# NSViewController Class Reference

**Cocoa > Graphics & Imaging**

**2007-05-02**

Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents


## NSViewController Class Reference 5

Overview 5
Tasks 6
- Creating A View Controller 6
- Represented Object 6
- Nib Properties 6
- View Properties 6
- NSEditor Conformance 7

Instance Methods 7
- commitEditing 7
- commitEditingWithDelegate:didCommitSelector:contextInfo: 7
- discardEditing 8
- initWithNibName:bundle: 8
- loadView 9
- nibBundle 10
- nibName 10
- representedObject 10
- setRepresentedObject: 11
- setTitle: 11
- setView: 12
- title 12
- view 13

## Document Revision History 15

## Index 17




2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

# NSViewController Class Reference

| | |
|---|---|
| **Inherits from** | NSResponder : NSObject |
| **Conforms to** | NSCoding (NSResponder)<br>NSObject (NSObject)<br>NSEditor (Informal Protocol)<br>NSEditorRegistration (Informal Protocol) |
| **Framework** | /System/Library/Frameworks/AppKit.framework |
| **Availability** | Available in Mac OS X v10.5 and later. |
| **Declared in** | NSViewController.h |
| **Related sample code** | CrossEvents<br>HIView-NSView<br>ViewController |

## Overview

An `NSViewController` object manages a view, typically loaded from a nib file.

This management includes:

- Memory management of top-level objects similar to that of the `NSWindowController` class, taking the same care to prevent reference cycles when controls are bound to the nib file's owner that `NSWindowController` began taking in Mac OS v 10.4.
- Declaring a generic `representedObject` (page 10) property, to make it easy to establish bindings in the nib to an object that isn't yet known at nib-loading time or readily available to the code that's doing the nib loading.
- Implementing the key-value binding `NSEditor` informal protocol, so that applications using `NSViewController` can easily make bound controls in the views commit or discard the changes the user is making.

`NSViewController` is meant to be highly reusable. For example, the `NSPageLayout` and `NSPrintPanel` `addAccessoryController:` methods take an `NSViewController` instance as the argument, and set the `representedObject` (page 10) to the `NSPrintInfo` that is to be shown to the user. This allows a developer to easily create new printing accessory views using bindings and the `NSPrintInfo` class' new key-value coding and key-value observing compliance. When the user dismisses a printing panel, `NSPageLayout` and `NSPrintPanel` each send `NSEditor` messages to each accessory view controller to ensure that the user's changes have been committed or discarded properly. The titles of the accessories are retrieved from the


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

view controllers and shown to the user in pulldown menus that the user can choose from. If your application needs to dynamically associate relatively complex views with the objects that they present to the user, you might be able to reuse `NSViewController` in similar ways.

# Tasks

## Creating A View Controller

- `initWithNibName:bundle:` (page 8)
  Returns an `NSViewController` object initialized to the nib file in the specified bundle.
- `loadView` (page 9)
  Instantiate the receiver's view and set it.

## Represented Object

- `setRepresentedObject:` (page 11)
  Sets the object whose value is being presented in the receiver's view.
- `representedObject` (page 10)
  Returns the object whose value is being presented in the receiver's view.

## Nib Properties

- `nibBundle` (page 10)
  Return the name of the nib bundle to be loaded to instantiate the receivers view.
- `nibName` (page 10)
  Return the name of the nib to be loaded to instantiate the receivers view

## View Properties

- `view` (page 13)
  Returns the receiver's view.
- `setView:` (page 12)
  Sets the receivers view to the specified object.
- `title` (page 12)
  Returns the localized title of the receiver's view.
- `setTitle:` (page 11)
  Sets the localized title of the receiver's view to the specified string.


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

### NSEditor Conformance

- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 7)
  Attempt to commit any currently edited results of the receiver.
- `commitEditing` (page 7)
  Returns whether the receiver was able to commit any pending edits.
- `discardEditing` (page 8)
  Causes the receiver to discard any changes, restoring the previous values.

