Maximizing the Power of the Digital Dashboard Widget

by Nick Decker

Shortly after BASIS released the Dashboard Utility with BBj® 14.0, ideas, suggestions, and requests started to trickle in from the Business BASIC community. Never being content to rest on their laurels, the BASIS engineers began to work in earnest to convert the flow of enhancement requests into a growing list of implemented features. The new widget and dashboard capabilities run the gamut from user-facing visual and aesthetic improvements to empowering developers with new filter controls and access to user's click events and data selections. This article tours some of the most notable additions to the Dashboard Utility, many of which you can experience first-hand by running some of the demos on the BBx BUI Showcase web page.

Customize the User Experience

Regardless of whether a developer embeds a single widget into an existing application or creates a full-blown dashboard, their choices for widget size, chart color theme and fonts can profoundly impact legibility. While the defaults are acceptable for many common situations, certain little touches like increasing font size or making widgets larger can improve clarity and help the user interpret the data more readily. Wouldn't it be great if the user could override some of those options chosen (or perhaps ignored) by the developer? The latest version of the Dashboard Utility gives the user complete control over widget sizing, color themes, chart fonts, and customized font scaling via the updated Customize screen shown in **Figure 1**.

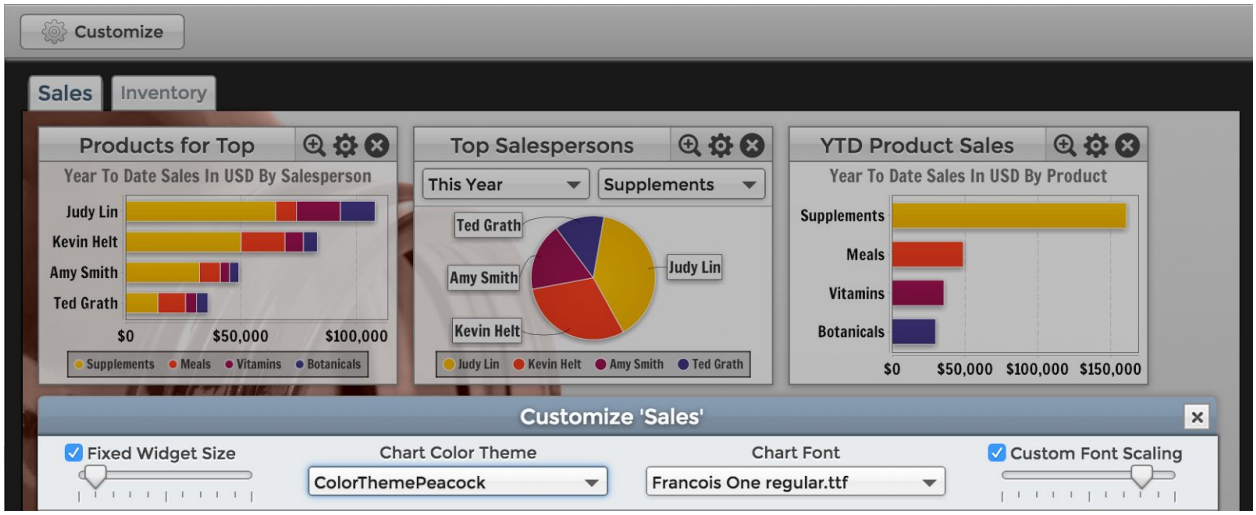

Figure 1. The updated Customize screen (bottom) allows the user to easily modify charts displayed in the dashboard (top)

For widget size, the user can choose to keep the default sizing model where the widgets are dynamically sized to maximize the use of the available dashboard space. That means that whenever the user resizes the dashboard window, the utility will perform a set of calculations and resize and reposition the widgets to fill the new window size. Alternatively, the user can choose custom widget sizes by interactively sliding the widget sizing control. As the slider value changes, the widgets change their size and placement in real-time so that the user may configure widgets that are tiny, enormous, or any size in between, as shown in **Figure 2.** Regardless of the available window size, the widgets keep their designated size and the number of rows and columns for the widget layout may change depending on the size of the window.

