

Add New Grid Selections to Your Toolbox

Ideas can be seeds that explode onto the computers of an entire workforce but a lack of time, energy, or funding can starve an idea before it even gets a chance to germinate. In order to carve off time to nurture ideas when resources are tight, you constantly need to be on the lookout for ways to be more productive with the resources you already have. And getting productivity improvements to the tools in your toolbox, like a more fully featured grid control, just might help you find the time to feed your ideas so that they can take root and grow. This article looks more closely at how new grid enhancements will make you more productive in BBJ® 15.0.

BASIS Grid Controls

BBj programmers commonly use various Grid* controls in the role of data entry, data verification, graphical control layouts, and browser interface layouts. In a business application, the importance of organizing your data into rows and columns cannot be overstated. It can be paramount for users to view and access information contained in databases of any size, so much so that the Grid has entire functionality created for that exact purpose. However, as important as it is to access and view the returned results in a Grid, the information has little value unless a user can select data in the Grid.

*References to "Grid" in this article are intentionally not specific but refer to one or all of the BBJGrid controls – [BBjStandardGrid](#), [BBjDataAwareGrid](#), and [BBjDataBoundGrid](#) – because they use the same model for row, column, and cell selection.

Default Selection Model

BASIS made some massive improvements to the selection model used in the Grid to aid you in nurturing your ideas and turning them into reality beginning in BBJ 15.0. In the past, the Grid used Oracle's Default Selection Model (DSM) to work with the JTable, which is the base upon which we built our Grid. For years, the JTable with the DSM has met developers' needs, however, *The Times They Are A-Changin'* (Dylan, 1964). Although the DSM was effective at Grid selection, it had its limitations.

For example, Oracle based the DSM on the concept of rows and columns. Clicking in a cell with a mouse effectively split the selected data into two separate values, a row and a column. The selected rows and columns did not have any knowledge of one another and the Grid worked with them separately; however, a user could work with them in conjunction with each other under special circumstances.

In the modern world of grid manipulation, selecting by row and column causes problems for our user base. The most commonly reported problem is the inability to select and deselect multiple individual cells within the Grid. The DSM was only able to select a single cell because it recorded that cell as the intersection of the currently selected row with the currently selected column, deducing which cell the user actually selected.



By Aaron Wantuck
Software Engineer

The blue lines in **Figure 1** illustrate using a currently selected row and column to identify an individual selected cell, represented by a blue square. The selected cell is identified as (2,3) because cell references follow the format (row, col), where row and col are zero-based indexes, ignoring any headers.

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 1. Selecting cell (2, 3) with zero-based indexes in the DSM

While selecting by row and column works well enough for a single cell or for multiple rows or columns, it does not work well for selecting multiple individual cells. For example, if you select cells (0, 5) and (2, 4), the DSM records that rows 0 and 2 are selected and columns 4 and 5 are selected. This produces a result that looks like **Figure 2**; you probably expected something like **Figure 3** instead.

Another common problem was that, at times, BUI selection would differ from GUI selection. This was a result of using the DSM for GUI applications while having completely different code duplicate the functionality of the DSM for BUI. It was difficult to maintain these two different code sets and still get the same selections under all conditions.

Enhanced Selection Model

To address these issues and provide an improved way to select items within the Grid, BASIS dropped both the DSM and the DSM-like BUI code, and combined them into a single shared code model, the “Enhanced Selection Model.” This means that multiple cell selection is now possible, and that future Grid code changes only need to be made once for both GUI and BUI.

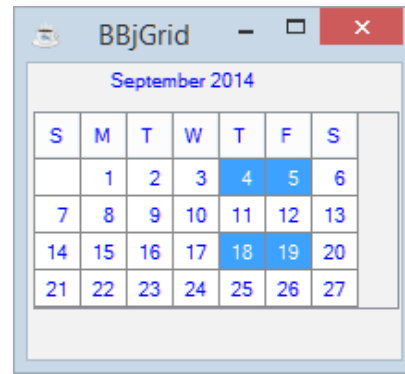
But what if you don’t want to change to the new model, what if the Grid selection you have used for years is “good enough”? You should not even notice the change. BASIS tested extensively to ensure that the default selection behavior in the Grid remains unchanged. We call this the legacy behavior, detailed further in the “Legacy Selection Model” section below. The Grid now offers you the new enhanced behavior, but only once you make a programmatic call to enable it by calling `setEnhancedSelectionModeEnabled(boolean)`. You can also call `isEnhancedSelectionModeEnabled()` at any time to find out whether your Grid is using the Enhanced Selection Model.

Improved Cell Selection

The Enhanced Selection Model does not use rows and columns, but records the individual cells selected either programmatically or through an input device such as the mouse. So what does this mean for Grid control users? Since each cell is recorded independently, you can now select multiple individual cells in a Grid. And you can use the standard [Ctrl]+click ([Command]+click on a Mac – hereafter just referenced as [Ctrl]) and [Shift]+click actions – to select or deselect one or more cells. For example, if you select cell (0, 5) and then hold down the [Ctrl] key while clicking cell (2, 4), you will now see the results in **Figure 3**.

