The Magic of the Widget Wizard

By Brian Hipple
Quality Assurance Supervisor

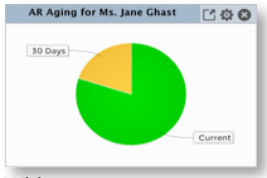
In BBj® 15.0 and higher, you have the ability to create widgets – one widget, a widget set, or a widget dashboard. But what is a widget? Not to be confused with a "whatchamacallit" or a "thingamabob," a widget is a small window that graphically displays some aspect of your data. You define the data it will display and set an optional refresh rate to make its display dynamic. So how do you create widgets without writing any code? Use magic – the new Widget Wizard!

The Widget Wizard is a BASIS development tool that generates BBj object-oriented (OO) code to create, manage, and display widgets. This easy-to-use WYSIWYG utility allows you to specify the attributes and data used to generate Javadoc-commented source code. It generates the code needed to create a dashboard of widgets or to embed your widgets in controls or windows, and to run them in a browser or in a thin client without needing to write a single line of code. This allows you to use your widgets on a desktop or a variety of mobile devices. The Widget Wizard not only stores the code it generates for you, it also displays the few lines of code to cut and paste into your own program that are necessary to invoke the generated code. Because the Widget Wizard is a BBj program, it runs on all supported BBj platforms and can be run in GUI or BUI (browser user interface) mode.

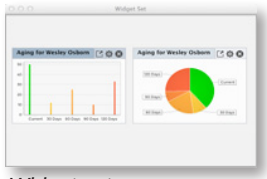
Why Widget Wizard?

So why do you need to use the Widget Wizard? After all, it only takes a limited number of lines of code to create a widget, widget set, or a widget dashboard as shown in **Figure 1**. There can be a number of reasons.

- You may have only seen how widgets work in BASIS products, demos, or Advantage articles, but not have any personal experience with writing code.
- You might not be sure where to begin and all you want is to see a good example.
- Maybe you just want to create widgets and add them to your own application without writing any code at all.
- Perhaps you want to see the differences between the widget, widget set, and widget dashboard using your own application data.
- You may not be familiar with the BBj OO code necessary to create and maintain widgets. The generated code can be inspected and manipulated until you achieve the desired knowledge and behavior.
- You might be wondering which widget would best display your application data.
- Even though you know your data, you may not be sure what SQL or recordset information is necessary to fill the widget with data, and you want to try out a few options.



Widget



Widget set

Widget dashboard

Figure 1. Widget creation choices

Using the Widget Wizard to learn, to experiment with the various option combinations, or to accelerate your development will save time that translates into less development cost. In no more than eight simple steps, the Widget Wizard asks all the right questions necessary to create your widgets. Let's walk through an example.

Choosing What to Build

In Step 1 of the Widget Wizard, select what you want the Wizard to build. The choices are a 'Widget', 'Widget Set', or 'Widget Dashboard' as shown in **Figure 1.** Each choice shows an image of how it might look when you are finished.

The simplest of the types is the 'Widget' that creates a single widget. For example, you may want to show information for a particular customer or account in a single pie chart. To create and display multiple widgets, choose a 'Widget Set'. This type is useful for displaying multiple charts or widgets depicting a graphical representation of datasets' key to the business in a single window. Lastly, the Widget Dashboard creates and displays multiple widget sets organized by category. A category is simply a logical grouping of widget sets that a tab control visually manages. AddonSoftware® by Barista® uses a widget dashboard called the "Digital Dashboard" (**Figure 2**) to create and display widgets in Accounting, Sales, and Manufacturing categories.

