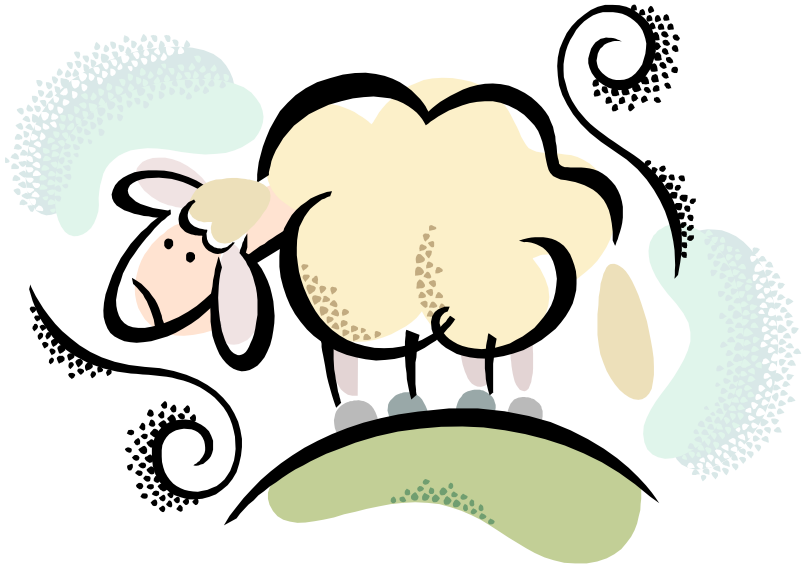


C# Programming IV-5:

System.Drawing & System.Drawing.Drawing2D Namespaces

Legal Stuff

This presentation as well as other presentations and documents found on the sheepsqueezers.com website may contain quoted material from outside sources such as books, articles and websites. It is our intention to diligently reference all outside sources. Occasionally, though, a reference may be missed. No copyright infringement whatsoever is intended, and all outside source materials are copyright of their respective author(s).

.NET Lecture Series

sheepsqueezers.com

	C# Programming I: Concepts of OOP	C# Programming II: Beginning C#	C# Programming III: Advanced C#	
C# Programming IV-1: System Namespace	C# Programming IV-2: System.Collections Namespace	C# Programming IV-3: System.Collections. Generic Namespace	C# Programming IV-4A: System.Data Namespace	C# Programming IV-4B: System.Data.Odbc Namespace
C# Programming IV-4C: System.Data.OleDb Namespace	C# Programming IV-4D: Oracle.DataAccess.Client Namespace	C# Programming IV-4E: System.Data.SqlClient Namespace	C# Programming IV-4F: System.Data.SqlTypes Namespace	C# Programming IV-5: System.Drawing/(2D) Namespace
C# Programming IV-6: System.IO Namespace	C# Programming IV-7: System.Numerics	C# Programming IV-8: System.Text and System.Text. RegularExpressions Namespaces	C# Programming V: Introduction to LINQ	

C# Self-Inflicted Project #1 Address Cleaning

C# Self-Inflicted Project #2 Large Intersection Problem

Charting Our Course

- The `System.Drawing/System.Drawing.Drawing2D` Namespaces
- What Next?

The `System.Drawing` and `System.Drawing.Drawing2D` Namespaces

The `System.Drawing` namespace is defined by Microsoft as follows:

The System.Drawing namespace provides access to GDI+ basic graphics functionality. More advanced functionality is provided in the System.Drawing.Drawing2D, System.Drawing.Imaging, and System.Drawing.Text namespaces. The Graphics class provides methods for drawing to the display device. Classes such as Rectangle and Point encapsulate GDI+ primitives. The Pen class is used to draw lines and curves, while classes derived from the abstract class Brush are used to fill the interiors of shapes.

The `System.Drawing.Drawing2D` namespace is defined by Microsoft as follows:

The System.Drawing.Drawing2D namespace provides advanced two-dimensional and vector graphics functionality.

I have combined these two related namespaces together in the slides below. The classes, enumerations, etc. from both namespaces appear intermingled. When writing code using these namespaces, include the following two lines at the top of your C# program:

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

The `System.Drawing` and `System.Drawing.Drawing2D` Namespaces

In the listing of the classes, enumerators, etc. below, I have placed things in an order which I believe is a little more useful than just an alphabetical order. Note that the pens come first, then brushes, fonts, colors, strings, images, bitmaps, graphics, etc.

Since these namespaces are a bit more complex than the other namespaces we've dealt with so far, the next section contains a brief explanation and a few examples on how to use these namespaces.

Introduction

Introduction

Back in the day, graphs were draw by hand on paper or board using a pen of some kind. Any graphic element that needed to be fill was filled in by using a brush. Letters were drawn on the graph either by hand or by using letter *rub-ons*.

These *ancient* concepts still hold today when using the `System.Drawing` and `System.Drawing.Drawing2D` namespaces.

You set up your pen by instantiating the `Pen` class and your brush by instantiating the `Brush` class. The `Pen` class helps with the drawing of lines (with or without end-caps and/or arrows). The `Brush` class helps with filling in areas such as bars for bar charts, etc.

Nowadays, the surface upon which you draw can be a bitmap of a specified size, a pre-existing image that you want to *pimp out*, or a GUI interface such as a Windows form. Now, you instantiate the `Bitmap` class to create/modify bitmaps. The `Image` class is an abstract class and the `Bitmap` class is derived from it. And you use the `Graphics` class to employ more complex drawing features such as lines and text rather than the limited features of the `Bitmap` class. The `Graphics` class can also be used to draw on Windows forms! (Although we haven't discussed GUI stuff yet, we will show you how to draw on a Windows form later on in this *Introduction*.)

Don't forget to include the following three lines at the top of your C# code:

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

Introduction

To make things a little easier, Microsoft has created the `Pens` and `Brushes` classes which hold all a range of standard colors for you to use such as BurlyWood and OliveDrab. Microsoft has also created the `SystemPens` and `SystemBrushes` classes containing colors related to system elements such as menu bars and scroll bars. Note that for the `Pens` and `SystemPens` classes, the width is set a 1, but you can change this. Let's see how to set up a pen and brush:

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

class DrawingExample {

 public static void Main() {

  //Set up pen and brush
  Pen oPenBurlyWood = Pens.BurlyWood;
  Brush oBrushOliveDrab = Brushes.OliveDrab;

 }

}
```

Next, we need something to draw on. Let's start out by creating a bitmap that is 800 pixels wide and 600 pixels tall. Note that the default horizontal and vertical dots per inch is 96. That is, there are 96 dots per inch on the bitmap. You can change this by using the `SetResolution` method of the bitmap.

Introduction


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

class DrawingExample {

 public static void Main() {

  //Set up pen and brush
  Pen oPenBurlyWood = Pens.BurlyWood;
  Brush oBrushOliveDrab = Brushes.OliveDrab;

  //Set up drawing surface
  Bitmap oBM = new Bitmap(800,600); //96dpi

 }

}
```

Finally, let's draw a line on the surface from the origin (0,0) to (250,250):

```
  //Draw a line on the graph
  for(Int32 indx=0; indx<250; indx++) {
   oBM.SetPixel(indx,indx, oPenBurlyWood.Color);
  }
```

Note that we are using the `SetPixel` method. The first parameter is the x-coordinate, the second parameter is the y-coordinate, and the third parameter is the color you want the pixel to be. Note that you could just as easily have used `Color.BurlyWood` instead of `oPenBurlyWood.Color`.

Introduction

Finally, let's save the image to disk:

```
//Save the bitmap
oBM.Save(@"C:\TEMP\TEST\Graph1.bmp",ImageFormat.Bmp);
```

Note that we are using the `Save` method. The first parameter is the location on disk where you want the image to be saved and the second parameter is the image format you want to use (here I save the image as a .bmp). Note that the `ImageFormat` class is located in the `System.Drawing.Imaging` namespace, so you will have to include this in your code as well as the `System.Drawing` and `System.Drawing.Drawing2D` namespaces.

Here is what our image looks like:

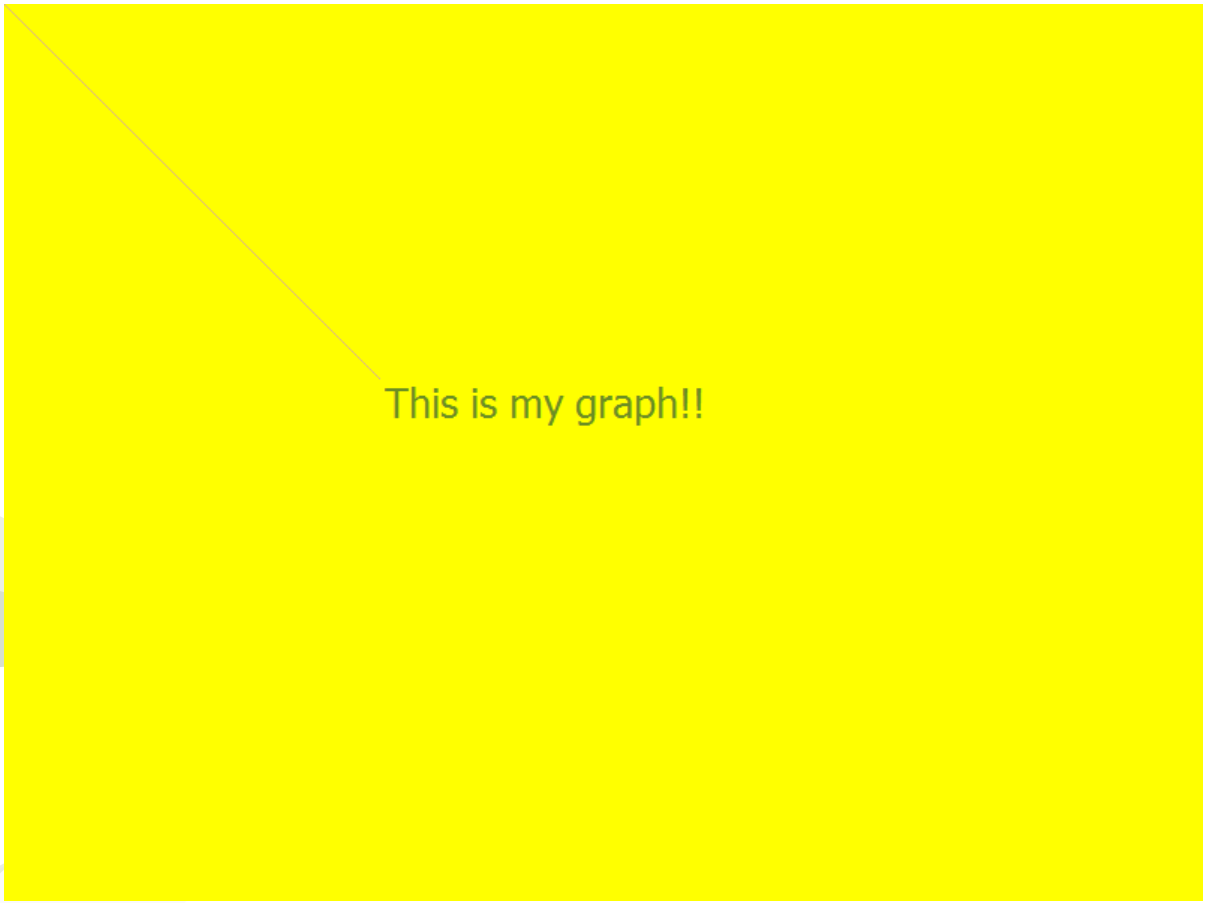
Now, if you look over the methods available in the `Bitmap` class, you will see that there are no "draw line", "place text", etc. methods. For that, we have to employ the `Graphics` class. We can obtain a `Graphics` object from our bitmap by using one of the `Graphic`s object's constructors.

Introduction

Here is an example (the results are on the next slide):

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

class DrawingExample {

 public static void Main() {

  //Set up pen and brush
  Pen oPenBurlyWood = Pens.BurlyWood;
  Brush oBrushOliveDrab = Brushes.OliveDrab;

  //Set up drawing surface
  Bitmap oBM = new Bitmap(800,600); //96dpi

  //Create a Graphics object from the existing Bitmap object
  Graphics oGR = Graphics.FromImage(oBM);

  //Clear the graph and fill it in with the color white
  oGR.Clear(Color.Yellow);

  //Draw line on graph using the DrawLine method.
  oGR.DrawLine(oPenBurlyWood,0,0,250,250);

  //Place some text on the graph
  oGR.DrawString("This is my graph!!",new Font("Tahoma",20),oBrushOliveDrab,new Point(250,250));

  //Place oGR on the bitmap
  oGR.DrawImage(oBM,new Point(0,0));

  //Save the bitmap
  oBM.Save(@"C:\TEMP\TEST\Graph1.bmp",ImageFormat.Bmp);

 }

}
```

Introduction



As you can see from the code on the previous page, our life is made considerably easier by using a `Graphics` object and the methods and properties that are available to it rather then using the methods and properties available to the `Bitmap` and `Image` objects.

Now, you've probably noticed that the line we drew started at the upper-left corner of the image rather than the "traditional" lower-left corner. It's true, the origin (0,0) is located in the upper-left corner! Also, this means that the Y-axis is pointing **down** rather than up as is traditional.

Introduction

But, never fear, because there is a way to change the origin to be the lower-left corner. It involves using a matrix transformation on your graphics object (our oGR in the example above). There are several matrix transformations you can perform:

Identity

This matrix transformation does not change anything at all. This is similar to multiplying a number by the number 1: no effect. But, as you will see, you can temporarily undo all transformations on your graphics object by using the identity matrix. This matrix looks like this:

```
Matrix oMAT_IDENTITY = new Matrix(1,0,0,1,0,0); //Default transformation when you create a Graphics object.
```

Scaling

This involves stretching or shrinking the object along either the x-axis, y-axis, or both axes simultaneously. This matrix looks like this:

```
Matrix oMAT_SCALE = new Matrix(Sx,0,0,Sy,0,0); //Sx and Sy are scaling factors.  If both are 1, then you get
                                               // the identity matrix and no scaling is performed.
```

Translation

This involves moving objects to another location on the graph; that is, changing their x- and y-coordinates. This matrix looks like this:

```
Matrix oMAT_TRANSLATE = new Matrix(1,0,0,1,CHANGE_X,CHANGE_Y); //Move objects to a different point.
```

Introduction

Rotation

This matrix transformation rotates the coordinate axes around the current origin point counter-clockwise (if theta<0) or clockwise (if theta>0). This matrix looks like this:

```
Matrix oMAT_ROTATION = new Matrix(COS_THETA,SIN_THETA,-SIN_THETA,COS_THETA,0,0); //Rotation theta radians
```

Note that you can combine one or more of these matrix transformations by using the Multiply method of the Matrix object. For example, given two matrix transformation, `oMAT1` and `oMAT2`, we multiply these two matrices like this:

```
oMAT1.Multiply(oMAT2); //oMAT1 is REPLACED  with the results of the multiplication.
```

Let's see some examples. Below we create a 100x100 pixel bitmap and place the letter "A" on the graph. This example uses no transformations at all (see next slide).

Introduction


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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //Based on the Width and Height of the letter A, place it dead center on the graph.
  String pLetter = "A";
  SizeF oLetterSize = new SizeF();
  oLetterSize = oGR.MeasureString(pLetter,new Font("Vendana",16));
  PointF pLetterLocation = new PointF(100/2 - oLetterSize.Width/2,100/2-oLetterSize.Height/2);
  oGR.DrawString("A",new Font("Vendana",16),Brushes.White,pLetterLocation);

  oGR.DrawImage(oBM,new PointF(0,0));
  oBM.Save(@"C:\TEMP\TEST\TESTGRAPH1.bmp",ImageFormat.Bmp);

 }

}
```

Next, using the scaling transformation, make the letter A twice as big.

Introduction

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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //Apply a matrix transformation
  Matrix oMAT_SCALE = new Matrix(2,0,0,2,0,0);
  oGR.Transform = oMAT_SCALE;

  //Based on the Width and Height of the letter A, place it dead center on the graph.
  String pLetter = "A";
  SizeF oLetterSize = new SizeF();
  oLetterSize = oGR.MeasureString(pLetter,new Font("Vendana",16));
  PointF pLetterLocation = new PointF(100/2 - oLetterSize.Width/2,100/2-oLetterSize.Height/2);
  oGR.DrawString("A",new Font("Vendana",16),Brushes.White,pLetterLocation);

  oGR.DrawImage(oBM,new PointF(0,0));
  oBM.Save(@"C:\TEMP\TEST\TESTGRAPH2.bmp",ImageFormat.Bmp);

 }

}
```

Now, you'll note that the letter A is not in the center of the graph, but rather centered around the lower-right corner. This is because the scaling makes the insides of the image behave as if it is twice as big…so, the point (50,50) is

now at the lower-right corner instead of the center. Again, this is because everything is scaled twice as big (since the scaling factors I used were 2 for the x-axis and 2 for the y-axis). Note that the graph itself is still 100 pixels by 100 pixels, though!! The scaling affects the *insides* of the bitmap and not the overall bitmap size!! Also, take note that the letter A, what you can see of it, is twice as big as before even with the same 16 point Verdana font!!

Next, let's translate the origin to the center of the graph.

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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //Apply a matrix transformation
  Matrix oMAT_TRANSLATE = new Matrix(1,0,0,1,50,50);
  oGR.Transform = oMAT_TRANSLATE;

  //Draw a circle centered around the new origin (0,0) which is the old origin (50,50)!!
  oGR.DrawEllipse(Pens.White,0,0,2,2);

  …continued on next slide…
```

Introduction


```
    //Based on the Width and Height of the letter A, place it dead center on the graph.
    String pLetter = "A";
    SizeF oLetterSize = new SizeF();
    oLetterSize = oGR.MeasureString(pLetter,new Font("Vendana",16));
    PointF pLetterLocation = new PointF(100/2 - oLetterSize.Width/2,100/2-oLetterSize.Height/2);
    oGR.DrawString("A",new Font("Vendana",16),Brushes.White,pLetterLocation);

    oGR.DrawImage(oBM,new PointF(0,0));
    oBM.Save(@"C:\TEMP\TEST\TESTGRAPH3.bmp",ImageFormat.Bmp);

  }

}
```

As you can see, the circle we've drawn at the point (0,0) is now in the center of the graph at the old point (50,50). Our letter A, really is located at the new point (50,50) which corresponds to the old point (100,100).

Next, let's rotate the graph 45 degrees counter-clockwise and see where the letter A winds up. See next slide.

Introduction


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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  oGR.DrawString("B",new Font("Vendana",16),Brushes.White,new PointF(50,50));

  //Apply a matrix transformation
  Single fTheta = (Single) Math.PI/4; //-45 degrees in radians
  Single fSinTheta = (Single) Math.Sin(-fTheta);
  Single fCosTheta = (Single) Math.Cos(-fTheta);
  Matrix oMAT_ROTATE = new Matrix(fCosTheta,fSinTheta,-fSinTheta,fCosTheta,0,0);
  oGR.Transform = oMAT_ROTATE;

  //Based on the Width and Height of the letter A, place it dead center on the graph.
  String pLetter = "A";
  SizeF oLetterSize = new SizeF();
  oLetterSize = oGR.MeasureString(pLetter,new Font("Vendana",16));
  PointF pLetterLocation = new PointF(100/2 - oLetterSize.Width/2,100/2-oLetterSize.Height/2);
  oGR.DrawString("A",new Font("Vendana",16),Brushes.White,pLetterLocation);

  oGR.DrawImage(oBM,new PointF(0,0));
  oBM.Save(@"C:\TEMP\TEST\TESTGRAPH4.bmp",ImageFormat.Bmp);

 }

}
```

Introduction

I placed the letter B on the graph before the rotation to show you that only objects added to the graph AFTER the transformation are affected by the transformation. Here, the coordinate axes have been rotated so that the original point (50,50) is now along the x-axis. Note that the rotation is around what is currently the origin; here, (0,0) is the origin.

Now, we can combine transformations together to produce the results we desire. For example, I want to keep the letter A at the center of the graph at the original point (50,50), but I want to rotate the letter A 45 degrees so that it appears at an angle. Here we need to translate the origin to (50,50) and then rotate 45 degrees. This involves multiplying the two matrixes together to get one combined matrix which will be stored in the Transform property of the graphics object. Here is the code:

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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //1. Create translation matrix
  Matrix oMAT_TRANSLATE = new Matrix(1,0,0,1,50,50); //old (50,50) becomes new (0,0)
```

…continued on next slide…

Introduction


```
    //2. Create rotation matrix
    Single fTheta = (Single) Math.PI/4; //45 degrees in radians
    Single fSinTheta = (Single) Math.Sin(-fTheta);
    Single fCosTheta = (Single) Math.Cos(-fTheta);
    Matrix oMAT_ROTATE = new Matrix(fCosTheta,fSinTheta,-fSinTheta,fCosTheta,0,0);

    //3. Perform multiplication
    Matrix oMAT_FINAL = oMAT_TRANSLATE.Clone();
    oMAT_FINAL.Multiply(oMAT_ROTATE);
    oGR.Transform = oMAT_FINAL;

    //Based on the Width and Height of the letter A, place it dead center on the graph.
    String pLetter = "A";
    SizeF oLetterSize = new SizeF();
    oLetterSize = oGR.MeasureString(pLetter,new Font("Vendana",16));
    PointF pLetterLocation = new PointF(0/2 - oLetterSize.Width/2,0/2-oLetterSize.Height/2);
    oGR.DrawString("A",new Font("Vendana",16),Brushes.White,pLetterLocation);

    oGR.DrawImage(oBM,new PointF(0,0));
    oBM.Save(@"C:\TEMP\TEST\TESTGRAPH5.bmp",ImageFormat.Bmp);

  }

}
```

Note that I am cloning (making a duplicate copy) of the `oMAT_TRANSLATE` matrix and calling it `oMAT_FINAL`. The reason I am doing this is because the Multiply method will overwrite the object of the multiplication based on its parameter matrix. In this case, I wanted to ensure that my matrix `oMAT_TRANSLATE` was not overwritten.

Introduction

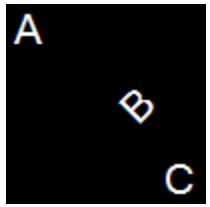
Now, if you take a look at the purple colored line above, you will see that I am specifying the origin to be (0,0) instead of (50,50) as before. Remember that we translated the origin to the center of the graph and, thus, it becomes the new (0,0).

Now, rather than having to constantly change the transformation matrix of the graphics object back to the original transformation every time you, say, want to rotate a word, or move the origin to another point, you can save the current graphics state and then restore it back to the original graphics state once you are done manipulating the graph. It's like restoring a piece of toast back to its original untoasted bread form. Okay, maybe it's not like that, but I think you get the idea. For example, below we place the letter A on the graph at the origin (0,0), we then move the origin to the center, rotate the axes 45 degrees and draw the letter B; we then restore the graphics state and place the letter C at the point (75,75).

