C# How-to 10: Add a context menu

Adding a context menu to a .NET window could not be easier. This short how-to demonstrates how to do this, again using Simple Editor as the test bed.

1. Add a context menu component

The .NET framework comes with a separate context menu component. You add this to the window from the toolbox in the usual way; it will be given the name 'contextMenu1' by default, but I changed this to 'popupMenu1' mainly because I still think in terms of popup menus rather than context menus!

When the context menu component is selected, the menu designer appears at the top of the window form in the same way as the main window menu does. The menu items can then be added from here, and event handlers added in the Properties window. I added three menu items to this menu: popupCut, popupCopy, and popupPaste. These are the same editing commands as were already present on the Edit menu and the toolbar. Event handlers need to be added for the Click event for each menu item.

We also need a handler for the Popup event of the context menu itself (not the individual menu items). This will allow us to change the menu before it is displayed, since the Popup event is called just prior to displaying the context menu.

2. Attach the context menu to the RichTextBox control

Many controls, including RichTextBox, have a ContextMenu property; you can add a context menu to the control simply by setting this property to one of the context menu components on the form. In this case, there is only one, the newly added popupMenu1 component.

That's all you need to do to activate the menu! If you build the project now, you will see that right-clicking the mouse in the text box brings up the context menu. Of course, this doesn't actually do anything useful yet, but it appears on cue, and it took no code at all to make this happen.

3. Add code to the event handlers

Firstly, we add code to the Popup event handler. This performs the same function as already carried out for the Edit menu and the toolbar - namely, to make sure that there is some text to copy or cut (and enable or disable the Cut/Copy entries accordingly) and to make sure that there is something to paste. There's nothing new about the code, which looks like this:

```
private void popupMenu1_Popup(object sender, System.EventArgs e) {
// check to see if we can cut/copy, paste anything; then enable/disable menu items
// as required
IDataObject iData=Clipboard.GetDataObject();
popupPaste.Enabled = iData.GetDataPresent(DataFormats.Text);
if (rtfMain.SelectionLength==0) {
popupCut.Enabled=false;
popupCopy.Enabled=false;
}
else {
popupCut.Enabled=true;
popupCopy.Enabled=true;
}
}
```

The other event handlers are much simpler, and you can see identical code in the handlers for the Edit menu items:

```
private void popupCut_Click(object sender, System.EventArgs e) {
rtfMain.Cut();
}
private void popupCopy_Click(object sender, System.EventArgs e) {
rtfMain.Copy();
}
private void popupPaste_Click(object sender, System.EventArgs e) {
rtfMain.Paste();
}
```

And that, in fact, is all you need to do. You could easily add other menu entries to this context menu, and you could add other context menus to (for example) the form itself, or the toolbar or status bar. It's a simple, effective system.

As always the latest version of Simple Editor incorporating these changes can be downloaded from the link below.

Download code for Simple Editor - latest version (71K)