NSTableView Class Reference

Cocoa > User Experience

2009-04-30

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

Simultaneously published in the United States and Canada.

Contents

NSTableView Class Reference 9

Class at a Glance 9
Overview 10
Adopted Protocols 10
Tasks 11
- Setting the Data Source 11
- Loading Data 11
- Target-action Behavior 11
- Configuring Behavior 11
- Setting Display Attributes 12
- Column Management 13
- Selecting Columns and Rows 13
- Table Dimensions 14
- Displaying Cell 14
- Editing Cells 14
- Setting Auxiliary Views 15
- Layout Support 15
- Drawing 16
- Scrolling 16
- Persistence 16
- Selecting in the Tableview 17
- Setting the Delegate 17
- Highlightable Column Headers 17
- Dragging 18
- Sorting 18
- Moving and Resizing Columns 18
- Responding to Mouse Events 18
- Text Delegate Methods 19
- Displaying Tooltips 19
- Deprecated Methods 19

Instance Methods 20
- addTableColumn: 20
- allowsColumnReordering 20
- allowsColumnResizing 21
- allowsColumnSelection 21
- allowsEmptySelection 22
- allowsMultipleSelection 22
- allowsTypeSelect 23
- autosaveName 23
- autosaveTableColumns 23
- backgroundColor 24

canDragRowsWithIndexes:atPoint: 24
clickedColumn 25
clickedRow 25
columnAtPoint: 26
columnAutoresizingStyle 26
columnIndexesInRect: 26
columnWithIdentifier: 27
cornerView 27
dataSource 28
delegate 28
deselectAll: 28
deselectColumn: 29
deselectRow: 30
doubleAction 30
dragImageForRowsWithIndexes:tableColumns:event:offset: 31
drawBackgroundInClipRect: 31
drawGridInClipRect: 31
drawRow:clipRect: 32
editColumn:row:withEvent:select: 32
editedColumn 33
editedRow 33
frameOfCellAtColumn:row: 33
gridColor 34
gridStyleMask 35
headerView 35
highlightedTableColumn 35
highlightSelectionInClipRect: 36
indicatorImageInTableColumn: 36
intercellSpacing 37
isColumnSelected: 37
isRowSelected: 37
moveColumn:toColumn: 38
noteHeightOfRowsWithIndexesChanged: 38
noteNumberOfRowsChanged 39
numberOfColumns 39
numberOfRows 40
numberOfSelectedColumns 40
numberOfSelectedRows 40
preparedCellAtColumn:row: 41
rectOfColumn: 41
rectOfRow: 42
reloadData 42
removeTableColumn: 43
rowAtPoint: 43
rowHeight 44
rowsInRect: 44

scrollColumnToVisible: 45
scrollRowToVisible: 45
selectAll: 45
selectColumnIndexes:byExtendingSelection: 46
selectedColumn 47
selectedColumnIndexes 47
selectedRow 47
selectedRowIndexes 48
selectionHighlightStyle 48
selectRowIndexes:byExtendingSelection: 49
setAllowsColumnReordering: 49
setAllowsColumnResizing: 49
setAllowsColumnSelection: 50
setAllowsEmptySelection: 50
setAllowsMultipleSelection: 51
setAllowsTypeSelect: 51
setAutosaveName: 52
setAutosaveTableColumns: 52
setBackgroundColor: 53
setColumnAutoresizingStyle: 53
setCornerView: 53
setDataSource: 54
setDelegate: 54
setDoubleAction: 55
setDraggingSourceOperationMask:forLocal: 55
setDropRow:dropOperation: 56
setGridColor: 56
setGridStyleMask: 57
setHeaderView: 57
setHighlightedTableColumn: 58
setIndicatorImage:inTableColumn: 58
setIntercellSpacing: 58
setRowHeight: 59
setSelectionHighlightStyle: 59
setSortDescriptors: 60
setUsesAlternatingRowBackgroundColors: 60
setVerticalMotionCanBeginDrag: 60
sizeLastColumnToFit 61
sizeToFit 61
sortDescriptors 62
tableColumns 62
tableColumnWithIdentifier: 62
textDidBeginEditing: 63
textDidChange: 63
textDidEndEditing: 64
textShouldBeginEditing: 64

textShouldEndEditing: 64
tile 65
usesAlternatingRowBackgroundColors 65
verticalMotionCanBeginDrag 65
Delegate Methods 66
selectionShouldChangeInTableView: 66
tableView:dataCellForTableColumn:row: 66
tableView:didClickTableColumn: 67
tableView:didDragTableColumn: 67
tableView:heightOfRow: 68
tableView:isGroupRow: 68
tableView:mouseDownInHeaderOfTableColumn: 68
tableView:nextTypeSelectMatchFromRow:toRow:forString: 69
tableView:selectionIndexesForProposedSelection: 69
tableView:shouldEditTableColumn:row: 70
tableView:shouldSelectRow: 70
tableView:shouldSelectTableColumn: 71
tableView:shouldShowCellExpansionForTableColumn:row: 71
tableView:shouldTrackCell:forTableColumn:row: 72
tableView:shouldTypeSelectForEvent:withCurrentSearchString: 72
tableView:toolTipForCell:rect:tableColumn:row:mouseLocation: 73
tableView:typeSelectStringForTableColumn:row: 73
tableView:willDisplayCell:forTableColumn:row: 74
tableViewColumnDidMove: 74
tableViewColumnDidResize: 75
tableViewSelectionDidChange: 75
tableViewSelectionIsChanging: 75
Constants 76
Drop Operations 76
Grid styles 76
Autoresizing Styles 77
Selection Styles 78
Notifications 79
NSTableViewColumnDidMoveNotification 79
NSTableViewColumnDidResizeNotification 79
NSTableViewSelectionDidChangeNotification 79
NSTableViewSelectionIsChangingNotification 80

Appendix A **Deprecated NSTableView Methods 81**

Deprecated in Mac OS X v10.3 81
drawsGrid 81
selectColumn:byExtendingSelection: 81
selectedColumnEnumerator 82
selectedRowEnumerator 82
selectRow:byExtendingSelection: 82

setDrawsGrid: 83
Deprecated in Mac OS X v10.4 and later 83
dragImageForRows:event:dragImageOffset: 83
setAutoresizesAllColumnsToFit: 83
Deprecated in Mac OS X v10.4 84
autoresizesAllColumnsToFit 84
Deprecated in Mac OS X v10.5 84
columnsInRect: 84

Document Revision History 87

Index 89

NSTableView Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTableView.h
Companion guides	Table View Programming Guide Drag and Drop Programming Topics for Cocoa
Related sample code	CoreRecipes iSpend Mountains MyPhoto QTAudioExtractionPanel

Class at a Glance

An `NSTableView` object displays record-oriented data in a table and allows the user to edit values and resize and rearrange columns.

Principal Attributes

- A data source
- Table columns

Commonly Used Methods

`dataSource` (page 28)
Returns the object providing the data that the table view displays.

`tableColumns` (page 62)
Returns the `NSTableColumn` objects representing attributes for the table view.

`selectedColumn` (page 47)
Returns the index of the selected column.

`selectedRow` (page 47)
Returns the index of the selected row.

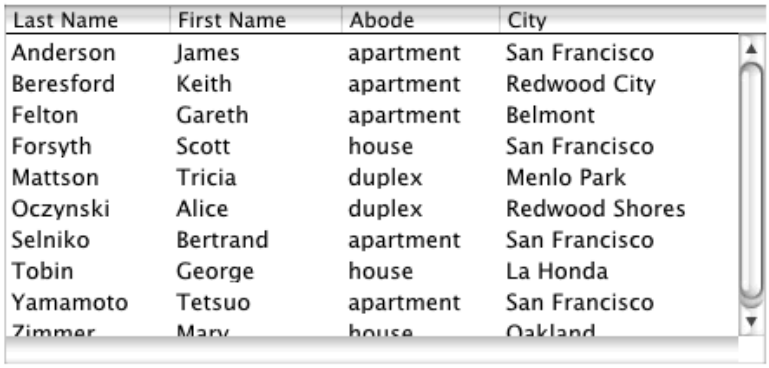
`numberOfRows` (page 40)
Returns the number of rows in the table view.

`reloadData` (page 42)
Informs the table view that data has changed and needs to be retrieved and displayed again.

Overview

An `NSTableView` object displays data for a set of related records, with rows representing individual records and columns representing the attributes of those records.

A table view is usually displayed in a scroll view, like this:



Last Name	First Name	Abode	City
Anderson	James	apartment	San Francisco
Beresford	Keith	apartment	Redwood City
Felton	Gareth	apartment	Belmont
Forsyth	Scott	house	San Francisco
Mattson	Tricia	duplex	Menlo Park
Oczynski	Alice	duplex	Redwood Shores
Selniko	Bertrand	apartment	San Francisco
Tobin	George	house	La Honda
Yamamoto	Tetsuo	apartment	San Francisco
Zimmer	Mary	house	Oakland

A table view does not store its own data, instead it retrieves data values as needed from a data source to which it has a weak reference (see Communicating With Objects). You should not, therefore, try to directly set data values programmatically in the table view; instead you should modify the values in the data source and allow the changes to be reflected in the table view. See the `NSTableDataSource` informal `protocol`, which declares the methods that an `NSTableView` object uses to access the contents of its data source object.

Adopted Protocols

NSUserInterfaceValidations
- `validateUserInterfaceItem:`

Tasks

Setting the Data Source

- `setDataSource:` (page 54)
  Sets the receiver's data source to a given object.
- `dataSource` (page 28)
  Returns the object that provides the data displayed by the receiver.

Loading Data

- `reloadData` (page 42)
  Marks the receiver as needing redisplay, so it will reload the data for visible cells and draw the new values.

Target-action Behavior

- `setDoubleAction:` (page 55)
  Sets the message sent to the target when the user double-clicks an uneditable cell or a column header to a given selector.
- `doubleAction` (page 30)
  Returns the message sent to the target when the user double-clicks a column header or an uneditable cell.
- `clickedColumn` (page 25)
  Returns the index of the column the user clicked to trigger an action message.
- `clickedRow` (page 25)
  Returns the index of the row the user clicked to trigger an action message.

Configuring Behavior

- `setAllowsColumnReordering:` (page 49)
  Controls whether the user can drag column headers to reorder columns.
- `allowsColumnReordering` (page 20)
  Returns a Boolean value that indicates whether the receiver allows the user to rearrange columns by dragging their headers.
- `setAllowsColumnResizing:` (page 49)
  Controls whether the user can resize columns by dragging between headers.
- `allowsColumnResizing` (page 21)
  Returns a Boolean value that indicates whether the receiver allows the user to resize columns by dragging between their headers.
- `setAllowsMultipleSelection:` (page 51)
  Controls whether the user can select more than one row or column at a time.

- `allowsMultipleSelection` (page 22)
  Returns a Boolean value that indicates whether the receiver allows the user to select more than one column or row at a time.
- `setAllowsEmptySelection:` (page 50)
  Controls whether the receiver allows zero rows or columns to be selected.
- `allowsEmptySelection` (page 22)
  Returns a Boolean value that indicates whether the receiver allows the user to select zero columns or rows.
- `setAllowsColumnSelection:` (page 50)
  Controls whether the user can select an entire column by clicking its header.
- `allowsColumnSelection` (page 21)
  Returns a Boolean value that indicates whether the receiver allows the user to select columns by clicking their headers.

Setting Display Attributes

- `setIntercellSpacing:` (page 58)
  Sets the width and height between cells to those in a given `NSSize` struct.
- `intercellSpacing` (page 37)
  Returns the horizontal and vertical spacing between cells.
- `setRowHeight:` (page 59)
  Sets the height for rows to a given value.
- `rowHeight` (page 44)
  Returns the height of each row in the receiver.
- `setBackgroundColor:` (page 53)
  Sets the receiver's background color to a given color.
- `backgroundColor` (page 24)
  Returns the color used to draw the background of the receiver.
- `setUsesAlternatingRowBackgroundColors:` (page 60)
  Sets whether the receiver uses the standard alternating row colors for its background.
- `usesAlternatingRowBackgroundColors` (page 65)
  Returns a Boolean value that indicates whether the receiver uses the standard alternating row colors for its background.
- `selectionHighlightStyle` (page 48)
  Returns the selection highlight style used by the receiver to indicate row and column selection.
- `setSelectionHighlightStyle:` (page 59)
  Sets the selection highlight style used by the receiver to indicate row and column selection.
- `setGridColor:` (page 56)
  Sets the color used to draw grid lines to a given color.
- `gridColor` (page 34)
  Returns the color used to draw grid lines.
- `setGridStyleMask:` (page 57)
  Sets the grid style mask to specify if no grid lines, vertical grid lines, or horizontal grid lines should be displayed.

- `gridStyleMask` (page 35)
  Returns the receiver's grid style mask.
- `indicatorImageInTableColumn:` (page 36)
  Returns the indicator image of a given table column.
- `setIndicatorImage:inTableColumn:` (page 58)
  Sets the indicator image of *`aTableColumn`* to *`anImage`*.

