NSBrowser

Inherits From:	NSControl : NSView : NSResponder : NSObject
Conforms To:	NSCoding (NSResponder) NSObject (NSObject)
Declared In:	AppKit/NSBrowser.h

Class Description

NSBrowser provides a user interface for displaying and selecting items from a list of data, or from hierarchically organized lists of data such as directory paths. When working with a hierarchy of data, the levels are displayed in columns, which are numbered from left to right, beginning with 0. Each column consists of an NSScrollView containing an NSMatrix filled with NSBrowserCells. NSBrowser relies on a delegate to provide the data in its NSBrowserCells. See the NSBrowserCell class description for more on its implementation.

Browser Selection

An entry in an NSBrowser's column can be either a branch node (such as a directory) or a leaf node (such as a file). When the user selects a single branch node entry in a column, the NSBrowser sends itself the **addColumn** message, which messages its delegate to load the next column. The user's selection can be represented as a character string; if the selection is hierarchical (for example, a filename within a directory), each component of the path to the selected node is separated by "/". To use some other character as the delimiter, invoke **setPathSeparator:**.

An NSBrowser can be set to allow selection of multiple entries in a column, or to limit selection to a single entry. When set for multiple selection, it can also be set to limit multiple selection to leaf nodes only, or to allow selection of both types of nodes together.

As a subclass of NSControl, NSBrowser has a target object and action message. Each time the user selects one or more entries in a column, the action message is sent to the target. NSBrowser also adds an action to be sent when the user double-clicks on an entry, which allows the user to select items without any action being taken, and then double-click to invoke some useful action such as opening a file.

User Interface Features

The user interface features of an NSBrowser can be changed in a number of ways. The NSBrowser may or may not have a horizontal scroller. (The NSBrowser's columns, by contrast, always have vertical scrollers—although a scroller's buttons and knob might be invisible if the column doesn't contain many entries.) You generally shouldn’t create an NSBrowser without a horizontal scroller; if you do, you must make sure the bounds rectangle of the NSBrowser is wide enough that all the columns can be displayed. An NSBrowser’s columns may be bordered and titled, bordered and untitled, or unbordered and untitled. A column’s title may be taken from the selected entry in the column to its left, or may be provided explicitly by the NSBrowser or its delegate.

NSBrowser’s Delegate

NSBrowser requires a delegate to provide it with data to display. The delegate is responsible for providing the data and for setting each item as a branch or leaf node, enabled or disabled. It can also receive notification of events like scrolling and requests for validation of columns that may have changed.

You can implement one of two delegate types: active or passive. An active delegate creates a column’s rows (that is, the NSBrowserCells) itself, while a passive one leaves that job to the NSBrowser. Normally, passive delegates are preferable, because they’re easier to implement. An active delegate must implement **browser:createRowsForColumn:inMatrix:** to create the rows of the specified column. A passive delegate, on the other hand, must implement **browser:numberOfRowsInColumn:** to let the NSBrowser know how many rows to create. These two methods are mutually exclusive; you can implement one or the other, but not both. (The NSBrowser ascertains what type of delegate it has by which method the delegate responds to.)

Both types of delegate implement **browser:willDisplayCell:atRow:column:** to set up state (such as the cell’s string value and whether the cell is a leaf or a branch) before an individual cell is displayed. (This delegate method doesn’t need to invoke NSBrowserCell’s **setLoaded:** method, because the NSBrowser can determine that state by itself.) An active delegate can instead set all the cells’ state at the time the cells are created, in which case it doesn’t need to implement **browser:willDisplayCell:atRow:column:**. However, a passive delegate must always implement this method.

Method Types

Setting component classes	+ cellClass
	– setCellClass:
	– cellPrototype
	– setCellPrototype:
	– matrixClass
	– setMatrixClass:

Getting matrices, cells, and rows	– selectedCell – selectedCellInColumn: – selectedCells – selectAll: – selectedRowInColumn: – selectRow:inColumn: – loadedCellAtRow:column: – matrixInColumn:
Getting and setting paths	– path – setPath: – pathToColumn: – pathSeparator – setPathSeparator:
Manipulating columns	– addColumn – displayAllColumns – displayColumn: – columnOfMatrix: – selectedColumn – lastColumn – setLastColumn: – firstVisibleColumn – numberOfVisibleColumns – lastVisibleColumn – validateVisibleColumns
Loading columns	– isLoaded – loadColumnZero – reloadColumn:
Setting selection characteristics	– allowsBranchSelection – setAllowsBranchSelection: – allowsEmptySelection – setAllowsEmptySelection: – allowsMultipleSelection – setAllowsMultipleSelection:

