This one’s a bit more involved. One reader asked about editing data during a TBrowse. What he wants to do is program the TBrowse so that when the user presses the Enter key, the program allows the user to edit all the visible columns.

The first thing you must decide is how to relate the displayed data to the underlying data source. You know that a TBColumn object’s block instance variable determines what is displayed in a particular column, so evaluating that code block gives you the contents. How do you change it, though? The key: Use a get/set block as the TBColumn’s block instance variable, as in:

```plaintext
// Database alias is Names
oTbc := TBColumnNew ()
oTbc:heading := "Lname Field"
oTbc:block := {|c| iif (c == NIL, ; Names -> Lname, ; Names -> Lname := c)}
```

The TBrowse system only GETS the block; it never SETS it. Your code, however, can access the get/set block and SET it to update the field following an edit.

So now that you know how to get and set a field, you can issue the get. To issue a GET just for the currently selected column, use:

```plaintext
// TBrowse object oTbr
// Get a reference to the current column
oTbc := oTbr:getColumn (oTbr:colPos)
// Edit to memvar copy
xTemp := Eval (oTbc:block)
@ Row (), Col () GET xTemp
READ
IF Updated ()
   IF Rlock () // Update the field
      Eval (oTbc:block, xTemp)
   ELSE // Unable to lock the record
      ENDIF
ENDIF
```

The TBrowse object’s colPos instance variable contains the ordinal position of the currently selected column. You pass this to GetColumn(), which then returns a reference to that column. Place the GET command at the current cursor position, as returned by Row() and Col().

To issue a GET for each visible column, you must determine two things:

* Which columns are visible?
* What is the row and column position of each column?

The TBrowse object contains two instance variables you can use to determine which columns are visible:

* leftVisible—Ordinal position of left-most visible column.
* rightVisible—Ordinal position of right-most visible column.

The Row() function returns the current row. The only thing you need to determine are the column positions of each TBColumn. Listing 1 shows TBCColPos(), a routine that returns an array containing the column positions of the visible columns.

The idea is simple. It visits each visible TBColumn object in turn, saving the column position using the Col() function. To do this you must assign to oTbr:colPos, then re-stabilize the display. So the user doesn’t see this behinds the scene action, your code first turns off the display using dispBegin().

Listing 1—Determining column positions of visible TBrowse objects.

```plaintext
FUNCTION TBCColPos (oTbr)
   LOCAL nSaveColNum := oTbr:colPos
   LOCAL aColPos := array (oTbr:rightVisible)
   LOCAL nColNum
   dispBegin ()
   FOR nColNum := oTbr:leftVisible TO oTbr:rightVisible
      oTbr:colPos := nColNum
      DO WHILE !oTbr:stabilize()
         ENDDO
      aColPos[nColNum] := col ()
      oTbr:deHilite ()
   NEXT
   // redisplay current highlight bar
   oTbr:colPos := nSaveColNum
   DO WHILE !oTbr:stabilize ()
```
Given this routine, it's straightforward to implement editing:

CASE nKey == K_ENTER
        aCols := TBCColPos(oTbr)
        // We will edit to elements of this array
        aVars := Array(Len (aCols))
        // Issue a GET for each visible TBColumn
        FOR i := oTbr:leftVisible TO oTbr:rightVisible
            oTbc := oTbr:getColumn (i)
            aVars [i] := Eval (oTbc:block)
            @ Row (), aCols [i] GET aVars [i]
        NEXT READ
        // Now apply updates if user did not abort the edit
        IF Updated() .AND. LastKey () != K_ESC
           IF Rlock ()
               FOR i := oTbr:leftVisible TO oTbr:rightVisible
                   oTbc := oTbr:getColumn (i)
                   Eval (oTbc:block, aVars [i])
               NEXT UNLOCK
           ENDIF
        ENDIF
        // Redraw the row
        oTbr:refreshCurrent ()
    END CASE

On detecting the Enter key, the code uses TBCColPos() to determine the starting column positions for each TBColumn object.

The GETs will edit copies of whatever the TBColumn's block instance variable returns to memory.

These are stored in the array aVars. You can see how the code sets up this array and issues the GET:

FOR i := oTbr:leftVisible TO oTbr:rightVisible
    oTbc := oTbr:getColumn (i)
    aVars [i] := Eval (oTbc:block)
    @ Row (), aCols [i] GET aVars [i]
NEXT

After the READ, if the user updates any of the entries, the code rewrites the data using the SET form of the TBColumn's get/set block:

FOR i := oTbr:leftVisible TO oTbr:rightVisible
    oTbc := oTbr:getColumn (i)
    Eval (oTbc:block, aVars [i])
NEXT

Listing 2 shows the entire code listing.

Listing 2--Editing in a TBrowse.