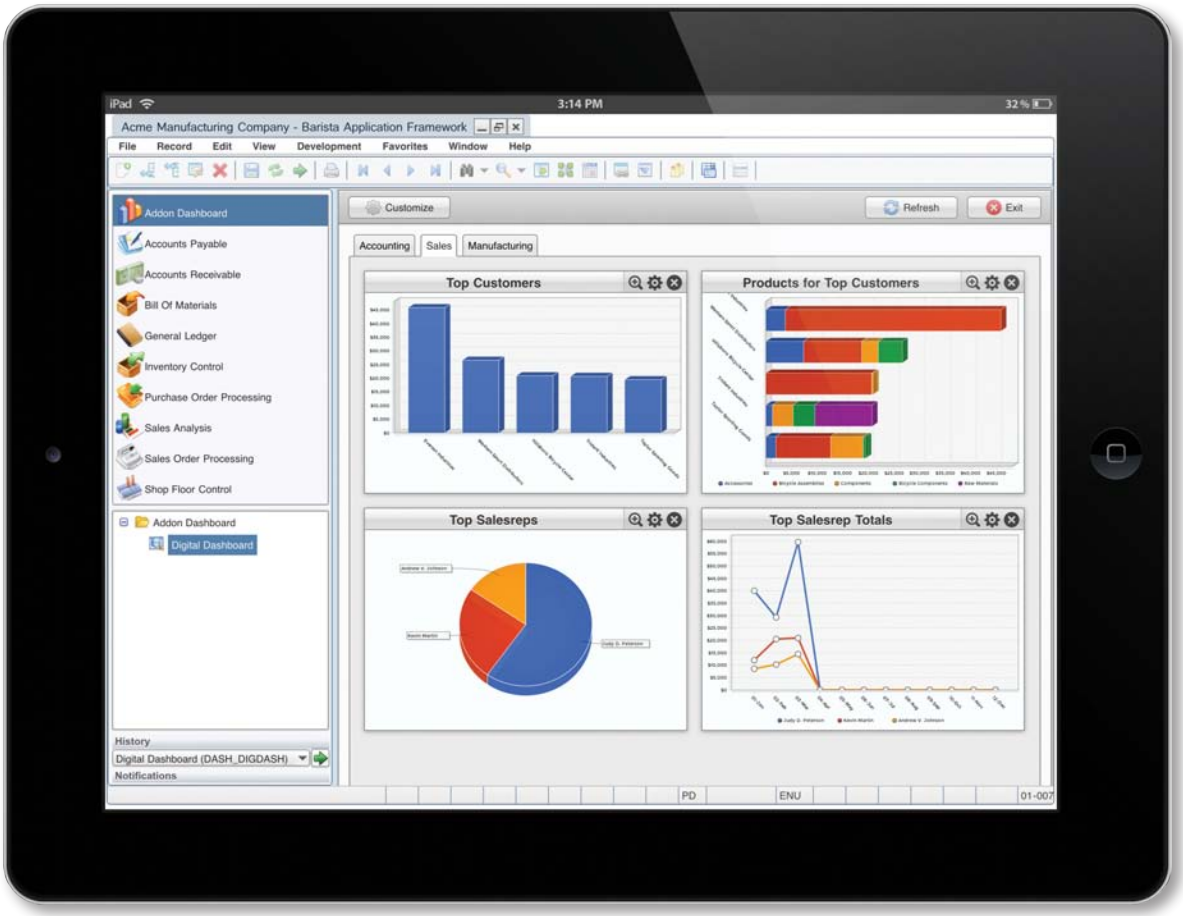


Figure 2. The Sales category in the AddonSoftware Digital Dashboard

Selecting a Widget Container

Step 2, shown in **Figure 3**, offers a choice of housing the widgets in either a window or a control.

Again, images provide a visual indication of what the build item might look like when selecting the associated option. The 'Window' option creates a BBjTopLevelWindow that is shown to the user in modal fashion. Program control will not return to the calling application until the window is closed by the user. The 'Control' option creates a BBjChildWindow that will embed the widgets into an existing BBjWindow. This option provides the caller with control over visibility of the widgets and the application program flow.

