GDI 基本知识

[转]GDI+基础(1)

System.Drawing 命名空间提供了对 GDI+ 基本图形功能的访问

System.Drawing.Drawing2D 命名空间提供高级的二维和矢量图形功能。此命名空间包含梯度型画刷、Matrix 类（用于定义几何变换）和 GraphicsPath 类

System.Drawing.Imaging 命名空间提供高级 GDI+ 图像处理功能

System.Drawing.Text 命名空间提供高级 GDI+ 排版功能

Bitmap 类封装 GDI+ 位图，此位图由图形图像及其属性的像素数据组成。Bitmap 对象是用于处理由像素数据定义的图像的对象

常用属性和方法

1.属性:

Height 获取此 Image 对象的高度

PixelFormat 获取此 Image 对象的像素格式

Size 获取此图像的以像素为单位的宽度和高度

Width 获取此 Image 对象的宽度

2.方法

Dispose 释放由此 Image 对象使用的所有资源

Save 将此图像以指定的格式保存到指定的流中

SetPixel 设置 Bitmap 对象中指定像素的颜色

示例

```
<%@ Page ContentType="image/gif" Language="C#" %>

<!--ContentType 设置页面类型-->

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(400, 200);

 //创建一个宽 400，高 200 的实例

 Random pixel = new Random();

 //创建一个随机实例

 for(int i=0;i<1000;i++) {

 bmp.SetPixel(pixel.Next(400), pixel.Next(200), Color.Red);

 //设置 Bitmap 对象中指定像素的颜色,构造函数参数为 x,y,color

 }

bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的
图像的格式

//向客户端输出数据流，并以此数据流形成 Gif 图片

}
```

</script>

Graphics 类表示封装 GDI+ 绘图面,比如画长方形,圆形等更加复杂的图形.

常用属性和方法:

属性 说明

Clip 获取或设置 Region 对象，该对象限定此 Graphics 对象的绘图区域。

ClipBounds 获取 RectangleF 结构，该结构限定此 Graphics 对象的剪辑区域。

DpiX 获取此 Graphics 对象的水平分辨率。

DpiY 获取此 Graphics 对象的垂直分辨率。

PageScale 获取或设置此 Graphics 对象的全局单位和页单位之间的比例。

PageUnit 获取或设置用于此 Graphics 对象中的页坐标的度量单位。

PixelOffsetMode 获取或设置一个值，该值指定在呈现此 Graphics 对象的过程中像素如何偏移。

RenderingOrigin 为抵色处理和阴影画笔获取或设置此 Graphics 对象的呈现原点。

SmoothingMode 获取或设置此 Graphics 对象的呈现质量。

TextRenderingHint 获取或设置与此 Graphics 对象关联的文本的呈现模式。

Transform 获取或设置此 Graphics 对象的全局变换。

VisibleClipBounds 获取或设置此 Graphics 对象的可见剪辑区域的边框。

方法 说明

Clear 清除整个绘图面并以指定背景色填充。

Dispose 释放由此 Graphics 对象使用的所有资源。

DrawArc 绘制一段弧线，它表示由一对坐标、宽度和高度指定的椭圆部分。

DrawEllipse 绘制一个由边框定义的椭圆。

DrawIcon 在指定坐标处绘制由指定的 Icon 对象表示的图像。

DrawIconUnstretched 绘制指定的 Icon 对象表示的图像，而不缩放该图像。

DrawImage 在指定位置并且按原始大小绘制指定的 Image 对象。

DrawImageUnscaled 在坐标对所指定的位置并且按其原始大小绘制指定的 Image 对象。

DrawLine 绘制一条连接由坐标对指定的两个点的线条。

DrawLines 绘制一系列连接一组 Point 结构的线段。

DrawPie 绘制一个扇形，该扇形由一个坐标对、宽度和高度以及两条射线所指定的椭圆定义。

DrawPolygon 绘制由一组 Point 结构定义的多边形。

DrawRectangle 绘制由坐标对、宽度和高度指定的矩形。

DrawRectangles 绘制一系列由 Rectangle 结构指定的矩形。

DrawString 在指定位置并且用指定的 Brush 和 Font 对象绘制指定的文本字符串。

FillClosedCurve 填充由 Point 结构数组定义的闭合基数样条曲线的内部。

FillEllipse 填充边框所定义的椭圆的内部，该边框由一对坐标、一个宽度和一个高度指定。

FillPie 填充由一对坐标、一个宽度、一个高度以及两条射线指定的椭圆所定义的扇形区的内部。

FillPolygon

填充 Point 结构指定的点数组所定义的多边形的内部。

FillRectangle 填充由一对坐标、一个宽度和一个高度指定的矩形的内部。

FillRectangles 填充由 Rectangle 结构指定的一系列矩形的内部。

FillRegion 填充 Region 对象的内部。

Flush 强制执行所有挂起的图形操作并立即返回而不等待操作完成。

FromHdc 从设备上下文的指定句柄创建新的 Graphics 对象。

FromHwnd 从窗口的指定句柄创建新的 Graphics 对象。

FromImage 从指定的 Image 对象创建新 Graphics 对象。

GetHdc 获取与此 Graphics 对象关联的设备上下文的句柄。

ReleaseHdc 释放通过以前对此 Graphics 对象 GetHdc 方法的调用获得的设备上下文句柄。

ResetClip 将此 Graphics 对象的剪辑区域重置为无限区域。

ResetTransform 将此 Graphics 对象的全局变换矩阵重置为单位矩阵。

画一个矩形

DrawRectangle 绘制由坐标对、宽度和高度指定的矩形

public void DrawRectangle(Pen, int, int, int, int);

绘制由坐标对、宽度和高度指定的矩形

```
  <%@ Page ContentType="image/gif" Language="C#" %>

  <!--ContentType 设置页面类型-->

  <%@ Import namespace="System.Drawing" %>

  <%@ Import namespace="System.Drawing.Imaging" %>
```

```
<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(500, 400);

 //创建一个宽 400，高 200 的实例

Graphics gph;

//从指定的 Image 对象创建新 Graphics 对象

gph = Graphics.FromImage(bmp);

//清除整个绘图面并以指定背景色填充

gph.Clear(Color.Olive);

//绘制由坐标对、宽度和高度指定的矩形

gph.DrawRectangle(Pens.Red, 100, 100, 50, 50);

//填充由笔刷颜色,一对坐标、一个宽度和一个高度指定的矩形的内部

gph.FillRectangle(Brushes.Green, 200, 200, 50, 50);

bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的图像的格式

//向客户端输出数据流，并以此数据流形成 Gif 图片

}

</script>
```

使用颜色

Color 结构表示 ARGB 颜色

颜色包含 a,R,G,B

```
Color mycolor;
mycolor = Color.FromName("blue");
//FromName 方法把颜色字符串换成 GDI+颜色
mycolor = Color.FromArgb(255, 0, 0);
//FromArgb 方法设置 RGB 值
mycolor = Color.FromArgb(100,255, 0, 0);
//FromArgb 方法设置 a,RGB 值
```

用 ColorTranslator 类在 HTML 颜色和 GDI+颜色之间转换

```
mycolor = ColorTranslator.FromHtml("#00FF00");
```

使用画笔

Brush 类用于填充图形形状（如矩形、椭圆形、扇形、多边形和封闭路径）内部的对象

System.Drawing.Drawing2D.HatchBrush 用阴影样式、前景色和背景色定义矩形画笔

System.Drawing.Drawing2D.LinearGradientBrush 使用线性渐变封装 Brush 对象

System.Drawing.Drawing2D.PathGradientBrush 封装 Brush 对象，它通过渐变填充

GraphicsPath 对象的内部

System.Drawing.SolidBrush 定义单色画刷

System.Drawing.TextureBrush 使用图像来填充形状的内部

PathGradientBrush 类似 LinearGradientBrush,但更加强大,不再介绍.

下面示例

```
<%@ Page ContentType="image/gif" Language="C#" %>

<!--ContentType 设置页面类型-->

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(600, 120);

 //创建一个宽 400，高 200 的实例

 Bitmap img=new Bitmap(Server.MapPath("001.jpg"));

Color mycolor;

mycolor = Color.FromArgb(0, 0, 255);

//FromArgb 方法设置 RGB 值

 Rectangle rect=new Rectangle(300,0,100,100);
```

```
SolidBrush solid = new SolidBrush(mycolor);

//实心颜色填充

HatchBrush hatch = new HatchBrush(HatchStyle.Cross, Color.Orange);

//阴影模式填充,从 HatchStyle 属性中选择样式

LinearGradientBrush linear;

linear = new LinearGradientBrush(rect, Color.Red, Color.Yellow, LinearGradientMode.ForwardDiagonal);

//渐变色填充，构造函数参数为(Rectangle, Color, Color, LinearGradientMode)

//一个矩形，起点颜色,结束色,指定如何呈现渐变;

//BackwardDiagonal 指定从右上到左下的渐变。

//ForwardDiagonal 指定从左上到右下的渐变。

//Horizontal 指定从左到右的渐变。

//Vertical 指定从上到下的渐变。

TextureBrush texture = new TextureBrush(img);

//以图片填充图形

Graphics gph;

//从指定的 Image 对象创建新 Graphics 对象

gph = Graphics.FromImage(bmp);

//清除整个绘图面并以指定背景色填充

gph.Clear(Color.Red);

//绘制由坐标对、宽度和高度指定的矩形

//填充由一对坐标、一个宽度和一个高度指定的矩形的内部
```

```
gph.FillRectangle(Brushes.Green, 0, 0, 100, 100);

gph.FillRectangle(solid, 100, 0, 100, 100);

gph.FillRectangle(hatch, 200, 0, 100, 100);

gph.FillRectangle(linear, rect);

gph.FillRectangle(texture, 400, 0, 100, 100);

bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的图像的格式

//向客户端输出数据流，并以此数据流形成 Gif 图片

}

</script>
```

[转]GDI+基础(2)

使用钢笔,画笔用来填充图形内部,钢笔则用来绘制带有一定宽度,样式和色彩的线条和曲线.

可以使用标准的 pens 类

```
<%@ Page ContentType="image/gif" Language="C#" %>

<!--ContentType 设置页面类型-->

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(600, 120);

 //创建一个宽 400，高 200 的实例

Color mycolor;

mycolor = Color.FromArgb(0, 0, 255);

//FromArgb 方法设置 RGB 值

Pen mypen = new Pen(Color.Blue, 5);

//创建自定义钢笔,color,float,参数为颜色和宽度

Graphics gph;

//从指定的 Image 对象创建新 Graphics 对象
gph = Graphics.FromImage(bmp);
//清除整个绘图面并以指定背景色填充
gph.Clear(Color.Red);
//绘制由坐标对、宽度和高度指定的矩形
gph.DrawRectangle(Pens.Green, 10, 10, 100, 100);
gph.DrawRectangle(mypen, 120, 10, 100, 100);
bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的图像的格式
//向客户端输出数据流，并以此数据流形成 Gif 图片
}
</script>
```

自定义钢笔样式

DashStyle 枚举 指定用 Pen 对象绘制的虚线的样式

成员名称	说明
Custom	指定用户定义的自定义划线段样式。
Dash	指定由划线段组成的直线。
DashDot	指定由重复的划线点图案构成的直线。
DashDotDot	指定由重复的划线点点图案构成的直线。

Dot 指定由点构成的直线。

Solid 指定实线。

示例

```
<%@ Page ContentType="image/gif" Language="C#" %>
<!--ContentType 设置页面类型-->
<%@ Import namespace="System.Drawing" %>
<%@ Import namespace="System.Drawing.Imaging" %>
<%@ Import namespace="System.Drawing.Drawing2D" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
Bitmap bmp = new Bitmap(600, 120);
 //创建一个宽 400，高 200 的实例
Graphics gph;
//从指定的 Image 对象创建新 Graphics 对象
gph = Graphics.FromImage(bmp);
//清除整个绘图面并以指定背景色填充
gph.Clear(Color.Green);
Pen mypen = new Pen(Color.Red, 2);
//创建自定义钢笔,color,float,参数为颜色和宽度
mypen.DashStyle = DashStyle.Custom;

//自定义钢笔样式

gph.DrawLine(mypen, 10, 10, 300, 10);

//绘制直线

mypen.DashStyle = DashStyle.Dash;

gph.DrawLine(mypen, 10, 30, 300, 30);

//(Pen, int, int, int, int)pen Pen 起点 x 坐标,起点 y 坐标,终点 x 坐标,终点 y 坐标

mypen.DashStyle = DashStyle.DashDot;

gph.DrawLine(mypen, 10, 50, 300, 50);

mypen.DashStyle = DashStyle.DashDotDot;

gph.DrawLine(mypen, 10, 70, 300, 70);

mypen.DashStyle = DashStyle.Dot;

gph.DrawLine(mypen, 10, 90, 300, 90);

mypen.DashStyle = DashStyle.Solid;

gph.DrawLine(mypen, 10, 110, 300, 110);

bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的图像的格式

//向客户端输出数据流，并以此数据流形成 Gif 图片

}

</script>
```

设置钢笔线帽样式,可以设置线条的起始和结束的线帽样式

StartCap 获取或设置用在通过此 Pen 对象绘制的直线起点的帽样式

EndCap 获取或设置用在通过此 Pen 对象绘制的直线终点的帽样式

DashCap 获取或设置用在短划线终点的帽样式，这些短划线构成通过此 Pen 对象绘制的虚线

LineCap 枚举

成员名称	说明
AnchorMask	指定用于检查线帽是否为锚头帽的掩码。
ArrowAnchor	指定箭头状锚头帽。
Custom	指定自定义线帽。
DiamondAnchor	指定菱形锚头帽。
Flat	指定平线帽。
NoAnchor	指定没有锚。
Round	指定圆线帽。
RoundAnchor	指定圆锚头帽。
Square	指定方线帽。
SquareAnchor	指定方锚头帽。
Triangle	指定三角线帽。

示例

```
<%@ Page ContentType="image/gif" Language="C#" %>

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<script language="C#" runat=server>

void Page_Load(Object sender , EventArgs e)

{

 Bitmap objBitmap;

 Graphics objGraphics;

 Pen objPen;

 objBitmap = new Bitmap( 400, 200 );

 objGraphics = Graphics.FromImage( objBitmap );

 objPen = new Pen( Color.White );

 objPen.Width = 6;

 objPen.EndCap = LineCap.ArrowAnchor;

 objGraphics.DrawLine( objPen, 10, 20, 350, 20 );

 objPen.EndCap = LineCap.DiamondAnchor;

 objGraphics.DrawLine( objPen, 10, 40, 350, 40 );

 objPen.EndCap = LineCap.Flat;

 objGraphics.DrawLine( objPen, 10, 60, 350, 60 );
```

```
  objPen.EndCap = LineCap.Round;
  objGraphics.DrawLine( objPen, 10, 80, 350, 80 );
  objPen.EndCap = LineCap.RoundAnchor;
  objGraphics.DrawLine( objPen, 10, 100, 350, 100 );
  objPen.EndCap = LineCap.Square;
  objGraphics.DrawLine( objPen, 10, 120, 350, 120 );
  objPen.EndCap = LineCap.SquareAnchor;
  objGraphics.DrawLine( objPen, 10, 140, 350, 140 );
  objPen.EndCap = LineCap.Triangle;
  objGraphics.DrawLine( objPen, 10, 160, 350, 160 );
  objBitmap.Save( Response.OutputStream, ImageFormat.Gif );
}
</Script>
```

设置钢笔的线条接头样式,通过修改 LineJoin 属性值,控制钢笔绘制的线条接头的外观.

LineJoin 枚举

成员名称	说明
Bevel	指定成斜角的联接。这将产生一个斜角。
Miter	指定斜联接。这将产生一个锐角或切除角，具体取决于斜联接的长度是否超过斜联接限制。

MiterClipped 指定斜联接。这将产生一个锐角或斜角，具体取决于斜联接的长度是否超过斜联接限制。

Round 指定圆形联接。这将在两条线之间产生平滑的圆弧。

示例

```
<%@ Page ContentType="image/gif" Language="C#" %>

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<script language="C#" runat=server>

void Page_Load(Object sender , EventArgs e)

{

  Bitmap objBitmap;

  Graphics objGraphics;

  Pen objPen;

  objBitmap = new Bitmap( 400, 300 );

  objGraphics = Graphics.FromImage( objBitmap );

  objPen = new Pen( Color.White );

  objPen.Width = 10;

  objPen.LineJoin = LineJoin.Bevel;

  objGraphics.DrawRectangle( objPen, 10, 20, 350, 40 );
  objPen.LineJoin = LineJoin.Miter;

  objGraphics.DrawRectangle( objPen, 10, 80, 350, 40 );

 objPen.LineJoin = LineJoin.MiterClipped;

 objGraphics.DrawRectangle(objPen, 10, 140, 350, 40);

  objPen.LineJoin = LineJoin.Round;

  objGraphics.DrawRectangle( objPen, 10, 200, 350, 40 );

  objBitmap.Save( Response.OutputStream, ImageFormat.Gif );

}

</Script>
```

[转]GDI+基础(3)

常用图形绘制

```
<%@ Page ContentType="image/gif" Language="C#" %>

<!--ContentType 设置页面类型-->

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(600, 500);

 //创建一个宽 400，高 200 的实例

Graphics gph;

//从指定的 Image 对象创建新 Graphics 对象

gph = Graphics.FromImage(bmp);

gph.SmoothingMode = SmoothingMode.HighQuality;

//设置图片质量,指定是否将平滑处理（消除锯齿）应用于直线、曲线和已填充区域的边缘

gph.Clear(Color.Red);

//清除整个绘图面并以指定背景色填充

gph.DrawRectangle(Pens.Blue, 10, 10, 100, 100);

gph.FillRectangle(Brushes.Blue, 120, 10, 100, 100);
```

```
//绘制矩形
gph.DrawEllipse(Pens.Blue, 10, 120, 100, 100);
gph.FillEllipse(Brushes.Blue, 120, 120, 100, 100);
//绘制椭圆
gph.DrawPie(Pens.Blue, 10, 230, 100, 100, 0, 270);
//绘制圆弧
gph.FillPie(Brushes.Blue, 120, 230, 100, 100, 0, 270);
//绘制饼图
Point[] line={new Point(10,340),new Point(60,30),new Point(110,30),new Point(160,340)};
gph.DrawCurve(Pens.Blue, line);
//绘制曲线
gph.DrawBezier(Pens.Blue, new Point(10, 340), new Point(60, 30), new Point(110, 30), new Point(160, 340));
//绘制贝塞尔曲线
Point[] line2 ={ new Point(10, 340), new Point(340, 100), new Point(190, 340), new Point(10, 340) };
gph.DrawPolygon(Pens.Blue, line2);
//gph.FillPolygon(Pens.Blue, line2);
//绘制多边形
Bitmap mybit=new Bitmap(Server.MapPath("001.jpg"));
gph.DrawImage(mybit,10,360,100,100);
//绘制图片
```

```
gph.DrawLine(Pens.Black, 10, 480, 300, 480);

//绘制直线

bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的
图像的格式

//向客户端输出数据流，并以此数据流形成 Gif 图片

}

</script>
```

绘制文本字符串

```
<%@ Page ContentType="image/gif" Language="C#" %>

<!--ContentType 设置页面类型-->

<%@ Import namespace="System.Drawing" %>

<%@ Import namespace="System.Drawing.Imaging" %>

<%@ Import namespace="System.Drawing.Drawing2D" %>

<%@ Import namespace="System.Drawing.Text" %>

<script runat="server">

protected void Page_Load(object sender, EventArgs e)

{

Bitmap bmp = new Bitmap(600, 500);

 //创建一个宽 400，高 200 的实例
Graphics gph;
//从指定的 Image 对象创建新 Graphics 对象
gph = Graphics.FromImage(bmp);
gph.TextRenderingHint = TextRenderingHint.ClearTypeGridFit;
//设置字符串质量
gph.Clear(Color.Red);
//清除整个绘图面并以指定背景色填充
gph.DrawString("无换行显示:绘制文本换行显示的字符串,应该使用文本外部边界的长方行", new Font("宋体", 16), Brushes.Blue, 10, 10);
//绘制文本字符串 string, Font, Brush, float, float(字符串,字体,x,y 坐标)
RectangleF rect = new RectangleF(10, 110, 300, 200);
string str = "绘制文本换行显示的字符串,应该使用文本外部边界的长方行";
gph.DrawString(str, new Font("宋体", 16), Brushes.Blue,rect);
//绘制有范围的字符串
bmp.Save(Response.OutputStream, ImageFormat.Gif);//ImageFormat 对象，它指定保存的图像的格式
//向客户端输出数据流，并以此数据流形成 Gif 图片
}
</script>
```

设置图片质量

SmoothingMode 枚举 指定是否将平滑处理（消除锯齿）应用于直线、曲线和已填充区域的边缘

成员名称	说明
AntiAlias	指定消除锯齿的呈现。
Default	指定默认模式。
HighQuality	指定高质量、低速度呈现。
HighSpeed	指定高速度、低质量呈现。
Invalid	指定一个无效模式。
None	指定不消除锯齿。

```
<%@ ContentType="image/jpeg" Language="C#" %>
<%@ Import namespace="System.Drawing" %>
<%@ Import namespace="System.Drawing.Imaging" %>
<%@ Import namespace="System.Drawing.Drawing2D" %>
<%@ Import namespace="System.Drawing.Text" %>
<script language="C#" runat=server>
void Page_Load(Object sender , EventArgs e)
{
  Bitmap objBitmap;
  Graphics objGraphics;
```

```
Pen objPen;

Brush objBrush;

Font objFont;

objBitmap = new Bitmap( 400, 400 );

objGraphics = Graphics.FromImage( objBitmap );

objPen = new Pen( Color.Yellow );

objBrush = new SolidBrush( Color.Yellow );

objFont = new Font( "Lucida Sans Unicode", 18 );

objGraphics.SmoothingMode = SmoothingMode.Default;

objGraphics.DrawString( "Default", objFont, objBrush, 50, 20 );

objGraphics.DrawEllipse( objPen, 10, 10, 200, 50 );

objGraphics.SmoothingMode = SmoothingMode.AntiAlias;

objGraphics.DrawString( "AntiAlias", objFont, objBrush, 50, 80 );

objGraphics.DrawEllipse( objPen, 10, 70, 200, 50 );

objGraphics.SmoothingMode = SmoothingMode.HighQuality;

objGraphics.DrawString( "HighQuality", objFont, objBrush, 50, 140 );

objGraphics.DrawEllipse( objPen, 10, 130, 200, 50 );

objGraphics.SmoothingMode = SmoothingMode.HighSpeed;

objGraphics.DrawString( "HighSpeed", objFont, objBrush, 50, 200 );

objGraphics.DrawEllipse( objPen, 10, 190, 200, 50 );

objGraphics.SmoothingMode = SmoothingMode.None;

objGraphics.DrawString( "None", objFont, objBrush, 50, 260 );
```

```
 objGraphics.DrawEllipse( objPen, 10, 250, 200, 50 );

 objBitmap.Save( Response.OutputStream, ImageFormat.Jpeg );

}

</Script>
```

设置文本质量

TextRenderingHint 枚举 指定文本呈现的质量

成员名称 说明

AntiAlias 指定在无提示的情况下使用每个字符的 AntiAlias 标志符号位图来绘制字符。由于采用了 AntiAlias，质量会得到改善。由于关闭了提示，主干宽度差可能会比较明显。

AntiAliasGridFit 指定在有提示的情况下使用每个字符的 AntiAlias 标志符号位图来绘制字符。由于采用了 AntiAlias，质量会得到大大改善，但同时会增加性能成本。

ClearTypeGridFit 指定在有提示的情况下使用每个字符的标志符号 CT 位图来绘制字符。这是质量最高的设置。用于利用 ClearType 字体功能。

SingleBitPerPixel 指定使用每个字符的标志符号位图来绘制字符。不使用提示。

SingleBitPerPixelGridFit 指定使用每个字符的标志符号位图来绘制字符。提示用于改善字符在主干和弯曲部分的外观。

SystemDefault 指定在有系统默认呈现提示的情况下使用每个字符的标志符号位图来绘制字符。将采用用户为系统选择的所有字体修匀设置来绘制文本。

```
<%@ Page ContentType="image/jpeg" Language="C#" %>
<%@ Import namespace="System.Drawing" %>
<%@ Import namespace="System.Drawing.Imaging" %>
<%@ Import namespace="System.Drawing.Text" %>
<script language="C#" runat=server>
void Page_Load(Object sender , EventArgs e)
{
 Bitmap objBitmap;
 Graphics objGraphics;
 string strString;
 objBitmap = new Bitmap( 600, 400 );
 objGraphics = Graphics.FromImage( objBitmap );
 objGraphics.Clear( Color.DarkBlue );
 Font objFont = new Font( "Times", 24 );
 strString = "ABCabc123 - AntiAlias";
 objGraphics.TextRenderingHint = TextRenderingHint.AntiAlias;
 objGraphics.DrawString( strString, objFont, Brushes.White, 10, 10 );
 strString = "ABCabc123 - AntiAliasGridFit";
 objGraphics.TextRenderingHint = TextRenderingHint.AntiAliasGridFit;
 objGraphics.DrawString( strString, objFont, Brushes.White, 10, 50 );
 strString = "ABCabc123 - ClearTypeGridFit";
 objGraphics.TextRenderingHint = TextRenderingHint.ClearTypeGridFit;
```

```
  objGraphics.DrawString( strString, objFont, Brushes.White, 10, 90 );

  strString = "ABCabc123 - SingleBitPerPixel";

  objGraphics.TextRenderingHint = TextRenderingHint.SingleBitPerPixel;

  objGraphics.DrawString( strString, objFont, Brushes.White, 10, 130 );

  strString = "ABCabc123 - SingleBitPerPixelGridFit";

  objGraphics.TextRenderingHint = TextRenderingHint.SingleBitPerPixel;

  objGraphics.DrawString( strString, objFont, Brushes.White, 10, 170 );

  strString = "ABCabc123 - SystemDefault";

  objGraphics.TextRenderingHint = TextRenderingHint.SystemDefault;

  objGraphics.DrawString( strString, objFont, Brushes.White, 10, 210 );

  objBitmap.Save( Response.OutputStream, ImageFormat.Jpeg );

}

</Script>
```

解析.Net 框架下的 GDI+编程

目前微软的.Net 框架正进一步发展，1.1 版本即将发布，伴随而来是增加了诸如命名空间（Namespace）、Windows Form、GDI+和 CLR 等新概念、新机制。本文就来向大家介绍一下.Net 框架下 GDI+编程的一些基本知识。

GDI+是从 GDI 演化而来的，但是在 Visual Studio 的以前版本中使用 GDI 是相当复杂的，而且工作量巨大。现在在 GDI+中，微软已经帮我们解决了许多问题，因而使用 GDI+编程将变得非常容易。

GDI+包含在 System.Drawing.Dll 集合中，所有的 GDI+类包含在 System.Drawing，System.Text，System.Printing，System.Internal，System.Imaging，System.Drawing2D 以及 System.Design 等命名空间中。

本文先向大家介绍一下图形类（Graphics Class）。然后，会向大家介绍一些最常用的类和结构，包括画笔（Pen）、画刷（Brush）、字体（Font）、颜色（Color）等类或结构。文章的最后还给出了一些很有用的用.Net 框架的原生语言 C#实现的例子。

图形类（Graphics Class）

我们用图形类的对象来表示 GDI+的图形表面。为了使用 GDI+，我们必须先建立一个图形类对象。通常，我们可以从 Paint 这个事件获得图形类对象的一个引用或是通过重载方法 OnPaint 来取得该对象。具体方法如下：

```
private void form1_Paint(object sender, PaintEventArgs pe)
{
Graphics g = pe.Graphics;
}
```

或是：

```
protected override void OnPaint(PaintEventArgs pe)
{
Graphics g = pe.Graphics;
}
```

建立好图形类对象后，我们就可以调用以下一些方法来完成基本的画图功能了。

DrawArc	（已重载的）画一段弧线
DrawClosedCurve	（已重载的）画一段由一些点确定的闭合折线
DrawCurve	（已重载的）画一段由一些点确定的折线
DrawEllipse	（已重载的）画一个椭圆
DrawImage	（已重载的）画一副图象
DrawLine	（已重载的）画一条直线
DrawPath	画一段路径（包括直线和曲线）
DrawPie	（已重载的）画一个馅饼区的轮廓
DrawPolygon	（已重载的）画一个多边形的轮廓
DrawRectangle	（已重载的）画一个矩形的轮廓
DrawString	（已重载的）画一串字符串
FillEllipse	（已重载的）填充一个椭圆形区域
FillPath	填充一个路径
FillPie	（已重载的）填充一个馅饼区域
FillPolygon	（已重载的）填充一个多边形
FillRectangle	用画刷填充一个矩形
FillRectangles	用画刷填充一系列矩形
FillRegion	填充一个区域

图形对象

建立图形对象后，我们就可以用它来画线、填充图形以及画文本等等。以下是一些主要的图形对象：

Brush	用来填充特定的表面
Pen	用来画直线、多边形、矩形、弧线以及馅饼区等
Font	用来设置文本的字体
Color	用来设置特定对象的颜色（在 GDI+中，颜色可以是 Alpha 混合的）

画刷类（Brush Class）

画刷类是一个抽象基类，我们不能直接将它实例化。我们必须实例化它的子类对象，它的子类包括：SolidBrush，TextureBrush，RectangleGradientBrush 以及 LinearGradientBrush。

举例如下：

```
LinearGradientBrush  lBrush = new  LinearGradientBrush(rect,
Color.Red,
Color.Yellow,
LinearGradientMode.BackwardDiagonal);
Brush  brsh = new  SolidBrush(Color.Red), 40, 40, 140, 140);
```

SolidBrush 类定义一把由单色构成的画刷。这个画刷可以用来填充像矩形、椭圆形、馅饼形、多边形以及路径这样的图形区域。

TextureBrush 类定义一把可以将一定区域用图象来填充的画刷。

LinearGradiantBrush 类可以定义一把两种颜色间变化的画刷，也可以定义一把多种颜色间变化的画刷。

画笔类（**Pen Class**）

画笔类用来画具有特定宽度和风格的直线和曲线。我们必须先用画笔类的构造函数初始化一个画笔对象，在实例化的时候还可以用到颜色和画刷。

用特定的颜色初始化新的画笔对象：

public Pen(Color);

用特定的画刷初始化新的画笔对象：

public Pen(Brush);

用特定的画刷以及宽度初始化新的画笔对象：

public Pen(Brush, float);

用特定的颜色以及宽度初始化新的画笔对象：

public Pen(Color, float);

举例如下：

```
Pen  pn = new  Pen( Color.Blue );
```

或是：

```
Pen  pn = new  Pen( Color.Blue, 100 );
```

以下是画笔类的一些最常用的属性：

Alignment	获得或设置用画笔画的对象的边界
Brush	获得或设置决定画笔特性的画刷
Color	获得或设置画笔的颜色
Width	获得或设置画笔的宽度

字体类（**Font Class**）

字体类决定了特定文本的字体格式，比如：字体类型、大小以及风格。我们用字体类的构造函数建立一种字体。

用特定的属性初始化新的字体对象：

```
public Font(string, float);
```

用特定的已存在的字体和字体风格初始化新的字体对象：

```
public Font(Font, FontStyle);
```

以下是一些字体风格：

Bold	粗体
Italic	斜体
Regular	正常字体
Strikeout	有删除线

Underline	有下划线

举例如下：

```
Graphics  g ;
Font  font = new  Font("Times New Roman", 26);
```

颜色结构（Color Structure）

一个颜色结构代表一种 ARGB 格式的颜色。以下是它的 ARGB 属性：

A：获得颜色的 Alpha 成分值

B：获得颜色的蓝色成分值

G：获得颜色的绿色成分值

R：获得颜色的红色成分值

下面是如何使用颜色结构的例子：

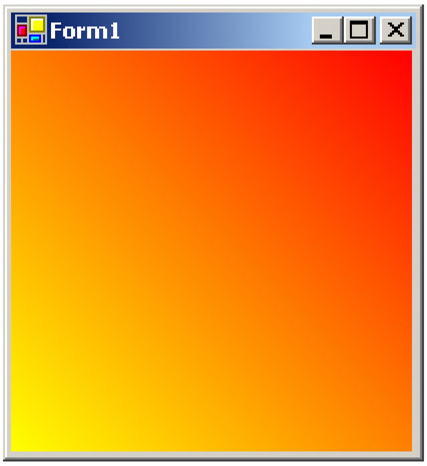
```
Pen  pn = new  Pen( Color.Blue );
```

到现在，我相信大家已经对.Net 框架下的 GDI+有了大致的了解。为了让大家更直观的理解有关 GDI+的编程知识，我特意为大家准备了以下一些基本但又很有用例子。通过对这些例子的学习，我相信大家对.Net 框架下的 GDI+编程会有更深刻的理解。同时要说明的是，下面的例子是用 C#语言实现的，如果你是一位 VB.net 的爱好者，不妨通过适当修改代码来实现同样的功能。

画一个矩形：

```
protected override void OnPaint(PaintEventArgs pe)
{
Graphics g = pe.Graphics ;
//设置矩形区域的位置和大小
Rectangle rect = new Rectangle(0, 0, 200, 200);
//使填充矩形的颜色从红色到黄色渐变
LinearGradientBrush lBrush = new LinearGradientBrush(rect, Color.Red,
Color.Yellow,
LinearGradientMode.BackwardDiagonal);
g.FillRectangle(lBrush, rect);
}
```

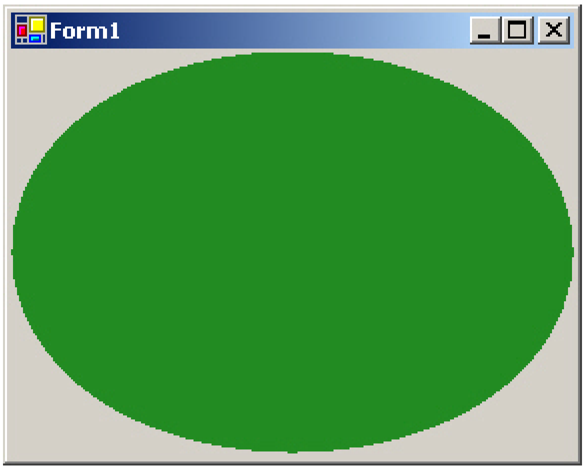
图示如下：



画一个椭圆：

```
protected override void OnPaint(PaintEventArgs pe)
{
Graphics g = pe.Graphics ;
//建立一只 100 象素宽、呈蓝色的画笔
Pen pn = new Pen( Color. ForestGreen, 100 );
Rectangle rect = new Rectangle(50, 50, 180, 100);
g.DrawEllipse( pn, rect );
}
```

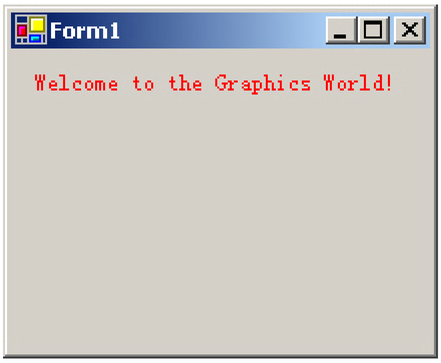
图示如下：



画一段文本：

```
protected override void OnPaint(PaintEventArgs pe)
{
Graphics g = pe.Graphics;
//文本内容为"Welcome to the Graphics World!"
g.DrawString("Welcome to the Graphics World!", this.Font, new
SolidBrush(Color.Red), 10,10);
}
```

图示如下：



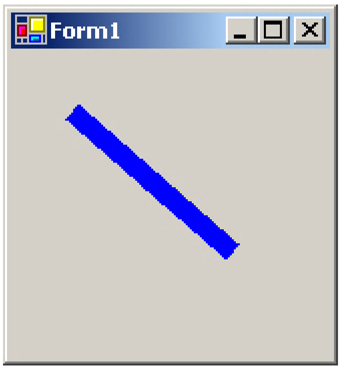
Welcome to the Graphics World!

画一条直线：

```
protected override void OnPaint(PaintEventArgs pe)
{
```

```
Graphics g = pe.Graphics ;
Pen pn = new Pen( Color.Blue, 10 );
//预先设定好两个点
Point pt1 = new Point( 30, 30);
Point pt2 = new Point( 110, 100);
g.DrawLine( pn, pt1, pt2 );
}
```

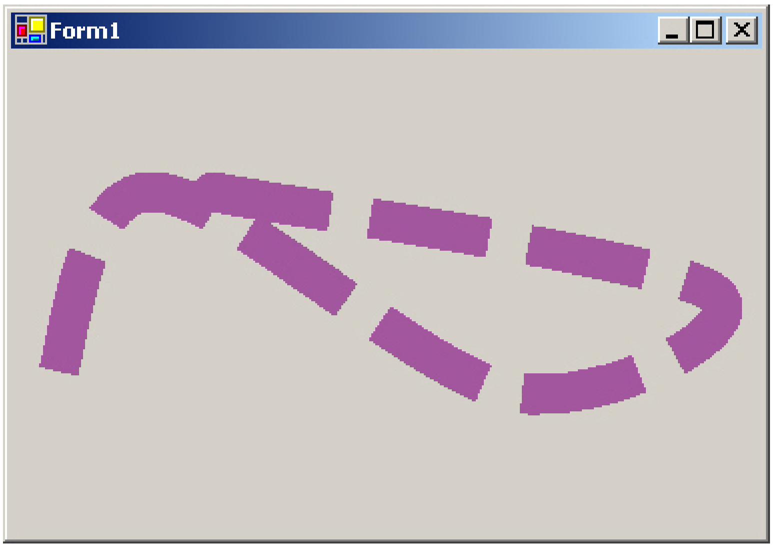
图示如下：



画一段折线：

```
protected override void OnPaint(PaintEventArgs pe)
{
Graphics g = pe.Graphics;
//建立一只 20 象素宽、呈粉红色且半透明的画笔
Pen penExample = new Pen(Color.FromArgb(150, Color.Purple), 20);
//使画笔画出虚线
penExample.DashStyle = DashStyle.Dash;
//将画笔开始和结束处都设置为圆形
penExample.StartCap = LineCap.Round;
penExample.EndCap = LineCap.Round;
//现在用画笔画曲线
g.DrawCurve(penExample, new Point[] {
new Point(100, 70),
new Point(350, 120),
new Point(250, 170),
new Point(70, 70),
new Point(20, 170),
});
}
```

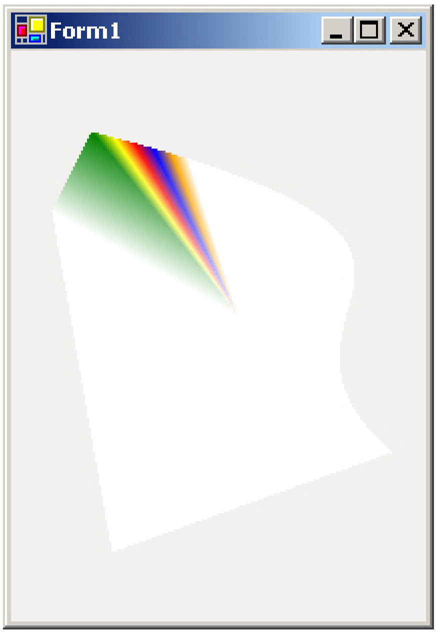
图示如下：



画一个填充颜色逐渐变化的多边形图形：

```
protected override void OnPaint(PaintEventArgs e)//这里是重载 OnPaint
函数
{
e.Graphics.TextRenderingHint =
System.Drawing.Text.TextRenderingHint.AntiAlias;
e.Graphics.FillRectangle(new
SolidBrush(Color.FromArgb(180,Color.White)),
ClientRectangle);
//建立图形路径
GraphicsPath path = new GraphicsPath(new Point[] {
new Point(40, 40),
new Point(275, 100),
new Point(105, 125),
new Point(190, 200),
new Point(50, 250),
new Point(20, 80),
}, new byte[] {
(byte)PathPointType.Start,
(byte)PathPointType.Bezier,
(byte)PathPointType.Bezier,
(byte)PathPointType.Bezier,
(byte)PathPointType.Line,
(byte)PathPointType.Line,
});
//建立一个 PathGradientBrush 对象
PathGradientBrush pgb = new PathGradientBrush(path);
pgb.SurroundColors = new Color[] {
Color.Green,
Color.Yellow,
Color.Red,
Color.Blue,
Color.Orange,
Color.White,
};
//最后进行填充
e.Graphics.FillPath(pgb, path);
}
```

图示如下：



（注：以上例子均需在 Windows2000 Server 版或 Windows XP Professional 版以及 Vs.net 环境下才能实现）

graph.Clear(backgroundColor01[i]);

为 image 填充一种颜色。

```
graph.DrawLine(new System.Drawing.Pen(System.Drawing.Color.Red), 1, 1, 30, 30);
```

在 image 上画一条从（1，1）到（30，30）的直线，直线的颜色是红色的。

```
System.Drawing.Drawing2D.LinearGradientBrush codeBrush = new LinearGradientBrush(new
Rectangle(0,0,image.Width,image.Height),System.Drawing.Color.While,System.Drawing.Color.Blue
,59,true);
```

同画刷的意思差不多，有五种画刷可供使用，都继承自 Brush 基类，LinearGradientBrush 是其中的一种，使用这种画刷可以实现线性渐变的功能，即写一个字，开头时是可以是红色，之后慢慢的渐变为蓝色画刷。它的构造函数，第一个参数是指定他在多大的范围里画画，第二个参数是画刷开始时的颜色，第三个参数是画刷结束时的颜色，第四、五个是高级应用，由于篇幅所限，暂时不做说明。

```
System.Drawing.Font codeFont=new Font("Arial
Bold",23,System.Drawing.FontStyle.Bold);
```

Font 定义字体的样式，如使用"宋体字"还是"罗马字"，字体大小等等。第一个参数是选择一个字体，第二个参数是字体的大小，第三个参数是"加粗、斜体"等信息。

```
graph.DrawString("aBA',codeFont,codeBrush,1 ,1);
```

将"aBA"写在 image 了，第一个参数是要写入的内容，第二个是 Font 类型，第三个是 Brush 类型，第四、五个参数是一个坐标，指示在 image 上从哪里开始写。

```
image.SetPixel(11,11,System.Drawing.Color.Yellow);
```

在 image 上画点，第一、二参数是一个坐标，指示画在哪里，第三个参数指示点的颜色。

```
graph.DrawRectangle(new
System.Drawing.Pen(System.Drawing.Color.Black),0,0,image.Width-1,image.Height-1);
```

为 image 画一个边框。Pen 的作用就像钢笔一样。

GDI+系列教程（二）：Graphics对象(C#) **ivanx**著 所有通过GDI+绘制的工作都会涉及到Graphics对象，本文介绍如何得到并操作Graphics对象，如何可以用它来做些什么等等...	C#,.NET 1.0,.NET 2.0 发表 **2007-10-22** 阅读 **2** 评论 **0**

翻译

The Graphics Object

简介

所有通过 GDI+绘制的工作都会涉及到 Graphics 对象，下面列举几种使用到它的场景：
1. 在绘制过程中，Windows Form 控件会通过 OnPaint 和 OnPaintBackground 方法传递 PaintEventArgs 参数，而 Graphics 对象就包括在其中；
2. 同样，该参数也被 OnPaint 事件引发的其它 Paint 事件所传递和处理；
3. 打印的时候，PrintPage 事件提供了 PrintPageEventArgs 参数，它也包括了操作打印机的 Graphics 对象，你可以直接操作这个对象绘制各种图形，它们会和显示在屏幕上一样被打印出来。

获得Graphics对象

当你写绘图程序的时候，你可以处理被封装在 PaintEventArgs 对象中的 Graphics 对象。这里有两种方法来控制 Graphics 对象。你可以重写保护类型的 OnPaint 或 OnPaintBackground 方法，或者，添加一个 Paint 事件句柄。在所有的这些场景中，Graphics 对象都是包含在 PaintEventArgs 对象中传递。

下面显示不同的方法：

```
protected override void OnPaint(PaintEventArgs e)
    {
      //get the Graphics object from the PaintEventArgs
      Graphics g=e.Graphics;
      g.DrawLine(....);

      //or use it directly
      e.Graphics.DrawLine(....);

      //Remember to call the base class or the Paint event won’t fire
      base.OnPaint (e);
    }
    //This is the Paint event handler
    private void Form1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
    {
      //get the Graphics object from the PaintEventArgs
      Graphics g=e.Graphics;
      g.DrawLine(....);

      //or use it directly
      e.Graphics.DrawLine(....);
    }
```

从上面的代码，可以看出来，你可以得到一个 Graphics 对象的引用，或者直接从 PaintEventArgs 中使用。

记住：千万别在这些方法的范围之外保存 Graphics 对象。

用它来做什么?

当你的程序已经得到 Graphics 对象后，你可以做以下几类事情，它们分成 5 类：

绘制图形(Stroke a shape): 例如使用画笔Pen对象绘制矩形、椭圆和线条。Pen对象有不同的线条宽度、颜色及其它许多属性，我们会在接下来的其它文章中细说。

填充图形(Fill a shape): 已经绘制出的图形，可以使用画刷Brush对象填充颜色。在GDI+中，Brush有许多复杂的设置，不过基本都是使用颜色填充一个面积。

绘制文本(Draw a string): 文本可以通过使用DrawString方法显示放置到Graphics的上面。

绘制图像(Draw an image): Image可以通过DrawImage方法以任何比例绘制。

修改Graphics对象(Modify the Graphics object): 有许多方法用于改变Graphics对象的效果。你可以在牺牲速度的基础上，提升图像的质量。你还可以改变Graphics的输出，做出一些旋转、缩放或扭曲等效果。

下面的代码演示如何绘制图形及填充图形：

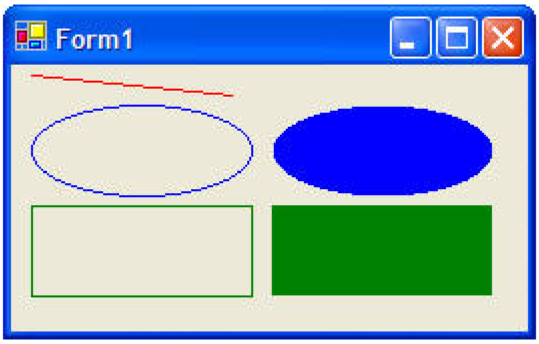


图 1 绘制及填充图形

```
protected  override void OnPaint(PaintEventArgs e)
    {
      //Get the Graphics object
      Graphics g=e.Graphics;
       //Draw a line
      g.DrawLine(Pens.Red,10,5,110,15);
       //Draw an ellipse
      g.DrawEllipse(Pens.Blue,10,20,110,45);
       //Draw a rectangle
      g.DrawRectangle(Pens.Green,10,70,110,45);
       //Fill an ellipse
      g.FillEllipse(Brushes.Blue,130,20,110,45);
       //Fill a rectangle
      g.FillRectangle(Brushes.Green,130,70,110,45);
      base.OnPaint (e);
    }
```

注意点：

总的来说，你必须记着只能使用系统传递给你的 Graphics 对象。

GDI+ is an "Immediate Mode" system. This is to say that the Graphics object has no knowledge of objects, only areas of colour. Shapes and lines are only remembered until they are painted over by a different colour.

你不可以在上面那些方法的作用范围之外使用 Graphics 对象，因为，它会在下次使用之前被销毁 Destroy。

Windows 窗体的.Net 框架绘图技术

[本文已于 06-9-27 在天极发表]

本文主要介绍 .Net 框架的基本绘图技术。通过简要的介绍和示例程序来探讨绘图技术的优势、劣势以及其它相关注意事项。

简介

幸运的是当编写一个典型的 Windows 窗体程序时，窗体和控件的绘制、效果等操作是不需要特别加以考虑的。这是为什么呢？因为通过使用 .Net 框架，开发人员可以拖动一系列的控件到窗体上，并书写一些简单的与事件相关联的代码然后在 IDE 中按 F5，一个完完全全的窗体程序就诞生了！所有控件都将自己绘制自己，窗体或者控件的大小和缩放都调整自如。在这里经常会用到的，且需要引起一点注意的就是控件效果。游戏，自定义图表控件以及屏幕保护程序的编写会需要程序员额外撰写用于响应 Paint 事件的代码。

本文针对那些 Windows 窗体开发人员并有助于他们在应用程序编制过程中使用简单的绘图技术。首先，我们会讨论一些基本的绘图概念。到底谁在负责进行绘制操作？Windows 窗体程序是如何知道何时该进行绘制的？那些绘制代码究竟被放置在哪里？之后，还将介绍图像绘制的双重缓冲区技术，你将会看到它是怎样工作的，怎样通过一个方法来实现缓存和实际显示的图像间的交替。最后，我们将会探讨”智能无效区域”，实际就是仅仅重绘或者清除应用程序窗体上的无效部分，加快程序的显示和响应速度。希望这些概念和技术能够引导读者阅读完本文，并且有助于更快和更有效的开发 Windows 窗体程序。

Windows 窗体使用 GDI+图像引擎，在本文中的所有绘图代码都会涉及使用托管的.Net 框架来操纵和使用 Windows GDI+图像引擎。

尽管本文用于基本的窗体绘图操作，但是它同样提供了快速的、有效的且有助于提高程序性能的技术和方法。所以，在通读本文之前建议读者对.Net 框架有个基本的了解，包括 Windows 窗体事件处理、简单的 GDI+对象譬如 Line，Pen 和 Brush 等。熟悉 Visual Basic .Net 或者 C#编程语言。

概念

Windows 应用程序是自己负责绘制的，当一个窗体”不干净”了，也就是说窗体改变了大小，或者部分被其它程序窗体遮盖，或者从最小化状态恢复时，程序都会收到需要绘制的信息。Windows 把这种”不干净”状态称为”无效的(Invalidated)”状态，我们理解为：需要重绘，当 Windows 窗体程序需要重绘窗体时它会从 Windows 消息队列中获取绘制的信息。这个信息经过.Net 框架封装然后传递到窗体的 PaintBackground 和 Paint 事件中去，在上述事件中适当的书写专门用于绘制的代码即可。

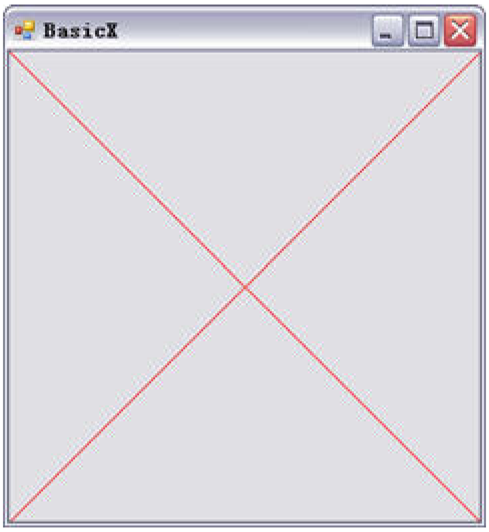
简单的绘图示例如下：

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class BasicX : Form
{
  public BasicX()
  {
    InitializeComponent();
  }

  private void BasicX_Paint(object sender, PaintEventArgs e)
  {
    Graphics g = e.Graphics;
    Pen p = new Pen(Color.Red);
    int width = ClientRectangle.Width;
    int height= ClientRectangle.Height;
    g.DrawLine(p, 0,0, width, height);
    g.DrawLine(p, 0, height, width, 0);
    p.Dispose();
  }

  private void InitializeComponent()
  {
    this.SetStyle(ControlStyles.ResizeRedraw, true);
    this.ClientSize = new System.Drawing.Size(300, 300);
    this.Text = "BasicX";
    this.Paint += new PaintEventHandler(this.BasicX_Paint);
  }

  [System.STAThreadAttribute()]
  public static void Main()
  {
    Application.Run(new BasicX());
  }
}
```

上述代码分成两个基本的步骤来创建示例程序。首先 InitializeComponent 方法包含一些属性的设置和附加窗体 Paint 事件的处理过程。注意，在方法中控件的样式也同时被设置，设置控件的样式也是自定义 Windows 窗体及控件行为的一种有效途径，譬如：控件的"ResizeRedraw"属性指示当窗体的大小变化发生以后需要对其完全进行重绘，也就是说重绘时总是需要对整个窗体的客户区域进行重绘。窗体的“客户区域”是指除了标题栏和边框的所有窗体区域。可以进行一个有趣的试验，取消该控件的属性然后再运行程序，我们可以很明显的看出为什么该属性会被经常的设置，因为窗体调整大小后的无效区域根本不会被重绘。

好了，我们需要注意一下 BasicX_Paint 方法，正如先前所提到的，Paint 事件在程序需要重绘时被激活，程序窗体利用 Paint 事件来负责回应需要重绘的系统消息，BasicX_Paint

方法的调用需要一个对象 sender 和一个 PaintEventArgs 类型的变量，PaintEventArgs 类的实例或称之为变量 e 封装了两个重要的数据，第一个就是窗体的 Graphics 对象，该对象表示窗体可绘制的表面也称之为画布用于绘制诸如线、文本以及图像等，第二个数据就是 ClipRectangle，该 Rectangle 对象表示窗体上无效的的矩形范围，或者说就是窗体需要重绘的区域。记住，当窗体的 ResizeRedDraw 设置后，调整大小后该 ClipRectangle 的大小实际就等于窗体整个客户区域的大小，或者是被其它程序窗体遮盖的那部分剪切区域。关于部分剪切区域的用处我们会在智能重绘章节作更详细的阐述。



BasicX 示例程序的运行界面

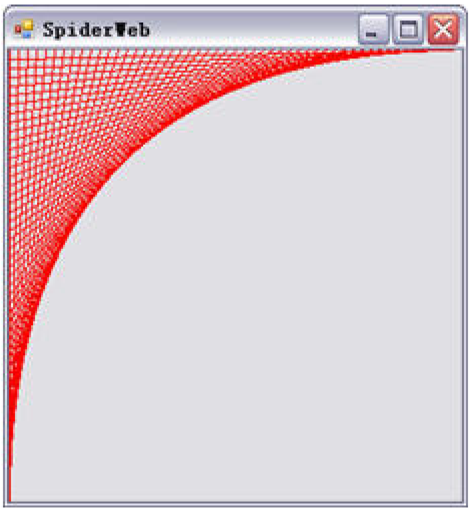
双重缓冲区绘图技术

双重缓冲区技术能够使程序的绘图更加快速和平滑，有效减少绘制时的图像闪烁。该技术的基本原理是先将图像绘制到内存中的一块画布上，一旦所有的绘制操作都完成了，再将内存中的画布推到窗体的或者控件的表面将其显示出来。通过这种操作后的程序能使用户感觉其更加快速和美观。

下面提供的示例程序能够阐明双重缓冲区的概念和实现方法，这个示例所包含的功能已相当完整，且完全可以在实际应用中使用。在该章节后面还会提及该技术应该配合控件的一些属性设置才能达到更好的效果。

要想领略双重缓冲区绘图技术所带来的好处就请运行 SpiderWeb 示例程序吧。程序启动并运行后对窗口大小进行调整，你会发现使用这种绘图算法的效率不高，并且在调整大小的过程

中有大量的闪烁出现。



不具备双重缓冲区技术的 SpiderWeb 示例程序

纵观程序的源码你会发现在程序Paint 事件激活后是通过调用LineDrawRoutine方法来实现线的绘制的。LineDrawRoutine 方法有两个参数，第一个是 Graphics 对象是用于绘制线条的地方，第二个是绘图工具 Pen 对象用来画线条。代码相当简单，一个循环语句，LINEFREQ 常量等，程序从窗体表面的左下一直划线到其右上。请注意，程序使用浮点数来计算在窗体上的绘制位置，这样做的好处就是当窗体的大小发生变化时位置数据会更加精确。

```
private void LineDrawRoutine(Graphics g, Pen p)
{
  float width = ClientRectangle.Width;
  float height = ClientRectangle.Height;
  float xDelta = width / LINEFREQ;
  float yDelta = height / LINEFREQ;
  for (int i = 0; i < LINEFREQ; i++)
  {
    g.DrawLine(p, 0, height - (yDelta * i), xDelta * i, 0);
  }
}
```

撰写很简单的用于响应 Paint 事件 SpiderWeb_Paint 的代码，正如前面所提到的，Graphics 对象就是从 Paint 事件参数 PaintEventArgs 对象中提取出来的表示窗体的绘制表面。这个

Graphics 对象连同新创建 Pen 对象一起传递给 LineDrawRoutine 方法来画出蜘蛛网似的线条，使用完 Graphics 对象和 Pen 对象后释放其占用的资源，那么整个绘制操作就完成了。

```
private void SpiderWeb_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  Pen redPen = new Pen(Color.Red);
  LineDrawRoutine(g, redPen);
  redPen.Dispose();
  g.Dispose();
}
```

那么到底作怎么样的改动才能使上面的 SpiderWeb 程序实现简单的双重缓冲区技术呢？原理其实相当简单，就是将应该画到窗体表面的绘制操作改成先画到内存中的位图上，LineDrawRoutine 向这个在内存中隐藏的画布执行同样的蜘蛛网绘制操作，等到绘制完毕再通过调用 Graphics.DrawImage 方法将隐藏的画布上内容推到窗体表面来显示出来，最后，再加上一些小的改动一个高性能的绘图窗体程序就完成了。

请比较下面双重缓冲区绘图事件与前面介绍的简单绘图事件间的区别：

```
private void SpiderWeb_DblBuff_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  Pen bluePen = new Pen(Color.Blue);
  Bitmap localBitmap = new Bitmap(ClientRectangle.Width,ClientRectangle.Height
);
  Graphics bitmapGraphics = Graphics.FromImage(localBitmap);
  LineDrawRoutine(bitmapGraphics, bluePen);
  //把在内存里处理的 bitmap 推向前台并显示
  g.DrawImage(localBitmap, 0, 0);
  bitmapGraphics.Dispose();
  bluePen.Dispose();
  localBitmap.Dispose();
  g.Dispose();
}
```

上面的示例代码创建了内存位图对象，它的大小等于窗体的客户区域(就是绘图表面)的大小，通过调用 Graphics.FromImage 将内存中位图的引用传递给 Graphics 对象，也就是说后面所有对该 Graphics 对象的操作实际上都是对内存中的位图进行操作的，该操作在 C++中等同于将位图对象的指针复制给 Graphics 对象，两个对象使用的是同一块内存地址。现在 Graphics 对象表示的是屏幕后方的一块画布，而它在双重缓冲区技术中起到至关重要的作用。所有的线条绘制操作都已经针对于内存中的位图对象，下一步就通过调用 DrawImage 方法将该位图复制到窗体，蜘蛛网的线条就会立刻显示在窗体的绘制表面而且丝毫没有闪烁出现。

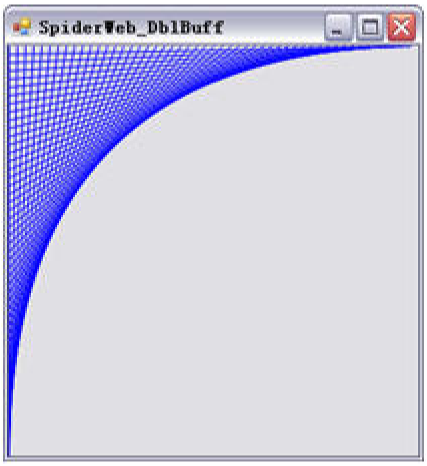
这一系列的操作完成后还不是特别有效，因为我们先前提到了，控件的样式也是定义 Windows 窗体程序行为的一条途径，为了更好的实现双重缓冲区必须设置控件的 Opaque 属性，这个属性指明窗体是不负责在后台绘制自己的，换句话说，如果这个属性设置了，那么必须为清除和重绘操作添加相关的代码。具备双重缓冲区版本的 SpiderWeb 程序通过以上的设置在每一次需要重绘时都表现良好，窗体表面用其自己的背景色进行清除，这样就更加减少了闪烁的出现。

```
public SpiderWeb_DblBuff()
{
  SetStyle(ControlStyles.ResizeRedraw | ControlStyles.Opaque, true);
}

private void SpiderWeb_DblBuff_Paint(object sender, PaintEventArgs e)
{
  Bitmap localBitmap = new Bitmap(ClientRectangle.Width,
  ClientRectangle.Height);
  Graphics bitmapGraphics = Graphics.FromImage(localBitmap);
  bitmapGraphics.Clear(BackColor);
  LineDrawRoutine(bitmapGraphics, bluePen);
}
```

结果怎么样？图像的绘制平滑多了。从内存中将蜘蛛网的线条推到前台以显示出来是完全没有闪烁的，但是我们还是稍微停顿一下，先将内存中的位图修整一下再显示出来，可以添加一行代码以便使线条看上去更加平坦。

```
    bitmapGraphics.SmoothingMode = SmoothingMode.AntiAlias;
```

在将内存中的位图对象赋给 Graphics 后通过放置这行代码，我们在画布上所画的每一个线条都使用了反锯齿，使凹凸不平的线条显得更加平坦。



具备双重缓冲区技术的且使用 AntiAliasing（反锯齿）属性的 SpiderWeb_DblBuff 示例程序

完成了简单的双重缓冲区应用后有两个问题需要向读者阐明，.Net 中的某些控件例如：Button、PictureBox、Label 还有 PropertyGrid 都已经很好的利用了该技术！这些控件在默认状态下会自动启用双重缓冲区技术，用户可以通过对“DoubleBuffer”属性的设置来就可以实现双重缓冲区技术。所以，用户若使用 PictureBox 来绘制蜘蛛网将会更有效率一些，而且也使程序变得更加简单了。

我们在这里讨论的双重缓冲区技术既不是完全被优化但也没有什么太大的负面影响。双重缓冲区技术是减少 Windows 窗体绘制时闪烁的一条重要途径，

但是它也确实消耗不少内存，因为它将会使用双倍的内存空间：应用程序所显示的图像和屏幕后方内存中的图像。每次 Paint 事件被激活时都会动态的创建位图对象，这种机制会相当耗费内存。而自带双重缓冲区技术的控件在使用 DoubleBuffer 属性后执行起来的优化程度则会更好一些。

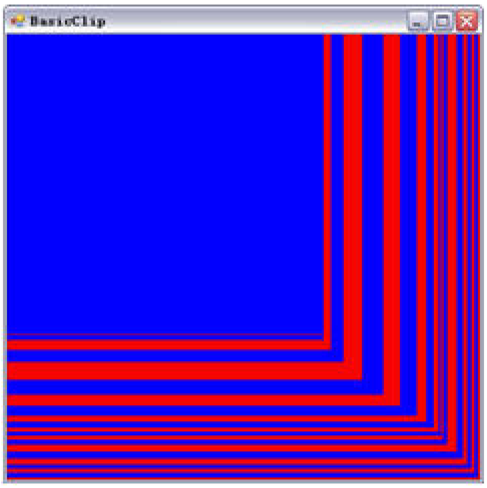
使用 GDI+的 DIB(与设备无关的位图)对象来实现这种画面以外的内存缓冲，自带双重缓冲区机制的控件则能好的利用该位图对象。DIB 是底层 Win32 的对象用于高效的屏幕绘制。同样，值得注意的是 GDI+的第一个版本 GDI 中仅与硬件加速有关以及一些简单功能可以直接使用，由于这样的限制，像反锯齿和半透明等屏幕绘制方法执行起来的速度则相当慢。尽管双重缓冲区机制消耗了一些内存但是它的使用不容置疑的增强了程序的执行性能。

智能重绘，在绘制前需要斟酌一下

“智能无效”(智能重绘)就是在暗示程序员应该明白仅应对程序中无效的区域进行重绘，对 Regions 对象所对应的无效区域进行重绘可以提高绘制性能，使用 Regions 对象你可以仅排除或绘制控件和窗体的部分区域已获得更好的性能。我们现在就开始来看一下 BasicClip 示例程序，这个程序使用保存在 PaintEventArgs 对象的 ClipRectangle 对象，之前我们已经提及，无论何时当程序的大小发生变化时 Paint 事件都会被激活。BasicClip 示例程序用红和蓝两种颜色填充剪切的矩形区域，利用不同的速度调整窗体的大小几次以后，你会发现绘制的矩形区域其实就是窗体的无效区域(包括大于原始窗体大小的区域部分和缩少了的区域部分)，示例程序的 Paint 事件代码如下：

```
private void BasicClip_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  if (currentBrush.Color == Color.Red)
    currentBrush.Color = Color.Blue;
  else
    currentBrush.Color = Color.Red;
  g.FillRectangle(currentBrush, e.ClipRectangle);
  g.Dispose();
}
```

该示例程序的唯一目的就是演示怎样仅针对部分区域进行图形绘制。



BasicClip 示例程序中的彩色矩形区域就是表示窗体的下方和右侧的无效区域。

Regions 是一种被用来定义 Windows 窗体或者控件区域的对象，调整窗体大小后所获得的 Regions 就是窗体重绘的最小区域。当程序需要进行绘制的时候仅绘制感兴趣的特殊区域，这样绘制更小的区域就会使程序的运行速度更快。

为了更好的演示 Regions 的用法，请查看 TextCliping 示例程序。该程序重载了 OnPaintBackground 和 OnPaint 方法，直接重载这些方法比侦听事件更能保证代码在其它的绘制操作之前被调用，而且对于自定义控件的绘制也更加有效。为了清楚起见，示例程序提供了一个 Setup 方法，该方法定义了全局的 Graphics 对象。

```
private void Setup()
{
  GraphicsPath textPath = new GraphicsPath();
  textPath.AddString(displayString, FontFamily.GenericSerif,
    0, 75, new Point(10, 50), new StringFormat());
  textRegion = new Region(textPath);
  backgroundBrush = new TextureBrush(new Bitmap("CoffeeBeanSmall.jpg"),
  WrapMode.Tile);
  foregroundBrush = new SolidBrush(Color.Red);
}
```

上面的 Setup 方法首先定义一个空的 GraphicsPath 对象变量 textPath，下一步字符串“Windows Forms”的边界被添加到该路径中，围绕这个轮廓创建 Region。这样，一个被绘制在窗体表面的以字符串轮廓为区域的 Region 就被创建了。最后，Setup 方法创建以材质刷子为背景和以实色刷子为前景来绘制窗体。

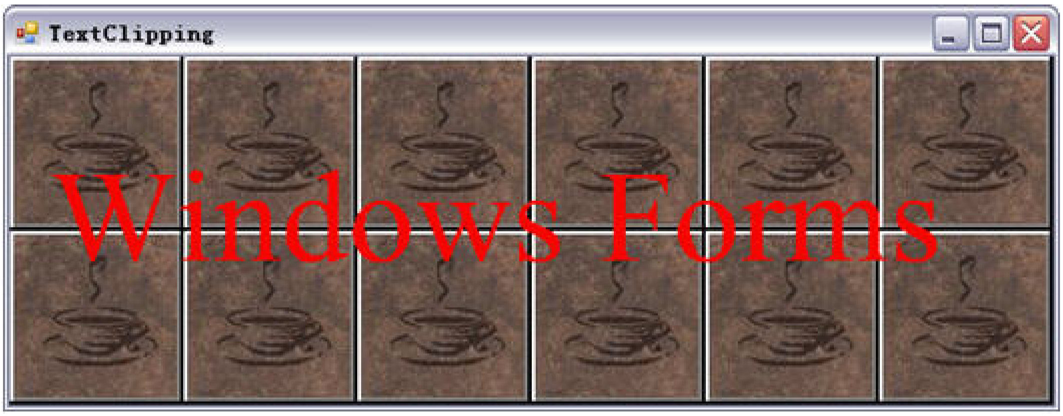
```
protected override void OnPaintBackground(PaintEventArgs e)
{
  base.OnPaintBackground(e);
  Graphics bgGraphics = e.Graphics;
  bgGraphics.SetClip(textRegion, CombineMode.Exclude);
  bgGraphics.FillRectangle(backgroundBrush, e.ClipRectangle);
  bgGraphics.Dispose();
}
```

上面定义的 OnPaintBackground 方法先立刻调用基类方法，这能够保证所有底层绘制的代码都能够被执行。下一步，从 PaintEventArgs 中获得 Graphics 对象，再将 Graphics 对象的剪切区域定义为 textRegion 对象。通过指定 CombineMode.Exclude 参数，明确无论在哪里绘制或怎样绘制 Graphics 对象都不绘制 textRegion 区域内部。

```
protected override void OnPaint(PaintEventArgs e)
{
  base.OnPaint(e);
  Graphics fgGraphics = e.Graphics;
  fgGraphics.FillRegion(foregroundBrush, textRegion);
  fgGraphics.Dispose();
}
```

最后，OnPaint 事件负责精确的绘制出字符串。可以很容易的通过调用 Graphics 的 FillRegion 方法来实现。通过指定的前景刷子 foregroundBrush 和 textRegion 且仅是该区

域被绘制。结果，Windows 窗体程序在运行之前确实“思考”该怎样进行绘制。



TextClipping 示例程序，通过 Region 定义的 Windows Forms 字符串。能够使程序在绘制时避开一个区域。

适当的组合使用区域和智能重绘你可以编写出运行速度快且不会引起闪烁的绘制代码，并且比单独使用双重缓冲区绘制还要节省内存的消耗。

结论

如果你的程序确定要进行绘制操作，使用几种技术可以增强绘制性能。确保争取设置控件属性以及适当的 Paint 事件处理是编写健壮程序的开始。在权衡好利弊后可以使用双重缓冲区技术产生非常“保护视力”的结果。最后，在实际绘制前进行思考到底哪些客户区域或 Region 需要被绘制将非常有益。希望通过这篇文章能够使读者更好的理解关于.net 框架的绘制技术及其应用。

.NET Windows编程系列(8)：.NET图形和图像编程

Posted on 2007-10-29 16:14 李永京 阅读(609) 评论(0) 编辑 收藏 所属分类: WinForm

.

.NET Windows 编程系列(8)：.NET 图形和图像编程

摘要：

在本节介绍我们如何在 Windows 界面上绘制各种图形、如何利用.NET FrameWork 对图像进行各种处理。

注：本系列节选自MSDN Webcasts上邵志东老师的《 .NET Windows编程系列课程》PPT内容，在我的Blog中整理发表，方便大家一同参考！此系列相关资料请到这里下载。

本节主要内容：

.NET 图形编程概述

画笔和画刷

图形的绘制

文本和字体

双缓冲

一、**.NET** 图形编程概述

GDI+技术简介

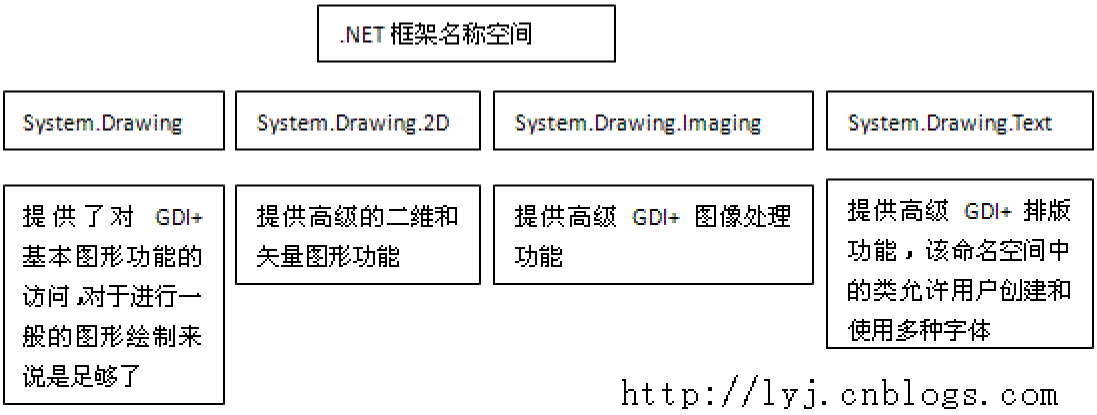
GDI：Graphics Device Interface.

GDI+是一种构成 Microsoft Windows XP 操作系统的子系统的应用程序编程接口(API)。GDI+ 负责在屏幕和打印机上显示信息。它是 GDI 的改进，同时也是.NET 框架结构的重要组成部分。和 GDI 一样它提供料对二维图形图像和文字排版处理的支持．通过 GDI+能够创建与设备无关的应用程序。

GDI+提供的新特性：

Alpha 混合技术、反锯齿处理技术、渐变色和纹理填充、宽线条、基本几何曲线样式、可缩放区域、浮点数坐标、嵌入画笔、高质量过滤和缩放、多种线条样式和端点选项

.NET 框架结构中对 GDI 的封装



提供了对 GDI+基本图形功能的访问,对于进行一般的图形绘制来说是足够了

提供高级的二维和矢量图形功能

提供高级 GDI+ 图像处理功能

提供高级 GDI+ 排版功能，该命名空间中的类允许用户创建和使用多种字体

坐标系统

GDI+在坐标系中绘制直线、矩形和其他形状。我们可以从各种各样的坐标系统中选择，但默认坐标系统的原点是在左上角，并且 x 轴指向右边，y 轴指向下边。默认坐标系统的度量单位是像素。

System.Drawing 中常用的结构

1．Color

Color 封装了对颜色的定义。该结构中封装了 数百个分别对应与标准调色板色彩的静态成员。如 Color.Red 代表红色，Color.Purple 代表紫色

有用的静态方法：

FromArgb:通过三原色构建 Color 对象

FromKnownColor:通过已知颜色构建 Color 对象

FromName:通过颜色名称来构建 Color 对象

例如：

Color temp1 = Color.Black;

Color temp2 = Color.FromArgb(0,0,0)

Color temp3 = Color.FromName(“Black”);

2．Size 和 SizeF

表示绘制平面上的一个尺寸，一个为整数，一个为浮点数

构造函数

Size sz1 = new Size(10,10)

属性：

Width：表示宽度值

Height：表示高度值

重载了加、减、比较、赋值操作

Size sz2 = sz1;

Size sz3 = sz1 -sz2;

Size sz4 = sz1+sz2;

3．Point 和 PointF

表示绘制平面上点的坐标，一个为整数，另外一个为浮点数

构造方法：

Point pt = new Point(20,20);

Point pt = new Point(new Size(10,10))

重载了加、减、比较、赋值操作

4．Rectangle 和 RectangleF

表示绘制平面上的一个矩形区域

属性

Bottom：矩形底部的纵座标

Top：矩形顶部的纵座标

Left：矩形坐部的横座标

Right：矩形右部的横座标

Height：矩形的高度

Width：矩形的宽度

Size：矩形的尺寸

IsEmpty：矩形是否为空（高度和宽度是否都是 0）

X：矩形左上角的横座标

Y：矩形左上角的纵座标

Graphics 类

使用 GDI+绘图，首先要创建 Graphics 类

1．Graphics 使用的两种方法（模板）

在 OnPaint 事件中使用

```
Protected override void OnPaint(PaintEventArgs e)
{
    Graphics g = e.Graphics;
}
```

在其他情况使用

```
Graphics g = this.CreateGraphics();
Try
{
    //作图
}
Finally
{
    if(g!=null)
        ((IDisposable)g).Dispose();
}
```

2．Graphics 对象绘图方法

DrawArc—绘制圆弧、DrawBezier—绘制贝塞尔曲线、DrawBeziers—绘制贝塞尔曲线组、DrawClosedCurve—绘制封闭曲线、DrawCurve—绘制曲线、DrawEllipse—绘制椭圆、DrawIcon—绘制图标、DrawIconUnstretched—无缩放绘制图标、DrawImage—绘制图像、DrawImageUnscaled—无缩放绘制图像、DrawLine—绘制直线、DrawLines—绘制直线组、DrawPath—绘制 GraphicsPath 对象、DrawPie—绘制圆饼、DrawPolygon—绘制多边形、DrawRectangle—绘制矩形、DrawRectangles—绘制矩形组、DrawString—绘制文本

3．Graphics 对象绘制实心图形方法

FillClosedCurve—绘制实心封闭曲线、FillEllipe—封闭实心椭圆、FillPath—GraphicsPath 对象、FillPie—绘制实心圆饼、FillPolygon—绘制实心多边形、FillRectangle—绘制实心矩形、FillRectangles—绘制实心矩形组、FillRegion—绘制实心 Region 对象

DEMO1：图形编程示例

二、画笔和画刷

Pen

在 System.Drawing 名称空间中，用来指定图形的轮廓，如颜色和宽度等。

1．画笔创建

Pen pen = new Pen(Color.Blue,5)

2．画笔的属性

属性	描述	取值
Alignment	指定相对于理论上、零宽度的线条的 Pen 对象的对齐方式	PenAlignment.Center：位于所绘制线条的中央 PenAlignment:Insert：位于所绘制线条的嵌入内部 PenAlignment.Left：位于所绘制线条的左侧 PenAlignment.OutSet：位于所绘制线条的嵌入外部 PenAlignment.Right：位于所绘制线条的右侧
DashStyle	绘制线条的虚线类型	DashStyle.Custom：用户自定义 DashStyle.Dash：线条由线段组成 DashStyle.DashDot：线条由线段和点组成 DashStyle.DashDotDot：线条由线段、点和点组成 DashStyle.Dot：线条由点组成 DashStyle.Solid：线条由实线组成
StartCap EndCap	绘制线条的起点和终点类型	AnchorMask 指定用于检查线帽是否为锚头帽的掩码。ArrowAnchor 指定箭头状锚头帽。Custom 指定自定义线帽。DiamondAnchor 指定菱形锚头帽。Flat 指定平线帽。 NoAnchor 指定没有锚。Round 指定圆线帽。 RoundAnchor 指定圆锚头帽。Square 指定方线帽。 SquareAnchor 指定方锚头帽。Triangle 指定三角线帽。

DEMO2：Pen 使用示例

Brush

Brush 对象是一个抽象类，不能被直接使用。它有 5 个派生类，分别实行不同类型的画刷。

画刷类型：SolidBrush：实心画刷（最简单）、HatchBrush：带阴影线的画刷、LinearGradientBrush：填充颜色线性渐变的画刷、PathGradientBrush：填充颜色沿路径渐变的画刷、TextureBrush：使用图像进行填充的画刷。

DEMO3：Brush 使用示例

三、绘制的图形

直线的绘制

DrawLine、DrawLines

DEMO4：线条绘制

圆弧、矩形和椭圆

1．圆弧

确定矩形边框，指定起始角度和跨越角度。角度以度数为单位，即 360 表示一周

2．椭圆：确定外接矩形

绘制曲线

DrawCurve、DrawClosedCurve、DrawBezier

练习

把直线练习修改为 DrawCurve

把直线练习修改为 DrawClosedCurve

绘制路径（**GraphicsPath**）

路径是一系列图形的组合

GraphicsPath 的方法

描述	方法
路径形成	? AddArc ? AddBezier ? AddBeziers ? AddClosedCurve ? AddEl ipse ? AddLine ? AddLines ? AddPath ? AddPie ? AddPolygon ? AddRectangle ? AddRectangles ? AddString

路径填充	? FillPath

DEMO5：路径绘制

四、文本和字体

Font 类

FontFamiliy：字体家族，如 Times New Roman、宋体等

字体大小：float 类型

字体风格

Bold：粗体

Italic：斜体

Regular：正规

Strikeout：加删除线

Underline：加下划线

例如：

Font myFont = new Font("宋体",16,FontStyle.Bold|FontStyle.Italic);

以上代码创建了宋体家族的字体对象，字体大小为 16，样式为粗斜体。

DrawString 的使用方法

DrawString(string,Font,Brush,PointF);

DrawString(string,Font,Brush,RectangleF);

DrawString(string,Font,Brush,PointF,StringFormat);

DrawString(string,Font,Brush,RectangleF,StringFormat);

DrawString(string,Font,Brush,float,float);

DrawString(string,Font,Brush,float,float,StringFormat);

DEMO6：文字绘制

反锯齿

消除锯齿：指对绘制的图形和文本的粗糙边缘进行平滑处理以改进它们的外观或可读性。

修改 Graphics 对象的 TextRenderingHint 属性，取值为：TextRenderingHint.AntiAlias、TextRenderingHint.AntiAliasGridFit、TextRenderingHint.ClearTypeCridFit、TextRenderingHint.SingleBitPerPixel、TextRenderingHint.SingleBitPerPixelGridFit、TextRenderingHint.SystemDefault

位于 System.Drawing.Text 名称空间中

TextRenderingHint 属性

属性	描述
AntiAlias	在无提示的情况下使用每个字符的消除锯齿效果标志符号位图来绘制字符。由于采用了 AntiAlias，质量会得到改善。由于关闭了提示，主干宽度差可能会比较明显。
AntiAliasGridFit	在有提示的情况下使用每个字符的消除锯齿效果标志符号位图来绘制字符。由于采用了 AntiAlias，质量会得到大大改善，但同时会增加性能成本。
ClearTypeGridFit	在有提示的情况下使用每个字符的标志符号 ClearType 位图来绘制字符。这是质量最高的设置。用于利用 ClearType 字体功能。
SingleBitPerPixel	使用每个字符的标志符号位图来绘制字符。不使用提示。
SingleBitPerPixelGridFit	使用每个字符的标志符号位图来绘制字符。提示用于改善字符在主干和弯曲部分的外观。
SystemDefault	有系统默认呈现提示的情况下使用每个字符的标志符号位图来绘制字符。将采用用户为系统选择的任何字体修匀设置来绘制文本。

DEMO7：反锯齿示例

五、双缓冲

闪烁是图形编程的一个常见问题。需要多重复杂绘制操作的图形操作会导致呈现的图像闪烁或具有其他不可接受的外观。

为解决这些问题，.NET Framework2.0 提供了对双缓冲的使用。

双缓冲使用内存缓冲区来解决由多重绘制操作造成的闪烁问题。当启用双缓冲时，所有绘制操作首先呈现到内存缓冲区，而不是屏幕上的绘图图面。所有绘制操作完成后，内存缓冲区直接复制到与其关联的绘图图面。因为在屏幕上只执行一个图形操作，所以消除了由复杂绘制操作造成的图像闪烁。

使用双缓冲

默认情况下，标准 Windows 窗体控件是双缓冲的。

可以通过两种方法对窗体和所创作的控件启用默认双缓冲。

一种方法是将 DoubleBuffered 属性设置为 true，另一种方法是通过调用 SetStyle 方法将 OptimizedDoubleBuffer 标志设置为 true。两种方法都将为窗体或控件启用默认双缓冲并提供无闪烁的图形呈现。建议仅对已为其编写所有呈现代码的自定义控件调用 SetStyle 方法。

双缓冲相关类

BufferedGraphicsContext：负责单独分配和管理图形缓冲区。Allocate 方法可以创建 BufferedGraphics。

BufferedGraphics：为双缓冲提供图形缓冲区。可以用 Render()方法写入图形缓冲区的内容。

手动管理缓冲图形

```
BufferedGraphicsContext currentContext= BufferedGraphicsManager.Current;

BufferedGraphics myBuffer = currentContext.Allocate(Form1.CreateGraphics(),

Form1.DisplayRectangle);

myBuffer.Graphics.DrawEllipse(Pens.Blue, Form1.DisplayRectangle);

myBuffer.Render();

myBuffer.Dispose();
```

DEMO8：双缓冲

本系列文章快速导航：

- .NET Windows编程系列(1)：.NET Windows编程概述 （2007-10-19 发布）
- .NET Windows编程系列(2)：C#语法和面向对象编程 （2007-10-20 发布）
- .NET Windows编程系列(3,4)：VisualStudio2005 常用控件（2007-10-21 发布）
- .NET Windows编程系列(5)：VisualStudio2005 自定义控件开发（2007-10-21 发布）
- .NET Windows编程系列(6)：对话框编程（2007-10-23 发布）
- .NET Windows编程系列(7)：.NET菜单、工具栏、状态栏编程（2007-10-25 发布）
- .NET Windows编程系列(8)：.NET图形和图像编程（2007-10-29 发布）

- .NET Windows编程系列(9)：.NET程序交互（2007-11-5 发布）
- .NET Windows编程系列(10)：.NET图像处理(上) （2007-11-6 发布）
- .NET Windows编程系列(11)：.NET图像处理(下) （2007-11-7 发布）
- .NET Windows编程系列(12)：.NET文件和流编程（2007-11-8 发布）
- .NET Windows编程系列(13)：.NET注册表编程（2007-11-9 发布）
- .NET Windows编程系列(14)：Windows服务（2007-11-12 发布）
- .NET Windows编程系列(15)：.NET多线程编程（2007-11-13 发布）
- .NET Windows编程系列(16)：.NET网络编程（2007-11-14 发布）

所属分类的其他文章:

· .NET Windows编程系列(16)：.NET网络编程
· .NET Windows编程系列(15)：.NET多线程编程
· .NET Windows编程系列(14)：Windows服务
· .NET Windows编程系列(13)：.NET注册表编程
· .NET Windows编程系列(12)：.NET文件和流编程
· .NET Windows编程系列(11)：.NET图像处理(下)
· .NET Windows编程系列(10)：.NET图像处理(上)
· .NET Windows编程系列(9)：.NET程序交互
· .NET Windows编程系列(7)：.NET菜单、工具栏、状态栏编程

博客园首页 小组 博问 闪存 新闻频道 招聘频道 专题

.NET Windows编程系列(10)：.NET图像处理(上)

摘要：

在.NET 编程中，由于 GDI+的出现，使得对于图像的处理功能大大增强。在本节介绍如何在.NET 中显示图像、如何对图像进行旋转和剪切等物理变换，并且我们将实现一个简易的图像处理程序。

注：本系列节选于MSDN Webcasts上邵志东老师的《 .NET Windows编程系列课程》，在我的Blog中整理发表，方便大家一同参考！此系列相关资料请到这里下载。

本节主要内容：

GDI+中的图像处理

图像物理变换

坐标系和变换

一、GDI+中的图像处理

GDI+中对图像处理提供了以下支持：

支持 BMP、GIF、JPEG、PNG、TIFF、ICON 等等广泛格式的图像文件。

提供了用于多种光栅图像格式进行编码和解码的公共接口。

支持为图像格式添加动态格式。

支持对图像的像素进行多种处理，包括亮度、对比度、颜色平衡、模糊、消弱等。

支持对图像进行旋转、剪切等操作。

主要通过 Image(抽象类)实现。

Bitmap 类

从 Image 派生，可以处理 BMP、Jpeg、GIF、PNG 等格式

1．构建

Bitmap bt1 = new Bitmap("c:\\1.bmp");

Bitmap bt2 = new Bitmap(bt1,200,300);

Bitmap bt3; bt3.FromFile("文件名称");

2．Bitmap 类常用属性

名称	描述
Height	图像的高度
HorizontalResolution	图像水平方向上的分辨率
Palette	图像所使用的调色板
PhysicalDimension	图像的物理维度
PixelFormat	图像像素的格式
RawFormat	图像的存储格式
Size	图像的尺寸
VerticalResolution	图像垂直方向上的分辨率
Width	图像的宽度

3．Bitmap 类常用方法

名称	描述
Dispose	释放图像资源
GetBounds	获得图像对象的矩形边界
GetFrameCount	得到指定维度上的边框数目
GetPropertyItem	得到图像的属性信息
GetThumbnailImage	得到缩略图
RemovePropertyItem	删除图像对象的属性信息
RotateFlip	对图像进行旋转或翻转
Save	按照指定格式保存图像
SaveAdd	添加编码信息后保存图像
SelectActiveFrame	选择图像的活动边界

ImageFormat

Bmp：获取位图图像格式(BMP)。

Emf：获取增强型 Windows 图元文件图像格式(EMF)。

Exif：获取可交换图像文件(Exif) 格式。

Gif：获取图形交换格式(GIF) 图像格式。

Guid：获取表示此 ImageFormat 对象的 Guid 结构。

Icon：获取 Windows 图标图像格式。

Jpeg：获取联合图像专家组(JPEG) 图像格式。

MemoryBmp：获取内存位图图像格式。

Png：获取 W3C 可移植网络图形(PNG) 图像格式。

Tiff：获取标签图像文件格式(TIFF) 图像格式。

Wmf：获取 Windows 图元文件(WMF) 图像格式。

DEMO1：图像的转换

二、图像物理变换

Image.RotateFlip 方法

Rotate180FlipNone：指定不进行翻转的 180 度旋转。

Rotate180FlipX：指定后接水平翻转的 180 度旋转。

Rotate180FlipXY：指定后接水平翻转和垂直翻转的 180 度旋转。

Rotate180FlipY：指定后接垂直翻转的 180 度旋转。

Rotate270FlipNone：指定不进行翻转的 270 度旋转。

Rotate270FlipX：指定后接水平翻转的 270 度旋转。

Rotate270FlipXY：指定后接水平翻转和垂直翻转的 270 度旋转。

Rotate270FlipY：指定后接垂直翻转的 270 度旋转。

Rotate90FlipNone：指定不进行翻转的 90 度旋转。

Rotate90FlipX：指定后接水平翻转的 90 度旋转。

Rotate90FlipXY：指定后接水平翻转和垂直翻转的 90 度旋转。

Rotate90FlipY：指定后接垂直翻转的 90 度旋转。

RotateNoneFlipNone：指定不进行旋转和翻转。

RotateNoneFlipX：指定没有后跟水平翻转的旋转。

RotateNoneFlipXY：指定没有后跟水平和垂直翻转的旋转。

RotateNoneFlipY：指定没有后跟垂直翻转的旋转。

DEMO2：图像的变换

三、坐标系和变换

坐标系

GDI+使用三个坐标空间：世界、页面和设备。

世界坐标是用于建立特殊图形世界模型的坐标系，也是在.NET Framework 中传递给方法的坐标系。

页面坐标系是指绘图图面（如窗体或控件）使用的坐标系。

设备坐标系是在其上进行绘制的物理设备（如屏幕或纸张）所使用的坐标系。

坐标系和变换

myGraphics.TranslateTransform(100, 50); myGraphics.DrawLine(myPen, 0, 0, 160, 80);

Graphics 变换相关方法

ResetTransform

TranslateTransform：平移

RotateTransform：旋转

ScaleTransform：缩放

使用 **Matrix** 类进行变换

Matrix.Rotate 方法：顺时针按照指定角度旋转

Matrix.Scale 方法：缩放

Matrix.Translate 方法：平移

DEMO3：坐标系变换示例

DEMO4：图像变换

.NET Windows编程系列(11)：.NET图像处理(下)

摘要：

在本节中，我们将继续学习有关图像处理的一些知识：包括颜色基本知识、图像的剪切和和粘贴、坐标系变换、各种图像特效的实现等。

注：本系列节选自MSDN Webcasts上邵志东老师的《 .NET Windows编程系列课程》PPT内容，在我的Blog中整理发表，方便大家一同参考！此系列相关资料请到这里下载。

本节主要内容：

从颜色谈起

图像的复制和粘贴

像素处理

动画

综合实例

一、从颜色谈起

颜色表示

RGB：即用红、绿、蓝和透明度的组合来表示计算机中所有颜色。

HSB：即用色调、饱和度、亮度的组合方式来表示颜色。

CMYK：代表印刷上用的四种颜色，C 代表青色，M 代表洋红色，Y 代表黄色，K 代表黑色。

1．RGB 模式

RGB 是色光的色彩模式。三种色彩叠加形成了其它的色彩。因为三种颜色都有 256 个亮度水平级，所以三种色彩叠加就形成 1670 万种颜色了。也就是真彩色，通过它们足以在现绚丽的世界。

在 RGB 模式中，由红、绿、蓝相叠加可以产生其它颜色，因此该模式也叫加色模式。所有显示器、投影设备以及电视机等等许多设备都依赖于这种加色模式来实现的。 就编辑图像而言，RGB 色彩模式也是最佳的色彩模式，因为它可以提供全屏幕的 24bit 的色彩范围，即真彩色显示。

但是，如果将 RGB 模式用于打印就不是最佳的了，因为 RGB 模式所提供的有些色彩已经超出了打印的范围之外，因此在打印一幅真彩色的图像时，就必然会损失一部分亮度，并且比较鲜艳的色彩肯定会失真的。

2．HSB 模式

在 HSB 模式中，H 表示色相，S 表示饱和度，B 表示亮度。

色相：是纯色，即组成可见光谱的单色。红色在 0 度，绿色在 120 度，蓝色在 240 度。它基本上是 RGB 模式全色度的饼状图。

饱和度：表示色彩的纯度，为 0 时为会色。白、黑和其他灰色色彩都没有饱和度的 在最大饱和度时，每一色相具有最纯的色光。

亮度：是色彩的明亮度。为 0 时即为黑色，最大亮度是色彩最鲜明的状态。

3．CMYK 模式

当阳光照射到个物体上时，这个物体将吸收一部分光线，并将剩下的光线进行反射，反射的光线就是我们所看见的物体颜色。这是一种减色色彩模式，同时也是与 RGB 模式的根本不同之处。不但我们看物体的颜色时用到了这种减色模式，而且在纸上印刷时应用的也是这种减色模式。按照这种减色模式，就衍变出了适合印刷的 CMYK 色彩模式。

因为在实际引用中，青色、洋红色和黄色很难叠加形成真正的黑色，最多不过是褐色而已。因此才引入了 k——黑色。黑色的作用是强化暗调，加深暗部色彩。

CMYK 模式是最佳的打印模式

DEMO1：颜色示例

二、图像的复制和粘贴

Clipboard 类

Clipboard 类提供了可以用来与 Windows 操作系统剪贴板功能交互的方法。许多应用程序都将剪贴板用作一个临时数据储存库。剪贴板还可用于从一个应用程序向另一个应用程序传输数据。

调用 SetDataObject，将数据置于剪贴板中。调用 GetDataObject，从剪贴板中检索数据。

DEMO2：图像复制和粘贴

三、像素处理

改变图像的分辨率

彩色图像变换（逆反处理、平滑处理、霓虹处理、浮雕处理、镶嵌处理、灰度处理）

DEMO3：更改图像分辨率

图像特效

单色处理：用原来颜色的一个分量代替其他分量。

反色处理：颜色所有分量，用 255 减去该分量代替。

平滑处理：像素点颜色用周围像素的平均值代替。

霓虹处理：首先计算像素 f(i,j)的红、绿、蓝分量与同行 f(i+1,j)以及同列 f(i,j+1)相邻像素的梯度，即差的平方之和的平方根，然后将梯度值作为处理后该像素的颜色值。

锐化处理：由 f(i,j)与 f(i-1,j-1)像素值之差的绝对值的百分比之和，作为新值。

浮雕处理：算法为：g(i,j) = f(i,j)-f(i-1,j)+常数（一般为 128）。

马赛克处理：处理后图像每一小矩阵内的所有像素都取此矩阵内源图像各像素值之和的平均值。本节示例为 5*5。

灰度处理：int gray=r*0.3+g*0.59+b*0.11;然后令 r=g=b=gray 即可。

透明度变换：修改像素透明度。

DEMO4：图像特效

四、动画

动画和图像切换

1．动画有三种：

位置不动，形态动；位置动，形态不动；位置和形态都动

2．图像切换技术

图像排列方法； DrawImage 方法

DEMO5：图像切换

五、综合实例

DEMO6：图像处理综合实例

C# GDI+ 简单绘图 （三）

感谢大家的支持,这几天从早忙到晚,一个字累呀!!!现在挺困的,但是又不习惯这么早睡觉,哎~~还是利用这个时间继续来写第三篇吧.

前两篇已经基本向大家介绍了绘图的基本知识.那么,我就用我们上两篇所学的,做几个例子.

我们先来做一个简单的----仿 QQ 截图,关于这个的例子其实网上已经有这方面的资料了,但是为了文章的完整性,还是觉得有必要讲解.

我们先来看一下效果：

（图 1）

(图 2)

接下来看看这是如何做到的.

思路:将整个屏幕画在一个新的全屏窗体上.然后在这个新窗体上画矩形,最后保存矩形中的内容.

步骤:

A.新建一个窗体. 命名为 Catch.然后设置这个窗体的 FormBorderStyle 为 None,WindowState 为 Maximized.

B.我们对代码进行编辑:

```
using System;

using System.Collections.Generic;

using System.ComponentModel;
using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;



namespace Client

    {

  public partial class Catch : Form

      {

    public Catch()

          {

      InitializeComponent();

    }


         用户变量


    //窗体初始化操作

    private void Catch_Load(object sender, EventArgs e)

          {

            this.SetStyle(ControlStyles.OptimizedDoubleBuffer | ControlStyles.AllPaintingInWmPai
nt | ControlStyles.UserPaint, true);

      this.UpdateStyles();
```

```
          //以上两句是为了设置控件样式为双缓冲，这可以有效减少图片闪烁的问题，关于这个大家可以自己去搜索下

          originBmp = new Bitmap(this.BackgroundImage);//BackgroundImage 为全屏图片，我们另用变量来保存全屏图片

        }



        //鼠标左键点击结束截图

        private void Catch_MouseClick(object sender, MouseEventArgs e)

             {

          if (e.Button == MouseButtons.Right)

               {

            this.DialogResult = DialogResult.OK;

            this.Close();

          }

        }



        //鼠标左键按下时动作

        private void Catch_MouseDown(object sender, MouseEventArgs e)

             {

          if (e.Button == MouseButtons.Left)

               {

            if (!CatchStart)

                 {//如果捕捉没有开始
```

```
              CatchStart = true;

              DownPoint = new Point(e.X, e.Y);//保存鼠标按下坐标

            }

          }

        }



        private void Catch_MouseMove(object sender, MouseEventArgs e)

             {

          if (CatchStart)

               {//如果捕捉开始

            Bitmap destBmp = (Bitmap)originBmp.Clone();//新建一个图片对象，并让它与原始图片相同

            Point newPoint = new Point(DownPoint.X, DownPoint.Y);//获取鼠标的坐标

            Graphics g = Graphics.FromImage(destBmp);//在刚才新建的图片上新建一个画板

            Pen p = new Pen(Color.Blue,1);

             int width = Math.Abs(e.X - DownPoint.X), height = Math.Abs(e.Y - DownPoint.Y);//获取矩形的长和宽

            if (e.X < DownPoint.X)

                 {

              newPoint.X = e.X;

            }

            if (e.Y < DownPoint.Y)

                 {
```

```
              newPoint.Y = e.Y;

            }

            CatchRect = new Rectangle(newPoint,new Size(width,height));//保存矩形

            g.DrawRectangle(p,CatchRect);//将矩形画在这个画板上

            g.Dispose();//释放目前的这个画板

            p.Dispose();

            Graphics g1 = this.CreateGraphics();//重新新建一个 Graphics 类

            //如果之前那个画板不释放，而直接 g=this.CreateGraphics()这样的话无法释放掉第一次创建的 g,因为只是把地址转到新的 g 了．如同 string 一样

            g1 = this.CreateGraphics();//在整个全屏窗体上新建画板

            g1.DrawImage(destBmp,new Point(0,0));//将刚才所画的图片画到这个窗体上

            //这个也可以属于二次缓冲技术，如果直接将矩形画在窗体上，会造成图片抖动并且会有无数个矩形．

            g1.Dispose();

            destBmp.Dispose();//要及时释放，不然内存将会被大量消耗



          }

        }



        private void Catch_MouseUp(object sender, MouseEventArgs e)

             {

          if (e.Button == MouseButtons.Left)

               {
```

```
        if (CatchStart)
                  {
            CatchStart = false;
            CatchFinished = true;


        }
    }
}


//鼠标双击事件，如果鼠标位于矩形内，则将矩形内的图片保存到剪贴板中
private void Catch_MouseDoubleClick(object sender, MouseEventArgs e)
          {
    if (e.Button == MouseButtons.Left&&CatchFinished)
              {
        if (CatchRect.Contains(new Point(e.X, e.Y)))
                  {
            Bitmap CatchedBmp = new Bitmap(CatchRect.Width, CatchRect.Height);//新建一个于矩形等大的空白图片
            Graphics g = Graphics.FromImage(CatchedBmp);
            g.DrawImage(originBmp, new Rectangle(0, 0, CatchRect.Width, CatchRect.Height), CatchRect, GraphicsUnit.Pixel);
            //把 orginBmp 中的指定部分按照指定大小画在画板上
            Clipboard.SetImage(CatchedBmp);//将图片保存到剪贴板
```

```
                    g.Dispose();

                    CatchFinished = false;

                    this.BackgroundImage = originBmp;

                    CatchedBmp.Dispose();

                    this.DialogResult = DialogResult.OK;

                    this.Close();

                }

            }

        }

    }

}
```

C.创建了 Catch 窗体后，我们在截图按钮上加入以下事件：

```
        private void bCatch_Click(object sender, EventArgs e)

             {



          if (bCatch_HideCurrent.Checked)

               {

            this.Hide();//隐藏当前窗体

            Thread.Sleep(50);//让线程睡眠一段时间，窗体消失需要一点时间

            Catch CatchForm = new Catch();

            Bitmap CatchBmp = new Bitmap(Screen.AllScreens[0].Bounds.Width, Screen.AllScreens[0].Bounds.Height);//新建一个和屏幕大小相同的图片
            Graphics g = Graphics.FromImage(CatchBmp);

            g.CopyFromScreen(new Point(0, 0), new Point(0, 0), new Size(Screen.AllScreens[0].Bounds.Width, Screen.AllScreens[0].Bounds.Height));//保存全屏图片

            CatchForm.BackgroundImage = CatchBmp;//将 Catch 窗体的背景设为全屏时的图片

            if (CatchForm.ShowDialog() == DialogResult.OK)

                {//如果 Catch 窗体结束,就将剪贴板中的图片放到信息发送框中

              IDataObject iData = Clipboard.GetDataObject();

              DataFormats.Format myFormat = DataFormats.GetFormat(DataFormats.Bitmap);

              if (iData.GetDataPresent(DataFormats.Bitmap))

                   {

                richtextbox1.Paste(myFormat);

                Clipboard.Clear();//清除剪贴板中的对象

              }

              this.Show();//重新显示窗体

            }

          }



        }
```

这样我们的截图功能便完成了.

我想对于初学者来说如何消去第一次绘制的图片是个比较困难的问题.如果没有采取措施，你会发现只要你鼠标移动，就会画一个矩形，这样便会出现 N 多的矩形，而我们只是要最后的那一个.

一般解决这种问题的方法有两种：

1.就是在绘制第二个图形时,我们先用与底色相同的颜色将上次绘制的图形重新绘制一下. 但这往往需要底色为纯色时使用.

2.我们并不直接将图形画在画板上,我们用一个图片 A 来保存原画板上的图

片．然后再新建一个与图片 A 相同的图片 B，将我们要绘制的图形画在该图片 B 上，然后再将该图片 B 画在画板上．这样图片 A 并没有被改变．于是第二次画的时候我们还是同样新建一个与图片 A 相同的图片进行绘制．那么上一次的图形就不会被保留下来．问题也就解决了．

下一次，向大家介绍如何做一个仿 windows 画板的程序．

Tag标签: 截图,.NET,C#,绘图

posted on 2008-03-19 19:42 stg609 阅读(1098) 评论(6) 编辑 收藏 所属分类: .Net 技术

Minesweeper: GDI+ 综述

Posted on 2008-03-02 21:37 xingd 阅读(387) 评论(5) 编辑 收藏 所属分类: GDI+ .

系列前面的两篇文章写的内容太简单了，本文对我理解的 GDI+做一个综述，不再涉及代码细节。

GDI+中共有三种坐标，全局坐标、页面坐标和设备坐标。在 GDI+的绘图调用中，传入的坐标位于全局坐标内，全局坐标经由全局变换转换到页面坐标，页面坐标再通过页面变换计算出设备坐标。

全局变换通过 Graphics.Transform 指定，其类型为 Matrix。GDI+中的矩阵为 3x3 浮点矩阵，可以通过 Matrix 类的方法和属性来修改全局变换，也可以通过 Graphics 类上的 TranslateTransform 等方法来设定。页面变换通过 Graphics 类的 PageUnit 和 PageScale 来设定坐标单位和缩放倍数。

Point, Size, Rectangle 是 GDI+中常用的度量类型，并且都具有对应的 float 类型。Color 则代表了 32 位 A8R8G8B8 的颜色。这一些都是基本的值类型，在实际使用的时候，要牢记其值类型的特征，类似 o.Size.Width = 100 的代码是没有作用的，因为.Width = 100 是作用在了 o.Size 返回的临时变量上了，对于 o 的状态没有任何影响。

GraphicsPath，Region, Image 则是 GDI+中的一些资源性的类型，在使用完成后要尽快 Dispose。GraphicsPath 是一系列连续的线，包含直线和曲线。Region 则表示封闭的一个区域，这个区域的边界可以由 GraphicsPath 来描述。Image 表示一个图形，其中表现像素组成的位图的派生类为 Bitmap，表现失量绘图指令组成的图形的派生类为 Metafile。计算机屏幕最擅长展现两维的数据，因此 Rectangle 视为最简单的一种 Region，并且应用面也非常广泛，计算包含整个 Region 的 Rectangle 也是非常常见的一种操作。

Brush 用来填充一个 Region，填充时可以使用单色填充，可以使用纹理（图片）填充，也可以使用线型填充和渐变填充，.NET 中封充的 GDI+提供了 SolidBrush，TextureBrush，HatchBrush，LinearGradientBrush 和 PathGradientBrush 类。Brush 也是需要及时 Dispose 的，对于 SolidBrush，可使用 SystemBrushes 和 Brushes 中的静态属性，获取预定义的 Brush 对表，免去 Dispose 的麻烦。

Pen 是用来画线的，GDI+的线是有宽度的，也就有其内部区域，因此 GDI+中的 Pen 需要一个 Brush 实例来构造。同样 SystemPens 和 Pens 中提供了预定义的单色 Pen 实例。

此外 Font 对象用来实现 GDI+中不同字体的输出，Graphics 类提供了一个 MeasureString 方法计算一段字符串绘制出来时占据的区域大小。

Graphics 类提供了一系列 Draw...方法，使用特定的 Pen 来绘制一定的形状，Fill...系列则使用特定的 Brush 来填充指定区域。

Graphics 的 Clip 属性通过一个 Region 类的实例指定 GDI+有效绘制区域，这是一个基础信息，Graphics 的属性 ClipBounds，IsClipEmpty，IsVisibleClipEmpty，VisibleClipBounds 均基于 Clip 属性，并且为只读的。

CompositingQuality，InterpolationMode，PixelOffsetMode，SmoothingMode，TextRenderingHint 用来控制绘制输出质量，质量越高，速度越慢。CompositingMode 用来启用 Alpha Blend，TextContrast 控制文本输出时的 Gamma 值，RenderingOrigin 用来控制 8bit/16bit 色深时的色彩拌动和 Hatch Brush 的起始点。

这些内容对之后的 Minesweeper 内容足够了，如果有问题，欢迎在评论中提出。

如何获取GRAPHICS对象

在.NET 中，可以通过以下方法获取 Graphics 对象。

1．从 Paint 事件的参数中获取。

窗体和许多控件都有一个 Paint 事件，有一个 PaintEventArgs 类型的参数 e。

```
 private void Form1_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
        {
          //获取 Graphic 对象
          Graphics g = e.Graphics;
          //书写绘图代码
          g.DrawRectangle(Pens.Red,10,10,100,50);
          //释放 Graphic 对象占用的资源
          g.Dispose();

      }
```

窗体的 Paint 事件是最常用于放置绘图代码的地方，每当窗体被其他窗体挡住，再次显示的时候，窗体的所有内容必须被重绘，否则会得到一个空白的窗体。

2．用 CreateGraphics 方法创建。

如果需要在 Paint 方法以外绘图，可以通过控件或窗体的 CreateGraphics 方法来获取 Graphics 对象。

```
 private void button1_Click(object sender, System.EventArgs e)
        {
          Graphics g = button1.CreateGraphics();
          //画一个椭圆
          g.DrawEllipse(Pens.Red,5,5,button1.Width-10,button1.Height-10);
          g.Dispose();
      }
```

3．对 Image 对象调用 Graphics.FromImage 获取。

```
 private void Form1_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
        {
          //创建 Image 对象
          Bitmap image1 = new Bitmap("football.jpg");
          //窗体的绘图对象
          Graphics formE = e.Graphics;
          //图像的绘图对象
          Graphics imageE = Graphics.FromImage(image1);
          //将第二幅图覆盖到第一幅的左上角
          imageE.DrawImage(new Bitmap("asdf.JPG"),new Rectangle(0,0,this.ClientSize.Width/2,this.ClientSize.Height/2));
```

```
│           //将合成好的图像绘制在窗体上
│           formE.DrawImage(image1,new Rectangle(0,0,this.ClientSize.Width,this.ClientSize.Height));
│
│           //释放所有资源
│           formE.Dispose();
│           imageE.Dispose();
│           image1.Dispose();
└       }
```

关于提高缩略图品质

所略图品质问题一直困扰着我,一直以为只要设置了 Graphics 对象的 SmoothingMode 属性就可以了

(Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality //设置高质量,低速度呈现平滑程度) 然而在使用的过程中总是觉得生成的图片品质总是与原图有一断肉眼就能分辨的出的差异,甚不合意...

于是翻些资料发现 Graphics 还有个属性 InterpolationMode(Graphics 对象的插值模式影响 GDI+ 缩放（拉伸和收缩）图像的方式)

将此属性设置为指定高质量双三次插值法,如下

Graphics.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;(HighQualityBicubic 是质量最好的模式)

还有如下几个属性设置可以提高图像合成及加水印时的品质

Graphics.TextRenderingHint = TextRenderingHint.AntiAlias;

Graphics.CompositingQuality = CompositingQuality.HighQuality;

详细资料参考 msdn

呵呵,现在生成的所略图品质有所提高,不过与某网站的所略图还是有所差距.....

不知道还有没有别的方法来生成高品质的图片..searching...

缩略图品质

测试了下我所知道的所有方法,发现关键的地方还是

```
ImageCodecInfo myImageCodecInfo;
Encoder myEncoder;
EncoderParameter myEncoderParameter;
EncoderParameters myEncoderParameters;
myImageCodecInfo = GetEncoderInfo("image/jpeg");
myEncoder = Encoder.Quality;
myEncoderParameters = new EncoderParameters(1);
myEncoderParameter = new EncoderParameter(myEncoder, 95L);
myEncoderParameters.Param[0] = myEncoderParameter;
```

同时比较下不同设置的图片品质

1.只设置了 graphics.SmoothingMode = SmoothingMode.HighQuality

2.只设置了 graphics.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic

3.同时设置了 1,2 两属性

4.只设置 ImageCodecInfo myImageCodecInfo;

```
Encoder myEncoder;
EncoderParameter myEncoderParameter;
EncoderParameters myEncoderParameters;
myImageCodecInfo = GetEncoderInfo("image/jpeg");
myEncoder = Encoder.Quality;
myEncoderParameters = new EncoderParameters(1);
myEncoderParameter = new EncoderParameter(myEncoder, 95L);
myEncoderParameters.Param[0] = myEncoderParameter;
```

5.同时设置了 1,2,4 步

综上图示,图像的品质关键在于设置第四步...
另外对于 myImageCodecInfo = GetEncoderInfo("image/jpeg");
我发现如果按原先的.gif 图片来设置 GetEncoderInfo("image/gif")的话,图片的象素将会丢失很严重,而设置为 jpeg 却很好,而关于象素的丢失如果原先的图片格式就为 gif(256 色)的话应该不会发生这么多象素丢失,如果原先图片是 jpg(65536 色)的话到是可能会有这么严重的丢失现象....怎么会这样呢..郁闷下....

GDI+中RenderingOrigin对TextureBrush的影响

Posted on 2005-02-02 15:40 xingd 阅读(975) 评论(4) 编辑 收藏 所属分类: GDI+ .

原文见http://www.cnblogs.com/mikespook/archive/2005/01/31/99912.html，作者将图像对齐的问题归到对FillRectangle的理解上，这是不准确的。

根本的原因在于TextureBrush的使用方式，GDI+中使用Brush时，会根据Graphics.RenderingOrigin确定Brush的原点，然后在FillRectangle，DrawString等使用Brush的方法中，将Brush的可见部分绘制出来。除了TextureBrush外，HatchBrush也会受RenderOrigin的影响。

要实现原文中需要的功能，应该采用如下的代码：

```
private void DrawTitle_Right(Graphics g)
{
    using(Image img = Files.GetImage("/Images/Top_Right.gif"))
    {
        using (Brush brush = new TextureBrush(img))
        {
            Point pt = g.RenderingOrigin;
            g.RenderingOrigin = new Point(this.ClientSize.Width-img.Width, 0);
            g.FillRectangle(brush, this.ClientSize.Width-img.Width, 0, img.Width, img.Height);
            g.RenderingOrigin = pt;
        }
    }
}
```

或者是：

```
private void DrawTitle_Right(Graphics g)
{
    using(Image img = Files.GetImage("/Images/Top_Right.gif"))
    {
        using (Brush brush = new TextureBrush(img))
        {
```

```
│           GraphicsState state = g.Save();
│           g.RenderingOrigin = new Point(this.ClientSize.Width-img.Width, 0);
│           g.FillRectangle(brush, this.ClientSize.Width-img.Width, 0, img.Width, img.Height);
│           g.Restore(state);
├       }
├   }
└}
```

我也试过了，可能我对 RenderingOrigin 理解的也不够，但 HatchBrush 是肯定受它的影响的，TextureBrush 的 Origin，我会再查一些资料，多试一下。mikespook 如果找到答案了，也请告诉我一下。

以下代码是可以解决问题的：

```
private void DrawTitle_Right(Graphics g)
{
  using(Image img = Files.GetImage("/Images/Top_Right.gif"))
  {
    using (TextureBrush brush = new TextureBrush(img))
    {
      brush.TranslateTransform(this.ClientSize.Width-image.Width, 0);
      g.FillRectangle(brush, this.ClientSize.Width-img.Width, 0, img.Width,
img.Height);
    }
  }
}
```

之前的 **gdi+**绘图,整理了一下,清晰了很多 - 南方小鬼的博客 - 博客园

柱状图

```
using System;
using System.IO;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace WanFangData.Chart
{
    public class Bar
    {
        public void Render(string title, string subTitle, int width, int height, DataSet chartData, Stream target)
        {
            const int SIDE_LENGTH = 400;
            const int CHART_TOP = 75;
            const int CHART_HEIGHT = 200;
            const int CHART_LEFT = 50;
            const int CHART_WIDTH = 300;
            DataTable dt = chartData.Tables[0];

            //计算最高的点
            float highPoint = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (highPoint < Convert.ToSingle(dr[1]))
                {
                    highPoint = Convert.ToSingle(dr[1]);
                }
            }
            //建立一个 Graphics 对象实例
            Bitmap bm = new Bitmap(width, height);
            Graphics g = Graphics.FromImage(bm);
            //设置条图图形和文字属性
            g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height)) / SIDE_LENGTH);
            g.SmoothingMode = SmoothingMode.Default;
            g.TextRenderingHint = TextRenderingHint.AntiAlias;

            //设定画布和边
            g.Clear(Color.White);
            g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
            //画大标题
            g.DrawString(title, new Font("Tahoma", 24), Brushes.Black, new PointF(5, 5));
            //画小标题
            g.DrawString(subTitle, new Font("Tahoma", 14), Brushes.Black, new PointF(7, 35));
            //画条形图
            float barWidth = CHART_WIDTH / (dt.Rows.Count * 2);
            PointF barOrigin = new PointF(CHART_LEFT + (barWidth / 2), 0);
            float barHeight = dt.Rows.Count;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
```

```
                barHeight = Convert.ToSingle(dt.Rows[i][1]) * 200 / highPoint;
                barOrigin.Y = CHART_TOP + CHART_HEIGHT - barHeight;
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), barOrigin.X, barOrigin.Y, barWidth, barHeight);
                barOrigin.X = barOrigin.X + (barWidth * 2);
            }
            //设置边
            g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT));
            g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT), new Point(CHART_LEFT + CHART_WIDTH, CHART_TOP + CHART_HEIGHT));
            //画图例框和文字
            g.DrawRectangle(new Pen(Color.Black, 1), 200, 300, 199, 99);
            g.DrawString("Legend", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new PointF(200, 300));

            //画图例
            PointF boxOrigin = new PointF(210, 330);
            PointF textOrigin = new PointF(235, 326);
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawString(dt.Rows[i][0].ToString() + " - " + dt.Rows[i][1].ToString(), new Font("Tahoma", 10), Brushes.Black, textOrigin);
                boxOrigin.Y += 15;
                textOrigin.Y += 15;
            }
            //输出图形
            bm.Save(target, ImageFormat.Gif);

            //资源回收
            bm.Dispose();
            g.Dispose();
        }
    }
}

```

饼状图

```
using System;
using System.IO;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace WanFangData.Chart
{
    public class Pie
    {
        public void Render(string title, string subTitle, int width, int height, DataSet chartData, Stream target)
        {
            const int SIDE_LENGTH = 400;
            const int PIE_DIAMETER = 200;
            DataTable dt = chartData.Tables[0];

            //通过输入参数，取得饼图中的总基数
            float sumData = 0;
            foreach (DataRow dr in dt.Rows)
            {
                sumData += Convert.ToSingle(dr[1]);
            }
            //产生一个 image 对象，并由此产生一个 Graphics 对象
            Bitmap bm = new Bitmap(width, height);
            Graphics g = Graphics.FromImage(bm);
            //设置对象 g 的属性
            g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height)) / SIDE_LENGTH);
            g.SmoothingMode = SmoothingMode.Default;
            g.TextRenderingHint = TextRenderingHint.AntiAlias;

            //画布和边的设定
            g.Clear(Color.White);
            g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
            //画饼图标题
            g.DrawString(title, new Font("Tahoma", 24), Brushes.Black, new PointF(5, 5));
            //画饼图的图例
            g.DrawString(subTitle, new Font("Tahoma", 14), Brushes.Black, new PointF(7, 35));
            //画饼图
            float curAngle = 0;
            float totalAngle = 0;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                curAngle = Convert.ToSingle(dt.Rows[i][1]) / sumData * 360;

                g.FillPie(new SolidBrush(Utils.GetChartItemColor(i)), 100, 65, PIE_DIAMETER, PIE_DIAMETER, totalAngle, curAngle);
                g.DrawPie(Pens.Black, 100, 65, PIE_DIAMETER, PIE_DIAMETER, totalAngle, curAngle);
                totalAngle += curAngle;
            }
            //画图例框及其文字
            g.DrawRectangle(Pens.Black, 200, 300, 199, 99);
            g.DrawString("Legend", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new PointF(200, 300));

            //画图例各项
            PointF boxOrigin = new PointF(210, 330);
            PointF textOrigin = new PointF(235, 326);
            float percent = 0;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 20, 10);
                percent = Convert.ToSingle(dt.Rows[i][1]) / sumData * 100;
                g.DrawString(dt.Rows[i][0].ToString() + " - " + dt.Rows[i][1].ToString() + " (" + percent.ToString("0") + "%)", new Font("Tahoma", 10), Brushes.Black, textOrigin);
                boxOrigin.Y += 15;
                textOrigin.Y += 15;
            }
            //通过 Response.OutputStream，将图形的内容发送到浏览器
            bm.Save(target, ImageFormat.Gif);
            //回收资源
            bm.Dispose();
            g.Dispose();
        }
    }
}
```

工具

```
using System;
using System.Drawing;

namespace WanFangData.Chart
{
    class Utils
    {
        public static Color GetChartItemColor(int itemIndex)
        {
            Color selectedColor;
            switch (itemIndex)
            {
                case 0:
                    selectedColor = Color.Blue;
                    break;
                case 1:
                    selectedColor = Color.Red;
                    break;
                case 2:
                    selectedColor = Color.Yellow;
                    break;
                case 3:
                    selectedColor = Color.Purple;
                    break;
                default:
                    selectedColor = Color.Green;
                    break;
            }
            return selectedColor;
        }
    }
}
```

调用柱状图

```
using System;
using System.Data;
using WanFangData.Chart;
using WanFangData.Common;

namespace WanFangData.Web
{
    public partial class CBar : WanFangData.Page.BasePage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Bar p = new Bar();
            p.Render("大标题", "小标题", 300,
300,WanFangData.Common.Database.ExecuteDataset(CommandType.Text,
"select * from
a"), Response.OutputStream);
        }
    }
}

```

调用饼状图

```
using System;
using System.Data;
using WanFangData.Chart;
using WanFangData.Common;

namespace WanFangData.Web
{
    public partial class CPie : WanFangData.Page.BasePage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Pie p = new Pie();
            p.Render("大标题", "小标题", 400, 400, Database.ExecuteDataset(CommandType.Text, "select * from a"), Response.OutputStream);
        }
    }
}

```

用 GDI+怎么实现绘制倾斜文字

Q:用 GDI+怎么实现绘制倾斜文字

A:Graphics g = this.CreateGraphics();

g.RotateTransform(30f);

g.DrawString("倾斜 ABCabc", this.Font, SystemBrushes.WindowText, 10f, 10f);

```
g.ResetTransform();

g.Dispose();

g = null;
```

简单的 GDI+处理图片大小(C#代码)

```
        /// <summary>
        /// 缩放图片
        /// </summary>
        /// <param name="img">原图片</param>
        /// <param name="xWith">缩放宽比例,如果想缩小图片,小于 100</param>
        /// <param name="yHeight">缩放高比例</param>
        /// <returns>返回处理后图片</returns>
        public Image scaleImg(System.Drawing.Image img, int xWith, int yHeight)
        {
            //计算处理后图片宽
            int i = Convert.ToInt32(img.Width * xWith / 100);
            //计算处理后图片高
            int j = Convert.ToInt32(img.Height * yHeight / 100);
            //格式化图片
            System.Drawing.Image imgScale = new System.Drawing.Bitmap(i, j,
System.Drawing.Imaging.PixelFormat.Format24bppRgb);
            System.Drawing.Graphics g =
System.Drawing.Graphics.FromImage(imgScale);
            System.Drawing.Rectangle srcRect = new
System.Drawing.Rectangle(0, 0, img.Width, img.Height);
            System.Drawing.Rectangle desRect = new
System.Drawing.Rectangle(0, 0, imgScale.Width, imgScale.Height);
            g.Clear(System.Drawing.Color.White);
            g.DrawImage(img, desRect, srcRect, 
System.Drawing.GraphicsUnit.Pixel);
            //处理后的图片另存
            imgScale.Save("E:\\1111.jpg", 
System.Drawing.Imaging.ImageFormat.Gif);
            g.Dispose();
            return imgScale;
        }
```

GDI 特殊效果处理与应用

GDI+简单使用例子

目录

1 介绍

本文档将以一个具体的例子，讲解 GDI+简单的使用例子。

2 约定

斜体字是真正的程序代码。

3 示例

GDI+主要使用的是 System.Drawing; System.Drawing.Drawing2D; System.Drawing.Imageing;命名空间。

```
System.Drawing.Bitmap image = new System.Drawing.Bitmap(32, 32);
```

此句是定义一个长 32、宽 32 的画板

```
System.Drawing.Graphics graph = System.Drawing.Graphics.FromImage(image);
```

在画板上创建一个绘图的实例，之后使用 graph，就可以在 image 上画图形了。

网站验证码源码(学习)

本代码直接复制进去就可以用了，刷新就可以看到验证码的变动。
这个验证码已经在 DotNet迅网 (http://www.16sw.com) 使用。
Default.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
//myself
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;

public partial class _Default : System.Web.UI.Page
{
    public System.Web.UI.WebControls.Image verifyNumber = new System.Web.UI.WebControls.Image();

    protected void Page_Load(object sender, EventArgs e)
    {
        string tmp = RndNum(Convert.ToInt16(6));
        Session["verify"] = tmp;
        ValidateCode(tmp);
    }

    private void ValidateCode(string VNum)
    {
        Bitmap Img = null;
        Graphics g = null;
        MemoryStream ms = null;
        int gheight = VNum.Length * 9;
        Img = new Bitmap(gheight, 18);
        g = Graphics.FromImage(Img);
        //背景颜色
        g.Clear(Color.WhiteSmoke);
        //文字字体
        Font f = new Font("Tahoma", 9);
        //文字颜色
        SolidBrush s = new SolidBrush(Color.Red);
        g.DrawString(VNum, f, s, 3, 3);
        ms = new MemoryStream();
        Img.Save(ms, ImageFormat.Jpeg);
        Response.ClearContent();
        Response.ContentType = "image/Jpeg";
        Response.BinaryWrite(ms.ToArray());
        g.Dispose();
        Img.Dispose();
        Response.End();
    }
    private string RndNum(int VcodeNum)
   {
    string MaxNum="";
    string MinNum="";
    for (int i = 0; i < VcodeNum; i++ )
    {
        MaxNum = MaxNum + "9";
    }
    MinNum=MaxNum.Remove(0,1);
    Random rd=new Random();
    string VNum=Convert.ToString(rd.Next(Convert.ToInt32(MinNum),Convert.ToInt32(MaxNu
m)));
    return VNum;
   }
}
```

使用GDI+绘制高质量图和字体

对于 GDI+,在正常的操作,Bitmap-- Graphcis -- DrawImage 或者 DrawString ,生成图片的话,会产生很多杂点,或者是图片质量不稳定..尤其是在读取图片后,生成缩略图之后,文件会被压缩而失真..

主要原因是因为没有重新设置 Graphics 的几个属性..

1.Graphics.SmoothingMode 属性：例如 SmoothingMode.HighQuality 可以产生高质量图片,但是效率低.

2.Graphics.CompositingQuality 属性：例如:CompositingQuality.HighQuality 也是产生高质量图,效率低下.

3.Graphics.InterpolationMode 属性,例如:InterpolationMode.HighQualityBicubic 与前两个也是同样的效果.

这三个属性的值都是 enum,具体的 enum 参数可以查看 MSDN 的说明..在这里就我不赘述..如果是对图片进行放大,缩小,可以调整 Graphics.CompositingQuality 和 Graphics.InterpolationMode 两个属性..如果是图片生成,则可以调整 Graphics.SmoothingMode 属性..

另外一个问题就是关于文字生成的..按照正常的模式生成的文字,可以很明显的看到文字带有锯齿..解决的办法也是需要修改 Graphics 的一个属性: Graphics.TextRenderingHint...注意一点,修改 TextRenderingHint 的话,需要引入 System.Drawing.Text,例如:Graphics.TextRenderingHint = System.Drawing.Text.TextRenderingHint.ClearTypeGridFit;

经过对这四个属性的修改,操作大部分的图片之后,产生的结果都是比较让人满意的..

在这里提供一个简单的例子..是生成印章签名的效果..提供四种字体选择..并产生一个图片..如果你的机器中没有安装指定的几种字体,你需要把字体修改一下...直接把代码拷贝..就可以看到效果...

ASPX 页面:

```
<%@ Page language="c#" Codebehind="underWrite.aspx.cs" AutoEventWireup="false" Inherits="testItem.movie.underWrite" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
```

```
        <title>underWrite</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
            <asp:Image id="Image1" style="Z-INDEX: 101; LEFT: 208px; POSITION: absolute; TOP: 360px" runat="server"></asp:Image>
            <asp:HyperLink id="HyperLink1" style="Z-INDEX: 105; LEFT: 304px; POSITION: absolute; TOP: 312px"
                runat="server" NavigateUrl="../mv/index.html">大头帖</asp:HyperLink>
            <asp:TextBox id="name" style="Z-INDEX: 104; LEFT: 208px; POSITION: absolute; TOP: 144px" runat="server"
                MaxLength="4"></asp:TextBox>
            <asp:RadioButtonList id="fontType" style="Z-INDEX: 103; LEFT: 208px; POSITION: absolute; TOP: 184px"
                runat="server" Width="240px">
                <asp:ListItem Value="方正黄草简体" Selected="True">方正黄草简体</asp:ListItem>
                <asp:ListItem Value="汉鼎繁淡古">汉鼎繁淡古</asp:ListItem>
                <asp:ListItem Value="汉鼎繁印篆">汉鼎繁印篆</asp:ListItem>
                <asp:ListItem Value="经典繁栈亭">经典繁栈亭</asp:ListItem>
            </asp:RadioButtonList>
            <asp:Button id="bu" style="Z-INDEX: 102; LEFT: 208px; POSITION: absolute; TOP: 312px" runat="server"
```

```
                Text="刻字"></asp:Button>
          </form>
      </body>
  </HTML>

```

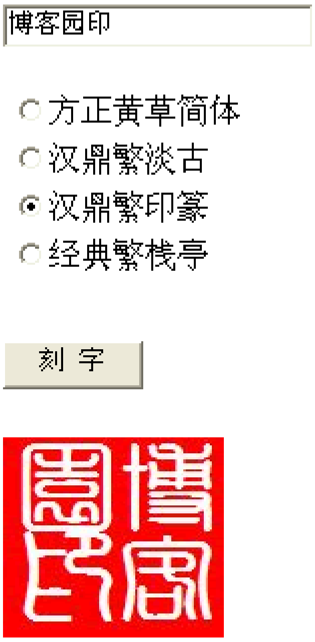
CS 文件:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace testItem.movie
{
    /// <summary>
    ///        生成印章签名
    /// </summary>
    public class underWrite : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Image Image1;
        protected System.Web.UI.WebControls.RadioButtonList fontType;
        protected System.Web.UI.WebControls.HyperLink HyperLink1;
        protected System.Web.UI.WebControls.Button bu;
        protected System.Web.UI.WebControls.TextBox name;

        private void Page_Load(object sender, System.EventArgs e)
         {
          Image1.Visible = false;
        }

         Web 窗体设计器生成的代码

        private void Button1_Click(object sender, System.EventArgs e)
         {
          string Name = name.Text;
           if ( Name.Length < 4 ) {
            Response.Write( "请输入最少 4 个字符!" );
            return;
          }

          string FontName = fontType.SelectedValue;

          int fontSize = 36;
          int x = 10;
          int y = 10;

           switch( FontName ) {
            case "方正黄草简体":
              x = -8;
```

```
                    y = -2;
                    fontSize = 42;
                    break;

                case "汉鼎繁淡古":
                    x = -10;
                    y = -2;
                    break;

                case "汉鼎繁印篆":
                    x = -15;
                    y = -2;
                    break;

                case "经典繁栈亭":
                    fontSize = 34;
                    x = -10;
                    y = -3;
                    break;
            }

            Bitmap bm = new Bitmap(110,100,System.Drawing.Imaging.PixelFormat.Format32bppArgb);
            Font font = new Font(FontName,fontSize,FontStyle.Bold);

            Graphics g = Graphics.FromImage( bm );

            g.TextRenderingHint =  System.Drawing.Text.TextRenderingHint.ClearTypeGridFit;

```

```
             /*   这里的三个属性可以根据情况开放.
             *
            //g.CompositingQuality = CompositingQuality.HighQuality;
            //g.SmoothingMode = SmoothingMode.HighQuality;
            //g.InterpolationMode = InterpolationMode.HighQualityBicubic;

            */
            g.Clear(Color.Red);       //赋予图像一个背景色

            g.DrawString( Name.Substring(2,1),font,new SolidBrush(Color.White),0,2 );
            g.DrawString( Name.Substring(3,1),font,new SolidBrush(Color.White),0,45-y );
            g.DrawString( Name.Substring(0,1),font,new SolidBrush(Color.White),35-x,2 );
            g.DrawString( Name.Substring(1,1),font,new SolidBrush(Color.White),35-x,45-y );
            g.Dispose();
            //设置保存路径
            bm.Save( Server.MapPath("../upload/c.jpg"),System.Drawing.Imaging.ImageFormat.Jpeg );
            bm.Dispose();
            //显示产生的图片
            Image1.ImageUrl = Server.MapPath("../upload/c.jpg") ;
            Image1.Visible = true;
        }
    }
}

```

效果:



博客园印

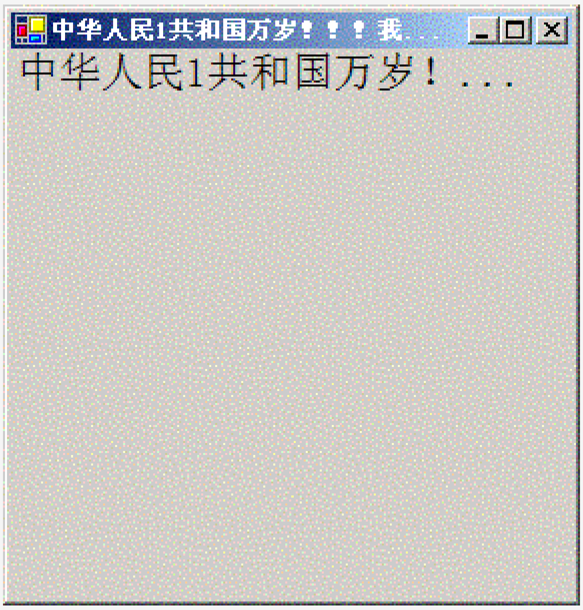
○ 方正黄草简体
○ 汉鼎繁淡古
● 汉鼎繁印篆
○ 经典繁栈亭

MinimizeName在C#中的实现代码（解决了我那个GDI+的问题）

代码如下：

```
//测试用的文字是：中华人民1共和国万岁！！！我爱北23京4天安门！！！
//字体：宋体，小三
int MagicNumber = 5;
protected override void OnPaint(PaintEventArgs e){
        Graphics g = e.Graphics;


        g.DrawString(MeasureString(g,Text,Font,this.ClientSize.Width-MagicNumber),Font,SystemBrushes.WindowText,0,0);

        base.OnPaint (e);
}

private string MeasureString(Graphics g,string text,Font font,int maxwidth){
        SizeF sf = g.MeasureString(text,font);
        if(sf.Width<maxwidth)return text;

        int len = text.Length;
        string s = "";
        StringBuilder sb = new StringBuilder(64);

        for(int i=len-1;i>=0;i--){
                sb.Remove(0,sb.Length);
                sb.Append(text.Substring(0,i));
                sb.Append("...");
                s = sb.ToString();
                sf = g.MeasureString(s,font);
                if(sf.Width<=maxwidth)break;
        }


        return s;
}

private void Form1_Resize(object sender, System.EventArgs e){
        Invalidate();
}
```

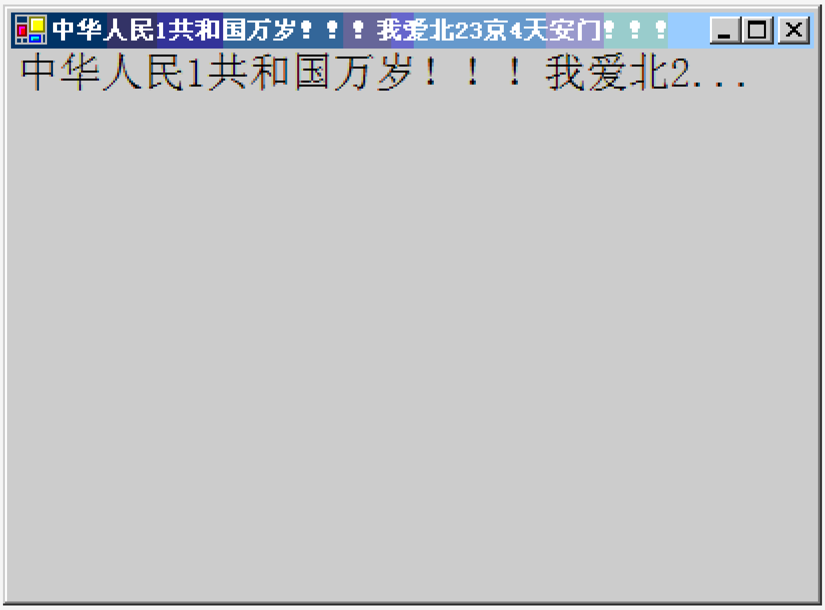
这是效果 1：



中华人民1共和国万岁！！！我...

中华人民1共和国万岁！...

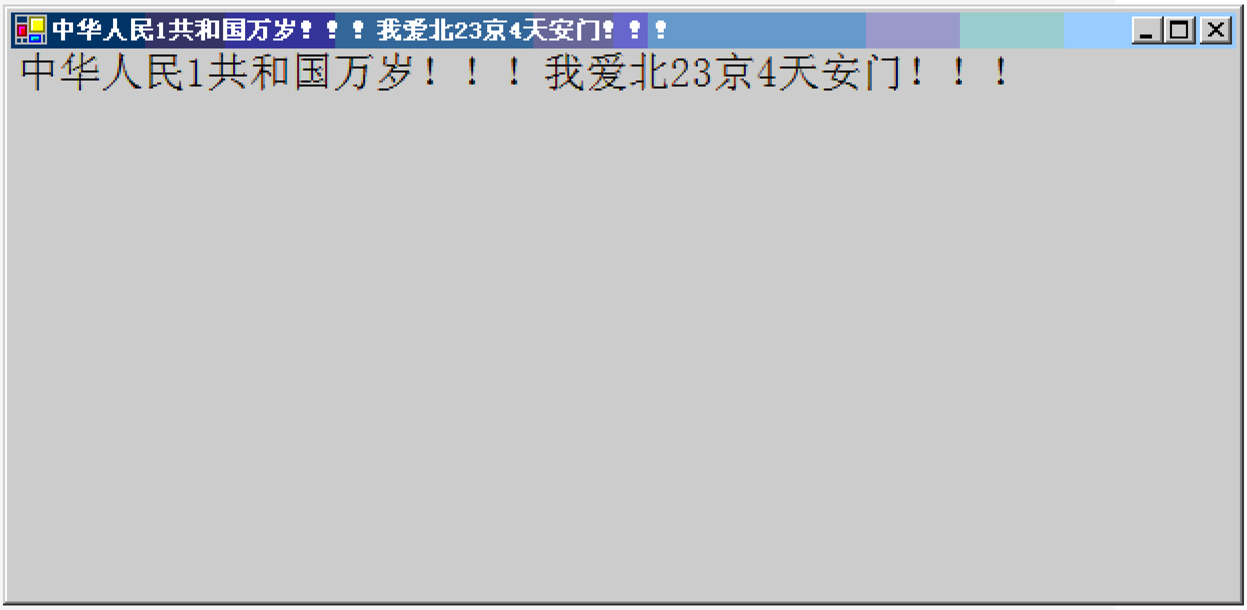
这是效果 2：



中华人民1共和国万岁！！！我爱北23京4天安门！！！

中华人民1共和国万岁！！！我爱北2...

这是效果 3：



中华人民1共和国万岁！！！我爱北23京4天安门！！！

你可以看看 StringFormat 的 Trimming 属性

当你在 DrawString 是给出一个 Rectangle 时，它会把 Text 画在指定的 Rectangle 内超出部分根据你的 Trimming 设置处理。Windows API DrawString 一直就有这个特性的。

```
protected override void OnPaint(PaintEventArgs e){
Graphics g = e.Graphics;

StringFormat sf = StringFormat.GenericTypographic;
sf.Trimming = StringTrimming.EllipsisCharacter;
g.DrawString(Text,Font,SystemBrushes.WindowText,new Rectangle(3,3,this.Width-6,this.Height-6),sf);
            //这行我自己的调用，就废掉了。
//g.DrawString(MeasureString(g,Text,Font,this.ClientSize.Width-MagicNumber),Font,SystemBrushes.WindowText,0,0);

base.OnPaint (e);
}
```

实现图片圆角

今天特意找了个关于图片去圆角的文章

找到http://www.cnblogs.com/lovecherry/archive/2006/05/17/402541.html

提供了主要的关键代码：

我稍微的去了些功能 只留下了一个方法 来处理 图片圆角的功能

嘿嘿

其实其他的几个方法还没来得及看

创建一个类

COPY进去

```
using System;

using System.Data;

using System.Configuration;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;
```

```
/// <summary>
/// Class1 的摘要说明
/// </summary>
public class MyGDI
{

    //创建 圆角图片的方法
    方法参数的说明
    public static void CreateRoundedCorner(string sSrcFilePath, string sDstFilePath, string sCornerLocation)
    {
        System.Drawing.Image image = System.Drawing.Image.FromFile(sSrcFilePath);
        Graphics g = Graphics.FromImage(image);
        g.SmoothingMode = SmoothingMode.HighQuality;
        g.InterpolationMode = InterpolationMode.HighQualityBicubic;
        g.CompositingQuality = CompositingQuality.HighQuality;
        Rectangle rect = new Rectangle(0, 0, image.Width, image.Height);
        GraphicsPath rectPath = CreateRoundRectanglePath(rect, image.Width / 10, sCornerLocation); //构建圆角外部路径
        Brush b = new SolidBrush(Color.White);//圆角背景白色
        g.DrawPath(new Pen(b), rectPath);
        g.FillPath(b, rectPath);
        g.Dispose();
        image.Save(sDstFilePath, ImageFormat.Jpeg);
        image.Dispose();
    }

    private static GraphicsPath CreateRoundRectanglePath(Rectangle rect, int radius, string sPosition)
```

```
{
 GraphicsPath rectPath = new GraphicsPath();
 switch (sPosition)
   {
     case "TopLeft":
         {
          rectPath.AddArc(rect.Left, rect.Top, radius * 2, radius * 2, 180, 90);
          rectPath.AddLine(rect.Left, rect.Top, rect.Left, rect.Top + radius);
          break;
       }

     case "TopRight":
         {
          rectPath.AddArc(rect.Right - radius * 2, rect.Top, radius * 2, radius * 2, 270, 90);
          rectPath.AddLine(rect.Right, rect.Top, rect.Right - radius, rect.Top);
          break;
       }

     case "BottomLeft":
         {
          rectPath.AddArc(rect.Left, rect.Bottom - radius * 2, radius * 2, radius * 2, 90, 90);
          rectPath.AddLine(rect.Left, rect.Bottom - radius, rect.Left, rect.Bottom);
          break;
       }

     case "BottomRight":
         {
          rectPath.AddArc(rect.Right - radius * 2, rect.Bottom - radius * 2, radius * 2, radius * 2, 0, 90);
```

```
                    rectPath.AddLine(rect.Right - radius, rect.Bottom, rect.Right, rect.Bottom);
                    break;
                }

            }
            return rectPath;
        }
    }
```

我使用了一个笨方法来实现 一个图片一次性的将四个圆角全部切出来

这个方法太笨了

```
    protected void Page_Load(object sender, EventArgs e)
    {
     //   MyGDI.
        MyGDI.CreateRoundedCorner(@"C:\01.JPG", @"C:\02.JPG", "TopLeft");
        MyGDI.CreateRoundedCorner(@"c:\02.jpg", @"c:\03.jpg", "TopRight");
        MyGDI.CreateRoundedCorner(@"C:\03.JPG", @"C:\04.JPG", "BottomLeft");
        MyGDI.CreateRoundedCorner(@"c:\04.jpg", @"c:\05.jpg", "BottomRight");


    }
```

我试过这么做

```
        //case "TopAll":
            //    {
            //        rectPath.AddArc(rect.Left, rect.Top, radius * 2, radius * 2, 180, 90);
            //        rectPath.AddLine(rect.Left, rect.Top, rect.Left, rect.Top + radius);
            //        rectPath.AddArc(rect.Right - radius * 2, rect.Top, radius * 2, radius * 2, 270, 90);
            //        rectPath.AddLine(rect.Right, rect.Top, rect.Right - radius, rect.Top);
            //        //rectPath.AddArc(rect.Left, rect.Bottom - radius * 2, radius * 2, radius * 2, 90, 90);
            //        //rectPath.AddLine(rect.Left, rect.Bottom - radius, rect.Left, rect.Bottom);
            //        //rectPath.AddArc(rect.Right - radius * 2, rect.Bottom - radius * 2, radius * 2, radius * 2, 0, 90);
            //        //rectPath.AddLine(rect.Right - radius, rect.Bottom, rect.Right, rect.Bottom);
            //        break;
            //    }
            //case "BottomAll":
            //    {
            //        //rectPath.AddArc(rect.Left, rect.Top, radius * 2, radius * 2, 180, 90);
            //        //rectPath.AddLine(rect.Left, rect.Top, rect.Left, rect.Top + radius);
            //        //rectPath.AddArc(rect.Right - radius * 2, rect.Top, radius * 2, radius * 2, 270, 90);
            //        //rectPath.AddLine(rect.Right, rect.Top, rect.Right - radius, rect.Top);
            //        rectPath.AddArc(rect.Left, rect.Bottom - radius * 2, radius * 2, radius * 2, 90, 90);
            //        rectPath.AddLine(rect.Left, rect.Bottom - radius, rect.Left, rect.Bottom);
            //        rectPath.AddArc(rect.Right - radius * 2, rect.Bottom - radius * 2, radius * 2, radius * 2, 0, 90);
            //        rectPath.AddLine(rect.Right - radius, rect.Bottom, rect.Right, rect.Bottom);
            //        break;
            //    }
```

但是 图片出来后 面目全非

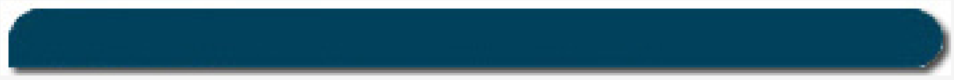
哪位朋友可以帮我看看 有没有更好点的方法

.NET下GDI＋的一些常用应用（水印，文字，圆角处理）

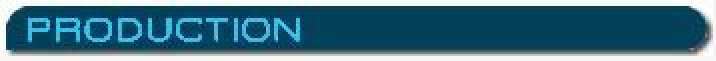
在某些情况下希望处理一些图片，比如

给图片添加一般文字：

原图



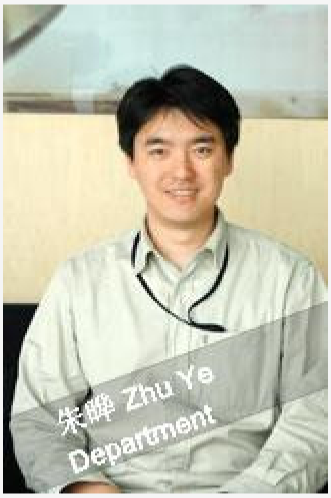
程序处理成



给图片添加水印：

原图

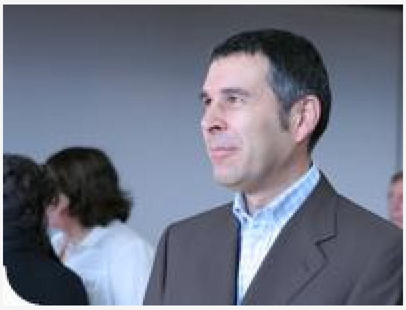
程序处理成



还有圆角效果：

原图

程序处理成

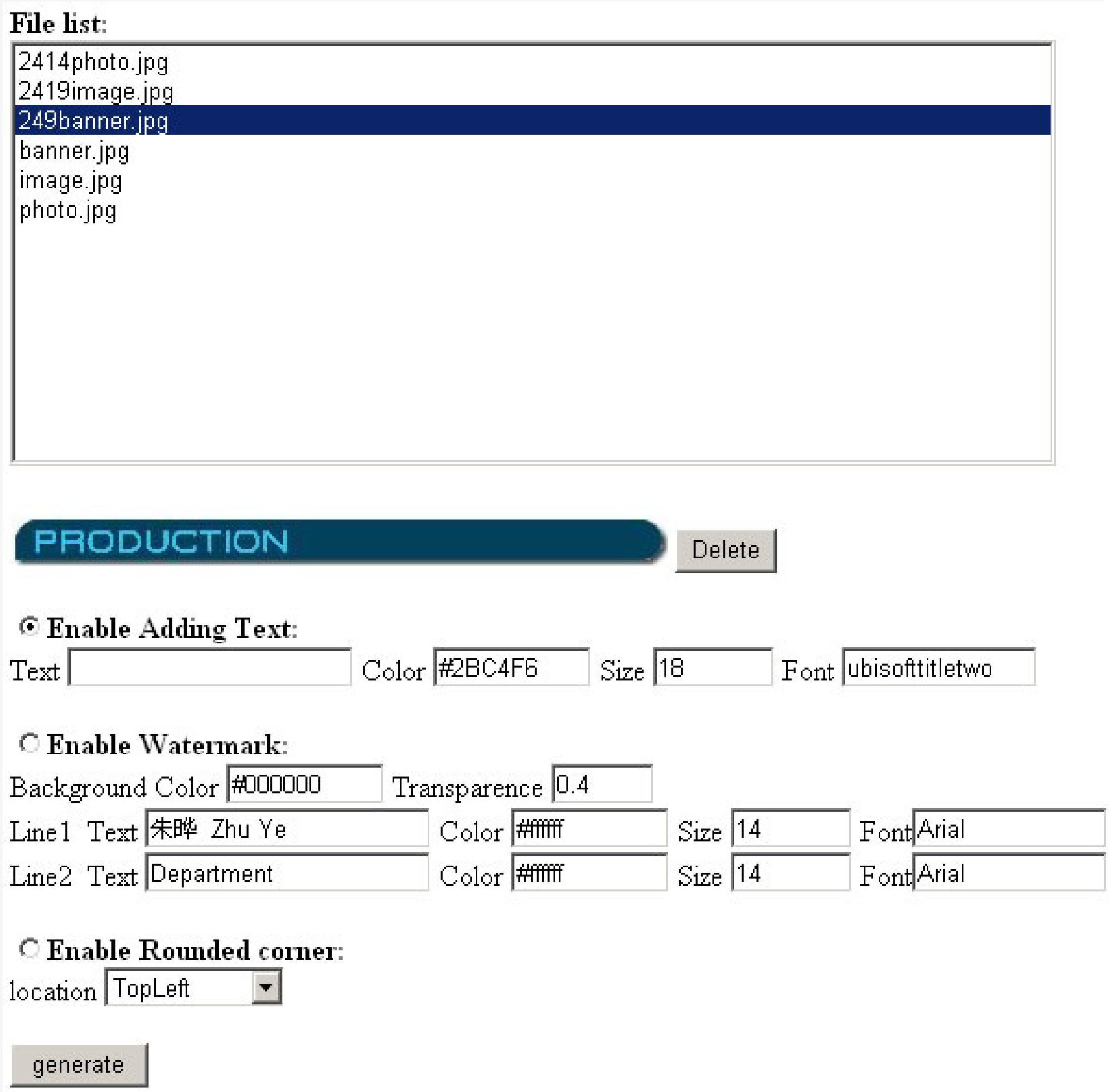


注：

（1）以下程序针对这个应用，也许这些对你没有用，但是基本的原理都差不多

（2）以上图为测试用，应该不算侵犯肖像权吧

程序界面：



File list:

2414photo.jpg
2419image.jpg
249banner.jpg
banner.jpg
image.jpg
photo.jpg

● Enable Adding Text:

Text | Color #2BC4F6 | Size 18 | Font ubisofttitletwo

○ Enable Watermark:

Background Color #000000 | Transparence 0.4

Line1 Text 朱晔 Zhu Ye | Color #ffffff | Size 14 | Font Arial

Line2 Text Department | Color #ffffff | Size 14 | Font Arial

○ Enable Rounded corner:

location TopLeft

程序点击这里下载。

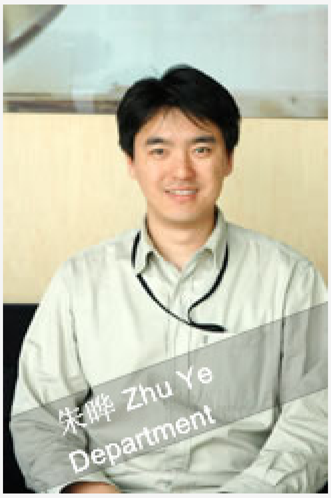
如果需要提高图片质量，把

image.Save(sDstFilePath, ImageFormat.Jpeg);

改成

```
ImageCodecInfo myImageCodecInfo;
        Encoder myEncoder;
        EncoderParameter myEncoderParameter;
        EncoderParameters myEncoderParameters;
        myImageCodecInfo = ImageCodecInfo.GetImageEncoders()[0];
        myEncoder = Encoder.Quality;
        myEncoderParameters = new EncoderParameters(1);
        myEncoderParameter = new EncoderParameter(myEncoder, 100L);  //  0-100
        myEncoderParameters.Param[0] = myEncoderParameter;
        image.Save(sDstFilePath, myImageCodecInfo, myEncoderParameters);
        myEncoderParameter.Dispose();
        myEncoderParameters.Dispose();
```

这是提高质量后的效果：



附一些关键代码：

```
public class MyGDI
{
    public static void CreateWatermark(string sSrcFilePath, string sDstFilePath, string sText1, string sColor1, string sSize1, string sFont1, string sText2, string sColor2, string sSize2, string sFont2, string sBgColor, string sTransparence)
    {
        System.Drawing.Image image = System.Drawing.Image.FromFile(sSrcFilePath);
        Graphics g = Graphics.FromImage(image);
        g.SmoothingMode = SmoothingMode.AntiAlias;
        g.InterpolationMode = InterpolationMode.HighQualityBicubic;
        g.CompositingQuality = CompositingQuality.HighQuality;
        g.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAlias; //文字抗锯齿
        g.DrawImage(image, 0, 0, image.Width, image.Height);
        Font f1 = new Font(sFont1, float.Parse(sSize1));
        Font f2 = new Font(sFont2, float.Parse(sSize2));
        Brush brushfortext1 = new SolidBrush(ColorTranslator.FromHtml(sColor1));
        Brush brushfortext2 = new SolidBrush(ColorTranslator.FromHtml(sColor2));
        Brush brushforbg = new SolidBrush(Color.FromArgb(Convert.ToInt16(255 * float.Parse(sTransparence)), ColorTranslator.FromHtml(sBgColor)));
        g.RotateTransform(-20);
        Rectangle rect = new Rectangle(-image.Width/2-50, image.Height - 50, image.Width * 2, 40);
        g.DrawRectangle(new Pen(brushforbg), rect);
        g.FillRectangle(brushforbg, rect);
        Rectangle rectfortext1 = new Rectangle(-image.Width/2 + image.Width / 5, image.Height - 45, image.Width * 2, 60);
        for (int i = 0; i < 10; i++)
            g.DrawString(sText1, f1, brushfortext1, rectfortext1);
        Rectangle rectfortext2 = new Rectangle(-image.Width / 2 + image.Width / 5, image.Height -25, image.Width * 2, 60);
        for (int i = 0; i < 10; i++)
            g.DrawString(sText2, f2, brushfortext2, rectfortext2);
        image.Save(sDstFilePath, ImageFormat.Jpeg);
        image.Dispose();

    }

    public static void CreateRoundedCorner(string sSrcFilePath, string sDstFilePath, string sCornerLocation)
    {
        System.Drawing.Image image = System.Drawing.Image.FromFile(sSrcFilePath);
        Graphics g = Graphics.FromImage(image);
        g.SmoothingMode = SmoothingMode.HighQuality;
        g.InterpolationMode = InterpolationMode.HighQualityBicubic;
        g.CompositingQuality = CompositingQuality.HighQuality;
        Rectangle rect = new Rectangle(0, 0, image.Width, image.Height);
        GraphicsPath rectPath = CreateRoundRectanglePath(rect, image.Width / 10, sCornerLocation); //构建圆角外部路径
        Brush b = new SolidBrush(Color.White);//圆角背景白色
        g.DrawPath(new Pen(b), rectPath);
        g.FillPath(b, rectPath);
        g.Dispose();
        image.Save(sDstFilePath, ImageFormat.Jpeg);
        image.Dispose();
    }

    public static void CreatePlainText(string sSrcFilePath, string sDstFilePath,string sText, string sColor, string sSize, string sFont)
    {
        System.Drawing.Image image = System.Drawing.Image.FromFile(sSrcFilePath);
        Graphics g = Graphics.FromImage(image);
        g.SmoothingMode = SmoothingMode.AntiAlias;
        g.InterpolationMode = InterpolationMode.HighQualityBicubic;
        g.CompositingQuality = CompositingQuality.HighQuality;
        g.DrawImage(image, 0, 0, image.Width, image.Height);
        g.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAlias; //文字抗锯齿
        Font f = new Font(sFont,float.Parse(sSize));
        Brush b = new SolidBrush(ColorTranslator.FromHtml(sColor));
        Rectangle rect = new Rectangle(10, 5, image.Width, image.Height); //适当空开一段距离
        for (int i = 0; i < 30; i++) //加强亮度
            g.DrawString(sText, f, b, rect);
        image.Save(sDstFilePath, ImageFormat.Jpeg);
        image.Dispose();
    }

    private static GraphicsPath CreateRoundRectanglePath(Rectangle rect, int radius, string sPosition)
    {
        GraphicsPath rectPath = new GraphicsPath();
        switch (sPosition)
        {
            case "TopLeft":
                {
                    rectPath.AddArc(rect.Left, rect.Top, radius * 2, radius * 2, 180, 90);
                    rectPath.AddLine(rect.Left, rect.Top, rect.Left, rect.Top + radius);
                    break;
                }

            case "TopRight":
                {
                    rectPath.AddArc(rect.Right - radius * 2, rect.Top, radius * 2, radius * 2, 270, 9
0);
                    rectPath.AddLine(rect.Right, rect.Top, rect.Right - radius, rect.Top);
                    break;
                }

            case "BottomLeft":
                {
                    rectPath.AddArc(rect.Left, rect.Bottom - radius * 2, radius * 2, radius * 2, 90, 9
0);
                    rectPath.AddLine(rect.Left, rect.Bottom - radius, rect.Left, rect.Bottom);
                    break;
                }

            case "BottomRight":
                {
                    rectPath.AddArc(rect.Right - radius * 2, rect.Bottom - radius * 2, radius * 2, rad
ius * 2, 0, 90);
                    rectPath.AddLine(rect.Right - radius, rect.Bottom, rect.Right, rect.Bottom);
                    break;
                }
        }
        return rectPath;
    }
}
```

动态生成文字图片解决方案

大家都知道我们如果想把网页上的文字做出比较炫的效果，便只能用 POTOSHOP、FIREWORK 等图像处理软件把文字做成图片来实现，因为这样才不会依赖浏览者的字体、浏览器类型等。可是在我们的 WEB 应用中又往往是动态的文字，我们便不能用图像处理软件来处理了，只能让 WEB 程序动态生成，幸运地是.Net Framework 给我们提供了便利，下面我们就利用 System.Drawing 命名空间下的 Bitmap 类与 Graphics 类来编写一个生成文字图片的类，使用该类生成图片时能满足以下需求：

1、可以指定文字字体、大小和颜色（注:指定的文字在 WEB 服务器上需要有该字库）；

2、可以加文字阴影；

3、可以指定文字的透明度；

4、可以指定背景图片或背景颜色；

5、可以指定生成的图片大小（宽度与高度）；

6、可以指定文字的位置（左边距和上边距）；

7、当用户设定的文字字号太大，能自动调整文字大小使之能适应生成图片的大小。

该类实现代码如下：

```
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace Ycweb.Controls.Utility
{
    /**//// <summary>
    /// WaterMark
    /// </summary>
    public class Watermark
    {
        private int _width;  //背景宽度
        private int _height; //背景高度
```

```
  private string _fontFamily;  //字体属性
  private int _fontSize;  //字体大小
  private bool _adaptable;  //若文字太大，是否根据背景图来调整文字大小，默认为适应
  private FontStyle _fontStyle;  //文字风格
  private bool _shadow; //水印文字是否使用阴影
  private string _backgroundImage;  //背景图片
  private Color _bgColor;  //背景颜色
  private int _left;    //水印文字的左边距
  private string _resultImage;  // 生成后的图片
  private string _text;  //文字
  private int _top;     // 水印文字的顶边距
  private int _alpha;  //透明度 0-255,255 表示不透明
  private int _red;
  private int _green;
private int _blue;
private long _quality;   //输出图片质量，质量范围 0-100,类型为 long



public Watermark()
   {
    //
    // TODO: Add constructor logic here
    //
    _width=460;
    _height=30;
    _fontFamily = "华文行楷";
    _fontSize = 20;
    _fontStyle=FontStyle.Regular;
    _adaptable=false ;
```

```
        _shadow=false;
        _left = 0;
        _top = 0;
        _alpha = 255;
        _red = 0;
        _green = 0;
        _blue = 0;
        _backgroundImage="";
        _quality=100;
        _bgColor=Color.FromArgb(255,229,229,229);

    }

      /**//// <summary>
    /// 字体
    /// </summary>
    public string FontFamily
      {
          set { this._fontFamily = value; }
    }

      /**//// <summary>
    /// 文字大小
    /// </summary>
    public int FontSize
      {
          set { this._fontSize = value; }
    }

      /**//// <summary>
```

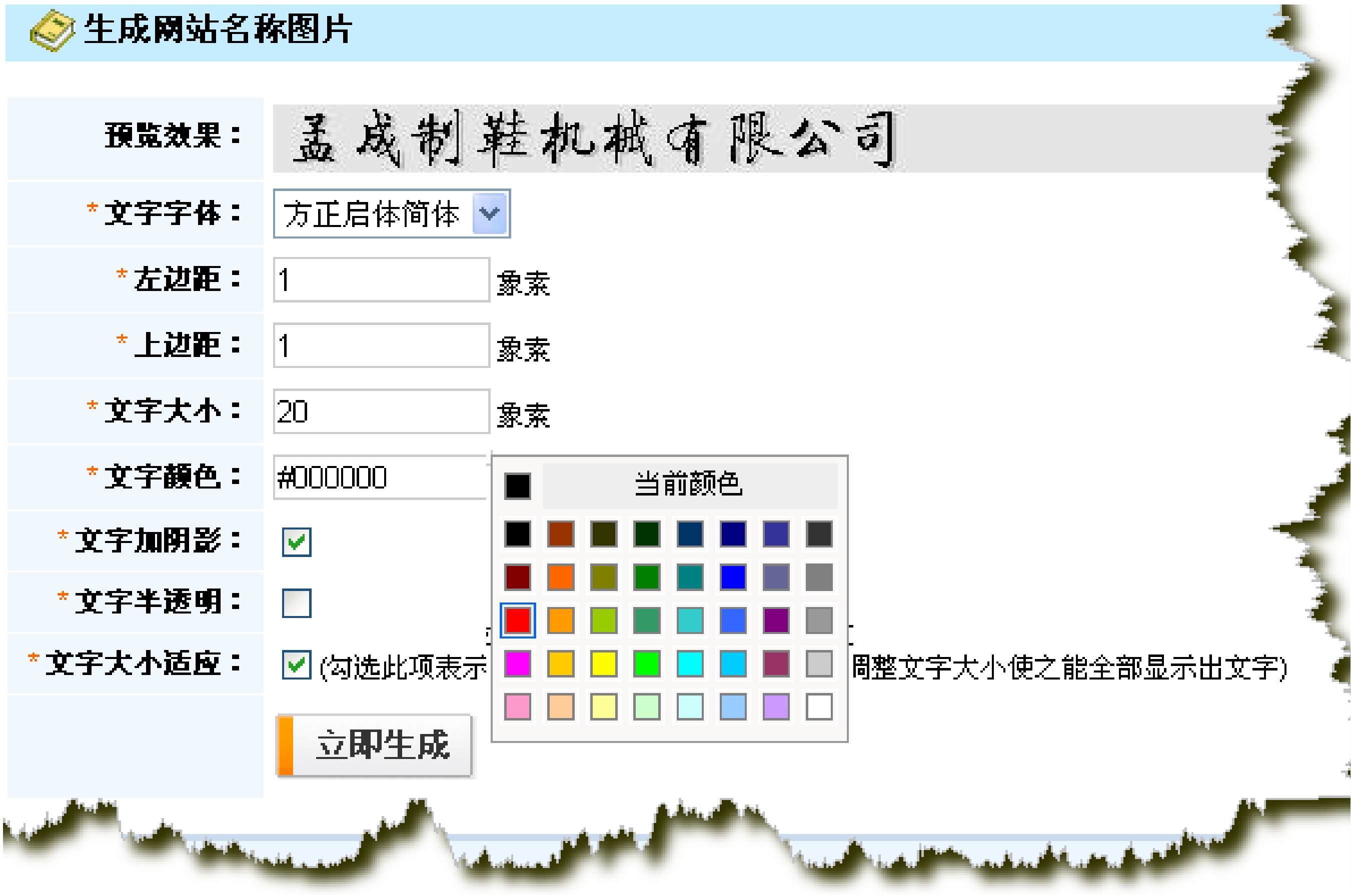
```
/// 文字风格
/// </summary>
public FontStyle FontStyle
 {
     get{return _fontStyle;}
     set{_fontStyle = value;}
}

 /**//// <summary>
/// 透明度 0-255,255 表示不透明
/// </summary>
public int Alpha
 {
     get { return _alpha; }
     set { _alpha = value; }
}

 /**//// <summary>
/// 水印文字是否使用阴影
/// </summary>
public bool Shadow
 {
     get { return _shadow; }
     set { _shadow = value; }
}

  public int Red
   {
       get { return _red; }
       set { _red = value; }
```

```
    }

    public int Green
     {
         get { return _green; }
         set { _green = value; }
    }

    public int Blue
     {
         get { return _blue; }
         set { _blue = value; }
    }

     /**/// <summary>
    /// 底图
    /// </summary>
    public string BackgroundImage
     {
         set { this._backgroundImage = value; }
    }

     /**/// <summary>
    /// 水印文字的左边距
    /// </summary>
    public int Left
     {
         set { this._left = value; }
    }


     /**//// <summary>
    /// 水印文字的顶边距
    /// </summary>
    public int Top
     {
         set { this._top = value; }
    }

     /**//// <summary>
    /// 生成后的图片
    /// </summary>
    public string ResultImage
     {
         set { this._resultImage = value; }
    }

     /**//// <summary>
    /// 水印文本
    /// </summary>
    public string Text
     {
         set { this._text = value; }
    }


     /**//// <summary>
    /// 生成图片的宽度
    /// </summary>
    public int Width
```

```
     {
         get { return _width; }
         set { _width = value; }
    }

     /**//// <summary>
    /// 生成图片的高度
    /// </summary>
    public int Height
     {
         get { return _height; }
         set { _height = value; }
    }

     /**//// <summary>
    /// 若文字太大，是否根据背景图来调整文字大小，默认为适应
    /// </summary>
    public bool Adaptable
     {
         get { return _adaptable; }
         set { _adaptable = value; }
    }

    public Color BgColor
     {
         get { return _bgColor; }
         set { _bgColor = value; }
    }

     /**//// <summary>
```

```
/// 输出图片质量，质量范围 0-100,类型为 long
/// </summary>
public long Quality
  {
     get { return _quality; }
     set { _quality = value; }
}

  /**//// <summary>
/// 立即生成水印效果图
/// </summary>
/// <returns>生成成功返回 true,否则返回 false</returns>
public bool Create()
  {
   try
     {
      Bitmap bitmap;
      Graphics g;

      //使用纯背景色
      if(this._backgroundImage.Trim()=="")
        {
         bitmap = new Bitmap(this._width, this._height, PixelFormat.Format64bppArgb);
         g = Graphics.FromImage(bitmap);
         g.Clear(this._bgColor);
      }
      else
        {
         bitmap = new Bitmap(Image.FromFile(this._backgroundImage));
         g = Graphics.FromImage(bitmap);
      }
      g.SmoothingMode = SmoothingMode.HighQuality;
      g.InterpolationMode = InterpolationMode.HighQualityBicubic;
      g.CompositingQuality=CompositingQuality.HighQuality;

      Font f = new Font(_fontFamily, _fontSize,_fontStyle);
      SizeF size = g.MeasureString(_text, f);

      // 调整文字大小直到能适应图片尺寸
      while(_adaptable==true && size.Width > bitmap.Width)
        {
         _fontSize--;
         f = new Font(_fontFamily, _fontSize, _fontStyle);
         size = g.MeasureString(_text, f);
      }

      Brush b = new SolidBrush(Color.FromArgb(_alpha, _red, _green, _blue));
      StringFormat StrFormat = new StringFormat();
      StrFormat.Alignment = StringAlignment.Near;

      if(this._shadow)
        {
         Brush b2=new SolidBrush(Color.FromArgb(90, 0, 0, 0));
         g.DrawString(_text, f, b2,_left+2, _top+1);
      }
      g.DrawString(_text, f, b, new PointF(_left, _top), StrFormat);

      bitmap.Save(this._resultImage, ImageFormat.Jpeg);
      bitmap.Dispose();
      g.Dispose();
```

```
            return true;
        }
        catch
         {
            return false;
        }
    }
 }


}
```



生成网站名称图片

预览效果：	孟成制鞋机械有限公司
* 文字字体：	方正启体简体
* 左边距：	1 象素
* 上边距：	1 象素
* 文字大小：	20 象素
* 文字颜色：	#000000
* 文字加阴影：	☑
* 文字半透明：	□
* 文字大小适应：	☑ (勾选此项表示 周整文字大小使之能全部显示出文字)
	立即生成

调用方法如下

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using Ycweb.Controls.Utility;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {


  Watermark _Watermark = new   Watermark();
 _Watermark.Top = 0;
 _Watermark.Left = 0;
 _Watermark.Height = 200;
 _Watermark.Width = 600;
 _Watermark.Quality = 100;
 _Watermark.Red = 20;
 _Watermark.ResultImage = @"c:\test.jpg"; //注意要有写的权限
 _Watermark.Shadow = true;
 _Watermark.Text = "你好啊？ ？ ？ ？ ";
 _Watermark.Adaptable = true;
 _Watermark.Alpha = 255;
 _Watermark.BgColor = System.Drawing.Color.Red;
_Watermark.Blue = 20;

_Watermark.FontFamily = "隶书";

_Watermark.FontSize = 14;


bool result=_Watermark.Create();


if(result)
{
Response.Write("OK");
}
else
{
Response.Write("Error");
}
Response.End();
}



   }
```

Asp.net(C#)给图片加上水印效果

下面的代码中，加文字水印和加图片水印的代码不能共存

我是为了方便显示才写在一块的

```
private void Btn_Upload_Click(object sender, System.EventArgs e)
        {
         if(UploadFile.PostedFile.FileName.Trim()!="")
            {
            //上传文件
            string extension = Path.GetExtension(UploadFile.PostedFile.FileName).ToUpper();
            string fileName = DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.ToString() + DateTime.Now.Second.ToString();
            string path = Server.MapPath(".") + "/UploadFile/" + fileName + extension;
            UploadFile.PostedFile.SaveAs(path);

            //加文字水印，注意，这里的代码和以下加图片水印的代码不能共存
            System.Drawing.Image image = System.Drawing.Image.FromFile(path);
            Graphics g = Graphics.FromImage(image);
            g.DrawImage(image, 0, 0, image.Width, image.Height);
            Font f = new Font("Verdana", 32);
            Brush b = new SolidBrush(Color.White);
            string addText = AddText.Value.Trim();
            g.DrawString(addText, f, b, 10, 10);
            g.Dispose();

            //加图片水印
            System.Drawing.Image image = System.Drawing.Image.FromFile(path);
            System.Drawing.Image copyImage = System.Drawing.Image.FromFile( Server.MapPath(".") + "/Alex.gif");
            Graphics g = Graphics.FromImage(image);
            g.DrawImage(copyImage, new Rectangle(image.Width-copyImage.Width, image.Height-copyImage.Height, copyImage.Width, copyImage.Height), 0, 0, copyImage.Width, copyImage.Height, GraphicsUnit.Pixel);
            g.Dispose();

            //保存加水印过后的图片,删除原始图片
            string newPath = Server.MapPath(".") + "/UploadFile/" + fileName + "_new" + extension;
            image.Save(newPath);
            image.Dispose();
            if(File.Exists(path))
              {
                File.Delete(path);
            }

            Response.Redirect(newPath);
        }
    }
```

给图片添加水印效果图的函数(可以在图片上添加自己的版权和LOGO图片的水印)【转载】

```
        protected void Button1_ServerClick(object sender, System.EventArgs e)
          {
            if(File1.PostedFile!=null)
              {
                string fileName = File1.PostedFile.FileName ;
                //取得上传文件的扩展名
                string fileExtName =fileName.Substring(fileName.LastIndexOf("."));
                //创建 GUID,作为上传文件的唯一标识
                System.Guid myGuid = System.Guid.NewGuid();
                string savePath = Server.MapPath("~").ToString() + "//uploadfile//" + myGuid.ToString() + "." + fileExtName;
                string newSavePath = Server.MapPath("~").ToString() + "//uploadfile1//" + myGuid.ToString() + "." + fileExtName;
                string waterMarkPath = Server.MapPath("~").ToString() + "//uploadfile//watermark.bmp";
                File1.PostedFile.SaveAs(savePath);
                watermark myWatermark = new watermark();
                myWatermark.MakeWatermark("北京朝阳",waterMarkPath,savePath,newSavePath);
//              //文件的相关属性:文件名,文件类型,文件大小
//              myFile.PostedFile.FileName;
//              myFile.PostedFile.ContentType ;
//              myFile.PostedFile.ContentLength.ToString();
            }

        }
```

看看 类的处理

```
/*
 * 给图片设置水印效果图的函数
 * 可以在图片上添加自己的版权和 LOGO 图片的水印
 *
 * 原作者:Joel Neubeck
 *
 * 本人在原作者的基础上进行了部分修改,并改为一个单独的函数
 * 大家可以根据需要修改此代码
 * 但请保留原作者信息.
 *
 * 星宿.net(zhuhee)
 * 修改于 2006-6-14
 *
 */
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;

namespace WebWaterMark.Components
{
    /// <summary>
    /// watermark 的摘要说明。
    /// </summary>
    public class watermark
    {
        public watermark()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
        /// <summary>
        /// 用于处理图片
        /// 给图片加入水印
        /// </summary>
        /// <param name="Copyright">需要写入的版权信息</param>
        /// <param name="MarkBmpPath">需要假如的 logo 图片</param>
        /// <param name="photoPath">需要处理的图片的路径</param>
        /// <param name="savePhotoPath">保存的图片路径</param>
        public void MakeWatermark(string Copyright,string MarkBmpPath,string photoPath,string savePhotoPath)
        {

            //创建一个 image 对象,即要被处理的图片
            Image imgPhoto = Image.FromFile(photoPath);
            int phWidth = imgPhoto.Width;
            int phHeight = imgPhoto.Height;

            //创建原始图片大小的 Bitmap
            Bitmap bmPhoto = new Bitmap(phWidth, phHeight, PixelFormat.Format24bppRgb);

            bmPhoto.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);

            //将位图 bmPhoto 加载到 Graphics 对象
            Graphics grPhoto = Graphics.FromImage(bmPhoto);

```

```
//创建一个需要填充水银的 Image 对象
Image imgWatermark = new Bitmap(MarkBmpPath);
int wmWidth = imgWatermark.Width;
int wmHeight = imgWatermark.Height;

//------------------------------------------------------------
//第一步,插入版权信息
//------------------------------------------------------------

//设置图象的成相质量
grPhoto.SmoothingMode = SmoothingMode.AntiAlias;

//将原始图象绘制到 grPhoto 上
grPhoto.DrawImage(
    imgPhoto,                              // 要绘制的 Image 对象
    new Rectangle(0, 0, phWidth, phHeight), // 绘制图象的位置和大小
    0,                                // 要绘制的原图象部分的左上角的 X 坐标
    0,                                // 要绘制的原图象部分的左上角的 Y 坐标
    phWidth,                               // 要绘制的原图象的高度
    phHeight,                              // 要绘制的原图象的宽度
    GraphicsUnit.Pixel);                  // 源矩形的度量单位

//-----------------------------------------------------------
//字体大小放在一个数组中,最大字体为 32.
//感觉用处不大,可以自己再修改这里
//-----------------------------------------------------------
  int[] sizes = new int[]{32,14,12,10,8,6,4};

Font crFont = null;
SizeF crSize = new SizeF();
```

```
        //循环测试数组中所定义的字体大小是否适合版权信息,如果合适就使用此种字体大小
        for (int i=0 ;i<6; i++)
          {
            //设置字体类型,可以单独提出,作为参数
            crFont = new Font("迷你繁篆书", sizes[0], FontStyle.Bold);
            //测量此种字体大小
            crSize = grPhoto.MeasureString(Copyright, crFont);

            if((ushort)crSize.Width < (ushort)phWidth)
                break;
        }

        //给底部保留%3 的空间
        int yPixlesFromBottom = (int)(phHeight *.03);

        //设置字体在图片中的位置
        float yPosFromBottom = ((phHeight - yPixlesFromBottom)-(crSize.Height/2));

        //float xCenterOfImg = (phWidth/2);
        float xCenterOfImg = (phWidth-(crSize.Width)/2);
        //设置字体居中
        StringFormat StrFormat = new StringFormat();
        StrFormat.Alignment = StringAlignment.Center;

        //设置绘制文本的颜色和纹理 (Alpha=153)
        SolidBrush semiTransBrush2 = new SolidBrush(Color.FromArgb(153, 0, 0, 0));

        //将版权信息绘制到图象上
        grPhoto.DrawString(Copyright,                    //版权信息
            crFont,                          //字体
            semiTransBrush2,                   //绘制文本的颜色及纹理
            new PointF(xCenterOfImg+1,yPosFromBottom+1),  //绘制文本的位置
            StrFormat);                        //格式

        //设置绘制文本的颜色和纹理 (Alpha=153)
        SolidBrush semiTransBrush = new SolidBrush(Color.FromArgb(153, 255, 255, 255));

        //重新绘制版权信息,以让其具有阴影效果
        //将文本向又下移动一个象素
        grPhoto.DrawString(Copyright,              //版权信息
            crFont,                          //字体
            semiTransBrush,                    //绘制文本的颜色及纹理
            new PointF(xCenterOfImg,yPosFromBottom),  //绘制文本的位置
            StrFormat);                        //格式



        //------------------------------------------------------------
        //第二步,插入图片水印
        //------------------------------------------------------------

        //在原来修改过的 bmPhoto 上创建一个水银位图
        Bitmap bmWatermark = new Bitmap(bmPhoto);

        bmWatermark.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);
        //将位图 bmWatermark 加载到 Graphics 对象
        Graphics grWatermark = Graphics.FromImage(bmWatermark);

```

```
//To achieve a transulcent watermark we will apply (2) color
//manipulations by defineing a ImageAttributes object and
//seting (2) of its properties.
ImageAttributes imageAttributes = new ImageAttributes();

//The first step in manipulating the watermark image is to replace
//the background color with one that is trasparent (Alpha=0, R=0, G=0, B=0)
//to do this we will use a Colormap and use this to define a RemapTable
ColorMap colorMap = new ColorMap();

//My watermark was defined with a background of 100% Green this will
//be the color we search for and replace with transparency
colorMap.OldColor = Color.FromArgb(255, 0, 255, 0);
colorMap.NewColor = Color.FromArgb(0, 0, 0, 0);

  ColorMap[] remapTable = { colorMap };

imageAttributes.SetRemapTable(remapTable, ColorAdjustType.Bitmap);

//The second color manipulation is used to change the opacity of the
//watermark.  This is done by applying a 5x5 matrix that contains the
//coordinates for the RGBA space.  By setting the 3rd row and 3rd column
//to 0.3f we achive a level of opacity
  float[][] colorMatrixElements = {
                        new float[] {1.0f,  0.0f,  0.0f,  0.0f, 0.0f},
                        new float[] {0.0f,  1.0f,  0.0f,  0.0f, 0.0f},
                        new float[] {0.0f,  0.0f,  1.0f,  0.0f, 0.0f},
                        new float[] {0.0f,  0.0f,  0.0f,  0.3f, 0.0f},
                        new float[] {0.0f,  0.0f,  0.0f,  0.0f, 1.0f}};
ColorMatrix wmColorMatrix = new ColorMatrix(colorMatrixElements);
```

```

            imageAttributes.SetColorMatrix(wmColorMatrix, ColorMatrixFlag.Default,
                ColorAdjustType.Bitmap);

            //For this example we will place the watermark in the upper right
            //hand corner of the photograph. offset down 10 pixels and to the
            //left 10 pixles

            int xPosOfWm = ((phWidth - wmWidth) - 10);
            int yPosOfWm = 10;

            grWatermark.DrawImage(imgWatermark,
                new Rectangle(xPosOfWm, yPosOfWm, wmWidth, wmHeight),  //Set the detination Position
                0,                  // x-coordinate of the portion of the source image to draw.
                0,                  // y-coordinate of the portion of the source image to draw.
                wmWidth,            // Watermark Width
                wmHeight,           // Watermark Height
                GraphicsUnit.Pixel, // Unit of measurment
                imageAttributes);   //ImageAttributes Object
            //Replace the original photgraphs bitmap with the new Bitmap
            imgPhoto = bmWatermark;
            grPhoto.Dispose();
            grWatermark.Dispose();
            //save new image to file system.
            imgPhoto.Save(savePhotoPath, ImageFormat.Jpeg);
            imgPhoto.Dispose();
            imgWatermark.Dispose();
        }
    }
```

```
}
```

GDI+位图透明

前段时间做了一个图片透明画的代码，基本思路是使用ColorMatrix设置位图的Alpha通道，使其透明化。这类代码可能高手都懒得写，像我等菜鸟要用时就得费一番周则研究了，所以我把做完的代码发上来，大家有需要用的就拿去用，如果高兴的话还可以评论里说声加油之类的话，呵呵。

利用 ColorMatrix 还可以调整整个位图的 RGB 值，看各位需要发挥了。

代码如下：

```
/// <summary>
/// 改变图像透明度（真透明）
/// </summary>
/// <param name="img">所要转变的图像</param>
/// <param name="alpha">透明度，最大为 1，最小为 0</param>
/// <returns>改变后的图像</returns>
public static Bitmap VitrificationImage(Image img, float alpha)
{
    Bitmap _newImg = new Bitmap(img.Width, img.Height);

    using (Graphics _g = Graphics.FromImage(_newImg))
    {
        using (ImageAttributes _imageAttrs = new ImageAttributes())
        {
            _imageAttrs.SetColorMatrix(new ColorMatrix(CreateAlphaMatrix(alpha)));

            _g.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height),
                        1, 1, img.Width, img.Height, GraphicsUnit.Pixel, _imageAttrs);
        }
    }

    return _newImg;
}

//// <summary>
/// 创建用于改变图像透明度的颜色矩阵
/// </summary>
/// <param name="alpha">所要设置的透明度</param>
/// <returns>返回用于图像转换的颜色矩阵</returns>
private static float[][] CreateAlphaMatrix(float alpha)
{
    if (alpha > 1)
        alpha = 1;

    if (alpha < 0)
        alpha = 0;

    float[][] _matrix =
      {
                   new float[] {1, 0, 0, 0, 0},
                   new float[] {0, 1, 0, 0, 0},
                   new float[] {0, 0, 1, 0, 0},
                   new float[] {0, 0, 0, alpha, 0},
                   new float[] {0, 0, 0, 0, 1}
    };

    return _matrix;
}
```

我是没有眼泪的人，为了保护自己，今后我可以对任何人残忍！

GDI+命中测试的效率[r]

问题的提出：

摄像头的分辨率从 30 万一直到 700 万，成本不停地降低，性能不停地提高，在项目开发过程中碰到这样的问题，为了将提高精度，使用 700 万的摄像头来进行微距拍摄，因此带来最直接的问题是相同的颗粒需要处理的像素多了很多多多多。

在低分辨率图像中，像素少，整体的处理时间也不多。但是一旦使用高分辨率的图像，为便于用户全局观察，图像显示时缩小了，对用户来说并未感觉到选择区域的变化，可是需要处理的像素增加了，处理时间则呈平方倍数增加，就会明显地察觉到停顿。为此，我们需要找到原来的程序的速度瓶颈，进行优化。

问题的分析：

在不改变算法逻辑的前提下，我们对程序的各部分的时间耗费进行了分析和测试。最后发现程序中有两中功能的代码应该还可以改进，减少所需的时间。

一个是对图像像素值的读写，也就是通过坐标(x,y)对像素的寻址定位，一般的方法就是通过

```
p = y*width + x;
```

来换算像素值在线形内存地址中的位置，这样的做法简单且直观，容易理解，但是每访问一个就需要进行 1 次乘法和 1 次加法。如果单个像素使用多个字节存储（一般 24 位 3 个字节），则还需要额外的乘法和加法。因此我们希望可以改变地址计算的方法，按照像素的存储顺序进行扫描，避免不必要的乘法运算。

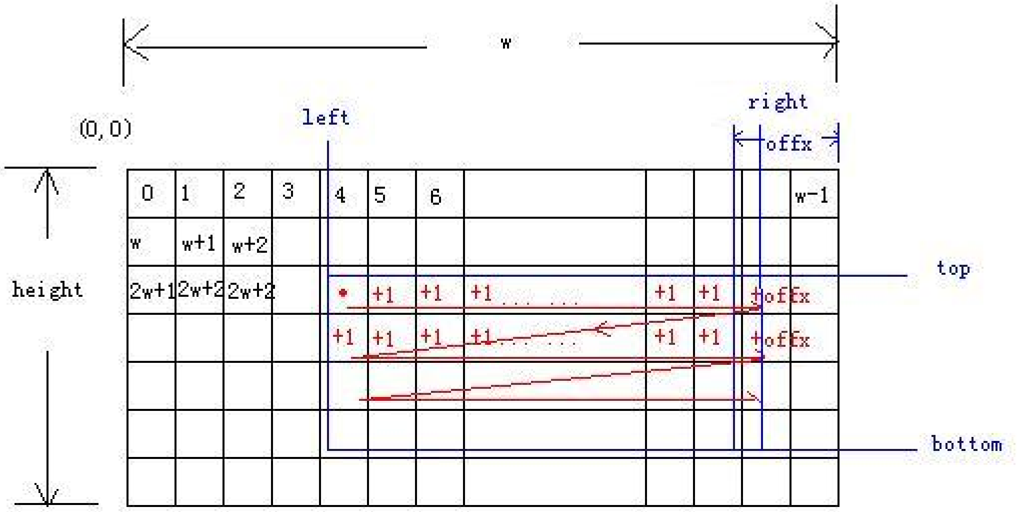
另外一方面，就是判断坐标为(x,y)的像素是否在选择的区域内。因为程序所使用的检测算法对杂质的影响非常敏感，已经定好的解决方案是让用户自己圈定一个区域进行检测，因此在对图像进行处理时就需要判断点是否在区域内，是的话则进行统计并处理，否则放弃。原来的程序是基于 GDI 的，使用 CRgn 类创建多边形区域，使用 CRgn::PtInRegion(x,y)函数判断命中。测试发现，对矩形区域（不判断）和对一个多边形区域进行处理，在代码中就是将判断命中的 if 语句注销掉和不去掉一行的差别，处理时间相差甚多。

