

介绍MSChart的常用属性和事件.....	2
数据源绑定（一），具体代码如下：	6
数据源绑定方式（二）	8
数据源绑定（三）：	9
数据源绑定（四）：	11
数据源绑定（四）——行绑定	13
数据源绑定（五）	15
MSChart的研究(二) 收藏	17
一.数据源——xml	18
二.数据源——Excel	18
三.数据源——CSV文件	20
继续对MSChart学习:本次主要针对MSChart的图形交互和AJAX的应用。	22
二. AJAX(一)——Click Event.....	23
三. AJAX(二)——Chart as Trriger	29
四. AJAX(三)——AJAX更新	31
五. AJAX(四)——Real Time Chart	35
解决MSChart底部横坐标显示不全的问题 收藏.....	38
mschart 上下限阈值 画横线 收藏.....	38
MSChart for vs2008 收藏	39
MSCHART全部属性方法介绍 收藏	41
在access 中创建库表（.net实现）	57
MSChart学习帮助-开发日志（二）.net 3.5 图表控件 收藏.....	62
三个电压，三个电流	70
.NET MSChart应用的一个简单例子	76
初试 ASP.NET Chart Control.....	83
微软图表控件使用初探--数据绑定(交叉表)	88
MSChart图表控件的一些使用	96
微软Chart控件导学--数据绑定	110
Mschart学习(MSchart介绍).....	116
微软图表控件MsChart使用初探	122
微软Chart控件导学--渲染图表	135

介绍 MSChart 的常用属性和事件

MSChart 的元素组成

最常用的属性包括

ChartAreas: 增加多个绘图区域，每个绘图区域包含独立的图表组、数据源，用于多个图表类型在一个绘图区不兼容时。

最常用的属性包括

ChartAreas: 增加多个绘图区域，每个绘图区域包含独立的图表组、数据源，用于多个图表类型在一个绘图区不兼容时。

AlignmentOrientation: 图表区对齐方向，定义两个绘图区域间的对齐方式。

AlignmentStyle: 图表区对齐类型，定义图表间用以对其的元素。

AlignWithChartArea: 参照对齐的绘图区名称。

InnerPlotPosition: 图表在绘图区内的位置属性。

Auto: 是否自动对齐。

Height: 图表在绘图区内的高度（百分比，取值在 0-100 ）

Width: 图表在绘图区内的宽度（百分比，取值在 0-100 ）

X,Y: 图表在绘图区内左上角坐标

Position: 绘图区位置属性，同 InnerPlotPosition。

Name: 绘图区名称。

Axis: 坐标轴集合

Title: 坐标轴标题

TitleAlignment: 坐标轴标题对齐方式

Interval: 轴刻度间隔大小

IntervalOffset: 轴刻度偏移量大小

MinorGrid: 次要辅助线

MinorTickMark: 次要刻度线

MajorGrid: 主要辅助线

MajorTickMark: 主要刻度线

DataSourceID: MSChart 的数据源。

Legends: 图例说明。

Palette: 图表外观定义。

Series: 最重要的属性，图表集合，就是最终看到的饼图、柱状图、线图、点图等构成的集合；可以将多种相互兼容的类型放在一个绘图区域内，形成复合图。

IsValueShownAsLabel: 是否显示数据点标签，如果为 true，在图表中显示每一个数据值

Label: 数据点标签文本

LabelFormat: 数据点标签文本格式

LabelAngle: 标签字体角度

Name: 图表名称

Points: 数据点集合

XValueType: 横坐标轴类型

YValueType: 纵坐标轴类型

XValueMember: 横坐标绑定的数据源（如果数据源为 Table，则填写横坐标要显示的字段名称）

YValueMembers: 纵坐标绑定的数据源（如果数据源为 Table，则填写纵坐标要显示的字段名称，纵坐标可以有多个）

ChartArea: 图表所属的绘图区域名称

ChartType: 图表类型（柱形、饼形、线形、点形等）

Legend: 图表使用的图例名称

Titles: 标题集合。

width: MSChart 的宽度。

height: MSChart 的高度。

常用事件:

Series1.Points.DataBind()

绑定数据点集合，如果要在一个 MSChart 控件的一个绘图区(ChartArea)内添加多个不同数据源的图表，就用这个主动绑定数据集合的方法。可以将表中指定字段的值绑定到指定的坐标轴上。

MSChart1.DataBind()

给整个 MSChart 绑定一个数据源，该 MSChart 中的图表全部可以使用该数据源作为统计来源。

本文来自 CSDN 博客，转载请标明出处：<http://blog.csdn.net/jetable/archive/2009/01/08/3727444.aspx>

下面给一个自己做的例子：

view plaincopy to clipboardprint?

```
int[] date = new int[] { 20, 40, 50, 80, 30, 10, 60 };
```

```
int[] test = new int[] { 10, 80, 70, 40, 20, 50, 90};
```

```
Chart1.Width = 600;           //图片宽度
```

```
Chart1.Height = 400;          //图片高度
```

```
Chart1.BackColor = Color.Azure;           //图片背景色
```

```
//建一个图表集合
```

```
Series series = new Series("test");
```

```
series.ChartType = SeriesChartType.Column;           //图标集类型，Line 为直线，SpLine
```

为曲线

```
series.Color = Color.Green;           //线条颜色
```

```
series.BorderWidth = 2;           //线条宽度
```

```
series.ShadowOffset = 1;           //阴影宽度
```

```
series.IsVisibleInLegend = false;           //是否显示数据说明
```

```
series.IsValueShownAsLabel = true;
```

```
series.MarkerStyle = MarkerStyle.Diamond;           //线条上的数据点标志类型
```

```
series.MarkerSize = 8;           // 标志的大小
```

```
DateTime date1 = DateTime.Now.Date;
```

```
for (int i = 0; i < date.Length; i++)
```

```
{
```

```
    series.Points.AddXY(date1, date[i]);
```

```
    date1 = date1.AddDays(1);
```

```

    }
    Chart1.Series.Add(series);           //把数据集添加到 Chart1 中

    //再建一个图表集合
    Series series1 = new Series("ok");
    series1.ChartType = SeriesChartType.Column;           //图标集类型，Line 为直线，SpLine
为曲线
    series1.Color = Color.Red;           //线条颜色
    series1.BorderWidth = 2;             //线条宽度
    series1.ShadowOffset = 1;            //阴影宽度
    series1.IsVisibleInLegend = false;    //是否显示数据说明
    series1.IsValueShownAsLabel = true;
    series1.MarkerStyle = MarkerStyle.Diamond;
    series1.MarkerSize = 8;

    DateTime date2 = DateTime.Now.Date;

    for (int i = 0; i < test.Length; i++)
    {
        series1.Points.AddXY(date2, test[i]);
        date2 = date2.AddDays(1);
    }
    Chart1.Series.Add(series1);           //把数据集添加到 Chart1 中

    //设置坐标轴
    Chart1.ChartAreas[0].AxisX.LineColor = Color.Blue;
    Chart1.ChartAreas[0].AxisY.LineColor = Color.Blue;
    Chart1.ChartAreas[0].AxisX.LineWidth = 2;
    Chart1.ChartAreas[0].AxisY.LineWidth = 2;
    Chart1.ChartAreas[0].AxisY.Title = "总额";
    //设置网格线
    Chart1.ChartAreas[0].AxisX.MajorGrid.LineColor = Color.Blue;
    Chart1.ChartAreas[0].AxisY.MajorGrid.LineColor = Color.Blue;
int[] date = new int[] { 20, 40, 50, 80, 30, 10, 60 };
int[] test = new int[] { 10, 80, 70, 40, 20, 50, 90 };

    Chart1.Width = 600;           //图片宽度
    Chart1.Height = 400;           //图片高度
    Chart1.BackColor = Color.Azure;           //图片背景色

    //建一个图表集合
    Series series = new Series("test");
    series.ChartType = SeriesChartType.Column;           //图标集类型，Line 为直线，SpLine
为曲线
    series.Color = Color.Green;           //线条颜色
    series.BorderWidth = 2;             //线条宽度

```

```

series.ShadowOffset = 1;           //阴影宽度
series.IsVisibleInLegend = false; //是否显示数据说明
series.IsValueShownAsLabel = true;
series.MarkerStyle = MarkerStyle.Diamond; //线条上的数据点标志类型
series.MarkerSize = 8;           // 标志的大小

```

```
DateTime date1 = DateTime.Now.Date;
```

```

for (int i = 0; i < date.Length; i++)
{
    series.Points.AddXY(date1, date[i]);
    date1 = date1.AddDays(1);
}

```

```
Chart1.Series.Add(series);           //把数据集添加到 Chart1 中
```

```
//再建一个图表集合
```

```
Series series1 = new Series("ok");
```

```
series1.ChartType = SeriesChartType.Column;           //图标集类型，Line 为直线，SpLine
```

为曲线

```

series1.Color = Color.Red;           //线条颜色
series1.BorderWidth = 2;             //线条宽度
series1.ShadowOffset = 1;           //阴影宽度
series1.IsVisibleInLegend = false; //是否显示数据说明
series1.IsValueShownAsLabel = true;
series1.MarkerStyle = MarkerStyle.Diamond;
series1.MarkerSize = 8;

```

```
DateTime date2 = DateTime.Now.Date;
```

```

for (int i = 0; i < test.Length; i++)
{
    series1.Points.AddXY(date2, test[i]);
    date2 = date2.AddDays(1);
}

```

```
Chart1.Series.Add(series1);           //把数据集添加到 Chart1 中
```

```
//设置坐标轴
```

```
Chart1.ChartAreas[0].AxisX.LineColor = Color.Blue;
```

```
Chart1.ChartAreas[0].AxisY.LineColor = Color.Blue;
```

```
Chart1.ChartAreas[0].AxisX.LineWidth = 2;
```

```
Chart1.ChartAreas[0].AxisY.LineWidth = 2;
```

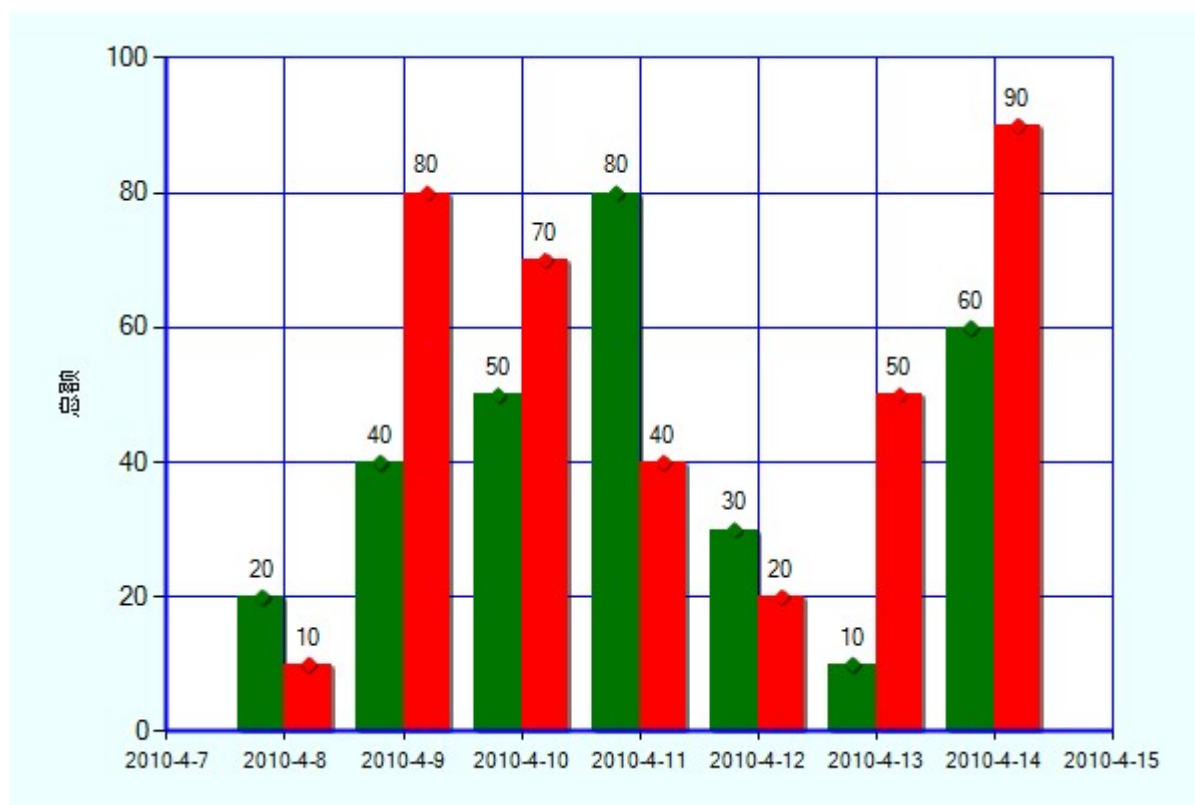
```
Chart1.ChartAreas[0].AxisY.Title = "总额";
```

```
//设置网格线
```

```
Chart1.ChartAreas[0].AxisX.MajorGrid.LineColor = Color.Blue;
```

```
Chart1.ChartAreas[0].AxisY.MajorGrid.LineColor = Color.Blue;
```

最后显示的图片：



数据源绑定（一），具体代码如下：

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    string fileName = Server.MapPath(".");
```

```
    fileName += "\\App_Data\\chartdata.mdb";
```

```
    Chart1.Width = 600;                                //图表宽度
```

```
    Chart1.Height = 400;                                //图表高度
```

```
    Chart1.BackColor = Color.Azure;                      //图表背景色
```

```
    Chart1.Titles.Add("房地产");                        //图表标题
```

```
    //新建连接
```

```
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source="+fileName))
```

```
    {
```

```
        OleDbCommand cmd = new OleDbCommand("SELECT Name, Sales FROM REPS", con);
```

```
        con.Open();
```

```
        OleDbDataReader read = cmd.ExecuteReader();
```

```
        Chart1.DataBindTable(read, "Name");
```

```
        read.Close();
```

```

    }
    //注意数据绑定后，它的 series 是 1 而不是 0
    Chart1.Series[1].IsValueShownAsLabel = true;           //是否显示数据
    Chart1.Series[1].IsVisibleInLegend = false;            //是否显示数据说明
    Chart1.Series[1].MarkerStyle = MarkerStyle.Circle;     //线条上的数据点标志类型
    Chart1.Series[1].MarkerSize = 8;                       //标志大小

    Chart1.ChartAreas[0].AxisX.LineColor = Color.Blue;     //X 轴颜色
    Chart1.ChartAreas[0].AxisY.LineColor = Color.Blue;     //Y 轴颜色
    Chart1.ChartAreas[0].AxisX.LineWidth = 2;             //X 轴宽度
    Chart1.ChartAreas[0].AxisY.LineWidth = 2;             //Y 轴宽度
    Chart1.ChartAreas[0].AxisY.Title = "价格";            //Y 轴标题

}

protected void Page_Load(object sender, EventArgs e)
{
    string fileName = Server.MapPath(".");
    fileName += "\\App_Data\\chartdata.mdb";

    Chart1.Width = 600;                                     //图表宽度
    Chart1.Height = 400;                                    //图表高度
    Chart1.BackColor = Color.Azure;                        //图表背景色
    Chart1.Titles.Add("房地产");                          //图表标题

    //新建连接
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source="+fileName))
    {
        OleDbCommand cmd = new OleDbCommand("SELECT Name, Sales FROM REPS", con);
        con.Open();
        OleDbDataReader read = cmd.ExecuteReader();
        Chart1.DataBindTable(read, "Name");
        read.Close();
    }
    //注意数据绑定后，它的 series 是 1 而不是 0
    Chart1.Series[1].IsValueShownAsLabel = true;           //是否显示数据
    Chart1.Series[1].IsVisibleInLegend = false;            //是否显示数据说明
    Chart1.Series[1].MarkerStyle = MarkerStyle.Circle;     //线条上的数据点标志类型
    Chart1.Series[1].MarkerSize = 8;                       //标志大小

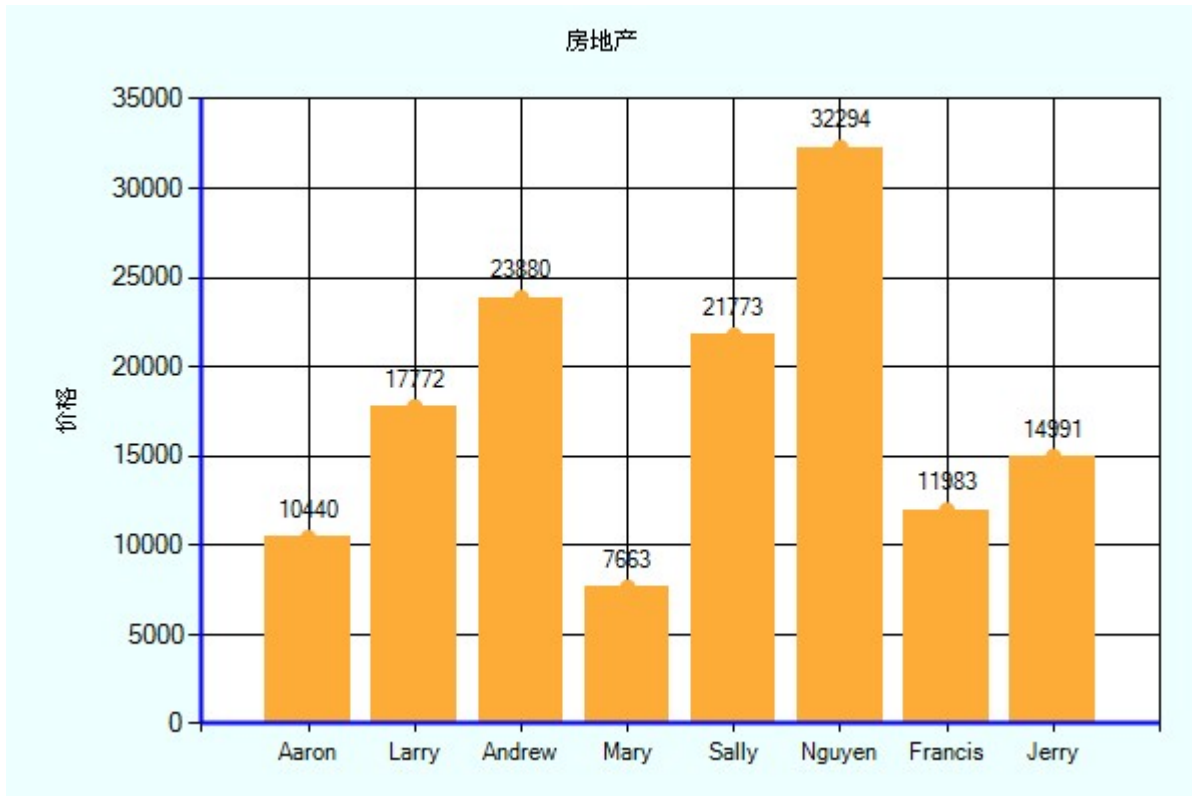
    Chart1.ChartAreas[0].AxisX.LineColor = Color.Blue;     //X 轴颜色
    Chart1.ChartAreas[0].AxisY.LineColor = Color.Blue;     //Y 轴颜色
    Chart1.ChartAreas[0].AxisX.LineWidth = 2;             //X 轴宽度
    Chart1.ChartAreas[0].AxisY.LineWidth = 2;             //Y 轴宽度

```

```
Chart1.ChartAreas[0].AxisY.Title = "价格"; //Y 轴标题
```

```
}
```

效果图如下：



数据源绑定方式（二）

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    string fileName = Server.MapPath(".");
```

```
    fileName += "\\App_Data\\chartdata.mdb";
```

```
    Chart1.Width = 600;
```

```
    //图表宽度
```

```
    Chart1.Height = 400;
```

```
    //图表高度
```

```
    Chart1.BackColor = Color.Azure;
```

```
    //图表背景色
```

```
    Chart1.Titles.Add("房地产");
```

```
    //图表标题
```

```
    //新建连接
```

```
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source="+fileName))
```

```

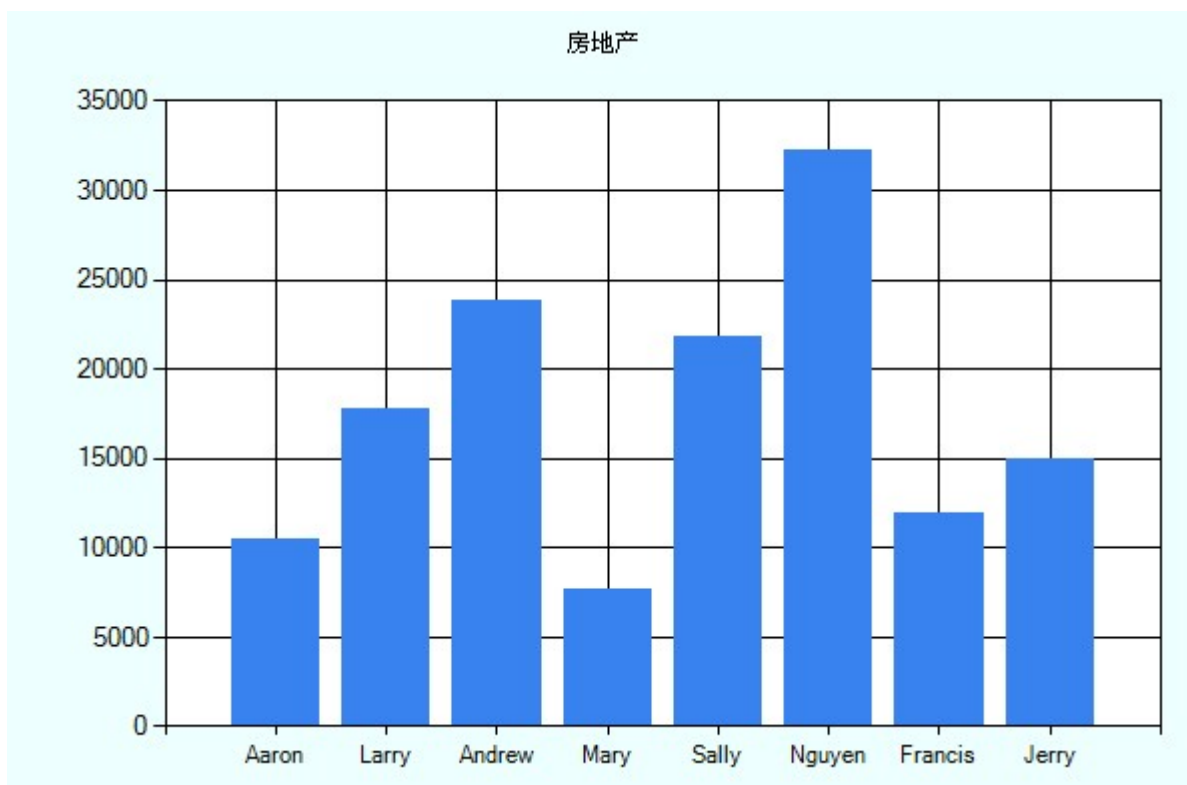
{
    OleDbCommand cmd = new OleDbCommand("SELECT Name, Sales FROM REPS", con);
    con.Open();
    //OleDbDataReader read = cmd.ExecuteReader();
    //Chart1.DataBindTable(read, "Name");
    //read.Close();
    Chart1.DataSource = cmd;

    Chart1.Series[0].XValueMember = "Name";
    Chart1.Series[0].YValueMembers = "Sales";
    con.Close();

    Chart1.DataBind();
}
}

```

效果图如下：



数据源绑定（三）：

view plaincopy to clipboardprint?

```

protected void Page_Load(object sender, EventArgs e)
{

```

```

string fileName = Server.MapPath(".");
fileName += "\\App_Data\\chartdata.mdb";

Chart1.Width = 600;           //图表宽度
Chart1.Height = 400;         //图表高度
Chart1.BackColor = Color.Azure; //图表背景色
Chart1.Titles.Add("房地产"); //图表标题

//新建连接
using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source="+fileName))
{
    OleDbCommand cmd = new OleDbCommand("SELECT GrossSales FROM SALES WHERE
QuarterEnding < #01/01/2002#", con);
    con.Open();

    OleDbDataReader read = cmd.ExecuteReader();
    Chart1.Series[0].Points.DataBindY(read, "GrossSales");
    read.Close();
}

}

protected void Page_Load(object sender, EventArgs e)
{
    string fileName = Server.MapPath(".");
    fileName += "\\App_Data\\chartdata.mdb";

    Chart1.Width = 600;           //图表宽度
    Chart1.Height = 400;         //图表高度
    Chart1.BackColor = Color.Azure; //图表背景色
    Chart1.Titles.Add("房地产"); //图表标题

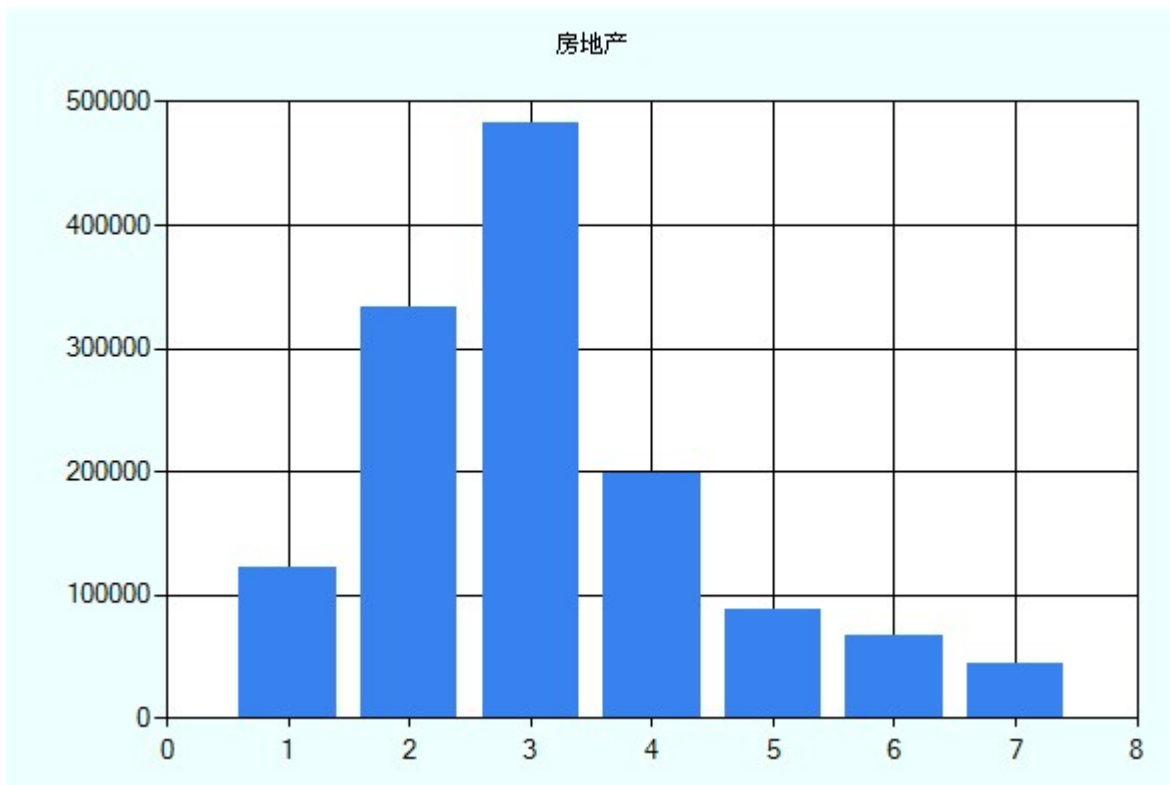
    //新建连接
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source="+fileName))
    {
        OleDbCommand cmd = new OleDbCommand("SELECT GrossSales FROM SALES WHERE
QuarterEnding < #01/01/2002#", con);
        con.Open();

        OleDbDataReader read = cmd.ExecuteReader();
        Chart1.Series[0].Points.DataBindY(read, "GrossSales");
        read.Close();
    }

}

```

效果图如下：



数据源绑定（四）：

```
view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)
{
    string fileName = Server.MapPath(".");
    fileName += "\\App_Data\\chartdata.mdb";

    Chart1.Width = 600;                //图表宽度
    Chart1.Height = 400;               //图表高度
    Chart1.BackColor = Color.Azure;    //图表背景色
    Chart1.Titles.Add("房地产");       //图表标题

    //新建连接
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source="+fileName))
    {
        OleDbCommand cmd = new OleDbCommand("SELECT * FROM REPSALES", con);
        con.Open();

        OleDbDataReader read = cmd.ExecuteReader();
        Chart1.DataBindCrossTable(read,
```

```

        "Name",
        "Year",
        "Sales",
        "Label=Commissions{C}");

    read.Close();
}

// Chart1.Legends.Add("Name");

}

protected void Page_Load(object sender, EventArgs e)
{
    string fileName = Server.MapPath(".");
    fileName += "\\App_Data\\chartdata.mdb";

    Chart1.Width = 600;           //图表宽度
    Chart1.Height = 400;         //图表高度
    Chart1.BackColor = Color.Azure; //图表背景色
    Chart1.Titles.Add("房地产"); //图表标题

    //新建连接
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source="+fileName))
    {
        OleDbCommand cmd = new OleDbCommand("SELECT * FROM REPSALES", con);
        con.Open();

        OleDbDataReader read = cmd.ExecuteReader();
        Chart1.DataBindCrossTable(read,
                                "Name",
                                "Year",
                                "Sales",
                                "Label=Commissions{C}");

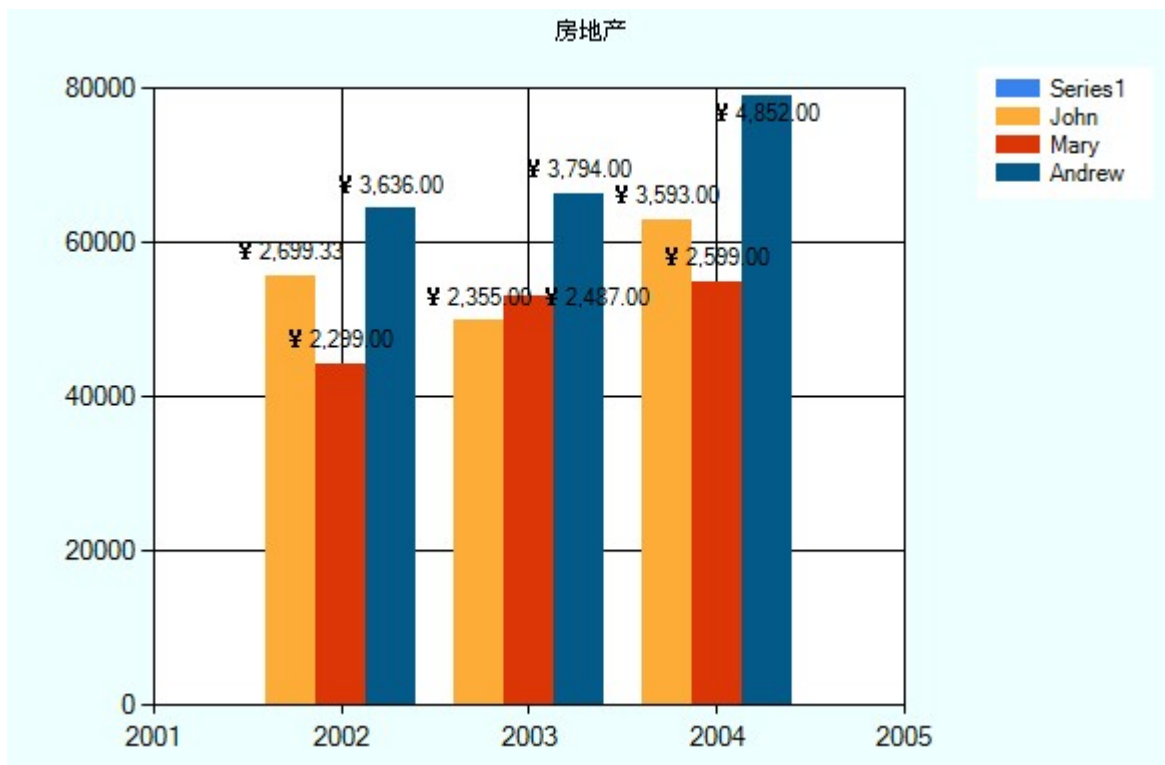
        read.Close();
    }

    // Chart1.Legends.Add("Name");

}

```

效果图如下：（有点不理想，Legend 中出现了 Series1，关于 legend 的设置还有待研究）



数据源绑定（四）——行绑定

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    string fileName = Server.MapPath(".");
```

```
    fileName += "\\App_Data\\chartdata.mdb";
```

```
    Chart1.Width = 600; //图表宽度
```

```
    Chart1.Height = 400; //图表高度
```

```
    Chart1.BackColor = Color.Azure; //图表背景色
```

```
    Chart1.Titles.Add("Bind By Rows"); //图表标题
```

```
    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source="+ fileName))
```

```
    {
```

```
        OleDbCommand cmd = new OleDbCommand("SELECT * FROM SALESCOUNTS", con);
```

```
        OleDbDataAdapter oda = new OleDbDataAdapter();
```

```
        oda.SelectCommand = cmd;
```

```
        DataSet ds = new DataSet();
```

```
        oda.Fill(ds);
```

```
        foreach (DataRow row in ds.Tables[0].Rows)
```

```
        {
```

```

        string seriesName=row["SalesRep"].ToString();
        Chart1.Series.Add(seriesName);
        Chart1.Series[seriesName].ChartType = SeriesChartType.Line;
        Chart1.Series[seriesName].BorderWidth = 2;
        for (int col = 1; col < ds.Tables[0].Columns.Count; col++)
        {
            string colName = ds.Tables[0].Columns[col].ColumnName;
            int yVal = Convert.ToInt32(row[colName].ToString());

            Chart1.Series[seriesName].Points.AddXY(colName, yVal);
        }
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    string fileName = Server.MapPath(".");
    fileName += "\\App_Data\\chartdata.mdb";

    Chart1.Width = 600;                //图表宽度
    Chart1.Height = 400;               //图表高度
    Chart1.BackColor = Color.Azure;    //图表背景色
    Chart1.Titles.Add("Bind By Rows"); //图表标题

    using (OleDbConnection con=new OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source=" + fileName))
    {
        OleDbCommand cmd = new OleDbCommand("SELECT * FROM SALESCOUNTS", con);
        OleDbDataAdapter oda = new OleDbDataAdapter();
        oda.SelectCommand = cmd;
        DataSet ds = new DataSet();
        oda.Fill(ds);

        foreach (DataRow row in ds.Tables[0].Rows)
        {
            string seriesName=row["SalesRep"].ToString();
            Chart1.Series.Add(seriesName);
            Chart1.Series[seriesName].ChartType = SeriesChartType.Line;
            Chart1.Series[seriesName].BorderWidth = 2;
            for (int col = 1; col < ds.Tables[0].Columns.Count; col++)
            {
                string colName = ds.Tables[0].Columns[col].ColumnName;
                int yVal = Convert.ToInt32(row[colName].ToString());

