Class

NSViewController

A controller that manages a view, typically loaded from a nib file.

macOS 10.5+

```
@MainActor
class NSViewController
```

Overview

View controller management includes:

- Memory management of top-level objects similar to that performed by the `NSWindowController` class, taking the same care to prevent reference cycles when controls are bound to the nib file's owner.
- Declaring a generic `view` 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 apps using a view controller can easily make bound controls in the views commit or discard changes by the user.

In macOS 10.10 and later, a view controller offers a full set of life cycle methods, allowing you to manage the content of a window in a way that is on a par with iOS view controller management. These methods, presented in order here to reflect a typical cycle, are:

View life cycle:

1. `viewDidLoad()`
2. `viewWillAppear()`
3. `viewDidAppear()`

User interaction cycle:

1. `updateViewConstraints()`
2. `viewWillLayout()`
3. `viewDidLayout()`
4. `viewWillDisappear()`
5. `viewDidDisappear()`

In addition, in macOS 10.10 and later, a view controller participates in the responder chain. You can implement action methods directly in the view controller. Corresponding actions that originate in the view controller's view proceed up the responder chain and are handled by those methods.

Prior to OS X v10.10, a typical usage pattern for loading a nib file was to subclass `NSViewController` and override its `loadView()` method to call `[super loadView]`. But in macOS 10.10 and later, the `loadView()` method automatically looks for a nib file with the same name as the view controller. To take advantage of this behavior, name a nib file after its corresponding view controller and pass `nil` to both parameters of the `init(nibName:bundle:)` method.

A view controller employs lazy loading of its view: Immediately after a view controller is loaded into memory, the value of its `isViewLoaded` property is `false`. The value changes to `true` after the `loadView()` method returns and just before the system calls the `viewDidLoad()` method.

A view controller is meant to be highly reusable, such as for dynamically representing various objects. For example, the `addAccessoryController(_:)` methods of the `NSPageLayout` and `NSPrintPanel` classes take an `NSViewController` instance as the argument, and set the `representedObject` property to the `NSPrintInfo` object 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's key-value coding and key-value observing compliance. When the user dismisses a printing dialog, the `NSPageLayout` and `NSPrintPanel` classes 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 view controllers and shown to the user in menus that the user can choose from.

Topics

Creating A View Controller

`init(nibName: NSNib.Name?, bundle: Bundle?)`

Returns a view controller object initialized to the nib file in the specified bundle.

`func loadView()`

Instantiates a view from a nib file and sets the value of the `view` property.

Represented Object

```
var representedObject: Any?
```

The object whose value is presented in the receiver's primary view.

Nib Properties

```
var nibBundle: Bundle?
```

The nib bundle to be loaded to instantiate the receiver's primary view.

```
var nibName: NSNib.Name?
```

The name of the nib file to be loaded to instantiate the receiver's primary view.

View Properties

```
var view: NSView
```

The view controller's primary view.

```
var title: String?
```

The localized title of the receiver's primary view.

View Property Wrappers

```
struct ViewLoading
```

A property wrapper that loads the view controller's view before accessing the property.

NSEditor Conformance

```
func commitEditing(withDelegate: Any?, didCommit: Selector?, context
Info: UnsafeMutableRawPointer?)
```

Attempt to commit any currently edited results of the receiver.

```
func commitEditing() -> Bool
```

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

```
func discardEditing()
```

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

Using a Storyboard

`var storyboard: NSStoryboard?`

The storyboard from which the view controller was loaded.

`func dismiss(Any?)`

Responding to View Events

`func viewDidLoad()`

Called after the view controller's view has been loaded into memory.

`func loadViewIfNeeded()`

`var isViewLoaded: Bool`

A Boolean value indicating whether the view controller's view is loaded into memory.

`var viewIfLoaded: NSView?`

`func viewWillAppear()`

Called after the view controller's view has been loaded into memory is about to be added to the view hierarchy in the window.

`func viewDidAppear()`

Called when the view controller's view is fully transitioned onto the screen.

`func viewWillDisappear()`

Called when the view controller's view is about to be removed from the view hierarchy in the window.

`func viewDidDisappear()`

Called after the view controller's view is removed from the view hierarchy in a window.

