Mike Ferris - February 16th, 1994

MiscNibController

Inherits From: Object

Declared In: misckit/MiscNibController.h

Class Description

MiscNibController instances manage a window which is stored in an InterfaceBuilder .nib file. MiscNibController handles both loading the nib file and keeping track of the window. MiscNibController acts as the nib "File's Owner". Automatic support for saving the window's frame (ie. size and location) in the defaults database through the standard Window methods is provided. The **-showWindow:**;MiscNibController.rtf;showWindow:;¬ method is used to display the window (loading the nib file first if necessary).

MiscNibController is an abstract superclass. Most subclasses of MiscNibController will only have one instance in your application. For example, you might create subclasses called PreferencesController, FindController, PaletteController, etc, and then create one instance of each for your application to manage those particular panels. Other subclasses (such as MiscDocController) are meant to have multiple instances, but this is less common, and if you find yourself wanting to do this, you should look closely at whether you shouldn't instead be subclassing MiscDocController.

Each subclass of MiscNibController should register a nib name for that class. This is done with the **+setClassNib:**;MiscNibController.rtf;setClassNib:;¬ method, and you should do it in your subclass's **+initialize**;MiscNibController.rtf;initialize;¬ method. MiscNibController uses a MiscClassVariable object to store these nib names so you needn't worry about reimplementing the class variable in your subclasses (like you'd have to do with NeXT's Control class). If you do not use **+setClassNib:**;MiscNibController.rtf;setClassNib:;¬ to set the name of the .nib file to load, then MiscNibController will load the .nib that matches the class name. For example, a subclasses called ªPreferencesº will, by default, load a .nib named ªPreferences.nibº.

It is possible to have a MiscNibController without a nib, although almost all of what it provides has to do with nib support, so you'd have to have a reason for this. (Two reasons might be if you want to keep a consistent interface between several objects, some of which have nib-file windows, or if you want to make a subclass of MiscDocController which supports a multiple-views-on-a-document type interface but without a top-level document window.)

When you create an instance, you can assign it a frame name to use to save the window's frame information in the defaults database. Passing NULL means the frame information is not restored from the defaults database, it is left as it is in the nib file. Specifying a name does two things. When the nib is loaded, the window's frame is set to the value in the defaults database corresponding to the frame name. If there isn't a defaults entry, the

frame in the nib file is used. Also, having a frame name causes the window to save its frame info into the defaults database any time the window changes size or location.

Instance Variables

```
id          window;
MiscString  *frameName;
BOOL        nibIsLoaded;
```

window	The window which the controller controls. This should almost always be set up as an outlet connection in the nib file.
frameName	frameName always points at a valid MiscString object. The MiscString contains the window's frame name. It contains NULL or a zero-length string if the window has no frame name.
nibIsLoaded	nibIsLoaded indicates whether the controller's nib file has been loaded (for this instance).

Method Types

Initializing the class	+ initialize;MiscNibController.rtf;initialize;¬ + startUnloading;¬;MiscNibController.rtf;startUnloading;¬
Class nib file	+ setClassNib:;MiscNibController.rtf;setClassNib:;¬ + classNib;MiscNibController.rtf;classNib;¬
Initializing instances	- init;MiscNibController.rtf;init;¬ - initWithFrameName:;MiscNibController.rtf;initWithFrameName:;¬ - free;MiscNibController.rtf;free;¬
The frame name	- setFrameName:;MiscNibController.rtf;setFrameName:;¬ - frameName;MiscNibController.rtf;frameName;¬
Loading the nib	- loadNib;MiscNibController.rtf;loadNib;¬ - loadNib:;MiscNibController.rtf;loadNib:;¬ - loadNib:fromBundle:;MiscNibController.rtf;loadNib:fromBundle:;¬ - loadNibIfNeeded;MiscNibController.rtf;loadNibIfNeeded;¬ - loadNib:withOwner:;MiscNibController.rtf;loadNib:withOwner:;¬ - loadNib:withOwner:fromBundle:;MiscNibController.rtf;loadNib:withOwner:fromBundle:;¬ - nibDidLoad;MiscNibController.rtf;nibDidLoad;¬ - nibWillLoad;MiscNibController.rtf;nibWillLoad;¬
Dealing with the window	- window;MiscNibController.rtf;window;¬

