Class

NSSplitViewItem

An item in a split view controller.

macOS 10.10+

```
class NSSplitViewItem
```

Overview

A split view item represents a single pane in a split view controller (`NSSplitViewController`). Each split view item contains information about a child view controller in the split view controller, like its preferred thickness, holding priority, and collapsed state.

To add one or more accessory views to the top or bottom of a split view item, such as a search field above a list, use the `topAlignedAccessoryViewControllers` and `bottomAlignedAccessoryViewControllers` properties to specify `NSSplitViewItemAccessoryViewController` types.

Topics

Creating a split view item

`convenience init(sidebarWithViewController: NSViewController)`

Creates a split view item that represents a sidebar for the specified view controller.

`convenience init(contentListWithViewController: NSViewController)`

Creates a split view item that represents a content list for the specified view controller.

`convenience init(viewController: NSViewController)`

Creates a split view item that represents the specified view controller.

`convenience init(inspectorWithViewController: NSViewController)`

Managing the item thickness

`var automaticMaximumThickness: CGFloat`

The maximum thickness of the split view item when it resizes due to automatic sizing.

`var preferredThicknessFraction: CGFloat`

The preferred thickness of the split view item relative to the split view.

`var minimumThickness: CGFloat`

The minimum thickness of the split view item.

`var maximumThickness: CGFloat`

The maximum thickness of the split view item.

`class let unspecifiedDimension: CGFloat`

A constant that resets a dimension's value.

Getting Auto Layout behaviors

`var holdingPriority: NSLayoutConstraint.Priority`

The priority for a split view item to hold its size.

`var automaticallyAdjustsSafeAreaInsets: Bool`

When YES, other items such as sidebars or inspectors may appear overlaid on top of this item's `viewController` and this item's `safeAreaInsets` will be adjusted with respect to overlaid content. Defaults to NO.

Getting the item behavior

`var behavior: NSSplitViewItem.Behavior`

The standard behavior type of the split view item.

`enum Behavior`

Constants that describe the behavior of the split view item.

Collapsing and expanding the item

`var isCollapsed: Bool`

A Boolean value that determines whether the child view controller that corresponds to the split view item is in a collapsed state in the split view controller.

`var canCollapse: Bool`

A Boolean value that determines whether a user interaction can collapse the child view controller that corresponds to the split view item.

`var collapseBehavior: NSSplitViewItem.CollapseBehavior`

The resizing behavior when the split view item toggles its collapsed state.

`enum CollapseBehavior`

Constants that describe the split view item's collapsing behavior.

`var isSpringLoaded: Bool`

A Boolean value that determines whether the split view item can temporarily expand during a drag.

`var canCollapseFromWindowResize: Bool`

Customizing appearance

`var allowsFullHeightLayout: Bool`

A Boolean value that indicates whether full-height sidebars appear in the window after you set a style mask.

`var titlebarSeparatorStyle: NSTitlebarSeparatorStyle`

The type of separator that the app displays between the title bar and content of a window.

`enum NSTitlebarSeparatorStyle`

Styles that determine the type of separator displayed between the title bar and content of a window.

Configuring accessory views

`var topAlignedAccessoryViewControllers: [NSSplitViewItemAccessoryViewController]`

The following methods allow you to add accessory views to the top/bottom of this splitViewItem. See NSSplitViewItemAccessoryViewController for more details.

```
var bottomAlignedAccessoryViewControllers: [NSSplitViewItemAccessory
ViewController]
```

```
func addTopAlignedAccessoryViewController(NSSplitViewItemAccessoryView
Controller)
```

```
func insertTopAlignedAccessoryViewController(NSSplitViewItemAccessory
ViewController, at: Int)
```

```
func removeTopAlignedAccessoryViewController(at: Int)
```

NOTE: you can use this method, or –removeFromParentViewController:, whichever is easier.

```
func addBottomAlignedAccessoryViewController(NSSplitViewItemAccessory
ViewController)
```

```
func insertBottomAlignedAccessoryViewController(NSSplitViewItem
AccessoryViewController, at: Int)
```

```
func removeBottomAlignedAccessoryViewController(at: Int)
```

NOTE: you can use this method, or –removeFromParentViewController:, whichever is easier.

```
class NSSplitViewItemAccessoryViewController
```

Getting the View Controller

```
var viewController: NSViewController
```

The view controller that the split view item represents.

Relationships

Inherits From

NSObject

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSAnimatablePropertyContainer
NSCoding
NSObjectProtocol

See Also

Split View Interface

`class` NSSplitViewController

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

`class` NSSplitView

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