BBj Databound Controls

By Brian Hipple

Data is the driving force for today's applications. Creating data presentation forms is the key task in the design of modern applications. Developers often end up spending a large amount of time writing code for the data presentation. Debugging and maintaining this code becomes a major effort as the data changes and the complexity of data increases. BASIS addresses these issues by supporting the concept of databound controls in BBj® 4.0.

Databound controls provide a simple, convenient, powerful, and transparent way for developers to create a read/write link between the controls on a form and the underlying data in their application. **Figure 1** presents a graphical description of the concept of databound controls.

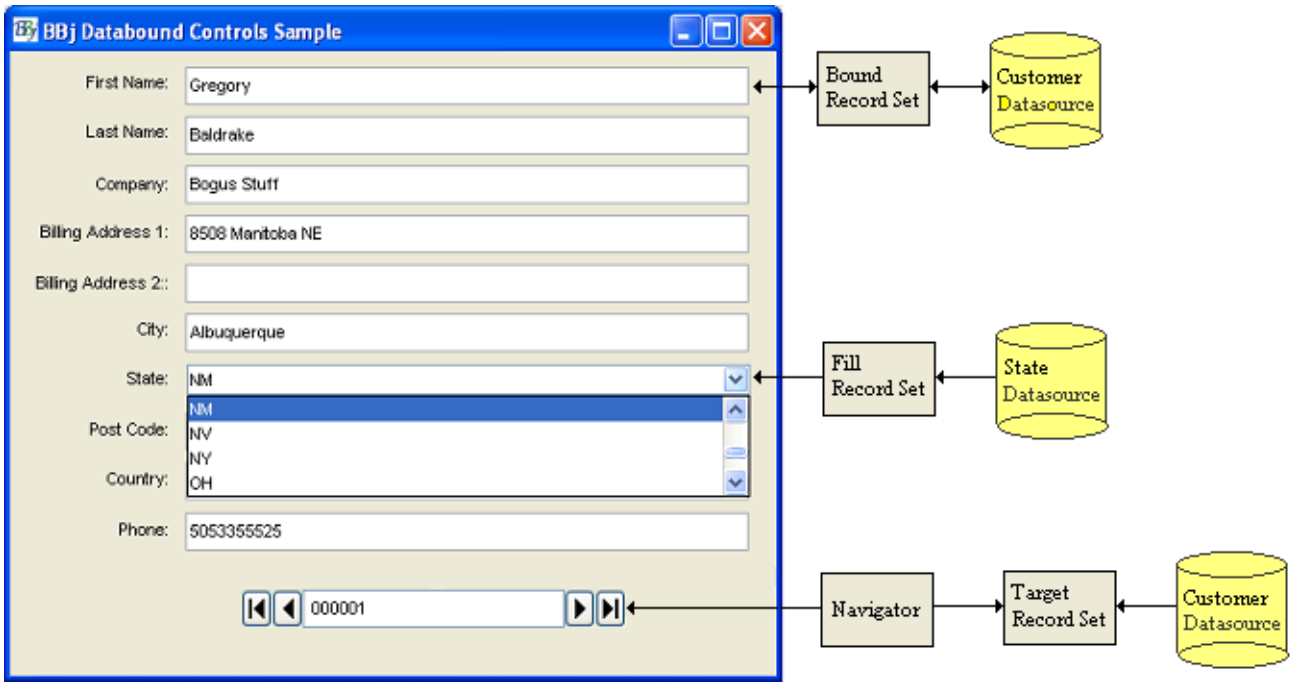

Figure 1. Databound Controls.

To explain the concept of databound controls, this article presents and discusses the ***BBjRecordSet*** object and the ***BBjNavigator*** control.

A BBjRecordSet contains a set of records either from a file or from the results of an SQL query. It allows for the navigating, updating, deleting, and inserting of these records. In addition, a BBjRecordSet provides a convenient way to perform a read record/write record or issue SQL commands.

The three categories of BBjRecordSets relating to databound controls include bound, fill, and target. A bound BBjRecordSet refers to a record set bound to one or more controls. It provides a read/write link between the controls and the data.