                Chart1.Series[seriesName].Points.AddXY(colName, yVal);
            }
        }
    }
}

```

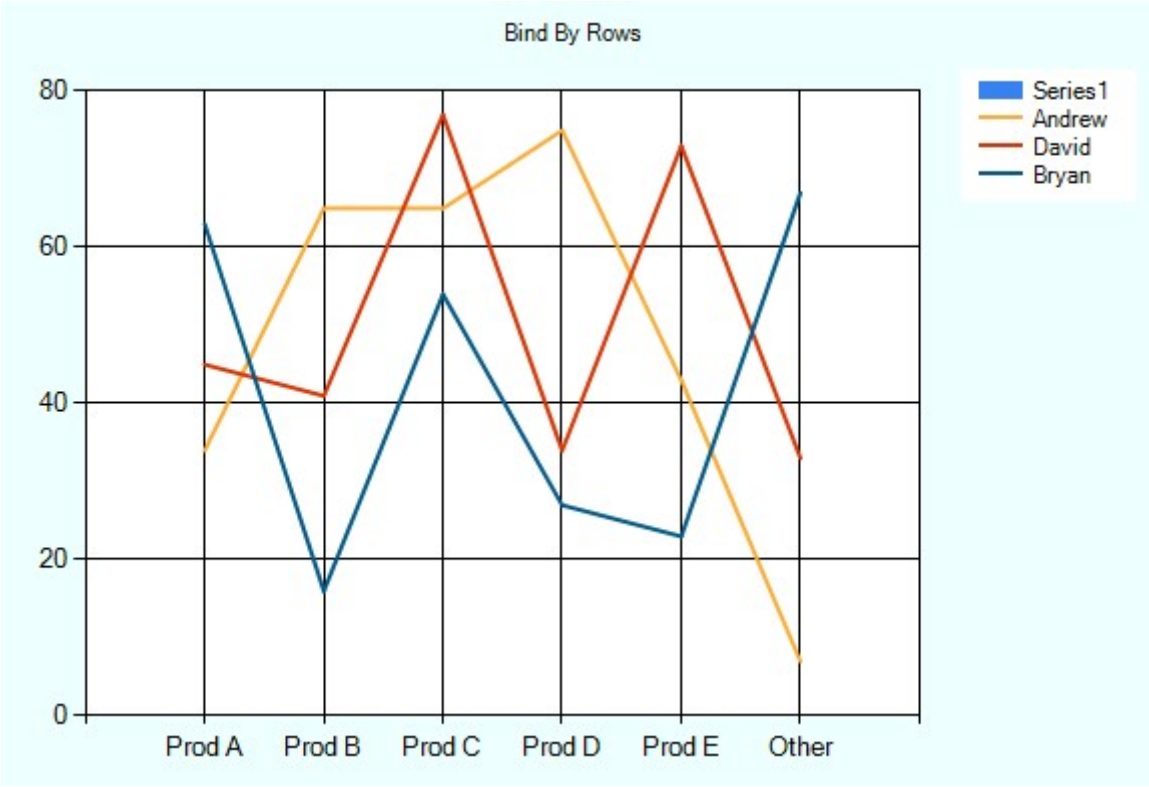
```

    }
}

}
}

```

效果图如下：



数据源绑定（五）

```

view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)
{
    // Initialize arrays for series 1
    double[] yval1 = { 2, 6, 5 };
    string[] xval1 = { "Peter", "Andrew", "Julie" };

    // Initialize arrays for series 2
    double[] yval2 = { 4, 5, 3 };
    string[] xval2 = { "Peter", "Andrew", "Dave" };

    // Initialize arrays for series 3
    double[] yval3 = { 6, 5 };
    string[] xval3 = { "Julie", "Mary" };
}

```

```

        // Bind the arrays to each data series
        Chart1.Series["Series1"].Points.DataBindXY(xval1, yval1);
        Chart1.Series["Series2"].Points.DataBindXY(xval2, yval2);
        Chart1.Series["Series3"].Points.DataBindXY(xval3, yval3);

        // Align series using their X axis labels
        Chart1.AlignDataPointsByAxisLabel();
        Chart1.ChartAreas[0].Area3DStyle.Enable3D = true;    //展示 3D

    }
protected void Page_Load(object sender, EventArgs e)
{
    // Initialize arrays for series 1
    double[] yval1 = { 2, 6, 5 };
    string[] xval1 = { "Peter", "Andrew", "Julie" };

    // Initialize arrays for series 2
    double[] yval2 = { 4, 5, 3 };
    string[] xval2 = { "Peter", "Andrew", "Dave" };

    // Initialize arrays for series 3
    double[] yval3 = { 6, 5 };
    string[] xval3 = { "Julie", "Mary" };

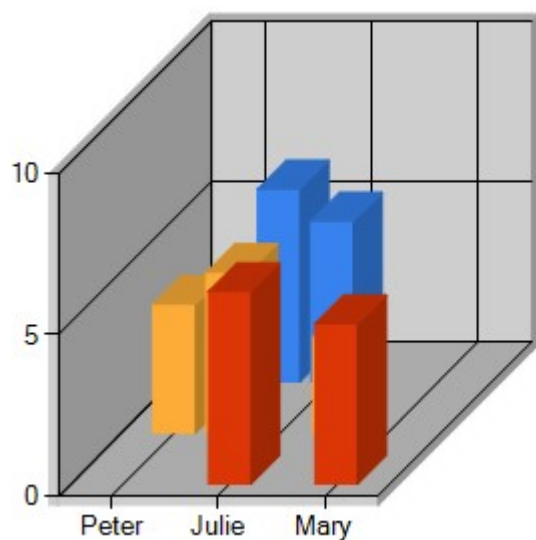
    // Bind the arrays to each data series
    Chart1.Series["Series1"].Points.DataBindXY(xval1, yval1);
    Chart1.Series["Series2"].Points.DataBindXY(xval2, yval2);
    Chart1.Series["Series3"].Points.DataBindXY(xval3, yval3);

    // Align series using their X axis labels
    Chart1.AlignDataPointsByAxisLabel();
    Chart1.ChartAreas[0].Area3DStyle.Enable3D = true;    //展示 3D

}

```

效果图如下：



MSChart 的研究(二) 收藏

继续上篇：

chart 可以有以下几种数据源：

- 1.视图对象
- 2.dataRead 对象 (xml, sql, oledb)
- 3.dataSet 对象
- 4.dataRow 对象
- 5.dataColumn 对象
- 6.数组对象
- 7.List 对象
- 8.sqlCommand 对象 (oledbCommand)
- 9.sqlDataAdapter 对象 (oledbDataAdapter)
- 10.所有实现 IEnumerable 借口的对象

一.数据源——xml

代码:

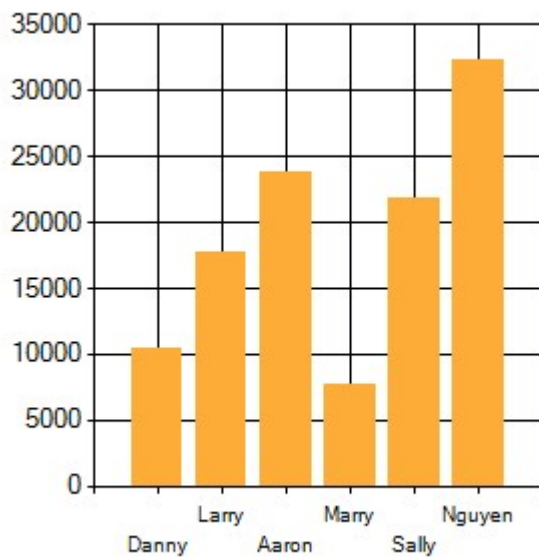
view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
{
    string fileNameString = Server.MapPath(".\\App_Data\\data.xml");
    string fileNameSchema = Server.MapPath(".\\App_data\\data.xsd");

    DataSet ds = new DataSet();
    ds.ReadXmlSchema(fileNameSchema);
    ds.ReadXml(fileNameString);

    DataView dv = new DataView(ds.Tables[0]);
    Chart1.DataBindTable(dv,"Name");
}
```

效果图:



二.数据源——Excel

代码:

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
{
    string fileNameString = Server.MapPath(".\\App_Data\\ExcelData.xls");

    //连接 Excel 数据源
    string conStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + fileNameString + ";Extended
Properties=\"Excel 8.0;HDR=YES\"";
    OleDbConnection con = new OleDbConnection(conStr);
    con.Open();
    OleDbCommand cmd = new OleDbCommand("select * from [data1$A1:E25]", con);

    OleDbDataReader read = cmd.ExecuteReader();

    chart1.DataBindTable(read, "HOUR");

    read.Close();
    con.Close();

    for (int i = 1; i < chart1.Series.Count; i++)
    {
        chart1.Series[i].ChartType = SeriesChartType.Line;
        chart1.Series[i].BorderWidth = 2;
    }

    chart1.Legends.Add("Default");
    //设置表
    chart1.Width = 600;
    chart1.Height = 400;
    chart1.BackColor = Color.Azure;
}

protected void Page_Load(object sender, EventArgs e)
{
    string fileNameString = Server.MapPath(".\\App_Data\\ExcelData.xls");

    //连接 Excel 数据源
    string conStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + fileNameString + ";Extended
Properties=\"Excel 8.0;HDR=YES\"";
    OleDbConnection con = new OleDbConnection(conStr);
    con.Open();
    OleDbCommand cmd = new OleDbCommand("select * from [data1$A1:E25]", con);

    OleDbDataReader read = cmd.ExecuteReader();

    chart1.DataBindTable(read, "HOUR");
```

```

read.Close();
con.Close();

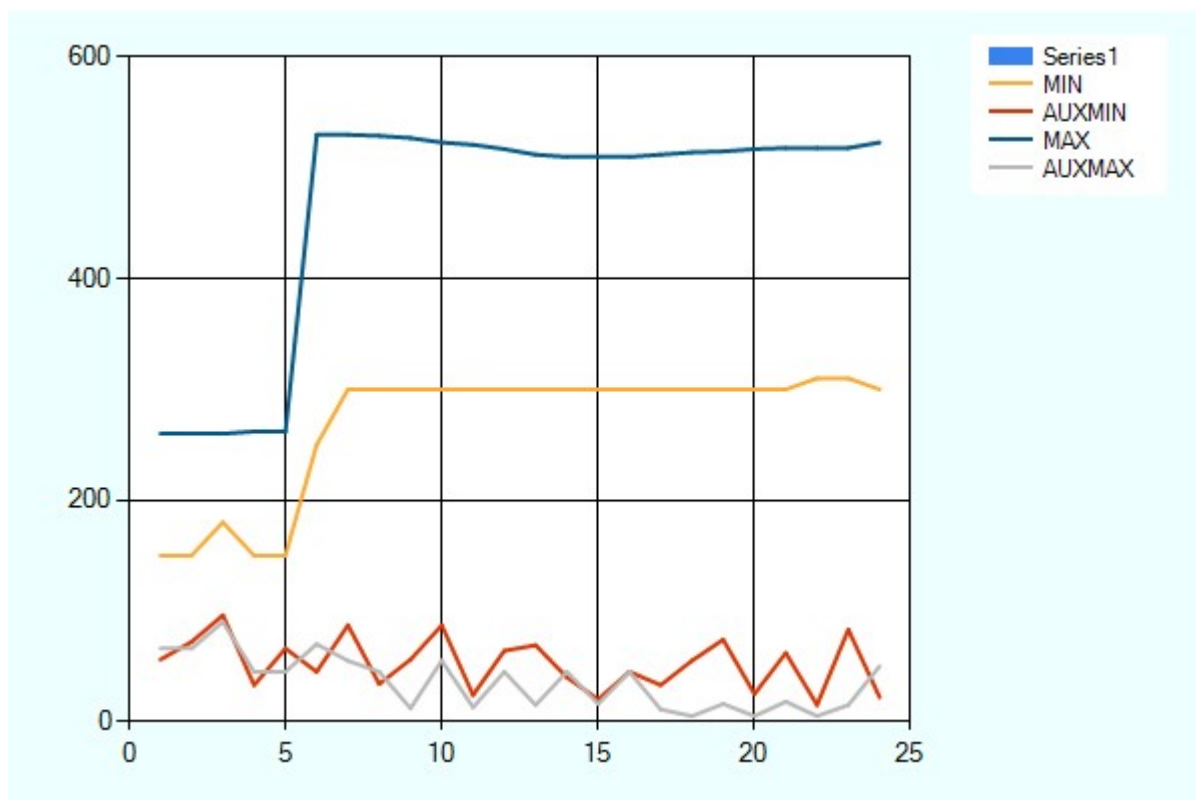
for (int i = 1; i < chart1.Series.Count; i++)
{
    chart1.Series[i].ChartType = SeriesChartType.Line;
    chart1.Series[i].BorderWidth = 2;
}

chart1.Legends.Add("Default");
//设置表
chart1.Width = 600;
chart1.Height = 400;
chart1.BackColor = Color.Azure;

```

}

效果图：



三.数据源——CSV 文件

代码：

view plaincopy to clipboardprint?

```

protected void Page_Load(object sender, EventArgs e)
{
    string file = "DataFile.csv";

    //获取路径
    string path = Server.MapPath(".\\App_Data\\");

    string sql = "select * from " + file;
    string conStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
    path + ";Extended Properties=\\\"Text;HDR=No;FMT=Delimited\\\"";

    OleDbConnection con=new OleDbConnection(conStr);
    con.Open();
    OleDbCommand cmd=new OleDbCommand(sql,con);

    OleDbDataReader read=cmd.ExecuteReader();

    //"1"表示第一列即时间列, "2"表示第二列
    Chart1.Series[0].Points.DataBindXY(read,"1",read,"2");
    Chart1.Series[0].ChartType = SeriesChartType.Line;
    Chart1.Width = 600;
    Chart1.Height = 400;
    Chart1.BackColor = Color.Azure;

    read.Close();
    con.Close();
}

```

```

protected void Page_Load(object sender, EventArgs e)
{
    string file = "DataFile.csv";

    //获取路径
    string path = Server.MapPath(".\\App_Data\\");

    string sql = "select * from " + file;
    string conStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
    path + ";Extended Properties=\\\"Text;HDR=No;FMT=Delimited\\\"";

    OleDbConnection con=new OleDbConnection(conStr);
    con.Open();
    OleDbCommand cmd=new OleDbCommand(sql,con);

    OleDbDataReader read=cmd.ExecuteReader();

    //"1"表示第一列即时间列, "2"表示第二列
    Chart1.Series[0].Points.DataBindXY(read,"1",read,"2");

```

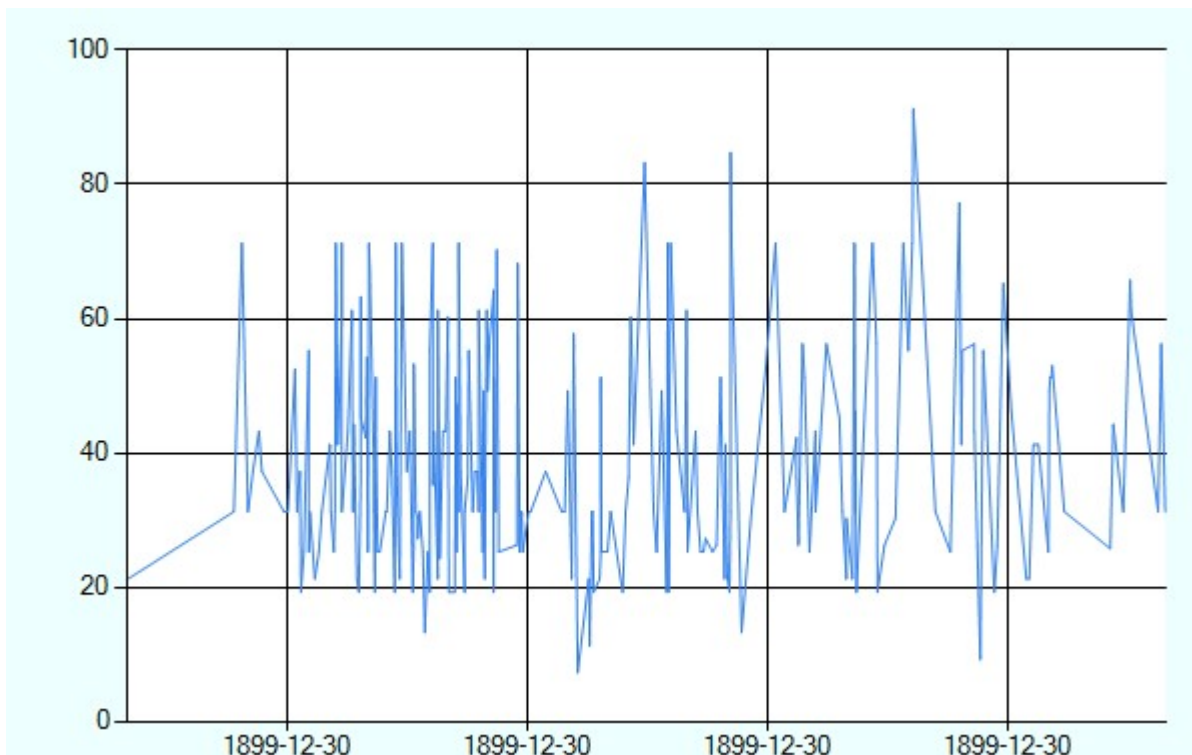
```

Chart1.Series[0].ChartType = SeriesChartType.Line;
Chart1.Width = 600;
Chart1.Height = 400;
Chart1.BackColor = Color.Azure;

read.Close();
con.Close();
}

```

效果图：



继续对 **MSChart** 学习:本次主要针对 **MSChart** 的图形交互和 **AJAX** 的应用。

一. 图形交互:

概述：当点击饼图或 legend 中相应的模块，饼图会发生缺省：

代码如下：

view plaincopy to clipboardprint?

```

protected void Page_Load(object sender, EventArgs e)
{
    Series series= chart1.Series.Add("My series");
    // Set series and legend tooltips
    series.ToolTip = "#VALX: #VAL {C} million";
    series.LegendToolTip = "#PERCENT";
    series.PostBackValue = "#INDEX";
    series.LegendPostBackValue = "#INDEX";
    // Populate series data
}

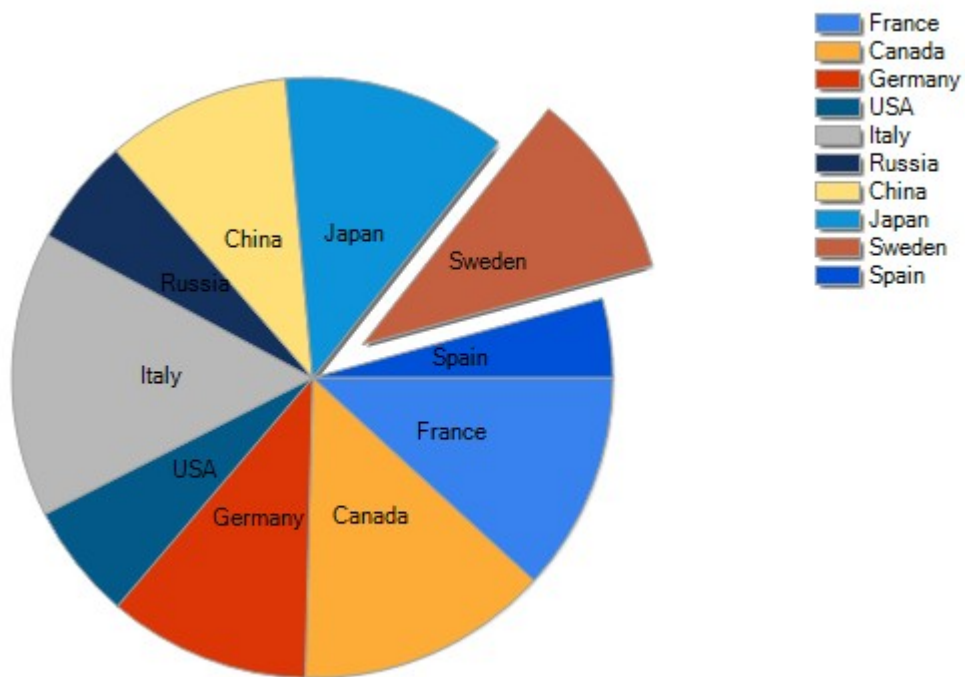
```

```

double[] yValues = { 65.62, 75.54, 60.45, 34.73, 85.42, 32.12, 55.18, 67.15, 56.24, 23.65 };
string[] xValues = { "France", "Canada", "Germany", "USA", "Italy", "Russia", "China", "Japan",
"Sweden", "Spain" };
series.Points.DataBindXY(xValues, yValues);
    // Set series visual attributes
series.ChartType = SeriesChartType.Pie;
series.ShadowOffset = 2;
series.BorderColor = Color.DarkGray;
// series.CustomAttributes = "LabelStyle=Outside";
chart1.Width = 600;
chart1.Height = 400;
}
protected void chart1_Click(object sender, ImageMapEventArgs e)
{
    int pointIndex = int.Parse(e.PostBackValue);
    Series series = chart1.Series["My series"];
    if (pointIndex >= 0 && pointIndex < series.Points.Count)
    {
        series.Points[pointIndex].CustomProperties += "Exploded=true";
    }
}

```

效果图：



二. AJAX(一)——Click Event

概述：当单击图中相应的列柱，标题 2 就会相应的变化，若单击 Chart 中其他地方，会捕捉 Chart 中的相应对象的名称。

先在前台设置 Chart, 添加三个 Series, 分别对其设置 Point, 添加数据, 如下: [view plaincopy to clipboardprint?](#)

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Chart ID="chart1" runat="server" onclick="chart1_Click">
                <Legends>
                    <asp:Legend Enabled="False" Name="Legend1">
                    </asp:Legend>
                </Legends>
                <Titles>
                    <asp:Title Font="Microsoft Sans Serif, 12pt" Name="Title1" Text="单击 Chart">
                    </asp:Title>
                    <asp:Title DockingOffset="-2" Name="Title2" Text="上次单击元素: 空">
                    </asp:Title>
                </Titles>
                <Series>
                    <asp:Series CustomProperties="DrawingStyle=Emboss" Name="Series1"
BorderColor="180, 26, 59, 105"
Color="220, 65, 140, 240" Legend="Legend1">
                        <Points>
                            <asp:DataPoint YValues="6" />
                            <asp:DataPoint YValues="8" />
                            <asp:DataPoint YValues="7" />
                            <asp:DataPoint YValues="7" />
                            <asp:DataPoint YValues="4" />
                        </Points>
                    </asp:Series>
                    <asp:Series CustomProperties="DrawingStyle=Emboss" BorderColor="180, 26, 59, 105"
Color="220, 252, 180, 65" Name="Series2" Legend="Legend1" >
                        <Points>
                            <asp:DataPoint YValues="6" />
                            <asp:DataPoint YValues="6" />
                            <asp:DataPoint YValues="7" />
                            <asp:DataPoint YValues="7" />
                            <asp:DataPoint YValues="8" />
                        </Points>
                    </asp:Series>
                    <asp:Series CustomProperties="DrawingStyle=Emboss" BorderColor="180, 26, 59, 10"
Color="220, 224, 64, 10" Name="Series3" Legend="Legend1">
                        <Points>
                            <asp:DataPoint YValues="6" />
                            <asp:DataPoint YValues="8" />
                            <asp:DataPoint YValues="3" />
                        </Points>
                    </asp:Series>
                </Series>
            </asp:Chart>
        </ContentTemplate>
    </asp:UpdatePanel>
</div>
```

```

                <asp:DataPoint YValues="5" />
                <asp:DataPoint YValues="6" />
            </Points>
        </asp:Series>
    </Series>
    <ChartAreas>
        <asp:ChartArea Name="Default" BorderColor="0, 0, 0, 0"
BackSecondaryColor="0, 0, 0, 0"
                BackColor="0, 0, 0, 0" ShadowColor="0, 0, 0, 0"
BackGradientStyle="TopBottom">
            <AxisY LineColor="64, 64, 64, 64">
                <LabelStyle Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold""></LabelStyle>
                <MajorGrid LineColor="64, 64, 64, 64"></MajorGrid>
            </AxisY>
            <AxisX LineColor="64, 64, 64, 64">
                <LabelStyle Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold""></LabelStyle>
                <MajorGrid LineColor="64, 64, 64, 64"></MajorGrid>
            </AxisX>
        </asp:ChartArea>
    </ChartAreas>
</asp:Chart>
</ContentTemplate>
</asp:UpdatePanel>
</div>
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Chart ID="chart1" runat="server" onclick="chart1_Click">
                <Legends>
                    <asp:Legend Enabled="False" Name="Legend1">
                    </asp:Legend>
                </Legends>
                <Titles>
                    <asp:Title Font="Microsoft Sans Serif, 12pt" Name="Title1" Text="单击 Chart">
                    </asp:Title>
                    <asp:Title DockingOffset="-2" Name="Title2" Text="上次单击元素：空">
                    </asp:Title>
                </Titles>
                <Series>
                    <asp:Series CustomProperties="DrawingStyle=Emboss" Name="Series1"
BorderColor="180, 26, 59, 105"
                    Color="220, 65, 140, 240" Legend="Legend1">

```

```

        <Points>
            <asp:DataPoint YValues="6" />
            <asp:DataPoint YValues="8" />
            <asp:DataPoint YValues="7" />
            <asp:DataPoint YValues="7" />
            <asp:DataPoint YValues="4" />
        </Points>
    </asp:Series>
    <asp:Series CustomProperties="DrawingStyle=Emboss" BorderColor="180, 26, 59, 105"
        Color="220, 252, 180, 65" Name="Series2" Legend="Legend1" >
        <Points>
            <asp:DataPoint YValues="6" />
            <asp:DataPoint YValues="6" />
            <asp:DataPoint YValues="7" />
            <asp:DataPoint YValues="7" />
            <asp:DataPoint YValues="8" />
        </Points>
    </asp:Series>
    <asp:Series CustomProperties="DrawingStyle=Emboss" BorderColor="180, 26, 59, 10"
        Color="220, 224, 64, 10" Name="Series3" Legend="Legend1">
        <Points>
            <asp:DataPoint YValues="6" />
            <asp:DataPoint YValues="8" />
            <asp:DataPoint YValues="3" />
            <asp:DataPoint YValues="5" />
            <asp:DataPoint YValues="6" />
        </Points>
    </asp:Series>
</Series>
<ChartAreas>
    <asp:ChartArea Name="Default" BorderColor="0, 0, 0, 0"
BackSecondaryColor="0, 0, 0, 0"
        BackColor="0, 0, 0, 0" ShadowColor="0, 0, 0, 0"
BackGradientStyle="TopBottom">
        <AxisY LineColor="64, 64, 64, 64">
            <LabelStyle Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"></LabelStyle>
            <MajorGrid LineColor="64, 64, 64, 64"></MajorGrid>
        </AxisY>
        <AxisX LineColor="64, 64, 64, 64">
            <LabelStyle Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"></LabelStyle>
            <MajorGrid LineColor="64, 64, 64, 64"></MajorGrid>
        </AxisX>
    </asp:ChartArea>
</ChartAreas>

```

```

        </asp:Chart>
    </ContentTemplate>
</asp:UpdatePanel>
</div>

```

后台代码:

view plaincopy to clipboardprint?

```

protected void Page_Load(object sender, EventArgs e)
{
    //这个事件在这没必要
    // chart1.Click+=new ImageMapEventHandler(chart1_Click);
    foreach (Series series in chart1.Series)
    {
        series.PostBackValue = "series:" + series.Name + ",#INDEX";
    }
    // transfer of click coordinates. getCoordinates is a javascript function.
    this.chart1.Attributes["onclick"] = ClientScript.GetPostBackEventReference(this.chart1,
"@").Replace("@", "chart:'+_getCoord(event)");
    // set position to relative in order to get proper coordinates.
    this.chart1.Style[HtmlTextWriterStyle.Position] = "relative";
    this.ClientScript.RegisterClientScriptBlock(
        typeof(Chart),
        "Chart",
        @"function _getCoord(event){if(typeof(event.x)=='undefined'){return
event.layerX+' '+event.layerY;}return event.x+' '+event.y;}",
        true);
}

protected void chart1_Click(object sender, ImageMapEventArgs e)
{
    chart1.Titles[1].Text = "Nothing";
    string[] input = e.PostBackValue.Split(':');
    //若长度等于 2
    if (input.Length == 2)
    {
        string[] seriesData = input[1].Split(',');
        //若是 series,则输出相应的列数据
        if (input[0] == "series")
        {
            chart1.Titles[1].Text = "最后单击元素;" + seriesData[0] + " Data Point #" + seriesData[1];
        }
        //否则输出 Chart 中相应的元素
        else if (input[0] == "chart")
        {
            //确定是 Chart 中的哪个元素
            HitTestResult
hitTestResult=chart1.HitTest(Int32.Parse(seriesData[0]),Int32.Parse(seriesData[1]));
            chart1.Titles[1].Text = "最后单击元素;" + hitTestResult.ChartElementType.ToString();
        }
    }
}

protected void Page_Load(object sender, EventArgs e)

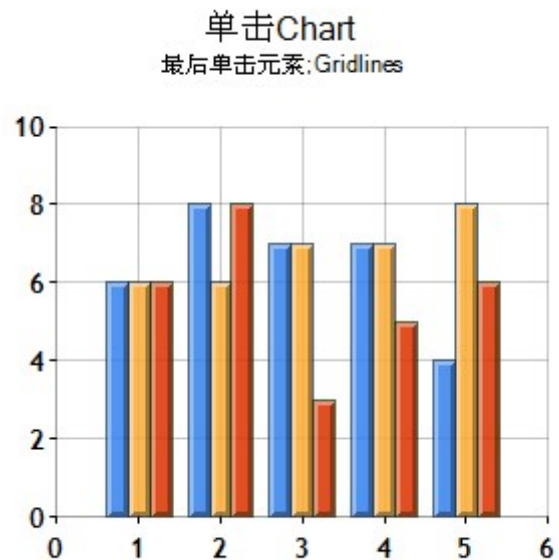
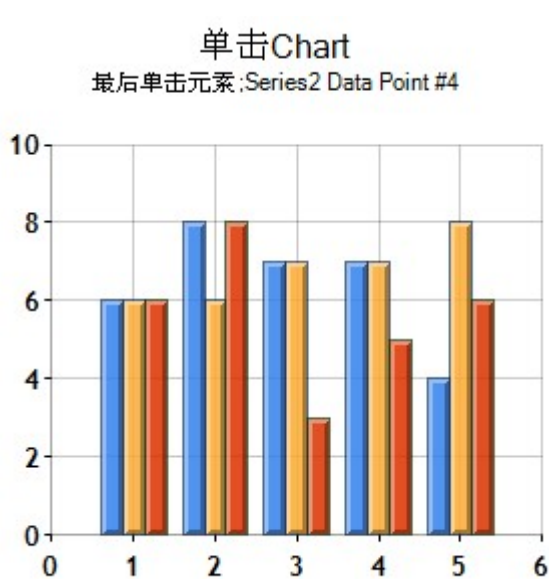
```

```

{
    //这个事件在这没必要
    // chart1.Click+=new ImageMapEventHandler(chart1_Click);
    foreach (Series series in chart1.Series)
    {
        series.PostBackValue = "series:" + series.Name + ",#INDEX";
    }
    // transfer of click coordinates. getCoordinates is a javascript function.
    this.chart1.Attributes["onclick"] = ClientScript.GetPostBackEventReference(this.chart1,
"@").Replace("@", "chart:+'_getCoord(event)");
    // set position to relative in order to get proper coordinates.
    this.chart1.Style[HtmlTextWriterStyle.Position] = "relative";
    this.ClientScript.RegisterClientScriptBlock(
        typeof(Chart),
        "Chart",
        @"function _getCoord(event){if(typeof(event.x)=='undefined'){return
event.layerX+'_'+event.layerY;}return event.x+'_'+event.y;}",
        true);
}
protected void chart1_Click(object sender, ImageMapEventArgs e)
{
    chart1.Titles[1].Text = "Nothing";
    string[] input = e.PostBackValue.Split(':');
    //若长度等于 2
    if (input.Length == 2)
    {
        string[] seriesData = input[1].Split(',');
        //若是 series,则输出相应的列数据
        if (input[0] == "series")
        {
            chart1.Titles[1].Text = "最后单击元素;" + seriesData[0] + " Data Point #" + seriesData[1];
        }
        //否则输出 Chart 中相应的元素
        else if (input[0] == "chart")
        {
            //确定是 Chart 中的哪个元素
            HitTestResult
hitTestResult=chart1.HitTest(Int32.Parse(seriesData[0]),Int32.Parse(seriesData[1]));
            chart1.Titles[1].Text = "最后单击元素;" + hitTestResult.ChartElementType.ToString();
        }
    }
}
}

```

效果图:



三. AJAX(二)——Chart as Trriger

概述：当单击图标上的列柱时，Gridview 相应的发生变化。主要靠 AJAX 的 UpdatePanel 控件的 Trriger 属性实现异步回调。

前台主要代码：

view plaincopy to clipboardprint?

```
<asp:AccessDataSource ID="MarterSource" runat="server"
    DataFile="~/App_Data/chartdata.mdb"
    SelectCommand="SELECT  REGIONS.RegionName, SUM(REPS_SALES.Sales)
AS SumOfSales FROM (REPS_SALES INNER JOIN REGIONS ON REPS_SALES.RegionID =
REGIONS.RegionID) GROUP BY REGIONS.RegionName">
</asp:AccessDataSource>
<asp:Chart ID="Chart1" runat="server" DataSourceID="MarterSource"
onclick="Chart1_Click">
    <Series>
        <asp:Series Name="Series1" XValueMember="RegionName"
YValueMembers="SumOfSales">
        </asp:Series>
    </Series>
    <ChartAreas>
        <asp:ChartArea Name="ChartArea1">
        </asp:ChartArea>
    </ChartAreas>
</asp:Chart>
<asp:AccessDataSource ID="MarterSource" runat="server"
    DataFile="~/App_Data/chartdata.mdb"
    SelectCommand="SELECT  REGIONS.RegionName, SUM(REPS_SALES.Sales)
AS SumOfSales FROM (REPS_SALES INNER JOIN REGIONS ON REPS_SALES.RegionID =
```

```

REGIONS.RegionID) GROUP BY REGIONS.RegionName">
        </asp:AccessDataSource>
        <asp:Chart ID="Chart1" runat="server" DataSourceID="MarterSource"
            onclick="Chart1_Click">
            <Series><asp:Series      Name="Series1"      XValueMember="RegionName"
YValueMembers="SumOfSales">
                </asp:Series>
            </Series>
            <ChartAreas>
                <asp:ChartArea Name="ChartArea1">
                </asp:ChartArea>
            </ChartAreas>
        </asp:Chart>

```

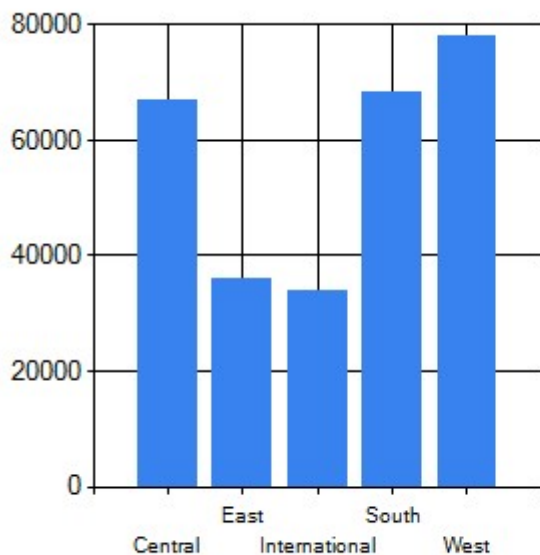
后台代码:

```

view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)
{
    this.Chart1.Series[0].PostBackValue = "#AXISLABEL";
    this.Chart1.Series[0].ToolTip = "#AXISLABEL Region: #VAL{C}";
}
protected void Chart1_Click(object sender, ImageMapEventArgs e)
{
    DetailsSource.SelectParameters["RegionName"].DefaultValue = e.PostBackValue;
    GridView1.Caption = string.Format("{0} Region", e.PostBackValue);
}
protected void Page_Load(object sender, EventArgs e)
{
    this.Chart1.Series[0].PostBackValue = "#AXISLABEL";
    this.Chart1.Series[0].ToolTip = "#AXISLABEL Region: #VAL{C}";
}
protected void Chart1_Click(object sender, ImageMapEventArgs e)
{
    DetailsSource.SelectParameters["RegionName"].DefaultValue = e.PostBackValue;
    GridView1.Caption = string.Format("{0} Region", e.PostBackValue);
}

```

效果图:



Name	Sales
Andrew	23880
Jerry	14991
Hannah	27887

四. AJAX(三)——AJAX 更新

概述：当点击图列时，自动弹出一框框，显示当前图列柱的值，当修改 txtBox 中的值点确定后，图列表相应的发生变化，框框消失。

具体前台代码:(panel 必须放在 UpdatePanel 中，不然 txtBox 中值不会出现)

view plaincopy to clipboardprint?

