Class

NSTableView

A set of related records, displayed in rows that represent individual records and columns that represent the attributes of those records.

macOS

```
@MainActor
class NSTableView
```

Overview

Table views are displayed in scroll views. Beginning with macOS v10.7, you can use `NSView` objects (most commonly customized `NSTableCellView` objects) instead of cells for specifying rows and columns. You can still use `NSCell` objects for each row and column item if you prefer.

A table view does not store its own data; it retrieves data values as needed from a data source to which it has a weak reference. You should not, therefore, directly set data values programmatically in the table view; instead, modify the values in the data source and allow the changes to be reflected in the table view. To learn about the methods that an `NSTableView` object uses to provide and access the contents of its data source object, see `NSTableViewDataSource`.

To customize a table view's behavior without subclassing `NSTableView`, use the methods defined by the `NSTableViewDelegate` protocol. For example, the delegate supports table column management, type-to-select functionality, row selection and editing, custom tracking, and custom views for individual columns and rows. To learn more about the table view delegate, see `NSTableViewDelegate`.

Important

It's possible that your data source methods for populating the table view may be called before `awakeFromNib()` is called if the data source is specified in Interface Builder. You should defend against this by having the data source's `numberOfRows(in:)` method return `0` for the number of rows when the data source has not yet been configured. In `awakeFromNib()`, when the data source is initialized you should always call `reloadData` on the table view.

Subclassing

Subclassing `NSTableView` is usually not necessary. Instead, you customize the table view using a delegate object (an object conforming to the `NSTableViewDelegate` protocol) and a data source object (conforming to the `NSTableViewDataSource` protocol), or by subclassing one of the following subcomponents: cells (when using `NSCell`-based table views), the row cell view or the row view (when using `NSView`-based table views), the table column class, or table column header classes.

Enabling the Table View

Use the `isEnabled` property to enable or disable the table view, which the view inherits from `NSControl`. This property affects the visual appearance of the table view differently depending on whether you use a view- or a cell-based table view. When you change the property's value for a cell-based table view, the system manages the visual appearance of that table view's rows, and updates them to a state that reflects the value. Because view-based table views permit complex items in their cells, it's the developer's responsibility to update each cell's appearance as appropriate.

Topics

Creating a Table

`init?(coder: NSCoder)`

`init(frame: NSRect)`

Managing the Table's Data

`var dataSource: (any NSTableViewDataSource)?`

The object that provides the data displayed by the table view.

```
var usesStaticContents: Bool
```

A Boolean value indicating whether the table uses static data.

```
func reloadData()
```

Marks the table view as needing redisplay, so it will reload the data for visible cells and draw the new values.

```
func reloadData(forRowIndexes: IndexSet, columnIndexes: IndexSet)
```

Reloads the data for only the specified rows and columns.

Creating Views to Display

```
func makeView(withIdentifier: NSUserInterfaceItemIdentifier, owner: Any?) -> NSView?
```

Returns a new or existing view with the specified identifier.

```
func rowView(atRow: Int, makeIfNecessary: Bool) -> NSTableRowView?
```

Returns a row view at the specified index, creating one if necessary.

```
func view(atColumn: Int, row: Int, makeIfNecessary: Bool) -> NSView?
```

Returns a view at the specified row and column indexes, creating one if necessary.

```
struct NSUserInterfaceItemIdentifier
```

Updating the Table View Arrangement

```
func beginUpdates()
```

Begins a group of updates for the table view.

```
func endUpdates()
```

Ends the group of updates for the table view.

```
func moveRow(at: Int, to: Int)
```

Moves the specified row to the new row location using animation.

```
func insertRows(at: IndexSet, withAnimation: NSTableView.AnimationOptions)
```

Inserts the rows using the specified animation.

```
func removeRows(at: IndexSet, withAnimation: NSTableView.AnimationOptions)
```

Removes the rows using the specified animation.

```
func row(for: NSView) -> Int
```

Returns the index of the row for the specified view.

```
func column(for: NSView) -> Int
```

Returns the column index for the specified view.

NSView-Based Table Nib File Registration

```
func register(NSNib?, forIdentifier: NSUserInterfaceItemIdentifier)
```

Registers a NIB for the specified identifier, so that view-based table views can use it to instantiate views.

```
var registeredNibsByIdentifier: [NSUserInterfaceItemIdentifier : NSNib
]?
```

The dictionary of all registered nib files for view-based table view identifiers.

