

Watch the Form Gen Wizard Trans “form” Data

By Robert Del Prete

The BASIS IDE now includes the Form Gen Wizard, a new rapid application development tool for developers. The Form “Gen”eration Wizard allows developers to create new forms quickly that are bound to data files and/or SQL result sets via record sets. From there, AppBuilder quickly creates a working application. Rapid development has never been so easy!

Overview

The secret to the wizardry behind the Form Gen Wizard is leveraging BBJRecordSet controls, first added to the BBJ® language back in 2003. Take time to refresh your memory of this powerful language concept by reviewing the foundational *BASIS International Advantage* articles on record sets referenced on page 22.



It is important to know that a record set definition does not need to rely on a data dictionary definition and can be as simple as defining a string template for an existing BASIS MKEYED file.

The Form Gen Wizard automatically creates controls and labels for each field the developer chooses to include in the form. It also adds a navigator, search buttons, and a databound grid, if desired. The Wizard places these controls in a window or child window and allows resizing of all controls via spinners. When added to the form, these controls are bound to the BBJRecordSet object and allow developers to deploy forms rapidly and as often as expanding applications require.

Create a Simple Form

To give you a clear picture of how the Form Gen Wizard works, follow this personal and straightforward how-to guide. Using the **ChileCompany** database, you will discover the power and magic of the Form Gen Wizard.

While I give specifics of how the Form Gen Wizard works, I am relying on your working knowledge of the BASIS FormBuilder and AppBuilder. If you are unfamiliar with these tools, it would be very beneficial to read the BASIS IDE articles referenced on page 22.

We are going to create a new “Master Detail” form in just a few steps using the Form Gen Wizard. First, create a new resource file in BBJ FormBuilder, which is included in the BASIS IDE. Choose **File | New** from the BASIS IDE, select the BBJResource template and name the new form **CCMasterDetail**.

With the form created and opened for editing, delete the default form and create a new BBJRecordSet. To create a record set in the resource, right-click on the **open_invoice.arc** in the BBJGui Inspector and choose **add RecordSet**. This displays the wizard that allows you to create the new record set in our resource. Choose an **SQL RecordSet** and apply the following query on the **ChileCompany** database:

```
select order_num, cust_num, order_date, ship_date, ship_zone, ship_method,
comment, salesperson, mdse_total, tax_total, frght_total from order_header
```

This creates record set 100 in our resource file. Let’s add another record set using the following query for our BBJDataBoundGrid control;

```
select line_num, item_num, qty_ordered, qty_shipped, price, disc_pct,
disc_amount, line_weight from order_line
```

After creating the record set, right-click on record set **RecordSet 100** in the **BBJGui Inspector** and select **Form Gen Wizard** from the popup menu. The dialog as shown in **Figure 1** now appears.

continued...



Robert Del Prete
Quality Assurance
Engineer

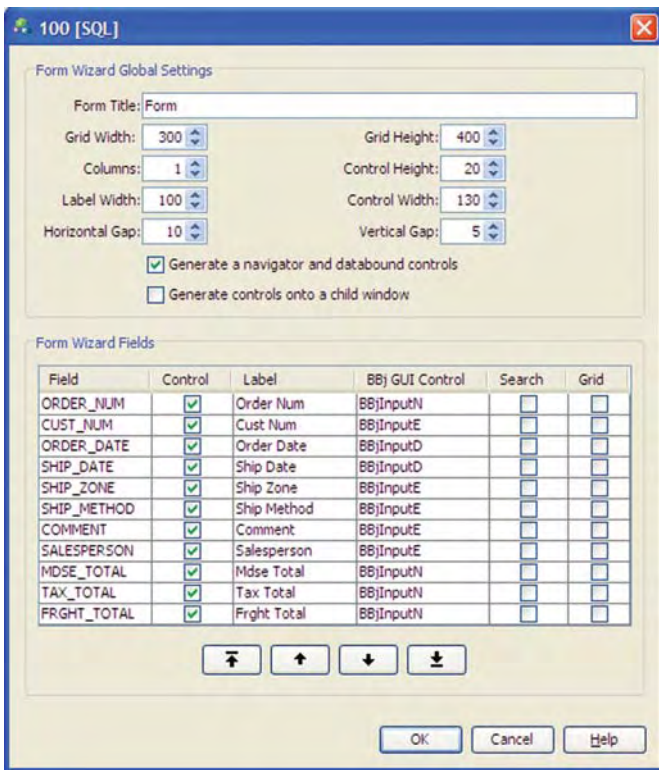


Figure 1. Default Form Gen Wizard dialog menu

Modify the Form

Let's explore some of the many options available in the Form Gen Wizard.