## Instance Methods

### commitEditing

Returns whether the receiver was able to commit any pending edits.

```
- (BOOL)commitEditing
```

**Return Value**
Returns `YES` if the changes were successfully applied to the model, `NO` otherwise.

**Discussion**
A commit is denied if the receiver fails to apply the changes to the model object, perhaps due to a validation error.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 7)
- `discardEditing` (page 8)

**Declared In**
`NSViewController.h`

### commitEditingWithDelegate:didCommitSelector:contextInfo:

Attempt to commit any currently edited results of the receiver.

```
- (void)commitEditingWithDelegate:(id)delegate
    didCommitSelector:(SEL)didCommitSelector contextInfo:(void *)contextInfo
```

**Parameters**
*`delegate`*
An object that can serve as the receiver's delegate. It should implement the method specified by *`didCommitSelector`*.

*`didCommitSelector`*
A selector that is invoked on delegate.


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

*contextInfo*

Contextual information that is sent as the `contextInfo` argument to delegate when *didCommitSelector* is invoked.

**Discussion**
The receiver must have been registered as the editor of an object using `objectDidBeginEditing:`, and has not yet been unregistered by a subsequent invocation of `objectDidEndEditing:`. When the committing has either succeeded or failed, send the *delegate* the message specified by *didCommitSelector*.

The *didCommitSelector* method must have the following method signature:.

```
- (void)editor:(id)editor didCommit:(BOOL)didCommit contextInfo:(void
*)contextInfo
```

If an error occurs while attempting to commit, for example if key-value coding validation fails, an implementation of this method should typically send the receiver's view a`presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` message, specifying the view's containing window.

You may find this method useful in some situations when you want to ensure that pending changes are applied before a change in user interface state. For example, you may need to ensure that changes pending in a text field are applied before a window is closed. See also `commitEditing` (page 7) which performs a similar function but which allows you to handle any errors directly, although it provides no information beyond simple success/failure.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `commitEditing` (page 7)
- `discardEditing` (page 8)

**Declared In**
`NSViewController.h`

## discardEditing

Causes the receiver to discard any changes, restoring the previous values.

```
- (void)discardEditing
```

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `commitEditing` (page 7)

**Declared In**
`NSViewController.h`

## initWithNibName:bundle:

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


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

```
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
```

**Parameters**

*nibNameOrNil*

The name of the nib file, without any leading path information.

*nibBundleOrNil*

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

**Discussion**

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

The specified nib should typically have the class of the file's owner set to `NSViewController`, or a custom subclass, with the `view` outlet connected to a view.

If you pass in a `nil` for *nibNameOrNil* then `nibName` (page 10) will return `nil` and `loadView` (page 9) will throw an exception; in this case you must invoke `setView:` (page 12) before `view` (page 13) is invoked, or override `loadView` (page 9).

**Availability**

Available in Mac OS X v10.5 and later.

**Related Sample Code**

ViewController

**Declared In**

`NSViewController.h`

## loadView

Instantiate the receiver's view and set it.

```
- (void)loadView
```

**Discussion**

The default implementation of this method invokes `nibName` (page 10) and `nibBundle` (page 10) and then uses the `NSNib` class to load the nib with the receiver as the file's owner. If the `view` outlet of the file's owner in the nib is properly connected, the regular nib loading machinery will send the receiver a `setView:` (page 12) message.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`NSViewController.h`


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

## nibBundle

Return the name of the nib bundle to be loaded to instantiate the receivers view.

```
- (NSBundle *)nibBundle
```

**Return Value**
The name of the nib bundle.

**Discussion**
The default implementation returns whatever value was passed to the initializer.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `initWithNibName:bundle:` (page 8)
- `nibName` (page 10)

