TABLEBROWSER

TableBrowser implements a simple mechanism for building applications that browse certain kinds of tabular data. It supplies a set of basic functions that maintain the window, allowing scrolling and selection of items; the application defines the items, how they print, and any higher-level operations to perform on them. FileBrowser is an example of an application built upon TableBrowser.

Installation

Load TABLEBROWSER.LCOM from the library.

During program development, you also need to load the declarations file TABLEBROWSERDECLS; this file is not needed when running a compiled application. The file manager coms for the typical application file should thus include the two commands

```
(FILES (SYSLOAD) TABLEBROWSER)
(DECLARE: EVAL@COMPILE DONTCOPY
     (FILES (SOURCE) TABLEBROWSERDECLS)
```

User Interface

An instance of a TableBrowser application is a window displaying a browser. The browser consists of an ordered set of items. Each item contains an item of application data and some bookkeeping information. TableBrowser maintains the display, supplying generic scrolling, painting, reshaping and selection mechanisms; the application supplies TableBrowser with a set of methods for displaying the contents of an item, what to do when an item is copy selected, etc.

Review the user interface of the FileBrowser module to get an idea of the selection mechanism and the sort of functionality available from TableBrowser. Briefly, items are selected by clicking in the browser window — left to select a single item, middle to add an item to the current selection, control-middle to remove an item, right to extend the selection, control-right to extend the selection, including deleted items.

The typical application creates a browser window, fills it with items, and attaches one or more menus to the window. The menu contains commands that may perform application-specific operations, usually on the set of currently selected items.

Records

There are two important records, TABLEITEM and TABLEBROWSER. These records are declared in the file TABLEBROWSERDECLS.

TABLEITEM Record

Each piece of application data (a line in the browser) is encapsulated in an instance of the datatype TABLEITEM, whose fields are as follows. The first four fields are supplied by the application; the others are maintained by TableBrowser and should be considered read-only from the point of view of the application:

TIDATA [Record field]

An arbitrary pointer to the application data.

TIUNSELECTABLE [Record field]

If T, the user is not permitted to select this item.

TIUNCOPYSELECTABLE [Record field]

If T, the user is not permitted to copy-select this item.

TIUNDELETABLE [Record field]

If T, the user is not permitted to delete this item. This field is not currently supported by TableBrowser, though the application may use it itself for this purpose.

TI# [Record field]

An integer denoting the position of this among the set of items in the browser. This field is maintained automatically by TableBrowser as items are removed or new items are inserted. The first item is numbered 1.

TISELECTED [Record field]

T if the item is currently selected; NIL otherwise.

TIDELETED [Record field]

T if the item is currently marked for deletion; NIL otherwise.

TABLEBROWSER Record

For each browser, there is an instance of the datatype TABLEBROWSER, whose fields are described below. TABLEBROWSERs are created by TB.MAKE.BROWSER (see the section "Functions," below). The first six fields listed here are "methods"— application functions called from the TableBrowser code when some event occurs or in order to carry out some operation. The application function is called with the browser object itself as the first argument, and possibly other arguments. Only the first method, TBPRINTFN, is required; the others may be NIL. For most applications, the fields listed here are set in the call to TB.MAKE.BROWSER and then never changed.

TBPRINTFN [Record field]

Application function invoked to print a single item in the browser. The three arguments are the browser, the item being painted, and the window in which the item is to be painted. At the time the method is called, TableBrowser has set the x,y position of the window to the left edge of the line where this item starts. It has done nothing to the line, including clear it; the application must clear the line before printing if it so desires. Somewhat smoother, but more complex, repainting is possible if the application clears only the areas that it is not otherwise overwriting. After the printing is finished, TableBrowser supplies a selection mark and deletion line if appropriate. TableBrowser worries about scrolling, reshaping, inserting new items, etc., and calls the printing function for each line that it has determined needs to be (re)displayed as the result of operations performed on the browser.