// Simple test routine.
#include "Inkey.ch"
MEMVAR GetList
PROC Test
    LOCAL nKey, lExitRequested := .F.
    LOCAL oTbr, aCols, aVars, oTbc, i
    USE Names
    oTbr := TBrowseDB (1, 1, 23, 78)
    AddAllFields (oTbr)
    DO WHILE !lExitRequested
        DO WHILE !oTbr:stabilize()
            ENDDO
        nKey := InKey (0)
        IF !StdMeth(nKey, oTbr)
            DO CASE
                CASE nKey == K_ESC
                    lExitRequested := .T.
                CASE nKey == K_ENTER
                    aCols := TBCColPos(oTbr)
                    aVars := Array(Len (aCols))
                    FOR i := oTbr:leftVisible TO oTbr:rightVisible
                        oTbc := oTbr:getColumn (i)
                        aVars [i] := Eval (oTbc:block)
                        @ Row (), aCols [i] GET aVars [i]
                    NEXT READ
                    IF Updated() .AND. LastKey () != K_ESC
                        IF Rlock ()
                            FOR i := oTbr:leftVisible TO oTbr:rightVisible
                                oTbc := oTbr:getColumn (i)
                                Eval (oTbc:block, aVars [i])
                            NEXT UNLOCK
                        ENDIF
                    ENDIF
                    // Redraw the row
                    oTbr:refreshCurrent ()
            END CASE
        ENDIF
    DO WHILE !lExitRequested
        ENDDO
    nKey := InKey (0)
    IF !StdMeth(nKey, oTbr)
        DO CASE
            CASE nKey == K_ESC
                lExitRequested := .T.
            CASE nKey == K_ENTER
                aCols := TBCColPos(oTbr)
                aVars := Array(Len (aCols))
                FOR i := oTbr:leftVisible TO oTbr:rightVisible
                    oTbc := oTbr:getColumn (i)
                    aVars [i] := Eval (oTbc:block)
                    @ Row (), aCols [i] GET aVars [i]
                NEXT READ
                IF Updated() .AND. LastKey () != K_ESC
                    IF Rlock ()
                        FOR i := oTbr:leftVisible TO oTbr:rightVisible
                            oTbc := oTbr:getColumn (i)
                            Eval (oTbc:block, aVars [i])
                        NEXT UNLOCK
                    ENDIF
                ENDIF
                // Redraw the row
                oTbr:refreshCurrent ()
        END CASE
// Return an array containing the starting columns
// of the visible TBColumn objects
FUNCTION TBCColPos (oTbr)
LOCAL nSaveColNum := oTbr:colPos
LOCAL aColPos := array (oTbr:rightVisible)
LOCAL nColNum
dispBegin()
FOR nColNum := oTbr:leftVisible TO oTbr:rightVisible
  nColNum := oTbr:colPos := nColNum
  DO WHILE !oTbr:stabilize()
  ENDdo
  aColPos[nColNum] := col()
  oTbr:deHilite()
ENDFOR
// redisplay current prompt
oTbr:colPos := nSaveColNum
DO WHILE !oTbr:stabilize()
ENDdo
dispEnd()
RETURN aColPos
// Apply a key to TBrowse object. Return .T. if handled, .F. otherwise.
FUNCTION StdMeth(nKey, oTbr)
LOCAL lKeyHandled := .T.
DO CASE
  CASE nKey == K_DOWN; oTbr:down()
  CASE nKey == K_UP; oTbr:up()
  CASE nKey == K_PGDN; oTbr:pageDown()
  CASE nKey == K_PGUP; oTbr:pageUp()
  CASE nKey == K_CTRL_PGUP; oTbr:goTop()
  CASE nKey == K_CTRL_PGDN; oTbr:goBottom()
  CASE nKey == K_RIGHT; oTbr:right()
  CASE nKey == K_LEFT; oTbr:left()
  CASE nKey == K_HOME; oTbr:home()
  CASE nKey == K_END; oTbr:end()
  CASE nKey == K_CTRL_LEFT; oTbr:panLeft()
  CASE nKey == K_CTRL_RIGHT; oTbr:panRight()
  CASE nKey == K_CTRL_HOME; oTbr:panHome()
  CASE nKey == K_CTRL_END; oTbr:panEnd()
  OTHERWISE; lKeyHandled := .F.
ENDCASE
RETURN lKeyHandled
// Add all the fields from the currently selected
// database to the passed TBrowse object. Use
// Get/Set blocks for the TBColumn's block instance variables.
FUNCTION AddAllFields(oTbr)
LOCAL oTbc, ; nFieldNum, ; nFields := fcount()
FOR nFieldNum := 1 TO nFields oTbc := TBColumnNew (field(nFieldNum), ;
  fieldblock(field[nFieldNum])) oTbr:addColumn (oTbc)
NEXT
RETURN NIL
http://www.accessmylibrary.com/coms2/summary_0286-9277846_ITM