NSNib Class Reference

Cocoa > Resource Management

2007-01-22

Contents

NSNib Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Resource Programming Guide
Declared in	NSNib.h
Related sample code	Aperture Edit Plugin - Borders & Titles Aperture Image Resizer CoreRecipes Departments and Employees

Overview

Instances of the `NSNib` class serve as object wrappers, or containers, for Interface Builder nib files. An `NSNib` object keeps the contents of a nib file resident in memory, ready for unarchiving and instantiation.

When you create an `NSNib` object using the contents of a nib file, the object loads the contents of the referenced nib bundle—the object graph as well as any images and sounds—into memory; but it does not yet unarchive it. To unarchive all of the nib data and thus truly instantiate the nib you must call one of the `instantiate...` methods of `NSNib`.

During the instantiation process, each object in the archive is unarchived and then initialized using the method befitting its type. View classes are initialized using their `initWithFrame:` method. Custom objects are initialized using their `init` method. In the case of Cocoa views (and custom views that have options on an associated Interface Builder palette) the initialization process also reads in any values set by the user in Interface Builder.

Once all objects have been instantiated and initialized from the archive, the nib loading code attempts to reestablish the connections between each object's outlets and the corresponding target objects. If your custom objects have outlets, the `NSNib` object attempts to reestablish any connections you created in Interface Builder. It starts by trying to establish the connections using your object's own methods first. For each outlet that needs a connection, the `NSNib` object looks for a method of the form `set`*OutletName*`:` in your object. If that method exists, the nib object calls it, passing the target object as a parameter. If you did not define a setter method with that exact name, the `NSNib` object searches the object for an instance variable (of type `IBOutlet id`) with the corresponding outlet name and tries to set its value directly. If an instance variable with the correct name cannot be found, initialization of that connection does not occur.

After all objects have been initialized and their connections reestablished, each object receives an `awakeFromNib` message. You can override this method in your custom objects to perform any additional initialization.

Subclassing Notes

You can subclass `NSNib` if you want to extend or specialize nib-loading behavior. For example, you could create a custom `NSNib` subclass that performs some post-processing on the top-level objects returned from the `instantiateNib...` methods. If you want to modify how nib instantiations are performed, it is recommended that you override the primitive method `instantiateNibWithExternalNameTable:` (page 8). Note that the instance variables of `NSNib` are private and thus are not available to subclasses. Any override of `initWithContentsOfURL:` (page 7) or `initWithNibNamed:bundle:` (page 7) should first invoke the superclass implementation.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Initializing a Nib

- `initWithContentsOfURL:` (page 7)
  Returns an `NSNib` object initialized to the nib file at the specified URL.
- `initWithNibNamed:bundle:` (page 7)
  Returns an `NSNib` object initialized to the nib file in the specified bundle.

Instantiating a Nib

- `instantiateNibWithOwner:topLevelObjects:` (page 8)
  Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and set of top level objects.
- `instantiateNibWithExternalNameTable:` (page 8)
  Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and top level objects.

Instance Methods

initWithContentsOfURL:

Returns an `NSNib` object initialized to the nib file at the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)nibFileURL
```

Parameters

nibFileURL

The location of the nib file.

Return Value

The initialized `NSNib` object or `nil` if there were errors during initialization or the nib file could not be located.

Discussion

When you instantiate the nib objects later, the `NSNib` object looks for an appropriate bundle from which to search for any additional resources referenced by the nib. Because you do not specify a bundle directory when calling this method, the receiver uses the bundle associated with the class of the nib file's owner. If the nib file does not have an owner, the receiver uses the application's main bundle instead.

Availability

Available in Mac OS X v10.3 and later.

See Also

`NSURL` class (Foundation)

Declared In

`NSNib.h`

initWithNibNamed:bundle:

Returns an `NSNib` object initialized to the nib file in the specified bundle.

```
- (id)initWithNibNamed:(NSString *)nibName bundle:(NSBundle *)bundle
```

Parameters

nibName

The name of the nib file, without any leading path information. Inclusion of the `.nib` extension on the nib file name is optional.

bundle

The bundle in which to search for the nib file. If you specify `nil`, this method looks for the nib file in the main bundle.

Return Value

The initialized `NSNib` object or `nil` if there were errors during initialization or the nib file could not be located.

Discussion

The `NSNib` object looks for the nib file in the bundle's language-specific project directories first, followed by the `Resources` directory.

After the nib file has been loaded, the `NSNib` object uses the bundle's resource map to locate additional resources referenced by the nib. If you specified `nil` for the *`bundle`* parameter, the `NSNib` object looks for those resources in the bundle associated with the class of the nib file's owner instead. If the nib file does not have an owner, the `NSNib` object looks for additional resources in the application's main bundle.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Aperture Image Resizer
CoreRecipes
Departments and Employees

Declared In
`NSNib.h`

instantiateNibWithExternalNameTable:

Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and top level objects.

```
- (BOOL)instantiateNibWithExternalNameTable:(NSDictionary *)externalNameTable
```

Parameters

`externalNameTable`
A dictionary containing entries for the nib file's owner and top-level objects. See the discussion for more information.

Return Value
`YES` if the nib file's contents were instantiated successfully; otherwise, `NO`.

Discussion
This is the primitive method for performing instantiations of a nib file. You may use this method to instantiate a nib file multiple times. Each instantiation of the nib must have a distinct owner object that is responsible for the resulting object tree.

If the nib file requires an owner, the *`externalNameTable`* parameter must contain the object representing the nib file's owner (associated with the `NSNibOwner` key). The parameter may optionally include an `NSMutableArray` object to be populated with the top-level objects nib file (associated with the `NSNibTopLevelObjects` key).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSNib.h`

instantiateNibWithOwner:topLevelObjects:

Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and set of top level objects.

```
- (BOOL)instantiateNibWithOwner:(id)owner topLevelObjects:(NSArray **)topLevelObjects
```

Parameters

owner
: The object to use as the owner of the nib file. If the nib file has an owner, you must specify a valid object for this parameter.

topLevelObjects
: On input, a variable capable of holding an `NSArray` object. On output, this variable contains an autoreleased `NSArray` object containing the top-level objects from the nib file. You may specify `nil` for this parameter if you are not interested in the top-level objects.

Return Value

`YES` if the nib file's contents were instantiated successfully; otherwise, `NO`.

Discussion

You may use this method to instantiate a nib file multiple times. This is a convenience method that composes the name-table dictionary and invokes the `instantiateNibWithExternalNameTable:` (page 8) method, passing it the name table.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CoreRecipes

Departments and Employees

Declared In

`NSNib.h`

Constants

Nib Loading Keys

The `NSNib` class uses the following constants which are used as keys in the dictionary passed to `instantiateNibWithExternalNameTable:` (page 8).

```
NSString *NSNibOwner;
NSString *NSNibTopLevelObjects;
```

Constants

`NSNibOwner`
: The external object that is responsible for the instantiated nib.

  This key is required.

  Available in Mac OS X v10.3 and later.

  Declared in `NSNib.h`.

`NSNibTopLevelObjects`
: An `NSMutableArray` object that, if present, is populated with the top-level objects of the newly instantiated nib.

  Because you must allocate this array, you are responsible for its disposal. This key is optional.

  Available in Mac OS X v10.3 and later.

  Declared in `NSNib.h`.

Declared In

`NSNib.h`

Document Revision History

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

Date	Notes
2007-01-22	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Index