TBCOPYFN [Record field]

Application function invoked when an item in the browser is copy-selected. Copy selection occurs when the Copy or Shift key is held down and the mouse is used to select a line in the browser. While the Copy key is down, TableBrowser highlights the item under the cursor with a dotted underline; when the copy key is released, the highlighting is removed, and the copy function is called with two arguments: the browser and the item selected. The Copy function typically calls BKSYSBUF on some string or structure representing the item selected. Copy selection can be aborted by moving the mouse outside the window with the mouse button still down. If an item's TIUNCOPYSELECTABLE field is true, copy selection is ignored while the mouse is inside the item. If TBCOPYFN is NIL, then copy selection is ignored for the entire browser.

TBCLOSEFN [Record field]

Application function invoked when the user tries to close or shrink the browser window. The three arguments are the browser, the window, and a flag whose value is one of the symbols CLOSE or SHRINK. If the function returns the symbol DON'T, then the close or shrink operation is aborted; if it returns NIL, it proceeds; otherwise, the value is a function to run as a separate cleanup process. In this last case, the window remains open and the value returned from the TBCLOSEFN application is called in a new process with the same three arguments (browser, window, flag). When this function finishes, it should call the function TB.FINISH.CLOSE (see the section "Functions," below) to complete the close or shrink operation, assuming it wishes it to proceed.

TBAFTERCLOSEFN [Record field]

Application function invoked when the browser window is about to be discarded. The two arguments are the browser and the window. This function is called *after* the TBCLOSEFN, if any, has permitted the window to close; note that it is not called when a window is merely shrunk. The application might, for example, want to remove the browser from its own structures, snap circular links, etc. The return value is ignored.

TBTITLEEVENTFN [Record field]

Application function invoked when the middle mouse button is pressed while the cursor is in the title bar of the browser (clicking in the body of the window is for selection and is handled by TableBrowser itself). The two arguments are the window and the browser. Note that this is the only method that does not take the browser as the first argument; this is to match the form of window system button event functions.

TBFONTCHANGEFN [Record field]

Application function invoked when the font of the window is changed (by TB.SET.FONT). The two arguments are the browser and the window. The application function might, for example, want to change cached information about the size of the font.

TBLINETHICKNESS [Record field]

The thickness of the horizontal lines drawn through deleted items. This defaults to the value of TB.DELETEDLINEHEIGHT, initially 1. For example, setting this field to the height of an item would result in items being completely blacked out when they are deleted.

TBHEADINGWINDOW [Record field]

An optional auxiliary window that is to be horizontally scrolled in parallel with the main window. The WIDTH of the window's EXTENT property is maintained in synch with that of the main window, and whenever the main window is horizontally scrolled, the heading window is scrolled by the same amount. You still need to create this auxiliary window, attach it where you want it and supply it with a REPAINTFN. (This is how FileBrowser implements its header line identifying the columns of attributes.)

TBUSERDATA [Record field]

An arbitrary pointer to application-dependent data.

The following fields describe the size and shape of items. The values are usually derived from information supplied when a browser is created (see TB.MAKE.BROWSER) and then never changed. If an application wishes to change any of these fields once a browser has been created, it should call TB.SET.FONT afterwards to notify TableBrowser of the change.

TBFONT [Record field]

A font descriptor, the default font in which items are painted.

TBFONTHEIGHT [Record field]

The height of the font.

TB#LINESPERITEM [Record field]

The number of lines (in units of the font's height) occupied by each item. TableBrowser requires that each item occupy the same number of lines. The default is 1. For multi-line items, the window is positioned at the first line when its print function is called, selection markers point at the first line, and deletion lines are drawn only through the first line.

TBITEMHEIGHT [Record field]

The total height of an item. This is normally the font height times the number of lines per item, but an application can set it explicitly, independent of the font, instead of specifying the number of lines per item.

TBBASELINE [Record field]

The distance of an item's baseline above the bottom of the item. This field has been renamed from TBFONTDESCENT. This field is used for two purposes:

- When the browser's PRINTFN is called, the y-position of the window is set to be at the baseline.
- Selection marks and deletion lines are centered between the baseline and the top of the item.