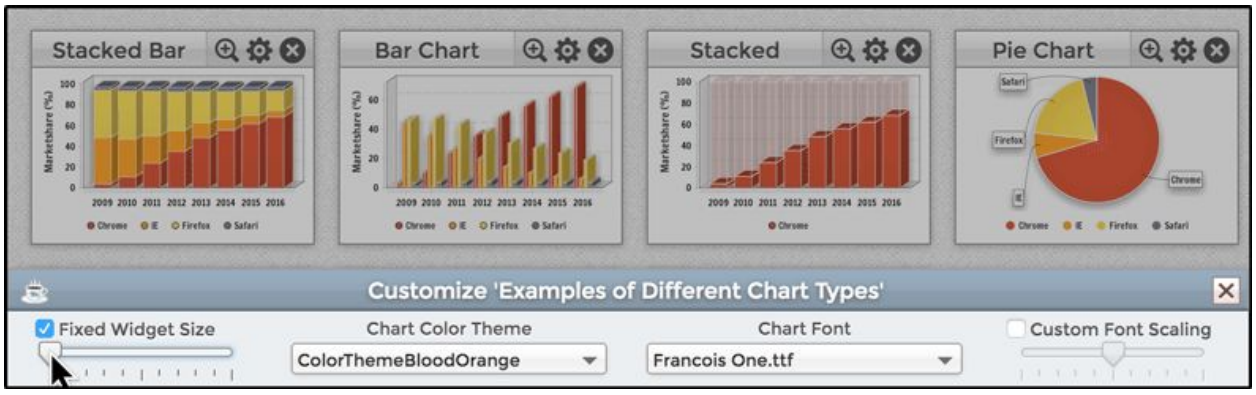

Figure 2. The effects of changing the widget size in the Customize screen

Each of these parameters allow the user to fine tune their frequently-used dashboards. Best of all, dashboards save these customizations on a per-user and per-dashboard basis. This means that the charts won't just look nicer and be easier to read this time, but whenever the user runs the same dashboard in the future the utility automatically applies all their chart refinements at

load time. If at any time the user wants to start from scratch, they can simply choose the default values to return everything back to its original state. To watch a video of the new customization capabilities, click on this link.

Filter the Data

Adding one or more filters to a widget goes a long way to increasing the flexibility of the widget, making it far more powerful and interactive. The Dashboard Utility has always provided filters using the BBjListButton control, giving the user a pre-selected list of choices. The filter values are set by the developer and are commonly used to change the underlying SQL query, such as restricting reports to particular time frames. In those cases, the developer must know the possible user choices when designing the widget. This makes sense for fixed periods such as last quarter, last year, or even what type of data to include in the report as shown in **Figure 3**.

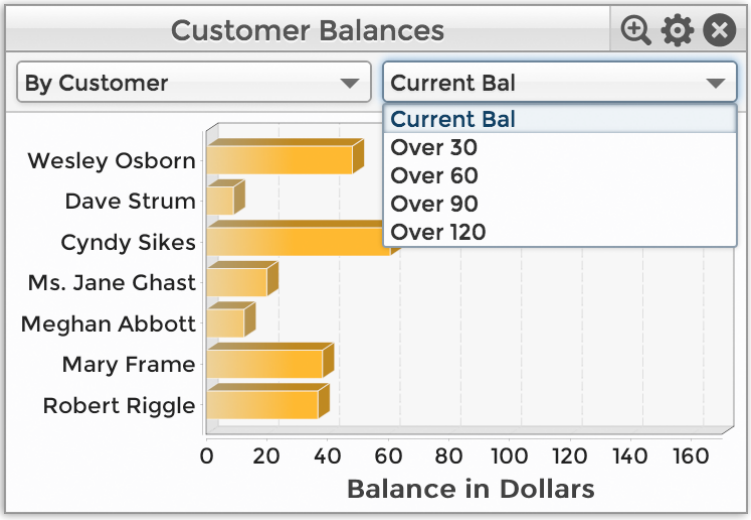

Figure 3. Choose what data to include in the chart via the widget's filters

But what about the cases when the developer may not know what a reasonable value may be, such as restricting reports based on total sales volume? That's where two new filters come in, based off of the BBjListEdit and BBjEditBox controls. The BBjListEdit-based filter is like the previous BBjListButton filters, with one major difference: the user can type in their own values in addition to selecting one from the drop-down list, as shown in **Figure 4**.