Column Management

- `addTableColumn:` (page 20)
  Adds a given column as the last column of the receiver.
- `removeTableColumn:` (page 43)
  Removes a given column from the receiver.
- `moveColumn:toColumn:` (page 38)
  Moves the column and heading at a given index to a new given index.
- `tableColumns` (page 62)
  Returns an array containing the the `NSTableColumn` objects in the receiver.
- `columnWithIdentifier:` (page 27)
  Returns the index of the first column in the receiver whose identifier is equal to a given identifier.
- `tableColumnWithIdentifier:` (page 62)
  Returns the `NSTableColumn` object for the first column whose identifier is equal to a given object.

Selecting Columns and Rows

- `selectColumnIndexes:byExtendingSelection:` (page 46)
  Sets the column selection using *`indexes`*.
- `selectRowIndexes:byExtendingSelection:` (page 49)
  Sets the row selection using *`indexes`*.
- `selectedColumnIndexes` (page 47)
  Returns an index set containing the indexes of the selected columns.
- `selectedRowIndexes` (page 48)
  Returns an index set containing the indexes of the selected rows.
- `deselectColumn:` (page 29)
  Deselects the column at a given index if it's selected.
- `deselectRow:` (page 30)
  Deselects the row at a given index if it's selected.
- `numberOfSelectedColumns` (page 40)
  Returns the number of selected columns.
- `numberOfSelectedRows` (page 40)
  Returns the number of selected rows.
- `selectedColumn` (page 47)
  Returns the index of the last column selected or added to the selection.

- `selectedRow` (page 47)
  Returns the index of the last row selected or added to the selection.
- `isColumnSelected:` (page 37)
  Returns a Boolean value that indicates whether the column at a given index is selected.
- `isRowSelected:` (page 37)
  Returns a Boolean value that indicates whether the row at a given index is selected.
- `selectAll:` (page 45)
  Selects all rows or all columns, according to whether rows or columns were most recently selected.
- `deselectAll:` (page 28)
  Deselects all selected rows or columns if empty selection is allowed; otherwise does nothing.
- `allowsTypeSelect` (page 23)
  Returns a Boolean value that indicates whether the receiver allows the user to type characters to select rows.
- `setAllowsTypeSelect:` (page 51)
  Sets whether the receiver allows the user to type characters to select rows.

Table Dimensions

- `numberOfColumns` (page 39)
  Returns the number of columns in the receiver.
- `numberOfRows` (page 40)
  Returns the number of rows in the receiver.

Displaying Cell

- `tableView:willDisplayCell:forTableColumn:row:` (page 74) *delegate method*
  Informs the delegate that *`aTableView`* will display the cell at *`rowIndex`* in *`aTableColumn`* using *`aCell`*.
- `preparedCellAtColumn:row:` (page 41)
  Returns the fully prepared cell that the receiver will use for drawing or processing of the specified row and column.
- `tableView:dataCellForTableColumn:row:` (page 66) *delegate method*
  Invoked to allow the delegate to return a custom data cell for a specified row and column.
- `tableView:shouldShowCellExpansionForTableColumn:row:` (page 71) *delegate method*
  Invoked to allow the delegate to control tooltip cell expansion for a specific row and column.
- `tableView:isGroupRow:` (page 68) *delegate method*
  Invoked to allow the delegate to indicate that a specified row is a group row.

Editing Cells

- `editColumn:row:withEvent:select:` (page 32)
  Edits the cell at *`columnIndex`* and *`rowIndex`*, selecting its entire contents if *`flag`* is `YES`.

- `editedColumn` (page 33)
  Returns the index of the column being edited.
- `editedRow` (page 33)
  Returns the index of the row being edited.
- `tableView:shouldEditTableColumn:row:` (page 70) *delegate method*
  Returns `YES` to permit *`aTableView`* to edit the cell at *`rowIndex`* in *`aTableColumn`*, `NO` to deny permission.

Setting Auxiliary Views

- `setHeaderView:` (page 57)
  Sets the receiver's header view to a given header view.
- `headerView` (page 35)
  Returns the `NSTableHeaderView` object used to draw headers over columns.
- `setCornerView:` (page 53)
  Sets the receiver's corner view to a given view.
- `cornerView` (page 27)
  Returns 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

- `rectOfColumn:` (page 41)
  Returns the rectangle containing the column at a given index.
- `rectOfRow:` (page 42)
  Returns the rectangle containing the row at a given index.
- `rowsInRect:` (page 44)
  Returns a range of indices for the rows that lie wholly or partially within the vertical boundaries of a given rectangle.
- `columnIndexesInRect:` (page 26)
  Returns the indexes of the receiver's columns that intersect the specified rectangle.
- `columnAtPoint:` (page 26)
  Returns the index of the column a given point lies in.
- `rowAtPoint:` (page 43)
  Returns the index of the row a given point lies in.
- `frameOfCellAtColumn:row:` (page 33)
  Returns a rectangle locating the cell that lies at the intersection of *`columnIndex`* and *`rowIndex`*.
- `columnAutoresizingStyle` (page 26)
  Returns the receiver's column autoresizing style.
- `setColumnAutoresizingStyle:` (page 53)
  Sets the column autoresizing style of the receiver to a given style.
- `sizeLastColumnToFit` (page 61)
  Resizes the last column if there's room so the receiver fits exactly within its enclosing clip view.

- `noteNumberOfRowsChanged` (page 39)
  Informs the receiver that the number of records in its data source has changed.
- `tile` (page 65)
  Properly sizes the receiver and its header view and marks it as needing display.
- `sizeToFit` (page 61)
  Changes the width of columns in the receiver so all columns are visible.
- `noteHeightOfRowsWithIndexesChanged:` (page 38)
  Informs the receiver that the rows specified in *indexSet* have changed height.
- `tableView:heightOfRow:` (page 68) *delegate method*
  Returns the height of *row* in *tableView*.
- `columnsInRect:` (page 84) Deprecated in Mac OS X v10.5
  Returns a range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of a given rectangle.

Drawing

- `drawRow:clipRect:` (page 32)
  Draws the cells for the row at *rowIndex* in the columns that intersect *clipRect*.
- `drawGridInClipRect:` (page 31)
  Draws the grid lines within *aRect*, using the grid color set with `setGridColor:` (page 56).
- `highlightSelectionInClipRect:` (page 36)
  Highlights the region of the receiver in *clipRect*.
- `drawBackgroundInClipRect:` (page 31)
  Draws the background in the clip rect specified by *clipRect*.

Scrolling

- `scrollRowToVisible:` (page 45)
  Scrolls the receiver vertically in an enclosing NSClipView so the row specified by *rowIndex* is visible.
- `scrollColumnToVisible:` (page 45)
  Scrolls the receiver and header view horizontally in an enclosing NSClipView so the column specified by *columnIndex* is visible.

Persistence

- `autosaveName` (page 23)
  Returns the name under which table information is automatically saved.
- `autosaveTableColumns` (page 23)
  Returns a Boolean value that indicates whether the order and width of the receiver's columns are automatically saved.
- `setAutosaveName:` (page 52)
  Sets the name under which table information is automatically saved to *name*.

- `setAutosaveTableColumns:` (page 52)
  Sets whether the order and width of this table view's columns are automatically saved.

Selecting in the Tableview

- `selectionShouldChangeInTableView:` (page 66) *delegate method*
  Returns `YES` to permit *`aTableView`* to change its selection (typically a row being edited), `NO` to deny permission.
- `tableView:shouldSelectRow:` (page 70) *delegate method*
  Returns `YES` to permit *`aTableView`* to select the row at *`rowIndex`*, `NO` to deny permission.
- `tableView:selectionIndexesForProposedSelection:` (page 69) *delegate method*
  Invoked to allow the delegate to modify the proposed selection.
- `tableView:shouldSelectTableColumn:` (page 71) *delegate method*
  Returns `YES` to permit *`aTableView`* to select *`aTableColumn`*, `NO` to deny permission.
- `tableViewSelectionIsChanging:` (page 75) *delegate method*
  Informs the delegate that the table view's selection is in the process of changing (typically because the user is dragging the mouse across a number of rows).
- `tableViewSelectionDidChange:` (page 75) *delegate method*
  Informs the delegate that the table view's selection has changed.
- `tableView:shouldTypeSelectForEvent:withCurrentSearchString:` (page 72) *delegate method*
  Invoked to allow the delegate to control type select for a specific event.
- `tableView:typeSelectStringForTableColumn:row:` (page 73) *delegate method*
  Invoked to allow the delegate to provide an alternate text value used for type selection for a specified row and column.
- `tableView:nextTypeSelectMatchFromRow:toRow:forString:` (page 69) *delegate method*
  Invoked to allow the delegate to allow the delegate to modify how type selection works.

Setting the Delegate

- `setDelegate:` (page 54)
  Sets the receiver's delegate to a given object.
- `delegate` (page 28)
  Returns the receiver's delegate.

Highlightable Column Headers

- `highlightedTableColumn` (page 35)
  Returns the table column highlighted in the receiver.
- `setHighlightedTableColumn:` (page 58)
  Sets *`aTableColumn`* to be the currently highlighted column header.

Dragging

- `dragImageForRowsWithIndexes:tableColumns:event:offset:` (page 31)
  Computes and returns an image to use for dragging.
- `canDragRowsWithIndexes:atPoint:` (page 24)
  Returns whether the receiver allows dragging the rows at *`rowIndexes`* with a drag initiated at *`mousedDownPoint`*.
- `setDraggingSourceOperationMask:forLocal:` (page 55)
  Sets the default operation mask returned by `draggingSourceOperationMaskForLocal:` to *`mask`*.
- `setDropRow:dropOperation:` (page 56)
  Used if you wish to "retarget" the proposed drop.
- `setVerticalMotionCanBeginDrag:` (page 60)
  Sets whether vertical motion is treated as a drag or selection change to *`flag`*.
- `verticalMotionCanBeginDrag` (page 65)
  Returns whether vertical motion is treated as a drag or selection change.

Sorting

- `setSortDescriptors:` (page 60)
  Sets the receiver's sort descriptors to the `NSSortDescriptor` objects in *`array`*.
- `sortDescriptors` (page 62)
  Returns the receiver's sort descriptors.

Moving and Resizing Columns

- `tableView:didDragTableColumn:` (page 67) *delegate method*
  Sent at the time the mouse button goes up in *`tableView`* and *`tableColumn`* has been dragged during the time the mouse button was down.
- `tableViewColumnDidMove:` (page 74) *delegate method*
  Informs the delegate that a column was moved by user action in the table view.
- `tableViewColumnDidResize:` (page 75) *delegate method*
  Informs the delegate that a column was resized in the table view.

Responding to Mouse Events

- `tableView:didClickTableColumn:` (page 67) *delegate method*
  Sent at the time the mouse button subsequently goes up in *`tableView`* and *`tableColumn`* has been "clicked" without having been dragged anywhere.
- `tableView:mouseDownInHeaderOfTableColumn:` (page 68) *delegate method*
  Sent to the delegate whenever the mouse button is clicked in *`tableView`* while the cursor is in a column header *`tableColumn`*.
- `tableView:shouldTrackCell:forTableColumn:row:` (page 72) *delegate method*
  Invoked to allow the delegate to control the tracking behavior for a specific cell.

Text Delegate Methods

- `textShouldBeginEditing:` (page 64)
  Queries the delegate using `control:textShouldBeginEditing:`, returning the delegate's response, or simply returning `YES` to allow editing of *`textObject`* if the delegate doesn't respond to that method.
- `textDidBeginEditing:` (page 63)
  Posts an `NSControlTextDidBeginEditingNotification` to the default notification center.
- `textDidChange:` (page 63)
  Sends `textDidChange:` (page 63) to the edited cell and posts an `NSControlTextDidChangeNotification` to the default notification center.
- `textShouldEndEditing:` (page 64)
  Validates the *`textObject`* cell being edited and queries the delegate using `control:textShouldEndEditing:`, returning the delegate's response if it responds to that method.
- `textDidEndEditing:` (page 64)
  Updates the data source based on the newly edited value and selects another cell for editing if possible according to the character that ended editing (Return, Tab, Backtab).

Displaying Tooltips

- `tableView:toolTipForCell:rect:tableColumn:row:mouseLocation:` (page 73) *delegate method*
  Returns a string that is displayed as a tooltip for *`aCell`* in *`aTableColumn`* of *`aTableView`*.

Deprecated Methods

- `drawsGrid` (page 81) Deprecated in Mac OS X v10.3
  Returns a Boolean value that indicates whether the receiver draws a grid. (Deprecated. Use `gridStyleMask` (page 35) instead.)
- `selectColumn:byExtendingSelection:` (page 81) Deprecated in Mac OS X v10.3
  Selects a column at a given index, optionally extending any existing selection. (Deprecated. Use `selectColumnIndexes:byExtendingSelection:` (page 46) instead.)
- `selectedColumnEnumerator` (page 82) Deprecated in Mac OS X v10.3
  This method has been deprecated. (Deprecated. Use `selectedColumnIndexes` (page 47) instead.)
- `selectedRowEnumerator` (page 82) Deprecated in Mac OS X v10.3
  This method has been deprecated. (Deprecated. Use `selectedRowIndexes` (page 48) instead.)
- `selectRow:byExtendingSelection:` (page 82) Deprecated in Mac OS X v10.3
  Selects a row at a given index, optionally extending any existing selection. (Deprecated. Use `selectRowIndexes:byExtendingSelection:` (page 49) instead.)
- `setDrawsGrid:` (page 83) Deprecated in Mac OS X v10.3
  Sets whether the receiver draws a grid. (Deprecated. Use `setGridStyleMask:` (page 57) instead.)
- `dragImageForRows:event:dragImageOffset:` (page 83) Deprecated in Mac OS X v10.4 and later
  Computes and returns an image to use for dragging. (Deprecated. Use `dragImageForRowsWithIndexes:tableColumns:event:offset:` (page 31) instead.)