我们使用了 GDI+中的 Region 对象及 Region::IsVisible(x,y)函数替代，处理时间有所缩短，但是，因为处理的像素实在太多（选择图像的一半就有 350 万个像素），判断和不判断区域命中的时间差距仍然比较多。因此开始怀疑 GDI+的 Region::IsVisible(x,y)函数的判断算法的效率有问题，于是决定自己实现这个判断的功能。实现的基本思想就是用空间来换时间。

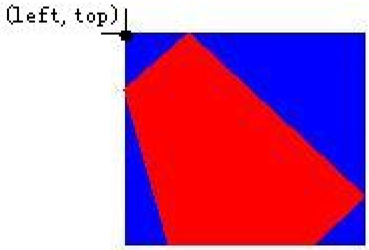
问题的解决方案：

首先是扫描线问题，这个比较简单，按行扫描则符合像素的存储顺序。图中兰色代表扫描的区域，红色是扫描的顺序也就是地址的递增操作，需要注意的是每行扫描结束后需要增加 offx，以跳到下一行行首地址，值应该为 w - right ，一般 right 都为选择区域内的最右边像素 x 坐标的

再往右一个像素的坐标。



命中测试问题用 FastHitTest 类来解决，初始化时创建一个选定区域的外接矩形大小的位图，并记录矩形的左上坐标，然后将位图先全部涂成指定的背景色，然后在选定的区域中涂上用指定的前景色。判定的时候，只要检查这个位图对应点是否前景色，是则在区域内，否则在区域外。在外接矩形外的点不需要判断了，直接否定。在 FastHitTest，记录了三种区域，如图中白色区域为外接矩形外的区域，通过左上坐标和位图大小来确定；兰色为指定的背景色，为选择区域的外接矩形；红色为指定的前景色，为所选择的区域。空间耗费就为位图所占用的空间，单次区域内判断的时间耗费就仅为查询位图内指定坐标的点的颜色。



项目中实际使用过的代码，集合放到IPLab程序中了，是VC6 下直接链接gdiplus.lib库，可以直接编译运行，在这里下载。程序运行后打开工程文件夹下的 700 万（3072*2304）像素的图片，显示默认缩放为原图的 15%，然后应该按住鼠标选定一块区域，快捷按钮 1-5 的功能分别为：1 对全图进行反色处理；2、3、4 都对选定的局部区域进行反色处理，但是判断命中的方式分别为CRgn方式、Region方式和FastHitTest方式，处理结束后分别给出处理时间；5 将选择区域（记录了鼠标的划过的路径），用DrawPath和Fill两种方式画出来以比较路径和区域的差异。

贴出关键代码的全文。

图像反色算法函数：

```
void CIPLabDoc::IPFuncInvInRegion_F(GraphicsPath* pWorkingRegion)
{
    if (!pPicture || !pWorkingRegion) {
        return;
    }

    BitmapData bitmapData;
    Rect imageRect(0, 0, pPicture->GetWidth(), pPicture->GetHeight());//900,900);
    Rect rect;

    pWorkingRegion->GetBounds(&rect);
    rect.Intersect(imageRect);

    BYTE inv[256*3];
    int i;
    for(i=0;i<256;i++)
        {
        inv[i+256*0]=255-i;    //b
        inv[i+256*1]=255-i;    //g
        inv[i+256*2]=255-i;    //r
    }

    Status status = pPicture->LockBits(
        &rect,
        ImageLockModeRead | ImageLockModeWrite,
        PixelFormat24bppRGB,
        &bitmapData);

```

```
  if (status != Ok ) {
   return;
}
unsigned char* pixels = (unsigned char*)bitmapData.Scan0;
int x,y;
int offx = bitmapData.Stride - bitmapData.Width*3;
BYTE* pHistogramProjection = inv;

FastHitTest fHitTest(pWorkingRegion);

for(y=rect.GetTop();y<rect.GetBottom();y++)
  {
   for(x=rect.GetLeft();x<rect.GetRight();x++)
     {
       if (fHitTest.IsVisible(x,y)) {

        //b
        *pixels = pHistogramProjection[*pixels];
        pixels++;
        pHistogramProjection += 256;

        //g
        *pixels = pHistogramProjection[*pixels];
        pixels++;
        pHistogramProjection += 256;

        //r
        *pixels = pHistogramProjection[*pixels];
        pixels++;
        pHistogramProjection -= 256*2;
```

```
            }
            else
               {
               pixels += 3;
            }
        }
        pixels += offx;
    }
    pPicture->UnlockBits(&bitmapData);

    UpdateAllViews(NULL);
}
```

FastHitTest 的构造函数：

```
void FastHitTest::init(const GraphicsPath* pPath,BOOL includeEdge)
{
    pPath->GetBounds(&m_Bounds);

    m_RegionBuffer = new Bitmap(m_Bounds.Width,m_Bounds.Height,PixelFormat32bppRGB);

    Graphics g(m_RegionBuffer);

    SolidBrush backgroundBrush(m_BackgroundColor);
    g.FillRectangle(&backgroundBrush,0,0,m_Bounds.Width,m_Bounds.Height);

    SolidBrush foregroundBrush(m_ForegroundColor);
    Pen foregroundPen(&foregroundBrush);

    GraphicsPath *path = pPath->Clone();
    Matrix mat;

    mat.Translate((float)(-1.0 * m_Bounds.GetLeft()), (float)(-1 * m_Bounds.GetTop()));
    path->Transform(&mat);

    g.FillPath(&foregroundBrush,path);

    //confirm required
    if (includeEdge) {
        g.DrawPath(&foregroundPen,path);
    }

    delete path;
}
```

判断命中的函数：

```
BOOL FastHitTest::IsVisible(int x, int y)
{
    if( !m_RegionBuffer )
    {
        return FALSE;
    }

    if ( x < m_Bounds.GetLeft() || x >= m_Bounds.GetRight()
        || y < m_Bounds.GetTop() || y >= m_Bounds.GetBottom() )
    {
        return FALSE;
    }

    UINT *p = (UINT*)m_Data.Scan0 + m_Data.Stride * (y-m_Bounds.GetTop() ) / 4 + (x - m_Bounds.GetLeft());
    if ( *p == m_ForegroundColor.GetValue() ) {
        return TRUE;
    }

    return FALSE;
}
```

总结：
当图特别大的时候，处理区域特别大的时候，空间换时间的方式可以将处理时间下降到可以容忍的程度。空间耗费嘛，及时释放的话还是能够忍受的。
如果需要进行多次区域命中判断，整体代价（时间、空间及复杂度等因素）太高时，而且算法跟区域无关时，可以选择对全图进行处理（不判断命中），然后再对结果进行剪切，这样就只需要进行一次命中判断。

除此之外，自己构造的类可以解决一些边缘的问题。无论在 GDI 或者 GDI+中，Rect 的边缘（Draw 出来的）和内部区域（Fill 出来的）都有 1 个像素的差别，对于多边形区域就更难发现其中的对应关系，如果边缘跟踪出来的颗粒利用区域命中的方法来计算面积，就会和预期的有所差距，特别在颗粒呈细长条状时特别明显。程序中的 HitTest 按钮将这个差别画出来了，不是很容易观察到，可以剪屏后到画笔中用大尺寸查看。因此 FastHitTest 的构造参数中有个 BOOL 变量，来决定是否包括边缘。
原来还希望加入边缘宽度，并以边缘框架的形式来测试命中，这样可以在判断单次点击是否点击命中这个路径时，决定敏感区域的宽窄。但是这个问题并不需要单独出一个类花这么多的空间来完成，直接使用 GDI+提供的方法已经足够了。因此也就没有进一步的扩展。

问题还很多，有兴趣或者有相关经验的朋友，不妨一起讨论。兴趣最重要，但是要肯花时间，真诚，不劳而获是可耻的。:-|

路漫漫其修远兮 吾将上下而求索

GDIDrawing3——GDI+绘图（三）

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace GDIDrawing3
{
    /// <summary>
    /// GDIDrawing3——GDI+绘图。
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        /// <summary>
        /// 必需的设计器变量。
        /// </summary>
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            // Windows 窗体设计器支持所必需的
            InitializeComponent();
            // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
        }

        /// <summary>
        /// 清理所有正在使用的资源。
/// </summary>
protected override void Dispose( bool disposing )
 {
  if( disposing )
   {
    if (components != null)
     {
      components.Dispose();
    }
  }
  base.Dispose( disposing );
}

 Windows Form Designer generated code

 /// <summary>
/// 应用程序的主入口点。
/// </summary>
[STAThread]
static void Main()
 {
  Application.Run(new Form1());
}
// 处理窗体显示事件。
private void Form1_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
 {
  Graphics g = e.Graphics;
  SolidBrush redBrush = new SolidBrush(Color.Red);
  g.FillEllipse(redBrush, 0, 0, 100, 60);
```

```
            g.FillPie(redBrush, 100, 0, 100, 70, 30, 300);
            g.FillRectangle(redBrush, 200, 10, 100, 50);
            Point[] points =
                  {
                    new Point(0, 100),
                    new Point(20, 120),
                    new Point(50, 100),
                    new Point(60, 200),
                    new Point(30, 220),
                    new Point(180, 200),
                    new Point(20, 110),
                    new Point(0, 220)
                };
            g.FillPolygon(redBrush, points);
            Rectangle fillRect = new Rectangle(200, 100, 200, 200);
            Region fillRegion = new Region(fillRect);
            g.FillRegion(redBrush, fillRegion);
        }
    }
}
```

C#中用GDI+生成饼状图和柱状图

```
using System;
using System.IO;//用于文件存取
using System.Data;//用于数据访问
using System.Drawing;//提供画 GDI+图形的基本功能
using System.Drawing.Text;//提供画 GDI+图形的高级功能
using System.Drawing.Drawing2D;//提供画高级二维，矢量图形功能
using System.Drawing.Imaging;//提供画 GDI+图形的高级功能
namespace lc.laili.Web.Code
{
 /// <summary>
 /// MyImage 的摘要说明。
 /// </summary>
 public class MyImage
 {
  public MyImage()
  {
   //
   // TODO: 在此处添加构造函数逻辑
   //

  }


  /// <summary>
  ///  数据源是 PieChartValue 对象的饼状图
  /// </summary>
  /// <param name="title">饼状图大标题</param>
  /// <param name="subTitle">饼状图小标题</param>
  /// <param name="width">图宽</param>
  /// <param name="height">图高</param>
  /// <param name="Mydata">PieChartValue 对象</param>
  /// <returns>饼状图路径</returns>
  public string Render(string title, string subTitle, int width, int height, PieChartValue[] Mydata)
  {
   const int SIDE_LENGTH = 400;
   const int PIE_DIAMETER = 200;
   //DataTable dt = DataTable;

   //通过输入参数，取得饼图中的总基数
   float sumData = 0;
   for (int i = 0; i < Mydata.Length; i++)
   {
    sumData += Convert.ToSingle(Mydata[i].MValue);
   }
   //产生一个 image 对象，并由此产生一个 Graphics 对象
   Bitmap bm = new Bitmap(width, height);
   Graphics g = Graphics.FromImage(bm);
   //设置对象 g 的属性
   g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height)) / SIDE_LENGTH);
```

```
 g.SmoothingMode = SmoothingMode.Default;
 g.TextRenderingHint = TextRenderingHint.AntiAlias;

 //画布和边的设定
 g.Clear(Color.White);
 g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
 //画饼图标题
 g.DrawString(title, new Font("Tahoma", 14), Brushes.Black, new PointF(5, 5));
 //画饼图的图例
 g.DrawString(subTitle, new Font("Tahoma", 12), Brushes.Black, new PointF(7, 35));
 //画饼图
 float curAngle = 0;
 float totalAngle = 0;
 for (int j = 0; j < Mydata.Length; j++)
 {
  curAngle = Convert.ToSingle(Mydata[j].MValue) / sumData * 360;

  g.FillPie(new SolidBrush(ChartUtil.GetChartItemColor(j)), 100, 65, PIE_DIAMETER,
PIE_DIAMETER, totalAngle, curAngle);
  g.DrawPie(Pens.Black, 100, 65, PIE_DIAMETER, PIE_DIAMETER, totalAngle, curAngle);
  totalAngle += curAngle;
 }
 //画图例框及其文字
 g.DrawRectangle(Pens.Black, 200, 300, 199, 99);
 g.DrawString("图表说明", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new
PointF(200, 300));

 //画图例各项
 PointF boxOrigin = new PointF(210, 330);
```

```
  PointF textOrigin = new PointF(235, 326);
  float percent = 0;
  for (int k = 0; k < Mydata.Length; k++)
  {
   g.FillRectangle(new SolidBrush(ChartUtil.GetChartItemColor(k)), boxOrigin.X,
boxOrigin.Y, 20, 10);
   g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 20, 10);
   percent = Convert.ToSingle(Mydata[k].MValue) / sumData * 100;
   g.DrawString(Mydata[k].MValue.ToString() + " - " + Mydata[k].Name.ToString() + " (" +
percent.ToString("0") + "%)", new Font("Tahoma", 10), Brushes.Black, textOrigin);
   boxOrigin.Y += 15;
   textOrigin.Y += 15;
  }
  //回收资源
  g.Dispose();
//此处存储路径需要自己改
  System.Drawing.Image MyImage=(System.Drawing.Image)bm;
  string TruePath= System.Web.HttpContext.Current.Server.MapPath("~/images");
  string
TrueName=DateTime.Now.ToString("yyyyMMddhhmmss")+DateTime.Now.Millisecond;
  MyImage.Save(TruePath+"\\"+TrueName+".jpg",
System.Drawing.Imaging.ImageFormat.Gif);
  return "~/images/"+TrueName+".jpg";
 }
```

```
  /// <summary>
  ///  数据源是 BarChartValue 对象的柱状图
  /// </summary>
  /// <param name="title">大标题</param>
  /// <param name="subTitle">小标题</param>
  /// <param name="width">图宽</param>
  /// <param name="height">图高</param>
  /// <param name="Mydata">BarChartValue 对象</param>
  /// <returns>柱状图路径</returns>
  public string Render(string title, string subTitle, int width, int height, BarChartValue[] Mydata)
  {
   const int SIDE_LENGTH = 400;
   const int CHART_TOP = 75;
   const int CHART_HEIGHT = 200;
   const int CHART_LEFT = 50;
   const int CHART_WIDTH = 300;

   //计算最高的点
   float highPoint = 0;
   for (int i=0; i < Mydata.Length;i++ )
   {
    if (highPoint < Convert.ToSingle(Mydata[i].MValue))
    {
     highPoint = Convert.ToSingle(Mydata[i].MValue);
    }
   }
   //建立一个 Graphics 对象实例
   Bitmap bm = new Bitmap(width, height);

    Graphics g = Graphics.FromImage(bm);
    //设置条图图形和文字属性
    g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height))
/ SIDE_LENGTH);
    g.SmoothingMode = SmoothingMode.Default;
    g.TextRenderingHint = TextRenderingHint.AntiAlias;

    //设定画布和边
    g.Clear(Color.White);
    g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
    //画大标题
    g.DrawString(title, new Font("Tahoma", 14), Brushes.Black, new PointF(5, 5));
    //画小标题
    g.DrawString(subTitle, new Font("Tahoma", 12), Brushes.Black, new PointF(7, 35));
    //画条形图
    float barWidth = CHART_WIDTH / (Mydata.Length * 2);
    PointF barOrigin = new PointF(CHART_LEFT + (barWidth / 2), 0);
    float barHeight = Mydata.Length;
    for (int i = 0; i < Mydata.Length; i++)
    {
     barHeight = Convert.ToSingle(Mydata[i].MValue) * 200 / highPoint * 1;
     barOrigin.Y = CHART_TOP + CHART_HEIGHT - barHeight;
     g.FillRectangle(new SolidBrush(ChartUtil.GetChartItemColor(i)), barOrigin.X,
barOrigin.Y, barWidth, barHeight);
```

```
     g.DrawString(Mydata[i].Name, new Font("Tahoma", 3), Brushes.Black, new PointF(barOrigin.X-3, 277));


     barOrigin.X = barOrigin.X + (barWidth * 2);
    }
    //设置边
    g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT));
    g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT), new Point(CHART_LEFT + CHART_WIDTH, CHART_TOP + CHART_HEIGHT));
    //画图例框和文字


    //g.DrawRectangle(Pens.Black, 200, 300, 199, 99);
    //g.DrawString("图表说明", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new PointF(200, 300));

    g.DrawRectangle(new Pen(Color.Black, 1), 10, 290, 299, 110);
    g.DrawString("图表说明", new Font("Tahoma", 10, FontStyle.Bold), Brushes.Black, new PointF(11, 290));

    //画图例
    PointF boxOrigin = new PointF(10, 310);
    PointF textOrigin = new PointF(35, 306);
    for (int i = 0; i < Mydata.Length; i++)
    {
     g.FillRectangle(new SolidBrush(ChartUtil.GetChartItemColor(i)), boxOrigin.X,
boxOrigin.Y, 20, 10);
     g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 2, 1);
     g.DrawString(Mydata[i].Name.ToString() + " - " + Mydata[i].MValue.ToString(), new
Font("Tahoma", 10), Brushes.Black, textOrigin);
     if(i<5)
     {
      boxOrigin.Y += 15;
      textOrigin.Y += 15;
     }
     else if(i==5)
     {
      boxOrigin.X = 150;
      boxOrigin.Y = 310;
      textOrigin.X = 176;
      textOrigin.Y = 306;

     }
     else
     {
      boxOrigin.Y += 15;
      textOrigin.Y += 15;
     }

    }
    //输出图形
    g.Dispose();
//此处存储路径需要自己改
```

```
  System.Drawing.Image MyImage=(System.Drawing.Image)bm;

  string TruePath= System.Web.HttpContext.Current.Server.MapPath("~/images");

  string
TrueName=DateTime.Now.ToString("yyyyMMddhhmmss")+DateTime.Now.Millisecond;

  MyImage.Save(TruePath+"\\"+TrueName+".jpg",
System.Drawing.Imaging.ImageFormat.Gif);

  return "~/images/"+TrueName+".jpg";

 }

}

/// <summary>

///  选择颜色

/// </summary>

public class ChartUtil

{

 public ChartUtil()

 {

 }

 public static Color GetChartItemColor(int itemIndex)

 {

  Color selectedColor;

  switch (itemIndex)

  {

   case 0:

    selectedColor = Color.AliceBlue;

    break;

   case 1:

    selectedColor = Color.Red;
    break;

   case 2:

    selectedColor = Color.Yellow;

    break;

   case 3:

    selectedColor = Color.AntiqueWhite;

    break;

   case 4:

    selectedColor = Color.Aqua;

    break;

   case 5:

    selectedColor = Color.Aquamarine;

    break;

   case 6:

    selectedColor = Color.Azure;

    break;

   case 7:

    selectedColor = Color.Beige;

    break;

   case 8:

    selectedColor = Color.Black;

    break;

   case 9:

    selectedColor = Color.Brown;

    break;

   case 10:

    selectedColor = Color.Coral;

    break;
```

```
   case 11:
    selectedColor = Color.DarkCyan;
    break;
   case 12:
    selectedColor = Color.DarkOrange;
    break;
   default:
    selectedColor = Color.DarkViolet;
    break;
  }
  return selectedColor;
 }
}

/// <summary>
/// 饼状图的数据对象
/// </summary>
public class PieChartValue
{
 public PieChartValue()
 {

 }
 public PieChartValue(string MyName, int Myvalue)
 {
  Name = MyName;
  MValue = Myvalue;
 }
 private string name;
 public string Name
 {
  get
  {
   return name;
  }
  set
  {
   name = value;
  }
 }

 private int mvalue;
 public int MValue
 {
  get
  {
   return mvalue;
  }
  set
  {
   mvalue = value;
  }
 }
}

/// <summary>
///  柱状图的数据对象
/// </summary>
```

```
public class BarChartValue

{

  public BarChartValue()

  {


  }

  public BarChartValue(string MyName, int Myvalue)

  {

   Name = MyName;

   MValue = Myvalue;

  }

  private string name;

  public string Name

  {

   get

   {

    return name;

   }

   set

   {

    name = value;

   }

  }


  private int mvalue;

  public int MValue

  {

   get

   {
    return mvalue;
   }
   set
   {
    mvalue = value;
   }
  }
 }
}
```

用法

```
   #region  生成柱状图

   MyImage MyMyImage = new MyImage();
   BarChartValue[] BarChartValue ={
         new BarChartValue("第1个月",5),
         new BarChartValue("第2个月",2),
         new BarChartValue("第3个月",3),
         new BarChartValue("第4个月",4),
         new BarChartValue("第5个月",5),
         new BarChartValue("第6个月",6),
         new BarChartValue("第7个月",7),
         new BarChartValue("第8个月",8),
         new BarChartValue("第9个月",9)
         };
   Image1.ImageUrl = MyMyImage.Render("全年统计", "98 年", 1000, 1000, BarChartValue);
   #endregion



   #region  生成饼状图
   PieChartValue[] PieChartValue ={
           new PieChartValue("第一个仓库",1),
           new PieChartValue("第一个仓库",2)
         };
   Image2.ImageUrl = MyMyImage.Render("仓库统计饼状图", "小仓库", 500, 500,
PieChartValue);
   #endregion
```

[GDI+]如何制作出高质量的缩略图

如何制作出高质量的缩略图是个关键的因素，最近项目中遇到了类似需要解决的问题。
一般情况下，我们制作成的缩略图都会保存为占用空间比较小的Jpeg类型，在使用GetThumbnail方法制作成的缩略图质量感觉不理想，
如何才能保证在压缩比例最优化的情况下产生高质量的缩略图呢，经过查阅相关资料，
发现在Graphics 对象的 InterpolationMode 属性中可以产生不同质量模式的缩放图，看到这里了，不再是缩略图，而是缩放图，就是说放大的时候也可以使用。
Graphics 对象的 InterpolationMode 属性枚举定义了几种模式，列表如下：
NearestNeighbor
Bilinear
HighQualityBilinear
Bicubic
HighQualityBicubic
从名字上就可以识别NearestNeighbor 是质量最差的模式，HighQualityBicubic 是质量最好的模式了，我们借此属性看看生成的图片怎么样吧，下段代码摘自网络，大家可以把下面的函数拿去使用，这里采用了HighQualityBilinear 。

代码引用地址：http://www.bobpowell.net/highqualitythumb.htm

```
Public Function GenerateThumbnail(original As Image, percentage As Integer) As Image

    If percentage < 1 Then
     Throw New Exception("Thumbnail size must be aat least 1% of the original size")
    End If

    Dim tn As New Bitmap(CInt(original.Width * 0.01F * percentage), _

                      CInt(original.Height * 0.01F * percentage))

    Dim g As Graphics = Graphics.FromImage(tn)

    g.InterpolationMode = InterpolationMode.HighQualityBilinear

    g.DrawImage(original, New Rectangle(0, 0, tn.Width, tn.Height), _

            0, 0, original.Width, original.Height, GraphicsUnit.Pixel)

    g.Dispose()
    Return CType(tn, Image)
End Function
```

C#实现捕获当前屏幕截图(转) - xugang - 博客园

xugang 从 2007 年开始，记录一个走入社会的.NET 程序员的成长

C#实现捕获当前屏幕截图(转)

编程思路（API 编程)：

先调用 GetForegroundWindow 获取当前活动程序窗口句柄,然后调用 GetWindowDC 获取窗口的设备句柄（或 GetDC 函数)，调用

BitBlt 位图传输函数将位图拷贝到兼容的设备场景中（拷贝时可以指定位置和大小)，最后保存位图文件。

以下源代码内容转自 CSDN 论坛。

要想完成这个功能，首先要了解一下在 C＃中如何调用 API（应用程序接口）函数。虽然在.Net 框架中已经提供了许多类库，这些类库的功能也十分强大，但对于一些 Windows 底层编程来说，还是要通过调用这些 API 函数才可以实现。所有 API 都在"Kernel"、"User

"和"GDI"三个库中得以运行：其中"Kernel"，他的库名为 "KERNEL32.DLL"，

他主要用于产生与操作系统之间的关联,譬如:程序加载,上下文选择,文件输入输出,内存管理等等。"User "这个类库在 Win32 中名叫 "USER32.DLL"。

它允许管理全部的用户接口。譬如：窗口 、菜单 、对话框

、图标等等。"GDI"(图象设备接口)，它在 Win32 中的库名为："GDI32.dll"，它是图形输出库。使用 GDI

Windows"画"出窗口、菜单以及对话框等；它能创建图形输出；它也能保存图形文件。由于本文所涉及到是图象问题，所有调用的类库是"GDI32.dll"。在本文程序中我们使用的 API 函数是"BitBlt"，这个函数对于广大程序员来说，一定不感觉到陌生，因为在图象处理方面他的用途是相对广的，在用其他程序语言编程中，时常也要和他打交道。在.Net

FrameWork

SDK 中有一个名字空间"System.Runtime.InteropServices"，此名字空间提供了一系列的类来访问 COM 对象，和调用本地的 API 函数。下面是在 C＃中声明此函数：

```
[ System.Runtime.InteropServices.DllImportAttribute ( "gdi32.dll" ) ]
private static extern bool BitBlt (
IntPtr hdcDest , // 目标 DC 的句柄
int nXDest ,
int nYDest ,
int nWidth ,
int nHeight ,
IntPtr hdcSrc , // 源 DC 的句柄
int nXSrc ,
int nYSrc ,
System.Int32 dwRop // 光栅的处理数值
) ;
```

通过上面这个声明，就可以在下面的代码中使用此函数了。

下面是用 C＃做屏幕捕获程序的具体实现步骤：

（1）.首先要获得当前屏幕的 graphic 对象，通过以下代码可以实现：

```
Graphics g1 = this.CreateGraphics ( ) ;
```

（2）.创建一个 Bitmap 对象，并且这个 Bitmap 对象的大小是当前屏幕：

首先要获得当前屏幕的大小，通过名字空间"System.Windows.Forms"中的"Screen"类的GetWorkingArea（）方法，可以实现。下面是得到当前屏幕的长（Height）和宽（Width）：

```
Rectangle rect = new Rectangle ( ) ;
rect = Screen.GetWorkingArea ( this ) ;
"屏幕宽"＝ rect.Width ;
"屏幕长"＝ rect.Height ;
```

至此就可以得到我们想要的 Bitmap 了，通过下列语句可以实现：

```
Image MyImage = new Bitmap ( rect.Width , rect.Height , g1 ) ;
//创建以屏幕大小为标准的位图
```

（3）.获得当前屏幕和此 Bitmap 对象的 DC，这可以通过下列语句实现：

```
//得到屏幕的 DC
IntPtr dc1 = g1.GetHdc ( ) ;
//得到 Bitmap 的 DC
IntPtr dc2 = g2.GetHdc ( ) ;
```

（4）.调用 API 函数，把当前屏幕拷贝到创建的 Bitmap 中：

```
BitBlt ( dc2 , 0 , 0 , rect.Width , rect.Height , dc1 , 0 , 0 , 13369376 ) ;
```

（5）.释放当前屏幕和此 Bitmap 对象的 DC，通过下面代码可以实现：

```
//释放掉屏幕的 DC
g1.ReleaseHdc ( dc1 ) ;
//释放掉 Bitmap 的 DC
g2.ReleaseHdc ( dc2 ) ;
```

（6）.保存 Bitmap 对象，形成 jpg 图片：

```
MyImage.Save ( @"c:\Capture.jpg" , ImageFormat.Jpeg );
```

当然你也可以根据自己的需要，把屏幕以其他图片的格式来保存，如果你想把图片保存为位图文件，可以把"ImageFormat.Jpeg"改换成"ImageFormat.Bmp"；想把图片保存为 Gif 文件，就把"ImageFormat.Jpeg"改换成"ImageFormat.Gif"。你可以保存的文件类型大概有十多种，这里就不一一介绍了，当然你也要相应改变保存文件的后缀。

用 C＃来捕获屏幕的源程序代码（Capture.cs）：

```
using System ;
using System.Drawing ;
using System.Collections ;
using System.ComponentModel ;
using System.Windows.Forms ;
using System.Data ;
using System.Drawing.Imaging ;
public class Form1 : Form
```

```
{
    private Button button1 ;
    private System.ComponentModel.Container components = null ;

    public Form1 ( )
    {
        //初始化窗体中的各个组件
        InitializeComponent ( ) ;
    }
    // 清除程序中使用过的资源
    protected override void Dispose ( bool disposing )
    {
        if ( disposing )
        {
            if ( components != null )
            {
                components.Dispose ( ) ;
            }
        }
        base.Dispose ( disposing ) ;
    }
    private void InitializeComponent ( )
    {
        button1 = new Button ( );
        SuspendLayout ( ) ;
        button1.Location = new System.Drawing.Point ( 64 , 40 ) ;
        button1.Name = "button1" ;
        button1.Size = new System.Drawing.Size ( 80 , 32 ) ;
        button1.TabIndex = 0 ;
        button1.Text = "捕获" ;
        button1.Click += new System.EventHandler ( button1_Click ) ;

        AutoScaleBaseSize = new System.Drawing.Size ( 6 , 14 ) ;
        ClientSize = new System.Drawing.Size ( 216 , 125 ) ;
        Controls.Add ( button1 ) ;
        MaximizeBox = false ;
        MinimizeBox = false ;
        Name = "Form1" ;
        Text = "C#捕获当前屏幕！" ;
        ResumeLayout ( false ) ;

    }
    //声明一个 API 函数
    [ System.Runtime.InteropServices.DllImportAttribute ( "gdi32.dll" ) ]
    private static extern bool BitBlt (
        IntPtr hdcDest , // 目标 DC 的句柄
        int nXDest ,
        int nYDest ,
        int nWidth ,
        int nHeight ,
        IntPtr hdcSrc , // 源 DC 的句柄
        int nXSrc ,
        int nYSrc ,
        System.Int32 dwRop // 光栅的处理数值
        ) ;

    static void Main ( )
    {
        Application.Run ( new Form1 ( ) ) ;
    }
    private void button1_Click ( object sender , System.EventArgs e )
    {
        //获得当前屏幕的大小
        Rectangle rect = new Rectangle ( ) ;
        rect = Screen.GetWorkingArea ( this ) ;
        //创建一个以当前屏幕为模板的图象
        Graphics g1 = this.CreateGraphics ( ) ;
        //创建以屏幕大小为标准的位图
        Image MyImage = new Bitmap ( rect.Width , rect.Height , g1 ) ;
        Graphics g2 = Graphics.FromImage ( MyImage ) ;
        //得到屏幕的 DC
        IntPtr dc1 = g1.GetHdc ( ) ;
        //得到 Bitmap 的 DC
        IntPtr dc2 = g2.GetHdc ( ) ;
        //调用此 API 函数，实现屏幕捕获
        BitBlt ( dc2 , 0 , 0 , rect.Width , rect.Height , dc1 , 0 , 0 , 13369376
) ;
        //释放掉屏幕的 DC
        g1.ReleaseHdc ( dc1 ) ;
        //释放掉 Bitmap 的 DC
        g2.ReleaseHdc ( dc2 ) ;
        //以 JPG 文件格式来保存
        MyImage.Save ( @"c:\Capture.jpg" , ImageFormat.Jpeg );
        MessageBox.Show ( "当前屏幕已经保存为 C 盘的 capture.jpg 文件！" ) ;
    }
}
```

来源：http://thewbb.spaces.live.com/Blog/cns!56C956FF8A430850!198.entry

posted on 2007-12-19 17:10 钢钢 阅读(168) 评论(2) 编辑 收藏 所属分类: C#

评论

#1 楼 [TrackBack] 2007-12-23 16:55 钢钢

由于工作需要，学习了 GDI

编程的一些知识。其中看到了一个比较好的 Demo，深入的了解后，却发现自己对如何用 C#调用非托管 DLL 函数也有了更好的理解，于是整理了一下，跟大家一起分享。其中引用了微软...

[引用提示]钢钢引用了该文章, 地址:

http://www.cnblogs.com/xugang/archive/2007/12/23/1011597.html 回复 引用 查看

#2 楼 [TrackBack] 2008-01-13 11:15 古晶

编程思路（API 编程）：先调用 GetForegroundWindow 获取当前活动程序窗口句柄，然后调用 GetWindowDC 获取窗口的设备句柄（或 GetDC 函数），调用 BitBlt...

[引用提示]古晶引用了该文章, 地址:

http://www.cnblogs.com/kingkoo/archive/2007/12/19/1036816.html 回复

C#中 Invalidate() 方法 - chinafine - 博客园

chinafine

Asp.net,Windows Mobile,Windows Server 醉心于.NET 博客园 首页 新随笔 新文章 联系 管理 订阅 随笔-

40 文章- 74 评论- 9 C#中 Invalidate() 方法

Control.Invalidate 方法

使控件的特定区域无效并向控件发送绘制消息。

重载列表

使控件的特定区域无效并向控件发送绘制消息。

受 .NET Framework 精简版的支持。

```
  [C#] public void Invalidate();
  [C++] public: void Invalidate();
```

使控件的特定区域无效并向控件发送绘制消息。还可以使分配给该控件的子控件无效。

```
  [C#] public void Invalidate(bool);
```

使控件的指定区域无效（将其添加到控件的更新区域，下次绘制操作时将重新绘制更新区域），并向控件发送绘制消息。

受 .NET Framework 精简版的支持。

```
  [C#] public void Invalidate(Rectangle);
```

使控件的指定区域无效（将其添加到控件的更新区域，下次绘制操作时将重新绘制更新区域），并向控件发送绘制消息。

```
  [C#] public void Invalidate(Region);
```

使控件的指定区域无效（将其添加到控件的更新区域，下次绘制操作时将重新绘制更新区域），并向控件发送绘制消息。还可以使分配给该控件的子控件无效。

```
  [C#] public void Invalidate(Rectangle, bool);
```

使控件的指定区域无效（将其添加到控件的更新区域，下次绘制操作时将重新绘制更新区域），并向控件发送绘制消息。还可以使分配给该控件的子控件无效。

```
  [C#] public void Invalidate(Region, bool);
```

示例

[Visual Basic, C#, C++] 下面的示例使用户能够将图像或图像文件拖到窗体上，并使它在放置点显示。每次绘制窗体时，都重写 OnPaint

方法以重新绘制图像；否则图像将保持到下一次重新绘制。DragEnter 事件处理方法决定拖到窗体中的数据的类型，并提供适当的反馈。如果 Image
可以从该数据中创建，则 DragDrop 事件处理方法就会在该窗体上显示此图像。因为 DragEventArgs.X 和 DragEventArgs.Y
值为屏幕坐标，所以示例使用 PointToClient 方法将它们转换成工作区坐标。

```
[C#]
private Image picture;
private Point pictureLocation;

public Form1()
{
   // Enable drag-and-drop operations and
   // add handlers for DragEnter and DragDrop.
   this.AllowDrop = true;
   this.DragDrop += new DragEventHandler(this.Form1_DragDrop);
   this.DragEnter += new DragEventHandler(this.Form1_DragEnter);
}

protected override void OnPaint(PaintEventArgs e)
{
   // If there is an image and it has a location,
   // paint it when the Form is repainted.
   base.OnPaint(e);
   if(this.picture != null && this.pictureLocation != Point.Empty)
   {
      e.Graphics.DrawImage(this.picture, this.pictureLocation);
   }
}

private void Form1_DragDrop(object sender, DragEventArgs e)
{
   // Handle FileDrop data.
   if(e.Data.GetDataPresent(DataFormats.FileDrop) )
   {
      // Assign the file names to a string array, in
      // case the user has selected multiple files.
      string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
      try
      {
         // Assign the first image to the picture variable.
         this.picture = Image.FromFile(files[0]);
         // Set the picture location equal to the drop point.
         this.pictureLocation = this.PointToClient(new Point(e.X, e.Y) );
      }
```

posted on 2007-11-20 16:33 chinaifne 阅读(70) 评论(0) 编辑 收藏 所属分类: C# 、GDI+编程

```
      catch(Exception ex)
      {
        MessageBox.Show(ex.Message);
        return;
      }
    }

    // Handle Bitmap data.
    if(e.Data.GetDataPresent(DataFormats.Bitmap) )
    {
      try
      {
        // Create an Image and assign it to the picture variable.
        this.picture = (Image)e.Data.GetData(DataFormats.Bitmap);
        // Set the picture location equal to the drop point.
        this.pictureLocation = this.PointToClient(new Point(e.X, e.Y) );
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.Message);
        return;
      }
    }
    // Force the form to be redrawn with the image.
    this.Invalidate();
  }

  private void Form1_DragEnter(object sender, DragEventArgs e)
  {
    // If the data is a file or a bitmap, display the copy cursor.
    if (e.Data.GetDataPresent(DataFormats.Bitmap) ||
      e.Data.GetDataPresent(DataFormats.FileDrop) )
    {
      e.Effect = DragDropEffects.Copy;
    }
    else
    {
      e.Effect = DragDropEffects.None;
    }
  }
```

posted on 2007-11-20 16:33 chinaifne 阅读(70) 评论(0) 编辑 收藏 所属分类: C# 、GDI+编程

gdi+实现多种统计图表(饼状,折线,柱状)支持负从标

```
using System;
using System.Web.UI;
using System.Data;
using System.Web.UI.WebControls;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Web;
using System.ComponentModel;
using System.Collections ;

namespace Tom.Control
{
        [ToolboxData("<{0}:Columniation runat=server></{0}:Columniation>")]
    public class Columniation: System.Web.UI.WebControls.WebControl
    {
        private const int _colorLimit = 12;   //颜色列表
       private Color[] _color =
            {
                Color.Chocolate,
                Color.YellowGreen,
                Color.Olive,
                Color.DarkKhaki,
                Color.Sienna,
                Color.PaleGoldenrod,
                Color.Peru,
                Color.Tan,
                Color.Khaki,
                Color.DarkGoldenrod,
                Color.Maroon,
                Color.OliveDrab
            };

        private DataTable items;//列表项名称和值

        private string text="数据";
        private string datastd="标准值";
        private string data="实时数据";


        int kds=10; //刻度数
        float kddw=100;   //没刻度大小
```

```
int zmheight=500; //真个图区高
int zmwidth=740;    //真个图区宽

int height=400;//呈现区高
int width=730;
int cxtop=30;//呈现区距顶距离
int cxleft=30;//呈现区左边距离

Color bzlink=Color.Black;//标准线颜色


int Chart_Flag=1;
Bitmap bm ;

int Displacement =0; //获取负坐标刻度数

[Bindable(true),
Category("Appearance"),
DefaultValue("")]
public string Text
{
        get
        {
              return text;
        }

        set
        {
              text = value;
        }
}

[Bindable(true),
Category("Appearance"),
DefaultValue("")]
public string DataStdName
{
        get
        {
              return datastd;
        }

        set
        {
                datastd = value;
        }
}

[Bindable(true),
Category("Appearance"),
DefaultValue("")]
public string DataName
{
        get
        {
                return data;
        }

        set
        {
                data = value;
        }
}

/**//// <summary>
/// 需要呈现的数据
/// </summary>
public DataTable Items
{
        set
        {items=value;}
}

/**//// <summary>
/// 需要显示的刻度量
/// </summary>
public int Kdcount
{
        set {kds=value;}
}

/**//// <summary>
/// 刻度大小
/// </summary>

public float Kddw
{set{kddw=value;}}
```

```
public int ChatStyle
{set{this.Chart_Flag=value;}}


/**//// <summary>
/// 将此控件呈现给指定的输出参数。
/// </summary>
/// <param text="output"> 要写出到的 HTML 代码 </param>
protected override void Render(HtmlTextWriter output)
{
    //if(dt==null)
    //{
    //      return "没有数据";
    //}
    //设计样式
    kd(items);
    output.Write(makeimage(items,"c:/"));


}

private string makeimage(DataTable dt,string imagefile)
{
    string url="";

    switch(Chart_Flag)
    {
        case 1:
        {

            this.Draw_X_Y_Bar(dt);
            url=this.Drar_Bar(dt);

            break;
        }
        case 2:
        {


            this.Draw_X_Y (dt);
            url=this.Drow_Lin(dt);
```

```
                break;
            }
            case 3:
            {

                url=this.Draw_Pie(dt);
                break;
            }

        }

            return    url;
}
/**//// <summary>
///  换算成实际值
/// </summary>
/// <param text="kd">提供的值</param>
/// <returns>返回换算后的实际值</returns>
private float bl(float kd)
{
        float bls=1;
        bls=(float)height/((float)kds*(float)kddw);
        return (float)kd*bls;
}


//通过数据计算刻度,和负坐标数#region //通过数据计算刻度,和负坐标数

private void    kd(DataTable dt)
{
        float       mintest=float.MaxValue;
        float       maxtest=float.MinValue;

            float       maxnegative=0f;



        for (int j=1;j<dt.Columns .Count ;j++)
        {
            for(int i=0;i<dt.Rows.Count;i++)
            {
                float test=System.Convert.ToSingle(dt.Rows[i][j]);
                //求最小值
                if(test<0    && test<maxnegative)
```

```
                    {
                         maxnegative=test;
                    }
                    //求最大值
                    if(test>maxtest)
                    {
                        maxtest=test;
                    }
                }
            }

            maxtest=maxtest+Math.Abs(maxnegative);

        //      this.Page .Response .Write (maxtest);

                 kddw=maxtest/kds;

            double exp = Convert.ToDouble(Math.Floor(Math.Log10(maxtest)));

            //      this.Page .Response .Write ("exp"+exp.ToString ()+"<br>");
            float tempMax = Convert.ToSingle(Math.Ceiling(maxtest / Math.Pow(10, exp)) *
Math.Pow(10, exp));

            // this.Page .Response .Write ("temp_Max"+exp.ToString ()+"<br>");


            kddw = tempMax / kds;
            double expTick = Convert.ToDouble(Math.Floor(Math.Log10(kddw)));
            kddw = Convert.ToSingle(Math.Ceiling(kddw / Math.Pow(10, expTick)) *
Math.Pow(10, expTick));
             this.Displacement =Math.Abs ((int) Math.Floor(maxnegative/kddw));
        //      this.Page .Response.Write (this.Displacement);


        }

        #endregion



   //
        绘制折线图#region    绘制折线图

         private string      Drow_Lin(DataTable dt)
```

```
            {


              //通过循环画出曲线图
              int x=-1;
              int bzy=-1;


              Graphics bp=Graphics.FromImage(bm);
              for (int j=1;j<dt.Columns.Count ;j++)
              {
                  for(int i=0;i<dt.Rows.Count;i++)
                  {

                      //标准刻度

                      float bzkd=bl(System.Convert.ToSingle(dt.Rows[i].ItemArray[j]));
                      //实际刻度
                      //      float sjkd=bl(System.Convert.ToSingle(dt.Rows[i].ItemArray[2]));

                      int bztop=cxtop+height-(int)bzkd-3-    (int) (bl(this.Displacement*kddw));
                      //填充标准柱(画刷，起点X，起点Y，宽，高）



                      bp.FillRectangle(new
SolidBrush(_color[j-1]),(i*40)+cxleft+20-3,bztop,6,6);
                      //      int sstop=zmheight-(int)sjkd-(zmheight-cxtop-height)-3;
                      //填充实时点(画刷，起点X，起点Y，宽，高）
                      //      bp.FillRectangle(new SolidBrush(ss),(i*30)+cxleft,sstop,6,6);

                      //绘制点到点连接线
                      if(x!=-1)
                      {
                          //绘制标准点连接线

                          bp.DrawLine(new   Pen(   _color[j-1]   ,1.6F),new   Point(x,bzy),new
Point((i*40)+cxleft+20,bztop));
                          //绘制实时点连接线
                          //          bp.DrawLine(new   Pen(ss,1.6F),new   Point(x+3,ssy+4),new
Point((i*30)+cxleft+10,sstop+4));
                      }
                      x=(i*40)+cxleft+20;
                      bzy=bztop;
```

```
                           //       ssy=sstop;
                         //      bp.DrawString(dt.Rows[i].ItemArray[0].ToString (),new Font("宋体
",9),new SolidBrush(Color.Black),new PointF((i*40)+cxleft+10,height+cxtop+1));


                 }


                     x=-1;
                   }
                 FileStream fs=new
FileStream(Page.Server.MapPath(Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")),FileMo
de.Create);
                 bm.Save(fs,ImageFormat.Jpeg);

                 bm.Dispose();
                 bp.Dispose();
                 fs.Close();
               return "<img src="+Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")+"
></img>";

             }

             #endregion



             画住状图#region  画住状图
  //画住状图


             private string  Drar_Bar(DataTable dt)
             { Graphics bp=Graphics.FromImage(bm);

                 int flag=0;
                    for(int i=0;i<dt.Rows.Count;i++)
                    {
                       for (int j=1;j<dt.Columns.Count ;j++)
                            {
                        float sjkd=bl(System.Convert.ToSingle(dt.Rows[i].ItemArray[j]));

                        float    top= cxtop+height-(int)sjkd-(int)(bl(this.Displacement*kddw));
                        if (sjkd<0)
                        {
                            top=top+sjkd;
                    sjkd=Math.Abs (sjkd);
                }

                bp.FillRectangle(new
SolidBrush(this._color[j]),(flag*40)+cxleft+20,top,20,sjkd);

            //          this.Page .Response .Write ("<br>"+   (flag*40+cxleft+20) );

                  flag++ ;


            }
        }

        FileStream                                                        fs=new
FileStream(Page.Server.MapPath(Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")),FileMo
de.Create);
        bm.Save(fs,ImageFormat.Jpeg);

        bm.Dispose();
        bp.Dispose();
        fs.Close();
        return      "<img   src="+Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")+"
></img>";


    }

    #endregion



    画   X,Y  轴  线和刻度#region   画   X,Y  轴  线和刻度

  private    void Draw_X_Y(DataTable dt )
    {

    //调整宽度
    width=45*dt.Rows.Count +30;

    zmwidth=width+cxleft+80;
    //创建一个画布

    bm=new Bitmap(zmwidth,zmheight);
```

```
        //在新建的画布上画一个图
        Graphics bp=Graphics.FromImage(bm);

        bp.Clear(Color.AliceBlue);
        //填充图表呈现区背景(画刷,起点 x，起点 Y，高，宽)
        bp.FillRectangle(new SolidBrush(Color.WhiteSmoke),cxleft,cxtop,width,height);
        //描绘呈现区边框
        bp.DrawRectangle(Pens.Black,cxleft,cxtop,width,height);

        //绘制图表名称
        bp.DrawString(text,new Font("宋体",9),new SolidBrush(Color.Black),new
PointF(zmwidth/2,10));


        //绘制图表说明
        bp.DrawRectangle(Pens.Black,cxleft+width+10,zmheight/2,60,15*dt.Columns.Count
-1);
        for (int j=0;j<dt.Columns .Count-1 ;j++)
        {
            bp.FillRectangle(new
SolidBrush(this._color[j]),cxleft+width+10+2,zmheight/2+6+16*j,8,8);
        //文字说明
            bp.DrawString(dt.Columns[j+1].ColumnName,new Font("宋体",9),new
SolidBrush(Color.Black),new PointF(cxleft+width+10+2+8,zmheight/2+4+j*12));
        }




        //通过循环绘制标准线
        for(int i=0;i<=kds;i++)
        {


           string Cur_Kd;
            if (i-this.Displacement!=0)
            {
                Cur_Kd=((i-this.Displacement)*(kddw)).ToString ("#,###.##");
            //       this.Page .Response .Write ("<br>abc"+Cur_Kd.ToString ());
            }
            else
            {
                 Cur_Kd="0";
            }
            bp.DrawString (Cur_Kd,new Font("宋体",9),new SolidBrush(Color.Black),new
PointF(2,zmheight-(bl(i*kddw)+(zmheight-cxtop-height)+4)));
            //填充标准线(画刷，起点X，起点Y，宽，高）
            int top=cxtop+height-(int)(bl(i*kddw));

            bp.DrawLine(new Pen(bzlink),new Point(cxleft-4,top),new
Point(cxleft+width,top));

        }

        for(int j=1;j<dt.Rows.Count+1;j++)
        {

            bp.DrawLine (new Pen (bzlink),new Point(cxleft+j*40, (int)
(cxtop+height-bl(this.Displacement*kddw) ) ),new Point (cxleft+j*40, (int)
(cxtop+height-bl(this.Displacement*kddw) )+4));

            bp.DrawString(dt.Rows[j-1].ItemArray[0].ToString (),new Font("宋体",9),new
SolidBrush(Color.Black),new PointF(((j-1)*40)+cxleft+10, (int)
(cxtop+height-bl(this.Displacement*kddw) ) +1));

        }
    }
        #endregion


        画 柱状 x,y 刻度#region 画 柱状 x,y 刻度

          private void Draw_X_Y_Bar (DataTable dt)
        {
              //调整宽度
              width=45*dt.Rows.Count * (dt.Columns .Count-1) +10;

              zmwidth=width+cxleft+80;
              //创建一个画布

              bm=new Bitmap(zmwidth,zmheight);
              //在新建的画布上画一个图
              Graphics bp=Graphics.FromImage(bm);

              bp.Clear(Color.AliceBlue);
              //填充图表呈现区背景(画刷,起点x，起点Y，高，宽)
              bp.FillRectangle(new
SolidBrush(Color.WhiteSmoke),cxleft,cxtop,width,height);
              //描绘呈现区边框
              bp.DrawRectangle(Pens.Black,cxleft,cxtop,width,height);

              //绘制图表名称
              bp.DrawString(text,new Font("宋体",9),new SolidBrush(Color.Black),new
PointF(zmwidth/2,10));


              //绘制图表说明

bp.DrawRectangle(Pens.Black,cxleft+width+10,zmheight/2,60,15*dt.Columns.Count -1);
              for (int j=0;j<dt.Columns .Count-1 ;j++)
              {
                  bp.FillRectangle(new
SolidBrush(this._color[j]),cxleft+width+10+2,zmheight/2+6+16*j,8,8);
                  //文字说明
                  bp.DrawString(dt.Columns[j+1].ColumnName,new Font("宋体",9),new
SolidBrush(Color.Black),new PointF(cxleft+width+10+2+8,zmheight/2+4+j*12));
              }




              //通过循环绘制标准线
              for(int i=0;i<=kds;i++)
              {


                  string Cur_Kd;
                  if (i-this.Displacement!=0)
                  {
                      Cur_Kd=((i-this.Displacement)*(kddw)).ToString ("#,###.##");
                      //
                  }
                  else
                  {
                      Cur_Kd="0";
                  }

                  bp.DrawString (Cur_Kd,new Font("宋体",9),new
SolidBrush(Color.Black),new PointF(2,zmheight-(bl(i*kddw)+(zmheight-cxtop-height)+4)));
                  //填充标准线(画刷，起点X，起点Y，宽，高）
                  int top=cxtop+height-(int)(bl(i*kddw));
```

```
                  bp.DrawLine(new Pen(bzlink),new Point(cxleft-4,top),new Point(cxleft+width,top));
                //    this.Page .Response .Write ("<br>abc"+Cur_Kd.ToString ());
              }

              for(int j=1;j<dt.Rows.Count+1;j++)
              {

                  bp.DrawLine (new Pen (bzlink),new Point(  cxleft+ (dt.Columns .Count -1)*j* 40, (int)  (cxtop+height-bl(this.Displacement*kddw) ) ),new Point (cxleft+ (dt.Columns .Count -1)*j* 40, (int)  (cxtop+height-bl(this.Displacement*kddw) )+12));

                  bp.DrawString(dt.Rows[j-1].ItemArray[0].ToString (),new Font("宋体",9),new SolidBrush(Color.Black),new PointF(( (dt.Columns .Count -1) *(j-1)*40)+cxleft+40, (int)(cxtop+height-bl(this.Displacement*kddw) ) +1));

              }

        }
        #endregion

        饼状图#region  饼状图

        private string  Draw_Pie(DataTable dt)
        {

            int width = 240;
            const int page_top_margin = 15;

            float total = 0.0F, tmp;
            int i;
            for (i=0; i < dt.Rows.Count; i++)
            {
                tmp = Convert.ToSingle(dt.Rows[i][1]);
                total += tmp;
            }



            Font fontLegend = new Font("Verdana", 10);

            Font fontTitle = new Font("Verdana", 12, FontStyle.Bold);
            int titleHeight = fontTitle.Height + page_top_margin;
```

```
                int row_gap = 6;
                int start_of_rect = 8;
                int rect_width = 14;
                int rect_height = 16;

                int row_height;
                if (rect_height > fontLegend.Height) row_height = rect_height; else row_height =
fontLegend.Height;
                row_height += row_gap;

                int legendHeight = row_height * (dt.Rows.Count+1);
                int height = width + legendHeight + titleHeight + page_top_margin;
                int pieHeight = width;
                Rectangle pieRect = new Rectangle(0, titleHeight, width, pieHeight);




                float currentDegree = 0.0F;



                Bitmap bm = new Bitmap(width, height);
            Graphics       objGraphics= Graphics.FromImage(bm) ;

                SolidBrush blackBrush = new SolidBrush(Color.Black);



                objGraphics.FillRectangle(new SolidBrush(Color.White), 0, 0, width, height);
                for (i = 0; i < dt.Rows.Count; i++)
                {
                     objGraphics.FillPie(
                           new SolidBrush (this._color[i]),
                          pieRect,
                          currentDegree,
                          Convert.ToSingle(dt.Rows[i][1]) / total * 360);


                     currentDegree += Convert.ToSingle(dt.Rows[i][1]) / total * 360;
                }


                StringFormat stringFormat = new StringFormat();
```

```
stringFormat.Alignment = StringAlignment.Center;
stringFormat.LineAlignment = StringAlignment.Center;

objGraphics.DrawString(this.datastd, fontTitle, blackBrush,
    new Rectangle(0, 0, width, titleHeight), stringFormat);



objGraphics.DrawRectangle(
    new Pen(Color.Gray, 1),
    0,
    height - legendHeight,
    width-4,
    legendHeight-1);

int y = height - legendHeight + row_gap;

for (i = 0; i < dt.Rows.Count; i++)
{

    objGraphics.FillRectangle(
        new SolidBrush (this._color[i]),
        start_of_rect,    // x
        y,
        rect_width,
        rect_height);

    objGraphics.DrawString(
        Convert.ToString(dt.Rows[i][0])
        + " - " +
        Convert.ToString(dt.Rows[i][1]),
        fontLegend,
        blackBrush,
        start_of_rect + rect_width + 4,
        y);

    y += rect_height + row_gap;


}
```

```
            FileStream                                                                 fs=new
FileStream(Page.Server.MapPath(Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")),FileMo
de.Create);
            bm.Save(fs,ImageFormat.Jpeg);

            bm.Dispose();
            objGraphics.Dispose();
            fs.Close();
            return     "<img   src="+Page.Request.Url.AbsolutePath.Replace(".aspx",".jpg")+"
></img>";

        }
        #endregion


}
}
```

调用页面

```
  using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace Web_Graphic
{
    /**//// <summary>
    /// WebForm1 的摘要说明。
    /// </summary>
    public class WebForm1 : System.Web.UI.Page
    {
        protected Tom.Control.Columniation Columniation1;

        private void Page_Load(object sender, System.EventArgs e)
        {
```

```
DataTable dt=new DataTable();
DataColumn dc;

dc=new DataColumn();
dc.DataType=System.Type.GetType("System.String");
dc.ColumnName="name";
dt.Columns.Add(dc);

dc=new DataColumn();
dc.DataType=System.Type.GetType("System.Int32");
dc.ColumnName="db";
dt.Columns.Add(dc);

dc=new DataColumn();
dc.DataType=System.Type.GetType("System.Int32");
dc.ColumnName="df";
dt.Columns.Add(dc);

DataRow dr=dt.NewRow();

dr["name"]="点1";

dr["db"]="1400";
dr["df"]="500";

dt.Rows.Add(dr);

dr=dt.NewRow();
dr["name"]="点2";

dr["db"]="200";
dr["df"]="200";
dt.Rows.Add(dr);

dr=dt.NewRow();
dr["name"]="点3";

dr["db"]="-300";
    dr["df"]="-600";
dt.Rows.Add(dr);

dr=dt.NewRow();
dr["name"]="点4";
```

```
            dr["db"]="200";
             dr["df"]="1500";
            dt.Rows.Add(dr);

            dr=dt.NewRow();
            dr["name"]="点 5";

            dr["db"]="400";
                  dr["df"]="2400";
            dt.Rows.Add(dr);

            Columniation1.Items=dt;
            Columniation1.ChatStyle=2;   //1 为柱状,2 为折线, 3 为屏状                  }

        Web 窗体设计器生成的代码#region Web 窗体设计器生成的代码
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /**//// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容。
        /// </summary>
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion
    }
}
```

效果如下...

\

GDI+显示 GIF 动画 - blog - 博客园

GDI+显示 GIF 动画
http://www.cppblog.com/TechLab/articles/862.html

多页图像是指图像中包含有多个图形页。每页可以看作图像帧。这些图像帧通过连续的显示就形成了一副动画。比如 GIF 文件。

GDI+的 Image 对象提供了直接的对 GIF、TIF 文件格式的支持。调用 Image 对象的成员函数 GetFrameDimensionsCount 可以得到 Image 对象的 Dimension 数。每个 Dimension 通过一个 GUID 标示。函数 GetFrameDimensionsList 可以返回所有 Dimension 的 GUID 值。第一个 GUID 值保存在函数参数 pDimensionsIDs 数组的索引 0 处。GetFrameCount 可以得到每个 Dimension 里有多少个 Frame。简单示例代码：

```
    Image* image = new Image(L"Multiframe.gif");
    UINT count = 0;
    count = image->GetFrameDimensionsCount();
    GUID *pDimensionIDs=(GUID*)new GUID[count];
    image->GetFrameDimensionsList(pDimensionIDs, count);
    WCHAR strGuid[39];
    StringFromGUID2(pDimensionIDs[0], strGuid, 39);
    UINT frameCount=image->GetFrameCount(&pDimensionIDs[0]);
    delete []pDimensionIDs;
```

并不是所有的 GIF 文件都是含有多帧的，所以我们在显示 GIF 的时候可以通过上面的代码根据 frameCount 的值判断这个 GIF 文件是否有多个帧。

在确认有多个帧的图像以后，还要得到每帧图像显示的间隔时间。GDI+的 Image 对象提供了
GetPropertyItem 获取图像的属性。GetPropertyItem 函数需要用户传递数据返回缓冲区和大小。所以在使用前先用 GetPropertyItemSize 得到需要的缓冲区大小，分配空间后再取得属性数据。

```
    //PropertyTagFrameDelay 是 GDI+中预定义的一个 GIG 属性 ID 值，表示标签帧数据的延迟时间
    int size = GetPropertySize(PropertyTagFrameDelay);
    PropertyItem* pItem = NULL;
    pItem = (PropertyItem*)malloc(size);
    image->GetPropertyItem(PropertyTagFrameDelay,size,pItem);
```

这样就把所有和 PropertyTagFrameDelay 属性相关的数据取到了 pItem 中。然后通过 pItem 访问结构中的 value。每两帧图像之间的间隔时间是不一定相同的，所以还需要得到当前正显示的帧图像的索引值。最后调用 Image 对象的 DrawImage 函数把每帧图像画出来。简单代码如下：

```
int        fcount=0;
//Guid 的值在显示 GIF 为 FrameDimensionTime，显示 TIF 时为 FrameDimensionPage
```

```
GUID        Guid = FrameDimensionTime;
while(thue)
{
            Graphics gh(hDC); //hDC 是外部传入的画图 DC
            gh.DrawImage(image,0,0,image->GetWidth(),image->GetHeight());
            //重新设置当前的活动数据帧
            image->SelectActiveFrame(&Guid,fcount++);
            if(fcount == frameCount) //frameCount 是上面 GetFrameCount 返回值
                      fcount= 0;         //如果到了最后一帧数据又重新开始
            //计算此帧要延迟的时间
            long lPause = ((long*)pItem->value)[fcount]*10;
            Sleep(lPause);               //这里简单使用了 sleep
}
```

posted on 2005-11-08 10:14 henry 阅读(155) 评论(0) 编辑 收藏 所属分类: 图形、图像

<img src ="http://www.cnblogs.com/henryzc/aggbug/271271.html?webview=1" width =
"1" h

解决 GDI+的图像质量的问题 - Kikee 每天都在变 - 博客园

Kikee 每天都在变

世界在变 我在变

博客园 首页 新随笔 联系 订阅 管理

随笔-84 评论-20 文章-1 trackbacks-0

解决 GDI+的图像质量的问题

最近在完成一个文章发布系统时需要给图片添加水印，采用 GDI+的 DrawString 后发现图像质量非常糟糕，后来从其他地方找到如下代码，经过试验
图像质量大幅提高：

//上传成功，加水印

```
                                    string path=UploadFileDestination + UploadFileName;
                                    System.Drawing.Image img =
System.Drawing.Image.FromFile(path);
                  System.Drawing.Imaging.ImageFormat thisFormat = img.RawFormat;

                  //Size newSize = NewSize(maxWidth, maxHeight, img.Width,
img.Height);
                  //Bitmap outBmp = new Bitmap(newSize.Width, newSize.Height);
                  Graphics g = Graphics.FromImage(img);

                  // 设置画布的描绘质量
                  g.CompositingQuality = CompositingQuality.HighQuality;
                  g.SmoothingMode = SmoothingMode.HighQuality;
                  g.InterpolationMode = InterpolationMode.HighQualityBicubic;
```

```
                g.DrawImage(img, 0, 0, img.Width, img.Height);

                Font f = new Font("Arial",9);
                    Brush b = new SolidBrush(Color.Black);
                    Brush c = new SolidBrush(Color.White);
                    string addText = ".com 在线";

                    g.DrawString(addText, f, b, 3, 3);
                    g.DrawString(addText, f, c, 2, 2);

                    g.DrawString(addText, f, b, img.Width-140, img.Height-15);
                    g.DrawString(addText, f, c, img.Width-141, img.Height-16);
                    g.Dispose();


                // 以下代码为保存图片时，设置压缩质量
                EncoderParameters encoderParams = new EncoderParameters();
                long[] quality = new long[1];
                quality[0] = 100;

                EncoderParameter encoderParam = new
EncoderParameter(System.Drawing.Imaging.Encoder.Quality, quality);
                encoderParams.Param[0] = encoderParam;

                //获得包含有关内置图像编码解码器的信息的 ImageCodecInfo 对象。
                ImageCodecInfo[] arrayICI = ImageCodecInfo.GetImageEncoders();
                ImageCodecInfo jpegICI = null;
                for (int x = 0; x < arrayICI.Length; x++)
                {
                    if (arrayICI[x].FormatDescription.Equals("JPEG"))
                    {
                        jpegICI = arrayICI[x];//设置 JPEG 编码
                        break;
                    }
                }

                string newPath = (UploadFileDestination+RNDFile())+newext;

                if (jpegICI != null)
                {
                    img.Save(newPath, jpegICI, encoderParams);
                }
                else
```

```
{
    img.Save(newPath, thisFormat);
}

img.Dispose();


    //保存加水印过后的图片,删除原始图片


    if(File.Exists(path))
    {
        File.Delete(path);
    }


        //水印添加完成
```

posted on 2005-11-06 20:18 龚祺 阅读(97) 评论(1) 编辑 收藏 所属分类: .Net / XML

如何在C#的WinForm中制作饼状图和柱状图 - 小马的天空 - 博客园

如何在C#的WinForm中制作饼状图和柱状图
当我们的软件需要各种饼状图和柱状图来表示数据时，我们或许会想到用Offices中的图形控件或是第三方控件，但现在的第三方控件大都需要注册，有些免费的控件会有开发商的标记等。而对于使用Offices的图形控件来说，并不能在程序中得于很好控制，其使用的简易程度也较低，所以在这我给出在C#中使用GDI+实现饼状图和柱状图跟数据库联接显示数据的方法。

```
using System;
using System.IO;//用于文件存取
using System.Data;//用于数据访问
using System.Drawing;//提供画GDI+图形的基本功能
using System.Drawing.Text;//提供画GDI+图形的高级功能
using System.Drawing.Drawing2D;//提供画高级二维，矢量图形功能
using System.Drawing.Imaging;//提供画GDI+图形的高级功能
namespace BaseLayer
{
public class PieChart
{
public PieChart()
{
}
```

```
//Render是图形大标题，图开小标题，图形宽度，图形长度，饼图的数据集和饼图的数据集要表示出来的数据
public Image Render(string title, string subTitle, int width, int height,
DataSet chartData,int DataLine)
{
const int SIDE_LENGTH = 400;
const int PIE_DIAMETER = 200;
DataTable dt = chartData.Tables[0];

//通过输入参数，取得饼图中的总基数
float sumData = 0;
foreach(DataRow dr in dt.Rows)
{
sumData += Convert.ToSingle(dr[DataLine]);
}
//产生一个image对象，并由此产生一个Graphics对象
Bitmap bm = new Bitmap(width,height);
Graphics g = Graphics.FromImage(bm);
//设置对象g的属性
g.ScaleTransform((Convert.ToSingle(width))/SIDE_LENGTH,(Convert.ToSingle(height))/SIDE_LENGTH);

g.SmoothingMode = SmoothingMode.Default;
g.TextRenderingHint = TextRenderingHint.AntiAlias;

//画布和边的设定
g.Clear(Color.White);
g.DrawRectangle(Pens.Black,0,0,SIDE_LENGTH-1,SIDE_LENGTH-1);
//画饼图标题
g.DrawString(title,new Font(Tahoma,14),Brushes.Black,new PointF(5,5));
//画饼图的图例
g.DrawString(subTitle,new Font(Tahoma,12),Brushes.Black,new PointF(7,35));
//画饼图
float curAngle = 0;
float totalAngle = 0;
for(int i=0;i<dt.Rows.Count;i++)
{
curAngle = Convert.ToSingle(dt.Rows[i][DataLine]) / sumData * 360;

g.FillPie(new SolidBrush(ChartUtil.GetChartItemColor(i)),100,65,PIE_DIAMETER,PIE_DIAMETER,totalAngle,curAngle);

g.DrawPie(Pens.Black,100,65,PIE_DIAMETER,PIE_DIAMETER,totalAngle,curAngle);
totalAngle += curAngle;
}
//画图例框及其文字
g.DrawRectangle(Pens.Black,200,300,199,99);
g.DrawString(图表说明,new Font(Tahoma,12,FontStyle.Bold),Brushes.Black,new
PointF(200,300));

//画图例各项
PointF boxOrigin = new PointF(210,330);
PointF textOrigin = new PointF(235,326);
float percent = 0;
for(int i=0;i<dt.Rows.Count;i++)
{
g.FillRectangle(new
SolidBrush(ChartUtil.GetChartItemColor(i)),boxOrigin.X,boxOrigin.Y,20,10);
g.DrawRectangle(Pens.Black,boxOrigin.X,boxOrigin.Y,20,10);
percent = Convert.ToSingle(dt.Rows[i][DataLine]) / sumData * 100;
g.DrawString(dt.Rows[i][1].ToString() + - + dt.Rows[i][0].ToString() + ( +
percent.ToString(0) + %),new Font(Tahoma,10),Brushes.Black,textOrigin);
boxOrigin.Y += 15;
textOrigin.Y += 15;
}
//回收资源
g.Dispose();
return (Image) bm;

}
}
//画条形图
public class BarChart
{
public BarChart()
{
}
//Render 是图形大标题，图开小标题，图形宽度，图形长度，饼图的数据集和饼图的数据集
public Image Render(string title, string subTitle, int width, int height,
DataSet chartData)
{
const int SIDE_LENGTH = 400;
const int CHART_TOP = 75;
const int CHART_HEIGHT = 200;
const int CHART_LEFT = 50;
const int CHART_WIDTH = 300;
DataTable dt = chartData.Tables[0];
```

```
//计算最高的点
float highPoint = 0;
foreach(DataRow dr in dt.Rows)
{
if(highPoint<Convert.ToSingle(dr[0]))
{
highPoint = Convert.ToSingle(dr[0]);
}
}
//建立一个 Graphics 对象实例
Bitmap bm = new Bitmap(width,height);
try
{
Graphics g = Graphics.FromImage(bm);
//设置条图图形和文字属性
g.ScaleTransform((Convert.ToSingle(width))/SIDE_LENGTH,(Convert.ToSingle(height))/SIDE_
LENGTH);

g.SmoothingMode = SmoothingMode.Default;
g.TextRenderingHint = TextRenderingHint.AntiAlias;

//设定画布和边
g.Clear(Color.White);
g.DrawRectangle(Pens.Black,0,0,SIDE_LENGTH-1,SIDE_LENGTH-1);
//画大标题
g.DrawString(title,new Font(Tahoma,14),Brushes.Black,new PointF(5,5));
//画小标题
g.DrawString(subTitle,new Font(Tahoma,12),Brushes.Black,new PointF(7,35));
//画条形图
float barWidth = CHART_WIDTH / (dt.Rows.Count * 2);
PointF barOrigin = new PointF(CHART_LEFT + (barWidth / 2),0);
float barHeight = dt.Rows.Count;
for(int i=0;i<dt.Rows.Count;i++)
{
barHeight = Convert.ToSingle(dt.Rows[i][0]) * 200 / highPoint * 1;
barOrigin.Y = CHART_TOP + CHART_HEIGHT - barHeight;
g.FillRectangle(new
SolidBrush(ChartUtil.GetChartItemColor(i)),barOrigin.X,barOrigin.Y,barWidth,barHeight);

barOrigin.X = barOrigin.X + (barWidth * 2);
}
//设置边
g.DrawLine(new Pen(Color.Black,2),new Point(CHART_LEFT,CHART_TOP),new
Point(CHART_LEFT,CHART_TOP + CHART_HEIGHT));
g.DrawLine(new Pen(Color.Black,2),new Point(CHART_LEFT,CHART_TOP +
CHART_HEIGHT),new Point(CHART_LEFT + CHART_WIDTH,CHART_TOP +
CHART_HEIGHT));
//画图例框和文字
g.DrawRectangle(new Pen(Color.Black,1),200,300,199,99);
g.DrawString(图表说明,new Font(Tahoma,12,FontStyle.Bold),Brushes.Black,new
PointF(200,300));

//画图例
PointF boxOrigin = new PointF(210,330);
PointF textOrigin = new PointF(235,326);
for(int i=0;i<dt.Rows.Count;i++)
{
g.FillRectangle(new
SolidBrush(ChartUtil.GetChartItemColor(i)),boxOrigin.X,boxOrigin.Y,20,10);
g.DrawRectangle(Pens.Black,boxOrigin.X,boxOrigin.Y,20,10);
g.DrawString(dt.Rows[i][1].ToString() + - + dt.Rows[i][0].ToString(),new
Font(Tahoma,10),Brushes.Black,textOrigin);
boxOrigin.Y += 15;
textOrigin.Y += 15;
}
//输出图形
g.Dispose();
return bm;
}
catch
{
return bm;
}
}
}
public class ChartUtil
{
public ChartUtil()
{
}
public static Color GetChartItemColor(int itemIndex)
{
Color selectedColor;
switch(itemIndex)
{
case 0:
selectedColor = Color.Blue;
```

```
break;
case 1:
selectedColor = Color.Red;
break;
case 2:
selectedColor = Color.Yellow;
break;
case 3:
selectedColor = Color.Purple;
break;
default:
selectedColor = Color.Green;
break;
}
return selectedColor;
}
}
}
```

posted on 2006-06-07 09:07 Vinson 阅读(122) 评论(0) 编辑 收藏

学习使用 GDI+绘制饼状图 - Leey - 博客园

学习使用 GDI+绘制饼状图

Posted on 2005-10-18 16:56 Leey 阅读(364) 评论(1) 编辑 收藏 所属分类: 学习笔记

早就申请了博客园的空间，但是一直也没有发过文章。
这次就算个开始吧

对于 GDI＋，我不是很熟悉，只是工作中用到了，才现找资料学习
这次画饼状图也是一样，Web 页面中需要加入统计的饼状图，没办法。
虽然最终把图画出来了，但是图上却有一条虚线去不掉，到现在也搞不懂为什么。
如下图，是最终的效果。

在黄色区域中间有一条细细的虚线。
代码如下：

```
using System;
using System.Collections;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
```

```
namespace JRJC.Web.QuanZheng
{
    /// <summary>
    /// XChart 的摘要说明。
    /// </summary>
    public class XChart
    {
        /// <summary>
        /// 根据传递的数组绘制饼图
        /// </summary>
        /// <param name="Width">宽度</param>
        /// <param name="Height">高度</param>
        /// <param name="dataArr">数据数组</param>
        /// <param name="FileName">文件名称</param>
        public static void DrawPieByArrayList(int Width, int Height, int
Magin, int Blank, DataAndColorPair[] arr, string FileName)
        {
            Bitmap bitmap = new Bitmap(Width, Height); //图片
            Graphics g = Graphics.FromImage(bitmap); //绘图
            g.DrawRectangle(new Pen(Color.White),0,0,Width,Height); //边框
            g.FillRectangle(new SolidBrush(Color.White), 1, 1,Width -
Magin, Height - Magin);
            SolidBrush brush = new SolidBrush(Color.Blue); //画笔
            g.SmoothingMode = SmoothingMode.AntiAlias; //柔化
            //g.DrawLine(new Pen(brush,10),1,1,10,10);
            Rectangle outline = new Rectangle(Blank,Blank,Width - Blank *
2, Height - Blank * 2); //饼图范围

            //绘制饼图
            float total = 0;
            float start = 0;
            float angle = 0;
            for(int i=0;i<arr.Length;i++)
            {
                total += arr[i].Data;
            }
            //Console.WriteLine(total.ToString());
            for(int i=0;i<arr.Length;i++)
            {
                angle = (float)(arr[i].Data/total) * 360;
                //Console.WriteLine(angle.ToString());
                if(angle<=180)
                {
                    Draw3DPie(ref g,arr[i].ShowColor,outline,start,angle);
                }
                else//如果大于 180 度，则将大 Pie 拆开两块画
                {
                    float acuteAngle=angle-180;
                    Draw3DPie(ref
g,arr[i].ShowColor,outline,start,acuteAngle);
                    Draw3DPie(ref
g,arr[i].ShowColor,outline,start+acuteAngle,angle-acuteAngle);
                }
                //Draw3DPie(ref g,arr[i].ShowColor,outline,start,angle);
                start += angle;
            }
            bitmap.Save(FileName, ImageFormat.Gif);
        }
        private static void Draw3DPie(ref Graphics g,Color
brushColor,Rectangle outline,float start,float angle)
        {
            //以下使用 HatchBrush 画阴影
            //if((start <135 ) || (start == 180))
            if((start <135 ))
            {
                //深度
                for(int iLoop2 = 0; iLoop2 < 10; iLoop2++)
                {
                    g.FillPie(new
HatchBrush(HatchStyle.Percent50,brushColor),
                        outline.X,
                        outline.Y + iLoop2,
                        outline.Width,
                        outline.Height,
                        start,
                        angle);
                }
            }
            if(start == 135)
            {
                for(int iLoop2 = 0; iLoop2 < 10; iLoop2++)
                {
                    g.FillPie(new
HatchBrush(HatchStyle.Percent50,brushColor),
                        outline.X - 30,
                        outline.Y + iLoop2 + 15,
                        outline.Width,
                                                outline.Height,
                                                start,
                                                angle);
                                }
                                g.FillPie(new SolidBrush(brushColor),
                                        outline.X - 30,
                                        outline.Y + 15,
                                        outline.Width,
                                        outline.Height,
                                        start,
                                        angle);
                        }
                        else //画 pie
                        {
                                g.FillPie(new SolidBrush(brushColor), outline.X,
outline.Y, outline.Width,outline.Height, start, angle);
                        }
                        //g.FillPie(new
SolidBrush(arr[i].ShowColor),outline,start,angle);
                }
        }
}
```

用 c#做的控件--用 vml 生成饼图

点滴 于 2004-7-13 23:15:08 加贴在 .NET 技术 ←返回版面：

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Text;
using System.Collections;
using System.Collections.Specialized;

namespace Controls.Custom
{
        /// <summary>
        /// ChartReport 的摘要说明。
/// </summary>
[DefaultProperty("Text"),
ToolboxData("<{0}:ChartPieReport runat=server></{0}:ChartReport>")]
public class ChartPieReport:Control
{
    private int max;
    private string  data;
    private string dataInfor;
    private string color;
    private string strPie;
    string[] tmpdata;//传的数值
    string[] tmpdatainfor;//传的对应内容
    string[] tmpColor;
    string str_temp="";
    string strdivdatainfo="";
    ArrayList resultData=new ArrayList();
    int  r=2000;

    int  colorInfoTop = -2000;
    int  dataInfoTop = 10;
    double total=0;
    #region 属性
    [Bindable(true),Category("Appearance"),DefaultValue("")]
    public int Max
    {
        get
        {
            return max;
        }
        set
        {
            max=value;
        }
    }
    [Bindable(true),Category("Data"),DefaultValue("")]
    public string Data
    {
        get
        {
            return data;
        }
        set
        {
            data=value;
        }
    }
    [Bindable(true),Category("Data"),DefaultValue("")]
    public string DataInfor
    {
        get
        {
            return DataInfor ;
        }
        set
        {
            dataInfor=value;
        }
    }
    [Bindable(true),Category("Data"),DefaultValue("")]
    public string Color
    {
        get
        {
            return color;
        }
        set
        {
            color=value;
        }
    }
    #endregion
    public ChartPieReport()
    {

        strPie+="<v:group id=\"p\" CoordOrig=\"3000,2700\" CoordSize=\"7200,4050\" style=\"WIDTH:480px;POSITION:relative;HEIGHT:270px\">";
        strPie+="<v:rect style=\"LEFT:-3000px;WIDTH:9600px;POSITION:relative;TOP:-2700px;HEIGHT:5400px\" fillcolor=\"white\"";
        strPie+="strokecolor=\"black\" coordsize=\"21600,21600\">";
        strPie+="<v:shadow on=\"t\" type=\"single\" color=\"silver\" offset=\"4pt,3pt\"></v:shadow>";
        strPie+="</v:rect>";
        strPie+="<v:group id=\"group1\" CoordSize=\"6600,5400\" style=\"WIDTH:8000px;POSITION:absolute;HEIGHT:5400px\">";
        strdivdatainfo="<div id=\"dataInfo\">";

    }
```

```
        protected override void Render(HtmlTextWriter output)
        {
            tmpdata=data.Split(new char[]{','});//传的数值
            tmpdatainfor=dataInfor.Split(new char[]{','});//传的对应内容
            tmpColor=color.Split(new char[]{','});//颜色
            showMorePie();
            this.Page.Response.Write (strPie);
        }
        #region  方法
        private void showMorePie()
        {
            //首先取数字的绝对值
            for(int i=0;i<tmpdata.Length;i++)
            {

tmpdata[i]=Convert.ToString(Math.Abs(Convert.ToDouble(tmpdata[i])));
                total+=Convert.ToDouble(tmpdata[i]);
            }
            //冒泡排序
            object temp;
            for(int i=0;i<tmpdata.Length-1;i++)
            {
                for(int j=i+1;j<tmpdata.Length;j++)
                {

if(Convert.ToDouble(tmpdata[i])<Convert.ToDouble(tmpdata[j]))
                    {
                        temp=Convert.ToDouble(tmpdata[i]);
                        tmpdata[i]=tmpdata[j];
                        tmpdata[j]=temp.ToString();

                        temp=tmpdatainfor[i];
                        tmpdatainfor[i]=tmpdatainfor[j];
                        tmpdatainfor[j]=temp.ToString();
                    }
                }
            }
            int loop=tmpdata.Length;
            //判断是不是给出的最大的值与实际给的内容不符合
            if(loop>max)
            {
                for(int i=max;i<tmpdata.Length;i++)
                    tmpdata[max-1]+=Convert.ToDouble(tmpdata[i]);
                tmpdatainfor[max-1]="其他";
```

```
                    loop=max;

                }
                for(int i=0;i<loop;i++)
                {
                    resultData.Add(Convert.ToDouble(tmpdata[i])/total);
                    showColorInfo(i);
                    showDataInfo(i);

                }
                strdivdatainfo+="</div>";

createPie(0,Convert.ToDouble(resultData[0]),tmpColor[0].ToString(),tmpdatainfor[0]+":"+tmpdat
a[0].ToString());
                double currentValue=Convert.ToDouble(resultData[0]);
                for(int i=1;i<loop-1;i++)
                {

createPie(currentValue,Convert.ToDouble(currentValue+Convert.ToDouble(resultData[i])),tmpCo
lor[i].ToString(),tmpdatainfor[i]+":"+tmpdata[i].ToString());
                    currentValue+=Convert.ToDouble(resultData[i]);
                }

createPie(currentValue,360,tmpColor[loop-1].ToString(),tmpdatainfor[loop-1]+":"+tmpdata[loop-
1].ToString());
                strPie+="<div id=\"colorInfo\">";
                strPie+= str_temp;
                strPie+="</div>";
                strPie+=strdivdatainfo;
                strPie+="</v:group>";
                strPie+="</v:group>";
            }
            public void showColorInfo(int i)
            {
                int top=colorInfoTop-300+i*373;
                string temp="<v:rect id='a"+i.ToString()+"'
style='position:relative;left:2600;top:"+top.ToString()+";WIDTH:180;HEIGHT:180;'
fillcolor='"+tmpColor[i]+"' strokecolor='black'><v:shadow on='t' type='single'
color='silver' offset='2pt,2pt'></v:shadow></v:rect>";
                str_temp=temp+str_temp;
            }
            public void showDataInfo(int i)
            {
                int top = dataInfoTop-10 + i*11;
                string nInfo="<Div align='left'
style='font-size:9pt;position:relative;top:"+top+";left:420'>"+tmpdatainfor[i]+"</Div>";
                strdivdatainfo+= nInfo;
            }
            public void createPie(double sa,double ea,string color,string n)
            {
                double fs=Convert.ToDouble(Math.PI*2*sa);
                double fe=Convert.ToDouble(Math.PI*2*ea);
                int sx=Convert.ToInt32(r*Math.Cos(fs));
                int sy=Convert.ToInt32(-r*Math.Sin(fs));
                int ex=Convert.ToInt32(r*Math.Cos(fe));
                int ey=Convert.ToInt32(-r*Math.Sin(fe));
                int index;
                if (sa<=0.3)
                {
                    index = 100 - Convert.ToInt32((sa * 100));
                    strPie+="<v:shape onclick='JavaScript:alert(\""+n+"\")'
title='"+n+"'
style='position:relative;left:-800;top:-1000;z-index:"+index.ToString()+";width:7000;height:6000'
 fillcolor='"+color+"' path='m 0,0 l "+sx+","+sy+" ar
-2000,-2000,2000,2000,"+sx+","+sy+","+ex+","+ey+","+" l 0,0 x e'>
                    <v:Extrusion on='t' foredepth='0' backdepth='20pt'
rotationangle='66,0' color='"+color+"'></v:Extrusion></v:shape>";

                }
                else if (sa>=0.75)
                {
                        index = (100 - Convert.ToInt32((sa * 100)))+100;
                        strPie+="<v:shape onclick='JavaScript:alert(\""+n+"\")'
title='"+n+"'
style='position:relative;left:-800;top:-1000;z-index:"+index.ToString()+";width:7000;height:6000'
fillcolor='"+color+"' path='m 0,0 l "+sx+","+sy+" ar
-2000,-2000,2000,2000,"+sx+","+sy+","+ex+","+ey+","+" l 0,0 x e'>
                    <v:Extrusion on='t' foredepth='0' backdepth='20pt'
rotationangle='66,0' color='"+color+"'></v:Extrusion></v:shape>";
                }
                else
                {
                        strPie+="<v:shape onclick='JavaScript:alert(\""+n+"\")'
title='"+n+"'
style='position:relative;left:-800;top:-1000;z-index:100;width:7000;height:6000'
fillcolor='"+color+"' path='m 0,0 l "+sx+","+sy+" ar
-2000,-2000,2000,2000,"+sx+","+sy+","+ex+","+ey+","+" l 0,0 x e'>
                    <v:Extrusion on='t' foredepth='0' backdepth='20pt'
rotationangle='66,0' color='"+color+"'></v:Extrusion></v:shape>";

                    }
            }
            #endregion
        }
}
```

//使用例子

```
private void Page_Load(object sender, System.EventArgs e)
            {
                    Chat1.Max=13;

                    Chat1.Data="12.5,25,19.5,17.5,12.5,16,9,21,3,7,8,6.2,9";


Chat1.DataInfor="杨浦区,闸北区,普陀区,虹口区,宝山区,嘉定区,黄浦区,闵行区,浦东区,静安,徐汇区,其他区,其他区";


Chat1.Color="red,#FFEBCD,#FFFF55,#00BFFF,#E6E6FA,#7B68EE,#FFCC33,#FFC0CB,#008B8B,#c27f34,#FFDEAD,#00FA9A,#D3D3D3";

            }
```

//保证程序运行正常请按下面做

下面是使用的时候，html 代码中需要加入的，注意<html>标记处，和<Style>标记处。

```
<HTML xmlns:v="urn:schemas-microsoft-com:vml">
        <HEAD>
                <title>WebForm2</title>
                <meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
                <meta name="CODE_LANGUAGE" Content="C#">
                <meta name="vs_defaultClientScript" content="JavaScript">
                <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
                <STYLE> v\:* { BEHAVIOR: url(#default#VML) } </STYLE>
        </HEAD>
        <body MS_POSITIONING="GridLayout">
                <form id="Form1" method="post" runat="server">
                        <cc1:ChartVolumeReport id="Chat1"
runat="server"></cc1:ChartVolumeReport>
                </form>
        </body>
</HTML>
```

计算机世界开发者俱乐部 http://dev-club.esnai.com

用 c#做的控件--用 vml 生成饼图 - 点滴 2004-7-13 23:15:08 [ID:2375465 点击:457] (9775 Bytes)
(11)

请问用 vml 作出的统计图，对浏览器版本有什么要求啊？98 浏览器支持吗？谢谢！ <无内容> - gongfan 2004-12-20
10:20:29 [ID:2462829 点击:268] (0 Bytes) (0)

改一下<summary>ToolboxData("<{0}:ChartPieReport
runat=server></{0}:ChartReport>")] - mingziweb 2004-7-15 11:45:06
[ID:2376548 点击:289] (284 Bytes) (0)

确实运行有问题,请看.protected override void Render(HtmlTextWriter output) -
mingziweb 2004-7-15 11:34:24 [ID:2376530 点击:327] (255 Bytes) (0)

我给打了一个包 - qushui 2004-7-15 10:25:10 [ID:2376405 点击:360] (94 Bytes) (0)

效果图 <无内容> - qushui 2004-7-15 10:21:51 [ID:2376399 点击:401] (0 Bytes) (0)

这段代码的确有问题，好像是没考全，兄弟再发一遍吧~~ <无内容> - icefox 2004-7-14 13:22:45 [ID:2375761
点击:310] (0 Bytes) (1)

把你那个控件的代码直接传上来多好呀 <无内容> - 小小浪子 2004-7-14 10:34:30 [ID:2375625 点击:240] (0
Bytes) (0)

有问题，上面的控件编译时好多的错误！！！ - qushui 2004-7-14 9:53:39 [ID:2375545 点击:300] (237
Bytes) (1)

之前的 gdi+绘图,整理了一下,清晰了很多 - 南方小鬼的博客 - 博客园

柱状图

```
using System;
using System.IO;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace WanFangData.Chart
{
    public class Bar
    {
        public void Render(string title, string subTitle, int width, int
height, DataSet chartData, Stream target)
        {
            const int SIDE_LENGTH = 400;
            const int CHART_TOP = 75;
            const int CHART_HEIGHT = 200;
            const int CHART_LEFT = 50;
            const int CHART_WIDTH = 300;
            DataTable dt = chartData.Tables[0];

            //计算最高的点
            float highPoint = 0;
            foreach (DataRow dr in dt.Rows)
            {
                if (highPoint < Convert.ToSingle(dr[1]))
                {
                    highPoint = Convert.ToSingle(dr[1]);
                }
            }
            //建立一个 Graphics 对象实例
            Bitmap bm = new Bitmap(width, height);
            Graphics g = Graphics.FromImage(bm);
            //设置条图图形和文字属性
            g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height)) / SIDE_LENGTH);
            g.SmoothingMode = SmoothingMode.Default;
            g.TextRenderingHint = TextRenderingHint.AntiAlias;

            //设定画布和边
            g.Clear(Color.White);
            g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
            //画大标题
            g.DrawString(title, new Font("Tahoma", 24), Brushes.Black, new PointF(5, 5));
            //画小标题
            g.DrawString(subTitle, new Font("Tahoma", 14), Brushes.Black, new PointF(7, 35));
            //画条形图
            float barWidth = CHART_WIDTH / (dt.Rows.Count * 2);
            PointF barOrigin = new PointF(CHART_LEFT + (barWidth / 2), 0);
            float barHeight = dt.Rows.Count;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                barHeight = Convert.ToSingle(dt.Rows[i][1]) * 200 / highPoint;
                barOrigin.Y = CHART_TOP + CHART_HEIGHT - barHeight;
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), barOrigin.X, barOrigin.Y, barWidth, barHeight);
                barOrigin.X = barOrigin.X + (barWidth * 2);
            }
            //设置边
            g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT));
            g.DrawLine(new Pen(Color.Black, 2), new Point(CHART_LEFT, CHART_TOP + CHART_HEIGHT), new Point(CHART_LEFT + CHART_WIDTH, CHART_TOP + CHART_HEIGHT));
            //画图例框和文字
            g.DrawRectangle(new Pen(Color.Black, 1), 200, 300, 199, 99);
            g.DrawString("Legend", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new PointF(200, 300));

            //画图例
            PointF boxOrigin = new PointF(210, 330);
            PointF textOrigin = new PointF(235, 326);
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawString(dt.Rows[i][0].ToString() + " - " + dt.Rows[i][1].ToString(), new Font("Tahoma", 10), Brushes.Black, textOrigin);
                boxOrigin.Y += 15;
                textOrigin.Y += 15;
            }
            //输出图形
            bm.Save(target, ImageFormat.Gif);

            //资源回收
            bm.Dispose();
            g.Dispose();
        }
    }
}

```

饼状图

```
using System;
using System.IO;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace WanFangData.Chart
{
    public class Pie
    {
        public void Render(string title, string subTitle, int width, int height, DataSet chartData, Stream target)
        {
            const int SIDE_LENGTH = 400;
            const int PIE_DIAMETER = 200;
            DataTable dt = chartData.Tables[0];

            //通过输入参数，取得饼图中的总基数
            float sumData = 0;
            foreach (DataRow dr in dt.Rows)
            {
                sumData += Convert.ToSingle(dr[1]);
            }
            //产生一个 image 对象，并由此产生一个 Graphics 对象
            Bitmap bm = new Bitmap(width, height);
            Graphics g = Graphics.FromImage(bm);
            //设置对象 g 的属性
            g.ScaleTransform((Convert.ToSingle(width)) / SIDE_LENGTH, (Convert.ToSingle(height)) / SIDE_LENGTH);
            g.SmoothingMode = SmoothingMode.Default;
            g.TextRenderingHint = TextRenderingHint.AntiAlias;

            //画布和边的设定
            g.Clear(Color.White);
            g.DrawRectangle(Pens.Black, 0, 0, SIDE_LENGTH - 1, SIDE_LENGTH - 1);
            //画饼图标题
            g.DrawString(title, new Font("Tahoma", 24), Brushes.Black, new PointF(5, 5));
            //画饼图的图例
            g.DrawString(subTitle, new Font("Tahoma", 14), Brushes.Black, new PointF(7, 35));
            //画饼图
            float curAngle = 0;
            float totalAngle = 0;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                curAngle = Convert.ToSingle(dt.Rows[i][1]) / sumData * 360;

                g.FillPie(new SolidBrush(Utils.GetChartItemColor(i)), 100, 65, PIE_DIAMETER, PIE_DIAMETER, totalAngle, curAngle);
                g.DrawPie(Pens.Black, 100, 65, PIE_DIAMETER, PIE_DIAMETER, totalAngle, curAngle);
                totalAngle += curAngle;
            }
            //画图例框及其文字
            g.DrawRectangle(Pens.Black, 200, 300, 199, 99);
            g.DrawString("Legend", new Font("Tahoma", 12, FontStyle.Bold), Brushes.Black, new PointF(200, 300));

            //画图例各项
            PointF boxOrigin = new PointF(210, 330);
            PointF textOrigin = new PointF(235, 326);
            float percent = 0;
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                g.FillRectangle(new SolidBrush(Utils.GetChartItemColor(i)), boxOrigin.X, boxOrigin.Y, 20, 10);
                g.DrawRectangle(Pens.Black, boxOrigin.X, boxOrigin.Y, 20, 10);
                percent = Convert.ToSingle(dt.Rows[i][1]) / sumData * 100;
                g.DrawString(dt.Rows[i][0].ToString() + " - " + dt.Rows[i][1].ToString() + " (" + percent.ToString("0") + "%)", new Font("Tahoma", 10), Brushes.Black, textOrigin);
                boxOrigin.Y += 15;
                textOrigin.Y += 15;
            }
            //通过 Response.OutputStream，将图形的内容发送到浏览器
            bm.Save(target, ImageFormat.Gif);
            //回收资源
            bm.Dispose();
            g.Dispose();
        }
    }
}
```

工具

```
using System;
using System.Drawing;

namespace WanFangData.Chart
```

```
{
    class Utils
    {
        public static Color GetChartItemColor(int itemIndex)
        {
            Color selectedColor;
            switch (itemIndex)
            {
                case 0:
                    selectedColor = Color.Blue;
                    break;
                case 1:
                    selectedColor = Color.Red;
                    break;
                case 2:
                    selectedColor = Color.Yellow;
                    break;
                case 3:
                    selectedColor = Color.Purple;
                    break;
                default:
                    selectedColor = Color.Green;
                    break;
            }
            return selectedColor;
        }
    }
}
```

调用柱状图

```
using System;
using System.Data;
using WanFangData.Chart;
using WanFangData.Common;

namespace WanFangData.Web
{
    public partial class CBar : WanFangData.Page.BasePage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Bar p = new Bar();
            p.Render("大标题", "小标题", 300,
300,WanFangData.Common.Database.ExecuteDataset(CommandType.Text, "select * from
a"), Response.OutputStream);
```

```
        }
    }
}

```

调用饼状图

```
using System;
using System.Data;
using WanFangData.Chart;
using WanFangData.Common;

namespace WanFangData.Web
{
    public partial class CPie : WanFangData.Page.BasePage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Pie p = new Pie();
            p.Render("大标题", "小标题", 400, 400,
Database.ExecuteDataset(CommandType.Text, "select * from a"),
Response.OutputStream);
        }
    }
}

```

用 GDI+怎么实现绘制倾斜文字

Q:用 GDI+怎么实现绘制倾斜文字

```
A:Graphics g = this.CreateGraphics();
g.RotateTransform(30f);
g.DrawString("倾斜 ABCabc", this.Font, SystemBrushes.WindowText, 10f, 10f);
g.ResetTransform();
g.Dispose();
g = null;
```

简单的 GDI+处理图片大小(C#代码)

```
        /// <summary>
        ///  缩放图片
        /// </summary>
        /// <param name="img">原图片</param>
        /// <param name="xWith">缩放宽比例,如果想缩小图片,小于 100</param>
        /// <param name="yHeight">缩放高比例</param>
        /// <returns>返回处理后图片</returns>
        public Image scaleImg(System.Drawing.Image img, int xWith, int yHeight)
        {
            //计算处理后图片宽
            int i = Convert.ToInt32(img.Width * xWith / 100);
            //计算处理后图片高
            int j = Convert.ToInt32(img.Height * yHeight / 100);
            //格式化图片
            System.Drawing.Image imgScale = new System.Drawing.Bitmap(i, j, System.Drawing.Imaging.PixelFormat.Format24bppRgb);
            System.Drawing.Graphics g = System.Drawing.Graphics.FromImage(imgScale);
            System.Drawing.Rectangle srcRect = new System.Drawing.Rectangle(0, 0, img.Width, img.Height);
            System.Drawing.Rectangle desRect = new System.Drawing.Rectangle(0, 0, imgScale.Width, imgScale.Height);
            g.Clear(System.Drawing.Color.White);
            g.DrawImage(img, desRect, srcRect, System.Drawing.GraphicsUnit.Pixel);
            //处理后的图片另存
            imgScale.Save("E:\\1111.jpg", System.Drawing.Imaging.ImageFormat.Gif);
            g.Dispose();
            return imgScale;
        }
```

方法 4 中获取文件扩展名可以使用：

```
System.IO.Path.GetExtension(strOldFilePath)
```

而不需要：

```
strExtension = strOldFilePath.Substring(strOldFilePath.LastIndexOf("."));
```

另外 6 中防止文件重名，我觉得直接使用 GUID 就可以了，不需要再加其他东西

ASP.NET2.0 打通文件图片处理任督二脉[转载]

ASP.NET2.0 打通文件图片处理任督二脉

作者：清清月儿

主页：http://blog.csdn.net/21aspnet/　　时间：2007.4.1

1.最简单的单文件上传**(**没花头**)**

2.多文件上传

3.客户端检查上传文件类型**(**以上传图片为例**)**

4.服务器端检查上传文件类型**(**以上传图片为例**)**

5.服务器端检查上传文件类型**(**可以检测真正文件名**)**

6.上传文件文件名唯一性处理**(**时间戳**+SessionID)**

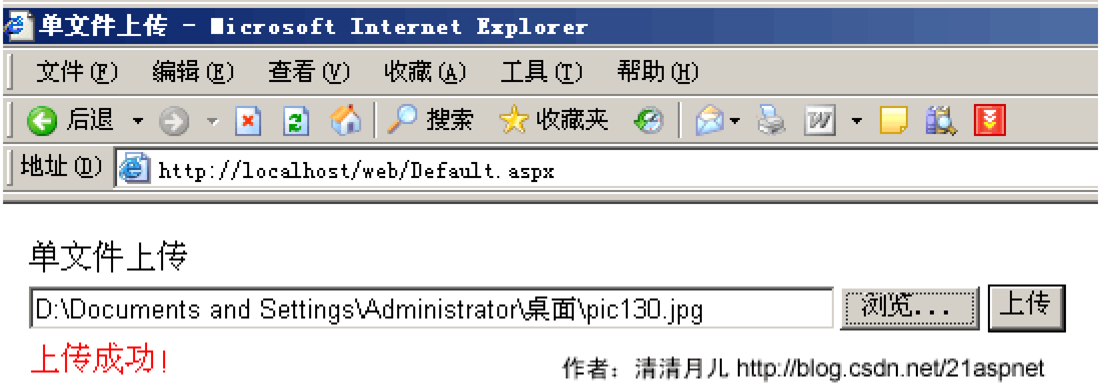
7.上传图片生成等比例缩略图

8.上传图片加水印**(**文字水印，图片水印，文字**+**图片水印**)**

9.

