Basic Steps R1C1

Intro

Same as Basic Steps except all the formulas and references are in R1C1 notation.

Here are some of the features shown in the sample:

1. Setting cell values and formulas
2. Adding a basic chart
3. Using the color palette
4. Applying styles and formatting
5. Page setup, including headers and footers
6. Setting "document properties" metadata
7. Adding external pictures to a worksheet
8. Defining clickable hyperlinks
9. Using cell comments
10. Setting Conditional Formatting
11. Defining workbook and worksheet protection

Code

```
class Basic_Steps_R1C1
    {
        ///This class creates a workbook with only one Worksheet.
            Declare ExcelApplication, Workbook, Worksheet, and Color
            objects    at the class-level.
        private ExcelApplication xlw;
        private Workbook wb;
        private Worksheet ws;
        private Color clrDarkBlue, clrDarkGreen, clrComment, clrTitleCell;

        ///
        /// Build the report with ExcelApplication
        ///
        public void GenerateReport()
        {

            ///Create an instance of ExcelApplication,
            a new Workbook, and create one Worksheet.
            Switch the workbook to RC notation.
            xlw = new ExcelApplication();
            wb = xlw.Create(ExcelApplication.FileFormat.Xlsx);

            ///Set this property to turn or off the R1C1 formula
             //parser.  When set to true, all properties and methods
             //that accept or return formula string will use R1C1 notation.

            wb.UseRCFormulaNotation = true;

            ws = wb.Worksheets[0];

            ///Demonstrate setting cell values
            this.PopulateWorksheet();

            ///Demonstrate setting formulas
            this.AddCellFormulas();
```

```
        ///Set up colors that will be used in
        //the workbook

        this.SetupColors();

        ///Add styles to the worksheet
        this.AddStyles();

        ///Set the Page Setup properties
        this.DoPageSetup();

        ///Set Document Properties
        this.AddDocumentProperties();

        ///Add a chart to the worksheet
        this.AddChart();

        ///Add some comments to the worksheet
        this.AddCellComment();

        ///Add a JPEG picture to the worksheet
        this.AddPicture();

        ///Add a hyperlink
        this.AddHyperlink();

        ///Use the CharacterRun object to
        //stylize individual characters
        //in a cell.

        this.StylizeHeaderCharacters();

        ///Add Conditional Formatting rules to the Worksheet
        this.AddConditionalFormating();

        ///Turn off gridlines display
        ws.ShowGridlines = false;

        ///Save the report by streaming it
         //to the client's browser
        xlw.Save(wb, @"..\..\ExcelOutputFiles\BasicStepsR1C1_output.xlsx");
    }

    ///  The Cell's getCharacters method returns a
    /// CharacterRun object that can be used to apply
    /// formatting to specific characters in the cell.
    /// Cell A1 is populated with "Welcome to SoftArtisans
    /// OfficeWriter" in the populateWorksheet() method.
    /// This stylizeHeaderCharacters() method formats
    /// the individual words in the cell.
    ///
    private void StylizeHeaderCharacters()
    {
        ///Get a reference to cell A1
        Cell cellHeader = ws[0, 0];

        ///Stylize the first word, starting from the
        //first character, ending at the 10th
```

```
            cellHeader.GetCharacters(0, 10).Font.Size = 12;

            ///Format the font for the next two words in the cell
            cellHeader.GetCharacters(11, 12).Font.Color =
wb.Palette.GetClosestColor(25, 116, 210);
            cellHeader.GetCharacters(24, 12).Font.Color =
wb.Palette.GetClosestColor(77, 82, 81);
        }

        ///  Add protection to workbooks and worksheets
        /// with various security-related properties.
        ///
        private void SetProtection()
        {
            ///The Workbook.setPasswordToModify property
            //will cause the user to be prompted
            //for a password when the workbook opens
            //in Excel. Users without the password
            //will get Read-Only access.
            //

            wb.PasswordToModify = "SomePassword";

            ///Worksheet.protect turns on Worksheet Protection.
            //This can be used to "lock" cells to prevent them
            //from being edited.

            ws.Protect("");