- `setAutoresizesAllColumnsToFit:` (page 83) Deprecated in Mac OS X v10.4 and later
  Controls whether the receiver proportionally resizes its columns to fit when its superview's frame changes. (Deprecated. Use `setColumnAutoresizingStyle:` (page 53) instead.)
- `autoresizesAllColumnsToFit` (page 84) Deprecated in Mac OS X v10.4
  Returns `YES` if the receiver proportionally resizes its columns to fit when its superview's frame changes, `NO` if it only resizes the last column. (Deprecated. Use `columnAutoresizingStyle` (page 26) instead.)

Instance Methods

addTableColumn:

Adds a given column as the last column of the receiver.

```
- (void)addTableColumn:(NSTableColumn *)aColumn
```

Parameters

aColumn
The column to add to the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sizeLastColumnToFit` (page 61)
- `removeTableColumn:` (page 43)

Declared In
`NSTableView.h`

allowsColumnReordering

Returns a Boolean value that indicates whether the receiver allows the user to rearrange columns by dragging their headers.

```
- (BOOL)allowsColumnReordering
```

Return Value
`YES` to allow the user to rearrange columns by dragging their headers, otherwise `NO`.

Discussion
The default is `YES`. You can rearrange columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveColumn:toColumn:` (page 38)
- `setAllowsColumnReordering:` (page 49)

Declared In
NSTableView.h

allowsColumnResizing

Returns a Boolean value that indicates whether the receiver allows the user to resize columns by dragging between their headers.

```
- (BOOL)allowsColumnResizing
```

Return Value
YES if the receiver allows the user to resize columns by dragging between their headers, otherwise NO.

Discussion
The default is YES. You can resize columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setWidth: (NSTableColumn)
- setAllowsColumnResizing: (page 49)

Declared In
NSTableView.h

allowsColumnSelection

Returns a Boolean value that indicates whether the receiver allows the user to select columns by clicking their headers.

```
- (BOOL)allowsColumnSelection
```

Return Value
YES if the receiver allows the user to select columns by clicking their headers, otherwise NO.

Discussion
The default is NO. You can select columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectColumn:byExtendingSelection: (page 81)
- allowsColumnReordering (page 20)
- setAllowsColumnSelection: (page 50)

Declared In
NSTableView.h

allowsEmptySelection

Returns a Boolean value that indicates whether the receiver allows the user to select zero columns or rows.

```
- (BOOL)allowsEmptySelection
```

Return Value
`YES` if the receiver allows the user to select zero columns or rows, otherwise `NO`.

Discussion
The default is `YES`.

You cannot set an empty selection programmatically if this setting is `NO`, unlike with the other settings that affect selection behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deselectAll:` (page 28)
- `deselectColumn:` (page 29)
- `deselectRow:` (page 30)
- `setAllowsEmptySelection:` (page 50)

Declared In
`NSTableView.h`

allowsMultipleSelection

Returns a Boolean value that indicates whether the receiver allows the user to select more than one column or row at a time.

```
- (BOOL)allowsMultipleSelection
```

Return Value
`YES` if the receiver allows the user to select more than one column or row at a time, otherwise `NO`.

Discussion
The default is `NO`. You can select multiple columns or rows programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectColumn:byExtendingSelection:` (page 81)
- `selectRow:byExtendingSelection:` (page 82)
- `setAllowsMultipleSelection:` (page 51)

Declared In
`NSTableView.h`

allowsTypeSelect

Returns a Boolean value that indicates whether the receiver allows the user to type characters to select rows.

```
- (BOOL)allowsTypeSelect
```

Return Value
`YES` if the receiver allows type selection, otherwise `NO`.

Discussion
The default value is `YES`.

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

See Also
`- setAllowsTypeSelect:` (page 51)

Declared In
`NSTableView.h`

autosaveName

Returns the name under which table information is automatically saved.

```
- (NSString *)autosaveName
```

Return Value
The name under which table information is automatically saved. If no name has been set, returns `nil`.

Discussion
The table information is saved separately for each user and for each application that user uses.

Note that even when a table view has an autosave name, it may not be saving table information automatically. To check whether table information is being saved automatically, use `autosaveTableColumns` (page 23).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autosaveTableColumns` (page 23)
`- setAutosaveName:` (page 52)

Declared In
`NSTableView.h`

autosaveTableColumns

Returns a Boolean value that indicates whether the order and width of the receiver's columns are automatically saved.

```
- (BOOL)autosaveTableColumns
```

Discussion
The table information is saved separately for each user and for each application that user uses. Note that if `autosaveName` (page 23) returns `nil`, this setting is ignored and table information isn't saved.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveName` (page 23)
- `setAutosaveTableColumns:` (page 52)
- `setAutosaveName:` (page 52)

Declared In
`NSTableView.h`

backgroundColor

Returns the color used to draw the background of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The color used to draw the background of the receiver.

Discussion
The default background color is light gray.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundColor:` (page 53)

Declared In
`NSTableView.h`

canDragRowsWithIndexes:atPoint:

Returns whether the receiver allows dragging the rows at *`rowIndexes`* with a drag initiated at *`mousedDownPoint`*.

```
- (BOOL)canDragRowsWithIndexes:(NSIndexSet *)rowIndexes
    atPoint:(NSPoint)mouseDownPoint
```

Discussion
Return `NO` to disallow the drag.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

clickedColumn

Returns the index of the column the user clicked to trigger an action message.

```
- (NSInteger)clickedColumn
```

Return Value
The index of the column the user clicked to trigger an action message. Returns `-1` if the user clicked in an area of the table view not occupied by columns.

Discussion
The return value of this method is meaningful only in the target's implementation of the action or double-action method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickedRow` (page 25)
- `setAction:` (`NSControl`)
- `setDoubleAction:` (page 55)

Declared In
`NSTableView.h`

clickedRow

Returns the index of the row the user clicked to trigger an action message.

```
- (NSInteger)clickedRow
```

Return Value
The index of the row the user clicked to trigger an action message. Returns `-1` if the user clicked in an area of the table view not occupied by table rows.

Discussion
The return value of this method is meaningful only in the target's implementation of the action or double-action method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickedColumn` (page 25)
- `setAction:` (`NSControl`)
- `setDoubleAction:` (page 55)

Related Sample Code
WhackedTV

Declared In
`NSTableView.h`

columnAtPoint:

Returns the index of the column a given point lies in.

```
- (NSInteger)columnAtPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point in the coordinate system of the receiver.

Return Value

The index of the column *aPoint* lies in, or -1 if *aPoint* lies outside the receiver's bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rowAtPoint: (page 43)

Declared In

NSTableView.h

columnAutoresizingStyle

Returns the receiver's column autoresizing style.

```
- (NSTableViewColumnAutoresizingStyle)columnAutoresizingStyle
```

Return Value

The receiver's column autoresizing style. For possible values, see "Autoresizing Styles" (page 77).

Availability

Available in Mac OS X v10.4 and later.

See Also

- setColumnAutoresizingStyle: (page 53)

Declared In

NSTableView.h

columnIndexesInRect:

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

```
- (NSIndexSet *)columnIndexesInRect:(NSRect)rect
```

Parameters

rect

The rectangle in the receiver's coordinate system to test for column enclosure.

Return Value

New NSIndexSet object containing the indexes of the receiver's columns that intersect with *rect*.

Discussion

Columns that return YES for the NSTableColumn method isHidden are excluded from the results.

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

Declared In
NSTableView.h

columnWithIdentifier:

Returns the index of the first column in the receiver whose identifier is equal to a given identifier.

```
- (NSInteger)columnWithIdentifier:(id)anObject
```

Parameters

anObject
A column identifier.

Return Value
The index of the first column in the receiver whose identifier is equal to *anObject* (when compared using `isEqual:`) or `-1` if no columns are found with the specified identifier.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tableColumnWithIdentifier:` (page 62)

Declared In
NSTableView.h

cornerView

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

```
- (NSView *)cornerView
```

Return Value
The view used to draw the area to the right of the column headers and above the vertical scroller of the enclosing `NSScrollView` object.

Discussion
This is by default a simple view that merely fills in its frame, but you can replace it with a custom view using `setCornerView:` (page 53).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `headerView` (page 35)

Declared In
NSTableView.h

dataSource

Returns the object that provides the data displayed by the receiver.

```
- (id)dataSource
```

Return Value
The object that provides the data displayed by the receiver.

Discussion
See Using a Table Data Source and the `NSTableDataSource` informal `protocol` specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDataSource:` (page 54)

Declared In
`NSTableView.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDelegate:` (page 54)

Related Sample Code
People

Declared In
`NSTableView.h`

deselectAll:

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

```
- (void)deselectAll:(id)sender
```

Parameters
sender
Typically the object that sent the message.

Discussion
Posts `NSTableViewSelectionDidChangeNotification` (page 79) to the default notification center if the selection does in fact change.

As a target-action method, `deselectAll:` checks with the delegate before changing the selection, using `selectionShouldChangeInTableView:` (page 66).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 22)
- `selectAll:` (page 45)
- `selectColumn:byExtendingSelection:` (page 81)

Related Sample Code
EnhancedAudioBurn
OpenGLCompositorLab

Declared In
`NSTableView.h`

deselectColumn:

Deselects the column at a given index if it's selected.

```
- (void)deselectColumn:(NSInteger)columnIndex
```

Parameters
columnIndex
The index of the column to deselect.

Discussion
Deselects the column at *columnIndex* if it's selected, regardless of whether empty selection is allowed.

If the selection does in fact change, posts `NSTableViewSelectionDidChangeNotification` (page 79) to the default notification center.

If the indicated column was the last column selected by the user, the column nearest it effectively becomes the last selected column. In case of a tie, priority is given to the column on the left.

This method doesn't check with the delegate before changing the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumn` (page 47)
- `allowsEmptySelection` (page 22)
- `selectRow:byExtendingSelection:` (page 82)

Declared In
`NSTableView.h`

deselectRow:

Deselects the row at a given index if it's selected.

```
- (void)deselectRow:(NSInteger)rowIndex
```

Parameters

rowIndex

The index of the row to deselect.

Discussion

Deselects the row at *rowIndex* if it's selected, regardless of whether empty selection is allowed.

If the selection does in fact change, posts `NSTableViewSelectionDidChangeNotification` (page 79) to the default notification center.

If the indicated row was the last row selected by the user, the row nearest it effectively becomes the last selected row. In case of a tie, priority is given to the row above.

This method doesn't check with the delegate before changing the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectedRow` (page 47)
- `allowsEmptySelection` (page 22)

Declared In

`NSTableView.h`

doubleAction

Returns the message sent to the target when the user double-clicks a column header or an uneditable cell.

```
- (SEL)doubleAction
```

Return Value

The message the receiver sends to its target when the user double-clicks a column header or an uneditable cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `action` (`NSControl`)
- `target` (`NSControl`)
- `setDoubleAction:` (page 55)

Declared In

`NSTableView.h`

dragImageForRowsWithIndexes:tableColumns:event:offset:

Computes and returns an image to use for dragging.

```
- (NSImage *)dragImageForRowsWithIndexes:(NSIndexSet *)dragRows
    tableColumns:(NSArray *)tableColumns
    event:(NSEvent *)dragEvent
    offset:(NSPointPointer)dragImageOffset
```

Discussion
Override this to return a custom image. *dragRows* represents the rows participating in the drag. *tableColumns* represents the table columns that should be in the output image. *dragEvent* is a reference to the mouse-down event that began the drag. *dragImageOffset* is an in/out parameter.

This method is called with *dragImageOffset* set to NSZeroPoint, but it can be modified to reposition the returned image. A *dragImageOffset* of NSZeroPoint will cause the image to be centered under the cursor.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTableView.h

drawBackgroundInClipRect:

Draws the background in the clip rect specified by *clipRect*.

```
- (void)drawBackgroundInClipRect:(NSRect)clipRect
```

Availability
Available in Mac OS X v10.3 and later

Declared In
NSTableView.h

drawGridInClipRect:

Draws the grid lines within *aRect*, using the grid color set with setGridColor: (page 56).