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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  …continued on next slide…
```

Introduction


```
//Place letter A at the origin (uncentered).
oGR.DrawString("A",new Font("Vendana",16),Brushes.White,new PointF(0,0));

//1. Create translation matrix
Matrix oMAT_TRANSLATE = new Matrix(1,0,0,1,50,50); //old (50,50) becomes new (0,0)

//2. Create rotation matrix
Single fTheta = (Single) Math.PI/4; //45 degrees in radians
Single fSinTheta = (Single) Math.Sin(-fTheta);
Single fCosTheta = (Single) Math.Cos(-fTheta);
Matrix oMAT_ROTATE = new Matrix(fCosTheta,fSinTheta,-fSinTheta,fCosTheta,0,0);

//3. Perform multiplication
Matrix oMAT_FINAL = oMAT_TRANSLATE.Clone();
oMAT_FINAL.Multiply(oMAT_ROTATE);

//4. Save the graphics state
GraphicsState oGS = oGR.Save();

//5. Modify the graphics object's Transform property.
oGR.Transform = oMAT_FINAL;

//Place the letter B on the graph at the new origin (0,0) which is the old origin (50,50).
oGR.DrawString("B",new Font("Vendana",16),Brushes.White,new PointF(0,0));

//6. Restore the graphics state to the previous state before the translation/rotation occurred.
oGR.Restore(oGS);

//Place the letter C on the graph
oGR.DrawString("C",new Font("Vendana",16),Brushes.White,new PointF(75,75));

oGR.DrawImage(oBM,new PointF(0,0));
oBM.Save(@"C:\TEMP\TEST\TESTGRAPH6.bmp",ImageFormat.Bmp);

}

}
```



Introduction

Note that there are several methods available to you when using a `Graphics` object to perform matrix-like transformation. For example, the `TranslateTransform` method performs a translation similar to our translation matrix, but "adds" it to the already existing transformations. This way you can build onto any pre-existing transformations. The `RotateTransform` method performs a rotation with its parameter in degrees. The `ScaleTransform` performs scaling. You can also perform matrix multiplication using the `MultiplyTransform` method on the graphics object.

Now, as stated before, when you first create a Graphics object, the origin (0,0) is located at the upper-left corner of the graph. The positive x-axis points to the right and the positive y-axis points to the bottom of the graph. This is very annoying when drawing charts and graphs, so here is a matrix transformation that moves the origin (0,0) to the bottom-left corner of the graph and also makes the positive y-axis point upwards:

```
Matrix oMAT_NEWORIGIN = new Matrix(1,0,0,-1,0,HEIGHT_OF_YOUR_CHART);
oGR.Transform=oMAT_NEWORIGIN;
```

Now, let's apply this transformation right off the bat once we create our graph object `oGR` and let's draw a line from the origin to the center of the graph. See the next slide.

Introduction


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

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //Transform the graph oGR so that the origin (0,0) is at the BOTTOM LEFT of the graph!!
  Matrix oMAT_NEWORIGIN = new Matrix(1,0,0,-1,0,100);
  oGR.Transform=oMAT_NEWORIGIN;

  //Place line on the graph.
  oGR.DrawLine(Pens.White,0,0,50,50);

  oGR.DrawImage(oBM,new PointF(0,0));
  oBM.Save(@"C:\TEMP\TEST\TESTGRAPH7.bmp",ImageFormat.Bmp);

 }

}
```

Clearly, something is wrong! We are not drawing two lines! What's going on??

If you take a look at the bold red line of code above, you'll see that I am asking for my graph object `oGR` to be placed on the bitmap `oBM` at the location (0,0) by using the `DrawImage` method. The second parameter to `DrawImage` is expecting a PointF structure that represents the location of the upper-left corner of the drawn image.

Introduction

But, (0,0) is no longer the upper-left corner of the graph, it's (0,-100)! Let's fix the code and try it again:

```
oGR.DrawImage(oBM,new PointF(0,-100));
```

Here is the correct bitmap image:

Another way around this, is to remove all transformations just before saving the graph to the bitmap, like this:

```
oGR.Transform=new Matrix(1,0,0,1,0,0);  //Remove all transformations by resetting to identity matrix.
oGR.DrawImage(oBM,new PointF(0,0));
oBM.Save(@"C:\TEMP\TEST\TESTGRAPH8.bmp",ImageFormat.Bmp);
```

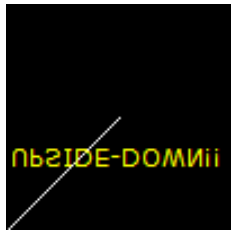
Now, one very important thing to remember when using this code...

```
//Transform the graph oGR so that the origin (0,0) is at the BOTTOM LEFT of the graph!!
Matrix oMAT_NEWORIGIN = new Matrix(1,0,0,-1,0,100);
oGR.Transform=oMAT_NEWORIGIN;
```

...is that, although the y-axis now points upwards, any text you place on the graph will be upside-down!! This is due to the negative one in the Matrix constructor above. It's something you have to live with, so get used to using the graphics save and restore state methods!

Introduction

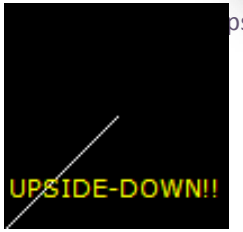

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;

class DrawingExample {

 public static void Main() {

  Bitmap oBM=new Bitmap(100,100);
  Graphics oGR = Graphics.FromImage(oBM);

  //Transform the graph oGR so that the origin (0,0) is at the BOTTOM LEFT of the graph!!
  Matrix oMAT_NEWORIGIN = new Matrix(1,0,0,-1,0,100);
  oGR.Transform=oMAT_NEWORIGIN;

  //Place line on the graph from the lower-left corner (0,0) to the center (50,50).
  oGR.DrawLine(Pens.White,0,0,50,50);

  //Place text on the graph...note that it will appear upside down!
  oGR.DrawString("UPSIDE-DOWN!!",new Font("Verdana",8),Brushes.Yellow,new Point(0,25));

  //Save the image to disk.
  oGR.Transform=new Matrix(1,0,0,1,0,0);
  oGR.DrawImage(oBM,new PointF(0,0));
  oBM.Save(@"C:\TEMP\TEST\TESTGRAPH9.bmp",ImageFormat.Bmp);

 }

}
```



As you see, the text is upside-down. To remedy this, you can save the graphics state, use a different transformation (like the identity), place the text on the image and then restore the graphics state.

Introduction

```
…
  //Place text on the graph...note that it will appear right-side up!
  GraphicsState oGS = oGR.Save();
  oGR.Transform = new Matrix(1,0,0,1,0,0);
  oGR.DrawString("UPSIDE-DOWN!!",new Font("Verdana",8),Brushes.Yellow,new Point(0,75));
  oGR.Restore(oGS);
…
```



As you see, the text is right-side up! Nice!! Note, though, that the point we are placing the text is based on the original origin (0,0) located at the upper-left corner of the graph! C'est la vie!

Finally, if you look closely at the image on this slide and the image on the previous slide, it looks like the text is much lower on this slide than on the previous slide. This is because the `DrawString` method uses the upper-left corner of the text string (where the letter U is in the images) as the point to start placing the text. But, when we use the `oMAT_NEWORIGIN` transformation, the text is flipped upside-down along with the starting point for the text!

Now, let's try to draw a graph of Microsoft's stock closing price. I have loaded this data into a table in SQL Server. This table contains the stock symbol (MSFT), the high, low, closing and volume and well as the date. I will only pull the date and the close and use a WHERE clause in my SQL to pull just data for MSFT. Be aware that the code you see below is a first pass and is not very pretty! Also, in the example below I am using LINQ to Datasets as well as Analytic Functions in my call to my SQL Server database.

Introduction


```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Data.Linq;
using System.Linq.Expressions;

class StockChart {

 String StockSymbol;
 String OutputFile;
 Int32 iChartSizeWidth;
 Int32 iChartSizeHeight;
 Color BackgroundColor;
 Color ForegrondColor;
 Font oFont_Title;
 Font oFont_Axes;
 Bitmap oBM;
 Graphics oGR;
 Pen oPenBlack;
 Int32 iOffset;
 Int32 iOrigin_X;
 Int32 iOrigin_Y;
 Int32 iAxisLength_X;
 Int32 iAxisLength_Y;
 Single sMaxClose;
 Int32 iNbrDataPoints;
 Int32 iNbrXAxisTicks;
 Array aDistXAxisValues;
 Array aDistYAxisValues;
```

Introduction


```
DataTable oStockDataTable;
Single sXAxisSkip;
Single sYAxisSkip;

//Constructor
public StockChart(String pStockSymbol,String pOutputFile) {

 iChartSizeWidth=800;
 iChartSizeHeight=600;
 oBM=new Bitmap(iChartSizeWidth,iChartSizeHeight);
 oGR = Graphics.FromImage(oBM);

 //Transform the graph oGR so that the origin (0,0) is at the BOTTOM LEFT of the graph!!
 Matrix oMAT = new Matrix(1,0,0,-1,0,iChartSizeHeight);
 oGR.Transform=oMAT;

 StockSymbol=pStockSymbol;
 OutputFile=pOutputFile;
 BackgroundColor = Color.White;
 ForegrondColor = Color.Black;
 oFont_Title = new Font("Verdana",16);
 oFont_Axes = new Font("Verdana",8);
 oPenBlack = Pens.Black;
 iOffset=30; //30 pixel offset
 //Determine the origin (0,0) of the xy-axes
 iOrigin_X = 0+2*iOffset;
 iOrigin_Y = iChartSizeHeight-2*iOffset;
 //Determine the length of the x-axis and y-axis
 iAxisLength_X = (iChartSizeWidth-2*iOffset) - (0+2*iOffset);
 iAxisLength_Y = (iChartSizeHeight-2*iOffset) - (0+2*iOffset);
 oStockDataTable = new DataTable();

}

//Smart Fields/Properties
```

Introduction


```
public Color Background {

 get {
  return(BackgroundColor);
 }

 set {
  BackgroundColor=value;
 }

}

public String Symbol {

 get {
  return(StockSymbol);
 }

 set {
  StockSymbol=value;
 }

}

//Methods (public)
public void CreateChart() {
 SetBackgroundColor();
 DrawChartArea();
 PlaceTitleOnChart("Stock Chart of " + StockSymbol);
 DrawChartAxes();
 PullData(StockSymbol);
 DrawXAxisTickmarks();
 DrawYAxisTickmarks();
 PlotData();
 SaveGraph();
```

Introduction


```
}

 //Methods (private)
 private void PlotData() {

  Single sCloseValue_PREVIOUS;
  Single sCloseValue_CURRENT;

  //Place the data into an array.
  Array aPlotData = (from A in oStockDataTable.AsEnumerable()
                     select A.Field<Double>("MarketClose")).ToArray();

  //Place the tick marks on the X-Axis
  GraphicsState oGS = oGR.Save();

  //Move the current origin (0,0) to where we want the origin of the axes intersection to be (60,60)
  oGR.TranslateTransform(60,60);

  //Draw data on grid as a series of short lines
  for(Int32 indx=1;indx<iNbrDataPoints;indx++) {

   //Get the current and previous closing data point
   sCloseValue_PREVIOUS=(Single)Math.Round( ((Double)sYAxisSkip)*((Double)aPlotData.GetValue(indx-1)) , 1);
   sCloseValue_CURRENT= (Single)Math.Round( ((Double)sYAxisSkip)*((Double)aPlotData.GetValue(indx  )) , 1);
   oGR.DrawLine(oPenBlack,((Single)(indx-
1))*sXAxisSkip,sCloseValue_PREVIOUS,((Single)indx)*sXAxisSkip,sCloseValue_CURRENT);

  }

  oGR.Restore(oGS);

 }
```

Introduction


```
private void DrawXAxisTickmarks() {

  //Get a distinct list of years on the X-Axis
  aDistXAxisValues = (from A in oStockDataTable.AsEnumerable()
                      select A.Field<String>("MARKET_YEAR")).Distinct().ToArray();

  for(Int32 indx=0;indx<aDistXAxisValues.Length;indx++) {
   Console.WriteLine("{0}",(String)aDistXAxisValues.GetValue(indx));
  }

  iNbrXAxisTicks=aDistXAxisValues.Length;
  Console.WriteLine("iNbrXAxisTicks={0}",iNbrXAxisTicks);

  //Based on the length of the X-Axis (iAxisLength_X,in pixels) and the number of data points (iNbrDataPoints),
determine the skip-interval to place the tick marks
  sXAxisSkip = (Single)Math.Round((Single)iAxisLength_X / (Single)iNbrDataPoints,1);
  Console.WriteLine("sXAxisSkip={0}",sXAxisSkip);

  //Place the tick marks on the X-Axis
  GraphicsState oGS = oGR.Save();

  //Move the current origin (0,0) to where we want the origin of the axes intersection to be (60,60)
  oGR.TranslateTransform(60,60);

  //Draw X-Axis Tickmarks
  for(Int32 indx=0;indx<iNbrDataPoints;indx++) {
   oGR.DrawLine(oPenBlack,((Single)indx)*sXAxisSkip,0,((Single)indx)*sXAxisSkip,-3);
  }

  oGR.Restore(oGS);

  //Next, place the words Stock Date under the X-Axis
  oGS = oGR.Save();
  oGR.Transform = new Matrix(1,0,0,1,0,0);
  String sXAxisTitle = "Stock Date";
```

Introduction


```
    SizeF oXAxisTitleSize = new SizeF();
    oXAxisTitleSize = oGR.MeasureString(sXAxisTitle,oFont_Axes);
    PointF pXAxisTitleLocation = new PointF( ((Single)iChartSizeWidth)/2 -
  oXAxisTitleSize.Width/2,iChartSizeHeight - 2*iOffset + oXAxisTitleSize.Height/2);
    oGR.DrawString(sXAxisTitle,oFont_Axes,Brushes.Black,pXAxisTitleLocation);
    oGR.Restore(oGS);

    Console.WriteLine("{0}/{1}",pXAxisTitleLocation.X,pXAxisTitleLocation.Y);
   }

   private void DrawYAxisTickmarks() {

    //Determine the skip pattern for the Y-axis tick marks
    sYAxisSkip = (Single)Math.Round((Single)iAxisLength_Y / (Single)sMaxClose,1);
    Console.WriteLine("sYAxisSkip={0}",sYAxisSkip);

    //Place the tick marks on the X-Axis
    GraphicsState oGS = oGR.Save();

    //Move the current origin (0,0) to where we want the origin of the axes intersection to be (60,60)
    oGR.TranslateTransform(60,60);

    //Draw Y-Axis Tickmarks (but, say, only 8 ticks instead of all of them)
    for(Int32 indx=0;indx<=8;indx++) {
     oGR.DrawLine(oPenBlack,0,((Single)indx)*4*sYAxisSkip,-3,((Single)indx)*4*sYAxisSkip);
    }

    oGR.Restore(oGS);

    //Next, place the Y-Axis tickmark values to the left of the Y-Axis
    oGS = oGR.Save();
    oGR.Transform = new Matrix(1,0,0,1,0,0);

    String sYAxisTitle;
    SizeF oYAxisTitleSize = new SizeF();
```

Introduction


```
    PointF pYAxisTitleLocation = new PointF(0,0);

    for(Int32 indx=0;indx<=8;indx++) {
     sYAxisTitle = ( ((Single)indx)*4 ).ToString();
     oYAxisTitleSize = oGR.MeasureString(sYAxisTitle,oFont_Axes);
     pYAxisTitleLocation.X = 2*iOffset - oYAxisTitleSize.Width;
     pYAxisTitleLocation.Y = (Single)iChartSizeHeight - 2*(Single)iOffset - ((Single)indx)*4*sYAxisSkip -
  oYAxisTitleSize.Height/2;
     oGR.DrawString(sYAxisTitle,oFont_Axes,Brushes.Black,pYAxisTitleLocation);
    }
    oGR.Restore(oGS);

   }

   private void PlaceTitleOnChart(String pTitle) {

    //Measure the size of the title based on the chosen font.
    SizeF oTitleSize = new SizeF();
    oTitleSize = oGR.MeasureString(pTitle,oFont_Title);

    //Based on the Width and Height of the title, place it gingerly on the top of the graph.
    PointF pTitleLocation = new PointF(iChartSizeWidth/2 - oTitleSize.Width/2,iOffset/2-oTitleSize.Height/2);

    //Place the title on the graph
    GraphicsState oGS = oGR.Save();
    oGR.Transform = new Matrix(1,0,0,1,0,0);
    oGR.DrawString(pTitle,oFont_Title,Brushes.Black,pTitleLocation);
    oGR.Restore(oGS);

   }

   //Draw the x- and y-axes on the chart.
   private void DrawChartAxes() {

    GraphicsState oGS = oGR.Save();
```

Introduction


```
   //Move the current origin (0,0) to where we want the origin of the axes intersection to be (60,60)
   oGR.TranslateTransform(60,60);

   //Draw X-Axis and Y-Axis
   oGR.DrawLine(oPenBlack,0,0,iAxisLength_X,0);
   oGR.DrawLine(oPenBlack,0,0,0,iAxisLength_Y);

   oGR.Restore(oGS);

  }

  //Place a smaller white rectangle within a grey rectangle.  The white rectangle will serve as the place to put
the chart.
  private void DrawChartArea() {

   Int32 iOffset=30;
   oGR.FillRectangle(Brushes.White,0+iOffset,0+iOffset,iChartSizeWidth-2*iOffset,iChartSizeHeight-2*iOffset);
   oGR.DrawRectangle(oPenBlack,0+iOffset,0+iOffset,iChartSizeWidth-2*iOffset,iChartSizeHeight-2*iOffset);

  }

  private void PullData(String pStockSymbol) {

   String sConn = @"Data Source=SCOTT-PC\SQLTEST;Initial Catalog=TESTDB;Integrated Security=SSPI;";
   using (SqlConnection oConn = new SqlConnection(sConn)) {

    oConn.Open();
    SqlCommand oCmd = new SqlCommand("SELECT MarketDate,MarketClose,ROW_NUMBER() OVER (ORDER BY MarketDate) AS
RNUM,COUNT(*) OVER () AS RTOT,MIN(MarketClose) OVER () AS MIN_CLOSE,MAX(MarketClose) OVER () AS
MAX_CLOSE,CONVERT(VARCHAR(4),YEAR(MarketDate)) AS MARKET_YEAR FROM STOCKDATA WHERE Symbol='" + StockSymbol + "'
ORDER BY MarketDate",oConn);
    SqlDataReader oDR = oCmd.ExecuteReader();
    oStockDataTable.Load(oDR);
```

Introduction

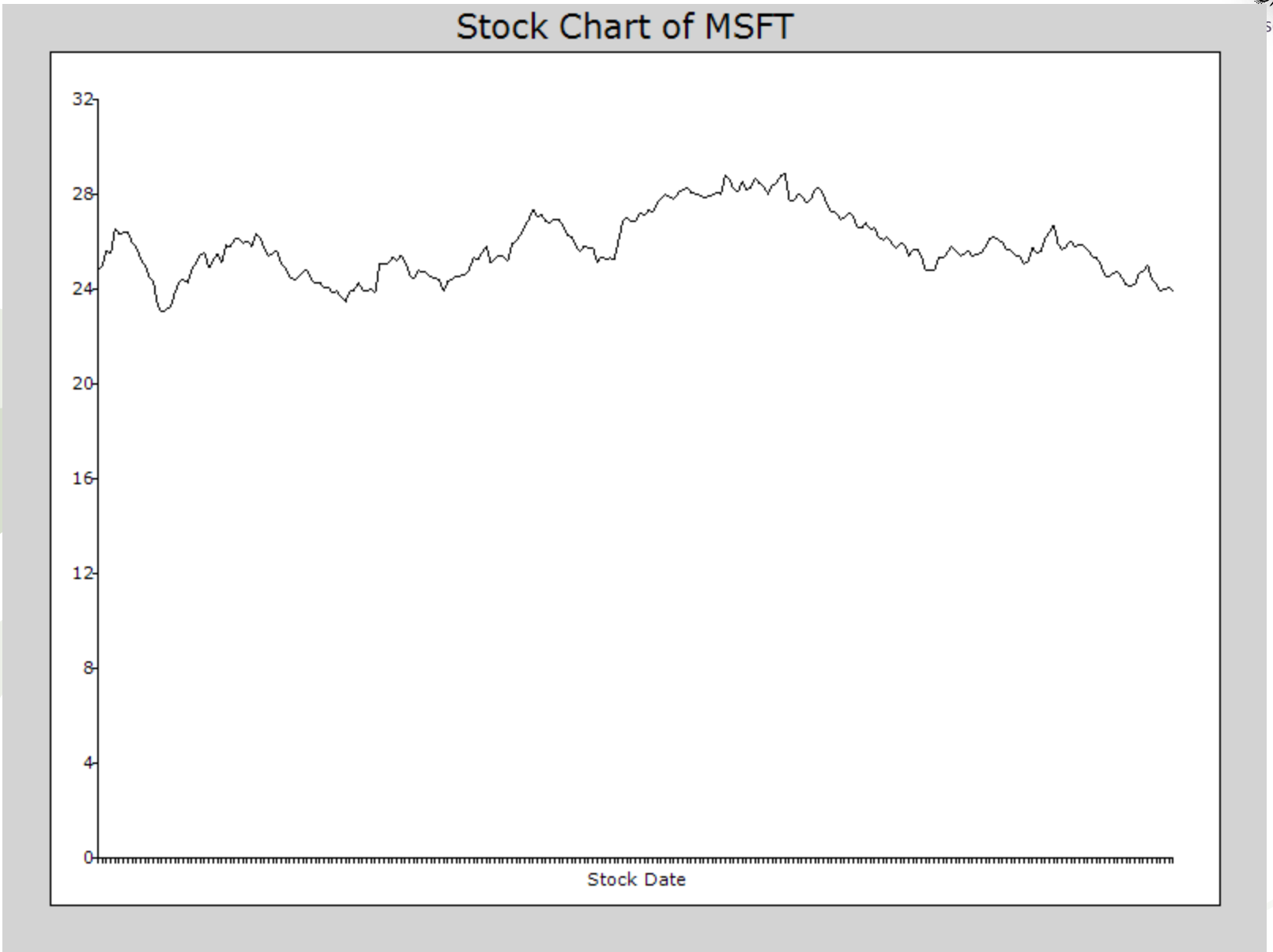