**Declared In**
`NSViewController.h`

## nibName

Return the name of the nib to be loaded to instantiate the receivers view

```
- (NSString *)nibName
```

**Return Value**
The name of the nib.

**Discussion**
The default implementation returns whatever value was passed to the initializer.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `initWithNibName:bundle:` (page 8)
- `nibBundle` (page 10)

**Declared In**
`NSViewController.h`

## representedObject

Returns the object whose value is being presented in the receiver's view.

```
- (id)representedObject
```

**Return Value**
The object whose value is presented in the receiver's view.


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

**Discussion**
This class is key-value coding and key-value observing compliant for `representedObject` (page 10) so when you use it as the file's owner of a view's nib you can bind controls to the file's owner using key paths that start with `representedObject`.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`- setRepresentedObject:` (page 11)

**Declared In**
`NSViewController.h`

## setRepresentedObject:

Sets the object whose value is being presented in the receiver's view.

```
- (void)setRepresentedObject:(id)representedObject
```

**Parameters**
*representedObject*
The object whose value is presented in the receiver's view.

**Discussion**
The default implementation of this method doesn't copy the passed-in object, it retains it. In another words, *representedObject* is a to-one relationship, not an attribute.

This class is key-value coding and key-value observing compliant for `representedObject` (page 10) so when you use it as the file's owner of a view's nib you can bind controls to the file's owner using key paths that start with `representedObject`.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`- representedObject` (page 10)

**Declared In**
`NSViewController.h`

## setTitle:

Sets the localized title of the receiver's view to the specified string.

```
- (void)setTitle:(NSString *)title
```

**Parameters**
*title*
The localized title of the receiver view.


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

**Discussion**
`NSViewController` does not use the title property directly. This property is here because so many anticipated uses of this class will involve letting the user choose among multiple named views using a pulldown menu or some other user interface.

This class is key value coding and key value compliant for this property.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`- title` (page 12)

**Declared In**
`NSViewController.h`

## setView:

Sets the receivers view to the specified object.

```
- (void)setView:(NSView *)view
```

**Parameters**

*view*

The view the receiver should manage.

**Discussion**
You can invoke this method immediately after creating the object to specify a view that's created in a different manner than the receiver's default implementation would provide.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`- view` (page 13)

**Declared In**
`NSViewController.h`

## title

Returns the localized title of the receiver's view.

```
- (NSString *)title
```

**Return Value**
The localized title of the receiver's view

**Discussion**
`NSViewController` does not use the title property directly. This property is here because so many anticipated uses of this class will involve letting the user choose among multiple named views using a pulldown menu or some other user interface.

This class is key value coding and key value compliant for this property.


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `setTitle:` (page 11)

**Declared In**
`NSViewController.h`

## view

Returns the receiver's view.

```
- (NSView *)view
```

**Return Value**
The receiver's view object.

**Discussion**
The default implementation of this method first invokes `loadView` (page 9) if the view hasn't been set yet.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
- `setView:` (page 12)
- `loadView` (page 9)

**Declared In**
`NSViewController.h`


2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

# Document Revision History

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

| Date | Notes |
| --- | --- |
| 2007-05-02 | New document that describes the class used to manage a view, usually a view stored in a nib file. |



2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

Document Revision History



2007-05-02 | © 2007 Apple Inc. All Rights Reserved.

# Index


## C

`commitEditing` instance method 7
`commitEditingWithDelegate:didCommitSelector:contextInfo:` instance method 7

## D

`discardEditing` instance method 8

## I

`initWithNibName:bundle:` instance method 8

## L

`loadView` instance method 9

## N

`nibBundle` instance method 10
`nibName` instance method 10

## R

`representedObject` instance method 10

## S

`setRepresentedObject:` instance method 11
`setTitle:` instance method 11
`setView:` instance method 12

## T

`title` instance method 12

## V

`view` instance method 13




2007-05-02 | © 2007 Apple Inc. All Rights Reserved.