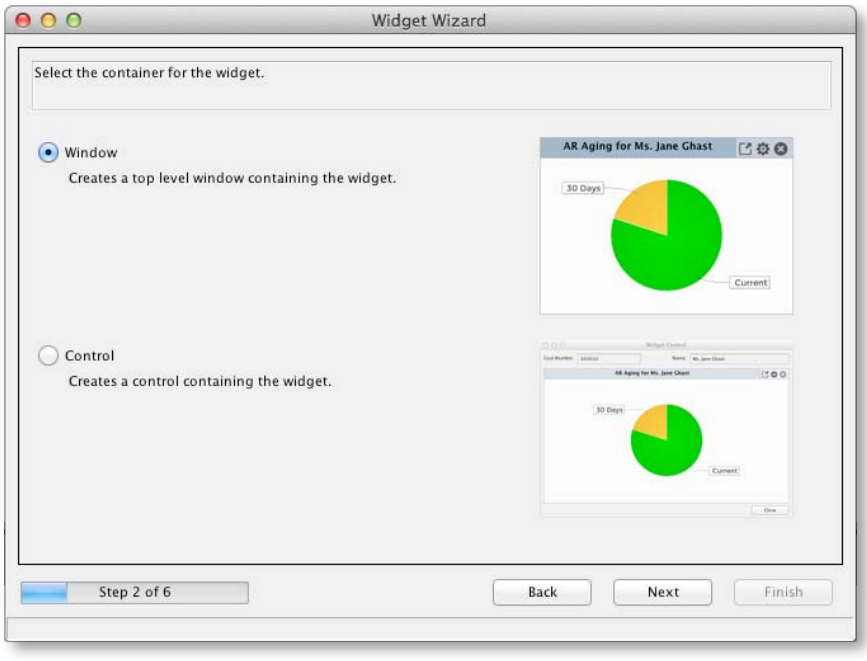


Figure 3. Select the container for the item to build

Choosing a Widget Type

There are thirteen (yes, thirteen!) different types of widgets that the Widget Wizard can create. The widget type can range