            ///setReadOnlyRecommended advises the user
            //that the workbook should be opened as Read-Only
            //when it's opened in Excel.  This is advisory only
            //and can be ignored by the user even if they don't have
            //a password.

            wb.ReadOnlyRecommended = true;
        }

        ///  Create hyperlinks in cells with the
        /// Cell.CreateHyperlink method.
        /// It returns a Hyperlink object
        ///
        private void AddHyperlink()
        {
            ///Write bold-faced text into a cell to describe the hyperlink
            ws[30, 0].Value = "Click here to visit SoftArtisans Support:";
            ws[30, 0].Style.Font.Bold = true;

            ///Create the hyperlink
            Hyperlink hl = ws[31,
0].CreateHyperlink("http://www.officewriter.com/support");

            ///Merge the cells containing the hyperlink together.
            //This is optional, but improves the way it looks.

            ws.CreateArea(31, 0, 1, 4).MergeCells();
        }

        /// <summary>Add some conditional formatting to the data in the worksheet.
```

```
        /// Use the wb.CreateConditionalFormat() method to create a conditional
formatting object
        /// that can be applied to a Cell, Area, or Range.
        /// Use the Cell, Area or Range SetConditionalFormat() method to apply a
conditinal format.
        /// </summary>
        private void AddConditionalFormating()
        {
            //Create a conditional formatting object
            ConditionalFormat cf1 = wb.CreateConditionalFormat();

            //Create a range of the value columns
            Area valuesArea = ws.CreateArea("R4C1:R24C3");

            //Create a set of conditions, and set the fonts to warmer colors as the
values go up
            Condition condition1 =
cf1.CreateCondition(Condition.Comparison.CellValueBetween, "=0", "=10");
            condition1.Style.Font.Color = Color.SystemColor.Red;

            Condition condition2 =
cf1.CreateCondition(Condition.Comparison.CellValueBetween, "=11", "=20");
            condition2.Style.Font.Color = wb.Palette.GetClosestColor(0, 25, 149);

            Condition condition3 =
cf1.CreateCondition(Condition.Comparison.CellValueGreaterThan, "=20");
            condition3.Style.Font.Color = wb.Palette.GetClosestColor(36, 122, 51);

            //Apply the conditional formatting object to the range of values
            valuesArea.SetConditionalFormat(cf1);
        }

        ///  Insert external picture files into worksheets
        /// with the Pictures.CreatePicture method.
        /// Specify its location with an Anchor object.
        ///
        private void AddPicture()
        {

            ///The path to the picture that will be imported
            string picturePath = @"..\..\ExcelImages\OWlogo.gif";

            ///Create an Anchor on which the picture will set
            Anchor anch = ws.CreateAnchor(0, 7, 0, 0);

            ///Insert the picture into the workeheet.
            //Its top left corner will be on the Anchor.

            Picture pic = ws.Pictures.CreatePicture(picturePath, anch);
        }

        ///  Comment items can be added,
        /// edited, or removed from cells.
        ///
        private void AddCellComment()
        {
            ///Add a comment to cell C3
            Cell c = ws[2, 2];
            Comment cmnt = c.Comment;
```

```
            ///Set the comment text (body of comment note),
            //author (shown in status bar),
            //and visibility (false by default).

            cmnt.Author = "John Doe";
            cmnt.Text = "Figures for March are projected.";
            cmnt.Visible = true;

            ///The Comment's Shape object
            //can be used to stylize the
            //comment tag itself.
            //Set the fill color and FitToText
            //properties.

            Shape shp = cmnt.Shape;

            shp.SetCustomFillColor(0, 204, 255);
            shp.FitToText = true;
        }

        ///  The DocumentProperties object is used for setting
        /// workbook metadata such as Author, Comments, Company,
        /// and also user-defined custom properties.
        ///
        private void AddDocumentProperties()
        {
            ///Get the DocumentProperties interface from Workbook
            DocumentProperties docprops = wb.DocumentProperties;

            ///Set built-in DocumentProperties values
            docprops.Author = "John Doe";
