A Tour of the BBjCharts API

By Shaun Haney

With the introduction of the BBjChart API in BBj® 7.0, it is now convenient to display application data graphically in the form of a chart. Charts add value to applications by allowing users to answer questions with a quick glance at a chart. Are we attaining our sales goals? Which geographic areas are generating the most revenue? Is our new Web site attracting customers?

Imagine Florence's Bakery, a local business that makes gourmet delicacies such as Lemon Raspberry White Chocolate Mousse Cake, Flourless Chocolate Torte, and Black Bottom Tiramisu. This bakery thrives locally and several customers gave feedback to Florence that no other bakery within the state makes such unique cakes. After some research and discussion, Florence, the proprietress, determined that an e-commerce system would be valuable for expanding her customer base outside of her neighborhood. Now, one year after the e-commerce system launched, it is in full swing. Still, one question looms, "Have sales really improved as much as it appears?" The BBjCharts API can help provide this information at a glance.

The Journey Begins

The BBjCharts API includes three "ready-to-use" charts and a highly customizable chart. The ready-to-use charts include a pie chart called BBjPieChart, a line chart called BBjLineChart, and a bar chart called BBjBarChart. BBjGenericChart is a highly customizable chart control that allows the BBj programmer to create any of the dozens of additional charts in the JFreeCharts API. This journey will show us how Florence can use the BBjCharts API to answer her business questions. Along the way, we will explore uses for the different kinds of BBjCharts and learn how to create them.

Figure 1. Pie chart created using AppBuilder

The First Stop – Creating a BBjPieChart in the BASIS IDE

Using AppBuilder, it is easy to create a chart like the one in **Figure 1** and populate it with just a few lines of code. Florence wants to know what portion of this year's revenue e-commerce generated. Pie charts represent proportional data and since Florence wants this data quickly, AppBuilder is the ideal tool.
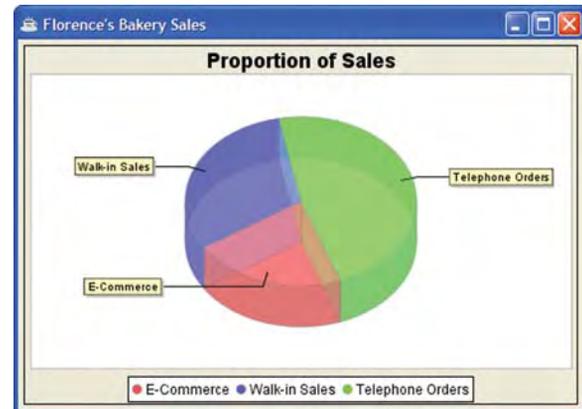
To create the pie chart in AppBuilder, first create a resource and place the BBjPieChart control onto the resource's main window. Right-click the resource file and select **Create AppBuilder File**. Double-click the new AppBuilder file to open it, and enter the code in **Figure 2** into the Init block (or cut text from the **appbuilder.txt** file in the downloadable .zip file referenced at the end of this article):

```
declare BBjPieChart pieChart!

REM Get the pie chart control from the window.
pieChart!=cast(BBjPieChart, BBjAPI().getSysGui().getActiveWindow().getControl(100))

REM Normally the data would be retrieved from a database, but the values
REM are read from the data statement for this self-inclusive example
while 1
    dread sliceName$, sliceAmount,end=*BREAK
    pieChart!.setSliceValue(sliceName$,sliceAmount)
wend

data "E-Commerce",50319.80, "Walk-in Sales",73506.80, "Telephone Orders",112055.10
```

Figure 2. Sample code from **appbuilder.txt**

After entering this code, build the AppBuilder file and run it to see the chart in all its brilliance.

Scenic Overlook – the BBjPieChart

AppBuilder creates a pie chart quickly and easily, doing all the work for us. However, would a BBj programmer generate the same pie chart using the BBjCharts API within a hand-coded program? The BBjPieChart is the easiest of the charts in the BBjCharts API to create and populate so that will be the next stop on our journey.

Shaun Haney
Quality Assurance
Engineer

BBjPieChart Creation

First, create the pie chart by invoking the method addPieChart. It takes seven arguments; five are common to all BBjControls, and the last two, which are boolean values, are specific to BBjPieChart. They indicate whether to display a legend and whether to display the chart with a 3D effect. For example, if creating a pie chart with the following line of code, the two 1's at the end of the call result in a pie chart with a legend and a 3D effect.

```
pieChart!=win!.addPieChart(101,0,0,640,480,1,1)
```

BBjPieChart Population

To populate a BBjPieChart, simply add slices of any size to the pie. BBjPieChart maintains a running total and displays each slice according to its portion of the total. Name each slice when adding it to the pie and the name appears in the legend. We add slices by invoking the method setSliceValue, which takes a slice name and a value as arguments and adds them to the pie chart. For example:

```
pieChart!.setSliceValue("A",2)
pieChart!.setSliceValue("B",5)
pieChart!.setSliceValue("C",3)
```

These statements will create a pie chart with a total value of 10. Slice **A** would occupy 20% of the pie; slice **B** would occupy 50% of the pie; and slice **C** would occupy 30% of the pie.