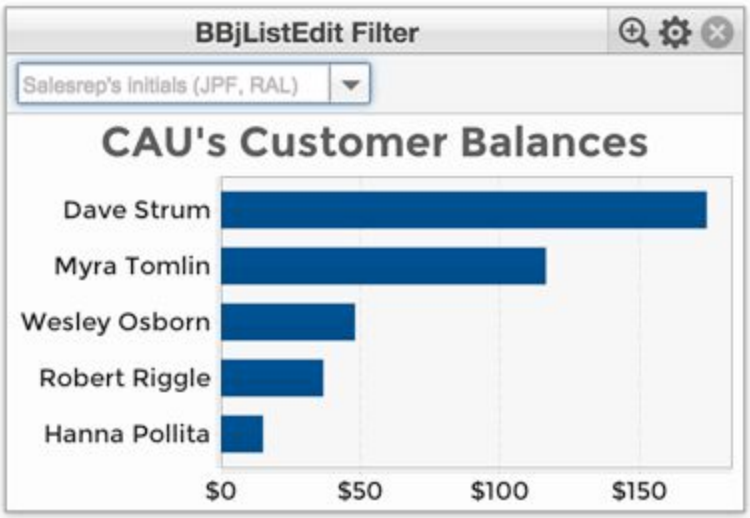

Figure 4. Selecting and editing values in the new filter control

The BBjEditBox filter takes this flexibility to another level, as it only contains a single value but is completely editable by the user. These filters allow for much greater input freedom, and the developer still has the ability to sanity check the user's values. The developer has access to all the relevant filter selection information in their custom callback section. So if the input provided by the user doesn't make sense for the query, such as non-numeric input, the developer can alert the user with a message box instead of modifying the widget. But, if the user provides reasonable input, the developer simply sets the new SQL query on the widget and calls the refresh method. The widget updates, showing new information based off of the user's input.

Because the new filters use BBj input controls, they also support BBj's new placeholder text. The placeholder specifies a short hint describing the expected value of an input field, as shown in **Figure 5**. This text appears when the input control is empty, and immediately disappears as the user begins to type in the control. The placeholder text goes a long way to improving the usability of many interfaces by guiding the user to enter correct data in the expected format.

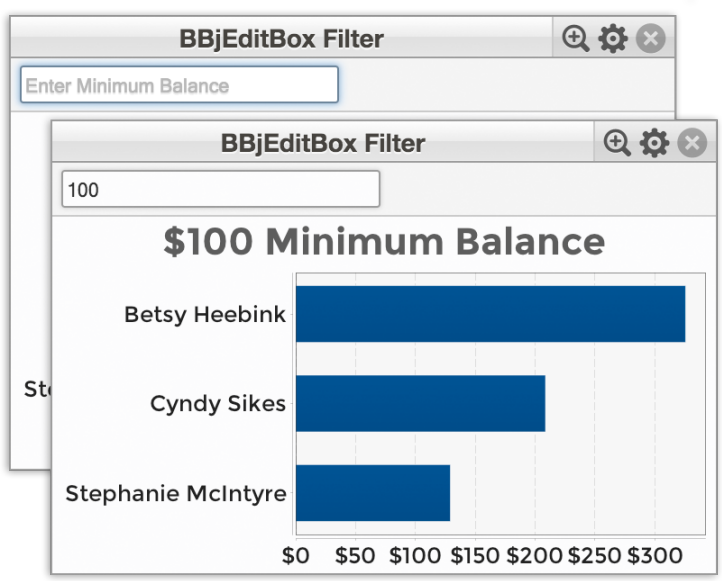

Figure 5. The BBjEditBox-based filter provides placeholder text when empty (above) and displays the text normally (below) after the user types in a value

Export the Data

The Dashboard Utility's widgets provide key metrics and data aggregation capabilities in a variety of formats. While it has always been possible to save an image of a dashboard's chart or email it to a colleague, sharing or manipulating the underlying data was not possible - until now! The latest round of updates gives users the power to export a grid or chart widget's data into a delimited file, providing complete control over the exported file name, the field and record delimiter types (such as a comma, tab, carriage return, etc.), and even text qualifiers. The end user can then import the resultant file into a variety of programs, such as spreadsheets and mathematical and analytical applications. Because the comma separated value (CSV) file format is ubiquitous and has extensive software support, users can streamline the export process and view the widget's data in Excel for Windows, Numbers for the Mac, or any other program that the user associated with the CSV extension on their computer. **Figure 6** shows how to access the widget's View Data option and how the underlying data appears when viewed in a spreadsheet.