            docprops.Comments = "A basic demonstration of OfficeWriter for Excel";
            docprops.Company = "SoftArtisans, Inc.";
            docprops.Title = "Basic Report w/ Chart";

            ///Set custom DocumentProperties key/value pairs
            docprops.SetCustomProperty("GeneratedBy", "SoftArtisans OfficeWriter for
Excel");
        }

        ///  Adds a simple clustered column chart to a worksheet
        /// Category data is what appears on the X axis of the chart
        /// Series items (there can be more than one) are the data that
        /// the chart is displaying in the plot area.
        ///
        private void AddChart()
        {
            ///Create an Anchor on which the Chart will be placed
            Anchor anch = ws.CreateAnchor(7, 4, 0, 50);

            ///Create a Column.Clustered chart on the Worksheet
            Chart columnChart = ws.Charts.CreateChart(ChartType.Column.Clustered,
anch);

            ///Use the chart's SeriesCollection to add Series items and set dategory
data
            SeriesCollection seriesCol = columnChart.SeriesCollection;
```

```
            ///Set the month header row as the Primary axis's category data
            seriesCol.CategoryData = "Sheet1!R3C1:R3C3";

            ///Add the totals row as a series and name it "Sales"
            Series chartSeries = seriesCol.CreateSeries("Sheet1!R25C1:R25C3");
            chartSeries.Name = "Sales";

            ///Change the chart's title
            columnChart.Title.Text = "Basic Column Chart";
        }

        ///  Every Worksheet has a PageSetup object that can be
        /// used to set print behavior.  Commonly used PageSetup
        /// capabilities are header/footers, and printer paper setup.
        ///
        private void DoPageSetup()
        {
            ///Get the PageSetup object from the Worksheet
            PageSetup ps = ws.PageSetup;

            ///Set the worksheet headers and footers.
            Note use of special header/footer variables.
            // Updated to new syntax
            HeaderFooterSection leftHeader =
ps.GetHeader(HeaderFooterSection.Section.Left);
            leftHeader.SetContent("Document name: &F");
            HeaderFooterSection centerHeader =
ps.GetHeader(HeaderFooterSection.Section.Center);
            centerHeader.SetContent("Basic Report");
            HeaderFooterSection rightHeader =
ps.GetHeader(HeaderFooterSection.Section.Right);
            rightHeader.SetContent("Generated by OfficeWriter");
            HeaderFooterSection leftFooter =
ps.GetFooter(HeaderFooterSection.Section.Left);
            leftFooter.SetContent("Created on date &D");
            HeaderFooterSection centerFooter =
ps.GetFooter(HeaderFooterSection.Section.Center);
            centerFooter.SetContent("Page &P of &N");
            HeaderFooterSection rightFooter =
ps.GetFooter(HeaderFooterSection.Section.Right);
            rightFooter.SetContent("Create at time &T");

            ///Turn off the printing of gridlines
            ps.PrintGridlines = false;

            ///Change printing orientation to Landscape,
            and. paper size to Legal
            ps.Orientation = PageSetup.PageOrientation.Landscape;
            ps.PaperSize = PageSetup.PagePaperSize.Legal;
        }

        ///  Apply some basic stylization to the data in the worksheet.
        /// Use the setStyle method to set a base style for a Cell, Area, or Range.
        /// Use the ApplyStyle method to "merge" or "layer" two Styles in one region.
        ///
        private void AddStyles()
        {
            ///Create a GlobalStyle to stylize the data region.
            //Set number formatting, and a surrounding border.
```

```
    GlobalStyle styleData = wb.CreateStyle();
    styleData.NumberFormat = "$#.##00";

    ///Style for the header row.
    //Add a surrounding border, and set
    //bold-faced font.

    GlobalStyle styleHeader = wb.CreateStyle();
    styleHeader.Font.Bold = true;

    ///Style for the title cell.  Enlarge the font
    //and change the color.

    GlobalStyle styleTitle = wb.CreateStyle();
    styleTitle.Font.Size = 14;
    styleTitle.Font.Color = clrTitleCell;

    ///The total row will have number formatting,
    //bold, and italicized text.

    GlobalStyle styleTotalRow = wb.CreateStyle();
    styleTotalRow.NumberFormat = "$#.##00";
    styleTotalRow.Font.Italic = true;
    styleTotalRow.Font.Bold = true;

    ///Create an Area that encompases the data
