Styles in ExcelWriter

ExcelWriter creates three style types:

- GlobalStyle
- NamedStyle
- CellStyle

All three style types derive from the Style class, and NamedStyle derives from GlobalStyle.

A style can be *set* or *applied* to cells, rows, columns, ranges, and areas.
When a style is *set*, the object to which the style is assigned acquires all of that style's properties, including font properties and number formatting. When a style is *applied*, only the differences between the new style and existing style properties (assigned through the ExcelWriter API or in Microsoft Excel) will take effect.

For example, if the cell has a background color and the new style applied does not contain a background color, the cell's color will not be affected. Note: Properties that were set on a pre-existing style in a workbook will not be propogated to the object to which a new style is *applied*.

> ⚠ Setting a style overwrites any pre-existing formatting on the cell, area, or range. Applying a style merges the new style with the previous one.

To *set* a style, use the Style property or the SetStyle method. To *apply* a style, call ApplyStyle. Both are accessible through the following objects: Cell, Area, Range, RowProperties, and ColumnProperties.

GlobalStyle

A GlobalStyle is global to a workbook. It can be *set* or *applied* to cells, rows, columns, areas, and ranges. When a GlobalStyle is *set* (using the Style property), subsequent changes to the style affect all cells on which the style was set. When a GlobalStyle is *applied* (using the ApplyStyle method), subsequent changes to the style will not affect cells to which the style was applied.

Unlike NamedStyles, GlobalStyles are not stored in a collection, so they may be created, but not retrieved. A GlobalStyle is not accessible after the workbook is saved.

To create a GlobalStyle, call Workbook.CreateStyle.

NamedStyle

The ExcelApplication object contains a collection of NamedStyles, which - unlike GlobalStyles are accessible after the workbook is saved. This collection includes Excel's built-in styles and any user-defined styles in a workbook opened with the ExcelApplication.Open method.

A NamedStyle is global to a workbook. It can be *set* or *applied* to cells, rows, columns, areas, and ranges. When a NamedStyle is *set* (using the Style property), subsequent changes to the style affect all cells on which the style was set. When a NamedStyle is retrieved from an existing workbook and *applied* (using the ApplyStyle method), subsequent changes to the style will not affect cells to which the style was applied.

To create a NamedStyle, call Workbook.CreateNamedStyle.

To return an existing NamedStyle, call Workbook.GetNamedStyle.

CellStyle

Every cell in an Excel workbook contains a unique cell style. This style is exposed by ExcelWriter through the CellStyle class. Changes to a CellStyle affect only the cell that owns the style. A CellStyle may be *set* on another cell or group of cells, but this action clones the style. Changes to the original CellStyle reference affect only the owning cell and changes to the cloned CellStyle affect only the cell on which the style was set. If the style is set on a cell grouping object - such as an area, range, row, or column - each cell in the grouping receives a unique clone of the style.

CellStyles are accessed through Cell.Style.

Example

```
ExcelApplication xla = new ExcelApplication();
Workbook wb = xla.Create(ExcelApplication.FileFormat.Xlsx);
Worksheet ws = wb.Worksheets[0];

//--- Write some random data into the cells to be stylized
System.Random rand = new System.Random();
for(int iRow = 0; iRow < 24; iRow++)
        for(int iCol = 0; iCol < 3; iCol++)
                ws.Cells[iRow, iCol].Value = rand.Next(100);

ws.Cells[24, 0].Formula = "=SUM(A1:A24)";
ws.Cells[24, 1].Formula = "=SUM(B1:B24)";
ws.Cells[24, 2].Formula = "=SUM(C1:C24)";

//--- This GlobalStyle, styleMoneyFormat, will be used to apply currency
//--- stylization to workbook values.
Style styleMoneyFormat = wb.CreateStyle();
styleMoneyFormat.NumberFormat = "$#,##.00";

//--- styleBold represents a bold-faced font.
Style styleBold = wb.CreateStyle();
styleBold.Font.Bold = true;

//--- Set Style on an Area of cells
//--- The SetStyle method sets a base style
//--- and overwrites all existing style attributes
Area areaData = ws.CreateArea("A1:C25");
areaData.SetStyle(styleMoneyFormat);

//--- ApplyStyle overlays a style with an existing
//--- style, changing only the attributes that have
//--- been defined in the applied style
Area areaTotalRow = ws.CreateArea("A25:C25");
areaTotalRow.ApplyStyle(styleBold);

xla.Save(wb, "Styles.xlsx");
```

Code Example: ExcelApplication Basic Steps

This sample shows many of the core features of ExcelApplication, including the use of Styles.

[C#] | [VB.NET]