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Panel runat="server">
            <asp:Chart ID="Chart1" runat="server" BackColor="243, 223, 193"
                BackGradientStyle="TopBottom" BorderlineColor="181, 64, 1"
                BorderlineDashStyle="Solid" onclick="Chart1_Click">
                <BorderSkin SkinStyle="Emboss"></BorderSkin>
                <Series>
                    <asp:Series Name="Series1" BorderColor="192, 192, 255"
                        ChartType="Bar" CustomProperties="DrawingStyle=Cylinder"
                        XValueType="Double" YValueType="Double">
                        <Points>
                            <asp:DataPoint XValue="1" YValues="56" />
                            <asp:DataPoint XValue="2" YValues="23" />
                            <asp:DataPoint XValue="3" YValues="45" />
                            <asp:DataPoint XValue="4" YValues="67" />
                            <asp:DataPoint XValue="5" YValues="34" />
                        </Points>
                    </asp:Series>
                    <asp:Series ChartArea="ChartArea1" ChartType="Bar"
                        CustomProperties="DrawingStyle=Cylinder" Name="Series2" XValueType="Double"
                        YValueType="Double">
```

```

        <Points>
            <asp:DataPoint XValue="1" YValues="87" />
            <asp:DataPoint XValue="2" YValues="54" />
            <asp:DataPoint XValue="3" YValues="32" />
            <asp:DataPoint XValue="4" YValues="68" />
            <asp:DataPoint XValue="5" YValues="21" />
        </Points>
    </asp:Series>
</Series>
    <Titles>
        <asp:Title Name="title1" Text=" 单 击 和 更 新 "></asp:Title>
</Titles>

    <ChartAreas>
    <asp:ChartArea Name="ChartArea1" BackGradientStyle="TopBottom"
        BorderDashStyle="Solid">
        <AxisX LineColor="64, 64, 64, 64">
        </AxisX>
        </asp:ChartArea>
    </ChartAreas>
</asp:Chart>
    <asp:Panel ID="InputPanel" runat="server" Visible="False" style=" position:absolute;top:
40px; left:100px; border:black 1px ridge; padding:3px; background-color:White;">
        <asp:HiddenField ID="hiderField" runat="server" />
        <asp:TextBox ID="txtValue" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button"
            Width="85px" />
    </asp:Panel>
</asp:Panel>
    </ContentTemplate>
</asp:UpdatePanel>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:Panel runat="server">
            <asp:Chart ID="Chart1" runat="server" BackColor="243, 223, 193"
BackGradientStyle="TopBottom" BorderlineColor="181, 64, 1"
BorderlineDashStyle="Solid" onclick="Chart1_Click">
                <BorderSkin SkinStyle="Emboss"></BorderSkin>
                <Series>
                    <asp:Series Name="Series1" BorderColor="192, 192, 255"
ChartType="Bar" CustomProperties="DrawingStyle=Cylinder"
                        XValueType="Double" YValueType="Double">
                        <Points>
                            <asp:DataPoint XValue="1" YValues="56" />
                            <asp:DataPoint XValue="2" YValues="23" />
                            <asp:DataPoint XValue="3" YValues="45" />
                            <asp:DataPoint XValue="4" YValues="67" />

```

```

        <asp:DataPoint XValue="5" YValues="34" />
    </Points>
</asp:Series>
<asp:Series ChartArea="ChartArea1" ChartType="Bar"
    CustomProperties="DrawingStyle=Cylinder"      Name="Series2"
XValueType="Double"
    YValueType="Double">
    <Points>
        <asp:DataPoint XValue="1" YValues="87" />
        <asp:DataPoint XValue="2" YValues="54" />
        <asp:DataPoint XValue="3" YValues="32" />
        <asp:DataPoint XValue="4" YValues="68" />
        <asp:DataPoint XValue="5" YValues="21" />
    </Points>
</asp:Series>
</Series>
<Titles>
    <asp:Title Name="title1" Text="单击和更新"></asp:Title>
</Titles>
<ChartAreas>
    <asp:ChartArea Name="ChartArea1" BackGradientStyle="TopBottom"
        BorderDashStyle="Solid">
        <AxisX LineColor="64, 64, 64, 64">
        </AxisX>
    </asp:ChartArea>
</ChartAreas>
</asp:Chart>
<asp:Panel ID="InputPanel" runat="server" Visible="False" style=" position:absolute;top:
40px; left:100px; border:black 1px ridge; padding:3px; background-color:White;">
    <asp:HiddenField ID="hidderField" runat="server" />
    <asp:TextBox ID="txtValue" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button"
        Width="85px" />
</asp:Panel>
</asp:Panel>
</ContentTemplate>
</asp:UpdatePanel>

```

后台代码:

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```

{
    //对每个 Series 赋值
    foreach (Series series in Chart1.Series)
    {
        series.PostBackValue = "#SERIESNAME;#INDEX";
    }
}

```

```

}
protected void Chart1_Click(object sender, ImageMapEventArgs e)
{
    string[] pointDate = e.PostBackValue.Split(';');
    //转化为相应的 Series
    Series series = Chart1.Series[pointDate[0]];
    //转化为相应的 DataPoint
    DataPoint dataPoint = series.Points[Int32.Parse(pointDate[1])];
    //给 textBox 和 Hiddfield 赋值
    hiddfield.Value = e.PostBackValue;
    txtValue.Text = dataPoint.YValues[0].ToString();
    //显示 Panel
    InputPanel.Visible = true;
    // PointValue.Focus();
}
protected void Button1_Click(object sender, EventArgs e)
{
    string[] pointData = hiddfield.Value.Split(';');
    //转化为相应的 Series
    Series series = Chart1.Series[pointData[0]];
    //转化为相应的 DataPoint
    DataPoint dataPoint = series.Points[Int32.Parse(pointData[1])];
    //给 textBox 和 Hiddfield 赋值
    dataPoint.YValues[0] = Double.Parse(txtValue.Text);
    //隐藏 Panel
    InputPanel.Visible = false;
}
protected void Page_Load(object sender, EventArgs e)
{
    //对每个 Series 赋值
    foreach (Series series in Chart1.Series)
    {
        series.PostBackValue = "#SERIESNAME;#INDEX";
    }
}
protected void Chart1_Click(object sender, ImageMapEventArgs e)
{
    string[] pointDate = e.PostBackValue.Split(';');
    //转化为相应的 Series
    Series series = Chart1.Series[pointDate[0]];
    //转化为相应的 DataPoint
    DataPoint dataPoint = series.Points[Int32.Parse(pointDate[1])];
    //给 textBox 和 Hiddfield 赋值
    hiddfield.Value = e.PostBackValue;
    txtValue.Text = dataPoint.YValues[0].ToString();
    //显示 Panel

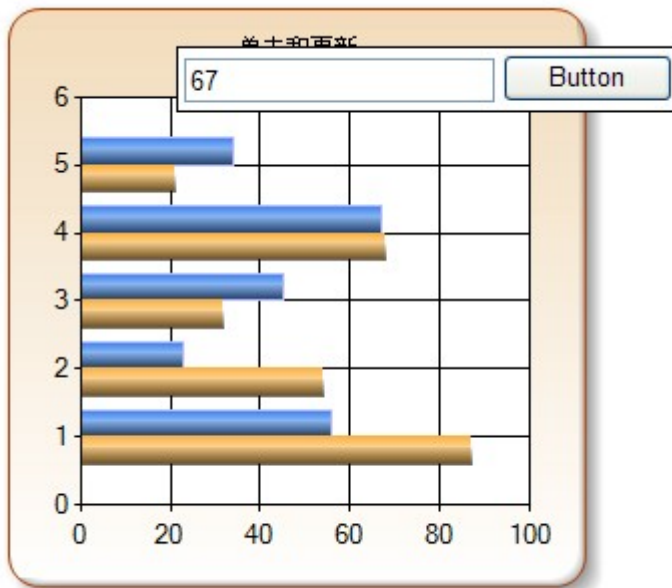
```

```

        InputPanel.Visible = true;
        // PointValue.Focus();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string[] pointData = hidderField.Value.Split(';');
        //转化为相应的 Series
        Series series = Chart1.Series[pointData[0]];
        //转化为相应的 DataPoint
        DataPoint dataPoint = series.Points[Int32.Parse(pointData[1])];
        //给 txtBox 和 Hidderfield 赋值
        dataPoint.YValues[0] = Double.Parse(txtValue.Text);
        //隐藏 Panel
        InputPanel.Visible = false;
    }
}

```

效果图:



五. AJAX(四)——Real Time Chart

概述: 每过一秒钟, Chart 就增加五个点, 动态的生成一条曲线。

代码: (前台代码一定要设 EnableViewState="true", 否则一直就是五个点)

view plaincopy to clipboardprint?

```

protected void Timer1_Tick(object sender, EventArgs e)
{
    Random random = new Random();

    foreach (Series series in Chart1.Series)

```

```

{
    double xLastValue = series.Points[series.Points.Count - 1].XValue+1;
    double yLastValue = series.Points[series.Points.Count - 1].YValues[0];

    for (int i = 0; i < 5; i++)
    {
        yLastValue += random.Next(-3, 4);
        if (yLastValue >= 100.0)
        {
            yLastValue -= 25.0;
        }
        if (yLastValue <= 10.0)
        {
            yLastValue += 25.0;
        }
        series.Points.AddXY(xLastValue++,yLastValue);
    }
}

while (Chart1.Series[0].Points.Count > 100)
{
    foreach (Series series in Chart1.Series)
    {
        series.Points.RemoveAt(0);
    }
}

double axisMinimum = this.Chart1.Series[0].Points[0].XValue;
this.Chart1.ChartAreas[0].AxisX.Minimum = axisMinimum;
this.Chart1.ChartAreas[0].AxisX.Maximum = axisMinimum + 100;

}
protected void Timer1_Tick(object sender, EventArgs e)
{
    Random random = new Random();

    foreach (Series series in Chart1.Series)
    {
        double xLastValue = series.Points[series.Points.Count - 1].XValue+1;
        double yLastValue = series.Points[series.Points.Count - 1].YValues[0];

        for (int i = 0; i < 5; i++)
        {
            yLastValue += random.Next(-3, 4);

```

```

        if (yLastValue >= 100.0)
        {
            yLastValue -= 25.0;
        }
        if (yLastValue <= 10.0)
        {
            yLastValue += 25.0;
        }
        series.Points.AddXY(xLastValue++,yLastValue);
    }
}

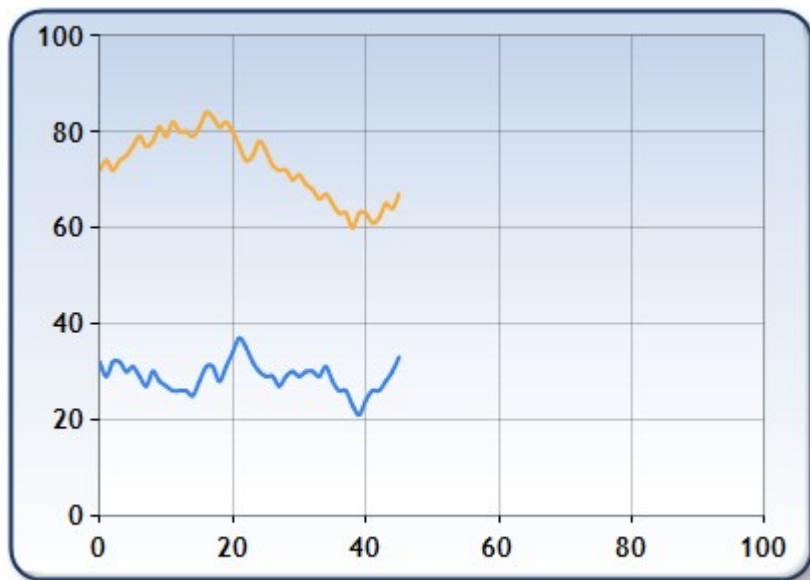
while (Chart1.Series[0].Points.Count > 100)
{
    foreach (Series series in Chart1.Series)
    {
        series.Points.RemoveAt(0);
    }
}

double axisMinimum = this.Chart1.Series[0].Points[0].XValue;
this.Chart1.ChartAreas[0].AxisX.Minimum = axisMinimum;
this.Chart1.ChartAreas[0].AxisX.Maximum = axisMinimum + 100;

}

```

效果图：



解决 MSChart 底部横坐标显示不全的问题 收藏

还是这个 MSChart，先前做一个 test 的时候都正常，正式用的时候出现这样一个问题，绑定数据之后，如果 x 轴坐标过多，会显示不全，我这里的情况是底部坐标一般最多显示八九个，再多了就隔一个显示一个，或者显示的更少。

开始的时候一直以为是绑定数据的问题，一步步的断点看，都没有问题。后来网上搜到这个页面 <http://www.cnblogs.com/tristinjet/archive/2009/11/09/1598981.html>

底下的留言里也有人问到了这个问题，联系了 blog 作者，发现只需要设置以下属性就可以了：

```
view plaincopy to clipboardprint?
```

```
//需要设置横坐标的间隔和偏移都设为 1 不然坐标多了会显示不全
```

```
Chart2.ChartAreas[0].AxisX.Interval = 1;
```

```
Chart2.ChartAreas[0].AxisX.IntervalOffset = 1;
```

```
Chart2.ChartAreas[0].AxisX.LabelStyle.IsStaggered = true;
```

```
//需要设置横坐标的间隔和偏移都设为 1 不然坐标多了会显示不全
```

```
Chart2.ChartAreas[0].AxisX.Interval = 1;
```

```
Chart2.ChartAreas[0].AxisX.IntervalOffset = 1;
```

```
Chart2.ChartAreas[0].AxisX.LabelStyle.IsStaggered = true;
```

一个是横坐标间隔，一个是偏移，IsStaggered 是设置是否交错显示，比如横坐标多的时候可以分成两行来显示。

花了我将近一天时间调试，原来这么简单。

mschart 上下限阈值 画横线 收藏

开发中，发现图表中，显示上下限很重要，于是想办法。mschart，开始想增加一个 Series，然后 Y 值 设置为固定值。发现效果： 有数据区域有上下限线，无法从纵轴开始。

后来 google，找到这种方法：

```
view plaincopy to clipboardprint?
```

```
StripLine sl1 = new StripLine();
```

```
sl1.BackColor = System.Drawing.Color.Red;
```

```
sl1.IntervalOffset = 1;
```

```
sl1.StripWidth = 0.001;
```

```
sl1.Text = "上限";
```

```
sl1.TextAlignment = StringAlignment.Near;
```

```
ChartLine1.ChartAreas[0].AxisY.StripLines.Add(sl1);
```

```
StripLine sl1 = new StripLine();
```

```
sl1.BackColor = System.Drawing.Color.Red;
```

```
sl1.IntervalOffset = 1;
```

```
sl1.StripWidth = 0.001;
```

```
sl1.Text = "上限";
```

```
sl1.TextAlignment = StringAlignment.Near;
```

```
ChartLine1.ChartAreas[0].AxisY.StripLines.Add(sl1);
```

其中 StripWidth 为线的宽度，我的 Y 轴值比较小，所以设为 0.001
设大值时：

IntervalOffset 为距离 Y 轴原点的距离。

X 轴方向类似！

MSChart for vs2008 收藏

最近在做统计图标，使用 mschart for vs2008，用 asp.net 开发，感觉不错

MSChart 下载

<http://download.microsoft.com/download/c/c/4/cc4dcac6-ea60-4868-a8e0-62a8510aa747/MSChart.exe>

如果要在 VS2008IDE 环境中使用，需要安装插件支持

MSChart_VisualStudioAddOn.exe

http://download.microsoft.com/download/e/6/f/e6fedd87-9119-4037-8da8-e5de429d940a/MSChart_VisualStudioAddOn.exe

装完这两个东西以后，会在工具箱 Data 栏出现一个 Chart 图标，可以使用了。

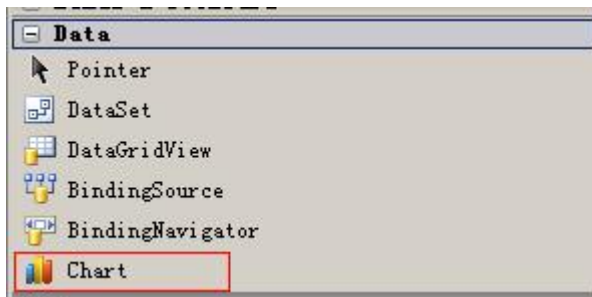
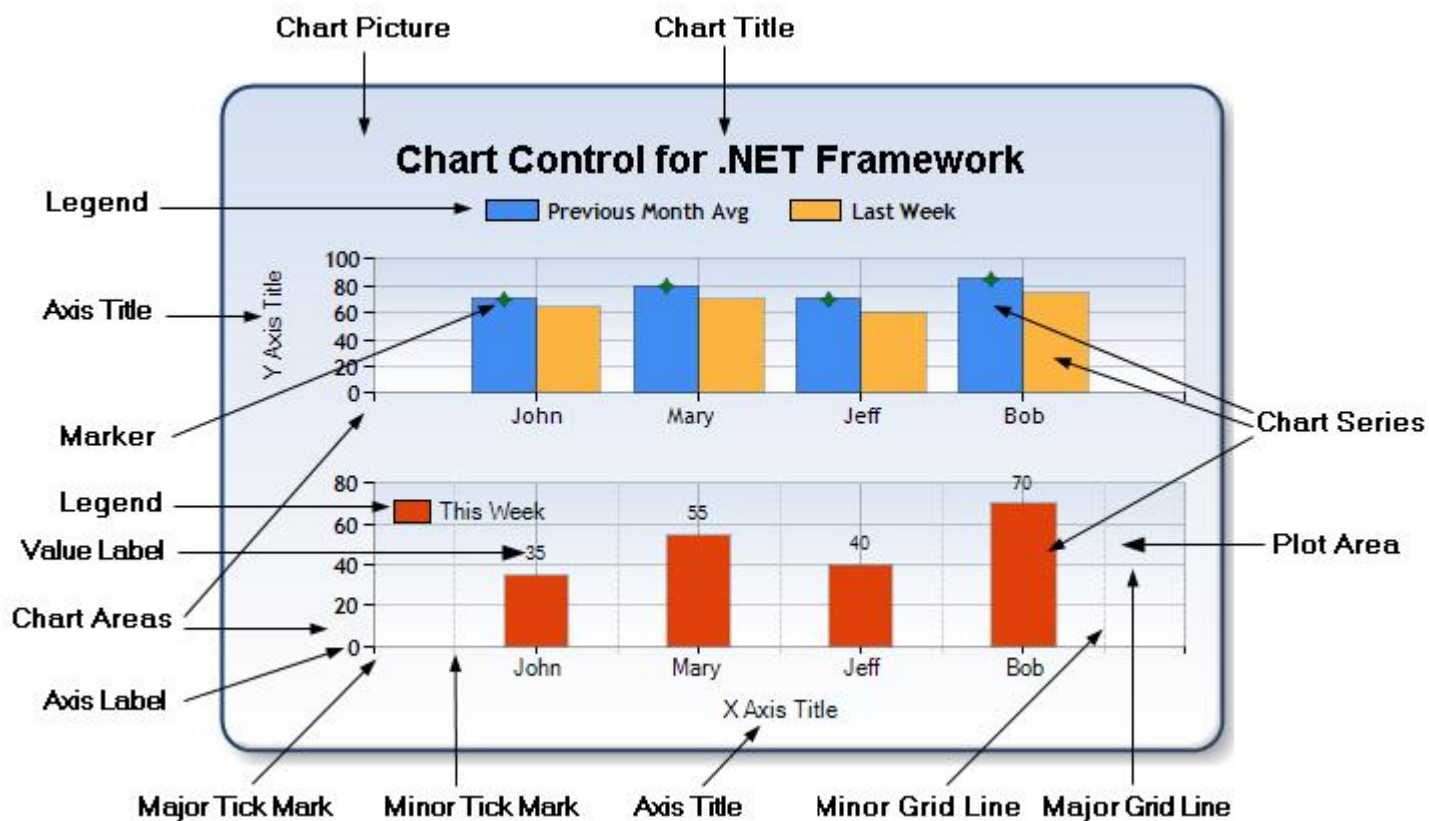


Chart 组件可以与 Excel 图标相媲美

下图是 Chart 组件一些关键元素组成：



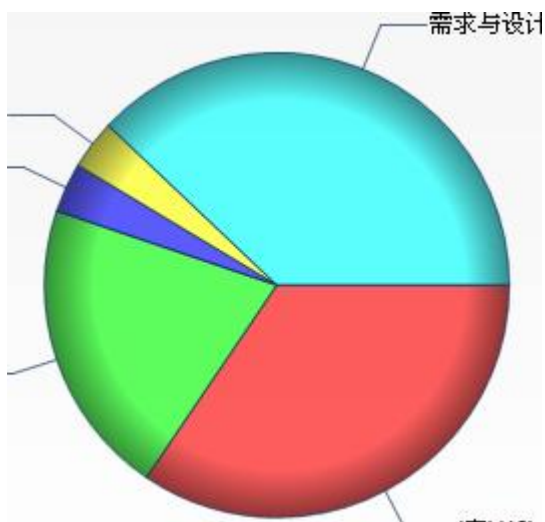
更多内容可以查看 MS 的 DEMO

DataVisChartControl.exe

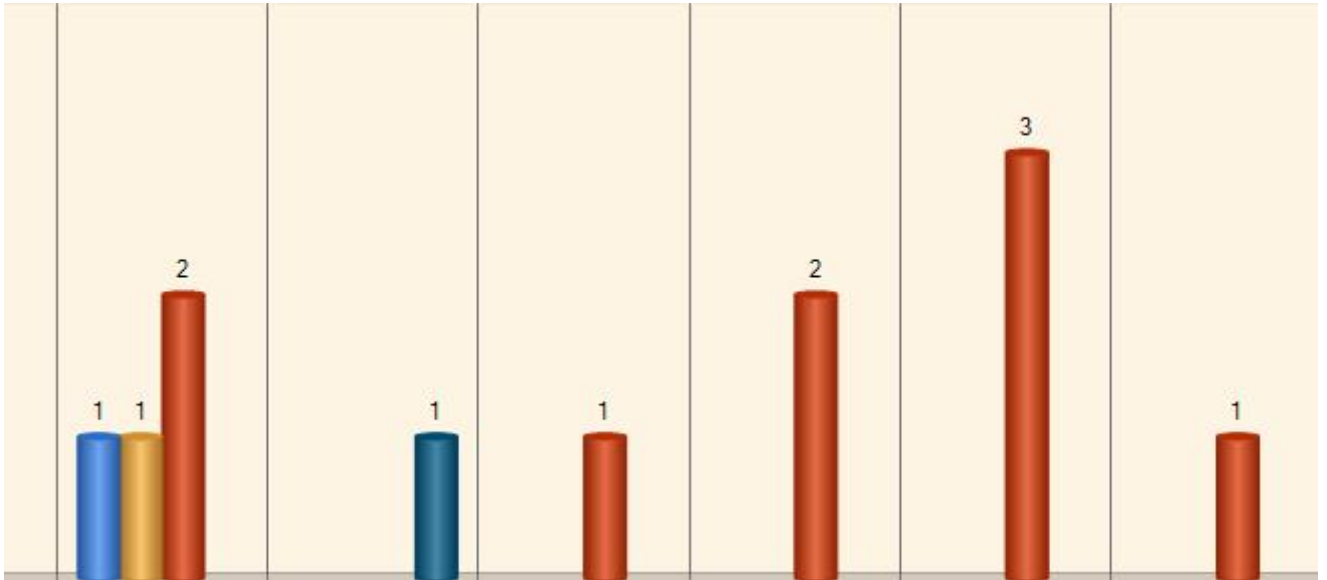
<http://download.microsoft.com/download/5/4/a/54ad5fde-dd31-4a16-82db-72885da83a1f/DataVisChartControl.exe>

这是这两天做的一些截图

饼图，比较好实现



条状图，花了些时间，主要是 X 轴有多个统计项，每个统计项又有好几个子项，还要把 X 轴的统计项用竖线隔开



在 MSChart 使用中可能碰到一些错误

1、“XXXTempImage 目录无效”，修改 web.config 中的

“<add key="ChartImageHandler" value="storage=file;timeout=20;dir=" 后面的内容即可。

2、“msc_cntr_0.txt”不可写，原因是临时目录中存在了这个文件，删除掉就 OK 了

3、“XXX ChartImg.axd XXX” 错误，也修改 web.config，添加一个"POST"即可

"<add path="ChartImg.axd" verb="GET,HEAD,POST""

MSCHART 全部属性方法介绍 收藏

```
void CMoreDMcode::OnCacluteDelta(int nRow)
{
    m_Chart.SetRedraw(false);
    m_Chart.SetRow(1);
    for(int i=1;i<=64;i++)
    {
        m_Chart.GetDataGrid().SetData(i,1,nDelta[i-1],0);
    }
    CRgn rgn;
    m_Chart.GetUpdateRgn(&rgn);

    m_Chart.SetRedraw(true);
    m_Chart.InvalidateRgn(&rgn,false);
}
```

```
void CMoreDMcode::InitChart()
```

```

{
VARIANT var;

m_Chart.SetColumnCount(1); //设置曲线条数
m_Chart.SetRowCount(64); //一条曲线有 64 个点
m_Chart.SetTitleText("二位形态码△t 变化趋势");
m_Chart.SetChartType(1|2); //
//m_Chart.SetShowLegend(TRUE);

//设置 X 轴
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetAuto(FALSE); // 不自动标注 X 轴刻度
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerLabel(8); // 每四刻度一个标注
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerTick(8); // 每刻度一个刻度线

//m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线
m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMaximum(64); // X 轴最大刻度
m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMinimum(0); // X 轴最小刻度
m_Chart.GetPlot().GetAxis(0,var).GetAxisTitle().SetText("二位形态码");

//设置 Y 轴
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetAuto(FALSE); // 不自动标注 Y 轴刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(100); // Y 轴最大刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinimum(0); // Y 轴最小刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMajorDivision(10); // Y 轴刻度 10 等分
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线
//m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().SetText("△t 变化趋势"); // Y 轴名称
//m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().GetTextLayout().SetOrientation(2); // Y 轴名称排列方式

m_Chart.GetPlot().SetUniformAxis(FALSE);

}

void CMoreDMcode::OnMSChartDraw()
{

CString str;
int nRow = m_Grid.GetFocusCell().row;
if (nRow<0)
{
MessageBox("请在左表选择要排序的行！","操作错误",MB_OK|MB_ICONWARNING);
return;
}

for(int i=0;i<64;i++)

```

```

{
str = m_Grid.GetItemText(nRow,i+3);
nDelta = atoi(str.GetBuffer(0));
}

InitChart();//初始化坐标轴

// 根据不同的数据设定不同的 Y 轴最大刻度
int m = GetMaxData(nDelta);
m=(m/50)*50+50;//取靠近 M 的比 M 大的 50 的倍数
str = m_Grid.GetItemText(nRow,0);
VARIANT var;
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(m);

OnCacluteDelta(nRow);//绘图
str = "当前绘制的是第"+str+"期的 $\Delta t$  变化趋势";
m_Tip.SetWindowText(str);
}

```

函数名字取得不好，大家见笑了，开始取名字错误，后来不想该了，呵呵^_^

=====

图表由于其直观明了的特性，在实际应用中十分很广泛。我们常常希望数据能通过图表来显示其特性。例如在 Delphi 和 C++Builder 编程中，我们可以很方便地实现数据图表。MsChart（6.0 或 5.0 版）是 Windows 系统中 Visual studio 自带的一个 ACTIVE X 控件，它功能强大，应用广泛，具有以下特点：

- 支持随机数据和随机数组，动态显示。
- 支持所有主要的图表类型。
- 支持三维显示。

MsChart 具有 45 个属性，9 个方法，49 事件，可灵活编程，可实现各类表的显示。

1 MsChart 应用编程

首先插入 MsChart 控件，

在工程中加入 mschart

菜单->Project->Add To Project->Components and Controls->Registered ActiveX Controls->Microsoft Chart Control, version 6.0 (OLEDB)

1.1 坐标系属性的设置

a) 纵轴初始化属性

Mschart 默认支持自动标准，将自动调整，可以缺省设置。

```
#include "mschart.h" //添加相关的头文件
#include "VcAxis.h"
#include "VcAxisTitle.h"
#include "VcTextLayout.h"
#include "VcDataGrid.h"
#include "VcPlot.h"
#include "VcValueScale.h"
#include "VcSeriesCollection.h"
#include "VcSeries.h"
#include "VcPen.h"
#include "VcCategoryScale.h"
#include "VcColor.h"
#include "VcDataGrid.h"
#include "VcBackdrop.h"
#include "VcFill.h"
#include "VcBrush.h"
#include "VcDataPoints.h"
#include "VcDataPoint.h"
#include "VcDataPointLabel.h"
#include "VcAxisTitle.h"
//最好添加全部头函数，
CMSChart m_Chart;//m_Chart 为图表变量
```

```
VARIANT var;
```

```
m_Chart.GetPlot().GetAxis(1,var)//获取纵轴
```

```
//设置是否支持自动标准;控件默认支持自动标准。
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetAuto(FALSE);
```

```
//设置最大刻度为 M;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMaximum(M);
```

```
//设置最小刻度为 m;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMinimum(m);
```

```
//设置轴的等分数 D;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMajorDivision(D);
```

```
//设置每等分的刻度线数 n;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMinorDivision(n);
```

b) 横轴初始化属性

```
VARIANT var;
```

```
m_Chart.GetPlot().GetAxis(0,var)//获取横轴
```

其他属性设置跟纵轴相同。

1.2 数据显示

a) 设置标题栏和标签

```
m_Chart.SetTitleText(“标题”);//设置标题栏
```

```
m_Chart.SetRowLabel((“第 I 行”));//设置第 i 行标签
```

```
m_Chart.SetColumnLabel((“第 j 列”));//设置第 j 列标签
```

b) 行列的显示布局

MSChart 的行列显示布局有其自身的特点：下面显示是一个行列 4×3（矩形图），即（四行，三列）的布局示意图。

```
m_Chart.SetRowCount(4); //没条曲线三个四个点 （曲线图）
```

```
m_Chart.SetColumnCount(3); //设置曲线条数为三条（曲线图）
```

c) 行列操作

```
// 操作行列第 i 行、第 j 列
```

```
m_Chart.SetRow(i);// 第 i 行
```

```
m_Chart.SetColumn(j);//第 j 行
```

```
m_Chart.SetRowLabel((“第 i 行”));//设置第 i 行标签
```

```
CString str=“ 90.5” ;
```

```
m_Chart.SetData((LPCTSTR(str)); //设置行列，j>的显示数据
```

```
m_Chart.Refresh();//刷新视图
```

d) 显示方式

获取当前的显示方式:

```
long nType =m_Chart.GetChartType();
```

设置显示方式:

```
m_Chart.SetChartType(0);//3D(三维) 显示
```

```
m_Chart.SetChartType(1);//2D(二维) 显示
```

```
m_Chart.Refresh();
```

其它常用组合方式为:

```
m_Chart.SetChartType(1|0) //2D 柱(条)形,
```

```
m_Chart.SetChartType(0|0) //3D 柱(条)形
```

```
m_Chart.SetChartType(1|2) //2D 线条型
```

```
m_Chart.SetChartType(0|2) //3D 线条型
```

```
m_Chart.SetChartType(1|4) //2D 区域型
```

```
m_Chart.SetChartType(0|4) //3D 区域型
```

```
m_Chart.SetChartType(1|6) //2D 阶梯型
```

```
m_Chart.SetChartType(0|6) //3D 阶梯型
```

```
m_Chart.SetChartType(1|8) //2D 复(混)合型
```

```
m_Chart.SetChartType(0|8) //3D 复(混)合型
```

另外,在 2D 方式中,还有两类: 饼型和 XY 型

```
m_Chart.SetChartType(14) //2D 饼型
```

```
m_Chart.SetChartType(16) //2DXY 型
```

e) 其他

其他属性, 比如设置字体, 颜色, 对齐方式等。

```
//=====
```

```
//VC 知识库上的参考文章
```

```
http://www.vckbase.com/document/viewdoc/?id=959
```

3.3 设置 m_Chart void CDemoView::InitChart()

```
{  
    // 设置标题  
    m_Chart.SetTitleText("mschart 示例 by thinkry@263.net");  
    // 下面两句改变背景色  
    m_Chart.GetBackdrop().GetFill().SetStyle(1);  
    m_Chart.GetBackdrop().GetFill().GetBrush().GetFillColor().Set(255, 255, 255);  
    // 显示图例  
    m_Chart.SetShowLegend(TRUE);  
    m_Chart.SetColumn(1);  
    m_Chart.SetColumnLabel((LPCTSTR)"1 号机");  
    m_Chart.SetColumn(2);  
    m_Chart.SetColumnLabel((LPCTSTR)"2 号机");  
    m_Chart.SetColumn(3);  
    m_Chart.SetColumnLabel((LPCTSTR)"3 号机");  
    // 栈模式  
    // m_Chart.SetStacking(TRUE);  
    // Y 轴设置  
    VARIANT var;  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetAuto(FALSE); // 不自动标注 Y 轴刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(100); // Y 轴最大刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinimum(0); // Y 轴最小刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMajorDivision(5); // Y 轴刻度 5 等分  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线  
    m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().SetText("小时"; // Y 轴名称  
    // 3 条曲线  
    m_Chart.SetColumnCount(3);  
    // 线色  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetPen().GetVtColor().Set(0, 0, 255);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetPen().GetVtColor().Set(255, 0, 0);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetPen().GetVtColor().Set(0, 255, 0);  
    // 线宽(对点线图有效)  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetPen().SetWidth(50);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetPen().SetWidth(100);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetPen().SetWidth(2);  
}
```

```

// 数据点类型显示数据值的模式(对柱柱状图和点线图有效)
// 0: 不显示 1: 显示在柱状图外
// 2: 显示在柱状图内上方 3: 显示在柱状图内中间 4: 显示在柱状图内下方
m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
}
3.4 设置数据 void CDemoView:rawChart()
{
int nRowCount = 6;
m_Chart.SetRowCount(nRowCount);
VARIANT var;

// 不自动标注 X 轴刻度
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetAuto(FALSE);

// 每刻度一个标注
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerLabel(1);

// 每刻度一个刻度线
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerTick(1);

// X 轴名称
m_Chart.GetPlot().GetAxis(0,var).GetAxisTitle().SetText("日期");
char buf[32];
srand( (unsigned)time( NULL ) );
for(int row = 1; row <= nRowCount; ++row)
{
m_Chart.SetRow(row);
sprintf(buf, "%d 号", row);
m_Chart.SetRowLabel((LPCTSTR)buf);
m_Chart.GetDataGrid().SetData(row, 1, rand() * 100 / RAND_MAX, 0);//设置第一条曲线的第 row 个点
m_Chart.GetDataGrid().SetData(row, 2, rand() * 100 / RAND_MAX, 0);//设置第二条曲线的第 row 个点
m_Chart.GetDataGrid().SetData(row, 3, rand() * 100 / RAND_MAX, 0);//设置第三条曲线的第 row 个点
}
m_Chart.Refresh();
}

/*
SetData (row, column, dataPoint, nullFlag)

```

SetData 方法语法包括以下组成部分:

row Integer 类型。标识包含数据点值的行。
column Integer 类型。标识包含数据点值的列。
dataPoint Double 类型。数据点值。
nullFlag Integer 类型。指示数据点值是否为空
*/

3.5 改变显示类型 // 折线图

```
void CDemoView::OnChartLine()
{
    m_Chart.SetChartType(3);
    DrawChart();
}
```

// 柱状图

```
void CDemoView::OnChartCombi()
{
    m_Chart.SetChartType(1);
    DrawChart();
}
```

// 饼状图

```
void CDemoView::OnChartPie()
{
    m_Chart.SetChartType(14);
    DrawChart();
}
```

MSCHART 全部属性方法介绍 收藏

```
void CMoreDMcode::OnCacluteDelta(int nRow)
{
    m_Chart.SetRedraw(false);
    m_Chart.SetRow(1);
    for(int i=1;i<=64;i++)
    {
        m_Chart.GetDataGrid().SetData(i,1,nDelta[i-1],0);
    }
    CRgn rgn;
    m_Chart.GetUpdateRgn(&rgn);

    m_Chart.SetRedraw(true);
    m_Chart.InvalidateRgn(&rgn,false);
}

void CMoreDMcode::InitChart()
{

```

```

VARIANT var;

m_Chart.SetColumnCount(1); //设置曲线条数
m_Chart.SetRowCount(64); //一条曲线有 64 个点
m_Chart.SetTitleText("二位形态码△t 变化趋势");
m_Chart.SetChartType(1|2); //
//m_Chart.SetShowLegend(TRUE);

//设置 X 轴
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetAuto(FALSE); // 不自动标注 X 轴刻度
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerLabel(8); // 每四刻度一个标注
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerTick(8); // 每刻度一个刻度线

//m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线
m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMaximum(64); // X 轴最大刻度
m_Chart.GetPlot().GetAxis(0,var).GetValueScale().SetMinimum(0); // X 轴最小刻度
m_Chart.GetPlot().GetAxis(0,var).GetAxisTitle().SetText("二位形态码");

//设置 Y 轴
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetAuto(FALSE); // 不自动标注 Y 轴刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(100); // Y 轴最大刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinimum(0); // Y 轴最小刻度
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMajorDivision(10); // Y 轴刻度 10 等分
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线
//m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().SetText("△t 变化趋势"); // Y 轴名称
//m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().GetTextLayout().SetOrientation(2); //Y 轴名称排列方式

m_Chart.GetPlot().SetUniformAxis(FALSE);

}

void CMoreDMcode::OnMSChartDraw()
{

CString str;
int nRow = m_Grid.GetFocusCell().row;
if (nRow<0)
{
MessageBox("请在左表选择要排序的行！","操作错误",MB_OK|MB_ICONWARNING);
return;
}

for(int i=0;i<64;i++)
{