A fill BBjRecordSet is bound to a list control, which includes a List Button, List Edit, and List Box controls. It provides a read only link from the data to the list control. Its main function is filling the list control with data.

The new BBjNavigator control governs the target BBjRecordSet by navigating between the first, previous, next, and last record. Please see ***Using the BBjRecordSet*** for more information about using the BBjRecordSet object.

BBj provides a navigator control that enables basic navigation through a record set by responding to user events. This control is similar to having four separate buttons on the form. However, instead of receiving the ON_BUTTON_PUSH event for the four

continued...

different buttons, the program now receives four separate navigator events: ON_NAV_FIRST, ON_NAV_PREVIOUS, ON_NAV_NEXT, and ON_NAV_LAST.

When generating these events, a typical application validates the current record and then writes the record by calling the update method on the record set. The application then moves the record set pointer by calling the first, previous, next, or last method on the record set object. Furthermore, using databound controls does not require using a navigator control, as the application developer can choose to navigate through a record set in response to other control events or program conditions. **Figure 2** shows a sample of how an application can handle an ON_NAV_NEXT event on a navigator control.

```
rem ' ------------------------------
rem ' Win=101 Ctl=120 BBj Navigator (Navigator) NAV_NEXT (N3:113)
rem ' ------------------------------

W101_C120_NAV_NEXT:
rem ' Notify Event - Navigator - Next Record [>]

gb__navigator!=bbjapi().getSysGui().getWindow(gb__win.FORM).getControl(120)
gb__recordset!=gb__navigator!.getTargetRecordSet(err=*return)

rem ' Retrieve the current record
gb__recorddata!=gb__recordset!.getRecordData()

rem ' TODO: Insert validation here

rem ' Update the current record
gb__recordset!.writeRecordData(gb__recorddata!)

rem ' Move the record pointer to the next record
gb__recordset!.next(err=*next)
RETURN
```

Figure 2. Handling a Navigator Event.

This article explained the simplicity, convenience, power, and transparency of databound controls. BASIS provides developers with the ability to create databound controls, record sets, and the BBjNavigator control using GUIBuilder®, ResBuilder®, and programmatically using object syntax. By using ResBuilder and GUIBuilder and following the steps below, developers can now build an application with databound controls in minutes, without ever writing a single line of code!

Step 1: Create a GUIBuilder project

1. Run GUIBuilder from a machine with BBj installed.
2. Create a new GUIBuilder project by selecting **New** from the **File** menu.
3. Browse to the **<bbj install dir>/guibuilder** directory and type in a project name, i.e. DataBound.
4. Click the **Save** button.

Step 2: Create a resource file

1. GUIBuilder now displays a dialog box asking if you want to create a resource file.
2. Select the **Yes** button to create the resource file, which launches ResBuilder .
3. Create the Customer DB record set by selecting **BBjRecordSet** node, right mouse click, then select **Add** from the menu.

    a. Select the **SQL radio button** on the Record Set Type tab.
    b. Click the **Build** button.
    c. Click the **Query** button.

continued...

d. Enter a valid user name and password (setup in Enterprise Manager, with default of admin/admin123).
e. Click the **Query Databases** button.
f. Select the **ChileCompany** database in the list box.
g. Click the **OK** button twice.
h. Select the **SQL Command** tab.
i. Type `"select * from customer"` in the edit box.
j. Select the **SQL Connection tab.**
k. Click the **Test** button. If the specified information is valid, a MessageBox containing `"Connection is Valid"` appears.
l. Type `"Customer DB"` in the **Name** property.