The following fields are maintained by TableBrowser, but may be of use to application code; they should be considered read-only.

TBWINDOW [Record field]

A pointer to the window containing the browser.

TBLOCK [Record field]

A monitor lock acquired by the TableBrowser when performing operations on the browser. Application code may want to hold this lock while performing a series of TableBrowser operations that it wishes to have occur atomically. Selection and scrolling are inhibited while this lock is busy.

Functions

This section describes the functions that create and manipulate TableBrowser windows and their contents. Typical use for most of these functions is from the code invoked by commands from a menu attached to the window by the application.

Creating a TableBrowser

(TB.MAKE.BROWSER *ITEMS WINDOWSPEC PROPS*) [Function]

Creates a new browser, browsing *ITEMS*, a list of TABLEITEM records. *ITEMS* may be NIL, in which case an empty browser is created. It is permissible for the TISELECTED and/or TIDELETED fields to be true in any item; this is a way of setting the initial selection, and/or pre-deleting some items.

If *WINDOWSPEC* is supplied, it is a window or region; otherwise, the user is prompted for a window.

PROPS is a list in property list format specifying initial values for some features of the browser. The properties PRINTFN, COPYFN, CLOSEFN, AFTERCLOSEFN, LINESPERITEM, ITEMHEIGHT, BASELINE, HEADINGWINDOW, LINETHICKNESS, and USERDATA set the corresponding fields of the new TABLEBROWSER record (see above). The following additional properties are recognized:

TITLE The title to put on the window. If NIL, the window will not be given a title.

FONT The font in which to paint items (a font descriptor or any other argument acceptable to FONTCREATE). This font is made the window's current font. If the browser's print function displays items in more than one font, the application should choose the tallest font for the FONT property, and it is responsible for ensuring that the correct font is always in use. If this property is NIL, the default display font is used. The browser fields TBFONT and TBFONTHEIGHT are set from this font. In addition, if the caller did not specify the ITEMHEIGHT property, the fields TBITEMHEIGHT and TBBASELINE are calculated from the font. TableBrowser uses the sizes to know where each line starts.

If the caller specified the ITEMHEIGHT property but no BASELINE, the baseline is taken to be zero. If the caller did not specify ITEMHEIGHT, then the baseline is calculated to be the font's descent, or in the case of multiple lines per item, the descent plus the font height times

(#LINESPERITEM–1), so that the behavior described above for TB#LINESPERITEM holds.

TB.MAKE.BROWSER returns a new TABLEBROWSER object describing the browser. The application is free to attach further windows to this one. The TABLEBROWSER object is also stored on the window's TABLEBROWSER property.

(TB.REPLACE.ITEMS *BROWSER NEWITEMS*) [Function]

Completely replaces the items of *BROWSER* with *NEWITEMS*, a list of TABLEITEM records. This is a lot like creating a new browser, except that the window structure is already there.

(TB.SET.FONT *BROWSER FONT*) [Function]

Changes *BROWSER*'s display font to *FONT*, which is of the same form as the FONT property given to TB.MAKE.BROWSER. If FONT is NIL, TB.SET.FONT makes its computations based on the browsers current display-related fields (TBFONT, TB#LINESPERITEM, etc).

TB.SET.FONT clears the window; the application is responsible for ensuring that the window is redisplayed, e.g., by calling REDISPLAYW. TB.SET.FONT does not do the redisplay itself, so as to avoid double redisplay in the case where the application also wants to change the items at the same time (by calling TB.REPLACE.ITEMS).

(TB.FINISH.CLOSE *BROWSER WINDOW CLOSEFLG* —) [Function]

Takes care of closing or shrinking *WINDOW*, occupied by *BROWSER*, after the cleanup performed by the browser's TBCLOSEFN (see above).

CLOSEFLG is one of the symbols CLOSE or SHRINK.