```

```

str = m_Grid.GetItemText(nRow,i+3);
nDelta = atoi(str.GetBuffer(0));
}

InitChart();//初始化坐标轴

// 根据不同的数据设定不同的 Y 轴最大刻度
int m = GetMaxData(nDelta);
m=(m/50)*50+50;//取靠近 M 的比 M 大的 50 的倍数
str = m_Grid.GetItemText(nRow,0);
VARIANT var;
m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(m);

OnCacluteDelta(nRow);//绘图
str = "当前绘制的是第"+str+"期的△t 变化趋势";
m_Tip.SetWindowText(str);
}

```

函数名字取得不好，大家见笑了，开始取名字错误，后来不想该了，呵呵^_^

=====

图表由于其直观明了的特性，在实际应用中十分很广泛。我们常常希望数据能通过图表来显示其特性。例如在 Delphi 和 C++Builder 编程中，我们可以很方便地实现数据图表。MsChart（6.0 或 5.0 版）是 Windows 系统中 Visual studio 自带的一个 ACTIVEX 控件，它功能强大，应用广泛，具有以下特点：

- 支持随机数据和随机数组，动态显示。
- 支持所有主要的图表类型。
- 支持三维显示。

MsChart 具有 45 个属性，9 个方法，49 事件，可灵活编程，可实现各类表的显示。

1 MsChart 应用编程

首先插入 MsChart 控件，

在工程中加入 mschart

菜单->Project->Add To Project->Components and Controls->Registered ActiveX Controls->Microsoft Chart Control, version 6.0 (OLEDB)

1.1 坐标系属性的设置

a) 纵轴初始化属性

Mschart 默认支持自动标准，将自动调整，可以缺省设置。

```
#include "mschart.h" //添加相关的头文件
```

```
#include "VcAxis.h"
#include "VcAxisTitle.h"
#include "VcTextLayout.h"
#include "VcDataGrid.h"
#include "VcPlot.h"
#include "VcValueScale.h"
#include "VcSeriesCollection.h"
#include "VcSeries.h"
#include "VcPen.h"
#include "VcCategoryScale.h"
#include "VcColor.h"
#include "VcDataGrid.h"
#include "VcBackdrop.h"
#include "VcFill.h"
#include "VcBrush.h"
#include "VcDataPoints.h"
#include "VcDataPoint.h"
#include "VcDataPointLabel.h"
#include "VcAxisTitle.h"
//最好添加全部头函数,
CMSChart m_Chart;//m_Chart 为图表变量
```

```
VARIANT var;
```

```
m_Chart.GetPlot().GetAxis(1,var)//获取纵轴
```

```
//设置是否支持自动标准;控件默认支持自动标准。
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetAuto(FALSE);
```

```
//设置最大刻度为 M;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMaximum(M);
```

```
//设置最小刻度为 m;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMinimum(m);
```

```
//设置轴的等分数 D;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMajorDivision(D);
```

```
//设置每等分的刻度线数 n;
```

```
m_Chart.GetPlot().GetAxis().GetValuesScale().SetMinorDivision(n);
```

b) 横轴初始化属性

```
VARIANT var;
```

```
m_Chart.GetPlot().GetAxis(0,var)//获取横轴
```

其他属性设置跟纵轴相同。

1.2 数据显示

a) 设置标题栏和标签

```
m_Chart.SetTitleText(“标题”); //设置标题栏
```

```
m_Chart.SetRowLabel((“第 I 行”)); //设置第 i 行标签
```

```
m_Chart.SetColumnLabel((“第 j 列”)); //设置第 j 列标签
```

b) 行列的显示布局

MSChart 的行列显示布局有其自身的特点：下面显示是一个行列 4×3 （矩形图），即（四行，三列）的布局示意图。

```
m_Chart.SetRowCount(4); //没条曲线三个四个点 （曲线图）
```

```
m_Chart.SetColumnCount(3); //设置曲线条数为三条（曲线图）
```

c) 行列操作

```
// 操作行列第 i 行、第 j 列
```

```
m_Chart.SetRow(i); // 第 i 行
```

```
m_Chart.SetColumn(j); //第 j 行
```

```
m_Chart.SetRowLabel((“第 i 行”)); //设置第 i 行标签
```

```
CString str= ” 90.5” ;
```

```
m_Chart.SetData((LPCTSTR(str)); //设置行列，j>的显示数据
```

```
m_Chart.Refresh(); //刷新视图
```

d) 显示方式

获取当前的显示方式：

```
long nType =m_Chart.GetChartType();
```

设置显示方式:

```
m_Chart.SetChartType(0);//3D(三维) 显示
```

```
m_Chart.SetChartType(1);//2D(二维) 显示
```

```
m_Chart.Refresh();
```

其它常用组合方式为:

```
m_Chart.SetChartType(1|0) //2D 柱(条)形,
```

```
m_Chart.SetChartType(0|0) //3D 柱(条)形
```

```
m_Chart.SetChartType(1|2) //2D 线条型
```

```
m_Chart.SetChartType(0|2) //3D 线条型
```

```
m_Chart.SetChartType(1|4) //2D 区域型
```

```
m_Chart.SetChartType(0|4) //3D 区域型
```

```
m_Chart.SetChartType(1|6) //2D 阶梯型
```

```
m_Chart.SetChartType(0|6) //3D 阶梯型
```

```
m_Chart.SetChartType(1|8) //2D 复(混)合型
```

```
m_Chart.SetChartType(0|8) //3D 复(混)合型
```

另外,在 2D 方式中,还有两类: 饼型和 XY 型

```
m_Chart.SetChartType(14) //2D 饼型
```

```
m_Chart.SetChartType(16) //2DXY 型
```

e) 其他

其他属性, 比如设置字体, 颜色, 对齐方式等。

```
//=====
```

```
//VC 知识库上的参考文章
```

```
http://www.vckbase.com/document/viewdoc/?id=959
```

3.3 设置 m_Chart void CDemoView::InitChart()

```
{  
    // 设置标题  
    m_Chart.SetTitleText("mschart 示例 by thinkry@263.net");  
    // 下面两句改变背景色  
    m_Chart.GetBackdrop().GetFill().SetStyle(1);  
    m_Chart.GetBackdrop().GetFill().GetBrush().GetFillColor().Set(255, 255, 255);  
    // 显示图例  
    m_Chart.SetShowLegend(TRUE);  
    m_Chart.SetColumn(1);  
    m_Chart.SetColumnLabel((LPCTSTR)"1 号机");  
    m_Chart.SetColumn(2);  
    m_Chart.SetColumnLabel((LPCTSTR)"2 号机");  
    m_Chart.SetColumn(3);  
    m_Chart.SetColumnLabel((LPCTSTR)"3 号机");  
    // 栈模式  
    // m_Chart.SetStacking(TRUE);  
    // Y 轴设置  
    VARIANT var;  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetAuto(FALSE); // 不自动标注 Y 轴刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMaximum(100); // Y 轴最大刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinimum(0); // Y 轴最小刻度  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMajorDivision(5); // Y 轴刻度 5 等分  
    m_Chart.GetPlot().GetAxis(1,var).GetValueScale().SetMinorDivision(1); // 每刻度一个刻度线  
    m_Chart.GetPlot().GetAxis(1,var).GetAxisTitle().SetText("小时"; // Y 轴名称  
    // 3 条曲线  
    m_Chart.SetColumnCount(3);  
    // 线色  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetPen().GetVtColor().Set(0, 0, 255);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetPen().GetVtColor().Set(255, 0, 0);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetPen().GetVtColor().Set(0, 255, 0);  
    // 线宽(对点线图有效)  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetPen().SetWidth(50);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetPen().SetWidth(100);  
    m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetPen().SetWidth(2);  
    // 数据点类型显示数据值的模式(对柱柱状图和点线图有效)
```

```

// 0: 不显示 1: 显示在柱状图外
// 2: 显示在柱状图内上方 3: 显示在柱状图内中间 4: 显示在柱状图内下方
m_Chart.GetPlot().GetSeriesCollection().GetItem(1).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
m_Chart.GetPlot().GetSeriesCollection().GetItem(2).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
m_Chart.GetPlot().GetSeriesCollection().GetItem(3).GetDataPoints().GetItem(-1).GetDataPointLabel().SetLocationType(1);
}
3.4 设置数据 void CDemoView:rawChart()
{
int nRowCount = 6;
m_Chart.SetRowCount(nRowCount);
VARIANT var;

// 不自动标注 X 轴刻度
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetAuto(FALSE);

// 每刻度一个标注
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerLabel(1);

// 每刻度一个刻度线
m_Chart.GetPlot().GetAxis(0,var).GetCategoryScale().SetDivisionsPerTick(1);

// X 轴名称
m_Chart.GetPlot().GetAxis(0,var).GetAxisTitle().SetText("日期");
char buf[32];
srand( (unsigned)time( NULL ) );
for(int row = 1; row <= nRowCount; ++row)
{
m_Chart.SetRow(row);
sprintf(buf, "%d 号", row);
m_Chart.SetRowLabel((LPCTSTR)buf);
m_Chart.GetDataGrid().SetData(row, 1, rand() * 100 / RAND_MAX, 0);//设置第一条曲线的第 row 个点
m_Chart.GetDataGrid().SetData(row, 2, rand() * 100 / RAND_MAX, 0);//设置第二条曲线的第 row 个点
m_Chart.GetDataGrid().SetData(row, 3, rand() * 100 / RAND_MAX, 0);//设置第三条曲线的第 row 个点
}
m_Chart.Refresh();
}

/*
SetData (row, column, dataPoint, nullFlag)

```

SetData 方法语法包括以下组成部分:

row Integer 类型。标识包含数据点值的行。
column Integer 类型。标识包含数据点值的列。
dataPoint Double 类型。数据点值。
nullFlag Integer 类型。指示数据点值是否为空
*/

3.5 改变显示类型 // 折线图

```
void CDemoView::OnChartLine()
{
    m_Chart.SetChartType(3);
    DrawChart();
}
```

// 柱状图

```
void CDemoView::OnChartCombi()
{
    m_Chart.SetChartType(1);
    DrawChart();
}
```

// 饼状图

```
void CDemoView::OnChartPie()
{
    m_Chart.SetChartType(14);
    DrawChart();
}
```

在access 中创建库表（.net实现）

```
1 Imports System.Data.OleDb
2
3 Public Class Form1
4 |
5 Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Lo
ad
6 | Dim cnn As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Documents and Set
tings\Administrator\My Documents\mydata.mdb")
7 | Dim cmd As New OleDbCommand("CREATE TABLE person (name varchar(20),sex varchar(6),id int primar
y key)", cnn)
8 | Dim adapter As OleDbDataAdapter
9 | Dim dst As New DataSet
10 | Try
11 |     cnn.Open()
12 |     cmd.ExecuteNonQuery()
13 |     cmd.CommandText = "insert into person values('zzq','male',1)"
```

```

14 | cmd.ExecuteNonQuery()
15 | cmd.CommandText = "select * from person"
16 | adapter = New OleDbDataAdapter(cmd)
17 | adapter.Fill(dst)
18 | DataGridView1.DataSource = dst.Tables(0).DefaultView
19 | cnn.Close()
20 | Catch ex As Exception
21 |     MessageBox.Show(ex.Message)
22 | End Try
23 | End Sub
24 |
25 |
26 | End Class
27

```

目前 access 中的控制命令好像只有 create table,alter table,create index，还有 drop 。不过网上有说可以用 CREATE PROCEDURE 创建存储过程，我一直没有找到。我用的是是 access 2003+Microsoft Jet 4.0 Service Pack 8

vs2008 开发 3D 报表-MSChart 学习帮助-开发日志(一)-带附件下载 收藏

< type=text/javascript> < src="http://pagead2.googlesyndication.com/pagead/show_ads.js" type=text/javascript>
 我 X 这是我第二次写这篇文章了，刚才写了半天 IE 浏览器给关闭了，我 X 不过我又狠下心来，为大家再服务一会 :)

首先看到这个报表的界面会不会感觉很爽，没错，这就是用 MSChart 的图形报表组件，完全免费的 开发的

如果你不看这篇文章，可能会碰很多钉子，写这个 就是为了让大家更快，更顺利的去使用这个.net FrameWork 3.5 新组件的功能

先发表了，继续修改，免得又出现关闭浏览器问题...

继续 首先你要安装 Vs2008 这我就不说了， .net FrameWork 3.5 这也是要有的，

然后下载 MSChart 的和 4 个补丁文件，在这里我都给大家提供好了，在我的资源里就有下载，5 分哦，不过下载完评论下就不扣分了。

然后依次安装

dotnetfx35setup.exe

MSChart_VisualStudioAddOn.exe

MSChartLP_chs.exe

MSChart.exe

这几个文件，然后 OK

启动 VS2008 准备写程序

创建一个 Web 项目

然后在左边的工具卡里添加 .net Framework 组件 Chart 这里添加 Web UI 的 Chart 组件如果是 WinForm 的话就添加 WinForm 的

然后在数据的选项卡里，把 Chart 的控件拖到界面上

HTML 代码如下

```
<asp:chart id="Chart1" runat="server" Height="296px" Width="412px"
ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)" Palette="BrightPastel" imagetype="Png"
BorderDashStyle="Solid" BackSecondaryColor="White" BackGradientStyle="TopBottom" BorderWidth="2"
backcolor="#D3DFF0" BorderColor="26, 59, 105">
    <legends>
        <asp:Legend IsTextAutoFit="False" Name="Default"
BackColor="Transparent" Font="Trebuchet MS, 8.25pt, style=Bold"></asp:Legend>
    </legends>
    <borderskin skinstyle="Emboss"></borderskin>
    <series>
        <asp:Series Name="Column" BorderColor="180, 26, 59, 105">
            <points>
                <asp:DataPoint YValues="45" />
                <asp:DataPoint YValues="34" />
            </points>
        </asp:Series>
    </series>
</asp:chart>
```

```

        <asp:DataPoint YValues="67" />
        <asp:DataPoint YValues="31" />
        <asp:DataPoint YValues="27" />
        <asp:DataPoint YValues="87" />
        <asp:DataPoint YValues="45" />
        <asp:DataPoint YValues="32" />
    </points>
</asp:Series>
</series>
<chartareas>
    <asp:ChartArea Name="ChartArea1" BorderColor="64, 64, 64, 64"
BorderDashStyle="Solid" BackSecondaryColor="White" BackColor="64, 165, 191, 228"
ShadowColor="Transparent" BackGradientStyle="TopBottom">
        <area3dstyle Rotation="10" perspective="10" Inclination="15"
IsRightAngleAxes="False" wallwidth="0" IsClustered="False"></area3dstyle>
        <axisy linecolor="64, 64, 64, 64">
            <labelstyle font="Trebuchet MS, 8.25pt, style=Bold" />
            <majorgrid linecolor="64, 64, 64, 64" />
        </axisy>
        <axisx linecolor="64, 64, 64, 64">
            <labelstyle font="Trebuchet MS, 8.25pt, style=Bold" />
            <majorgrid linecolor="64, 64, 64, 64" />
        </axisx>
    </asp:ChartArea>
</chartareas>
</asp:chart>

```

< type=text/javascript> < src="http://pagead2.googlesyndication.com/pagead/show_ads.js" type=text/javascript>
 后台代码如下

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Web.UI.DataVisualization.Charting;

```

```

namespace WebApplication2
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Create new data series and set it's visual attributes
            Series series = new Series("Spline");
            series.ChartType = SeriesChartType.Spline;
            series.BorderWidth = 3;
            series.ShadowOffset = 2;

            // Populate new series with data
            series.Points.AddY(67);
            series.Points.AddY(57);
            series.Points.AddY(83);
            series.Points.AddY(23);
            series.Points.AddY(70);
            series.Points.AddY(60);
            series.Points.AddY(90);
            series.Points.AddY(20);

            // Add series into the chart's series collection
            Chart1.Series.Add(series);
        }
    }
}

```

ok，你可以按 F5 运行了

是不是报错了？

哈哈，当然要报错 还要配置 Web.Config 开个玩笑：)

Web.Config 还要配置如下

```

<appSettings>
    <add key="ChartImageHandler" value="Storage=file;Timeout=20;Url=~/tempImages/;" />
</appSettings>

```

```

<httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    <add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    <add verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" validate="false"/>
    <add path="ChartImg.axd" verb="GET,HEAD"
type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler, System.Web.DataVisualization,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" validate="false"/>
</httpHandlers>
<handlers>
    <remove name="ChartImageHandler"/>
    <add name="ChartImageHandler" precondition="integratedMode" verb="GET,HEAD"
path="ChartImg.axd" type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
System.Web.DataVisualization, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
</handlers>

```

ok, 再次运行, 你会看到你看到的东西了, 哈哈~~~感谢大家关注~~~Gavin Luo 的博客

MSChart 学习帮助-开发日志（二）.net 3.5 图表控件 收藏

```

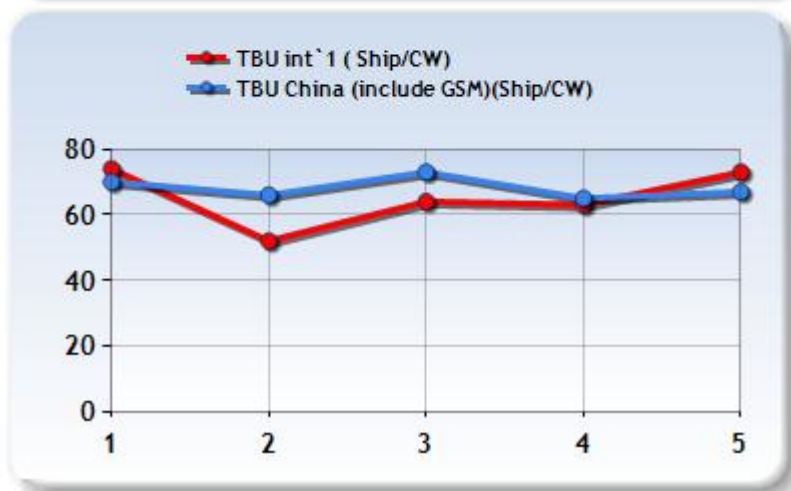
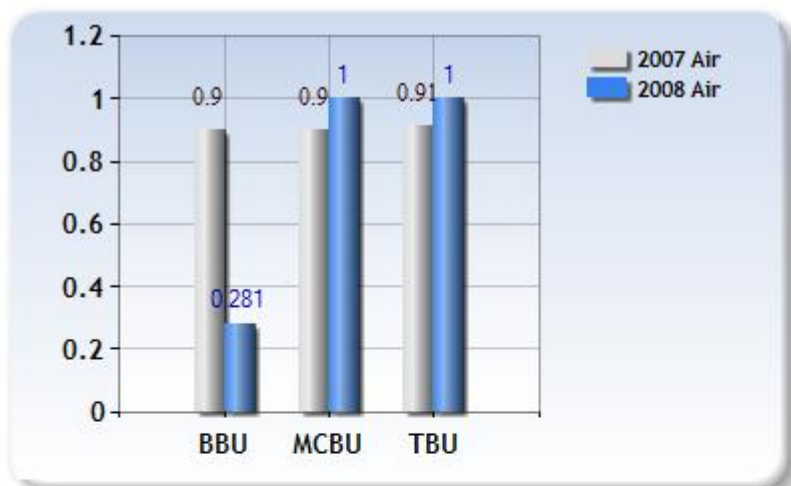
< type="text/javascript"> < src="http://pagead2.googlesyndication.com/pagead/show_ads.js"
type="text/javascript">

```

仔细看了 Scott 写的如何开发 MSChart, 但始终还是有一个地方不明白, 原话上说: 可用于免费使用 ASP.NET 3.5。但在官方网站上的 MSChart 板块又看到了 license, 如果真有人想用的话, 可以打微软的销售电话 8008203800, 或者等我知道了, 我会为大家解答, 因为这个好东西, 我肯定会找机会用在我的项目里

今天又把 MSChart 拿出来研究, 由于没有帮助文档, 网上也没有案例和资料, 唯一的一个帮助就是 Microsoft 给的 200 个例子。于是就看源码, 看属性, 尝试着每个属性的作用 -_- 这是很辛苦的

ok 终于基本搞懂了一些 首先介绍一下这两个图的做法



我把源码贴出来这是前台的

view plaincopy to clipboardprint?

```
<asp:chart id="Chart1" runat="server" Height="250px" Width="400px"
    ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)" Palette="BrightPastel"
    imagetype="Png" BorderDashStyle="Solid" BackSecondaryColor="White"
    BackGradientStyle="TopBottom" BorderWidth="2" backcolor="#D3DFF0" >
    <legends>
        <asp:Legend IsTextAutoFit="true" Name="Default" BackColor="Transparent"
            Font="Trebuchet MS, 8.25pt, style="Bold" mce_style="Bold" />
    </legends>
    <borderskin skinstyle="Emboss" />
    <series>
        <asp:Series Name="2007 Air"
            CustomProperties="MinPixelPointWidth=1, PixelPointWidth=30,
            DrawingStyle=Cylinder, MaxPixelPointWidth=30, DrawSideBySide=True, PointWidth=0"
            Color="224, 224, 224" Font="Times New Roman, 9pt"
            MarkerColor="Navy" MarkerSize="12"
            BorderWidth="3" ShadowOffset="2" IsValueShownAsLabel="True"
            LabelForeColor="64, 0, 0" >
            <points>
                <asp:DataPoint YValues="0.9" AxisLabel="BBU" />
```

```

        <asp:DataPoint YValues="0.9" AxisLabel="MCBU" />
        <asp:DataPoint YValues="0.91" AxisLabel="TBU" />
    </points>
    <EmptyPointStyle IsValueShownAsLabel="True" LabelAngle="15"
        LabelBackColor="Olive" LabelForeColor="RosyBrown"
MarkerSize="12" />
    <SmartLabelStyle CalloutLineColor="DarkRed"
CalloutLineAnchorCapStyle="Diamond"
        CalloutStyle="None" />
    </asp:Series>
</series>
<chartareas>
    <asp:ChartArea Name="Air" BorderColor="64, 64, 64, 64"
BorderDashStyle="Solid" BackSecondaryColor="White" BackColor="64, 165, 191, 228"
ShadowColor="Transparent" BackGradientStyle="TopBottom">
        <area3dstyle PointDepth="150" PointGapDepth="150" Rotation="15"
perspective="2" Inclination="20" LightStyle="Realistic" wallwidth="0" IsClustered="False"></area3dstyle>

        <axisy linecolor="64, 64, 64, 64" >
            <labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

            <majorgrid linecolor="64, 64, 64, 64" />
        </axisy>
        <axisx linecolor="64, 64, 64, 64" >
            <labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

            <majorgrid linecolor="64, 64, 64, 64" />
        </axisx>
    </asp:ChartArea>
</chartareas>
</asp:chart><br />
<asp:chart id="Chart2" runat="server" Height="250px" Width="400px"
    ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)" Palette="BrightPastel"
    imagetype="Png" BorderDashStyle="Solid" BackSecondaryColor="White"
    BackGradientStyle="TopBottom" BorderWidth="2" backcolor="#D3DFF0" >
    <legends>
        <asp:Legend IsTextAutoFit="true" Alignment="Center" Docking="Top"
Name="Default" BackColor="Transparent" Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold""></asp:Legend>
    </legends>
    <borderskin skinstyle="Emboss"></borderskin>
    <series>
        <asp:Series Name="TBU int`1 ( Ship/CW)" MarkerSize="8"
BorderWidth="3"
            XValueType="Double" ChartType="Line" MarkerStyle="Circle"
ShadowColor="Black"

```

```

        BorderColor="180, 26, 59, 105" Color="Red" ShadowOffset="2"
        YValueType="Double">
</asp:Series>
<asp:Series Name="TBU China (include GSM)(Ship/CW)"
ChartType=Line
        BorderWidth="3" ShadowOffset="2" MarkerSize="8"
MarkerStyle="Circle"
        ShadowColor="Black" BorderColor="180, 26, 59, 105">
</asp:Series>
</series>
<chartareas>
<asp:ChartArea Name="Ship/CW" BorderColor="64, 64, 64, 64"
        BorderDashStyle="Solid" BackSecondaryColor="White"
        BackColor="64, 165, 191, 228" ShadowColor="Transparent"
        BackGradientStyle="TopBottom" IsSameFontSizeForAllAxes="True" >
<area3dstyle PointDepth="150" PointGapDepth="150" Rotation="15"
perspective="2" Inclination="20" LightStyle="Realistic" wallwidth="0" IsClustered="False"></area3dstyle>

<axisy linecolor="64, 64, 64, 64" >
<labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

<majorgrid linecolor="64, 64, 64, 64" />
</axisy>
<axisx linecolor="64, 64, 64, 64" >
<labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

<majorgrid linecolor="64, 64, 64, 64" />
</axisx>
</asp:ChartArea>
</chartareas>
</asp:chart>
<asp:chart id="Chart1" runat="server" Height="250px" Width="400px"
        ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)" Palette="BrightPastel"
        imagetype="Png" BorderDashStyle="Solid" BackSecondaryColor="White"
        BackGradientStyle="TopBottom" BorderWidth="2" backcolor="#D3DFF0" >
<legends>
<asp:Legend IsTextAutoFit="true" Name="Default" BackColor="Transparent"
Font="Trebuchet MS, 8.25pt, style="Bold" mce_style="Bold""></asp:Legend>
</legends>
<borderskin skinstyle="Emboss"></borderskin>
<series>
<asp:Series Name="2007 Air"
        CustomProperties="MinPixelPointWidth=1, PixelPointWidth=30,
DrawingStyle=Cylinder, MaxPixelPointWidth=30, DrawSideBySide=True, PointWidth=0"
        Color="224, 224, 224" Font="Times New Roman, 9pt"
MarkerColor="Navy" MarkerSize="12"

```

```

        BorderWidth="3" ShadowOffset="2" IsValueShownAsLabel="True"
        LabelForeColor="64, 0, 0" >
        <points>
            <asp:DataPoint YValues="0.9" AxisLabel="BBU" />
            <asp:DataPoint YValues="0.9" AxisLabel="MCBU" />
            <asp:DataPoint YValues="0.91" AxisLabel="TBU" />
        </points>
        <EmptyPointStyle IsValueShownAsLabel="True" LabelAngle="15"
            LabelBackColor="Olive" LabelForeColor="RosyBrown"
MarkerSize="12" />
        <SmartLabelStyle CalloutLineColor="DarkRed"
CalloutLineAnchorCapStyle="Diamond"
            CalloutStyle="None" />
        </asp:Series>
    </series>
    <chartareas>
        <asp:ChartArea Name="Air" BorderColor="64, 64, 64, 64"
BorderDashStyle="Solid" BackSecondaryColor="White" BackColor="64, 165, 191, 228"
ShadowColor="Transparent" BackGradientStyle="TopBottom">
            <area3dstyle PointDepth="150" PointGapDepth="150" Rotation="15"
perspective="2" Inclination="20" LightStyle="Realistic" wallwidth="0" IsClustered="False"></area3dstyle>

            <axisy linecolor="64, 64, 64, 64" >
                <labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

                <majorgrid linecolor="64, 64, 64, 64" />
            </axisy>
            <axisx linecolor="64, 64, 64, 64" >
                <labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

                <majorgrid linecolor="64, 64, 64, 64" />
            </axisx>
        </asp:ChartArea>
    </chartareas>
</asp:chart><br />
    <asp:chart id="Chart2" runat="server" Height="250px" Width="400px"
        ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)" Palette="BrightPastel"
        imagetype="Png" BorderDashStyle="Solid" BackSecondaryColor="White"
        BackGradientStyle="TopBottom" BorderWidth="2" backcolor="#D3DFF0" >
        <legends>
            <asp:Legend IsTextAutoFit="true" Alignment="Center" Docking="Top"
Name="Default" BackColor="Transparent" Font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"></asp:Legend>
        </legends>
        <borderskin skinstyle="Emboss"></borderskin>
        <series>

```

```

        <asp:Series Name="TBU int`1 ( Ship/CW)" MarkerSize="8"
BorderWidth="3"
XValueType="Double" ChartType="Line" MarkerStyle="Circle"
ShadowColor="Black"
BorderWidth="3" BorderColor="180, 26, 59, 105" Color="Red" ShadowOffset="2"
YValueType="Double">
</asp:Series>
<asp:Series Name="TBU China (include GSM)(Ship/CW)"
ChartType=Line
BorderWidth="3" ShadowOffset="2" MarkerSize="8"
MarkerStyle="Circle"
ShadowColor="Black" BorderColor="180, 26, 59, 105">
</asp:Series>
</series>
<chartareas>
<asp:ChartArea Name="Ship/CW" BorderColor="64, 64, 64, 64"
BorderDashStyle="Solid" BackSecondaryColor="White"
BackColor="64, 165, 191, 228" ShadowColor="Transparent"
BackGradientStyle="TopBottom" IsSameFontSizeForAllAxes="True" >
<area3dstyle PointDepth="150" PointGapDepth="150" Rotation="15"
perspective="2" Inclination="20" LightStyle="Realistic" wallwidth="0" IsClustered="False"></area3dstyle>

<axisy linecolor="64, 64, 64, 64" >
<labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

<majorgrid linecolor="64, 64, 64, 64" />
</axisy>
<axisx linecolor="64, 64, 64, 64" >
<labelstyle font="Trebuchet MS, 8.25pt, style="Bold"
mce_style="Bold"" />

<majorgrid linecolor="64, 64, 64, 64" />
</axisx>
</asp:ChartArea>
</chartareas>
</asp:chart>

```

这是后台的

view plaincopy to clipboardprint?

```

protected void Page_Load(object sender, EventArgs e)
{
    showAirExample();
    ShowPointExample();
}
protected void showAirExample()
{

```

```

Series series = new Series("2008 Air");
series.ChartType = SeriesChartType.Column;
series["DrawingStyle"] = "Cylinder";
series.IsValueShownAsLabel = true;
series.BorderWidth = 3;
series.ShadowOffset = 2;
series["PixelPointWidth"] = "30";
// Populate new series with data
series.Points.AddY(0.281);
series.Points.AddY(1);
series.Points.AddY(1);
// Add series into the chart's series collection
Chart1.Series.Add(series);
Chart1.Series["2008 Air"].ChartType = SeriesChartType.Column;
// Show as 3D
Chart1.ChartAreas["Air"].Area3DStyle.Enable3D = false;
// Draw as 3D Cylinder
Chart1.Series["2008 Air"]["DrawingStyle"] = "Cylinder";
Chart1.Series["2008 Air"].LabelForeColor = Color.FromArgb(0, 0, 192);
}
protected void ShowPointExample() {
    Random random = new Random();
    for (int pointIndex = 0; pointIndex < 5; pointIndex++)
    {
        Chart2.Series[0].Points.AddY(random.Next(45, 95));
        Chart2.Series[1].Points.AddY(random.Next(5, 75));
    }

    // Set point labels
    //Chart2.Series["TBU int`1 ( Ship/CW)"].IsValueShownAsLabel = true;

    //Chart2.Series["TBU China (include GSM)(Ship/CW)"].IsValueShownAsLabel = true;
    // Enable X axis margin
    Chart2.ChartAreas["Ship/CW"].AxisX.IsMarginVisible = false;
    // Enable 3D, and show data point marker lines
    Chart2.ChartAreas["Ship/CW"].Area3DStyle.Enable3D = false;
    Chart2.Series[0]["ShowMarkerLines"] = "True";
    Chart2.Series[1]["ShowMarkerLines"] = "True";
}
protected void Page_Load(object sender, EventArgs e)
{
    showAirExample();
    ShowPointExample();
}
protected void showAirExample()
{

```

```

Series series = new Series("2008 Air");
series.ChartType = SeriesChartType.Column;
series["DrawingStyle"] = "Cylinder";
series.IsValueShownAsLabel = true;
series.BorderWidth = 3;
series.ShadowOffset = 2;
series["PixelPointWidth"] = "30";
// Populate new series with data
series.Points.AddY(0.281);
series.Points.AddY(1);
series.Points.AddY(1);
// Add series into the chart's series collection
Chart1.Series.Add(series);
Chart1.Series["2008 Air"].ChartType = SeriesChartType.Column;
// Show as 3D
Chart1.ChartAreas["Air"].Area3DStyle.Enable3D = false;
// Draw as 3D Cylinder
Chart1.Series["2008 Air"]["DrawingStyle"] = "Cylinder";
Chart1.Series["2008 Air"].LabelForeColor = Color.FromArgb(0, 0, 192);
}

protected void ShowPointExample() {
    Random random = new Random();
    for (int pointIndex = 0; pointIndex < 5; pointIndex++)
    {
        Chart2.Series[0].Points.AddY(random.Next(45, 95));
        Chart2.Series[1].Points.AddY(random.Next(5, 75));
    }

    // Set point labels
    //Chart2.Series["TBU int`1 ( Ship/CW)"].IsValueShownAsLabel = true;

    //Chart2.Series["TBU China (include GSM)(Ship/CW)"].IsValueShownAsLabel = true;
    // Enable X axis margin
    Chart2.ChartAreas["Ship/CW"].AxisX.IsMarginVisible = false;
    // Enable 3D, and show data point marker lines
    Chart2.ChartAreas["Ship/CW"].Area3DStyle.Enable3D = false;
    Chart2.Series[0]["ShowMarkerLines"] = "True";
    Chart2.Series[1]["ShowMarkerLines"] = "True";
}

```

三个电压，三个电流

C# code

```
string myConnectionString = ReportForm.Properties.Settings.Default.CEC710ConnectionString;
string mySelectQuery="SELECT  vol_va,vol_vb,vol_vc,cur_ia,cur_ib,cur_ic,opertime  FROM
[AmmeterData103]";

SqlConnection myConnection = new SqlConnection(myConnectionString);
SqlCommand myCommand = new SqlCommand(mySelectQuery, myConnection);
// Set chart data source
chart1.DataSource = myCommand;
chart1.BorderlineWidth = 3;
// Set series members names for the X and Y values
chart1.Series["A 相电压"].XValueMember = "OperTime";
chart1.Series["A 相电压"].YValueMembers = "vol_va";

chart1.Series["B 相电压"].XValueMember = "OperTime";
chart1.Series["B 相电压"].YValueMembers = "vol_vb";

chart1.Series["C 相电压"].XValueMember = "OperTime";
chart1.Series["C 相电压"].YValueMembers = "vol_vc";

chart1.Series["A 相电流"].XValueMember = "OperTime";
chart1.Series["A 相电流"].YValueMembers = "cur_ia";

chart1.Series["B 相电流"].XValueMember = "OperTime";
chart1.Series["B 相电流"].YValueMembers = "cur_ib";

chart1.Series["C 相电流"].XValueMember = "OperTime";
chart1.Series["C 相电流"].YValueMembers = "cur_ic";