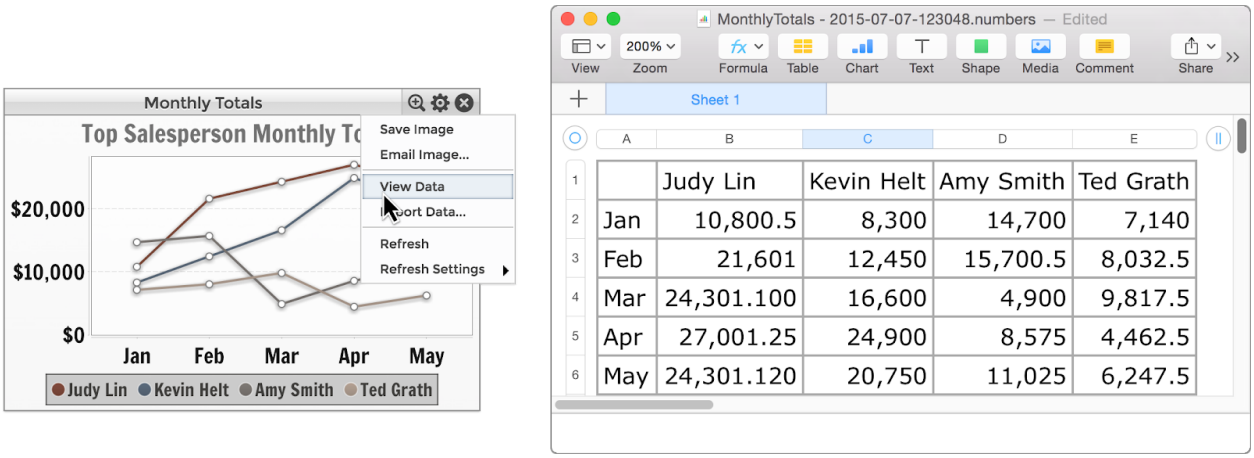

Figure 6. Selecting a widget's View Data option (left) and viewing the exported data in a spreadsheet (right)

After the widget has exported its data into the spreadsheet, it is possible to massage or analyze the data by making use of the application's built-in formula library. Merging the data with another spreadsheet is a simple matter of copying and pasting, and printing the data is just a click away.

To watch a video of the new export capability, click on this link.

Click for Tips and Drill Down for More Detailed Data

With the introduction of click events, chart tips, and developer-defined click routines many widgets now respond to the user's clicks in a context sensitive manner. For example, clicking on a slice in a PieChartWidget or a bar in a BarChartWidget causes the widget to briefly show a helpful tip describing the clicked data entity. Charts do this automatically, so anyone can click on a bar in a bar chart to find out more information about the chart's row key, column key, and value associated with the bar in question.

Developers can also write custom code handlers for both left and right click events, extending the usefulness of the widgets in a variety of ways. One of the more popular use cases for reacting to the user's mouse click is the ability to drill down from summary level information to a more detailed view of the data. This means that the user can now click on a salesperson's year-to-date sales figures and get a detailed breakdown of their sales by region, store, customer, or whatever metric makes the most sense for your data analysis. Each chart entity, such as a pie slice or grid cell or row, acts like a standard Web hyperlink. The developer can choose between taking the user from one report to the next in the same widget, or displaying a standalone widget as exemplified by the GridWidget in **Figure 7.**