(TB.BROWSER.BUSY *BROWSER*) [Function]

Briefly changes the cursor to a large "X", the conventional way TableBrowser indicates that an operation attempted with the mouse cannot be performed because the browser is busy.

Simple Item Operations

(TB.SELECT.ITEM *BROWSER ITEM*) [Function]

Marks *ITEM* in *BROWSER* selected. Ordinarily, selection occurs by means of the mouse. However, applications may want to programmatically select items in response to a user command. Selected items are indicated by a small triangle in the left margin.

(TB.UNSELECT.ITEM *BROWSER ITEM*) [Function]

Marks *ITEM* in *BROWSER* not selected.

(TB.UNSELECT.ALL.ITEMS *BROWSER*) [Function]

Marks *all* items in *BROWSER* not selected. This is considerably faster than unselecting items one at a time. The typical use for this function is prior to calling TB.SELECT.ITEM, to ensure that the new selection is the browser's *only* selection.

(TB.DELETE.ITEM *BROWSER ITEM*) [Function]

Marks *ITEM* in *BROWSER* for deletion. The display shows a line drawn through the item. It is permissible to delete a deleted item (it is a no-op).

(TB.UNDELETE.ITEM *BROWSER ITEM*) [Function]

Removes the deletion mark from *ITEM* in *BROWSER*.

(TB.INSERT.ITEM *BROWSER NEWITEM BEFOREITEM*) [Function]

Adds *NEWITEM* to *BROWSER*'s set of items, inserting it immediately before the item *BEFOREITEM*, or at the end if *BEFOREITEM* is NIL.

(TB.REMOVE.ITEM *BROWSER ITEM*) [Function]

Removes *ITEM* from *BROWSER*'s set of items. This is the operation that an application's "Expunge" function would typically use, but it need not be in any way correlated with deleted items.

(TB.NORMALIZE.ITEM *BROWSER ITEM*) [Function]

Scrolls *BROWSER*'s window, if necessary, so that *ITEM* is visible.

(TB.CLEAR.LINE *BROWSER ITEM LEFT WIDTH*) [Function]

Clears the contents of *ITEM*'s line in *BROWSER*, starting at the x-position *LEFT* and clearing a region *WIDTH* pixels wide. *LEFT* defaults to zero, *WIDTH* to infinity, so omitting both clears the whole line.

This function is typically used by a browser's print function to clear the line before displaying fresh contents. An application wanting to print with minimal visual disruption may want to use TB.CLEAR.LINE only on those portions of the line not being printed to explicitly, so that repainting a line with only slight changes minimizes the apparent display activity.

(TB.REDISPLAY.ITEMS *BROWSER FIRSTITEM LASTITEM*) [Function]

Ordinarily, TableBrowser takes care of deciding when items need to be redisplayed (e.g., when scrolling, reshaping, inserting, etc.). However, there may be times when circumstances beyond the knowledge of TableBrowser require that an item be redisplayed, e.g., when the contents of the item are changed by the application. In such cases, the application is responsible for telling TableBrowser that repainting is needed.

TB.REDISPLAY.ITEMS explicitly invokes *BROWSER*'s repaint method to redisplay items *FIRSTITEM* through *LASTITEM*, which may be the same item in the case of redisplaying a single item. The item arguments may be given as TABLEITEM objects or as a number. *FIRSTITEM* defaults to the browser's first item and *LASTITEM* defaults to the last one; thus (TB.REDISPLAY.ITEMS *BROWSER*) forces redisplay of the entire browser, and is equivalent to calling REDISPLAYW on the window. Only those items currently visible in the window are actually repainted.

Operations on Multiple Items

(TB.NUMBER.OF.ITEMS *BROWSER TYPE*) [Function]

Returns the number of items in *BROWSER* of the specified *TYPE*, one of the following symbols:

NIL	Returns the total number of items.
SELECTED	Returns the number of items currently selected.
DELETED	Returns the number of items currently deleted.