Managing View Layout

`var preferredContentSize: NSSize`

The desired size of the view controller's view, in screen units.

`func updateViewConstraints()`

Called during Auto Layout constraint updating to enable the view controller to mediate the process.

`func viewWillLayout()`

Called just before the `layout()` method of the view controller's view is called.

`func viewDidLayout()`

Called immediately after the `layout()` method of the view controller's view is called.

Managing Child View Controllers in a Custom Container

`func addChild(NSViewController)`

A convenience method for adding a child view controller at the end of the `children` array.

`var children: [NSViewController]`

An array of view controllers that are hierarchical children of the view controller.

`func transition(from: NSViewController, to: NSViewController, options: NSViewController.TransitionOptions, completionHandler: (() -> Void)?)`

Performs a transition between two sibling child view controllers of the view controller.

`func insertChild(NSViewController, at: Int)`

Inserts a specified child view controller into the `children` array at a specified position.

`func removeChild(at: Int)`

Removes a specified child controller from the view controller.

`func removeFromParent()`

Removes the called view controller from its parent view controller.

`func preferredContentSizeDidChange(for: NSViewController)`

Called when there is a change in value of the `preferredContentSize` property of a child view controller or a presented view controller.

Presenting Another View Controller's Content

`func present(NSViewController, animator: any NSViewControllerPresentationAnimator)`

Presents another view controller using a specified, custom animator for presentation and dismissal.

`func dismiss(NSViewController)`

Dismisses a presented view controller, using the same animator that presented it.

`func present(NSViewController, asPopoverRelativeTo: NSRect, of: NSView, preferredEdge: NSRectEdge, behavior: NSPopover.Behavior)`

Presents another view controller as a popover.

`func present(NSViewController, asPopoverRelativeTo: NSRect, of: NSView, preferredEdge: NSRectEdge, behavior: NSPopover.Behavior, hasFullSizeContent: Bool)`

`func presentAsModalWindow(NSViewController)`

Presents another view controller as a modal window, also known as an alert.

`func presentAsSheet(NSViewController)`

Presents another view controller as a sheet.

~~`func present(inWidget: NSViewController)`~~ Deprecated

Getting Related View Controllers

`var parent: NSViewController?`

The immediate ancestor view controller of the view controller.

`var presentedViewControllers: [NSViewController]?`

The view controllers, if any, that are currently presented by the view controller.

`var presentingViewController: NSViewController?`

The view controller that presented the view controller or that presented its farthest ancestor view controller.

Configuring an App Extension View Controller

`var extensionContext: NSExtensionContext?`

For a view controller that is part of an app extension, the app extension context.

`var preferredScreenOrigin: NSPoint`

For a view controller that is part of an app extension, the preferred screen origin.

`var preferredMaximumSize: NSSize`

For a view controller that is part of an app extension, the largest allowable size for the app extension's primary view, in screen units.

`var preferredMinimumSize: NSSize`

For a view controller that is part of an app extension, the smallest allowable size for the app extension's primary view, in screen units.

`func viewWillTransition(to: NSSize)`

For a view controller that is part of an app extension, called when its view is about to be resized.

`var sourceItemView: NSView?`

Constants

`struct TransitionOptions`

Animation options for view transitions in a view controller.

Initializers

`init?(coder: NSCoder)`

Relationships

Inherits From

`NSResponder`

Inherited By

`NSCollectionViewItem`
`NSPageController`
`NSSplitViewController`
`NSSplitViewItemAccessoryViewController`
`NSTabViewController`
`NSTitlebarAccessoryViewController`

Conforms To

`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`

Equatable
Hashable
NSCoding
NSEditor
NSExtensionRequestHandling
NSObjectProtocol
NSSeguePerforming
NSStandardKeyBindingResponding
NSTouchBarProvider
NSUserActivityRestoring
NSUserInterfaceItemIdentification
PlaygroundLiveViewable
Sendable
SendableMetatype

See Also

Content Controllers

`class` NSWindowController

A controller that manages a window, usually a window stored in a nib file.

`class` NSTitlebarAccessoryViewController

An object that manages a custom view—known as an accessory view—in the title bar–toolbar area of a window.