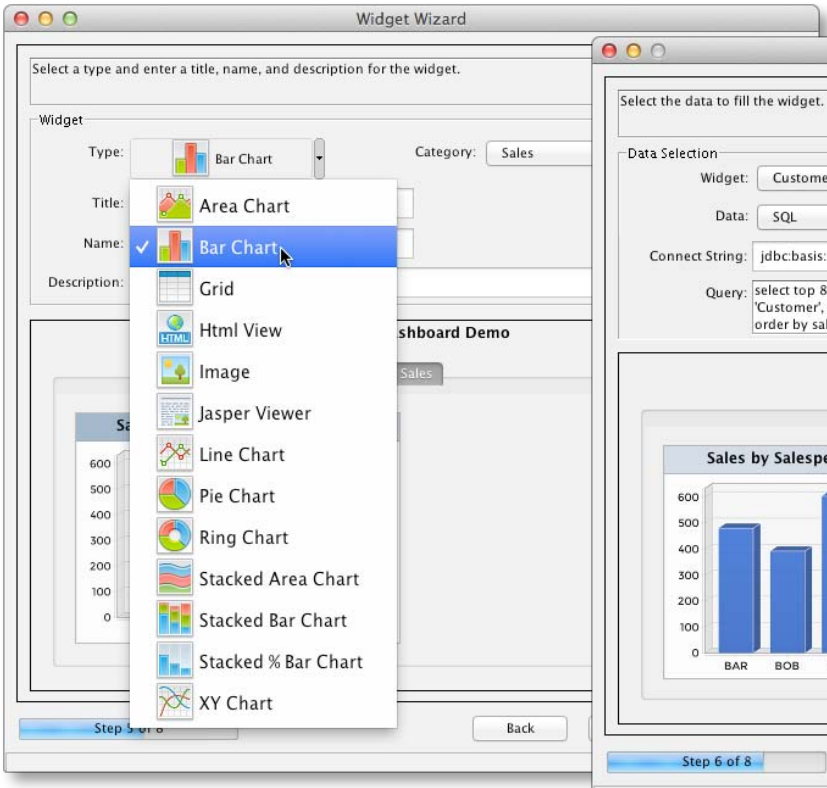


Figure 4. The available widget types

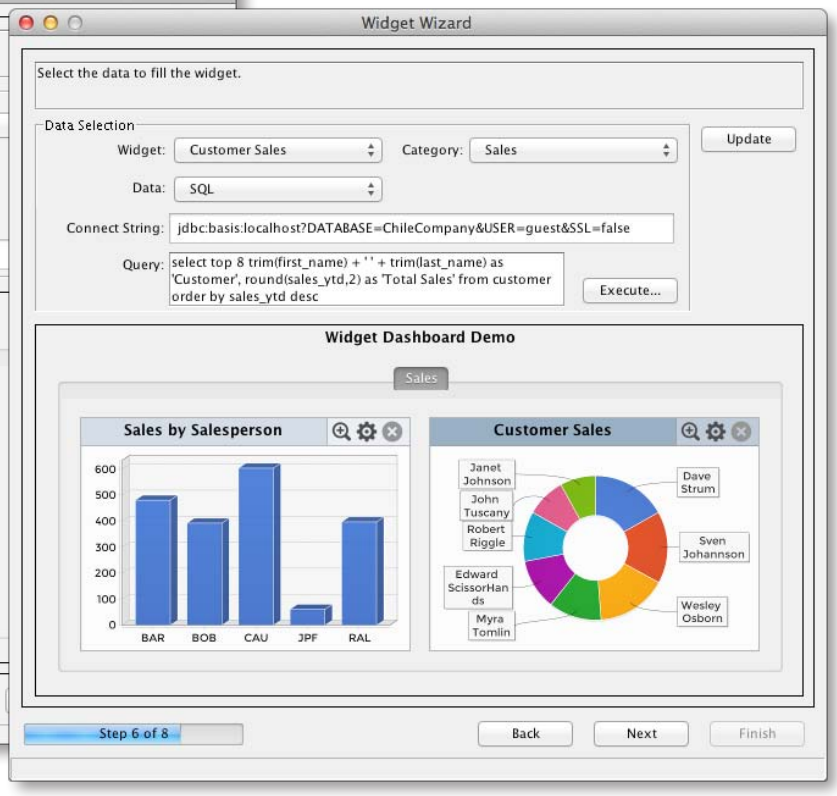


Figure 5. Select the data to fill the widget

from a basic chart such as area, bar, line, pie, or ring charts, to more advanced chart types such as stacked area, stacked bar, stacked percentage bar, or XY charts. The Widget Wizard also includes a 'Jasper Viewer' widget for displaying JasperReports, an 'Image' widget for images, and an 'Html View' widget for displaying HTML. You can even put a 'Grid' in your widget. Icons in the 'Widget Type' control provide a visual representation of each widget type. See **Figure 4.**

Obtaining Data

The most time consuming part of widget creation is figuring out what data is needed for the selected widget type and how to get it. Different widgets use different methods for obtaining the data. An 'Image' widget uses either a path or URL to the image, an 'HTML View' widget uses HTML text or a URL, and a 'Jasper Viewer' widget uses a connect string and JasperReport file for data. Data for the various chart widgets can be obtained either by SQL or a recordset. Depending on your use case, you may choose not to define data for the widget, in which case you can skip this step and proceed to the next wizard screen. For example, you may wish to integrate a widget into an existing application by taking advantage of the code that the Widget Wizard generates. In this scenario, the existing application already has the requisite data, so the Widget Wizard does not need to obtain the data again when creating the widget definition.