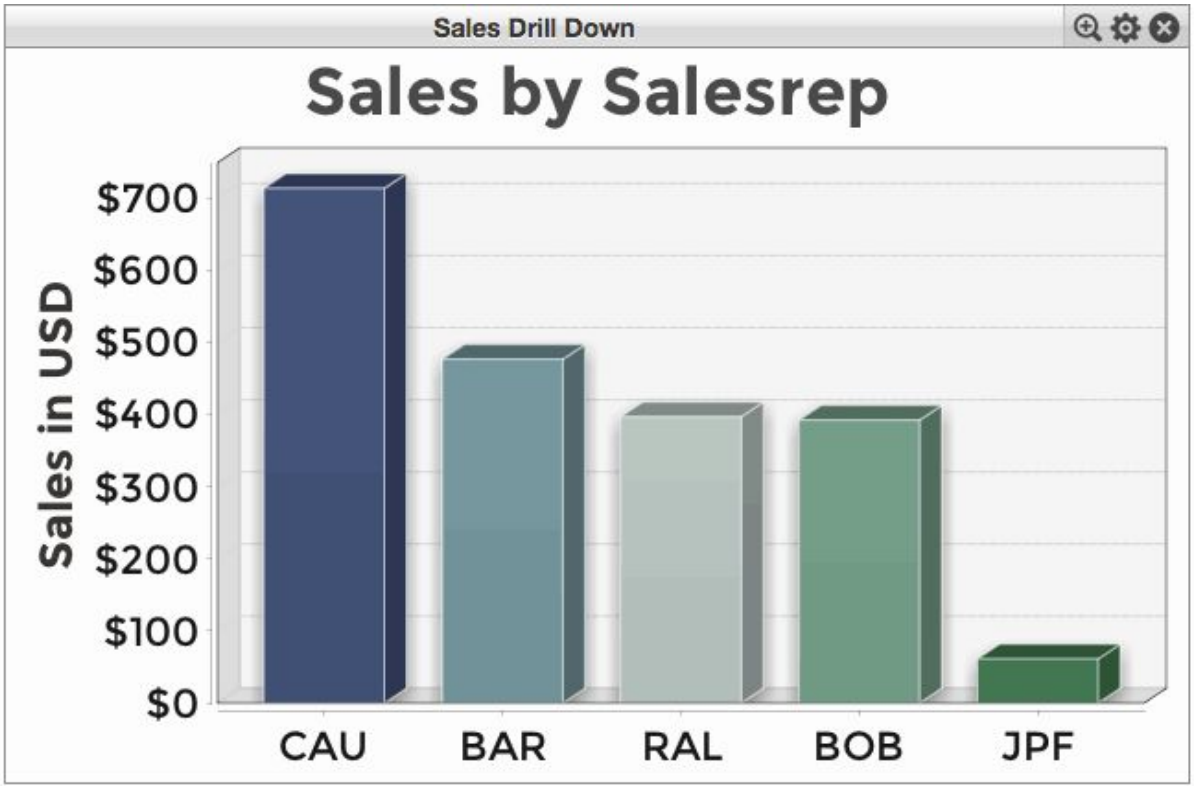

Figure 7. Clicking in a widget to drill down deeper into the data

It is also possible to provide a multi-level drill-down widget, so clicking on the sales figures first breaks it down by region, and the next click breaks those figures down by store. One more click drills down further and reports on store sales by customer. Clicking the right mouse button is a convenient way to either back up one level in the stack or restore the widget to its original report.

To watch a video of the new chart tips and drill down capability, click on this link.

Resize Widget Popouts

If you'd like to see a larger version of a dashboard widget or embedded widget, clicking on the widget's popout toolbutton ( ) displays a larger clone of the widget in short order. This feature has been available from the beginning and is a convenient way to focus everyone's attention on a particular widget during a presentation. When defining the widget, the developer may choose to override the default popout size and call the DashboardWidget's `k]l0ghgml3ar]ÝÞ` method. While this allows the developer to define a large, tall popout for a JasperViewerWidget, ultimately the end user knows the best size for the popout given their computer's screen size. That is why BASIS recently upgraded the popout widgets, displaying them in a user-resizable window. Users can now click and drag on a popout window's corner and resize the widget as desired, further extending the built-in behavior for all widgets.

Configure and Secure the JDBC Connection

When creating a widget, the developer can populate the widget's grid or chart automatically by providing a BBjRecordSet or a database connection string and SQL query to the widget's creation method. This kind of $YlY&addYZd]7a\ ]l now supports a new creation method that also takes a JDBC connection mode string. The mode string is merely a list of properties and values separated by commas that widgets pass to the database or JDBC connection at connect time. For example, a connection mode string may look like:

```
53%2đeqmk]jfYe]Ä07$đeqhYkkÄ!54/#/--)4đ.
```

This allows the developer to provide specific key/value pairs that not only provide authentication, but also configure the connection and specify optional parameters such as timeout or auto commit settings. Therefore, it is now possible to remove the authentication settings from the database connection string and instead set them in the connection mode string. This not only lends more flexibility when it comes to widget creation and modification, it also means that authentication information is no longer retrievable from the widget. That is because these data fillable widgets provide methods so that the developer can efficiently get and/or set the widget's JDBC connection string and SQL query. While this is usually done in response to a filter selection event or drill down click, it may be beneficial to protect the authentication parameters. If the developer moves the authentication information into the connection mode string it will remain unchanged for the life of the widget and cannot be discovered by asking for the widget's JDBC connection string.

Transform Empty Charts

Due to the dynamic nature of SQL queries, the underlying data, and the presence of filters, it is occasionally possible to end up with an empty widget. Perhaps the filtering criteria were too stringent, or maybe there are not any sales figures available yet when reporting on the first day of the quarter. Regardless, the user can probably figure out that a GridWidget or BarChartWidget doesn't have any data to display as they still see the grid headers and the bar chart's axes and domain/range labels and titles. But with a PieChartWidget you're left with a completely empty widget that may confuse users. After all, if the widget has no data then there won't be any pie slices or legend to display in the widget. To address these situations, all chart widgets now offer two new methods: k]l#`Yjl%ehlq$akhdYq4]plÝÞ and k]l#`Yjl%ehlq$akhdYq)eY]ÝÞ. This allows the developer to overlay custom text onto the rendered chart or completely replace the chart with any desired image. For example, adding the following line of code:

```
widget!.setChartEmptyDisplayText("There is no data to display")
```