```
- (void)drawGridInClipRect:(NSRect)aRect
```

Discussion
This method draws a grid regardless of whether the receiver is set to draw one automatically.

Subclasses can override this method to draw grid lines other than the standard ones.

Availability
Available in Mac OS X v10.0 and later.

See Also
- gridColor (page 34)
- setIntercellSpacing: (page 58)
- drawsGrid (page 81)

- drawRow:clipRect: (page 32)
- highlightSelectionInClipRect: (page 36)

Declared In
NSTableView.h

drawRow:clipRect:

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

```
- (void)drawRow:(NSInteger)rowIndex
    clipRect:(NSRect)clipRect
```

Discussion
Sends tableView:willDisplayCell:forTableColumn:row: (page 74) to the delegate before drawing each cell.

Subclasses can override this method to customize their appearance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- columnsInRect: (page 84)
- highlightSelectionInClipRect: (page 36)
- drawGridInClipRect: (page 31)

Declared In
NSTableView.h

editColumn:row:withEvent:select:

Edits the cell at *columnIndex* and *rowIndex*, selecting its entire contents if *flag* is YES.

```
- (void)editColumn:(NSInteger)columnIndex
    row:(NSInteger)rowIndex
    withEvent:(NSEvent *)theEvent
    select:(BOOL)flag
```

Discussion
This method is invoked automatically in response to user actions; you should rarely need to invoke it directly. *theEvent* is usually the mouse event that triggered editing; it can be nil when starting an edit programmatically.

This method scrolls the receiver so that the cell is visible, sets up the field editor, and sends selectWithFrame:inView:editor:delegate:start:length: and editWithFrame:inView:editor:delegate:event: to the field editor's NSCell object with the NSTableView as the text delegate.

The row at *rowIndex* must be selected prior to calling editColumn:row:withEvent:select:, or an exception will be raised.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `editedColumn` (page 33)
- `editedRow` (page 33)

Declared In
`NSTableView.h`

editedColumn

Returns the index of the column being edited.

```
- (NSInteger)editedColumn
```

Return Value
If sent during `editColumn:row:withEvent:select:` (page 32), the index of the column being edited; otherwise `-1`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
People

Declared In
`NSTableView.h`

editedRow

Returns the index of the row being edited.

```
- (NSInteger)editedRow
```

Return Value
If sent during `editColumn:row:withEvent:select:` (page 32), the index of the row being edited; otherwise `-1`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
People

Declared In
`NSTableView.h`

frameOfCellAtColumn:row:

Returns a rectangle locating the cell that lies at the intersection of *`columnIndex`* and *`rowIndex`*.

```
- (NSRect)frameOfCellAtColumn:(NSInteger)columnIndex
    row:(NSInteger)rowIndex
```

Parameters

columnIndex

The index of the column containing the cell whose rectangle you want.

rowIndex

The index of the row containing the cell whose rectangle you want.

Return Value

A rectangle locating the cell that lies at the intersection of *columnIndex* and *rowIndex*. Returns `NSZeroRect` if *columnIndex* or *rowIndex* is greater than the number of columns or rows in the receiver.

Discussion

You can use this method to update a single cell more efficiently than sending the table view a reloadData (page 42) message.

```
[aTableView setNeedsDisplayInRect:[aTableView frameOfCellAtColumn:column
row:row]];
```

The result of this method is used in a `drawWithFrame:inView:` message to the table column's data cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `rectOfColumn:` (page 41)
- `rectOfRow:` (page 42)

Declared In

`NSTableView.h`

gridColor

Returns the color used to draw grid lines.

```
- (NSColor *)gridColor
```

Return Value

The color used to draw grid lines.

Discussion

The default color is gray.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `drawsGrid` (page 81)
- `drawGridInClipRect:` (page 31)
- `setGridColor:` (page 56)

Declared In

`NSTableView.h`

gridStyleMask

Returns the receiver's grid style mask.

```
- (NSUInteger)gridStyleMask
```

Return Value
The receiver's grid style mask. Possible return values are described in "Grid styles" (page 76).

Availability
Available in Mac OS X v10.3 and later.

See Also
- setGridStyleMask: (page 57)

Declared In
NSTableView.h

headerView

Returns the `NSTableHeaderView` object used to draw headers over columns.

```
- (NSTableHeaderView *)headerView
```

Return Value
The `NSTableHeaderView` object used to draw headers over columns, or `nil` if the receiver has no header view

Discussion
See The Parts of a Table and the `NSTableHeaderView` class specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHeaderView: (page 57)

Declared In
NSTableView.h

highlightedTableColumn

Returns the table column highlighted in the receiver.

```
- (NSTableColumn *)highlightedTableColumn
```

Return Value
The table column highlighted in the receiver.

Discussion
A highlightable column header can be used in conjunction with row selection to highlight a particular column of the table. An example of this is how the Mail application indicates the currently sorted column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHighlightedTableColumn:` (page 58)

Declared In
`NSTableView.h`

highlightSelectionInClipRect:

Highlights the region of the receiver in *clipRect*.

```
- (void)highlightSelectionInClipRect:(NSRect)clipRect
```

Discussion
This method is invoked before `drawRow:clipRect:` (page 32).

Subclasses can override this method to change the manner in which they highlight selections.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawGridInClipRect:` (page 31)

Declared In
`NSTableView.h`

indicatorImageInTableColumn:

Returns the indicator image of a given table column.

```
- (NSImage *)indicatorImageInTableColumn:(NSTableColumn *)aTableColumn
```

Parameters
aTableColumn
A table column in the receiver.

Discussion
An indicator image is an arbitrary (small) image that is rendered on the right side of the column header. An example of its use is in Mail to indicate the sorting direction of the currently sorted column in a mailbox.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIndicatorImage:inTableColumn:` (page 58)

Declared In
`NSTableView.h`

intercellSpacing

Returns the horizontal and vertical spacing between cells.

```
- (NSSize)intercellSpacing
```

Return Value
The horizontal and vertical spacing between cells.

Discussion
The default spacing is `(3.0, 2.0)`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsGrid:` (page 83)
- `setIntercellSpacing:` (page 58)

Related Sample Code
MP3 Player

Declared In
`NSTableView.h`

isColumnSelected:

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

```
- (BOOL)isColumnSelected:(NSInteger)columnIndex
```

Parameters
`columnIndex`
The index of the column to test.

Return Value
`YES` if the column at *`columnIndex`* is selected, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumn` (page 47)
- `selectedColumnEnumerator` (page 82)
- `selectColumn:byExtendingSelection:` (page 81)

Declared In
`NSTableView.h`

isRowSelected:

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

```
- (BOOL)isRowSelected:(NSInteger)rowIndex
```

Parameters

rowIndex

The index of the row to test.

Return Value

YES if the row at *rowIndex* is selected, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- selectedRow (page 47)
- selectedRowEnumerator (page 82)
- selectRow:byExtendingSelection: (page 82)

Declared In

NSTableView.h

moveColumn:toColumn:

Moves the column and heading at a given index to a new given index.

```
- (void)moveColumn:(NSInteger)columnIndex
    toColumn:(NSInteger)newIndex
```

Parameters

columnIndex

The current index of the column to move.

newIndex

The new index for the moved column.

Discussion

This method posts NSTableViewColumnDidMoveNotification (page 79) to the default notification center.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTableView.h

noteHeightOfRowsWithIndexesChanged:

Informs the receiver that the rows specified in *indexSet* have changed height.

```
- (void)noteHeightOfRowsWithIndexesChanged:(NSIndexSet *)indexSet
```

Discussion

If the delegate implements tableView:heightOfRow: (page 68) this method immediately re-tiles the table view using the row heights the delegate provides.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

noteNumberOfRowsChanged

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

```
- (void)noteNumberOfRowsChanged
```

Discussion
This method allows the receiver to update the scrollers in its scroll view without actually reloading data into the receiver. It's useful for a data source that continually receives data in the background over a period of time, in which case the table view can remain responsive to the user while the data is received.

See the `NSTableDataSource` informal protocol specification for information on the messages an `NSTableView` object sends to its data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reloadData` (page 42)
- `numberOfRowsInTableView:` (`NSTableDataSource` informal protocol)

Declared In
`NSTableView.h`

numberOfColumns

Returns the number of columns in the receiver.

```
- (NSInteger)numberOfColumns
```

Return Value
The number of columns in the receiver.

Discussion
The value returned includes table columns that are currently hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfRows` (page 40)

Related Sample Code
NSOperationSample

Declared In
`NSTableView.h`

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

Discussion
Typically you should not ask the table view how many rows it has; instead you should interrogate the table view's data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfColumns` (page 39)
- `numberOfRowsInTableView:` (`NSTableDataSource` informal protocol)

Related Sample Code
EnhancedAudioBurn
iSpend
MP3 Player
QTAudioExtractionPanel

Declared In
`NSTableView.h`

numberOfSelectedColumns

Returns the number of selected columns.

```
- (NSInteger)numberOfSelectedColumns
```

Return Value
The number of selected columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfSelectedRows` (page 40)
- `selectedColumnEnumerator` (page 82)

Declared In
`NSTableView.h`

numberOfSelectedRows

Returns the number of selected rows.

```
- (NSInteger)numberOfSelectedRows
```

Return Value
The number of selected rows.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfSelectedColumns` (page 40)
- `selectedRowEnumerator` (page 82)

Declared In
`NSTableView.h`

preparedCellAtColumn:row:

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

```
- (NSCell *)preparedCellAtColumn:(NSInteger)column
    row:(NSInteger)row
```

Parameters

column
The column index for which to return the appropriate cell.

row
The row index for which to return the appropriate cell.

Return Value
New `NSCell` subclass instance to use for the specified *row* and *column*. The value for the cell is correctly set, and the delegate method `tableView:willDisplayCell:forTableColumn:row:` (page 74) will have been called.

Discussion
You can override this method to do any additional cell set up that is required, or invoke it to retrieve a cell that has its contents configured for the specified *column* and *row*.

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

Declared In
`NSTableView.h`

rectOfColumn:

Returns the rectangle containing the column at a given index.

```
- (NSRect)rectOfColumn:(NSInteger)columnIndex
```

Parameters

columnIndex
The index of a column in the receiver.

Return Value
The rectangle containing the column at *columnIndex*. Returns NSZeroRect if *columnIndex* lies outside the range of valid column indices for the receiver.

Discussion
You can use this method to update a single column more efficiently than sending the table view a reloadData (page 42) message.

```
[aTableView setNeedsDisplayInRect:[aTableView rectOfColumn:column]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- frameOfCellAtColumn:row: (page 33)
- rectOfRow: (page 42)
- headerRectOfColumn: (NSTableHeaderView)

Declared In
NSTableView.h

rectOfRow:

Returns the rectangle containing the row at a given index.

```
- (NSRect)rectOfRow:(NSInteger)rowIndex
```

Return Value
The rectangle containing the row at *rowIndex*. Returns NSZeroRect if *rowIndex* lies outside the range of valid row indices for the receiver.

Discussion
You can use this method to update a single row more efficiently than sending the table view a reloadData (page 42) message.

```
[aTableView setNeedsDisplayInRect:[aTableView rectOfRow:row]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- frameOfCellAtColumn:row: (page 33)
- rectOfColumn: (page 41)

Declared In
NSTableView.h

reloadData

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

```
- (void)reloadData
```

Discussion
This method forces redraw of all the visible cells in the receiver. If you want to update the value in a single cell, column, or row, it is more efficient to use frameOfCellAtColumn:row: (page 33), rectOfColumn: (page 41), or rectOfRow: (page 42) in conjunction with `setNeedsDisplayInRect:` (`NSView`). If you just want to update the scroller, use noteNumberOfRowsChanged (page 39); if the height of a set of rows changes, use noteHeightOfRowsWithIndexesChanged: (page 38).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `noteNumberOfRowsChanged` (page 39)
- `noteHeightOfRowsWithIndexesChanged:` (page 38)
- `frameOfCellAtColumn:row:` (page 33)
- `rectOfColumn:` (page 41)
- `rectOfRow:` (page 42)

Related Sample Code
ABPresence
WhackedTV

Declared In
`NSTableView.h`

removeTableColumn:

Removes a given column from the receiver.

```
- (void)removeTableColumn:(NSTableColumn *)aTableColumn
```

Parameters
aTableColumn
The column to remove from the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sizeLastColumnToFit` (page 61)
- `addTableColumn:` (page 20)

Declared In
`NSTableView.h`

rowAtPoint:

Returns the index of the row a given point lies in.

```
- (NSInteger)rowAtPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point in the coordinate system of the receiver.

Return Value

The index of the row *aPoint* lies in, or -1 if *aPoint* lies outside the receiver's bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- columnAtPoint: (page 26)

Declared In

NSTableView.h

rowHeight

Returns the height of each row in the receiver.

```
- (CGFloat)rowHeight
```

Return Value

The height of each row in the receiver.

Discussion

The default row height is 16.0.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setRowHeight: (page 59)

Related Sample Code

Mountains

Declared In

NSTableView.h

rowsInRect:

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

```
- (NSRange)rowsInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the coordinate system of the receiver.

Return Value

A range of indices for the receiver's rows that lie wholly or partially within the horizontal boundaries of *aRect*. If the width or height of *aRect* is 0, returns an NSRange whose length is 0.

Discussion
The location of the range is the first such row's index, and the length is the number of rows that lie in *aRect*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- columnsInRect: (page 84)

Declared In
NSTableView.h

scrollColumnToVisible:

Scrolls the receiver and header view horizontally in an enclosing NSClipView so the column specified by *columnIndex* is visible.

