Databound Grid – Data Abstraction Extended

By Jim Douglas

The BBjDataBoundGrid provides an easy way to display and edit data from a BASIS multikeyed file (MKEYED, XKEYED or VKEYED) or from any SQL-accessible database. It builds on recordsets and databound controls first introduced in BBj 4.0. Databound controls define the association between fields in the database and GUI controls on the screen. With the databound controls automatically transferring data between the screen and the database, the developer can focus on application details to build data-oriented applications more quickly and easily than ever before.

Figure 1. BBjDataBoundGrid state table

File Maintenance Sample

Figure 1 shows a typical grid-based maintenance screen for viewing and editing state codes.

Figure 2 shows the complete source that produced this maintenance screen.

continued...

```
rem ' BBjDataBoundGrid State Table

rem ' (1) Create a BBjRecordSet
database$ = "ChileCompany"
options$ = "user=admin,password=admin123"
select$ = "select * from state"
recordset! = BBJAPI().createSQLRecordSet(database$,options$,select$)

rem ' (2) Create a BBjWindow with a BBjDataBoundGrid
sysgui! = bbjapi().openSysGui("X0")
window! = sysgui!.addWindow(100,100,200,200,"States",$00110083$)
window!.setCallback(window!.ON_CLOSE,"eoj")
dbgrid! = window!.addDataBoundGrid(101,10,10,180,180,$81ce$)
dbgrid!.setColumnHeaderAlignment(dbgrid!.GRID_ALIGN_LEFT)
dbgrid!.setRowHeight(25)
dbgrid!.setEditable(1)
dbgrid!.setFitToGrid(1)
dbgrid!.focus()

rem ' (3) Bind the BBjRecordSet to the BBjDataBoundGrid
dbgrid!.bindRecordSet(recordset!)
dbgrid!.setDefaultColumnHeaders()
dbgrid!.setDefaultAlignment(dbgrid!.GRID_ALIGN_LEFT)

rem ' (4) On requested row change, update any edits
dbgrid!.setCallback(dbgrid!.ON_DB_GRID_ROW_CHANGE_REQUEST,"Row_Change")

process_events

eoj:
release

Row_Change:
  rem ' (4) Update the changes to the underlying database
  if (dbgrid!.isEditable() and recordset!.isCurrentRecordDirty()) then
    recorddata! = recordset!.getCurrentRecordData()
    recordset!.update(recorddata!,err=*next)
  endif
  rem ' Move to the requested row and column
  rowchange! = sysgui!.getLastEvent()
  recordset!.moveToRecord(rowchange!.getRow())
  dbgrid!.setSelectedColumn(rowchange!.getColumn())
return
```

Figure 2. BBjDataBoundGrid state table – listing `(State.src)`

Jim Douglas
Software Engineer
Contractor

The main sections of this program are:

(1) `Create a BBjRecordSet`
Uses an SQLRecordSet that maps to the state table of the ChileCompany database, which is an MKEYED file with an associated entry in the Data Dictionary. Recordsets can also be associated with BBj multikeyed files (FileRecordSet) via a string template or with memory-based data structures (MemoryRecordSet) that might have been constructed from READ or READ RECORD statements applied to non-normalized data files.

(2) `Create a BBjWindow with a BBjDataBoundGrid`

(3) `Bind the BBjRecordSet to the BBjDataBoundGrid`
Associates the databound grid to the recordset and creates an ongoing two-way linkage between the grid and the underlying data structure.

(4) `Update the changes to the underlying database`
When the user attempts to move to a new row, BBj fires a BBjDBGridRowChangeRequestEvent for which the programmer may choose how to respond. This sample calls BBjRecordSet::isCurrentRecordDirty() to see if any changes were made; if so, it writes the changed record to the underlying database. Finally, it moves to the requested row and column.

To see the two-way linkage in action, run two copies of `State.src` at the same time. Change one of the state names on one screen and move away from that row to commit the change to the database. When navigating again to that row on the other screen, the value updates automatically to reflect the change.

Property Sheet Sample

Databound grids are a convenient way to display any table-oriented data. **Figure 3** displays a typical property sheet that shows current Java system properties. The source code list in **Figure 4** uses a memory-based databound grid for the property sheet sample.