1.最简单的单文件上传**(**没花头**)**

效果图：


单文件上传 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/web/Default.aspx

单文件上传

D:\Documents and Settings\Administrator\桌面\pic130.jpg

上传成功!

作者：清清月儿 http://blog.csdn.net/21aspnet

说明：这是最基本的文件上传，在 asp.net1.x 中没有这个 FileUpload 控件，只有 html 的上传控件，那时候要把 html 控件转化为服务器控件，很不好用。其实所有文件上传的美丽效果都是从这个 FileUpload 控件衍生，第一个例子虽然简单却是根本。

后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                this.lb_info.Text = "请选择文件！ ";
            }
            else
            {
```

```
│               string filepath = FileUpload1.PostedFile.FileName;
│               string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
│               string serverpath = Server.MapPath("images/") + filename;
│               FileUpload1.PostedFile.SaveAs(serverpath);
│               this.lb_info.Text = "上传成功！";
├           }
├       }
│       catch (Exception ex)
        {
│           this.lb_info.Text = "上传发生错误！原因是：" + ex.ToString();
├       }
├   }
└}
```

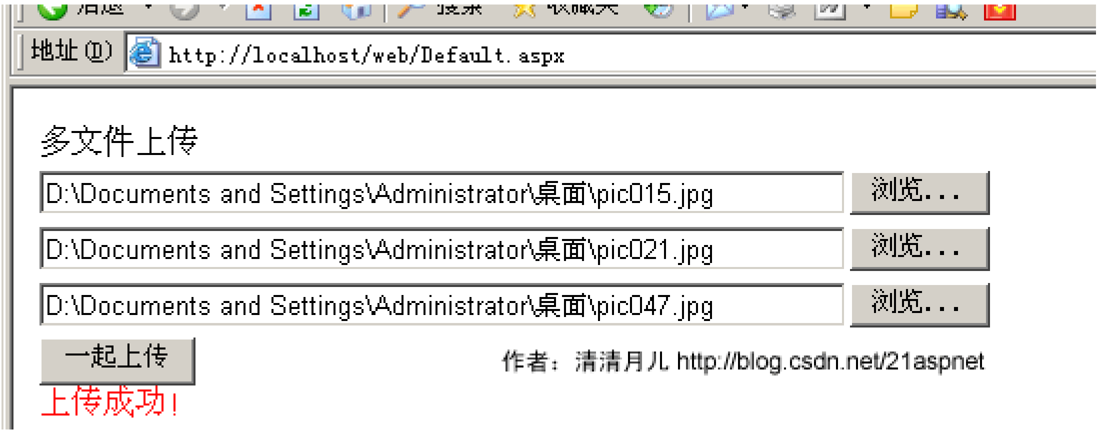
前台代码：

```
<table style="width: 343px">
        <tr>
            <td style="width: 100px">
                单文件上传</td>
            <td style="width: 100px">
            </td>
        </tr>
        <tr>
            <td style="width: 100px">
                <asp:FileUpload ID="FileUpload1" runat="server" Width="475px" />
                </td>
            <td style="width: 100px">
```

```
                <asp:Button ID="bt_upload" runat="server" OnClick="bt_upload_Click" Text="上传" /></td>
            </tr>
            <tr>
                <td style="width: 100px; height: 21px;">
                    <asp:Label ID="lb_info" runat="server" ForeColor="Red" Width="183px"></asp:Label></td>
                <td style="width: 100px; height: 21px">
                </td>
            </tr>
        </table>
```

2.多文件上传

效果图：



地址(D) http://localhost/web/Default.aspx

多文件上传

D:\Documents and Settings\Administrator\桌面\pic015.jpg

D:\Documents and Settings\Administrator\桌面\pic021.jpg

D:\Documents and Settings\Administrator\桌面\pic047.jpg

作者：清清月儿 http://blog.csdn.net/21aspnet

上传成功！

后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {

        if ((FileUpload1.PostedFile.FileName == "" && FileUpload2.PostedFile.FileName == "")&&FileUpload3.PostedFile.FileName == "")
        {
            this.lb_info.Text = "请选择文件！ ";
        }
        else
        {
            HttpFileCollection myfiles = Request.Files;
            for (int i = 0; i < myfiles.Count; i++)
            {
                HttpPostedFile mypost = myfiles[i];
                try
                {
                    if (mypost.ContentLength > 0)
                    {
```

```
                    string filepath = mypost.FileName;
                    string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                    string serverpath = Server.MapPath("images/") + filename;
                    mypost.SaveAs(serverpath);
                    this.lb_info.Text = "上传成功！";
                }
            }
            catch (Exception error)
            {
                this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
            }

        }

    }
}

}
```

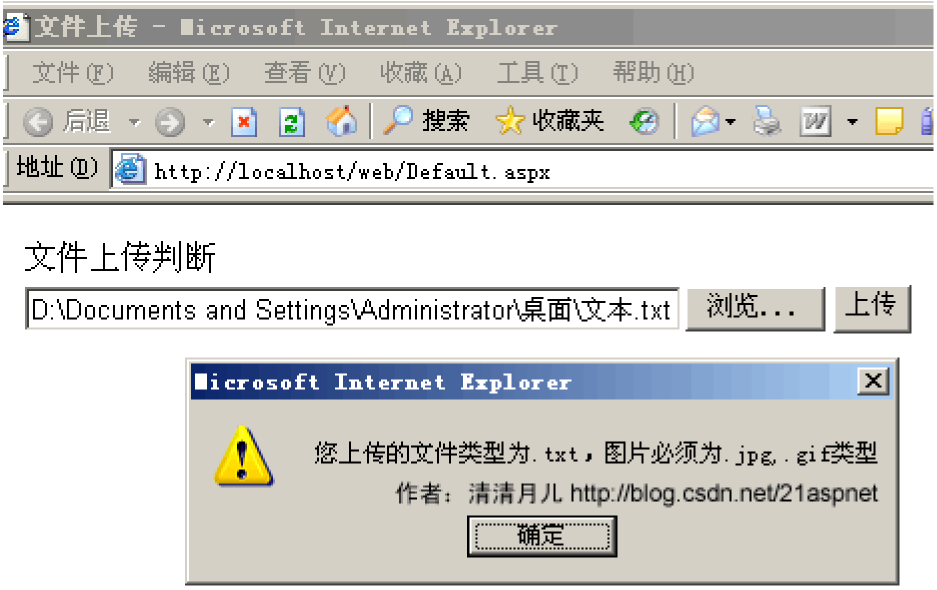
前台代码：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits
="_Default" %>

 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/T
R/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>多文件上传 清清月儿 http://blog.csdn.net/21aspnet/</title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        <table style="width: 343px">

            <tr>

                <td style="width: 100px">

                    多文件上传</td>

                <td style="width: 100px">

                </td>

            </tr>

            <tr>

                <td style="width: 100px">

                    <asp:FileUpload ID="FileUpload1" runat="server" Width="475px" />

                    </td>

                <td style="width: 100px">

                    </td>

            </tr>

            <tr>

                <td style="width: 100px">

                    <asp:FileUpload ID="FileUpload2" runat="server" Width="475px" /></td>

                <td style="width: 100px">

                </td>

            </tr>

            <tr>

                <td style="width: 100px">

                    <asp:FileUpload ID="FileUpload3" runat="server" Width="475px" /></td>
                <td style="width: 100px">
                </td>
            </tr>
            <tr>
                <td style="width: 100px">
                    <asp:Button ID="bt_upload" runat="server" OnClick="bt_upload_Click" Text="一起上传" />
                    <asp:Label ID="lb_info" runat="server" ForeColor="Red" Width="183px"></asp:Label></td>
                <td style="width: 100px">
                </td>
            </tr>
        </table>

    </div>
    </form>
</body>
</html>
```

3.客户端检查上传文件类型**(**以上传图片为例**)**

效果图：


文件上传 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/web/Default.aspx
浏览...
上传
Microsoft Internet Explorer
确定

文件上传判断

D:\Documents and Settings\Administrator\桌面\文本.txt

您上传的文件类型为.txt，图片必须为.jpg,.gif类型

作者：清清月儿 http://blog.csdn.net/21aspnet

后台代码和 **1.**最简单的单文件上传一样；

前台代码：

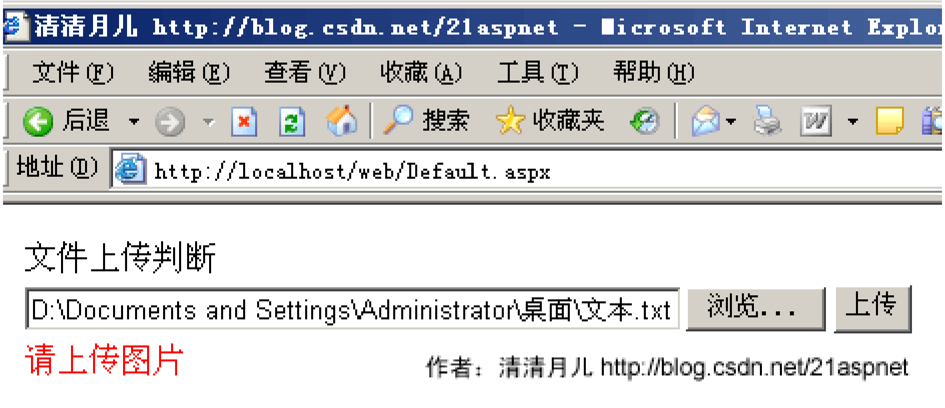
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits
="_Default" %>


  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/T
R/xhtml1/DTD/xhtml1-transitional.dtd">


  <html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
     <title>清清月儿 http://blog.csdn.net/21aspnet</title>
<script   language="javascript">
 function Check_FileType()
{
var str=document.getElementById("FileUpload1").value;
 var pos = str.lastIndexOf(".");
 var lastname = str.substring(pos,str.length)
 if (lastname.toLowerCase()!=".jpg" && lastname.toLowerCase()!=".gif")
{
    alert("您上传的文件类型为"+lastname+"，图片必须为.jpg,.gif 类型");
    return false;
}
else
{
  return true;
}
}
</script>

  </head>
  <body>
     <form id="form1" runat="server">
     <div>
         <table style="width: 343px">
             <tr>
                 <td style="width: 104px">
                     文件上传判断</td>
                 <td style="width: 100px">
                 </td>
             </tr>
             <tr>
                 <td style="width: 104px">
                     <asp:FileUpload ID="FileUpload1" runat="server" Width="400px" />
                     </td>
                 <td style="width: 100px">
                     <asp:Button ID="bt_upload" runat="server" OnClick="bt_upload_Click" Text="
上传"  OnClientClick="return Check_FileType()"/></td>
             </tr>
```

```
            <tr>
                <td style="width: 104px; height: 21px;">
                    <asp:Label ID="lb_info" runat="server" ForeColor="Red" Width="183px"></asp:Label></td>
                <td style="width: 100px; height: 21px">
                </td>
            </tr>
        </table>

    </div>
    </form>
</body>
</html>
```

说明：点击上传时先触发客户端事件 Check_FileType；

4.服务器端检查上传文件类型(以上传图片为例)

效果图：


清清月儿 http://blog.csdn.net/21aspnet - Microsoft Internet Explor
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/web/Default.aspx
浏览...
上传

文件上传判断

D:\Documents and Settings\Administrator\桌面\文本.txt

请上传图片

作者：清清月儿 http://blog.csdn.net/21aspnet

后台代码：

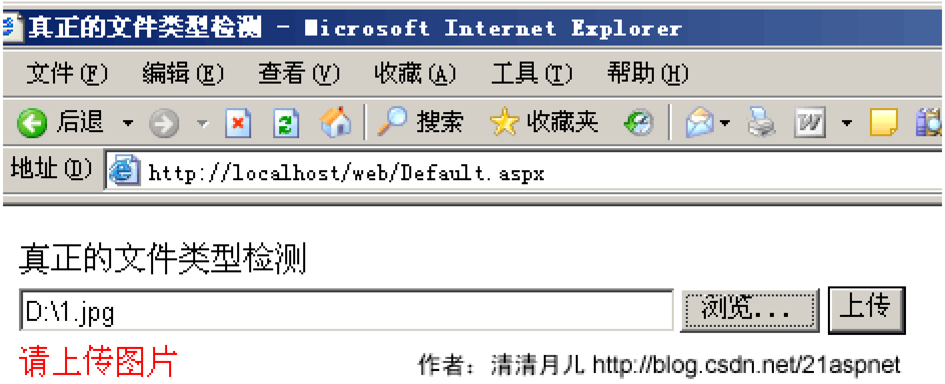
```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                this.lb_info.Text = "请选择文件！ ";
            }
            else
            {
                string filepath = FileUpload1.PostedFile.FileName;
                if (IsAllowedExtension(FileUpload1) == true)
                {

                    string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                    string serverpath = Server.MapPath("images/") + filename;
                    FileUpload1.PostedFile.SaveAs(serverpath);
                    this.lb_info.Text = "上传成功！";
                }
                else
                {
                    this.lb_info.Text = "请上传图片";
                }
            }
        }
        catch (Exception error)
        {
            this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
        }
    }
public static bool IsAllowedExtension(FileUpload hifile)
    {
        string strOldFilePath = "", strExtension = "";
        string[] arrExtension =  { ".gif", ".jpg", ".jpeg", ".bmp", ".png" };
        if (hifile.PostedFile.FileName != string.Empty)
        {
            strOldFilePath = hifile.PostedFile.FileName;
            strExtension = strOldFilePath.Substring(strOldFilePath.LastIndexOf("."));
            for (int i = 0; i < arrExtension.Length; i++)
            {
                if (strExtension.Equals(arrExtension[i]))
                {
                    return true;
                }
            }
```

```
├      }
│      return false;
├   }
│
└}
```

5.服务器端检查上传文件类型(可以检测真正文件名)

其实方法 4 并不好，因为用户可以把 XXX.txt 伪装为 XXX.jpg。

效果图：


真正的文件类型检测 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D)
浏览...
上传

http://localhost/web/Default.aspx

真正的文件类型检测

D:\1.jpg

请上传图片

作者：清清月儿 http://blog.csdn.net/21aspnet

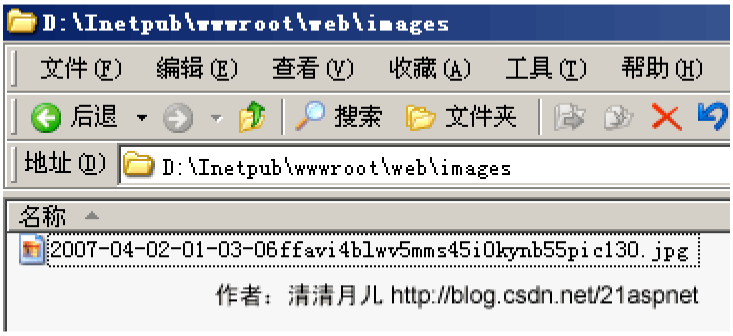
后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    //清清月儿 http://blog.csdn.net/21aspnet
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                this.lb_info.Text = "请选择文件！ ";
            }
            else
            {
                string filepath = FileUpload1.PostedFile.FileName;
                if (IsAllowedExtension(FileUpload1) == true)
                {
                    string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                    string serverpath = Server.MapPath("images/") + filename;
                    FileUpload1.PostedFile.SaveAs(serverpath);
                    this.lb_info.Text = "上传成功！ ";
                }
                else
                {
                    this.lb_info.Text = "请上传图片";
                }
            }
        }
        catch (Exception error)
        {
            this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
        }
    }
    public static bool IsAllowedExtension(FileUpload hifile)
    {
        System.IO.FileStream fs = new System.IO.FileStream(hifile.PostedFile.FileName, System.IO.FileMode.Open, System.IO.FileAccess.Read);
        System.IO.BinaryReader r = new System.IO.BinaryReader(fs);
        string fileclass = "";
        byte buffer;
        try
        {
            buffer = r.ReadByte();
            fileclass = buffer.ToString();
            buffer = r.ReadByte();
            fileclass += buffer.ToString();

        }
        catch
        {

        }
        r.Close();
        fs.Close();
        if (fileclass == "255216" || fileclass == "7173")//说明 255216 是jpg;7173 是gif;6677 是B
```

```
MP,13780 是PNG;7790 是exe,8297 是rar
            {
                return true;
            }
            else
            {
                return false;
            }

        }

}
```

6.上传文件文件名唯一性处理**(**时间戳**+SessionID)**

效果图：



D:\Inetpub\wwwroot\web\images
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 文件夹
地址(D) D:\Inetpub\wwwroot\web\images
名称

2007-04-02-01-03-06ffavi4blwv5mms45i0kynb55pic130.jpg

作者：清清月儿 http://blog.csdn.net/21aspnet

说明：年月日时分秒+临时 session+原文件名 如果大家怕还会重复可以加 GUID

后台代码：

```
    try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
              this.lb_info.Text = "请选择文件！";
          }
          else
            {
              string filepath = FileUpload1.PostedFile.FileName;
              string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
              string serverpath = Server.MapPath("images/") + System.DateTime.Now.ToString
("yyy-MM-dd-hh-mm-ss") + Session.SessionID + filename;
              FileUpload1.PostedFile.SaveAs(serverpath);
              this.lb_info.Text = "上传成功！";
          }
      }
      catch (Exception error)
        {
          this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
      }
```

注：GUID 的方法：Guid myGuid=Guid.NewGuid();

7.上传图片生成等比例缩略图

效果图：

缩略图代码：

```
ImageThumbnail.cs
using System;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;

public class ImageThumbnail
{
    public Image ResourceImage;
    private int ImageWidth;
    private int ImageHeight;
    public string ErrorMessage;

    public ImageThumbnail(string ImageFileName)
    {
        ResourceImage = Image.FromFile(ImageFileName);
        ErrorMessage = "";
    }

```

```
public bool ThumbnailCallback()
 {
   return false;
}


// 方法 1，按大小
public bool ReducedImage(int Width, int Height, string targetFilePath)
 {
   try
     {
       Image ReducedImage;
       Image.GetThumbnailImageAbort callb = new Image.GetThumbnailImageAbort(ThumbnailCallback);
       ReducedImage = ResourceImage.GetThumbnailImage(Width, Height, callb, IntPtr.Zero);
       ReducedImage.Save(@targetFilePath, ImageFormat.Jpeg);
       ReducedImage.Dispose();
       return true;
   }
   catch (Exception e)
     {
       ErrorMessage = e.Message;
       return false;
   }
}


// 方法 2，按百分比  缩小 60% Percent 为 0.6 targetFilePath 为目标路径
public bool ReducedImage(double Percent, string targetFilePath)
```

后台代码：

```
{
    try
    {
        Image ReducedImage;
        Image.GetThumbnailImageAbort callb = new Image.GetThumbnailImageAbort(ThumbnailCallback);
        ImageWidth = Convert.ToInt32(ResourceImage.Width * Percent);
        ImageHeight = (ResourceImage.Height)*ImageWidth/ ResourceImage.Width;//等比例缩放
        ReducedImage = ResourceImage.GetThumbnailImage(ImageWidth, ImageHeight, callb, IntPtr.Zero);
        ReducedImage.Save(@targetFilePath, ImageFormat.Jpeg);
        ReducedImage.Dispose();
        return true;
    }
    catch (Exception e)
    {
        ErrorMessage = e.Message;
        return false;
    }
  }


}
```

后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class _Default : System.Web.UI.Page
{

    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                this.lb_info.Text = "请选择文件！ ";
            }
            else
            {
                string filepath = FileUpload1.PostedFile.FileName;
                string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                string serverpath1 = Server.MapPath("images/") + filename;
                string serverpath2 = Server.MapPath("images/") + System.DateTime.Now.ToString("yyy-MM-dd-hh-mm-ss") + Session.SessionID + filename;
                FileUpload1.PostedFile.SaveAs(serverpath1);
                ImageThumbnail img = new ImageThumbnail(filepath);
                img.ReducedImage(0.4, serverpath2);//0.4 表示缩小 40%
                this.lb_info.Text = "上传成功！";
            }
        }
        catch (Exception error)
        {
            this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
        }
    }


}
```

8.上传图片加水印**(**文字水印，图片水印，文字**+**图片水印**)**

效果图：

原图

水印

给图片加水印以后（注意右上角+正下方）

代码：

DrawImg.cs 出自**http://www.codeproject.com/csharp/watermark.asp**

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
public class DrawImg
{
 private string  WorkingDirectory = string.Empty ; //路径
 private string  ImageName = string.Empty;   //被处理的图片
 private string  ImageWater = string.Empty;  //水印图片
 private string  FontString = string.Empty;  //水印文字


   enum DealType{NONE,WaterImage,WaterFont,DoubleDo}; //枚举命令

 private DealType dealtype;

```

```

public DrawImg()
{}

public string PublicWorkingDirectory
{
 get
 {
  return WorkingDirectory;
 }
 set
 {
  WorkingDirectory = value;
 }
}

public string PublicImageName
{
 get
 {
  return ImageName;
 }
 set
 {
  ImageName = value;
 }
}


public string PublicImageWater
```

```
{
 get
  {
  return ImageWater;
 }
 set //设置了水印图片的话说明是要水印图片效果的
  {
  dealtype = DealType.WaterImage;
  ImageWater = value;
 }
}

public string PublicFontString
{
 get
  {
  return FontString;
 }
 set //设置了水印文字的话说明是要水印文字效果的
  {
  dealtype = DealType.WaterFont;
  FontString = value;
 }
}



public void DealImage()
{
 IsDouble();
```

```

  switch( dealtype )
   {
    case DealType.WaterFont: WriteFont(); break;
    case DealType.WaterImage: WriteImg(); break;
    case DealType.DoubleDo: WriteFontAndImg(); break;
  }

}

private void IsDouble()
{
  if(ImageWater+""!="" && FontString+""!="")
   {
          dealtype = DealType.DoubleDo;
  }
}

private void WriteFont()
{
  //set a working directory
  //string WorkingDirectory = @"C:\Watermark_src\WaterPic";

  //define a string of text to use as the Copyright message
  //string Copyright = "Copyright ?2002 - AP Photo/David Zalubowski";

  //create a image object containing the photograph to watermark
  Image imgPhoto = Image.FromFile(WorkingDirectory + ImageName);
  int phWidth = imgPhoto.Width;
  int phHeight = imgPhoto.Height;
```

```

//create a Bitmap the Size of the original photograph
Bitmap bmPhoto = new Bitmap(phWidth, phHeight, PixelFormat.Format24bppRgb);

bmPhoto.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);

//load the Bitmap into a Graphics object
Graphics grPhoto = Graphics.FromImage(bmPhoto);

//------------------------------------------------------------
//Step #1 - Insert Copyright message
//------------------------------------------------------------

//Set the rendering quality for this Graphics object
grPhoto.SmoothingMode = SmoothingMode.AntiAlias;

//Draws the photo Image object at original size to the graphics object.
grPhoto.DrawImage(
 imgPhoto,                               // Photo Image object
 new Rectangle(0, 0, phWidth, phHeight), // Rectangle structure
 0,                                      // x-coordinate of the portion of the source image to draw.
 0,                                      // y-coordinate of the portion of the source image to draw.
 phWidth,                                // Width of the portion of the source image to draw.
 phHeight,                               // Height of the portion of the source image to draw.
 GraphicsUnit.Pixel);                    // Units of measure

//------------------------------------------------------
//to maximize the size of the Copyright message we will
//test multiple Font sizes to determine the largest posible
//font we can use for the width of the Photograph
```

```
//define an array of point sizes you would like to consider as possiblities
//-------------------------------------------------------
 int[] sizes = new int[]{16,14,12,10,8,6,4};

Font crFont = null;
SizeF crSize = new SizeF();

//Loop through the defined sizes checking the length of the Copyright string
//If its length in pixles is less then the image width choose this Font size.
for (int i=0 ;i<7; i++)
 {
 //set a Font object to Arial (i)pt, Bold
 //crFont = new Font("arial", sizes[i], FontStyle.Bold);

 crFont = new Font("arial",sizes[i],FontStyle.Bold);

 //Measure the Copyright string in this Font
 crSize = grPhoto.MeasureString(FontString, crFont);

 if((ushort)crSize.Width < (ushort)phWidth)
  break;
}

//Since all photographs will have varying heights, determine a
//position 5% from the bottom of the image
int yPixlesFromBottom = (int)(phHeight *.05);

//Now that we have a point size use the Copyrights string height
//to determine a y-coordinate to draw the string of the photograph
float yPosFromBottom = ((phHeight - yPixlesFromBottom)-(crSize.Height/2));
```

```

//Determine its x-coordinate by calculating the center of the width of the image
float xCenterOfImg = (phWidth/2);

//Define the text layout by setting the text alignment to centered
StringFormat StrFormat = new StringFormat();
StrFormat.Alignment = StringAlignment.Center;

//define a Brush which is semi trasparent black (Alpha set to 153)
SolidBrush semiTransBrush2 = new SolidBrush(Color.FromArgb(153, 0, 0, 0));

//Draw the Copyright string
grPhoto.DrawString(FontString,                //string of text
 crFont,                                      //font
 semiTransBrush2,                             //Brush
 new PointF(xCenterOfImg+1,yPosFromBottom+1),  //Position
 StrFormat);

//define a Brush which is semi trasparent white (Alpha set to 153)
SolidBrush semiTransBrush = new SolidBrush(Color.FromArgb(153, 255, 255, 255));

//Draw the Copyright string a second time to create a shadow effect
//Make sure to move this text 1 pixel to the right and down 1 pixel
grPhoto.DrawString(FontString,                //string of text
 crFont,                                      //font
 semiTransBrush,                              //Brush
 new PointF(xCenterOfImg,yPosFromBottom),  //Position
 StrFormat);

imgPhoto = bmPhoto;
```

```
grPhoto.Dispose();

//save new image to file system.
imgPhoto.Save(WorkingDirectory + ImageName + "_finally.jpg", ImageFormat.Jpeg);
imgPhoto.Dispose();

//Text alignment
}


private void WriteImg()
{
//set a working directory
//string WorkingDirectory = @"C:\Watermark_src\WaterPic";

//create a image object containing the photograph to watermark
Image imgPhoto = Image.FromFile(WorkingDirectory + ImageName);
int phWidth = imgPhoto.Width;
int phHeight = imgPhoto.Height;

//create a Bitmap the Size of the original photograph
Bitmap bmPhoto = new Bitmap(phWidth, phHeight, PixelFormat.Format24bppRgb);

bmPhoto.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);

//load the Bitmap into a Graphics object
Graphics grPhoto = Graphics.FromImage(bmPhoto);

//create a image object containing the watermark
Image imgWatermark = new Bitmap(WorkingDirectory + ImageWater);
```

```
int wmWidth = imgWatermark.Width;
int wmHeight = imgWatermark.Height;

//Set the rendering quality for this Graphics object
grPhoto.SmoothingMode = SmoothingMode.AntiAlias;

//Draws the photo Image object at original size to the graphics object.
grPhoto.DrawImage(
 imgPhoto,                               // Photo Image object
 new Rectangle(0, 0, phWidth, phHeight), // Rectangle structure
 0,                                      // x-coordinate of the portion of the source image to draw.
 0,                                      // y-coordinate of the portion of the source image to draw.
 phWidth,                                // Width of the portion of the source image to draw.
 phHeight,                               // Height of the portion of the source image to draw.
 GraphicsUnit.Pixel);                    // Units of measure


//------------------------------------------------------------
//Step #2 - Insert Watermark image
//------------------------------------------------------------

//Create a Bitmap based on the previously modified photograph Bitmap
Bitmap bmWatermark = new Bitmap(bmPhoto);
bmWatermark.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);
//Load this Bitmap into a new Graphic Object
Graphics grWatermark = Graphics.FromImage(bmWatermark);

//To achieve a translucent watermark we will apply (2) color
//manipulations by defineing a ImageAttributes object and
```

```
//seting (2) of its properties.
ImageAttributes imageAttributes = new ImageAttributes();

//The first step in manipulating the watermark image is to replace
//the background color with one that is trasparent (Alpha=0, R=0, G=0, B=0)
//to do this we will use a Colormap and use this to define a RemapTable
ColorMap colorMap = new ColorMap();

//My watermark was defined with a background of 100% Green this will
//be the color we search for and replace with transparency
colorMap.OldColor = Color.FromArgb(255, 0, 255, 0);
colorMap.NewColor = Color.FromArgb(0, 0, 0, 0);

 ColorMap[] remapTable = {colorMap};

imageAttributes.SetRemapTable(remapTable, ColorAdjustType.Bitmap);

//The second color manipulation is used to change the opacity of the
//watermark.  This is done by applying a 5x5 matrix that contains the
//coordinates for the RGBA space.  By setting the 3rd row and 3rd column
//to 0.3f we achive a level of opacity
 float[][] colorMatrixElements = {
        new float[] {1.0f,  0.0f,  0.0f,  0.0f, 0.0f},
        new float[] {0.0f,  1.0f,  0.0f,  0.0f, 0.0f},
        new float[] {0.0f,  0.0f,  1.0f,  0.0f, 0.0f},
        new float[] {0.0f,  0.0f,  0.0f,  0.3f, 0.0f},
        new float[] {0.0f,  0.0f,  0.0f,  0.0f, 1.0f}};
ColorMatrix wmColorMatrix = new ColorMatrix(colorMatrixElements);

imageAttributes.SetColorMatrix(wmColorMatrix, ColorMatrixFlag.Default,
 ColorAdjustType.Bitmap);

//For this example we will place the watermark in the upper right
//hand corner of the photograph. offset down 10 pixels and to the
//left 10 pixles

int xPosOfWm = ((phWidth - wmWidth)-10);
int yPosOfWm = 10;

grWatermark.DrawImage(imgWatermark,
 new Rectangle(xPosOfWm,yPosOfWm,wmWidth,wmHeight),  //Set the detination Position
 0,                  // x-coordinate of the portion of the source image to draw.
 0,                  // y-coordinate of the portion of the source image to draw.
 wmWidth,            // Watermark Width
 wmHeight,      // Watermark Height
 GraphicsUnit.Pixel, // Unit of measurment
 imageAttributes);   //ImageAttributes Object

//Replace the original photgraphs bitmap with the new Bitmap
imgPhoto = bmWatermark;
grPhoto.Dispose();
grWatermark.Dispose();

//save new image to file system.
imgPhoto.Save(WorkingDirectory + ImageName +"_finally.jpg", ImageFormat.Jpeg);
imgPhoto.Dispose();
imgWatermark.Dispose();

}

```

```

private void WriteFontAndImg()
{

  //create a image object containing the photograph to watermark
  Image imgPhoto = Image.FromFile(WorkingDirectory + ImageName);
  int phWidth = imgPhoto.Width;
  int phHeight = imgPhoto.Height;

  //create a Bitmap the Size of the original photograph
  Bitmap bmPhoto = new Bitmap(phWidth, phHeight, PixelFormat.Format24bppRgb);

  bmPhoto.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);

  //load the Bitmap into a Graphics object
  Graphics grPhoto = Graphics.FromImage(bmPhoto);

  //create a image object containing the watermark
  Image imgWatermark = new Bitmap(WorkingDirectory + ImageWater);
  int wmWidth = imgWatermark.Width;
  int wmHeight = imgWatermark.Height;

  //------------------------------------------------------------
  //Step #1 - Insert Copyright message
  //------------------------------------------------------------

  //Set the rendering quality for this Graphics object
  grPhoto.SmoothingMode = SmoothingMode.AntiAlias;

  //Draws the photo Image object at original size to the graphics object.
```

```
grPhoto.DrawImage(
 imgPhoto,                             // Photo Image object
 new Rectangle(0, 0, phWidth, phHeight), // Rectangle structure
 0,                                    // x-coordinate of the portion of the source image to draw.
 0,                                    // y-coordinate of the portion of the source image to draw.
 phWidth,                              // Width of the portion of the source image to draw.
 phHeight,                             // Height of the portion of the source image to draw.
 GraphicsUnit.Pixel);                  // Units of measure

//------------------------------------------------------
//to maximize the size of the Copyright message we will
//test multiple Font sizes to determine the largest posible
//font we can use for the width of the Photograph
//define an array of point sizes you would like to consider as possiblities
//------------------------------------------------------
 int[] sizes = new int[]{16,14,12,10,8,6,4};

Font crFont = null;
SizeF crSize = new SizeF();

//Loop through the defined sizes checking the length of the Copyright string
//If its length in pixles is less then the image width choose this Font size.
for (int i=0 ;i<7; i++)
 {
 //set a Font object to Arial (i)pt, Bold
 crFont = new Font("arial", sizes[i], FontStyle.Bold);
 //Measure the Copyright string in this Font
 crSize = grPhoto.MeasureString(FontString, crFont);

 if((ushort)crSize.Width < (ushort)phWidth)
```

```
  break;
}

//Since all photographs will have varying heights, determine a
//position 5% from the bottom of the image
int yPixlesFromBottom = (int)(phHeight *.05);

//Now that we have a point size use the Copyrights string height
//to determine a y-coordinate to draw the string of the photograph
float yPosFromBottom = ((phHeight - yPixlesFromBottom)-(crSize.Height/2));

//Determine its x-coordinate by calculating the center of the width of the image
float xCenterOfImg = (phWidth/2);

//Define the text layout by setting the text alignment to centered
StringFormat StrFormat = new StringFormat();
StrFormat.Alignment = StringAlignment.Center;

//define a Brush which is semi trasparent black (Alpha set to 153)
SolidBrush semiTransBrush2 = new SolidBrush(Color.FromArgb(153, 0, 0, 0));

//Draw the Copyright string
grPhoto.DrawString(FontString,             //string of text
 crFont,                                   //font
 semiTransBrush2,                          //Brush
 new PointF(xCenterOfImg+1,yPosFromBottom+1),  //Position
 StrFormat);

//define a Brush which is semi trasparent white (Alpha set to 153)
SolidBrush semiTransBrush = new SolidBrush(Color.FromArgb(153, 255, 255, 255));
```

```
//Draw the Copyright string a second time to create a shadow effect
//Make sure to move this text 1 pixel to the right and down 1 pixel
grPhoto.DrawString(FontString,                //string of text
 crFont,                                  //font
 semiTransBrush,                          //Brush
 new PointF(xCenterOfImg,yPosFromBottom),  //Position
 StrFormat);                              //Text alignment



//------------------------------------------------------------
//Step #2 - Insert Watermark image
//------------------------------------------------------------

//Create a Bitmap based on the previously modified photograph Bitmap
Bitmap bmWatermark = new Bitmap(bmPhoto);
bmWatermark.SetResolution(imgPhoto.HorizontalResolution, imgPhoto.VerticalResolution);
//Load this Bitmap into a new Graphic Object
Graphics grWatermark = Graphics.FromImage(bmWatermark);

//To achieve a transulcent watermark we will apply (2) color
//manipulations by defineing a ImageAttributes object and
//seting (2) of its properties.
ImageAttributes imageAttributes = new ImageAttributes();

//The first step in manipulating the watermark image is to replace
//the background color with one that is trasparent (Alpha=0, R=0, G=0, B=0)
//to do this we will use a Colormap and use this to define a RemapTable
```

```
ColorMap colorMap = new ColorMap();

//My watermark was defined with a background of 100% Green this will
//be the color we search for and replace with transparency
colorMap.OldColor = Color.FromArgb(255, 0, 255, 0);
colorMap.NewColor = Color.FromArgb(0, 0, 0, 0);

 ColorMap[] remapTable = {colorMap};

imageAttributes.SetRemapTable(remapTable, ColorAdjustType.Bitmap);

//The second color manipulation is used to change the opacity of the
//watermark.  This is done by applying a 5x5 matrix that contains the
//coordinates for the RGBA space.  By setting the 3rd row and 3rd column
//to 0.3f we achive a level of opacity
 float[][] colorMatrixElements = {
         new float[] {1.0f,  0.0f,  0.0f,  0.0f, 0.0f},
         new float[] {0.0f,  1.0f,  0.0f,  0.0f, 0.0f},
         new float[] {0.0f,  0.0f,  1.0f,  0.0f, 0.0f},
         new float[] {0.0f,  0.0f,  0.0f,  0.3f, 0.0f},
         new float[] {0.0f,  0.0f,  0.0f,  0.0f, 1.0f}};
ColorMatrix wmColorMatrix = new ColorMatrix(colorMatrixElements);

imageAttributes.SetColorMatrix(wmColorMatrix, ColorMatrixFlag.Default,
 ColorAdjustType.Bitmap);

//For this example we will place the watermark in the upper right
//hand corner of the photograph. offset down 10 pixels and to the
//left 10 pixles

```

```
 int xPosOfWm = ((phWidth - wmWidth)-10);

 int yPosOfWm = 10;


 grWatermark.DrawImage(imgWatermark,

  new Rectangle(xPosOfWm,yPosOfWm,wmWidth,wmHeight),  //Set the detination Position

  0,                  // x-coordinate of the portion of the source image to draw.

  0,                  // y-coordinate of the portion of the source image to draw.

  wmWidth,            // Watermark Width

  wmHeight,      // Watermark Height

  GraphicsUnit.Pixel, // Unit of measurment

  imageAttributes);   //ImageAttributes Object


 //Replace the original photgraphs bitmap with the new Bitmap

 imgPhoto = bmWatermark;

 grPhoto.Dispose();

 grWatermark.Dispose();


 //save new image to file system.

 imgPhoto.Save(WorkingDirectory + ImageName +"_finally.jpg", ImageFormat.Jpeg);

 imgPhoto.Dispose();

 imgWatermark.Dispose();




}

}




 //水印图片加水印文字

//   ReDrawImg img = new ReDrawImg();

//   img .PublicWorkingDirectory = @"C:\Watermark_src\WaterPic\";
//   img .PublicImageName = "watermark_photo.jpg";

//   img .PublicImageWater = "watermark.bmp";

//   img .PublicFontString = "清清月儿";

//   img .DealImage();


  //水印文字

  ReDrawImg img = new ReDrawImg();

  img .PublicWorkingDirectory = @"C:\Watermark_src\WaterPic\";

  img .PublicImageName = "watermark_photo.jpg";

  img .PublicFontString = @"清清月儿";

  img .DealImage();



  //水印图片

//   ReDrawImg img = new ReDrawImg();

//   img .PublicWorkingDirectory = @"C:\Watermark_src\WaterPic\";

//   img .PublicImageName = "watermark_photo.jpg";

//   img .PublicImageWater = "watermark.bmp";

//   img .DealImage();
```

后台代码：

```
using System;

using System.Data;

using System.Configuration;

using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class _Default : System.Web.UI.Page
{

    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void bt_upload_Click(object sender, EventArgs e)
    {
        try
        {
            if (FileUpload1.PostedFile.FileName == "")
            {
                this.lb_info.Text = "请选择文件！";
            }
            else
            {
                string filepath = FileUpload1.PostedFile.FileName;
                string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1);
                string serverpath1 = Server.MapPath("images/") + filename;
                //string serverpath2 = Server.MapPath("images/") + System.DateTime.Now.ToString("yyy-MM-dd-hh-mm-ss") + Session.SessionID + filename;
                FileUpload1.PostedFile.SaveAs(serverpath1);
                //ImageThumbnail img = new ImageThumbnail(filepath);
                //img.ReducedImage(0.4, serverpath2);
                DrawImg img = new DrawImg();
                img.PublicWorkingDirectory = Server.MapPath("images/");
                img.PublicImageName = filename;
                img.PublicFontString = "http://blog.csdn.net/21aspnet";
                img.PublicImageWater = "yyy.jpg";
                img.DealImage();
                this.lb_info.Text = "上传成功！";
            }
        }
        catch (Exception error)
        {
            this.lb_info.Text = "上传发生错误！原因：" + error.ToString();
        }
    }

}
```

此文未完，待续

OWC 组件使用

asp.net 2.0 中，要显示图型的话，可以用 ms office 2003 的 owc 组件，可以十分方便地看到图表，在工程中，

首先添加 microsoft office web components 11.0 的引用就可以了，然后要

```
using Microsoft.Office.Interop.Owc11;
```

1 生成柱状图

```
//创建 X 坐标的值，表示月份

        int[] Month = new int[3] { 1, 2, 3 };

        //创建 Y 坐标的值，表示销售额

        double[] Count = new double[3] { 120,240,220};

        //创建图表空间

        ChartSpace mychartSpace = new ChartSpace();

        //在图表空间内添加一个图表对象

        ChChart mychart = mychartSpace.Charts.Add(0);

        //设置图表类型，本例使用柱形

        mychart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;

        //设置图表的一些属性

        //是否需要图例
```

```
mychart.HasLegend = true;

//是否需要主题

mychart.HasTitle = true;

//主题内容

mychart.Title.Caption = "一季度总结";

//设置 x,y 坐标

mychart.Axes[0].HasTitle = true;

mychart.Axes[0].Title.Caption = "月份";

mychart.Axes[1].HasTitle = true;

mychart.Axes[1].Title.Caption = "销量";

//添加三个图表块

mychart.SeriesCollection.Add(0);

mychart.SeriesCollection.Add(0);

mychart.SeriesCollection.Add(0);

//设置图表块的属性

//标题

mychart.SeriesCollection[0].Caption = "一月份";

//X 坐标的值属性
```

```
mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[0]);

//y 坐标的值属性

mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[0]);

//第二个块

mychart.SeriesCollection[1].Caption = "二月份";

//X 坐标的值属性

mychart.SeriesCollection[1].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[1]);

//y 坐标的值属性

mychart.SeriesCollection[1].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[1]);

//第三个块

mychart.SeriesCollection[2].Caption = "三月份";

//X 坐标的值属性

mychart.SeriesCollection[2].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[2]);

//y 坐标的值属性
```

```
        mychart.SeriesCollection[2].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[2]);

        //生成图片

        mychartSpace.ExportPicture(Server.MapPath(".") + @"\test.jpg", "jpg", 500, 450);

        //加载图片

        Image1.ImageUrl = Server.MapPath(".") + @"\test.jpg";

    }
```

2　生成饼状图

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

    {

        //创建X坐标的值，表示月份

        int[] Month ={ 1, 2, 3 };

        //创建Y坐标的值，表示销售额

        double[] Count ={ 120, 240, 220 };

        string strDataName = "";

        string strData = "";

        //创建图表空间

        ChartSpace mychartSpace = new ChartSpace();
```

```
//在图表空间内添加一个图表对象

ChChart mychart = mychartSpace.Charts.Add(0);

//设置每块饼的数据

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

{

    strDataName += Month[i] + "\t";

    strData += Count[i].ToString() + "\t";

}

//设置图表类型，本例使用柱形

mychart.Type = ChartChartTypeEnum.chChartTypePie;

//设置图表的一些属性

//是否需要图例

mychart.HasLegend = true;

//是否需要主题

mychart.HasTitle = true;

//主题内容

mychart.Title.Caption = "一季度总结";

//添加图表块
```

```
            mychart.SeriesCollection.Add(0);

            //设置图表块的属性

            //分类属性

            mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories,
(int)ChartSpecialDataSourcesEnum.chDataLiteral, strDataName);

            //值属性

            mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues,
(int)ChartSpecialDataSourcesEnum.chDataLiteral, strData);

           //显示百分比

            ChDataLabels mytb= mychart.SeriesCollection[0].DataLabelsCollection.Add();

            mytb.HasPercentage = true;

            //生成图片

            mychartSpace.ExportPicture(Server.MapPath(".") + @"\test.gif", "gif", 500, 450);

            //加载图片

            Image1.ImageUrl = Server.MapPath(".") + @"\test.gif";

      }
```

一个 OWC 开发范例

最近在开发一个报告生成系统，其中使用到了.NET 图表第三方控件。以下是使用 OWC9 画一个BAR 图的范例：

```
public class ChartClass

 {

   public ChartClass()

   {

   }

public void GetChart(string category,string value1,string value2,string

title,System.Web.HttpServerUtility Server,System.Web.UI.WebControls.PlaceHolder

ChartHolder,string filename,string sdate,string edate,bool y)

   {

     OWC.ChartSpace objCSpace = new OWC.ChartSpaceClass ();

     OWC.WCChart objChart = objCSpace.Charts.Add (0);

     objChart.Type = OWC.ChartChartTypeEnum.chChartTypeBarStacked;    //图表类型

     objChart.HasLegend = true;

     objChart.HasTitle = true;

     if(y==true)
```

```
    objChart.Title.Caption= title + " 前十二个月广告主花费 (" + sdate + " 至 " + edate +
")";
   else
    objChart.Title.Caption= title + " 上个月广告主花费 (" + sdate + " 至 " + edate + ")";
   objChart.Axes[0].HasTitle = false;
   objChart.Axes[1].HasTitle = false;
   objChart.SeriesCollection.Add(0);
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimSeriesNames,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, "报纸");
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimCategories,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, category);
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimValues,(int)OWC.ChartSpecialDataSourcesEnum.chDat
aLiteral,
value1);
   objChart.SeriesCollection.Add(1);
```

```
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimSeriesNames,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, "杂志");
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimCategories,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, category);
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimValues,(int)OWC.ChartSpecialDataSourcesEnum.chDat
aLiteral,
value2);
    object ob = "#66FF66";
    objChart.SeriesCollection[0].Interior.set_Color(ref ob);
//不知道为什么，VB 里，可以使用 Interior.Color="#66FF66"来直接设置颜色，而 C#里必须使
用 set_Color 方法，而且参数必须是 ref
object。
    objChart.SeriesCollection[0].Border.set_Color(ref ob);
    ob="#6699ff";
    objChart.SeriesCollection[1].Interior.set_Color(ref ob);
    objChart.SeriesCollection[1].Border.set_Color(ref ob);
objChart.SeriesCollection[0].Border.set_Weight(OWC.LineWeightEnum.owcLineWeightThin
);

objChart.SeriesCollection[1].Border.set_Weight(OWC.LineWeightEnum.owcLineWeightThin
);

    ob="#FFFFFF";

    objChart.PlotArea.Border.set_Color(ref ob);

    objChart.PlotArea.Interior.set_Color(ref ob);

    objChart.Axes[0].Font.set_Name("Arial");

    objChart.Axes[0].Font.set_Size(9);

    objChart.Axes[1].MajorTickMarks=OWC.ChartTickMarkEnum.chTickMarkInside;

    objChart.Axes[1].MinorTickMarks=OWC.ChartTickMarkEnum.chTickMarkNone;

    ob="#CCCCCC";

    objChart.Axes[0].MajorGridlines.Line.set_Color(ref ob);

    objChart.Axes[1].Font.set_Name("宋体");

    objChart.Axes[1].Font.set_Size(9);

//      objChart.SeriesCollection[0].DataLabelsCollection.Add();

//      objChart.SeriesCollection[1].DataLabelsCollection.Add();

//      objChart.SeriesCollection[0].DataLabelsCollection[0].HasValue=true;
//     ob="red";

//     objChart.SeriesCollection[0].DataLabelsCollection[0].Font.set_Color(ref

ob);

//     objChart.SeriesCollection[1].DataLabelsCollection[0].Font.set_Color(ref

ob);

//

objChart.SeriesCollection[0].DataLabelsCollection[0].Position=OWC.ChartDataLabelPositio
nEnum.chLabelPositionOutsideBase;

//

objChart.SeriesCollection[1].DataLabelsCollection[0].Position=OWC.ChartDataLabelPosition
Enum.chLabelPositionOutsideEnd;

//上面几段注释掉的内容，是在数据柱上显示数据值的设置代码



    objChart.Legend.Font.set_Name("宋体");

    objChart.Legend.Font.set_Size(9);

    objChart.Title.Font.set_Name("宋体");

    objChart.Title.Font.set_Size(11);

    string strAbsolutePath = Server.MapPath(".")+ @"\TempFiles\" + filename;

//文件保存的位置
```

```
     objCSpace.ExportPicture(strAbsolutePath, "GIF", 600, 350);    //文件的分辨率其他属性

     string strRelativePath = "./TempFiles/" + filename;

     string strImageTag = "<IMG SRC='" + strRelativePath + "'/>";

     ChartHolder.Controls.Add(new LiteralControl(strImageTag));

//将图片填充到 PlaceHolder 对象 ChartHoler 中。

   }

  }
```

OWC 学习笔记-Spreadsheet 插入行/列

在 owc 提供的 Spreadsheet api 中，没有直接添加行列的方法，可以使用执行命令的方式实现

添加新行在第 3 行，代码如下：

```
     var ssConstants = Spreadsheet1.Constants;

     Spreadsheet1.ActiveSheet.Row(3).Select();

     Spreadsheet1.Commands(ssConstants.ssCommandInsertRows).Execute();
```

添加新列在第 3 列，代码如下：

```
     var ssConstants = Spreadsheet1.Constants;

     Spreadsheet1.ActiveSheet.cells(2,3).Select();

     Spreadsheet1.Commands(ssConstants.ssCommandInsertCols).Execute();
```

C#编写 OWC11 组件源代码(2)

```
using System;

using System.Collections;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Web;

using System.Web.SessionState;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;

using Microsoft.Office.Interop;

namespace FishPro

{

  /// <summary>

  ///  利用 OWC 组件生成柱状图示例

  /// </summary>

  public class TestOWC : System.Web.UI.Page
```

```
 {
  private void Page_Load(object sender, System.EventArgs e)
  {
    string strCategory = "1" + '\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "5" + '\t' + "6" + '\t'+"7" + '\t' + "8" + '\t' + "9" + '\t'+"10" + '\t' + "11" + '\t' + "12" + '\t';
    string strValue = "9000" + '\t' + "8000" + '\t' + "4007" + '\t'+"10" + '\t' + "12760" + '\t' + "6678" + '\t'+"10000" + '\t' + "20999" + '\t' + "3567" + '\t'+"456" + '\t' + "125" + '\t' + "66765" + '\t';
    string mTitle="建科院月报表分析图";
    string xTitle="月份";
    string yTitle="工作量";
    int imgWidth=780;
    int imgHeight=600;
    int chartType=0;

//this.CreateChartSmoothLine(strCategory,strValue,mTitle,xTitle,yTitle,imgWidth,imgHeight,chartType);
    FishPro.OWCChart11 chart = new OWCChart11(Server.MapPath("."),"费用",mTitle,1,xTitle,yTitle,imgWidth,imgHeight);
    chart.OCategory=strCategory;
    chart.OValue=strValue;

    if(chart.Create())

    {

      Response.Write( "<IMG SRC='"+chart.FileName+"'/>");

    }

    else

    {

      Response.Write("shibai");

    }

//     // 在此处放置用户代码以初始化页面

//     string strCategory = "1" + '\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "5" + '\t' + "6" + '\t'+"7"
+ '\t' + "8" + '\t' + "9" + '\t'+"10" + '\t' + "11" + '\t' + "12" + '\t';

//     string strValue = "9" + '\t' + "8" + '\t' + "4" + '\t'+"10" + '\t' + "12" + '\t' + "6" + '\t'+"1" +
'\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "12" + '\t' + "6" + '\t';

//

//     //声明对象

//     Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new
Microsoft.Office.Interop.Owc11.ChartSpaceClass();

//     Microsoft.Office.Interop.Owc11.ChChart ThisChChart    = ThisChart.Charts.Add(0);
//      Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries = ThisChChart.SeriesCollection.Add(0);

//

//

//      //显示图例

//      ThisChChart.HasLegend = true;

//      //标题

//      ThisChChart.HasTitle = true;

//      ThisChChart.Title.Caption = "统计图";

//      //给定 x,y 轴图示说明

//      ThisChChart.Axes[0].HasTitle = true;

//      ThisChChart.Axes[1].HasTitle = true;

//      ThisChChart.Axes[0].Title.Caption = "月份";

//      ThisChChart.Axes[1].Title.Caption = "数量";

//

//      //图表类型

//      //ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered3D;//3D 柱状图
//      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeArea; //折线面积图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeArea3D;//折线 3D 面积图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaOverlapped3D;//折线 3D 面积图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked;//折线面积图加边框

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked100;//折线面积图加边框百分比图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked1003D;//折线 3D 面积图加边框百分比图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked3D;//折线 3D 面积图加边框
```

```
////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarClustered;//横道图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarClustered3D;//横道图 3D

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked;//横道图 3D

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked100;//横道图 3D 百分比图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked1003D;//横道图 3D 百分比图

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBubble; //测试不通过

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBubbleLine;//测试不通过

////      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D
```

```
////     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered;//柱状图 3D

////     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered3D;//柱状图 3D

////     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked;//柱状图

////     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked100;//柱状图 3D  百分比图

////     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked1003D;//柱状图 3D 百分比图

//     //ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartLegendPositionEnum.chLegendPositionBottom;

//     //旋转

//     ThisChChart.Rotation    = 360;

//     ThisChChart.Inclination = 10;

//     //背景颜色

//     ThisChChart.PlotArea.Interior.Color = "red";

//     ThisChChart.PlotArea.Floor.Interior.Color = "green";
```

```
//
//      ThisChChart.Overlap = 50;
//
//      /////给定 series 的名字
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesNames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),"日期");
//      //给定分类
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategories,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strCategory);
//      //给定值
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strValue);
//
//      //导出图像文件
//      try
//     {
//
//       ThisChart.ExportPicture(Server.MapPath("chart.gif"),"gif",600,350);
//       Response.Write( "<IMG SRC='chart.gif'/>");
//     }
//
//     catch(Exception ee)
//
//     {
//
//     }
   }
   //平滑曲线图
   //ChartType 0  默认柱状图  1  横道图  2  平滑曲线图
   public    void    CreateChartSmoothLine(string strCategory,string strValue,string
mTitle,string xTitle,string yTitle,int imgWidth,int imgHeight,int chartType)
   {
     //声明对象
```

```
Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new Microsoft.Office.Interop.Owc11.ChartSpaceClass();
Microsoft.Office.Interop.Owc11.ChChart ThisChChart  = ThisChart.Charts.Add(0);
Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries = ThisChChart.SeriesCollection.Add(0);

//显示图例
ThisChChart.HasLegend = true;
//显示标题选项
ThisChChart.HasTitle = true;
ThisChChart.Title.Font.Name="黑体";
ThisChChart.Title.Font.Size=14;
ThisChChart.Title.Caption = mTitle;//from
//x,y 轴说明
//x
ThisChChart.Axes[0].HasTitle=true;
ThisChChart.Axes[0].Title.Font.Name="黑体";
ThisChChart.Axes[0].Title.Font.Size=12;
ThisChChart.Axes[0].Title.Caption=xTitle;
```

```
    ThisChChart.Axes[1].HasTitle=true;

    ThisChChart.Axes[1].Title.Font.Name="黑体";

    ThisChChart.Axes[1].Title.Font.Size=12;

    ThisChChart.Axes[1].Title.Caption=yTitle;

    //图表类型

    switch(chartType)

    {

      case 0:

        ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D

        break;

      case 1:

        ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D

        break;

      case 2:

        ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图
      break;
     case 3:
      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypePie;//圆饼图
      break;
    }
    //ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线
图
    //旋转
    ThisChChart.Rotation  = 360;
    ThisChChart.Inclination = 10;
    //背景颜色
    ThisChChart.PlotArea.Interior.Color = "red";
    ThisChChart.PlotArea.Floor.Interior.Color = "green";
    //ThisChChart.Overlap = 50;
    /////给定 series 的名字

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesN
ames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHash
Code(),"日期");

   //给定分类

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategori
es,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCod
e(),strCategory);

   //给定值

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,
Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode()
,strValue);

   //导出图像文件

   try

   {

    ThisChart.ExportPicture(Server.MapPath("chart.gif"),"gif",imgWidth,imgHeight);

    Response.Write( "<IMG SRC='chart.gif'/>");

   }

   catch(Exception ee)

   {
   }
  }
  #region Web 窗体设计器生成的代码
  override protected void OnInit(EventArgs e)
  {
   //
   // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
   //
   InitializeComponent();
   base.OnInit(e);
  }
  /// <summary>
  /// 设计器支持所需的方法 - 不要使用代码编辑器修改
  /// 此方法的内容。
  /// </summary>
  private void InitializeComponent()
  {
   this.Load += new System.EventHandler(this.Page_Load);
```

```
   }

   #endregion

  }

}
```

使用方法：

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

    {

        MyOWC1.DataSource = createTable();

        MyOWC1.Attribute.KeyField = "name";

        MyOWC1.Attribute.ValueFileds = new string[] { "vistis", "showname" };

        MyOWC1.Attribute.signValues = new string[] { "美国", "中国" };

        MyOWC1.Attribute.startDate = new DateTime(2007, 4, 1);

        MyOWC1.Attribute.endDate = new DateTime(2007, 4, 19);

        MyOWC1.Draw();

    }

    DataTable createTable()

    {
    }

        Random rnd = new Random();

        DataTable dt = new DataTable();

        dt.Columns.Add("name");

        dt.Columns.Add("vistis");

        dt.Columns.Add("showname");

        dt.Columns.Add("percent");

        for (int i = 0; i < 8; i++)

        {

            DataRow dr = dt.NewRow();

            dr["name"] = "2007-4-"+(i+1);

            dr["vistis"] = rnd.Next(100, 1000).ToString();

            dr["showname"] = rnd.Next(100, 1000).ToString();

            dr["percent"] = rnd.Next(100, 1000).ToString();

            dt.Rows.Add(dr);

        }

        return dt;

    }
```

C#编写OWC11 组件(转贴)

```
 using System;

using System.Data;

using System.Text;



namespace FishPro

         {

            /// <summary>

    /// 使用 OWCChart11 生成各种图表

    ///

    ///

    /// </summary>

    public class OWCChart11

             {

                #region 属性

        private string m_SavePath;

        private string m_Category;

        private string m_Value;

        private DataTable m_DataSource;

        private string m_SeriesName;

        private string m_Title;

        private string m_AxesXTitle;
private string m_AxesYTitle;

private int m_PicWidth;

private int m_PicHeight;

private int m_Type;

private string m_FileName;



        /// <summary>

/// 保存图片的路径和名称,物理路径

/// </summary>

public string SavePath

        {

            get{return m_SavePath;}

            set{m_SavePath=value;}

}



        /// <summary>

/// 直接获得类型

/// </summary>

public string OCategory

        {

            get{return m_Category;}

            set{m_Category=value;}
```

```
}

        /// <summary>
/// 直接获得值
/// </summary>
public string OValue
        {
            get{return m_Value;}
            set{m_Value=value;}
}



        /// <summary>
/// 以表格 DataTable 的形式获取原始数据
/// </summary>
public DataTable DataSource
        {
            get{return m_DataSource;}
   set
            {
       m_DataSource=value;
       m_Category=GetColumnsStr(m_DataSource);
```

```
        m_Value=GetValueStr(m_DataSource);
    }
}

        /// <summary>
/// 简要说明
/// </summary>
public string SeriesName
        {
            get{return m_SeriesName;}
            set{m_SeriesName=value;}
}

        /// <summary>
/// 图表的总标题，说明图表的简单意思
/// </summary>
public string Title
        {
            get{return m_Title;}
            set{m_Title=value;}
}
```

```
    /// <summary>
/// 图表横坐标标题，说明横坐标的意义
/// </summary>
public string AxesXTitle
    {
      get{return m_AxesXTitle;}
      set{m_AxesXTitle=value;}
}

    /// <summary>
/// 图表纵坐标标题，说明纵坐标的意义
/// </summary>
public string AxesYTitle
    {
      get{return m_AxesYTitle;}
      set{m_AxesYTitle=value;}
}

    /// <summary>
///  生成的图片宽度
/// </summary>
public int PicWidth
```

```
        {
            get{return m_PicWidth;}
            set{m_PicWidth=value;}
}

        /// <summary>
///  生成的图片高度
/// </summary>
public int PicHeight
        {
            get{return m_PicHeight;}
            set{m_PicHeight=value;}
}

        /// <summary>
/// 类型
/// chChartTypeColumnStacked100 =2
///chChartTypeColumnStacked1003D = 49
///chChartTypeColumnStacked3D = 48
///chChartTypeCombo = -1
///chChartTypeCombo3D = -2
///chChartTypeDoughnut = 32
```

```
///chChartTypeDoughnutExploded = 33

///chChartTypeLine = 6

///chChartTypeLine3D = 54

///chChartTypeLineMarkers=  7

///chChartTypeLineOverlapped3D=  55

///chChartTypeLineStacked = 8

///chChartTypeLineStacked100  =10

///chChartTypeLineStacked1003D=  57

///chChartTypeLineStacked100Markers = 11

///chChartTypeLineStacked3D = 56

///chChartTypeLineStackedMarkers = 9

///chChartTypePie = 18

///chChartTypePie3D =58

///chChartTypePieExploded = 19

///chChartTypePieExploded3D = 59

///chChartTypePieStacked = 20

///chChartTypePolarLine = 42

///chChartTypePolarLineMarkers = 43

///chChartTypePolarMarkers = 41

///chChartTypePolarSmoothLine = 44

///chChartTypePolarSmoothLineMarkers = 45

///chChartTypeRadarLine=  34
```

```
///chChartTypeRadarLineFilled = 36
///chChartTypeRadarLineMarkers=  35
///chChartTypeRadarSmoothLine = 37
///chChartTypeRadarSmoothLineMarkers = 38
///chChartTypeScatterLine = 25
///chChartTypeScatterLineFilled = 26
///chChartTypeScatterLineMarkers = 24
///chChartTypeScatterMarkers = 21
///chChartTypeScatterSmoothLine = 23
///chChartTypeScatterSmoothLineMarkers = 22
///chChartTypeSmoothLine = 12
///chChartTypeSmoothLineMarkers = 13
///chChartTypeSmoothLineStacked = 14
///chChartTypeSmoothLineStacked100 = 16
///chChartTypeSmoothLineStacked100Markers = 17
///chChartTypeSmoothLineStackedMarkers = 15
///chChartTypeStockHLC = 39
///chChartTypeStockOHLC = 40
/// </summary>
public int Type
    {
      get{return m_Type;}
```

```
            set{m_Type=value;}
        }


        public string FileName
            {
                get{return m_FileName;}
          set
                {
              m_FileName=value;
              if(m_FileName==null||m_FileName=="")
                    {
                  m_FileName=DateTime.Now.ToString("yyyyMMddHHmmssff")+".gif";//生成
图片名称
              }
          }
        }


            /// <summary>
        /// 是 DataTable 的转换
        /// </summary>
        /// <param name="dt"></param>
        /// <returns></returns>
```

```
private string GetColumnsStr(DataTable dt)
    {
  StringBuilder strList=new StringBuilder();
  foreach(DataRow r in dt.Rows)
      {
    strList.Append(r[0].ToString()+'\t');
  }
  return strList.ToString();
}

    /// <summary>
/// 是 DataTable 的转换
/// </summary>
/// <param name="dt"></param>
/// <returns></returns>
private string GetValueStr(DataTable dt)
    {
  StringBuilder strList=new StringBuilder();
  foreach(DataRow r in dt.Rows)
      {
    strList.Append(r[1].ToString()+'\t');
  }
```

```
    return strList.ToString();

}



#endregion



         #region 枚举类型

         /// <summary>

/// 枚举类型 对应于 OWC 的图表类型

/// </summary>

public  enum ChartType:int

         {



             /// <summary>

    /// 面积图

    /// </summary>

    chChartTypeArea = 29 ,

             /// <summary>

    /// 面积图 3D

    /// </summary>

    chChartTypeArea3D = 60 ,

             /// <summary>

    /// 面积图重复
    /// </summary>
    chChartTypeAreaOverlapped3D = 61 ,
    /// <summary>
    /// 堆积面积图
    /// </summary>
    chChartTypeAreaStacked  =30 ,
    /// <summary>
    /// 堆积面积图百分比图
    /// </summary>
    chChartTypeAreaStacked100 = 31 ,
             /// <summary>
    /// 堆积面积图百分比图 3D
    /// </summary>
    chChartTypeAreaStacked1003D  =63 ,
             /// <summary>
    /// 堆积面积图 3D
    /// </summary>
    chChartTypeAreaStacked3D = 62 ,
             /// <summary>
    /// 横道图 3D
    /// </summary>
    chChartTypeBar3D = 50 ,
            /// <summary>
    /// 横道图串风格
    /// </summary>
    chChartTypeBarClustered=  3 ,
            /// <summary>
    /// 横道图串风格 3D
    /// </summary>
    chChartTypeBarClustered3D=  51 ,
            /// <summary>
    /// 堆横道图
    /// </summary>
    chChartTypeBarStacked = 4 ,
            /// <summary>
    /// 堆横道图百分比图
    /// </summary>
    chChartTypeBarStacked100  =5 ,
            /// <summary>
    /// 堆横道图百分比图 3D
    /// </summary>
    chChartTypeBarStacked1003D = 53 ,
            /// <summary>
    /// 堆横道图 3D
    /// </summary>

    chChartTypeBarStacked3D  =52 ,

            /// <summary>

    /// 气泡图

    /// </summary>

    chChartTypeBubble = 27 ,

            /// <summary>

    /// 线形气泡图

    /// </summary>

    chChartTypeBubbleLine =28 ,

    /// <summary>

    /// 柱形图

    /// </summary>

    chChartTypeColumn3D  =46 ,

    /// <summary>

    /// 3D 柱形图

    /// </summary>

    chChartTypeColumnClustered = 0 ,

    /// <summary>

    /// 3D 串柱形图

    /// </summary>

    chChartTypeColumnClustered3D = 47 ,
    /// <summary>
    /// 重叠柱形图
    /// </summary>
    chChartTypeColumnStacked =1 ,
    /// <summary>
    /// 100%重叠柱形图
    /// </summary>
    chChartTypeColumnStacked100 =2 ,
    /// <summary>
    /// 100%3D 重叠柱形图
    /// </summary>
    chChartTypeColumnStacked1003D = 49 ,
    /// <summary>
    /// 3D 柱形图
    /// </summary>
    chChartTypeColumnStacked3D = 48 ,
    /// <summary>
    /// 组合图
    /// </summary>
    chChartTypeCombo = -1 ,
    /// <summary>
    /// 3D 组合图
    /// </summary>
    chChartTypeCombo3D = -2 ,
    /// <summary>
    /// 环形图
    /// </summary>
    chChartTypeDoughnut = 32 ,
    /// <summary>
    /// 破式环形图
    /// </summary>
    chChartTypeDoughnutExploded = 33 ,
    /// <summary>
    /// 折线图
    /// </summary>
    chChartTypeLine = 6,
    /// <summary>
    /// 3D 折线图
    /// </summary>
    chChartTypeLine3D = 54 ,
    /// <summary>
    /// 制造折线图
    /// </summary>
    chChartTypeLineMarkers=  7,
    /// <summary>
    /// 重复折线图
    /// </summary>
    chChartTypeLineOverlapped3D=  55,
    /// <summary>
    /// 重叠折线图
    /// </summary>
    chChartTypeLineStacked = 8 ,
    /// <summary>
    /// 100%重叠折线图
    /// </summary>
    chChartTypeLineStacked100  =10 ,
    /// <summary>
    /// 100%3D 重叠折线图
    /// </summary>
    chChartTypeLineStacked1003D=  57,
    /// <summary>
    /// 制造 100%重叠折线图
    /// </summary>
    chChartTypeLineStacked100Markers = 11 ,
    /// <summary>
    /// 3D 重叠折线图
    /// </summary>
    chChartTypeLineStacked3D = 56 ,
    /// <summary>
    /// 制造重叠折线图
    /// </summary>
    chChartTypeLineStackedMarkers = 9 ,
    /// <summary>
    /// 饼图
    /// </summary>
    chChartTypePie = 18 ,
    /// <summary>
    /// 3D 饼图
    /// </summary>
    chChartTypePie3D =58 ,
    /// <summary>
    /// 破式饼图
    /// </summary>
    chChartTypePieExploded = 19 ,
    /// <summary>
    /// 3D 破式饼图
    /// </summary>
    chChartTypePieExploded3D = 59 ,
    /// <summary>

    /// 重叠饼图

    /// </summary>

    chChartTypePieStacked = 20 ,

    /// <summary>

    /// 极坐标图

    /// </summary>

    chChartTypePolarLine = 42 ,

    /// <summary>

    /// 制造线形极坐标图

    /// </summary>

    chChartTypePolarLineMarkers = 43,

    /// <summary>

    /// 制造极坐标图

    /// </summary>

    chChartTypePolarMarkers = 41 ,

    /// <summary>

    /// 平滑线形极坐标图

    /// </summary>

    chChartTypePolarSmoothLine = 44 ,

    /// <summary>

    /// 制造平滑线形极坐标图
    /// </summary>
    chChartTypePolarSmoothLineMarkers = 45 ,
            /// <summary>
    /// 雷达图
    /// </summary>
    chChartTypeRadarLine=  34 ,
            /// <summary>
    /// 填充雷达图
    /// </summary>
    chChartTypeRadarLineFilled = 36 ,
            /// <summary>
    /// 制造雷达图
    /// </summary>
    chChartTypeRadarLineMarkers=  35 ,
            /// <summary>
    /// 平滑雷达图
    /// </summary>
    chChartTypeRadarSmoothLine = 37 ,
            /// <summary>
    /// 制造平滑雷达图
    /// </summary>
    chChartTypeRadarSmoothLineMarkers = 38 ,
            /// <summary>
    /// 线形散点图
    /// </summary>
    chChartTypeScatterLine = 25 ,
            /// <summary>
    /// 填充线形散点图
    /// </summary>
    chChartTypeScatterLineFilled = 26 ,
            /// <summary>
    /// 制造线形散点图
    /// </summary>
    chChartTypeScatterLineMarkers = 24 ,
            /// <summary>
    /// 制造散点图
    /// </summary>
    chChartTypeScatterMarkers = 21 ,
            /// <summary>
    /// 平滑散点图
    /// </summary>
    chChartTypeScatterSmoothLine = 23 ,
            /// <summary>
    /// 制造平滑散点图
    /// </summary>
    chChartTypeScatterSmoothLineMarkers = 22 ,
            /// <summary>
    /// 平滑线图
    /// </summary>
    chChartTypeSmoothLine = 12 ,
            /// <summary>
    /// 制造平滑线图
    /// </summary>
    chChartTypeSmoothLineMarkers = 13 ,
            /// <summary>
    /// 重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked = 14 ,
            /// <summary>
    /// 100%重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked100 = 16 ,
            /// <summary>
    /// 制造 100%重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked100Markers = 17 ,
        /// <summary>
    /// 制造重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStackedMarkers = 15 ,
        /// <summary>
    /// 股价图
    /// </summary>
    chChartTypeStockHLC = 39 ,
        /// <summary>
    /// 股价图 O 型
    /// </summary>
    chChartTypeStockOHLC = 40

}
#endregion

     #region 构造函数
public OWCChart11()
     {
    //
    // TODO: 在此处添加构造函数逻辑
    //
```

```
        }

        public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType)
            {
          m_SavePath=SavePath;
          m_SeriesName=SeriesName;
          m_Title=Title;
          m_Type=ChartType;
        }

         public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType,string AxesXTitle,string AxesYTitle)
            {
          m_SavePath=SavePath;
          m_SeriesName=SeriesName;
          m_Title=Title;
          m_AxesXTitle=AxesXTitle;
          m_AxesYTitle=AxesYTitle;
          m_Type=ChartType;
        }

         public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType,string AxesXTitle,string AxesYTitle,int PicWidth,int PicHeight)
```

```
      {
   m_SavePath=SavePath;
   m_SeriesName=SeriesName;
   m_Title=Title;
   m_AxesXTitle=AxesXTitle;
   m_AxesYTitle=AxesYTitle;
   m_PicWidth=PicWidth;
   m_PicHeight=PicHeight;
   m_Type=ChartType;
}
#endregion


public bool Create()
      {
   //声明对象
   Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new  Microsoft.Office.Interop.Owc11.ChartSpaceClass();
   Microsoft.Office.Interop.Owc11.ChChart ThisChChart  = ThisChart.Charts.Add(0);
   Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries = ThisChChart.SeriesCollection.Add(0);


   //显示图例
   ThisChChart.HasLegend = true;
```

```
            //显示标题选项

            ThisChChart.HasTitle = true;

            ThisChChart.Title.Font.Name="黑体";

            ThisChChart.Title.Font.Size=14;

            ThisChChart.Title.Caption = m_Title;//from


            //x,y 轴说明

            //x

            ThisChChart.Axes[0].HasTitle=true;

            ThisChChart.Axes[0].Title.Font.Name="黑体";

            ThisChChart.Axes[0].Title.Font.Size=12;

            ThisChChart.Axes[0].Title.Caption=m_AxesXTitle;


            ThisChChart.Axes[1].HasTitle=true;

            ThisChChart.Axes[1].Title.Font.Name="黑体";

            ThisChChart.Axes[1].Title.Font.Size=12;

            ThisChChart.Axes[1].Title.Caption=m_AxesYTitle;


            //图表类型

             ThisChChart.Type=(Microsoft.Office.Interop.Owc11.ChartChartTypeEnum) m_Type;

//           switch(m_Type)
```

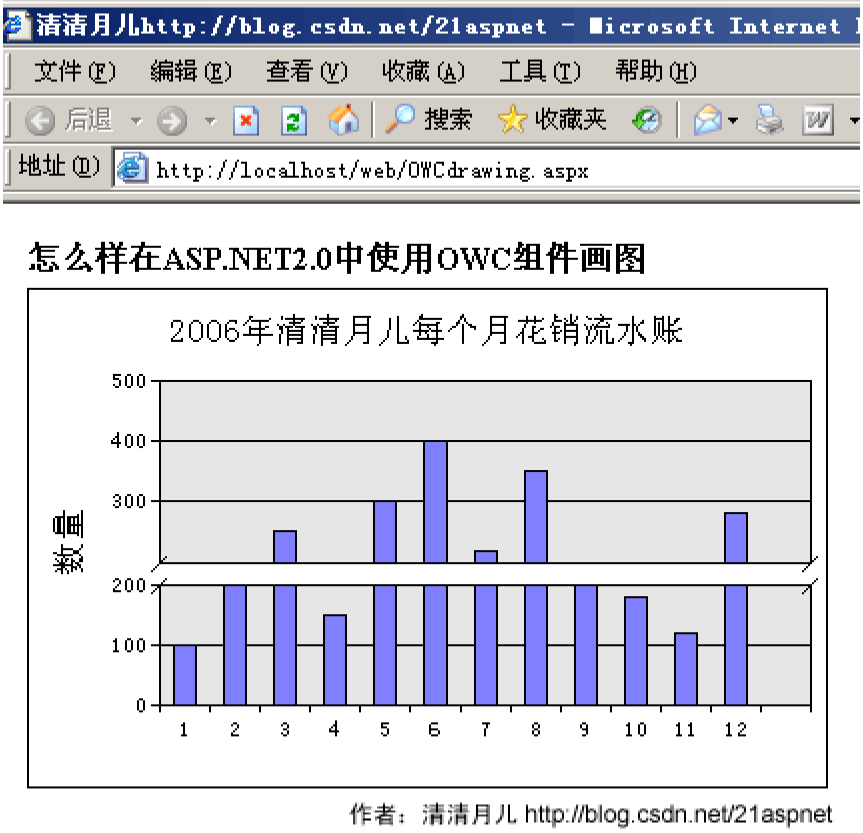
```
     //        {
     //          case 0:
//            ThisChChart.Type =(Microsoft.Office.Interop.Owc11.ChartChartTypeEnum) m_Type;// Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D
     //            break;
     //          case 1:
//            ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D
     //            break;
     //          case 2:
//            ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图
     //            break;
     //          case 3:
//            ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypePie;//圆饼图
     //
     //
     //            break;
     //        }
            //旋转
            ThisChChart.Rotation  = 360;
            ThisChChart.Inclination = 10;

            //背景颜色

            ThisChChart.PlotArea.Interior.Color = "red";

            //底座颜色

            ThisChChart.PlotArea.Floor.Interior.Color = "green";



            //ThisChChart.Overlap = 50;



            //给定 series 的名字

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesNames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_SeriesName);

            //给定分类

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategories,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_Category);

            //给定值

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_Value);



             Microsoft.Office.Interop.Owc11.ChDataLabels dl=ThisChChart.SeriesCollection[0].DataLabelsCollection.Add();

            dl.HasValue=true;

            //dl.HasPercentage=true;
```

```
                    //导出图像文件
                    try
                            {
                        if(m_FileName==null||m_FileName=="")
                                {
                            m_FileName=DateTime.Now.ToString("yyyyMMddHHmmssff")+".gif";//生成图片名称
                        }
                        ThisChart.ExportPicture(m_SavePath+"\\"+m_FileName,"gif",m_PicWidth,m_PicHeight);
                        return true;
                    }
                    catch(Exception ee)
                            {
                        return false;
                    }
                    return false;
                }


            }
        }
```

ASP.NET2.0 轻松搞定统计图表

本文讲述如何绘制**条形图，折线图，柱形图，面积图**等常见图形。

效果图：



清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F)
编辑(E)
查看(V)
收藏(A)
工具(T)
帮助(H)
后退
搜索
收藏夹
地址(D)
http://localhost/web/OWCdrawing.aspx

怎么样在ASP.NET2.0中使用OWC组件画图

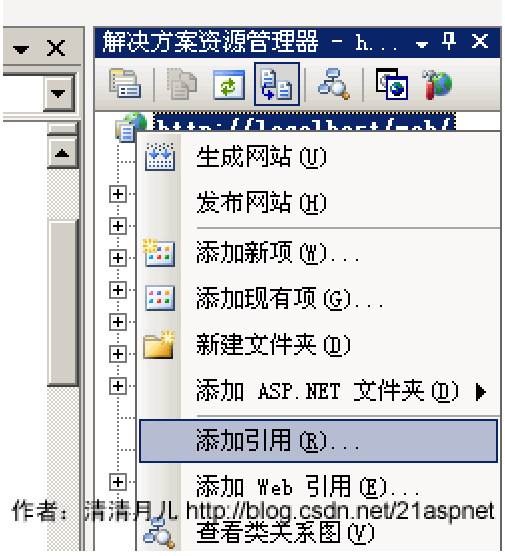
2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12

作者：清清月儿 http://blog.csdn.net/21aspnet

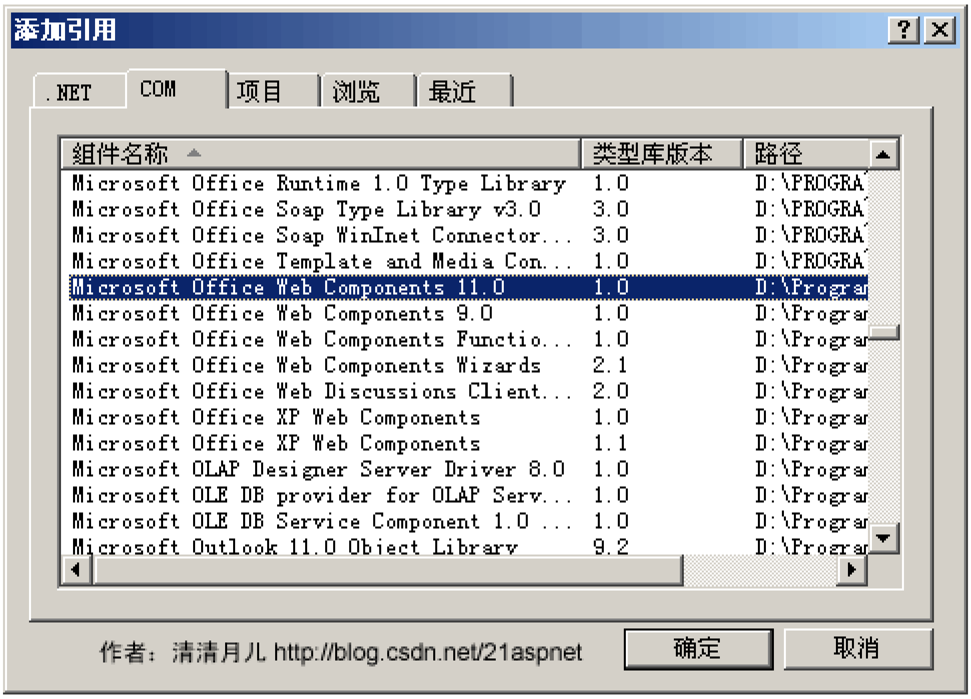
手把手教程：

原理：**OWC** 是 Office　Web　Compent 的缩写，即 Microsoft 的 Office　Web 组件，它为在 Web 中绘制图形提供了灵活的同时也是最基本的机制。在一个 intranet 环境中，如果可以假设客户机上存在特定的浏览器和一些功能强大的软件（如 IE6 和 Office　2000/XP/2003），那么就有能力利用 Office　Web 组件提供一个交互式图形开发环境。这种模式下，客户端工作站将在整个任务中分担很大的比重。理论上说 **Excel** 能做的图都可以通过 OWC 画。

第一步：
右键点击网站根目录引用。如图所示：



第二步：
点击"添加引用"后弹出一个窗口，添加 OWC 的引用。如图所示：



组件名称	类型库版本	路径
Microsoft Office Runtime 1.0 Type Library	1.0	D:\PROGRA
Microsoft Office Soap Type Library v3.0	3.0	D:\PROGRA
Microsoft Office Soap WinInet Connector...	3.0	D:\PROGRA
Microsoft Office Template and Media Con...	1.0	D:\PROGRA
Microsoft Office Web Components 11.0	1.0	D:\Progra
Microsoft Office Web Components 9.0	1.0	D:\Progra
Microsoft Office Web Components Functio...	1.0	D:\Progra
Microsoft Office Web Components Wizards	2.1	D:\Progra
Microsoft Office Web Discussions Client...	2.0	D:\Progra
Microsoft Office XP Web Components	1.0	D:\Progra
Microsoft Office XP Web Components	1.1	D:\Progra
Microsoft OLAP Designer Server Driver 8.0	1.0	D:\Progra
Microsoft OLE DB provider for OLAP Serv...	1.0	D:\Progra
Microsoft OLE DB Service Component 1.0 ...	1.0	D:\Progra
Microsoft Outlook 11.0 Object Library	9.2	D:\Progra

点"确定"。

第三步：
代码中引用 Microsoft.Office.Interop.Owc11。

全部代码
后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;   //添加数据操作引用
using Microsoft.Office.Interop.Owc11;//添加 Office 组件引用

public partial class OWCdrawing : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //连接数据库并获取特定字符串
        string strSeriesName = "图例 1";
        string ConnectString = "Server=(local);DataBase=web;Uid=sa;Pwd=sa";
        string Sql = "SELECT month,Allcount FROM Chart";
        SqlConnection myConn = new SqlConnection(ConnectString);
        myConn.Open();
        SqlDataAdapter Da = new SqlDataAdapter(Sql, myConn);
        DataSet ds = new DataSet();
        Da.Fill(ds);

        //存放月
        string[] MonNum = new string[12];
        //存放数据
        string[] MonCount = new string[12];
        //为数组赋值
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            MonNum[i] = ds.Tables[0].Rows[i][0].ToString();
            MonCount[i] = ds.Tables[0].Rows[i][1].ToString();
        }
        //为 x 轴指定特定字符串，以便显示数据
        string strXdata = String.Empty;
        foreach (string strData in MonNum)
        {
            strXdata += strData + "\t";
        }
        string strYdata = String.Empty;
        //为 y 轴指定特定的字符串，以便与 x 轴相对应
        foreach (string strValue in MonCount)
        {
```

```
        strYdata += strValue + "\t";
    }
    //创建 ChartSpace 对象来放置图表
    ChartSpace laySpace = new ChartSpaceClass();
    //在 ChartSpace 对象中添加图表
    ChChart InsertChart = laySpace.Charts.Add(0);
    //指定绘制图表的类型。类型可以通过 OWC.ChartChartTypeEnum 枚举值得到
    //InsertChart.Type = ChartChartTypeEnum.chChartTypeLine;//折线图
    //InsertChart.Type = ChartChartTypeEnum.chChartTypeArea;//面积图
    //InsertChart.Type = ChartChartTypeEnum.chChartTypeBarClustered;//条形图
    InsertChart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;//柱形图


    //指定图表是否需要图例标注
    InsertChart.HasLegend = false;


    InsertChart.HasTitle = true;//为图表添加标题
    InsertChart.Title.Caption = "2006 年清清月儿每个月花销流水账";//标题名称
    //为 x,y 轴添加图示说明
    InsertChart.Axes[0].HasTitle = true;
    InsertChart.Axes[0].Title.Caption = "";//月份
    InsertChart.Axes[1].HasTitle = true;
    InsertChart.Axes[1].Scaling.SplitMinimum = 200;
    InsertChart.Axes[1].Title.Caption = "数量";
    //添加一个 series 系列
    InsertChart.SeriesCollection.Add(0);
    //给定 series 系列的名字
    InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimSeriesNames, +(int)ChartSpecialDataSourcesEnum.chDataLiteral, strSeriesName);
    //给定分类
    InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, +(int)ChartSpecialDataSourcesEnum.chDataLiteral, strXdata);
    //给定值
    InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, strYdata);
    //输出文件.
    string strAbsolutePath = (Server.MapPath(".")) + "\\ShowData.gif";
    laySpace.ExportPicture(strAbsolutePath, "GIF", 400, 250);
```

```
        //创建 GIF 文件的相对路径.
        string strRelativePath = "./ShowData.gif";

        //把图片添加到 placeholder 中，并在页面上显示
        string strImageTag = "<IMG SRC='" + strRelativePath + "'/>";
        this.PlaceHolder1.Controls.Add(new LiteralControl(strImageTag));
    }
}
```