```
- (void)scrollColumnToVisible:(NSInteger)columnIndex
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollRowToVisible: (page 45)
- scrollToPoint: (NSClipView)

Declared In
NSTableView.h

scrollRowToVisible:

Scrolls the receiver vertically in an enclosing NSClipView so the row specified by *rowIndex* is visible.

```
- (void)scrollRowToVisible:(NSInteger)rowIndex
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollColumnToVisible: (page 45)
- scrollToPoint: (NSClipView)

Related Sample Code
OpenGLCompositorLab

Declared In
NSTableView.h

selectAll:

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

```
- (void)selectAll:(id)sender
```

Parameters

sender

Typically the object that sent the message.

Discussion

If the table allows multiple selection, this action method selects all rows or all columns, according to whether rows or columns were most recently selected. If nothing has been recently selected, this method selects all rows. If this table doesn't allow multiple selection, this method does nothing.

If the selection does change, this method posts `NSTableViewSelectionDidChangeNotification` (page 79) to the default notification center.

As a target-action method, `selectAll:` checks with the delegate before changing the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allowsMultipleSelection` (page 22)
- `deselectAll:` (page 28)
- `selectColumn:byExtendingSelection:` (page 81)

Declared In

`NSTableView.h`

selectColumnIndexes:byExtendingSelection:

Sets the column selection using *indexes*.

```
- (void)selectColumnIndexes:(NSIndexSet *)indexes
    byExtendingSelection:(BOOL)extend
```

Discussion

If the *extend* flag is `NO` the selected columns are specified by *indexes*. If *extend* is `YES`, the columns indicated by *indexes* are added to the collection of already selected columns, providing multiple selection.

If a subclass implements only the deprecated `selectColumn:byExtendingSelection:` (page 81) method, then this method will be invoked in a loop. If a subclass implements this method, then `selectColumn:byExtendingSelection:` is not used. This allows subclasses that already implement `selectColumn:byExtendingSelection:` to still receive all selection messages. To avoid cycles, implementations of this method and `selectColumn:byExtendingSelection:` should not invoke each other.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `selectRowIndexes:byExtendingSelection:` (page 49)

Declared In

`NSTableView.h`

selectedColumn

Returns the index of the last column selected or added to the selection.

```
- (NSInteger)selectedColumn
```

Return Value
The index of the last column selected or added to the selection, or `-1` if no column is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumnEnumerator` (page 82)
- `numberOfSelectedColumns` (page 40)
- `selectColumn:byExtendingSelection:` (page 81)
- `deselectColumn:` (page 29)

Declared In
`NSTableView.h`

selectedColumnIndexes

Returns an index set containing the indexes of the selected columns.

```
- (NSIndexSet *)selectedColumnIndexes
```

Return Value
An index set containing the indexes of the selected columns.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedRowIndexes` (page 48)
- `selectColumnIndexes:byExtendingSelection:` (page 46)

Declared In
`NSTableView.h`

selectedRow

Returns the index of the last row selected or added to the selection.

```
- (NSInteger)selectedRow
```

Return Value
The index of the last row selected or added to the selection, or `-1` if no row is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRowEnumerator` (page 82)
- `numberOfSelectedRows` (page 40)
- `selectRow:byExtendingSelection:` (page 82)
- `deselectRow:` (page 30)

Related Sample Code
CocoaEcho
EnhancedDataBurn
NSOperationSample

Declared In
`NSTableView.h`

selectedRowIndexes

Returns an index set containing the indexes of the selected rows.

```
- (NSIndexSet *)selectedRowIndexes
```

Return Value
An index set containing the indexes of the selected rows.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedColumnIndexes` (page 47)
- `selectRowIndexes:byExtendingSelection:` (page 49)

Declared In
`NSTableView.h`

selectionHighlightStyle

Returns the selection highlight style used by the receiver to indicate row and column selection.

```
- (NSTableViewSelectionHighlightStyle)selectionHighlightStyle
```

Return Value
The selection highlight style used by the receiver to use to indicate row and column selection. See "Selection Styles" (page 78) for the possible values.

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

Declared In
`NSTableView.h`

selectRowIndexes:byExtendingSelection:

Sets the row selection using *indexes*.

```
- (void)selectRowIndexes:(NSIndexSet *)indexes
    byExtendingSelection:(BOOL)extend
```

Discussion
If the *extend* flag is NO the selected rows are specified by *indexes*. If *extend* is YES, the rows indicated by *indexes* are added to the collection of already selected rows, providing multiple selection.

If a subclass implements only the deprecated selectRow:byExtendingSelection: (page 82) method, then that method will be invoked in a loop. This allows subclasses that already implement selectRow:byExtendingSelection: to still receive all selection messages. If a subclass implements selectRowIndexes:byExtendingSelection:, then selectRow:byExtendingSelection: is not used. Note that to avoid cycles, implementations of this method and selectRow:byExtendingSelection: should not invoke each other.

Availability
Available in Mac OS X v10.3 and later.

See Also
- selectColumnIndexes:byExtendingSelection: (page 46)

Declared In
NSTableView.h

setAllowsColumnReordering:

Controls whether the user can drag column headers to reorder columns.

```
- (void)setAllowsColumnReordering:(BOOL)flag
```

Parameters
flag
YES to allow the user to reorder columns, otherwise NO.

Discussion
The default is YES. You can rearrange columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- moveColumn:toColumn: (page 38)
- allowsColumnReordering (page 20)

Declared In
NSTableView.h

setAllowsColumnResizing:

Controls whether the user can resize columns by dragging between headers.

```
- (void)setAllowsColumnResizing:(BOOL)flag
```

Parameters

flag

YES to allow the user to resize columns, otherwise NO.

Discussion

The default is YES. You can resize columns programmatically regardless of this setting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setWidth: (NSTableColumn)
- allowsColumnResizing (page 21)

Declared In

NSTableView.h

setAllowsColumnSelection:

Controls whether the user can select an entire column by clicking its header.

```
- (void)setAllowsColumnSelection:(BOOL)flag
```

Parameters

flag

YES to allow the user to select columns, otherwise NO.

Discussion

The default is NO. You can select columns programmatically regardless of this setting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- selectColumn:byExtendingSelection: (page 81)
- setAllowsColumnReordering: (page 49)
- allowsColumnSelection (page 21)

Declared In

NSTableView.h

setAllowsEmptySelection:

Controls whether the receiver allows zero rows or columns to be selected.

```
- (void)setAllowsEmptySelection:(BOOL)flag
```

Parameters

flag

YES if an empty selection is allowed, otherwise NO.

Discussion
The default is `YES`.

Unlike with the other settings that affect selection behavior, you cannot set an empty selection programmatically if empty selection is disallowed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deselectAll:` (page 28)
- `deselectColumn:` (page 29)
- `deselectRow:` (page 30)
- `allowsEmptySelection` (page 22)

Declared In
`NSTableView.h`

setAllowsMultipleSelection:

Controls whether the user can select more than one row or column at a time.

```
- (void)setAllowsMultipleSelection:(BOOL)flag
```

Parameters
flag
`YES` to allow the user to select multiple rows or columns, otherwise `NO`.

Discussion
The default is `NO`. You can select multiple columns or rows programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectColumn:byExtendingSelection:` (page 81)
- `selectRow:byExtendingSelection:` (page 82)
- `allowsMultipleSelection` (page 22)

Declared In
`NSTableView.h`

setAllowsTypeSelect:

Sets whether the receiver allows the user to type characters to select rows.

```
- (void)setAllowsTypeSelect:(BOOL)value
```

Parameters
value
`YES` if the receiver allows type selection, otherwise `NO`.

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

See Also
- `allowsTypeSelect` (page 23)

Declared In
`NSTableView.h`

setAutosaveName:

Sets the name under which table information is automatically saved to *name*.

```
- (void)setAutosaveName:(NSString *)name
```

Discussion
If *name* is different from the current name, this method also reads in the saved information and sets the order and width of this table view's columns to match.

The table information is saved separately for each user and for each application that user uses. Note that even though a table view has an autosave name, it may not be saving table information automatically. To set whether table information is being saved automatically, use `setAutosaveTableColumns:` (page 52).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveName` (page 23)
- `setAutosaveTableColumns:` (page 52)

Declared In
`NSTableView.h`

setAutosaveTableColumns:

Sets whether the order and width of this table view's columns are automatically saved.

```
- (void)setAutosaveTableColumns:(BOOL)flag
```

Discussion
If *flag* is different from the current value, this method also reads in the saved information and sets the table options to match.

The table information is saved separately for each user and for each application that user uses. Note that if `autosaveName` (page 23) returns `nil`, this setting is ignored and table information isn't saved.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveTableColumns` (page 23)
- `setAutosaveName:` (page 52)

Declared In
NSTableView.h

setBackgroundColor:

Sets the receiver's background color to a given color.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setNeedsDisplay: (NSView)
- backgroundColor (page 24)

Declared In
NSTableView.h

setColumnAutoresizingStyle:

Sets the column autoresizing style of the receiver to a given style.

```
- (void)setColumnAutoresizingStyle:(NSTableViewColumnAutoresizingStyle)style
```

Parameters

style

The column autoresizing style for the receiver. For possible values, see "Autoresizing Styles" (page 77).

Availability
Available in Mac OS X v10.4 and later.

See Also
- columnAutoresizingStyle (page 26)

Declared In
NSTableView.h

setCornerView:

Sets the receiver's corner view to a given view.

```
- (void)setCornerView:(NSView *)aView
```

Parameters

aView

The corner view for the receiver.

Discussion
The default corner view merely draws a bezeled rectangle using a blank `NSTableHeaderCell` object, but you can replace it with a custom view that displays an image or with a control that can handle mouse events, such as a select all button. Your custom corner view should be as wide as a vertical `NSScroller` object and as tall as the receiver's header view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHeaderView:` (page 57)
- `cornerView` (page 27)

Declared In
`NSTableView.h`

setDataSource:

Sets the receiver's data source to a given object.

```
- (void)setDataSource:(id)anObject
```

Parameters
anObject
The data source for the receiver. The object must implement the appropriate methods of the `NSTableDataSource` informal protocol.

Discussion
In a managed memory environment, the receiver maintains a weak reference to the data source (that is, it does not retain the data source, see Communicating With Objects). After setting the data source, this method invokes `tile` (page 65).

This method raises an `NSInternalInconsistencyException` if *anObject* doesn't respond to either `numberOfRowsInTableView:` or `tableView:objectValueForTableColumn:row:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dataSource` (page 28)

Declared In
`NSTableView.h`

setDelegate:

Sets the receiver's delegate to a given object.

```
- (void)setDelegate:(id)anObject
```

Parameters
anObject
The delegate for the receiver.

Discussion
In a managed memory environment, the receiver maintains a weak reference to the delegate (that is, it does not retain the delegate, see Communicating With Objects).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 28)

Declared In
`NSTableView.h`

setDoubleAction:

Sets the message sent to the target when the user double-clicks an uneditable cell or a column header to a given selector.

```
- (void)setDoubleAction:(SEL)aSelector
```

Parameters

aSelector

The message the receiver sends to its target when the user double-clicks an uneditable cell or a column header.

Discussion
If the double-clicked cell is editable, this message isn't sent and the cell is edited instead. You can use this method to implement features such as sorting records according to the column that was double-clicked. See also `clickedRow` (page 25) which you can use to determine if a row was clicked rather than the column heading.

For the method to have any effect, the receiver's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (`NSControl`)
- `setTarget:` (`NSControl`)
- `doubleAction` (page 30)

Declared In
`NSTableView.h`

setDraggingSourceOperationMask:forLocal:

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

```
- (void)setDraggingSourceOperationMask:(NSDragOperation)mask
    forLocal:(BOOL)isLocal
```

Discussion
If *isLocal* is YES then *mask* applies when the destination object is in the same application. If *isLocal* is NO then *mask* applies when the destination object in an application outside the receiver's application. NSTableView will archive the operation mask you set for each *isLocal* setting.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTableView.h

setDropRow:dropOperation:

Used if you wish to "retarget" the proposed drop.

```
- (void)setDropRow:(NSInteger)row
    dropOperation:(NSTableViewDropOperation)operation
```

Discussion
To specify a drop on the second row, one would specify *row* as 1, and *operation* as NSTableViewDropOn. To specify a drop below the last row, one would specify *row* as [self numberOfRows] and *operation* as NSTableViewDropAbove. Passing a value of -1 for *row*, and NSTableViewDropOn as the *operation* causes the entire table view to be highlighted rather than a specific row. This is useful if the data displayed by the receiver does not allow the user to drop items at a specific row location.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
bMoviePalette
bMoviePaletteCocoa
EnhancedAudioBurn
iSpend
MP3 Player

Declared In
NSTableView.h

setGridColor:

Sets the color used to draw grid lines to a given color.

```
- (void)setGridColor:(NSColor *)aColor
```

Parameters
aColor
The color to use to draw grid lines.

Discussion
The default color is gray.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsGrid:` (page 83)
- `drawGridInClipRect:` (page 31)
- `gridColor` (page 34)

Declared In
`NSTableView.h`

setGridStyleMask:

Sets the grid style mask to specify if no grid lines, vertical grid lines, or horizontal grid lines should be displayed.

