I've been working with grids a lot lately, and find setting them up very tedious: set ColumnCount to the desired value, set ControlSource and other properties such as InputMask and Alignment for every column, set Caption and other properties for every header, etc. To make matters worse, if you change the RecordSource for the grid, ControlSource gets reset for every column so now you have to go back and fix them. To matters even worse, if you need to add a new column between two existing ones, you have to move all the property settings for the subsequent columns. (Yes, I know you can just increment ColumnCount and set ColumnOrder for the new column to the proper value, but then the visual appearance of the grid doesn't match the column order in the Properties window. Do that a couple of times and then have fun trying to find a specific column.)

So, I decided to format grids programmatically. At first, I wrote a lot of code like this:

```
with This.grdOrders
   .ColumnCount  = 22
   .RecordSource = 'orders'

   .Column1.ControlSource   = 'orders.line'
   .Column1.Width           = 30
   .Column1.Header1.Caption = 'Line'

   .Column2.ControlSource   = 'sorder.item'
   .Column2.Width           = 40
   .Column2.Header1.Caption = 'Item #'

   * more code here for the rest of the columns
endwith
```

That too grew tedious quickly. So, I decided to dynamically format the grid at run time rather than design time. I added a SetupColumns method to my grid base class with code that accepts a format string and applies it to the grid. Then in an instance of the grid, I write code that sets up the format string and calls SetupColumns. Here's an example:

```
text to lcColumns noshow
Field    |Width  |Caption    |Alignment  |InputMask  |ReadOnly |Control
Invnum   |70     |Invoice #  |           |           |.T.      |
Date     |70     |Date       |           |           |.T.      |
Name     |*      |Project    |           |           |.T.      |
Amount   |60     |Amount     |1          |99,999.99  |.T.      |
Hours    |40     |Hours      |1          |999.99     |.T.      |
Paid     |70     |Date Paid  |           |           |         |Checkbox
Received |60     |Received   |1          |99,999.99  |         |
endtext
Thisform.grdInvoices.SetupColumns(lcColumns)
```

Here are some rules for the format string:

- The format settings must be in this order and have a header row like shown above
- Separate format settings with tabs and a pipe
- "Field" is the field name (aliased or not)

- Use a tab for an unspecified value, such as Alignment and InputMask for most of the entries in the example
- Use "*" for Width to "auto-size" a column; that is, size it to the rest of the space after the other columns are sized
- Control specifies what control to use for the column. Currently only Checkbox is supported but others could easily be added

Now setting up a grid is just a matter of filling out an easy-to-understand "table" of column settings. Need to add a new column in the middle of the grid? Just add a new line to the table.

However, it's still slightly tedious because you have to run the form the grid is on to see what it looks like, close the form, edit the table (adjust column widths and alignment, for example), run the form again, rinse and repeat. So, I created a builder named SFGridBuilder that does the same thing at design time. Select the format string without the "text" and "endtext" statements, copy it, invoke the builder, and ta da, the grid is formatted at design time. Tweak the string, copy it, run the builder again ... repeat until perfect.

How do you invoke the builder? Any way you normally would:

- I added a Builder property to my grid base class and set it to SFGridBuilder.prg
- You can register it with Thor and invoke it through a Thor hot key or menu
- You can register it with Builder.app
- You can run it directly: do (path + 'SFGridBuilder')

Rick Schummer did a session at Virtual Fox Fest 2020 about builders, so watch the video for details.