First, it is a good idea to rename the form, so change the name to **Chile Company Master/Detail Form**. Change **Columns** to 2. This will generate the controls in two rows. Change **Label Width** to 70. See **Figure 2**.

By default, the Wizard includes the option **Generate a navigator and databound controls**. Adding a BBJNavigator to the window provides an easy way to page through the records available from the SQL query. Another option is **Generate controls onto a child window**. This allows you to include the form in an already-created window and then adds that generated form as a child window.

The Form Gen Wizard displays all fields based on the SQL statement that we included earlier and by default, selects them to appear in the form. In our example, unselect the ORDER_NUM field since the navigator will use it. Our changes to the default settings are shown in **Figure 2** and our form starts to look like **Figure 3**. For display purposes, you can reorder the fields using the arrow buttons located under the list of fields, as desired.

Now, click [OK] and the Form Gen Wizard creates the form and adds the labels, fields, and BBJNavigator to the form. During this process, the Wizard binds the controls to the BBJRecordSet object.

Using FormBuilder, we can modify the resultant form and its controls like any other resource. Double-click on the newly created form in the BBJGui Inspector to load it into FormBuilder. Note that all of the record set bindings are complete and all controls appear on the form in a logical layout. Change the size of the window to a height of 335 and a width of 495. Also, change the bound field of the Navigator from CUST_NUM to ORDER_NUM using the **boundfield** property under the Navigator's properties.

Move the controls and labels in the second column as shown in **Figure 3**. Add a BBJGroupBox around these controls titled **Order Master Information**. Add an **Order Num:** label next to the BBJNavigator and a second BBJGroupBox below the first to accommodate the BBJDataboundGrid.

continued...

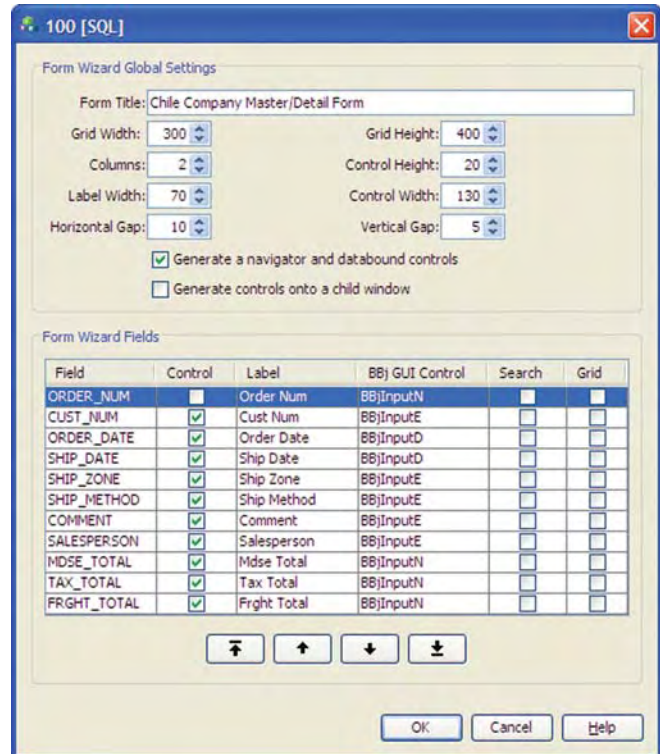


Figure 2. Edited version of the Wizard's default settings

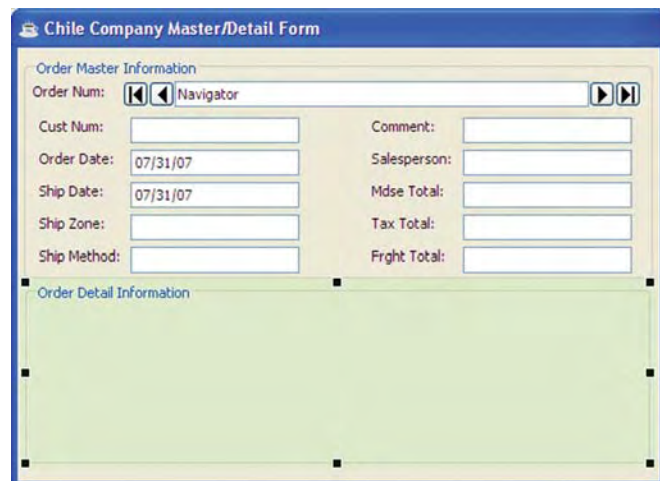


Figure 3. Resized controls and new group boxes

Field	Control	Label	BBJ GUI Control	Search	Grid
LINE_NUM	<input type="checkbox"/>	Line Num	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ITEM_NUM	<input type="checkbox"/>	Item Num	BBjInputE	<input type="checkbox"/>	<input checked="" type="checkbox"/>
QTY_ORDERED	<input type="checkbox"/>	Qty Ordered	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
QTY_SHIPPED	<input type="checkbox"/>	Qty Shipped	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PRICE	<input type="checkbox"/>	Price	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
DISC_PCT	<input type="checkbox"/>	Disc Pct	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
DISC_AMOUNT	<input type="checkbox"/>	Disc Amount	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>
LINE_WEIGHT	<input type="checkbox"/>	Line Weight	BBjInputN	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 4. Edited Form Gen Wizard for the BBJDataBoundGrid