```
- (void)setGridStyleMask:(NSUInteger)gridType
```

Parameters

gridType

The grid style mask. Possible values for *gridType* are described in "Grid styles" (page 76).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `gridStyleMask` (page 35)

Declared In
`NSTableView.h`

setHeaderView:

Sets the receiver's header view to a given header view.

```
- (void)setHeaderView:(NSTableHeaderView *)aHeaderView
```

Parameters

aHeaderView

The header view for the receiver.

Discussion
If *aHeaderView* is `nil`, the current header view is removed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setCornerView:` (page 53)
- `headerView` (page 35)

Declared In
`NSTableView.h`

setHighlightedTableColumn:

Sets *aTableColumn* to be the currently highlighted column header.

```
- (void)setHighlightedTableColumn:(NSTableColumn *)aTableColumn
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- highlightedTableColumn (page 35)

Related Sample Code
NSOperationSample

Declared In
NSTableView.h

setIndicatorImage:inTableColumn:

Sets the indicator image of *aTableColumn* to *anImage*.

```
- (void)setIndicatorImage:(NSImage *)anImage
    inTableColumn:(NSTableColumn *)aTableColumn
```

Discussion
anImage is retained and released by the table view as appropriate.

The default sorting order indicators are available as named NSImage objects. These images are accessed using [NSImage imageNamed:] passing either @"NSAscendingSortIndicator" (the "^" icon), and @"NSDescendingSortIndicator" (the "v" icon).

Availability
Available in Mac OS X v10.0 and later.

See Also
- indicatorImageInTableColumn: (page 36)

Related Sample Code
NSOperationSample

Declared In
NSTableView.h

setIntercellSpacing:

Sets the width and height between cells to those in a given NSSize struct.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters
aSize
An NSSize struct that defines the width and height between cells in the receiver.

Discussion
The receiver redisplays after the new value is set.

The default intercell spacing is `(3.0, 2.0)`.

Table views normally have a 1 pixel separation between consecutively selected rows or columns. An intercell spacing of `(1.0, 1.0)` or greater is required if you want this separation. An intercell spacing of `(0.0, 0.0)` forces there to be no separation between consecutive selections.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- intercellSpacing` (page 37)

Declared In
`NSTableView.h`

setRowHeight:

Sets the height for rows to a given value.

```
- (void)setRowHeight:(CGFloat)rowHeight
```

Parameters
rowHeight
The height for rows.

Discussion
After the height is set, this method invokes `tile` (page 65).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- rowHeight` (page 44)

Related Sample Code
Mountains

Declared In
`NSTableView.h`

setSelectionHighlightStyle:

Sets the selection highlight style used by the receiver to indicate row and column selection.

```
- (void)setSelectionHighlightStyle:(NSTableViewSelectionHighlightStyle)selectionHighlightStyle
```

Parameters
selectionHighlightStyle
The selection highlight style to use to indicate row and column selection. See "Selection Styles" (page 78) for the possible values.

Discussion
Setting the selection highlight style to `NSTableViewSelectionHighlightStyleSourceList` (page 78) causes the receiver to draw its background using the source list style.

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

Declared In
`NSTableView.h`

setSortDescriptors:

Sets the receiver's sort descriptors to the `NSSortDescriptor` objects in *array*.

```
- (void)setSortDescriptors:(NSArray *)array
```

Discussion
A table column is considered sortable if it has a sort descriptor that specifies the sorting direction, a key to sort by, and a selector defining how to sort. The array of sort descriptors is archived. Sort descriptors persist along with other column information if an autosave name is set.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- sortDescriptors` (page 62)

Declared In
`NSTableView.h`

setUsesAlternatingRowBackgroundColors:

Sets whether the receiver uses the standard alternating row colors for its background.

```
- (void)setUsesAlternatingRowBackgroundColors:(BOOL)useAlternatingRowColors
```

Parameters
useAlternatingRowColors
`YES` to specify standard alternating row colors for the background, `NO` to specify a solid color.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- usesAlternatingRowBackgroundColors` (page 65)

Declared In
`NSTableView.h`

setVerticalMotionCanBeginDrag:

Sets whether vertical motion is treated as a drag or selection change to *flag*.

```
- (void)setVerticalMotionCanBeginDrag:(BOOL)flag
```

Discussion
If *flag* is NO then vertical motion will not start a drag. The default is YES.

Note that horizontal motion is always a valid motion to begin a drag. Most often, you would want to disable vertical dragging when it's expected that horizontal dragging is the natural motion.

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalMotionCanBeginDrag (page 65)

Declared In
NSTableView.h

sizeLastColumnToFit

Resizes the last column if there's room so the receiver fits exactly within its enclosing clip view.

```
- (void)sizeLastColumnToFit
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAutoresizesAllColumnsToFit: (page 83)
- minWidth (NSTableColumn)
- maxWidth (NSTableColumn)

Declared In
NSTableView.h

sizeToFit

Changes the width of columns in the receiver so all columns are visible.

```
- (void)sizeToFit
```

Discussion
All columns are resized to the same size, up to a column's maximum size. This method then invokes tile (page 65).

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTableView.h

sortDescriptors

Returns the receiver's sort descriptors.

```
- (NSArray *)sortDescriptors
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- setSortDescriptors: (page 60)

Declared In
NSTableView.h

tableColumns

Returns an array containing the the NSTableColumn objects in the receiver.

```
- (NSArray *)tableColumns
```

Return Value
An array containing the the NSTableColumn objects in the receiver.

Discussion
The array returned by tableColumns contains all receiver's columns, including those that are hidden.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
bMoviePalette
bMoviePaletteCocoa
NSOperationSample
People

Declared In
NSTableView.h

tableColumnWithIdentifier:

Returns the NSTableColumn object for the first column whose identifier is equal to a given object.

```
- (NSTableColumn *)tableColumnWithIdentifier:(id)anObject
```

Parameters
anObject
A column identifier.

Return Value
The NSTableColumn object for the first column whose identifier is equal to *anObject*, as compared using isEqual:, or nil if no columns are found with the specified identifier.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `columnWithIdentifier:` (page 27)

Declared In
`NSTableView.h`

textDidBeginEditing:

Posts an `NSControlTextDidBeginEditingNotification` to the default notification center.

```
- (void)textDidBeginEditing:(NSNotification *)aNotification
```

Parameters

aNotification

The notification posted by the field editor; see the `NSText` class specifications for more information on this text delegate method.

Discussion
For more details, see the `NSControl` class specification.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textShouldBeginEditing:` (page 64)

Declared In
`NSTableView.h`

textDidChange:

Sends `textDidChange:` (page 63) to the edited cell and posts an `NSControlTextDidChangeNotification` to the default notification center.

```
- (void)textDidChange:(NSNotification *)aNotification
```

Parameters

aNotification

The notification posted by the field editor.

Discussion
See the `NSText` class specification for more information on this text delegate method. For additional details, see the `NSControl` class specification.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

textDidEndEditing:

Updates the data source based on the newly edited value and selects another cell for editing if possible according to the character that ended editing (Return, Tab, Backtab).

```
- (void)textDidEndEditing:(NSNotification *)aNotification
```

Discussion
aNotification is the NSNotification posted by the field editor; see the NSText class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textShouldEndEditing: (page 64)

Declared In
NSTableView.h

textShouldBeginEditing:

Queries the delegate using control:textShouldBeginEditing:, returning the delegate's response, or simply returning YES to allow editing of *textObject* if the delegate doesn't respond to that method.

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject
```

Discussion
See the NSText class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textDidBeginEditing: (page 63)

Declared In
NSTableView.h

textShouldEndEditing:

Validates the *textObject* cell being edited and queries the delegate using control:textShouldEndEditing:, returning the delegate's response if it responds to that method.

```
- (BOOL)textShouldEndEditing:(NSText *)textObject
```

Discussion
If it doesn't, it returns YES if the cell's new value is valid and NO if it isn't. See the NSText class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textDidEndEditing:` (page 64)

Declared In
`NSTableView.h`

tile

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

```
- (void)tile
```

Discussion
Also resets cursor rectangles for the header view and line scroll amounts for the `NSScrollView` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNeedsDisplay:` (`NSView`)

Declared In
`NSTableView.h`

usesAlternatingRowBackgroundColors

Returns a Boolean value that indicates whether the receiver uses the standard alternating row colors for its background.

```
- (BOOL)usesAlternatingRowBackgroundColors
```

Return Value
`YES` if the receiver uses standard alternating row colors for the background, `NO` if it uses a solid color.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setUsesAlternatingRowBackgroundColors:` (page 60)

Declared In
`NSTableView.h`

verticalMotionCanBeginDrag

Returns whether vertical motion is treated as a drag or selection change.

```
- (BOOL)verticalMotionCanBeginDrag
```

Discussion
`NO` means that vertical motion will not start a drag. Note that horizontal motion is always a valid motion to begin a drag.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setVerticalMotionCanBeginDrag:` (page 60)

Declared In
`NSTableView.h`

Delegate Methods

selectionShouldChangeInTableView:

Returns `YES` to permit *`aTableView`* to change its selection (typically a row being edited), `NO` to deny permission.

```
- (BOOL)selectionShouldChangeInTableView:(NSTableView *)aTableView
```

Discussion
The user can select and edit different cells within the same row, but can't select another row unless the delegate approves. The delegate can implement this method for complex validation of edited rows based on the values of any of their cells.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

tableView:dataCellForTableColumn:row:

Invoked to allow the delegate to return a custom data cell for a specified row and column.

```
- (NSCell *)tableView:(NSTableView *)tableView
    dataCellForTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row
```

Parameters

`tableView`
The table view that sent the message.

`tableColumn`
A column in *`tableView`*.

`row`
A row in *`tableView`*.

Return Value
An `NSCell` subclass that is used for the specified *`row`* and *`tableColumn`*. The returned cell must properly implement `copyWithZone:`.

Discussion
A different data cell can be returned for any particular table column and row, or a cell that will be used for the entire row (a full width cell).

If the tableColumn is non-`nil`, you should return a cell, and generally that will be the result of sending *`tableColumn`* a `dataCellForRow:` message.

This method will be invoked with a *`tableColumn`* value of `nil` to allow you to return a group cell–a cell that will be used to draw the entire row, acting as a separator. If you return a cell when *`tableColumn`* is `nil`, any implemented datasource and delegate methods must be prepared to handle a `nil` table column value.

Implementation of this method is optional.

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

Declared In
`NSTableView.h`

tableView:didClickTableColumn:

Sent at the time the mouse button subsequently goes up in *`tableView`* and *`tableColumn`* has been "clicked" without having been dragged anywhere.

```
- (void)tableView:(NSTableView *)tableView
    didClickTableColumn:(NSTableColumn *)tableColumn
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

tableView:didDragTableColumn:

Sent at the time the mouse button goes up in *`tableView`* and *`tableColumn`* has been dragged during the time the mouse button was down.

```
- (void)tableView:(NSTableView *)tableView
    didDragTableColumn:(NSTableColumn *)tableColumn
```

Special Considerations

The behavior of this method on Mac OS X v10.5 is different from prior versions. In version 10.5 the dragged column is sent to the delegate. In earlier versions the table column that is currently located at the dragged column's original index is sent.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

tableView:heightOfRow:

Returns the height of *row* in *tableView*.

```
- (CGFloat)tableView:(NSTableView *)tableView
    heightOfRow:(NSInteger)row
```

Discussion
You should implement this method if your table supports varying row heights. The height returned should not include intercell spacing and must be greater than zero.

Although table views may cache the returned values, you should ensure that this method is efficient. When you change a row's height you must invalidate the existing row height by calling `noteHeightOfRowsWithIndexesChanged:` (page 38). `NSTableView` automatically invalidates its entire row height cache when `reloadData` (page 42) and `noteNumberOfRowsChanged` (page 39) are called.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

tableView:isGroupRow:

Invoked to allow the delegate to indicate that a specified row is a group row.

```
- (BOOL)tableView:(NSTableView *)tableView
    isGroupRow:(NSInteger)row
```

Parameters

tableView
The table view that sent the message.

row
A row in *tableView*.

Return Value
`YES` if the specified row should have the group row style drawn, `NO` otherwise.

Discussion
If the cell in *row* is an `NSTextFieldCell` and contains only a string, the group row style attributes will automatically be applied to the cell.

Implementation of this method is optional.

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

Declared In
`NSTableView.h`

tableView:mouseDownInHeaderOfTableColumn:

Sent to the delegate whenever the mouse button is clicked in *tableView* while the cursor is in a column header *tableColumn*.

```
- (void)tableView:(NSTableView *)tableView
    mouseDownInHeaderOfTableColumn:(NSTableColumn *)tableColumn
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableView:nextTypeSelectMatchFromRow:toRow:forString:

Invoked to allow the delegate to allow the delegate to modify how type selection works.

```
- (NSInteger)tableView:(NSTableView *)tableView
    nextTypeSelectMatchFromRow:(NSInteger)startRow
    toRow:(NSInteger)endRow
    forString:(NSString *)searchString
```

Parameters

tableView
The table view that sent the message.

startRow
The starting row of the search range.

endRow
The ending row of the search range.

searchString
A string containing the typed selection.

Return Value
The first row in the range of *startRow* through *endRow* (excluding *endRow* itself) that matches *selectionString*. Return -1 if no match is found.

Discussion
It is possible for endRow to be less than startRow if the search will wrap.

Implementation of this method is optional.

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

Declared In
NSTableView.h

tableView:selectionIndexesForProposedSelection:

Invoked to allow the delegate to modify the proposed selection.

