C# How-to 1: Develop an application using Visual Studio .NET

Since you're reading this, presumably you are thinking of programming in C#. There are several books already devoted to this, and most give an example of the sort of program presented below. The difference here is that I want to show how to produce a very simple application using the Visual Studio .NET development system, rather than using the command-line compiler as most other references do. After all, Windows is a GUI-based operating system, right?

So, here is how you go about using Visual Studion .NET (VS.NET from now on) to develop a simple program. It starts with the assumption that you have opened VS.NET and are looking at the Start page.

Steps to take

1. To start a new project, click the 'New project' button on the start page, or select New->Project... from the File menu.

2. The New Project dialog box appears; click the link below for a screenshot.

VS.NET new project dialog (36K)

As shown in the screenshot, select 'Visual C# projects' and then 'Console Application'.

You can change the project name (shown in the screenshot as 'HelloWorld') and the location. VS.NET initially has a default location of a folder called 'Visual Studio Projects' in the 'MyDocuments' folder on your machine. You can change this default location with the Options... entry from the Tools menu.

3. Click OK and the project is created. A code window appears with the following code:

```
using System;

namespace HelloWorld
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class Class1
{
static void Main(string[] args)
{
//
// TODO: Add code to start application here
//
}
}
}
```

4. Note the creation of a namespace 'HelloWorld'. If you look in the Class View for the project, you will see that the project is HelloWorld and underneath it is a pair of curly braces with a '+' sign to their left and the namespace name, also 'HelloWorld', to the right.

5. If you wish, change the class name from 'Class1' to 'HWorld'. In Class View, the class name is shown as a child of the 'HelloWorld' namespace.

6. Console applications always start at the function Main(). So we need to add some code to this function. Helpfully, VS.NET tells us where to do so - in the commented line which says 'TODO: Add code to start application here'

7. Replace the 'TODO' line with the following (don't forget to remove the comment '//' characters!):

Console.WriteLine("Hello world, from Microbion!!!");

Note that C# is case-sensitive; the upper case letters are important. Note also that when you type the dot character after 'Console', VS.NET gives you a drop-down list of possible methods or properties. If you don't get this, you mis-typed the word 'Console'.

8. That's basically it. Save the file as 'HelloWorld.cs' (at the moment it's still 'Class1.cs').

9. Now compile the project. Choose 'Build' from the 'Build' menu. The program compiles (assuming you typed it in correctly) and you should get a series of messages in the output window below the code window as it happens.

10. Now you can run this masterpiece. It's a console app, so launch a command window and navigate to the folder in which the compiled app 'HelloWorld.exe' is located. You'll find it in your project's directory in folder HelloWorld\bin\Debug. When run, it prints a line to the screen and exits. (You can also run it by using Explorer to navigate to the program directory and double-click on the executable, but if you do then the program just flashes on the screen very quickly and then exits.)

So there you go - how to use Visual Studio.NET to write a simple application. To close this project, choose 'Close Solution' from the File menu.

---

C# How-to 2: Develop a Windows application using Visual Studio .NET

C# How-to #1 showed how to produce the simplest application: display the message ‘Hello, World’ in a console application. This how-to will produce the equivalent in a windows application.

Our application will have a window and two buttons. The first button displays a message box when clicked; the second exits the application by closing the form.

Steps to take

1. In the .NET environment, select Project… from the File menu, then choose Visual C# Projects then Windows Application. Type a suitable project name into the name field (e.g. ‘WinHelloWord’) and ensure the project is going to be saved in an appropriate location. Click OK.

2. You should now have a blank form in the design view of the project. The form is called Form1 (in best VB6 tradition!) but you can change this. Open the Properties window (from the View menu, or by pressing F4). If you have Auto-hide selected, the window will disappear when you move the mouse away from it; to stop this, click the pin in the top right of the window.

Now you can change the name to whatever you like. You want the Text property; if you can’t see it, expand the Appearance section of the window by clicking on the plus sign. Note that there is no Caption property any more.

3. Add two buttons to the form. Move the mouse over the Toolbox window to open it, click the Windows Forms section if this isn’t showing, and look for the button control. To add a button, either click the control then click the form, or just double-click the control. Make sure that two buttons are added to the form; they will have captions of ‘button1’ and ‘button2’.

4. Resize the form and move the buttons so that it is aesthetically pleasing. Change the caption for button1 to 'Say hello' and that of button2 to 'Quit'. Remember that once again you need to change the Text property – there is no Caption property in .NET. Your form should now look something like this:

VS.NET WinForm and buttons

5. Right now the form doesn't do anything. We need to add some code to make it work, and that means adding events to both buttons. This is both similar to VB6, but also very different. In VB6, the events were already added to each control and you just needed to write the appropriate code. In .NET, you have to add each event handler and link it to the control, then write the handler's code.

Adding a click event to a button is very simple. Just double click the button in design mode. If you do that for button1, you will find that .NET has added the following code (to see the code, right click the design window and choose View code from the popup menu):

```
// this is added to the InitializeComponent() function
this.button1.Click += new System.EventHandler(this.button1_Click);
```

and a new function has been added:

```
private void button1_Click(object sender, System.EventArgs e)
{
// add code here to respond to the event
}
```

This looks a bit different to VB, but it's relatively simple. Each control (and the form itself, of course) can respond to different events. To see the events a control can respond to, use the Properties window and click the events button at the top (it has a yellow lightning flash on it). If you do this for button1, you'll see that the events are divided up into sections, and the Click event is in the Action section. The event handler is shown in the rightmost column and you can see that this is shown as button1_Click – which is the name of the new function .NET added to your code. Incidentally, the function doesn't have to be called button1_Click(); you can name it whatever you like, but then you have to change the line which assigns the handler to the button1.Click event.

Anyway, all this means is that when the user clicks the button, the function button1_Click() is called, just as in VB6. Now do the same for button2, and another event handler is added – button2_Click().

6. We now have to add code to the handlers so that something happens when they are clicked. When button1 is clicked, a message box is shown. Add this code to the button1_Click() handler:

```
MessageBox.Show("Hello World, from Microbion!!!", "First message", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
```

The MessageBox class displays a message box (well, obviously) but it has a number of ways of doing it (i.e. the Show function is overloaded). Details are in the .NET documentation. I have used the version which allows a caption, buttons, and icon to be shown. MessageBoxButtons is just an enumeration of the various buttons which can appear in the box, while MessageBoxIcon is an enumeration of the various possible icons.

7. Finally, we add code to the button2_Click() function to close the window (and exit the program) when the button is clicked. Add the following code to the function:

```
this.Close();
```

This at least couldn't be simpler.

8. Save the form and run it by pressing F5. Assuming it compiles okay, you should see a form with two buttons, one of which displays an alert box while the other quits the program.

9. Don't remove this project; I'll be using it in the next how-to, which covers adding more events to the controls.