Multiple Chart Report

Intro

Import data to an existing template with multiple charts. Manipulate existing charts with ExcelApplication.

This demo shows how to open an existing Excel workbook and programmatically edit properties of existing charts with OfficeWriter. Select the major product categories you would like to see in the report. The SeriesCollection of the existing chart will be changed at runtime to reflect the number and types of categories you chose.

Code

```
class MultiChart
    {
        private ExcelApplication xlw;
        private Workbook wb;
        private Worksheet ws;
        private int[] catIdList = { 1, 2, 3, 4 };
        private bool bLegend = true;

        /// <summary>
        /// Build the report with ExcelApplication
        /// </summary>
        public void GenerateReport()
        {
            xlw = new ExcelApplication();

            //Open the template workbook
            string templatePath = @"..\..\ExcelTemplateFiles\MultiChart.xlsx";
            wb = xlw.Open(templatePath);

            ws = wb.Worksheets[0];

            this.PopulatePieCharts();
            this.PopulateMainChart();

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


        /// <summary> Populate the worksheet with data for the main chart.
        /// Reset the series collection objects and the category data
        /// based on how many products were chosen for display.
        /// </summary>
        private void PopulateMainChart()
        {
            //Get Data for Quarterly Sales.
            DataTable dtQuart = new DataTable();
                        for (int i=0;i<5;i++)
                dtQuart.Columns.Add();
            dtQuart.Rows.Add(new string[] { "Name", "Q1", "Q2", "Q3", "Q4" });
            dtQuart.Rows.Add(new string[] { "Bikes", "6099507.645869",
"7022584.175748", "10510509.832846", "11290678.586395" });
            dtQuart.Rows.Add(new string[] {"Components", "459073.391596",
```

```
"1111225.617307", "2524193.294746", "1391022.528312" });
            dtQuart.Rows.Add(new string[] { "Clothing", "105669.610552",
"191410.762794", "388216.721639", "326687.409399" });
            dtQuart.Rows.Add(new string[] { "Accessories", "15623.156936",
"32654.322865", "258987.196838", "282992.908554" });

            //Find the main chart based on its title text, and get
            //a reference to it.

            Chart mainChart = FindChart("Quarterly Sales 2008");

            //Clear all the existing series objects from the collection
            SeriesCollection seriesCol = mainChart.SeriesCollection;
            //while (seriesCol.Count > 0)
            //{
            //    seriesCol.Remove(0);
            //}

            DataImportProperties dProps = wb.CreateDataImportProperties();
            dProps.UseColumnNames = false;
            dProps.ConvertStrings = true;
            ws.ImportData(dtQuart, ws.Cells["B29"], dProps);


            //Re-set the category data.  The size will vary
            //depending on the selected categories.

            seriesCol.CategoryData = ws.CreateArea(29, 1, 4, 1).Dimensions;

            //Add a legend to the chart if desired
            if (bLegend)
            {
                mainChart.Legend.Visible = true;
                mainChart.Legend.Location = Legend.LegendLocation.Top;
            }
            else
            {
                //The legend is hidden in the template workbook.
                //Hide it again in case it's made visible.

                mainChart.Legend.Visible = false;
            }
        }


        /// <summary> Obtain the database data and populate data for the
        /// two pie charts on the right-hand side of the sheet.
        /// </summary>
        private void PopulatePieCharts()
        {
            DataTable dCS = new DataTable();
            for (int i = 0; i < 4; i++)
                dCS.Columns.Add();
            dCS.Rows.Add(new string[] { "Clothing", null, "588594.532331" });
            dCS.Rows.Add(new string[] { "Bikes", null, "22579811.983331" });
            dCS.Rows.Add(new string[] { "Accessories", null, "568844.582411" });
            dCS.Rows.Add(new string[] { "Components", null, "2091511.003980" });

            DataImportProperties dProps = wb.CreateDataImportProperties();
```

```
            dProps.UseColumnNames = false;
            dProps.ConvertStrings = true;
            ws.ImportData(dCS, ws.Cells["H27"], dProps);


            DataTable topSales =  new DataTable();
            for (int i = 0; i < 4; i++)
                topSales.Columns.Add();
            topSales.Rows.Add(new string[]{ "Mitchell, Linda", null, "2126790.5635"
});
            topSales.Rows.Add(new string[]{ "Blythe,Michael", null, "1732868.0076" });
            topSales.Rows.Add(new string[]{ "Pak, Jae", null, "2037152.8042" });
            topSales.Rows.Add(new string[]{ "Varkey Chudukatil, Ranjit", null,
"1549000.1782" });
            topSales.Rows.Add(new string[]{ "Carson, Jillian", null, "1546519.7757"
});
            ws.ImportData( topSales, ws.Cells["H52"], dProps);
        }


         private Chart FindChart(string title)
         {
             Console.WriteLine("Chart Count " + ws.Charts.Count);
             for (int iChart = 0; iChart < ws.Charts.Count; iChart++)
             {
                 Chart chrt = ws.Charts[iChart];
                 if (chrt.Title.Text == title)
                     return chrt;
             }
             return null;
         }

        //loops through datatable and converts values to doubles
        private void toDoubles(DataTable dt)
        {
            for (int i = 0; i < 4; i++)
            {
                for (int j = 1; j < 5; j++)
                {
                    String s = dt.Rows[j].ItemArray[i].ToString();
                    Console.WriteLine(s);
                    Convert.ToDouble(s);

                    Double d1 = Convert.ToDouble(s);
                    dt.Rows[j].ItemArray[i] = d1;
                    Console.WriteLine(d1.GetType());
                }
            }
        }
    }
```

Downloads

- Template MultiChart.xlsx
- Output MultiChartReport_output.xlsx