(TB.NTH.ITEM *BROWSER N*) [Function]

Returns the *N*th item in *BROWSER*. *N* is an integer; the first item is numbered 1. Returns NIL if *N* is less than 1 or greater than the number of items in the browser.

Most of the following functions accept a predicate or mapping function. The results of the mapping are unpredictable if the mapping function adds or removes items, so an application wishing to do so should first collect the items of interest and map over that list itself.

(TB.COLLECT.ITEMS *BROWSER PREDFN*) [Function]

Returns a list, in browser order, of all the items in *BROWSER* of the type specified by *PREDFN*. *PREDFN* can be one of the symbols NIL, SELECTED or DELETED, which are interpreted as for TB.NUMBER.OF.ITEMS. Otherwise, *PREDFN* is a predicate function of two arguments, *BROWSER* and an item from the browser; *PREDFN* should return T if the item is to be collected.

(TB.MAP.ITEMS *BROWSER MAPFN NULLFN*) [Function]

Applies the function *MAPFN* successively to each item in *BROWSER*. *MAPFN* should accept two arguments, *BROWSER* and the item.

If the browser is empty, TB.MAP.ITEMS instead calls *NULLFN*, if specified, with the single argument *BROWSER*.

(TB.MAP.SELECTED.ITEMS *BROWSER MAPFN NULLFN*) [Function]

Applies the function *MAPFN* successively to each selected item in *BROWSER*. *MAPFN* should accept two arguments, *BROWSER* and the item.

If no items are currently selected in the browser, TB.MAP.SELECTED.ITEMS instead calls *NULLFN*, if specified, with the single argument *BROWSER*.

A typical application calls TB.MAP.SELECTED.ITEMS in response to a menu selection to carry out the operation on the items you have selected.

(TB.MAP.DELETED.ITEMS *BROWSER MAPFN NULLFN*) [Function]

Applies the function *MAPFN* successively to each deleted item in *BROWSER*. *MAPFN* should accept two arguments, *BROWSER* and the item.

If no items are currently deleted in the browser, TB.MAP.DELETED.ITEMS instead calls *NULLFN*, if specified, with the single argument *BROWSER*.

(TB.FIND.ITEM *BROWSER PREDFN FIRST# LAST# BACKWARDSFLG*) [Function]

Returns the first item in *BROWSER* in the range of items numbered *FIRST#* through *LAST#* that satisfies the predicate *PREDFN* (a function of two arguments), *BROWSER* and the item.

PREDFN can also be one of the symbols SELECTED or DELETED to search for selected or deleted items.

FIRST# defaults to 1, *LAST#* defaults to the number of items in the browser, so omitting both searches the whole browser.

If *BACKWARDSFLG* is true, the range is searched in reverse order.

Miscellaneous Access Functions

These provide functional access to some of the fields defined in the Records section.

(TB.ITEM.SELECTED? *BROWSER ITEM*) [Function]

Returns T if *ITEM* in *BROWSER* is selected.

(TB.ITEM.DELETED? *BROWSER ITEM*) [Function]

Returns T if *ITEM* in *BROWSER* is deleted.

(TB.WINDOW *BROWSER*) [Function]

Returns a pointer to the window containing the browser.

(TB.USERDATA *BROWSER NEWVALUE*) [Function]

Returns the value of the TBUSERDATA field of the browser; if *NEWVALUE* is supplied, it is stored as the new value of this field.

TB.LEFT.MARGIN [Constant]

The left margin, in pixels, of the start of each item in the browser. Space to the left of this point is used by the selection marker. This constant is compiled into TableBrowser.

Limitations

In the current implementation, the items in a browser are maintained as a simple list. This means that some operations that might be expected to take constant time (e.g., returning the nth item) instead take linear time (or worse). TableBrowser currently optimizes its operations for sequential access, so that the most typical operations are not adversely affected. However, note that performance may not be acceptable for very large browsers (on the order of 1000 items) when accessing items in random order, or in particular, searching a browser in reverse order.

[This page intentionally left blank]