// Data bind to the selected data source
chart1.DataBind();

foreach (Series s in chart1.Series)
{
    s.ChartType = SeriesChartType.Spline;
}
```

我的数据库的 **AmmeterData103** 表里有多个字段，但我这里画图只需要三个电压，三个电流，请问如何画到一张图中呢？

#1 楼 得分: 0 回复于: 2010-05-13 10:22:12 问题已经解决，贴出实现 线程控制 spline 图的显示。

【实现的 CS 代码】

C# code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Data.SqlClient;
using ReportForm.DataSet;
using System.Threading;
using wl.CommonUtil.Sys;

namespace ReportForm
{
    public partial class BrokenlinegraphForm : Form
    {
        public MainForm mForm = (MainForm)Context.GetAttribute("main");

        // Chart data adding thread
        private Thread addDataRunner;

        // Thread Add Data delegate
        public delegate void AddDataDelegate();
        public AddDataDelegate addDataDel;
        public string bnumber = string.Empty;
        public string search = string.Empty;
        public string drawerName = string.Empty;
        public static DateTime dt1 = new DateTime();
        public static DateTime dt2 = new DateTime();
        public DateTime dateTest = DateTime.Now;
        public double maxVal = 0.0;
        public int i = 0;

        public BrokenlinegraphForm()
        {
            InitializeComponent();
        }

        private void BrokenlinegraphForm_Load(object sender, EventArgs e)
        {

            //"AmS1031AT32A 2010-05-12 2010-05-12S"
```

```

        bnumber = mForm.Sqlstring.Substring(3, 3);
        drawerName = mForm.Sqlstring.Substring(7, 5);
        search = "D" + mForm.Sqlstring.Substring(3, 3) + mForm.Sqlstring.Substring(7, 5);
        dt1 = Convert.ToDateTime(mForm.Sqlstring.Substring(13, 10) + " " + "0:00:00");
        dt2 = Convert.ToDateTime(mForm.Sqlstring.Substring(24, 10) + " " + "0:00:00");
        dt2 = dt2.AddDays(1);

        addDataRunner = new Thread(new ThreadStart(CallData));
        addDataRunner.Start();
        addDataRunner.IsBackground = true;

        // update chart title
        chart1.Titles[0].Text = bnumber.ToString() + "号楼变电站 " + "抽屉名: " + drawerName;
    }

    public void CallData()
    {
        for (; )
        {
            AddDataThreadLoop();
            Thread.Sleep(1000);
        }
    }
    /// <summary>
    /// ^
    /// </summary>
    private void AddDataThreadLoop()
    {
        AddDataDelegate addDataDel = new AddDataDelegate(AddData);
        this.BeginInvoke(addDataDel);
    }
    /// <summary>
    /// Method which is invoked from the thread to add series points
    /// </summary>
    public void AddData()
    {
        double tempMax = 0.0;
        string myConnectionString = ReportForm.Properties.Settings.Default.CEC710ConnectionString;
        //string mySelectQuery = "SELECT vol_va,vol_vb,vol_vc,cur_ia,cur_ib,cur_ic,opertime FROM
[AmmeterData103] where ammeterid like 'd103at32a%' and opertime between '" + dt1 + "' and '" + dt2 + "'
order by opertime asc";
        string mySelectQuery = "SELECT vol_va,vol_vb,vol_vc,cur_ia,cur_ib,cur_ic,opertime FROM
[AmmeterData"+bnumber+"] where ammeterid like '"
                                + search + "' and opertime between '" + dt1 + "' and '" + dt2 + "'
order by opertime asc";
    }

```

```

SqlConnection myConnection = new SqlConnection(myConnectionString);
SqlDataAdapter da = null;
AmmeterDataSet.DataTableDataTable amData = new AmmeterDataSet.DataTableDataTable();
da = new SqlDataAdapter(mySelectQuery, myConnection);
da.Fill(amData);
for (int k = 0; k < amData.Rows.Count; k++)
{
    for (int m = 0; m < 6; m++)
    {
        if (maxValue < Convert.ToDouble(amData.Rows[k][m].ToString()))
        {
            maxValue = Convert.ToDouble(amData.Rows[k][m].ToString());
        }
    }
}

tempMax = Convert.ToInt16(maxValue * 11 / 50) * 5;
//设置 Y 轴的最大值
chart1.ChartAreas[0].AxisY.Maximum = tempMax;

chart1.Palette = ChartColorPalette.Pastel;

DateTime dateTest = DateTime.Now;

if (i < amData.Rows.Count)
{

    dateTest = Convert.ToDateTime(amData.Rows[i].ItemArray[6].ToString());

    chart1.Series[0].Points.AddXY(dateTest, amData.Rows[i].ItemArray[0]);

    chart1.Series[1].Points.AddXY(dateTest, amData.Rows[i].ItemArray[1]);

    chart1.Series[2].Points.AddXY(dateTest, amData.Rows[i].ItemArray[2]);

    chart1.Series[3].Points.AddXY(dateTest, amData.Rows[i].ItemArray[3]);

    chart1.Series[4].Points.AddXY(dateTest, amData.Rows[i].ItemArray[4]);

    chart1.Series[5].Points.AddXY(dateTest, amData.Rows[i].ItemArray[5]);

    i++;
}

```

```

//移走时间
double removeDate = dateTest.AddMinutes((double)(90) * (-1)).ToOADate();

for (int j = 0; j < 6; j++)
{
    //判断当前最小的时间值是否小于移走时间
    while (chart1.Series[j].Points[0].XValue < removeDate)
    {
        chart1.Series[j].Points.RemoveAt(0);
    }
}

chart1.ChartAreas[0].AxisX.Minimum = chart1.Series[0].Points[0].XValue;
chart1.ChartAreas[0].AxisX.Maximum
DateTime.FromOADate(chart1.Series[0].Points[0].XValue).AddMinutes(100).ToOADate();
}

//标签提示
chart1.Series[0].MarkerStyle = MarkerStyle.Square;
chart1.Series[0].MarkerColor = Color.Red;
chart1.Series[0].Label = "#VAL{G}";
chart1.Series[0].LabelToolTip = "A 相电压";

chart1.Series[1].MarkerStyle = MarkerStyle.Square;
chart1.Series[1].MarkerColor = Color.Green;
chart1.Series[1].Label = "#VAL{G}";
chart1.Series[1].LabelToolTip = "B 相电压";

chart1.Series[2].MarkerStyle = MarkerStyle.Square;
chart1.Series[2].MarkerColor = Color.Yellow;
chart1.Series[2].Label = "#VAL{G}";
chart1.Series[2].LabelToolTip = "C 相电压";

chart1.Invalidate();
}

//关闭按钮事件
private void BrokenlinegraphForm_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        addDataRunner.Abort();
    }
}

```

```

        catch
        {

        }
    }

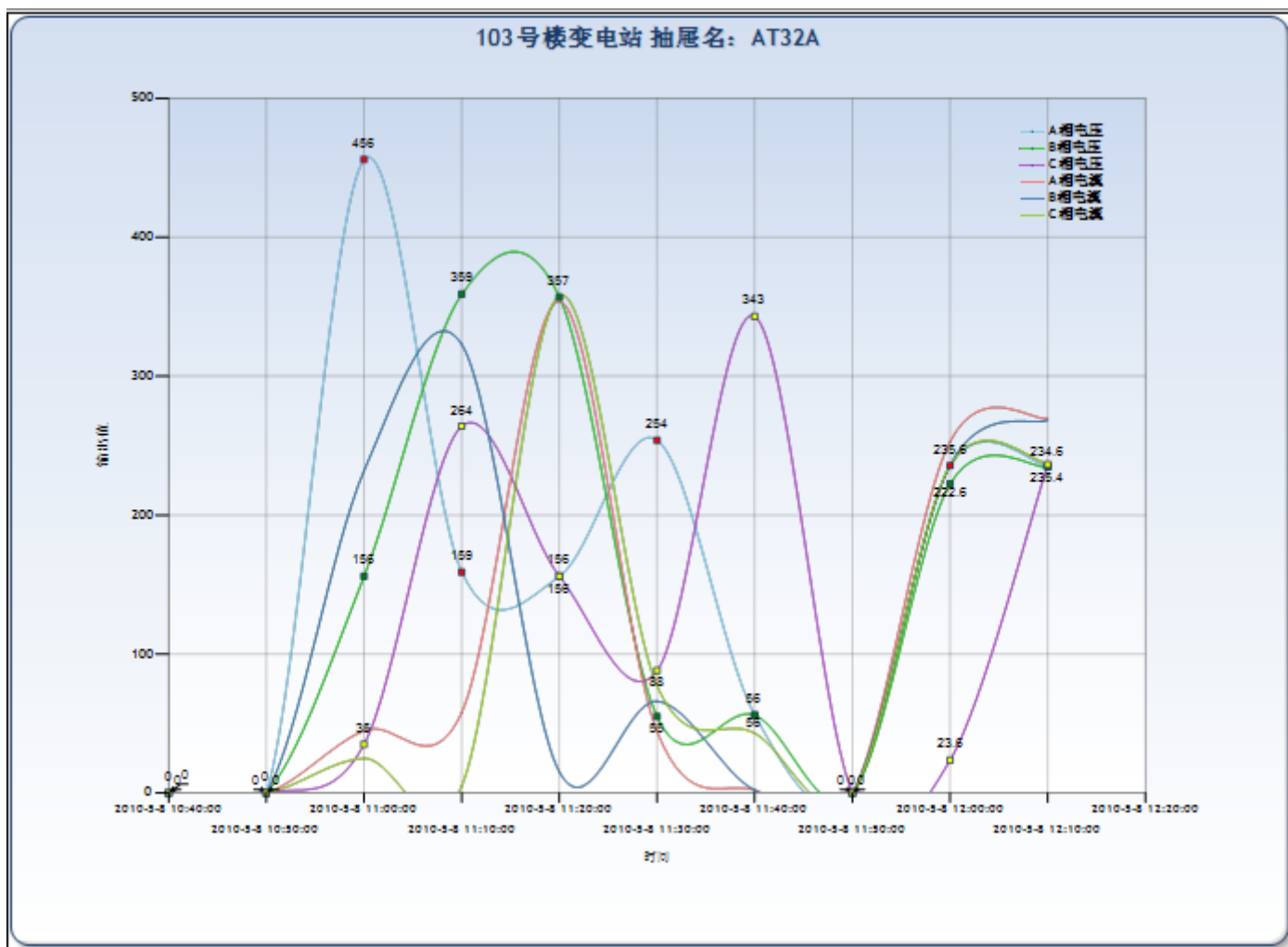
    /// <summary>
    /// 停止线程
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void stopTrending_Click(object sender, EventArgs e)
    {
        if (addDataRunner.IsAlive == true)
        {
            addDataRunner.Suspend();
        }
        continueTrending.Enabled = true;
    }

    /// <summary>
    /// 继续线程
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void continueTrending_Click(object sender, EventArgs e)
    {
        if (addDataRunner.ThreadState == ThreadState.Suspended)
        {
            addDataRunner.Resume();
        }
        else if (addDataRunner.IsAlive == false)
        {
            addDataRunner.Start();
        }
    }

}

```

复于：2010-05-13 15:48:09



.NET MSChart应用的一个简单例子

这几天都在研究微软才发布的 MSChart 图表工具，我写了简单的例子发上来让大家看看。

使用前需要做得工作：1.把.net 平台升级为 3.5sp1 的。

2.安装 MSChart 控件。

3.在使用时必须在项目中的 web.config 中添加一些内容；



```
<appSettings>
  <add key="ChartImageHandler" value="Storage=file;Timeout=20;Url=~\tempImages/;" />
</appSettings>
```

```

        </appSettings>

        <httpHandlers>

            <add path="ChartImg.axd" verb="GET,HEAD" type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler, System.Web.DataVisualization, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" validate="false"/>
        </httpHandlers>
        <handlers>
            <remove name="ChartImageHandler"/>
            <add name="ChartImageHandler" preCondition="integratedMode" verb="GET,HEAD" path="ChartImg.axd" type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler, System.Web.DataVisualization, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
        </handlers>

```

这样我们就可以使用控件来画图表了

我在实例中主要对实例 2 做了详细的说明，实例 1 是 ms 上的例子！

前台代码：

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="test1.aspx.cs" Inherits="webchart_test1.test1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Chart ID="Chart1" runat="server" Height="296px" Width="412px" ImageLocation="~/TempImages/ChartPic_#SEQ(300,3)"
                Palette="BrightPastel" ImageType="Png" BackSecondaryColor="White" BackGradientStyle="TopBottom"
                BorderWidth="2" BackColor="#D3DFF0" BorderColor="26, 59, 105" AntiAliasing="None"
                BorderlineDashStyle="Solid">

```

```

    <Legends>
        <asp:Legend IsTextAutoFit="False" Name="Default" BackColor="Transparent" Font="Trebuchet MS, 8.25pt, style=Bold">
        </asp:Legend>
    </Legends>
    <BorderSkin SkinStyle="Emboss"></BorderSkin>
    <Series>
        <asp:Series Name="Column" BorderColor="Fuchsia" Color="Green">
            <Points>
                <asp:DataPoint YValues="45" />
                <asp:DataPoint YValues="34" />
                <asp:DataPoint YValues="67" />
                <asp:DataPoint YValues="31" />
                <asp:DataPoint YValues="27" />
                <asp:DataPoint YValues="87" />
                <asp:DataPoint YValues="45" />
                <asp:DataPoint YValues="32" />
            </Points>
        </asp:Series>
    </Series>
    <ChartAreas>
        <asp:ChartArea Name="ChartArea1" BorderColor="64, 64, 64, 64" BorderDashStyle="Solid"
            BackSecondaryColor="White" BackColor="64, 165, 191, 228" ShadowColor="Transparent"
            BackGradientStyle="TopBottom">
            <Position Auto="true"/>
            <Area3DStyle Rotation="10" Perspective="10" Inclination="15" IsRightAngleAxes="False"
                WallWidth="0" IsClustered="False"></Area3DStyle>
            <AxisY LineColor="Lime">
                <LabelStyle Font="Trebuchet MS, 8.25pt, style=Bold" />
                <MajorGrid LineColor="red" />
            </AxisY>
            <AxisX LineColor="Blue">
                <LabelStyle Font="Trebuchet MS, 8.25pt, style=Bold" />
                <MajorGrid LineColor="red" />
            </AxisX>
        </asp:ChartArea>
    </ChartAreas>

```

```

        </asp:Chart>
        <asp:Chart ID="chart2" runat="server" BackColor="#FFFFCC" Palette="BrightPastel" BorderWidth = "2" BorderColor = "#cc9900" >
            <Legends>
                <asp:Legend IsTextAutoFit="False" Name="Default" BackColor="Transparent" Font="Trebuchet MS, 8.25pt, style=Bold">
                    </asp:Legend>
                </Legends> <%-- 控制小图标的显示格式 --%>
                <BorderSkin ></BorderSkin> <%--<!--控制整个图片边缘的格式 -->--%>

                <Series><%-- 数据(一个数据集为<asp:Series></asp:Series>) --%>
            </Series>
            <ChartAreas> <%-- 作图区的修饰 --%>
                <asp:ChartArea Name="ChartArea1">
                    </asp:ChartArea>
                </ChartAreas>
            </asp:Chart>
        </div>
    </form>
</body>
</html>

```

后台代码:

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Web.UI.DataVisualization.Charting;
using System.Drawing;
namespace webchart_test1
{

```

```

public partial class test1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Create new data series and set it's visual attributes
        ////////////////////////////////////////实验 1//////////////////////////////////////
        ////////////////////////////////////////

        Series series = new Series("曲线");
        series.ChartType = SeriesChartType.Spline;
        series.BorderWidth = 3;
        series.ShadowOffset = 2;

        // Populate new series with data
        series.Points.AddY(167);
        series.Points.AddY(157);
        series.Points.AddY(183);
        series.Points.AddY(123);
        series.Points.AddY(170);
        series.Points.AddY(160);
        series.Points.AddY(190);
        series.Points.AddY(120);
        series["ShowMarkerLines"] = "true";

        // Add series into the chart's series collection
        Chart1.Series.Add(series);
        Title t = new Title("一个小实验");
        Chart1.Titles.Add(t);
        ////////////////////////////////////////实例 2//////////////////////////////////////
        ////////////////////////////////////////

        //*****绘图区*****
        *****

        //chart 代表整个图片； legends 代表数据显示标识； Series 图片中的数据集； C
hartAreas 代表图片中的绘图区（里面能包含多个 Series 数据集的显示）；

        //数据源
        int[] test = { 100, 70, 40, 30, 20, 60, 50, 30, 50, 90, 80, 70, 9

```

```

0, 10, 80, 60, 50, 40, 30, 20, 10, 30 }; //内销
    int[] test1 = { 80, 70, 90, 10, 80, 60, 50, 40, 30, 20, 10, 0, 5
0, 90, 100, 70, 40, 30, 20, 60, 50, 30 }; //出口

//图片属性设置
chart2.Width = (test.Count() * 25 + 200) >= 600 ? test.Count() * 2
5 + 200 : 600; //图片宽度
chart2.Height = 400; //图片高度
chart2.BackColor = Color.Azure; //图片背景色
Title t2 = new Title("试验 2"); //图片标题
chart2.Titles.Add(t2);

//数据集显示属性设置

//数据集"出口"显示属性设置
Series series1 = new Series("出口"); //数据集声明
series1.ChartType = SeriesChartType.Line; //数据显示方式 Line:为折
线 Spline:曲线
series1.Color = Color.Green; //线条颜色
series1.BorderWidth = 2; //线条宽度
series1.ShadowOffset = 1; //阴影宽度
series1.IsVisibleInLegend = true; //是否显示数据说明
series1.IsValueShownAsLabel = false; //线条上是否给吃数据的显示
series1.MarkerStyle = MarkerStyle.Circle; //线条上的数据点标志类型
series1.MarkerSize = 8; //标志的大
小

DateTime date1 = DateTime.Now.Date;
for (int i = 0; i < test1.Count(); i++) //向数据集绑定数据
{
    series1.Points.AddXY(date1, test1[i]); //分别往 X,Y 轴添加数据(可以
为多种类型) (有多中添加方式)
    date1 = date1.AddDays(1);
}
chart2.Series.Add(series1); //把数据集添加到 chart 中

//数据集"内销"显示属性设置

```

```

Series series2 = new Series("内销");
series2.ChartType = SeriesChartType.Line; //数据显示方式 Line:为折
线 Spline:曲线
series2.Color = Color.Red; //线条颜色
series2.BorderWidth = 2; //线条宽度
series2.ShadowOffset = 1; //阴影宽度
series2.IsVisibleInLegend = true; //是否显示数据说明
series2.IsValueShownAsLabel = false; //线条上是否给吃数据的显示
series2.MarkerStyle = MarkerStyle.Circle; //线条上的数据点标志类型
series2.MarkerSize = 8; //标志的大
小

DateTime date = DateTime.Now.Date;
for (int i = 0; i < test.Count(); i++)
{
    series2.Points.AddXY(date, test[i]);
    date = date.AddDays(1);
}

chart2.Series.Add(series2);

//作图区的显示属性设置
chart2.ChartAreas["ChartArea1"].AxisX.IsMarginVisible = false;
chart2.ChartAreas["ChartArea1"].Area3DStyle.Enable3D = false;
//背景色设置
chart2.ChartAreas["ChartArea1"].ShadowColor = Color.Transparent;
chart2.ChartAreas["ChartArea1"].BackColor = Color.Azure; //
该处设置为了由天蓝到白色的逐渐变化
chart2.ChartAreas["ChartArea1"].BackGradientStyle = GradientStyle.To
pBottom;

chart2.ChartAreas["ChartArea1"].BackSecondaryColor = Color.White;
//X,Y 坐标线颜色和大小
chart2.ChartAreas["ChartArea1"].AxisX.LineColor = Color.Blue;
chart2.ChartAreas["ChartArea1"].AxisY.LineColor = Color.Blue;
chart2.ChartAreas["ChartArea1"].AxisX.LineWidth = 2;
chart2.ChartAreas["ChartArea1"].AxisY.LineWidth = 2;
chart2.ChartAreas["ChartArea1"].AxisY.Title = "百分比";
//中间 X,Y 线条的颜色设置
chart2.ChartAreas["ChartArea1"].AxisX.MajorGrid.LineColor = Color.Bl
ue;

```

```

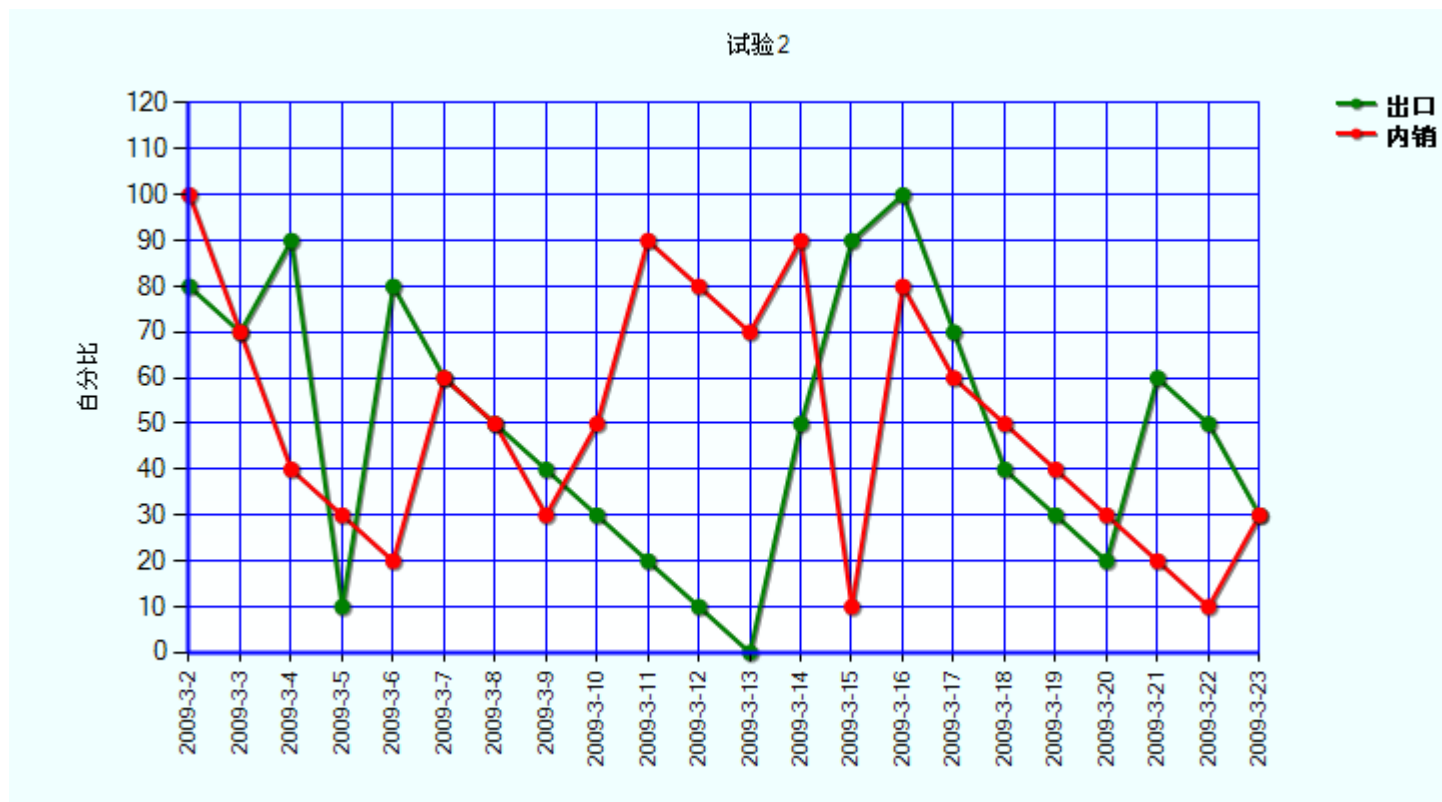
        chart2.ChartAreas["ChartArea1"].AxisY.MajorGrid.LineColor = Color.Blue;

        //X.Y 轴数据显示间隔
        chart2.ChartAreas["ChartArea1"].AxisX.Interval = 1; //X 轴数据显示间隔
        chart2.ChartAreas["ChartArea1"].AxisY.Interval = 10;
        //X 轴线条显示间隔
        chart2.ChartAreas["ChartArea1"].AxisX.MajorGrid.Interval = 1;
        //*****
        *****

    }
}
}

```

实例 2 的图片效果：



初试 ASP.NET Chart Control

.NET3.5 中推出了图表控件，可以同时支持 Web 和 WinForm 两种方式，由于平时很少使用，一直没有玩玩，闲来无事，简单研究了下，感觉功能真的很强大，基本上可以满足各种图表的应用，感觉这么好用的东西才研究，真是有些落伍啊，哈哈，想必很多人已经早已玩过，这里帮大家复习下，高手绕过。

安装 MSChart

由于是和.NET3.5 一起推出来的，所以只能在最新的开发环境中使用，需要.Net 3.5 Sp1 和 VS 2008 的开发环境。

控件下载地址： Microsoft .NET Framework 3.5 的 Microsoft 图表控件(Microsoft Chart Controls for Microsoft .NET Framework 3.5)

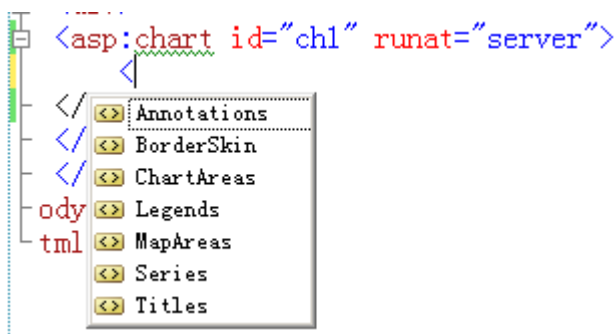
微软提供的 Demo: <http://code.msdn.microsoft.com/mschart>

按照上面的地址，下载安装就可以了。

使用 MSChart

安装好后，我们就可以使用 MSChart 了。

1.主要属性：



Annotations--图形注解集合，ChartAreas--图表区域集合，Legends--图例集合，Series--图表序列集合（即图表数据对象集合），Titles--图标的标题集合。

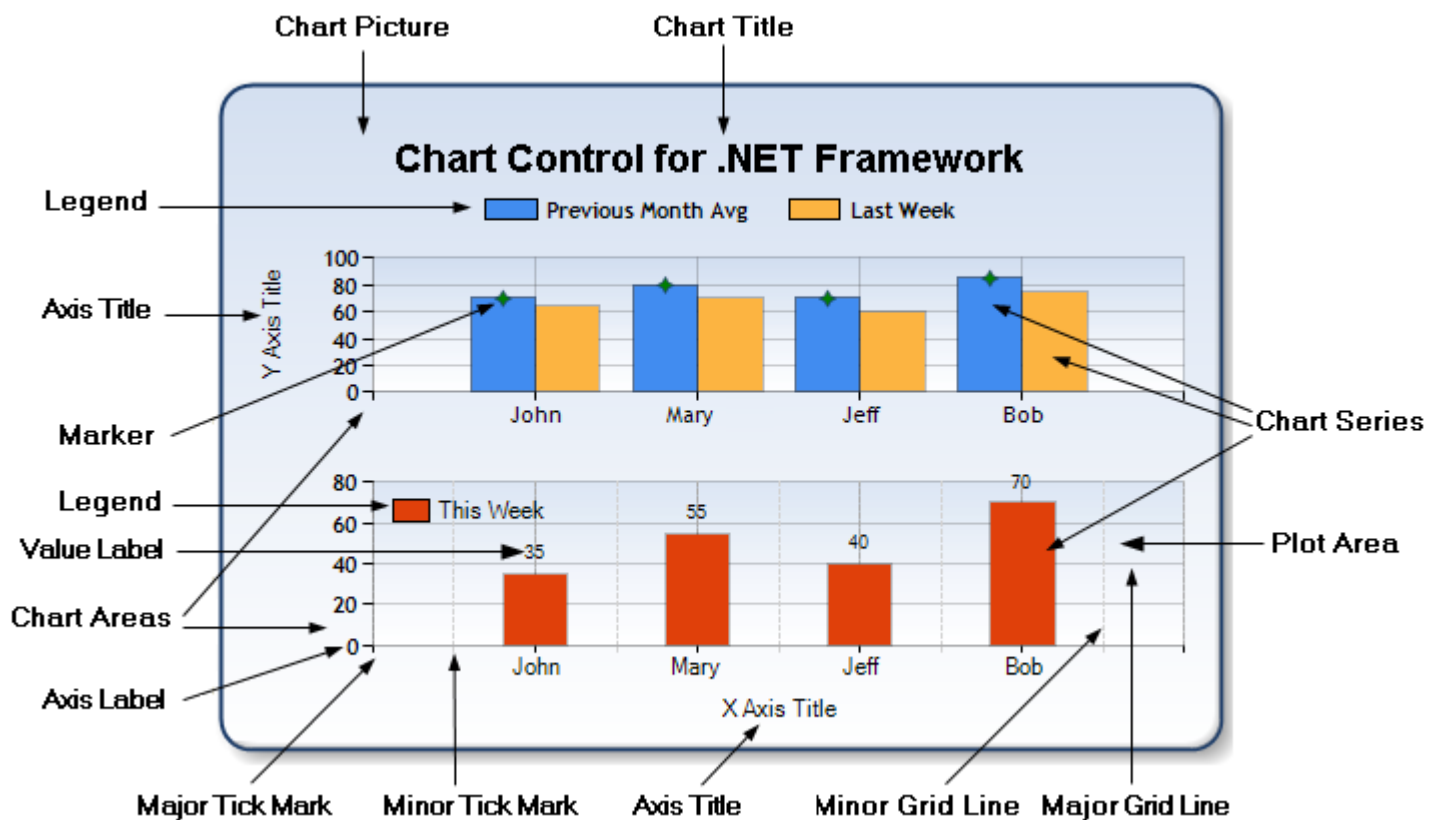
(1)Annotations: 是一个对图形的一些注解对象的集合，所谓注解对象，类似于对某个点的详细或者批注的说明。一个图形上可以拥有多个注解对象，可以添加十多种图形样式的注解对象，包括常见的箭头、云朵、矩形、图片等等注解符号，通过各个注解对象的属性，可以方便的设置注解对象的放置位置、呈现的颜色、大小、文字内容样式等常见的属性。

(2)ChartAreas: 是一个图表的绘图区，比如在一幅图中显示多个绘图。图表控件并不限制你添加多少个绘图区域，你可以根据你的需要进行添加。对于每一个绘图区域，你可以设置各自的属性，如：X,Y 轴属性、背景等。

(3)Legends: 是一个图例的集合，即标注图形中各个线条或颜色的含义，同样，一个图片也可以包含多个图例说明。

(4)Series: 是表数据对象集合，应该说是 MSChart 关键部分。即是实际的绘图数据区域，实际呈现的图形形状，由此集合中的每一个图表来构成的，可以往集合里面添加多个图表，每一个图表可以有自己的绘制形状、样式、独立的数据等。

(5)Titles: 图标的标题集合, 不难理解, 就是图表的标题配置, 同样可以添加多个标题。



其他属性:

AlignmentOrientation: 图表区对齐方向, 定义两个绘图区域间的对齐方式。

AlignmentStyle: 图表区对齐类型, 定义图表间用以对其的元素。

AlignWithChartArea: 参照对齐的绘图区名称。

InnerPlotPosition: 图表在绘图区内的位置属性。

Auto: 是否自动对齐。

Height: 图表在绘图区内的高度 (百分比, 取值在 0-100)

Width: 图表在绘图区内的宽度 (百分比, 取值在 0-100)

X,Y: 图表在绘图区内左上角坐标

Position: 绘图区位置属性, 同 InnerPlotPosition。

Name: 绘图区名称。

Axis: 坐标轴集合

TitleAlignment: 坐标轴标题对齐方式

Interval: 轴刻度间隔大小

IntervalOffset: 轴刻度偏移量大小

MinorGrid: 次要辅助线

MinorTickMark: 次要刻度线

MajorGrid: 主要辅助线

MajorTickMark: 主要刻度线

DataSourceID: MSChart 的数据源。

Palette: 图表外观定义。

IsValueShownAsLabel: 是否显示数据点标签，如果为 true，在图表中显示每一个数据值

Label: 数据点标签文本

LabelFormat: 数据点标签文本格式

LabelAngle: 标签字体角度

Name: 图表名称

Points: 数据点集合

XValueType: 横坐标轴类型

YValueType: 纵坐标轴类型

XValueMember: 横坐标绑定的数据源（如果数据源为 Table，则填写横坐标要显示的字段名称）

YValueMembers: 纵坐标绑定的数据源（如果数据源为 Table，则填写纵坐标要显示的字段名称，纵坐标可以有多个）

ChartType: 图表类型（柱形、饼形、线形、点形等）

width: MSChart 的宽度。

height: MSChart 的高度。

2.数据绑定方式

MSChart 提供了多种绑定数据的方式:

数组绑定:

```
double [] yval = { 2,6,4,5,3};
string [] xval = { "Peter", "Andrew", "Julie", "Mary", "Dave"};
Chart1.Series["Series 1"].Points.DataBindXY(xval,yval);
```

DataReader 绑定:

代码

```
string fileNameString = this.MapPath(".");
fileNameString += "..\\..\\..\\..\\data\\chartdata.mdb";
string myConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=" +
fileNameString;
string mySelectQuery="SELECT Name, Sales FROM REPS WHERE RegionID < 3;";
OleDbConnection myConnection = new OleDbConnection(myConnectionString);
OleDbCommand myCommand = new OleDbCommand(mySelectQuery, myConnection);
myCommand.Connection.Open();
OleDbDataReader myReader =
myCommand.ExecuteReader(CommandBehavior.CloseConnection);
Chart1.Series["Default"].Points.DataBindXY(myReader, "Name", myReader, "Sales");
myReader.Close();
myConnection.Close();
```

DataTable 绑定:

代码

```
string fileNameString = this.MapPath(".");
fileNameString += "..\\..\\..\\..\\data\\chartdata.mdb";
string myConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=" +
fileNameString;
string mySelectQuery="SELECT Name, Sales FROM REPS;";
OleDbConnection myConnection = new OleDbConnection(myConnectionString);
OleDbCommand myCommand = new OleDbCommand(mySelectQuery, myConnection);
myCommand.Connection.Open();
OleDbDataReader myReader =
myCommand.ExecuteReader(CommandBehavior.CloseConnection);
Chart1.DataBindTable(myReader, "Name");
myReader.Close();
myConnection.Close();
```

Excel 绑定:

代码

```
string fileNameString = this.MapPath(".");
    fileNameString += "..\\..\\..\\..\\data\\ExcelData.xls";
    string sConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
        fileNameString + ";Extended Properties=\\\"Excel 8.0;HDR=YES\\\"";
    OleDbConnection myConnection = new OleDbConnection( sConn );
    myConnection.Open();
    OleDbCommand myCommand = new OleDbCommand( "Select * From [data1$A1:E25]",
myConnection );
    OleDbDataReader myReader =
myCommand.ExecuteReader(CommandBehavior.CloseConnection);
    Chart1.DataBindTable(myReader, "HOUR");
    myReader.Close();
    myConnection.Close();
    foreach(Series ser in Chart1.Series)
    {
        ser.ShadowOffset = 1;
        ser.BorderWidth = 3;
        ser.ChartType = SeriesChartType.Line;
    }
```

还有其他形式的数据绑定，大家可以下载微软的 DEMO 研究下。

微软图表控件使用初探--数据绑定(交叉表)

本来在考虑是否继续写图表控件的使用文章,毕竟,微软自带的例子已经非常详细了,花点时间参考一下基本上可以了解用法和使用特性,不过后来想想,还是写一下咯,抽出其中大家关心的内容,写出其中常用的使用方法,可以省去大家的时间,在需要深入的时候再去详细查看例子,也当是学习笔记吧:)(BTW:我也是边学习边写的,因此,写的文章中难免有不足或失误的地方,希望各位朋友指正出来咯,大家一起提高~本文中基本上只会写出最简单常用的功能,复杂的数据操作请参考微软详细的例程)

一.数据源

说到绘制图表,可能很多人的第一反应除了是图表呈现的方式外,更关心的便是数据如何添加,记得在很多年前,哪时要绘制一个数据曲线图,一直找不到好的呈现方式,后来使用了 **SVG** 的绘图模式,不过在添加数据的时候可谓吃尽了苦头,毕竟,**SVG** 只是一种描述语言,要动态的实现一个图表的绘制,是非常困难的.对于微软的图表控件,数据添加是一件很简单的方式,它支持多种数据添加方式,如:

可以在图表的设计界面,在属性设置窗口中的 **Series** 属性下的 **Points** 中添加需要的数据.

可以在图表的设计界面中,在属性中绑定一个数据源.

可以在后台代码中,动态添加数据.

可以在后台代码中设置一个或多个数据源,直接绑定到图表中.