Figure 5 shows the 'Connect String' and the 'Query' string required to use SQL. The 'Connect String' defaults to the **`ChileCompany`** database on the local machine as the guest user. You can easily modify this to connect to your database and machine with the appropriate credentials. To view query results, select [Execute] after entering a query.

Once your query gives you satisfactory results, select the [Update] button to update the widget and fill it with data. If there is an SQL issue, a message box appears with the type of results required for the particular type of widget, and with a tip for the SQL query. See **Figure 6.**

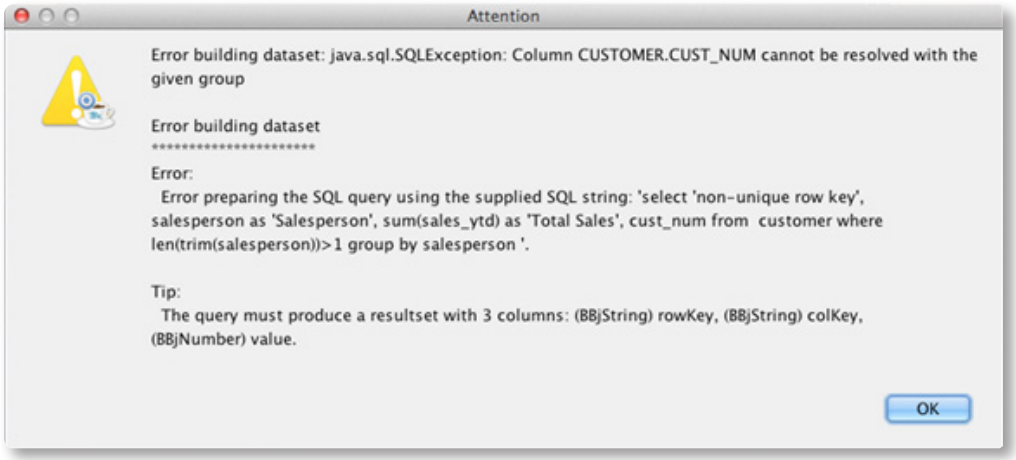


Figure 6. A message box showing the error and a tip

Code Generation and Execution

The end of the Widget Wizard bears the fruit of this process. You are ready to generate the BBj OO code and let the wizard write it out to a source file that you specify. The Javadoc-commented source code includes an error handler and is ready to launch if you select either of the 'Run in GUI' or 'Run in BUI' checkboxes. See **Figure 7**.

Clicking the [Next] button causes the wizard to actually generate and save the OO code. The generated code contains a public class that creates the widgets, which a BBj program can then access to instantiate and display the widgets. The code also includes a short sample at the top of the file (shown in **Figure 8**) that does exactly that so that your widgets actually run.

The public class definition, partly shown in **Figure 9**, contains all of the code necessary to replicate all of the widgets and categories you defined in the previous wizard steps. The code is complete as-is, but can be easily augmented to further modify any of the widgets defined within. For example, you can find your chart-based widget object in the program and add a couple of lines of code to modify the chart's colors or set a background image. Download the code sample at links.basis.com/14code.

