

---

[AppKit](#) / NSSplitViewController

## Class

# NSSplitViewController

An object that manages an array of adjacent child views, and has a split view object for managing dividers between those views.

macOS 10.10+

```
@MainActor
class NSSplitViewController
```

---

## Overview

A split view controller manages a set of child views that it displays next to each other in a side-by-side or top-to-bottom arrangement.

A split view controller owns an array of split view items ([NSSplitViewItem](#)), each of which has a view controller ([NSViewController](#)) and corresponding view. The split view controller's [splitView](#) object manages those child views and the dividers between them.

By default, a split view arranges its child views vertically from top to bottom. To specify a horizontal (side-by-side) arrangement, implement the [isVertical](#) property of the [splitView](#) object to return [true](#).

The split view controller serves as the delegate of its [splitView](#) object. If you override a split view delegate method, your override must call `super`.

To use a split view controller, you must use Auto Layout for the child views and to support animations that collapse and reveal child views. For example, if you design a layout that contains two views, a content area and an optional sidebar, you employ Auto Layout constraints to specify whether the content area shrinks or remains the same size when the sidebar becomes visible.

A split view controller employs lazy loading of its views. For example, adding a collapsed split view item as a new child doesn't load the associated view until it shows.

For more information about using [NSSplitViewController](#) in your app, see [Navigating Hierarchical Data Using Outline and Split Views](#).

---

## Topics

### Configuring and Managing a Split View Controller

`var splitView: NSSplitView`

The split view that the split view controller manages.

`func splitViewItem(for: NSViewController) -> NSSplitViewItem?`

Returns the corresponding split view item for the specified child view controller of the split view controller.

`var splitViewItems: [NSSplitViewItem]`

The array of split view items that correspond to the split view controller's child view controllers.

`class NSSplitViewItem`

An item in a split view controller.

### Modifying a Split View Controller

`func addSplitViewItem(NSSplitViewItem)`

Adds a split view item to the end of the array of split view items.

`func insertSplitViewItem(NSSplitViewItem, at: Int)`

Adds a split view item to the array of split view items at the specified index position.

`func removeSplitViewItem(NSSplitViewItem)`

Removes a specified split view item from the split view controller.

### Managing Sidebars

`func toggleSidebar(Any?)`

Collapses or expands the first sidebar in the split view controller using an animation.

`var minimumThicknessForInlineSidebars: CGFloat`

The minimum thickness for a sidebar before it automatically collapses.

```
class let automaticDimension: CGFloat
```

The default value to apply to a dimension.

## Managing Inspectors

```
func toggleInspector(Any?)
```

## Responding to View Events

```
func viewDidLoad()
```

Configures the split view controller after its view loads into memory.

## Supporting Protocol Requirements

☰ Protocol Implementations

Access the split view controller's implementations of protocol methods.

---

# Relationships

## Inherits From

NSViewController

## Conforms To

CVarArg

CustomDebugStringConvertible

CustomStringConvertible

Equatable

Hashable

NSCoding

NSEditor

NSExtensionRequestHandling

NSObjectProtocol

NSSeguePerforming

NSSplitViewDelegate

NSStandardKeyBindingResponding

NSTouchBarProvider  
NSUserActivityRestoring  
NSUserInterfaceItemIdentification  
NSUserInterfaceValidations  
Sendable  
SendableMetatype

---

## See Also

### Split View Interface

`class NSSplitView`

A view that arranges two or more views in a linear stack running horizontally or vertically.

`class NSSplitViewItem`

An item in a split view controller.