Setting column characteristics	– reusesColumns – setReusesColumns: – maxVisibleColumns – setMaxVisibleColumns: – minColumnWidth – setMinColumnWidth: – separatesColumns – setSeparatesColumns: – takesTitleFromPreviousColumn – setTakesTitleFromPreviousColumn:
Manipulating column titles	– titleOfColumn: – setTitle:ofColumn: – isTitled – setTitled: – drawTitle:inRect:ofColumn: – titleHeight – titleFrameOfColumn:
Scrolling an NSBrowser	– scrollColumnToVisible: – scrollColumnsLeftBy: – scrollColumnsRightBy: – updateScroller – scrollViaScroller:
Showing a horizontal scroller	– hasHorizontalScroller – setHasHorizontalScroller:
Setting the behavior of arrow keys	– acceptsArrowKeys – setAcceptsArrowKeys: – sendsActionOnArrowKeys – setSendsActionOnArrowKeys:
Getting column frames	– frameOfColumn: – frameOfInsideOfColumn:
Arranging browser components	– tile
Setting the delegate	– delegate – setDelegate:
Target and action	– doubleAction – setDoubleAction: – sendAction
Event handling	– doClick: – doDoubleClick:

Class Methods

cellClass

+ (Class)**cellClass**

Returns the NSBrowserCell class (regardless of whether a **setCellClass:** message has been sent to a particular instance)

See also: – **cellPrototype**,– s**etCellPrototype:**

Instance Methods

acceptsArrowKeys

– (BOOL)**acceptsArrowKeys**

Returns YES if the arrow keys are enabled.

See also: – **setAcceptsArrowKeys:**

addColumn

– (void)**addColumn**

Adds a column to the right of the last column.

See also: – **columnOfMatrix:**, – **displayColumn:**, – **selectedColumn**

allowsBranchSelection

– (BOOL)**allowsBranchSelection**

Returns whether the user can select branch items when multiple selection is enabled.

See also: – **setAllowsBranchSelection:**

allowsEmptySelection

– (BOOL)**allowsEmptySelection**

Returns whether there can be nothing selected.

See also: – **setAllowsEmptySelection:**

allowsMultipleSelection

– (BOOL)**allowsMultipleSelection**

Returns whether the user can select multiple items.

See also: – **setAllowsMultipleSelection:**

cellPrototype

– (id)**cellPrototype**

Returns the NSBrowser’s prototype NSCell.

See also: – **setCellPrototype:**, – **setCellClass:**

columnOfMatrix:

– (int)**columnOfMatrix:**(NSMatrix *)*matrix*

Returns the column number in which *matrix* is located.

See also: – **matrixInColumn:**

delegate

– (id)**delegate**

Returns the NSBrowser’s delegate.

See also: – **setDelegate:**

displayAllColumns

– (void)**displayAllColumns**

Updates the NSBrowser to display all loaded columns.

See also: – **addColumn**, – **validateVisibleColumns**

displayColumn:

– (void)**displayColumn:**(int)*column*

Updates the NSBrowser to display the column with the given index.

See also: – **addColumn**, – **validateVisibleColumns**

doClick:

– (void)**doClick:**(id)*sender*

Responds to (single) mouse clicks in a column of the NSBrowser.

See also: – **sendAction**

doDoubleClick:

– (void)**doDoubleClick:**(id)*sender*

Responds to double-clicks in a column of the NSBrowser.

See also: – **setDoubleAction:**

doubleAction

– (SEL)**doubleAction**

Returns the NSBrowser's double-click action method.

See also: – **setDoubleAction:**

drawTitle:inRect:ofColumn:

– (void)**drawTitle:**(NSString *)*title* **inRect:**(NSRect)*aRect* **ofColumn:**(int)*column*

Draws the title for the column at index column.

See also: – **titleFrameOfColumn:**, – **titleHeight**

firstVisibleColumn

– (int)**firstVisibleColumn**

Returns the index of the first visible column.

See also: – **lastVisibleColumn**, – **numberOfVisibleColumns**

frameOfColumn:

– (NSRect)**frameOfColumn:**(int)*column*

Returns the rectangle containing the column at index column.

frameOfInsideOfColumn:

– (NSRect)**frameOfInsideOfColumn:**(int)*column*

Returns the rectangle containing the column at index column, not including borders.

hasHorizontalScroller

– (BOOL)**hasHorizontalScroller**

Returns whether an NSScroller is used to scroll horizontally.

See also: – **setHasHorizontalScroller:**

isLoaded

– (BOOL)**isLoaded**

Returns whether column zero is loaded.