- window:;MiscNibController.rtf;window:;¬
- showWindow:;MiscNibController.rtf;showWindow:;¬

Archiving

- awake;MiscNibController.rtf;awake;¬
- read:;MiscNibController.rtf;read:;¬
- write:;MiscNibController.rtf;write:;¬

Class Methods

initialize;¬initialize

+ **initialize**

Sets the class version number. Loads the MiscString and MiscClassVariable classes if necessary. Initializes the nib name class variable. Subclasses should override this method and call **+setClassNib:**;MiscNibController.rtf;setClassNib:;¬ to set up the name of the .nib file handled by the particular subclass in question.

See also: + **startUnloading**;MiscNibController.rtf;startUnloading;¬

classNib;¬classNib

+ (const char *)**classNib**

Returns the nib file name for the calling subclass. This is done by looking up the value in the MiscClassVariable object used to store the nib names.

See also: + **setClassNib:**;MiscNibController.rtf;setClassNib:;¬

setClassNib:;¬setClassNib:

+ **setClassNib:**(const char *)*nibName*

Sets the nib file name for the calling subclass. Nib names are stored in a MiscClassVariable, so each subclass (and all their subclasses, etc...) get different values.

See also: + **classNib**;MiscNibController.rtf;classNib;¬

startUnloading;¬startUnloading

+ **startUnloading**

Frees the nib name class variable.

See also: + **initialize**;MiscNibController.rtf;initialize;¬

Instance Methods

awake;¬awake
± **awake**

Initializes the instance variables that are not archived after an instance has been read from a typed stream. Basically this just makes sure the instance looks like it hasn't had its nib loaded.

See also: ± **read:**, ± **write:**

frameName;¬frameName
± (const char *)**frameName**

Returns the frame name used to save the window's frame information in the defaults database. Returns NULL if the frame information is not stored in the defaults database.

See also: ± **setFrameName:**

free;¬free
± **free**

This method frees the frameName and the window. If your subclass uses other windows (including popup menus), you should override this and free them. Only the window pointed at by the window instance variable is

freed.

See also: ± **init**, ± **initWithFrameName:**;MiscNibController.rtf;initWithFrameName:;¬

init;¬init
± **init**

Calls **±initWithFrameName:**;MiscNibController.rtf;initWithFrameName:;¬ using NULL as the argument.

See also: ± **initWithFrameName:**;MiscNibController.rtf;initWithFrameName:;¬, ± **free**

initWithFrameName:;¬initWithFrameName:
± **initWithFrameName:**(const char *)*name*

This is the designated initializer for the class. This allocates and initializes the frameName MiscString and initializes the other instance variables. The nib file is not loaded until it is needed (that is, until **-showWindow:** or **-window:**YES are called).

See also: ± **init**, ± **free**

loadNib;¬loadNib
± **loadNib**

Cover method which calls `[self loadNib:NULL withOwner:nil fromBundle:nil]`.

See also: + **setClassNib:**, ± **nibDidLoad** and ± **nibWillLoad**

loadNib:;¬loadNib:
± **loadNib:**(const char *)*name*

Cover method which calls `[self loadNib:name withOwner:nil fromBundle:nil]`.

See also: + **setClassNib:**, ± **nibDidLoad** and ± **nibWillLoad**

loadNib:fromBundle:;¬loadNib:fromBundle:
± **loadNib:**(const char *)*name* **fromBundle:***bundle*

Cover method which calls `[self loadNib:name withOwner:nil fromBundle:bundle]`.

