Using CellContainers with Dataview

About:

DataView includes a special functionality through the included CellContainer class. This class allows you to create custom cell layouts using ContainerControls.

The CellContainer Class was originally intended as a display only feature, but was updated to allow interacting with the container by clicking a cell. When the user clicks in a cell with a CellContainer, the container is populated and moved into the position of the cell, allowing the user to interact with controls in the Container. The container can also be activated without user interaction using DataView.ActivateCell(row, column)

General Use:

To use a Container in a cell, first add a containerControl to your project and set the super to piDogDataView.CellContainer. Next, layout the container with the desired controls and appearance.

The container can be added to a cell or column by assigning a value to cellContainer(row, column) or DataView.Column(i).CellContainer. Using the Column version will use the container for each cell in that column unless overriden by CellContainer.

Important: When setting a CellContainer, DataView uses only the class, and not the instance!

For example:

```
me.cellContainer(0,1)=new myCellContainer
me.cellContainer(1,1)=new myCellContainer
```

is the same as

```
dim cc as new myCellContainer
me.cellContainer(0,1)=cc
me.cellContainer(1,1)=cc
```

You should only access values in the container during the PopulateValues event and DepopulateValues event or from, for example, an event of a control in the container like valueChanged or textChanged. The Open and Close events only fire during initialization of the container after it is added to a DataView, and not for the individual cells.

The PopulateValues event will fire when the container is being readied for use either to render the cell, or for user interaction. When Populating the values in the Container, you can check the destination cell in the DataView using me.Row and me.Column. me.view refers to the DataView that owns the CellContainer. Using these values, you can access values stored in DataView.

For example:

```
me.label1.text=me.view.cell(me.row,me.column)
```

The Depopulate event is called when the cell has been active and is becoming inactive. This is a good time to save any values.

Example:

```
me.view.cell(me.row,me.column)=me.textarea1.text
```

You can set WantsEvents to false for a display-only Container that will not allow user interaction.

You can check me.isActiveCell to determine if the cell is being rendered or activated for user interaction.

The PrintCell event allows you to do custom drawing for printing that is different from the UI.

The SetPrintHeight event allows you to modify the desired height based on a given column width.