See also: – **loadColumnZero**, – **reloadColumn:**

isTitled

– (BOOL)**isTitled**

Returns whether columns display titles.

See also: – **setTitled:**

lastColumn

– (int)**lastColumn**

Returns the index of the last column loaded.

See also: – **selectedColumn**, – **setLastColumn:**

lastVisibleColumn

– (int)**lastVisibleColumn**

Returns the index of the last visible column.

See also: – **firstVisibleColumn**, – **numberOfVisibleColumns**

loadColumnZero

– (void)**loadColumnZero**

Loads column zero; unloads previously loaded columns.

See also: – **isLoaded**, – **reloadColumn:**

loadedCellAtRow:column:

– (id)**loadedCellAtRow:**(int)*row* **column:**(int)*column*

Loads if necessary and returns the NSCell at *row* in *column*.

See also: – **selectedCellInColumn:**

matrixClass

– (Class)**matrixClass**

Returns the class of NSMatrix used in the NSBrowser's columns.

See also: – **setMatrixClass:**

matrixInColumn:

– (NSMatrix *)**matrixInColumn:**(int)*column*

Returns the matrix located in the column identified by index *column*.

maxVisibleColumns

– (int)**maxVisibleColumns**

Returns the maximum number of visible columns.

See also: – **setMaxVisibleColumns:**

minColumnWidth

– (float)**minColumnWidth**

Returns the minimum column width in pixels.

See also: – **setMinColumnWidth:**

numberOfVisibleColumns

– (int)**numberOfVisibleColumns**

Returns the number of columns visible.

See also: – **validateVisibleColumns**

path

– (NSString *)**path**

Returns the browser’s current path.

See also: – **setPath:**

pathSeparator

– (NSString *)**pathSeparator**

Returns the path separator. The default is “/”.

See also: – **setPathSeparator:**

pathToColumn:

– (NSString *)**pathToColumn:**(int)*column*

Returns a string representing the path from the first column to the column at index *column*.

See also: – **path**, – **setPath:**

reloadColumn:

– (void)**reloadColumn:**(int)*column*

Reloads *column* if it is loaded; sets it as the last column.

See also: – **isLoaded**, – **loadColumnZero**

reusesColumns

– (BOOL)**reusesColumns**

Returns YES if NSMatrix objects aren't freed when their columns are unloaded.

See also: – **setReusesColumns**

scrollColumnToVisible:

– (void)**scrollColumnToVisible:**(int)*column*

Scrolls to make the column at index column visible.

See also: – **scrollViaScroller:**, – **updateScroller**

scrollColumnsLeftBy:

– (void)**scrollColumnsLeftBy:**(int)*shiftAmount*

Scrolls columns left by shiftAmount columns.

See also: – **scrollViaScroller:**, – **updateScroller**

scrollColumnsRightBy:

– (void)**scrollColumnsRightBy:**(int)*shiftAmount*

Scrolls columns right by shiftAmount columns.

See also: – **scrollViaScroller:**, – **updateScroller**

scrollViaScroller:

– (void)**scrollViaScroller:**(NSScroller *)*sender*

Scrolls columns left or right based on an NSScroller.

See also: – **updateScroller**

selectAll:

– (void)**selectAll:**(id)*sender*

Selects all NSCells in the last column of the NSBrowser.

See also: – **selectedCell**, – **selectedCells**, – **selectedColumn**

selectRow:inColumn:

– (void)**selectRow:**(int)*row* **inColumn:**(int)*column*

Selects the cell at index *row* in the column identified by index *column*.

See also: – **loadedCellAtRow:column:**, – **selectedRowInColumn:**

selectedCell

– (id)**selectedCell**

Returns the last (rightmost and lowest) selected NSCell.

See also: – **loadedCellAtRow:column:**, – **selectedCell**s, – **selectRow:inColumn:**

selectedCellInColumn:

– (id)**selectedCellInColumn:**(int)*column*

Returns the last (lowest) NSCell that's selected in column.

See also: – **loadedCellAtRow:column:**, – **selectedCell**, – **selectedRowInColumn:**

selectedCells

– (NSArray *)**selectedCells**

Returns all cells selected in the rightmost column.

See also: – **selectAll:**, – **selectedCell**

selectedColumn

– (int)**selectedColumn**

Returns the index of the last column with a selected item.

See also: – **columnOfMatrix**, – **selectAll:**

selectedRowInColumn:

– (int)**selectedRowInColumn:**(int)*column*

Returns the row index of the selected cell in the column specified by index *column*.