BBjPieChart Example

Read and run the sample code **piechart.src** shown in **Figure 3**. The resulting chart in **Figure 4** showed Florence that e-commerce generated approximately 20% of her revenue for this year.

```
REM Florence's Bakery: Portion of Sales From E-Commerce

REM declare our standard GUI Objects
declare BBjSysGui sysgui!
declare BBjTopLevelWindow win!
declare BBjPieChart pieChart!

sysgui!=BBjAPI().openSysGui("X0")
win!=sysgui!.addWindow(10,10,500,325,
: "Florence's Bakery: Portion of Sales from E-Commerce")

REM Create our pie chart
pieChart!=win!.addPieChart(win!.getAvailableControlID(),
: 5,5,490,315,1,1)

REM Populate the pie chart's data.  These values are
REM hardcoded for simplicity.  Data would normally
REM be read from a database or data files.
pieChart!.setSliceValue("E-Commerce",5031.98)
pieChart!.setSliceValue("Walk-in Sales",7350.68)
pieChart!.setSliceValue("Telephone Orders",11205.51)

win!.setCallback(win!.ON_CLOSE,"goodbye")

process_events

goodbye:
release
```

Figure 3. BBjAPI code sample for the pie chart

Figure 4. Pie chart created using BBjAPI

Scenic Overlook – BBjLineChart

Line charts are especially helpful to show the progress of a series of data over a given period of time. One common use for the line chart, for example, is showing the rise and fall of prices on stocks. The line chart in **Figure 5** shows how sales from the e-commerce site compare with sales from telephone orders and walk-in sales.

The BBjLineChart Dataset: Series and XYValues

Every chart in the BBjCharts API has a dataset. The BBjPieChart's dataset is very simple and does not require preparation before creating the chart. By comparison, BBjLineChart's dataset requires slightly more preparation. The dataset consists of one or more series and each series consists of any number of Cartesian coordinates known as XYValues. A series represents one of the lines in the chart and the XYValues are the points that this line connects. In order to create a BBjLineChart, it is necessary to specify the number of series, i.e. lines, in the chart.

Figure 5. Line chart created using BBjAPI

Creating a BBjLineChart
Using the method addLineChart, specify a label for the X axis, a label for the Y axis, the number of series, and whether to display a legend as shown below:

```
lineChart!=win!.addLineChart(101,0,0,640,480,"Month",
: "Sales in U.S. Dollars",3,1)
```

Populating a BBjLineChart With a Series
Populating the line chart is a matter of naming the series and then specifying a set of XYValues for each series. Each series displays as a line connecting its specified XYValues. There is no limit to the number of XYValues that one can specify and no need to adhere to any discrete interval along the X-axis.

To name the series, use setSeriesName. This method takes a zero-based index, and a string indicating the name of the series. For example, to name the second line in the chart **Walk-in Sales**, type:

```
lineChart!.setSeriesName(1,"Walk-in Sales")
```

After naming the series, add XYValues to the series for its line to appear by calling setXYValue. This method takes the series' zero-based index, an x-value, and a y-value. In the example, the x-value represents a month and the y-value represents a sales amount for the month.

Suppose that in January, the bakery's first month, walk-in sales were \$5000 and then rose to \$7500 in February. The code for these facts, resulting in a rapidly rising line in the graph, would be as follows:

```
lineChart!.setXYValue(1,1,5000)
lineChart!.setXYValue(1,2,7500)
```

BBjLineChart Example
Run the sample `linechart.src`, which shows the progress of three types of revenue for Florence's Bakery: sales from the e-commerce site, which did not launch until the 13th month; walk-in sales, and telephone orders. From the data in this example, Florence was able to determine that e-commerce was a significant source of revenue.

Scenic Overlook – the BBjBarChart

Bar charts use ingots or bars to give side-by-side comparisons of different series of data. Florence wants to know whether sales improved after the introduction of e-commerce. A bar chart would be useful for a monthly comparison of this year's and last year's sales.
Before creating the BBjBarChart, it is important to understand how its dataset is organized.

The BBjBarChart Dataset: Series, Categories, and Values
The BBjBarChart's dataset consists of series, categories, and values as shown in **Figure 6**.

