C# How-to 5: Use the MessageBox class

The MessageBox is probably the easiest way to get some feedback to the user. In C# and .NET, this is very similar to the function provided in VB6, although just different enough to make a quick look worthwhile.

In VB6, to display a message box you used a function called MsgBox() which had a variety of optional parameters in addition to the mandatory parameter which was the text of the box. .NET is very similar but the functionality has changed slightly and of course everything is now an object.

1. Displaying a message box

To do this call the Show() method of the MessageBox class. This method is overloaded so you can display the following box types:

Text only

Text, Caption

Text, Caption, Button(s)

Text, Caption, Button(s), Icon

Text, Caption, Button(s), Icon, DefaultButton

Text, Caption, Button(s), Icon, DefaultButton, Options

The Text and Caption are strings which are displayed in the body and title bar of the box respectively. The buttons parameter is a single value from the MessageBoxButtons enumeration. Slightly disappointingly these are the same old combinations of buttons that you may have seen before, namely:

Abort, Retry, Ignore

OK

OK, Cancel

Retry, Cancel

Yes, No

Yes, No, Cancel

The icon parameter is again a value from the MessageBoxIcon enumeration, and at first sight has more possible icons than previously, such as Error, Hand, Stop, etc. But unfortunately the same four old icons are mapped to these new values (exclamation mark, question mark, information sign, and the cross-in-red-circle sign).

The DefaultButton is also an enumerated value and has the possible values of DefaultButton1, DefaultButton2, or DefaultButton3.

That just leaves the Options parameter. There's a slight difference from VB6 here in that in addition to right-aligned text and right-to-left text display, which were present in earlier versions, there appears to be support for multiple desktops but only to the extent that you can force the box to be displayed on the active desktop.

Note that the ability to associate a help file topic to the message box is no longer present in .NET.

Finally, all these overloaded methods come in two flavours. The other group of methods take an additional parameter which is any object which implements the IWin32Window interface (such as controls and forms). This is supposed to let you display a message box in front of the object rather than in the centre of the screen, but I haven't been able to get this to work yet.

2. Instantiating a message box

To cut this short, you can't. In other words you can't do:

```
MessageBox mb;
mb = new MessageBox();
```

If you do, you get a compiler error:

```
'System.Windows.Forms.MessageBox.MessageBox()' is inaccessible due to its protection level
```

But you can display a message box from another box. This code works fine:

```
if (MessageBox.Show("Do you want to show another box?", "Confirm another box",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)==DialogResult.Yes)
MessageBox.Show("Here is another box!", "Second box", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
```

Note though that the first box is removed before the second box appears and control returns directly to the calling routine not to the first message box.

3. Returning values from a message box

In VB6, the MsgBox function returned an Integer which was one of a series of constants representing the button that was clicked. In .NET MessageBox returns one of the DialogResult enumerated values, again corresponding to the button which was clicked.

If you want to check the return value in a box with two buttons, the best way is something like:

```
if (MessageBox.Show("Click a button", "", MessageBoxButtons.YesNo)==
DialogResult.Yes ) {
// code if Yes clicked here
} else {
// code if No clicked here
}
```

If you have three buttons you can't do this as you would have to call the Show() method twice. Instead you can do this:

```
DialogResult dr;
dr=MessageBox.Show("Click a button", "", MessageBoxButtons.YesNoCancel);
if (dr==DialogResult.Yes) {
// code for Yes here
} else if (dr=DialogResult.No) {
// code for No here
} else {
// code for Cancel here
}
```

That's the end of the look at the MessageBox function in C#. I hope that's been of some help. The technique is just sufficiently different in .NET to cause confusion for previous VB programmers, but it's clear enough when you understand what's going on.