See also: – **loadedCellAtRow:column:**, – **selectedCell**, – **selectedCellInColumn:**

sendAction

– (BOOL)**sendAction**

Sends the action message to the target. Returns YES upon success, NO if no target for the message could be found.

sendsActionOnArrowKeys

– (BOOL)**sendsActionOnArrowKeys**

Returns NO if pressing an arrow key only scrolls the browser, YES if it also sends the action message specified by **setAction:**.

See also: – **acceptsArrowKeys**, – **setSendsActionOnArrayKeys:**

separatesColumns

– (BOOL)**separatesColumns**

Returns whether columns are separated by bezeled borders.

See also: – **setSeparatesColumns:**

setAcceptsArrowKeys:

– (void)**setAcceptsArrowKeys:**(BOOL)*flag*

Enables or disables the arrow keys as used for navigating within and between browsers.

See also: – **acceptsArrowKeys**, – **sendsActionOnArrowKeys**

setAllowsBranchSelection:

– (void)**setAllowsBranchSelection:**(BOOL)*flag*

Sets whether the user can select branch items when multiple selection is enabled.

See also: – **allowsBranchSelection**

setAllowsEmptySelection:

– (void)**setAllowsEmptySelection:**(BOOL)*flag*

Sets whether there can be nothing selected.

See also: – **allowsEmptySelection**

setAllowsMultipleSelection:

– (void)**setAllowsMultipleSelection:**(BOOL)*flag*

Sets whether the user can select multiple items.

See also: – **allowsMultipleSelection**

setCellClass:

– (void)**setCellClass:**(Class)*factoryId*

Sets the class of NSCell used in the columns of the NSBrowser.

See also: + **cellClass:**, – **cellPrototype**

setCellPrototype:

– (void)**setCellPrototype:**(NSCell *)*aCell*

Sets the NSCell instance copied to display items in the columns of NSBrowser.

See also: + **cellClass:**, – **cellPrototype**

setDelegate:

– (void)**setDelegate:**(id)*anObject*

Sets the NSBrowser's delegate to *anObject*. Raises NSBrowserIllegalDelegateException if the delegate specified by *anObject* doesn't respond to **browser:willDisplayCell:atRow:column:** and either of the methods **browser:numberOfRowsInColumn:** or **browser:createRowsForColumn:inMatrix:**.

See also: – **delegate**

setDoubleAction:

– (void)**setDoubleAction:**(SEL)*aSelector*

Sets the NSBrowser's double-click action to *aSelector*.

See also: – **doubleAction**, – **sendAction**

setHasHorizontalScroller:

– (void)**setHasHorizontalScroller:**(BOOL)*flag*

Sets whether an NSScroller is used to scroll horizontally.

See also: – **hasHorizontalScroller**

setLastColumn:

– (void)**setLastColumn:**(int)*column*

Sets the last column to column.

See also: – **lastColumn**, – **lastVisibleColumn**

setMatrixClass:

– (void)**setMatrixClass:**(Class)*factoryId*

Sets the matrix class (NSMatrix or an NSMatrix subclass) used in the NSBrowser’s columns.

See also: – **matrixClass**

setMaxVisibleColumns:

– (void)**setMaxVisibleColumns:**(int)*columnCount*

Sets the maximum number of columns displayed.

See also: – **maxVisibleColumns**

setMinColumnWidth:

– (void)**setMinColumnWidth:**(float)*columnWidth*

Sets the minimum column width in pixels.

See also: – **minColumnWidth**

setPath:

– (BOOL)**setPath:**(NSString *)*path*

Parses *path* and selects corresponding items in the NSBrowser columns.

See also: – **path**, – **pathToColumn:**

setPathSeparator:

– (void)**setPathSeparator:**(NSString *)*newString*

Sets the path separator to *newString*.

See also: – **pathSeparator**

setReusesColumns:

– (void)**setReusesColumns:**(BOOL)*flag*

If *flag* is YES, prevents NSMatrix objects from being freed when their columns are unloaded, so they can be reused.

See also: – **resusesColumns**

setSendsActionOnArrowKeys:

– (void)**setSendsActionOnArrowKeys:**(BOOL)*flag*

Sets whether pressing an arrow key will cause the action message to be sent (in addition to causing scrolling).

See also: – **sendsActionOnArrowKeys**

setSeparatesColumns:

– (void)**setSeparatesColumns:**(BOOL)*flag*

Sets whether to separate columns with bezeled borders.

See also: – **separatesColumns**

setTakesTitleFromPreviousColumn:

– (void)**setTakesTitleFromPreviousColumn:**(BOOL)*flag*

Sets whether the title of a column is set to the string value of the selected NSCell in the previous column.