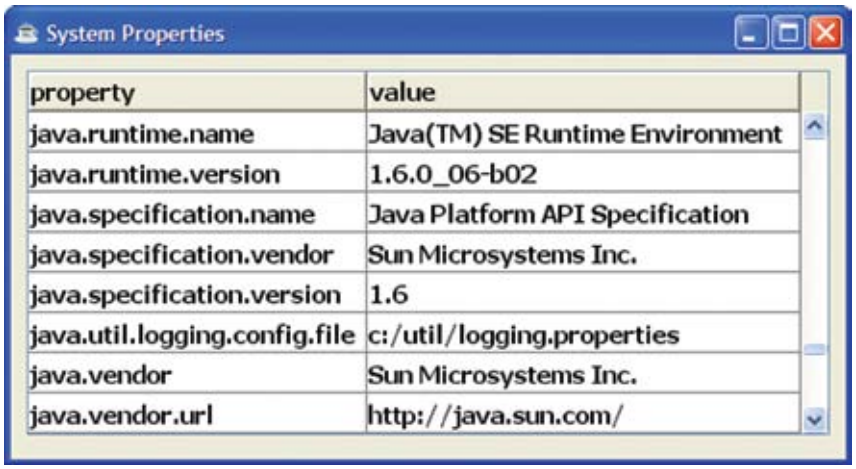

Figure 3. BBjDataBoundGrid property sheet

The main sections of this program are similar to the first sample:

(1) `Create a BBjRecordSet`
Uses a MemoryRecordSet of properties and corresponding values.

(2) `Populate the BBjRecordSet with the Java system properties`
Lists the current Java system properties in alphabetical order.

(3) `Create a BBjWindow with a BBjDataBoundGrid`

(4) `Bind the BBjRecordSet to the BBjDataBoundGrid`
Binds (associates) the databound grid to the recordset.

(5) `Accept any requested row changes`
Fires a BBjDBGridRowChangeRequestEvent when the user attempts to move to a new row. This sample just displays the data without needing to allow for updates and moves to the requested row and column.

continued...

```
rem ' BBjDataBoundGrid Property Sheet

rem ' (1) Create a BBjRecordSet
template$ = "property:c(1*),value:c(1*)"
recordset! = bbjapi().createMemoryRecordSet(Template$)

rem ' (2) Populate the BBjRecordSet with the Java system properties
p! = java.lang.System.getProperties()
k! = new java.util.TreeMap(p!)
i! = k!.keySet().iterator()
while i!.hasNext()
  key! = i!.next()
  val! = p!.get(key!)
  rec! = recordset!.getEmptyRecordData()
  rec!.setFieldValue("property",str(key!))
  rec!.setFieldValue("value",str(val!))
  recordset!.insert(rec!)
wend

rem ' (3) Create a BBjWindow with a BBjDataBoundGrid
sysgui! = bbjapi().openSysGui("X0")
window! = sysgui!.addWindow(100,100,400,500,"System Properties",$00110083$)
window!.setCallback(window!.ON_CLOSE,"eoj")
window!.setCallback(window!.ON_RESIZE,"resize")
dbgrid! = window!.addDataBoundGrid(101,10,10,380,480,$81ce$)
dbgrid!.setFont(sysgui!.makeFont("Tahoma",11,1))
dbgrid!.setColumnHeaderAlignment(dbgrid!.GRID_ALIGN_LEFT)
dbgrid!.setRowHeight(25)
dbgrid!.setFitToGrid(1)
dbgrid!.focus()

rem ' (4) Bind the BBjRecordSet to the BBjDataBoundGrid
dbgrid!.bindRecordSet(recordset!)
dbgrid!.setDefaultColumnHeaders()
dbgrid!.setDefaultAlignment(dbgrid!.GRID_ALIGN_LEFT)

rem ' (5) Accept any requested row changes
dbgrid!.setCallback(dbgrid!.ON_DB_GRID_ROW_CHANGE_REQUEST,"Row_Change")

process_events

eoj:
release

Row_Change:
  rem ' (5) Move to the requested row and column
  rowchange! = sysgui!.getLastEvent()
  recordset!.moveToRecord(rowchange!.getRow())
  dbgrid!.setSelectedColumn(rowchange!.getColumn())
return

resize:
  rem ' Resize the grid when the screen is resized
  event! = sysgui!.getLastEvent()
  dbgrid!.setSize(event!.getWidth()-20,event!.getHeight()-20)
return
```

Figure 4. BBjDataBoundGrid property sheet – listing `(Properties.src)`

Database Table Editor Sample

The final sample program uses a databound grid to implement a powerful and flexible database editor as displayed in **Figure 5** and **Figure 6**. In addition to being able to change individual fields by editing grid cells, the user can also add, change, or delete records using a dynamically created full-screen editor.

Databound vs. Data-Aware

The databound grid is more powerful and flexible than the older data-aware grid. The data-aware grid automatically writes any changes to the underlying database, without possible programmer intervention. This can allow for slightly less programming, but at the cost of flexibility. With the databound grid, developers always choose when to update changes to the database, allowing more control over data integrity. Furthermore, they can use the databound grid in coordination with other databound controls, all interacting with a single recordset.

continued...