```
    //Print out data
    foreach(DataRow drThisRow in oStockDataTable.Rows) {
     Console.Write("\t");
     foreach(DataColumn dtThisColumn in oStockDataTable.Columns) {
      Console.Write("{0} ",drThisRow[dtThisColumn]);
     }
     Console.WriteLine();
    }

    //Fill in the max close value plus 10% for this stock
    sMaxClose = Convert.ToSingle(Math.Round( 1.10 * ( (Double) oStockDataTable.Rows[0]["MAX_CLOSE"] )));
    Console.WriteLine("sMaxClose={0}",sMaxClose);

    //Get the count of the data points for this stock.
    iNbrDataPoints = (Int32) oStockDataTable.Rows[0]["RTOT"];
    Console.WriteLine("iNbrDataPoints={0}",iNbrDataPoints);

    oConn.Close();
    oConn.Dispose();
   }

  }

  private void SaveGraph() {

   //Place oGR on the bitmap and save the bitmap to disk.
   oGR.Transform = new Matrix(1,0,0,1,0,0);
   oGR.DrawImage(oBM,new Point(0,0));
   oBM.Save(OutputFile,ImageFormat.Bmp);

  }
```

Introduction


```
 private void SetBackgroundColor() {

  oGR.Clear(Color.LightGray);

 }

}
```

```
class DrawingExample {

 public static void Main() {

  StockChart oSC = new StockChart("MSFT",@"C:\TEMP\TEST\MSFT.bmp");
  oSC.CreateChart();