前台代码：

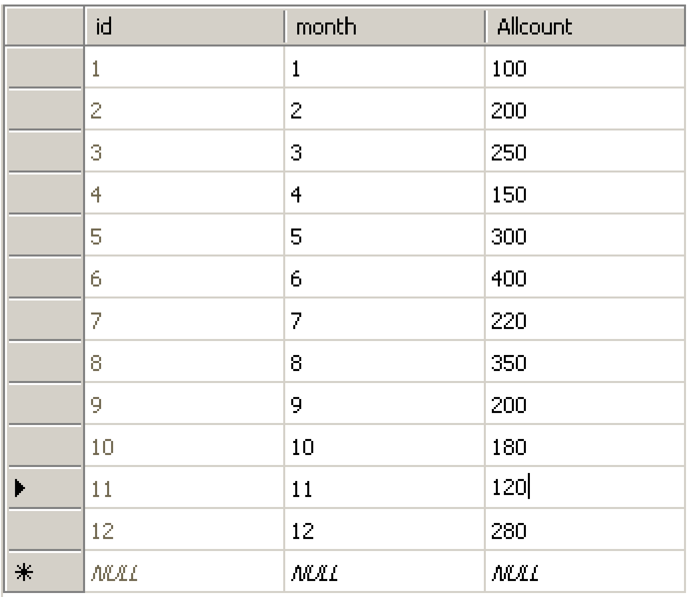
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="OWCdrawing.aspx.cs" Inherits="OWCdrawing" %>

<!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>清清月儿 http://blog.csdn.net/21aspnet</title>
</head>
<body>
    <form id="form1" runat="server">
   <div style="text-align: left">
        <table style="width: 600px">
            <tr>
                <td colspan="3" style="height: 20px">
                    <strong>怎么样在ASP.NET2.0 中使用OWC组件画图</strong></td>
            </tr>
            <tr>
                <td colspan="3" rowspan="2" style="height: 21px">
        <asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
                </td>
            </tr>
            <tr>
            </tr>
        </table>

    </div>
    </form>
</body>
</html>
```

数据库 **SQL** 脚本：

```
USE [web]
GO
/****** 对象：  Table [dbo].[Chart]    脚本日期: 03/27/2007 22:26:00 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Chart](
 [id] [int] IDENTITY(1,1) NOT NULL,
 [month] [smallint] NULL,
 [Allcount] [int] NULL
) ON [PRIMARY]
```

在数据库建好表以后要自己手动假想有 12 条数据，手动添加，最终结果类似下图：



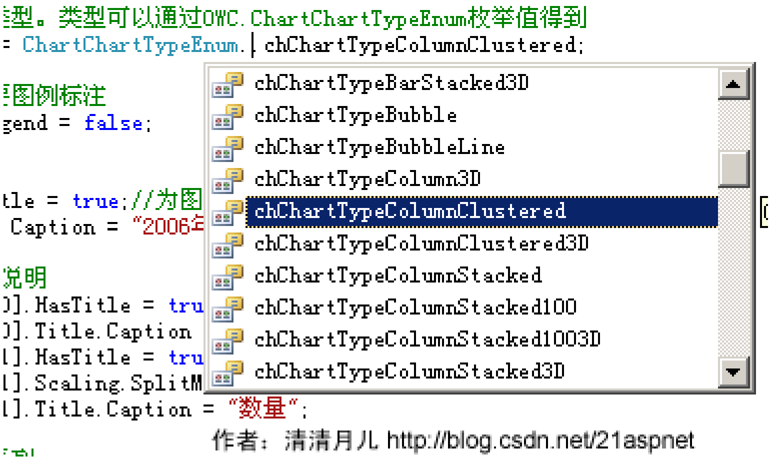
	id	month	Allcount
	1	1	100
	2	2	200
	3	3	250
	4	4	150
	5	5	300
	6	6	400
	7	7	220
	8	8	350
	9	9	200
	10	10	180
▶	11	11	120
	12	12	280
*	NULL	NULL	NULL

作者：清清月儿 http://blog.csdn.net/21aspnet

后台程序说明：

最关键就是 InsertChart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;

你可以在 ChartChartTypeEnum 后点出其他方法。如图所示：



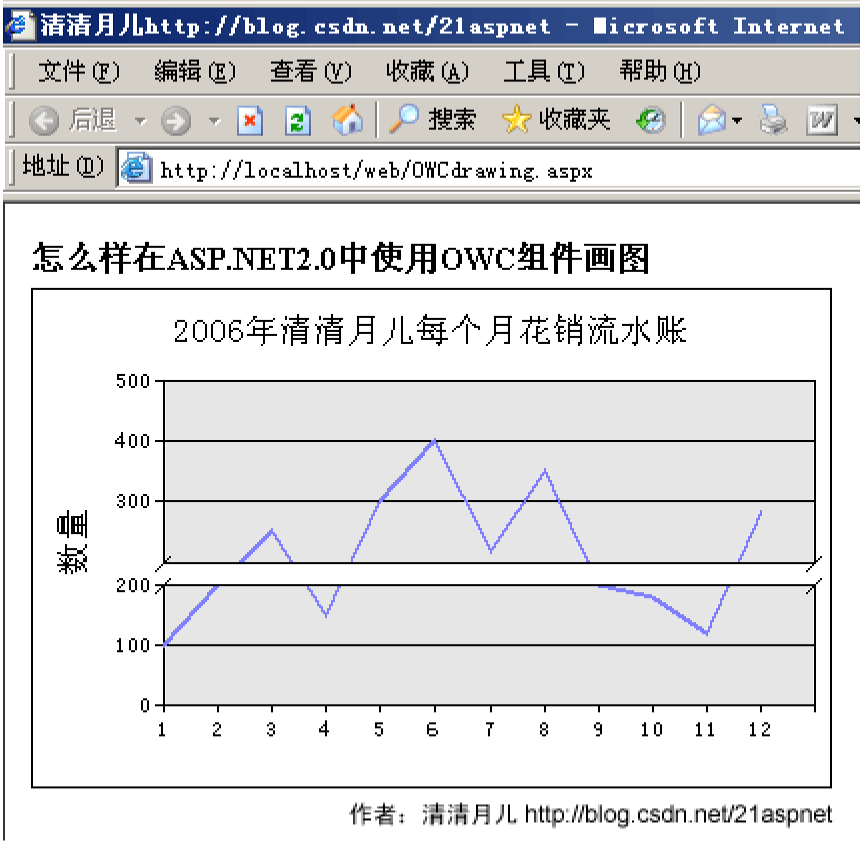
```
型。类型可以通过OWC.ChartChartTypeEnum枚举值得到
= ChartChartTypeEnum.| chChartTypeColumnClustered;

图例标注
gend = false;


tle = true;//为图
 Caption = "2006

说明
]].HasTitle = tru
]].Title.Caption
l].HasTitle = tru
l].Scaling.SplitM
l].Title.Caption = "数量";
```

chChartTypeBarStacked3D
chChartTypeBubble
chChartTypeBubbleLine
chChartTypeColumn3D
chChartTypeColumnClustered
chChartTypeColumnClustered3D
chChartTypeColumnStacked
chChartTypeColumnStacked100
chChartTypeColumnStacked1003D
chChartTypeColumnStacked3D

作者：清清月儿 http://blog.csdn.net/21aspnet

下面列出的是其他类型图：

折线图：



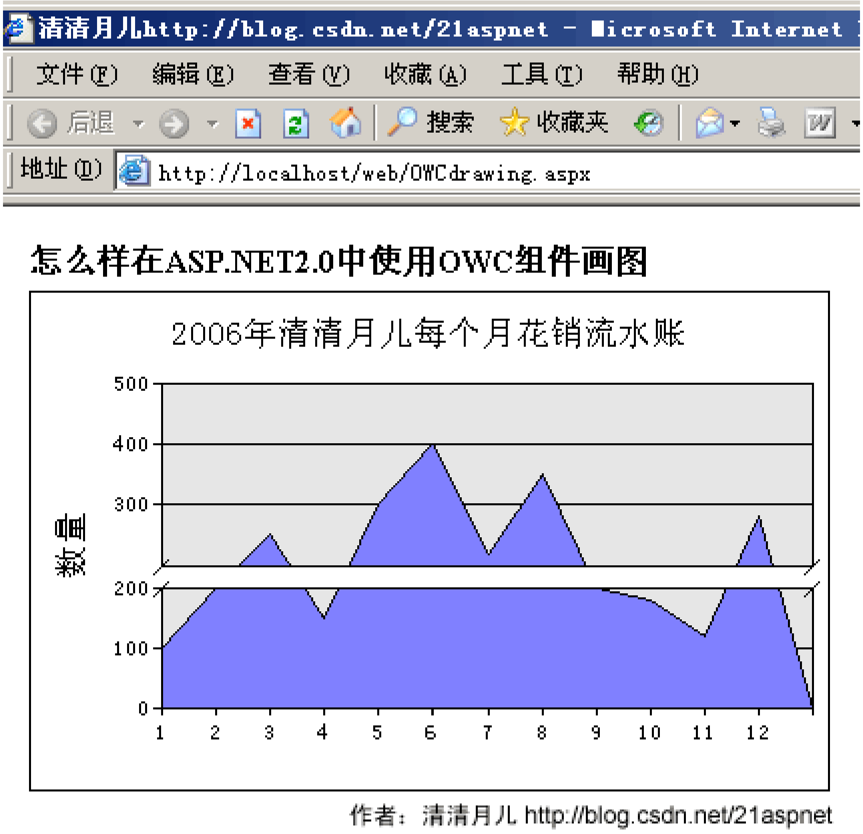
清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退
搜索
收藏夹
地址(D)
http://localhost/web/OWCdrawing.aspx

怎么样在ASP.NET2.0中使用OWC组件画图

2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12

作者：清清月儿 http://blog.csdn.net/21aspnet

面积图：



清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D)

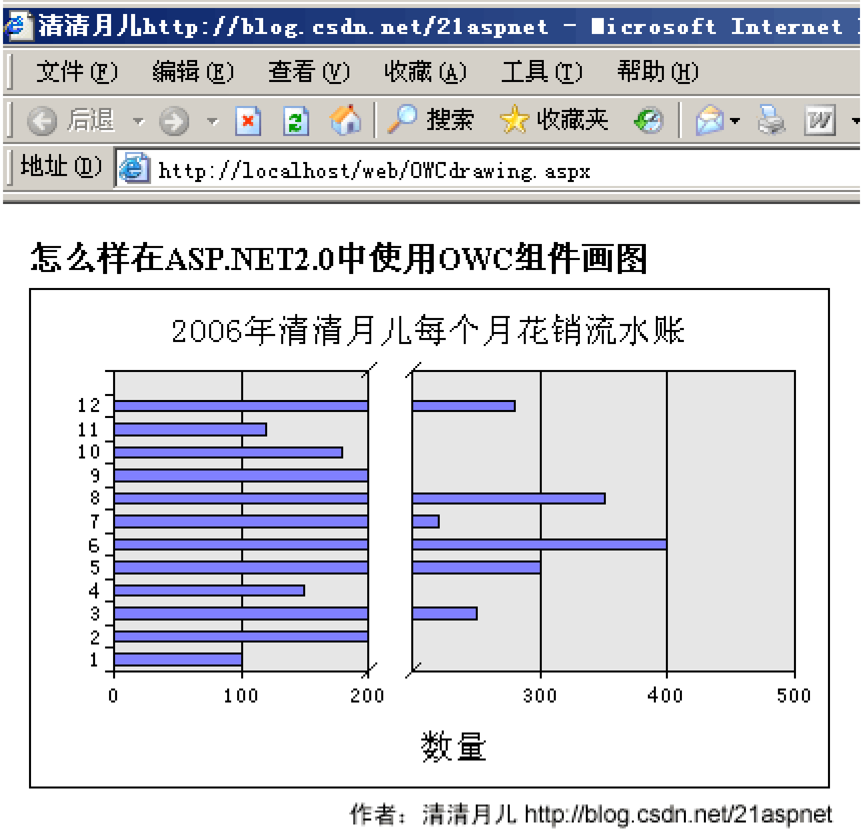
http://localhost/web/OWCdrawing.aspx

怎么样在ASP.NET2.0中使用OWC组件画图

2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12

作者：清清月儿 http://blog.csdn.net/21aspnet

条形图：



清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D)

http://localhost/web/OWCdrawing.aspx

怎么样在ASP.NET2.0中使用OWC组件画图

2006年清清月儿每个月花销流水账
12
11
10
9
8
7
6
5
4
3
2
1
0
100
200
300
400
500
数量

作者：清清月儿 http://blog.csdn.net/21aspnet

OWC 什么图形都可以画，还能画立体的，请大家自己尝试。

可以参考 OWC 手册，具体位置：
C:\Program Files\Common Files\Microsoft Shared\Web Components\11\2052\OWCVBA11.CHM

• asp.net 中调用 Office 来制作 3D 统计图

1、首先下载 owc11 COM 组件

2、注册 owc11

在工程中添加 C：\Program Files\Common Files\Microsoft Shared\Web Components\11 文件下的 owc11.dll 引用

3、在工程中添加

using OWC11；

4、开始 coding 举例如下：

public class ChartFactory { public ChartFactory（）

{ InitTypeMap（）；/// TODO： 在此处添加构造函数逻辑// } protected System.Web.UI.WebControls.Image imgHondaLineup；private string[] chartCategoriesArr；private string[] chartValuesArr；private OWC11.ChartChartTypeEnum chartType = OWC11.ChartChartTypeEnum.chChartTypeColumn3D；//默认值 private static Hashtable chartMap = new Hashtable（）；private static string chartTypeCh = "垂直柱状图" ；private static string chartTitle = ""；

private void InitTypeMap（）

{ chartMap.Clear（）；OWC11.ChartChartTypeEnum[] chartTypes = new OWC11.ChartChartTypeEnum[]{ ChartChartTypeEnum.chChartTypeColumnClustered，ChartChartTypeEnum.chChartTypeColumn3D，ChartChartTypeEnum.chChartTypeBarClustered，ChartChartTypeEnum.chChartTypeBar3D，

ChartChartTypeEnum.chChartTypeArea，
ChartChartTypeEnum.chChartTypeArea3D，
ChartChartTypeEnum.chChartTypeDoughnut，
ChartChartTypeEnum.chChartTypeLineStacked，
ChartChartTypeEnum.chChartTypeLine3D，
ChartChartTypeEnum.chChartTypeLineMarkers，
ChartChartTypeEnum.chChartTypePie，
ChartChartTypeEnum.chChartTypePie3D，

ChartChartTypeEnum.chChartTypeRadarSmoothLine，
ChartChartTypeEnum.chChartTypeSmoothLine}；

string[] chartTypesCh = new string [] {"垂直柱状统计图"，"3D 垂直柱状统计图"，"水平柱状统计图"，"3D 水平柱状统计图"，"区域统计图"，"3D 区域统计图"，"中空饼图"，"折线统计图"，"3D 折线统计图"，"折线带点统计图"，"饼图"，"3D 饼图"，"网状统计图"，"弧线统计图"}；

for（int i=0；i<chartTypes.Length；i++）

{ chartMap.Add（chartTypesCh[i]，chartTypes[i]）；} public ChartSpaceClass BuildCharts （）

{ string chartCategoriesStr = String.Join（"\t"， chartCategoriesArr）；string chartValuesStr = String.Join （"\t"，chartValuesArr）；

OWC11.ChartSpaceClass oChartSpace = new OWC11.ChartSpaceClass （）；

// ——// Give pie and doughnut charts a legend on the bottom. For the rest of // them let the control figure it out on its own. // ——

chartType = （ChartChartTypeEnum）chartMap[chartTypeCh]；

if （chartType == ChartChartTypeEnum.chChartTypePie || chartType == ChartChartTypeEnum.chChartTypePie3D || chartType == ChartChartTypeEnum.chChartTypeDoughnut）

{ oChartSpace.HasChartSpaceLegend = true；oChartSpace.ChartSpaceLegend.Position = ChartLegendPositionEnum.chLegendPositionBottom；}

oChartSpace.Border.Color = "blue"；oChartSpace.Charts.Add（0）；oChartSpace.Charts[0].HasTitle = true；oChartSpace.Charts[0].Type = chartType；oChartSpace.Charts[0].ChartDepth = 125；oChartSpace.Charts[0].AspectRatio = 80；oChartSpace.Charts[0].Title.Caption = chartTitle；oChartSpace.Charts[0].Title.Font.Bold = true；

oChartSpace.Charts[0].SeriesCollection.Add（0）；oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection.Add （）；

// ——// If you're charting a pie or a variation thereof percentages make a lot // more sense than values……

// ——

if （chartType == ChartChartTypeEnum.chChartTypePie || chartType == ChartChartTypeEnum.chChartTypePie3D || chartType == ChartChartTypeEnum.chChartTypeDoughnut）

{ oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].HasPercentage = true；oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].HasValue = false；} // ——// Not so for other chart types where values have more meaning than // percentages. // ——else { oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].HasPercentage = false；oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].HasValue = true；}

// ——// Plug your own visual bells and whistles here // ——oChartSpace.Charts[0].SeriesCollection[0].Caption = String.Empty；

oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Name = "verdana"；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Size = 10；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Bold = true；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Color = "red"；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Position = ChartDataLabelPositionEnum.chLabelPositionCenter；

if （chartType == ChartChartTypeEnum.chChartTypeBarClustered || chartType == ChartChartTypeEnum.chChartTypeBar3D || chartType == ChartChartTypeEnum.chChartTypeColumnClustered || chartType == ChartChartTypeEnum.chChartTypeColumn3D）

{ oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Position = ChartDataLabelPositionEnum.chLabelPositionOutsideEnd；}

oChartSpace.Charts[0].SeriesCollection[0].SetData（OWC11.ChartDimensionsEnum.chDimCategories，Convert.ToInt32（OWC11.ChartSpecialDataSourcesEnum.chDataLiteral）， chartCategoriesStr）；

oChartSpace.Charts[0].SeriesCollection[0].SetData（OWC11.ChartDimensionsEnum.chDimValues，Convert.ToInt32（OWC11.ChartSpecialDataSourcesEnum.chDataLiteral）， chartValuesStr）；

return oChartSpace；}

#region 属性设置 public string[] chartCategoriesArrValue { get { return chartCategoriesArr；} set { chartCategoriesArr = value；}

public string[] chartValuesArrValue { get { return chartValuesArr；

} set { chartValuesArr = value；} public string chartTypeValue { get { return chartTypeCh；} set { chartTypeCh = value；} public string chartTitleValue { get { return chartTitle；} set { chartTitle = value；} #endregion }

//调用 首先需要在页面上放置一个Image来显示产生的统计图

public void ShowChart（）

{

//初始化赋值 chartFactory.chartCategoriesArrValue = chartCategories；chartFactory.chartValuesArrValue = chartValues；chartFactory.chartTitleValue = chartTitle；chartFactory.chartTypeValue = chartType；

OWC11.ChartSpaceClass oChartSpace = chartFactory.BuildCharts（）；string path = Server.MapPath（"."）+ @"\images\Chart.jpeg"；//产生图片并保存 页可以是 png gif 图片 oChartSpace.ExportPicture（path，"jpeg"， 745，500）；Image1.ImageUrl = path； // 显示统计图}

// 保存统计图请参照上一篇文章

//由于每次生成的统计图都会覆盖原来的图片所以有必要的话可以用日期加时间的方式来作为图片的名字，但是这样将会产生很多图片需及时处理，如不需要只需取同名覆盖原来图片即可。

OWC使用方法

OWC==> Office Web Components

是微软 Office 的轻量级开发工具，随 Office 安装时一起安装

Office2003 版本对应 OWC11，里面有 5 个控件对象

图表工作区对象

数据源控件对象

记录集控件对象

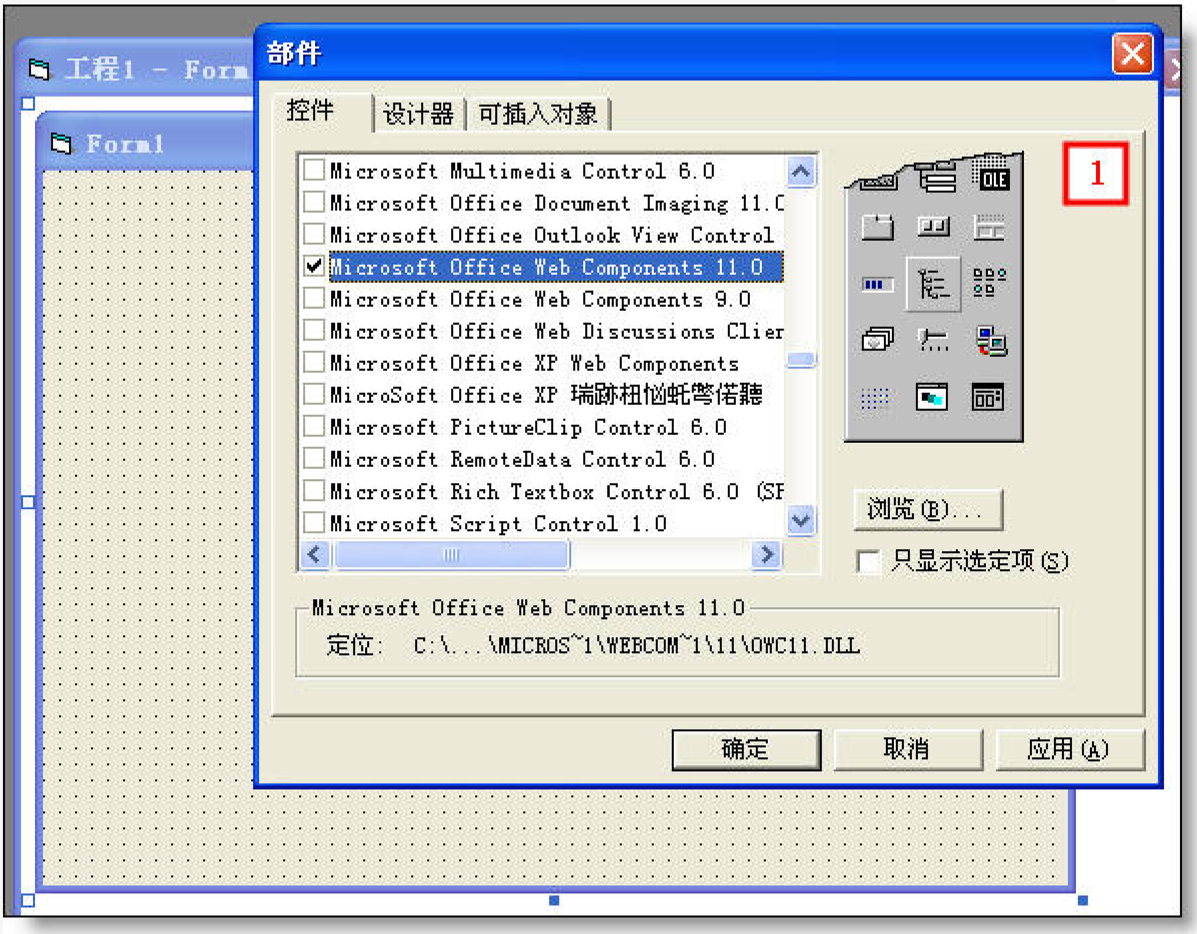
数据透视表列表对象

电子表格对象

很久以前用过的，最近有人问就翻出来看看

其实很简单，只是可能没有到注意文档罢了

我简单用 VB 做了下常用的 SpreadSheet 例子



部件

控件 | 设计器 | 可插入对象

- □ Microsoft Multimedia Control 6.0
- □ Microsoft Office Document Imaging 11.C
- □ Microsoft Office Outlook View Control
- ☑ Microsoft Office Web Components 11.0
- □ Microsoft Office Web Components 9.0
- □ Microsoft Office Web Discussions Clier
- □ Microsoft Office XP Web Components
- □ MicroSoft Office XP 瑞跡杻惱蚝誓偌聽
- □ Microsoft PictureClip Control 6.0
- □ Microsoft RemoteData Control 6.0
- □ Microsoft Rich Textbox Control 6.0 (SF
- □ Microsoft Script Control 1.0

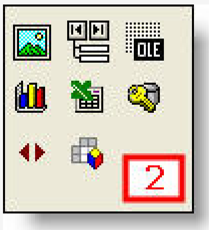
浏览(B)...

□ 只显示选定项(S)

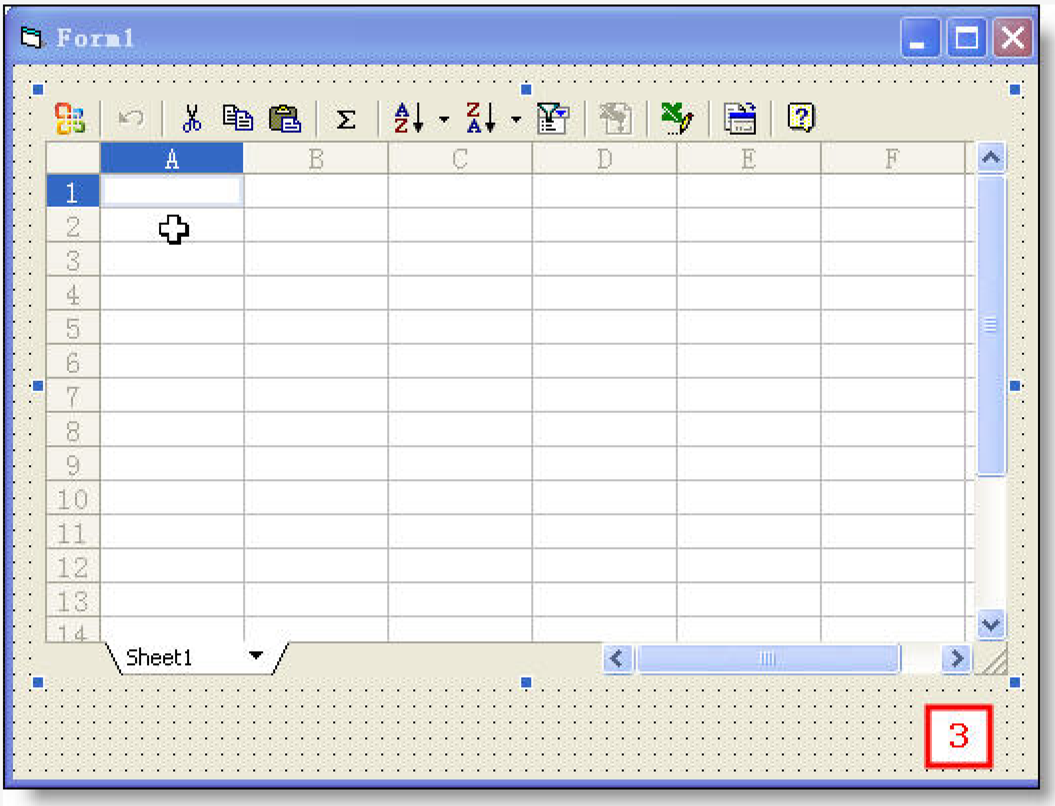
Microsoft Office Web Components 11.0

定位:　C:\...\MICROS~1\WEBCOM~1\11\OWC11.DLL

确定　取消　应用(A)



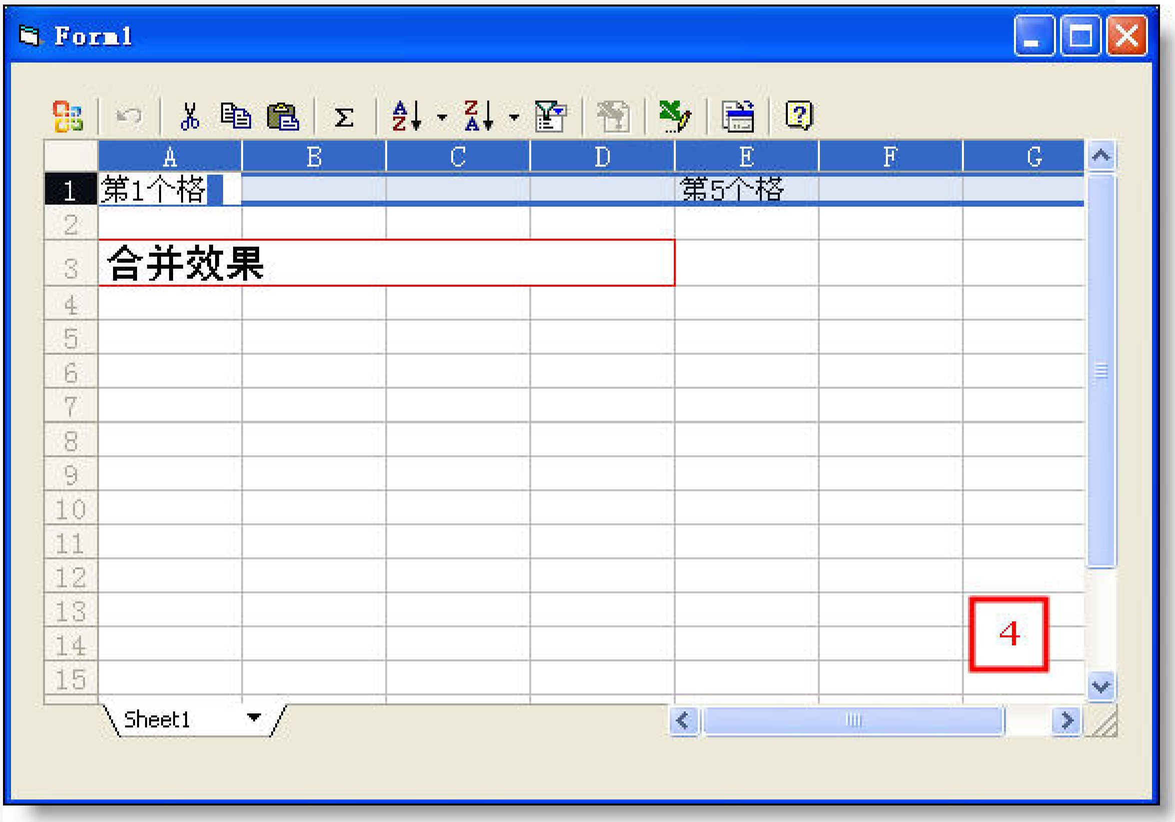
Form 上加一个 Spreadsheet 控件



代码，很简单，没什么好说的

```
Private Sub Form_Load()


    '单元格
    '可以使用循环的方式填充单元格
    Spreadsheet1.Cells(1, 1) = "第 1 个格"
    Spreadsheet1.Cells(1, 5) = "第 5 个格"
    '区域
    '合并
    Spreadsheet1.Range("A3:D3").Merge
    Spreadsheet1.Range("A3:D3").Value = "合并效果"
    '格式设置
    Spreadsheet1.Range("A3:D3").Font.Name = "黑体"
    Spreadsheet1.Range("A3:D3").Font.Size = 15


    Spreadsheet1.Range("A3:D3").Borders(xlEdgeTop).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeBottom).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeLeft).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeRight).Color = vbRed

''    '不提示直接保存为本地文件
''    Spreadsheet1.Export "c:\xxx.xls", ssExportActionNone
''    '直接在 Excel 中打开
''    Spreadsheet1.Export "c:\xxx.xls", ssExportActionOpenInExcel
End Sub
```

运行出来就是这个样子



第1个格

第5个格

合并效果

4

因为整个编程中使用了 VBA 语法，所以还是很容易理解的

如果你的机器上安装了 Office2000/xp/2003

那么你的机器上已经有使用文档及开发文档了，很详尽，参照一下，应该没啥问题。

我的机器装的是 2003，在如下位置上有相关文档

```
C:\Program Files\Common Files\Microsoft Shared\Web Components\10\2052
```

C:\Program Files\Common Files\Microsoft Shared\Web Components\11\2052

前面的文档都是使用帮助

OWCVBA11.CHM 是开发文档

文本的小例子在此处下载 ，VB的

http://www.cnblogs.com/Files/babyt/Owc1.rar

我想用 spreadsheet 做个表格,不知道是不是可以用这个控件直接导入 xml 文件呢,因为它有一个 export 方法,可以导出为 xml 文件.我查了它的所有方法发现都没有发现导入这个功能,但是它里面却又可以导入的,啊泰知道如何在代码中实现导入 xml 文件吗?谢谢啊

1：首先保证你用的是OWC11 版本
2：注意你的XML文件的格式，不是所有的格式都能被支持的。严格说就是只支持指定格式，你可以把你的Excel文件另存为xml文件，然后看下格式规定
3：仍以vb6 的使用为例，你导入spreadsheet控件后，使用如下代码即可导入xml文件（注意格式）

```
Spreadsheet1.XMLURL = App.Path & "\test.xml"
```

以下文章可能对你进一步的工作有所帮助 ^_^
http://www.microsoft.com/china/msdn/library/office/office/xl2k3xmllist.mspx?mfr=true

利用GDI+的双缓冲技术来提高绘图效率

GDI+在一种与设备无关的环境下提供了一套统一的绘图编程模型，极大的提高了 Windows 绘图编程的方便性，我们再也不用创建什么各种各样复杂的设备环境了，说实话，我现在想起来都头疼。

题归正传，关于如何进行 GDI+的基本编程，我不能过多的加以描述，如果有对此概念还不太清楚的朋友，建议先去了解一下相关的资料，我们在这里主要讨论的是一种提高绘图效率（主要是动画效率）的双缓冲技术在 GDI+中的应用和实现。

实现目的

为了能清楚的对比应用双缓冲技术前后的效果，我编写了一段程序来进行测试。首先，我创建了一个普通的 Windows Application，在主 Form 中，我放置了一个定时器：timer1，然后将它的 Interval 属性设置为 10，然后在 Form 上放置两个按纽，分别用来控制定时器的开启和关闭，最后，我还放置了一个 label 控件，用来显示绘图的帧数。

测试程序

在timer1的timer1_Tick事件中,我写下了如下的代码(其中flag是一个bool型标志变量)：

```
DateTime t1 = DateTime.Now;

Graphics g = this.CreateGraphics();

if(flag)

{

      brush = new LinearGradientBrush(new PointF(0.0f, 0.0f),

            new PointF(700.0f, 300.0f), Color.Red, Color.Blue);

      flag = false;

}

else
```

```
{
      brush = new LinearGradientBrush(new PointF(0.0f, 0.0f),
            new PointF(700.0f, 300.0f), Color.Blue, Color.Red);
      flag = true;
}
for(int j = 0; j < 60; j ++)
{
      for(int i = 0; i < 60; i++)
      {
            g.FillEllipse(brush, i * 10, j * 10, 10, 10);
      }
}
DateTime t2 = DateTime.Now;
TimeSpan sp = t2 - t1;
float per = 1000 / sp.Milliseconds;
this.label1.Text = "速度：" + per.ToString() + "帧/秒";
```

运行后，我点击“开始”按纽，效果如下图所示：

应用双缓冲以前的效果图（帧数：5 帧/秒）

正如大家所看到的，我在程序中使用循环画了几百个圆形，然后在每次的定时器脉冲事件中使用不同方向的线性渐变来对它们进行填充，形成了一个动画效果。不过不幸的是，程序运行起来闪烁很严重，几乎每次刷新的时候都可以看到一条很明显的扫描线从上慢慢的刷到下

来完成整幅图形的刷新动作。如果你不是要模拟老式雷达的区域扫描的话，这种速度不会满足你的要求。

改进代码

下面是我改进以后的代码：

```
DateTime t1 = DateTime.Now;
Bitmap bmp = new Bitmap(600, 600);
Graphics g = Graphics.FromImage(bmp);
if(flag)
{
      brush = new LinearGradientBrush(new PointF(0.0f, 0.0f),
                  new PointF(700.0f, 300.0f), Color.Red, Color.Blue);
      flag = false;
}
else
{
      brush = new LinearGradientBrush(new PointF(0.0f, 0.0f),
                  new PointF(700.0f, 300.0f), Color.Blue, Color.Red);
      flag = true;
}
for(int j = 0; j < 60; j ++)
{
      for(int i = 0; i < 60; i++)
      {
            g.FillEllipse(brush, i * 10, j * 10, 10, 10);
      }
}
this.CreateGraphics().DrawImage(bmp, 0, 0);
DateTime t2 = DateTime.Now;
TimeSpan sp = t2 - t1;
float per = 1000 / sp.Milliseconds;
this.label1.Text = "速度：" + per.ToString() + "帧/秒";
```

运行后，我点击“开始”按纽，效果如下图所示：

应用双缓冲以后的效果图（帧数：9 帧/秒）

经过改进后，画面刷新速度大大加快，绝对看不到任何的“扫描线”，帧数也从 5 帧一下就提高到了 9 帧，几乎是两倍于前的速度。这究竟是什么原因呢？让我来讲述其中的道理。

因为圆是要一个一个画上去，所以每画一个圆，系统就要做一次图形的绘制操作，图形的重绘是很占用资源的，当需要重绘的图形数量很多的时候，所造成的系统开销就特别大，造成我们看到的那种刷新缓慢的情况。那么如何来解决这个问题呢？

答案就是双缓冲，何谓“双缓冲”？它的基本原理就是：先在内存中开辟一块虚拟画布，然后将所有需要画的图形先画在这块“虚拟画布”上，最后在一次性将整块画布画到真正的窗体上。因为所有的单个图形的绘制都不是真正的调用显示系统来“画”，所以不会占用显示系统的开销，极大的提高的绘图效率。

实现双缓冲的具体步骤

我再来详细解释一下刚才实现双缓冲的具体步骤：

1、在内存中建立一块“虚拟画布”：

```
Bitmap bmp = new Bitmap(600, 600);
```

2、获取这块内存画布的 Graphics 引用：

```
Graphics g = Graphics.FromImage(bmp);
```

3、在这块内存画布上绘图：

```
g.FillEllipse(brush, i * 10, j * 10, 10, 10);
```

4、将内存画布画到窗口中

```
this.CreateGraphics().DrawImage(bmp, 0, 0);
```

总结

怎么样？是不是很简单？但是正是这个简单的操作大大提高了绘图效率,所以如果你需要进行 GDI+图形编程，双缓冲技术一定要掌握，特别是在进行大量图形绘制刷新的情况下要尽量采用。

如何设置Image保存时的文件扩展名

Posted on 2005-03-08 15:33 xingd 阅读(2211) 评论(4) 编辑 收藏 所属分类: GDI+ .

假如有一个从 Stream 构造的 Image，那么在保存成文件时应该使用什么样的扩展名呢？

在 IE 中使用图片中，图片格式是从图片内容中析取出来的，与扩展名无关，但是，保存 Image 时不指定扩展名，想起来都有点不爽。

目前我使用的方法是根据 Image.RawFormat 与 ImageFormat 提供的静态属性进行 Guid 比较，以确定 Image 的格式，然后返回相应的扩展名，代码如下：

```
using (Image image = Image.FromStream(stream))
{
    Guid guid = image.RawFormat.Guid;
    string ext = ".tmp";

    if (guid == ImageFormat.Bmp.Guid)
    {
        ext = ".bmp";
    }
    else if (guid == ImageFormat.Gif.Guid)
    {
        ext = ".gif";
    }
    else if (guid == ImageFormat.Jpeg.Guid)
    {
        ext = ".jpg";
    }

    image.Save(file + ext);
}
```

不知道还有没有更好的方法。

那请问如何用 GDI＋读入图像，设置一个窗口，显示图像，再保存图像呢？（我是新手，多谢指点）

读取图像可以用 Image.FromFile，显示图像用 PictureBox 控件，Image.Save 方法可用于保存图像。

学习使用**GDI+**绘制饼状图

Posted on 2005-10-18 16:56 Leey 阅读(366) 评论(1) 编辑 收藏 所属分类: 学习笔记

.

早就申请了博客园的空间，但是一直也没有发过文章。

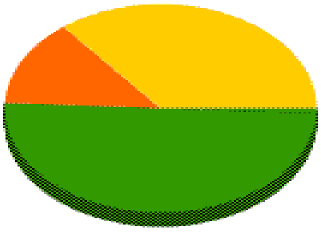
这次就算个开始吧

对于 GDI＋，我不是很熟悉，只是工作中用到了，才现找资料学习

这次画饼状图也是一样，Web 页面中需要加入统计的饼状图，没办法。

虽然最终把图画出来了，但是图上却有一条虚线去不掉，到现在也搞不懂为什么。

如下图，是最终的效果。



在黄色区域中间有一条细细的虚线。

代码如下：

```
using System;

using System.Collections;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;


namespace JRJC.Web.QuanZheng

{

    /// <summary>

    /// XChart 的摘要说明。

    /// </summary>

    public class XChart

    {
        /// <summary>
        /// 根据传递的数组绘制饼图
        /// </summary>
        /// <param name="Width">宽度</param>
        /// <param name="Height">高度</param>
        /// <param name="dataArr">数据数组</param>
        /// <param name="FileName">文件名称</param>
        public static void DrawPieByArrayList(int Width, int Height, int Magin, int Blank, DataAndColorPair[] arr, string FileName)
        {
            Bitmap bitmap = new Bitmap(Width, Height); //图片
            Graphics g = Graphics.FromImage(bitmap); //绘图
            g.DrawRectangle(new Pen(Color.White),0,0,Width,Height); //边框
            g.FillRectangle(new SolidBrush(Color.White), 1, 1,Width - Magin, Height - Magin);
            SolidBrush brush = new SolidBrush(Color.Blue); //画笔
            g.SmoothingMode = SmoothingMode.AntiAlias; //柔化
            //g.DrawLine(new Pen(brush,10),1,1,10,10);
            Rectangle outline = new Rectangle(Blank,Blank,Width - Blank * 2, Height - Blank * 2); //饼图范围

            //绘制饼图
            float total = 0;
            float start = 0;
            float angle = 0;
            for(int i=0;i<arr.Length;i++)
            {
                total += arr[i].Data;
            }
            //Console.WriteLine(total.ToString());
            for(int i=0;i<arr.Length;i++)
            {
                angle = (float)(arr[i].Data/total) * 360;
                //Console.WriteLine(angle.ToString());
                if(angle<=180)
                {
                    Draw3DPie(ref g,arr[i].ShowColor,outline,start,angle);
                }
                else//如果大于 180 度，则将大 Pie 拆开两块画
                {
                    float acuteAngle=angle-180;
                    Draw3DPie(ref g,arr[i].ShowColor,outline,start,acuteAngle);
                    Draw3DPie(ref g,arr[i].ShowColor,outline,start+acuteAngle,angle-acuteAngle);
                }
                //Draw3DPie(ref g,arr[i].ShowColor,outline,start,angle);
                start += angle;
            }
            bitmap.Save(FileName, ImageFormat.Gif);
        }
        private static void Draw3DPie(ref Graphics g,Color brushColor,Rectangle outline,float start,float angle)
        {
            //以下使用 HatchBrush 画阴影
            //if((start <135 ) || (start == 180))
            if((start <135 ))
            {
                //深度
                for(int iLoop2 = 0; iLoop2 < 10; iLoop2++)
                {
            }
                    g.FillPie(new HatchBrush(HatchStyle.Percent50,brushColor),
                        outline.X,
                        outline.Y + iLoop2,
                        outline.Width,
                        outline.Height,
                        start,
                        angle);
                }
            }
            if(start == 135)
            {
                for(int iLoop2 = 0; iLoop2 < 10; iLoop2++)
                {
                    g.FillPie(new HatchBrush(HatchStyle.Percent50,brushColor),
                        outline.X - 30,
                        outline.Y + iLoop2 + 15,
                        outline.Width,
                        outline.Height,
                        start,
                        angle);
                }
                g.FillPie(new SolidBrush(brushColor),
                    outline.X - 30,
                    outline.Y + 15,
                    outline.Width,
                    outline.Height,
                    start,
                    angle);
            }
            else //画 pie
```

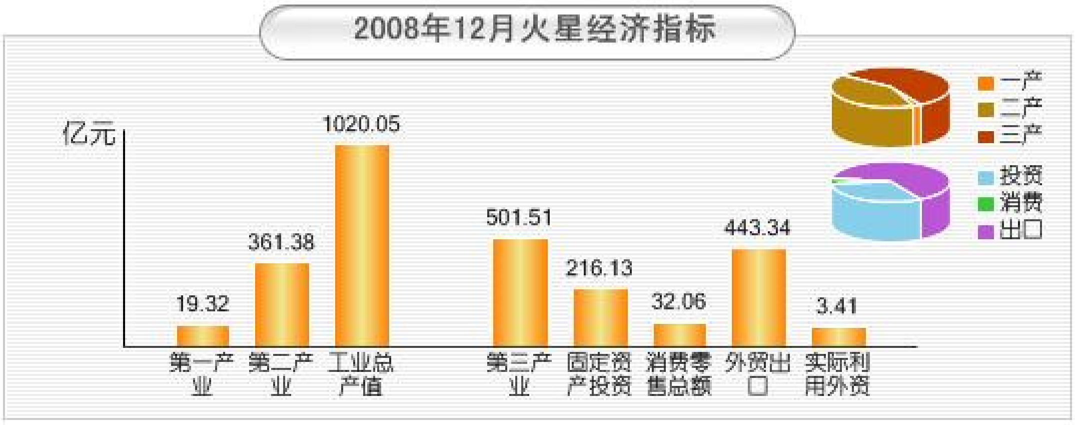
```
            {
                g.FillPie(new SolidBrush(brushColor), outline.X, outline.Y, outline.Width,outline.Height, start, angle);
            }
            //g.FillPie(new SolidBrush(arr[i].ShowColor),outline,start,angle);
        }
    }
}
```

一个功能强大超级好用的图表组件Dundas Chart

最近因为项目需要，需要为客户的统计数据生成图表，包括柱状图和饼图。我找来了Dundas Software的Dundas Chart，大家可以上它的网站http://www.dundas.com/查看最新版本的信息以及下载该软件它确实是一个功能强大的组件，而且用户友好性方面很值得国产软件学习。

闲话少说，进入该软件的使用。先安装该软件，安装后它包括两个非常有用的功能，一个是做的非常类似 msdn 的帮助文档，相信对.net 程序员非常友好，就当是 msdn 用吧。另外一个是它的 sample，这是一个完整的.net 解决方案，你可以在 IIS 里面配置个网站来访问它。接着你就可以开始通过 sample 来了解和学习 Dundas Chart 的强大功能了。

下面给个我学习中生成的图片。



下面是程序部分源码

```
private void Page_Load(object sender, System.EventArgs e)
        {
            // 在此处放置用户代码以初始化页面
            if(!this.IsPostBack)
            {
                double industry1 = 19.32;
                double industry2 = 361.38;
                double industry3 = 501.51;
                double industryfull = 1020.05;
                double fixedasserts = 216.13;
                double retailtrade = 32.06;
                double foreigntrade = 443.34;
                double foreigncapital = 3.41;
                //------------------------生产统计图表-------------------------
                 Dundas.Charting.WebControl.Chart Chart1 = new Dundas.Charting.WebControl.Chart();
                 Chart1.BackImage = this.Server.MapPath("cn/images/target/back_img.gif");
                Chart1.BackGradientEndColor = Color.White;
                Chart1.BorderLineColor = Color.White;
                Chart1.BorderLineWidth = 0;
                Chart1.BorderSkin.FrameBackColor = Color.MediumTurquoise;
                Chart1.BorderSkin.FrameBackGradientEndColor = Color.Teal;
                Chart1.Palette = ChartColorPalette.SemiTransparent;
                Chart1.Width = 545;
                Chart1.Height = 215;
                Chart1.ImageType = ChartImageType.Jpeg;
                Chart1.AntiAliasing = AntiAliasing.All;
                Chart1.Titles.Add("Default");
                Chart1.Titles[0].Text = "2008 年 12 月火星经济指标";
                Chart1.Titles[0].Alignment = ContentAlignment.TopCenter;
                Chart1.Titles[0].Font = new Font("黑体", 12, FontStyle.Bold);
                Chart1.Titles[0].Color = Color.FromArgb(72, 72, 72);
    ⊞           饼图("一产", "二产", "三产")
    ⊞           饼图("投资", "消费", "出口")
    ⊞           柱状图
```

```
                Chart1.Save(this.Server.MapPath("Stat.jpg"), ChartImageFormat.Jpe
g);

            }
        }

  private double MaxValue(double[] yValue)
          {
            double maxvalue = 0;
            for(int i=0;i<yValue.Length;i++)
              {
                if(yValue[i] > maxvalue) maxvalue = yValue[i];
            }
            return maxvalue;
        }
```

重提：上传图片生成略缩图“最佳解决”

Posted on 2008-02-24 17:36 刀晟容 阅读(569) 评论(5) 编辑 收藏 .

从用 .Net Web 开发到现在所看到的略缩生成代码都不尽人意，要不太局限，要不失真厉害。

为此写了一个相对完善的函数供大家学习。

其中的 SaveIamge 函数提供了失真解决方法，对于处理过的图片（如加水印......）要求保持高品质可以直接调用。

```
using System;
using System.IO;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Drawing.Text;
using System.Text.RegularExpressions;

/// <summary>
// 图片处理。
// http://www.hulaka.com/
// (c) 2008 Dao
/// </summary>
public class Image
{
    /// <summary>
    /// 缩略模式。
    /// </summary>
    public enum ThumbMode : byte
    {
        /// <summary>
        /// 完整模式
        /// </summary>
        Full = 1,
        /// <summary>
        /// 最大尺寸
        /// </summary>
        Max
    }

    /// <summary>
    /// 缩略图。
    /// </summary>
    /// <param name="image">要缩略的图片</param>
    /// <param name="size">要缩放的尺寸</param>
    /// <param name="mode">缩略模式</param>
    /// <param name="contentAlignment">对齐方式</param>
    /// <returns>返回已经缩放的图片。</returns>
    public static Bitmap Thumbnail(Bitmap image, Size size, ThumbMode mode, ContentAlignment contentAlignment)
    {
        if (!size.IsEmpty && !image.Size.IsEmpty && !size.Equals(image.Size))
        {
            //先取一个宽比例。
            double scale = (double)image.Width / (double)size.Width;
            //缩略模式
            switch (mode)
            {
                case ThumbMode.Full:
                    if (image.Height > image.Width)
                        scale = (double)image.Height / (double)size.Height;
                    break;
                case ThumbMode.Max:
                    if (image.Height / scale < size.Height)
                        scale = (double)image.Height / (double)size.Height;
                    break;
            }
            SizeF newSzie = new SizeF((float)(image.Width / scale), (float)(image.Height / scale));
            Bitmap result = new Bitmap(size.Width, size.Height);
            using (Graphics g = Graphics.FromImage(result))
            {
                g.FillRectangle(Brushes.White, new Rectangle(new Point(0, 0), size));
                g.InterpolationMode = InterpolationMode.HighQualityBicubic;
                g.SmoothingMode = SmoothingMode.HighQuality;
                g.PixelOffsetMode = PixelOffsetMode.HighQuality;
                g.CompositingMode = CompositingMode.SourceOver;
                g.CompositingQuality = CompositingQuality.HighQuality;
                g.TextRenderingHint = TextRenderingHint.AntiAliasGridFit;
                //对齐方式
                RectangleF destRect;
                switch (contentAlignment)
                {
                    case ContentAlignment.TopCenter:
                        destRect = new RectangleF(new PointF(-(float)((newSzie.Width - size.Width) * 0.5), 0), newSzie);
                        break;
                    case ContentAlignment.TopRight:
                        destRect = new RectangleF(new PointF(-(float)(newSzie.Width - size.Width), 0), newSzie);
                        break;
                    case ContentAlignment.MiddleLeft:
                        destRect = new RectangleF(new PointF(0, -(float)((newSzie.Heig
ht - size.Height) * 0.5)), newSzie);
                        break;
                    case ContentAlignment.MiddleCenter:
                        destRect = new RectangleF(new PointF(-(float)((newSzie.Widt
h - size.Width) * 0.5), -(float)((newSzie.Height - size.Height) * 0.5)), newSzie);
                        break;
                    case ContentAlignment.MiddleRight:
                        destRect = new RectangleF(new PointF(-(float)(newSzie.Widt
h - size.Width), -(float)((newSzie.Height - size.Height) * 0.5)), newSzie);
                        break;
                    case ContentAlignment.BottomLeft:
                        destRect = new RectangleF(new PointF(0, -(float)(newSzie.Heigh
t - size.Height)), newSzie);
                        break;
                    case ContentAlignment.BottomCenter:
                        destRect = new RectangleF(new PointF(-(float)((newSzie.Widt
h - size.Width) * 0.5), -(float)(newSzie.Height - size.Height)), newSzie);
                        break;
                    case ContentAlignment.BottomRight:
                        destRect = new RectangleF(new PointF(-(float)(newSzie.Widt
h - size.Width), -(float)(newSzie.Height - size.Height)), newSzie);
                        break;
                    default:
                        destRect = new RectangleF(new PointF(0, 0), newSzie);
                        break;
                }
                g.DrawImage(image, destRect, new RectangleF(new PointF(0F, 0F), i
mage.Size), GraphicsUnit.Pixel);
                image.Dispose();
            }
            return result;
        }
        else
            return image;
    }

    /// <summary>
    /// 保存图片。
    /// </summary>
    /// <param name="image">要保存的图片</param>
    /// <param name="quality">品质（1L~100L之间，数值越大品质越好）</param>
    /// <param name="filename">保存路径</param>
    public static void SaveIamge(Bitmap image, long quality, string filename)
    {
        using (EncoderParameters encoderParams = new EncoderParameters(1))
        {
            using (EncoderParameter parameter = (encoderParams.Param[0] = new EncoderParameter(Encoder.Quality, quality)))
            {
                ImageCodecInfo encoder = null;
                //取得扩展名
                string ext = Path.GetExtension(filename);
                if (string.IsNullOrEmpty(ext))
                    ext = ".jpg";
                //根据扩展名得到解码、编码器
                foreach (ImageCodecInfo codecInfo in ImageCodecInfo.GetImageEncoders())
                {
                    if (Regex.IsMatch(codecInfo.FilenameExtension, string.Format(@"(;|^)\*\{0}(;|$)", ext), RegexOptions.IgnoreCase))
```

调用方法如下：

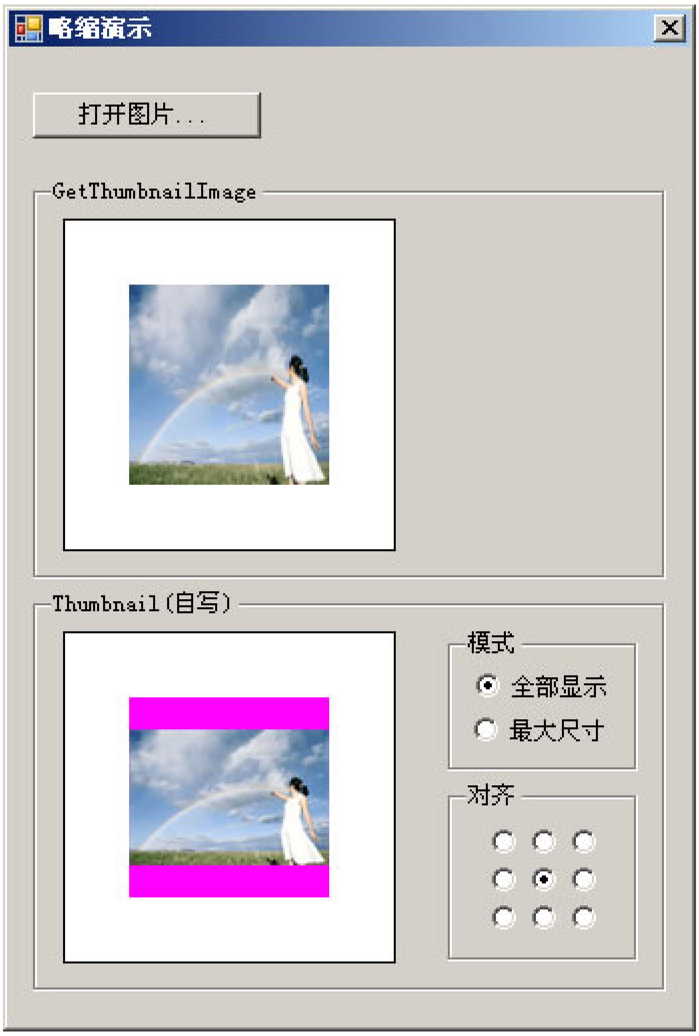
```
                    {
                        encoder = codecInfo;
                        break;
                    }
                }
                Directory.CreateDirectory(Path.GetDirectoryName(filename));
                image.Save(filename, encoder, encoderParams);
            }
        }
    }

    /// <summary>
    /// 保存图片。
    /// </summary>
    /// <param name="stream">要保存的流</param>
    /// <param name="quality">品质（1L~100L之间，数值越大品质越好）</param>
    /// <param name="filename">保存路径</param>
    public static void SaveIamge(Stream stream, long quality, string filename)
    {
        using (Bitmap bmpTemp = new Bitmap(stream))
        {
            SaveIamge(bmpTemp, quality, filename);
        }
    }
}
```

调用方法如下：

```
using (Bitmap bmpAvatar = Image.Thumbnail(new Bitmap(/*<流，FileUpload 控件的 PostedFile.InputStream 属性>*/), new Size(48, 48), Image.ThumbMode.Full, ContentAlignment.MiddleCenter))
```

```
{
    Image.SaveIamge(bmpAvatar, 95L, Server.MapPath("~/upfiles/Avatar.jpg"));
}
```

演示程序：



下载演示程序

Tag标签：略缩,asp.net,C#

Image 类就有这样的方法。

Image img=new Image()

img.GetThumbnailImage(newWidth, newHeigth, callback, new System.IntPtr());

如果图片大小和你要求缩放的大小不一样的话（宽高比列刚好相反） GetThumbnailImage 拉伸的太严重了

自己写的一个asp.net的生成曲线图的过程

效果如下:

http://zx.8dao.net/imginfo.aspx?sendid=20061123085937

这里是从Dataset里的数据生成曲线图.

我的Dataset是从表Sendrec里读取的数据,分别有Id,Sendid(订单号),Sendtime(记录时间),Sendnum(单位时间发送量/我这里是五分钟)几个字段

过程如下:

public void draw(Page page,DataSet ds,int Tnum){}

其中page是用来传递引用这个过程的页面,这样让页面是JPG方式直接向客户端输出生成的曲线图.

ds就是取出来的数据集了

Tnum只是我这里要用到的一个参数,不想让这个类去接触读取过程,所以把订单的总量直接取出后传递给它的.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Drawing;
using System.IO;



public class imgdraw
```

```
{
    public imgdraw()
    {

    }
    public void draw(Page page,DataSet ds,int Tnum)
    {
        //取得记录数量
        int count = ds.Tables[0].Rows.Count;
        //记算图表宽度
        int wd = 80 + 20 * (count - 1);
        //设置最小宽度为 800
        if (wd < 800) wd = 800;
        //生成 Bitmap 对像
        Bitmap img=new Bitmap(wd,400);
        //生成绘图对像
        Graphics g = Graphics.FromImage(img);
        //定义黑色画笔
        Pen Bp = new Pen(Color.Black);
        //定义红色画笔
        Pen Rp = new Pen(Color.Red);
        //定义银灰色画笔
        Pen Sp = new Pen(Color.Silver);
        //定义大标题字体
        Font Bfont = new Font("Arial", 12, FontStyle.Bold);
        //定义一般字体
        Font font = new Font("Arial", 6);
        //定义大点的字体
        Font Tfont = new Font("Arial", 9);
        //绘制底色
        g.DrawRectangle(new Pen(Color.White, 400), 0, 0, img.Width, img.Height);
        //定义黑色过渡型笔刷
         LinearGradientBrush brush = new LinearGradientBrush(new Rectangle(0, 0, img.Width, img.Height), Color.Black, Color.Black, 1.2F, true);
        //定义蓝色过渡型笔刷
        LinearGradientBrush Bluebrush = new LinearGradientBrush(new Rectangle(0, 0, img.Width, img.Height), Color.Blue, Color.Blue, 1.2F, true);
        //绘制大标题
        g.DrawString(ds.Tables[0].Rows[0]["sendid"].ToString() + "号订单发送情况曲线图", Bfont, brush, 40, 5);
        //取得当前发送量
        int nums=0;
        for (int i = 0; i < count; i++)
          {
           nums+=Convert.ToInt32(ds.Tables[0].Rows[i]["sendnum"]);
        }
        //绘制信息简报
        string info="订单发送时间："+ds.Tables[0].Rows[0]["sendtime"].ToString()+"  曲线图生成时间："+DateTime.Now.ToString()+"  订单总量："+Tnum.ToString()+"  当前发送总量："+nums.ToString();
        g.DrawString(info, Tfont, Bluebrush, 40, 25);
        //绘制图片边框
        g.DrawRectangle(Bp, 0, 0, img.Width - 1, img.Height - 1);

        //绘制竖坐标线
        for (int i = 0; i < count; i++)
          {
           g.DrawLine(Sp, 40+20 * i, 60, 40+20 * i, 360);
        }
        //绘制时间轴坐标标签
        for (int i = 0; i < count; i+=2)
          {
             string st = Convert.ToDateTime(ds.Tables[0].Rows[i]["sendtime"]).ToString("hh:mm");
             g.DrawString(st, font, brush, 30 + 20 * i, 370);
        }
        //绘制横坐标线
        for (int i = 0; i < 10; i++)
          {
             g.DrawLine(Sp, 40, 60+30*i, 40+20*(count-1), 60+30*i);
             int s = 2500 - 50 * i * 5;
             //绘制发送量轴坐标标签
             g.DrawString(s.ToString(), font, brush, 10, 60 + 30 * i);
        }

        //绘制竖坐标轴
        g.DrawLine(Bp, 40, 55, 40, 360);
        //绘制横坐标轴
        g.DrawLine(Bp, 40, 360, 45 + 20 * (count - 1), 360);

        //定义曲线转折点
        Point[] p = new Point[count];
        for (int i = 0; i < count; i++)
          {
             p[i].X = 40 + 20 * i;
             p[i].Y = 360- Convert.ToInt32(ds.Tables[0].Rows[i]["sendnum"]) / 5*3/5;
        }
        //绘制发送曲线
        g.DrawLines(Rp, p);

        for (int i = 0; i < count; i++)
          {
            //绘制发送记录点的发送量
            g.DrawString(ds.Tables[0].Rows[i]["sendnum"].ToString(), font, Bluebrush, p[i].X, p[i].Y - 10);
            //绘制发送记录点
            g.DrawRectangle(Rp, p[i].X - 1, p[i].Y - 1, 2, 2);
        }
        //绘制竖坐标标题
        g.DrawString("发送量", Tfont, brush, 5, 40);
        //绘制横坐标标题
        g.DrawString("发送时间", Tfont, brush, 40, 385);


        //保存绘制的图片
        MemoryStream stream = new MemoryStream();
        img.Save(stream, ImageFormat.Jpeg);
        //图片输出
        page.Response.Clear();
        page.Response.ContentType = "image/jpeg";
        page.Response.BinaryWrite(stream.ToArray());

    }
}
```

嘿嘿!发表这个,就是一个图片的绘制方法的记录了.这里面还有很多不当之久,望指教.

posted on 2006-11-23 10:29 轻剑傲风 阅读(383) 评论(5) 编辑 收藏 引用 网摘 所属分类: 学习日记

OWC 组件使用

asp.net 2.0 中，要显示图型的话，可以用 ms office 2003 的 owc 组件，可以十分方便地看到图表，在工程中，

首先添加 microsoft office web components 11.0 的引用就可以了，然后要

```
using Microsoft.Office.Interop.Owc11;
```

1 生成柱状图

```
    //创建 X 坐标的值，表示月份
        int[] Month = new int[3] { 1, 2, 3 };
        //创建 Y 坐标的值，表示销售额
        double[] Count = new double[3] { 120,240,220};
        //创建图表空间
        ChartSpace mychartSpace = new ChartSpace();
        //在图表空间内添加一个图表对象
        ChChart mychart = mychartSpace.Charts.Add(0);
        //设置图表类型，本例使用柱形
        mychart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;
        //设置图表的一些属性
```

```
//是否需要图例
mychart.HasLegend = true;
//是否需要主题
mychart.HasTitle = true;
//主题内容
mychart.Title.Caption = "一季度总结";
//设置 x,y 坐标
mychart.Axes[0].HasTitle = true;
mychart.Axes[0].Title.Caption = "月份";
mychart.Axes[1].HasTitle = true;
mychart.Axes[1].Title.Caption = "销量";
//添加三个图表块
mychart.SeriesCollection.Add(0);
mychart.SeriesCollection.Add(0);
mychart.SeriesCollection.Add(0);
//设置图表块的属性
//标题
mychart.SeriesCollection[0].Caption = "一月份";
```

```
//X 坐标的值属性

mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[0]);

//y 坐标的值属性

mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[0]);

//第二个块

mychart.SeriesCollection[1].Caption = "二月份";

//X 坐标的值属性

mychart.SeriesCollection[1].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[1]);

//y 坐标的值属性

mychart.SeriesCollection[1].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[1]);

//第三个块

mychart.SeriesCollection[2].Caption = "三月份";

//X 坐标的值属性

mychart.SeriesCollection[2].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Month[2]);

//y 坐标的值属性
```

```
            mychart.SeriesCollection[2].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, Count[2]);

            //生成图片

            mychartSpace.ExportPicture(Server.MapPath(".") + @"\test.jpg", "jpg", 500, 450);

            //加载图片

            Image1.ImageUrl = Server.MapPath(".") + @"\test.jpg";

        }
```

2　生成饼状图

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

        {

            //创建 X 坐标的值，表示月份

            int[] Month ={ 1, 2, 3 };

            //创建 Y 坐标的值，表示销售额

            double[] Count ={ 120, 240, 220 };

            string strDataName = "";

            string strData = "";

            //创建图表空间

            ChartSpace mychartSpace = new ChartSpace();
```

```
//在图表空间内添加一个图表对象
ChChart mychart = mychartSpace.Charts.Add(0);
//设置每块饼的数据
for (int i = 0; i < Count.Length; i++)
{
    strDataName += Month[i] + "\t";
    strData += Count[i].ToString() + "\t";
}
//设置图表类型，本例使用柱形
mychart.Type = ChartChartTypeEnum.chChartTypePie;
//设置图表的一些属性
//是否需要图例
mychart.HasLegend = true;
//是否需要主题
mychart.HasTitle = true;
//主题内容
mychart.Title.Caption = "一季度总结";
//添加图表块
```

```
        mychart.SeriesCollection.Add(0);

        //设置图表块的属性

        //分类属性

        mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories,
(int)ChartSpecialDataSourcesEnum.chDataLiteral, strDataName);

        //值属性

        mychart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues,
(int)ChartSpecialDataSourcesEnum.chDataLiteral, strData);

       //显示百分比

        ChDataLabels mytb= mychart.SeriesCollection[0].DataLabelsCollection.Add();

        mytb.HasPercentage = true;

        //生成图片

        mychartSpace.ExportPicture(Server.MapPath(".") + @"\test.gif", "gif", 500, 450);

        //加载图片

        Image1.ImageUrl = Server.MapPath(".") + @"\test.gif";

    }
```

一个 OWC 开发范例

最近在开发一个报告生成系统，其中使用到了.NET 图表第三方控件。以下是使用 OWC9 画一个 BAR 图的范例：

```
public class ChartClass

 {

   public ChartClass()

   {

   }

public void GetChart(string category,string value1,string value2,string

title,System.Web.HttpServerUtility Server,System.Web.UI.WebControls.PlaceHolder

ChartHolder,string filename,string sdate,string edate,bool y)

   {

     OWC.ChartSpace objCSpace = new OWC.ChartSpaceClass ();

     OWC.WCChart objChart = objCSpace.Charts.Add (0);

     objChart.Type = OWC.ChartChartTypeEnum.chChartTypeBarStacked;    //图表类型

     objChart.HasLegend = true;

     objChart.HasTitle = true;

     if(y==true)
```

```
    objChart.Title.Caption= title + " 前十二个月广告主花费（" + sdate + " 至 " + edate +
")";
   else
    objChart.Title.Caption= title + " 上个月广告主花费（" + sdate + " 至 " + edate + ")";
   objChart.Axes[0].HasTitle = false;
   objChart.Axes[1].HasTitle = false;
   objChart.SeriesCollection.Add(0);
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimSeriesNames,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, "报纸");
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimCategories,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, category);
   objChart.SeriesCollection[0].SetData
(OWC.ChartDimensionsEnum.chDimValues,(int)OWC.ChartSpecialDataSourcesEnum.chDat
aLiteral,
value1);
   objChart.SeriesCollection.Add(1);
```

```
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimSeriesNames,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, "杂志");
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimCategories,
(int)OWC.ChartSpecialDataSourcesEnum.chDataLiteral, category);
    objChart.SeriesCollection[1].SetData
(OWC.ChartDimensionsEnum.chDimValues,(int)OWC.ChartSpecialDataSourcesEnum.chDat
aLiteral,
value2);
    object ob = "#66FF66";
    objChart.SeriesCollection[0].Interior.set_Color(ref ob);
//不知道为什么，VB 里，可以使用 Interior.Color="#66FF66"来直接设置颜色，而 C#里必须使
用 set_Color 方法，而且参数必须是 ref
object。
    objChart.SeriesCollection[0].Border.set_Color(ref ob);
    ob="#6699ff";
    objChart.SeriesCollection[1].Interior.set_Color(ref ob);
    objChart.SeriesCollection[1].Border.set_Color(ref ob);
objChart.SeriesCollection[0].Border.set_Weight(OWC.LineWeightEnum.owcLineWeightThin
);
objChart.SeriesCollection[1].Border.set_Weight(OWC.LineWeightEnum.owcLineWeightThin
);
    ob="#FFFFFF";
    objChart.PlotArea.Border.set_Color(ref ob);
    objChart.PlotArea.Interior.set_Color(ref ob);
    objChart.Axes[0].Font.set_Name("Arial");
    objChart.Axes[0].Font.set_Size(9);
    objChart.Axes[1].MajorTickMarks=OWC.ChartTickMarkEnum.chTickMarkInside;
    objChart.Axes[1].MinorTickMarks=OWC.ChartTickMarkEnum.chTickMarkNone;
    ob="#CCCCCC";
    objChart.Axes[0].MajorGridlines.Line.set_Color(ref ob);
    objChart.Axes[1].Font.set_Name("宋体");
    objChart.Axes[1].Font.set_Size(9);
//      objChart.SeriesCollection[0].DataLabelsCollection.Add();
//      objChart.SeriesCollection[1].DataLabelsCollection.Add();
//      objChart.SeriesCollection[0].DataLabelsCollection[0].HasValue=true;
//     ob="red";

//     objChart.SeriesCollection[0].DataLabelsCollection[0].Font.set_Color(ref

ob);

//     objChart.SeriesCollection[1].DataLabelsCollection[0].Font.set_Color(ref

ob);

//

objChart.SeriesCollection[0].DataLabelsCollection[0].Position=OWC.ChartDataLabelPositio
nEnum.chLabelPositionOutsideBase;

//

objChart.SeriesCollection[1].DataLabelsCollection[0].Position=OWC.ChartDataLabelPosition
Enum.chLabelPositionOutsideEnd;

//上面几段注释掉的内容，是在数据柱上显示数据值的设置代码



    objChart.Legend.Font.set_Name("宋体");

    objChart.Legend.Font.set_Size(9);

    objChart.Title.Font.set_Name("宋体");

    objChart.Title.Font.set_Size(11);

    string strAbsolutePath = Server.MapPath(".")+ @"\TempFiles\" + filename;

//文件保存的位置
```

```
   objCSpace.ExportPicture(strAbsolutePath, "GIF", 600, 350);   //文件的分辨率其他属性

   string strRelativePath = "./TempFiles/" + filename;

   string strImageTag = "<IMG SRC='" + strRelativePath + "'/>";

   ChartHolder.Controls.Add(new LiteralControl(strImageTag));

//将图片填充到 PlaceHolder 对象 ChartHoler 中。

  }

 }
```

OWC 学习笔记-Spreadsheet 插入行/列

在 owc 提供的 Spreadsheet api 中，没有直接添加行列的方法，可以使用执行命令的方式实现

添加新行在第 3 行，代码如下：

```
   var ssConstants = Spreadsheet1.Constants;

   Spreadsheet1.ActiveSheet.Row(3).Select();

   Spreadsheet1.Commands(ssConstants.ssCommandInsertRows).Execute();
```

添加新列在第 3 列，代码如下：

```
   var ssConstants = Spreadsheet1.Constants;

   Spreadsheet1.ActiveSheet.cells(2,3).Select();

   Spreadsheet1.Commands(ssConstants.ssCommandInsertCols).Execute();
```

C#编写OWC11组件源代码(2)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Microsoft.Office.Interop;
namespace FishPro
{
  /// <summary>
  /// 利用OWC组件生成柱状图示例
  /// </summary>
  public class TestOWC : System.Web.UI.Page
```

```
 {

  private void Page_Load(object sender, System.EventArgs e)

  {

    string strCategory = "1" + '\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "5" + '\t' + "6" + '\t'+"7" +
'\t' + "8" + '\t' + "9" + '\t'+"10" + '\t' + "11" + '\t' + "12" + '\t';

    string strValue = "9000" + '\t' + "8000" + '\t' + "4007" + '\t'+"10" + '\t' + "12760" + '\t' +
"6678" + '\t'+"10000" + '\t' + "20999" + '\t' + "3567" + '\t'+"456" + '\t' + "125" + '\t' +
"66765" + '\t';

    string mTitle="建科院月报表分析图";

    string xTitle="月份";

    string yTitle="工作量";

    int imgWidth=780;

    int imgHeight=600;

    int chartType=0;


//this.CreateChartSmoothLine(strCategory,strValue,mTitle,xTitle,yTitle,imgWidth,imgHeigh
t,chartType);

    FishPro.OWCChart11 chart = new OWCChart11(Server.MapPath("."),"费用
",mTitle,1,xTitle,yTitle,imgWidth,imgHeight);

    chart.OCategory=strCategory;
```

```
      chart.OValue=strValue;

      if(chart.Create())

      {

        Response.Write( "<IMG SRC='"+chart.FileName+"'/>");

      }

      else

      {

        Response.Write("shibai");

      }

//      // 在此处放置用户代码以初始化页面

//      string strCategory = "1" + '\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "5" + '\t' + "6" + '\t'+"7"
+ '\t' + "8" + '\t' + "9" + '\t'+"10" + '\t' + "11" + '\t' + "12" + '\t';

//      string strValue = "9" + '\t' + "8" + '\t' + "4" + '\t'+"10" + '\t' + "12" + '\t' + "6" + '\t'+"1" +
'\t' + "2" + '\t' + "3" + '\t'+"4" + '\t' + "12" + '\t' + "6" + '\t';

//

//      //声明对象

//      Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new
Microsoft.Office.Interop.Owc11.ChartSpaceClass();

//      Microsoft.Office.Interop.Owc11.ChChart ThisChChart    = ThisChart.Charts.Add(0);
//      Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries =
ThisChChart.SeriesCollection.Add(0);

//

//

//      //显示图例

//      ThisChChart.HasLegend = true;

//      //标题

//      ThisChChart.HasTitle = true;

//      ThisChChart.Title.Caption = "统计图";

//      //给定 x,y 轴图示说明

//      ThisChChart.Axes[0].HasTitle = true;

//      ThisChChart.Axes[1].HasTitle = true;

//      ThisChChart.Axes[0].Title.Caption = "月份";

//      ThisChChart.Axes[1].Title.Caption = "数量";

//

//      //图表类型

//      //ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered3D;//3D
柱状图
```

//　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeArea; //折线面积图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeArea3D;//折线 3D 面积图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaOverlapped3D;//折线 3D 面积图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked;//折线面积图加边框

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked100;//折线面积图加边框百分比图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked1003D;//折线 3D 面积图加边框百分比图

////　ThisChChart.Type =

Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeAreaStacked3D;//折线 3D 面积图加边框

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarClustered;//横道图

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarClustered3D;//横道图 3D

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked;//横道图 3D

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked100;//横道图 3D 百分比图

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBarStacked1003D;//横道图 3D 百分比图

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBubble; //测试不通过

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBubbleLine;//测试不通过

//// ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D

```
////     ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered;//柱状
图 3D

////     ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnClustered3D;//柱
状图 3D

////     ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked;//柱状图

////     ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked100;//柱
状图 3D  百分比图

////     ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumnStacked1003D;//
柱状图 3D 百分比图

//     //ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartLegendPositionEnum.chLegendPositionBottom;

//     //旋转

//     ThisChChart.Rotation    = 360;

//     ThisChChart.Inclination = 10;

//     //背景颜色

//     ThisChChart.PlotArea.Interior.Color = "red";

//     ThisChChart.PlotArea.Floor.Interior.Color = "green";
```

```
//
//      ThisChChart.Overlap = 50;
//
//      /////给定 series 的名字
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesNames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),"日期");
//      //给定分类
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategories,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strCategory);
//      //给定值
//
ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strValue);
//
//      //导出图像文件
//      try
//    {
//
//       ThisChart.ExportPicture(Server.MapPath("chart.gif"),"gif",600,350);
//       Response.Write( "<IMG SRC='chart.gif'/>");
//    }
//
//    catch(Exception ee)
//
//    {
//
//    }
  }
  //平滑曲线图
  //ChartType 0 默认柱状图 1 横道图 2 平滑曲线图
  public   void   CreateChartSmoothLine(string strCategory,string strValue,string
mTitle,string xTitle,string yTitle,int imgWidth,int imgHeight,int chartType)
  {
   //声明对象
```

```
Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new Microsoft.Office.Interop.Owc11.ChartSpaceClass();
Microsoft.Office.Interop.Owc11.ChChart ThisChChart   = ThisChart.Charts.Add(0);
Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries = ThisChChart.SeriesCollection.Add(0);


//显示图例
ThisChChart.HasLegend = true;
//显示标题选项
ThisChChart.HasTitle = true;
ThisChChart.Title.Font.Name="黑体";
ThisChChart.Title.Font.Size=14;
ThisChChart.Title.Caption = mTitle;//from
//x,y 轴说明
//x
ThisChChart.Axes[0].HasTitle=true;
ThisChChart.Axes[0].Title.Font.Name="黑体";
ThisChChart.Axes[0].Title.Font.Size=12;
ThisChChart.Axes[0].Title.Caption=xTitle;
```

```
    ThisChChart.Axes[1].HasTitle=true;

    ThisChChart.Axes[1].Title.Font.Name="黑体";

    ThisChChart.Axes[1].Title.Font.Size=12;

    ThisChChart.Axes[1].Title.Caption=yTitle;

    //图表类型

    switch(chartType)

    {

     case 0:

      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D

      break;

     case 1:

      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D

      break;

     case 2:

      ThisChChart.Type =
Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线
图
```

```
     break;
    case 3:
     ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypePie;//圆饼图
     break;
   }
   //ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图
   //旋转
   ThisChChart.Rotation  = 360;
   ThisChChart.Inclination = 10;
   //背景颜色
   ThisChChart.PlotArea.Interior.Color = "red";
   ThisChChart.PlotArea.Floor.Interior.Color = "green";
   //ThisChChart.Overlap = 50;
   /////给定 series 的名字

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesN
ames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),"日期");

   //给定分类

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategories,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strCategory);

   //给定值

ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),strValue);

   //导出图像文件

   try

   {

    ThisChart.ExportPicture(Server.MapPath("chart.gif"),"gif",imgWidth,imgHeight);

    Response.Write( "<IMG SRC='chart.gif'/>");

   }

   catch(Exception ee)

   {
   }
  }
  #region Web 窗体设计器生成的代码
  override protected void OnInit(EventArgs e)
  {
   //
   // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
   //
   InitializeComponent();
   base.OnInit(e);
  }
  /// <summary>
  /// 设计器支持所需的方法 - 不要使用代码编辑器修改
  /// 此方法的内容。
  /// </summary>
  private void InitializeComponent()
  {
   this.Load += new System.EventHandler(this.Page_Load);
```

```
  }

  #endregion

 }

}
```

使用方法：

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

    {

        MyOWC1.DataSource = createTable();

        MyOWC1.Attribute.KeyField = "name";

        MyOWC1.Attribute.ValueFileds = new string[] { "vistis", "showname" };

        MyOWC1.Attribute.signValues = new string[] { "美国", "中国" };

        MyOWC1.Attribute.startDate = new DateTime(2007, 4, 1);

        MyOWC1.Attribute.endDate = new DateTime(2007, 4, 19);

        MyOWC1.Draw();

    }

    DataTable createTable()

    {
```

```
    Random rnd = new Random();

    DataTable dt = new DataTable();

    dt.Columns.Add("name");

    dt.Columns.Add("vistis");

    dt.Columns.Add("showname");

    dt.Columns.Add("percent");

    for (int i = 0; i < 8; i++)

    {

        DataRow dr = dt.NewRow();

        dr["name"] = "2007-4-"+(i+1);

        dr["vistis"] = rnd.Next(100, 1000).ToString();

        dr["showname"] = rnd.Next(100, 1000).ToString();

        dr["percent"] = rnd.Next(100, 1000).ToString();

        dt.Rows.Add(dr);

    }

    return dt;

}
```

C#编写OWC11 组件(转贴)

```
 using System;

using System.Data;

using System.Text;



using System.Text;
```

```
private string m_AxesYTitle;

private int m_PicWidth;

private int m_PicHeight;

private int m_Type;

private string m_FileName;



        /// <summary>

/// 保存图片的路径和名称,物理路径

/// </summary>

public string SavePath

        {

            get{return m_SavePath;}

            set{m_SavePath=value;}

}



        /// <summary>

/// 直接获得类型

/// </summary>

public string OCategory

        {

            get{return m_Category;}

            set{m_Category=value;}
```

```
}

        /// <summary>
/// 直接获得值
/// </summary>
public string OValue
        {
            get{return m_Value;}
            set{m_Value=value;}
}



        /// <summary>
/// 以表格 DataTable 的形式获取原始数据
/// </summary>
public DataTable DataSource
        {
            get{return m_DataSource;}
    set
            {
       m_DataSource=value;
       m_Category=GetColumnsStr(m_DataSource);
```

```
        m_Value=GetValueStr(m_DataSource);

    }

}



         /// <summary>

/// 简要说明

/// </summary>

public string SeriesName

         {

             get{return m_SeriesName;}

             set{m_SeriesName=value;}

}



         /// <summary>

/// 图表的总标题，说明图表的简单意思

/// </summary>

public string Title

         {

             get{return m_Title;}

             set{m_Title=value;}

}
```

```
        /// <summary>
/// 图表横坐标标题，说明横坐标的意义
/// </summary>
public string AxesXTitle
        {
            get{return m_AxesXTitle;}
            set{m_AxesXTitle=value;}
}

        /// <summary>
/// 图表纵坐标标题，说明纵坐标的意义
/// </summary>
public string AxesYTitle
        {
            get{return m_AxesYTitle;}
            set{m_AxesYTitle=value;}
}

        /// <summary>
///  生成的图片宽度
/// </summary>
public int PicWidth
```

```
         {
             get{return m_PicWidth;}
             set{m_PicWidth=value;}
}

         /// <summary>
///  生成的图片高度
/// </summary>
public int PicHeight
         {
             get{return m_PicHeight;}
             set{m_PicHeight=value;}
}

         /// <summary>
/// 类型
/// chChartTypeColumnStacked100 =2
///chChartTypeColumnStacked1003D = 49
///chChartTypeColumnStacked3D = 48
///chChartTypeCombo = -1
///chChartTypeCombo3D = -2
///chChartTypeDoughnut = 32
```

```
///chChartTypeDoughnutExploded = 33

///chChartTypeLine = 6

///chChartTypeLine3D = 54

///chChartTypeLineMarkers=  7

///chChartTypeLineOverlapped3D=  55

///chChartTypeLineStacked = 8

///chChartTypeLineStacked100  =10

///chChartTypeLineStacked1003D=  57

///chChartTypeLineStacked100Markers = 11

///chChartTypeLineStacked3D = 56

///chChartTypeLineStackedMarkers = 9

///chChartTypePie = 18

///chChartTypePie3D =58

///chChartTypePieExploded = 19

///chChartTypePieExploded3D = 59

///chChartTypePieStacked = 20

///chChartTypePolarLine = 42

///chChartTypePolarLineMarkers = 43

///chChartTypePolarMarkers = 41

///chChartTypePolarSmoothLine = 44

///chChartTypePolarSmoothLineMarkers = 45

///chChartTypeRadarLine=  34
```

```
///chChartTypeRadarLineFilled = 36
///chChartTypeRadarLineMarkers=  35
///chChartTypeRadarSmoothLine = 37
///chChartTypeRadarSmoothLineMarkers = 38
///chChartTypeScatterLine = 25
///chChartTypeScatterLineFilled = 26
///chChartTypeScatterLineMarkers = 24
///chChartTypeScatterMarkers = 21
///chChartTypeScatterSmoothLine = 23
///chChartTypeScatterSmoothLineMarkers = 22
///chChartTypeSmoothLine = 12
///chChartTypeSmoothLineMarkers = 13
///chChartTypeSmoothLineStacked = 14
///chChartTypeSmoothLineStacked100 = 16
///chChartTypeSmoothLineStacked100Markers = 17
///chChartTypeSmoothLineStackedMarkers = 15
///chChartTypeStockHLC = 39
///chChartTypeStockOHLC = 40
/// </summary>
public int Type
     {
        get{return m_Type;}
```

```
            set{m_Type=value;}
    }

    public string FileName
        {
            get{return m_FileName;}
      set
            {
        m_FileName=value;
        if(m_FileName==null||m_FileName=="")
                {
            m_FileName=DateTime.Now.ToString("yyyyMMddHHmmssff")+".gif";//生成
图片名称
        }
      }
    }

        /// <summary>
    /// 是 DataTable 的转换
    /// </summary>
    /// <param name="dt"></param>
    /// <returns></returns>
```

```
private string GetColumnsStr(DataTable dt)

     {

  StringBuilder strList=new StringBuilder();

  foreach(DataRow r in dt.Rows)

       {

    strList.Append(r[0].ToString()+'\t');

  }

  return strList.ToString();

}



     /// <summary>

/// 是 DataTable 的转换

/// </summary>

/// <param name="dt"></param>

/// <returns></returns>

private string GetValueStr(DataTable dt)

     {

  StringBuilder strList=new StringBuilder();

  foreach(DataRow r in dt.Rows)

       {

    strList.Append(r[1].ToString()+'\t');

  }
```

```
    return strList.ToString();

}


#endregion


         #region 枚举类型

         /// <summary>

/// 枚举类型 对应于 OWC 的图表类型

/// </summary>

public  enum ChartType:int

         {


             /// <summary>

    /// 面积图

    /// </summary>

    chChartTypeArea = 29 ,

             /// <summary>

    /// 面积图 3D

    /// </summary>

    chChartTypeArea3D = 60 ,

             /// <summary>

    /// 面积图重复
    /// </summary>

    chChartTypeAreaOverlapped3D = 61 ,

             /// <summary>

    /// 堆积面积图

    /// </summary>

    chChartTypeAreaStacked  =30 ,

             /// <summary>

    /// 堆积面积图百分比图

    /// </summary>

    chChartTypeAreaStacked100 = 31 ,

             /// <summary>

    /// 堆积面积图百分比图 3D

    /// </summary>

    chChartTypeAreaStacked1003D  =63 ,

             /// <summary>

    /// 堆积面积图 3D

    /// </summary>

    chChartTypeAreaStacked3D = 62 ,

             /// <summary>

    /// 横道图 3D

    /// </summary>

    chChartTypeBar3D = 50 ,
             /// <summary>

    /// 横道图串风格

    /// </summary>

    chChartTypeBarClustered=  3 ,

             /// <summary>

    /// 横道图串风格 3D

    /// </summary>

    chChartTypeBarClustered3D=  51 ,

             /// <summary>

    /// 堆横道图

    /// </summary>

    chChartTypeBarStacked = 4 ,

             /// <summary>

    /// 堆横道图百分比图

    /// </summary>

    chChartTypeBarStacked100  =5 ,

             /// <summary>

    /// 堆横道图百分比图 3D

    /// </summary>

    chChartTypeBarStacked1003D = 53 ,

             /// <summary>

    /// 堆横道图 3D
    /// </summary>
    chChartTypeBarStacked3D  =52 ,
             /// <summary>
    /// 气泡图
    /// </summary>
    chChartTypeBubble = 27 ,
             /// <summary>
    /// 线形气泡图
    /// </summary>
    chChartTypeBubbleLine =28 ,
             /// <summary>
    /// 柱形图
    /// </summary>
    chChartTypeColumn3D  =46 ,
             /// <summary>
    /// 3D 柱形图
    /// </summary>
    chChartTypeColumnClustered = 0 ,
             /// <summary>
    /// 3D 串柱形图
    /// </summary>
    chChartTypeColumnClustered3D = 47 ,
             /// <summary>
    /// 重叠柱形图
    /// </summary>
    chChartTypeColumnStacked =1 ,
             /// <summary>
    /// 100%重叠柱形图
    /// </summary>
    chChartTypeColumnStacked100 =2 ,
             /// <summary>
    /// 100%3D 重叠柱形图
    /// </summary>
    chChartTypeColumnStacked1003D = 49 ,
             /// <summary>
    /// 3D 柱形图
    /// </summary>
    chChartTypeColumnStacked3D = 48 ,
             /// <summary>
    /// 组合图
    /// </summary>
    chChartTypeCombo = -1 ,
             /// <summary>
    /// 3D 组合图
    /// </summary>

    chChartTypeCombo3D = -2 ,

             /// <summary>

    /// 环形图

    /// </summary>

    chChartTypeDoughnut = 32 ,

             /// <summary>

    /// 破式环形图

    /// </summary>

    chChartTypeDoughnutExploded = 33 ,

             /// <summary>

    /// 折线图

    /// </summary>

    chChartTypeLine = 6,

             /// <summary>

    /// 3D 折线图

    /// </summary>

    chChartTypeLine3D = 54 ,

             /// <summary>

    /// 制造折线图

    /// </summary>

    chChartTypeLineMarkers=  7,
             /// <summary>
    /// 重复折线图
    /// </summary>
    chChartTypeLineOverlapped3D=  55,
             /// <summary>
    /// 重叠折线图
    /// </summary>
    chChartTypeLineStacked = 8 ,
             /// <summary>
    /// 100%重叠折线图
    /// </summary>
    chChartTypeLineStacked100  =10 ,
             /// <summary>
    /// 100%3D 重叠折线图
    /// </summary>
    chChartTypeLineStacked1003D=  57,
             /// <summary>
    /// 制造 100%重叠折线图
    /// </summary>
    chChartTypeLineStacked100Markers = 11 ,
             /// <summary>
    /// 3D 重叠折线图
    /// </summary>
    chChartTypeLineStacked3D = 56 ,
             /// <summary>
    /// 制造重叠折线图
    /// </summary>
    chChartTypeLineStackedMarkers = 9 ,
             /// <summary>
    /// 饼图
    /// </summary>
    chChartTypePie = 18 ,
             /// <summary>
    /// 3D 饼图
    /// </summary>
    chChartTypePie3D =58 ,
             /// <summary>
    /// 破式饼图
    /// </summary>
    chChartTypePieExploded = 19 ,
             /// <summary>
    /// 3D 破式饼图
    /// </summary>
    chChartTypePieExploded3D = 59 ,
             /// <summary>
    /// 重叠饼图
    /// </summary>
    chChartTypePieStacked = 20 ,
             /// <summary>
    /// 极坐标图
    /// </summary>
    chChartTypePolarLine = 42 ,
             /// <summary>
    /// 制造线形极坐标图
    /// </summary>
    chChartTypePolarLineMarkers = 43,
             /// <summary>
    /// 制造极坐标图
    /// </summary>
    chChartTypePolarMarkers = 41 ,
             /// <summary>
    /// 平滑线形极坐标图
    /// </summary>
    chChartTypePolarSmoothLine = 44 ,
             /// <summary>
    /// 制造平滑线形极坐标图
    /// </summary>

    chChartTypePolarSmoothLineMarkers = 45 ,

             /// <summary>

    /// 雷达图

    /// </summary>

    chChartTypeRadarLine=  34 ,

             /// <summary>

    /// 填充雷达图

    /// </summary>

    chChartTypeRadarLineFilled = 36 ,

             /// <summary>

    /// 制造雷达图

    /// </summary>

    chChartTypeRadarLineMarkers=  35 ,

             /// <summary>

    /// 平滑雷达图

    /// </summary>

    chChartTypeRadarSmoothLine = 37 ,

             /// <summary>

    /// 制造平滑雷达图

    /// </summary>

    chChartTypeRadarSmoothLineMarkers = 38 ,
             /// <summary>
    /// 线形散点图
    /// </summary>
    chChartTypeScatterLine = 25 ,
             /// <summary>
    /// 填充线形散点图
    /// </summary>
    chChartTypeScatterLineFilled = 26 ,
             /// <summary>
    /// 制造线形散点图
    /// </summary>
    chChartTypeScatterLineMarkers = 24 ,
             /// <summary>
    /// 制造散点图
    /// </summary>
    chChartTypeScatterMarkers = 21 ,
             /// <summary>
    /// 平滑散点图
    /// </summary>
    chChartTypeScatterSmoothLine = 23 ,
             /// <summary>
    /// 制造平滑散点图
    /// </summary>
    chChartTypeScatterSmoothLineMarkers = 22 ,
             /// <summary>
    /// 平滑线图
    /// </summary>
    chChartTypeSmoothLine = 12 ,
             /// <summary>
    /// 制造平滑线图
    /// </summary>
    chChartTypeSmoothLineMarkers = 13 ,
             /// <summary>
    /// 重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked = 14 ,
             /// <summary>
    /// 100%重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked100 = 16 ,
             /// <summary>
    /// 制造 100%重叠平滑线图
    /// </summary>
    chChartTypeSmoothLineStacked100Markers = 17 ,
            /// <summary>