在此处,我只着重讲解一下第 3,4 两点.对于第 3 点,相对来说比较简单,在后台代码中,找到要添加代码的 Series,通过它下面 Points 的 Add、AddXY、AddY 等方法,即可以实现数据的添加.例如:

```
Codedouble t;
```

```
for(t = 0; t <= (2.5 * Math.PI); t += Math.PI/6)
{
double ch1 = Math.Sin(t);
double ch2 = Math.Sin(t-Math.PI/2);
Chart1.Series["Channel 1"].Points.AddXY(t, ch1);
Chart1.Series["Channel 2"].Points.AddXY(t, ch2);
}
```

(注: 代码摘自微软的例子, 上例中, Chart1 为图表的名字, Channel 1、Channel 2 分别表示两个 Series 数据序列)

二.绑定数据

先来看看图表控件都支持什么类型的数据绑定, 根据官方文档的说法, 只要是实现了 **IEnumerable** 接口的数据类型, 都可以绑定到图表控件中, 例如: **DataView**, **DataReader**, **DataSet**, **DataRow**, **DataColumn**, **Array**, **List**, **SqlCommand**, **OleDbCommand**, **SqlDataAdapter**, 及 **OleDbDataAdapter** 对象。

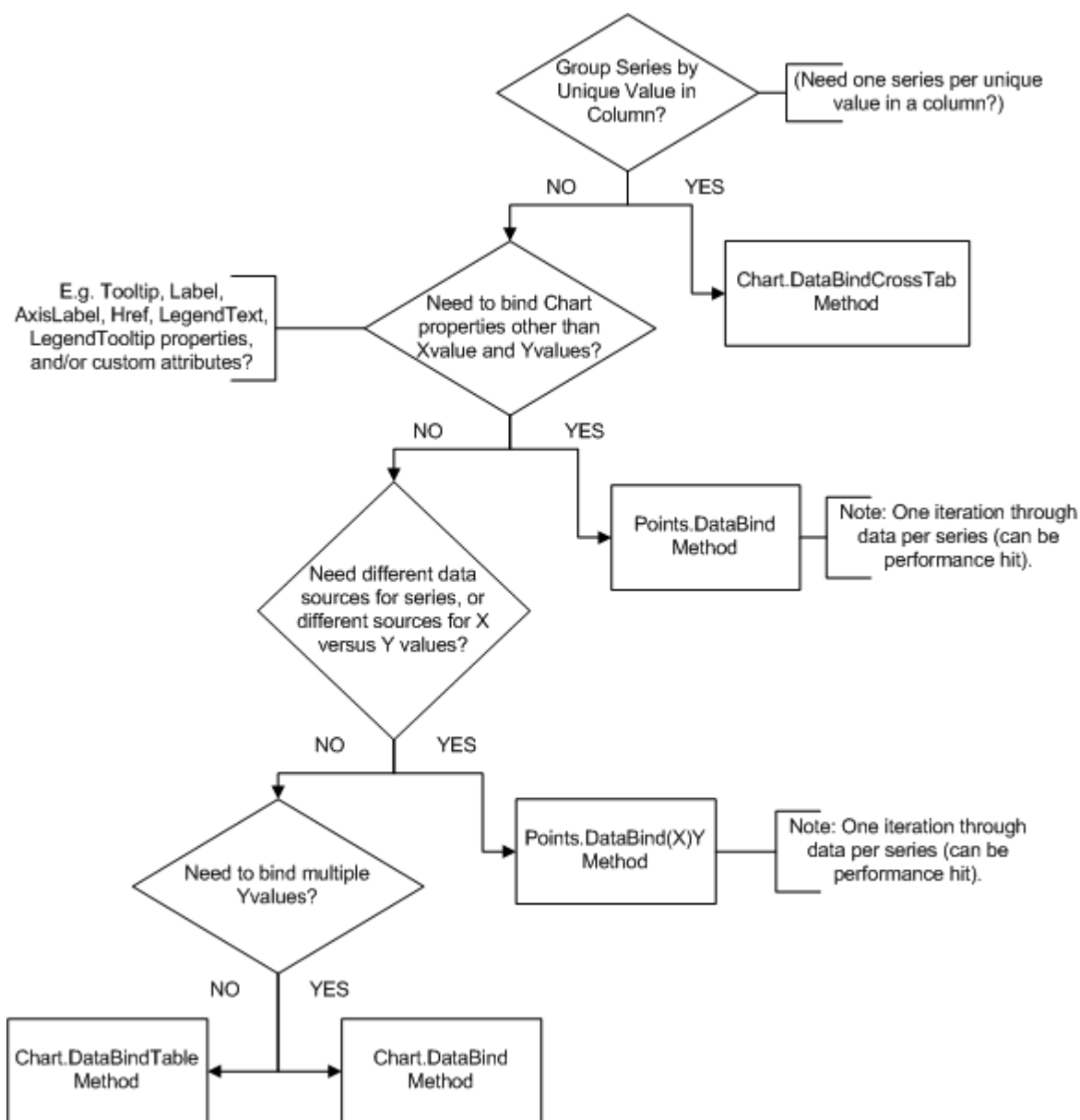
对于开发来说, 最常用的还是 **DataView**、**DataReader**、**DataSet**、**DataRow**、**Array**、**List** 这几种类型了, 有几点需要注意一下:

图表控件支持多数据源的绑定, 例如: X 轴绑定一个数据集 **ds1**, Y 轴可以绑定另一个数据集 **ds2**, 当然也可以是 X 轴绑定到一个 **List** 数据对象, Y 轴绑定到一个 **DataView** 对象等等。

图表控件的绑定方式一般有两种, 常规绑定及交叉表的绑定。

图表控件的 Y 轴数据, 支持一次绑定多个值, 以绘制时间、区域、使用量等之类的图形。

绑定数据的流程如下:



大意是，绑定数据里面是否有分组数据需要绑定，如果有，则调用交叉表绑定的方法。否则判断是否时绑定 X 轴和 Y 轴（包括标签、超链接、图例文字等自定义属性），如果是，则调用 **Points.DataBind** 方法进行绑定操作。再判断是否有不同的 X 轴或 Y 轴数据，如果有，则分别调用 X,Y 轴的绑定方法 **Points.DataBindX**, **Points.DataBindY** 进行数据绑定。最后，再判断是否需要进行多个 Y 轴值的绑定。

下面分别对几种数据绑定的方法进行一下说明：

1.绑定一张数据表

绑定一张数据表，例如绑定一张普通的数据表,表数据如下：

REPS ID Name RegionID Sales

1 Aaron 1 10440

2 Larry 2 17772

3 Andrew 3 23880

4 Mary 1 7663

5 Sally 4 21773
6 Nguyen 2 32294
7 Francis 4 11983
8 Jerry 3 14991

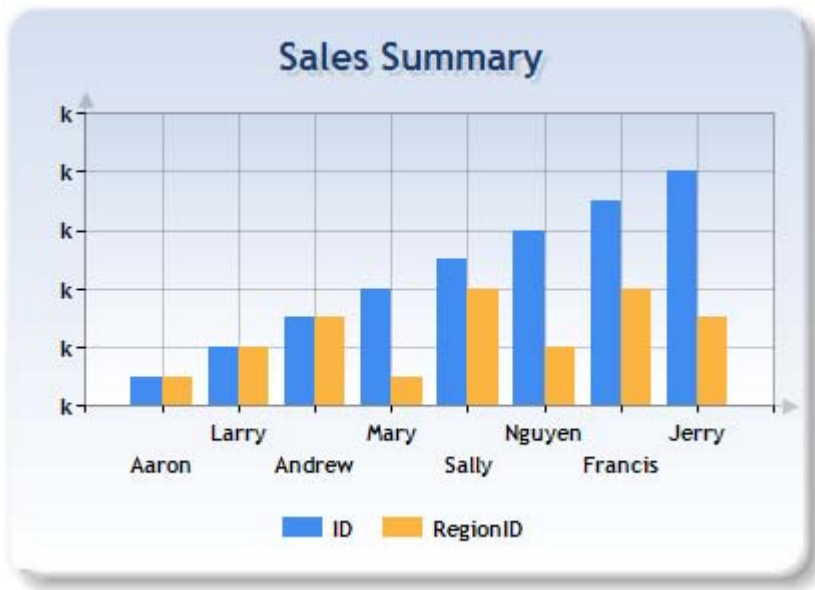
绑定方法的代码:

```
Codestring mySelectQuery="SELECT Name, Sales FROM REPS;";  
OleDbConnection myConnection = new OleDbConnection(myConnectionString);  
OleDbCommand myCommand = new OleDbCommand(mySelectQuery, myConnection);  
myCommand.Connection.Open();  
OleDbDataReader myReader =  
myCommand.ExecuteReader(CommandBehavior.CloseConnection);  
Chart1.DataBindTable(myReader, "Name");  
myReader.Close();myConnection.Close();
```

因为数据源中只有两列 **Name** 和 **Sales**,因此在调用 **Chart1.DataBindTable** 方法的时候,告诉了图表 X 轴的名称为 **Name**,因此自动将 **Sales** 设置为 Y 轴的数据了.生成的图形如下:



如果修改一下 **Sql** 语句为:**SELECT Name,ID,RegionID FROM REPS**;其它都不变化,再看看图表是如何处理另外两个字段的,生成的图表如下:



此时,图表自动将 ID,RegionID 字段当成了 Y 值,生成了两个 Series, 因此每个用户都有两个值,生成的图形也有两个柱状图。

上面是一种动态绑定的方式,有人可能会问了,如果我想确定我要绑定的列,比如:X 轴绑定某个字段,Y 轴绑定某个字段如何操作呢?对于这种绑定,有几种方法可以实现。

第一种:这种方法可能是很常见的,在原来的.NET 编程中出现的机率非常之高,方法如下:

Code

```
// 设置数据源,myDv 是一个取出数据集的 DataView
chart1.DataSource = myDv;
// 分别设置图表的 X 值和 Y 值
chart1.Series["Series1"].XValueMember = "Name";
chart1.Series["Series1"].YValueMembers = "Sales";
// 绑定设置的数据
chart1.DataBind();
```

第二种:即直接调用点的绑定方法

Code//myReader 为取得的 DataReader 对象

```
Chart1.Series["Series1"].Points.DataBindXY(myReader, "Name", myReader, "Sales");
```

第三种:调用 DataBind 的方法实现

```
CodeChart1.Series["Series1"].Points.DataBind(myReader, "Name", "Sales", "");
```

上面几种方法得到的效果都是一样的.当然了,上面三种方法需要自己建立 **Series**, 要显示两个柱状图, 像上面的例子中的图形, 那么得手动建立两个 **Series**, 然后分别进行上面的绑定操作。

此处说一下 **Label** 和 **Tooltip** 的绑定方式,要在绑定的图表中显示标签(**Label**)及提示(**Tooltip**),可以在绑定的时候,设置绑定的属性.对于上面的第一、二种方法, 可以调用如下的方法来设置 **Label** 和 **Tooltip**; 例如, 两个数据序列名称分别为 **Series1** 和 **Series2**, 设置代码如下:

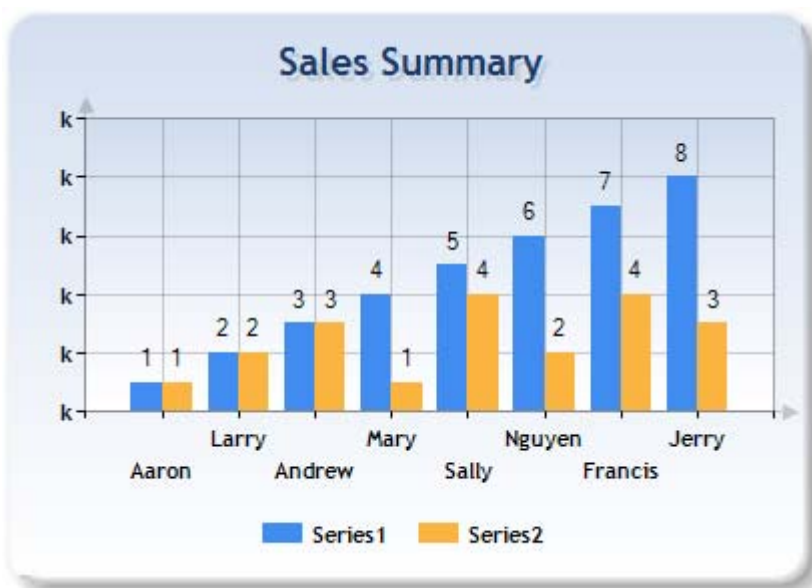
```
CodeChart1.Series["Series1"].Label = "#VAL";  
Chart1.Series["Series1"].Points.DataBind(myDs.Tables[0].DefaultView, "Name", "ID", "");  
Chart1.Series["Series2"].Label = "#VAL";  
Chart1.Series["Series2"].Points.DataBind(myDs.Tables[0].DefaultView, "Name",  
"RegionID", "");
```

其中的#VAL 是 **Label** 和 **Tooltip** 的通配符,表示取默认 Y 轴变量的意思。具体的详细操作可以参考例程: **Chart Features/Labels** 下面的内容。以后有空我也写一篇 **Label** 的和其它变量的设置吧。

对于第三种调用的绑定则稍微不同, 如下:

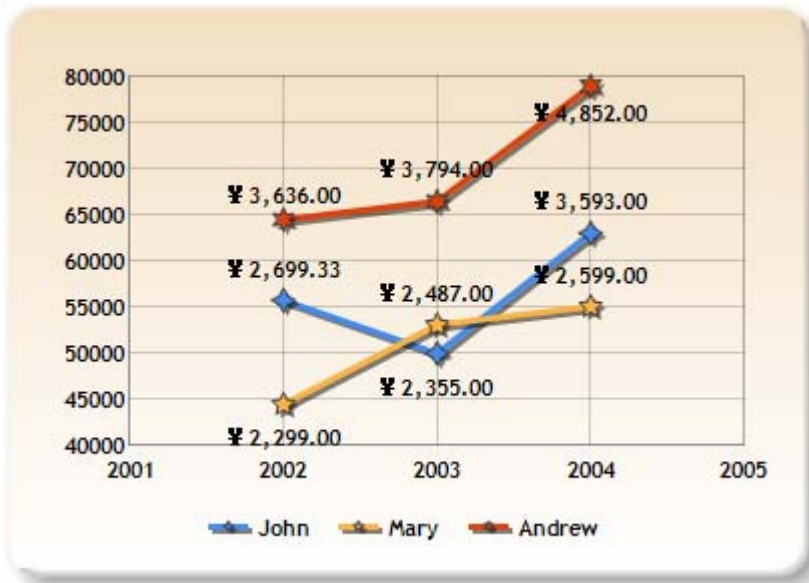
```
CodeChart1.Series["Series1"].Points.DataBind(myDs.Tables[0].DefaultView, "Name", "ID",  
"Label=ID,ToolTip=RegionID");  
Chart1.Series["Series2"].Points.DataBind(myDs.Tables[0].DefaultView, "Name",  
"RegionID", "Label=RegionID,ToolTip=ID");
```

其实就是利用第三个属性, 通过格式化的字符串来设置绑定属性, **Label** 表示标签, **ToolTip** 表示提示信息, **Url** 表示超链接等等。最后生成的图片如下:



2. 绑定一个交叉表

微软的图表控件提供了一个交叉表数据的绑定方法 `DataBindCrossTable`，它可以根据数据动态的生成数据序列（`Series`），借用官方的例子，数据库的表数据如下：



REPSALES Name Year Sales Commissions

John 2002 55676.55 2699.33

Mary 2002 44333 2299

Andrew 2002 64455.4 3636

John 2003 49884 2355

Mary 2003 52994 2487

Andrew 2003 66449 3794

John 2004 62994 3593

Mary 2004 54993 2599

Andrew 2004 78993 4852

我们先看看 `DataBindCrossTable` 的参数，它有两个重载方法，分别是：

```
public void DataBindCrossTable (  
    IEnumerable dataSource,  
    string seriesGroupByField,  
    string xField,  
    string yFields,  
    string otherFields,  
    PointSortOrder sortingOrder  
)
```

以及

```
public void DataBindCrossTable (  
    IEnumerable dataSource,  
    string seriesGroupByField,  
    string xField,  
    string yFields,  
    string otherFields  
)
```

每个参数的含义如下：

dataSource

要绑定的数据源.

seriesGroupByField

要分组统计的数据字段名称，例如按姓名、日期等.

xField

X 轴绑定的字段名称.

yFields

Y 轴绑定的字段名称，如果需要绑定多个字段，则用逗号将字段名分开.

otherFields

其它数据字段属性，就是上面刚讲的，标签、提示、Url 等属性.

sortingOrder

设置数据是正确还是逆序排列.

此时要以统计每个用户的年销售曲线，那么分组统计的字段名应该设置为 **Name**，如下：

```
CodeChart1.DataBindCrossTable(  
    myReader,  
    "Name",  
    "Year" ,  
    "Sales",  
    "Label=Commissions{C}");
```

用如上的方法绑定，生成的图形如下：

相反，如果要统计用户每年的曲线，则将字段反转一下即可，如下：

```
CodeChart1.DataBindCrossTable(
myReader,
"Year",
"Name",
"Sales",
"Label=Commissions{C}");
```

生成的曲线图如下：



这次就先说到这里咯，说几个需要注意的地方：

在进行 Y 轴数据绑定的时候，如果要绑定多个字段，默认情况会出错，那是因为需要设置 Y 轴的可保存值数量，设置为你需要保存的数量即可，设置的地点在：**Series-》YValuesPrePoint**，设置为你需要显示的个数即可。

在进行 **DataTable** 绑定的时候，**Label**、**ToolTip** 等属性的字段格式化比较困难（**otherFields** 属性），我试了半天，也就试出了一次只能绑定一个字段，因为是和数据集绑定，如果要在标签上增加文字的话，可以使用：**:Field{xxxx#xxxx}**，其中#会替换为相应的文字，例如：**Field** 的值为 **45**，那么最后的呈现的结果就是：**xxxx45XXXX**。

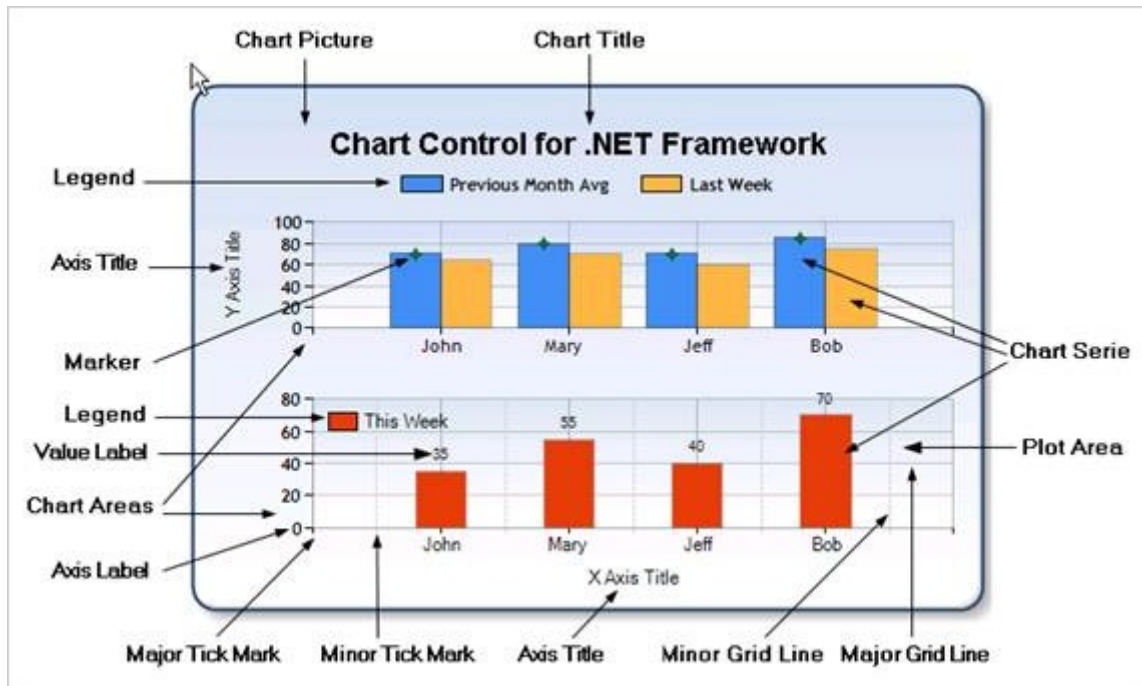
BTW:有很多东西我自己也在研究中，因此说得不是很完善，希望大家一起研究吧~下次再研究一下数据操作方面的东东以及标签等的显示。

MSChart图表控件的一些使用

最近使用 MSChart 做了几张图表，微软提供的例子中对这个图表控件的使用已经说的比较详细了，我这里记录一些需要注意的。

1. Chart 图表的元素

要使用该图表首先要了解该图表的元素组成，就是下面这张图了，



图表主要包含：

Annotations （图形注解集合）
ChartAreas （图表区域集合）
Legends （图例集合）
Series 图表序列集合即图表数据对象集合
Title （图标的标题集合）

其他大家可以参考 Samples 中的 Char Features 一节。

2. 如何在 ASP.NET MVC 中使用 Chart 控件

2.1. 需要在“<system.web><pages><controls>”中添加如下：

```
<add tagPrefix="asp" namespace="System.Web.UI.DataVisualization.Charting"
assembly="System.Web.DataVisualization,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
```

2.2. 在“<httpHandlers>”中添加如下部分：

```
<add path="ChartImg.axd" verb="GET,HEAD"
type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
System.Web.DataVisualization, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" validate="false"/>
```

2.3. 有了上面的配置就可以测试了：

2.3.1.View 中添加：

```
<asp:chart id="Chart1" runat="server" Height="296px" Width="412px"
Palette="BrightPastel" imagetype="Png"
BorderDashStyle="Solid" BackSecondaryColor="White" BackGradientStyle="TopBottom"
BorderWidth="2" backcolor="#D3DFF0"
BorderColor="26, 59, 105">
<Titles>
<asp:Title Text="With datasource in code behind" />
</Titles>
<legends>
<asp:Legend IsTextAutoFit="False" Name="Default" BackColor="Transparent"
Font="Trebuchet MS, 8.25pt, style=Bold">
</asp:Legend></legends>
<borderskin skinstyle="Emboss"></borderskin>
<series>
<asp:Series Name="Column" BorderColor="180, 26, 59, 105">
</asp:Series>
</series>
<chartareas>
<asp:ChartArea Name="ChartArea1" BorderColor="64, 64, 64, 64"
BorderDashStyle="Solid" BackSecondaryColor="White"
BackColor="64, 165, 191, 228" ShadowColor="Transparent"
BackGradientStyle="TopBottom">
<area3dstyle Rotation="10" perspective="10" Inclination="15"
IsRightAngleAxes="False" wallwidth="0"
IsClustered="False">
</area3dstyle>
<axisy linecolor="64, 64, 64, 64">
<labelstyle font="Trebuchet MS, 8.25pt, style=Bold" />
<majorgrid linecolor="64, 64, 64, 64" />
</axisy>
<axisx linecolor="64, 64, 64, 64">
<labelstyle font="Trebuchet MS, 8.25pt, style=Bold" />
<majorgrid linecolor="64, 64, 64, 64" />
</axisx>
</asp:ChartArea>
</chartareas>
</asp:chart>
```

2.3.2. index.aspx.cs 中添加

```
protected void Page_Load(object sender, System.EventArgs e)
{
    foreach (int value in (List<int>)this.ViewData["Chart"])
    {
        this.Chart1.Series["Column"].Points.Add(value);
    }
}
```

2.3.3. Controller 中:

```
public ActionResult Index()
{
    ViewData["Title"] = "Home Page";
    ViewData["Message"] = "Welcome to ASP.NET MVC!";
    List<int> chartList = new List<int>();
    chartList.Add(1);
    chartList.Add(2);
    chartList.Add(6);
    chartList.Add(5);
    chartList.Add(4);
    ViewData["Chart"] = chartList;
    return View();
}
```

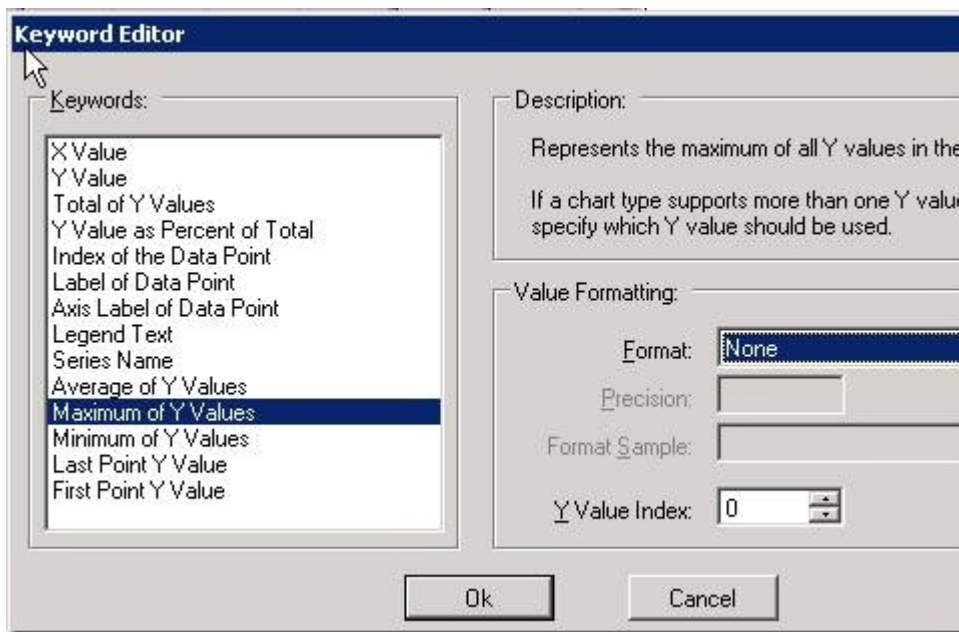
2.3.4. 添加 System.Web.ui.DataVisualization 引用

3. Tooltip 的使用

ToolTip 用于在各个关键点，如：标签、图形关键点、标题等当鼠标移动上去的时候，提示用户一些相关的详细或说明信息。我们可以通过如下代码来设置：

```
Chart1.Series["Series1"].ToolTip = "#VALY";
```

这里的主要目的是说下，系统已经提供了很多 Keyword 供我们使用，如下图：



4. “Error executing child request for ChartImg.axd”的处理

```
<add name="ChartImageHandler" preCondition="integratedMode" verb="POST, GET, HEAD"
path="ChartImg.axd"
和
```

```
<add
path="ChartImg.axd" verb="POST, GET, HEAD" type="System.Web.UI.DataVisualization.Ch
arting.ChartHttpHandler,
```

这两个地方都加上 POST 就好了

5. 图表的呈现

Chartk 控件的 ImageStorageMode 属性是决定图片的存储方式的。

5.1. 当 Chart.ImageStorageMode=UseHttpHandler 时

使用的是 web.config 中的配置。即指定由 HTTP 处理程序来存储图形文件，web.config 的配置如下：

```
<add key="ChartImageHandler"
value="storage=file;timeout=20;dir=c:\TempImageFiles\;deleteAfterServicing=false;
webDevServerUseConfigSettings=false" />
```

5.1.1. 其中 c:\TempImageFiles\就是 HTTP 处理程序来存储图形文件的目录。

5.1.2. `deleteAfterServicing`, 在图片被浏览器成功下载后, 确定服务器上图片是否删除, 默认值为 `false`。如果不指定 `deleteAfterServicing`, 图片是不会被存储为文件的, 应当只是暂存在内存中。

5.1.3. `webDevServerUseConfigSettings`, 当在 VS 开发服务器中, 即运行在完全信任级别 (`full trust level`) 环境时, 决定生成的图片是否被保存在内存中。

5.1.4 `storage` 存储 3 种方式: `storage=file`, `storage=session`, `storage=memory`。

5.2. 当 `Chart.ImageStorageMode= UseImageLocation` 时

图表会存储在我们指定的文件夹中, 我们也可以指定图片生成的命名规则。

5.2.1. 其中 `Chart.ImageType` 表示生成图形文件的格式, 可以是 `BMG`、`JPEG`、`PNG`、`EMF`。

5.2.2. `Chart.RenderType`

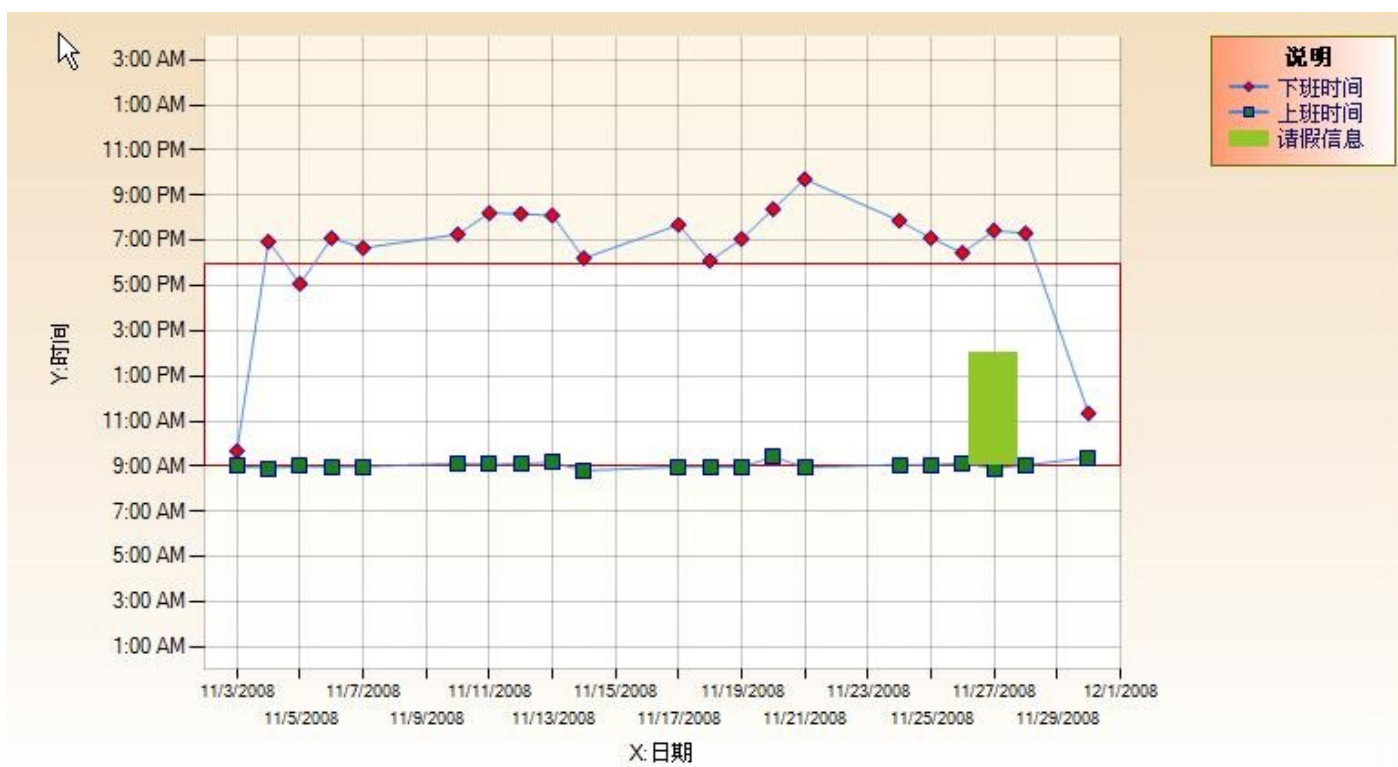
此属性决定 `MSChart` 生成的图形以何种方式送到客户端, 有三种:

- `ImageTag`, 图形在服务端保存为临时文件, 并将临时文件的 URL 作为 HTML 中 `<IMG>` 标签的 `SRC` 属性值。而此临时文件的 URL 规则及临时文件如何生成可由 `ImageLocation` 属性确定。
- `BinaryStreaming`, 不在服务端生成图形文件, `<IMG>` 标签的 `SRC` 属性值将指向另一个负责生成图形的 `.aspx` 页面。
- `ImageMap`, 不显示实际的图形, 仅创建图片热区 (`image map`)。

默认值是 `ImageTag`, 就是会在服务端生成临时文件。

5.2.3. 只要在服务端生成图形文件, 无论存储在哪个目录下, ASP.NET 帐户都需要拥有该目录的写入权限。

下面是我的一张图表:



微软Chart控件导学--从头开始

微软在.NET 3.5 的时候出了一个 Chart 控件,但是那时候要单独安装才可以使用,在.NET 4.0 中,微软将这个控件集成到了 Visual Studio 2010 中,现在可是大大的提高了.NET 做报表的能力,是一个非常有用的控件。呵呵,越有用的东西用起来就越不容易,这几天在研究这个东西,就把自己所学到的东西大家分享一下。

一. .NET 3.5 控件的下载安装。

该图表控件支持多种图表,如饼图,柱状图,曲线图,散点图,雷达图,面积图,股票图等,同时支持 Winform 和 Asp.net。 该控件只能用在.Net Framework 3.5 和 4.0 上。据说在.NET 2.0 上也是有的,但是要花钱,呵呵。

这个就不多说了,给一下下载的连接:

控件: Microsoft .NET Framework 3.5 的 Microsoft 图表控件(Microsoft Chart Controls for Microsoft .NET Framework 3.5)

包含英文版,中文版。上面提供的链接是中文版的,可以更改为英文版。

文档: Microsoft Chart Controls for .NET Framework Documentation

这个只有英文的,没找到中文的。只是些函数的 API,只要英文好一点很有用处。

这个据说安装后还要配制什么的,没有用过,自己也不太清楚,想用的可以去网上找些资料。

二. .NET 4.0 Chart 控件使用基础

新建一个 ASP.NET 的网站后，在工具栏中的 Data 分类中，就会出现一个 Chart 控件，和其他控件一样，你可以把它拖到页面上，就可以添加一个“图表”了。

生成的代码如下：

```
view plaincopy to clipboardprint?
<asp:Chart ID="Chart1" runat="server">

    <Series>

        <asp:Series Name="Series1">

            </asp:Series>

        </Series>

    <ChartAreas>

        <asp:ChartArea Name="ChartArea1">

            </asp:ChartArea>

        </ChartAreas>

</asp:Chart>
<asp:Chart ID="Chart1" runat="server">

    <Series>

        <asp:Series Name="Series1">

            </asp:Series>

        </Series>

    <ChartAreas>

        <asp:ChartArea Name="ChartArea1">

            </asp:ChartArea>

        </ChartAreas>

</asp:Chart>
```

2.1 Chart 控件组成

由上面的代码段可以清楚的看到，Chart 控件由 Series（数据列）和 ChartArea（成图区域）两部分组成（这两部分的命名是自己的理解，现在还没有在网上看到中文的命名）。这两部分都是可以有一个或者多个的，例如当一个“图表”中要画多条曲线的时候我们就可能会用到多个“数据列”，并且把多个数据列的 ChartArea 属性设置为指定的“成图区域”。当我们想在一个“图表”中分区域多形式的显示一种或多种数据的时候，我们就需要多个 ChartArea 了。（从 XML 元素的命名也是可以看出这一点的，要不然在<asp:Series>和<asp:ChartArea>前面就不会再有一层复数的 XML 标签了。）

对于简单的图表，我们只用默认的风格就足够了，所以可以不用对 ChartArea 进行太多的修改，只要在“数据列”中添加一定数量的点就可以成图了，相当简单。

2.2 向 Chart 中添加数据点

方法一：用 XML 在设计时直接添加数据点。

先找一个例子，下面的例子是 NBA 几个队伍胜利的场数据的“图表”，代码如下：

view plaincopy to clipboardprint?

```
<asp:Chart ID="Chart1" runat="server">
```

```
    <Series>
```

```
        <asp:Series Name="Series1" YValuesPerPoint="4">
```

```
            <Points>
```

```
                <asp:DataPoint AxisLabel="Celtics" YValues="17" />
```

```
                <asp:DataPoint AxisLabel="Lakers" YValues="15" />
```

```
                <asp:DataPoint AxisLabel="Bulls" YValues="6" />
```

```
                <asp:DataPoint AxisLabel="Spurs" YValues="4" />
```

```
                <asp:DataPoint AxisLabel="76ers" YValues="3" />
```

```
                <asp:DataPoint AxisLabel="Pistons" YValues="3" />
```

```
                <asp:DataPoint AxisLabel="Warriors" YValues="3" />
```

```
            </Points>
```

```
        </asp:Series>
```

```

</Series>

<ChartAreas>

    <asp:ChartArea Name="ChartArea1">

        </asp:ChartArea>

    </ChartAreas>

</asp:Chart>
    <asp:Chart ID="Chart1" runat="server">

        <Series>

            <asp:Series Name="Series1" YValuesPerPoint="4">

                <Points>

                    <asp:DataPoint AxisLabel="Celtics" YValues="17" />

                    <asp:DataPoint AxisLabel="Lakers" YValues="15" />

                    <asp:DataPoint AxisLabel="Bulls" YValues="6" />

                    <asp:DataPoint AxisLabel="Spurs" YValues="4" />

                    <asp:DataPoint AxisLabel="76ers" YValues="3" />

                    <asp:DataPoint AxisLabel="Pistons" YValues="3" />

                    <asp:DataPoint AxisLabel="Warriors" YValues="3" />

                </Points>

            </asp:Series>

        </Series>

    <ChartAreas>

        <asp:ChartArea Name="ChartArea1">

            </asp:ChartArea>

        </ChartAreas>