See also: + **setClassNib:**, ± **nibDidLoad** and ± **nibWillLoad**

loadNibIfNeeded;¬loadNibIfNeeded
± **loadNibIfNeeded**

Loads the controller's nib file, if it has not already been loaded. If the nib file appears to exist, then ±**nibWillLoad** is called; if you need to do any prepatory initialization, that method should be overridden to do it. If the nib file actually was loaded successfully, the window's frame is set if the frameName of this instance is non-NULL and the name is registered as the window's autosave frame name. Finally, if the nib was actually loaded, the **-nibDidLoad** method is called. The method ±**nibDidLoad** should be overridden instead of **-loadNibIfNeeded** if you have further initialization to do after the nib file loads. By default, the name of the nib file that is loaded is the name of the class with ªº appended. For example, ªMiscNibController.nibº would be the nib file loaded by this class. Use **+setClassNib:** to change the name of the .nib that will be loaded.

See also: + **setClassNib:**, ± **nibDidLoad** and ± **nibWillLoad**

loadNib:withOwner:;¬loadNib:withOwner:

± **loadNib:**(const char *)*name* **withOwner:***owner*

Cover method which calls `[self loadNib:name withOwner:owner fromBundle:nil]`.

See also: + **setClassNib:**, ± **nibDidLoad** and ± **nibWillLoad**

loadNib:withOwner:fromBundle:;¬loadNib:withOwner:fromBundle:

± **loadNib:**(const char *)*name* **withOwner:***owner* **fromBundle:***bundle*

Loads a nib file named ª*name*.nibº from inside the NXBundle *bundle* with *owner* as the file's owner . If *name* is

NULL then the instance's name (for a named object) is used; if that name is NULL, then the name of the class is used. If *owner* is nil, then *self* is used. If *bundle* is nil, then the receiver's class bundle is used. This method is used by ±**loadNibIfNeeded** to actually load the nib file. This method does *not* call the ±**nibDidLoad** or ±**nibWillLoad** methods. Normally this method wouldn't be used directly; ±**loadNibIfNeeded** would be used instead.

See also: ± **loadNibIfNeeded**

nibDidLoad;¬nibDidLoad
± **nibDidLoad**

This is called by **-loadNibIfNeeded** after the nib file is loaded and is intended to be overridden. Although the current implementation in the MiscNibController class does nothing, be sure to call super's implementation first if you override this method.

See also: ± **loadNib...** methods and ± **nibWillLoad**

nibWillLoad;¬nibWillLoad
± **nibDidLoad**

This is called by **-loadNibIfNeeded** before the nib file is loaded and is intended to be overridden. Although the current implementation in the MiscNibController class does nothing, be sure to call super's implementation first

if you override this method.

See also: ± **loadNib...** methods and ± **nibDidLoad**

read:;¬read:
± **read:**(NXTypedStream *)*stream*

Reads the instance from the typed stream. Only the frame name is archived.

See also: ± **awake**, ± **write:**

setFrameName:;¬ setFrameName:
± **setFrameName:**(const char *)*name*

Sets the window's frame name. This will only work if the nib file hasn't been loaded yet. Normally, the frame name is set with the **-initFrameName:** method.

See also: ± **frameName**

showWindow:;¬showWindow:
± **showWindow:***sender*

If the nib file is not loaded, this method loads it. Then it puts the window on-screen and makes it key.

See also: ± **window**, ± **window:**

window;¬window
± **window**

This just calls **±window:** with NO as the argument.

See also: ± **window:**, ± **showWindow:**

window:;¬window:
± **window:**(BOOL)*loadFlag*

Returns the controller's window. If the nib file is not loaded and loadFlag is YES, the nib file is loaded first. If the nib file is not loaded and loadFlag is NO, nil is returned.

See also: ± **window**, ± **showWindow:**

write:;¬write:
± **write:**(NXTypedStream *)*stream*

Writes the instance to the typed stream. Only the frame name is archived.

See also: ± **awake**, ± **read:**