Target-action Behavior

```
var doubleAction: Selector?
```

The message sent to the table view's target when the user double-clicks a cell or column header.

```
var clickedColumn: Int
```

The index of the column the user clicked.

```
var clickedRow: Int
```

The index of the row the user clicked.

Configuring Behavior

```
var allowsColumnReordering: Bool
```

A Boolean value indicating whether the table view allows the user to rearrange columns by dragging their headers.

```
var allowsColumnResizing: Bool
```

A Boolean value indicating whether the table view allows the user to resize columns by dragging between their headers.

```
var allowsMultipleSelection: Bool
```

A Boolean value indicating whether the table view allows the user to select more than one column or row at a time.

`var allowsEmptySelection: Bool`

A Boolean value indicating whether the table view allows the user to select zero columns or rows.

`var allowsColumnSelection: Bool`

A Boolean value indicating whether the table view allows the user to select columns by clicking their headers.

`var usesAutomaticRowHeights: Bool`

A Boolean value that indicates whether the table view uses autolayout to calculate the height of rows.

Setting Display Attributes

`var intercellSpacing: NSSize`

The horizontal and vertical spacing between cells.

`var rowHeight: CGFloat`

The height of each row in the table.

`var backgroundColor: NSColor`

The color used to draw the background of the table.

`var usesAlternatingRowBackgroundColors: Bool`

A Boolean value indicating whether the table view uses alternating row colors for its background.

`var style: NSTableView.Style`

The style that the table view uses.

`var effectiveStyle: NSTableView.Style`

The effective style that the table uses.

`enum Style`

Contains the possible style values for a table view.

`var selectionHighlightStyle: NSTableView.SelectionHighlightStyle`

The selection highlight style used by the table view to indicate row and column selection.

`var gridColor: NSColor`

The color used to draw grid lines.

`var gridStyleMask: NSTableView.GridLineStyle`

The grid lines drawn by the table view.

`func indicatorImage(in: NSTableColumn) -> NSImage?`

Returns the indicator image of the specified table column.

`func setIndicatorImage(NSImage?, in: NSTableColumn)`

Sets the indicator image of the specified column.

Getting and Setting Row Size Styles

`var effectiveRowSizeStyle: NSTableView.RowSizeStyle`

The effective row size style for the table.

`var rowSizeStyle: NSTableView.RowSizeStyle`

The row size style (small, medium, large, or custom) used by the table view.

Column Management

`func addTableColumn(NSTableColumn)`

Adds the specified column as the last column of the table view.

`func removeTableColumn(NSTableColumn)`

Removes the specified column from the table view.

`func moveColumn(Int, toColumn: Int)`

Moves the column and heading at the specified index to the new specified index.

`var tableColumns: [NSTableColumn]`

An array containing the current table column objects.

`func column(withIdentifier: NSUserInterfaceItemIdentifier) -> Int`

Returns the index of the first column in the table view whose identifier is equal to the specified identifier.

`func tableColumn(withIdentifier: NSUserInterfaceItemIdentifier) -> NSTableColumn?`

Returns the `NSTableColumn` object for the first column whose identifier is equal to the specified object.

Selecting Columns and Rows

```
func selectColumnIndexes(IndexSet, byExtendingSelection: Bool)
```

Sets the column selection using `indexes` possibly extending the selection.

```
var selectedColumn: Int
```

The index of the last selected column (or the last column added to the selection).

```
var selectedColumnIndexes: IndexSet
```

An index set containing the indexes of the selected columns.

```
func deselectColumn(Int)
```

Deselects the column at the specified index if it's selected.

```
var numberOfSelectedColumns: Int
```

The number of selected columns.

```
func isColumnSelected(Int) -> Bool
```

Returns a Boolean value that indicates whether the column at the specified index is selected.

```
func selectRowIndexes(IndexSet, byExtendingSelection: Bool)
```

Sets the row selection using `indexes` extending the selection if specified.

```
var selectedRow: Int
```

The index of the last selected row (or the last row added to the selection).

```
var selectedRowIndexes: IndexSet
```

An index set containing the indexes of the selected rows.

```
func deselectRow(Int)
```

Deselects the row at the specified index if it's selected.

```
var numberOfSelectedRows: Int
```

The number of selected rows.

```
func isRowSelected(Int) -> Bool
```

Returns a Boolean value that indicates whether the row at the specified index is selected.

```
func selectAll(Any?)
```

Selects all rows or all columns, according to whether rows or columns were most recently selected.

```
func deselectAll(Any?)
```

Deselects all selected rows or columns if empty selection is allowed; otherwise does nothing.