    /// 制造重叠平滑线图

    /// </summary>

    chChartTypeSmoothLineStackedMarkers = 15 ,

            /// <summary>

    /// 股价图

    /// </summary>

    chChartTypeStockHLC = 39 ,

            /// <summary>

    /// 股价图 O 型

    /// </summary>

    chChartTypeStockOHLC = 40



}

#endregion



        #region 构造函数

public OWCChart11()

        {

    //

    // TODO: 在此处添加构造函数逻辑

    //
```

```
}

public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType)
     {
  m_SavePath=SavePath;
  m_SeriesName=SeriesName;
  m_Title=Title;
  m_Type=ChartType;
}

 public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType,string AxesXTitle,string AxesYTitle)
     {
  m_SavePath=SavePath;
  m_SeriesName=SeriesName;
  m_Title=Title;
  m_AxesXTitle=AxesXTitle;
  m_AxesYTitle=AxesYTitle;
  m_Type=ChartType;
}

 public OWCChart11(string SavePath,string SeriesName,string Title,int ChartType,string AxesXTitle,string AxesYTitle,int PicWidth,int PicHeight)
```

```
      {
    m_SavePath=SavePath;
    m_SeriesName=SeriesName;
    m_Title=Title;
    m_AxesXTitle=AxesXTitle;
    m_AxesYTitle=AxesYTitle;
    m_PicWidth=PicWidth;
    m_PicHeight=PicHeight;
    m_Type=ChartType;
}
#endregion

public bool Create()
      {
    //声明对象
    Microsoft.Office.Interop.Owc11.ChartSpace ThisChart = new  Microsoft.Office.Interop.Owc11.ChartSpaceClass();
    Microsoft.Office.Interop.Owc11.ChChart ThisChChart  = ThisChart.Charts.Add(0);
    Microsoft.Office.Interop.Owc11.ChSeries ThisChSeries = ThisChChart.SeriesCollection.Add(0);

    //显示图例
    ThisChChart.HasLegend = true;
```

```
            //显示标题选项
            ThisChChart.HasTitle = true;
            ThisChChart.Title.Font.Name="黑体";
            ThisChChart.Title.Font.Size=14;
            ThisChChart.Title.Caption = m_Title;//from

            //x,y 轴说明
            //x
            ThisChChart.Axes[0].HasTitle=true;
            ThisChChart.Axes[0].Title.Font.Name="黑体";
            ThisChChart.Axes[0].Title.Font.Size=12;
            ThisChChart.Axes[0].Title.Caption=m_AxesXTitle;

            ThisChChart.Axes[1].HasTitle=true;
            ThisChChart.Axes[1].Title.Font.Name="黑体";
            ThisChChart.Axes[1].Title.Font.Size=12;
            ThisChChart.Axes[1].Title.Caption=m_AxesYTitle;

            //图表类型
             ThisChChart.Type=(Microsoft.Office.Interop.Owc11.ChartChartTypeEnum) m_Type;
//          switch(m_Type)
```

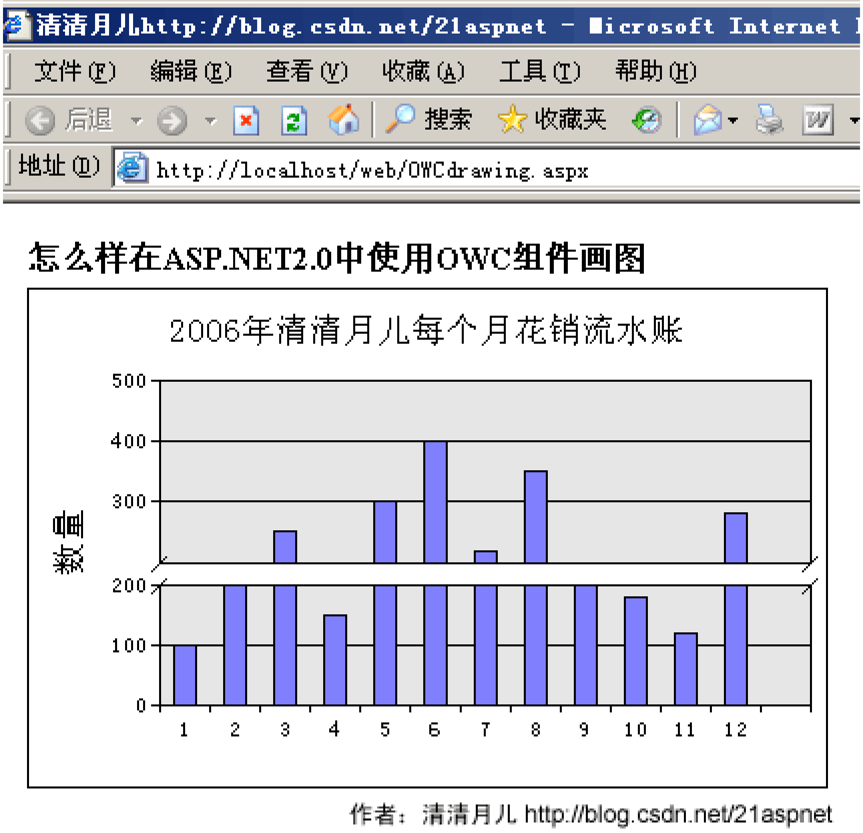
```
        //            {
        //                case 0:
//                    ThisChChart.Type =(Microsoft.Office.Interop.Owc11.ChartChartTypeEnum) m_Type;// Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeColumn3D;//柱状图 3D
        //                    break;
        //                case 1:
//                    ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeBar3D;//横道图 3D
        //                    break;
        //                case 2:
//                    ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypeSmoothLine;//平滑曲线图
        //                    break;
        //                case 3:
//                    ThisChChart.Type = Microsoft.Office.Interop.Owc11.ChartChartTypeEnum.chChartTypePie;//圆饼图
        //
        //
        //                    break;
        //            }
                    //旋转
                    ThisChChart.Rotation  = 360;
```

```
            ThisChChart.Inclination = 10;

            //背景颜色

            ThisChChart.PlotArea.Interior.Color = "red";

            //底座颜色

            ThisChChart.PlotArea.Floor.Interior.Color = "green";



            //ThisChChart.Overlap = 50;



            //给定 series 的名字

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimSeriesNames,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_SeriesName);

            //给定分类

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimCategories,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_Category);

            //给定值

             ThisChSeries.SetData(Microsoft.Office.Interop.Owc11.ChartDimensionsEnum.chDimValues,Microsoft.Office.Interop.Owc11.ChartSpecialDataSourcesEnum.chDataLiteral.GetHashCode(),m_Value);



             Microsoft.Office.Interop.Owc11.ChDataLabels dl=ThisChChart.SeriesCollection[0].DataLabelsCollection.Add();

            dl.HasValue=true;

            //dl.HasPercentage=true;
            //导出图像文件
            try
                 {
              if(m_FileName==null||m_FileName=="")
                   {
                m_FileName=DateTime.Now.ToString("yyyyMMddHHmmssff")+".gif";//生成图片名称
              }
              ThisChart.ExportPicture(m_SavePath+"\\"+m_FileName,"gif",m_PicWidth,m_PicHeight);
              return true;
            }
            catch(Exception ee)
                 {
              return false;
            }
            return false;
          }


        }
      }
```

ASP.NET2.0 轻松搞定统计图表

本文讲述如何绘制**条形图，折线图，柱形图，面积图**等常见图形。

效果图：



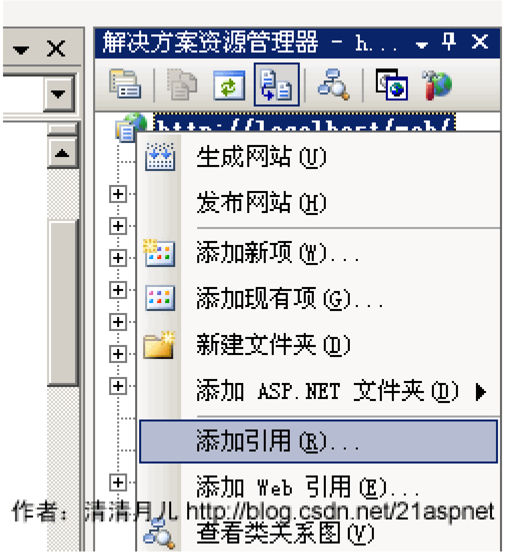
清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F)
编辑(E)
查看(V)
收藏(A)
工具(T)
帮助(H)
后退
搜索
收藏夹
地址(D)
http://localhost/web/OWCdrawing.aspx
怎么样在ASP.NET2.0中使用OWC组件画图
2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12
作者：清清月儿 http://blog.csdn.net/21aspnet

手把手教程：

原理：**OWC** 是 Office Web Compent 的缩写，即 Microsoft 的 Office Web 组件，它为在 Web 中绘制图形提供了灵活的同时也是最基本的机制。在一个 intranet 环境中，如果可以假设客户机上存在特定的浏览器和一些功能强大的软件（如 IE6 和 Office 2000/XP/2003），那么就有能力利用 Office Web 组件提供一个交互式图形开发环境。这种模式下，客户端工作站将在整个任务中分担很大的比重。理论上说 **Excel** 能做的图都可以通过 OWC 画。

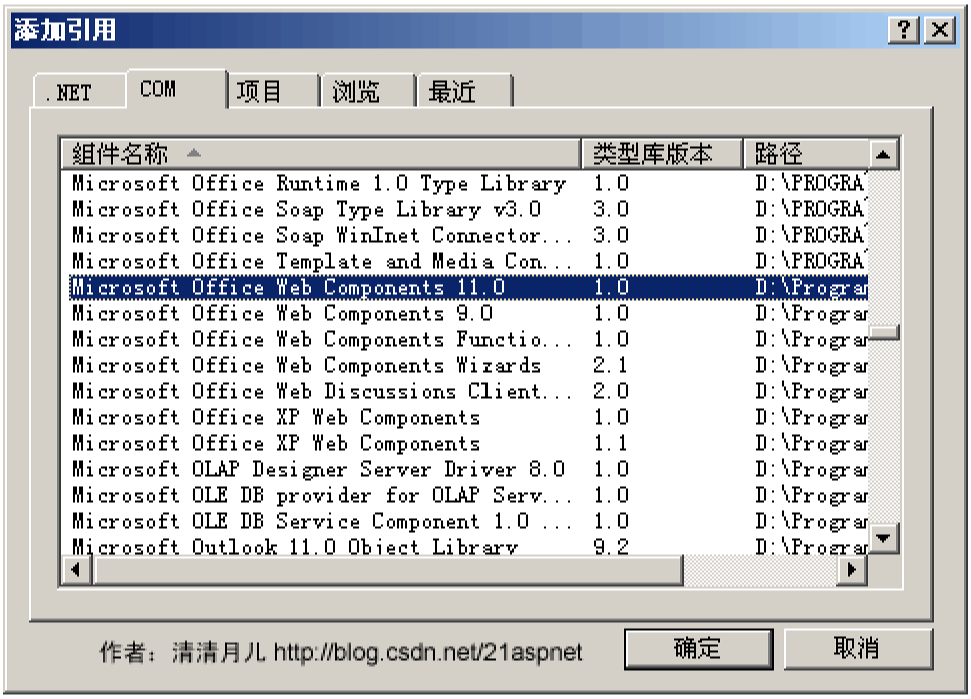
第一步：

右键点击网站根目录引用。如图所示：



第二步：

点击“添加引用”后弹出一个窗口，添加 OWC 的引用。如图所示：



组件名称	类型库版本	路径
Microsoft Office Runtime 1.0 Type Library	1.0	D:\PROGRA
Microsoft Office Soap Type Library v3.0	3.0	D:\PROGRA
Microsoft Office Soap WinInet Connector...	3.0	D:\PROGRA
Microsoft Office Template and Media Con...	1.0	D:\PROGRA
Microsoft Office Web Components 11.0	1.0	D:\Progra
Microsoft Office Web Components 9.0	1.0	D:\Progra
Microsoft Office Web Components Functio...	1.0	D:\Progra
Microsoft Office Web Components Wizards	2.1	D:\Progra
Microsoft Office Web Discussions Client...	2.0	D:\Progra
Microsoft Office XP Web Components	1.0	D:\Progra
Microsoft Office XP Web Components	1.1	D:\Progra
Microsoft OLAP Designer Server Driver 8.0	1.0	D:\Progra
Microsoft OLE DB provider for OLAP Serv...	1.0	D:\Progra
Microsoft OLE DB Service Component 1.0 ...	1.0	D:\Progra
Microsoft Outlook 11.0 Object Library	9.2	D:\Progra

点“确定”。

第三步：

代码中引用 Microsoft.Office.Interop.Owc11。

全部代码

后台代码：

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.Data.SqlClient;   //添加数据操作引用
using Microsoft.Office.Interop.Owc11;//添加 Office 组件引用

public partial class OWCdrawing : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //连接数据库并获取特定字符串
        string strSeriesName = "图例 1";
        string ConnectString = "Server=(local);DataBase=web;Uid=sa;Pwd=sa";
        string Sql = "SELECT month,Allcount FROM Chart";
        SqlConnection myConn = new SqlConnection(ConnectString);
        myConn.Open();
        SqlDataAdapter Da = new SqlDataAdapter(Sql, myConn);
        DataSet ds = new DataSet();
        Da.Fill(ds);

        //存放月
        string[] MonNum = new string[12];
        //存放数据
        string[] MonCount = new string[12];
        //为数组赋值
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            MonNum[i] = ds.Tables[0].Rows[i][0].ToString();
            MonCount[i] = ds.Tables[0].Rows[i][1].ToString();
        }
        //为 x 轴指定特定字符串，以便显示数据
        string strXdata = String.Empty;
        foreach (string strData in MonNum)
        {
            strXdata += strData + "\t";
        }
        string strYdata = String.Empty;
        //为 y 轴指定特定的字符串，以便与 x 轴相对应
        foreach (string strValue in MonCount)
        {
```

```
    strYdata += strValue + "\t";
}
//创建 ChartSpace 对象来放置图表
ChartSpace laySpace = new ChartSpaceClass();
//在 ChartSpace 对象中添加图表
ChChart InsertChart = laySpace.Charts.Add(0);
//指定绘制图表的类型。类型可以通过 OWC.ChartChartTypeEnum 枚举值得到
//InsertChart.Type = ChartChartTypeEnum.chChartTypeLine;//折线图
//InsertChart.Type = ChartChartTypeEnum.chChartTypeArea;//面积图
//InsertChart.Type = ChartChartTypeEnum.chChartTypeBarClustered;//条形图
InsertChart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;//柱形图


//指定图表是否需要图例标注
InsertChart.HasLegend = false;


InsertChart.HasTitle = true;//为图表添加标题
InsertChart.Title.Caption = "2006 年清清月儿每个月花销流水账";//标题名称
//为 x,y 轴添加图示说明
InsertChart.Axes[0].HasTitle = true;
InsertChart.Axes[0].Title.Caption = "";//月份
InsertChart.Axes[1].HasTitle = true;
InsertChart.Axes[1].Scaling.SplitMinimum = 200;
InsertChart.Axes[1].Title.Caption = "数量";
//添加一个 series 系列
InsertChart.SeriesCollection.Add(0);
//给定 series 系列的名字
InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimSeriesNames, +(int)ChartSpecialDataSourcesEnum.chDataLiteral, strSeriesName);
//给定分类
InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, +(int)ChartSpecialDataSourcesEnum.chDataLiteral, strXdata);
//给定值
InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSourcesEnum.chDataLiteral, strYdata);
//输出文件.
string strAbsolutePath = (Server.MapPath(".")) + "\\ShowData.gif";
laySpace.ExportPicture(strAbsolutePath, "GIF", 400, 250);
```

```
        //创建 GIF 文件的相对路径.
        string strRelativePath = "./ShowData.gif";

        //把图片添加到 placeholder 中，并在页面上显示
        string strImageTag = "<IMG SRC='" + strRelativePath + "'/>";
        this.PlaceHolder1.Controls.Add(new LiteralControl(strImageTag));
    }
}
```

前台代码：

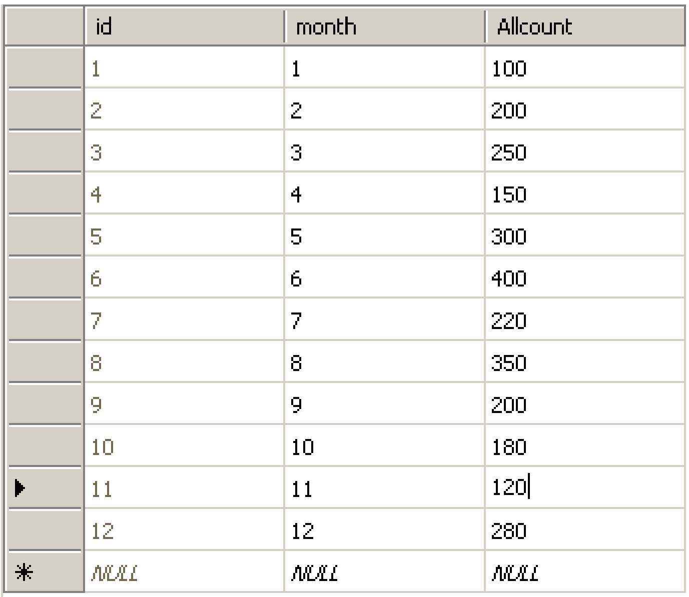
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="OWCdrawing.aspx.cs" Inherits="OWCdrawing" %>

<!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>清清月儿 http://blog.csdn.net/21aspnet</title>
</head>
<body>
    <form id="form1" runat="server">
   <div style="text-align: left">
        <table style="width: 600px">
            <tr>
                <td colspan="3" style="height: 20px">
                    <strong>怎么样在 ASP.NET2.0 中使用 OWC 组件画图</strong></td>
            </tr>
            <tr>
                <td colspan="3" rowspan="2" style="height: 21px">
        <asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
                </td>
            </tr>
            <tr>
            </tr>
        </table>

    </div>
    </form>
</body>
</html>
```

数据库 **SQL** 脚本：

```
USE [web]
GO
/****** 对象：  Table [dbo].[Chart]    脚本日期：03/27/2007 22:26:00 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Chart](
 [id] [int] IDENTITY(1,1) NOT NULL,
 [month] [smallint] NULL,
 [Allcount] [int] NULL
) ON [PRIMARY]
```

在数据库建好表以后要自己手动假想有 12 条数据，手动添加，最终结果类似下图：



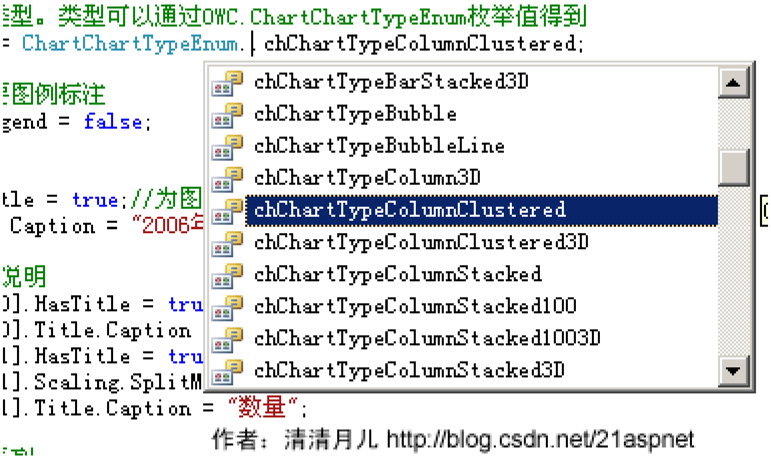
	id	month	Allcount
	1	1	100
	2	2	200
	3	3	250
	4	4	150
	5	5	300
	6	6	400
	7	7	220
	8	8	350
	9	9	200
	10	10	180
▶	11	11	120
	12	12	280
*	NULL	NULL	NULL

作者：清清月儿 http://blog.csdn.net/21aspnet

后台程序说明：

最关键就是 InsertChart.Type = ChartChartTypeEnum.chChartTypeColumnClustered;

你可以在 ChartChartTypeEnum 后点出其他方法。如图所示：



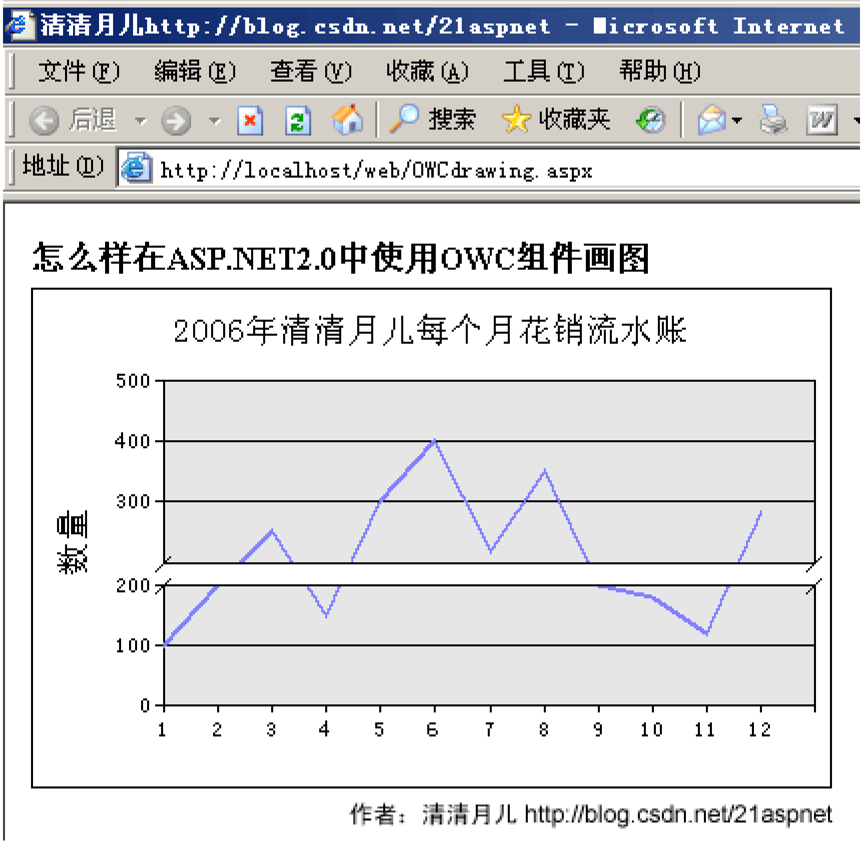
```
型。类型可以通过OWC.ChartChartTypeEnum枚举值得到
= ChartChartTypeEnum.| chChartTypeColumnClustered;

图例标注
gend = false;


tle = true;//为图
 Caption = "2006年

说明
]].HasTitle = tru
]].Title.Caption
l].HasTitle = tru
l].Scaling.SplitM
l].Title.Caption = "数量";
```

chChartTypeBarStacked3D
chChartTypeBubble
chChartTypeBubbleLine
chChartTypeColumn3D
chChartTypeColumnClustered
chChartTypeColumnClustered3D
chChartTypeColumnStacked
chChartTypeColumnStacked100
chChartTypeColumnStacked1003D
chChartTypeColumnStacked3D

作者：清清月儿 http://blog.csdn.net/21aspnet

下面列出的是其他类型图：

折线图：



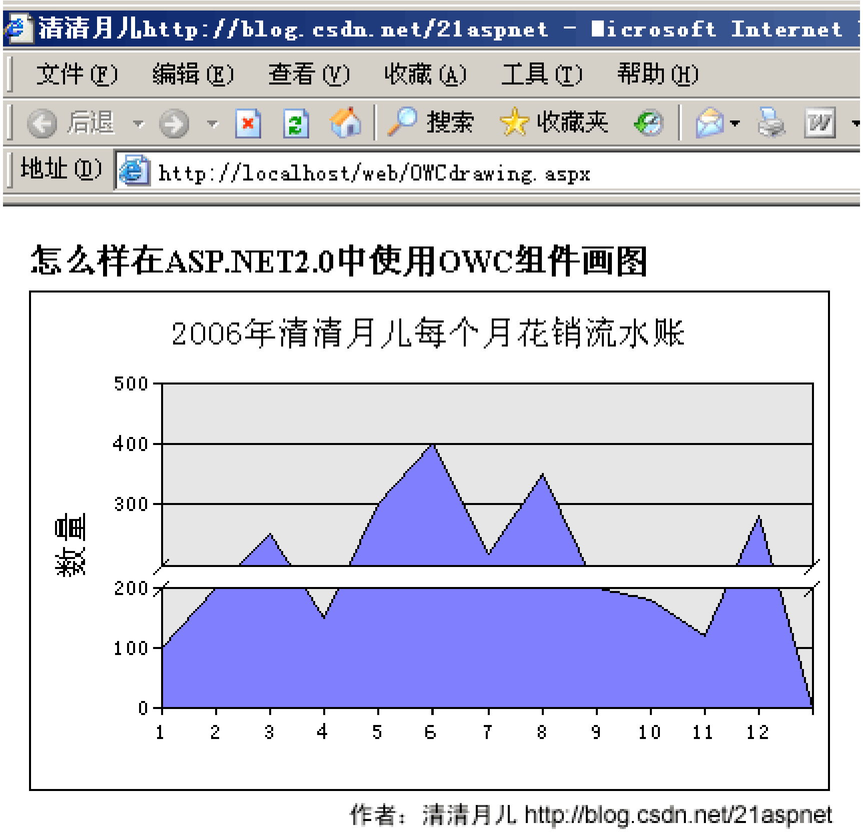
清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F)
编辑(E)
查看(V)
收藏(A)
工具(T)
帮助(H)
后退
搜索
收藏夹
地址(D)
http://localhost/web/OWCdrawing.aspx

怎么样在ASP.NET2.0中使用OWC组件画图

2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12

作者：清清月儿 http://blog.csdn.net/21aspnet

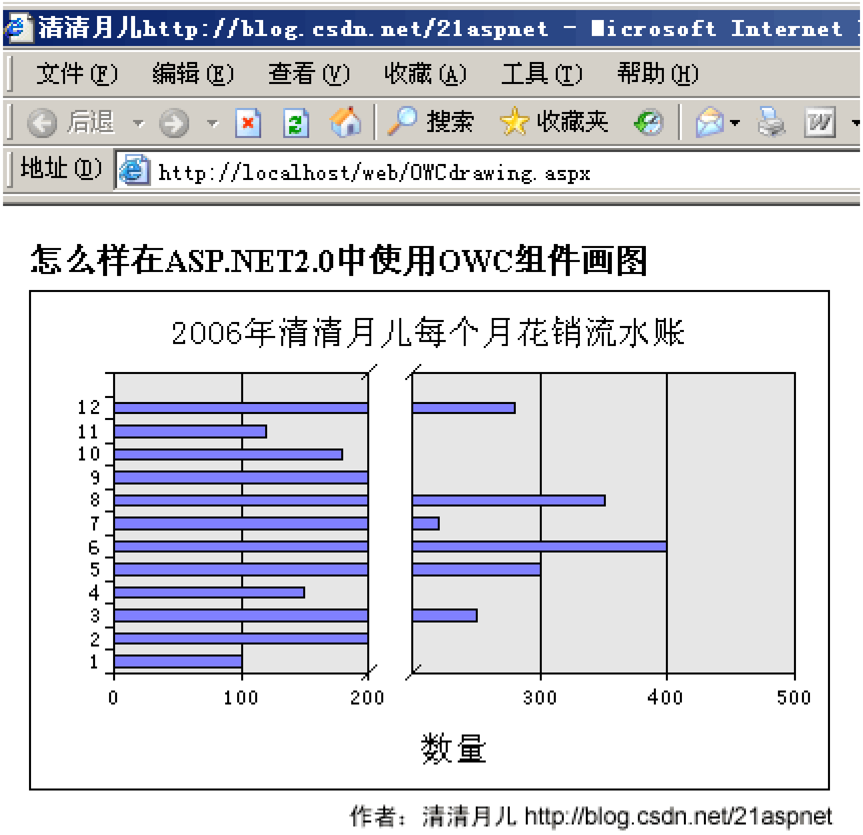
面积图：


清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/web/OWCdrawing.aspx
2006年清清月儿每个月花销流水账
数量
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12

怎么样在ASP.NET2.0中使用OWC组件画图

作者：清清月儿 http://blog.csdn.net/21aspnet

条形图：


清清月儿http://blog.csdn.net/21aspnet - Microsoft Internet
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://localhost/web/OWCdrawing.aspx
2006年清清月儿每个月花销流水账
12
11
10
9
8
7
6
5
4
3
2
1
0
100
200
300
400
500
数量

怎么样在ASP.NET2.0中使用OWC组件画图

作者：清清月儿 http://blog.csdn.net/21aspnet

OWC 什么图形都可以画，还能画立体的，请大家自己尝试。

可以参考 OWC 手册，具体位置：

C:\Program Files\Common Files\Microsoft Shared\Web Components\11\2052\OWCVBA11.CHM

- ## asp.net 中调用 Office 来制作 3D 统计图

1、首先下载 owc11 COM 组件

2、注册 owc11

在工程中添加 C：\Program Files\Common Files\Microsoft Shared\Web Components\11 文件下的 owc11.dll 引用

3、在工程中添加

using OWC11；

4、开始 coding 举例如下：

public class ChartFactory { public ChartFactory（）

{ InitTypeMap（）；/// TODO：在此处添加构造函数逻辑// } protected System.Web.UI.WebControls.Image imgHondaLineup；private string[] chartCategoriesArr；private string[] chartValuesArr；private OWC11.ChartChartTypeEnum chartType = OWC11.ChartChartTypeEnum.chChartTypeColumn3D；//默认值 private static Hashtable chartMap = new Hashtable（）；private static string chartTypeCh = "垂直柱状图"；private static string chartTitle = ""；

private void InitTypeMap（）

{ chartMap.Clear（）；OWC11.ChartChartTypeEnum[] chartTypes = new OWC11.ChartChartTypeEnum[]{ ChartChartTypeEnum.chChartTypeColumnClustered，ChartChartTypeEnum.chChartTypeColumn3D，ChartChartTypeEnum.chChartTypeBarClustered，ChartChartTypeEnum.chChartTypeBar3D，

```
ChartChartTypeEnum.chChartTypeArea，
ChartChartTypeEnum.chChartTypeArea3D，
ChartChartTypeEnum.chChartTypeDoughnut，
ChartChartTypeEnum.chChartTypeLineStacked，
ChartChartTypeEnum.chChartTypeLine3D，
ChartChartTypeEnum.chChartTypeLineMarkers，
ChartChartTypeEnum.chChartTypePie，
ChartChartTypeEnum.chChartTypePie3D，

    ChartChartTypeEnum.chChartTypeRadarSmoothLine，
ChartChartTypeEnum.chChartTypeSmoothLine}
；

    string[] chartTypesCh = new string [] {"垂直柱状统计图"，"3D 垂直柱状统计图"，"水平柱状统计图"，"3D 水平柱状统计图"，"区域统计图"，"3D 区域统计图"，"中空饼图"，"折线统计图"，"3D 折线统计图"，"折线带点统计图"，"饼图"，"3D 饼图"，"网状统计图"，"弧线统计图"}；

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

    { chartMap.Add（chartTypesCh[i]，
chartTypes[i]）；} public ChartSpaceClass
BuildCharts （）

    { string chartCategoriesStr = String.Join
（"\t"， chartCategoriesArr）；string
chartValuesStr = String.Join （"\t"，
chartValuesArr）；

    OWC11.ChartSpaceClass oChartSpace = new
OWC11.ChartSpaceClass （）；

    // ——// Give pie and doughnut charts a
legend on the bottom. For the rest of // them let
the control figure it out on its own. // ——

    chartType = （ChartChartTypeEnum）
chartMap[chartTypeCh]；

    if （chartType ==
ChartChartTypeEnum.chChartTypePie ||
chartType ==
ChartChartTypeEnum.chChartTypePie3D ||
chartType ==
ChartChartTypeEnum.chChartTypeDoughnut）
```

```
    { oChartSpace.HasChartSpaceLegend = true；
oChartSpace.ChartSpaceLegend.Position =
ChartLegendPositionEnum.chLegendPositionBott
om；}

    oChartSpace.Border.Color = "blue"；
oChartSpace.Charts.Add（0）；
oChartSpace.Charts[0].HasTitle = true；
oChartSpace.Charts[0].Type = chartType；
oChartSpace.Charts[0].ChartDepth = 125；
oChartSpace.Charts[0].AspectRatio = 80；
oChartSpace.Charts[0].Title.Caption =
chartTitle；
oChartSpace.Charts[0].Title.Font.Bold = true；

    oChartSpace.Charts[0].SeriesCollection.Add
（0）；
oChartSpace.Charts[0].SeriesCollection[0].Data
LabelsCollection.Add （）；

    // ——// If you're charting a pie or a variation
thereof percentages make a lot // more sense
than values……

    // ——

    if （chartType ==
ChartChartTypeEnum.chChartTypePie ||
chartType ==
ChartChartTypeEnum.chChartTypePie3D ||
chartType ==
ChartChartTypeEnum.chChartTypeDoughnut）

    { oChartSpace.Charts[0].SeriesCollection[0].
DataLabelsCollection[0].HasPercentage = true；
oChartSpace.Charts[0].SeriesCollection[0].Data
LabelsCollection[0].HasValue = false；} // ——//
Not so for other chart types where values have
more meaning than // percentages. // ——else
{ oChartSpace.Charts[0].SeriesCollection[0].Da
taLabelsCollection[0].HasPercentage = false；
oChartSpace.Charts[0].SeriesCollection[0].Data
LabelsCollection[0].HasValue = true；}

    // ——// Plug your own visual bells and
whistles here //
——oChartSpace.Charts[0].SeriesCollection[0].
Caption = String.Empty；
```

```
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Name = "verdana"；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Size = 10；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Bold = true；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Font.Color = "red"；
oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Position = ChartDataLabelPositionEnum.chLabelPositionCenter；

    if （chartType == ChartChartTypeEnum.chChartTypeBarClustered || chartType == ChartChartTypeEnum.chChartTypeBar3D || chartType == ChartChartTypeEnum.chChartTypeColumnClustered || chartType == ChartChartTypeEnum.chChartTypeColumn3D）

    { oChartSpace.Charts[0].SeriesCollection[0].DataLabelsCollection[0].Position = ChartDataLabelPositionEnum.chLabelPositionOutsideEnd；}

    oChartSpace.Charts[0].SeriesCollection[0].SetData（OWC11.ChartDimensionsEnum.chDimCategories，Convert.ToInt32（OWC11.ChartSpecialDataSourcesEnum.chDataLiteral），  chartCategoriesStr）；

    oChartSpace.Charts[0].SeriesCollection[0].SetData（OWC11.ChartDimensionsEnum.chDimValues，Convert.ToInt32（OWC11.ChartSpecialDataSourcesEnum.chDataLiteral），  chartValuesStr）；

    return oChartSpace；}

    #region 属性设置 public string[] chartCategoriesArrValue { get { return chartCategoriesArr；} set { chartCategoriesArr = value；}
```

public string[] chartValuesArrValue { get { return chartValuesArr；

} set { chartValuesArr = value；} public string chartTypeValue { get { return chartTypeCh；} set { chartTypeCh = value； } public string chartTitleValue { get { return chartTitle； } set { chartTitle = value； } #endregion }

//调用 首先需要在页面上放置一个Image来显示产生的统计图

public void ShowChart（）

{

//初始化赋值 chartFactory.chartCategoriesArrValue = chartCategories； chartFactory.chartValuesArrValue = chartValues；chartFactory.chartTitleValue = chartTitle；chartFactory.chartTypeValue = chartType；

OWC11.ChartSpaceClass oChartSpace = chartFactory.BuildCharts（）；string path = Server.MapPath（"."） + @"\images\Chart.jpeg"；//产生图片并保存 页可以是png gif图片 oChartSpace.ExportPicture（path，"jpeg"， 745，500）；Image1.ImageUrl = path； // 显示统计图}

// 保存统计图请参照上一篇文章

//由于每次生成的统计图都会覆盖原来的图片所以有必要的话可以用日期加时间的方式来作为图片的名字，但是这样将会产生很多图片需及时处理，如不需要只需取同名覆盖原来图片即可。

OWC使用方法

OWC==> Office Web Components

是微软 Office 的轻量级开发工具，随 Office 安装时一起安装

Office2003 版本对应 OWC11，里面有 5 个控件对象

图表工作区对象

数据源控件对象

记录集控件对象

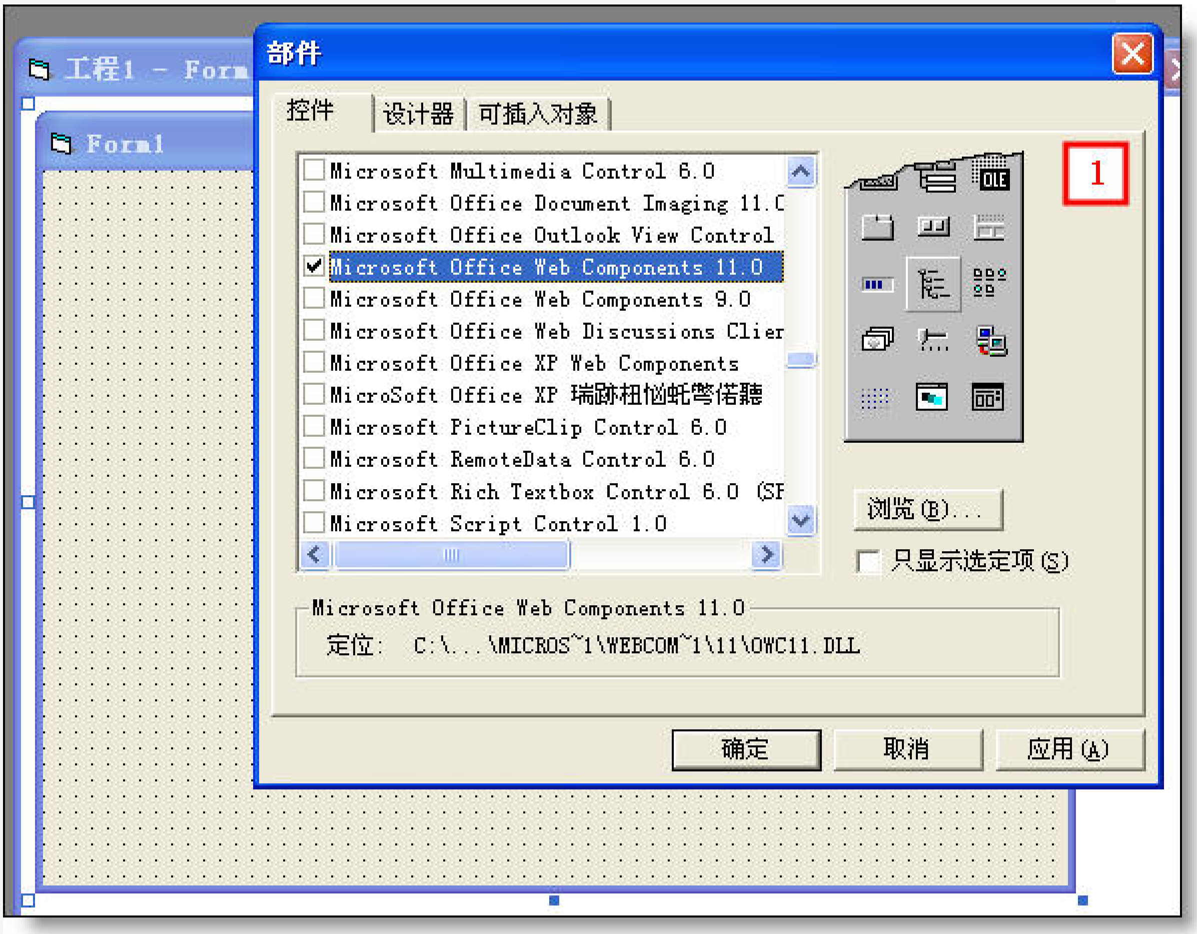
数据透视表列表对象

电子表格对象

很久以前用过的，最近有人问就翻出来看看

其实很简单，只是可能没有到注意文档罢了

我简单用 VB 做了下常用的 SpreadSheet 例子



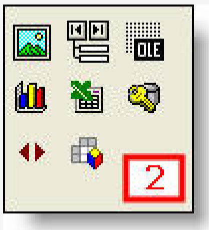
- □ Microsoft Multimedia Control 6.0
- □ Microsoft Office Document Imaging 11.C
- □ Microsoft Office Outlook View Control
- ☑ Microsoft Office Web Components 11.0
- □ Microsoft Office Web Components 9.0
- □ Microsoft Office Web Discussions Clier
- □ Microsoft Office XP Web Components
- □ MicroSoft Office XP 瑞跡租恼蚝弩偌聽
- □ Microsoft PictureClip Control 6.0
- □ Microsoft RemoteData Control 6.0
- □ Microsoft Rich Textbox Control 6.0 (SF
- □ Microsoft Script Control 1.0

浏览(B)...

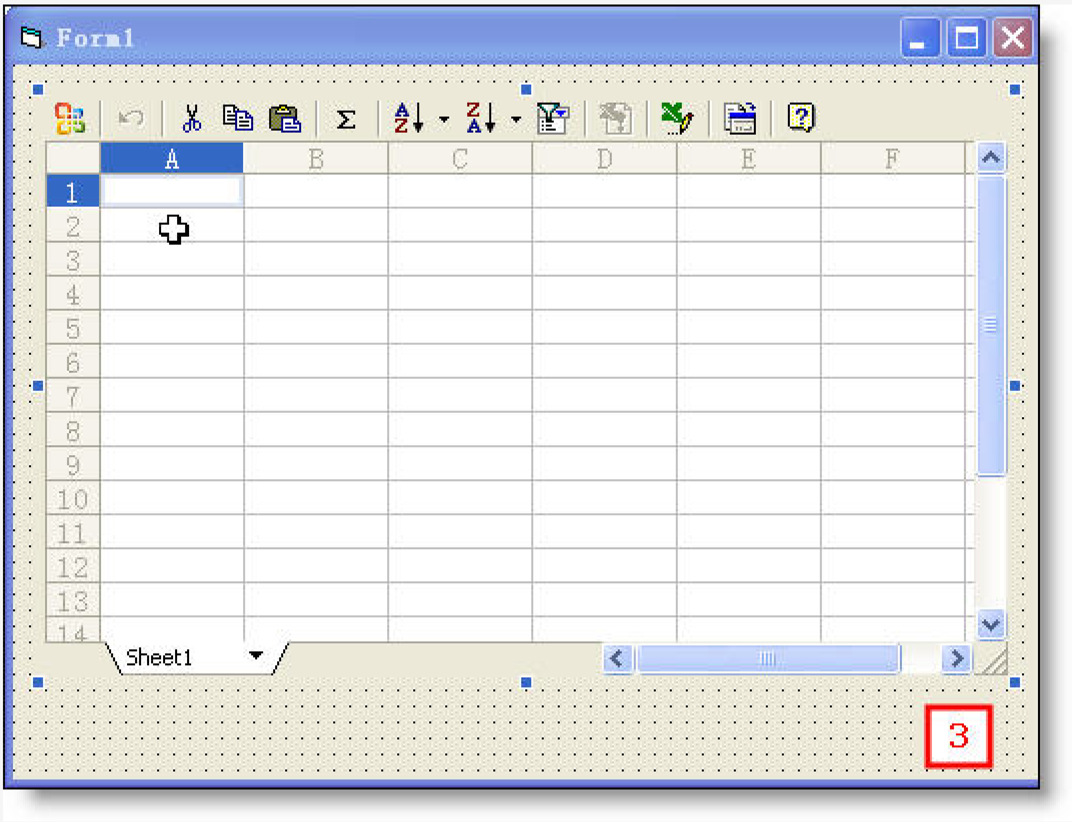
□ 只显示选定项(S)

Microsoft Office Web Components 11.0

定位： C:\...\MICROS~1\WEBCOM~1\11\OWC11.DLL



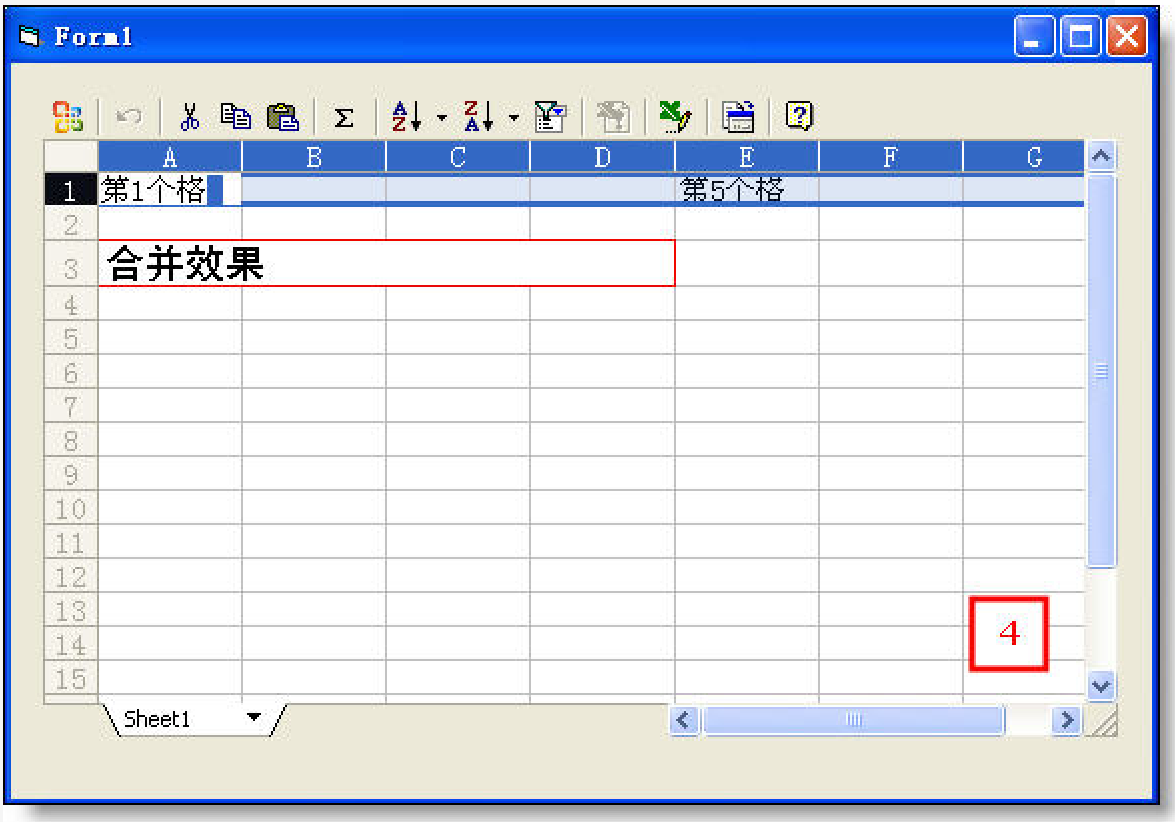
Form 上加一个 Spreadsheet 控件



代码，很简单，没什么好说的

```
Private Sub Form_Load()


    '单元格
    '可以使用循环的方式填充单元格
    Spreadsheet1.Cells(1, 1) = "第 1 个格"
    Spreadsheet1.Cells(1, 5) = "第 5 个格"
    '区域
    '合并
    Spreadsheet1.Range("A3:D3").Merge
    Spreadsheet1.Range("A3:D3").Value = "合并效果"
    '格式设置
    Spreadsheet1.Range("A3:D3").Font.Name = "黑体"
    Spreadsheet1.Range("A3:D3").Font.Size = 15


    Spreadsheet1.Range("A3:D3").Borders(xlEdgeTop).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeBottom).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeLeft).Color = vbRed
    Spreadsheet1.Range("A3:D3").Borders(xlEdgeRight).Color = vbRed


"    '不提示直接保存为本地文件
"    Spreadsheet1.Export "c:\xxx.xls", ssExportActionNone
"    '直接在 Excel 中打开
"    Spreadsheet1.Export "c:\xxx.xls", ssExportActionOpenInExcel
End Sub
```

运行出来就是这个样子



因为整个编程中使用了 VBA 语法，所以还是很容易理解的

如果你的机器上安装了 Office2000/xp/2003

那么你的机器上已经有使用文档及开发文档了，很详尽，参照一下，应该没啥问题。

我的机器装的是 2003，在如下位置上有相关文档

```
C:\Program Files\Common Files\Microsoft Shared\Web Components\10\2052
```

C:\Program Files\Common Files\Microsoft Shared\Web Components\11\2052

前面的文档都是使用帮助

OWCVBA11.CHM 是开发文档

文本的小例子在此处下载 ，VB的

http://www.cnblogs.com/Files/babyt/Owc1.rar

我想用 spreadsheet 做个表格,不知道是不是可以用这个控件直接导入 xml 文件呢,因为它有一个 export 方法,可以导出为 xml 文件.我查了它的所有方法发现都没有发现导入这个功能,但是它里面却又可以导入的,啊泰知道如何在代码中实现导入 xml 文件吗?谢谢啊

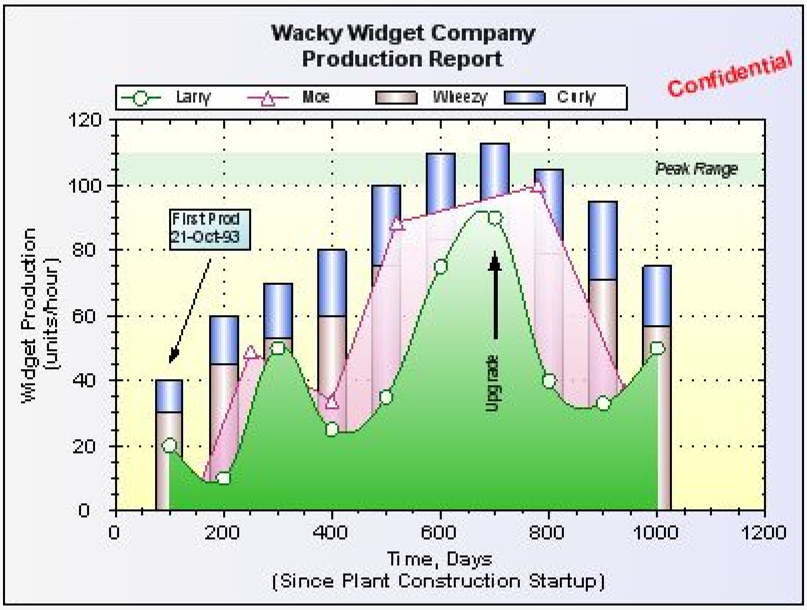
1：首先保证你用的是 OWC11 版本
2：注意你的 XML 文件的格式，不是所有的格式都能被支持的。严格说就是只支持指定格式，你可以把你的 Excel 文件另存为 xml 文件，然后看下格式规定
3：仍以 vb6 的使用为例，你导入 spreadsheet 控件后，使用如下代码即可导入 xml 文件（注意格式）

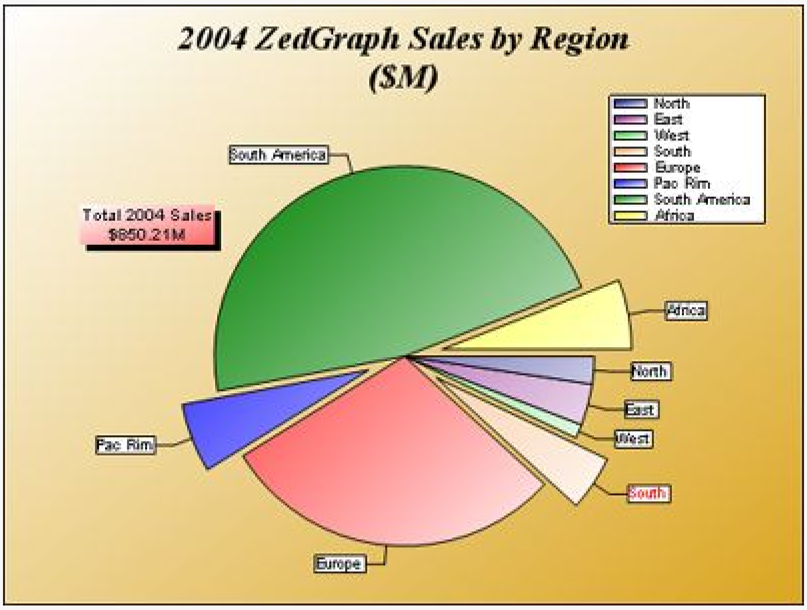
```
Spreadsheet1.XMLURL = App.Path & "\test.xml"
```

以下文章可能对你进一步的工作有所帮助 ^_^
http://www.microsoft.com/china/msdn/library/office/office/xl2k3xmllist.mspx?mfr=true

ZedGraph 组件使用

- ZedGraph 是 C#编写的.NET 类库，提供了用户控件和 web 控件。它可以创建 2D 的线性图、条形图和饼图。它功能完整且有详细的功能自定义，不过使用默认的选项就足够好用了。





在 vs 中使用 ZedGraph 控件的一些记录

ZedGraph 是一个非常优秀的开源的作图控件

ZedGraph 来源：http://sourceforge.net/project/showfiles.php?group_id=114675

ZedGraph 相关例子资源：http://zedgraph.org/wiki/index.php?title=Sample_Graphs

1、 在 vs 中使用 ZedGraph

2、 基本一些概念

几个注意点：

图片的保存路径设置：RenderedImagePath 属性中设置，程序对该文件夹应该是有写和修改权限的

图片的输出格式：OutputFormat 属性中设置，Png 的推荐，比较清晰。

Chart

ChartBorder	图表区域的边框设置
ChartFill	图表区域的背景填充
Legend	图表的注释标签显示设置项目，一组数据对应一种颜色的注释
IsHStack	当有多个显示项的时候设置 Y 轴数据是叠加的还是分开的
Xaxis	图表区域的 X 轴相关信息设置
AxisColor	坐标轴颜色
Cross	坐标的原点，可以设置坐标的偏移程度
CrossAuto	原点自动设置：True 的话 Cross 的设置就无效了。
FontSpec	X 轴标题字体相关信息
Angle	X 轴标题字体显示时候的角度，0 为水平 90 为垂直
Fill	X 轴标题字体填充信息
ColorOpacity	透明度
IsScaled	设置 X 轴标题字体显示大小是否根据图的比例放大缩小
RangeMax	填充时候的最大倾斜度（有过渡色，没试过）

RangeMin	填充时候的最小倾斜度（有过渡色，没试过）
StringAlignment	X 轴标题字体排列（不清楚，没试过）
IsOmitMag	是否显示指数幂（10 次方，没试过，似乎与 IsUseTenPower 有关系）
IsPreventLabelOverlap	坐标值显示是否允许重叠，如果 False 的话，控件会根据坐标值长度自动消除部分坐标值的显示状态
IsShowTitle	X 轴标题是否显示
IsTicsBetweenLabels	两个坐标值之间是否自动显示分隔标志
IsUseTenPower	是否使用 10 次幂指数
IsVisible	是否显示 X 轴
IsZeroLine	当数据为 0 时候是否显示（在饼状图显示的时候有用）
MajorGrid	大跨度的 X 轴表格虚线线显示信息
DashOff	虚线中孔间距
DashOn	虚线单位长度
MajorTic	大跨度的 X 轴刻度信息
IsInside	在 Chart 内部是否显示
IsOutSide	在 Chart 外部是否显示
IsOpposite	在对面的轴上是否显示
MinorGrid	小跨度的 X 轴表格虚线显示信息
MinorTic	小跨度的 x 轴刻度信息
MinSpace	刻度和轴之间的距离（没试过）
Scale	刻度值的一些设定
IsReverse	X 轴的刻度值从高到低还是从低到高

MajorStep	大刻度步长
MajorStepAuto	是否自动设置大刻度步长
MajorUnit	大刻度步长单位
Max	刻度最大值
MaxAuto	根据输入数据自动设置刻度最大值
Min	刻度最小值
MinAuto	根据输入数据自动设置刻度最小值
MinGrace	不清楚，没试过
MinorStep	小刻度步长
MinorStepAuto	是否自动设置小刻度步长
MinorUnit	小刻度单位
Type	数据显示方式
Liner	直接现实（自动）
Date	按日期方式显示
Log	按指数幂方式显示
Ordinal	顺序显示
Y2Axis	第二个 Y 轴坐标信息显示（具体设置看 X 轴）
Yaxis	第一个 Y 轴坐标信息显示（具体设置看 X 轴）
GraphPane	
BarBase	在生成柱状图的时候设置柱状是基于 X 轴还是其他轴
BarType	柱状的类型叠加或其他。
IsFontsScaled	图比例变化时候图表上的文字是否跟着自动缩放
IsIgnoreInitial	是否忽略初始值
IsIgnoreMissing	是否忽略缺省值
IsPenWidthScaled	图比例变化时候图表上的画笔的粗细是否跟着自动缩放

IsShowTitle	图表标题是否显示
PaneFill	Pane的一些填充信息
MasterPane	
BaseDimension	缩放比例基数（可以试试效果）
数据	未测试过。不知道如何和数据库绑定
外观	
IsImageMap	不清楚干吗用的
行为	
AxisChaneged	是否允许自动绘图（没试过，一般都true，动态绘图）
CacheDuration	Cache保存时间0
OutputFormat	输出格式
RenderedImagePath	输出路径
RenderMode	输出模式，不太清楚一般都是ImageTag，另一个输出的是乱码不是图片,

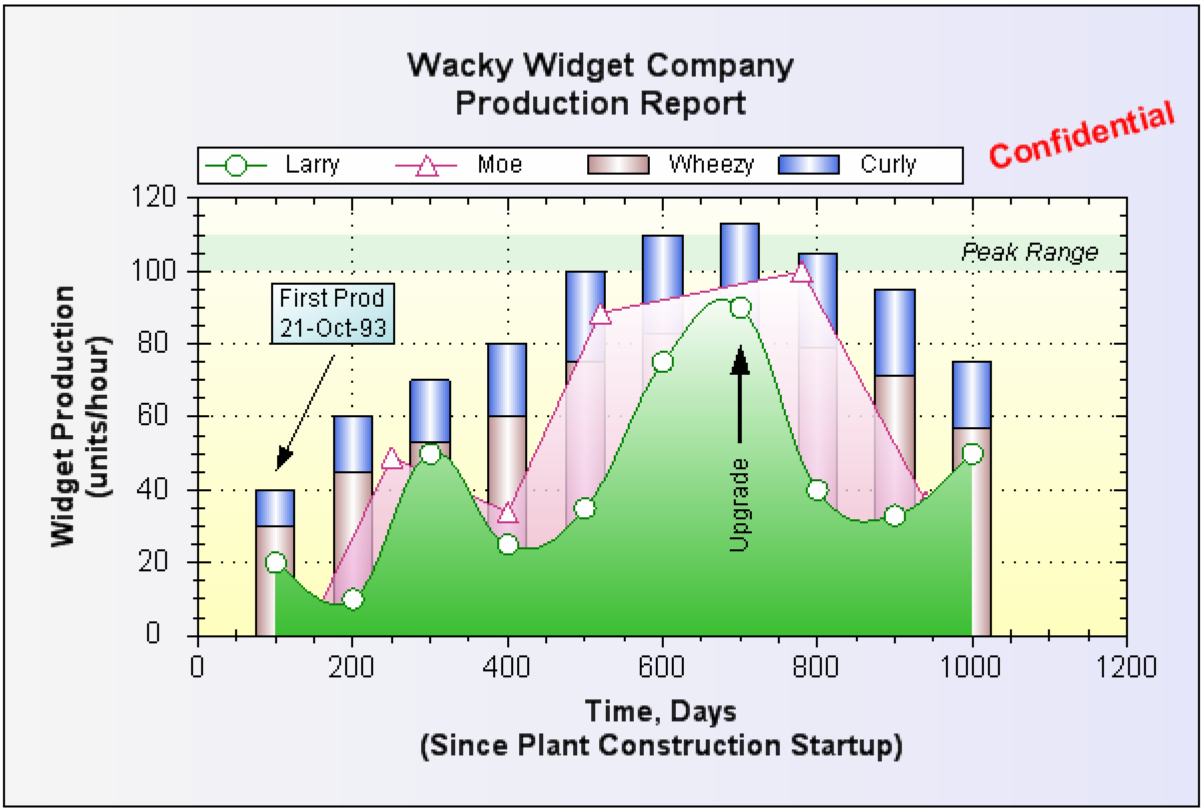
对于图表而言，一般是三种表现形式：柱状图、饼状图和点线图。

.NET控件ZedGraph使用帮助

原文地址：

http://www.codeproject.com/csharp/ZedGraph.asp

译文：



序言

ZedGraph是用于创建任意数据的二维线型、条型、饼型图表的一个类库，也可以作为Windows窗体用户控件和ASP网页控件（这里有个web-accessible 不知道该怎么翻译）。这个类库具有高度的适应性，几乎所有式样的图表都能够被创建。这个类库的用法在于通过提供所有图表属性的省缺值来保持使用性的简单。这个类库包含了基于要绘制的数值范围内的可选择适当度量范围和跨度的代码。

ZedGraph保持作为建立在SourceForge的开源项目。基于这个立场（又找不到合适的词翻译了!—_—），你可以获得项目信息、文件、更新支持和所有发布版本。

一套图表实例连同所有的源代码对于SourceForge也是同样有用的。

背景

世面上有许多的图表组件，但是没有一个符合我的要求。我发现MSChart太诡异，而切许多配置选项没有我所需要的适应性用来完成一个漂亮的视觉效果。当然大多数的商业包能够作到这些，但是我需要一些免费的东西，所以ZedGragh诞生了。

这些类将在一个窗体上产生一个线条，条型，或者图表，并给予一个矩形和一些数据点。ZedGraph 可以构建二维的线性/离散的图形，横/纵坐标轴，花哨的条形图，精致的百分比条形图，错误图和饼型图——他不能构建三维的外观或图表。最近还加入了对于ASP.NET页面中图形处理方法。可以参看下载资料中的关于ASP的文档。所绘制的图表能够配上轴标签和标题，一个图例，文本标签和箭头，图片等等。演示项目中的标记包括能够通过自由拖放自由改变大小的Normski's GDIDB double-buffering class。（这里翻译不清楚，大概意思就是图表能自由调整大小吧）。尝试根据可见窗体来达到尺寸要求。

文档(ZedGraph.chm) 包括了完整的类组件源代码。涉及更多的详细资料——ZedGraph有很多在说明文档中没有说明的配置选项。这些内容在网上可以找到。

使用代码

在简单的图形中，一个图表是通过一些简单的步骤创建的。你可以通过一个类库，一个窗体控件，或者一个ASP.NET网页控件，这些方式中的任何一种来使用ZedGraph。本文将主要介绍类库的使用。当然，你可以使用GraphPane 或者 MasterPane等工具来完成与ZedGragh相同的功能。

作为Web控件使用ZedGraph

ZedGraph现在具有一个可以应用到ASPX的Web控件类。提供下载演示项目示范了这个功能。要使用Web控件，你的页面必须包含以下图片：
<img src="graph1.aspx" />

在这个例子中，graph1.aspx是一个声名了这个控件的文件，这个声明要包含一个叫做“graph1.aspx.cs ”的后台代码文件，这个文件实际上负责绘制图形。所以，ZedGraph.dll文件必须未于和graph1.aspx同级的"bin"目录下。

作为用户控件使用ZedGraph

可以在Visual Studio .NET的工具箱中添加ZedGraph控件。首先，打开Visual Studio .NET，新建一个Windows项目，打开窗体设计器显示当前窗口。要查看工具箱，使用主菜单的视图-工具箱命令。
右键单击工具箱上的“我的用户控件”或“组件”栏，然后选择“添加/删除项”选项。点击“浏览”，选取“*ZedGraph.dll* ”文件。一旦文件添加了，你可以看到一个ZedGraphControl选项在工具栏中。将它拖到窗体设计器中，拉伸到合适大小。这样就在你的窗体中创建了一个ZedGraph控件。这个控件具备了ZedGraph控件的所有功能。一个ZedGraph控件就这样简单的创建了，他带有一个初始的图形窗格（又想不到词...）。ZedGraph.dll文件可以作为用户控件或组件。一些示例程序示范了这个用户控件不同编程语言（Visual Basic, Visual C#, and Visual C++）的用法。本文只关注组件的用法。

作为组件使用ZedGraph

在你的项目中添加组件，步骤如下:

1.在项目中，选择项目菜单下的“添加”选项。通过浏览按扭找到ZedGraph.dll，点击OK。这将使你的项目包含了ZedGraph的所有功能。

2.在主窗体代码中添加使用ZedGraph的代码.

3.用如下的声明在窗体类定义代码中添加窗格(这个词总不好翻译,意思就是说ZedGraph画出来的那个表图形):

```
GraphPane myPane;
```

4.在你的窗体 Load 方法(如:Form1_Load())中添加下列代码:

```
// 在坐标(40,40)处创建一个新图形, 大小为 600x400
    myPane = new GraphPane( new Rectangle( 40, 40, 600, 400 ),
        "My Test Graph\n(For CodeProject Sample)",
        "My X Axis",
        "My Y Axis" );


    // 设置初始数据
    double x, y1, y2;
    PointPairList list1 = new PointPairList();
    PointPairList list2 = new PointPairList();
    for ( int i=0; i<36; i++ )
```

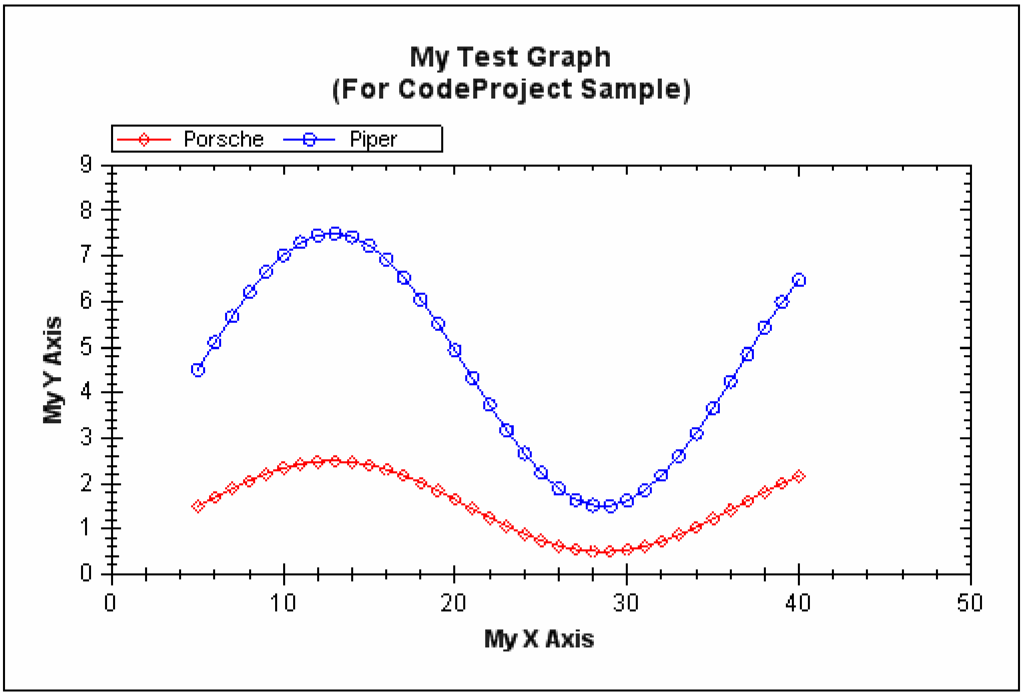
```
   {
      x = (double) i + 5;
      y1 = 1.5 + Math.Sin( (double) i * 0.2 );
      y2 = 3.0 * ( 1.5 + Math.Sin( (double) i * 0.2 ) );
      list1.Add( x, y1 );
      list2.Add( x, y2 );
    }
   // 创建红色的菱形曲线
   // 标记, 图中的 "Porsche"
   LineItem myCurve = myPane.AddCurve( "Porsche",list1, Color.Red, SymbolType.Diamond );
   // 创建蓝色的圆形曲线
   // 标记, 图中的 "Piper"
  LineItem myCurve2 = myPane.AddCurv( "Piper",list2, Color.Blue, SymbolType.Circle );
   // 在数据变化时绘制图形
    myPane.AxisChange( this.CreateGraphics() );
```

AxisChange() 方法在你的增加或者改变数据时被调用,它通知 ZedGraph 重新计算所有轴的范围.(AxisChange() 方法可以添加自己的代码,他将根据当前配置更新轴的范围),ZedGraph 提供了一个参数较少的 AxisChange() 方法,因此你也可以不调用 CreateGraphics()方法.

5.为了确保图形被绘制,你可以添加一行代码到你的 Form_Paint() 方法(Paint 事件调用的方法):

```
 myPane.Draw( e.Graphics );
```

以上的代码产生的输出如下:



好了，文章翻译到这里也差不多了，当然原文还有很多内容，不过那些是高级应用部分了，到这里算了解了 ZedGraph 的用法，后面的内容很容易看懂了，这里就不再翻译了。

posted on 2006-01-11 23:52 sharping 阅读(187) 评论(1) 编辑 收藏 引用 收藏至 365Key 所属分类: .NET .

Feedback

re: .NET控件ZedGraph使用帮助 2006-05-22 00:48 fly2689

用ZedGraph在asp.net中做一个X轴，两个Y轴的图的时候，发现两个Y轴怎么都靠在左边，不是一个左边，一个右边，我按照例子里的设置了

pane.YAxis.IsOppositeTic = false;

pane.YAxis.IsMinorOppositeTic = false;

和

pane.Y2Axis.IsOppositeTic = false;

pane.Y2Axis.IsMinorOppositeTic = false;

发现没用

搞了老半天了也搞出来，唉，郁闷,还请指点！ 回复

gridview结合Zgraphy使用用法 [日期：2007-07-27] [来自：本站原创]

Zgraph图表控件的强大功能令人出乎意料，与OWC相比我想应该毫不逊色，近来需求要求作出相关数据统计，不想使用BI这类的强大东西，所以搜索到了免费的开源的Zgraph控件。使用起来也非常方便，生成的效果可以说是五花八门，千姿百态。废话少说，先看看俺生成的样子.

图 1:

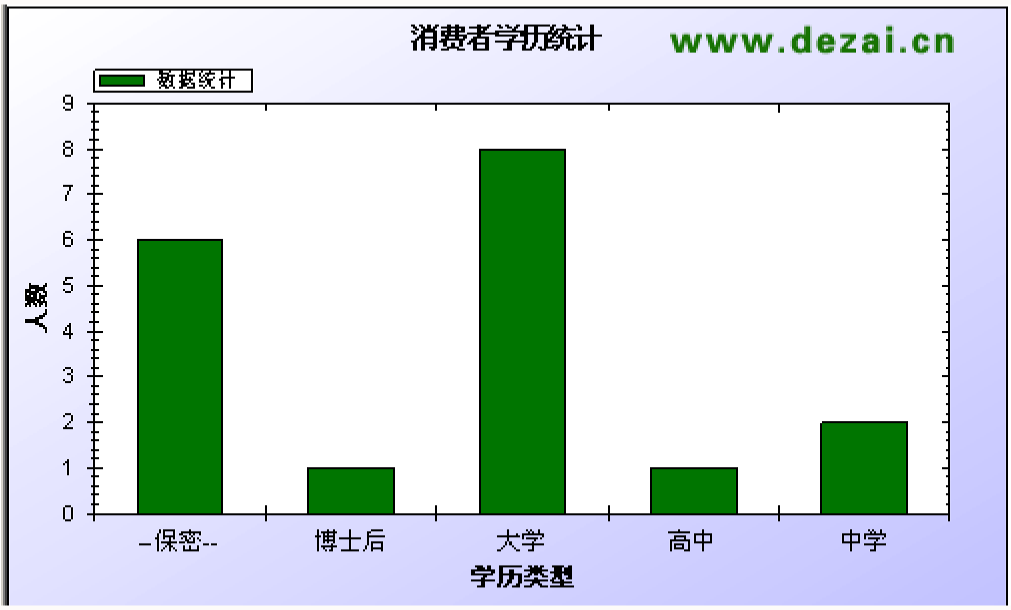


图 2:

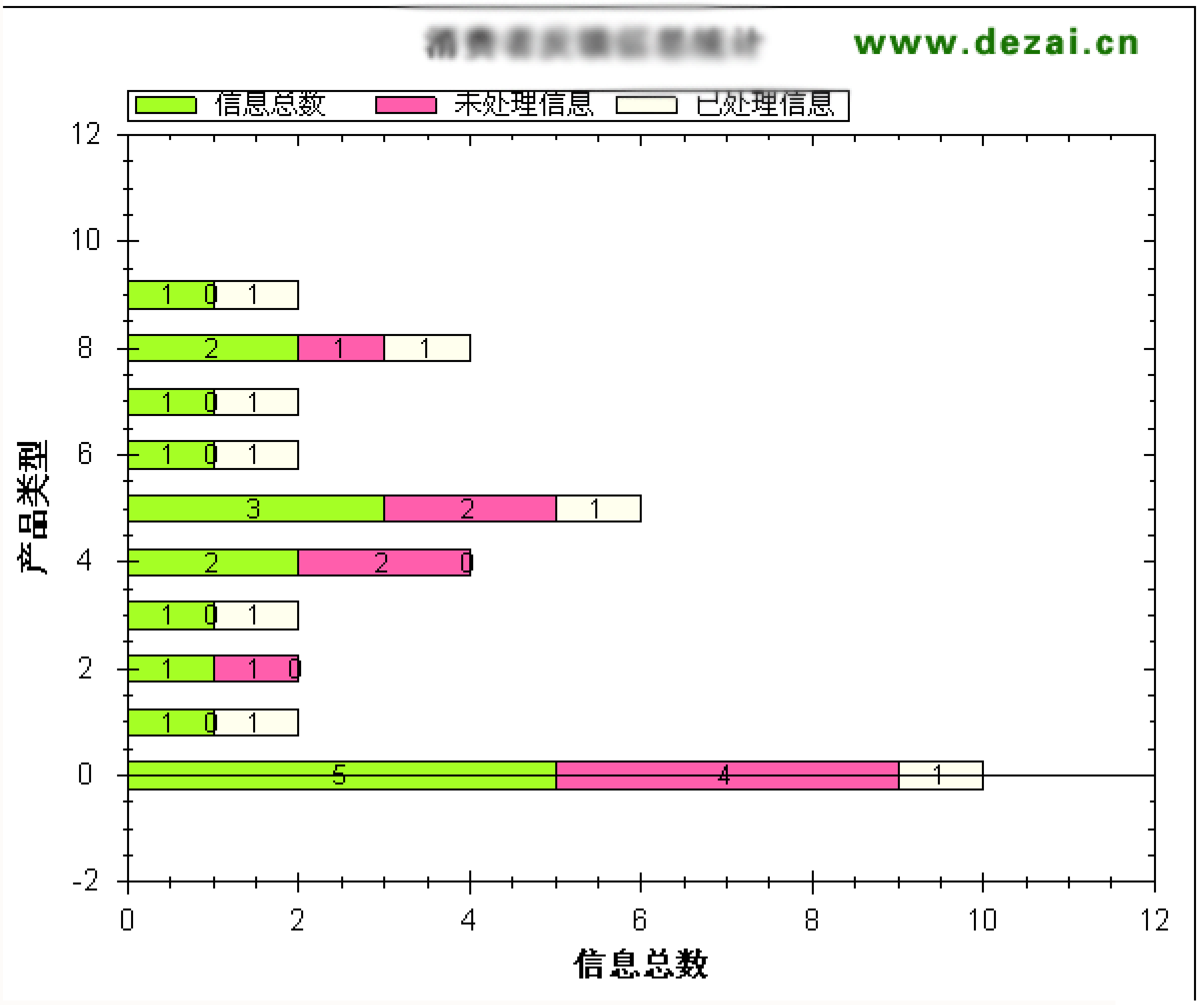
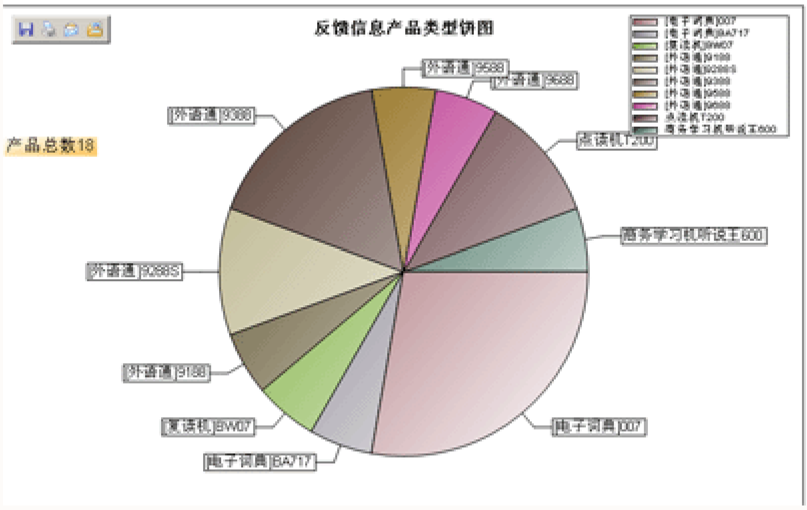
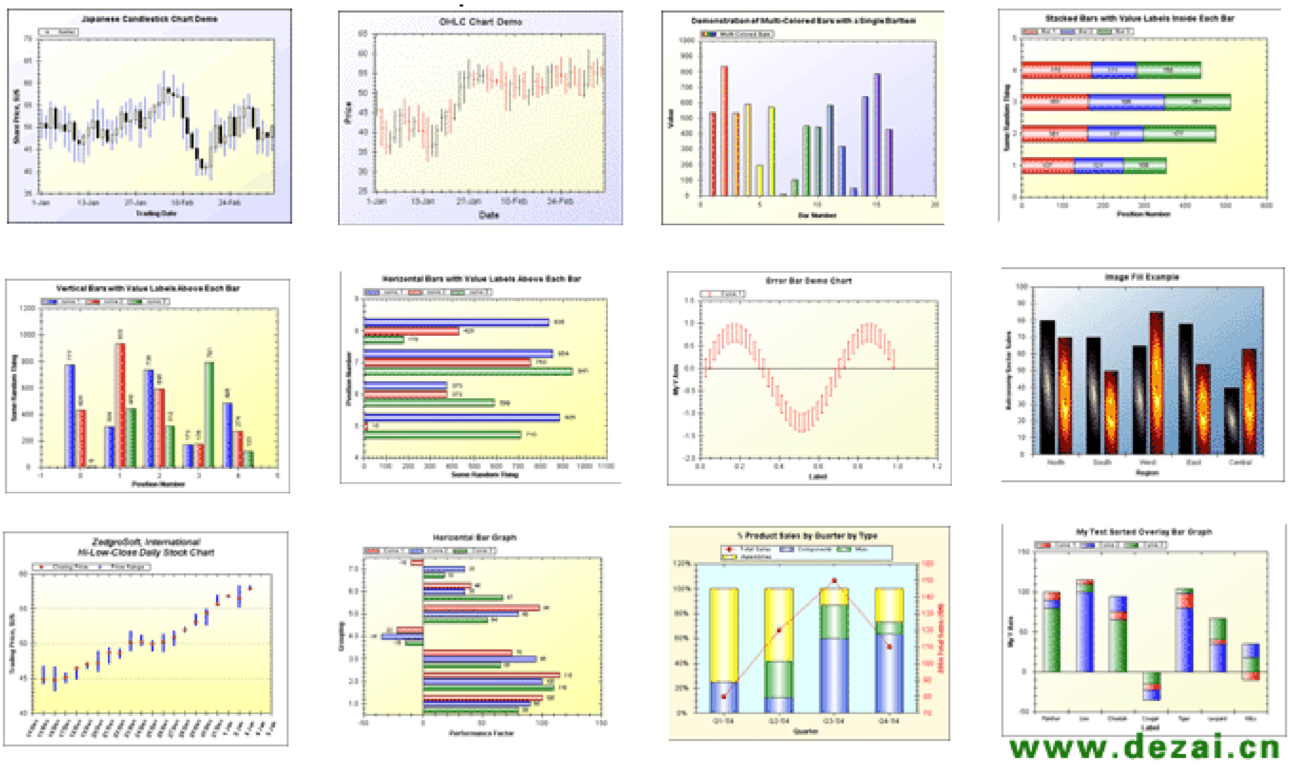

www.dezai.cn
信息总数
未处理信息
已处理信息
产品类型
信息总数
12
10
8
6
4
2
0
-2
0
2
4
6
8
10
12
1 0 1
2 1 1
1 0 1
1 0 1
3 2 1
2 2 0
1 0 1
1 1 0
1 0 1
5 4 1

图 3:


反馈信息产品类型饼图
产品总数18
[外语通]9588
[外语通]9688
[外语通]9388
点读机T200
商务学习机听说王600
[外语通]9288S
[外语通]9188
[复读机]BW07
[电子词典]BA717
[电子词典]007

这些根据俺的需求与相关统计数据，与Gridview结合起来绘制出来的图，而从此控件的官方网站中，我们还可以看到zgraph不仅能在WebForm中实现图表功能，在WinForm中同样可以实现，我在这里就只介绍一下WebForm的使用了。（而对应WinForm的使用，官方的Wiki里有很好的example有VB的啊，有C#的啊，这里就不用啰嗦了)

图 4:



网上在介绍Zedgraph在webForm中使用的大多数是根据Zedgraph自身所带的数据实现，没有与Net的相关数据控件进行结合，我在这里就根据自己在project中所使用的gridview与Zedgraph使用的心得，与大家一起分享一下.同时欢迎大家多多指教，指正错误。

Step1:下载Zgraph控件

Zgraph控件现在已经出到 5.0.10 版本了，分WinForm版和WebForm两种版本，你可以直接到官方下载，也可以在我这下载哈

本站下载地址：http://www.dezai.cn/lesson_info.asp?SoftID=71 (俺可是好人，不会放虫在里面)

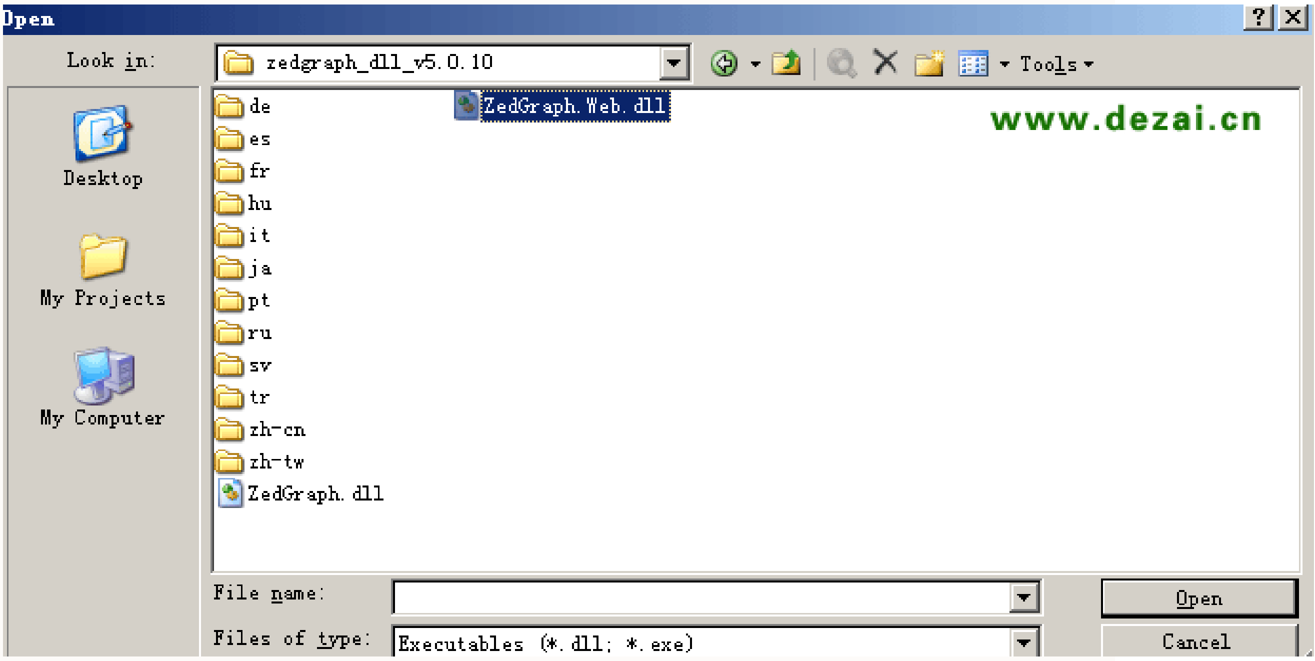
官方下载地址：http://sourceforge.net/project/showfiles.php?group_id=114675

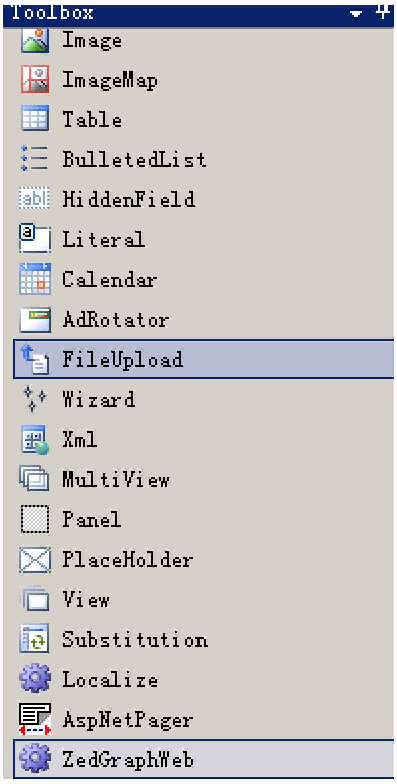
Step2:在Project中引用控件

一般情况下，此控件应该是使用在表现层（UI层），所以你可以直接在你的UI层直接引用，当然，为了你方便使用，你可以先把他加到控件箱里头(ToolBox)

方法： 对着控件箱右键点击->选择Choose Item -> 浏览 ->找到你下载的控件-> 选择 就可以了

这样你就可以在控件箱里看到ZedGraph的图标了，你就可以像其它控件一样使用拖拽了





上面说了这么多废话都是前奏，我们还没开始真正使用。

Step3:设置属性

先把控件拖拉到页面中或用户控件中,ZedGraph的属性很多，我这里只说明一下常用的几个属性哈，其实大多数是默认的就OK的，大多数都是对图表的外表进行相关设置，如果你想做出非常精美的图表，那就要对这些属性下一番功夫了。

ID:控件ID
BarBase: 设计图表的基准轴，默认为X
BarType:图表的类型（饼图，柱状图，曲线图等等)
Title:图表标题
OutPutFormat:输出的图表文件类型(Png,Gif,Jpeg,Ico)
width:图表宽度
height:图表高度
ChartFill: 图表背景直充（俺根据字面理解，可能不太准确)
ChartBoder: 设计图表边框样式
lineType:线条类型...... ....
(老实说，我一般在提供的属性里面只是去整宽度跟高度，其它的就很少动了）

Step4:引用控件

这几个东西，你就不要丢了，记得搞上去

```
using ZedGraph;
using ZedGraph.Web;
using System.Drawing;
using System.Drawing.Imaging;
```

Step5:初始化

不管你在页面中(aspx)直接使用，还是在用户控件中(ascx)，均要对ZedGraph进行初始化操作，其实很简单，你就把下面这段COPY就到你的CS里面就可以了

```
 private void InitializeComponent()
   {
       this.ZedGraphWeb1.RenderGraph += new ZedGraph.Web.ZedGraphWebControlEventHandler(this.
OnRenderGraph);//注册事件
       this.Load += new System.EventHandler(this.Page_Load);

   }
```

注意：这只是放置一张ZedGraph中生成的图片，那假如是放两个呢？两个就要加一个喔

```
  private void InitializeComponent()
   {
        this.ZedGraphWeb1.RenderGraph += new ZedGraph.Web.ZedGraphWebControlEventHandler(this.
OnRenderGraph1);//注册事件
       this.ZedGraphWeb2.RenderGraph += new ZedGraph.Web.ZedGraphWebControlEventHandler(this.
OnRenderGraph2);//注册事件
       this.Load += new System.EventHandler(this.Page_Load);

   }
```

当你COPY了这段代码后，还要记得在page_load()中加载

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


       InitializeComponent();

   }
```

Step6:绑定数据

之前在网上找到的相关教程中 90%是采用官主提供的默认数组数据绑定的，最好的绑定接口是IList，由于我这没必要用我就没去研究哈，有兴趣的兄弟们可去去试一下，ZedGraph绑定的数据源可以是很多，起初我以为只能绑定DataSet，但后面还是可以绑定DataReader的，不仅如此，数组就更不在话下了。我这里只用DataReader，不过话说回来，所呈现的数据都是以数组整的

6.1 数据源绑定，这个就不用多说了

我在前台加了一个GridVidw，其Gridview的数据源是一个dataReader

```
 protected void Show()
   {
       this.gdvEduList.DataSource = FeedBackInfoDataCount.EduDataCount();
       gdvEduList.DataBind();
   }
```

记得在page_load()加载，那么我们的page_load就这样的啦

```
 protected void Page_Load(object sender, EventArgs e)
   {
       Show();

       InitializeComponent();
   }
```

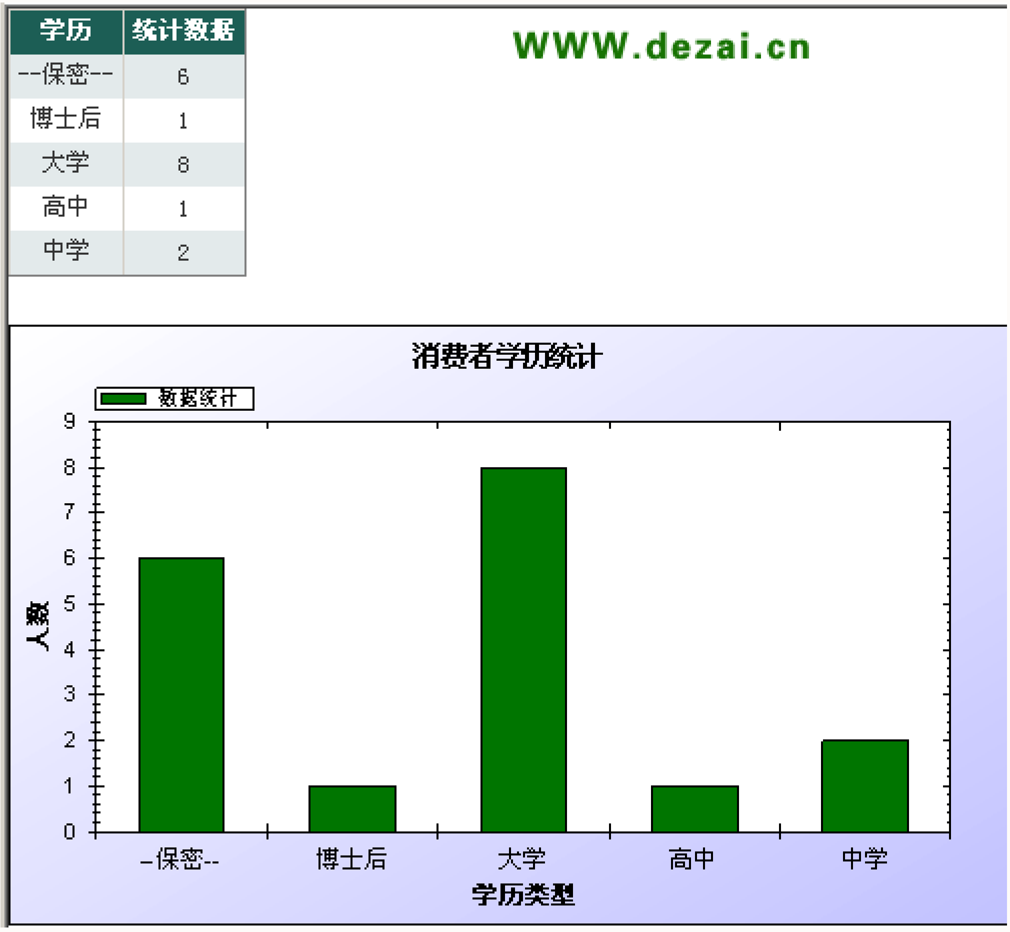
6.2 设定图表数组绑定

这个就是核心代码了

```
 private void OnRenderGraph(ZedGraphWeb zgw, Graphics g, MasterPane masterPane)
   {
       GraphPane myPane = masterPane[0];

       myPane.Title.Text = "消费者学历统计"; //设计图表的标题
       myPane.XAxis.Title.Text = "学历类型"; //X轴标题
       myPane.YAxis.Title.Text = "人数"; //Y轴标题

       PointPairList list = new PointPairList(); //初始化一个PointPariList对象，说白了就是一条曲线或一条
柱子要生出来了

       //PointPairList list2 = new PointPairList(); 如果你要在一个图表城显示多条曲线或柱子你就可以增
加一个PointPariList对像，相对应下面的 y2,List2 也是这个List相关的东西了
```

```
    //PointPairList list3 = new PointPairList(); //原理同上

    for (int x = 0; x < this.gdvEduList.Rows.Count; x++) //循环,x的初始值为 0,最终值是我数据源中的记录总数 这里主要是gdvEdulist.Rows.Count的设定，根据你的数据源来设定
    {
        int careerCount = Convert.ToInt32(this.gdvEduList.Rows[x].Cells[1].Text.Trim()); 这个是获取所对应的每条柱子所对应的数据的文本，我是从gridvew中直接取到的

        int y = careerCount;//这句就是要充的值了，将CareerCount的值都整到Y里面去了，其实你也可以直接将值赋予Y，注意，这里的Y应该代表Y轴

        //double y2 = rand.NextDouble() * 300;
        //double y3 = rand.NextDouble() * 300;

        list.Add(x, y); // 将XY的值整到list里面存储起来

        //list2.Add(x, y2);
        //list3.Add(x, y3);
    }

    BarItem myCurve = myPane.AddBar("数据统计", list, Color.Green); //增加一个Bar,List里面是不是包含了（X，Y)

    myCurve.Bar.Fill = new Fill(Color.Green, Color.Green, Color.Green); //将颜色直译

    //BarItem myCurve2 = myPane.AddBar("续费", list2, Color.Red);
    //myCurve2.Bar.Fill = new Fill(Color.Red, Color.White, Color.Red);
    //BarItem myCurve3 = myPane.AddBar("升级", list3, Color.Green);
    //myCurve3.Bar.Fill = new Fill(Color.Green, Color.White, Color.Green);

    myPane.XAxis.MajorTic.IsBetweenLabels = true; //这个我不知道乍么解释，不好意思

    string[] labels = new string[gdvEduList.Rows.Count];

    for (int i = 0; i < this.gdvEduList.Rows.Count; i++)  //这个循环主要是取到里面的说明文字，用了一个数组的方法
    {
         labels[i] = this.gdvEduList.Rows[i].Cells[0].Text.Trim();
    }

  //以下这些是无关痛庠的属性设置了，一般默认就可以

    myPane.XAxis.Scale.TextLabels = labels;
    myPane.XAxis.Type = AxisType.Text;

    myPane.Fill = new Fill(Color.White, Color.FromArgb(200, 200, 255), 45.0f);

    myPane.Chart.Fill = new Fill(Color.White, Color.White, 45.0f);

    masterPane.AxisChange(g); //这句话不可少
  }
```

当你整完上页这段后，你就可以得到下面这个东西了



学历	统计数据
--保密--	6
博士后	1
大学	8
高中	1
中学	2

我想大家应该大至明白乍样用这东西了吧。

发个源码包： 点击下载此文件

上面这个是一个柱子的生成图，那其它图呢？

我只把主要代码丢出来给大家哈，有错你就要说给我听哈，不要我会贬你的哈

饼图：

点击下载此文件

多条柱子的：

点击下载此文件

WebZedGraphDemo

点击下载此文件

其实WinForm的跟WebForm的用法差不多，大家只要认真比较一下，就很容易从官方的example里面的提供的在WinForm的使用转成WebForm的啦，俺们一起努力吧.