After creating the Customer DB set, the ResBuilder screen looks like **Figure 3**:

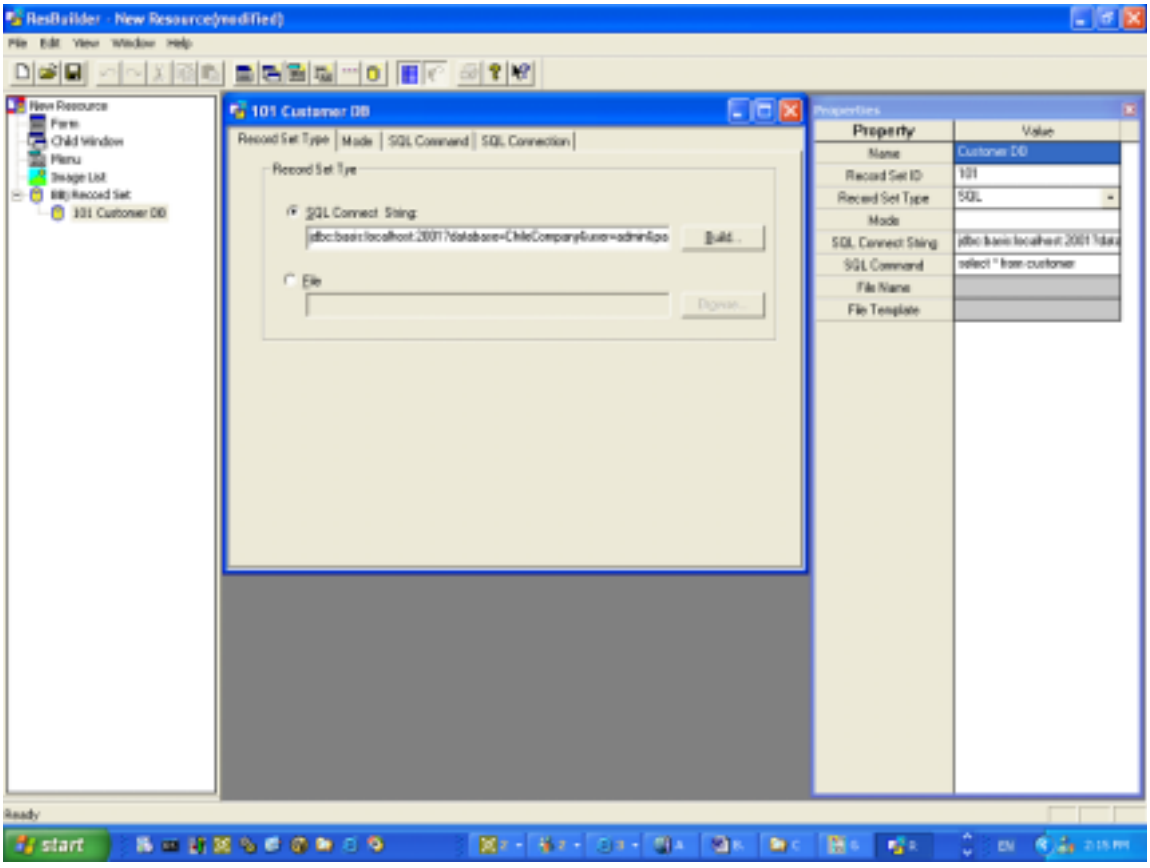

Figure 3. ResBuilder Customer DB.

continued...

4. Create the State File record set by selecting the **BBj Record set** node, right mouse click, then select **Add** from the menu

   a. Select the **File** radio button on the Record Set Type tab.
   b. Click the **Browse** button.
   c. Select the `<bbj install dir>/demos/chiledd/data/STATE` file.
   d. Click the **Select** button on the File Template tab
   e. Enter a valid user name and password (setup in Enterprise Manager, with default of admin/admin123).
   f. Click the **Get Template** button.
   g. Click on the OK button.
   h. Type `"State File"` in the Name property.

After creating the State File record set, the ResBuilder screen looks like **Figure 4**:

