

General addenda and comments

Random

You can obtain a floating point random number using `rndObj.NextDouble()` which returns a value equal or greater than 0.0 but less than 1, as follows:

```
double randomDouble = 15 + 10*rndObj.NextDouble();    // gives 15 - 24.999
```

Finding your PC's IP address:

The following code will display your PC's name and IP address:
You need to add 'using System.Net;'

```
string hostName = Dns.GetHostName();
MessageBox.Show ("Host Name = " + hostName);
IPHostEntry local = Dns.GetHostEntry(hostName);
foreach (IPAddress ipaddress in local.AddressList)
{
    MessageBox.Show("IPAddress = " + ipaddress.ToString());
}
```

Reading and writing a single pixel

There is no graphics command to read or write a single pixel. You could draw a line from x,y to $x,y+1$, but that draws two pixels.

To read or write to a pixel on the screen you have to import some DLL methods from `gdi32` (see chapter 18.6). The calls are `CreateDC`, `DeleteDC`, `SetPixel` and `GetPixel`.

To draw to the screen you have to 'borrow' it from Windows. To do this you have to create a device context (using `CreateDC`) and return it when finished (using `DeleteDC`). `CreateDC` returns a pointer to the drawing object, which is used by `SetPixel` and `GetPixel`.

The following code displays the colour value of a panel when it is clicked and then generates a new random background colour for it.

At the top of your code, add the declaration 'using System.Runtime.InteropServices;'

Add the following `DLLImport` code before 'public Form1':

```
// DLL to create a device context for a resource.
[DllImport("gdi32.dll")]
public static extern IntPtr CreateDC(string strDriver, string strDevice,
string strOutput, IntPtr pData);
```

```

// DLL to release the resource
[DllImport("gdi32.dll")]
public static extern bool DeleteDC(IntPtr hdc);

// DLL to set a pixel - pass DC pointer, x, y and colour
[DllImport("gdi32.dll")]
static extern uint SetPixel(IntPtr hdc, int X, int Y, uint crColor);

// DLL to get pixel value - pass DC pointer, x and y, colour value is
returned
[DllImport("gdi32.dll")]
public static extern uint GetPixel(IntPtr hdc, int nXPos, int nYPos);

```

Place a panel on a form and enter the following code for its mouse down event:

```

private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    // Obtain a Device Context for the panel.
    IntPtr ptrScreen = CreateDC("Display", null, null, IntPtr.Zero);

    // and use it to obtain the pixel value of the mouse position
    // GetPixel
    int pixelValue = (int)GetPixel(ptrScreen, e.X, e.Y);

    // and release it
    DeleteDC(ptrScreen);

    // Convert the pixel value colour to a .NET Color object
    Color clr = Color.FromArgb((pixelValue & 0x000000FF),
        (pixelValue & 0x0000FF00) >> 8,
        (pixelValue & 0x00FF0000) >> 16);

    string str = "red: " + clr.R + " green: " + clr.G + "blue: " + clr.B;
    MessageBox.Show(str);

    // generate new panel background colour
    Random rndObj = new Random();
    int red = rndObj.Next(255);
    int green = rndObj.Next(255);
    int blue = rndObj.Next(255);
    panel1.BackColor = Color.FromArgb(red, green, blue);
}

```

Note the use of `e.X` and `e.Y` from `MouseEventArgs` to obtain the X and Y position of the mouse.

`Color.FromArgb` creates a `Color` structure, `clr`, from the 8-bit alpha, red, green, and blue pixel values. Here we only use RGB. Note that having got a `Color` value, you can extract the individual ARGB values; see the use of `clr.R`. The program creates the new background colour using `Color.FromArgb` from three random

You can also set a pixel in the same way. The following code snippet writes to the screen on the panel's mouse move event. Note that the X and Y positions are obtained from the panel, and they are used as X and Y positions to write to the screen (not the panel).

```
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    Random rndObj = new Random();
    int red = rndObj.Next(255);
    int green = rndObj.Next(255);
    int blue = rndObj.Next(255);
    // Obtain a Device Context for the Display.
    IntPtr ptrScreen = CreateDC("Display", null, null, IntPtr.Zero);
    // and use it to obtain the pixel value of the mouse position

    // SetPixel
    SetPixel(ptrScreen, e.X, e.Y, (uint)((red << 16) | (green << 8) | (blue)));
    // and release resource
    DeleteDC(ptrScreen);
}
```

Reading and saving nulls in database

You may read or write a null from/to a database, rather than using a default value.

The item to check for is **DBNull.Value**

Reading Null values from a database

Null values read from a database can be converted to a string without a problem, e.g.:

```
txtTemp.Text = this.DataSet.Table.Rows[rowPosition]["temp"].ToString();
```

However, if you want to use a value such as an integer, you have to check that it's not a null entry on the database before using it. For example the following code will check for a null in the database and use the (integer) value if not:

```
if (this.DataSet.Table.Rows[rowPosition]["sp"] != DBNull.Value)
```

```

    {
      int tempSP = (int)this.DataSet1.Table.Rows[rowPosition]["sp"];
    }

```

Of course, if you want to use a default value instead of a null you can do so, just add an else clause. (else tempSP = -1;)

Writing Null values to a database

In this example a floating point value is read from a textbox. If the textbox is empty, DBNull.Value is saved in the database, otherwise the textbox's value is used:

```

if (txtTemp.Text == String.Empty)
{
  drAdd["temp"] = DBNull.Value;
}
else
{
  drAdd["temp"] = float.Parse(txtTemp.Text);
}

```

You might end up with lots of lines like the following:

```

if (txtSP.Text == String.Empty) { drAdd["sp"] = DBNull.Value; }
else { drAdd["sp"] = int.Parse(txtSP.Text); }
if (txtName.Text == String.Empty) { drAdd["name"] = DBNull.Value; }
else { drAdd["name"] = txtName.Text; }
if (txtTemp.Text == String.Empty) { drAdd["temp"] = DBNull.Value; }
else { drAdd["temp"] = float.Parse(txtTemp.Text); }

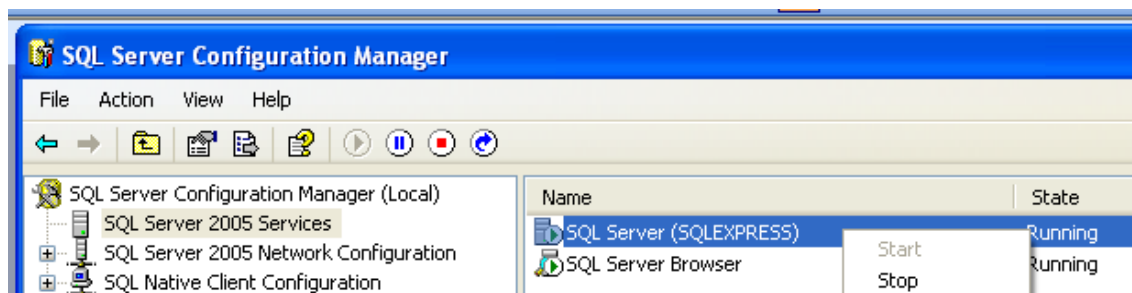
```

SQLServer

If your database isn't working, make sure the SQLServer is running.

Try: All Programs > Microsoft SQLServer > Configuration tools > SQL Server Configuration Manager

Select SQL Server services, and check that the SQL Server is running:



John