Grouping and Subtotals

Intro

Add Excel's grouping and outlining functionality to a workbook with ExcelWriter to display nested data.

The sales data is based on the 2008 sales data from the sample AdventureWorks data base. The sales data is written into the worksheet and grouped by major product category. Subtotal lines are written after each major product category. Each group of product sales line items is also grouped into a collapsible outline.

If you need to display nested, subtotaled data in your Excel report, this demo will be particularly useful to you.

This sample stores some of the data in a CSV file, which is available for download under **Downloads**. The CSV parser used in the example code was developed by Andrew Rissing and can be downloaded from Code Project.

Code

```
 public class GroupingSubtotals
{
    private ExcelApplication xlw;
    private Workbook wb;
    private Worksheet ws;
    private int year = 2008;
    private bool bCollapsed = true;

    /// <summary>
    /// Build the report with ExcelApplication
    /// </summary>
    public void GenerateReport()
    {
        // Creat a blank workbook and a single worksheet
        xlw = new ExcelApplication();
        wb = xlw.Create(ExcelApplication.FileFormat.Xlsx);
        ws = wb.Worksheets[0];

        // Populate
        this.PopulateWorksheet(year);

        //TODO Not implemented
        //ws.createRangeOfColumns

        // Save the report
        xlw.Save(wb, @"..\..\ExcelOutputFiles\GroupingSubtotals_output.xlsx");
    }

    /// <summary>
    /// Write values and import data into the  Worksheet.
    /// Also, sets up outlining and formulas.
    /// </summary>
    /// <param name="year">Display orders from this year</param>
    /// <param name="bCollapsed">Show outlines as collapsed?</param>
    /// <param name="bAddChart">Add chart to worksheet?</param>
    private void PopulateWorksheet(int year)
    {
        // Create a style for the total rows.
         //Make the font bold-faced, and shade
```

```
             //the cells with light grey.

            GlobalStyle totalStyle = wb.CreateStyle();
            totalStyle.Font.Bold = true;
            totalStyle.BackgroundColor = wb.Palette.GetClosestColor(211, 211, 211);

            // This area will be used to determine how
             //many rows of data were imported from the DataTable

            Area area = null;

            // These row and col variables are counters to
             //keep track of the current row and column

            int row = 0;
            int col = 0;

            // Cell references for the total rows will be
             //added into these vectors for the later use in
             //a chart

            ArrayList categoryList = new ArrayList();
            ArrayList seriesList = new ArrayList();

            string[] categories = { "Bike", "Component", "Clothing", "Accessory" };

            // Loop through the DataTables returned
             //from the getData method

            for (int i = 0; i < 4; i++)
            {
                Cell c = ws[row, col];

                // Write the category name
                c.Value = categories[i];

                // Add the category cell to the category vector
                categoryList.Add(String.Format("{0}!{1}", ws.Name, ws[row,
col].Name));

                ws[row, col + 1].Value = "Total:";

                // Apply totalStyle to the Area,
                 //and create a border around the row.

                Area totalArea = ws.CreateArea(row, col, 1, 3);
                totalArea.SetStyle(totalStyle);
                ws[row, col + 1].Style.HorizontalAlignment = Style.HAlign.Right;
                totalArea.BorderAround.Style = Border.LineStyle.Thin;

                DataImportProperties dProps = wb.CreateDataImportProperties();
                dProps.ConvertStrings = true;
                dProps.Truncate = true;

                // Import the data from the DataTable
                DataTable dt = GetData(i + 1);
                area = ws.ImportData(dt, ws[row + 1, 1], dProps);

                // Compute where the last row of data was imported to
```

```
                int lastRow = row + area.RowCount + 1;

                // Loop through all rows that were just imported
                 //and set the outline level to group the rows.

                for (int iRow = row + 1; iRow < lastRow; iRow++)
                {
                    RowProperties rp = ws.GetRowProperties(iRow);
                    rp.OutlineCollapsed = bCollapsed;
                    rp.OutlineLevel = 1;
                }

                // Add a SUBTOTAL formula to the total row.
                string formula = String.Format("=SUBTOTAL(9,{0}:{1})", ws[row + 1,
2].Name, ws[lastRow, 2].Name);
                ws[row, 2].Formula = formula;

                // Add the total cell reference to the series vector
                seriesList.Add(String.Format("{0}!{1}", ws.Name, ws[row, 2].Name));

                // Increment the row counter to the end of the
                 //imported data area, and skip two.

                row = row + area.RowCount + 2;
            }

            // TODO Not implemented
            //ws.createRange("A1:C1").autoFitWidth();

            // Set column widths
            ws.GetColumnProperties(0).Width = 80;
            ws.GetColumnProperties(1).Width = 195;
            ws.GetColumnProperties(2).Width = 110;

            // Create a style to format the currency column
            GlobalStyle stylCurrency = wb.CreateStyle();

            // Create and apply a typical Accounting format
             //to the currency column.

            NumberFormat nf = wb.NumberFormat;
            String fmtString = nf.CreateAccounting(2, true, null);
            stylCurrency.NumberFormat = fmtString;
            Area ar = ws.CreateArea(0, 2, row - 1, 1);
            ar.ApplyStyle(stylCurrency);

            // Join the cell reference vectors into formula
             //strings for the charts.

            string categoryFormula = String.Join(",",
(string[])categoryList.ToArray(typeof(string)));
            string seriesFormula = String.Join(",",
(string[])seriesList.ToArray(typeof(string)));
            Console.WriteLine(categoryFormula);
            // Create a 3D pie chart on a chartsheet

            Chartsheet wsChart =
                wb.Worksheets.CreateChartsheet(ChartType.Pie.Pie3D, "Chart");
            Chart chrt = wsChart.Chart;
```

```
            chrt.SeriesCollection.CategoryData = categoryFormula;

            Series srsSales = chrt.SeriesCollection.CreateSeries(seriesFormula);
        }

        ///<summary>
        /// Uses a 3rd party generic CSV parser
        /// DataTable of product line items
        /// </summary>
        /// <returns></returns>
        private DataTable GetData(int catId)
        {
            DataTable dt=new DataTable();
            if (catId==1)
                dt = GetCSVData(@"..\..\ExcelData\GroupingBikes.csv");
            if (catId==2)
                dt = GetCSVData(@"..\..\ExcelData\GroupingClothing.csv");
            if (catId==3)
                dt = GetCSVData(@"..\..\ExcelData\GroupingClothing.csv");
            if (catId==4)
                dt = GetCSVData(@"..\..\ExcelData\GroupingAccessories.csv");
            return dt;
        }

         #region Utility Methods
        //Uses CSV reader
        System.Data.DataTable GetCSVData(string csvFileName)
        {
            DataTable dt;
            using (GenericParserAdapter parser = new
GenericParserAdapter(csvFileName))
            {
                parser.ColumnDelimiter = ',';
                parser.FirstRowHasHeader = true;

                dt = parser.GetDataTable();
            }
            return dt;
        }
```

```
        #endregion
        }
```

Downloads

- Data:
    - GroupingComponents.csv
    - GroupingBikes.csv, GroupingClothing.csv
    - GroupingAccessories.csv
- Template: GroupingAndNestingTemplate.xlsx
- Output: GroupingSubtotals_output.xlsx