    and apply money formatting to the cells
    Area areaData = ws.CreateArea(3, 0, 21, 3);
    areaData.SetStyle(styleData);

    ///Create an area that encompasses the totals row
    Use the setStyle method to set the base style
    Use the ApplyStyle to layer other styles on
    the base style.
    Area areaTotalRow = ws.CreateArea(24, 0, 1, 3);
    areaTotalRow.SetStyle(styleTotalRow);

    ///Stylize the title cell
    ws[0, 0].Style = styleTitle;

    ///Make the month header row bold
    ws.CreateArea(2, 0, 1, 3).SetStyle(styleHeader);
}

///  Add some data to the workbook by using the Cell.setValue() method.
/// Cells can be addressed by Excel address (eg, A1, B2), or by row/column
/// ordinal for iteration.
/// Cell.setValue() is just one way to add values to cells.
///
private void PopulateWorksheet()
{
    ///Cells can be referenced by Excel name (eg, A1, B2)
    or by 0-based row number/column number

    ///Refers to cell A1
    Could also write as ws["A1").setValue(...)
    ws.Cells[0, 0].Value = "Welcome to SoftArtisans OfficeWriter";
```

```
    ws[2, 0].Value = "Jan";
    ws[2, 1].Value = "Feb";
    ws[2, 2].Value = "Mar";

    ///Use row/column indices to easily iterate over cells
    for (int iRow = 3; iRow <= 23; iRow++)
        for (int iCol = 0; iCol <= 2; iCol++)
            ws[iRow, iCol].Value = iRow + iCol;
}

///  The Cell.setFormula() method is for adding Excel formulas
/// to cells.  The result of the formulas will be calculated when
/// the workbook is opened in Excel.
///
private void AddCellFormulas()
{
    ///Apply forumlas to cells with the Cell.setFormula() method using
    a relative formula created with RC notation.
    string formula = "=SUM(R4C:R[-1]C)";

    ///Display the formula in cell A25
    ws[24, 0].Formula = formula;

    ///Display that formula in cell B25
    ws[24, 1].Formula = formula;

    ///Create a Range to sum using RC notation.
    Range rngMarch = ws.CreateNamedRange("R4C3:R24C3", "MarchRange");

    ///Display a formula that sums the range in cell C25.
    ws[24, 2].Formula = "=SUM(MarchRange)";
}

///  Colors are managed by the Palette object.
/// Palette has several methods for retrieving colors
/// from the workbook's internal palette.
/// Also, there are build-in colors.
///
private void SetupColors()
{
    ///Note: clrDarkBlue, clrDarkGreen, and clrComment
    //are declared above as private class members.


    ///Workbook.getPalette returns the Palette object
    Palette pal = wb.Palette;

    ///GetClosestColor finds the color in the palette
    //that most closely matches the specified Red,
    //Green, and Blue values.

    clrDarkBlue = pal.GetClosestColor(0, 35, 149);
    clrTitleCell = pal.GetClosestColor(0, 35, 149);

    try
    {
        ///getColor will find the exact color you request.
        //If the color does not exist in the palette
        //it will throw an Exception.
```

```
            clrDarkGreen = pal.GetColor(51, 153, 102);
        }
        catch
        {
            // The color you requested was not found in the palette
        }

        ///getColorAt gets the color from the palette at
        //the specified index.  This retrieves the second
        //color from the palette.

        clrComment = pal.GetColorAt(1);

        ///setColor and setColorAt are for changing
        //the colors in the palette. Specify a Color to
        //change for setColor, the index of the color for
        //setColorAt, and then the RGB values for the desired
        //appearance.



        pal.SetColorAt(0, 0, 255, 0);

        ///The Color object has built-in colors
        Color clrSysGreen = Color.SystemColor.Green;
        Color clrSysBlack = Color.SystemColor.Black; ;
    }

    ///  This method is here for your convenience.  You might find it useful
    /// in your project.  It simplifies building formulas in this basic form only:
    /// =[formulaName]([lBound]:[uBound])
    /// For example, =SUM(A1:C3) or =AVERAGE(D2:C20)
    ///
    private string BuildFormula(string formulaName, Cell lBound, Cell uBound)
    {
        System.Text.StringBuilder sb = new System.Text.StringBuilder();
        sb.Append("=");
        sb.Append(formulaName);
        sb.Append("(");
        sb.Append(lBound.Name);
        sb.Append(":");
        sb.Append(uBound.Name);
        sb.Append(")");
        return sb.ToString();
```

```
        }
    }
```

Downloads

Output: BasicStepsR1C1_output.xlsx

Image: OWlogo.gif