modal session (**runModal** or **runModalForDirectory:file:**) because the filename information is only updated just before the modal session ends.

initWithCoder:

– (id)**initWithCoder:**(NSCoder *)*coder*

Overrides the superclass implementation of this NSCoding protocol method to raise an exception. The NSSavePanel does not get decoded.

See also: – **encodeWithCoder:**

ok:

– **(void)ok:***(id)sender*

Invoked when the user clicks the panel's OK button.

prompt

– (NSString *)**prompt**

Returns the prompt of the Save panel field that holds the current pathname or file name. By default this prompt is "Name:".

See also: – **setPrompt:**

requiredFileType

– (NSString *)**requiredFileType**

Returns the required file type (if any). A file specified in the Save panel is saved with the designated file name and this file type as an extension. Examples of common file types are "rtf", "tiff", and "ps". An empty NSString return value indicates that the user can save to any ASCII file.

See also: – **setRequiredFileType:**

runModal

– (int)**runModal**

Displays the panel and begins its event loop with the current working (or last selected) directory as the default starting point. Invokes **runModalForDirectory:file:** (file argument is an empty NSString), which in turn performs NSApplication's **runModalForWindow:** method with **self** as the argument. Returns NSOKButton (if the user clicks the OK button) or NSCancelButton (if the user clicks the Cancel button).

Do not invoke **filename** or **directory** within a modal loop because the information that these methods fetch is updated only upon return.

See also: – **runModalForDirectory:file:**, – **runModalForWindow:** (NSApplication)

runModalForDirectory:file:

– (int)**runModalForDirectory:**(NSString *)*path* **file:**(NSString *)*filename*

Initializes the panel to the directory specified by *path* and, optionally, the file specified by *filename*, then displays it and begins its modal event loop; *path* and *filename* can be empty strings, but cannot be **nil**. The method invokes Application's **runModalForWindow:** method with **self** as the argument. Returns NSOKButton (if the user clicks the OK button) or NSCancelButton (if the user clicks the Cancel button). Do not invoke **filename** or **directory** within a modal loop because the information that these methods fetch is updated only upon return.

See also: – **runModal**, – **runModalForWindow:** (Application)

selectText:

– **(void)selectText:***(id)sender*

Advances the current browser selection one line when Tab or the up-arrow key is pressed, and goes back one line when Shift-Tab or the down-arrow key is pressed; after it makes the new selection it writes the selected item in the field after the prompt. The argument *sender* identifies the object invoking this method. This method is primarily of interest to those who want to override it to get different behavior.

setAccessoryView:

– (void)**setAccessoryView:**(NSView *)*aView*

Customizes the panel for the application by adding a custom NSView (aView) to the panel. The custom NSView that's added appears just above the OK and Cancel buttons at the bottom of the panel. The NSSavePanel automatically resizes itself to accommodate *aView*. You can invoke this method repeatedly to change the accessory view as needed. If *aView* is **nil**, the NSSavePanel removes the current accessory view.

See also: – **accessoryView**

setDelegate:

– (void)**setDelegate:**(id)*anObject*

Makes *anObject* the NSSavePanel's delegate, after verifying which delegate methods are implemented. Use NSWindow's **delegate** method to retrieve the NSSavePanel's delegate.

setDirectory:

– (void)**setDirectory:**(NSString *)*path*

Sets the current path name in the Save panel's browser. The *path* argument must be an absolute path name.

See also: – **directory**

setPrompt:

– (void)**setPrompt:**(NSString *)*prompt*

Sets the prompt of the field that holds the current pathname or file name. This prompt appears on all NSSavePanels (or all NSOpenPanels if the receiver of this message is an NSOpenPanel) in your application. "Name:" is the default prompt string.

See also: – **prompt**

setRequiredFileType:

– (void)**setRequiredFileType:**(NSString *)*type*

Specifies the *type*, a file name extension to be appended to any selected files that don't already have that extension; "nib" and "rtf" are examples. The argument *type* should not include the period that begins the extension. You need to invoke this method each time the Save panel is used for another file type within the application.

See also: – **requiredFileType**

setTreatsFilePackagesAsDirectories:

– (void)**setTreatsFilePackagesAsDirectories:**(BOOL)*flag*

Sets the NSSavePanel's behavior for displaying file packages (for example, MyApp.app) to the user. If *flag* is YES, the user is shown files and subdirectories within a file package. If NO, the NSSavePanel shows each file package as a file, thereby giving no indication that it is a directory.

See also: – **treatsFilePackagesAsDirectories**

setTitle:

– (void)**setTitle:**(NSString *)*title*

Sets the title of the NSSavePanel to *title*. By default, “Save” is the title string. If you adapt the NSSavePanel for other uses, its title should reflect the user action that brings it to the screen.

See also: – **title**

treatsFilePackagesAsDirectories

– (BOOL)**treatsFilePackagesAsDirectories**

Use to determine whether the Save panel displays file packages to the user as directories. Returns YES if the user is shown files and subdirectories within a file package; returns NO (the default) if the user is shown only file-package names, with no indication that they are directories.

See also: – **setTreatsFilePackagesAsDirectories:**

validateVisibleColumns

– (void)**validateVisibleColumns**

Validates and possibly reloads the browser columns visible in the Save panel by causing the delegate method **panel:shouldShowFilename:** to be invoked. One situation in which this method would find use is whey you want the browser show only files with certain extensions based on the selection made in an accessory-view pop-up list. When the user changes the selection, you would invoke this method to revalidate the visible columns.

Methods Implemented by the Delegate

panel:compareFilename:with:caseSensitive:

– (NSComparisonResult)**panel:***(id)sender*
compareFilename:(NSString *)*fileName1*
with:(NSString *)*fileName2*
caseSensitive:(BOOL)*flag*

Controls the ordering of files presented by the NSSavePanel. This method should return:

- NSOrderedAscending if *fileName1* should precede *fileName2*
- NSOrderedSame if the two names are equivalent
- NSOrderedDescending if *fileName2* should precede *fileName1*

The *flag* argument, if YES, indicates that the ordering is to be case-sensitive.

Don’t reorder file names in the Save panel without good reason, since it may confuse the user to have files in one Save panel or Open panel ordered differently than those in other such panels or in the Workspace Manager. The default behavior of Save and Open panel is to order files as they appear in the Workspace Manager file viewer. Note also that by implementing this method you will reduce the operating performance of the panel.

panel:shouldShowFilename:

– (BOOL)**panel:(id)***sender*
shouldShowFilename:*(NSString *)filename*

The NSSavePanel sends this message to the panel’s delegate for each file or directory (*filename*) it is about to load in the browser. This method gives the delegate the opportunity to filter out items that it doesn’t want the user to see or choose. The delegate returns YES if *filename* should be displayed, and NO if the NSSavePanel should ignore the file or directory.

panel:isValidFilename:

– (BOOL)**panel:(id)*sender***
isValidFilename:*(NSString *)filename*

The NSSavePanel sends this message just before the end of a modal session for each file name displayed or selected (including file names in multiple selections). The delegate determines whether it wants the file identified by *filename*; it returns YES if the file name is valid, or NO if the NSSavePanel should stay in its modal loop and wait for the user to type in or select a different file name or names. If the delegate refuses a file name in a multiple selection, none of the file names in the selection are accepted.