 }

}
```

Introduction



The `System.Drawing` Namespace

- → Classes
  - → `Pens`

Pens

The `Pens` class represents pens for all the standard colors.

Properties

- AliceBlue - A system-defined Pen object with a width of 1
- AntiqueWhite - A system-defined Pen object with a width of 1
- Aqua - A system-defined Pen object with a width of 1
- Aquamarine - A system-defined Pen object with a width of 1
- Azure - A system-defined Pen object with a width of 1
- Beige - A system-defined Pen object with a width of 1
- Bisque - A system-defined Pen object with a width of 1
- Black - A system-defined Pen object with a width of 1
- BlanchedAlmond - A system-defined Pen object with a width of 1
- Blue - A system-defined Pen object with a width of 1
- BlueViolet - A system-defined Pen object with a width of 1
- Brown - A system-defined Pen object with a width of 1
- BurlyWood - A system-defined Pen object with a width of 1
- CadetBlue - A system-defined Pen object with a width of 1
- Chartreuse - A system-defined Pen object with a width of 1
- Chocolate - A system-defined Pen object with a width of 1
- Coral - A system-defined Pen object with a width of 1
- CornflowerBlue - A system-defined Pen object with a width of 1
- Cornsilk - A system-defined Pen object with a width of 1
- Crimson - A system-defined Pen object with a width of 1
- Cyan - A system-defined Pen object with a width of 1
- DarkBlue - A system-defined Pen object with a width of 1
- DarkCyan - A system-defined Pen object with a width of 1
- DarkGoldenrod - A system-defined Pen object with a width of 1
- DarkGray - A system-defined Pen object with a width of 1
- DarkGreen - A system-defined Pen object with a width of 1
- DarkKhaki - A system-defined Pen object with a width of 1
- DarkMagenta - A system-defined Pen object with a width of 1
- DarkOliveGreen - A system-defined Pen object with a width of 1
- DarkOrange - A system-defined Pen object with a width of 1

Properties (continued)

- DarkOrchid - A system-defined Pen object with a width of 1
- DarkRed - A system-defined Pen object with a width of 1
- DarkSalmon - A system-defined Pen object with a width of 1
- DarkSeaGreen - A system-defined Pen object with a width of 1
- DarkSlateBlue - A system-defined Pen object with a width of 1
- DarkSlateGray - A system-defined Pen object with a width of 1
- DarkTurquoise - A system-defined Pen object with a width of 1
- DarkViolet - A system-defined Pen object with a width of 1
- DeepPink - A system-defined Pen object with a width of 1
- DeepSkyBlue - A system-defined Pen object with a width of 1
- DimGray - A system-defined Pen object with a width of 1
- DodgerBlue - A system-defined Pen object with a width of 1
- Firebrick - A system-defined Pen object with a width of 1
- FloralWhite - A system-defined Pen object with a width of 1
- ForestGreen - A system-defined Pen object with a width of 1
- Fuchsia - A system-defined Pen object with a width of 1
- Gainsboro - A system-defined Pen object with a width of 1
- GhostWhite - A system-defined Pen object with a width of 1
- Gold - A system-defined Pen object with a width of 1
- Goldenrod - A system-defined Pen object with a width of 1
- Gray - A system-defined Pen object with a width of 1
- Green - A system-defined Pen object with a width of 1
- GreenYellow - A system-defined Pen object with a width of 1
- Honeydew - A system-defined Pen object with a width of 1
- HotPink - A system-defined Pen object with a width of 1
- IndianRed - A system-defined Pen object with a width of 1
- Indigo - A system-defined Pen object with a width of 1
- Ivory - A system-defined Pen object with a width of 1
- Khaki - A system-defined Pen object with a width of 1
- Lavender - A system-defined Pen object with a width of 1
- LavenderBlush - A system-defined Pen object with a width of 1
- LawnGreen - A system-defined Pen object with a width of 1

Properties (continued)

- LemonChiffon - A system-defined Pen object with a width of 1
- LightBlue - A system-defined Pen object with a width of 1
- LightCoral - A system-defined Pen object with a width of 1
- LightCyan - A system-defined Pen object with a width of 1
- LightGoldenrodYellow - A system-defined Pen object with a width of 1
- LightGray - A system-defined Pen object with a width of 1
- LightGreen - A system-defined Pen object with a width of 1
- LightPink - A system-defined Pen object with a width of 1
- LightSalmon - A system-defined Pen object with a width of 1
- LightSeaGreen - A system-defined Pen object with a width of 1
- LightSkyBlue - A system-defined Pen object with a width of 1
- LightSlateGray - A system-defined Pen object with a width of 1
- LightSteelBlue - A system-defined Pen object with a width of 1
- LightYellow - A system-defined Pen object with a width of 1
- Lime - A system-defined Pen object with a width of 1
- LimeGreen - A system-defined Pen object with a width of 1
- Linen - A system-defined Pen object with a width of 1
- Magenta - A system-defined Pen object with a width of 1
- Maroon - A system-defined Pen object with a width of 1
- MediumAquamarine - A system-defined Pen object with a width of 1
- MediumBlue - A system-defined Pen object with a width of 1
- MediumOrchid - A system-defined Pen object with a width of 1
- MediumPurple - A system-defined Pen object with a width of 1
- MediumSeaGreen - A system-defined Pen object with a width of 1
- MediumSlateBlue - A system-defined Pen object with a width of 1
- MediumSpringGreen - A system-defined Pen object with a width of 1
- MediumTurquoise - A system-defined Pen object with a width of 1
- MediumVioletRed - A system-defined Pen object with a width of 1
- MidnightBlue - A system-defined Pen object with a width of 1
- MintCream - A system-defined Pen object with a width of 1
- MistyRose - A system-defined Pen object with a width of 1
- Moccasin - A system-defined Pen object with a width of 1

Properties (continued)

- NavajoWhite - A system-defined Pen object with a width of 1
- Navy - A system-defined Pen object with a width of 1
- OldLace - A system-defined Pen object with a width of 1
- Olive - A system-defined Pen object with a width of 1
- OliveDrab - A system-defined Pen object with a width of 1
- Orange - A system-defined Pen object with a width of 1
- OrangeRed - A system-defined Pen object with a width of 1
- Orchid - A system-defined Pen object with a width of 1
- PaleGoldenrod - A system-defined Pen object with a width of 1
- PaleGreen - A system-defined Pen object with a width of 1
- PaleTurquoise - A system-defined Pen object with a width of 1
- PaleVioletRed - A system-defined Pen object with a width of 1
- PapayaWhip - A system-defined Pen object with a width of 1
- PeachPuff - A system-defined Pen object with a width of 1
- Peru - A system-defined Pen object with a width of 1
- Pink - A system-defined Pen object with a width of 1
- Plum - A system-defined Pen object with a width of 1
- PowderBlue - A system-defined Pen object with a width of 1
- Purple - A system-defined Pen object with a width of 1
- Red - A system-defined Pen object with a width of 1
- RosyBrown - A system-defined Pen object with a width of 1
- RoyalBlue - A system-defined Pen object with a width of 1
- SaddleBrown - A system-defined Pen object with a width of 1
- Salmon - A system-defined Pen object with a width of 1
- SandyBrown - A system-defined Pen object with a width of 1
- SeaGreen - A system-defined Pen object with a width of 1
- SeaShell - A system-defined Pen object with a width of 1
- Sienna - A system-defined Pen object with a width of 1
- Silver - A system-defined Pen object with a width of 1
- SkyBlue - A system-defined Pen object with a width of 1
- SlateBlue - A system-defined Pen object with a width of 1
- SlateGray - A system-defined Pen object with a width of 1

Properties (continued)

- Snow - A system-defined Pen object with a width of 1
- SpringGreen - A system-defined Pen object with a width of 1
- SteelBlue - A system-defined Pen object with a width of 1
- Tan - A system-defined Pen object with a width of 1
- Teal - A system-defined Pen object with a width of 1
- Thistle - A system-defined Pen object with a width of 1
- Tomato - A system-defined Pen object with a width of 1
- Transparent - A system-defined Pen object with a width of 1
- Turquoise - A system-defined Pen object with a width of 1
- Violet - A system-defined Pen object with a width of 1
- Wheat - A system-defined Pen object with a width of 1
- White - A system-defined Pen object with a width of 1
- WhiteSmoke - A system-defined Pen object with a width of 1
- Yellow - A system-defined Pen object with a width of 1
- YellowGreen - A system-defined Pen object with a width of 1

Note that you can see the colors associated with these names on the following webpage:

http://samples.msdn.microsoft.com/workshop/samples/author/dhtml/colors/ColorTable.htm.

The `System.Drawing` Namespace

- → Classes
  - → `SystemPens`

SystemPens

The `SystemPens` class represents a Pen that is the color of a Windows display element and that has a width of 1 pixel.

Properties

- ActiveBorder - Gets a Pen that is the color of the active window's border
- ActiveCaption - Gets a Pen that is the color of the background of the active window's title bar
- ActiveCaptionText - Gets a Pen that is the color of the text in the active window's title bar
- AppWorkspace - Gets a Pen that is the color of the application workspace
- ButtonFace - Gets a Pen that is the face color of a 3-D element
- ButtonHighlight - Gets a Pen that is the highlight color of a 3-D element
- ButtonShadow - Gets a Pen that is the shadow color of a 3-D element
- Control - Gets a Pen that is the face color of a 3-D element
- ControlDark - Gets a Pen that is the shadow color of a 3-D element
- ControlDarkDark - Gets a Pen that is the dark shadow color of a 3-D element
- ControlLight - Gets a Pen that is the light color of a 3-D element
- ControlLightLight - Gets a Pen that is the highlight color of a 3-D element
- ControlText - Gets a Pen that is the color of text in a 3-D element
- Desktop - Gets a Pen that is the color of the Windows desktop
- GradientActiveCaption - Gets a Pen that is the lightest color in the color gradient of an active window's title bar
- GradientInactiveCaption - Gets a Pen that is the lightest color in the color gradient of an inactive window's title bar
- GrayText - Gets a Pen that is the color of dimmed text
- Highlight - Gets a Pen that is the color of the background of selected items
- HighlightText - Gets a Pen that is the color of the text of selected items
- HotTrack - Gets a Pen that is the color used to designate a hot-tracked item
- InactiveBorder - Gets a Pen is the color of the border of an inactive window
- InactiveCaption - Gets a Pen that is the color of the title bar caption of an inactive window
- InactiveCaptionText - Gets a Pen that is the color of the text in an inactive window's title bar
- Info - Gets a Pen that is the color of the background of a ToolTip
- InfoText - Gets a Pen that is the color of the text of a ToolTip
- Menu - Gets a Pen that is the color of a menu's background
- MenuBar - Gets a Pen that is the color of the background of a menu bar

Properties (continued)

- MenuHighlight - Gets a Pen that is the color used to highlight menu items when the menu appears as a flat menu
- MenuText - Gets a Pen that is the color of a menu's text
- ScrollBar - Gets a Pen that is the color of the background of a scroll bar
- Window - Gets a Pen that is the color of the background in the client area of a window
- WindowFrame - Gets a Pen that is the color of a window frame
- WindowText - Gets a Pen that is the color of the text in the client area of a window

Methods

- FromSystemColor - Creates a Pen from the specified Color structure.

The `System.Drawing` Namespace

- → Classes
  - → `Pen`

Pen

The `Pen` class defines an object used to draw lines and curves. According to Microsoft's website: *Pen draws a line of specified width and style. Use the DashStyle property to draw several varieties of dashed lines. The line drawn by a Pen can be filled in a variety of fill styles, including solid colors and textures. The fill style depends on brush or texture that is used as the fill object.*

Constructors

- Pen(Brush) - Initializes a new instance of the Pen class with the specified Brush.
- Pen(Color) - Initializes a new instance of the Pen class with the specified color.
- Pen(Brush, Single) - Initializes a new instance of the Pen class with the specified Brush and Width.
- Pen(Color, Single) - Initializes a new instance of the Pen class with the specified Color and Width properties.

Properties

- Alignment - Gets or sets the alignment for this Pen.
- Brush - Gets or sets the Brush that determines attributes of this Pen.
- Color - Gets or sets the color of this Pen.
- CompoundArray - Gets or sets an array of values that specifies a compound pen. A compound pen draws a compound line made up of parallel lines and spaces.
- CustomEndCap - Gets or sets a custom cap to use at the end of lines drawn with this Pen.
- CustomStartCap - Gets or sets a custom cap to use at the beginning of lines drawn with this Pen.
- DashCap - Gets or sets the cap style used at the end of the dashes that make up dashed lines drawn with this Pen.
- DashOffset - Gets or sets the distance from the start of a line to the beginning of a dash pattern.
- DashPattern - Gets or sets an array of custom dashes and spaces.
- DashStyle - Gets or sets the style used for dashed lines drawn with this Pen.
- EndCap - Gets or sets the cap style used at the end of lines drawn with this Pen.
- LineJoin - Gets or sets the join style for the ends of two consecutive lines drawn with this Pen.
- MiterLimit - Gets or sets the limit of the thickness of the join on a mitered corner.
- PenType - Gets the style of lines drawn with this Pen.
- StartCap - Gets or sets the cap style used at the beginning of lines drawn with this Pen.
- Transform - Gets or sets a copy of the geometric transformation for this Pen.
- Width - Gets or sets the width of this Pen, in units of the Graphics object used for drawing.

Methods

- Clone - Creates an exact copy of this Pen.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this Pen.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- MultiplyTransform(Matrix) - Multiplies the transformation matrix for this Pen by the specified Matrix.
- MultiplyTransform(Matrix, MatrixOrder) - Multiplies the transformation matrix for this Pen by the specified Matrix in the specified order.
- ResetTransform - Resets the geometric transformation matrix for this Pen to identity.
- RotateTransform(Single) - Rotates the local geometric transformation by the specified angle. This method prepends the rotation to the transformation.
- RotateTransform(Single, MatrixOrder) - Rotates the local geometric transformation by the specified angle in the specified order.
- ScaleTransform(Single, Single) - Scales the local geometric transformation by the specified factors. This method prepends the scaling matrix to the transformation.
- ScaleTransform(Single, Single, MatrixOrder) - Scales the local geometric transformation by the specified factors in the specified order.
- SetLineCap - Sets the values that determine the style of cap used to end lines drawn by this Pen.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- TranslateTransform(Single, Single) - Translates the local geometric transformation by the specified dimensions. This method prepends the translation to the transformation.
- TranslateTransform(Single, Single, MatrixOrder) - Translates the local geometric transformation by the specified dimensions in the specified order.

The `System.Drawing` Namespace

- → Classes
  - → `Brushes`

Brushes

The `Brushes` class defines brushes for all the standard colors. According to Microsoft's website: *The Brushes class contains static read-only properties that return a Brush object of the color indicated by the property name. You typically do not have to explicitly dispose of the brush returned by a property in this class, unless it is used to construct a new brush.*

Properties

- See the list of Properties for the Pens class.

The `System.Drawing` Namespace

- → Classes
  - → `SolidBrush`

SolidBrush

The `SolidBrush` class defines a brush of a single color. Brushes are used to fill graphics shapes, such as rectangles, ellipses, pies, polygons, and paths. According to Microsoft's website: *This class inherits from Brush.*

Constructors

- SolidBrush - Initializes a new SolidBrush object of the specified color.

Properties

- Color - Gets or sets the color of this SolidBrush object.

Methods

- Clone - Creates an exact copy of this SolidBrush object. (Overrides Brush.Clone().)
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Brush object. (Inherited from Brush.)
- Dispose(Boolean) - Releases the unmanaged resources used by the Brush and optionally releases the managed resources. (Inherited from Brush.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Brush.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetNativeBrush - In a derived class, sets a reference to a GDI+ brush object. (Inherited from Brush.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `TextureBrush`

TextureBrush

The `TextureBrush` class represents a Brush object that uses an image to fill the interior of a shape.

Constructors

- TextureBrush(Image) - Initializes a new TextureBrush object that uses the specified image.
- TextureBrush(Image, WrapMode) - Initializes a new TextureBrush object that uses the specified image and wrap mode.
- TextureBrush(Image, Rectangle) - Initializes a new TextureBrush object that uses the specified image and bounding rectangle.
- TextureBrush(Image, RectangleF) - Initializes a new TextureBrush object that uses the specified image and bounding rectangle.
- TextureBrush(Image, WrapMode, Rectangle) - Initializes a new TextureBrush object that uses the specified image, wrap mode, and bounding rectangle.
- TextureBrush(Image, WrapMode, RectangleF) - Initializes a new TextureBrush object that uses the specified image, wrap mode, and bounding rectangle.
- TextureBrush(Image, Rectangle, ImageAttributes) - Initializes a new TextureBrush object that uses the specified image, bounding rectangle, and image attributes.
- TextureBrush(Image, RectangleF, ImageAttributes) - Initializes a new TextureBrush object that uses the specified image, bounding rectangle, and image attributes.

Properties

- Image - Gets the Image object associated with this TextureBrush object.
- Transform - Gets or sets a copy of the Matrix object that defines a local geometric transformation for the image associated with this TextureBrush object.
- WrapMode - Gets or sets a WrapMode enumeration that indicates the wrap mode for this TextureBrush object.

Methods

- Clone - Creates an exact copy of this TextureBrush object. (Overrides Brush.Clone().)
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Brush object. (Inherited from Brush.)
- Dispose(Boolean) - Releases the unmanaged resources used by the Brush and optionally releases the managed resources. (Inherited from Brush.)

Methods (continued)

- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Brush.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- MultiplyTransform(Matrix) - Multiplies the Matrix object that represents the local geometric transformation of this TextureBrush object by the specified Matrix object by prepending the specified Matrix object.
- MultiplyTransform(Matrix, MatrixOrder) - Multiplies the Matrix object that represents the local geometric transformation of this TextureBrush object by the specified Matrix object in the specified order.
- ResetTransform - Resets the Transform property of this TextureBrush object to identity.
- RotateTransform(Single) - Rotates the local geometric transformation of this TextureBrush object by the specified amount. This method prepends the rotation to the transformation.
- RotateTransform(Single, MatrixOrder) - Rotates the local geometric transformation of this TextureBrush object by the specified amount in the specified order.
- ScaleTransform(Single, Single) - Scales the local geometric transformation of this TextureBrush object by the specified amounts. This method prepends the scaling matrix to the transformation.
- ScaleTransform(Single, Single, MatrixOrder) - Scales the local geometric transformation of this TextureBrush object by the specified amounts in the specified order.
- SetNativeBrush - In a derived class, sets a reference to a GDI+ brush object. (Inherited from Brush.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- TranslateTransform(Single, Single) - Translates the local geometric transformation of this TextureBrush object by the specified dimensions. This method prepends the translation to the transformation.
- TranslateTransform(Single, Single, MatrixOrder) - Translates the local geometric transformation of this TextureBrush object by the specified dimensions in the specified order.

The `System.Drawing` Namespace

- → Classes
  - → `SystemBrushes`

SystemBrushes

The `SystemBrushes` class represents a Brush object that uses an image to fill the interior of a shape. According to Microsoft's website: *This class inherits from Brush.*

Properties

- ActiveBorder - Gets a SolidBrush that is the color of the active window's border
- ActiveCaption - Gets a SolidBrush that is the color of the background of the active window's title bar
- ActiveCaptionText - Gets a SolidBrush that is the color of the text in the active window's title bar
- AppWorkspace - Gets a SolidBrush that is the color of the application workspace
- ButtonFace - Gets a SolidBrush that is the face color of a 3-D element
- ButtonHighlight - Gets a SolidBrush that is the highlight color of a 3-D element
- ButtonShadow - Gets a SolidBrush that is the shadow color of a 3-D element
- Control - Gets a SolidBrush that is the face color of a 3-D element
- ControlDark - Gets a SolidBrush that is the shadow color of a 3-D element
- ControlDarkDark - Gets a SolidBrush that is the dark shadow color of a 3-D element
- ControlLight - Gets a SolidBrush that is the light color of a 3-D element
- ControlLightLight - Gets a SolidBrush that is the highlight color of a 3-D element
- ControlText - Gets a SolidBrush that is the color of text in a 3-D element
- Desktop - Gets a SolidBrush that is the color of the desktop
- GradientActiveCaption - Gets a SolidBrush that is the lightest color in the color gradient of an active window's title bar
- GradientInactiveCaption - Gets a SolidBrush that is the lightest color in the color gradient of an inactive window's title bar
- GrayText - Gets a SolidBrush that is the color of dimmed text
- Highlight - Gets a SolidBrush that is the color of the background of selected items
- HighlightText - Gets a SolidBrush that is the color of the text of selected items
- HotTrack - Gets a SolidBrush that is the color used to designate a hot-tracked item
- InactiveBorder - Gets a SolidBrush that is the color of an inactive window's border
- InactiveCaption - Gets a SolidBrush that is the color of the background of an inactive window's title bar
- InactiveCaptionText - Gets a SolidBrush that is the color of the text in an inactive window's title bar
- Info - Gets a SolidBrush that is the color of the background of a ToolTip
- InfoText - Gets a SolidBrush that is the color of the text of a ToolTip

SystemBrushes

Properties (continued)

- Menu - Gets a SolidBrush that is the color of a menu's background
- MenuBar - Gets a SolidBrush that is the color of the background of a menu bar
- MenuHighlight - Gets a SolidBrush that is the color used to highlight menu items when the menu appears as a flat menu
- MenuText - Gets a SolidBrush that is the color of a menu's text
- ScrollBar - Gets a SolidBrush that is the color of the background of a scroll bar
- Window - Gets a SolidBrush that is the color of the background in the client area of a window
- WindowFrame - Gets a SolidBrush that is the color of a window frame
- WindowText - Gets a SolidBrush that is the color of the text in the client area of a window

Methods

- FromSystemColor - Creates a Brush from the specified Color structure.

The `System.Drawing` Namespace

- → Classes
  - → `Brush`

Brush

The `Brush` class defines objects used to fill the interiors of graphical shapes such as rectangles, ellipses, pies, polygons, and paths. According to Microsoft's website: *This is an abstract base class and cannot be instantiated. To create a brush object, use classes derived from Brush, such as SolidBrush, TextureBrush, and LinearGradientBrush.*

Constructors

- Brush - Initializes a new instance of the Brush class.

Methods

- Clone - When overridden in a derived class, creates an exact copy of this Brush.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Brush object.
- Dispose(Boolean) - Releases the unmanaged resources used by the Brush and optionally releases the managed resources.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Overrides Object.Finalize().)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetNativeBrush - In a derived class, sets a reference to a GDI+ brush object.
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `HatchBrush`

HatchBrush

The `HatchBrush` class defines a rectangular brush with a hatch style, a foreground color, and a background color. According to Microsoft's website: *A hatch pattern is made from two colors: one defined by the BackgroundColor, which fills the background and one for the lines that form the pattern over the background defined by the ForegroundColor property. The HatchStyle property defines what type of pattern the brush has and can be any value from the HatchStyle enumeration. There are more than fifty elements in the HatchStyle enumeration.*

Constructors

- HatchBrush(HatchStyle, Color) Initializes a new instance of the HatchBrush class with the specified HatchStyle enumeration and foreground color.
- HatchBrush(HatchStyle, Color, Color) Initializes a new instance of the HatchBrush class with the specified HatchStyle enumeration, foreground color, and background color.

Properties

- BackgroundColor Gets the color of spaces between the hatch lines drawn by this HatchBrush object.
- ForegroundColor Gets the color of hatch lines drawn by this HatchBrush object.
- HatchStyle Gets the hatch style of this HatchBrush object.

Methods

- Clone Creates an exact copy of this HatchBrush object. (Overrides Brush.Clone().)
- CreateObjRef Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() Releases all resources used by this Brush object. (Inherited from Brush.)
- Dispose(Boolean) Releases the unmanaged resources used by the Brush and optionally releases the managed resources. (Inherited from Brush.)
- Equals(Object) Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Brush.)

Methods (continued)

- GetHashCode Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetNativeBrush In a derived class, sets a reference to a GDI+ brush object. (Inherited from Brush.)
- ToString Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `LinearGradientBrush`

LinearGradientBrush

The `LinearGradientBrush` class encapsulates a Brush with a linear gradient. According to Microsoft's website: *This class encapsulates both two-color gradients and custom multicolor gradients. All linear gradients are defined along a line specified either by the width of a rectangle or by two points. By default, a two-color linear gradient is an even horizontal linear blend from the starting color to the ending color along the specified line. Customize the blend pattern using the Blend class, the SetSigmaBellShape methods, or the SetBlendTriangularShape methods. Customize the direction of the gradient by specifying the LinearGradientMode enumeration or the angle in the constructor. Use the InterpolationColors property to create a multicolor gradient. The Transform property specifies a local geometric transform applied to the gradient.*

Constructors

- LinearGradientBrush(Point, Point, Color, Color) - Initializes a new instance of the LinearGradientBrush class with the specified points and colors.
- LinearGradientBrush(PointF, PointF, Color, Color) - Initializes a new instance of the LinearGradientBrush class with the specified points and colors.
- LinearGradientBrush(Rectangle, Color, Color, LinearGradientMode) - Creates a new instance of the LinearGradientBrush class based on a rectangle, starting and ending colors, and orientation.
- LinearGradientBrush(Rectangle, Color, Color, Single) - Creates a new instance of the LinearGradientBrush class based on a rectangle, starting and ending colors, and an orientation angle.
- LinearGradientBrush(RectangleF, Color, Color, LinearGradientMode) - Creates a new instance of the LinearGradientBrush based on a rectangle, starting and ending colors, and an orientation mode.
- LinearGradientBrush(RectangleF, Color, Color, Single) - Creates a new instance of the LinearGradientBrush class based on a rectangle, starting and ending colors, and an orientation angle.
- LinearGradientBrush(Rectangle, Color, Color, Single, Boolean) - Creates a new instance of the LinearGradientBrush class based on a rectangle, starting and ending colors, and an orientation angle.
- LinearGradientBrush(RectangleF, Color, Color, Single, Boolean) - Creates a new instance of the LinearGradientBrush class based on a rectangle, starting and ending colors, and an orientation angle.

.

LinearGradientBrush

Properties

- Blend - Gets or sets a Blend that specifies positions and factors that define a custom falloff for the gradient.
- GammaCorrection - Gets or sets a value indicating whether gamma correction is enabled for this LinearGradientBrush.
- InterpolationColors - Gets or sets a ColorBlend that defines a multicolor linear gradient.
- LinearColors - Gets or sets the starting and ending colors of the gradient.
- Rectangle - Gets a rectangular region that defines the starting and ending points of the gradient.
- Transform - Gets or sets a copy Matrix that defines a local geometric transform for this LinearGradientBrush.
- WrapMode - Gets or sets a WrapMode enumeration that indicates the wrap mode for this LinearGradientBrush.

Methods

- Clone - Creates an exact copy of this LinearGradientBrush. (Overrides Brush.Clone().)
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Brush object. (Inherited from Brush.)
- Dispose(Boolean) - Releases the unmanaged resources used by the Brush and optionally releases the managed resources. (Inherited from Brush.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Brush.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- MultiplyTransform(Matrix) - Multiplies the Matrix that represents the local geometric transform of this LinearGradientBrush by the specified Matrix by prepending the specified Matrix.
- MultiplyTransform(Matrix, MatrixOrder) - Multiplies the Matrix that represents the local geometric transform of this LinearGradientBrush by the specified Matrix in the specified order.
- ResetTransform - Resets the Transform property to identity.

Methods

- RotateTransform(Single) - Rotates the local geometric transform by the specified amount. This method prepends the rotation to the transform.
- RotateTransform(Single, MatrixOrder) - Rotates the local geometric transform by the specified amount in the specified order.
- ScaleTransform(Single, Single) - Scales the local geometric transform by the specified amounts. This method prepends the scaling matrix to the transform.
- ScaleTransform(Single, Single, MatrixOrder) - Scales the local geometric transform by the specified amounts in the specified order.
- SetBlendTriangularShape(Single) - Creates a linear gradient with a center color and a linear falloff to a single color on both ends.
- SetBlendTriangularShape(Single, Single) - Creates a linear gradient with a center color and a linear falloff to a single color on both ends.
- SetNativeBrush - In a derived class, sets a reference to a GDI+ brush object. (Inherited from Brush.)
- SetSigmaBellShape(Single) - Creates a gradient falloff based on a bell-shaped curve.
- SetSigmaBellShape(Single, Single) - Creates a gradient falloff based on a bell-shaped curve.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- TranslateTransform(Single, Single) - Translates the local geometric transform by the specified dimensions. This method prepends the translation to the transform.
- TranslateTransform(Single, Single, MatrixOrder) - Translates the local geometric transform by the specified dimensions in the specified order.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `Blend`

Blend

The `Blend` class defines a blend pattern for a LinearGradientBrush object. According to Microsoft's website: *Gradients are commonly used to smoothly shade the interiors of shapes. A blend pattern is defined by two arrays (Factors and Positions) that each contain the same number of elements. Each element of the Positions array represents a proportion of the distance along the gradient line. Each element of the Factors array represents the proportion of the starting and ending colors in the gradient blend at the position along the gradient line represented by the corresponding element in the Positions array. For example, if corresponding elements of the Positions and Factors arrays are 0.2 and 0.3, respectively, for a linear gradient from blue to red along a 100-pixel line, the color 20 pixels along that line (20 percent of the distance) consists of 30 percent blue and 70 percent red.*