相关IT同仁的学习文章，大家都可以看看哈

ZedGraph Wiki:

http://zedgraph.org/wiki/index.php?title=Main_Page

（codeproject.com） http://www.codeproject.com/csharp/ZedGraph.asp

ZedGraph 总论(WinForm篇)http://blog.csdn.net/tjvictor/archive/2006/11/24/1412550.aspx

ZedGraph在Asp.net中的应用 http://tech.it168.com/n/2007-06-13/200706131521812.shtml

ZedGraph在Asp.net中的应用(2) http://www.cnblogs.com/wxukie/archive/2007/05/16/748922.html

请大家多多指导哈.

声明：本Blog文章均为德仔原创，欢迎转载，转载时请您注明出处，尊重他人成果，人人有责。

ZEDGRAPH给LINECHART添加数值

作为支持.net 的强大的开源图表控件 ZedGraph，最新版本是

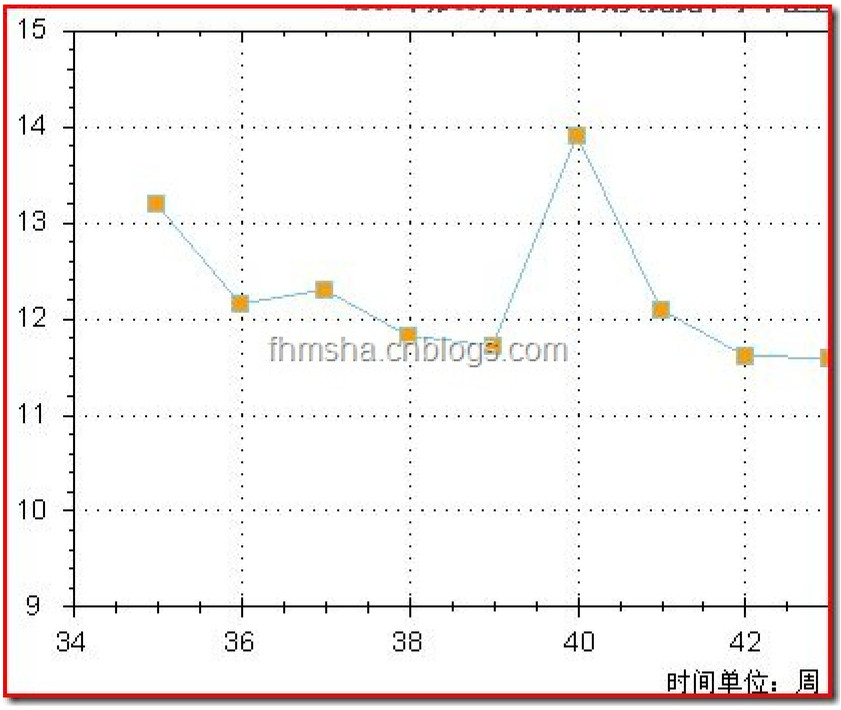
New Update as of 28-Nov-2007 Version 5.1.4 + 4.6.4

官方主页：http://zedgraph.org/

SOURCEFORG主页：http://zedgraph.sourceforge.net/

5.x 的是支持.net2.0 的，4.x 的是支持.net1.1 的，由于个人原因，这里使用的版本为 5.1.1 的，方法大同小异。

由于是开源，很多代码可以从 sample 中找到，但是例子中没有 LineChart 的添加数值的方法。通过参考例子中给 BarChart 添加数值的方法，这里给出一个相对可行的解决方案。



ZedGraph 中所有的元素都是以 GDI+的技术画上去的，实际上我们要做的事情就是建立几个 TextObject 添加到图表中去，在这之前还要找到每个点所在的位置 。

实际做法也很简单，在

masterPane.AxisChange(g);

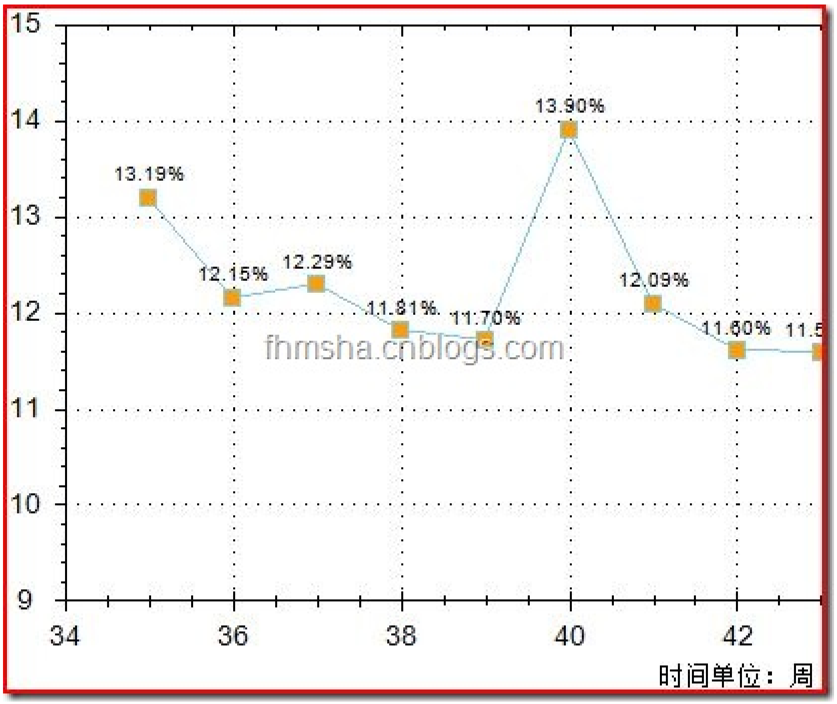
后面加一些代码：

```
int ord = 0;
    foreach (CurveItem mycurve in myPane.CurveList)
    {
        LineItem line = curve as LineItem;

        if (line != null)
        {
            IPointList points = mycurve.Points;

            for (int i = 0; i < points.Count; i++)
            {
                double xVal = points[i].X;
                double yVal = points[i].Y;

                string lab = yVal.ToString("0.00") + "%";

                TextObj text = new TextObj(lab, (float)xVal , (float)yVal*1.01);

                text.Location.CoordinateFrame = CoordType.AxisXYScale;
                text.FontSpec.Size = 10;

                text.Location.AlignH = AlignH.Center ;
                text.Location.AlignV = AlignV.Bottom;
                text.FontSpec.Border.IsVisible = false;

                text.FontSpec.Angle = 0;
                text.FontSpec.Fill.IsVisible = false;

                myPane.GraphObjList.Add(text);
            }
        }
        ord++;
    }
```

注意：

1. 在 ZedGraph 早期的版本中 TextObj 的名称为 TextItem
2. 画图的 masterPane.AxisChange(g);语句在早期版本中为 base.ZedGraphControl.AxisChange();
3. curve.Points 在现在的版本类型为 IPointList ， 早期为 PointPairList
4. 本代码中(float)yVal*1.01 是为是产生一个向上的偏移，由于不知道 Y 轴的具体高度， 如果用强位移的话会因 Y 轴刻度过短而达不到偏移效果。

最后的效果图：


15
14
13
12
11
10
9
34
36
38
40
42
13.19%
12.15%
12.29%
11.81%
11.70%
13.90%
12.09%
11.60%
11.5
fhmsha.cnblogs.com
时间单位：周

转：使用ZedGraph制作动态更新的统计图

Posted on 2007-03-20 15:46 俺喜欢打篮球 阅读(253) 评论(0) 编辑 收藏 所属分类: .NET 、C# .

使用ZedGraph制作动态更新的统计图

ZedGraph是很好的.net下的统计图开源项目，在以前的一篇随笔中提到，与其他的一些统计图控件相比，ZedGraph由于是直接在画布上作画，而不是生成图片显示，所以性能比较好，在诸如股市的实时走势图，显示cpu使用率等实时性较强的应用中有很好的表现，方法并不难，但是由于很少有人写这方面的文章，又正巧在其他论坛中看到有这方面的问题，所以写了下面的例子。ZedGraph 在描画折线图的时候，将所有的坐标点都保存在 PointPairList 中，在画线的时候以这个为 X，Y 坐标。要作动态的折线图，实际上就是不断在这个 PointPairList 中添加点坐标，然后刷新就可以了。

代码很简单：

```
Random ran = new Random();

PointPairList list = new PointPairList();

LineItem myCurve ;
```

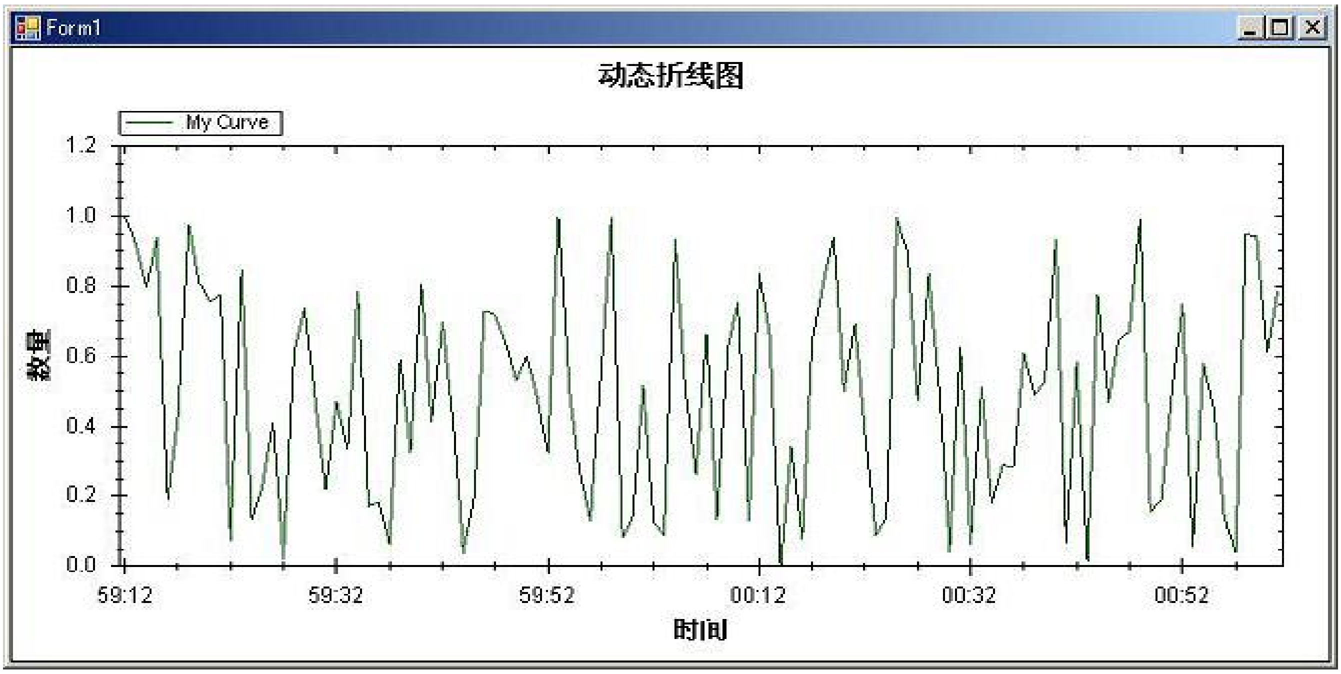
Random 用来生成示例数据，也就是 Y 坐标，PointPairList 用来存放点集合。myCarve 就是要画的线了。当然，不能忘了在窗体上添加 zedGraph 的控件。

为了突出效果，我们在 Form 的 Load 事件中加上下面的代码：

```
this.zedGraphControl1.GraphPane.Title.Text = "动态折线图";

this.zedGraphControl1.GraphPane.XAxis.Title.Text = "时间";

this.zedGraphControl1.GraphPane.YAxis.Title.Text = "数量";

this.zedGraphControl1.GraphPane.XAxis.Type = ZedGraph.AxisType.DateAsOrdinal;


for (int i = 0; i <= 100; i++)

{

    double x = (double)new XDate(DateTime.Now.AddSeconds(-(100 - i)));

    double y = ran.NextDouble();

    list.Add(x, y);

}

DateTime dt = DateTime.Now;
```

```
myCurve = zedGraphControl1.GraphPane.AddCurve("My Curve",
    list, Color.DarkGreen, SymbolType.None);


this.zedGraphControl1.AxisChange();

this.zedGraphControl1.Refresh();
```

这样，窗体加载后就可以看到已经画出了一条折线图。可能象下面的样子：



但是现在，这条线现在还不会动，为了让它动起来就要定时给 PointPairList 中添加坐标。

添加一个 Timer 控件，设置 Interval 属性为 1000，然后在 Timer 的 Tick 事件中添加代码：

```
zedGraphControl1.GraphPane.XAxis.Scale.MaxAuto = true;

double x = (double)new XDate(DateTime.Now);

double y = ran.NextDouble();

list.Add(x, y);

this.zedGraphControl1.AxisChange();

this.zedGraphControl1.Refresh();
```

运行，就会看到线条动起来了。

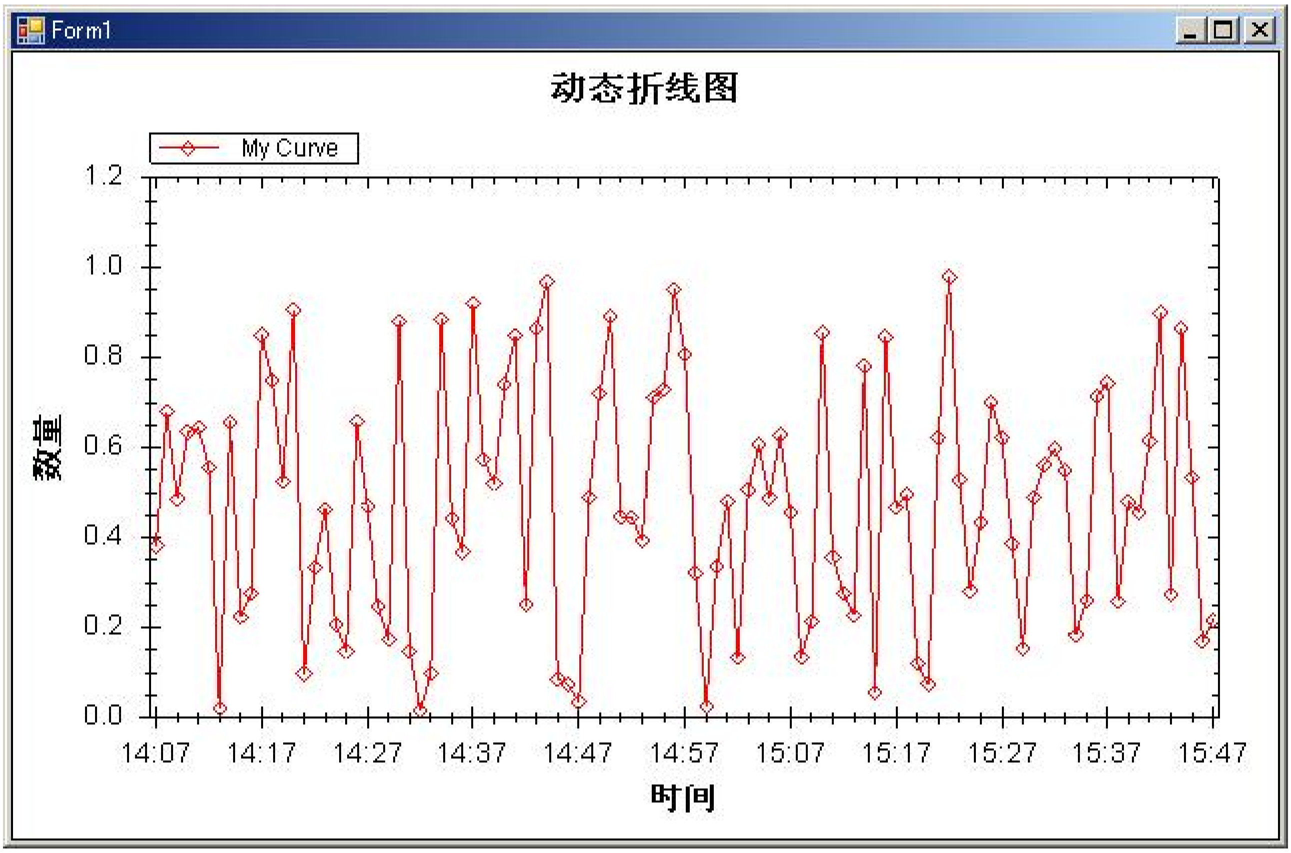
如果要在折线图内显示指定数量的点，只需要在添加坐标之前把第一个坐标点去掉：

```
if (list.Count >= 100)

{

    list.RemoveAt(0);

}
```

如果要象 windows 任务管理器中的 cpu 使用率那样，刚开始的时候是空的，随着时间的推移才逐渐画满，可以在初始化的时候填几个 Y 坐标为 0 的点：

```
for (int i = 0; i <= 100; i++)

{

    double x = (double)new XDate(DateTime.Now.AddSeconds(-(100 - i)));

    double y = 0;

    list.Add(x, y);

}
```

实际上，代码是比较简单的，关键就在于性能，在上面的代码中，在生成折线的时候使用的是 SymbolType.None，如果使用其他几种，折点可以表示为方型，星形等图形，性能就要下降很多，例如，按照上面的代码，在我 windows2000 专业版，赛扬 1.7G，512 内存的条件下，可以显示 10000 个点，而且没有明显的停顿现象，但是如果将折点的图形设置为 SymbolType.Diamond，例如下图这样：



在 10000 个点的情况下停顿现象非常严重，实际上，不到 2000 个点就已经有明显的感觉了。同时在描线的时候没有使用抗锯齿，一样可以提高性能，不过，对性能的提升还是很有限的。

如果以股市实时走势图为例，每天 4 个小时，如果每 10 秒更新一次，6×60×4＝1440，可以看出 ZedGraph 完全可以适用。

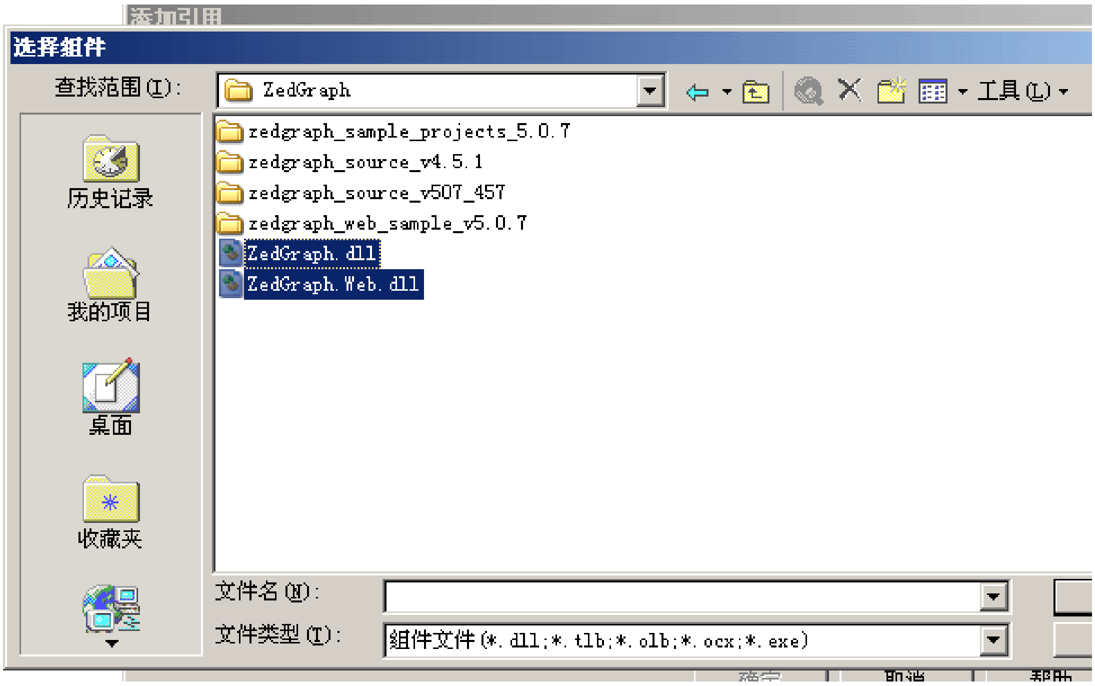
ZedGraph 在 Asp.net 中的应用 2007-07-19 13:05

由于项目的需求图表显示数据,今天在网上找了一天,终于找到一个不错的控件----ZedGraph,它支持 asp,asp.net,vc.

现在最新的版本是 5.0,些版本支持 .NET 2.0.5.0 版本以下的支持.NET 1.1

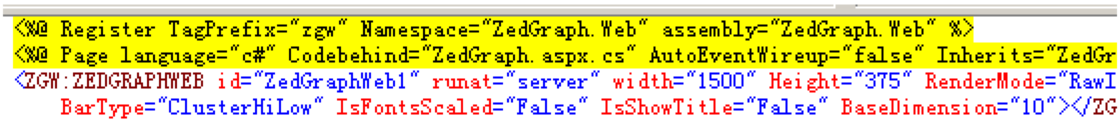
我们现在的项目是.NET1.1 开发的.我在网上找了一天也没有发现一个例子,下面我将介绍下其在 ASP.NET 下的用做 WEB 控件的用法

1.先将它提供的两个 DLL 文件添加引用



2.新建一个 ASPX 页面 ZedGraph.aspx,引用 ZedGraph 用户控件

ZedGraph.aspx 页面代码



```
<%@ Register TagPrefix="zgw" Namespace="ZedGraph.Web" assembly="ZedGraph.Web" %>
<%@ Page language="c#" Codebehind="ZedGraph.aspx.cs" AutoEventWireup="false" Inherits="ZedGr
<ZGW:ZEDGRAPHWEB id="ZedGraphWeb1" runat="server" width="1500" Height="375" RenderMode="RawI
    BarType="ClusterHiLow" IsFontsScaled="False" IsShowTitle="False" BaseDimension="10"></ZG
```

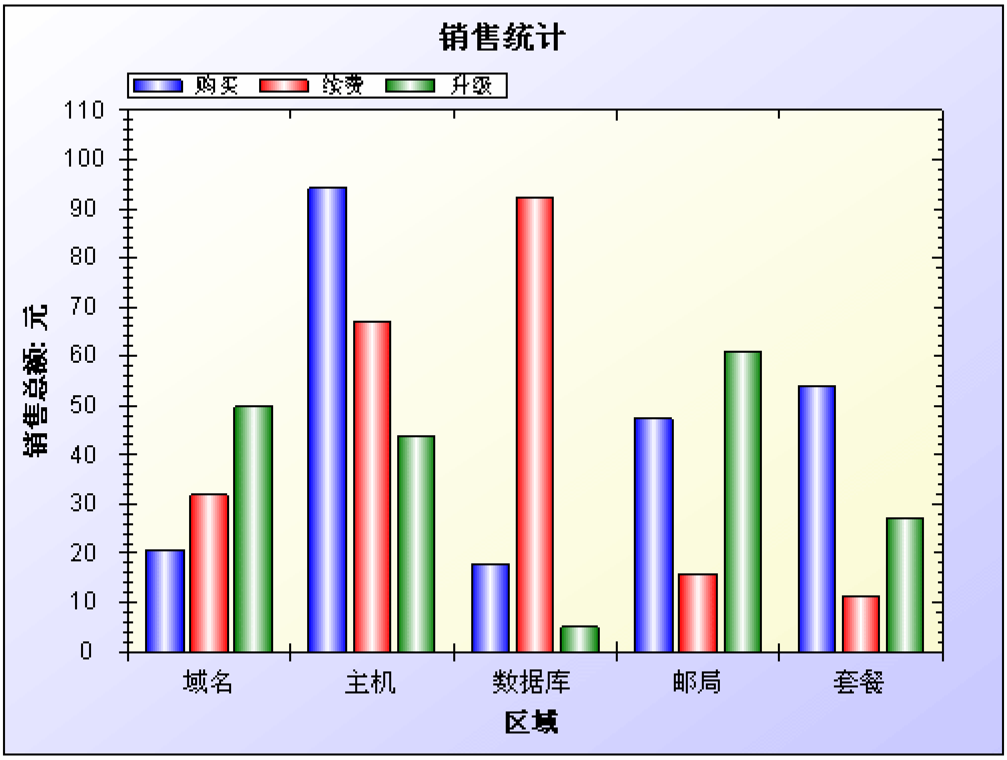
ZedGraph.aspx.cs 代码

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using ZedGraph;
using ZedGraph.Web;

namespace ZedGraph
{
    /// <summary>
    /// ZedGraph 的摘要说明。
    /// </summary>
    public class ZedGraph : System.Web.UI.Page
     {
        protected ZedGraphWeb ZedGraphWeb1;
        private void Page_Load(object sender, System.EventArgs e)
         {
         }

        #region Web 窗体设计器生成的代码
        override protected void OnInit(EventArgs e)
         {
            //
            // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的。
            //
             InitializeComponent();
            base.OnInit(e);
         }

        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容。
        /// </summary>
        private void InitializeComponent()
         {
            this.Load += new System.EventHandler(this.Page_Load);
            this.ZedGraphWeb1.RenderGraph +=new ZedGraphWebControlEventHandler(this.OnRenderGraph);

         }
        #endregion
        private void OnRenderGraph(ZedGraphWeb zgw, Graphics g, MasterPane masterPane)
         {
            // Get the GraphPane so we can work with it
            GraphPane myPane = masterPane[0];

            myPane.Title.Text = "销售统计";
            myPane.XAxis.Title.Text = "区域";
            myPane.YAxis.Title.Text = "销售总额: 元";

            PointPairList list = new PointPairList();
            PointPairList list2 = new PointPairList();
            PointPairList list3 = new PointPairList();
            Random rand = new Random();

            for (double x = 0; x < 5; x += 1.0)
            {
               double y = rand.NextDouble() * 100;
               double y2 = rand.NextDouble() * 100;
               double y3 = rand.NextDouble() * 100;
                list.Add(x, y);
                list2.Add(x, y2);
                list3.Add(x, y3);
            }

            BarItem myCurve = myPane.AddBar("购买", list, Color.Blue);
            myCurve.Bar.Fill = new Fill(Color.Blue, Color.White, Color.Blue);
            BarItem myCurve2 = myPane.AddBar("续费", list2, Color.Red);
            myCurve2.Bar.Fill = new Fill(Color.Red, Color.White, Color.Red);
            BarItem myCurve3 = myPane.AddBar("升级", list3, Color.Green);
            myCurve3.Bar.Fill = new Fill(Color.Green, Color.White, Color.Green);

            myPane.XAxis.MajorTic.IsBetweenLabels = true;
            string[] labels = { "域名", "主机", "数据库", "邮局", "套餐" };
            myPane.XAxis.Scale.TextLabels = labels;
            myPane.XAxis.Type = AxisType.Text;
            myPane.Fill = new Fill(Color.White, Color.FromArgb(200, 200, 255), 45.0f);
            myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45.0f);

            masterPane.AxisChange(g);
        }
    }
}

```

3.新一个测试页面 test.aspx

在这个页面中插入一个图片,其 URL 为 ZedGraph.aspx

4



ZedGraph 在项目中的应用

将数据库数据提取出来，显示成曲线图（饼状、柱状或立体图）是项目中最常见的需求。 网上搜索到的解决方法，大多归为两类，一种是利用 ActiveX 组件，另一种是使用.net 框架自带的画图的类。前者开发比较方便，但浏览时需要用户下载 ActiveX 插件（而这个往往是忌讳的，插件带毒）。后者，需要你自己写绘图、生成图片的类库，不过在开源社区已有不少的项目针对这个问题，做出了实现。ZedGraph 是其中之一，网上的评论大多如是说：使用方便，易上手。

ZedGraph 是 c#编写的类库，提供了用户控件和 web 控件。官网的例子提供了在 windows 窗体上显示图片的源码，而如何将它应用到 web 项目中，却没有任何介绍(或许是我没找到)，web 控件才是项目中有价值的东东。一番搜索后，搂到了一些实现，确实！它已经封装到相当优秀的程度，举个例子：

我们在画曲线图的时候，往往为如何设计 X、Y 轴的坐标而头疼，如何设置它数值范围、设置单元间距等，才能让我们的数据显示得尽量完整和精确。而它的设计是，只要你提供数据，X/Y 轴的问题不用管——它会找出数据范围，然后设置合理的显示域和单元间距。下面是一个简单的实现，将数据库数据生成一张曲线图和柱状图，如此它便可以被这样的方式，显示到页面上：

```
<body>
    <img src="Pic.aspx">
</body>
```

首先，你需要将两个 dll 引用至你的工程，ZedGraph.dll 和 ZedGraph.web.dll。接下来，是你要生成图片的页面，暂且命名为 pic.aspx，将它的 html 标签部分去掉，添加 ZEDGRAPHWEB 控件：

```
<%@ Page language="c#" Codebehind="PriceTimePic.aspx.cs" AutoEventWireup="false"
    Inherits="Herbalife.HelpDesk.UI.ReportManage.PurchaseReport.PriceTimePic" %>
<%@ Register TagPrefix="zgw" Namespace="ZedGraph.Web" Assembly="ZedGraph.Web"%>
<ZGW:ZEDGRAPHWEB id="ZG" runat="server" width="500" Height="375" RenderMode="RawImage" />
```

然后，我们到 pic.aspx.cs 下，添加包引用：using ZedGraph; using ZedGraph.Web;，添加控件对象的声明：protected ZedGraph.Web.ZedGraphWeb ZG;。接下来在

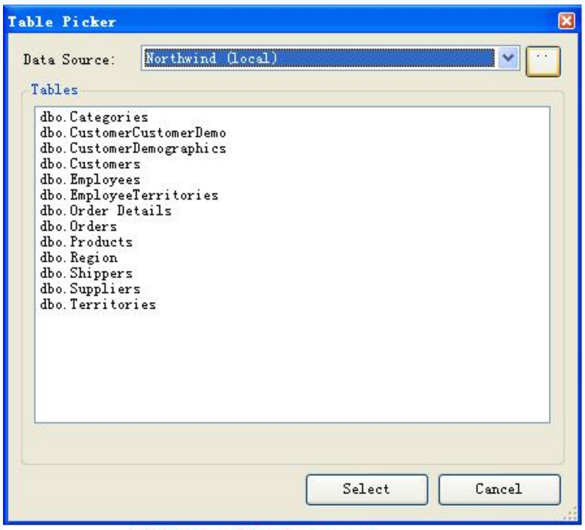
InitializeComponent()方法中注册一个事件：

```
// OnRenderGraph 绘图方法
this.ZG.RenderGraph += new ZedGraph.Web.ZedGraphWebControlEventHandler(this.OnRenderGraph);
```

下面便是绘图方法了：

```
private void OnRenderGraph(System.Drawing.Graphics g, ZedGraph.MasterPane masterPane)
{
    GraphPane myPane = masterPane[0];

    // Title
    myPane.Title.Text = "供应商 价格-时间曲线图";
    myPane.XAxis.Title.Text = "时间";
    myPane.YAxis.Title.Text = "价格";

    // 坐标对集合
    PointPairList line = new PointPairList();

    // 取数据部分
    DataTable dt = GetData();

    // 将数据转为(X,Y)坐标对
    for(int i=0;i<dt.Rows.Count;i++)
    {
        /*
         *   添加坐标对，需要将 X/Y 值全部转为 double 类型。原本以为他们会支持 DateTime 类型。
         *   个人觉得还是很有必要的，毕竟很多图都是时间-**曲线图，有需求。
         */
        string date = Convert.ToDateTime(dt.Rows[i]["RecTime"]).ToString("yyyyMMdd");
        line.Add(Convert.ToDouble(date), Convert.ToDouble(dt.Rows[i]["Price"]));
    }

    // 绘制曲线
    LineItem li = myPane.AddCurve("aaa",line, Color.Blue);
    masterPane.AxisChange(g);
}
```

效果图如下：

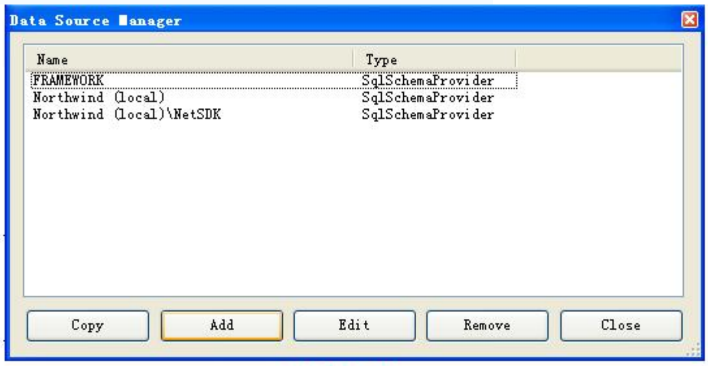


Data Source: Northwind (local)

Tables

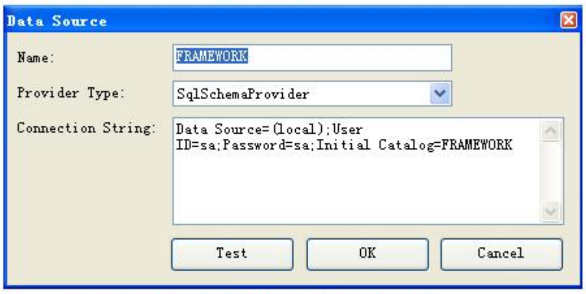
dbo.Categories
dbo.CustomerCustomerDemo
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders
dbo.Products
dbo.Region
dbo.Shippers
dbo.Suppliers
dbo.Territories

点击按钮“..”新建一个DataSource



Name	Type
FRAMEWORK	SqlSchemaProvider
Northwind (local)	SqlSchemaProvider
Northwind (local)\NetSDK	SqlSchemaProvider

点击按钮“Add”



Name: FRAMEWORK

Provider Type: SqlSchemaProvider

Connection String: Data Source=(local);User ID=sa;Password=sa;Initial Catalog=FRAMEWORK

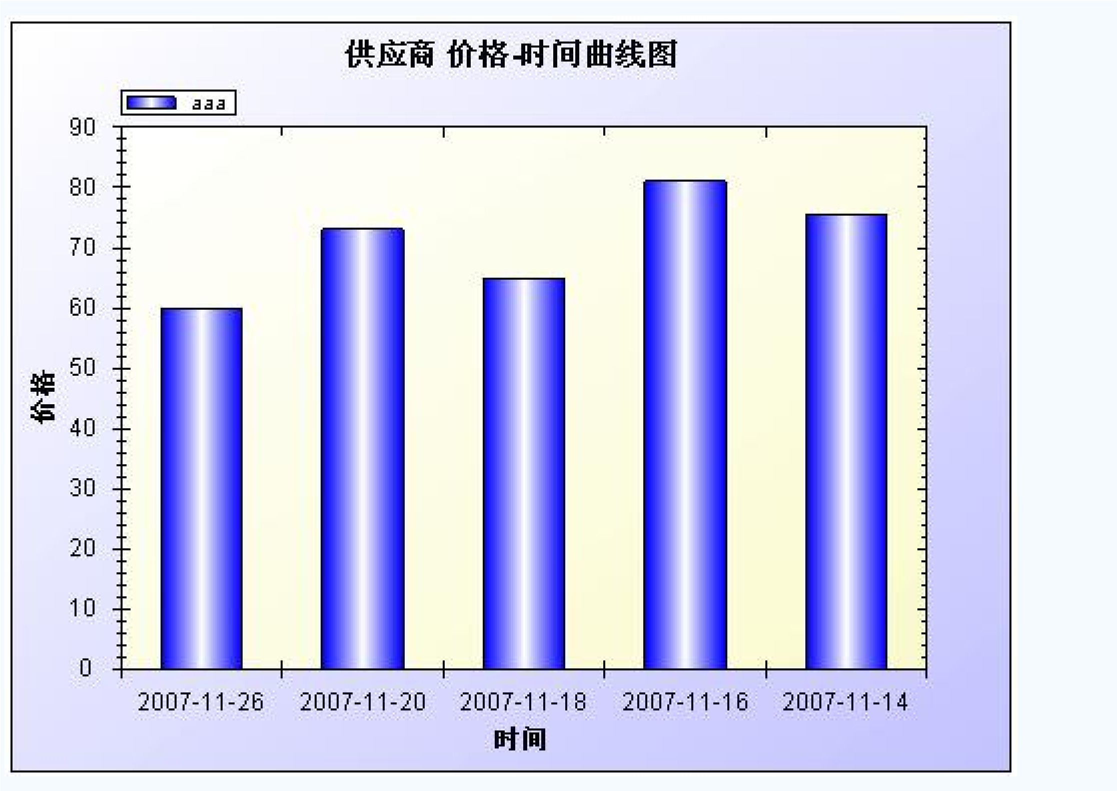
Test测试连接，若成功则点OK退出。回到图1，下拉列表中就有了DataSoure的选项，选择对应的表格

由于日期类型转换成 double 型，结果 X 轴还是比较难看的。目前正在研究它的类库，应该可以更改的。

饼状图，则需要修改 OnRenderGraph 方法：

```
private void OnRenderGraph(System.Drawing.Graphics g, ZedGraph.MasterPane masterPane)
{
    GraphPane myPane = masterPane[0];

    // Title
    myPane.Title.Text = "供应商 价格-时间曲线图";
    myPane.XAxis.Title.Text = "时间";
    myPane.YAxis.Title.Text = "价格";

    // 坐标对集
    PointPairList line = new PointPairList();
    DataTable dt = GetData();

    for(int i=0;i<dt.Rows.Count;i++)
    {
        string date = Convert.ToDateTime(dt.Rows[i]["RecTime"]).ToString("yyyyM
Mdd");
        line.Add(Convert.ToDouble(date), Convert.ToDouble(dt.Rows[i]["Price"]));
    }

    BarItem myCurve = myPane.AddBar("aaa", line, Color.Blue);
    myCurve.Bar.Fill = new Fill(Color.Blue, Color.White, Color.Blue);

    myPane.XAxis.MajorTic.IsBetweenLabels = true;
     // X 轴 Label
    string[] labels = new string[dt.Rows.Count];
    for(int i=0;i<dt.Rows.Count;i++)
        ls[i] = Convert.ToDateTime(dt.Rows[i]["RecTime"]).ToString("yyyy/MM/dd"
);
    myPane.XAxis.Scale.TextLabels = labels;
    myPane.XAxis.Type = AxisType.Text;

    // 颜色填充
    myPane.Fill = new Fill(Color.White, Color.FromArgb(200, 200, 255), 45.0f);
    myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45.0f
);

    masterPane.AxisChange(g);
}
```

效果图：



ZedGraph使用

最好的教程在程序老家的网站.
zedgraph.org 是一个 wiki.(.nettier 的官方网站也是 wiki).
首页是大概介绍.
左边 Navigator 里,ClassDocument 是 API 文档,Sample Graph 里是各种类型的例子(这个很重要).
另外,codeproject 里有这个程序的专门文档.http://www.codeproject.com/csharp/zedgraph.asp

ZedGraph 是开源的.有.net1.1 和.net2.0 不同的版本.

源码中有两个项目,ZedGraph 和 ZedGraph.Web.
如果是开发 winform 程序,只需引用第一个就行了.
要是 asp.net,就得引用这两个了.

ZedGraph 非常灵活,可配置的非常多.
在 asp.net 中有两种方法使用它.
第一种方法用得比较少.
RenderMode.RawImage .
专门制作一个网页来生成图像,在别的网页中通过<img src=this.aspx />来引用这个图片.
具体见:http://zedgraph.org/wiki/index.php?title=Use_RenderMode.RawImage_in_a_web_page

另一种方法是常用字的,把 zedgraph 作为一个控件.
对每一次请求,但会生成缓存图片,得指定存放的文件夹(第一种是即时生成,没有图片要存在服务器上)
在工具箱上"添加项...",浏览取 zedgraph.web.dll.
把这个拖到网页中.

由于参数太多, zedgraph 在网页上生成大量代码,很多参数要在这些小控件上进行配置.
Page_load 事件中自动添加了 zedgraphweb 的 RenderGraph 事件.

对象也非常多.
今天的工作进行得还比较顺利.作出了 Alexa 网站上的效果.
花时间最多还是在配置样式.因为是第一天弄这个,不知道各个对象的意义和包含关系.

这是我用的第一个画图控件,感觉很不错.简单灵活,性能也非常高.
下载了 DundasWinChart,dotnetCHARTING,TeeChart .这些都不是开源的.而且要安装.
水晶报表,想用,但还是没试,觉得做绘图有点大材小用.

在vs中使用ZedGraph控件的一些记录

ZedGraph 是一个非常优秀的开源的作图控件

1、在 vs 中使用 ZedGraph

2、 基本一些概念

几个注意点:

图片的保存路径设置:RenderedImagePath 属性中设置,程序对该文件夹应该是有写和修改权限的

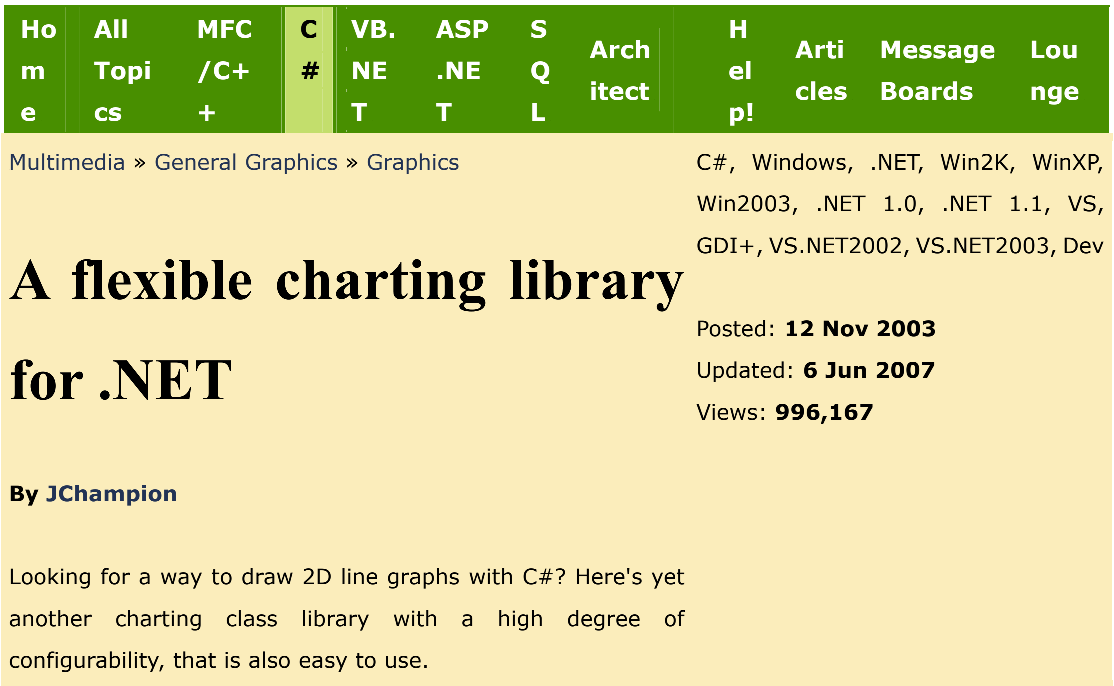
图片的输出格式:OutputFormat 属性中设置,Png 的推荐,比较清晰。

Chart

ChartBorder	图表区域的边框设置
ChartFill	图表区域的背景填充
Legend	图表的注释标签显示设置项目,一组数据对应一种颜色的注释
IsHStack	当有多个显示项的时候设置 Y 轴数据是叠加的还是分开的
Xaxis	图表区域的 X 轴相关信息设置
AxisColor	坐标轴颜色
Cross	坐标的原点,可以设置坐标的偏移程度
CrossAuto	原点自动设置:True 的话 Cross 的设置就无效了。
FontSpec	X 轴标题字体相关信息
Angle	X 轴标题字体显示时候的角度,0 为水平 90 为垂直
Fill	X 轴标题字体填充信息
ColorOpacity	透明度
IsScaled	设置 X 轴标题字体显示大小是否根据图的比例放大缩小
RangeMax	填充时候的最大倾斜度(有过渡色,没试过)
RangeMin	填充时候的最小倾斜度(有过渡色,没试过)
StringAlignment	X 轴标题字体排列(不清楚,没试过)

IsOmitMag	是否显示指数幂（10 次方，没试过，似乎与 IsUseTenPower 有关系）
IsPreventLabelOverlap	坐标值显示是否允许重叠，如果 False 的话，控件会根据坐标值长度自动消除部分坐标值的显示状态
IsShowTitle	X 轴标题是否显示
IsTicsBetweenLabels	两个坐标值之间是否自动显示分隔标志
IsUseTenPower	是否使用 10 次幂指数
IsZeroLine	当数据为 0 时候是否显示（在饼状图显示的时候有用）
IsVisible	是否显示 X 轴
MajorGrid	大跨度的 X 轴表格虚线线显示信息
DashOff	虚线中孔间距
DashOn	虚线单位长度
MajorTic	大跨度的 X 轴刻度信息
IsInside	在 Chart 内部是否显示
IsOutSide	在 Chart 外部是否显示
IsOpposite	在对面的轴上是否显示
MinorGrid	小跨度的 X 轴表格虚线显示信息
MinorTic	小跨度的 x 轴刻度信息
MinSpace	刻度和轴之间的距离（没试过）
Scale	刻度值的一些设定
IsReverse	X 轴的刻度值从高到低还是从低到高
MajorStep	大刻度步长
MajorStepAuto	是否自动设置大刻度步长
MajorUnit	大刻度步长单位
Max	刻度最大值
MaxAuto	根据输入数据自动设置刻度最大值
Min	刻度最小值
MinAuto	根据输入数据自动设置刻度最小值
MinGrace	不清楚，没试过
MinorStep	小刻度步长
MinorStepAuto	是否自动设置小刻度步长
MinorUnit	小刻度单位
Type	数据显示方式
Liner	直接现实（自动）
Date	按日期方式显示
Log	按指数幂方式显示
Ordinal	顺序显示
Y2Axis	第二个 Y 轴坐标信息显示（具体设置看 X 轴）
Yaxis	第一个 Y 轴坐标信息显示（具体设置看 X 轴）
BarBase	在生成柱状图的时候设置柱状是基于 X 轴还是其他轴
BarType	柱状的类型叠加或其他。
IsFontsScaled	图比例变化时候图表上的文字是否跟着自动缩放
IsIgnoreInitial	是否忽略初始值
IsIgnoreMissing	是否忽略缺省值

IsPenWidthScaled	图比例变化时候图表上的画笔的粗细是否跟着自动缩放
IsShowTitle	图表标题是否显示
PaneFill	Pane 的一些填充信息
BaseDimension	缩放比例基数（可以试试效果）
IsImageMap	不清楚干吗用的
AxisChaneged	是否允许自动绘图（没试过，一般都 true，动态绘图）
CacheDuration	Cache 保存时间 0
OutputFormat	输出格式
RenderedImagePath	输出路径
RenderMode	输出模式，不太清楚一般都是 ImageTag，另一个输出的是乱码不是图片。对于图表而言，一般是三种表现形式：柱状图、饼状图和点线图。



Multimedia » General Graphics » Graphics

C#, Windows, .NET, Win2K, WinXP, Win2003, .NET 1.0, .NET 1.1, VS, GDI+, VS.NET2002, VS.NET2003, Dev

A flexible charting library for .NET

Posted: **12 Nov 2003**
Updated: **6 Jun 2007**
Views: **996,167**

By JChampion

Looking for a way to draw 2D line graphs with C#? Here's yet another charting class library with a high degree of configurability, that is also easy to use.

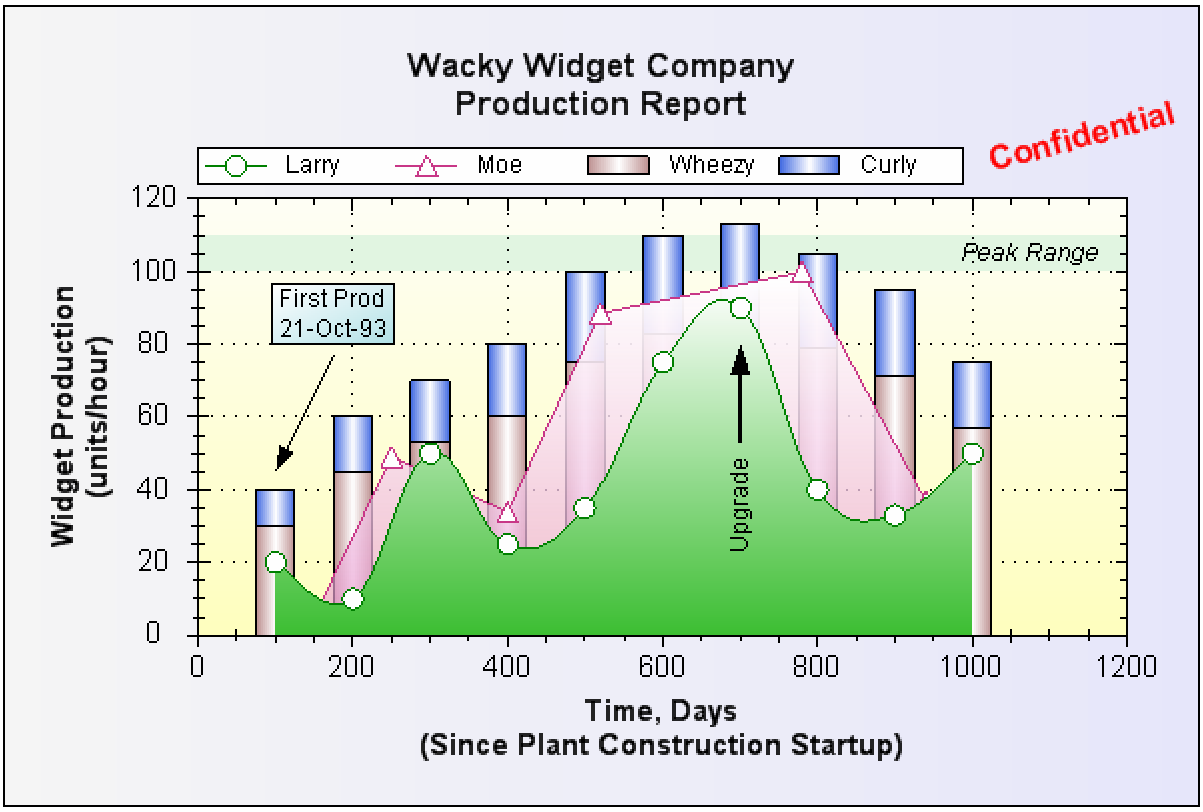
725 votes for this Article.

Print Broken Article? Add Bookmark Discuss Send to a friend

Popularity: 13.13

Rating: **4.59** out of 5

- Download source files (.NET 1.1 and .NET 2.0)
- Download dll only (.NET 1.1 and .NET 2.0)
- Download C# Sample Project
- Download VB Sample Project
- Download the **latest versions** from SourceForge



Introduction

ZedGraph is a class library, Windows Forms UserControl, and ASP web-accessible control for creating 2D line, bar, and pie graphs of arbitrary datasets. The classes provide a high degree of flexibility - almost every aspect of the graph can be user-modified. At the same time, usage of the classes is kept simple by providing default values for all of the graph attributes. The classes include code for choosing appropriate scale ranges and step sizes based on the range of data values being plotted. Moreover, ZedGraph is compatible with .NET 2.0, and VS .NET 2005.

ZedGraph is maintained as an open-source development project on SourceForge. The site includes a project Wiki, documentation, interim (CVS) updates, and all release versions.

A set of sample graphs is also available on the Wiki, complete with source code (many of the samples include C# and VB code).

ZedGraph now supports both .NET 2.0 and .NET 1.1.

- For .NET 2.0, use ZedGraph version 5.0+.
- For .NET 1.1, use ZedGraph version 4.5+.

Background

There are many graphing libraries out there, but none seemed to fit what I needed. I found MSChart too quirky, and many of the other options did not have the flexibility I needed to achieve a polished look. Of course, most of the commercial packages would do the trick, but I needed something that was free. So, ZedGraph was born.

These classes will generate a variety of line, bar, or pie charts on a form or web page, given a location rectangle and some data points. ZedGraph handles 2D line/scatter graphs, horizontal/vertical bar graphs, stacked bar charts, stacked percent bar charts, error bar charts, open-high-low-close charts, Japanese candlestick charts, and pie charts - it does not yet handle 2.5D or 3D surfaces or charts. The charts can be dressed up with axis labels and titles, a legend, text labels and arrows (as shown in the example above), images, etc.

The ZedGraphWiki and the online class documentation provide lots of helpful tips and descriptions. Refer to them for more details - ZedGraph has a tremendous number of options that are not documented in this tutorial.

VB users

This article uses C# exclusively for examples, however, all the code samples for the tutorial, plus other examples are available for Visual Basic, on the ZedGraphWiki Sample Graphs section.

Using ZedGraph as a web control

ZedGraph now includes a class derived from the Control class that facilitates access from ASPX web page code. The demo project download above demonstrates this functionality. Also, the ZedGraphWiki includes detailed examples of web control usage for two rendering modes, RawImage and ImageTag.

Using ZedGraph as a UserControl

ZedGraph is accessible as a control from the control toolbox in Visual Studio .NET. To access ZedGraph, first launch Visual Studio .NET, and create a new Windows Application (Forms) project. Open the form design so that it appears in the current window. View the toolbox using the *View/Toolbox* menu command. Right-click inside the "General" or "Components" sub-pane of the tool box, and select the "Choose Items..." option. Click "Browse...", and navigate to the *zedgraph.dll* file. Once this file is added, you should see a ZedGraphControl option in the toolbox.

1. In your project, under the *Project* menu, select the "Add Reference..." option. Use the *Browse* button to find *ZedGraph.dll*, and click OK. Repeat this for ZedGraph.Web.dll. This will include all the functionality of ZedGraph into your project.
2. Add a `using ZedGraph;` entry to your main form code.
3. In the form designer, drag the `ZedGraphControl` from the Toolbox over to the form, and drag/size it as desired. You now have a ZedGraph control in your form.
4. All of the ZedGraph functionality is accessible through the `ZedGraphControl.MasterPane` property. A `ZedGraphControl.GraphPane` is also provided, which simply references the first `GraphPane` in the `MasterPane` list (this is explained below).
5. In the Form designer, double click the form (not the `ZedGraphControl`). This will place a `Form1_Load()` template in your code file.
6. Again, in the Form designer, with the form selected, go to Properties, pick the yellow lightning bolt to see the events, put the cursor in the box to the right of the `Resize` event, and hit Enter to add a `Form1_Resize` event method to your code file.
7. Modify the `Form1_Load()` and `Form1_Resize` methods, and add the `CreateGraph()` and `SetSize()` methods as shown below (this code assumes the name of your control is '`zedGraphControl1`'). <Note: A complete solution containing this code can be download from the ZedGraph Wiki:

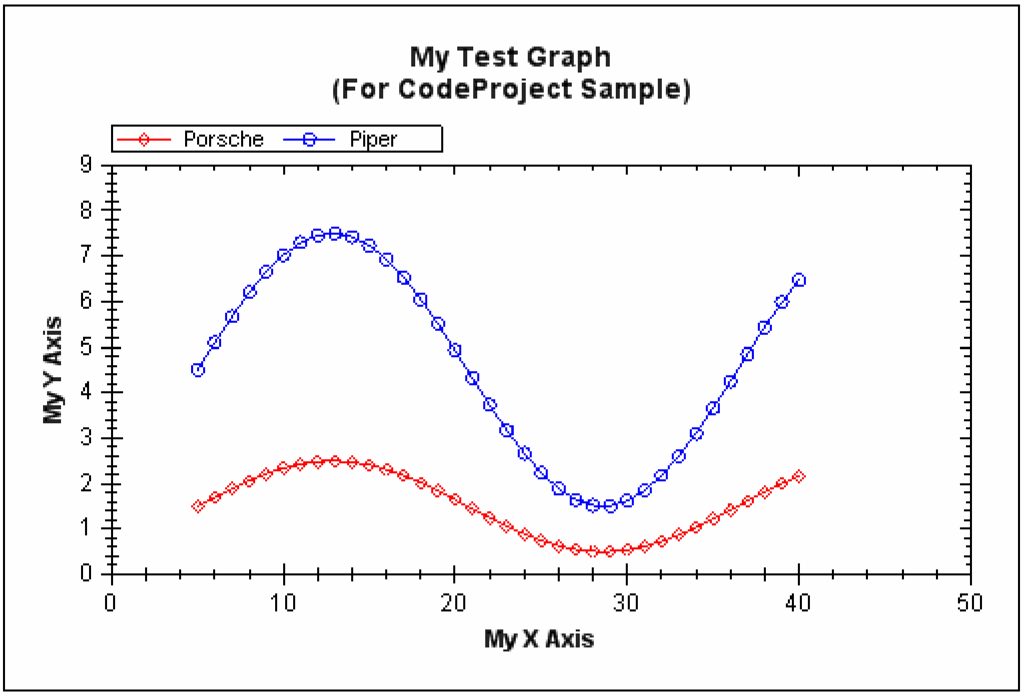
Collapse

```
// Respond to the form 'Resize' event
private void Form1_Resize( object sender, EventArgs e )
{
   SetSize();
}

// SetSize() is separate from Resize() so we can
// call it independently from the Form1_Load() method
// This leaves a 10 px margin around the outside of the control
// Customize this to fit your needs
private void SetSize()
{
   zedGraphControl1.Location = new Point( 10, 10 );
   // Leave a small margin around the outside of the control
   zedGraphControl1.Size = new Size( ClientRectangle.Width - 20,
                                     ClientRectangle.Height - 20 );
}

// Respond to form 'Load' event
private void Form1_Load( object sender, EventArgs e )
{
   // Setup the graph
   CreateGraph( zedGraphControl1 );
   // Size the control to fill the form with a margin
   SetSize();
}

// Build the Chart
private void CreateGraph( ZedGraphControl zgc )
{
   // get a reference to the GraphPane
   GraphPane myPane = zgc.GraphPane;

   // Set the Titles
   myPane.Title.Text = "My Test Graph\n(For CodeProject Sample)";
   myPane.XAxis.Title.Text = "My X Axis";
   myPane.YAxis.Title.Text = "My Y Axis";

   // Make up some data arrays based on the Sine function
   double x, y1, y2;
   PointPairList list1 = new PointPairList();
   PointPairList list2 = new PointPairList();
   for ( int i = 0; i < 36; i++ )
   {
```

```
            x = (double)i + 5;
            y1 = 1.5 + Math.Sin( (double)i * 0.2 );
            y2 = 3.0 * ( 1.5 + Math.Sin( (double)i * 0.2 ) );
            list1.Add( x, y1 );
            list2.Add( x, y2 );
        }

        // Generate a red curve with diamond
        // symbols, and "Porsche" in the legend
        LineItem myCurve = myPane.AddCurve( "Porsche",
              list1, Color.Red, SymbolType.Diamond );

        // Generate a blue curve with circle
        // symbols, and "Piper" in the legend
        LineItem myCurve2 = myPane.AddCurve( "Piper",
              list2, Color.Blue, SymbolType.Circle );

        // Tell ZedGraph to refigure the
        // axes since the data have changed
        zgc.AxisChange();
    }
```

Note that the AxisChange() method call must be made any time you add or change the data. This tells ZedGraph to go ahead and recalculate all the axis ranges. (Note: This is all AxisChange() does - you can call it anytime you like, and it will update the axis ranges based on the current set of data points. You can also avoid calling AxisChange() if you do not want the axes rescaled.)

The above code will generate the following output:



Enhancing the graph

ZedGraph allows you to modify the graph attributes in a wide variety of ways. Each of the parts of the graph is encapsulated in a class structure, which has modifiable properties to control the output. The following are some of the classes provided in ZedGraph (Note that these classes are XML documented. See the ZedGraph documentation for details about each class.):

Class	Description
MasterPane	A class to manage multiple GraphPane objects, derived from PaneBase. Use of the MasterPane class is optional, as the GraphPane class can be used directly for a single pane. Also provides methods for layout, arrangement, and management of the individual GraphPane objects.
GraphPane	The primary class for the graph, derived from PaneBase. Includes all other classes as properties. Also controls the pane title, the pane frame and axis frame, backgrounds, etc.
XAxis, YAxis, Y2Axis	Children of the Axis class. These classes include many aspects of the axis display, including tics, grids, colors, pens, fonts, labels, and styles.
Scale	A class instance maintained by the Axis class. Contains the scale range, step sizes, formats, and display options for the scale. Comes in variants for Linear, Log, Text, Date, Ordinal,

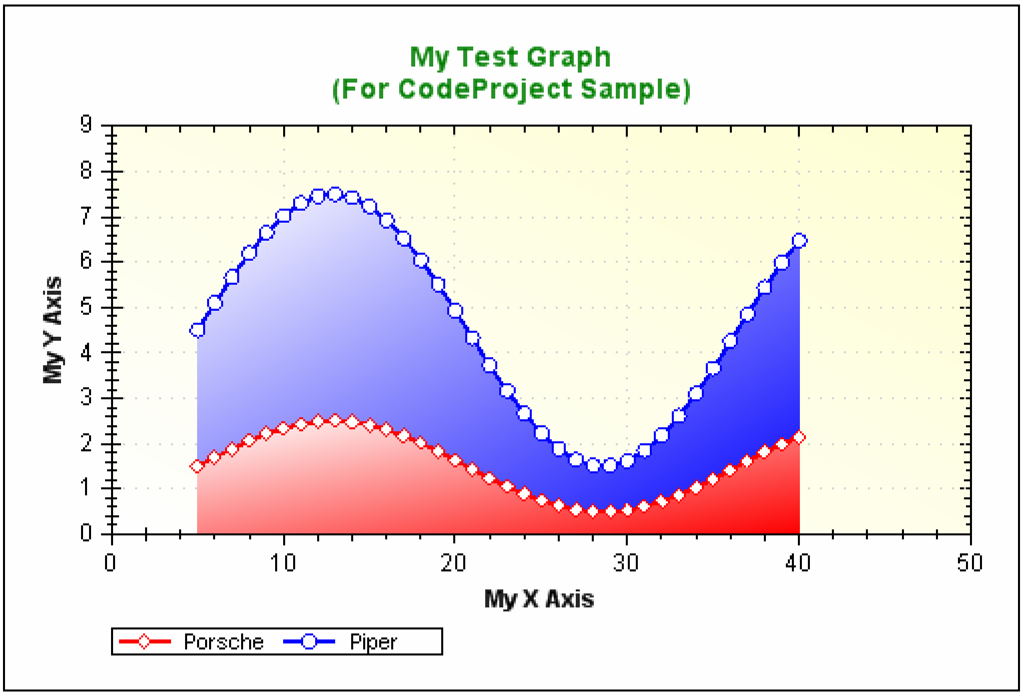
	Exponent, LinearAsOrdinal, and DateAsOrdinal scales.
Legend	The class that describes the location, font, colors, etc., used to draw the legend.
CurveItem	An abstract base class that contains data for a single curve. LineItem, BarItem, HiLowBarItem, ErrorBarItem, PieItem, StickItem, OHLCBarItem, and JapaneseCandleStickItem are all derived from this class.
CurveList	A collection class to maintain a list of CurveItem objects. The order of the curves in the list controls the Z-Order for drawing. The last curve in the list will appear behind all other curves.
GraphObj	An abstract base class that includes position information for a variety of supplemental graphic objects on a plot. TextObj, ImageObj, LineObj, ArrowObj, EllipseObj, BoxObj, and PolyObj are derived from GraphObj.
GraphObjList	A collection class to maintain a list of GraphObj objects. The order of the objects in the list, plus a ZOrder property, control the Z-Order for drawing. The last item in the list will appear behind all other items with the same ZOrder value.
FontSpec	A utility class that includes information about the font family, color, angle, size, style, frame, and background fill of the text on the graph. Each class that includes text information will contain one or more FontSpec objects to specifically describe the associated fonts.
Fill	A utility class that includes characteristics of background color fills. Each object that has color fill capability will contain one or more Fill objects to specifically describe the associated color fill.
Border	A utility class that includes characteristics of object borders. Each object that has border capability will contain one or more Border objects to specifically describe the associated border color and line properties.
Location	A general class for handling the location of graphic objects on the plot.
PointPair	A data struct that encapsulates a single pair of double values representing an (X,Y) data point. This is the internal data storage format for the value arrays in each CurveItem.
PointPairList	A collection class to maintain a list of PointPair objects.
XDate	This class encapsulates a single date-time value (stored as a System.Double), plus a wide array of methods to convert between XL date, Astronomical Julian Day number, Gregorian Calendar date, fractional year, etc. See the discussion of Date-Time axes below, for details.

The graph is modified by accessing properties in each of the above classes. For example, if you include the following lines of code in your CreateGraph() method, after the code samples shown previously, the plot will be modified accordingly:

Collapse

```
// Change the color of the title
myPane.Title.FontSpec.FontColor = Color.Green;

// Add gridlines to the plot, and make them gray
myPane.XAxis.MajorGrid.IsVisible = true;
myPane.YAxis.MajorGrid.IsVisible = true;
myPane.XAxis.MajorGrid.Color = Color.LightGray;
myPane.YAxis.MajorGrid.Color = Color.LightGray;

// Move the legend location
myPane.Legend.Position = ZedGraph.LegendPos.Bottom;

// Make both curves thicker
myCurve.Line.Width = 2.0F;
myCurve2.Line.Width = 2.0F;

// Fill the area under the curves
myCurve.Line.Fill = new Fill( Color.White, Color.Red, 45F );
myCurve2.Line.Fill = new Fill( Color.White, Color.Blue, 45F );

// Increase the symbol sizes, and fill them with solid white
myCurve.Symbol.Size = 8.0F;
myCurve2.Symbol.Size = 8.0F;
myCurve.Symbol.Fill = new Fill( Color.White );
myCurve2.Symbol.Fill = new Fill( Color.White );

// Add a background gradient fill to the axis frame
myPane.Chart.Fill = new Fill( Color.White,
    Color.FromArgb( 255, 255, 210 ), -45F );

// Add a caption and an arrow
TextObj myText = new TextObj( "Interesting\nPoint", 230F, 70F );
myText.FontSpec.FontColor = Color.Red;
myText.Location.AlignH = AlignH.Center;
myText.Location.AlignV = AlignV.Top;
myPane.GraphObjList.Add( myText );
ArrowObj myArrow = new ArrowObj( Color.Red, 12F, 230F, 70F, 280F, 55F );
myPane.GraphObjList.Add( myArrow );
```

The final "dressed-up" graph will look like this:



Many more graph attributes can be modified, but listing them all in this article would be tedious. Please refer to the online documentation for more details. Also see the demo graphs on the Wiki for samples with code.

Interesting tidbits

Drawing line graphs is no big deal, but there are some aspects of the drawing classes that proved to be interesting. The flexibility afforded by the transform matrix of the GDI+ drawing library is very cool. This allowed the same code to be employed for drawing all three of the axes. The coordinate system is transformed to accommodate the axis location and orientation. In each case, the coordinate system is translated and rotated so that the axis is oriented along the X direction and the origin is located at the left edge of the axis when facing from the label side. A few "if" exceptions were required to account for the fact that the "left" side of the X and Y2 axes is the minimum value and the "left" side of the Y axis is the maximum value.

Choosing scales

Range and step size

ZedGraph is set up to automatically select appropriate scale minimum, maximum, and step size values based on the range of data values in the curves. Alternatively, you can manually set any or all of the values, and the scale picking logic will attempt to pick the appropriate values for the remaining parameters that are left in the automatic mode. The scale picking logic is based on the assumption that the most humanly palatable step sizes will be even divisors of 10. That is, step sizes should be 1, 2, or 5 times some power of 10. The heart of the scale picking logic is found in the CalcStepSize() method:

```
protected double CalcStepSize( double range, double targetSteps )
{
    // Calculate an initial guess at step size
    double tempStep = range / targetSteps;

    // Get the magnitude of the step size
    double mag = Math.Floor( Math.Log10( tempStep ) );
    double magPow = Math.Pow( (double) 10.0, mag );

    // Calculate most significant digit of the new step size
    double magMsd =  ( (int) (tempStep / magPow + .5) );

    // promote the MSD to either 1, 2, or 5
    if ( magMsd > 5.0 )
        magMsd = 10.0;
    else if ( magMsd > 2.0 )
        magMsd = 5.0;
    else if ( magMsd > 1.0 )
        magMsd = 2.0;

    return magMsd * magPow;
}
```

The initial guess at the step size is calculated using a default number of steps named targetSteps (this value is defaulted to 7), which is the typical number of major steps you want on an axis. The actual number of steps will usually end up being this number or more. A magnitude of the initial step size is determined using Floor(Log10(stepSize)). This magnitude is reduced to a most significant digit, which is then promoted to either 1, 2, or 5 to make the scale an even divisor of 10. In general, I find that this logic is very good at picking scales based on the data range.

Zero lever

Another aspect of the scale selection is whether or not the scale should be extended to include the zero value. For example, if the scale range is 1 to 10, you would typically want to go ahead and start it at zero. This is accomplished with the ZeroLever default parameter. The ZeroLever is the allowable fraction that the scale range can be extended to include the zero value. This applies below the scale range for positive scales, or above the scale range for negative scale values. As an example, if the ZeroLever is 0.25 and the data range is from 2.0 to 12.0, then the scale would be extended to a range of 0.0 to 12.0 since the zero value lies 20% outside the actual data range (which is within the 25% allowed).

Grace

Finally, ZedGraph includes "grace" properties that allow you to include extra space on the scale range before and after the minimum and maximum data values, respectively. The reason for this is to avoid having the minimum and/or maximum data points fall right on the axes. Note that the grace values apply only to the non-ordinal axis types (AxisType.Log, AxisType.Linear, and AxisType.Date). The grace values are controlled by Axis.Scale.MinGrace and Axis.Scale.MaxGrace. These values are expressed as a fraction of the total data range. For example, if MinGrace and MaxGrace are both set to 0.1, then 10% of the data range will be padded before and after the actual data range. For example, if the data values go from 50.0 to 150.0, then the data range is 100.0. 10% of this range is 10.0. Adding this before and after the range gives an effective data range of 40.0 to 160.0. The grace properties will not cause the range to extend across the zero point. That is, if both the min and max data values are zero or greater, then the axis will not be extended to negative values regardless of the grace setting. The default values for MinGrace and MaxGrace are handled by Scale.Default.MinGrace and Scale.Default.MaxGrace, respectively. Both values are set to 0.1 initially.

Axis types

Each axis now has a property called Type, which can be set to one of these eight values:

- AxisType.Linear: This is the default type, which is just a regular Cartesian axis.

- AxisType.Log: This is a base 10 logarithmic scale.
- AxisType.Exponent: This is an exponential scale.
- AxisType.Date: This is a date-time axis, in which the corresponding values are XDate types. See Date-Time Axes below.
- AxisType.Text: This is an ordinal type, in which the data values for this axis are actually not used. All points or bars will be evenly spaced on the graph. The first point will have a value of 1.0, the second will be 2.0, etc. The actual value labels will be determined by an array of strings in Axis.TextLabels. See Text Axes below.
- AxisType.Ordinal: This is another ordinal type, in which the data values for this axis are not used. All points or bars will be evenly spaced on the graph. The first point will have a value of 1.0, the second will be 2.0, etc. The actual value labels are just determined by the numeric ordinal values (1.0, 2.0, etc.).
- AxisType.LinearAsOrdinal: This is another ordinal type, essentially the same as AxisType.Ordinal, except that the axis scale values will be displayed based on the data values (although the axis will still be treated as ordinal, with 1.0 for the first label, 2.0 for the second, etc.).
- AxisType.DateAsOrdinal: This is another ordinal type, essentially the same as AxisType.Ordinal, except that the axis scale values will be displayed based on the data values as dates. This mode is a good choice if you have weekday values and you want to skip the weekends without leaving gaps in the data.

Date-time axes

ZedGraph includes the capability to handle date-time axes, e.g., the axis labels can be based on an encoded date-time value, displayed in anything from seconds to years. At the heart of the date-time axis is the XDate struct, which captures a time value, and handles conversions to/from a wide range of date-time formats. You're probably wondering why I reinvented the wheel on this, rather than just using the built-in DateTime class (or equivalent). The main reason I made my own class is because an XDate stores its date-time information as a System.Double value. Therefore, XDate values can just be stored in an ordinary array of doubles, just like any other array of data that is passed to ZedGraph. The XDate struct and format are described in detail in the Wiki. If you are using DateTime structs to store your data, you can convert directly to doubles which are compatible with ZedGraph, by using the DateTime.ToOADate() method.

To use a date-time axis in ZedGraph, you need to change to AxisType.Date as follows:

```
myPane.XAxis.Type = AxisType.Date;
```

ZedGraph will then assume that the X array double values are actually XDate values. XAxis.Scale.Min and XAxis.Scale.Max will also be XDate values, but the units of XAxis.Scale.MajorStep and XAxis.Scale.MinorStep will depend on the setting of XAxis.Scale.MajorUnit and XAxis.Scale.MinorUnit, respectively. XAxis.Scale.MajorUnit and XAxis.Scale.MinorUnit can be set to one of the following values: DateUnit.Year, DateUnit.Month, DateUnit.Day, DateUnit.Hour, DateUnit.Minute, DateUnit.Second, and DateUnit.Millisecond. Note again that XDate values are actually units of days (actually, days since a reference date). However, if XAxis.Scale.MajorUnit = DateUnit.Year, then a value of 0.25 for XAxis.Scale.MajorStep means that the step size is 1/4th of a year. One more extra property is used for the date-time axes: the Scale.Format. This is a string that defines the format of the major tic labels for the axis. For example, XAxis.Scale.Format = "dd-MMM-yy" would give labels like "12-Dec-95". A wide range of formatting options is available - see the Wiki for details. Note that the formatting is based on DateTimeFormatInfo, which is the same as the DateTime struct.

Assuming XAxis.Scale.MinAuto, XAxis.Scale.MajorStepAuto, and XAxis.Scale.MaxAuto are all set to true, ZedGraph will attempt to select an axis range that fits the span of the dates. This means that the auto-scaling feature will automatically set the following properties: Axis.Scale.Min, Axis.Scale.Max, Axis.Scale.MajorStep, Axis.Scale.MinorStep, Axis.Scale.MajorUnit, Axis.Scale.MinorUnit, and Axis.Scale.Format.

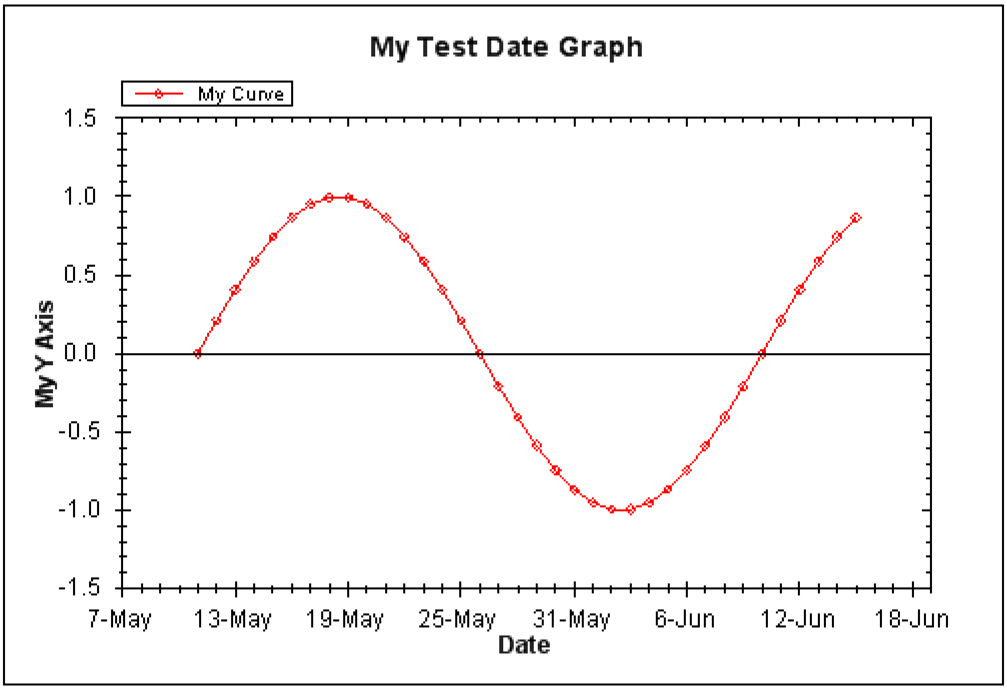
If you want to try this out, here's the code for an example graph, which has a data point for the first day of each month, for 30 months starting with 1-Jan-1995:

Collapse

```
// Build the Chart
private void CreateGraph( ZedGraphControl zg1 )
{
   // Get a reference to the GraphPane
   GraphPane myPane = zg1.GraphPane;

   // Set the titles
   myPane.Title.Text = "My Test Date Graph";
   myPane.XAxis.Title.Text = "Date";
   myPane.XAxis.Title.Text = "My Y Axis";

   // Make up some random data points
```

```
    double x, y;
    PointPairList list = new PointPairList();
    for ( int i=0; i<36; i++ )
    {
        x = (double) new XDate( 1995, 5, i+11 );
        y = Math.Sin( (double) i * Math.PI / 15.0 );
        list.Add( x, y );
    }

    // Generate a red curve with diamond
    // symbols, and "My Curve" in the legend
    CurveItem myCurve = myPane.AddCurve( "My Curve",
          list, Color.Red, SymbolType.Diamond );

    // Set the XAxis to date type
    myPane.XAxis.Type = AxisType.Date;

    // Tell ZedGraph to refigure the axes since the data
    // have changed
    zg1.AxisChange();
}
```

The above code generates the following graph:



In this case, ZedGraph selected a major step size of six days, and a minor step size of one day. You can easily adjust the selected format, if desired. For example, if you want to use 1 month for the minor step size, just set myPane.XAxis.Scale.MinorStep = 1.0 and myPane.XAxis.Scale.MinorUnit = DateUnit.Month just before the AxisChange() call. If you change the XAxis.Scale.MajorStep value, you will also have to manually set XAxis.Scale.MajorUnit, XAxis.Scale.MinorUnit, XAxis.Scale.MinorStep, and XAxis.Scale.Format. This is because XAxis.Scale.MajorStepAuto == false since you have chosen to manually select the step size.

Text axis

ZedGraph also handles a text axis. This is an axis in which the tic labels are arbitrary, user supplied text strings instead of value labels. Internally, a text axis is handled using ordinal values just like an ordinary axis. In this case, the first major label has a value of 1.0, the second major label has a value of 2.0, etc. It is permissible to use fractional values if you want to place points in-between the labels.

To make a text axis, you set Axis.Type = AxisType.Text. This informs ZedGraph to use the labels supplied by the user in Axis.Scale.TextLabels. The number of labels will determine the axis range. That is, 10 labels means the axis will be ranged from 1.0 to 10.0. Optionally, when you add a curve to ZedGraph, you can just skip any value array that is associated with a text axis. A default array of ordinal values will be generated. For example, if the XAxis is of type Text with 10 labels, you can add a curve, leaving the X array null, and an X array will be generated internally with values from 1.0 to 10.0. The bar chart below shows the usage of the AxisType.Text.

Bar charts

ZedGraph includes bar charting capability for vertical and horizontal bar charts, stacked bar charts, percent stacked bar charts, overlay bar charts, error bar charts, high-low bar charts, open-high-low-close bar charts, and Japanese candlestick bar charts. A bar chart is created similar to a line graph, except that you use GraphPane.AddBar(), GraphPane.AddErrorBar(), or GraphPane.AddHiLowBar() to create the bar instance. It is possible to mix bars, lines, and symbols on the same graph, by simply adding the different types.

Bars can be made horizontal or vertical, by setting the "base" axis to the X or Y axis. Under the ZedGraph terminology, the "base" axis determines the bar position, and the "value" axis determines the height of the bar.

Typically, bar charts would be created with XAxis.Type = AxisType.Text or XAxis.Type = AxisType.Ordinal (both types use ordinal values), such that the bars are drawn at integral values along the "base" axis, starting with 1 (e.g., the first bar cluster is at 1.0, the second is at 2.0, etc.). However, the ordinal axis type is not a requirement for bar charts. It is possible to create a bar chart that is not evenly spaced, by providing X values and using AxisType.Linear (in this case, you may need to use the GraphPane.ClusterScaleWidth property to tell ZedGraph how wide the bars should be. See this wiki page for details). For bar charts, the tic marks are typically between the bar clusters, which can be accomplished with the Axis.MajorTic.IsBetweenLabels property. However, this property is only applicable for AxisType.Text axes.

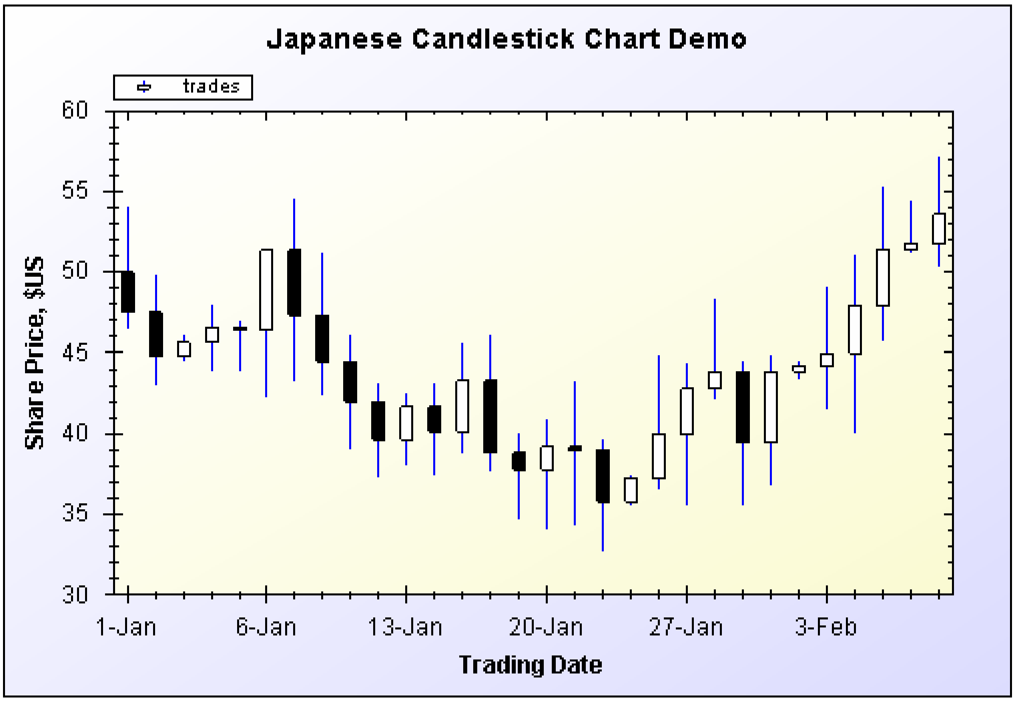
ZedGraph actually has six distinct bar types, any of which can be horizontal or vertical:

- BarItem - is for regular bars, stacked bars, percent stacked bars, overlay bars, and sorted overlay bars.
- ErrorBarItem - is for error bars, which are "I-Beam" bars with a symbol at each end, based on the upper and lower values that are user-defined.
- HiLowBarItem - is for rectangular bars that have both upper and lower values that are user-defined.

Other "bar-like" types are also available:

- OHLCBarItem - is for open-high-low-close stock charts, similar to an error bar, and showing the open, close, high, and low prices for the day.
- JapaneseCandleStickItem - another stock chart type showing a narrow vertical line depicting the high-low range of prices for the day, plus a colored bar showing the open and close with a different color for rising and falling days.
- StickItem - a chart with a narrow line at each value going back to the X axis.

The following is an example of a JapaneseCandleStickItem chart, used to show High-Low-Open-Close data for the stock market:



The orientation (horizontal or vertical) and the size of the BarItem bars are determined globally by GraphPane.BarSettings.Base and other GraphPane.BarSettings properties. Therefore, all BarItem bars will have similar properties, and the size of the bars is scaled automatically to fill the available space. In contrast, the size of ErrorBarItem bars and HiLowBarItem bars are controlled by individual properties for each bar item, e.g., ErrorBarItem.Bar.Size. These bar types are actually similar to symbols, since the bar width is specified in points (1/72 inch). A single plot can have a variety of different ErrorBarItems and HiLowBarItems with different sizes.

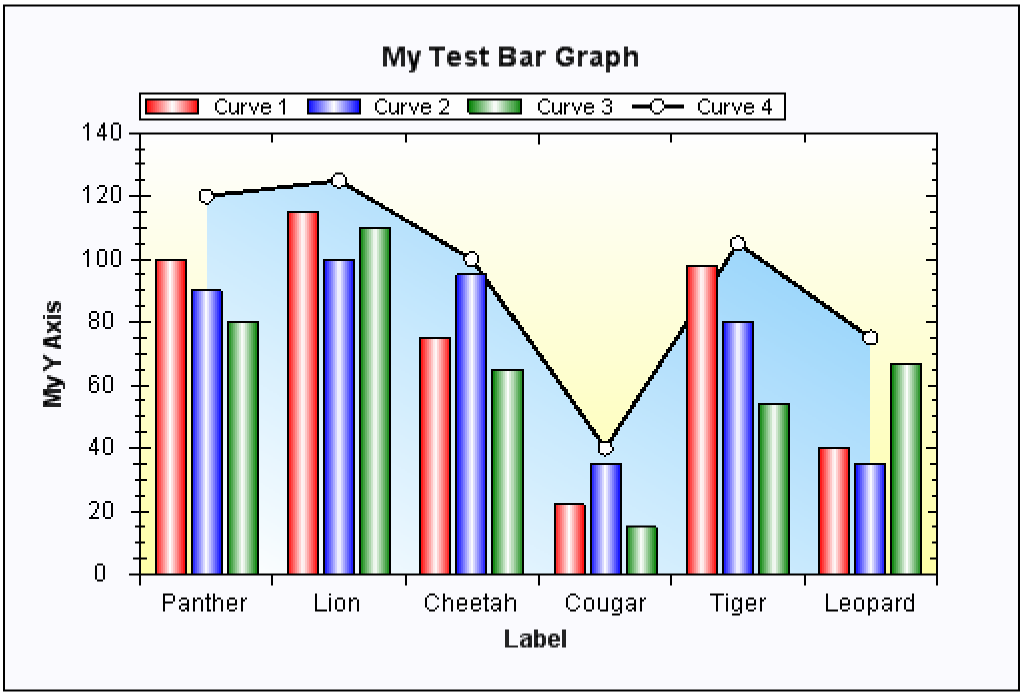
Two properties are included in the GraphPane class to control the gaps between BarItem bars; GraphPane.BarSettings.MinBarGap (default = 0.2) is the minimum size of the gap between each bar within a bar cluster (multiple bars that share the same X value), and GraphPane.BarSettings.MinClusterGap (default = 1.0) is the minimum size of the gap between the bar clusters. Both of these parameters are expressed as a fraction of the individual bar size, i.e., a value of 1.0 would make the gap the same size as the bars. Note that these properties apply only to BarItem bars (not ErrorBarItem or HiLowBarItem bars). A new Bar class has been added to the BarItem class to control the properties of the bars. This Bar class has properties for Fill, FrameColor, FrameWidth, and IsFramed. The following example generates a simple bar chart:

Collapse

```
// Build the Chart
private void CreateGraph( ZedGraphControl zg1 )
```

```
{
    // get a reference to the GraphPane
    GraphPane myPane = zg1.GraphPane;

    // Set the Titles
    myPane.Title.Text = "My Test Bar Graph";
    myPane.XAxis.Title.Text = "Label";
    myPane.YAxis.Title.Text = "My Y Axis";

    // Make up some random data points
    string[] labels = { "Panther", "Lion", "Cheetah",
                       "Cougar", "Tiger", "Leopard" };
    double[] y = { 100, 115, 75, 22, 98, 40 };
    double[] y2 = { 90, 100, 95, 35, 80, 35 };
    double[] y3 = { 80, 110, 65, 15, 54, 67 };
    double[] y4 = { 120, 125, 100, 40, 105, 75 };

    // Generate a red bar with "Curve 1" in the legend
    BarItem myBar = myPane.AddBar( "Curve 1", null, y,
                                            Color.Red );
    myBar.Bar.Fill = new Fill( Color.Red, Color.White,
                                            Color.Red );