```

</asp:Chart>

生成的图表如下所示：

图表 1

对几个重要的属性解释如下：

AxisLabel：当 X 轴没有值时可以设置这个值，会在 X 轴上生成一些文字，效果就如上面所示。

XValue：这个更容易理解，一个点的 X 坐标值。

YValues：一般情况下只用一个 YValue 就可以生成图表了，但是在某些情况下要两个甚至多个值，多个值是只有当图表类型为 bubble, candlestick 和 stock 时才有意义。

方法二：在程序运行时动态添加数据点。

下面是代码段：

```
view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)
{
    System.Web.UI.DataVisualization.Charting.Chart chart1 =
        new System.Web.UI.DataVisualization.Charting.Chart();

    var serial1 = chart1.Series.Add("serial");

    serial1.Points.AddXY("Celtics", 17);

    serial1.Points.AddXY("Lakers", 15);

    serial1.Points.AddXY("Bulls", 6);

    serial1.Points.AddXY("Spurs", 4);

    serial1.Points.AddXY("76ers", 3);

    serial1.Points.AddXY("Pistons", 3);
```

```

        serial1.Points.AddXY("Warriors", 3);

        var chartarea = chart1.ChartAreas.Add("default");

        serial1.ChartArea = "default";

        form1.Controls.Add(chart1);

        chart1.Visible = true;
    }

    protected void Page_Load(object sender, EventArgs e)
    {

        System.Web.UI.DataVisualization.Charting.Chart chart1 =
            new System.Web.UI.DataVisualization.Charting.Chart();

        var serial1 = chart1.Series.Add("serial");

        serial1.Points.AddXY("Celtics", 17);

        serial1.Points.AddXY("Lakers", 15);

        serial1.Points.AddXY("Bulls", 6);

        serial1.Points.AddXY("Spurs", 4);

        serial1.Points.AddXY("76ers", 3);

        serial1.Points.AddXY("Pistons", 3);

        serial1.Points.AddXY("Warriors", 3);

        var chartarea = chart1.ChartAreas.Add("default");

        serial1.ChartArea = "default";

        form1.Controls.Add(chart1);

        chart1.Visible = true;
    }

```

生成的图形还是上面那个，就不多说了。和上面用 XML 生成图表结构是一样，但注意最后要加入 form1 中，否则无法显示。

上面的两个例子是比较极端的，在实际应用时最多的是两种方法的结合。一般情况下我们知道哪里要有一个报表，但是其中的数据是动态的，所以可以在设计时用 XML 生成一部分的图表（无数据的图表），然后在程序中填充数据。

有人会觉得用程序来添加点是一件很麻烦的事，事实上也并不是太麻烦，在实际应用时多是在循环中使用加点的动作，所以可以很快的生成大量的点。

方法三：数据绑定。

在很多的情况下，我们都面临着这样一种情况，那就是从数据库中读取数据然后再生成图表。更普遍的是我们将数组中的数据拿出来成图尽管可以用上面的方法实现，但是如果能让自己的程序更简单那么何乐而不为呢？

首先向页面中添加一个 Chart 图表：代码段如上面的代码段一所示。

然后我们连接 NorthWind 数据库（没有这个数据库可以去下一个，实在不想下自己编点数据也行），然后用 DataBindTable 来进行绑定。代码段如下所示：

```
view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)
{
    SqlConnection cn = new SqlConnection();

    cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial
Catalog=Northwind;Integrated Security=True";

    SqlCommand cmd=cn.CreateCommand();

    cmd.CommandText="SELECT CategoryName, COUNT(*) as ProductCount FROM Products p
INNER JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY CategoryName ORDER
BY CategoryName";

    cn.Open();

    SqlDataReader dr = cmd.ExecuteReader();
```

```

Chart1.DataBindTable(dr, "CategoryName");

dr.Close();

cn.Close();

}

protected void Page_Load(object sender, EventArgs e)

{

    SqlConnection cn = new SqlConnection();

    cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial
Catalog=Northwind;Integrated Security=True";

    SqlCommand cmd=cn.CreateCommand();

    cmd.CommandText="SELECT CategoryName, COUNT(*) as ProductCount FROM
Products p INNER JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY
CategoryName ORDER BY CategoryName";

    cn.Open();

    SqlDataReader dr = cmd.ExecuteReader();

    Chart1.DataBindTable(dr, "CategoryName");

    dr.Close();

    cn.Close();

}

```

这样就生成了下面的这个表：（数据和文字是正常的，但是让我不明白的是颜色是橙色的，以前的默认色不是蓝色嘛！！）。

上面的 DataBindTable 函数有两个重载，具体的情况可以参照文档，第一个变量不一定要是 SqlDataReader，只要实现了 IEnumerable 接口就行了。对于数组元素的绑定，在下一节的教程中再做详述。

还有一种绑定方式就是：

```
Chart1.Series("Series1").XValueMember = "CategoryName" ;
```

```
Chart1.Series("Series1").YValueMembers = "ProductCount" ;
```

```
Chart1.DataSource = myReader ;
```

```
Chart1.DataBind();
```

用 DataSource 和 DataBindTable 的不同就在于用 DataSource 还要进一步的指定 X 和 Y 轴所对应的值。

现在我们就可以很轻松的做出一个符合自己心意的图表了吧。

想说这个是自己原创的有点不太不好意思，想说这个是翻译自己又做了好多自己的工作在里面，而且以前是针对 .NET 3.5 的，但是 .NET 3.5 和 4.0 对本文没有太大的影响，把参考的文章也放出来：

本文来自 CSDN 博客，转载请标明出处：
<http://blog.csdn.net/grlsir/archive/2010/01/10/5167178.aspxpx>

微软 Chart 控件导学--数据绑定

一个图表如果对外观没有太大的要求（本来图表的外观就是在很大程度上定制了的，并且微软也提供了很多的样式，对于一般的应用足够了），只要把正确的数据输入就一切 OK 了。从根本上讲，所有的数据的输入都是一个个的添加点的封装。只要掌握好这一种方法就一切问题都可以解决了。但是微软也提供了更加方便的方法，就是数据绑定。数据绑定内部一定也是一个点的添加的（呵呵，想想也是很容易的，对所有的实现 IEnumerator 接口的都可以用 foreach 语句，这样就可以一个一个的添加点了，我想应该是这样的）。

下面我们就用循环来把上一节用 DataBindTable 实现的那个图表再实现一遍。用起来也不是很容易，主要代码如下：

```
view plaincopy to clipboardprint?
protected void Page_Load(object sender, EventArgs e)

{
```

```

SqlConnection cn = new SqlConnection();

cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial Catalog=Northwind;Integrated
Security=True";

SqlCommand cmd = cn.CreateCommand();

cmd.CommandText = "SELECT CategoryName, COUNT(*) as ProductCount FROM Products p INNER
JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY CategoryName ORDER BY CategoryName";

cn.Open();

SqlDataReader dr = cmd.ExecuteReader();

//Chart1.DataBindTable(dr, "CategoryName");

while (dr.Read())
{
    Chart1.Series[0].Points.AddXY(Convert.ToString(dr["CategoryName"]),
Convert.ToInt32(dr["ProductCount"]));
}

dr.Close();

cn.Close();
}

protected void Page_Load(object sender, EventArgs e)
{
    SqlConnection cn = new SqlConnection();

    cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial
Catalog=Northwind;Integrated Security=True";

    SqlCommand cmd = cn.CreateCommand();

    cmd.CommandText = "SELECT CategoryName, COUNT(*) as ProductCount FROM Products p
INNER JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY CategoryName ORDER BY
CategoryName";

```

```

cn.Open();

SqlDataReader dr = cmd.ExecuteReader();

//Chart1.DataBindTable(dr, "CategoryName");

while (dr.Read())

{

    Chart1.Series[0].Points.AddXY(Convert.ToString(dr["CategoryName"]),
Convert.ToInt32(dr["ProductCount"]));

}

dr.Close();

cn.Close();

}

```

生成的图表如下：

图片 1（呵呵，上一节的图片）

这样就是用一个循环遍历 `dr` 这个实现了 `IEnumerator` 的对象，实现了与第一节中相同的功能（上文中注释的地方就是上一节所用的方法，用数据绑定的方法数据量会少一些）。

一个点一个点的添加是最为普遍的方式（但未必是最常用的方式）。再有时我们会到这种情况，那就是 `X` 轴和 `Y` 轴的数据都存到了一个数组中，对于这种情况 `Chart` 控件也提供了一个好用的方法来添加数据。这个方法就是 `DataBindXY`，`DataBindXY` 有两个重载，一个是 `DataPointCollection.DataBindXY (IEnumerable, IEnumerable[])`。这个很好理解，第二个参数为什么是一个数据也和 `Point` 的 `Y` 值为什么会是数组的理由是一样的。下面是一个例子，还是用的 `NBA` 的那个数据。

view plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```

{

    List<string> teams = new List<string>();

    List<int> score = new List<int>();

```

```

teams.Add("Celtics"); score.Add(17);

teams.Add("Lakers"); score.Add(15);

teams.Add("Bulls"); score.Add(6);

teams.Add("Spurs"); score.Add(4);

teams.Add("76ers"); score.Add(3);

teams.Add("Pistons"); score.Add(3);

teams.Add("Warriors"); score.Add(3);

Chart1.Series["Series1"].Points.DataBindXY(teams, score);
}

protected void Page_Load(object sender, EventArgs e)

{

    List<string> teams = new List<string>();

    List<int> score = new List<int>();

    teams.Add("Celtics"); score.Add(17);

    teams.Add("Lakers"); score.Add(15);

    teams.Add("Bulls"); score.Add(6);

    teams.Add("Spurs"); score.Add(4);

    teams.Add("76ers"); score.Add(3);

    teams.Add("Pistons"); score.Add(3);

    teams.Add("Warriors"); score.Add(3);

    Chart1.Series["Series1"].Points.DataBindXY(teams, score);

}

```

图片还是再贴一遍吧：

图表 2（还是上一节的图片）

第二个重载中 `DataPointCollection.DataBindXY (IEnumerable, String, IEnumerable, String)`，这个就和在用 `DataBind` 时的情况有点像了，也只给出一个例子就可以了。

+ expand sourceview plaincopy to clipboardprint?

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    SqlConnection cn = new SqlConnection();
```

```
    cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=True";
```

```
    SqlCommand cmd = cn.CreateCommand();
```

```
    cmd.CommandText = "SELECT CategoryName, COUNT(*) as ProductCount FROM Products p INNER JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY CategoryName ORDER BY CategoryName";
```

```
    cn.Open();
```

```
    SqlDataReader dr = cmd.ExecuteReader();
```

```
    //Chart1.DataBindTable(dr, "CategoryName");
```

```
    //while (dr.Read())
```

```
    //{
```

```
    //                                Chart1.Series[0].Points.AddXY(Convert.ToString(dr["CategoryName"]), Convert.ToInt32(dr["ProductCount"]));
```

```
    //}
```

```
    Chart1.Series[0].Points.DataBindXY(dr, "CategoryName", dr, "ProductCount");
```

```
    dr.Close();
```

```
    cn.Close();
```

```
}
```

```
    protected void Page_Load(object sender, EventArgs e)
```

```

{

    SqlConnection cn = new SqlConnection();

    cn.ConnectionString = @"Data Source=GERENLEI\SQLEXPRESS;Initial
Catalog=Northwind;Integrated Security=True";

    SqlCommand cmd = cn.CreateCommand();

    cmd.CommandText = "SELECT CategoryName, COUNT(*) as ProductCount FROM Products p
INNER JOIN Categories c ON c.CategoryID = p.CategoryID GROUP BY CategoryName ORDER BY
CategoryName";

    cn.Open();

    SqlDataReader dr = cmd.ExecuteReader();

    //Chart1.DataBindTable(dr, "CategoryName");

    //while (dr.Read())

    //{

        //
        Chart1.Series[0].Points.AddXY(Convert.ToString(dr["CategoryName"]),
Convert.ToInt32(dr["ProductCount"]));

    //}

    Chart1.Series[0].Points.DataBindXY(dr, "CategoryName", dr, "ProductCount");

    dr.Close();

    cn.Close();

}

```

呵呵，效果不用贴了吧。。。代码都大部分是上面复制的。。。

DataBind 就不用多说了吧，第一节中有例子。而且微软也是靠这个来赚人气的。说句实在话，这个自己其实也不是太懂，反正能用就行了。

最后一个数据绑定用的就是 **DataBindTable** 函数了，从这个名字上就能看到这个是很来做表格数据（**Tabular Data**）绑定的。第一节中也用过，就不多说了。

现在的所有的数据绑定我们都是用程序代码来实现的，微软还提供了一种在设计时不用写代码就可以实现数据绑定的控件：“数据源”，就是下图所示的样子，点击 Config Data Source，就会出现提示，然后一步一步的进行也是可以的。

好，数据绑定大致就这些了吧。

Mschart学习(MSchart介绍)

1.什么是 MSChart

Chart: Microsoft Chart Controls for Microsoft .NET Framework 3.5

- 适用于.NET Framework 3.5 SP1 的 ASP.NET 和 Windows Form 图标控制项
- 开发商：2007 年 Dundas 开发出收费的 Chart 控件 2008/9/8 微软发布免费的 MSChart 控件
- Support Service: system :win2003 sp2 ,win2008,windows vista,winxp sp3 .net Framework: .NET Framework 3.5 SP1(VS2008)
- 处理器：400 MHz Pentium(最低)；1GHz Pentium（建议） RAM：96 MB（最小）256MB（建议） Hard： 500MB 可用磁盘空间
- 显卡： 800*600、256 色（小） 1024*768、256 色

2. MSChart 安装

- 安装完成VS2008（.NET Framework 3.5 SP1）[下载地址](#)
 - 1) dotnetfx35setup.exe
 - 2) MSChart.exe
 - 3) MSChart_VisualStudioAddOn.exe
 - 4) MSChartLP_chs.exe
- 安装顺序：

(1) dotnetfx35setup.exe

(2) MSChart_VisualStudioAddOn.exe

(3) MSChartLP_chs.exe

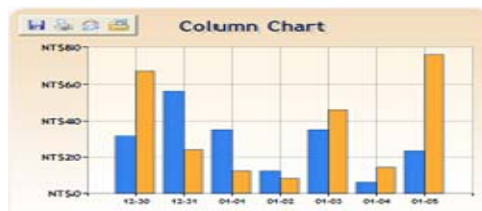
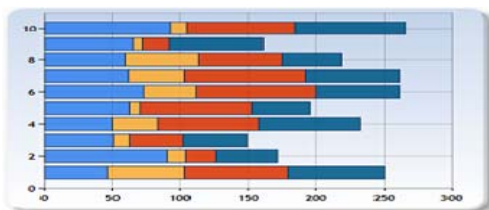
(4) MSChart.exe

打开 VS2008 在工具中新增了 Data Chart 控件：



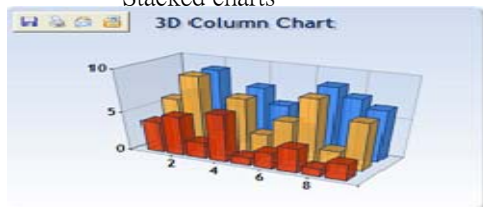
MSChart 图表类型

1.Bar and Column Charts 3D or 2D Bar or Column Charts like this page:



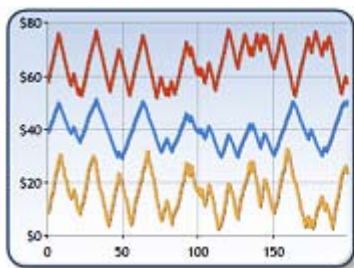
Stacked charts

Bar and Column

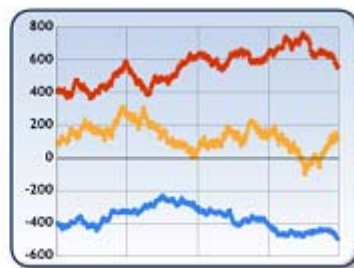


3D Bar and Column

2.Line Charts

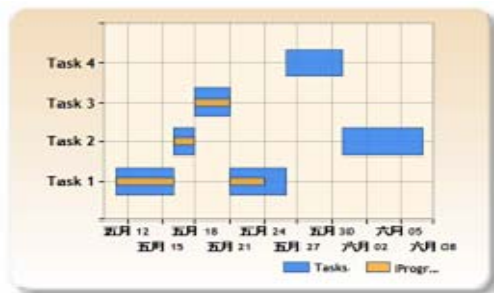


2D Line Chart

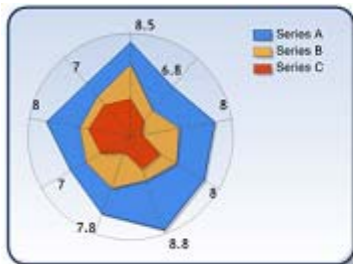


2D Fast Line Chart

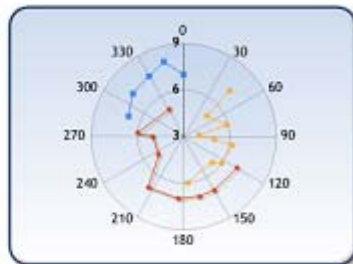
3. Range Charts



4. Circular charts

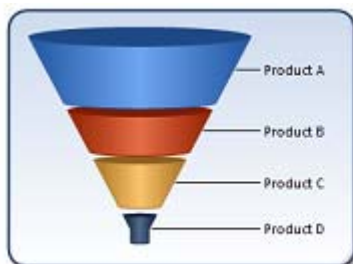


2D Radar Area Chart

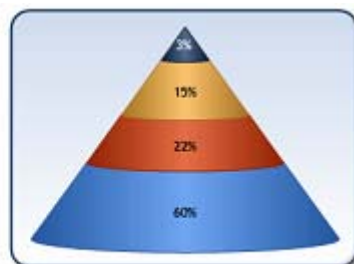


2D Polar Line Chart

5. Accumulation charts

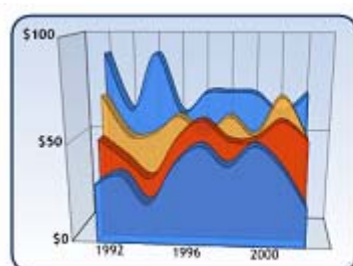


3D Funnel Chart with Point Gaps

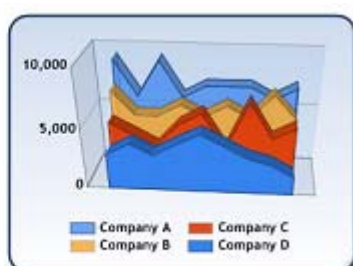


3D Pyramid Chart

6. Area Chart



3D Spline Area



3D Area Chart

7. Pie and Doughnut charts

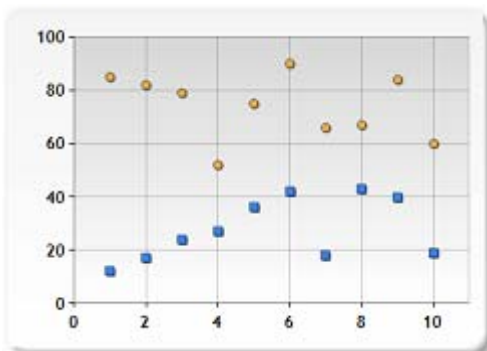


2D Doughnut Chart

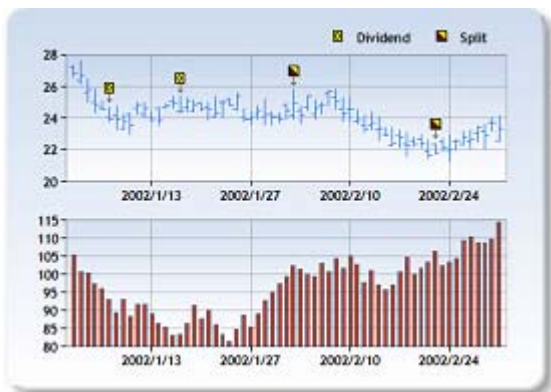


2D Grouped Pie Chart

8. Point charts



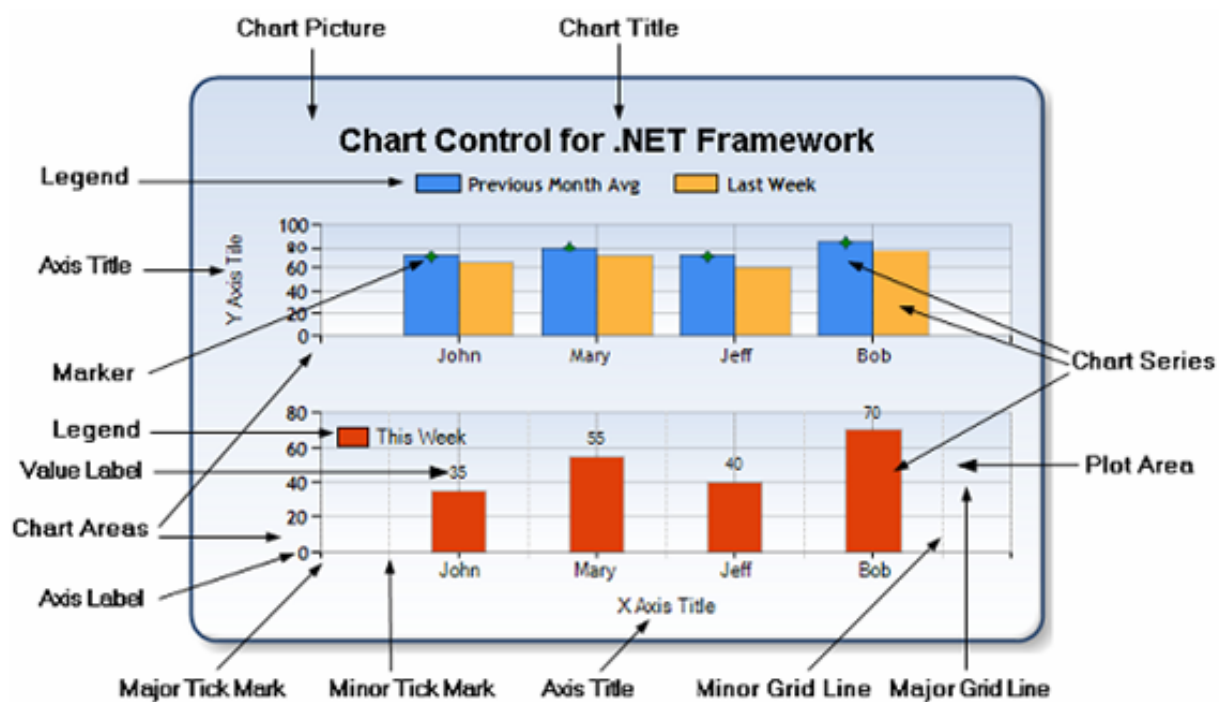
9. Advanced Financial charts



10. Combinational Charts

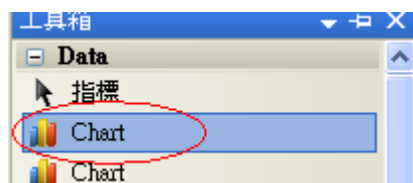


3.MSChart 属性介绍



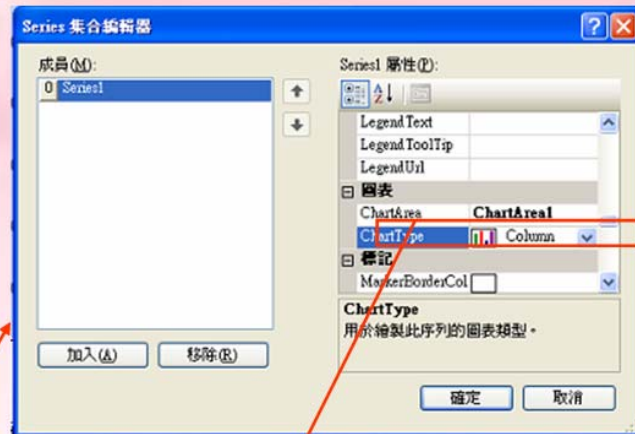
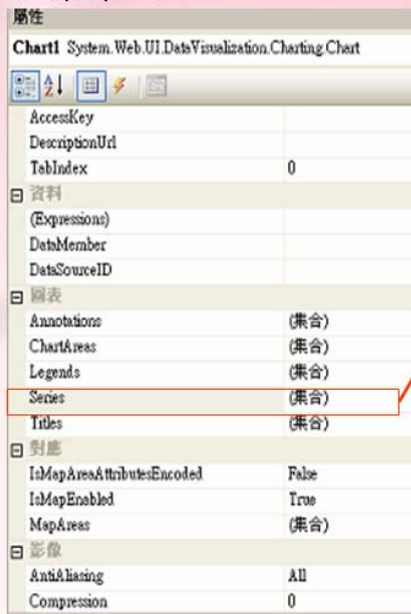
这些属性在添加 chart 的时候会有。

开启 VS2008 · 新建网站 ChartTest 打开工具列:



点击 Chart 在页面中添加 Chart 图形，在属性栏中选择 Chart 类型

属性修改：



4.MSChart inVS2008 实现

- 查询数据结果对应到 Chart 的 XY 轴

代码

```
1 1) string strSql = "SELECT Top 5 empid, COUNT(*) AS lNum FROM log
where empid <> '' GROUP BY empid ORDER BY empid";
2 DataSet ds = SqlHelper.ExecuteDataset(strConnISPD, CommandType.
Text, strSql);
3 DataView dv = new DataView(ds.Tables[0]);
4 Chart1.Series["login"].Points.DataBindXY(dv, "empid", dv, "lNum
");
5 2) SqlCommand cd = new SqlCommand(strSql, conn);
6 SqlDataAdapter da = new SqlDataAdapter();
7 da.SelectCommand = cd;
8 DataSet ds = new DataSet();
9 da.Fill(ds, "log");
```

```
10    DataView dv = new DataView(ds.Tables[0]);
11    dv.Sort="empid";
12    Chart1.Series["login"].Points.DataBindXY(dv, "empid", dv, "lNum");
```

使用现有数组数据到 Chart 的 XY 轴

代码

```
1 double[] yValues = {12, 67, 45, double.NaN, 67, 89, 35, 12, 78, 54};

2 double[] xValues = {1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999};

3 Chart1.Series["Series1"].Points.DataBindXY(xValues, yValues);
```

有错误的地方欢迎大家拍砖,希望交流和共享。

微软图表控件 MsChart 使用初探

微软图表控件 MsChart 使用初探 收藏

昨天在网上看到了微软发布了.NET 3.5 框架下的图表控件，第一时间抓下来看了一下，发觉功能很强劲，基本上能想到的图表都可以使用它绘制出来，给图形统计和报表图形显示提供了很好的解决办法，同时支持 Web 和 WinForm 两种方式，不过缺点也比较明显，只能在最新的开发环境中使用，需要.Net 3.5 Sp1 以及 VS 2008 的开发环境。

下面是下载地址：

[下载免费的微软图表控件](#)

[下载 VS 2008 对图表控件的工具支持](#)

[下载微软图表控件例程](#)

[下载微软图表控件文档](#)

[访问微软图表控件论坛](#)

下了它的示例程序后，运行了一下，非常的强大，可以支持各种各样的图形显示，常见的：点状图、饼图、柱状图、曲线图、面积图、排列图等等，同时也支持 3D 样式的图表显示，不过我觉得最有用的功能还是支持图形上各个点的属性操作，它可以定义图形上各个点、标签、图形的提示信息（Tooltip）以及超级链接、Javascript 动作等，而不是像其它图形类库仅生成一幅图片而已，通过这些，加上微软自己的 Ajax 框架，可以建立一个可以互动的图形统计报表了。

一。安装

控件的安装相对比较简单，下载完后，先执行“MSChart.exe”程序，它会自动检测你的环境，安装到系统目录中去，如果要在 VS 2008 环境中直接使用，那么需要安装 For Vs2008 的插件，MSChart_VisualStudioAddOn.exe，还有一个中文语言包 MSChartLP_chs.exe。安装完后，打开 Vs2008，在建立项目的时候，你就能看到在工具栏中看到有一个 Chart 的控件了，如下图：



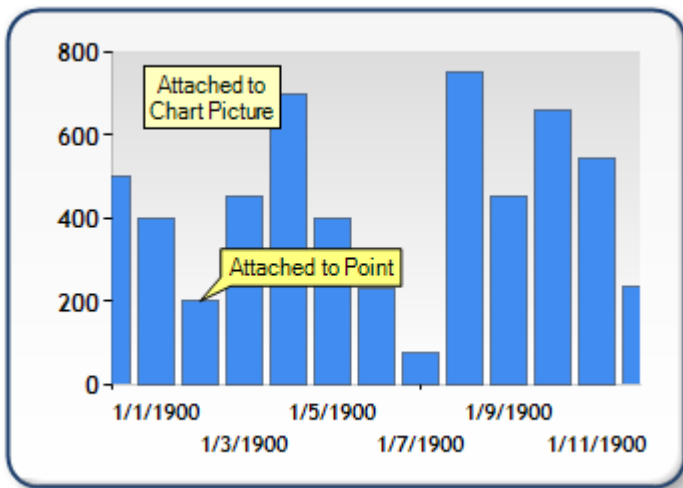
二。使用

安装好后，建立一个.NET3.5 的 Web 项目，像使用普通控件一样拖放到要使用的 Web 界面即可。初步研究了一下，整个图形控件主要由以下几个部份组成：

- 1.Annotations --图形注解集合
- 2.ChartAreas --图表区域集合
- 3.Legends --图例集合
- 4.Series --图表序列集合（即图表数据对象集合）
- 5.Titles --图标的标题集合

Annotations 注解集合

Annotations 是一个对图形的一些注解对象的集合，所谓注解对象，类似于对某个点的详细或者批注的说明，比如，在图片上实现各个节点的关键信息，如下图方框和黄色的小方框：



一个图形上可以拥有多个注解对象，可以添加十多种图形样式的注解对象，包括常见的箭头、云朵、矩形、图片等等注解符号，通过各个注解对象的属性，可以方便的设置注解对象的放置位置、呈现的颜色、大小、文字内容样式等常见的属性。

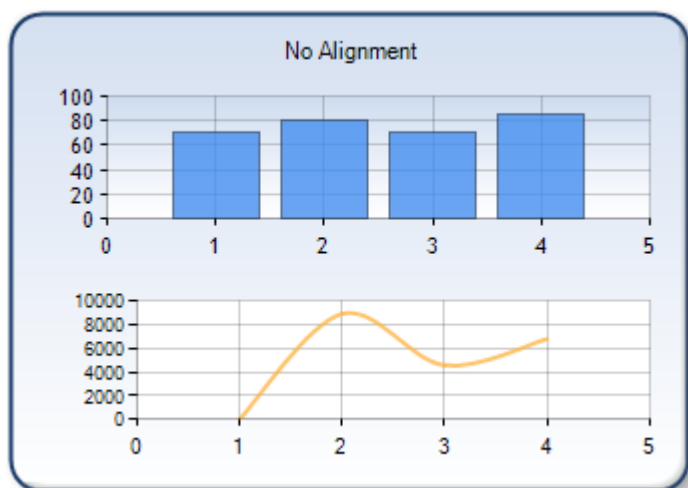
ChartAreas 图表区域集合

ChartAreas 可以理解为一个图表的绘图区，例如，你想在一幅图上呈现两个不同属性的内容，一个是用户流量，另一个则是系统资源占用情况，那么你要在一个图形上绘制这两种情况，明显是不合理的，对于这种情况，可以建立两个 ChartArea，一个用于呈现用户流量，另一个则用于呈现系统资源的占用情况。

当然了，图表控件并不限制你添加多少个绘图区域，你可以根据你的需要进行添加。对于每一个绘图区域，你可以设置各自的属性，如：X,Y 轴属性、背景等。

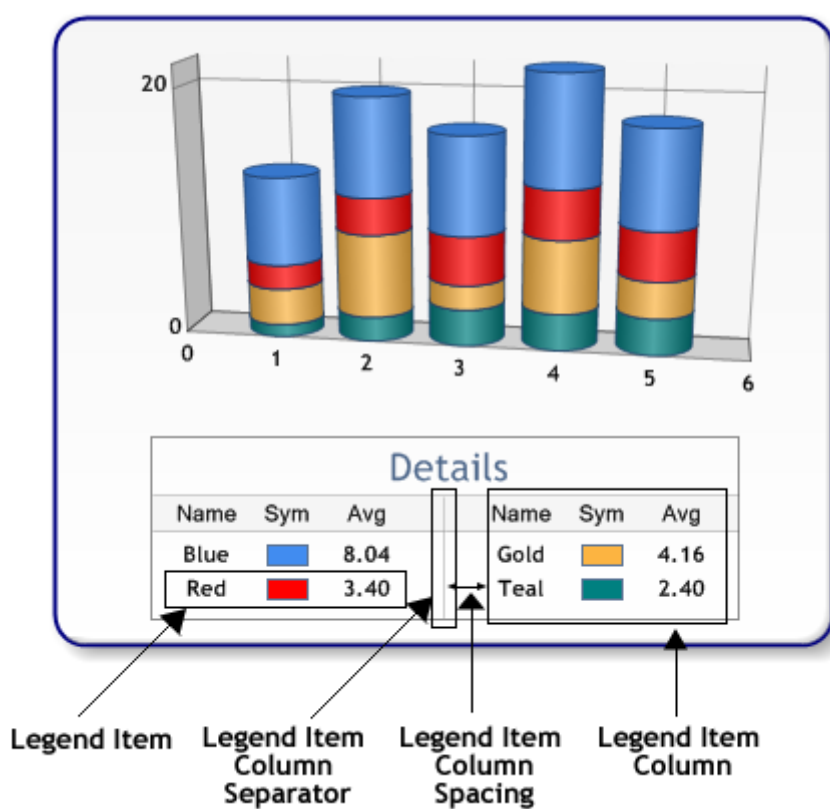
需要注意的是，绘图区域只是一个可以作图的区域范围，它本身并不包含要作图形的各种属性数据。

多绘图区效果图如下，分为上下两个绘图区域，分别表示不同的绘图数据：



Legends 图例集合

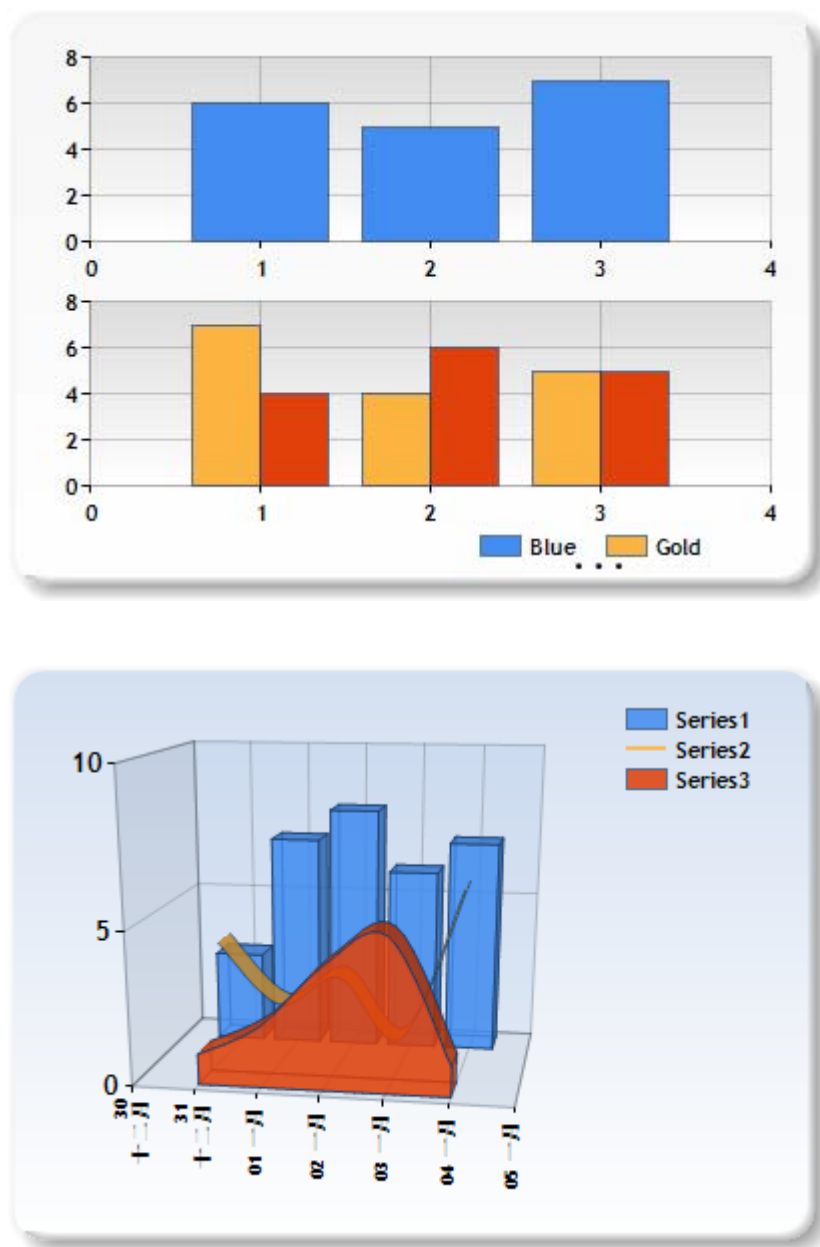
Legends 是一个图例的集合，即标注图形中各个线条或颜色的含义，同样，一个图片也可以包含多个图例说明，比如像上面说的多个图表区域的方式，则可以建立多个图例，每别说明各个绘图区域的信息，具体的图例配置说明此处就不详细说明了，可以参考一下官网的例子，写得丰富的详细了:) 也上一张图例的效果图吧~