Constructors

- Blend() - Initializes a new instance of the Blend class.
- Blend(Int32) - Initializes a new instance of the Blend class with the specified number of factors and positions.

Properties

- Factors - Gets or sets an array of blend factors for the gradient.
- Positions - Gets or sets an array of blend positions for the gradient.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `ColorBlend`

ColorBlend

The `ColorBlend` class defines arrays of colors and positions used for interpolating color blending in a multicolor gradient.

Constructors

- ColorBlend() - Initializes a new instance of the ColorBlend class.
- ColorBlend(Int32) - Initializes a new instance of the ColorBlend class with the specified number of colors and positions.

Properties

- Colors - Gets or sets an array of colors that represents the colors to use at corresponding positions along a gradient.
- Positions - Gets or sets the positions along a gradient line.

The `System.Drawing` Namespace

- → Classes
  - → `PathGradientBrush`

PathGradientBrush

The `PathGradientBrush` class encapsulates a Brush object that fills the interior of a GraphicsPath object with a gradient. According to Microsoft's website: *The color gradient is a smooth shading of colors from the center point of the path to the outside boundary edge of the path. Blend factors, positions, and style affect where the gradient starts and ends, and how fast it changes shade. Path gradient brushes do not obey the SmoothingMode property of the Graphics object used to do the drawing. Areas filled using a PathGradientBrush object are rendered the same way (aliased) regardless of the smoothing mode.*

Constructors

- PathGradientBrush(GraphicsPath) - Initializes a new instance of the PathGradientBrush class with the specified path.
- PathGradientBrush(Point[]) - Initializes a new instance of the PathGradientBrush class with the specified points.
- PathGradientBrush(PointF[]) - Initializes a new instance of the PathGradientBrush class with the specified points.
- PathGradientBrush(Point[], WrapMode) - Initializes a new instance of the PathGradientBrush class with the specified points and wrap mode.
- PathGradientBrush(PointF[], WrapMode) - Initializes a new instance of the PathGradientBrush class with the specified points and wrap mode.

Properties

- Blend - Gets or sets a Blend that specifies positions and factors that define a custom falloff for the gradient.
- CenterColor - Gets or sets the color at the center of the path gradient.
- CenterPoint - Gets or sets the center point of the path gradient.
- FocusScales - Gets or sets the focus point for the gradient falloff.
- InterpolationColors - Gets or sets a ColorBlend that defines a multicolor linear gradient.
- Rectangle - Gets a bounding rectangle for this PathGradientBrush.
- SurroundColors - Gets or sets an array of colors that correspond to the points in the path this PathGradientBrush fills.
- Transform - Gets or sets a copy of the Matrix that defines a local geometric transform for this PathGradientBrush.
- WrapMode - Gets or sets a WrapMode that indicates the wrap mode for this PathGradientBrush.

PathGradientBrush

Methods

- Clone - Creates an exact copy of this PathGradientBrush. (Overrides Brush.Clone().)
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Brush object. (Inherited from Brush.)
- Dispose(Boolean) - Releases the unmanaged resources used by the Brush and optionally releases the managed resources. (Inherited from Brush.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Brush.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- MultiplyTransform(Matrix) - Updates the brush's transformation matrix with the product of brush's transformation matrix multiplied by another matrix.
- MultiplyTransform(Matrix, MatrixOrder) - Updates the brush's transformation matrix with the product of the brush's transformation matrix multiplied by another matrix.
- ResetTransform - Resets the Transform property to identity.
- RotateTransform(Single) - Rotates the local geometric transform by the specified amount. This method prepends the rotation to the transform.
- RotateTransform(Single, MatrixOrder) - Rotates the local geometric transform by the specified amount in the specified order.
- ScaleTransform(Single, Single) - Scales the local geometric transform by the specified amounts. This method prepends the scaling matrix to the transform.
- ScaleTransform(Single, Single, MatrixOrder) - Scales the local geometric transform by the specified amounts in the specified order.
- SetBlendTriangularShape(Single) - Creates a gradient with a center color and a linear falloff to one surrounding color.
- SetBlendTriangularShape(Single, Single) - Creates a gradient with a center color and a linear falloff to each surrounding color.

Methods (continued)

- SetNativeBrush - In a derived class, sets a reference to a GDI+ brush object. (Inherited from Brush.)
- SetSigmaBellShape(Single) - Creates a gradient brush that changes color starting from the center of the path outward to the path's boundary. The transition from one color to another is based on a bell-shaped curve.
- SetSigmaBellShape(Single, Single) - Creates a gradient brush that changes color starting from the center of the path outward to the path's boundary. The transition from one color to another is based on a bell-shaped curve.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- TranslateTransform(Single, Single) - Applies the specified translation to the local geometric transform. This method prepends the translation to the transform.
- TranslateTransform(Single, Single, MatrixOrder) - Applies the specified translation to the local geometric transform in the specified order.

The `System.Drawing` Namespace

- → Classes
  - → `FontFamily`

FontFamily

The `FontFamily` class encapsulates a defines a group of type faces having a similar basic design and certain variations in styles.

Constructors

- FontFamily(GenericFontFamilies) - Initializes a new FontFamily from the specified generic font family.
- FontFamily(String) - Initializes a new FontFamily with the specified name.
- FontFamily(String, FontCollection) - Initializes a new FontFamily in the specified FontCollection with the specified name.

Properties

- Families - Returns an array that contains all the FontFamily objects associated with the current graphics context.
- GenericMonospace - Gets a generic monospace FontFamily.
- GenericSansSerif - Gets a generic sans serif FontFamily object.
- GenericSerif - Gets a generic serif FontFamily.
- Name - Gets the name of this FontFamily.

Methods

- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this FontFamily.
- Equals - Indicates whether the specified object is a FontFamily and is identical to this FontFamily. (Overrides Object.Equals(Object).)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetCellAscent - Returns the cell ascent, in design units, of the FontFamily of the specified style.
- GetCellDescent - Returns the cell descent, in design units, of the FontFamily of the specified style.
- GetEmHeight - Gets the height, in font design units, of the em square for the specified style.
- GetFamilies - Obsolete. Returns an array that contains all the FontFamily objects available for the specified graphics context.
- GetHashCode - Gets a hash code for this FontFamily. (Overrides Object.GetHashCode().)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)

Methods (continued)

- GetLineSpacing - Returns the line spacing, in design units, of the FontFamily of the specified style. The line spacing is the vertical distance between the base lines of two consecutive lines of text.
- GetName - Returns the name, in the specified language, of this FontFamily.
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- IsStyleAvailable - Indicates whether the specified FontStyle enumeration is available.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- ToString - Converts this FontFamily to a human-readable string representation. (Overrides Object.ToString().)

The `System.Drawing` Namespace

- → Classes
  - → `SystemFonts`

SystemFonts

The `SystemFonts` class specifies the fonts used to display text in Windows display elements. According to Microsoft's website: *Each property of the SystemFonts class returns a Font used to display text in a particular Windows display element, such as the title bar of a window. These fonts reflect the current settings in Control Panel.*

Properties

- CaptionFont - Gets a Font that is used to display text in the title bars of windows.
- DefaultFont - Gets the default font that applications can use for dialog boxes and forms.
- DialogFont - Gets a font that applications can use for dialog boxes and forms.
- IconTitleFont - Gets a Font that is used for icon titles.
- MenuFont - Gets a Font that is used for menus.
- MessageBoxFont - Gets a Font that is used for message boxes.
- SmallCaptionFont - Gets a Font that is used to display text in the title bars of small windows, such as tool windows.
- StatusFont - Gets a Font that is used to display text in the status bar.

The `System.Drawing` Namespace

- → Classes
  - → `Font`

Font

The `Font` class specifies defines a particular format for text, including font face, size, and style attributes. According to Microsoft's website: *For more information about how to construct fonts, see How to: Construct Font Families and Fonts. Windows Forms applications support TrueType fonts and have limited support for OpenType fonts. If you attempt to use a font that is not supported, or the font is not installed on the machine that is running the application, the Microsoft Sans Serif font will be substituted.*

Constructors

- Font(Font, FontStyle) - Initializes a new Font that uses the specified existing Font and FontStyle enumeration.
- Font(FontFamily, Single) - Initializes a new Font using a specified size.
- Font(String, Single) - Initializes a new Font using a specified size.
- Font(FontFamily, Single, FontStyle) - Initializes a new Font using a specified size and style.
- Font(FontFamily, Single, GraphicsUnit) - Initializes a new Font using a specified size and unit. Sets the style to FontStyle.Regular.
- Font(String, Single, FontStyle) - Initializes a new Font using a specified size and style.
- Font(String, Single, GraphicsUnit) - Initializes a new Font using a specified size and unit. The style is set to FontStyle.Regular.
- Font(FontFamily, Single, FontStyle, GraphicsUnit) - Initializes a new Font using a specified size, style, and unit.
- Font(String, Single, FontStyle, GraphicsUnit) - Initializes a new Font using a specified size, style, and unit.
- Font(FontFamily, Single, FontStyle, GraphicsUnit, Byte) - Initializes a new Font using a specified size, style, unit, and character set.
- Font(String, Single, FontStyle, GraphicsUnit, Byte) - Initializes a new Font using a specified size, style, unit, and character set.
- Font(FontFamily, Single, FontStyle, GraphicsUnit, Byte, Boolean) - Initializes a new Font using a specified size, style, unit, and character set.
- Font(String, Single, FontStyle, GraphicsUnit, Byte, Boolean) - Initializes a new Font using the specified size, style, unit, and character set.

Font

Properties

- Bold - Gets a value that indicates whether this Font is bold.
- FontFamily - Gets the FontFamily associated with this Font.
- GdiCharSet - Gets a byte value that specifies the GDI character set that this Font uses.
- GdiVerticalFont - Gets a Boolean value that indicates whether this Font is derived from a GDI vertical font.
- Height - Gets the line spacing of this font.
- IsSystemFont - Gets a value indicating whether the font is a member of SystemFonts.
- Italic - Gets a value that indicates whether this font has the italic style applied.
- Name - Gets the face name of this Font.
- OriginalFontName - Infrastructure. Gets the name of the font originally specified.
- Size - Gets the em-size of this Font measured in the units specified by the Unit property.
- SizeInPoints - Gets the em-size, in points, of this Font.
- Strikeout - Gets a value that indicates whether this Font specifies a horizontal line through the font.
- Style - Gets style information for this Font.
- SystemFontName - Gets the name of the system font if the IsSystemFont property returns true.
- Underline - Gets a value that indicates whether this Font is underlined.
- Unit - Gets the unit of measure for this Font.

Methods

- Clone - Creates an exact copy of this Font.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this Font.
- Equals - Indicates whether the specified object is a Font and has the same FontFamily, GdiVerticalFont, GdiCharSet, Style, Size, and Unit property values as this Font. (Overrides Object.Equals(Object).)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- FromHdc - Creates a Font from the specified Windows handle to a device context.
- FromHfont - Creates a Font from the specified Windows handle.
- FromLogFont(Object) - Creates a Font from the specified GDI logical font (LOGFONT) structure.
- FromLogFont(Object, IntPtr) - Creates a Font from the specified GDI logical font (LOGFONT) structure.
- GetHashCode - Gets the hash code for this Font. (Overrides Object.GetHashCode().)

Methods (continued)

- GetHeight() - Returns the line spacing, in pixels, of this font.
- GetHeight(Graphics) - Returns the line spacing, in the current unit of a specified Graphics, of this font.
- GetHeight(Single) - Returns the height, in pixels, of this Font when drawn to a device with the specified vertical resolution.
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- ToHfont - Returns a handle to this Font.
- ToLogFont(Object) - Creates a GDI logical font (LOGFONT) structure from this Font.
- ToLogFont(Object, Graphics) - Creates a GDI logical font (LOGFONT) structure from this Font.
- ToString - Returns a human-readable string representation of this Font. (Overrides Object.ToString().)

The `System.Drawing` Namespace

- → Classes
  - → `FontConverter`

FontConverter

The `FontConverter` class converts Font objects from one data type to another.

Constructors

- FontConverter - Initializes a new FontConverter object.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines whether this converter can convert an object in the specified source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Gets a value indicating whether this converter can convert an object to the given destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the specified object to the native type of the converter. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to another type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)

Methods (continued)

- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an object of this type by using a specified set of property values for the object. (Overrides TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary).)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows the FontConverter to attempt to free resources and perform other cleanup operations before the FontConverter is reclaimed by garbage collection. (Overrides Object.Finalize().)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Determines whether changing a value on this object should require a call to the CreateInstance method to create a new value. (Overrides TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext).)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Retrieves the set of properties for this type. By default, a type does not have any properties to return. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Determines whether this object supports properties. The default is false. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)

Methods (continued)

- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `ColorConverter`

ColorConverter

The `ColorConverter` class converts colors from one data type to another. Access this class through the TypeDescriptor. From Microsoft's website: *When converting from a string to a Color the ColorConverter expects the unqualified color name; otherwise, an exception will occur in the conversion process. For example, you should pass "Blue", not "System.Drawing.Color.Blue" or "Color.Blue", to the ConvertFrom method.*

Constructors

- ColorConverter - Initializes a new instance of the ColorConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines if this converter can convert an object in the given source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Returns a value indicating whether this converter can convert an object to the given destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the given object to the converter's native type. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)

Methods (continued)

- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to another type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of the type that this TypeConverter is associated with, using the specified context, given a set of property values for the object. (Inherited from TypeConverter.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Returns whether changing a value on this object requires a call to CreateInstance to create a new value, using the specified context. (Inherited from TypeConverter.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)

Methods (continued)

- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context and attributes. (Inherited from TypeConverter.)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Returns whether this object supports properties, using the specified context. (Inherited from TypeConverter.)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Retrieves a collection containing a set of standard values for the data type for which this validator is designed. This will return null if the data type does not support a standard set of values. (Overrides TypeConverter.GetStandardValues(ITypeDescriptorContext).)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Determines if this object supports a standard set of values that can be chosen from a list. (Overrides TypeConverter.GetStandardValuesSupported(ITypeDescriptorContext).)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `ColorTranslator`

ColorTranslator

The `ColorTranslator` class translates colors to and from GDI+ Color structures.

Methods

- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- FromHtml - Translates an HTML color representation to a GDI+ Color structure.
- FromOle - Translates an OLE color value to a GDI+ Color structure.
- FromWin32 - Translates a Windows color value to a GDI+ Color structure.
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- ToHtml - Translates the specified Color structure to an HTML string color representation.
- ToOle - Translates the specified Color structure to an OLE color.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- ToWin32 - Translates the specified Color structure to a Windows color.

The `System.Drawing` Namespace

- → Classes
  - → `SystemColors`

SystemColors

The `SystemColors` class contains properties each of which is a Color structure that is the color of a Windows display element. According to Microsoft's website: *Better performance is achieved by using the properties of the SystemPens or SystemBrushes classes rather than creating a new pen or brush based on a value from SystemColors. For example, if you wanted to get a brush for the face color of a 3-D element, use the SystemBrushes.Control property because it gets a brush that already exists, whereas calling the SolidBrush constructor with a parameter value of SystemColors.Control will create a new brush.*

Properties

- ActiveBorder - Gets a Color structure that is the color of the active window's border.
- ActiveCaption - Gets a Color structure that is the color of the background of the active window's title bar.
- ActiveCaptionText - Gets a Color structure that is the color of the text in the active window's title bar.
- AppWorkspace - Gets a Color structure that is the color of the application workspace.
- ButtonFace - Gets a Color structure that is the face color of a 3-D element.
- ButtonHighlight - Gets a Color structure that is the highlight color of a 3-D element.
- ButtonShadow - Gets a Color structure that is the shadow color of a 3-D element.
- Control - Gets a Color structure that is the face color of a 3-D element.
- ControlDark - Gets a Color structure that is the shadow color of a 3-D element.
- ControlDarkDark - Gets a Color structure that is the dark shadow color of a 3-D element.
- ControlLight - Gets a Color structure that is the light color of a 3-D element.
- ControlLightLight - Gets a Color structure that is the highlight color of a 3-D element.
- ControlText - Gets a Color structure that is the color of text in a 3-D element.
- Desktop - Gets a Color structure that is the color of the desktop.
- GradientActiveCaption - Gets a Color structure that is the lightest color in the color gradient of an active window's title bar.
- GradientInactiveCaption - Gets a Color structure that is the lightest color in the color gradient of an inactive window's title bar.
- GrayText - Gets a Color structure that is the color of dimmed text.
- Highlight - Gets a Color structure that is the color of the background of selected items.
- HighlightText - Gets a Color structure that is the color of the text of selected items.

Properties (continued)

- HotTrack - Gets a Color structure that is the color used to designate a hot-tracked item.
- InactiveBorder - Gets a Color structure that is the color of an inactive window's border.
- InactiveCaption - Gets a Color structure that is the color of the background of an inactive window's title bar.
- InactiveCaptionText - Gets a Color structure that is the color of the text in an inactive window's title bar.
- Info - Gets a Color structure that is the color of the background of a ToolTip.
- InfoText - Gets a Color structure that is the color of the text of a ToolTip.
- Menu - Gets a Color structure that is the color of a menu's background.
- MenuBar - Gets a Color structure that is the color of the background of a menu bar.
- MenuHighlight - Gets a Color structure that is the color used to highlight menu items when the menu appears as a flat menu.
- MenuText - Gets a Color structure that is the color of a menu's text.
- ScrollBar - Gets a Color structure that is the color of the background of a scroll bar.
- Window - Gets a Color structure that is the color of the background in the client area of a window.
- WindowFrame - Gets a Color structure that is the color of a window frame.
- WindowText - Gets a Color structure that is the color of the text in the client area of a window.

The `System.Drawing` Namespace

- → Classes
  - → `StringFormat`

StringFormat

The `StringFormat` class encapsulates text layout information (such as alignment, orientation and tab stops) display manipulations (such as ellipsis insertion and national digit substitution) and OpenType features. According to Microsoft's website: *Many common formats are provided through the StringFormatFlags enumeration. StringFormat objects can be changed.*

Constructors

- StringFormat() - Initializes a new StringFormat object.
- StringFormat(StringFormat) - Initializes a new StringFormat object from the specified existing StringFormat object.
- StringFormat(StringFormatFlags) - Initializes a new StringFormat object with the specified StringFormatFlags enumeration.
- StringFormat(StringFormatFlags, Int32) - Initializes a new StringFormat object with the specified StringFormatFlags enumeration and language.

Properties

- Alignment - Gets or sets horizontal alignment of the string..
- DigitSubstitutionLanguage - Gets the language that is used when local digits are substituted for western digits.
- DigitSubstitutionMethod - Gets the method to be used for digit substitution.
- FormatFlags - Gets or sets a StringFormatFlags enumeration that contains formatting information.
- GenericDefault - Gets a generic default StringFormat object.
- GenericTypographic - Gets a generic typographic StringFormat object.
- HotkeyPrefix - Gets or sets the HotkeyPrefix object for this StringFormat object.
- LineAlignment - Gets or sets the vertical alignment of the string.
- Trimming - Gets or sets the StringTrimming enumeration for this StringFormat object.

Methods

- Clone - Creates an exact copy of this StringFormat object.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this StringFormat object.

Methods (continued)

- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetTabStops - Gets the tab stops for this StringFormat object.
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetDigitSubstitution - Specifies the language and method to be used when local digits are substituted for western digits.
- SetMeasurableCharacterRanges - Specifies an array of CharacterRange structures that represent the ranges of characters measured by a call to the MeasureCharacterRanges method.
- SetTabStops - Sets tab stops for this StringFormat object.
- ToString - Converts this StringFormat object to a human-readable string. (Overrides Object.ToString().)

The `System.Drawing` Namespace

- → Classes
  - → `Icon`

Icon

The `Icon` class represents a Windows icon, which is a small bitmap image that is used to represent an object. Icons can be thought of as transparent bitmaps, although their size is determined by the system. According to Microsoft's website: *An icon resource can contain multiple icon images. One icon file may contain images in several sizes and color depths. The image that is used in an application depends on the operating system and settings. The following list details the typical sizes for an icon: 16 pixels x 16 pixels, 32 pixels x 32 pixels, 48 pixels x 48 pixels.*

Constructors

- Icon(Stream) - Initializes a new instance of the Icon class from the specified data stream.
- Icon(String) - Initializes a new instance of the Icon class from the specified file name.
- Icon(Icon, Size) - Initializes a new instance of the Icon class and attempts to find a version of the icon that matches the requested size.
- Icon(Stream, Size) - Initializes a new instance of the Icon class of the specified size from the specified stream.
- Icon(String, Size) - Initializes a new instance of the Icon class of the specified size from the specified file.
- Icon(Type, String) - Initializes a new instance of the Icon class from a resource in the specified assembly.
- Icon(Icon, Int32, Int32) - Initializes a new instance of the Icon class and attempts to find a version of the icon that matches the requested size.
- Icon(Stream, Int32, Int32) - Initializes a new instance of the Icon class from the specified data stream and with the specified width and height.
- Icon(String, Int32, Int32) - Initializes a new instance of the Icon class with the specified width and height from the specified file.

Properties

- Handle - Gets the Windows handle for this Icon. This is not a copy of the handle; do not free it.
- Height - Gets the height of this Icon.
- Size - Gets the size of this Icon.
- Width - Gets the width of this Icon.

Methods

- Clone - Clones the Icon, creating a duplicate image.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this Icon.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- ExtractAssociatedIcon - Returns an icon representation of an image that is contained in the specified file.
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- FromHandle - Creates a GDI+ Icon from the specified Windows handle to an icon (HICON).
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- Save - Saves this Icon to the specified output Stream.
- ToBitmap - Converts this Icon to a GDI+ Bitmap.
- ToString - Gets a human-readable string that describes the Icon. (Overrides Object.ToString().)

The `System.Drawing` Namespace

- → Classes
  - → `IconConverter`

IconConverter

The `IconConverter` class converts an Icon object from one data type to another. Access this class through the TypeDescriptor object.

Constructors

- IconConverter - Initializes a new instance of the IconConverter class.

Methods

- CanConvertFrom(Type)- Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines whether this IconConverter can convert an instance of a specified type to an Icon, using the specified context. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type)- Determines whether this IconConverter can convert an Icon to an instance of a specified type, using the specified context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts a specified object to an Icon. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts an Icon (or an object that can be cast to an Icon) to a specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)

Methods (continued)

- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of the type that this TypeConverter is associated with, using the specified context, given a set of property values for the object. (Inherited from TypeConverter.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Returns whether changing a value on this object requires a call to CreateInstance to create a new value, using the specified context. (Inherited from TypeConverter.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Gets a collection of properties for the type of object specified by the value parameter. (Inherited from ExpandableObjectConverter.)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Gets a value indicating whether this object supports properties using the specified context. (Inherited from ExpandableObjectConverter.)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)

Methods (continued)

- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `SystemIcons`

SystemIcons

The `SystemIcons` class's properties represent an Icon object for Windows system-wide icons.

Properties

- Application - Gets an Icon object that contains the default application icon (WIN32: IDI_APPLICATION).
- Asterisk - Gets an Icon object that contains the system asterisk icon (WIN32: IDI_ASTERISK).
- Error - Gets an Icon object that contains the system error icon (WIN32: IDI_ERROR).
- Exclamation - Gets an Icon object that contains the system exclamation icon (WIN32: IDI_EXCLAMATION).
- Hand - Gets an Icon object that contains the system hand icon (WIN32: IDI_HAND).
- Information - Gets an Icon object that contains the system information icon (WIN32: IDI_INFORMATION).
- Question - Gets an Icon object that contains the system question icon (WIN32: IDI_QUESTION).
- Shield - Gets an Icon object that contains the shield icon.
- Warning - Gets an Icon object that contains the system warning icon (WIN32: IDI_WARNING).
- WinLogo - Gets an Icon object that contains the Windows logo icon (WIN32: IDI_WINLOGO).

The `System.Drawing` Namespace

- → Classes
  - → `Image`

Image

The `Image` class is an abstract base class that provides functionality for the Bitmap and Metafile descended classes. According to Microsoft's website: *To draw an Image on a Windows Form, you should use one of the DrawImage(Image, Point) methods.*

Properties

- Flags - Gets attribute flags for the pixel data of this Image.
- FrameDimensionsList - Gets an array of GUIDs that represent the dimensions of frames within this Image.
- Height - Gets the height, in pixels, of this Image.
- HorizontalResolution - Gets the horizontal resolution, in pixels per inch, of this Image.
- Palette - Gets or sets the color palette used for this Image.
- PhysicalDimension - Gets the width and height of this image.
- PixelFormat - Gets the pixel format for this Image.
- PropertyIdList - Gets IDs of the property items stored in this Image.
- PropertyItems - Gets all the property items (pieces of metadata) stored in this Image.
- RawFormat - Gets the file format of this Image.
- Size - Gets the width and height, in pixels, of this image.
- Tag - Gets or sets an object that provides additional data about the image.
- VerticalResolution - Gets the vertical resolution, in pixels per inch, of this Image.
- Width - Gets the width, in pixels, of this Image.

Methods

- Clone - Creates an exact copy of this Image.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Image.
- Dispose(Boolean) - Releases the unmanaged resources used by the Image and optionally releases the managed resources.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Overrides Object.Finalize().)

Methods (continued)

- FromFile(String) - Creates an Image from the specified file.
- FromFile(String, Boolean) - Creates an Image from the specified file using embedded color management information in that file.
- FromHbitmap(IntPtr) - Creates a Bitmap from a handle to a GDI bitmap.
- FromHbitmap(IntPtr, IntPtr) - Creates a Bitmap from a handle to a GDI bitmap and a handle to a GDI palette.
- FromStream(Stream) - Creates an Image from the specified data stream.
- FromStream(Stream, Boolean) - Creates an Image from the specified data stream, optionally using embedded color management information in that stream.