Enumerating Table Rows

```
func enumerateAvailableRowViews((NSTableRowView, Int) -> Void)
```

Allows the enumeration of all the table rows that are known to the table view.

Managing Type Select

```
var allowsTypeSelect: Bool
```

A Boolean value indicating whether the table view allows the user to type characters to select rows.

Table Dimensions

```
var numberOfColumns: Int
```

The number of columns in the table.

```
var numberOfRows: Int
```

The number of rows in the table.

Getting and Setting Floating Rows

```
var floatsGroupRows: Bool
```

A Boolean value indicating whether the table view draws grouped rows as if they are floating.

Editing Cells

```
func editColumn(Int, row: Int, with: NSEvent?, select: Bool)
```

Edits the cell at the specified column and row using the specified event and selection behavior.

```
var editedColumn: Int
```

The index of the column being edited.

`var editedRow: Int`

The index of the row being edited.

Adding and Deleting Row Views

`func didAdd(NSTableRowView, forRow: Int)`

Invoked when a row view is added to the table.

`func didRemove(NSTableRowView, forRow: Int)`

Invoked when a row view is removed from the table.

Setting Auxiliary Views

`var headerView: NSTableHeaderView?`

The view object used to draw headers over columns.

`var cornerView: NSView?`

The view used to draw the area to the right of the column headers and above the vertical scroller of the enclosing scroll view.

Layout Support

`var userInterfaceLayoutDirection: NSUserInterfaceLayoutDirection`

The layout direction of the user interface.

`func rect(ofColumn: Int) -> NSRect`

Returns the rectangle containing the column at the specified index.

`func rect(ofRow: Int) -> NSRect`

Returns the rectangle containing the row at the specified index.

`func rows(in: NSRect) -> NSRange`

Returns a range of indexes for the rows that lie wholly or partially within the vertical boundaries of the specified rectangle.

`func columnIndexes(in: NSRect) -> IndexSet`

Returns the indexes of the table view's columns that intersect the specified rectangle.

`func column(at: NSPoint) -> Int`

Returns the index of the column the specified point lies in.

```
func row(at: NSPoint) -> Int
```

Returns the index of the row the specified point lies in.

```
func frameOfCell(atColumn: Int, row: Int) -> NSRect
```

Returns a rectangle locating the cell that lies at the intersection of the specified column and row.

```
var columnAutoresizingStyle: NSTableView.ColumnAutoresizingStyle
```

The table view's column autoresizing style.

```
func sizeLastColumnToFit()
```

Resizes the last column so the table view fits exactly within its enclosing clip view.

```
func noteNumberOfRowsChanged()
```

Informs the table view that the number of records in its data source has changed.

```
func tile()
```

Properly sizes the table view and its header view and marks it as needing display.

```
func sizeToFit()
```

Sizes the table view based on a uniform column autoresizing style.

```
func noteHeightOfRows(withIndexesChanged: IndexSet)
```

Informs the table view that the rows specified in `indexSet` have changed height.

Drawing

```
func drawRow(Int, clipRect: NSRect)
```

Draws the cells for the row at `rowIndex` in the columns that intersect `clipRect`.

```
func drawGrid(inClipRect: NSRect)
```

Draws the grid lines within the supplied rectangle.

```
func highlightSelection(inClipRect: NSRect)
```

Highlights the region of the table view in the specified rectangle.

```
func drawBackground(inClipRect: NSRect)
```

Draws the background of the table view in the clip rect specified by the rectangle.

Scrolling

`func scrollRowToVisible(Int)`

Scrolls the view so the specified row is visible.

`func scrollColumnToVisible(Int)`

Scrolls the view so the specified column is visible.

Table Column State Persistence

`var autosaveTableColumns: Bool`

A Boolean value indicating whether the order and width of the table view's columns are automatically saved.

`var autosaveName: NSTableView.AutosaveName?`

The name under which table information is automatically saved.

`typealias AutosaveName`

Accessing the Delegate

`var delegate: (any NSTableViewDelegate)?`

The table view's delegate.

Highlightable Column Headers

`var highlightedTableColumn: NSTableColumn?`

The column highlighted in the table.

Dragging

`func dragImageForRows(with: IndexSet, tableColumns: [NSTableColumn], event: NSEvent, offset: NSPointPointer) -> NSImage`

Computes and returns an image to use for dragging.

`func canDragRows(with: IndexSet, at: NSPoint) -> Bool`

Returns a Boolean value indicating whether the table view allows dragging the rows with the drag initiated at the specified point.

`func setDraggingSourceOperationMask(NSDragOperation, forLocal: Bool)`

Sets the default operation mask returned by `draggingSourceOperationMaskForLocal:` to `mask`.