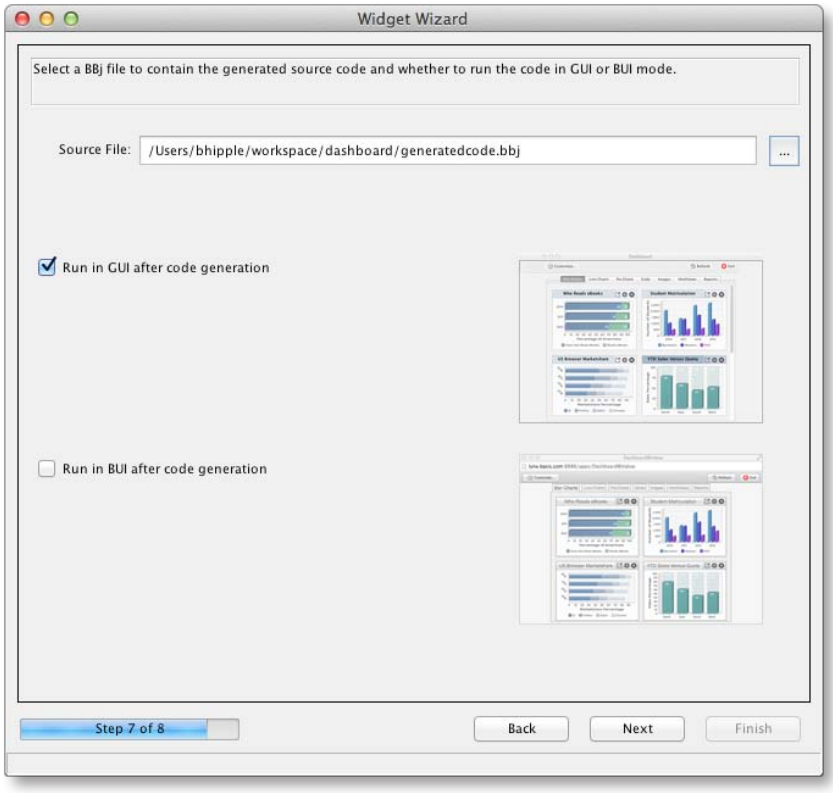


Figure 7. Specify a file for the generated code and choose how to run

```
rem /**
rem  * Sample code
rem  */
BBjAPI().getConfig().setOptionSetting("ERROR_UNWINDS",1)
seterr ErrorHandler
declare WidgetDashboardDemoWindow widgetDashboardDemoWindow!
rem Embedded Code Begin
use ::/Users/ndecker/Desktop/WWTest1.bbj::WidgetDashboardDemoWindow
widgetDashboardDemoWindow! = new WidgetDashboardDemoWindow()
widgetDashboardDemoWindow!.doModal()
rem Embedded Code End
release

ErrorHandler:
x=MsgBox(errmes(-1))
release
```

Figure 8. An excerpt from the generated code that instantiates and displays the widgets

```
rem /** WidgetDashboardDemoWindow
rem  * Generated class
rem  */
class public WidgetDashboardDemoWindow

    field public Dashboard Dashboard!
    field public DashboardWindow DashboardWindow!
    field public DashboardCategory SalesDashboardCategory!
    field public DashboardWidget SalesBySalespersonDashboardWidget!
    field public PieChartWidget SalesBySalespersonWidget!

    rem /**
    rem  * Constructor WidgetDashboardDemoWindow
    rem  */
    method public WidgetDashboardDemoWindow()
        rem Create the Dashboard
        dashboardName$ = "WidgetDashboardDemo"
        dashboardTitle$ = "Widget Dashboard Demo"
        #Dashboard! = new Dashboard(dashboardName$,dashboardTitle$)

        rem Create DashboardCategory
        categoryName$ = "Sales"
        categoryTitle$ = "Sales"
        #SalesDashboardCategory! =  #Dashboard!.addDashboardCategory(categoryName$,categoryTitle$)
```

Figure 9. The class definition in the generated code

The summary screen displays the code necessary for embedding the widget inside your application logic (see **Figure 10**). You can easily copy this code from the edit box and paste it into your program.