Series 图表序列

图表序列，应该是整个绘图中最关键的内容了，通俗点说，即是实际的绘图数据区域，实际呈现的图形形状，就是由此集合中的每一个图表来构成的，可以往集合里面添加多个图表，每一个图表可以有自己的绘制形状、样式、独立的数据等。

需要注意的是，每一个图表，你可以指定它的绘制区域（见 `ChartAreas` 的说明），让此图表呈现在某个绘图区域，也可以让几个图表在同一个绘图区域叠加，如下图：



上面两幅图，分别表示了把图表放在不同的绘制区域和放在同一个绘制区域的情况。

继续回到 `ChartAreas` 章节举的例子，同时要显示用户的流量还要显示系统的占用情况，对于这种时候，应该建立两个 `Series`，一个用于呈现用户的流量，另一个则用于呈现系统的占用情况。它们分别属于

各自的绘图区域。

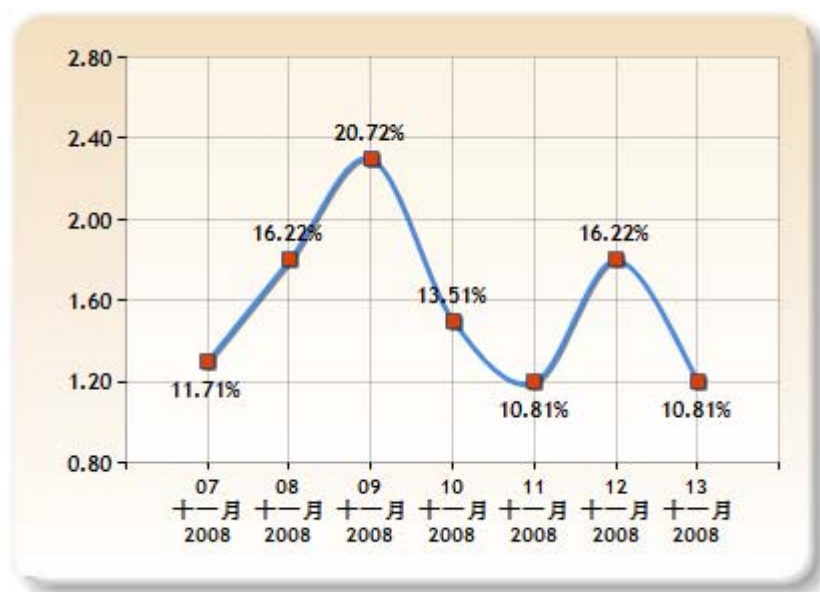
Titles 标题合集

根据字面含义即可以理解，是图表的标题配置，同样可以添加多个标题，以及设置标题的样式及文字、位置等属性。多看一下它的属性即能明白各自的含义。

三。其它属性

相对来说，我觉得比较有用的属性有三个，分别是：Label、Tooltip 以及 Url 链接。

Label 即标签的含义，可以在图片的关键位置进行一些关键数字或文字的描述，如下图：



像上图：X 轴和 Y 轴的文字便是标签，以及图表曲线中的红点上的文字，也是标签，添加了标签，可以让人更容易的对内容进行理解。

Tooltip 即提示的含义，用于在各个关键点，如：标签、图形关键点、标题等当鼠标移动上去的时候，提示用户一些相关的详细或说明信息，例如上图，可以给曲线中的每一个点增加 Tooltip 的属性，写上需要详细说明的内容，比如：详细的销售明细，那么，在鼠标移动到这个点的时候，会自动弹出提示信息。

Tooltip 可以支持简单方式以及自定义的方式，简单方式即像平时 Html 页面设置的 title 之类的属性效果，而自定义的方式，则可以实现图形、文本等各种复杂的提示信息显示。详细的方式请参考官方例子的：Interactivity and AJAX/Tooltips 以及 Interactivity and AJAX/Client Side Scripts 下面的相关例子。

Url 链接，图表控件中，有一大半的控件都有 Url 及 Tooltip 的属性，你可以设置此属性，在鼠标点击的时候，代到其它相应的页面去。

建议大家看看官方例子中的 Interactivity and AJAX 部份，很精彩：)

例子：建立一个 Cpu 信息和内存使用的实时统计表

下面写一个小例子，建立一个系统的内存实时统计图表，使用到了 Ajax 的方法，以及 Windows Api 取得系统内存的方法。

首先，建立一个 Aspx 页面，拖动一个图表控件到页面，设置图表控件的属性如下：

其中，MEMORY_INFO, ComputerInfo 是一个定义的结构体及调用 Win32 API 接口的一个访问类。程序分别取得每一个图表对象，每次加载的时候，都重新取得当前的内存和 Cpu 信息，再在图表上添加一个点，需要注意的是，一定要设置图表控件的 EnableViewState 属性为 True，否则无法记录状态。

Code

```
<asp:Chart ID="ChartMemory" runat="server" BackColor="LightSteelBlue"
BackGradientStyle="TopBottom" BackSecondaryColor="White" EnableTheming="False"
EnableViewState="True" Height="363px" Width="415px">
<Legends>
<asp:Legend Alignment="Center" Docking="Bottom" Name="Legend1" Title="图例">
</asp:Legend>
</Legends>
<Titles>
<asp:Title Font="微软雅黑, 16pt" Name="Title1" Text="系统内存监控图表">
</asp:Title>
</Titles>
<Series>
<asp:Series BorderColor="White" BorderWidth="3" ChartArea="ChartArea1"
ChartType="Spline" Legend="Legend1" Name="已使用物理内存" XValueType="Double"
YValueType="Double">
</asp:Series>
<asp:Series BorderWidth="3" ChartArea="ChartArea1" ChartType="Spline"
Legend="Legend1" Name="全部占用内存">
</asp:Series>
<asp:Series ChartArea="ChartArea2" ChartType="StackedArea" Legend="Legend1"
Name="CPU">
</asp:Series>
</Series>
<ChartAreas>
<asp:ChartArea BackColor="224, 224, 224" BackGradientStyle="LeftRight"
Name="ChartArea1">
</asp:ChartArea>
<asp:ChartArea Name="ChartArea2">
</asp:ChartArea>
</ChartAreas>
</asp:Chart>
```

一共建立了两个绘图区，一个用于呈现内存使用情况的在 ChartArea1 区域，另一个则是呈现 Cpu 使用情况的，放置在 ChartArea2 区域了。一共有三个图表，分别表示已使用的物理内存、全部占用的物理内存，以及 Cpu 使用显示的情况。

添加一个 Ajax 的计时器以及 Ajax 的 ScriptManager, UpdatePanel, 把计时器和图表控件都拖进 UpdatePanel

里面。设置计时器的间隔时间为一秒钟（1000），双击计时器，写如下代码：

Code

```
static PerformanceCounter pc = new PerformanceCounter("Processor", "% Processor Time", "_Total");
protected void Timer1_Tick(object sender, EventArgs e)
{
    MEMORY_INFO MemInfo = new MEMORY_INFO();
    ComputerInfo.GlobalMemoryStatus(ref MemInfo);
    //UseMemory
    Series series = ChartMemory.Series[0];
    int xCount = series.Points.Count == 0 ? 0 : series.Points.Count - 1;
    double lastXValue = series.Points.Count == 0 ? 1 : series.Points[xCount].XValue + 1;
    double lastYValue = (double)(MemInfo.dwTotalPhys-MemInfo.dwAvailPhys)/1024/1024;
    series.Points.AddXY(lastXValue, lastYValue);
    //Total Memory
    series = ChartMemory.Series[1];
    lastYValue = (double)(MemInfo.dwTotalVirtual+MemInfo.dwTotalPhys-MemInfo.dwAvailPhys - MemInfo.dwAvailVirtual)/1024/1024;
    series.Points.AddXY(lastXValue, lastYValue);

    //CPU
    series = ChartMemory.Series[2];
    lastYValue = (double)pc.NextValue();
    series.Points.AddXY(lastXValue, lastYValue);

    // Remove points from the left chart side if number of points exceeds 100.
    while (this.ChartMemory.Series[0].Points.Count > 80)
    {
        // Remove series points
        foreach (Series s in this.ChartMemory.Series)
        {
            s.Points.RemoveAt(0);
        }
    }
    // Adjust categorical scale
    double axisMinimum = this.ChartMemory.Series[0].Points[0].XValue;
    this.ChartMemory.ChartAreas[0].AxisX.Minimum = axisMinimum;
    this.ChartMemory.ChartAreas[0].AxisX.Maximum = axisMinimum + 99;
}
```

附上取得内存信息的类代码：

Code

```
/// <summary>
///取得计算机的系统信息
/// </summary>
public class ComputerInfo
```

```

{
/// <summary>
/// 取得 Windows 的目录
/// </summary>
/// <param name="WinDir"></param>
/// <param name="count"></param>
[DllImport("kernel32")]
public static extern void GetWindowsDirectory(StringBuilder WinDir, int count);
/// <summary>
/// 获取系统路径
/// </summary>
/// <param name="SysDir"></param>
/// <param name="count"></param>
[DllImport("kernel32")]
public static extern void GetSystemDirectory(StringBuilder SysDir, int count);
/// <summary>
/// 取得 CPU 信息
/// </summary>
/// <param name="cpuinfo"></param>
[DllImport("kernel32")]
public static extern void GetSystemInfo(ref CPU_INFO cpuinfo);
/// <summary>
/// 取得内存状态
/// </summary>
/// <param name="meminfo"></param>
[DllImport("kernel32")]
public static extern void GlobalMemoryStatus(ref MEMORY_INFO meminfo);
/// <summary>
/// 取得系统时间
/// </summary>
/// <param name="stinfo"></param>
[DllImport("kernel32")]
public static extern void GetSystemTime(ref SYSTEMTIME_INFO stinfo);

public ComputerInfo()
{
}
}

//定义 CPU 的信息结构
[StructLayout(LayoutKind.Sequential)]
public struct CPU_INFO
{
public uint dwOemId;
public uint dwPageSize;
public uint lpMinimumApplicationAddress;

```

```

public uint lpMaximumApplicationAddress;
public uint dwActiveProcessorMask;
public uint dwNumberOfProcessors;
public uint dwProcessorType;
public uint dwAllocationGranularity;
public uint dwProcessorLevel;
public uint dwProcessorRevision;
}

```

//定义内存的信息结构

```

[StructLayout(LayoutKind.Sequential)]
public struct MEMORY_INFO
{
    public uint dwLength;
    public uint dwMemoryLoad;
    public uint dwTotalPhys;
    public uint dwAvailPhys;
    public uint dwTotalPageFile;
    public uint dwAvailPageFile;
    public uint dwTotalVirtual;
    public uint dwAvailVirtual;
}

```

//定义系统时间的信息结构

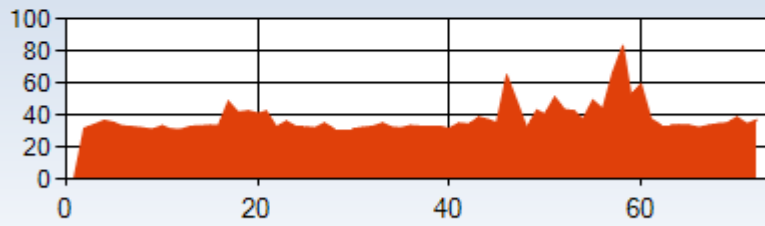
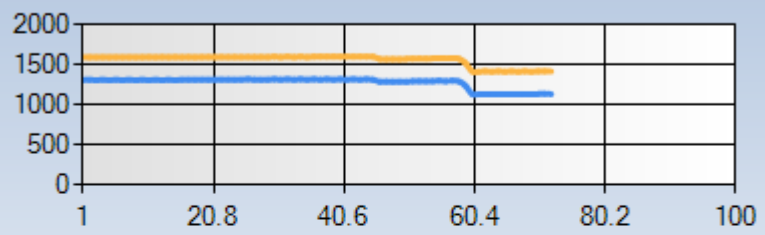
```

[StructLayout(LayoutKind.Sequential)]
public struct SYSTEMTIME_INFO
{
    public ushort wYear;
    public ushort wMonth;
    public ushort wDayOfWeek;
    public ushort wDay;
    public ushort wHour;
    public ushort wMinute;
    public ushort wSecond;
    public ushort wMilliseconds;
}

```

运行的效果图如下：

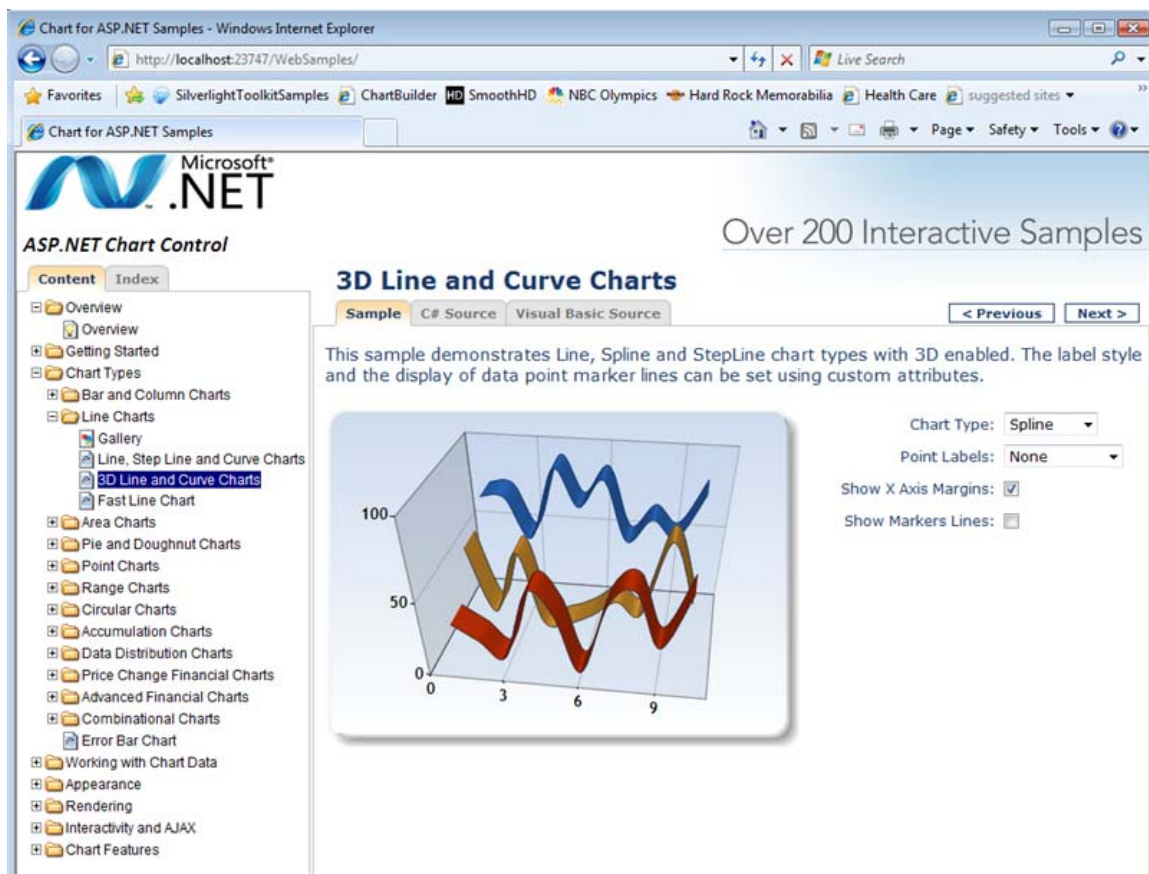
系统内存监控图表



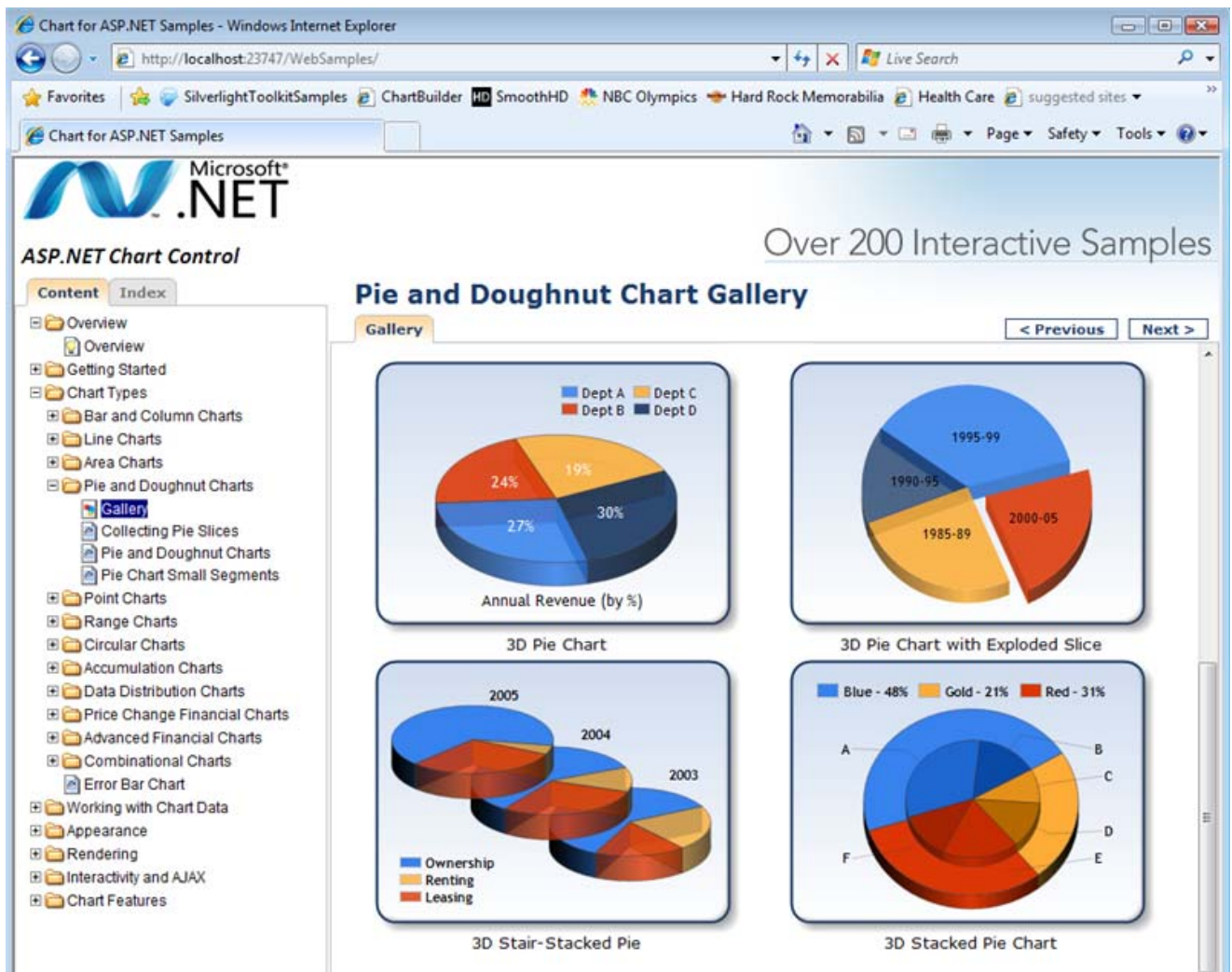
图例

■ CPU — 全部占用内存 — 已使用物理内存

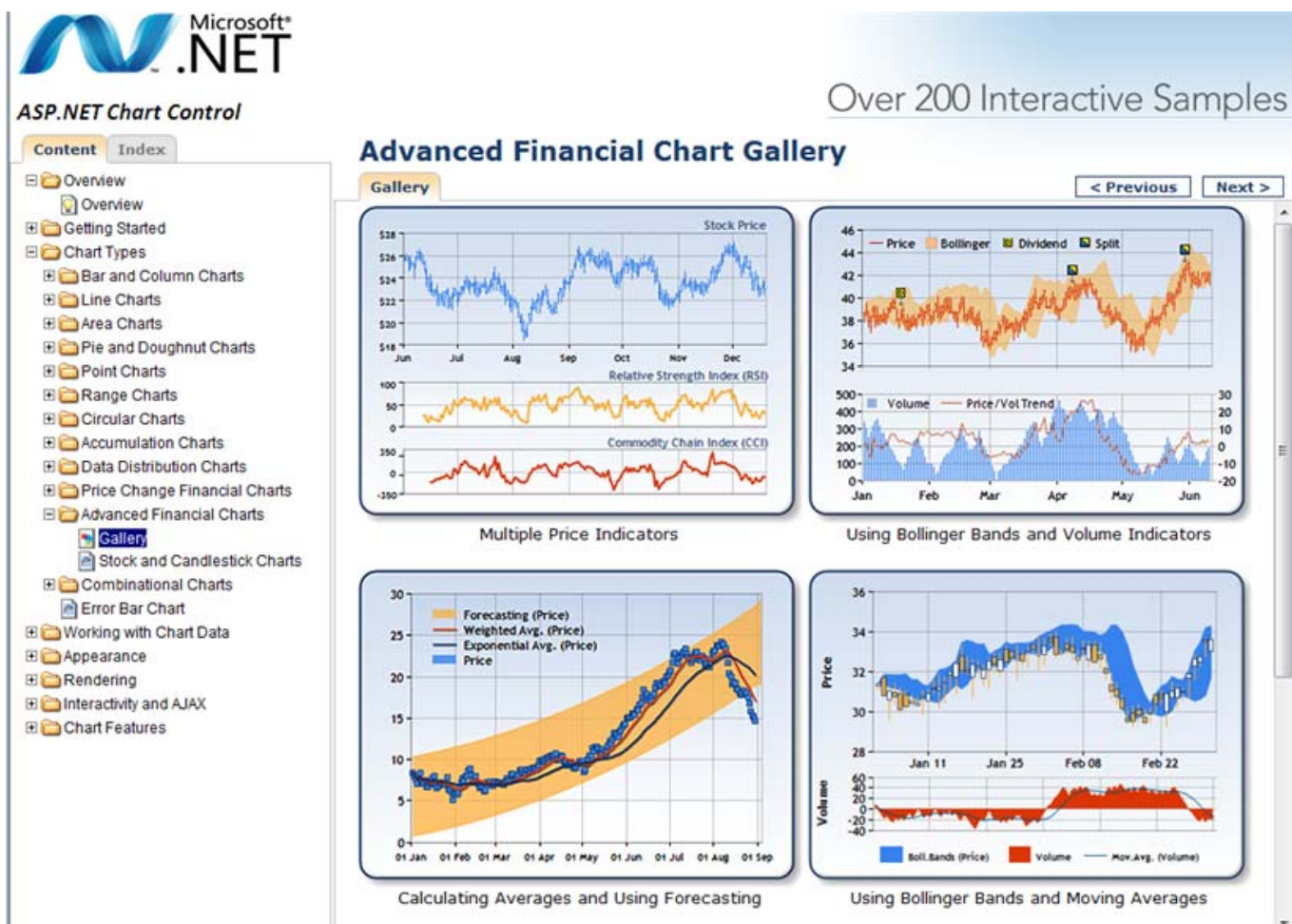
下面的例子（在 Chart Types->Line Charts->3D Line and Curve Charts，图表类型->曲线图->3D 曲线图下）示范了如何做 Line, Spline 和 StepLine 图表：



下面的例子(在 Chart Types->Pie and Doughnut Charts, 图表类型->饼图和甜甜圈图下)示范了各种饼图和 3D 甜甜圈图选项:



下面的例子(在 Chart Types->Advanced Financial Charts, 图表类型->高级财务图表下)示范了一些曲线图表:



除了上面的例子外，你可以下载微软图表控件文档或在图表控件论坛上询问，以了解更多详情。

这应该给你的标准 ASP.NET 功能工具箱又提供了一个非常有用（而且免费）的工具，允许你轻松地将更丰富的视觉效果和数据流程场景添加到你的 ASP.NET 应用中去。

希望本文对你有所帮助，

本文来自CSDN博客，转载请标明出处：<http://blog.csdn.net/liutengt/archive/2009/02/20/3908396.aspx>

微软 Chart 控件导学--渲染图表

这一章几乎是纯理论的东西，自己不太擅长，所以就完全翻译了一篇文章。我们都知道图片操作是一种很费时的操作，如果一个网站是以图表为基础的话那么渲染方式就可以很大程度上影响网站的速度，即使是不以图表为基础的网站，有时不同的渲染方法安全性也不同。当然并没有什么完美的方法，只能从自己网站的实际情况出发来决定用什么样渲染方式。只有深入了解各种渲染方式的优劣才能做出更好的选择。

先解释一下这一章写什么吧，并不是说怎么渲染让图片更好看一些，而是讲如何生成图片，保存图片发送图片让我们的网站（如果是单机应用程序那么这一章的意义就不太大了，那时不会有多用户竞争访问和安

全问题)更有效率也更加安全,好下面就是正文了,主要是翻译完全来的内容,原文是 Using Microsoft's Chart Controls In An ASP.NET Application: Rendering the Chart。

一、生成图片并静态存储

渲染图片最简单直接的方法就是生成图片并将图片存储在服务器上,然后再显示出来。具体的过程为:当用户请求图表时,Chart 控件在服务器上生成图片并将图片存储在服务器的特定目录下。然后 Chart 控件再生成一个元素,并将的 src 属性设置为刚才生成的图片。

静态存储图片时最重要的两个属性是: ImageLocation 和 ImageStorageMode。如果想要静态存储图片则 ImageStorageMode 设置为 UseImageLocation, ImageLocation 设置为存储图片的路径和图片名(但要注意不需要有图片的扩展名,扩展名可以用 ImageType 来指定),下面的代码段为一个简单的示例:

view plaincopy to clipboardprint?

```
<asp:Chart ID="chtSalesPerDOW" runat="server" Width="500" Height="350"
```

```
    ImageLocation="~/Images/Chart"
```

```
    ImageStorageMode="UseImageLocation"
```

```
    ImageType="Png">
```

...

```
</asp:Chart>
```

```
<asp:Chart ID="chtSalesPerDOW" runat="server" Width="500" Height="350"
```

```
    ImageLocation="~/Images/Chart"
```

```
    ImageStorageMode="UseImageLocation"
```

```
    ImageType="Png">
```

...

```
</asp:Chart>
```

这样就会在服务器中的 Images 文件夹下生成 Chart.png 这个图片(注意用户对这个特定位置的权限问题),并呈现到网页中,但是这并不是完美的,主要有以下几个问题:

名字冲突问题:如果用以上的代码段两个用户访问网站时就会生成同样名字的两张图片,这样的问题如何解决。

如何辨别当前用户:如果两个用户同时访问同一个页面,如果这个页面中的图表是固定不变的,那不会有什么问题(呵呵,一般不会出现这种问题,如果都不变那就用一张静态图片就好了,用这个控件那么麻烦干嘛。。。)。但是通常情况下我们会根据不同的用户和用户的不同的选择生成不同的图片,那么问题就出现

了。设想一下，如果第一个用户首先请求，Chart 控件给他生成了一张图片，但还没有发送之前第二个用户又发出请求，也生成一张图片。两张图片会不会出现覆盖问题（就是第二张产生的时候把第一张删除了），如果不会如何来分辨这两个用哪个呢？

存储问题：如果真的和上面那样每次生成的图片都是一样的，服务器中只有一张图片，那不会出现存储问题，但上面的两个问题的出现我想大家也可能想到那就是要从文件名上下功夫，如果那样每一个访问这个页面都会产生一张新的图片，一直存储下去服务器的硬盘空间肯定是有问题的，如何来删除不用的图片呢？

我想有很多人也想到了解决这个问题就要从文件名上下功夫，只要每一个用户所产生的文件名不同，那么各自用户产生并使用自己的图片就好了。用户当然可以自己来用类似于随机数的方法来使每一个用户产生的文件名不同，但是微软给我们提供更加简单的方法。可用的关键字列表如下：

关键字

功能描述

#UID

#UID 会产生一个全局唯一标志符（globally unique identifier）这个标志符就会保证你生成的字符中唯一的（根据名字，这个很可能是全球唯一的，呵呵）。例如：如果设置 ImageLocation 为 "~/Images/Chart_#UID" 将会产生如 ~/Images/Chart_7795fa87-ef6b-4e9f-9c66-abb2b9980185.png 这样的文件名，这可以保证文件名唯一（但是这样的话要注意文件存储的问题，设定删除图片的条件）。

#SEQ(max,minutes)

这个关键字可以生成一个从 1 开始直到 max 的序列，minutes 表示某个元素再生成时的最少间隔（例如如果 minutes 设为 2 则表示在 2 分钟内不能生成重复的数字）。例如设置 ImageLocation 为 "~/Images/Chart_#SEQ(50,5)" 时将会生成图上名如： ~/Images/Chart_000001.png. 下一个名称为：~/Images/Chart_000002.png,直到 ~/Images/Chart_000050.png.

#NOGUIDPARAM

为了加速显示，浏览器会缓存一些东西，例如如果两次请求两张名称相同的图片（在用#SEQ()时可能会出现这种情况）浏览器就会默认使用上一次已经接收到的图片，这时就可能会显示错误的图片，为了解决这个问题，通常的做法是在图片的后面加一个唯一的查询字符串，生成类似于 ~/Images/Chart_000002.png?7795fa87-ef6b-4e9f-9c66-abb2b9980185 这样的图片地址，就不会重复了。如果用这个选项就会阻止使用这样的方法使每次都从服务器请求图片。

呵呵，现在我们就可以把 ImageLocation 设置为~/Images/TempChartImages/Chart_#SEQ(25,2)这样不仅能保证每一次请求的地址都不同，还能让图片的张数不超过 25 张（如果服务器的用户很多可以将 max 设置为更大，时间也可以设置的短一些，当然在不会出现问题的前提下）。

二、用 ChartHttpHandler 来动态生成图表

用上面的方法将图片存储在大家网络用户可以访问到的地方如果对保密性很高的应用就会显得不怎么安全，所以就出现了将数据放到一些网络用户无法访问到的地方的解决方案。放到用户无法访问到的地方但用户还必须访问到其中的某些资源，所以需要有一个中介，这个中介就是 `ChartHttpHandler`。ASP.NET 用户可能对 `HttpHandler` 并不陌生。是的，`ChartHttpHandler` 就是 `Chart` 控件自己的 `HttpHandler`。在 `web.config` 文件中可以找到以下的代码段：

```
view plaincopy to clipboardprint?
```

```
<handlers>
```

```
.....
```

```
<add name="ChartImageHandler" preCondition="integratedMode" verb="GET,HEAD,POST"
```

```
    path="ChartImg.axd"                type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,  
System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
```

```
</handlers>
```

```
<handlers>
```

```
.....
```

```
<add name="ChartImageHandler" preCondition="integratedMode" verb="GET,HEAD,POST"
```

```
    path="ChartImg.axd"                type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,  
System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
```

```
</handlers>
```

就是这个了。用户无法访问到的地方通常是指文件系统，内存和 `Session` 中，并且存放的地方是可以配制的，`web.config` 文件中有类似于以下的代码段：

```
view plaincopy to clipboardprint?
```

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
    <add key="ChartImageHandler" value="storage=memory;deleteAfterServicing=true;" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
    <add key="ChartImageHandler" value="storage=memory;deleteAfterServicing=true;" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

这一段代码表示将图片存储在内存中（storage=memory）并且在图片一使用完后就释放内存（deleteAfterServicing=true）。

view plaincopy to clipboardprint?

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
    <add key="ChartImageHandler" value="storage=file;timeout=20;dir=c:\TempImageFiles\" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
    <add key="ChartImageHandler" value="storage=file;timeout=20;dir=c:\TempImageFiles\" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

这一段表示将图片存储在文件系统中，目录为 c:\TempImageFiles\，20 分钟超时。

这些都设置好了之后我们就可以使用 ChartHttpHandler 来动态生成图表了，我们可以什么都不设置，因为这些都是默认的，要设置的话只要把 ImageStorageMode 设置为 UseHttpHandler 就可以了。最后会生成类似于以下代码的图片：

```

```

至于为什么 src 设置成 ChartImg.axd 以及具体是如何处理的去查一下有关 HttpHandler 的资料我想并不难理解。

三、从 Chart 控件直接取得数据

上面的两种方法都是先让 Chart 控件生成图片，存储在一定位置后再发送给用户。我们也可以让 Chart 控件直接生成图片，并将 2 进制数据流直接发送到我们的页面。

使用 2 进制数据流的方法首先我们要生成一个单独的页面来产生 2 进制数据流，这个页面只能一唯一的一个 Chart 控件（也不能有其他的控件）。我们先假设这个页面为 BStreaming.aspx，那么我们在要用这个图表的时候只要用就可以了。

在生成 BStreaming.aspx 页面时可以用以下的格式：

view plaincopy to clipboardprint?

```
<%@ Page Language="C#" AutoEventWireup="false" CodeFile="SalesChart.aspx.cs"  
Inherits="Demos_Rendering_SalesChart" %>
```

```
<asp:Chart ID="chtSalesPerDOW" runat="server" Width="500" Height="350"
```

```
RenderType="BinaryStreaming">
```

...

```
</asp:Chart>
```

```
<%@ Page Language="C#" AutoEventWireup="false" CodeFile="SalesChart.aspx.cs"  
Inherits="Demos_Rendering_SalesChart" %>
```

```
<asp:Chart ID="chtSalesPerDOW" runat="server" Width="500" Height="350"
```

```
RenderType="BinaryStreaming">
```

...

```
</asp:Chart>
```

注意两点：1、页面中不能有其他的元素。2、要将 `RenderType` 设置成 `BinaryStreaming`。这样就可以了。

四、技术选择

在默认情况下是使用 `ChartHttpHandler` 并用以下配制：

view plaincopy to clipboardprint?

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
  <add key="ChartImageHandler" value="storage=file;timeout=20;dir=c:\TempImageFiles\" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

```
<configuration>
```

```
...
```

```
<appSettings>
```

```
  <add key="ChartImageHandler" value="storage=file;timeout=20;dir=c:\TempImageFiles\" />
```

```
</appSettings>
```

```
...
```

```
</configuration>
```

但这个选项并不是在所有的情况下都是最好的，下面是三种技术的比较：

技术类型

优点

缺点

直接生成图片

容易设置和理解

容易控制文件路径和名称

不安全：用户可以通过改变图片的路径来获得其他用户的图表

增加硬盘读写，影响效率

需要要配制文件夹的读写权限

ChartHttpHandler

可以锁定图片，只让生成该图片的用户对这个图片有访问权限（需要配制 ChartImageHandler 的 privateImages 属性）

可以通过将图片放在内存中来减少硬盘读写

具有配制图片存储位置的能力

在 web.config 中配制，超出了件本身属性的范围

每一样配制是针对整个网站，无法不同的图片存放在不同的位置

2 进制数据流

直接发送数据，避免了“成/等待/发送”模式

可以用后台代码进行详细的权限控制

代码量大

不支持 AJAX 等交互方式

结论：呵呵，无非是根据自己的实际情况选用方式罢了，不多说了。

本文来自 CSDN 博客，转载请标明出处：<http://blog.csdn.net/grlsir/archive/2010/01/11/5177179.aspx>