See also: – **takesTitleFromPreviousColumn:**

setTitle:ofColumn:

– (void)**setTitle:**(NSString *)*aString* **ofColumn:**(int)*column*

Sets the title of the column at index *column* to *aString*.

See also: – **titleOfColumn:**

setTitled:

– (void)**setTitled:**(BOOL)*flag*

Sets whether columns display titles.

See also: – **isTitled**

takesTitleFromPreviousColumn

– (BOOL)**takesTitleFromPreviousColumn**

Returns YES if the title of a column is set to the string value of the selected NSCell in the previous column.

See also: – **setTakesTitleFromPreviousColumn:**

tile

– (void)**tile**

Adjusts the various subviews of NSBrowser—scrollers, columns, titles, and so on—without redrawing. Your code shouldn’t send this message. It’s invoked any time the appearance of the NSBrowser changes.

titleFrameOfColumn:

– (NSRect)**titleFrameOfColumn:**(int)*column*

Returns the bounds of the title frame for the column at index *column*.

See also: – **drawTitle:inRect:ofColumn:**

titleHeight

– (float)**titleHeight**

Returns the height of column titles.

See also: – **drawTitle:inRect:ofColumn:**

titleOfColumn:

– (NSString *)**titleOfColumn:**(int)*column*

Returns the title displayed for the column at index *column*.

See also: – **setTitle:ofColumn:**

updateScroller

– (void)**updateScroller**

Updates the horizontal scroller to reflect column positions.

See also: – **scrollViaScroller:**

validateVisibleColumns

– (void)**validateVisibleColumns**

Invokes delegate method **browser:isColumnValid:** for visible columns.

See also: – **numberOfVisibleColumn**

Methods Implemented By the Delegate

browser:createRowsForColumn:inMatrix:

– (void)**browser:**(NSBrowser *)*sender*
  createRowsForColumn:(int)*column*
  inMatrix:(NSMatrix *)*matrix*

Creates a row in matrix for each row of data to be displayed in column of the browser. Either this method or **browser:numberOfRowsInColumn:** must be implemented, but not both (or an NSBrowserIllegalDelegateException will be raised).

See also: – **browser:willDisplayCell:atRow:column:**

browser:isColumnValid:

– (BOOL)**browser:**(NSBrowser *)*sender* **isColumnValid:**(int)*column*

Returns whether the contents of the specified column are valid. If NO is returned, *sender* reloads the column. This method is invoked in response to **validateVisibleColumns** being sent to *sender*.

browser:numberOfRowsInColumn:

– (int)**browser:**(NSBrowser *)*sender* **numberOfRowsInColumn:**(int)*column*

Returns the number of rows of data in the column at index *column*. Either this method or **browser:createRowsForColumn:inMatrix:** must be implemented, but not both.

See also: – **browser:willDisplayCell:atRow:column:**

browser:selectCellWithString:inColumn:

– (BOOL)**browser:**(NSBrowser *)*sender*
    selectCellWithString:(NSString *)*title*
    inColumn:(int)*column*

Asks the delegate to select the NSCell with title *title* in the column at index *column*. If the delegate returns NO, the NSCell is not selected.

See also: – **selectedCellInColumn:**

browser:selectRow:inColumn:

– (BOOL)**browser:**(NSBrowser *)*sender*
    selectRow:(int)*row*
    inColumn:(int)*column*

Asks the delegate to select the NSCell at row *row* in the column at index *column*. If the delegate returns NO, the NSCell is not selected.

See also: – **selectedRowInColumn:**, – **selectRow:inColumn:**

browser:titleOfColumn:

– (NSString *)**browser:**(NSBrowser *)*sender* **titleOfColumn:**(int)*column*

Asks the delegate for the title to display above the column at index *column*.

See also: – **setTitle:ofColumn:**, – **titleOfColumn:**

browser:willDisplayCell:atRow:column:

– (void)**browser:**(NSBrowser *)*sender*
willDisplayCell:(id)*cell*
atRow:(int)*row*
column:(int)*column*

Notifies the delegate before the NSBrowser displays the specified *cell* at *row* in *column*. The delegate should set any state necessary for the correct display of the cell.

See also: – **browser:createRowsForColumn:inMatrix:**, – **browser:numberOfRowsInColumn:**

browserDidScroll:

– (void)**browserDidScroll:**(NSBrowser *)*sender*

Notifies the delegate when the NSBrowser has scrolled.

browserWillScroll:

– (void)**browserWillScroll:**(NSBrowser *)*sender*

Notifies the delegate when the NSBrowser will scroll.