Series - represents particular entities that the chart compares. Each bar color represents an entity and the names of these entities will appear in the legend. The entities compared in this example are **Last Year** and **This Year**.

Category - a group of the entity's values. In this case, the category is **months** so every **month** has a value for **Last Year** and **This Year**.

Value - the actual unit of data. In this case, **a value** represents total sales during a particular month for a particular year such as sales for the month of March 2006.

In another example, Florence wanted to find out the most popular types of cake that each sales associate sold. To compare types of cake sold by five sales associates, set `Employee` as the series and `Cake Type` as the category. This resulting bar chart would display a label for each type of cake at the bottom of the chart and one bar color would represent each employee.
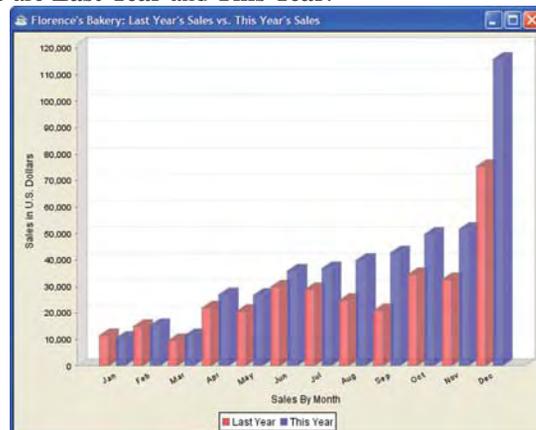
Figure 6. Bar chart created using BBjChart API

Creating the BBjBarChart

To create a BBjBarChart, use BBjWindow's factory method, addBarChart. This method takes twelve parameters, including a label for the X axis, a label for the Y axis, the number of series, the number of categories, and boolean values specifying whether to show a legend, display the bar chart with 3D bars, and whether the bar chart is a horizontal bar chart.

For example, to create a bar chart with a control ID of 101 in the top left corner of the window with dimensions of 600 by 450 pixels, an X axis labeled **Sales By Month** and a Y axis labeled **Sales in U.S. Dollars**, 12 categories, 2 series, a legend, the bars in 3D, and without horizontal orientation, enter this statement:

```
barchart!=win!.addBarChart(101,0,0,600,450,"Sales By Month","Sales in U.S. Dollars",12,2,1,0,0)
```

Populating the BBjBarChart

In order to populate the bar chart, first give each series and category names, then set values for each series within a given category. If a category is unnamed, it will not appear on the chart even though the category exists and can accept values.

The method provided for setting the series names is setSeriesName. This method takes a 0-based index and a string for the name of the series. The code to set up our series, This Year and Last Year looks like this:

```
barChart!.setSeriesName(0,"Last Year")
barChart!.setSeriesName(1,"This Year")
```

Similarly, the method to set each category name is setCategoryName and the index is 0-based so indexes 0 through 11 represent the 12 months of the year. The code to set a category name looks like the following:

```
barChart!.setCategoryName(6,"Jul")
```

The dataset for a bar chart is conceptually a two-dimensional array with the series as the first index and the category as the second index. Use these indexes to set each value. The method to set values in BBjBarChart, setBarValue, takes a series index, a category index, and a value. For example, to set this year's sales for July to $4,358, use the following line of code:

```
barChart!.setBarValue(1, 6, 4358)
```

BBjBarChart Example

Study and run the sample named **barchart.src**. The sample shows that sales indeed experienced an improvement over last year's sales with the introduction of e-commerce.

Invitation to Explore Deeper Terrain

The three basic charts in the BBjCharts API are sufficient for most purposes, but sometimes an application requires a different kind of chart. The online supplement to this article referenced at the end of this article covers how to customize the three basic charts using getClientChart and how to create dozens of other types of charts using BBjGenericChart. BBjGenericChart allows developers to create ring charts, box and whisker charts, Gantt charts, and other charts listed at www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/ChartFactory.html.

The Final Stop

Even for such simple business needs as Florence's Bakery, the new BBjCharts API will display data graphically, easily, and effortlessly. If the familiar bar charts, line charts, and pie charts do not meet specific needs or do not present the data clearly enough, developers have all the options available under the sun using BBjGenericChart to access the JFreeChart API and produce many diverse charts. Clearly, the new BBjCharts API will be useful in a wide variety of BBj applications. ⏵BASIS⏴

Download the sample code from
www.basis.com/advantage/mag-v11n1/charts.zip

There is more!

For a deeper look into charts go to
www.basis.com/advantage/mag-v11n1/ChartsDeeperAdv07.pdf

See all of the available charts in the JFreeChart library by visiting
www.jfree.org/jfreechart/samples.html