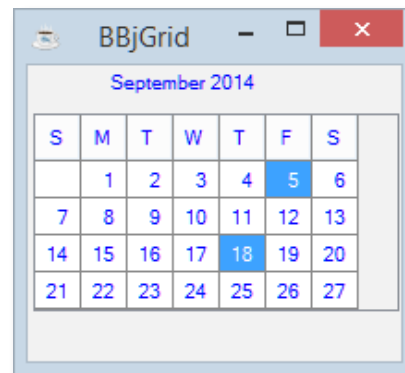
Also, you can now select a range of cells by holding down the [Shift] key while clicking a second cell. If after that you [Ctrl]+click individual selected cells, those cells will be deselected, resulting in never-before achievable selections such as those shown in **Figure 4**.

You do need to be aware of a few changes when using the new Enhanced Selection Model. The first is that the model brings with it the concept of the currently selected cell. Since you can now select multiple individual cells, which one should be returned when you call `getSelectedCell()`? BASIS now defines a *currently selected cell* as “the last cell selected, but only if it is still selected.” But what happens if you deselect the currently selected cell?



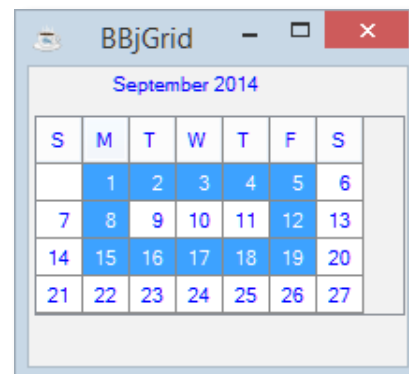
S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 2. The intersection of rows 0 and 2 with columns 4 and 5



S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 3. Selected cells (0, 5) and (2, 4)



S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 4. An example of individual cell selection and deselection

Let's look at an example. If a user selects cell (1, 5), and then [Ctrl]+clicks cells (0, 1) and (3, 3) in that order, the currently selected cell would be (3, 3); `getSelectedCell()` would return (3, 3), and `getSelectedCells()` would return [(0, 1), (1, 5), (3, 3)]. The user will see the selections shown in **Figure 5**.

If the user then deselects cell (3, 3) with a [Ctrl]+click, then there is no longer a currently selected cell. Method `getSelectedCell()` would return (-1, -1) because the user removed the currently selected cell from the set of selected cells, but `getSelectedCells()` would still return [(0, 1), (1, 5), (3, 3)], and the user would see the selection shown in **Figure 6**.

Selection for Rows and Columns

The second change is that the Enhanced Selection Model no longer supports the `setSelectedRow()`, `setSelectedRows()`, `setSelectedColumn()`, and `setSelectedColumns()` methods. After all, you cannot select specific rows if there is no concept of a selected row; instead, you select cells. BASIS designed several new enhanced methods to allow you to set a selection; `setSelectedCells(Vector<intPair>)` and `getSelectedCells()`. In BBj 15.0, these methods will allow you to select any combination of individual cells programmatically. The `setSelectedCells()` method takes a collection of integer pairs, each of which holds a zero-based row and column index representing an individual cell.

The `getSelectedRow()`, `getSelectedRows()`, `getSelectedColumn()`, and `getSelectedColumns()` methods are still available in the Enhanced Selection Model, but they work slightly differently than they did under the DSM. Now they return rows or columns that contain at least one cell that is selected.

To allow users to highlight an entire row, use the existing `setShouldHighlightSelectedRow()` method. So perhaps you are wondering, "If I use `setShouldHighlightSelectedRow()` to highlight full rows, how do I highlight full columns? A `setShouldHighlightSelectedColumns()` method doesn't exist." Well, I am glad you asked. The Grid now offers a `setShouldHighlightSelectedColumn()` method that works with columns the same way as `setShouldHighlightSelectedRow()` works with rows. This new method is present in both the enhanced and legacy selection models, giving you full control over the Grid to highlight full rows and columns.

Legacy Selection Model

What happens if I already use methods that are unsupported in the Enhanced Selection Model? Well, the Legacy Selection Model mimics the behavior of the old Oracle DSM, and stores selection values in a row and column format just as it always has. You can use this legacy model to avoid changing your program, if you don't need to select individual cells, or if you simply prefer row and column selection. The Legacy Selection Model is still available for you and is in fact the current Grid default, so no change is required.

Summary

If you have an application that requires users to select individual cells in a Grid, then you need to use the new Enhanced Selection Model. The BASIS Grid allows you to have more control over how your users make their selections. It makes maintaining the code easier, and eliminates some of the shortfalls in the legacy Default Selection Model ... and it is optional. The Enhanced Selection Model is another example of BASIS providing you with better tools for developing your applications, while increasing your productivity, helping you turn your ideas into reality. ■

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 5. Multiple cell selection with (3, 3) as the currently selected cell

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

Figure 6. Multiple cell selection after [Ctrl]+click to deselect cell (3,3)