`var verticalMotionCanBeginDrag: Bool`

A Boolean value indicating whether vertical motion is treated as a drag or selection change.

`var draggingDestinationFeedbackStyle: NSTableView.DraggingDestinationFeedbackStyle`

The feedback style displayed when the user drags over the table view.

`func setDropRow(Int, dropOperation: NSTableView.DropOperation)`

Retargets the proposed drop operation.

Sorting

`var sortDescriptors: [NSSortDescriptor]`

The table view's sort descriptors.

Row Actions

`var rowActionsVisible: Bool`

A Boolean value indicating whether a table row's actions are visible.

Hiding and Showing Table Rows

`func hideRows(at: IndexSet, withAnimation: NSTableView.AnimationOptions)`

Hides the specified table rows.

`func unhideRows(at: IndexSet, withAnimation: NSTableView.AnimationOptions)`

Unhides the specified table rows.

`var hiddenRowIndexes: IndexSet`

The indexes of all hidden table rows.

Deprecated Methods

~~`func focusedColumn() -> Int`~~

Returns the currently focused column.

Deprecated

~~`func setFocusedColumn(Int)`~~

Sets the currently focused column to the specified index.

Deprecated

~~`func shouldFocusCell(NSCell, atColumn: Int, row: Int) -> Bool`~~

Returns whether the fully prepared cell at the specified row and column can be made the focused cell.

Deprecated

~~`func performClickOnCell(atColumn: Int, row: Int)`~~

Performs a click action on the cell at the specified row and column.

Deprecated

~~`func preparedCell(atColumn: Int, row: Int) -> NSCell?`~~

Returns the fully prepared cell that the table view will use for drawing or processing of the specified row and column.

Deprecated

Constants

Specifying a Custom Row View in a Nib File

View-based table view instances use `NSTableViewRowKey` to identify the nib file containing the template row view. You can specify a custom row view (without any code) by associating this key with the appropriate nib name in Interface Builder.

`enum DraggingDestinationFeedbackStyle`

These constants specify the drag styles displayed by the table view. They're used by `draggingDestinationFeedbackStyle`.

`enum DropOperation`

`NSTableView` defines these constants to specify drop operations.

`struct GridLineStyle`

`NSTableView` defines these constants to specify grid styles. These constants are used by the `gridStyleMask` property. The mask can be either `NSTableViewGridNone` or it can contain either or both of the other options combined using the C bitwise OR operator.

`enum ColumnAutoresizingStyle`

The following constants specify the autoresizing styles. These constants are used by the `columnAutoresizingStyle` property.

`enum` `SelectionHighlightStyle`

The following constants specify the selection highlight styles. These constants are used by the `selectionHighlightStyle` property.

`struct` `AnimationOptions`

Specifies the animation effects to apply when inserting or removing rows.

`enum` `RowSizeStyle`

The row size style constants define the size of the rows in the table view. They are used by the `effectiveRowSizeStyle` and `rowSizeStyle` properties. You can also query the row size in the `NSTableCellView` class' property `rowSizeStyle`.

`enum` `RowActionEdge`

These constants define table row edges on which row actions are attached. They are used by the `tableView:rowActionsForRow:edge:` delegate method.

Notifications

`class let columnDidMoveNotification: NSNotification.Name`

Posted whenever a column is moved by user action in an `NSTableView` object.

`class let columnDidResizeNotification: NSNotification.Name`

Posted whenever a column is resized in an `NSTableView` object.

`class let selectionDidChangeNotification: NSNotification.Name`

Posted after an `NSTableView` object's selection changes.

`class let selectionIsChangingNotification: NSNotification.Name`

Posted as an `NSTableView` object's selection changes (while the mouse button is still down).

Relationships

Inherits From

`NSControl`

Inherited By

NSOutlineView

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSAccessibilityElementProtocol
NSAccessibilityGroup
NSAccessibilityProtocol
NSAccessibilityTable
NSAnimatablePropertyContainer
NSAppearanceCustomization
NSCoding
NSDraggingDestination
NSDraggingSource
NSObjectProtocol
NSStandardKeyBindingResponding
NSTextDelegate
NSTextViewDelegate
NSTouchBarProvider
NSUserActivityRestoring
NSUserInterfaceItemIdentification
NSUserInterfaceValidations
Sendable
SendableMetatype

See Also

Related Documentation

Table View Programming Guide for Mac

Views

class NSTableCellView

A reusable container view shown for a particular cell in a table view that uses rows for content.