    // Generate a blue bar with "Curve 2" in the legend
    myBar = myPane.AddBar( "Curve 2", null, y2, Color.Blue );
    myBar.Bar.Fill = new Fill( Color.Blue, Color.White,
                                            Color.Blue );

    // Generate a green bar with "Curve 3" in the legend
    myBar = myPane.AddBar( "Curve 3", null, y3, Color.Green );
    myBar.Bar.Fill = new Fill( Color.Green, Color.White,
                                            Color.Green );

    // Generate a black line with "Curve 4" in the legend
    LineItem myCurve = myPane.AddCurve( "Curve 4",
          null, y4, Color.Black, SymbolType.Circle );
    myCurve.Line.Fill = new Fill( Color.White,
                          Color.LightSkyBlue, -45F );

    // Fix up the curve attributes a little
    myCurve.Symbol.Size = 8.0F;
    myCurve.Symbol.Fill = new Fill( Color.White );
    myCurve.Line.Width = 2.0F;
    // Draw the X tics between the labels instead of
    // at the labels
    myPane.XAxis.MajorTic.IsBetweenLabels = true;

    // Set the XAxis labels
    myPane.XAxis.Scale.TextLabels = labels;
    // Set the XAxis to Text type
    myPane.XAxis.Type = AxisType.Text;

    // Fill the Axis and Pane backgrounds
    myPane.Chart.Fill = new Fill( Color.White,
          Color.FromArgb( 255, 255, 166), 90F );
    myPane.Fill = new Fill( Color.FromArgb( 250, 250, 255) );

    // Tell ZedGraph to refigure the
    // axes since the data have changed
    zg1.AxisChange();
}
```

The above code generates the following graph:

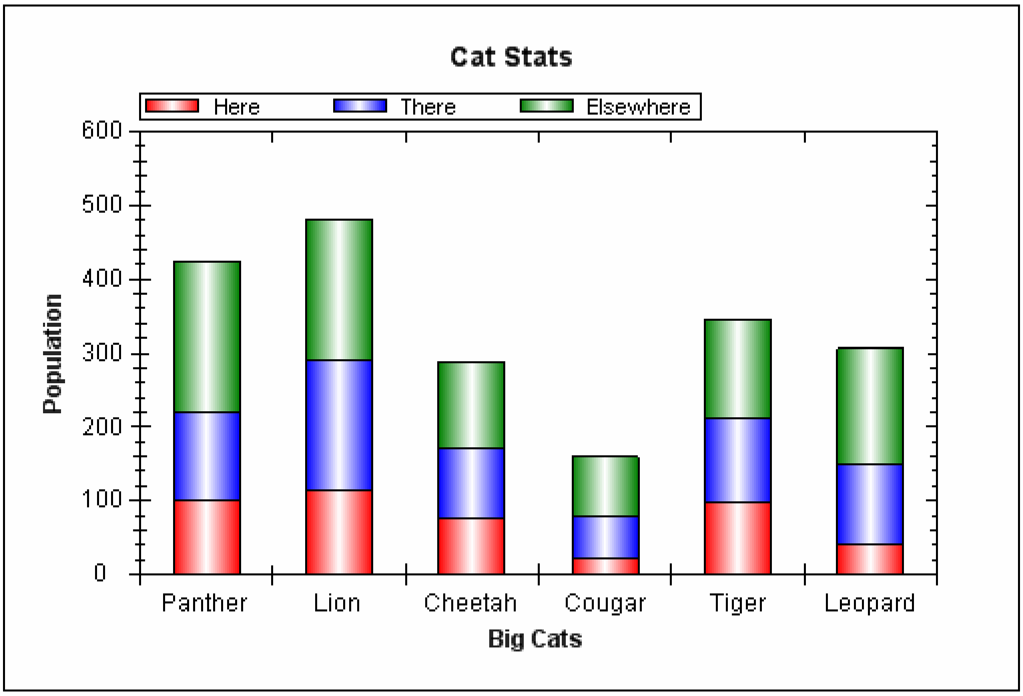


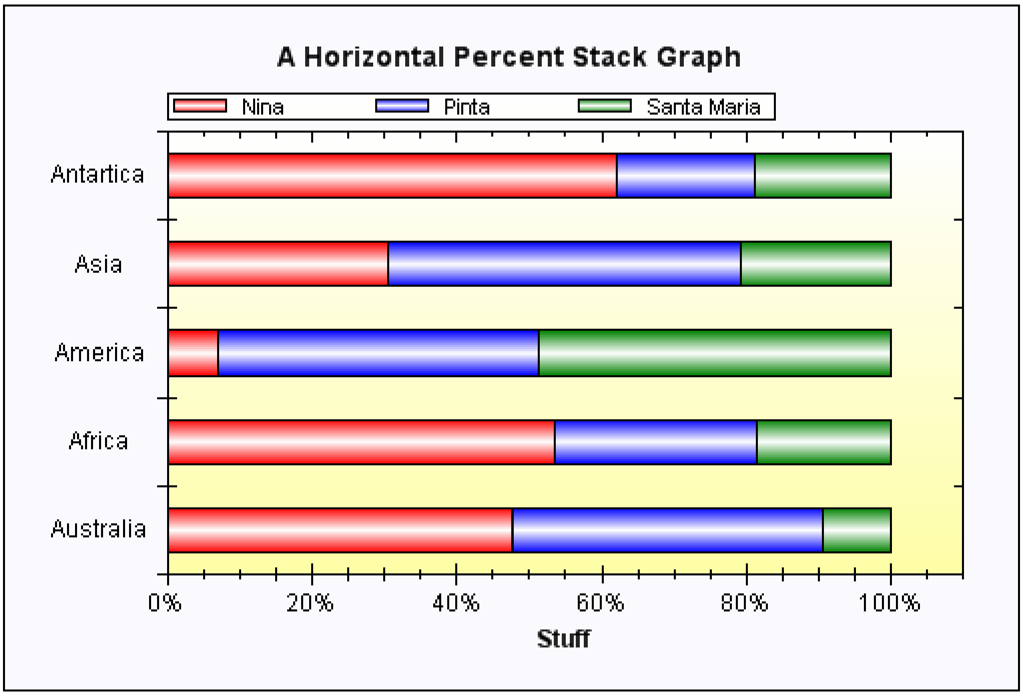
Bar types

ZedGraph can draw BarItem bar charts in a variety of types according to the GraphPane.BarSettings.Type property. This can be one of the following values:

BarType	Description
BarType.Cluster	This is the normal format in which various bar series are grouped together in clusters at each base value (like the first example chart above).
BarType.ClusterHiLow	This format draws a hi-low (bars have a top and bottom that are user defined) in a cluster format, so multiple high-low bars can be grouped together at each base value.
BarType.Overlay	In this format, the bars are drawn on top of each other, with the first BarItem drawn at the back, and the last BarItem drawn at the front.
BarType.SortedOverlay	This is similar to Overlay, but the bars are sorted on value, and the highest value is drawn at the back, and the lowest value is drawn at the front.
BarType.Stack	The bars are stacked on top of each other, accumulating in value.
BarType.PercentStack	The bars are stacked on top of each other, and plotted as a percentile, with the total height always being 100%.

The following samples show horizontal and stacked bar types:



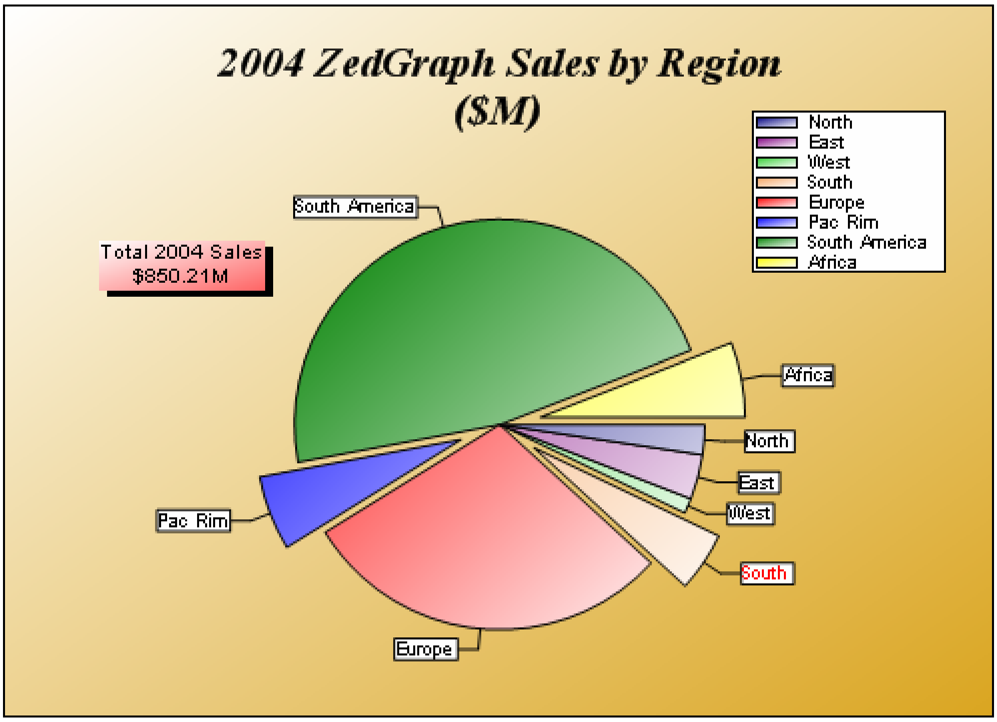


Pie charts

Pie charts are created in the normal fashion using the GraphPane.AddPieSlice(), which returns a PieItem. One PieItem is added for each slice of the pie. Note that, unlike the other CurveItem - derived classes, the PieItem does not use the PointPairList to store the data value. Since the pie has only a single data value, it is stored in PieItem.PieValue. The pie charts support text labels, a legend, color fills, etc. in the typical ZedGraph fashion.

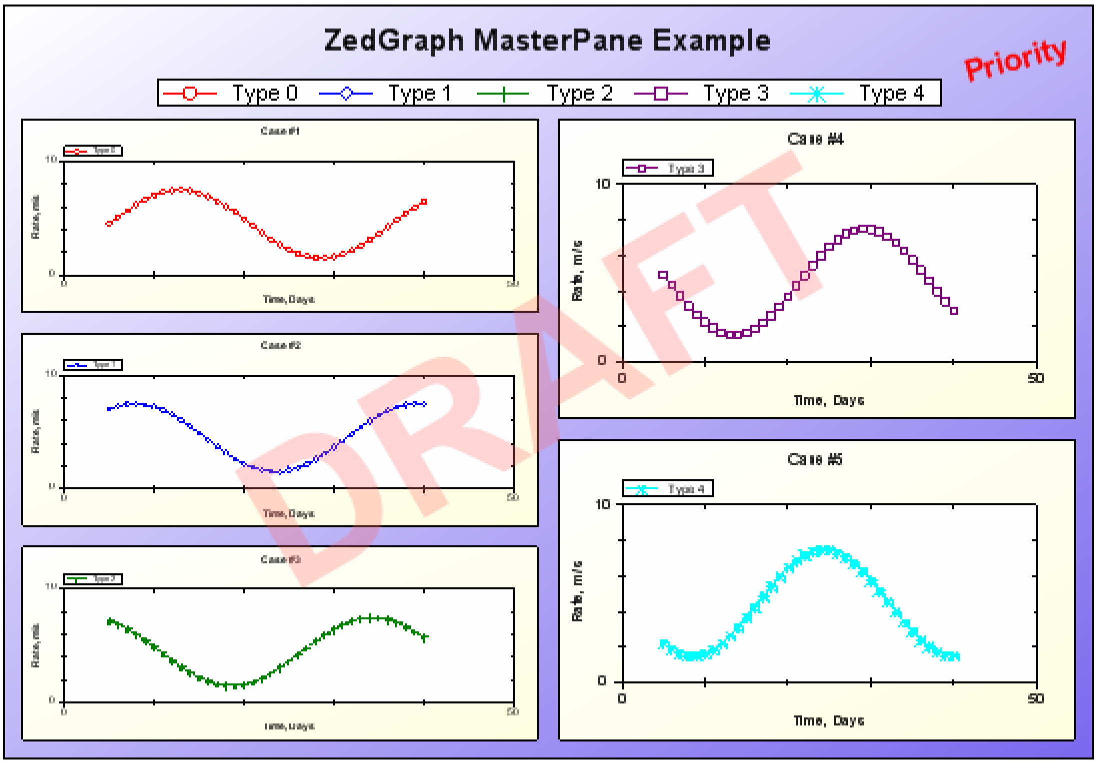
Although it is technically possible to combine pie charts with line graphs on the same GraphPane, it is not recommended. If a particular GraphPane contains only PieItem objects, then the AxisChange() method will automatically make the axes invisible by setting the Axis.IsVisible property to false.

The following is an example of a ZedGraph pie chart:



The MasterPane

The MasterPane is designed to help facilitate the handling of multiple GraphPane objects on a page. This is done by maintaining a list of GraphPane objects, and providing utility functions for layout, rendering, mouse point location, etc. See the ZedGraphWiki for details on the MasterPane class. The following is an example of a chart based on the MasterPane class:



Pan/Zoom functions

The ZedGraphControl class now provides some interactive functionality for zooming and panning. To zoom, left click inside the AxisRect area, and drag out a new rectangle to indicate the scale region into which to zoom. To pan, either click with the middle mouse button or hold down the shift key, and left click to drag the graph around (the zoom/pan key combinations are user-modifiable as well). You can also add scrollbars to the control with IsShowHScrollBar and IsShowVScrollBar. The mouse-wheel can also be used for zooming. There is also a context menu that allows you to un-zoom and un-pan to prior states, to restore the scale to full auto mode, to show point value tool tips, and to copy the graph (bitmap form) to the clipboard. Event options are provided to allow for customization of pan, zoom, point values, etc.

The Fill class

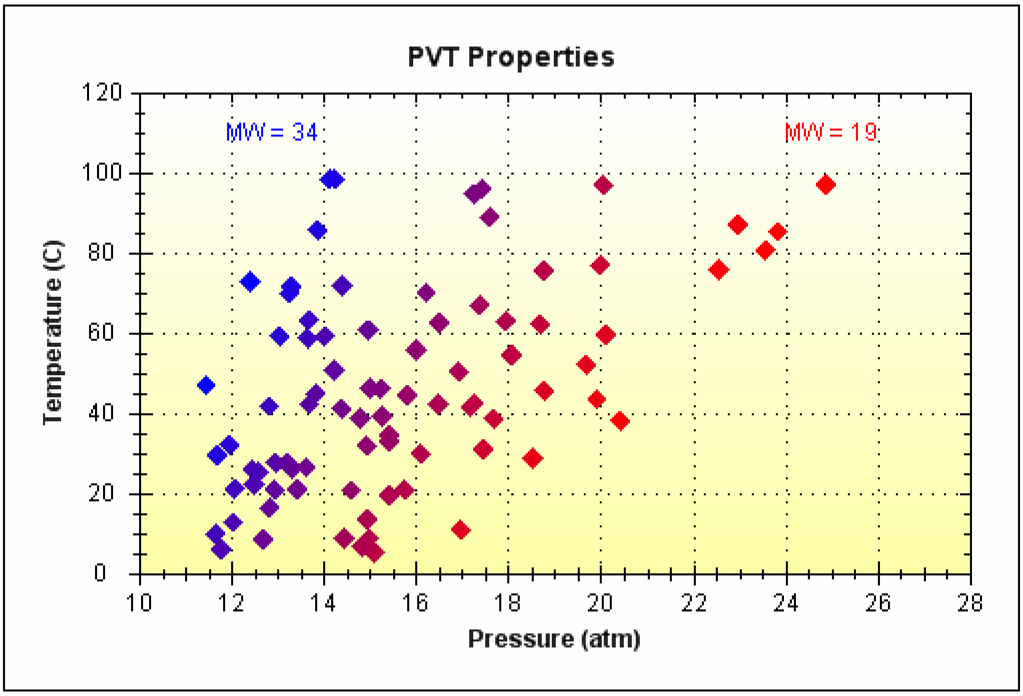
The Fill class is an important addition to ZedGraph, so it deserves a special mention. This class handles solid, linear gradient, and texture fills for the pane background, the axis background, the legend background, all text backgrounds, the symbol fill, and for filling the area under a line. Each of these classes will have one or more Fill classes to control the fill properties (e.g., CurveItem.Line.Fill controls the filling of the area under the curve). The Fill class has a number of constructors to make it flexible and easy to use:

Fill(Color)	Makes a solid color fill with the specified color.
Fill(Color, Color)	Makes a linear gradient fill from color1 to color2, with a gradient angle of 0 degrees.
Fill(Color, Color, float angle)	Makes a linear gradient fill from color1 to color2, with a gradient angle as specified.
Fill(Image, WrapMode)	Specifies an image to be used for filling.
Fill(Brush)	Uses the specified brush directly, scaling the brush to fill the destination object bounding box.
Fill(Brush, bool isScaled)	Uses the specified brush, allowing you to disable the scaling.
Fill(Brush, AlignH, AlignV)	Uses the specified brush, with no scaling, but the source brush alignment is specified.

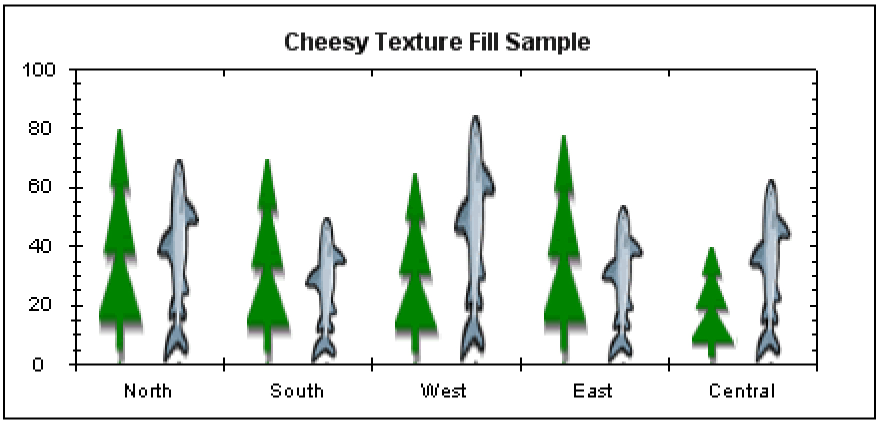
Internally, the Fill class keeps a FillType enumeration (the Type property) to determine what type of fill is used, as follows:

- FillType.None: No filling will be done - the background will be transparent.
- FillType.Solid: A solid color fill will be done using the SolidBrush class with the Fill.Color value.
- FillType.Brush: A fancy fill will be done using the user-supplied brush specified in Fill.Brush. If Fill.Brush is null, then a LinearGradientBrush will be automatically created using Fill.Color and Color.White as the gradient colors, with a gradient angle of zero degrees.
- FillType.GradientByX: This mode is intended to be used with Symbol.Fill. A linear gradient is maintained internally in the Fill class, and each symbol point is filled with a solid color that is taken from the internal gradient based on the X data value for that point. Essentially, you get a scatter plot in which each data point is colored according to its X value. The data ranges for color assignment are user assigned with the Fill.RangeMin and Fill.RangeMax properties.
- FillType.GradientByY: This is the same as GradientByX except that the Y data values are used instead of X data values.
- FillType.GradientByZ: This is the same as GradientByX except that the Z data values are used instead of X data values.
- FillType.GradientByColorValue: This is the same as GradientByX except that the 'ColorValue' property values are used instead of X data values.

The following graph is an example of FillType.GradientByZ usage, in which seemingly scattered data gains visual coherency using the color attribute:



The following shows an example of using an image to fill in the bars of a bar chart:



Z-Order

The graphic items in the display are contained in two lists; GraphPane.CurveList and GraphPane.GraphObjList. CurveList contains all of the curves, including bars, lines, etc. GraphObjList contains text items, images, shapes, arrows, etc. In both lists, the Z-Order is controlled by the order of the objects in the list; the first objects in the list appear in front of the later objects in the list. You can modify the order of any object relative to other objects in the same collection (list), with the Move() method in the collection class. Further, the GraphObj class has a ZOrder property that controls the depth of each individual GraphObj relative to other, non-GraphObj objects. The ZOrder is an enumeration type with the following values:

ZOrder	Description
ZOrder.A_InFront	The topmost depth, in front of all other objects.
ZOrder.B_BehindLegend	Drawn behind the Legend object.
ZOrder.C_BehindChartBorder	Drawn behind the Chart frame border.
ZOrder.D_BehindAxis	Drawn behind the Axis objects (behind the scale labels, etc.).
ZOrder.E_BehindCurves	Drawn behind all the CurveItem objects.
ZOrder.F_BehindTitle	Drawn behind the GraphPane title.
ZOrder.G_BehindAll	Drawn behind the Chart rectangle fill (but still in front of the pane rectangle fill).

Utility methods

ZedGraph is being used as a class library in interactive parent applications that need information about the graph. The following are some utility methods that are worth mentioning (these are documented in the *ZedGraph.chm* doc file):

- FindPane() is a method in the MasterPane class that, given a mouse point location, returns the GraphPane object under the mouse.
- FindAxisRect() is a method in the MasterPane class that, given a mouse point location, returns the GraphPane object for the AxisRect in which the mouse point lies.
- FindNearestObject() is a method in the GraphPane class that, given a mouse point location, returns the closest object and a corresponding index number for the object. This method may return Curve points, the GraphPane, GraphItems, axes, the legend, etc., depending on what was clicked.
- FindNearestPoint() is a method in the GraphPane class that, given a mouse point location, returns the closest CurveItem and the index number of the closest point within that CurveItem. This routine will only consider points that are within Def.Pane.NearestTol pixels of the specified mouse point location. The default tolerance is 7 pixels.
- ReverseTransform() is a method in the GraphPane class that, given a mouse point location, returns the X, Y, and Y2 axis values that correspond to that location.

- GeneralTransform() is a method in the GraphPane class that, given an (X,Y) value pair, returns the corresponding (X,Y) screen pixel coordinates. The (X,Y) value pair can be expressed in a variety of coordinate systems, as follows:
    - CoordType.AxisXYScale: the coordinate values are from the X and left Y axes.
    - CoordType.AxisXY2Scale: the coordinate values are from the X and right Y (Y2) axes.
    - CoordType.ChartFraction: the coordinate values are expressed as a fraction of the total Chart.Rect width and height (0.5 is the center of the Chart.Rect, 1.1 is just to the right or just above the AxisRect).
    - CoordType.PaneFraction: the coordinate values are expressed as a fraction of the total PaneRect width and height. The value should be from 0.0 to 1.0, since values outside this range are clipped.
    - XChartFractionYPaneFraction: the X coordinate is a fraction of the total Chart.Rect, and the Y coordinate is a fraction of the total Pane.Rect.
    - XPaneFractionYChartFraction: the X coordinate is a fraction of the total Pane.Rect, and the Y coordinate is a fraction of the total Chart.Rect.
    - XScaleYChartFraction: the X coordinate is an X scale value, and the Y coordinate is fraction of the total Chart.Rect.
    - XChartFractionYScale: the X coordinate is a fraction of the total Chart.Rect, and the Y coordinate is a Y scale value.
    - XChartFractionY2Scale: the X coordinate is a fraction of the total Chart.Rect, and the Y coordinate is a Y2 scale value.

- CalcChartRect() is a method in the GraphPane class that will calculate the chart rectangle (screen coordinates), given the current configuration. If, for some reason, you need to manually set the chart rectangle (to make it line up perfectly with other items, etc.), you can use this method to get the default rectangle, modify it as desired, then set GraphPane.Chart.Rect = yourRect. Changing the Chart.Rect will automatically set GraphPane.Chart.IsRectAuto to false so that the Chart.Rect will remain as you set it.

Acknowledgements

I would like to thank all those who have tested the code, reported issues, and contributed code to the ZedGraph project. I would especially like to acknowledge the contributions of the volunteer developers who made

ZedGraph possible: Jerry R. Vos, Bob Kaye, Darren Martz, and Benjamin Mayrargue.

Latest update

- See the Revision History page for the latest updates.
- Also, check out the Main Wiki Page for the latest news.

License

About the Author

JChampion

Occupation: Engineer

Location: United States

Other popular General Graphics articles:

- CxImage

  CxImage is a C++ class to load, save, display, transform BMP, JPEG, GIF, PNG, TIFF, MNG, ICO, PCX, TGA, WMF, WBMP, JBG, J2K images.

- 3D Pie Chart

  A class library for drawing 3D pie charts.

- Really cool visual FX

  A set of classes for doing stunning visual effects, including water, plasma and fire.

- ImageStone

  An article on a library for image manipulation.

- Gios PDF .NET library

  A .NET library for generating impressive PDF reports.

Add this article to:

Article Top	***Rate this Article for us!***	*Poor*	○ ○ ○ ○ ○	*Excellent* Vote

FAQ	Noise Tolerance Medium ▾	Search comments	Set Options
	Layout Normal ▾	Per page 25 ▾	

New Message Msgs 1 to 25 of 2,676 (Total in Forum: 2,676) (Refresh) FirstPrev Next

Subject	Author	Date
Professional web chart for ASP.NET NEW	Florentin BADEA	6hrs 16mins ago

If you are looking for a professional asp.net chart[^] control, I recommend ExpertChart. It's easy to use, unexpensive and creates absolutely marvelous charts.

ExpertComponents WebChart[^]

Features[^]

Download[^]

Live demo[^]

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

Using ZedGraph from PowerShell	BradleyC	20:00 19 Mar '08

I have ported one of the C# samples from the ZedGraph Wiki to PowerShell to see if it could be easily done. Turns out it's not too bad. I updated the wiki sample page with PowerShell source code here:
http://zedgraph.org/wiki/index.php?title=BaseTic_Demo[^]

or see my blog here:

http://bloggerbc.blogspot.com/2008/03/using-zedgraph-from-powershell.html[^]

Have fun!

---

bc

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

labels in a log 2D plot   TopoGigio   6:56 18 Mar '08

Hello,
I need to make a plot using the log scale on the x axis, with x values between, let's say, 200 and 8000. I do this with the AxisType.Log command, but the only label visible on the x axis in this case is 1000. If I expand the interval from 100 to 10000, the labels visible on the x axis become 100, 1000 and 10000. In other words the plot shows only the power of 10 labels.
The problem is that I need to show also other values inbetween like 200, 500, 1000, 2000, 4000 and 6000 and not just 100, 1000, 10000 (always keeping the log scale on the x axis).
Would this be possible?

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

How to display different coloured bars   mik_nomad   9:01 16 Mar '08

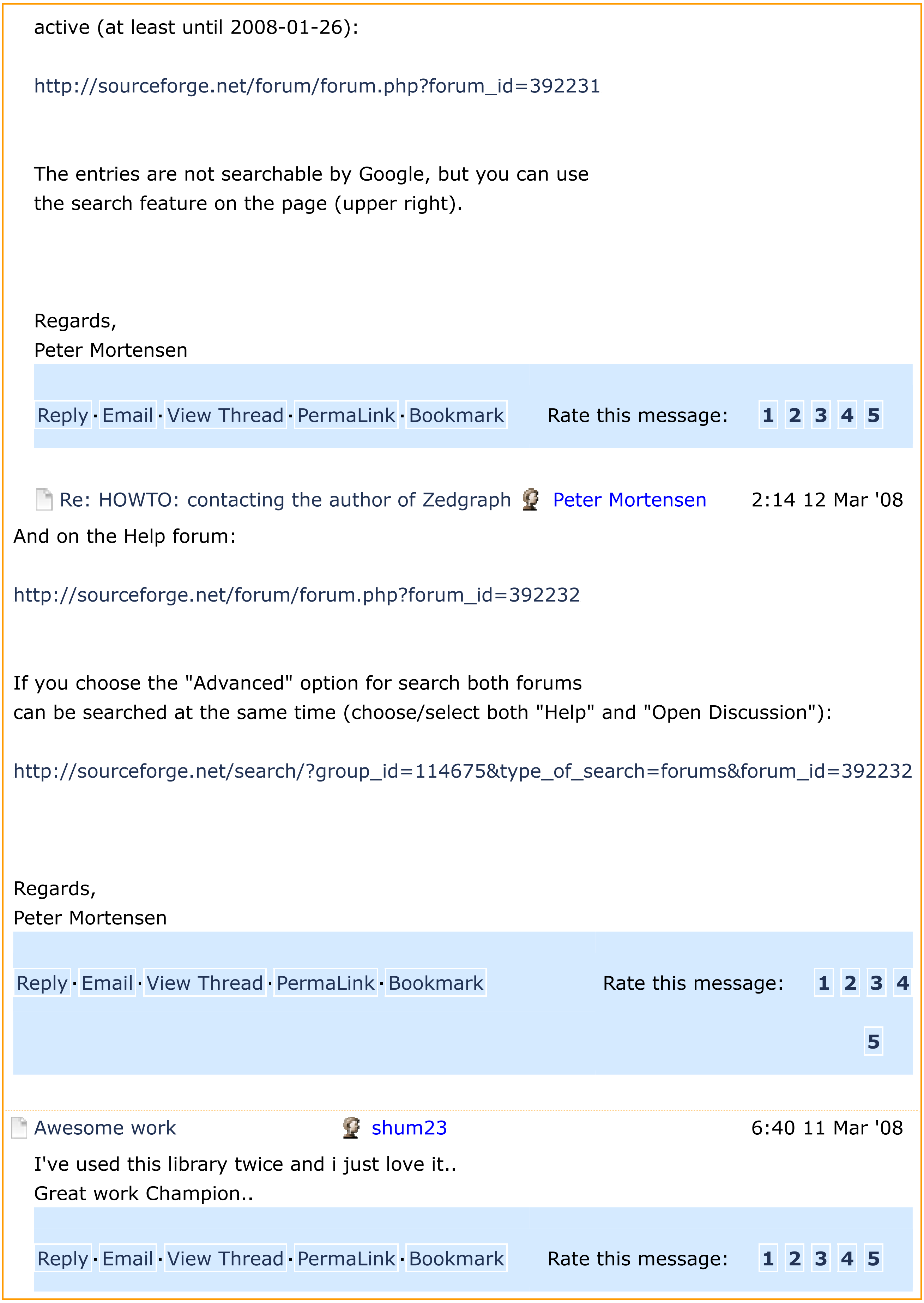
Hi, I am making a survey tool and we need to show the results in different coloured bar graphs for each option. Right now, all of them are coming in one colour.
Can anybody point out what i am missing...

Regards,
Mikhail

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

HOWTO: contacting the author of Zedgraph   Peter Mortensen   2:01 12 Mar '08

The open discussion forum on SourceForge appears to be the
place where the author of Zedgraph, John Champion, is most



active (at least until 2008-01-26):

http://sourceforge.net/forum/forum.php?forum_id=392231

The entries are not searchable by Google, but you can use the search feature on the page (upper right).

Regards,
Peter Mortensen

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

Re: HOWTO: contacting the author of Zedgraph   Peter Mortensen   2:14 12 Mar '08

And on the Help forum:

http://sourceforge.net/forum/forum.php?forum_id=392232

If you choose the "Advanced" option for search both forums
can be searched at the same time (choose/select both "Help" and "Open Discussion"):

http://sourceforge.net/search/?group_id=114675&type_of_search=forums&forum_id=392232

Regards,
Peter Mortensen

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

Awesome work   shum23   6:40 11 Mar '08

I've used this library twice and i just love it..
Great work Champion..

Reply · Email · View Thread · PermaLink · Bookmark   Rate this message: 1 2 3 4 5

Image Resize [modified] TeachesOfPeaches 3:13 10 Mar '08

Hi,
I'm displaying a horizontal bar graph in a web form. My problem is that when the numbers on the XAxis become bigger (there are for example values of several thousands) the last digit hangs over the Xaxis causing the whole image to shrink. Is there a possibility to drop the last digit or to leave more space at the right side of the graph?

Many thanks for any kind of help.

Andreas

modified on Monday, March 10, 2008 10:06 AM

Reply · Email · View Thread · PermaLink · Bookmark
2.00/5 (1 vote) Rate this message:
1 2 3 4 5

Org Chart? Joeblast 11:32 6 Mar '08

Could I use this to draw an Organizational Chart?

Reply · Email · View Thread · PermaLink · Bookmark
Rate this message: 1 2 3 4 5

Interacting in real time brunomas 8:08 6 Mar '08

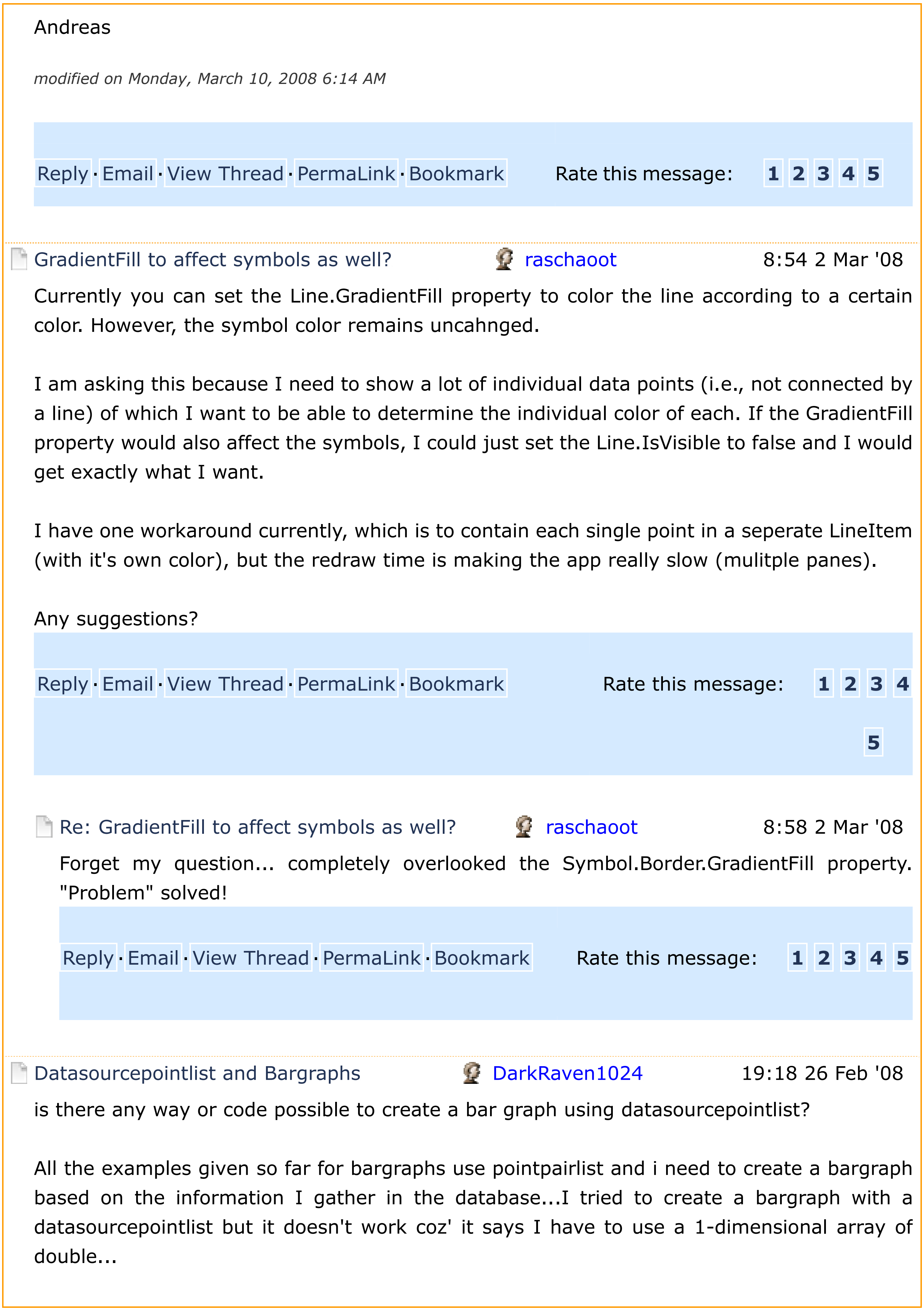
Hi. I would like to know if is it possible to draw lines(by user) on the graphic in real-time.
Thanks

Reply · Email · View Thread · PermaLink · Bookmark
1.00/5 (1 vote) Rate this message: 1 2 3 4 5

Taransparent background [modified] TeachesOfPeaches 7:29 5 Mar '08

Hi,
I would like to create a pie chart with transparent background in my web form.
Setting the MasterPaneFill.Color, GraphPane.Color and ChartFill.Color to transparent results only in a white background.

Many thanks for any kind of help



Andreas

modified on Monday, March 10, 2008 6:14 AM

Reply · Email · View Thread · PermaLink · Bookmark Rate this message: 1 2 3 4 5

GradientFill to affect symbols as well? raschaoot 8:54 2 Mar '08

Currently you can set the Line.GradientFill property to color the line according to a certain color. However, the symbol color remains uncahnged.

I am asking this because I need to show a lot of individual data points (i.e., not connected by a line) of which I want to be able to determine the individual color of each. If the GradientFill property would also affect the symbols, I could just set the Line.IsVisible to false and I would get exactly what I want.

I have one workaround currently, which is to contain each single point in a seperate LineItem (with it's own color), but the redraw time is making the app really slow (mulitple panes).

Any suggestions?

Reply · Email · View Thread · PermaLink · Bookmark Rate this message: 1 2 3 4 5

Re: GradientFill to affect symbols as well? raschaoot 8:58 2 Mar '08

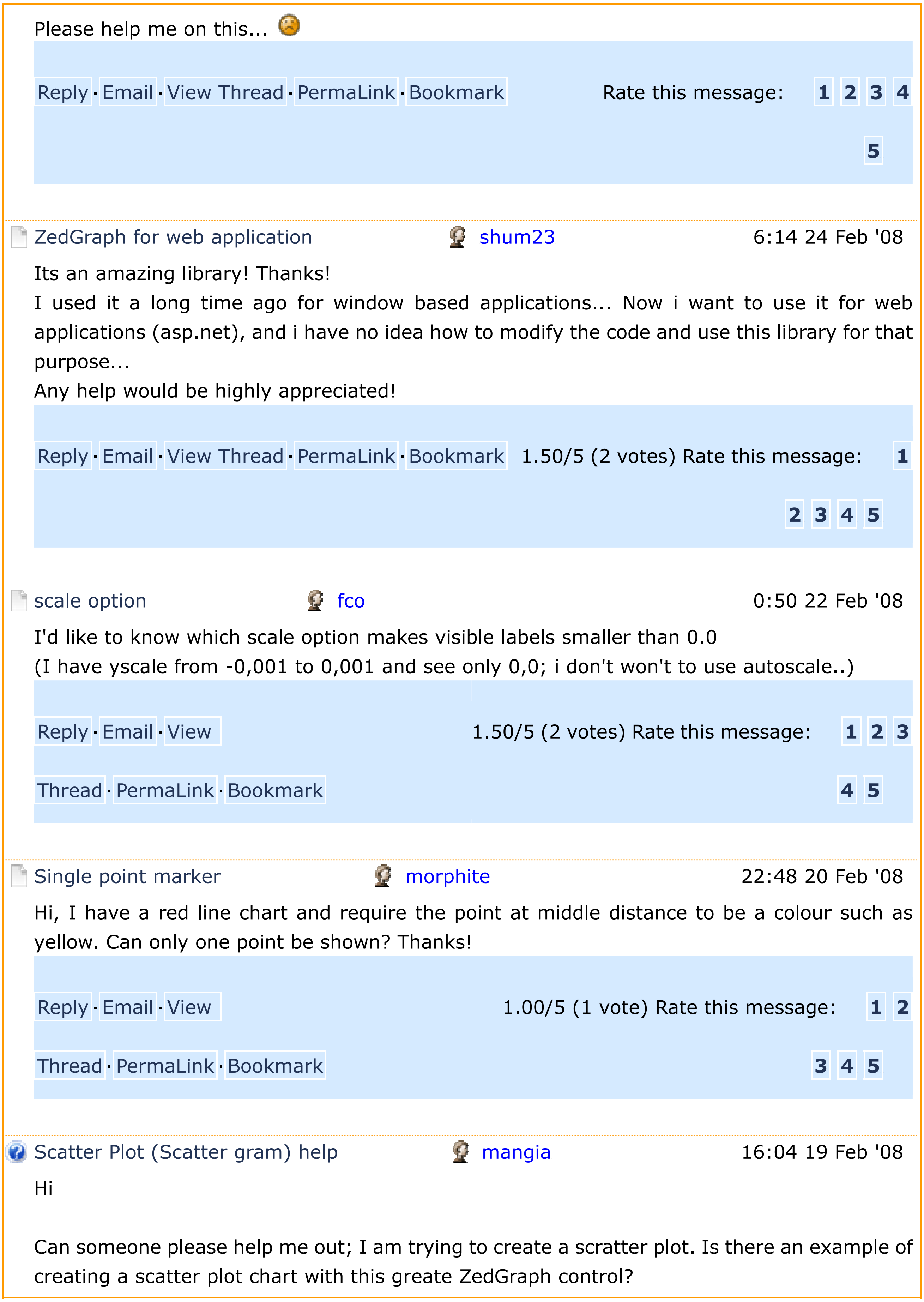
Forget my question... completely overlooked the Symbol.Border.GradientFill property. "Problem" solved!

Reply · Email · View Thread · PermaLink · Bookmark Rate this message: 1 2 3 4 5

Datasourcepointlist and Bargraphs DarkRaven1024 19:18 26 Feb '08

is there any way or code possible to create a bar graph using datasourcepointlist?

All the examples given so far for bargraphs use pointpairlist and i need to create a bargraph based on the information I gather in the database...I tried to create a bargraph with a datasourcepointlist but it doesn't work coz' it says I have to use a 1-dimensional array of double...



Please help me on this... ☹

Reply · Email · View Thread · PermaLink · Bookmark Rate this message: 1 2 3 4 5

ZedGraph for web application shum23 6:14 24 Feb '08

Its an amazing library! Thanks!

I used it a long time ago for window based applications... Now i want to use it for web applications (asp.net), and i have no idea how to modify the code and use this library for that purpose...

Any help would be highly appreciated!

Reply · Email · View Thread · PermaLink · Bookmark 1.50/5 (2 votes) Rate this message: 1 2 3 4 5

scale option fco 0:50 22 Feb '08

I'd like to know which scale option makes visible labels smaller than 0.0

(I have yscale from -0,001 to 0,001 and see only 0,0; i don't won't to use autoscale..)

Reply · Email · View Thread · PermaLink · Bookmark 1.50/5 (2 votes) Rate this message: 1 2 3 4 5

Single point marker morphite 22:48 20 Feb '08

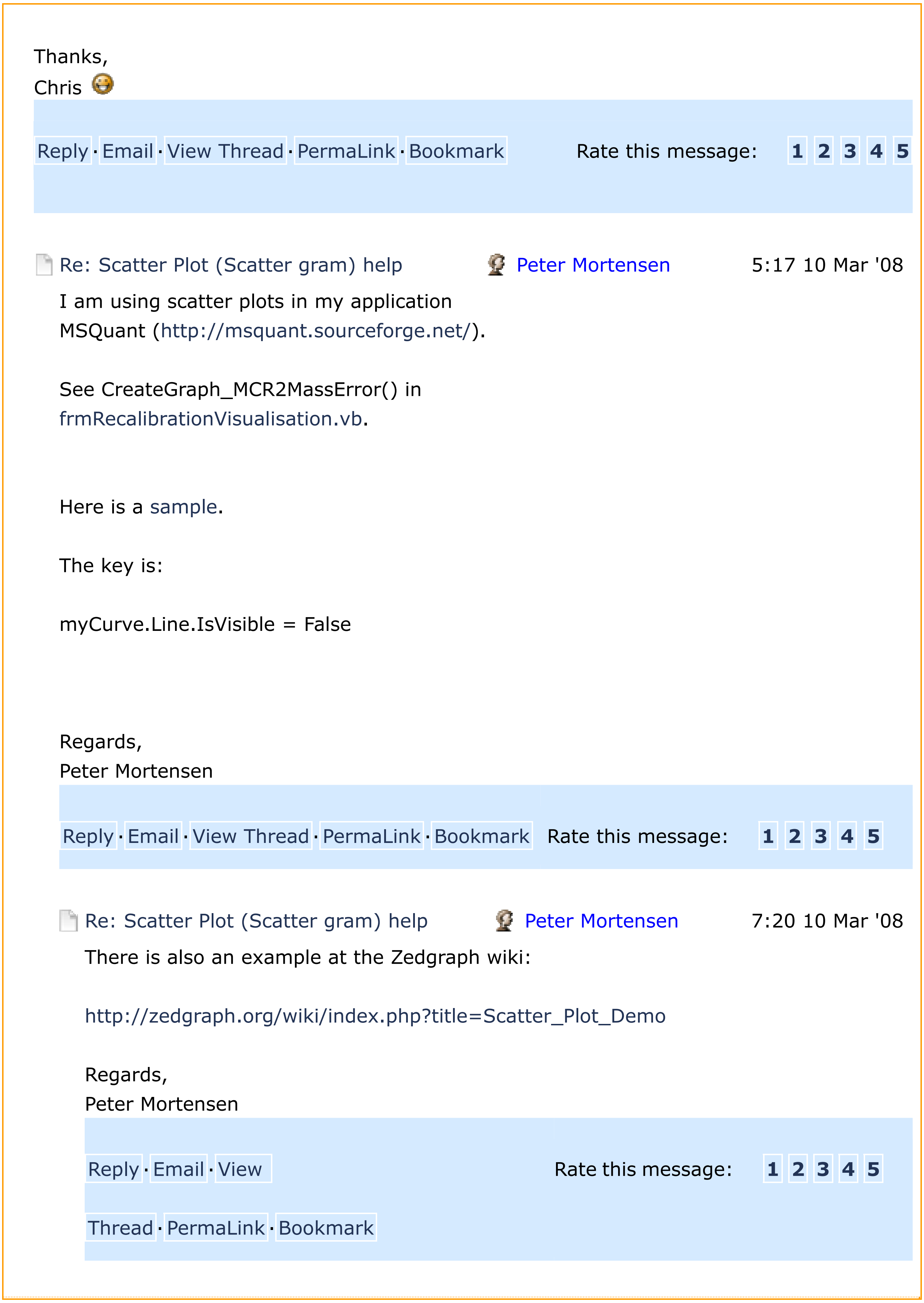
Hi, I have a red line chart and require the point at middle distance to be a colour such as yellow. Can only one point be shown? Thanks!

Reply · Email · View Thread · PermaLink · Bookmark 1.00/5 (1 vote) Rate this message: 1 2 3 4 5

Scatter Plot (Scatter gram) help mangia 16:04 19 Feb '08

Hi

Can someone please help me out; I am trying to create a scratter plot. Is there an example of creating a scatter plot chart with this greate ZedGraph control?



Thanks,
Chris

Reply · Email · View Thread · PermaLink · Bookmark
Rate this message: 1 2 3 4 5

Re: Scatter Plot (Scatter gram) help   Peter Mortensen   5:17 10 Mar '08

I am using scatter plots in my application MSQuant (http://msquant.sourceforge.net/).

See CreateGraph_MCR2MassError() in frmRecalibrationVisualisation.vb.

Here is a sample.

The key is:

myCurve.Line.IsVisible = False

Regards,
Peter Mortensen

Reply · Email · View Thread · PermaLink · Bookmark
Rate this message: 1 2 3 4 5

Re: Scatter Plot (Scatter gram) help   Peter Mortensen   7:20 10 Mar '08

There is also an example at the Zedgraph wiki:

http://zedgraph.org/wiki/index.php?title=Scatter_Plot_Demo

Regards,
Peter Mortensen

Text wrapping axis labels S Batham 7:45 18 Feb '08

I'm using the graph to show average scores (based on a scale of 1 to 5) in response to certain statements. The statements themselves as being used as the axis labels down the left hand side with horizontal bars displaying the results.

In some cases, the labels are so long, the graph itself is squashed up to the right hand side and not easily readable.

Is there a way to text wrap the labels, or restrict the space used by the axis labels, so the graph will display using, say, no more than 30% of the total width for the labels, leaving 70% of the total width for the graph itself?

Cheers

Simon.

Reply · Email · View Thread · PermaLink · Bookmark 3.50/5 (2 votes) Rate this message: 1 2 3 4 5

Save Zedgraph as Byte[] ntuyen01 9:38 13 Feb '08

Is there a way for me to save the zedgraph as byte[]
Thanks and Regards,

Thank You
Regards,
ntuyen01

Reply · Email · View Thread · PermaLink · Bookmark 4.20/5 (2 votes) Rate this message: 1 2 3 4 5

Bar Graph loganj1999 9:21 11 Feb '08

One you use the AddBar for the graph pane. Is there any way to clear that bar? I am using a dynamic application, where I need to create and delete bars depending on the button that is checked. Have been looking around, but I don't see any way to basically delete a bar.

Reply · Email · View Thread · PermaLink · Bookmark 1.00/5 (1 vote) Rate this message: 1 2 3 4 5

Very good graphing library... Fuzzychaos 1:48 11 Feb '08

Thank you for such a good piece of software. I'm using it in C# Express and it works great, as expected.

Jeremy

Props to the family: New Dawn Engineering

Reply · Email · View Thread · PermaLink · Bookmark 2.00/5 (1 vote) Rate this message: 1 2 3 4 5

What an incredible program! Thank you John Champion! Member 371910 19:01 9 Feb '08

I bow down to you. Great work.

Reply · Email · View Thread · PermaLink · Bookmark Rate this message: 1 2 3 4 5

Last Visit: Thursday, March 20, 2008 6:07 PM Last Update:Friday, March 21, 2008 9:16 PM **1** 2 3 4 5 6 7 8 9 10 Next »

General News Question Answer Joke Admin

PermaLink | Privacy | Terms of Use

Last Updated: 6 Jun 2007

Editor: Sean Ewington

Web07 | Advertise on the Code Project

Display Custom Tooltips

From ZedGraphWiki

Jump to: navigation, search

Contents

Display Custom Tooltips

ZedGraph can display tooltips as the mouse cursor hovers over each point or bar on the graph. Note that this tooltip capability is different from the "Link" class, which allows "Drill Down" functionality (see the Use an ImageMap page).

To enable the tooltip display, you first need to set the ZedGraphControl.IsShowPointValues = true. This value can be set within the program code or it can be set from the context menu. There are three different "modes" of operation for the tooltips:

Default Mode

In this mode, the tooltips are displayed using a default format for the X and Y values. The actual numeric value can be controlled with the ZedGraphControl.PointValueFormat property and/or the ZedGraphControl.PointDateFormat property. This properties are string values in the form of NumberFormatInfo and DateTimeFormatInfo.

Tag Mode

In this mode, the tooltip is made up of a string that is stored in the PointPair.Tag property for each point. If any PointPair.Tag value is non-null, and is of type "string", then this string will replace the default tooltip text. It can be of any format.

Here’s a basic example how to use PointPair.Tag

```
   public void createGraph()
        {
            double[] pointsY = new double[5];
            double[] pointsX = new double[5];
            String toolhint = "";
            PointPairList PPL = new PointPairList();
            GraphPane myPane = zedGraphControl1.GraphPane;
            for (int i = 0; i < 5; i++)
            {
                //Change X and Y points for the values that you want
                pointsY[i] = (double)(i*i);
                pointsX[i] = (double)i;
                toolhint = String.Format("Put your hint to Point [{0},{1}]", i,
i*i);
                PointPair pp = new PointPair(pointsX[i], pointsY[i], toolhint);
                PPL.Add(pp);
            }

            zedGraphControl1.IsShowPointValues = true;
            // Fill the axis background with a color gradient
            myPane.Chart.Fill = new Fill(Color.FromArgb(255, 255, 245),
Color.FromArgb(255, 255, 190), 90F);

            LineItem myCurve = myPane.AddCurve("Name of Graphic", PPL, Color.Red);
            myCurve.Symbol.Fill = new Fill(Color.PowderBlue);

            // Manually set the x axis range
            myPane.XAxis.Scale.Min = 0;
            myPane.XAxis.Scale.Max = 5;
            myPane.YAxis.Scale.Max = 25;
            // Display the Y axis grid lines
            myPane.YAxis.MajorGrid.IsVisible = true;
            myPane.YAxis.MinorGrid.IsVisible = true;

            // Calculate the Axis Scale Ranges
              zedGraphControl1.AxisChange();
       }
```

PointValueEvent

In this case, you can subscribe to the ZedGraphControl.PointValueEvent to provide a custom tooltip formatter. This allows you to provide customized tooltips for each point which are prepared "on the fly" as the mouse hovers over any given point. To make a custom formatter, first add a ZedGraphControl.PointValueHandler to your Form_Load method as follows:

```
zedGraphControl1.PointValueEvent += new
ZedGraphControl.PointValueHandler( MyPointValueHandler );
```

Also, add a method like this to your Form class:

```
private string MyPointValueHandler( object sender, GraphPane pane, CurveItem curve,
int iPt )
{
   PointPair pt = curve[iPt];
   return "This value is " + pt.Y.ToString("f2") + " gallons";
}
```

Retrieved from "http://zedgraph.org/wiki/index.php?title=Display_Custom_Tooltips"

Vary the colors of the points or bars on a single CurveItem

From ZedGraphWiki

Jump to: navigation, search

ZedGraph can use different colors for the individual points or bars of a single CurveItem instance. This is accomplished using the Fill class with FillType.GradientByX, FillType.GradientByY, or FillType.GradientByZ. Each of these types will apply a color in the fill according to the X, Y, or Z data value, respectively. In the rest of this article, we will use GradientByZ to represent any of the three types.

To use this FillType, you first define a normal linear gradient fill, like this:

```
Fill myFill = new Fill( Color.Blue, Color.Red );
myFill.Type = FillType.GradientByZ;
```

which defines a continuous linear color gradient the runs from blue to red (substitute any colors you like here). Also, the Fill is defined with the FillType.GradientByZ.

Next, you need to define the range of values that span the range of colors. The following defines the ranges of values to be from 1.0 to 2.0 (e.g., PointPair.Z), with a default value of 1.0:

```
myFill.RangeMin = 1;
myFill.RangeMax = 2;
myFill.RangeDefault = 1;
```

Depending on how you define your Z data values, you can use discrete colors or continuous colors. For example, if the Z value is 1.0, you will get a color of blue. A value of 2.0 will yield a color of red. If all Z values are either 1.0 or 2.0, then all points will be either red or blue. On the other hand, if you have a Z value of 1.5 then you will get a color that is a mix of blue and red.

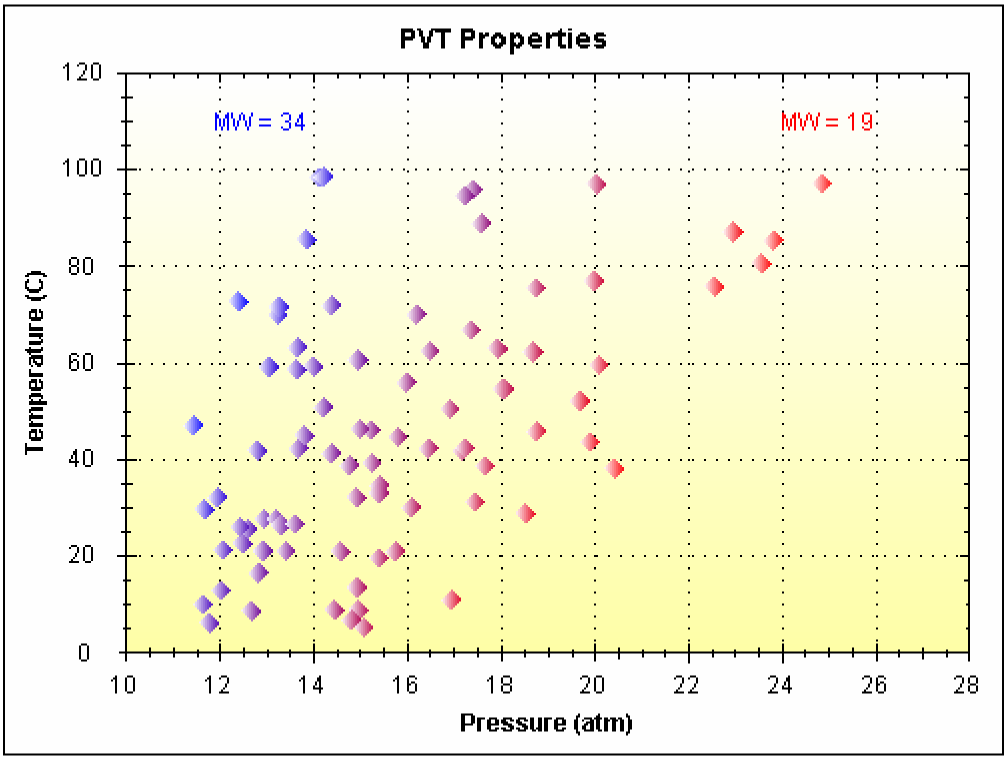
Finally, you need to define a secondary color if desired. This is the color that will be used in combination with the color above (as determined by the Z value), to make a gradient for filling the data point or bar. Typically, this will be Color.White or Color.Empty.

```
myFill.SecondaryValueGradientColor = Color.White;
```

As an example, assume the Z value is 2.0 and the SecondaryValueGradientColor is Color.White. The result would generate a gradient fill from white to red on the specified point or bar.

Note that 'myFill' needs to be assigned to a CurveItem in order for it to have an effect

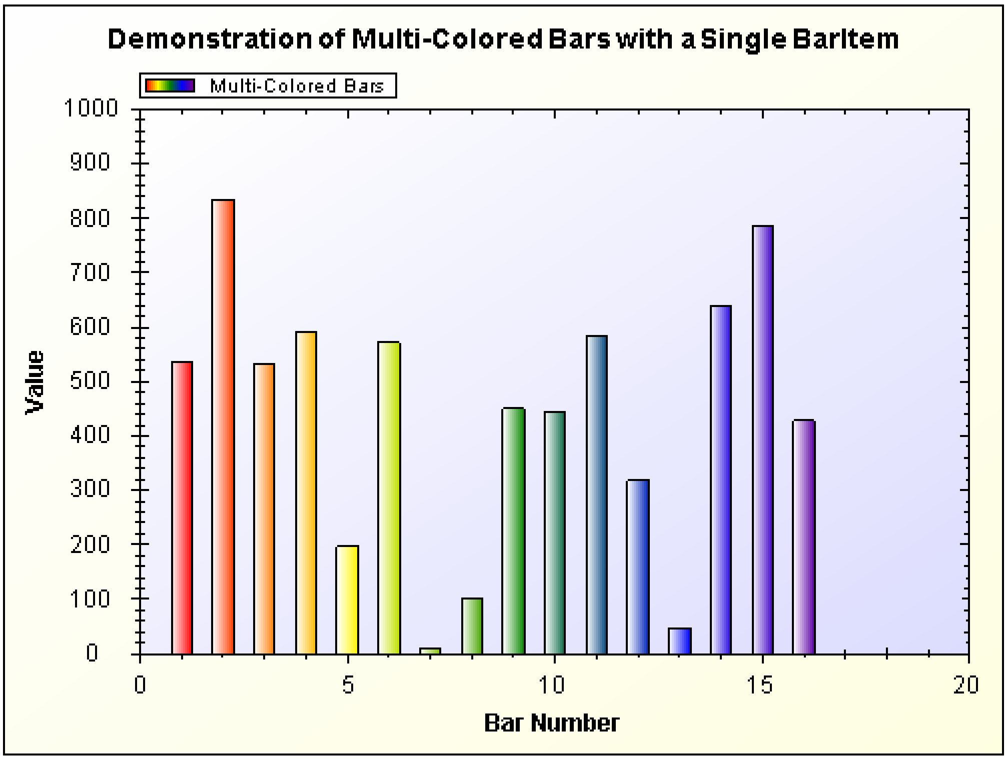
There is an example Gradient-By-Value_Demo showing coloring of data points in the sample graphs section, which looks like this:



More complex examples of gradient-by-value fills are possible. The following is an example of a five-color gradient-by-value fill:

```
Color[] colors = { Color.Red, Color.Yellow, Color.Green, Color.Blue,
Color.Purple };
myFill = new Fill( colors );
myFill.Type = FillType.GradientByZ;
myFill.RangeMin = 1;
myFill.RangeMax = 5;
myFill.RangeDefault = 1;
myFill.SecondaryValueGradientColor = Color.White;
```

The following graph is a full example using the above fill type. The code for this example can be found in the Sample Graph Section.



Retrieved from "http://zedgraph.org/wiki/index.php?title=Vary_the_colors_of_the_points_or_bars_on_a_single_CurveItem"

Customize the scale format

From ZedGraphWiki

Jump to: navigation, search

There are a variety of ways to use custom formats for the scale labels (the value labels on the major tics) on a ZedGraph chart. This article presents some options for custom formatting.

Using Text Labels

One of the simplest but most limited methods is to provide an array of custom text labels. These labels are simple text strings of any type. For example, they might be "First", "Second", "abc", "123", "xyz". In this case, there will be five labels with text exactly matching the previous strings. Note that using the text labels requires AxisType.Text, which is an ordinal type only.

You can view source code for a text axis in the Tutorial:Text Axis Chart Demo.

Setting a custom format template

For non-ordinal axis types, such as AxisType.Linear or AxisType.Date, the scale labels are formatted using a format string. If Axis.Scale.FormatAuto is true, this format string well be automatically set by ZedGraph to accommodate the actual data.

For example, for an AxisType.Date chart, if the range of the data span less than ten minutes, then the Scale.Format property will be set to the string "HH:mm", which will show values of hours and minutes. On the other hand, if the range of the data span less than ten days, the format property will be set to "d-MMM", which will show values in day-month format such as "5-Jan". For AxisType.Linear, the default format is always "g", which gives a reasonable format for most numeric values.

You can manually set this format to suit your own needs, overriding the ZedGraph defaults. Note that modifying this value will automatically set Axis.Scale.FormatAuto to false. You can re-enable the ZedGraph default settings by changing this value back to true.

There are a wide range of custom formats available, which are specified as DateTimeFormatInfo for date formats, and NumberFormatInfo for numeric formats.

These format strings allow just about any custom formatting, including literal strings. Some examples:

Format String	Axis Type	Sample Result
"f2"	AxisType.Linear	34.86
"c"	AxisType.Linear	$34.86
"c0"	AxisType.Linear	$35

"e"	AxisType.Linear	3.486000e+001
"e2"	AxisType.Linear	3.49e+001
"0.0'%'"	AxisType.Linear	34.9%
"d"	AxisType.Date	1/1/2005
"D"	AxisType.Date	Saturday, January 01, 2005
"f"	AxisType.Date	Saturday, January 01, 2005 12:00 AM
"g"	AxisType.Date	1/1/2005 12:00 AM
"G"	AxisType.Date	1/1/2005 12:00:00 AM
"r"	AxisType.Date	Sat, 01 Jan 2005 00:00:00 GMT
"t"	AxisType.Date	12:00 AM
"T"	AxisType.Date	12:00:00 AM
"r"	AxisType.Date	Sat, 01 Jan 2005 00:00:00 GMT
"y"	AxisType.Date	January, 2005
"dd-MMM"	AxisType.Date	01-Jan
"dd-MMM-yyyy"	AxisType.Date	01-Jan-2005
"h tt"	AxisType.Date	2 PM
"HH:mm"	AxisType.Date	14:32
"HH:mm:ss.fff"	AxisType.Date	14:32:24.751

Fully custom labels using the ScaleFormatEvent

A third option for scale label formatting is the Axis.ScaleFormatEvent. This option allows you to manually format each scale label individually. In effect, ZedGraph will ask your ScaleFormatEvent handler method to provide a label (formatted as a string) for each major tic as the chart is being drawn. In this case, you can use any format you like, and each label can be a different format if desired.

To utilize this option, you first need to subscribe to the AxisScaleFormatEvent. You can do this by adding the following line to your graph creation method (this assumes the name of your ZedGraphControl is 'z1'):

```
z1.GraphPane.XAxis.ScaleFormatEvent += new
Axis.ScaleFormatHandler( XScaleFormatEvent );
```

Next, provide the handler method that was specified in the code above (XScaleFormatEvent):

```
public string XScaleFormatEvent( GraphPane pane, Axis axis, double val, int index )
{
   return val.ToString("f2") + "cm";
}
```

This method just returns a string that represents the formatted version of the specified data value. In this case, the number is formatted using "f2", which is fixed format with 2 decimal places, and a " cm" string is added to the end. Of course, you can do any processing you like, you just have to return the resultant formatted value as a string.

Note that you can actually use the ScaleFormatEvent to completely change the scale of the graph. For example, if the scale of a linear graph runs from 0 to 100, you can modify the scale labels so it looks like the scale actually runs from 50 to 150 with the following method:

```
public string XScaleFormatEvent( GraphPane pane, Axis axis, double val, int index )
{
   return (val+50).ToString();
}
```

In this case, an actual X value of 50 will be plotted at what appears to be an X value of 100 on the scale. Note that the tooltip from "Show Point Values" will still reflect the actual X value of 50. Although modifying the scale in this way may seem odd, it opens up possibilities to rescale data with out actually modifying the source data. For example, rather than showing actual data values, you can plot the data as a percent of range without changing the PointPairList.

Retrieved from "http://zedgraph.org/wiki/index.php?title=Customize_the_scale_format"

Use the DataSourcePointList to access database data

From ZedGraphWiki

Jump to: navigation, search

It is often desirable to access database data directly by ZedGraph. This saves the trouble of having to copy and convert the data into a PointPairList class. This is done using the DataSourcePointList class, which is an IPointList interface type. This means that you can use a DataSourcePointList in place of a PointPairList when you call the GraphPane.AddCurve(), GraphPane.AddBar(), etc. methods.

The DataSourcePointList is intended to be very simple to use. All you do is create an instance of a DataSourcePointList, specify the data source name, and then you specify the column names in the data table that will be used for each property. The following are the pertinent properties in the DataSourcePointList:

DataSource

typically the name of a DataTable containing the data of interest. This can also be an ArrayList or other data collection type.

XDataMember

the name of the column in DataSource that will contain the PointPair.X data values

YDataMember

the name of the column in DataSource that will contain the PointPair.Y data values

ZDataMember

the name of the column in DataSource that will contain the PointPair.Z data values

TagDataMember

the name of the column in DataSource that will contain the PointPair.Tag data values

The sample source code below demonstrates the use of these properties.

For an example, we will use the Northwind database, which is a sample MS access database. First, you need to link the database into your Visual Studio 2005 project with the following steps:

1. Right-click on your project in the solution explorer and select "Add Existing Item..."
2. Set the file type to "Data Files", navigate to the folder that contains the Northwind.mdb file, select the file and click "Add"
3. Open up the Tables tree, and check-mark the "Orders" table

4. Leave the dataset name as the default ("NorthwindDataSet")
5. Click "Finish"

You now have a "NorthwindDataSet" class added to your project, along with a table adapter that will fill a data table with data from the Orders table.

The following demonstates how to generate a graph from the database data using the DataSourcePointList. Once the graph is displayed, you will be able to see a scatter plot showing the freight cost versus order date. If you mouse-over the individual points, you will see that the tooltips are sourced from the "ShipName" column of the data table.

```
private void CreateGraph_DataSource( ZedGraphControl z1 )
{
   GraphPane myPane = z1.GraphPane;

   // Set the titles
   myPane.Title.Text = "DataSourcePointList Test";
   myPane.XAxis.Title.Text = "Date";
   myPane.YAxis.Title.Text = "Freight Charges ($US)";

   // Create a new DataSourcePointList to handle the database connection
   DataSourcePointList dspl = new DataSourcePointList();
   // Create a TableAdapter instance to access the database
   NorthwindDataSetTableAdapters.OrdersTableAdapter adapter =
            new NorthwindDataSetTableAdapters.OrdersTableAdapter();
   // Create a DataTable and fill it with data from the database
   NorthwindDataSet.OrdersDataTable table = adapter.GetData();

   // Specify the table as the data source
   dspl.DataSource = table;
   // The X data will come from the "OrderDate" column
   dspl.XDataMember = "OrderDate";
   // The Y data will come from the "Freight" column
   dspl.YDataMember = "Freight";
   // The Z data are not used
   dspl.ZDataMember = null;
   // The Tag data will come from the "ShipName" column
   // (note that this will just set PointPair.Tag = ShipName)
   dspl.TagDataMember = "ShipName";

   // X axis will be dates
   z1.GraphPane.XAxis.Type = AxisType.Date;

   // Make a curve
   LineItem myCurve = z1.GraphPane.AddCurve( "Orders", dspl, Color.Blue );
   // Turn off the line so it's a scatter plot
   myCurve.Line.IsVisible = false;

   // Show the point values.  These are derived from the "ShipName" database column,
   // which is set as the "Tag" property.
   z1.IsShowPointValues = true;

   // Auto set the scale ranges
   z1.AxisChange();
}
```

Retrieved from "http://zedgraph.org/wiki/index.php?title=Use_the_DataSourcePointList_to_access_database_data"

Edit Data at runtime after the graph is created

From ZedGraphWiki

Jump to: navigation, search

Often, there is a need to access data associated with the current graph after the graph is created. So you generally have a reference to the ZedGraphControl, but you don't have references to the GraphPane, MasterPane, Curves, GraphObj's, etc. This note documents several ways to access the objects and data.

Contents

Getting the GraphPane

First, you need to get a reference to the GraphPane of interest. If there is only one GraphPane on your control, then (assuming your ZedGraphControl is named 'zg1') you can get it like this:

```
GraphPane myPane = zg1.GraphPane
```

If you have multiple GraphPane's, you can refer to them in several ways. First, the GraphPane's are maintained by the MasterPane in a PaneList. You can access them from the PaneList in several ways. First, by their zero-based ordinal position. The first GraphPane is position 0, the second position 1, etc. For example, to get the third GraphPane, you would do this:

```
GraphPane myPane = zg1.MasterPane.PaneList[2];
```

You can also access the GraphPane by its title text. Note that in order to access a GraphPane by name, the specified name must match exactly the title of one of the GraphPane's in the list, including spaces, capitalization, etc. As an example, it would work like this:

```
GraphPane myPane = zg1.MasterPane.PaneList["My Pane Title"];
if ( myPane == null )
   <throw exception here>
```

Lastly, you can access a GraphPane by giving it an arbitrary "Tag" value. This can be any object, including an integer, string, etc. For example, assume that when you created your GraphPane, you set the myPane.Tag = "pane1". You can then recall that particular GraphPane, no matter what position it holds in the PaneList, with the following code:

```
int i = master.PaneList.IndexOfTag( "pane1" );
if ( i < 0 )
   <exception here>
GraphPane myPane = master.PaneList[i];
```

Getting the CurveItem or GraphObj

The above logic also applies to CurveItem's and GraphObj's. In the case of CurveItem's, and assuming that your CurveItem's are actually BarItem's, you would use:

```
BarItem myBar = mypane.CurveList[2] as BarItem;
```

To access the 3rd item in the list. Similarly, you would use:

```
BarItem myBar = myPane.CurveList["My Curve Title"] as BarItem;
if ( myBar == null )
   <throw exception>
```

Finally, you can also use the CurveItem.Tag property to recall a CurveItem by its user-defined Tag property. Assume myBar.Tag was set to the string "third". You can access it like this:

```
int i = myPane.CurveList.IndexOfTag( "third" );
if ( i < 0 )
   <exception here>
BarItem myBar = myPane.CurveList[i];
```

Similar logic applies to GraphObj's.

Getting the Point Data

Once you have access to the CurveItem, you can access or modify the data values like this:

```
IPointListEdit list = myBar.Points as IPointListEdit;
```

Clearing the Data

Refer to Clear all the data in a graph

Retrieved from "http://zedgraph.org/wiki/index.php?title=Edit_Data_at_runtime_after_the_graph_is_created"

Do not list a curve in the legend

From ZedGraphWiki

Jump to: navigation, search

If you have curves you do not want to display in the legend, make the Label invisible:

```
ZedGraph.PointPairList minList = new ZedGraph.PointPairList();
```

```
minList.Capacity = 2;
minList.Add(new ZedGraph.PointPair(pane.XAxis.Scale.Min, tag.minVal));
minList.Add(new ZedGraph.PointPair(pane.XAxis.Scale.Max, tag.minVal));
ZedGraph.LineItem minCurve = pane.AddCurve("Min: " + tag.Name, minList, c,
ZedGraph.SymbolType.None);
minCurve.YAxisIndex = yaxis;
minCurve.Label.IsVisible = false; // finally they do not show up in the Legend...
```

Retrieved from "http://zedgraph.org/wiki/index.php?title=Do_not_list_a_curve_in_the_legend"

Create a difference curve using interpolation

From ZedGraphWiki

Jump to: navigation, search

Interpolation

In some applications, you may want to create a new set of data based on a comparison of two existing sets of arbitrary data. If the existing data sets are not co-located at the same exact X value locations, then the data must interpolated in order to generate the new curve. The ZedGraph PointPairList has built-in functionality to handle both linear and spline-based data interpolation.

Given an existing dataset within a PointPairList, and assuming that the data are monotonically increasing in X with no PointPair.Missing values, you can interpolate the corresponding Y value at any arbitrary X value like this:

Linear interpolation

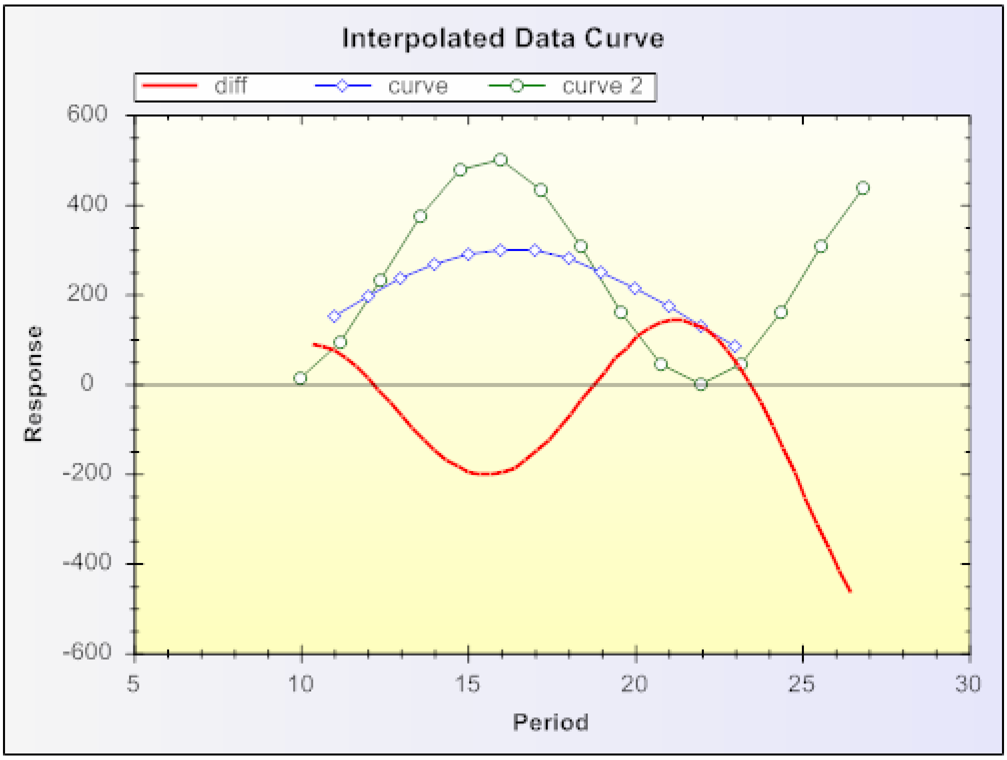
```
double y = myPointPairList.InterpolateX( x );
// where
//    x is any arbitrary x value
//       This method will extrapolate the data outside the existing range if
necessary
//    y is the interpolated result
```

Cardinal spline-based interpolation

```
double y = myPointPairList.SplineInterpolateX( x, tension );
// where:
//    x is any arbitrary x value within the defined data range
//       Any value outside the existing data range with result in PointPair.Missing
//    tension is a curvature setting, typically between 0 and 1
//       0 results in linear interpolation
//       0.5 is a typical smooth curve
//       > 1.0 can generate wild curvature results
//    y is the interpolated result
```

Sample Application

As an example, this article will use two existing sets of data to create an independent third dataset from the difference. The following graph shows the result:



The green and blue curves are the original data sets with dissimilar X spacing and different data ranges. The red curve spans the whole data file with higher point density and represents the difference between curve 1 and curve 2. The point values are calculated using cardinal spline interpolation.

```
// Simple plot with interpolated difference curve
private void CreateGraph_DifferencePlot( ZedGraphControl z1 )
```

```
{
  GraphPane myPane = z1.GraphPane;

  // Generate the first data set
  PointPairList list1 = new PointPairList();
  for ( int i = 0; i < 13; i++ )
  {
    double x = i + 11.0;
    double y = 150.0 * ( 1.0 + Math.Sin( i * 0.3 ) );
    list1.Add( x, y );
  }

  // Generate a second data set that is unrelated to the first
  PointPairList list2 = new PointPairList();
  for ( int i = 0; i < 15; i++ )
  {
    double x = i * 1.2 + 10.0;
    double y = 250.0 * ( 1.0 + Math.Sin( x * 0.5 ) );

    list2.Add( x, y );
  }

  // Make sure the data are sorted and monotonically increasing
  list1.Sort();
  list2.Sort();
  // Get the lower and upper limit of the data
  // This code can throw an exception if either list is empty
  double xMin = Math.Min( list1[0].X, list2[0].X );
  double xMax = Math.Max( list1[list1.Count - 1].X, list2[list2.Count - 1].X );

  // Create a new list that will hold the difference points
  PointPairList diffList = new PointPairList();

  // Select the number of points for the new difference curve
  // This is completely arbitrary, but more points will make it smoother in the
  // case of SplineInterpolation
  const int count = 50;

  // Loop for each data point to be created in the new PointPairList
  for ( int i=0; i<count; i++ )
  {
    // Calculated X values will be evenly spaced over the range of available data
    double x = xMin + (double) i * ( xMax - xMin ) / count;
    // Use spline interpolation to create the Y values for the new curve
    // A tension value of 0.5 is used, but anywhere between 0 and 1 is reasonable
    double y1 = list1.SplineInterpolateX( x, 0.5 );
    double y2 = list2.SplineInterpolateX( x, 0.5 );

    // Add the new Point to the list taking the difference between the Y values
    // If either value is Missing, it means that a point was extrapolated beyond
    // the available data, which is not allowed for SplineInterpolateX()
    // This won't happen with InterpolateX, since it allows extrapolation
    if ( y1 == PointPair.Missing || y2 == PointPair.Missing )
      diffList.Add( x, PointPair.Missing );
    else
      diffList.Add( x, y1 - y2 );
  }

  // Create the three curves -- two datasets, plus a difference curve
  LineItem diffCurve = myPane.AddCurve( "diff", diffList, Color.Red,
SymbolType.None );
  LineItem myCurve1 = myPane.AddCurve( "curve", list1, Color.Blue,
SymbolType.Diamond );
  LineItem myCurve2 = myPane.AddCurve( "curve 2", list2, Color.Green,
SymbolType.Circle );


  // Add some "pretty" stuff (optional)
  myCurve1.Symbol.Fill = new Fill( Color.White );
  myCurve2.Symbol.Fill = new Fill( Color.White );
  diffCurve.Line.Width = 2.0f;
  //diffCurve.Symbol.Fill = new Fill( Color.White );
  myPane.Title.Text = "Interpolated Data Curve";
  myPane.XAxis.Title.Text = "Period";
  myPane.YAxis.Title.Text = "Response";
  myPane.Legend.FontSpec.Size = 14;
  myPane.Fill = new Fill( Color.WhiteSmoke, Color.Lavender, 0F );
  myPane.Chart.Fill = new Fill( Color.FromArgb( 255, 255, 245 ),
     Color.FromArgb( 255, 255, 190 ), 90F );

  z1.AxisChange();
}
```

Retrieved from "http://zedgraph.org/wiki/index.php?title=Create_a_difference_curve_using_interpolation"

Use a gradient fill to individually color line segments

From ZedGraphWiki

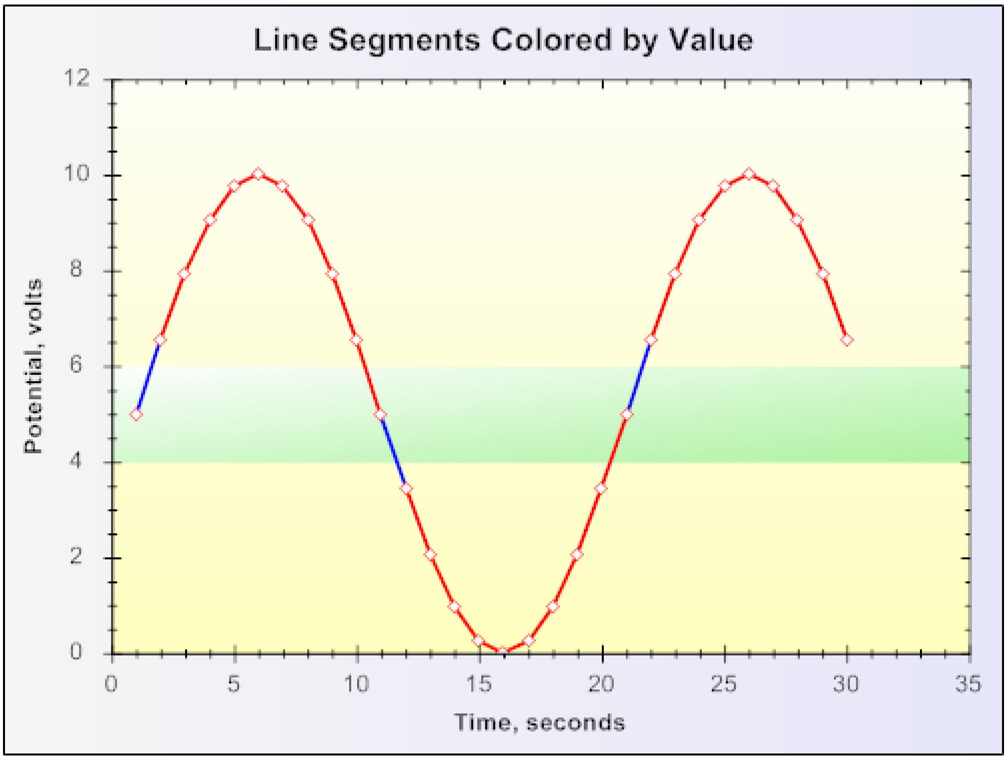
Jump to: navigation, search

Gradient Fills to Color Line Segments

Here's a technique you can use to apply different colors to individual segments of a LineItem. The technique involves using a GradientByValue fill applied to the LineItem.Line property of the LineItem. This is similar to the Multi-Colored Bar Demo, except that it applies to lines instead of bars.

A Simple Application

A simple application is shown below. This application will color all line segments red, unless they are within a certain data range (from Y=4.0 to 6.0), in which case they will be colored blue. Note that the coloration applies from any given point up to the following point. That is, even though the coloration applies to an individual point, the line segment is colored all the way until the next point in the PointPairList or IPointList. The code shown below produces the following graph:



This graph clearly demonstrates that the line segments are colored from any given point, up to the following point. In fact, you can see that the a point that crosses into the green zone will be colored red since it starts outside the green zone, and vice-versa. Following is the associated code:

```
private void CreateGraph_LineColorGradient( ZedGraphControl zgc )
{
  GraphPane myPane = zgc.GraphPane;

  PointPairList list = new PointPairList();
  const int count = 30;

  for ( int i = 0; i < count; i++ )
  {
    // Use an ordinary sine function to generate the curve
    double x = i + 1;
    double y = 5 * Math.Sin( (double) i * Math.PI * 3 / count ) + 5.0;

    // Set the Z value to be 2.0 if y is between 4 and 6, otherwise, it's 1.0
    list.Add( x, y, y > 4 && y < 6 ? 2.0 : 1.0 );
  }

  // Create a curve
  LineItem myCurve = myPane.AddCurve( "Test Curve", list, Color.Red,
SymbolType.Diamond );
```

```
  // use a gradient fill to color the each line segment according to its Z value
  // Color will be blue for Z = 2, and red for Z = 1
  Fill fill = new Fill( Color.Red, Color.Blue );
  fill.RangeMin = 1;
  fill.RangeMax = 2;
  fill.Type = FillType.GradientByZ;
  myCurve.Line.GradientFill = fill;
  // make the line fat
  myCurve.Line.Width = 2.0f;

  // Fill the symbols with white
  myCurve.Symbol.Fill = new Fill( Color.White );

  // Create a band of green to show the highlighted region
  BoxObj box = new BoxObj( 0.0, 6.0, 1.0, 2.0, Color.Empty,
      Color.FromArgb( 150, Color.LightGreen ) );
  // Use CoordType.XChartFractionYScale, so that Y values are regular scale values,
and
  // X values are chart fraction, ranging from 0 to 1
  box.Location.CoordinateFrame = CoordType.XChartFractionYScale;
  box.Fill = new Fill( Color.White, Color.FromArgb( 200, Color.LightGreen ),
45.0F );
  box.ZOrder = ZOrder.F_BehindGrid;
  box.IsClippedToChartRect = true;
  myPane.GraphObjList.Add( box );

  // Pretty it up
  myPane.Title.Text = "Line Segments Colored by Value";
  myPane.Title.FontSpec.Size = 18;
  myPane.XAxis.Title.Text = "Time, seconds";
  myPane.YAxis.Title.Text = "Potential, volts";
  myPane.Legend.IsVisible = false;
  myPane.Fill = new Fill( Color.WhiteSmoke, Color.Lavender, 0F );
  myPane.Chart.Fill = new Fill( Color.FromArgb( 255, 255, 245 ),
     Color.FromArgb( 255, 255, 190 ), 90F );
  zgc.IsAntiAlias = true;

  zgc.AxisChange();
}
```

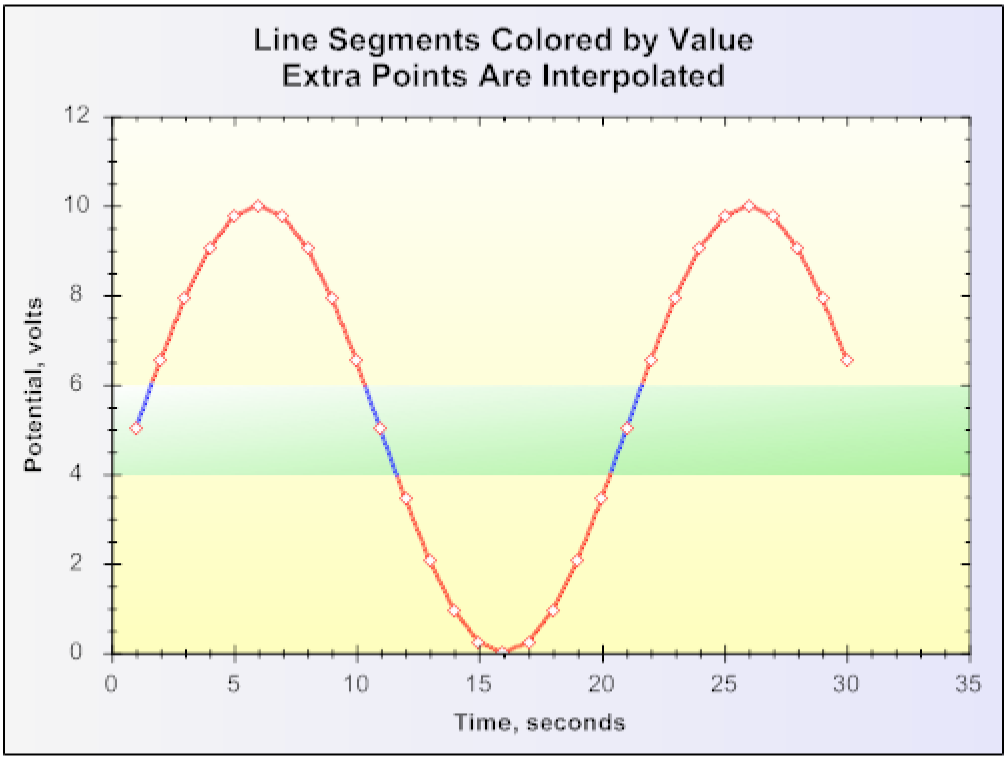
A Detailed Approach

In order to solve the problem of points sometimes crossing into or out of the green band, we need to improve the resolution of the graph. Although there are many possible approaches, one of the simplest is to just increase the number of points on the graph to improve the resolution. However, we don't want to increase the number of symbols on the chart. So, we can accomplish this by making two LineItems -- one LineItem to show only the symbols (and only the number of symbols in the original PointPairList), and a second LineItem to show the line with no symbols at a much higher resolution. The higher resolution curve will be able to better display when the curve is inside the green band, and when it is outside.

Note that, for this method to work, the second curve (the high-resolution one) must be displayed last so that the symbols from the first curve appear on top. In this case, the legend is not displayed, but you can just hide the legend for any curve by setting:

```
myCurve.Label.IsVisible = false;
```

Following is a modified graph that shows line segments that appear to change color right at the edge of the green zone.



This graph uses a technique called linear interpolation to "fill in" the extra points. See the topic Create a difference curve using interpolation for more information. The following is the modified version of the code:

```
private void CreateGraph_LineColorGradient2( ZedGraphControl zgc )
```

```
{
  GraphPane myPane = zgc.GraphPane;

  PointPairList list = new PointPairList();
  const int count = 30;

  for ( int i = 0; i < count; i++ )
  {
    // Use an ordinary sine function to generate the curve
    double x = i + 1;
    double y = 5 * Math.Sin( (double) i * Math.PI * 3 / count ) + 5.0;

    // Set the Z value to be 2.0 if y is between 4 and 6, otherwise, it's 1.0
    list.Add( x, y, y > 4 && y < 6 ? 2.0 : 1.0 );
  }

  // Create a curve with symbols only
  LineItem myCurve = myPane.AddCurve( "Test Curve", list, Color.Red,
SymbolType.Diamond );
  myCurve.Line.IsVisible = false;
  myCurve.Symbol.Fill = new Fill( Color.White );

  // Create a second curve, with lots of extra points
  const int count2 = 1000;
  PointPairList list2 = new PointPairList();
  // Points are equal-spaced, across all the X range
  double dx = ( list[list.Count - 1].X - list[0].X ) / (double) count2;

  // Calculate the extra points values using linear interpolation
  for ( int i = 0; i <= count2; i++ )
  {
    double x2 = list[0].X + dx * (double) i;
    double y2 = list.InterpolateX( x2 );

    list2.Add( x2, y2, y2 > 4 && y2 < 6 ? 2.0 : 1.0 );
  }

  // Add the second curve with no symbols
  LineItem myCurve2 = myPane.AddCurve( "Curve2", list2, Color.Blue,
SymbolType.None );

  // use a gradient fill to color the each line segment according to its Z value
  // Color will be blue for Z = 2, and red for Z = 1
  Fill fill = new Fill( Color.Red, Color.Blue );
  fill.RangeMin = 1;
  fill.RangeMax = 2;
  fill.Type = FillType.GradientByZ;
  myCurve2.Line.GradientFill = fill;
  // make the line fat
  myCurve2.Line.Width = 2.0f;

  // Create a band of green to show the highlighted region
  BoxObj box = new BoxObj( 0.0, 6.0, 1.0, 2.0, Color.Empty,
      Color.FromArgb( 150, Color.LightGreen ) );
  // Use CoordType.XChartFractionYScale, so that Y values are regular scale values,
and
  // X values are chart fraction, ranging from 0 to 1
  box.Location.CoordinateFrame = CoordType.XChartFractionYScale;
  box.Fill = new Fill( Color.White, Color.FromArgb( 200, Color.LightGreen ),
45.0F );
  box.ZOrder = ZOrder.F_BehindGrid;
  box.IsClippedToChartRect = true;
  myPane.GraphObjList.Add( box );

  // Pretty it up
  myPane.Title.Text = "Line Segments Colored by Value\nExtra Points Are
Interpolated";
  myPane.Title.FontSpec.Size = 18;
  myPane.XAxis.Title.Text = "Time, seconds";
  myPane.YAxis.Title.Text = "Potential, volts";
  myPane.Legend.IsVisible = false;
  myPane.Fill = new Fill( Color.WhiteSmoke, Color.Lavender, 0F );
  myPane.Chart.Fill = new Fill( Color.FromArgb( 255, 255, 245 ),
     Color.FromArgb( 255, 255, 190 ), 90F );
  zgc.IsAntiAlias = true;

  zgc.AxisChange();
}
```

Retrieved from "http://zedgraph.org/wiki/index.php?title=Use_a_gradient_fill_to_individually_color_line_segments"