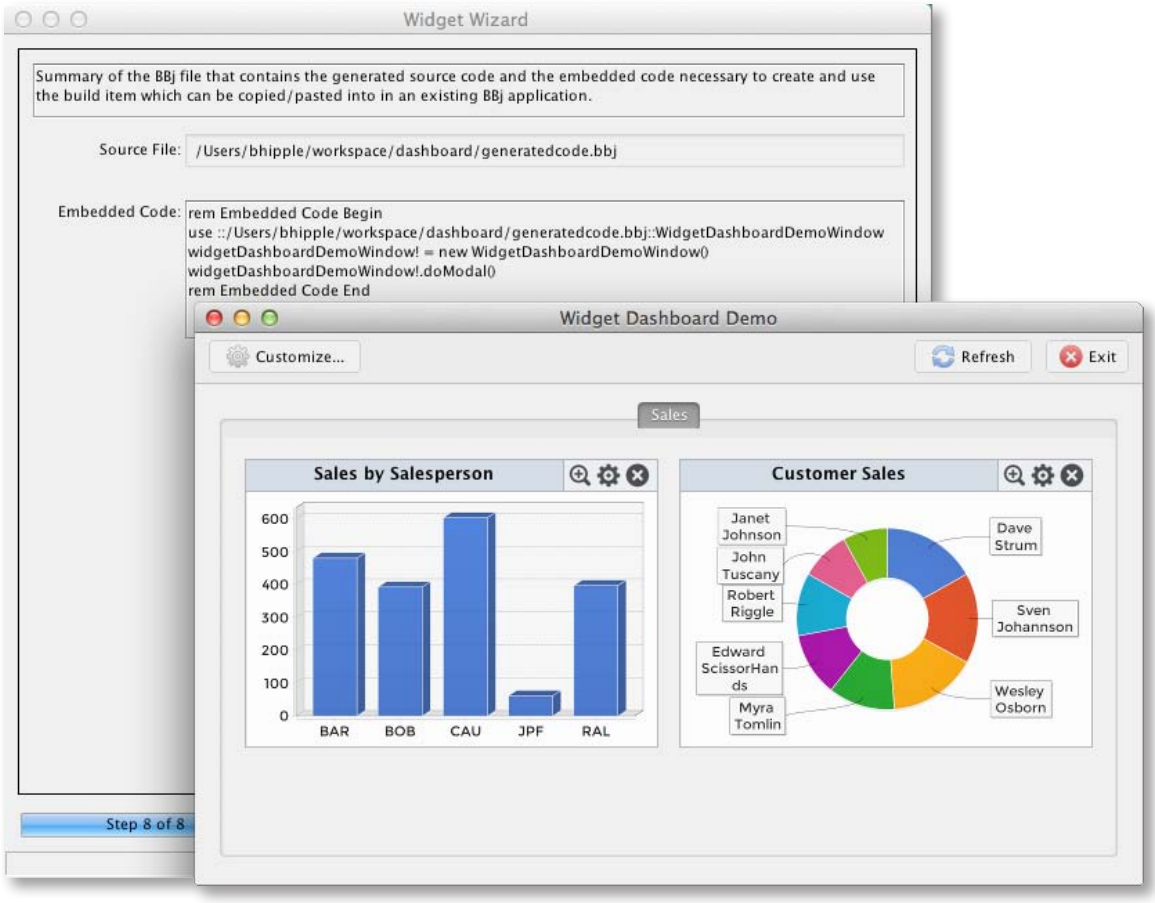


Figure 10. Running your code in GUI with a summary display

Summary

Using the Widget Wizard as a sandbox to create widget code is a fantastic way for you to get familiar with using widgets. To try it out, go to links.basis.com/getbbj and select the newest product for download. For assistance running the wizard, check out the tutorial-in-process at links.basis.com/widgetwizard. It has been our experience that once you create the first couple of widgets, creating more becomes a snap. The Widget Wizard eliminates the learning curve – like magic! ■

- Read related articles in this issue
  - *Easier Decision Making With the Dashboard Utility*
  - *Dash Boredom With the Dashboard Utility*
  - *AddonSoftware's Digital Dashboard Takes Off*
- Refer to *Dashboard Utility Overview* in the online Help
- Watch these past Java Breaks
  - *Adding the New Digital Dashboard to Your App*
  - *Embedding Widgets in Your BBx App*
  - *The New Widget Wizard - Dashboards and Widgets Without Any Code!*
- Download the code sample