```
- (NSIndexSet *)tableView:(NSTableView *)tableView
    selectionIndexesForProposedSelection:(NSIndexSet *)proposedSelectionIndexes
```

Parameters

tableView

The table view that sent the message.

proposedSelectionIndexes

An index set containing the indexes of the proposed selection.

Return Value

An NSIndexSet instance containing the indexes of the new selection. Return *proposedSelectionIndexes* if the proposed selection is acceptable, or the value of the table view's existing selection to avoid changing the selection.

Discussion

This method may be called multiple times with one new index added to the existing selection to find out if a particular index can be selected when the user is extending the selection with the keyboard or mouse.

Implementation of this method is optional. If implemented, this method will be called instead of `tableView:shouldSelectRow:` (page 70).

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSTableView.h

tableView:shouldEditTableColumn:row:

Returns `YES` to permit *aTableView* to edit the cell at *rowIndex* in *aTableColumn*, `NO` to deny permission.

```
- (BOOL)tableView:(NSTableView *)aTableView
    shouldEditTableColumn:(NSTableColumn *)aTableColumn
    row:(NSInteger)rowIndex
```

Discussion

The delegate can implement this method to disallow editing of specific cells.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTableView.h

tableView:shouldSelectRow:

Returns `YES` to permit *aTableView* to select the row at *rowIndex*, `NO` to deny permission.

```
- (BOOL)tableView:(NSTableView *)aTableView
    shouldSelectRow:(NSInteger)rowIndex
```

Discussion

The delegate can implement this method to disallow selection of particular rows.

For better performance and finer-grain control over the selection, use `tableView:selectionIndexesForProposedSelection:` (page 69).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableView:shouldSelectTableColumn:

Returns YES to permit *aTableView* to select *aTableColumn*, NO to deny permission.

```
- (BOOL)tableView:(NSTableView *)aTableView
    shouldSelectTableColumn:(NSTableColumn *)aTableColumn
```

Discussion
The delegate can implement this method to disallow selection of particular columns.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableView:shouldShowCellExpansionForTableColumn:row:

Invoked to allow the delegate to control tooltip cell expansion for a specific row and column.

```
- (BOOL)tableView:(NSTableView *)tableView
    shouldShowCellExpansionForTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row
```

Parameters

tableView
The table view that sent the message.

tableColumn
A column in *tableView*.

row
A row in *tableView*.

Return Value
YES if the cell should expand, NO otherwise.

Discussion
Implementation of this method is optional.

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

Declared In
NSTableView.h

tableView:shouldTrackCell:forTableColumn:row:

Invoked to allow the delegate to control the tracking behavior for a specific cell.

```
- (BOOL)tableView:(NSTableView *)tableView
    shouldTrackCell:(NSCell *)cell
    forTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row
```

Parameters

tableView

The table view that sent the message.

cell

The cell to track.

tableColumn

A column in *tableView*.

row

A row in *tableView*.

Return Value

YES if the cell should track, NO otherwise.

Discussion

Normally, only selectable or selected cells can be tracked. If you implement this method, cells which are not selectable or selected can be tracked, and vice-versa.

For example, this allows you to have an `NSButtonCell` in a table which does not change the selection, but can still be clicked on and tracked.

Implementation of this method is optional.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSTableView.h`

tableView:shouldTypeSelectForEvent:withCurrentSearchString:

Invoked to allow the delegate to control type select for a specific event.

```
- (BOOL)tableView:(NSTableView *)tableView
    shouldTypeSelectForEvent:(NSEvent *)event
    withCurrentSearchString:(NSString *)searchString
```

Parameters

tableView

The table view that sent the message.

event

The event.

searchString

The search string or `nil` if no type select has began.

Return Value
YES to allow type select for event, NO otherwise.

Discussion
Typically, this is called from the table view `keyDown:` implementation and the event will be a key event.

Implementation of this method is optional.

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

Declared In
`NSTableView.h`

tableView:toolTipForCell:rect:tableColumn:row:mouseLocation:

Returns a string that is displayed as a tooltip for *aCell* in *aTableColumn* of *aTableView*.

```
- (NSString *)tableView:(NSTableView *)aTableView
    toolTipForCell:(NSCell *)aCell
    rect:(NSRectPointer)rect
    tableColumn:(NSTableColumn *)aTableColumn
    row:(NSInteger)row
    mouseLocation:(NSPoint)mouseLocation
```

Discussion
The *row* is the row of the cell and *aTableColumn* is the NSTableColumn that contains the cell. The *rect* represents the proposed active area of the tooltip. By default, *rect* is computed as `[cell drawingRectForBounds:cellFrame]`. You can modify *rect* to provide an alternative active area. Return `nil` or the empty string if no tooltip is desired.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

tableView:typeSelectStringForTableColumn:row:

Invoked to allow the delegate to provide an alternate text value used for type selection for a specified row and column.

```
- (NSString *)tableView:(NSTableView *)tableView
    typeSelectStringForTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row
```

Parameters

tableView
The table view that sent the message.

tableColumn
A column in *tableView*.

row
A row in *tableView*.

Return Value
A string that is used in type select comparison for *row* and *tableColumn*. Return `nil` if the *row* or *tableColumn* should not be searched.

Discussion
Implement this method to change the string value that is searched for based on what is displayed. By default, all cells with text in them are searched.

If this delegate method is not implemented the string value is:

```
[[tableView preparedCellAtColumn:tableColumn row:row] stringValue]
```

This value can be returned from the delegate method if desired.

Implementation of this method is optional.

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

Declared In
`NSTableView.h`

tableView:willDisplayCell:forTableColumn:row:

Informs the delegate that *aTableView* will display the cell at *rowIndex* in *aTableColumn* using *aCell*.

```
- (void)tableView:(NSTableView *)aTableView
    willDisplayCell:(id)aCell
    forTableColumn:(NSTableColumn *)aTableColumn
    row:(NSInteger)rowIndex
```

Discussion
The delegate can modify the display attributes of *aCell* to alter the appearance of the cell. Because *aCell* is reused for every row in *aTableColumn*, the delegate must set the display attributes both when drawing special cells and when drawing normal cells.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

tableViewColumnDidMove:

Informs the delegate that a column was moved by user action in the table view.

```
- (void)tableViewColumnDidMove:(NSNotification *)aNotification
```

Discussion
aNotification is an `NSTableViewColumnDidMoveNotification` (page 79).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableViewColumnDidResize:

Informs the delegate that a column was resized in the table view.

```
- (void)tableViewColumnDidResize:(NSNotification *)aNotification
```

Discussion
aNotification is an NSTableViewColumnDidResizeNotification (page 79).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableViewSelectionDidChange:

Informs the delegate that the table view's selection has changed.

```
- (void)tableViewSelectionDidChange:(NSNotification *)aNotification
```

Discussion
aNotification is an NSTableViewSelectionDidChangeNotification (page 79).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

tableViewSelectionIsChanging:

Informs the delegate that the table view's selection is in the process of changing (typically because the user is dragging the mouse across a number of rows).

```
- (void)tableViewSelectionIsChanging:(NSNotification *)aNotification
```

Discussion
aNotification is an NSTableViewSelectionIsChangingNotification (page 80).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableView.h

Constants

Drop Operations

`NSTableView` defines these constants to specify drop operations.

```
enum {
    NSTableViewDropOn,
    NSTableViewDropAbove
};
typedef NSUInteger NSTableViewDropOperation;
```

Constants

`NSTableViewDropOn`

Specifies that the drop should occur on the specified row.

Available in Mac OS X v10.0 and later.

Declared in `NSTableView.h`.

`NSTableViewDropAbove`

Specifies that the drop should occur above the specified row.

Available in Mac OS X v10.0 and later.

Declared in `NSTableView.h`.

Discussion

For example, given a table with `n` rows (numbered with row `0` at the top visually), a row of `n-1` and operation of `NSTableViewDropOn` would specify a drop on the last row. To specify a drop below the last row, you use a row of `n` and `NSTableViewDropAbove` for the operation.

Declared In

`NSTableView.h`

Grid styles

`NSTableView` defines these constants to specify grid styles.

```
enum {
    NSTableViewGridNone                   = 0,
    NSTableViewSolidVerticalGridLineMask  = 1 << 0,
    NSTableViewSolidHorizontalGridLineMask = 1 << 1
};
```

Constants

`NSTableViewGridNone`

Specifies that no grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

`NSTableViewSolidVerticalGridLineMask`

Specifies that vertical grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

`NSTableViewSolidHorizontalGridLineMask`
Specifies that horizontal grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

Discussion
These constants are used by `gridStyleMask` (page 35) and `setGridStyleMask:` (page 57). The mask can be either `NSTableViewGridNone` (page 76) or it can contain either or both of the other options combined using the C bitwise OR operator.

Declared In
`NSTableView.h`

Autoresizing Styles

The following constants specify the autoresizing styles. These constants are used by `columnAutoresizingStyle` (page 26) and `setColumnAutoresizingStyle:` (page 53).

```
enum {
    NSTableViewNoColumnAutoresizing = 0,
    NSTableViewUniformColumnAutoresizingStyle,
    NSTableViewSequentialColumnAutoresizingStyle,
    NSTableViewReverseSequentialColumnAutoresizingStyle,
    NSTableViewLastColumnOnlyAutoresizingStyle,
    NSTableViewFirstColumnOnlyAutoresizingStyle
};
typedef NSUInteger NSTableViewColumnAutoresizingStyle;
```

Constants
`NSTableViewNoColumnAutoresizing`
Disable table column autoresizing.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewUniformColumnAutoresizingStyle`
Autoresize all columns by distributing space equally, simultaneously.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewSequentialColumnAutoresizingStyle`
Autoresize each table column sequentially, from the last auto-resizable column to the first auto-resizable column; proceed to the next column when the current column has reached its minimum or maximum size.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewReverseSequentialColumnAutoresizingStyle`
Autoresize each table column sequentially, from the first auto-resizable column to the last auto-resizable column; proceed to the next column when the current column has reached its minimum or maximum size.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewLastColumnOnlyAutoresizingStyle`
Autoresize only the last table column.

When that table column can no longer be resized, stop autoresizing. Normally you should use one of the sequential autoresizing modes instead.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewFirstColumnOnlyAutoresizingStyle`
Autoresize only the first table column.

When that table column can no longer be resized, stop autoresizing. Normally you should use one of the sequential autoresizing modes instead.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

Declared In
`NSTableView.h`

Selection Styles

The following constants specify the selection highlight styles. These constants are used by `selectionHighlightStyle` (page 48) and `setSelectionHighlightStyle:` (page 59).

```
enum {
    NSTableViewSelectionHighlightStyleRegular = 0,
    NSTableViewSelectionHighlightStyleSourceList = 1,
};
typedef NSInteger NSTableViewSelectionHighlightStyle;
```

Constants
`NSTableViewSelectionHighlightStyleRegular`
The regular highlight style of NSTableView. On Mac OS X v10.5 a light blue (returned by sending `NSColor` a `alternateSelectedControlColor` message) or light gray color (returned by sending NSColor a `secondarySelectedControlColor` message).

Available in Mac OS X v10.5 and later.

Declared in `NSTableView.h`.

`NSTableViewSelectionHighlightStyleSourceList`
The source list style of NSTableView. On 10.5, a light blue gradient is used to highlight selected rows.

Note: When using this style, cell subclasses that implement `drawsBackground` must set the value to `NO`. Otherwise, the cells will draw over the tableview's highlighting.

Available in Mac OS X v10.5 and later.

Declared in `NSTableView.h`.

Declared In
`NSTableView.h`

Notifications

NSTableViewColumnDidMoveNotification

Posted whenever a column is moved by user action in an `NSTableView` object. The notification object is the table view in which a column moved. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldColumn"`	An `NSNumber` object containing the integer value of the column's original index.
`@"NSNewColumn"`	An `NSNumber` object containing the integer value of the column's present index.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- moveColumn:toColumn:` (page 38)

Declared In
`NSTableView.h`

NSTableViewColumnDidResizeNotification

Posted whenever a column is resized in an `NSTableView` object. The notification object is the table view in which a column was resized. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSTableColumn"`	The column that was resized.
`@"NSOldWidth"`	An NSNumber containing the integer value of the column's original width.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

NSTableViewSelectionDidChangeNotification

Posted after an `NSTableView` object's selection changes. The notification object is the table view whose selection changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

NSTableViewSelectionIsChangingNotification

Posted as an `NSTableView` object's selection changes (while the mouse button is still down). The notification object is the table view whose selection is changing. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

Deprecated NSTableView Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.3

drawsGrid

Returns a Boolean value that indicates whether the receiver draws a grid. (Deprecated in Mac OS X v10.3. Use `gridStyleMask` (page 35) instead.)

