ISSN 2250-3153

Dot Net Charting (Integration with ASP.NET)

Anubhav Tiwari

* R&D Dept., Syscom Corporation Ltd.

Abstract- This paper talks about
•Why Dot Net Charting?
•Features of Dot Net Charting?
•How to use?

Index Terms- Why Dot Net Charting, Features, How to use;

I. INTRODUCTION

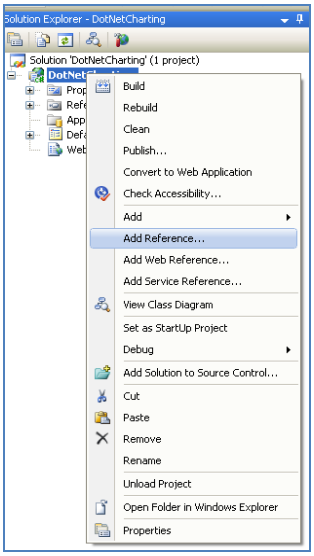
Dear Readers,
First of all, I would say this paper is just for ASP.net developers.
Many-a-times we need to generate charts in our ASP.Net project which becomes a cumbersome task for dot net developers. But dot net charting has made it so easy.
It makes our source for the easiest to use and makes most visually stunning charts available. With a few simple lines of codes Dot Net Charting can generate stylish charts from our own custom databases
Now in this paper I have consolidated most of the charts and how to use them. I have also included code for each chart as well.

Features of Dot Net Charting:

- .Net Framework 2.0, 3.0, 3.5 and 4.0+ supports – Dot Net charting support all versions of dot net framework.
- 2D and 3D rendering of charts – In Dot Net charting, we can customize our charts as per requirement 2-dimensional or 3-dimensional.
- Printer Friendly text – The text generated in the charts are printer friendly and are clearly printed from printer.
- Color Synchronization – Dot Net charting has a very good color synchronization and support multicolor.
- Rich Tool Tip support – In Dot Net charting we can also customize tool tip to appear on the charts.
- Transparent color support – We can also apply transparency in colors in Dot Net charting.

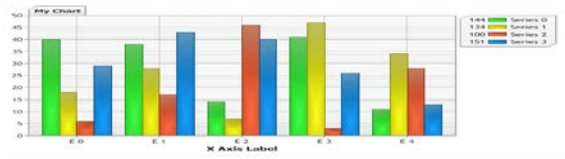
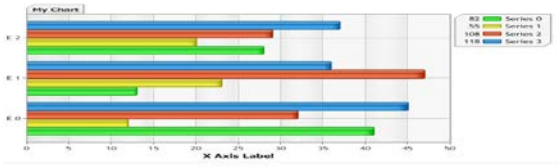
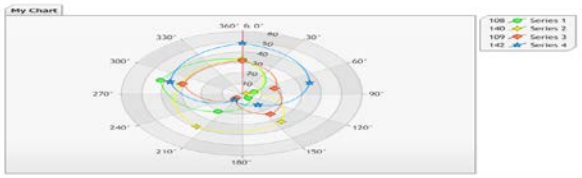
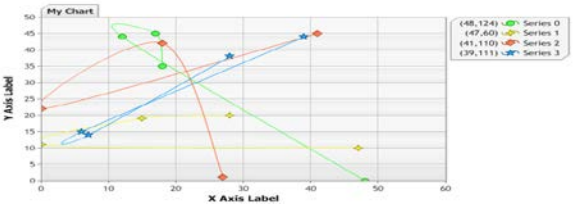
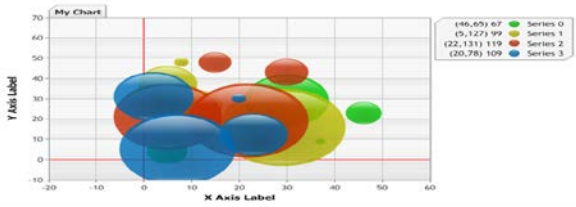
How to Use:

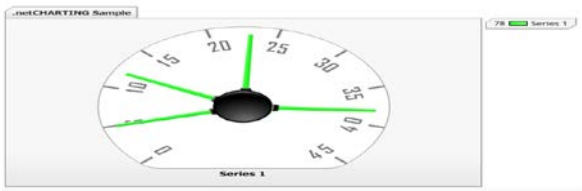
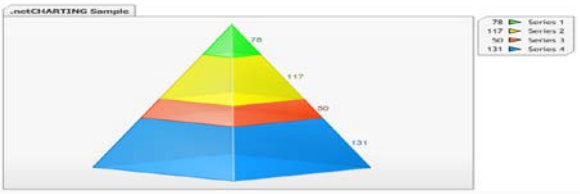
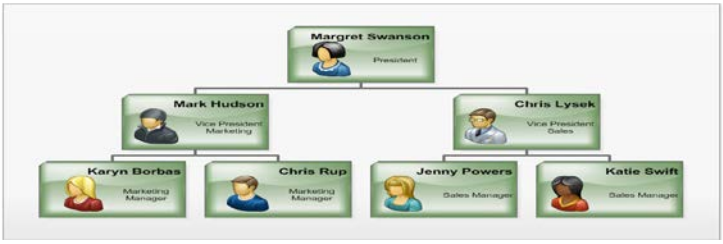
Download DLL (licensed) from this link www.dotnetcharting.com/download.aspx and provide a reference of DLL in our project and register the assembly on the page and put the control on aspx page as shown below:



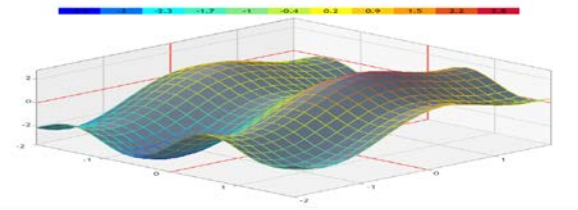

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Do
<%@ Register Assembly="dotnetCHARTING" Namespace="dotnetCHARTING" TagPrefix="dotnetCHAR
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xh
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <dotnetCHARTING:Chart ID="chart" runat="server">
        </dotnetCHARTING:Chart>
    </div>
    </form>
</body>
</html>
```

Chart Types available in Dot Net Charting:

Chart Types	
Combo	
Combo Horizontal	
Radar	
Pie	
Scatter	
Bubble	

Financial	
Gauge	
Multiple	
Funnel	
Cone	
Pyramid	
Organizational	

Surface



Code for generating above charts:

I. Combo Chart:

```
chart.Title="Combo sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name";
chart.Type=ChartType.Combo;
chart.Series.SqlStatement = "your query";
chart.Series.DataFields = "xaxis=DataField,yaxis=DataField";
chart.SeriesCollection.Add();
```

II. Combo Horizontal Chart:

```
chart.Title="Combo Horizontal sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.ComboHorizontal;
chart.Series.SqlStatement = "your query";
chart.Series.DataFields = "xaxis=DataField,yaxis=DataField";
chart.SeriesCollection.Add();
```

III. Radar Chart:

```
chart.Title="Radar sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.Radar;
chart.Series.SqlStatement = "your query";
chart.Series.DataFields = "xaxis=DataField,yaxis=DataField";
chart.SeriesCollection.Add();
```

IV. Pie Chart:

```
chart.Title="Pie sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type=ChartType.Pies;
chart.Series.SqlStatement = "your query";
chart.SeriesCollection.Add();
```

V. Scatter Chart:

```
chart.Title="Scatter sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type=ChartType.Scatter;
chart.Series.SqlStatement = "your query";
chart.SeriesCollection.Add();
```

VI. Bubble Chart:

```
chart.Title="Bubble sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type = ChartType.Bubble;
chart.Series.SqlStatement = "your query";
chart.SeriesCollection.Add();
```

VII. Financial Chart:

```
chart.Title="Financial sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.Financial;
chart.Series.SqlStatement = "your query";
chart.DefaultSeries.Type = SeriesTypeFinancial.CandleStick;
chart.SeriesCollection.Add();
```

VIII. Gauge Chart:

```
chart.Title="Gauge sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.Gauges;
chart.Series.SqlStatement = "your query";
chart.SeriesCollection.Add();
```

IX. Multiple Chart:

```
chart.Title="Multiple chart sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.Multiple;
chart.Series.SqlStatement = "your query";
chart.DefaultSeries.Type = SeriesTypeMultiple.Pyramid;
chart.SeriesCollection.Add();
```

X. Funnel Chart:

```
chart.Title="Funnel sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.MultipleGrouped;
chart.Series.SqlStatement = "your query";
chart.DefaultSeries.Type = SeriesTypeMultiple.FunnelPyramid;
chart.SeriesCollection.Add();
```

XI. Cone Chart:

```
chart.Title="Cone sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.MultipleGrouped;
chart.Series.SqlStatement = "your query";
chart.DefaultSeries.Type = SeriesTypeMultiple.Cone;
chart.SeriesCollection.Add();
```

XII. Pyramid Chart:

```
chart.Title="Pyramid sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type= ChartType.MultipleGrouped;
chart.Series.SqlStatement = "your query";
chart.DefaultSeries.Type = SeriesTypeMultiple.Pyramid;
chart.SeriesCollection.Add();
```

XIII. Organizational Chart:

```
chart.Title="Organization chart sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type = ChartType.Organizational;
chart.Series.SqlStatement = "your query";
chart.DefaultElement.Annotation = new Annotation("<block hAlign='right'
fStyle='bold' fSize='11'>%Name<row><img:../images/Org%image.png>%position");
chart.SeriesCollection.Add();
```

XIV. Surface Chart:

```
chart.Title="Surface chart sample";
chart.DefaultSeries.ConnectionString = "your connection string";
chart.TempDirectory="directory name ";
chart.Type = ChartType.Surface;
chart.Series.SqlStatement = "your query";
chart.SeriesCollection.Add();
```

IV. CONCLUSION

Dot Net Charting is very helpful for developers for generating charts. It reduces the developer's effort and generates high quality charts fulfilling the requirements in less time.
Thus a developer becomes more productive in chart generating.
Moreover, one more advantage of using Dot Net Charting is that it is always fully backward compatible. If we used the previous version of Dot Net Charting in our project & we want to have new look of chart that Dot Net charting provided in its new version, we won't need to bother about it. We can easily maintain it with a simple web.config entry and enable a new look on the chart..

ACKNOWLEDGMENT

This work was supported by Syscom Corporation Ltd.

REFERENCES

[1] http:// www.dotnetcharting.com
[2] https://en.wikipedia.org/wiki/Chart

AUTHORS

First Author – Anubhav Tiwari, M.C.A, Software Engineer, anubhavtiwari87@gmail.com