

Charts in ExcelApplication

ExcelApplication supports all Excel chart types, and allows you to insert a chart in a worksheet, create a chart sheet (a worksheet that only contains a chart).

If you open an existing Excel file with ExcelApplication.Open, you can use ExcelWriter's charting objects to [modify charts in the spreadsheet](#).

To create a simple column chart:

1. Get a Charts collection:

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

The Charts collection contains all charts in a specified worksheet.

2. Create an anchor in the worksheet:

```
Anchor anch = ws.CreateAnchor(7, 4, 0, 50);
```

An anchor represents the position of a floating object (e.g., a chart) within a worksheet. The chart's top left corner will be placed at the anchor.

3. Create a blank chart of a specified type, at the anchor you created:

```
Chart chrt = chrts.CreateChart(ChartType.Column.Clustered, anch);
```

The ChartType class contains all available chart types (e.g., Column) and sub-types (e.g., Clustered).

4. Return a SeriesCollection object representing the set of data series in the chart:

```
SeriesCollection sc = chrt.SeriesCollection;
```

5. Set the range of category (x) axis values:

```
sc.CategoryData = "Sheet1!A3:C3";
```

6. Add a data series to the chart:

```
Series s = sc.CreateSeries("Sheet1!A25:C25");
```

The formula passed to CreateSeries represents cells that contain the source values for the new data series.

Example: Creating a Chart

```

using System;
using SoftArtisans.OfficeWriter.ExcelWriter;

class ChartDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, System.EventArgs e)
    {
        //--- Create ExcelApplication, a Workbook, and a Worksheet
        ExcelApplication xla = new ExcelApplication();
        Workbook wb = xla.Create();
        Worksheet ws = wb.Worksheets[0];

        //--- Put some values into the cells that the
        //--- chart will reference
        System.Random rand = new System.Random();

        //--- Headers
        ws.Cells[0, 0].Value = "Month";
        ws.Cells[0, 1].Value = "Sales";

        //--- Month and sales
        ws.Cells[1, 0].Value = "Jan";
        ws.Cells[1, 1].Value = rand.Next(1000);
        ws.Cells[2, 0].Value = "Feb";
        ws.Cells[2, 1].Value = rand.Next(1000);
        ws.Cells[3, 0].Value = "Mar";
        ws.Cells[3, 1].Value = rand.Next(1000);

        //--- Create an Anchor on Cell B6
        Anchor anch = ws.CreateAnchor(4, 2, 0, 0);

        //--- Create the chart
        Chart chrt = ws.Charts.CreateChart(ChartType.Column.Clustered, anch);

        //--- Set series collection
        string seriesFormula =
            String.Format("={0}!{1}:{2}", ws.Name,
                ws.Cells[1, 1].Name, ws.Cells[3, 1].Name);
        Series srs = chrt.SeriesCollection.CreateSeries(seriesFormula);
        srs.Name = "Sales";

        //--- Set category data
        chrt.SeriesCollection.CategoryData =
            String.Format("{0}!{1}:{2}", ws.Name,
                ws.Cells[1, 0].Name, ws.Cells[3, 0].Name);

        xla.Save(wb, Page.Response, "Charts.xls", false);
    }
}

```

Modifying an Existing Chart

Code sample: Modifying an Existing Chart

When you use ExcelApplication to open a spreadsheet, you can access charts through their names or titles. Chart.getTitle returns a ChartText

object representing the title region. `ChartText.GetText` returns the title text. The following function takes a chart's title and returns a `Chart` object representing the chart. This code can be useful if you want to locate a specific chart in a worksheet but only know its title text:

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

The following example uses the `FindChart` code to locate a specific chart in a worksheet. The chart series collection is then cleared and re-set to the data imported from a database:

```

private void PopulateMainChart()
{
    ///--- Get the data from the database.
    DataTable[] dtArr = GetCategoryQuarterlySales();

    ///--- Find the main chart based on its title text, and get
    ///--- a reference to it.
    Chart mainChart = FindChart("Quarterly Sales 2003");

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

    ///--- Each DataTable has a single row of data.
    int iRow = 29;
    for (int i = 0; i < dtArr.Length; i++)
    {
        DataTable dt = dtArr[i];

        ///--- Import data from the DataTable.
        ws.ImportData(dt, ws[iRow, 1]);

        ///--- Add the imported data as a new Series object
        ///--- in the chart's collection. There will be a variable
        ///--- number of series objects depending on which categories
        ///--- were selected for display.
        Area a = ws.CreateArea(iRow, 2, 1, 4);
        Series srs = seriesCol.CreateSeries(a);
        srs.NameFormula = ws[iRow, 1].Name;

        iRow++;
    }

    ///--- Re-set the category data. The size will vary
    ///--- depending on the selected categories.
    seriesCol.CategoryData =
        ws.CreateArea(29, 1, dtArr.Length, 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;
    }
}

```