- FromStream(Stream, Boolean, Boolean) - Creates an Image from the specified data stream, optionally using embedded color management information and validating the image data.
- GetBounds - Gets the bounds of the image in the specified unit.
- GetEncoderParameterList - Returns information about the parameters supported by the specified image encoder.
- GetFrameCount - Returns the number of frames of the specified dimension.
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetPixelFormatSize - Returns the color depth, in number of bits per pixel, of the specified pixel format.
- GetPropertyItem - Gets the specified property item from this Image.
- GetThumbnailImage - Returns a thumbnail for this Image.
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- IsAlphaPixelFormat - Returns a value that indicates whether the pixel format for this Image contains alpha information.
- IsCanonicalPixelFormat - Returns a value that indicates whether the pixel format is 32 bits per pixel.
- IsExtendedPixelFormat - Returns a value that indicates whether the pixel format is 64 bits per pixel.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- RemovePropertyItem - Removes the specified property item from this Image.
- RotateFlip - Rotates, flips, or rotates and flips the Image.

Methods (continued)

- Save(String) - Saves this Image to the specified file or stream.
- Save(Stream, ImageFormat) - Saves this image to the specified stream in the specified format.
- Save(String, ImageFormat) - Saves this Image to the specified file in the specified format.
- Save(Stream, ImageCodecInfo, EncoderParameters) - Saves this image to the specified stream, with the specified encoder and image encoder parameters.
- Save(String, ImageCodecInfo, EncoderParameters) - Saves this Image to the specified file, with the specified encoder and image-encoder parameters.
- SaveAdd(EncoderParameters) - Adds a frame to the file or stream specified in a previous call to the Save method. Use this method to save selected frames from a multiple-frame image to another multiple-frame image.
- SaveAdd(Image, EncoderParameters) - Adds a frame to the file or stream specified in a previous call to the Save method.
- SelectActiveFrame - Selects the frame specified by the dimension and index.
- SetPropertyItem - Stores a property item (piece of metadata) in this Image.
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `ImageAnimator`

ImageAnimator

The `ImageAnimator` class animates an image that has time-based frames.

Methods

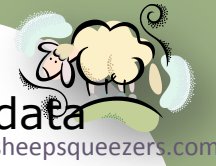
- Animate - Displays a multiple-frame image as an animation.
- CanAnimate - Returns a Boolean value indicating whether the specified image contains time-based frames.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- StopAnimate - Terminates a running animation.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- UpdateFrames() - Advances the frame in all images currently being animated. The new frame is drawn the next time the image is rendered.
- UpdateFrames(Image) - Advances the frame in the specified image. The new frame is drawn the next time the image is rendered. This method applies only to images with time-based frames.

Note that in order to create animated GIFs you must use another method. With that said, you can read in an animated GIF and play it using the `Animate` method above.

The `System.Drawing` Namespace

- → Classes
  - → `ImageConverter`

ImageConverter



The `ImageConverter` class can be used to convert Image objects from one data type to another. Access this class through the TypeDescriptor object.

Constructors

- ImageConverter - Initializes a new instance of the ImageConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines whether this ImageConverter can convert an instance of a specified type to an Image, using the specified context. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Determines whether this ImageConverter can convert an Image to an instance of a specified type, using the specified context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts a specified object to an Image. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts an Image (or an object that can be cast to an Image) to the specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)

Methods (continued)

- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of the type that this TypeConverter is associated with, using the specified context, given a set of property values for the object. (Inherited from TypeConverter.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Returns whether changing a value on this object requires a call to CreateInstance to create a new value, using the specified context. (Inherited from TypeConverter.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Gets the set of properties for this type. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Indicates whether this object supports properties. By default, this is false. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)

Methods (continued)

- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `ImageFormatConverter`

ImageFormatConverter

The `ImageFormatConverter` class can be used to convert ImageFormat objects from one data type to another. Access this class through the TypeDescriptor object. According to Microsoft's website: *A type converter is used to convert values between data types. A type converter also supports property configuration at design time by providing text-to-value conversion or a list of values that users can select from. Access the ImageFormatConverter class through the TypeDescriptor class by calling the GetConverter method. The ImageFormatConverter converts to and from known image formats, as specified in the ImageFormat class. If you want to convert between image formats, such as converting between BMP and JPEG, use one of the Image.Save methods that takes an ImageFormat object as a parameter.*

Constructors

- ImageFormatConverter - Initializes a new instance of the ImageFormatConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Indicates whether this converter can convert an object in the specified source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Gets a value indicating whether this converter can convert an object to the specified destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the specified object to an ImageFormat object. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)

Methods (continued)

- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to the specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of the type that this TypeConverter is associated with, using the specified context, given a set of property values for the object. (Inherited from TypeConverter.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Returns whether changing a value on this object requires a call to CreateInstance to create a new value, using the specified context. (Inherited from TypeConverter.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)

Methods (continued)

- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context and attributes. (Inherited from TypeConverter.)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Returns whether this object supports properties, using the specified context. (Inherited from TypeConverter.)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Gets a collection that contains a set of standard values for the data type this validator is designed for. Returns null if the data type does not support a standard set of values. (Overrides TypeConverter.GetStandardValues(ITypeDescriptorContext).)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Indicates whether this object supports a standard set of values that can be picked from a list. (Overrides TypeConverter.GetStandardValuesSupported(ITypeDescriptorContext).)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `Bitmap`

Bitmap

The `Bitmap` class encapsulates a GDI+ bitmap, which consists of the pixel data for a graphics image and its attributes. A Bitmap is an object used to work with images defined by pixel data. According to Microsoft's website: *A bitmap consists of the pixel data for a graphics image and its attributes. There are many standard formats for saving a bitmap to a file. GDI+ supports the following file formats: BMP, GIF, EXIF, JPG, PNG and TIFF.*

Constructors

- Bitmap(Image) - Initializes a new instance of the Bitmap class from the specified existing image.
- Bitmap(Stream) - Initializes a new instance of the Bitmap class from the specified data stream.
- Bitmap(String) - Initializes a new instance of the Bitmap class from the specified file.
- Bitmap(Image, Size) - Initializes a new instance of the Bitmap class from the specified existing image, scaled to the specified size.
- Bitmap(Int32, Int32) - Initializes a new instance of the Bitmap class with the specified size.
- Bitmap(Stream, Boolean) - Initializes a new instance of the Bitmap class from the specified data stream.
- Bitmap(String, Boolean) - Initializes a new instance of the Bitmap class from the specified file.
- Bitmap(Type, String) - Initializes a new instance of the Bitmap class from a specified resource.
- Bitmap(Image, Int32, Int32) - Initializes a new instance of the Bitmap class from the specified existing image, scaled to the specified size.
- Bitmap(Int32, Int32, Graphics) - Initializes a new instance of the Bitmap class with the specified size and with the resolution of the specified Graphics object.
- Bitmap(Int32, Int32, PixelFormat) - Initializes a new instance of the Bitmap class with the specified size and format.
- Bitmap(Int32, Int32, Int32, PixelFormat, IntPtr) - Initializes a new instance of the Bitmap class with the specified size, pixel format, and pixel data.

Properties

- Flags - Gets attribute flags for the pixel data of this Image. (Inherited from Image.)
- FrameDimensionsList - Gets an array of GUIDs that represent the dimensions of frames within this Image. (Inherited from Image.)
- Height - Gets the height, in pixels, of this Image. (Inherited from Image.)
- HorizontalResolution - Gets the horizontal resolution, in pixels per inch, of this Image. (Inherited from Image.)
- Palette - Gets or sets the color palette used for this Image. (Inherited from Image.)

Properties (continued)

- PhysicalDimension - Gets the width and height of this image. (Inherited from Image.)
- PixelFormat - Gets the pixel format for this Image. (Inherited from Image.)
- PropertyIdList - Gets IDs of the property items stored in this Image. (Inherited from Image.)
- PropertyItems - Gets all the property items (pieces of metadata) stored in this Image. (Inherited from Image.)
- RawFormat - Gets the file format of this Image. (Inherited from Image.)
- Size - Gets the width and height, in pixels, of this image. (Inherited from Image.)
- Tag - Gets or sets an object that provides additional data about the image. (Inherited from Image.)
- VerticalResolution - Gets the vertical resolution, in pixels per inch, of this Image. (Inherited from Image.)
- Width - Gets the width, in pixels, of this Image. (Inherited from Image.)

Methods

- Clone() - Creates an exact copy of this Image. (Inherited from Image.)
- Clone(Rectangle, PixelFormat) - Creates a copy of the section of this Bitmap defined by Rectangle structure and with a specified PixelFormat enumeration.
- Clone(RectangleF, PixelFormat) - Creates a copy of the section of this Bitmap defined with a specified PixelFormat enumeration.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this Image. (Inherited from Image.)
- Dispose(Boolean) - Releases the unmanaged resources used by the Image and optionally releases the managed resources. (Inherited from Image.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Image.)
- FromHicon - Creates a Bitmap from a Windows handle to an icon.
- FromResource - Creates a Bitmap from the specified Windows resource.
- GetBounds - Gets the bounds of the image in the specified unit. (Inherited from Image.)
- GetEncoderParameterList - Returns information about the parameters supported by the specified image encoder. (Inherited from Image.)
- GetFrameCount - Returns the number of frames of the specified dimension. (Inherited from Image.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetHbitmap() - Creates a GDI bitmap object from this Bitmap.

Methods (continued)

- GetHbitmap(Color) - Creates a GDI bitmap object from this Bitmap.
- GetHicon - Returns the handle to an icon.
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetPixel - Gets the color of the specified pixel in this Bitmap.
- GetPropertyItem - Gets the specified property item from this Image. (Inherited from Image.)
- GetThumbnailImage - Returns a thumbnail for this Image. (Inherited from Image.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- LockBits(Rectangle, ImageLockMode, PixelFormat) - Locks a Bitmap into system memory.
- LockBits(Rectangle, ImageLockMode, PixelFormat, BitmapData) - Locks a Bitmap into system memory
- MakeTransparent() - Makes the default transparent color transparent for this Bitmap.
- MakeTransparent(Color) - Makes the specified color transparent for this Bitmap.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- RemovePropertyItem - Removes the specified property item from this Image. (Inherited from Image.)
- RotateFlip - Rotates, flips, or rotates and flips the Image. (Inherited from Image.)
- Save(String) - Saves this Image to the specified file or stream. (Inherited from Image.)
- Save(Stream, ImageFormat) - Saves this image to the specified stream in the specified format. (Inherited from Image.)
- Save(String, ImageFormat) - Saves this Image to the specified file in the specified format. (Inherited from Image.)
- Save(Stream, ImageCodecInfo, EncoderParameters) - Saves this image to the specified stream, with the specified encoder and image encoder parameters. (Inherited from Image.)
- Save(String, ImageCodecInfo, EncoderParameters) - Saves this Image to the specified file, with the specified encoder and image-encoder parameters. (Inherited from Image.)
- SaveAdd(EncoderParameters) - Adds a frame to the file or stream specified in a previous call to the Save method. Use this method to save selected frames from a multiple-frame image to another multiple-frame image. (Inherited from Image.)
- SaveAdd(Image, EncoderParameters) - Adds a frame to the file or stream specified in a previous call to the Save method. (Inherited from Image.)
- SelectActiveFrame - Selects the frame specified by the dimension and index. (Inherited from Image.)
- SetPixel - Sets the color of the specified pixel in this Bitmap.
- SetPropertyItem - Stores a property item (piece of metadata) in this Image. (Inherited from Image.)

Methods (continued)

- SetResolution - Sets the resolution for this Bitmap.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- UnlockBits - Unlocks this Bitmap from system memory.

The `System.Drawing` Namespace

- → Classes
  - → `Graphics`

Graphics

The `Graphics` class encapsulates a GDI+ drawing surface. According to Microsoft's website: *The Graphics class provides methods for drawing objects to the display device. A Graphics is associated with a specific device context. You can obtain a Graphics object by calling the Control.CreateGraphics method on an object that inherits from System.Windows.Forms.Control, or by handling a control's Control.Paint event and accessing the Graphics property of the System.Windows.Forms.PaintEventArgs class. You can also create a Graphics object from an image by using the FromImage method. For more information about creating a Graphics object, see How to: Create Graphics Objects for Drawing. You can draw many different shapes and lines by using a Graphics object. For more information about how to draw lines and shapes, see the specific DrawGraphicalElement method for the line or shape you want to draw. These methods include DrawLine, DrawArc, DrawClosedCurve, DrawPolygon, and DrawRectangle. For more information about how to draw lines and shapes, see Using a Pen to Draw Lines and Shapes and Using a Brush to Fill Shapes. You can also draw images and icons by using the DrawImage and DrawIcon methods, respectively. To perform a bit-block transfer of color data from the screen to the drawing surface of the Graphics object, see CopyFromScreen. For more information about how to draw images with a Graphics object, see Working with Images, Bitmaps, Icons, and Metafiles. In addition, you can manipulate the coordinate system used by the Graphics object.*

Properties

- Bitmap(Image) - Initializes a new instance of the Bitmap class from the specified existing image.
- Bitmap(Stream) - Initializes a new instance of the Bitmap class from the specified data stream.
- Bitmap(String) - Initializes a new instance of the Bitmap class from the specified file.
- Bitmap(Image, Size) - Initializes a new instance of the Bitmap class from the specified existing image, scaled to the specified size.
- Bitmap(Int32, Int32) - Initializes a new instance of the Bitmap class with the specified size.
- Bitmap(Stream, Boolean) - Initializes a new instance of the Bitmap class from the specified data stream.
- Bitmap(String, Boolean) - Initializes a new instance of the Bitmap class from the specified file.
- Bitmap(Type, String) - Initializes a new instance of the Bitmap class from a specified resource.
- Bitmap(Image, Int32, Int32) - Initializes a new instance of the Bitmap class from the specified existing image, scaled to the specified size.
- Bitmap(Int32, Int32, Graphics) - Initializes a new instance of the Bitmap class with the specified size and with the resolution of the specified Graphics object.
- Bitmap(Int32, Int32, PixelFormat) - Initializes a new instance of the Bitmap class with the specified size and format.
- Bitmap(Int32, Int32, Int32, PixelFormat, IntPtr) - Initializes a new instance of the Bitmap class with the specified size, pixel format, and pixel data.

Methods

- AddMetafileComment Adds a comment to the current Metafile.
- BeginContainer() Saves a graphics container with the current state of this Graphics and opens and uses a new graphics container.
- BeginContainer(Rectangle, Rectangle, GraphicsUnit) Saves a graphics container with the current state of this Graphics and opens and uses a new graphics container with the specified scale transformation.
- BeginContainer(RectangleF, RectangleF, GraphicsUnit) Saves a graphics container with the current state of this Graphics and opens and uses a new graphics container with the specified scale transformation.
- Clear Clears the entire drawing surface and fills it with the specified background color.
- CopyFromScreen(Point, Point, Size) Performs a bit-block transfer of color data, corresponding to a rectangle of pixels, from the screen to the drawing surface of the Graphics.
- CopyFromScreen(Point, Point, Size, CopyPixelOperation) Performs a bit-block transfer of color data, corresponding to a rectangle of pixels, from the screen to the drawing surface of the Graphics.
- CopyFromScreen(Int32, Int32, Int32, Int32, Size) Performs a bit-block transfer of the color data, corresponding to a rectangle of pixels, from the screen to the drawing surface of the Graphics.

Methods (continued)

- CopyFromScreen(Int32, Int32, Int32, Int32, Size, CopyPixelOperation) Performs a bit-block transfer of the color data, corresponding to a rectangle of pixels, from the screen to the drawing surface of the Graphics.
- CreateObjRef Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose Releases all resources used by this Graphics.
- DrawArc(Pen, Rectangle, Single, Single) Draws an arc representing a portion of an ellipse specified by a Rectangle structure.
- DrawArc(Pen, RectangleF, Single, Single) Draws an arc representing a portion of an ellipse specified by a RectangleF structure.
- DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32) Draws an arc representing a portion of an ellipse specified by a pair of coordinates, a width, and a height.
- DrawArc(Pen, Single, Single, Single, Single, Single, Single) Draws an arc representing a portion of an ellipse specified by a pair of coordinates, a width, and a height.
- DrawBezier(Pen, Point, Point, Point, Point) Draws a Bézier spline defined by four Point structures.
- DrawBezier(Pen, PointF, PointF, PointF, PointF) Draws a Bézier spline defined by four PointF structures.
- DrawBezier(Pen, Single, Single, Single, Single, Single, Single, Single, Single) Draws a Bézier spline defined by four ordered pairs of coordinates that represent points.
- DrawBeziers(Pen, Point[]) Draws a series of Bézier splines from an array of Point structures.
- DrawBeziers(Pen, PointF[]) Draws a series of Bézier splines from an array of PointF structures.
- DrawClosedCurve(Pen, Point[]) Draws a closed cardinal spline defined by an array of Point structures.
- DrawClosedCurve(Pen, PointF[]) Draws a closed cardinal spline defined by an array of PointF structures.
- DrawClosedCurve(Pen, Point[], Single, FillMode) Draws a closed cardinal spline defined by an array of Point structures using a specified tension.
- DrawClosedCurve(Pen, PointF[], Single, FillMode) Draws a closed cardinal spline defined by an array of PointF structures using a specified tension.
- DrawCurve(Pen, Point[]) Draws a cardinal spline through a specified array of Point structures.
- DrawCurve(Pen, PointF[]) Draws a cardinal spline through a specified array of PointF structures.
- DrawCurve(Pen, Point[], Single) Draws a cardinal spline through a specified array of Point structures using a specified tension.
- DrawCurve(Pen, PointF[], Single) Draws a cardinal spline through a specified array of PointF structures using a specified tension.
- DrawCurve(Pen, PointF[], Int32, Int32) Draws a cardinal spline through a specified array of PointF structures. The drawing begins offset from the beginning of the array.
- DrawCurve(Pen, Point[], Int32, Int32, Single) Draws a cardinal spline through a specified array of Point structures using a specified tension.

Methods (continued)

- DrawCurve(Pen, PointF[], Int32, Int32, Single) Draws a cardinal spline through a specified array of PointF structures using a specified tension. The drawing begins offset from the beginning of the array.
- DrawEllipse(Pen, Rectangle) Draws an ellipse specified by a bounding Rectangle structure.
- DrawEllipse(Pen, RectangleF) Draws an ellipse defined by a bounding RectangleF.
- DrawEllipse(Pen, Int32, Int32, Int32, Int32) Draws an ellipse defined by a bounding rectangle specified by coordinates for the upper-left corner of the rectangle, a height, and a width.
- DrawEllipse(Pen, Single, Single, Single, Single) Draws an ellipse defined by a bounding rectangle specified by a pair of coordinates, a height, and a width.
- DrawIcon(Icon, Rectangle) Draws the image represented by the specified Icon within the area specified by a Rectangle structure.
- DrawIcon(Icon, Int32, Int32) Draws the image represented by the specified Icon at the specified coordinates.
- DrawIconUnstretched Draws the image represented by the specified Icon without scaling the image.
- DrawImage(Image, Point) Draws the specified Image, using its original physical size, at the specified location.
- DrawImage(Image, Point[]) Draws the specified Image at the specified location and with the specified shape and size.
- DrawImage(Image, PointF) Draws the specified Image, using its original physical size, at the specified location.
- DrawImage(Image, PointF[]) Draws the specified Image at the specified location and with the specified shape and size.
- DrawImage(Image, Rectangle) Draws the specified Image at the specified location and with the specified size.
- DrawImage(Image, RectangleF) Draws the specified Image at the specified location and with the specified size.
- DrawImage(Image, Int32, Int32) Draws the specified image, using its original physical size, at the location specified by a coordinate pair.
- DrawImage(Image, Single, Single) Draws the specified Image, using its original physical size, at the specified location.
- DrawImage(Image, Point[], Rectangle, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, PointF[], RectangleF, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Rectangle, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, RectangleF, RectangleF, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes) Draws the specified portion of the specified Image at the specified location.
- DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes) Draws the specified portion of the specified Image at the specified location and with the specified size.

Methods (continued)

- DrawImage(Image, Int32, Int32, Rectangle, GraphicsUnit) Draws a portion of an image at a specified location.
- DrawImage(Image, Int32, Int32, Int32, Int32) Draws the specified Image at the specified location and with the specified size.
- DrawImage(Image, Single, Single, RectangleF, GraphicsUnit) Draws a portion of an image at a specified location.
- DrawImage(Image, Single, Single, Single, Single) Draws the specified Image at the specified location and with the specified size.
- DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort, Int32) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort, Int32) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort, IntPtr) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes, Graphics.DrawImageAbort, IntPtr) Draws the specified portion of the specified Image at the specified location and with the specified size.
- DrawImageUnscaled(Image, Point) Draws a specified image using its original physical size at a specified location.
- DrawImageUnscaled(Image, Rectangle) Draws a specified image using its original physical size at a specified location.
- DrawImageUnscaled(Image, Int32, Int32) Draws the specified image using its original physical size at the location specified by a coordinate pair.
- DrawImageUnscaled(Image, Int32, Int32, Int32, Int32) Draws a specified image using its original physical size at a specified location.

Methods (continued)

- DrawImageUnscaledAndClipped Draws the specified image without scaling and clips it, if necessary, to fit in the specified rectangle.
- DrawLine(Pen, Point, Point) Draws a line connecting two Point structures.
- DrawLine(Pen, PointF, PointF) Draws a line connecting two PointF structures.
- DrawLine(Pen, Int32, Int32, Int32, Int32) Draws a line connecting the two points specified by the coordinate pairs.
- DrawLine(Pen, Single, Single, Single, Single) Draws a line connecting the two points specified by the coordinate pairs.
- DrawLines(Pen, Point[]) Draws a series of line segments that connect an array of Point structures.
- DrawLines(Pen, PointF[]) Draws a series of line segments that connect an array of PointF structures.
- DrawPath Draws a GraphicsPath.
- DrawPie(Pen, Rectangle, Single, Single) Draws a pie shape defined by an ellipse specified by a Rectangle structure and two radial lines.
- DrawPie(Pen, RectangleF, Single, Single) Draws a pie shape defined by an ellipse specified by a RectangleF structure and two radial lines.
- DrawPie(Pen, Int32, Int32, Int32, Int32, Int32, Int32) Draws a pie shape defined by an ellipse specified by a coordinate pair, a width, a height, and two radial lines.
- DrawPie(Pen, Single, Single, Single, Single, Single, Single) Draws a pie shape defined by an ellipse specified by a coordinate pair, a width, a height, and two radial lines.
- DrawPolygon(Pen, Point[]) Draws a polygon defined by an array of Point structures.
- DrawPolygon(Pen, PointF[]) Draws a polygon defined by an array of PointF structures.
- DrawRectangle(Pen, Rectangle) Draws a rectangle specified by a Rectangle structure.
- DrawRectangle(Pen, Int32, Int32, Int32, Int32) Draws a rectangle specified by a coordinate pair, a width, and a height.
- DrawRectangle(Pen, Single, Single, Single, Single) Draws a rectangle specified by a coordinate pair, a width, and a height.
- DrawRectangles(Pen, Rectangle[]) Draws a series of rectangles specified by Rectangle structures.
- DrawRectangles(Pen, RectangleF[]) Draws a series of rectangles specified by RectangleF structures.
- DrawString(String, Font, Brush, PointF) Draws the specified text string at the specified location with the specified Brush and Font objects.
- DrawString(String, Font, Brush, RectangleF) Draws the specified text string in the specified rectangle with the specified Brush and Font objects.
- DrawString(String, Font, Brush, PointF, StringFormat) Draws the specified text string at the specified location with the specified Brush and Font objects using the formatting attributes of the specified StringFormat.
- DrawString(String, Font, Brush, RectangleF, StringFormat) Draws the specified text string in the specified rectangle with the specified Brush and Font objects using the formatting attributes of the specified StringFormat.
- DrawString(String, Font, Brush, Single, Single) Draws the specified text string at the specified location with the specified Brush and Font objects.

Methods (continued)

- DrawString(String, Font, Brush, Single, Single, StringFormat) Draws the specified text string at the specified location with the specified Brush and Font objects using the formatting attributes of the specified StringFormat.
- EndContainer Closes the current graphics container and restores the state of this Graphics to the state saved by a call to the BeginContainer method.
- EnumerateMetafile(Metafile, Point, Graphics.EnumerateMetafileProc) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, Point[], Graphics.EnumerateMetafileProc) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, PointF, Graphics.EnumerateMetafileProc) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, PointF[], Graphics.EnumerateMetafileProc) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, Rectangle, Graphics.EnumerateMetafileProc) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, RectangleF, Graphics.EnumerateMetafileProc) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, Point, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, Point[], Graphics.EnumerateMetafileProc, IntPtr) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, PointF, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, PointF[], Graphics.EnumerateMetafileProc, IntPtr) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, Rectangle, Graphics.EnumerateMetafileProc, IntPtr) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, RectangleF, Graphics.EnumerateMetafileProc, IntPtr) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, Point, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point using specified image attributes.
- EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, Point[], Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram using specified image attributes.
- EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram.

Methods (continued)

- EnumerateMetafile(Metafile, PointF, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in the specified Metafile, one at a time, to a callback method for display at a specified point using specified image attributes.
- EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, PointF[], Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in the specified Metafile, one at a time, to a callback method for display in a specified parallelogram using specified image attributes.
- EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, Rectangle, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle using specified image attributes.
- EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, RectangleF, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records of the specified Metafile, one at a time, to a callback method for display in a specified rectangle using specified image attributes.
- EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point.
- EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram.
- EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle.
- EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point using specified image attributes.
- EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram using specified image attributes.

Methods (continued)

- EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display at a specified point using specified image attributes.
- EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records in a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified parallelogram using specified image attributes.
- EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle using specified image attributes.
- EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics.EnumerateMetafileProc, IntPtr, ImageAttributes) Sends the records of a selected rectangle from a Metafile, one at a time, to a callback method for display in a specified rectangle using specified image attributes.
- Equals(Object) Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- ExcludeClip(Rectangle) Updates the clip region of this Graphics to exclude the area specified by a Rectangle structure.
- ExcludeClip(Region) Updates the clip region of this Graphics to exclude the area specified by a Region.
- FillClosedCurve(Brush, Point[]) Fills the interior of a closed cardinal spline curve defined by an array of Point structures.