```
- (BOOL)drawsGrid
```

Availability
Deprecated in Mac OS X v10.3.

Declared In
`NSTableView.h`

selectColumn:byExtendingSelection:

Selects a column at a given index, optionally extending any existing selection. (Deprecated in Mac OS X v10.3. Use `selectColumnIndexes:byExtendingSelection:` (page 46) instead.)

```
- (void)selectColumn:(NSInteger)columnIndex
    byExtendingSelection:(BOOL)flag
```

Availability
Deprecated in Mac OS X v10.3.

See Also
- `allowsMultipleSelection` (page 22)
- `allowsColumnSelection` (page 21)
- `deselectColumn:` (page 29)
- `selectedColumn` (page 47)
- `selectRow:byExtendingSelection:` (page 82)

Declared In
`NSTableView.h`

selectedColumnEnumerator

This method has been deprecated. (Deprecated in Mac OS X v10.3. Use `selectedColumnIndexes` (page 47) instead.)

```
- (NSEnumerator *)selectedColumnEnumerator
```

Availability
Deprecated in Mac OS X v10.3.

See Also
- `numberOfSelectedColumns` (page 40)
- `selectedColumn` (page 47)
- `selectedRowEnumerator` (page 82)

Declared In
`NSTableView.h`

selectedRowEnumerator

This method has been deprecated. (Deprecated in Mac OS X v10.3. Use `selectedRowIndexes` (page 48) instead.)

```
- (NSEnumerator *)selectedRowEnumerator
```

Availability
Deprecated in Mac OS X v10.3.

See Also
- `numberOfSelectedRows` (page 40)
- `selectedRow` (page 47)
- `selectedColumnEnumerator` (page 82)

Declared In
`NSTableView.h`

selectRow:byExtendingSelection:

Selects a row at a given index, optionally extending any existing selection. (Deprecated in Mac OS X v10.3. Use `selectRowIndexes:byExtendingSelection:` (page 49) instead.)

```
- (void)selectRow:(NSInteger)rowIndex
    byExtendingSelection:(BOOL)flag
```

Availability
Deprecated in Mac OS X v10.3.

See Also
- `allowsMultipleSelection` (page 22)
- `deselectRow:` (page 30)
- `selectedRow` (page 47)
- `selectColumn:byExtendingSelection:` (page 81)

Related Sample Code
EnhancedAudioBurn
OpenGLCompositorLab
WhackedTV

Declared In
NSTableView.h

setDrawsGrid:

Sets whether the receiver draws a grid. (Deprecated in Mac OS X v10.3. Use `setGridStyleMask:` (page 57) instead.)

```
- (void)setDrawsGrid:(BOOL)flag
```

Availability
Deprecated in Mac OS X v10.3.

Declared In
NSTableView.h

Deprecated in Mac OS X v10.4 and later

dragImageForRows:event:dragImageOffset:

Computes and returns an image to use for dragging. (Deprecated in Mac OS X v10.4 and later. Use `dragImageForRowsWithIndexes:tableColumns:event:offset:` (page 31) instead.)

```
- (NSImage *)dragImageForRows:(NSArray *)dragRows
    event:(NSEvent *)dragEvent
    dragImageOffset:(NSPointPointer)dragImageOffset
```

Discussion
Override this to return a custom image. *dragRows* represents the rows participating in the drag. *dragEvent* is a reference to the mouse-down event that began the drag. *dragImageOffset* is an in/out parameter.

This method is called with *dragImageOffset* set to `NSZeroPoint`, but it can be modified to reposition the returned image. A *dragImageOffset* of `NSZeroPoint` will cause the image to be centered under the cursor.

Availability
Deprecated in Mac OS X v10.4 and later.

Declared In
NSTableView.h

setAutoresizesAllColumnsToFit:

Controls whether the receiver proportionally resizes its columns to fit when its superview's frame changes. (Deprecated in Mac OS X v10.4 and later. Use `setColumnAutoresizingStyle:` (page 53) instead.)

```
- (void)setAutoresizesAllColumnsToFit:(BOOL)flag
```

Discussion
If *flag* is YES, the difference in width is distributed among the receiver's table columns; if *flag* is NO, only the last column is resized to fit.

To preserve compatibility this method sets the autoresizing style to NSTableViewUniformColumnAutoresizingStyle, if *flag* is YES. Otherwise the autoresizing style is set to NSTableViewLastColumnOnlyAutoresizingStyle.

Availability
Deprecated in Mac OS X v10.4 and later.

See Also
- setColumnAutoresizingStyle: (page 53)

Declared In
NSTableView.h

Deprecated in Mac OS X v10.4

autoresizesAllColumnsToFit

Returns YES if the receiver proportionally resizes its columns to fit when its superview's frame changes, NO if it only resizes the last column. (Deprecated in Mac OS X v10.4. Use columnAutoresizingStyle (page 26) instead.)

```
- (BOOL)autoresizesAllColumnsToFit
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- columnAutoresizingStyle (page 26)
- setColumnAutoresizingStyle: (page 53)

Declared In
NSTableView.h

Deprecated in Mac OS X v10.5

columnsInRect:

Returns a range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of a given rectangle. (Deprecated in Mac OS X v10.5.)

```
- (NSRange)columnsInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the coordinate system of the receiver.

Return Value

A range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of *aRect*. If the width or height of *aRect* is 0, returns an NSRange whose length is 0.

Discussion

The location of the range is the first such column's index, and the length is the number of columns that lie in *aRect*.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

See Also

- rowsInRect: (page 44)

Declared In

NSTableView.h

Document Revision History

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

Date	Notes
2009-04-30	Updated availability and deprecation information.
2007-10-31	Clarified the ownership semantics for the setDelegate: and setDataSource: methods.
2007-07-24	Updated to include API introduced in Mac OS X v10.5. Fixed minor bugs.
2006-06-28	Enhanced the description of data source methods.
2006-05-23	Included an explicit definition of the NSTableViewColumnAutoresizingStyle and NSTableViewDropOperation types.
	First publication of this content as a separate document.

Document Revision History

Index

A

addTableColumn: instance method 20
allowsColumnReordering instance method 20
allowsColumnResizing instance method 21
allowsColumnSelection instance method 21
allowsEmptySelection instance method 22
allowsMultipleSelection instance method 22
allowsTypeSelect instance method 23
autoresizesAllColumnsToFit instance method 84
Autoresizing Styles 77
autosaveName instance method 23
autosaveTableColumns instance method 23

B

backgroundColor instance method 24

C

canDragRowsWithIndexes:atPoint: instance method 24
clickedColumn instance method 25
clickedRow instance method 25
columnAtPoint: instance method 26
columnAutoresizingStyle instance method 26
columnIndexesInRect: instance method 26
columnsInRect: instance method 84
columnWithIdentifier: instance method 27
cornerView instance method 27

D

dataSource instance method 28
delegate instance method 28
deselectAll: instance method 28
deselectColumn: instance method 29
deselectRow: instance method 30
doubleAction instance method 30
dragImageForRows:event:dragImageOffset: instance method 83
dragImageForRowsWithIndexes:tableColumns:event:offset: instance method 31
drawBackgroundInClipRect: instance method 31
drawGridInClipRect: instance method 31
drawRow:clipRect: instance method 32
drawsGrid instance method 81
Drop Operations 76

E

editColumn:row:withEvent:select: instance method 32
editedColumn instance method 33
editedRow instance method 33

F

frameOfCellAtColumn:row: instance method 33

G

Grid styles 76
gridColor instance method 34
gridStyleMask instance method 35

H

headerView instance method 35
highlightedTableColumn instance method 35
highlightSelectionInClipRect: instance method 36

I

indicatorImageInTableColumn: instance method 36
intercellSpacing instance method 37
isColumnSelected: instance method 37
isRowSelected: instance method 37

M

moveColumn:toColumn: instance method 38

N

noteHeightOfRowsWithIndexesChanged: instance method 38
noteNumberOfRowsChanged instance method 39
NSTableViewColumnDidMoveNotification notification 79
NSTableViewColumnDidResizeNotification notification 79
NSTableViewDropAbove constant 76
NSTableViewDropOn constant 76
NSTableViewFirstColumnOnlyAutoresizingStyle constant 78
NSTableViewGridNone constant 76
NSTableViewLastColumnOnlyAutoresizingStyle constant 78
NSTableViewNoColumnAutoresizing constant 77
NSTableViewReverseSequentialColumnAutoresizing-Style constant 77
NSTableViewSelectionDidChangeNotification notification 79
NSTableViewSelectionHighlightStyleRegular constant 78
NSTableViewSelectionHighlightStyleSourceList constant 78
NSTableViewSelectionIsChangingNotification notification 80
NSTableViewSequentialColumnAutoresizingStyle constant 77
NSTableViewSolidHorizontalGridLineMask constant 77
NSTableViewSolidVerticalGridLineMask constant 76
NSTableViewUniformColumnAutoresizingStyle constant 77
numberOfColumns instance method 39
numberOfRows instance method 40
numberOfSelectedColumns instance method 40
numberOfSelectedRows instance method 40

P

preparedCellAtColumn:row: instance method 41

R

rectOfColumn: instance method 41
rectOfRow: instance method 42
reloadData instance method 42
removeTableColumn: instance method 43
rowAtPoint: instance method 43
rowHeight instance method 44
rowsInRect: instance method 44

S

scrollColumnToVisible: instance method 45
scrollRowToVisible: instance method 45
selectAll: instance method 45
selectColumn:byExtendingSelection: instance method 81
selectColumnIndexes:byExtendingSelection: instance method 46
selectedColumn instance method 47
selectedColumnEnumerator instance method 82
selectedColumnIndexes instance method 47
selectedRow instance method 47
selectedRowEnumerator instance method 82
selectedRowIndexes instance method 48
Selection Styles 78
selectionHighlightStyle instance method 48
selectionShouldChangeInTableView: <NSObject> delegate method 66
selectRow:byExtendingSelection: instance method 82
selectRowIndexes:byExtendingSelection: instance method 49
setAllowsColumnReordering: instance method 49
setAllowsColumnResizing: instance method 49
setAllowsColumnSelection: instance method 50
setAllowsEmptySelection: instance method 50
setAllowsMultipleSelection: instance method 51
setAllowsTypeSelect: instance method 51
setAutoresizesAllColumnsToFit: instance method 83
setAutosaveName: instance method 52
setAutosaveTableColumns: instance method 52
setBackgroundColor: instance method 53
setColumnAutoresizingStyle: instance method 53
setCornerView: instance method 53

setDataSource: instance method 54
setDelegate: instance method 54
setDoubleAction: instance method 55
setDraggingSourceOperationMask:forLocal: instance method 55
setDrawsGrid: instance method 83
setDropRow:dropOperation: instance method 56
setGridColor: instance method 56
setGridStyleMask: instance method 57
setHeaderView: instance method 57
setHighlightedTableColumn: instance method 58
setIndicatorImage:inTableColumn: instance method 58
setIntercellSpacing: instance method 58
setRowHeight: instance method 59
setSelectionHighlightStyle: instance method 59
setSortDescriptors: instance method 60
setUsesAlternatingRowBackgroundColors: instance method 60
setVerticalMotionCanBeginDrag: instance method 60
sizeLastColumnToFit instance method 61
sizeToFit instance method 61
sortDescriptors instance method 62

T

tableColumns instance method 62
tableColumnWithIdentifier: instance method 62
tableView:dataCellForTableColumn:row: <NSObject> delegate method 66
tableView:didClickTableColumn: <NSObject> delegate method 67
tableView:didDragTableColumn: <NSObject> delegate method 67
tableView:heightOfRow: <NSObject> delegate method 68
tableView:isGroupRow: <NSObject> delegate method 68
tableView:mouseDownInHeaderOfTableColumn: <NSObject> delegate method 68
tableView:nextTypeSelectMatchFromRow:toRow:forString: <NSObject> delegate method 69
tableView:selectionIndexesForProposedSelection: <NSObject> delegate method 69
tableView:shouldEditTableColumn:row: <NSObject> delegate method 70
tableView:shouldSelectRow: <NSObject> delegate method 70
tableView:shouldSelectTableColumn: <NSObject> delegate method 71
tableView:shouldShowCellExpansionForTableColumn:row: <NSObject> delegate method 71
tableView:shouldTrackCell:forTableColumn:row: <NSObject> delegate method 72
tableView:shouldTypeSelectForEvent:withCurrentSearchString: <NSObject> delegate method 72
tableView:toolTipForCell:rect:tableColumn:row:mouseLocation: <NSObject> delegate method 73
tableView:typeSelectStringForTableColumn:row: <NSObject> delegate method 73
tableView:willDisplayCell:forTableColumn:row: <NSObject> delegate method 74
tableViewColumnDidMove: <NSObject> delegate method 74
tableViewColumnDidResize: <NSObject> delegate method 75
tableViewSelectionDidChange: <NSObject> delegate method 75
tableViewSelectionIsChanging: <NSObject> delegate method 75
textDidBeginEditing: instance method 63
textDidChange: instance method 63
textDidEndEditing: instance method 64
textShouldBeginEditing: instance method 64
textShouldEndEditing: instance method 64
tile instance method 65

U

usesAlternatingRowBackgroundColors instance method 65

V

verticalMotionCanBeginDrag instance method 65