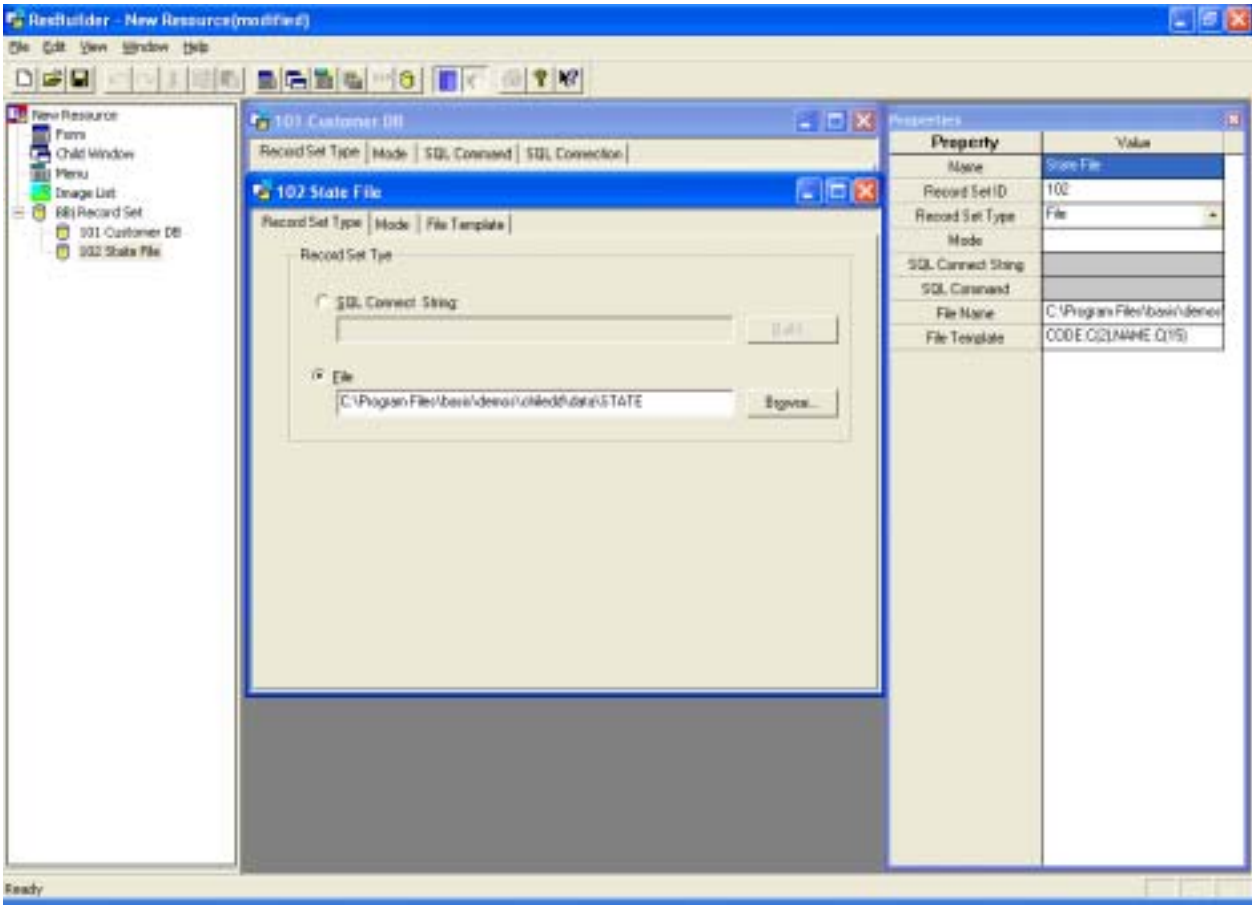

Figure 4. ResBuilder State File Record Set.

5. Create a Form by selecting the Form node, right mouse click, and select **Add** from the menu

   a. Type `"BBj Databound Controls Sample"` in the Name property

   b. For every column in the Customer DB except for state:
      - Add a static field and edit box to the Form
      - For each edit box, select **101 Customer DB** for the Bound Record Set
      - For each edit box, select the appropriate column for the Record Field

continued...

c. For the state column:

- Add a static field and a list button to the Form
- For the list button, select **102 State File** for the Fill Record Set
- For the list button, select **code** for the Fill Field

d. Add a BBjNavigator control (the last control tool button)

- Select **101 Customer DB** for the Target Record Set
- Select **101 Customer DB** for the Bound Record Set
- Select **cust_num** for the Record Field

After creating the form, ResBuilder looks like **Figure 5**:

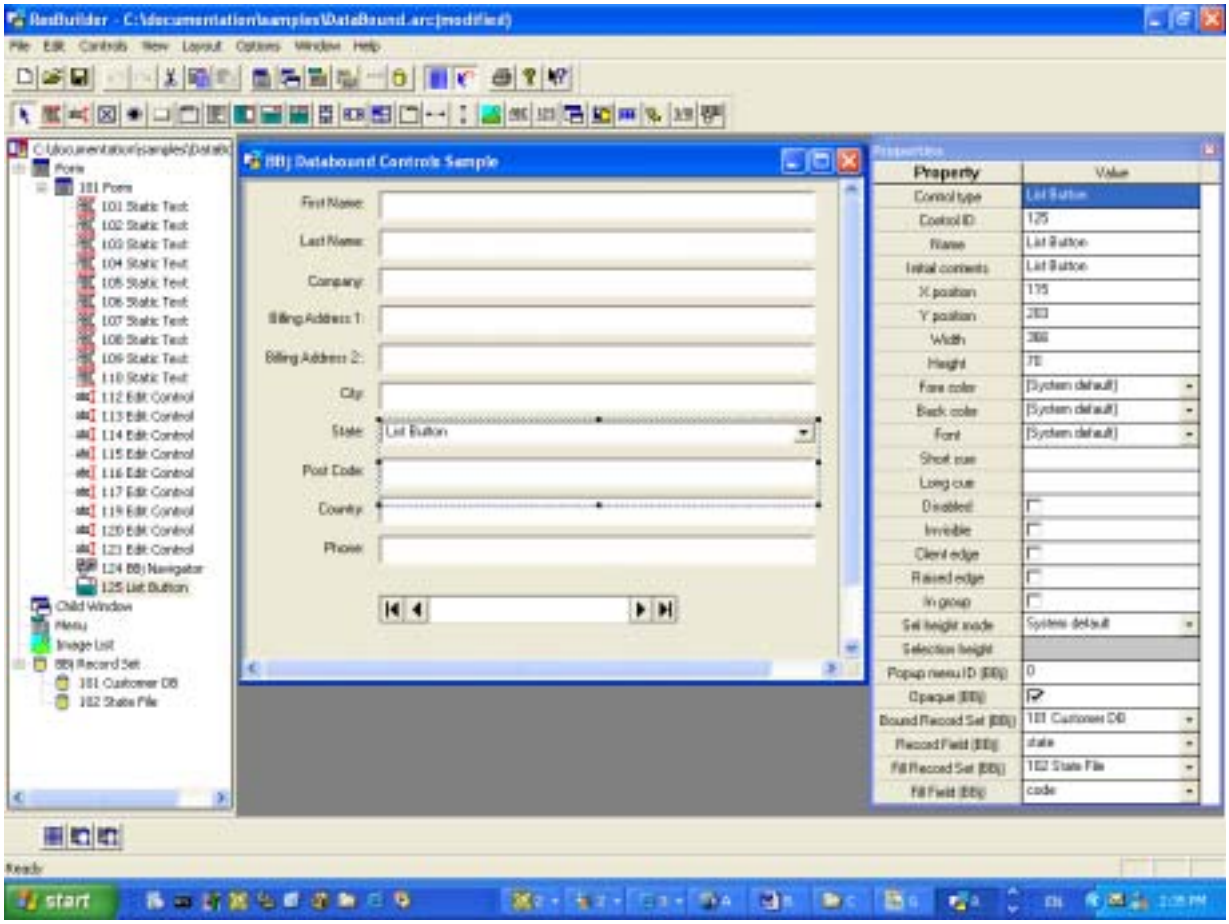

Figure 5. Completed Resource in ResBuilder.

6. Save the resource in `.arc` file format in the **`<bbj install dir>/GUIBuilder`** directory , i.e, **`DataBound.arc`**.
7. Exit ResBuilder

Step 3: Select the resource file

1. GUIBuilder now displays a dialog box for opening the resource file
2. Browse to and select the newly created `.arc` resource file
3. Click **OK** on the program options dialog

Step 4: Execute the program

Click the **Run** program toolbutton in GUIBuilder to see a working prototype of your new data bound application.