- FillClosedCurve(Brush, PointF[]) Fills the interior of a closed cardinal spline curve defined by an array of PointF structures.
- FillClosedCurve(Brush, Point[], FillMode) Fills the interior of a closed cardinal spline curve defined by an array of Point structures using the specified fill mode.
- FillClosedCurve(Brush, PointF[], FillMode) Fills the interior of a closed cardinal spline curve defined by an array of PointF structures using the specified fill mode.
- FillClosedCurve(Brush, Point[], FillMode, Single) Fills the interior of a closed cardinal spline curve defined by an array of Point structures using the specified fill mode and tension.
- FillClosedCurve(Brush, PointF[], FillMode, Single) Fills the interior of a closed cardinal spline curve defined by an array of PointF structures using the specified fill mode and tension.
- FillEllipse(Brush, Rectangle) Fills the interior of an ellipse defined by a bounding rectangle specified by a Rectangle structure.
- FillEllipse(Brush, RectangleF) Fills the interior of an ellipse defined by a bounding rectangle specified by a RectangleF structure.
- FillEllipse(Brush, Int32, Int32, Int32, Int32) Fills the interior of an ellipse defined by a bounding rectangle specified by a pair of coordinates, a width, and a height.
- FillEllipse(Brush, Single, Single, Single, Single) Fills the interior of an ellipse defined by a bounding rectangle specified by a pair of coordinates, a width, and a height.
- FillPath Fills the interior of a GraphicsPath.

Methods (continued)

- FillPie(Brush, Rectangle, Single, Single) Fills the interior of a pie section defined by an ellipse specified by a RectangleF structure and two radial lines.
- FillPie(Brush, Int32, Int32, Int32, Int32, Int32, Int32) Fills the interior of a pie section defined by an ellipse specified by a pair of coordinates, a width, a height, and two radial lines.
- FillPie(Brush, Single, Single, Single, Single, Single, Single) Fills the interior of a pie section defined by an ellipse specified by a pair of coordinates, a width, a height, and two radial lines.
- FillPolygon(Brush, Point[]) Fills the interior of a polygon defined by an array of points specified by Point structures.
- FillPolygon(Brush, PointF[]) Fills the interior of a polygon defined by an array of points specified by PointF structures.
- FillPolygon(Brush, Point[], FillMode) Fills the interior of a polygon defined by an array of points specified by Point structures using the specified fill mode.
- FillPolygon(Brush, PointF[], FillMode) Fills the interior of a polygon defined by an array of points specified by PointF structures using the specified fill mode.
- FillRectangle(Brush, Rectangle) Fills the interior of a rectangle specified by a Rectangle structure.
- FillRectangle(Brush, RectangleF) Fills the interior of a rectangle specified by a RectangleF structure.
- FillRectangle(Brush, Int32, Int32, Int32, Int32) Fills the interior of a rectangle specified by a pair of coordinates, a width, and a height.
- FillRectangle(Brush, Single, Single, Single, Single) Fills the interior of a rectangle specified by a pair of coordinates, a width, and a height.
- FillRectangles(Brush, Rectangle[]) Fills the interiors of a series of rectangles specified by Rectangle structures.
- FillRectangles(Brush, RectangleF[]) Fills the interiors of a series of rectangles specified by RectangleF structures.
- FillRegion Fills the interior of a Region.
- Finalize Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- Flush() Forces execution of all pending graphics operations and returns immediately without waiting for the operations to finish.
- Flush(FlushIntention) Forces execution of all pending graphics operations with the method waiting or not waiting, as specified, to return before the operations finish.
- FromHdc(IntPtr) Creates a new Graphics from the specified handle to a device context.
- FromHdc(IntPtr, IntPtr) Creates a new Graphics from the specified handle to a device context and handle to a device.
- FromHdcInternal Infrastructure. Returns a Graphics for the specified device context.
- FromHwnd Creates a new Graphics from the specified handle to a window.
- FromHwndInternal Infrastructure. Creates a new Graphics for the specified windows handle.
- FromImage Creates a new Graphics from the specified Image.

Methods (continued)

- GetContextInfo Infrastructure. Gets the cumulative graphics context.
- GetHalftonePalette Gets a handle to the current Windows halftone palette.
- GetHashCode Serves as a hash function for a particular type. (Inherited from Object.)
- GetHdc Gets the handle to the device context associated with this Graphics.
- GetLifetimeService Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetNearestColor Gets the nearest color to the specified Color structure.
- GetType Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- IntersectClip(Rectangle) Updates the clip region of this Graphics to the intersection of the current clip region and the specified Rectangle structure.
- IntersectClip(RectangleF) Updates the clip region of this Graphics to the intersection of the current clip region and the specified RectangleF structure.
- IntersectClip(Region) Updates the clip region of this Graphics to the intersection of the current clip region and the specified Region.
- IsVisible(Point) Indicates whether the specified Point structure is contained within the visible clip region of this Graphics.
- IsVisible(PointF) Indicates whether the specified PointF structure is contained within the visible clip region of this Graphics.
- IsVisible(Rectangle) Indicates whether the rectangle specified by a Rectangle structure is contained within the visible clip region of this Graphics.
- IsVisible(RectangleF) Indicates whether the rectangle specified by a RectangleF structure is contained within the visible clip region of this Graphics.
- IsVisible(Int32, Int32) Indicates whether the point specified by a pair of coordinates is contained within the visible clip region of this Graphics.
- IsVisible(Single, Single) Indicates whether the point specified by a pair of coordinates is contained within the visible clip region of this Graphics.
- IsVisible(Int32, Int32, Int32, Int32) Indicates whether the rectangle specified by a pair of coordinates, a width, and a height is contained within the visible clip region of this Graphics.
- IsVisible(Single, Single, Single, Single) Indicates whether the rectangle specified by a pair of coordinates, a width, and a height is contained within the visible clip region of this Graphics.
- MeasureCharacterRanges Gets an array of Region objects, each of which bounds a range of character positions within the specified string.
- MeasureString(String, Font) Measures the specified string when drawn with the specified Font.
- MeasureString(String, Font, SizeF) Measures the specified string when drawn with the specified Font within the specified layout area.

Methods (continued)

- MeasureString(String, Font, Int32) Measures the specified string when drawn with the specified Font.
- MeasureString(String, Font, PointF, StringFormat) Measures the specified string when drawn with the specified Font and formatted with the specified StringFormat.
- MeasureString(String, Font, SizeF, StringFormat) Measures the specified string when drawn with the specified Font and formatted with the specified StringFormat.
- MeasureString(String, Font, Int32, StringFormat) Measures the specified string when drawn with the specified Font and formatted with the specified StringFormat.
- MeasureString(String, Font, SizeF, StringFormat, Int32, Int32) Measures the specified string when drawn with the specified Font and formatted with the specified StringFormat.
- MemberwiseClone() Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- MultiplyTransform(Matrix) Multiplies the world transformation of this Graphics and specified the Matrix.
- MultiplyTransform(Matrix, MatrixOrder) Multiplies the world transformation of this Graphics and specified the Matrix in the specified order.
- ReleaseHdc() Releases a device context handle obtained by a previous call to the GetHdc method of this Graphics.
- ReleaseHdc(IntPtr) Releases a device context handle obtained by a previous call to the GetHdc method of this Graphics.
- ReleaseHdcInternal Infrastructure. Releases a handle to a device context.
- ResetClip Resets the clip region of this Graphics to an infinite region.
- ResetTransform Resets the world transformation matrix of this Graphics to the identity matrix.
- Restore Restores the state of this Graphics to the state represented by a GraphicsState.
- RotateTransform(Single) Applies the specified rotation to the transformation matrix of this Graphics.
- RotateTransform(Single, MatrixOrder) Applies the specified rotation to the transformation matrix of this Graphics in the specified order.
- Save Saves the current state of this Graphics and identifies the saved state with a GraphicsState.
- ScaleTransform(Single, Single) Applies the specified scaling operation to the transformation matrix of this Graphics by prepending it to the object's transformation matrix.
- ScaleTransform(Single, Single, MatrixOrder) Applies the specified scaling operation to the transformation matrix of this Graphics in the specified order.
- SetClip(Graphics) Sets the clipping region of this Graphics to the Clip property of the specified Graphics.
- SetClip(GraphicsPath) Sets the clipping region of this Graphics to the specified GraphicsPath.
- SetClip(Rectangle) Sets the clipping region of this Graphics to the rectangle specified by a Rectangle structure.
- SetClip(RectangleF) Sets the clipping region of this Graphics to the rectangle specified by a RectangleF structure.

Methods (continued)

- SetClip(Graphics, CombineMode) Sets the clipping region of this Graphics to the result of the specified combining operation of the current clip region and the Clip property of the specified Graphics.
- SetClip(GraphicsPath, CombineMode) Sets the clipping region of this Graphics to the result of the specified operation combining the current clip region and the specified GraphicsPath.
- SetClip(Rectangle, CombineMode) Sets the clipping region of this Graphics to the result of the specified operation combining the current clip region and the rectangle specified by a Rectangle structure.
- SetClip(RectangleF, CombineMode) Sets the clipping region of this Graphics to the result of the specified operation combining the current clip region and the rectangle specified by a RectangleF structure.
- SetClip(Region, CombineMode) Sets the clipping region of this Graphics to the result of the specified operation combining the current clip region and the specified Region.
- ToString Returns a string that represents the current object. (Inherited from Object.)
- TransformPoints(CoordinateSpace, CoordinateSpace, Point[]) Transforms an array of points from one coordinate space to another using the current world and page transformations of this Graphics.
- TransformPoints(CoordinateSpace, CoordinateSpace, PointF[]) Transforms an array of points from one coordinate space to another using the current world and page transformations of this Graphics.
- TranslateClip(Int32, Int32) Translates the clipping region of this Graphics by specified amounts in the horizontal and vertical directions.
- TranslateClip(Single, Single) Translates the clipping region of this Graphics by specified amounts in the horizontal and vertical directions.
- TranslateTransform(Single, Single) Changes the origin of the coordinate system by prepending the specified translation to the transformation matrix of this Graphics.
- TranslateTransform(Single, Single, MatrixOrder) Changes the origin of the coordinate system by applying the specified translation to the transformation matrix of this Graphics in the specified order.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `GraphicsContainer`

GraphicsContainer

The `GraphicsContainer` class represents the internal data of a graphics container. This class is used when saving the state of a Graphics object using the BeginContainer and EndContainer methods.

Methods

- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `GraphicsPath`

GraphicsPath

The `GraphicsPath` class represents a series of connected lines and curves. According to Microsoft's website: *Applications use paths to draw outlines of shapes, fill the interiors of shapes, and create clipping regions. The graphics engine maintains the coordinates of geometric shapes in a path in world coordinate space. A path may be composed of any number of figures (subpaths). Each figure is either composed of a sequence of connected lines and curves or a geometric shape primitive. The starting point of a figure is the first point in the sequence of connected lines and curves. The ending point is the last point in the sequence. The starting and ending points of a geometric shape primitive are defined by the primitive specification. A figure that consists of a sequence of connected lines and curves (whose starting and ending points may be coincident) is an open figure, unless it is closed explicitly. A figure can be closed explicitly, by using the CloseFigure method, which closes the current figure by connecting a line from the ending point to the starting point. A figure that consists of a geometric shape primitive is a closed figure. For purposes of filling and clipping (for example, if a path is rendered using FillPath), all open figures are closed by adding a line from the figure's first point to its last point. A new figure is implicitly started when a path is created or when a figure is closed. A new figure is explicitly created when the StartFigure method is called. When a geometric shape primitive is added to a path, it adds a figure containing the geometric shape, and also implicitly starts a new figure. Consequently, there is always a current figure in a path. When lines and curves are added to a path, an implicit line is added as needed to connect the ending point of the current figure to the starting point of the new lines and curves to form a sequence of connected lines and curves.*

GraphicsPath

A figure has a direction that describes how line and curve segments are traced between the starting point and the ending point. The direction is defined in the order that lines and curves are added to a figure, or is defined by the geometric shape primitive. The direction is used in determining the path interiors for clipping and fill.

Constructors

- GraphicsPath() - Initializes a new instance of the GraphicsPath class with a FillMode value of Alternate.
- GraphicsPath(FillMode) - Initializes a new instance of the GraphicsPath class with the specified FillMode enumeration.
- GraphicsPath(Point[], Byte[]) - Initializes a new instance of the GraphicsPath class with the specified PathPointType and Point arrays.
- GraphicsPath(PointF[], Byte[]) - Initializes a new instance of the GraphicsPath array with the specified PathPointType and PointF arrays.
- GraphicsPath(Point[], Byte[], FillMode) - Initializes a new instance of the GraphicsPath class with the specified PathPointType and Point arrays and with the specified FillMode enumeration element.
- GraphicsPath(PointF[], Byte[], FillMode) - Initializes a new instance of the GraphicsPath array with the specified PathPointType and PointF arrays and with the specified FillMode enumeration element.

Properties

- FillMode - Gets or sets a FillMode enumeration that determines how the interiors of shapes in this GraphicsPath are filled.
- PathData - Gets a PathData that encapsulates arrays of points (points) and types (types) for this GraphicsPath.
- PathPoints - Gets the points in the path.
- PathTypes - Gets the types of the corresponding points in the PathPoints array.
- PointCount - Gets the number of elements in the PathPoints or the PathTypes array.

Methods

- AddArc(Rectangle, Single, Single) - Appends an elliptical arc to the current figure.
- AddArc(RectangleF, Single, Single) - Appends an elliptical arc to the current figure.
- AddArc(Int32, Int32, Int32, Int32, Single, Single) - Appends an elliptical arc to the current figure.
- AddArc(Single, Single, Single, Single, Single, Single) - Appends an elliptical arc to the current figure.
- AddBezier(Point, Point, Point, Point) - Adds a cubic Bézier curve to the current figure.
- AddBezier(PointF, PointF, PointF, PointF) - Adds a cubic Bézier curve to the current figure.
- AddBezier(Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32) - Adds a cubic Bézier curve to the current figure.
- AddBezier(Single, Single, Single, Single, Single, Single, Single, Single) - Adds a cubic Bézier curve to the current figure.
- AddBeziers(Point[]) - Adds a sequence of connected cubic Bézier curves to the current figure.
- AddBeziers(PointF[]) - Adds a sequence of connected cubic Bézier curves to the current figure.

Methods (continued)

- AddClosedCurve(Point[]) - Adds a closed curve to this path. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddClosedCurve(PointF[]) - Adds a closed curve to this path. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddClosedCurve(Point[], Single) - Adds a closed curve to this path. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddClosedCurve(PointF[], Single) - Adds a closed curve to this path. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddCurve(Point[]) - Adds a spline curve to the current figure. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddCurve(PointF[]) - Adds a spline curve to the current figure. A cardinal spline curve is used because the curve travels through each of the points in the array.
- AddCurve(Point[], Single) - Adds a spline curve to the current figure.
- AddCurve(PointF[], Single) - Adds a spline curve to the current figure.
- AddCurve(Point[], Int32, Int32, Single) - Adds a spline curve to the current figure.
- AddCurve(PointF[], Int32, Int32, Single) - Adds a spline curve to the current figure.
- AddEllipse(Rectangle) - Adds an ellipse to the current path.
- AddEllipse(RectangleF) - Adds an ellipse to the current path.
- AddEllipse(Int32, Int32, Int32, Int32) - Adds an ellipse to the current path.
- AddEllipse(Single, Single, Single, Single) - Adds an ellipse to the current path.
- AddLine(Point, Point) - Appends a line segment to this GraphicsPath.
- AddLine(PointF, PointF) - Appends a line segment to this GraphicsPath.
- AddLine(Int32, Int32, Int32, Int32) - Appends a line segment to the current figure.
- AddLine(Single, Single, Single, Single) - Appends a line segment to this GraphicsPath.
- AddLines(Point[]) - Appends a series of connected line segments to the end of this GraphicsPath.
- AddLines(PointF[]) - Appends a series of connected line segments to the end of this GraphicsPath.
- AddPath - Appends the specified GraphicsPath to this path.
- AddPie(Rectangle, Single, Single) - Adds the outline of a pie shape to this path.
- AddPie(Int32, Int32, Int32, Int32, Single, Single) - Adds the outline of a pie shape to this path.
- AddPie(Single, Single, Single, Single, Single, Single) - Adds the outline of a pie shape to this path.
- AddPolygon(Point[]) - Adds a polygon to this path.
- AddPolygon(PointF[]) - Adds a polygon to this path.
- AddRectangle(Rectangle) - Adds a rectangle to this path.
- AddRectangle(RectangleF) - Adds a rectangle to this path.
- AddRectangles(Rectangle[]) - Adds a series of rectangles to this path.
- AddRectangles(RectangleF[]) - Adds a series of rectangles to this path.
- AddString(String, FontFamily, Int32, Single, Point, StringFormat) - Adds a text string to this path.
- AddString(String, FontFamily, Int32, Single, PointF, StringFormat) - Adds a text string to this path.
- AddString(String, FontFamily, Int32, Single, Rectangle, StringFormat) - Adds a text string to this path.

Methods (continued)

- AddString(String, FontFamily, Int32, Single, RectangleF, StringFormat) - Adds a text string to this path.
- ClearMarkers - Clears all markers from this path.
- Clone - Creates an exact copy of this path.
- CloseAllFigures - Closes all open figures in this path and starts a new figure. It closes each open figure by connecting a line from its endpoint to its starting point.
- CloseFigure - Closes the current figure and starts a new figure. If the current figure contains a sequence of connected lines and curves, the method closes the loop by connecting a line from the endpoint to the starting point.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this GraphicsPath.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- Flatten() - Converts each curve in this path into a sequence of connected line segments.
- Flatten(Matrix) - Applies the specified transform and then converts each curve in this GraphicsPath into a sequence of connected line segments.
- Flatten(Matrix, Single) - Converts each curve in this GraphicsPath into a sequence of connected line segments.
- GetBounds() - Returns a rectangle that bounds this GraphicsPath.
- GetBounds(Matrix) - Returns a rectangle that bounds this GraphicsPath when this path is transformed by the specified Matrix.
- GetBounds(Matrix, Pen) - Returns a rectangle that bounds this GraphicsPath when the current path is transformed by the specified Matrix and drawn with the specified Pen.
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLastPoint - Gets the last point in the PathPoints array of this GraphicsPath.
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- IsOutlineVisible(Point, Pen) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen.
- IsOutlineVisible(PointF, Pen) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen.
- IsOutlineVisible(Int32, Int32, Pen) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen.
- IsOutlineVisible(Point, Pen, Graphics) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen and using the specified Graphics.
- IsOutlineVisible(PointF, Pen, Graphics) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen and using the specified Graphics.

Methods (continued)

- IsOutlineVisible(Single, Single, Pen) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen.
- IsOutlineVisible(Int32, Int32, Pen, Graphics) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen and using the specified Graphics.
- IsOutlineVisible(Single, Single, Pen, Graphics) - Indicates whether the specified point is contained within (under) the outline of this GraphicsPath when drawn with the specified Pen and using the specified Graphics.
- IsVisible(Point) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(PointF) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(Int32, Int32) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(Point, Graphics) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(PointF, Graphics) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(Single, Single) - Indicates whether the specified point is contained within this GraphicsPath.
- IsVisible(Int32, Int32, Graphics) - Indicates whether the specified point is contained within this GraphicsPath, using the specified Graphics.
- IsVisible(Single, Single, Graphics) - Indicates whether the specified point is contained within this GraphicsPath in the visible clip region of the specified Graphics.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- Reset - Empties the PathPoints and PathTypes arrays and sets the FillMode to Alternate.
- Reverse - Reverses the order of points in the PathPoints array of this GraphicsPath.
- SetMarkers - Sets a marker on this GraphicsPath.
- StartFigure - Starts a new figure without closing the current figure. All subsequent points added to the path are added to this new figure.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- Transform - Applies a transform matrix to this GraphicsPath.
- Warp(PointF[], RectangleF) - Applies a warp transform, defined by a rectangle and a parallelogram, to this GraphicsPath.
- Warp(PointF[], RectangleF, Matrix) - Applies a warp transform, defined by a rectangle and a parallelogram, to this GraphicsPath.
- Warp(PointF[], RectangleF, Matrix, WarpMode) - Applies a warp transform, defined by a rectangle and a parallelogram, to this GraphicsPath.
- Warp(PointF[], RectangleF, Matrix, WarpMode, Single) - Applies a warp transform, defined by a rectangle and a parallelogram, to this GraphicsPath.
- Widen(Pen) - Adds an additional outline to the path.
- Widen(Pen, Matrix) - Adds an additional outline to the GraphicsPath.
- Widen(Pen, Matrix, Single) - Replaces this GraphicsPath with curves that enclose the area that is filled when this path is drawn by the specified pen.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `PathData`

PathData

The `PathData` class represents contains the graphical data that makes up a GraphicsPath object.

Constructors

- PathData - Initializes a new instance of the PathData class.

Properties

- Points - Gets or sets an array of PointF structures that represents the points through which the path is constructed.
- Types - Gets or sets the types of the corresponding points in the path.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `GraphicsPathIterator`

GraphicsPathIterator

The `GraphicsPathIterator` class provides the ability to iterate through subpaths in a GraphicsPath and test the types of shapes contained in each subpath.

Constructors

- GraphicsPathIterator - Initializes a new instance of the GraphicsPathIterator class with the specified GraphicsPath object.

Properties

- Count - Gets the number of points in the path.
- SubpathCount - Gets the number of subpaths in the path.

Methods

- CopyData - Copies the PathPoints property and PathTypes property arrays of the associated GraphicsPath into the two specified arrays.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this GraphicsPathIterator object.
- Enumerate - Copies the PathPoints property and PathTypes property arrays of the associated GraphicsPath into the two specified arrays.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- HasCurve - Indicates whether the path associated with this GraphicsPathIterator contains a curve.
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- NextMarker(GraphicsPath) - This GraphicsPathIterator object has a GraphicsPath object associated with it. The NextMarker method increments the associated GraphicsPath to the next marker in its path and copies all the points contained between the current marker and the next marker (or end of path) to a second GraphicsPath object passed in to the parameter.

GraphicsPathIterator

Methods (continued)

- NextMarker(Int32, Int32) - Increments the GraphicsPathIterator to the next marker in the path and returns the start and stop indexes by way of the [out] parameters.
- NextPathType - Gets the starting index and the ending index of the next group of data points that all have the same type.
- NextSubpath(GraphicsPath, Boolean) - Gets the next figure (subpath) from the associated path of this GraphicsPathIterator.
- NextSubpath(Int32, Int32, Boolean) - Moves the GraphicsPathIterator to the next subpath in the path. The start index and end index of the next subpath are contained in the [out] parameters.
- Rewind - Rewinds this GraphicsPathIterator to the beginning of its associated path.
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `GraphicsState`

GraphicsState

The `GraphicsState` class represents the state of a Graphics object. This object is returned by a call to the Save methods.

Methods

- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `Region`

Region

The `Region` class describes the interior of a graphics shape composed of rectangles and paths. According to Microsoft's website: *A region is scalable because its coordinates are specified in world coordinates. On a drawing surface, however, its interior is dependent on the size and shape of the pixels representing it. An application can use regions to clip the output of drawing operations. These regions are called clipping regions. An application can also use regions in hit-testing operations, such as checking whether a point or a rectangle intersects a region. An application can fill a region by using the Graphics.FillRegion method and a Brush object.*

Constructors

- Region() - Initializes a new Region.
- Region(GraphicsPath) - Initializes a new Region with the specified GraphicsPath.
- Region(Rectangle) - Initializes a new Region from the specified Rectangle structure.
- Region(RectangleF) - Initializes a new Region from the specified RectangleF structure.
- Region(RegionData) - Initializes a new Region from the specified data.

Methods

- Clone - Creates an exact copy of this Region.
- Complement(GraphicsPath) - Updates this Region to contain the portion of the specified GraphicsPath that does not intersect with this Region.
- Complement(Rectangle) - Updates this Region to contain the portion of the specified Rectangle structure that does not intersect with this Region.
- Complement(RectangleF) - Updates this Region to contain the portion of the specified RectangleF structure that does not intersect with this Region.
- Complement(Region) - Updates this Region to contain the portion of the specified Region that does not intersect with this Region.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this Region.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)

Methods (continued)

- Equals(Region, Graphics) - Tests whether the specified Region is identical to this Region on the specified drawing surface.
- Exclude(GraphicsPath) - Updates this Region to contain only the portion of its interior that does not intersect with the specified GraphicsPath.
- Exclude(Rectangle) - Updates this Region to contain only the portion of its interior that does not intersect with the specified Rectangle structure.
- Exclude(RectangleF) - Updates this Region to contain only the portion of its interior that does not intersect with the specified RectangleF structure.
- Exclude(Region) - Updates this Region to contain only the portion of its interior that does not intersect with the specified Region.
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- FromHrgn - Initializes a new Region from a handle to the specified existing GDI region.
- GetBounds - Gets a RectangleF structure that represents a rectangle that bounds this Region on the drawing surface of a Graphics object.
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetHrgn - Returns a Windows handle to this Region in the specified graphics context.
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetRegionData - Returns a RegionData that represents the information that describes this Region.
- GetRegionScans - Returns an array of RectangleF structures that approximate this Region after the specified matrix transformation is applied.
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- Intersect(GraphicsPath) - Updates this Region to the intersection of itself with the specified GraphicsPath.
- Intersect(Rectangle) - Updates this Region to the intersection of itself with the specified Rectangle structure.
- Intersect(RectangleF) - Updates this Region to the intersection of itself with the specified RectangleF structure.
- Intersect(Region) - Updates this Region to the intersection of itself with the specified Region.
- IsEmpty - Tests whether this Region has an empty interior on the specified drawing surface.
- IsInfinite - Tests whether this Region has an infinite interior on the specified drawing surface.
- IsVisible(Point) - Tests whether the specified Point structure is contained within this Region.
- IsVisible(PointF) - Tests whether the specified PointF structure is contained within this Region.
- IsVisible(Rectangle) - Tests whether any portion of the specified Rectangle structure is contained within this Region.
- IsVisible(RectangleF) - Tests whether any portion of the specified RectangleF structure is contained within this Region.
- IsVisible(Point, Graphics) - Tests whether the specified Point structure is contained within this Region when drawn using the specified Graphics.
- IsVisible(PointF, Graphics) - Tests whether the specified PointF structure is contained within this Region when drawn using the specified Graphics.

Methods (continued)

- IsVisible(Rectangle, Graphics) - Tests whether any portion of the specified Rectangle structure is contained within this Region when drawn using the specified Graphics.
- IsVisible(RectangleF, Graphics) - Tests whether any portion of the specified RectangleF structure is contained within this Region when drawn using the specified Graphics.
- IsVisible(Single, Single) - Tests whether the specified point is contained within this Region.
- IsVisible(Int32, Int32, Graphics) - Tests whether the specified point is contained within this Region object when drawn using the specified Graphics object.
- IsVisible(Single, Single, Graphics) - Tests whether the specified point is contained within this Region when drawn using the specified Graphics.
- IsVisible(Int32, Int32, Int32, Int32) - Tests whether any portion of the specified rectangle is contained within this Region.
- IsVisible(Single, Single, Single, Single) - Tests whether any portion of the specified rectangle is contained within this Region.