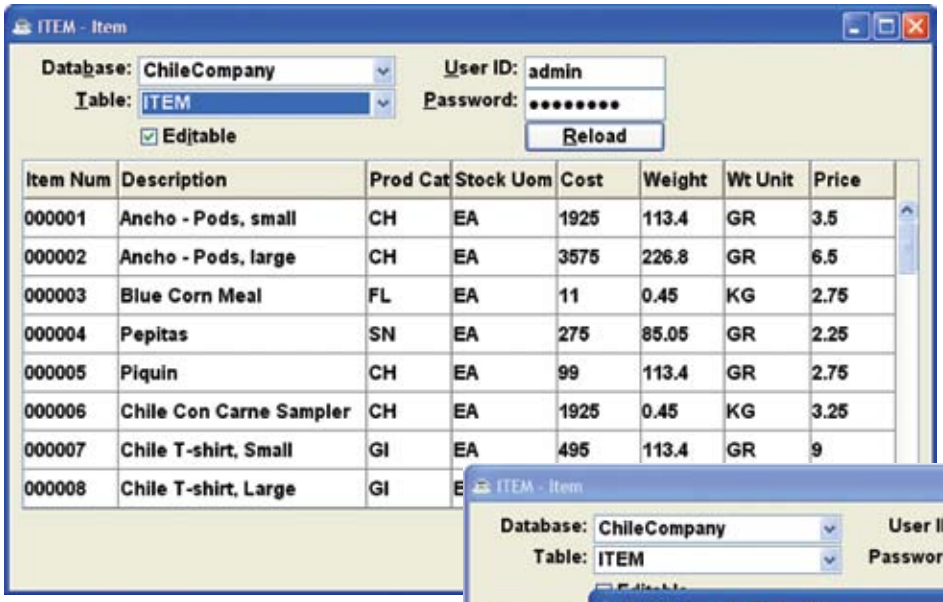

Figure 5. Database table editor (`DBGrid.src`)

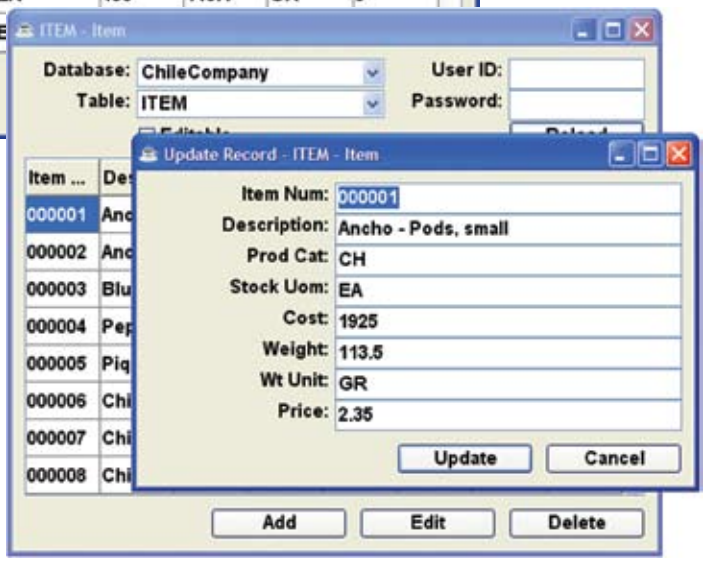

Figure 6. Database table editor – full-screen record editor

Summary

When using databound controls, BBj automatically manages the housekeeping details of copying data from the database record to the screen, and back again. This leaves developers free to focus on the business rules, reducing the time it takes to build powerful and flexible data-oriented applications.

Download the code samples for this article from www.basis.com/advantage/mag-v12n1/databoundgrid.zip

Refer to the online documentation at www.basis.com/onlinedocs/documentation/flash
and search the index for:

BBjRecordSet
BBjDataBoundGrid

Recordsets provide a consistent interface to the data in a BBj multikeyed file, SQL database, or (as of BBj 7.0) an in-memory data structure. For more information about the recordsets, see:

Using the BBjRecordSet
www.basis.com/advantage/mag-v7n3/bbjrecordset.pdf

Why Use the BBjRecordSet?
www.basis.com/advantage/mag-v8n1/recordset.html

Databound controls are BBj GUI controls (e.g. ListButton, InputE, InputN) with an ongoing link to a field in a recordset. For more information about databound controls, review:

BBjDataBound Controls
www.basis.com/advantage/mag-v7n3/bbjdataboundcontrols.pdf

Watch the Form Gen Wizard Trans"form" Data
www.basis.com/advantage/mag-v11n1/FormGenWizard01adv07_Links.pdf