Figure 5. Form with the new grid

The second record set that we created previously will allow the generation of the databound grid with the columns required for the application. The Form Gen Wizard can also create the **BBjDataBoundGrid**. Right-click on the second record set in the BBJGui Inspector and choose Form Gen Wizard. Deselect the **Generate a navigator and Databound controls** option and choose a **Grid Width of 455 and a Grid Height of 110**. Select the **Generate controls onto a child window** option. In the **Form Wizard Fields**, deselect all of the check boxes in the **Control** column and check all of the controls in the **Grid** column, **Figure 4**. Since we are only going to use the resulting child window, we do not need to modify the **Form Title**. Click [OK] to generate the new form.

Next, delete the extraneous form and drag the child window into the main form. Move it into the second group box we previously created. Change the child window association name to **Childdbgrid** so that we can reference it in the Init block. The updated form appears in **Figure 5**.

Create the Application

Now, we can convert the form into a running application via AppBuilder. Right-click on the **.arc** file in the Filesystems pane and choose **Create AppBuilder File**. We want the **.gbf** file to appear in the same directory so use AppBuilder's default settings for the project.

To complete the application, let's add a few more items in AppBuilder. Double-click on the newly created **.gbf** file to open up the project in AppBuilder. Register for a **WIN_CLOSE** event on the form and add a release to the **WIN_CLOSE** subroutine.

Next, add the Init event to registered events and add the following code to get the record set and record data for later use:

```

declare BBJNavigator          gb__navigator!
declare BBJRecordSet         gb__recordset!
declare BBJRecordData        gb__recorddata!
declare BBJDataBoundGrid     gb__dbgrid!
declare BBJRecordSet         gb__orderLineRS!
declare BBJWindow            gb__window!

REM - Get the recordset and recorddata for later use
gb__window! = bbjapi().getSysGui().getWindow(gb__sysgui_fin.current_context)
gb__childwindow! = bbjapi().getSysGui().getWindow(gb__win.Childdbgrid)
gb__navigator! = cast(BBJNavigator, gb__window!.getControl(100))
gb__recordset! = gb__navigator!.getTargetRecordSet()
gb__dbgrid! = cast(BBJDataBoundGrid, gb__childwindow!.getControl(100))
gb__orderLineRS! = gb__dbgrid!.getBoundRecordSet()

gosub Sync_Detail_Grid

```

continued...

Create a new code block for the application called `Sync_Detail_Grid` to synchronize the `BBjDataBoundGrid` to the current record as follows:

```

rem -----
rem Sync_Detail_Grid
rem -----

Sync_Detail_Grid:
rem Retrieve the current record:
gb_recorddata!=gb_recordset!.getCurrentRecordData()

rem Get the order number
orderNumber$ = gb_recorddata!.getFieldValue("order_num")

rem Detach the grid and close the orderline recordset
gb_dbgrid!.unbindRecordSet()
gb_orderLineRS!.close()

rem Recreate the orderline recordset based off of the new order number
connect$ = "jdbc:basis:localhost:2001?database=ChileCompany&user=admin&pwd=admin123"
modes$ = ""
sql$ = "select line_num, item_num, qty_ordered, qty_shipped, price, disc_pct, "
sql$ = sql$ + "disc_amount, line_weight from order_line where order_line=" + orderNumber$
gb_orderLineRS! = bbjapi().createSQLRecordSet(connect$, modes$, sql$)

rem Attach the DB Grid to the new orderline recordset
gb_dbgrid!.bindRecordSet(gb_orderLineRS!)

return

```

We also need code to handle the navigator events. Choose the **Navigator** in the form and add `NAV_FIRST`, `NAV_LAST`, `NAV_NEXT`, and `NAV_PREVIOUS` to registered events. The default code that AppBuilder includes will suffice but we need to un-remark two statements and add our code block for syncing the grid as shown in **Figure 6**.

continued...