- IsVisible(Int32, Int32, Int32, Int32, Graphics) - Tests whether any portion of the specified rectangle is contained within this Region when drawn using the specified Graphics.
- IsVisible(Single, Single, Single, Single, Graphics) - Tests whether any portion of the specified rectangle is contained within this Region when drawn using the specified Graphics.
- MakeEmpty - Initializes this Region to an empty interior.
- MakeInfinite - Initializes this Region object to an infinite interior.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- ReleaseHrgn - Releases the handle of the Region.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- Transform - Transforms this Region by the specified Matrix.
- Translate(Int32, Int32) - Offsets the coordinates of this Region by the specified amount.
- Translate(Single, Single) - Offsets the coordinates of this Region by the specified amount.
- Union(GraphicsPath) - Updates this Region to the union of itself and the specified GraphicsPath.
- Union(Rectangle) - Updates this Region to the union of itself and the specified Rectangle structure.
- Union(RectangleF) - Updates this Region to the union of itself and the specified RectangleF structure.
- Union(Region) - Updates this Region to the union of itself and the specified Region.
- Xor(GraphicsPath) - Updates this Region to the union minus the intersection of itself with the specified GraphicsPath.
- Xor(Rectangle) - Updates this Region to the union minus the intersection of itself with the specified Rectangle structure.
- Xor(RectangleF) - Updates this Region to the union minus the intersection of itself with the specified RectangleF structure.
- Xor(Region) - Updates this Region to the union minus the intersection of itself with the specified Region.

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `RegionData`

RegionData

The `RegionData` class encapsulates the data that makes up a Region object.

Properties

- Data - Gets or sets an array of bytes that specify the Region object.

The `System.Drawing` Namespace

- → Classes
  - → `PointConverter`

PointConverter

The `PointConverter` class converts a Point object from one data type to another. According to Microsoft's website: *The PointConverter class is typically used for creating a design-time experience for controls that have properties that are of type Point, and is used to convert points to strings for display in a designer and to convert strings entered in a designer to a Point object. Access this class through the TypeDescriptor object.*

Constructors

- PointConverter - Initializes a new instance of the PointConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines if this converter can convert an object in the given source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Gets a value indicating whether this converter can convert an object to the given destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the specified object to a Point object. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to the specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)

Methods (continued)

- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of this type given a set of property values for the object. (Overrides TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary).)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Determines if changing a value on this object should require a call to CreateInstance to create a new value. (Overrides TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext).)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Retrieves the set of properties for this type. By default, a type does not return any properties. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Determines if this object supports properties. By default, this is false. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)

Methods (continued)

- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `RectangleConverter`

RectangleConverter

The `RectangleConverter` class converts rectangles from one data type to another. Access this class through the TypeDescriptor.

Constructors

- RectangleConverter - Initializes a new instance of the RectangleConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines if this converter can convert an object in the given source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Gets a value indicating whether this converter can convert an object to the given destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the given object to a Rectangle object. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to the specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)

Methods (continued)

- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of this type given a set of property values for the object. This is useful for objects that are immutable but still want to provide changeable properties. (Overrides TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary).)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Determines if changing a value on this object should require a call to CreateInstance to create a new value. (Overrides TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext).)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Retrieves the set of properties for this type. By default, a type does not return any properties. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Determines if this object supports properties. By default, this is false. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)

Methods (continued)

- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `SizeConverter`

SizeConverter

The `SizeConverter` class is used to convert from one data type to another. Access this class through the TypeDescriptor object.

Constructors

- SizeConverter - Initializes a new SizeConverter object.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Determines whether this converter can convert an object in the specified source type to the native type of the converter. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Gets a value indicating whether this converter can convert an object to the given destination type using the context. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the specified object to the converter's native type. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the specified object to the specified type. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)
- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)

Methods (continued)

- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an object of this type by using a specified set of property values for the object. This is useful for creating non-changeable objects that have changeable properties. (Overrides TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary).)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Determines whether changing a value on this object should require a call to the CreateInstance method to create a new value. (Overrides TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext).)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Retrieves the set of properties for this type. By default, a type does not have any properties to return. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Determines whether this object supports properties. By default, this is false. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)
- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)

Methods (continued)

- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

- → Classes
  - → `SizeFConverter`

SizeFConverter

The `SizeFConverter` class converts SizeF objects from one type to another. According to Microsoft's website: *Most commonly, the SizeFConverter type converter is used to convert Font instances to and from their corresponding string representations. Access the FontConverter class through the TypeDescriptor.*

Constructors

- SizeFConverter - Initializes a new instance of the SizeFConverter class.

Methods

- CanConvertFrom(Type) - Returns whether this converter can convert an object of the given type to the type of this converter. (Inherited from TypeConverter.)
- CanConvertFrom(ITypeDescriptorContext, Type) - Returns a value indicating whether the converter can convert from the type specified to the SizeF type, using the specified context. (Overrides TypeConverter.CanConvertFrom(ITypeDescriptorContext, Type).)
- CanConvertTo(Type) - Returns whether this converter can convert the object to the specified type. (Inherited from TypeConverter.)
- CanConvertTo(ITypeDescriptorContext, Type) - Returns a value indicating whether the SizeFConverter can convert a SizeF to the specified type. (Overrides TypeConverter.CanConvertTo(ITypeDescriptorContext, Type).)
- ConvertFrom(Object) - Converts the given value to the type of this converter. (Inherited from TypeConverter.)
- ConvertFrom(ITypeDescriptorContext, CultureInfo, Object) - Converts the given object to the type of this converter, using the specified context and culture information. (Overrides TypeConverter.ConvertFrom(ITypeDescriptorContext, CultureInfo, Object).)
- ConvertFromInvariantString(String) - Converts the given string to the type of this converter, using the invariant culture. (Inherited from TypeConverter.)
- ConvertFromInvariantString(ITypeDescriptorContext, String) - Converts the given string to the type of this converter, using the invariant culture and the specified context. (Inherited from TypeConverter.)
- ConvertFromString(String) - Converts the specified text to an object. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, String) - Converts the given text to an object, using the specified context. (Inherited from TypeConverter.)
- ConvertFromString(ITypeDescriptorContext, CultureInfo, String) - Converts the given text to an object, using the specified context and culture information. (Inherited from TypeConverter.)
- ConvertTo(Object, Type) - Converts the given value object to the specified type, using the arguments. (Inherited from TypeConverter.)
- ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type) - Converts the given value object to the specified type, using the specified context and culture information. (Overrides TypeConverter.ConvertTo(ITypeDescriptorContext, CultureInfo, Object, Type).)

Methods (continued)

- ConvertToInvariantString(Object) - Converts the specified value to a culture-invariant string representation. (Inherited from TypeConverter.)
- ConvertToInvariantString(ITypeDescriptorContext, Object) - Converts the specified value to a culture-invariant string representation, using the specified context. (Inherited from TypeConverter.)
- ConvertToString(Object) - Converts the specified value to a string representation. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, Object) - Converts the given value to a string representation, using the given context. (Inherited from TypeConverter.)
- ConvertToString(ITypeDescriptorContext, CultureInfo, Object) - Converts the given value to a string representation, using the specified context and culture information. (Inherited from TypeConverter.)
- CreateInstance(IDictionary) - Re-creates an Object given a set of property values for the object. (Inherited from TypeConverter.)
- CreateInstance(ITypeDescriptorContext, IDictionary) - Creates an instance of a SizeF with the specified property values using the specified context. (Overrides TypeConverter.CreateInstance(ITypeDescriptorContext, IDictionary).)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetConvertFromException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetConvertToException - Returns an exception to throw when a conversion cannot be performed. (Inherited from TypeConverter.)
- GetCreateInstanceSupported() - Returns whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Inherited from TypeConverter.)
- GetCreateInstanceSupported(ITypeDescriptorContext) - Returns a value indicating whether changing a value on this object requires a call to the CreateInstance method to create a new value. (Overrides TypeConverter.GetCreateInstanceSupported(ITypeDescriptorContext).)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetProperties(Object) - Returns a collection of properties for the type of array specified by the value parameter. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object) - Returns a collection of properties for the type of array specified by the value parameter, using the specified context. (Inherited from TypeConverter.)
- GetProperties(ITypeDescriptorContext, Object, Attribute[]) - Retrieves a set of properties for the SizeF type using the specified context and attributes. (Overrides TypeConverter.GetProperties(ITypeDescriptorContext, Object, Attribute[]).)
- GetPropertiesSupported() - Returns whether this object supports properties. (Inherited from TypeConverter.)
- GetPropertiesSupported(ITypeDescriptorContext) - Returns whether the SizeF type supports properties. (Overrides TypeConverter.GetPropertiesSupported(ITypeDescriptorContext).)
- GetStandardValues() - Returns a collection of standard values from the default context for the data type this type converter is designed for. (Inherited from TypeConverter.)

Methods (continued)

- GetStandardValues(ITypeDescriptorContext) - Returns a collection of standard values for the data type this type converter is designed for when provided with a format context. (Inherited from TypeConverter.)
- GetStandardValuesExclusive() - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list. (Inherited from TypeConverter.)
- GetStandardValuesExclusive(ITypeDescriptorContext) - Returns whether the collection of standard values returned from GetStandardValues is an exclusive list of possible values, using the specified context. (Inherited from TypeConverter.)
- GetStandardValuesSupported() - Returns whether this object supports a standard set of values that can be picked from a list. (Inherited from TypeConverter.)
- GetStandardValuesSupported(ITypeDescriptorContext) - Returns whether this object supports a standard set of values that can be picked from a list, using the specified context. (Inherited from TypeConverter.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- IsValid(Object) - Returns whether the given value object is valid for this type. (Inherited from TypeConverter.)
- IsValid(ITypeDescriptorContext, Object) - Returns whether the given value object is valid for this type and for the specified context. (Inherited from TypeConverter.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- SortProperties - Sorts a collection of properties. (Inherited from TypeConverter.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `AdjustableArrowCap`

AdjustableArrowCap

The `AdjustableArrowCap` class represents an adjustable arrow-shaped line cap.

Constructors

- AdjustableArrowCap(Single, Single) - Initializes a new instance of the AdjustableArrowCap class with the specified width and height. The arrow end caps created with this constructor are always filled.
- AdjustableArrowCap(Single, Single, Boolean) - Initializes a new instance of the AdjustableArrowCap class with the specified width, height, and fill property. Whether an arrow end cap is filled depends on the argument passed to the isFilled parameter.

Properties

- BaseCap - Gets or sets the LineCap enumeration on which this CustomLineCap is based. (Inherited from CustomLineCap.)
- BaseInset - Gets or sets the distance between the cap and the line. (Inherited from CustomLineCap.)
- Filled - Gets or sets whether the arrow cap is filled.
- Height - Gets or sets the height of the arrow cap.
- MiddleInset - Gets or sets the number of units between the outline of the arrow cap and the fill.
- StrokeJoin - Gets or sets the LineJoin enumeration that determines how lines that compose this CustomLineCap object are joined. (Inherited from CustomLineCap.)
- Width - Gets or sets the width of the arrow cap.
- WidthScale - Gets or sets the amount by which to scale this CustomLineCap Class object with respect to the width of the Pen object. (Inherited from CustomLineCap.)

Methods

- Clone - Creates an exact copy of this CustomLineCap. (Inherited from CustomLineCap.)
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this CustomLineCap object. (Inherited from CustomLineCap.)
- Dispose(Boolean) - Releases the unmanaged resources used by the CustomLineCap and optionally releases the managed resources. (Inherited from CustomLineCap.)
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an CustomLineCap to attempt to free resources and perform other cleanup operations before the CustomLineCap is reclaimed by garbage collection. (Inherited from CustomLineCap.)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetStrokeCaps - Gets the caps used to start and end lines that make up this custom cap. (Inherited from CustomLineCap.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)

Methods (continued)

- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetStrokeCaps - Sets the caps used to start and end lines that make up this custom cap. (Inherited from CustomLineCap.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `CustomLineCap`

CustomLineCap

The `CustomLineCap` class encapsulates a custom user-defined line cap. According to Microsoft's website: *Line caps are used at the beginnings and ends of lines or curves drawn by GDI+ Pen objects. GDI+ supports several predefined cap styles, and also allows users to define their own cap styles. This class is used to create custom cap styles.*

Constructors

- CustomLineCap(GraphicsPath, GraphicsPath) - Initializes a new instance of the CustomLineCap class with the specified outline and fill.
- CustomLineCap(GraphicsPath, GraphicsPath, LineCap) - Initializes a new instance of the CustomLineCap class from the specified existing LineCap enumeration with the specified outline and fill.
- CustomLineCap(GraphicsPath, GraphicsPath, LineCap, Single) - Initializes a new instance of the CustomLineCap class from the specified existing LineCap enumeration with the specified outline, fill, and inset.

Properties

- BaseCap - Gets or sets the LineCap enumeration on which this CustomLineCap is based.
- BaseInset - Gets or sets the distance between the cap and the line.
- StrokeJoin - Gets or sets the LineJoin enumeration that determines how lines that compose this CustomLineCap object are joined.
- WidthScale - Gets or sets the amount by which to scale this CustomLineCap Class object with respect to the width of the Pen object.

Methods

- Clone - Creates an exact copy of this CustomLineCap.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose() - Releases all resources used by this CustomLineCap object.
- Dispose(Boolean) - Releases the unmanaged resources used by the CustomLineCap and optionally releases the managed resources.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an CustomLineCap to attempt to free resources and perform other cleanup operations before the CustomLineCap is reclaimed by garbage collection. (Overrides Object.Finalize().)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)

Methods (continued)

- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- GetStrokeCaps - Gets the caps used to start and end lines that make up this custom cap.
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- SetStrokeCaps - Sets the caps used to start and end lines that make up this custom cap.
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing.Drawing2D` Namespace

- → Classes
  - → `Matrix`

Matrix

The `Matrix` class encapsulates a 3-by-3 affine matrix that represents a geometric transform. According to Microsoft's website: *In GDI+ you can store an affine transformation in a Matrix object. Because the third column of a matrix that represents an affine transformation is always (0, 0, 1), you specify only the six numbers in the first two columns when you construct a Matrix object.*

Constructors

- Matrix() - Initializes a new instance of the Matrix class as the identity matrix.
- Matrix(Rectangle, Point[]) - Initializes a new instance of the Matrix class to the geometric transform defined by the specified rectangle and array of points.
- Matrix(RectangleF, PointF[]) - Initializes a new instance of the Matrix class to the geometric transform defined by the specified rectangle and array of points.
- Matrix(Single, Single, Single, Single, Single, Single) - Initializes a new instance of the Matrix class with the specified elements.

Properties

- Elements - Gets an array of floating-point values that represents the elements of this Matrix.
- IsIdentity - Gets a value indicating whether this Matrix is the identity matrix.
- IsInvertible - Gets a value indicating whether this Matrix is invertible.
- OffsetX - Gets the x translation value (the dx value, or the element in the third row and first column) of this Matrix.
- OffsetY - Gets the y translation value (the dy value, or the element in the third row and second column) of this Matrix.

Methods

- Clone - Creates an exact copy of this Matrix.
- CreateObjRef - Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
- Dispose - Releases all resources used by this Matrix.
- Equals - Tests whether the specified object is a Matrix and is identical to this Matrix. (Overrides Object.Equals(Object).)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
- GetHashCode - Returns a hash code. (Overrides Object.GetHashCode().)
- GetLifetimeService - Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)

Methods (continued)

- GetType - Gets the Type of the current instance. (Inherited from Object.)
- InitializeLifetimeService - Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
- Invert - Inverts this Matrix, if it is invertible.
- MemberwiseClone() - Creates a shallow copy of the current Object. (Inherited from Object.)
- MemberwiseClone(Boolean) - Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
- Multiply(Matrix) - Multiplies this Matrix by the matrix specified in the matrix parameter, by prepending the specified Matrix.
- Multiply(Matrix, MatrixOrder) - Multiplies this Matrix by the matrix specified in the matrix parameter, and in the order specified in the order parameter.
- Reset - Resets this Matrix to have the elements of the identity matrix.
- Rotate(Single) - Prepend to this Matrix a clockwise rotation, around the origin and by the specified angle.
- Rotate(Single, MatrixOrder) - Applies a clockwise rotation of an amount specified in the angle parameter, around the origin (zero x and y coordinates) for this Matrix.
- RotateAt(Single, PointF) - Applies a clockwise rotation to this Matrix around the point specified in the point parameter, and by prepending the rotation.
- RotateAt(Single, PointF, MatrixOrder) - Applies a clockwise rotation about the specified point to this Matrix in the specified order.
- Scale(Single, Single) - Applies the specified scale vector to this Matrix by prepending the scale vector.
- Scale(Single, Single, MatrixOrder) - Applies the specified scale vector (scaleX and scaleY) to this Matrix using the specified order.
- Shear(Single, Single) - Applies the specified shear vector to this Matrix by prepending the shear transformation.
- Shear(Single, Single, MatrixOrder) - Applies the specified shear vector to this Matrix in the specified order.
- ToString - Returns a string that represents the current object. (Inherited from Object.)
- TransformPoints(Point[]) - Applies the geometric transform represented by this Matrix to a specified array of points.
- TransformPoints(PointF[]) - Applies the geometric transform represented by this Matrix to a specified array of points.
- TransformVectors(Point[]) - Applies only the scale and rotate components of this Matrix to the specified array of points.
- TransformVectors(PointF[]) - Multiplies each vector in an array by the matrix. The translation elements of this matrix (third row) are ignored.
- Translate(Single, Single) - Applies the specified translation vector (offsetX and offsetY) to this Matrix by prepending the translation vector.
- Translate(Single, Single, MatrixOrder) - Applies the specified translation vector to this Matrix in the specified order.
- VectorTransformPoints - Multiplies each vector in an array by the matrix. The translation elements of this matrix (third row) are ignored.

The `System.Drawing` Namespace

- → Classes
  - → `BufferedGraphics`
  - → `BufferedGraphicsContext`
  - → `BufferedGraphicsManager`

BufferedGraphics,BufferedGraphicsContext,BufferedGraphicsManager

According to Microsoft's website: *The BufferedGraphics class allows you to implement custom double buffering for your graphics. It provides a wrapper for a graphics buffer, along with methods that you can use to write to the buffer and render its contents to an output device. Graphics that use double buffering can reduce or eliminate flicker that is caused by redrawing a display surface. When you use double buffering, updated graphics are first drawn to a buffer in memory, and the contents of this buffer are then quickly written to some or all of the displayed surface. This relatively brief overwrite of the displayed graphics typically reduces or eliminates the flicker that sometimes occurs when graphics are updated.*

Please see Microsoft's website for more on these three classes.

The `System.Drawing` Namespace

→ Attributes

Attributes

Below are the attributes in the `System.Drawing` namespace.

Attributes

- ToolboxBitmapAttribute - Allows you to specify an icon to represent a control in a container, such as the Microsoft Visual Studio Form Designer.

The `System.Drawing` Namespace

→ EventArgs

EventArgs

The are no `EventArgs` in the `System.Drawing` namespace. With that said, there is an important eventarg that you should be aware of. The `PaintEventArgs` class, in the `System.Windows.Forms` namespace, provides data for the `Paint` event in Windows forms and is used a lot with the `Graphics` class and its members. The `Paint` event is called when a control is redrawn and it is the `PaintEventArgs` which specified the `Graphics` to use to paint the control. Below are the `PaintEventArgs` members:

Constructors

- PaintEventArgs - Initializes a new instance of the PaintEventArgs class with the specified graphics and clipping rectangle.

Properties

- ClipRectangle - Gets the rectangle in which to paint.
- Graphics - Gets the graphics used to paint.

Methods

- Dispose() - Releases all resources used by the PaintEventArgs.
- Dispose(Boolean) - Releases the unmanaged resources used by the PaintEventArgs and optionally releases the managed resources.
- Equals(Object) - Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
- Finalize - Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Overrides Object.Finalize().)
- GetHashCode - Serves as a hash function for a particular type. (Inherited from Object.)
- GetType - Gets the Type of the current instance. (Inherited from Object.)
- MemberwiseClone - Creates a shallow copy of the current Object. (Inherited from Object.)
- ToString - Returns a string that represents the current object. (Inherited from Object.)

The `System.Drawing` Namespace

→ Structures

Structures

The are several structures available in the `System.Drawing` namespace.

Structures

- CharacterRange - Specifies a range of character positions within a string.
- Color - Represents an ARGB (alpha, red, green, blue) color.
- Point - Represents an ordered pair of integer x- and y-coordinates that defines a point in a two-dimensional plane.
- PointF - Represents an ordered pair of floating-point x- and y-coordinates that defines a point in a two-dimensional plane.
- Rectangle - Stores a set of four integers that represent the location and size of a rectangle
- RectangleF - Stores a set of four floating-point numbers that represent the location and size of a rectangle. For more advanced region functions, use a Region object.
- Size - Stores an ordered pair of integers, which specify a Height and Width.
- SizeF - Stores an ordered pair of floating-point numbers, typically the width and height of a rectangle.

The `System.Drawing` Namespace

→ Interfaces

Interfaces

There is one interfaces available in the `System.Drawing` namespace.

Interfaces

- IDeviceContext - Defines methods for obtaining and releasing an existing handle to a Windows device context.

The `System.Drawing` Namespace

→ Delegates

Delegates

Below are the delegates in the `System.Drawing` namespace.

Delegates

- Graphics.DrawImageAbort - Provides a callback method for deciding when the DrawImage method should prematurely cancel execution and stop drawing an image.
- Graphics.EnumerateMetafileProc - Provides a callback method for the EnumerateMetafile method.
- Image.GetThumbnailImageAbort - Provides a callback method for determining when the GetThumbnailImage method should prematurely cancel execution.

The `System.Drawing` Namespace

→ Enumerations

Enumerations

The following are the enumerations available in the `System.Drawing` namespace.

Enumerations

- CombineMode - Specifies how different clipping regions can be combined.
- CompositingMode - Specifies how the source colors are combined with the background colors.
- CompositingQuality - Specifies the quality level to use during compositing.
- ContentAlignment - Specifies alignment of content on the drawing surface.
- CoordinateSpace - Specifies the system to use when evaluating coordinates.
- CopyPixelOperation - Determines how the source color in a copy pixel operation is combined with the destination color to result in a final color.
- DashCap - Specifies the type of graphic shape to use on both ends of each dash in a dashed line.
- DashStyle - Specifies the style of dashed lines drawn with a Pen object.
- FillMode - Specifies how the interior of a closed path is filled.
- FlushIntention - Specifies whether commands in the graphics stack are terminated (flushed) immediately or executed as soon as possible.
- FontStyle - Specifies style information applied to text.
- GraphicsUnit - Specifies the unit of measure for the given data.
- HatchStyle - Specifies the different patterns available for HatchBrush objects.
- InterpolationMode - The InterpolationMode enumeration specifies the algorithm that is used when images are scaled or rotated.
- KnownColor - Specifies the known system colors.
- LineCap - Specifies the available cap styles with which a Pen object can end a line.
- LineJoin - Specifies how to join consecutive line or curve segments in a figure (subpath) contained in a GraphicsPath object.
- LinearGradientMode - Specifies the direction of a linear gradient.
- MatrixOrder - Specifies the order for matrix transform operations.
- PathPointType - Specifies the type of point in a GraphicsPath object.
- PenAlignment - Specifies the alignment of a Pen object in relation to the theoretical, zero-width line.
- PenType - Specifies the type of fill a Pen object uses to fill lines.
- PixelOffsetMode - Specifies how pixels are offset during rendering.
- QualityMode - Specifies the overall quality when rendering GDI+ objects.
- RotateFlipType - Specifies how much an image is rotated and the axis used to flip the image.
- SmoothingMode - Specifies whether smoothing (antialiasing) is applied to lines and curves and the edges of filled areas.
- StringAlignment - Specifies the alignment of a text string relative to its layout rectangle.
- StringDigitSubstitute - The StringDigitSubstitute enumeration specifies how to substitute digits in a string according to a user's locale or language.
- StringFormatFlags - Specifies the display and layout information for text strings.
- StringTrimming - Specifies how to trim characters from a string that does not completely fit into a layout shape.
- StringUnit - Specifies the units of measure for a text string.
- WarpMode - Specifies the type of warp transformation applied in a Warp method.
- WrapMode - Specifies how a texture or gradient is tiled when it is smaller than the area being filled.

The `System.Drawing` Namespace

→ Exceptions

Exceptions

The are no exceptions in the `System.Drawing` namespace.

What Next?

In *C# Programming IV-#*, we look at specific classes within specific namespaces such as the System namespace, the System.Drawing namespace, etc.

References

Click the book titles below to read more about these books on Amazon.com's website.

- Introducing Microsoft LINQ, Paolo Pialorsi and Marco Russo, Microsoft Press, ISBN:9780735623910
- LINQ Pocket Reference, Joseph Albahari and Ben Albahari, O'Reilly Press, ISBN:9780596519247
- Inside C#, Tom Archer and Andrew Whitechapel, Microsoft Press, ISBN:0735616485
- C# 4.0 In a Nutshell, O'Reilly Press, Joseph Albahari and Ben Albahari, ISBN:9780596800956
- The Object Primer, Scott W. Ambler, Cambridge Press, ISBN:0521540186
- CLR via C#, Jeffrey Richter, Microsoft Press, ISBN:9780735621633

Support sheepsqueezers.com

If you found this information helpful, please consider supporting sheepsqueezers.com. There are several ways to support our site:

- ☐ Buy me a cup of coffee by clicking on the following link and donate to my PayPal account: Buy Me A Cup Of Coffee?.
- ☐ Visit my Amazon.com Wish list at the following link and purchase an item: http://amzn.com/w/3OBK1K4EIWIR6

Please let me know if this document was useful by e-mailing me at `comments@sheepsqueezers.com`.