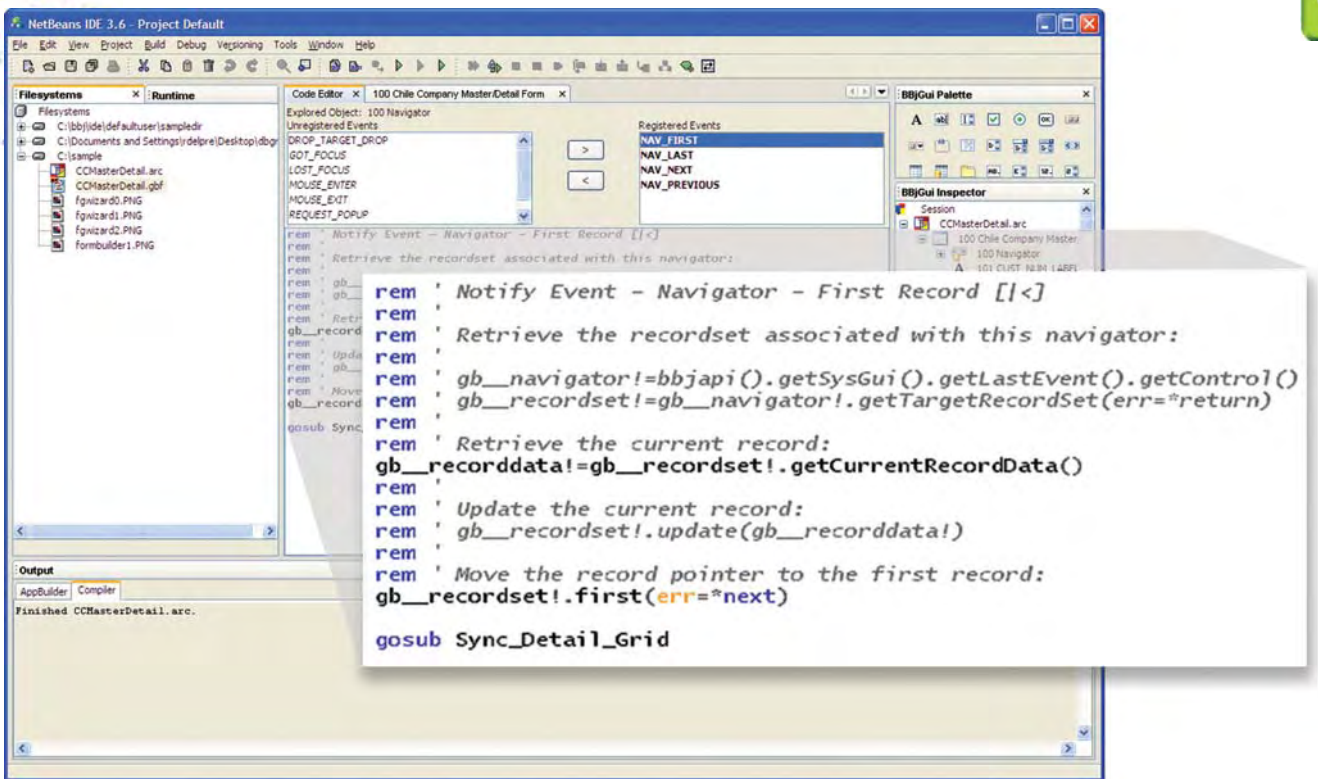


Figure 6. Modified `NAV_FIRST` code block

Once you have added the appropriate blocks of code, build and execute the application via AppBuilder. The resulting application appears in **Figure 7**.




Line Num	Item Num	Qty Ord...	Qty Ship...	Price	Disc Pct	Disc Am...	Line We...
1	000016	3	3	1.75	0	0	0.08
2	000002	12	12	6.5	0	0	2.72
3	000005	1	1	2.75	0	0	0.11

Figure 7. Chile Company Master/Detail Form

Results

The newly generated application displays order information based on order number. The navigator is functional and all of our BBJInputE and BBJInputN fields are bound to a record set allowing the form to display the selected fields from the database for each record. The databound grid displays the Order Detail Information.

Summary

In just a few steps, we created a functional application using the Form Gen Wizard. The form includes all the data we requested with appropriately created labels and fields as well as the databound grid and navigator. The best part of all, it took very little time and minimal effort to create. 



For information on record sets, read

Why Use the BBJRecordSet?

www.basis.com/advantage/mag-v8n1/recordset.pdf

Using the BBJRecordSet

www.basis.com/advantage/mag-v7n3/bbjrecordset.pdf

For information on the BASIS IDE, read

*AppBuilder: The BASIS IDE Gets RAD GUI
Development Integration*

www.basis.com/advantage/mag-v10n1/appbuilder.pdf

The State of the IDE

www.basis.com/advantage/mag-v9n2/ide.pdf

For information on databound controls, read

BBJ Databound Controls

www.basis.com/advantage/mag-v7n3/databound.pdf