

A Tax Form Application in VB.NET

Illustrating the Form Printing Features of PrintForm.NET



Overview

This sample demonstrates how to use `PrintForm` and `PrintChainManager` to create a finished application with multiple forms printed as a single print job.

In its simplest form `PrintForm` is designed to print a single Windows Form, or part thereof - so what do you do if you want to present a number of forms to the user and then have them all printed out as a single print job? Obviously you can add a `PrintForm` control to each form and then call `Print` on each one in turn, but this will result in each page being printed as a separate print job; and it also means you can't print preview them in one go either.

What is called for is a solution that allows you to chain together the output of multiple `PrintDocument`-based components, passing the .NET `PrintController` and `PrinterSettings` between them so that they all print their output into the same print job. We have included this functionality in the `PrintChaining` assembly supplied with `PrintForm`.

The `PrintChaining` assembly contains the interface definition for `IChainedDocument`, and the `PrintDocument`-based component `PrintChainManager`. When `PrintChainManager` is supplied with a list of objects that implement the interface `IChainedDocument` (`PrintForm` implements this, as do all the components in our `PrintAdapters` printing library, see http://www.winformreports.co.uk/features_pa.htm) it does the necessary work to chain them all together when printing.

Application Architecture

The `TaxForm` application consists of a main form (`frmMain`), and a set of subsidiary forms (`frmInstructions`, `frmTaxForm1`, `frmTaxForm2`). In the Form Load event handler each of these is reparented to a panel on `frmMain` using a call to the following subroutine, which removes their borders and `TopLevel` setting, and sets them to fill the client area of the new Parent:

```
' Reparent the form to the parentControl and setup standard options
Private Sub SetupTaxFormPage(ByVal form As Form, ByVal parentControl As Control)
    form.TopLevel = False
    form.FormBorderStyle = FormBorderStyle.None
    form.Parent = parentControl
    form.Dock = DockStyle.Fill
    form.Visible = True
End Sub
```

Each form has been created to mirror a real tax form page, based on the pdf file of the UK Income Tax return for 2003

(http://www.inlandrevenue.gov.uk/pdfs/2002_03/tax_return/sa100.pdf). Where complex formatted text was required on screen, we used a `RichTextBox` control with the corresponding RTF files produced by cutting and pasting from Adobe Acrobat into Wordpad, then saving as an RTF file. Each of these RTF files is included in the application as an embedded resource (include in project using 'Add Existing Item...', then set the **Build Action** property to **Embedded Resource**).

To load the RTF embedded resource into the corresponding `RichTextBox` control at runtime, we use code similar to:

```
Me.rtbInstructions0.LoadFile(GetType(frmInstructions).Module.Assembly.GetManifestResourceStream("TaxForm.frmInstructions.rtbInstructions0.rtf"), RichTextBoxStreamType.RichText)
```

Note: the key point here is to get the fully qualified name of the embedded resource (highlighted in the above code line) exactly right. A handy tip for checking what you should use is to open the `ILDASM.exe` tool that comes with Visual Studio, point it at your compiled executable, and then double click on the **MANIFEST** section. This lists, among other things, the fully qualified names of all the embedded resources.

The final code for the main application is some logic to show a specific child form when a 'toolbar' label is clicked. Since the child forms (`frmTaxForm1`, `frmTaxForm2` etc) have all been reparented to the same panel, and set `DockStyle.Fill`, the only one which will be visible is the one at the top of the Z-Order. The Z-Order of contained controls in .NET is governed by their position in the container's `Controls` collection, so to bring a given child form to the top we move it to the start of the `Controls` collection as follows:

```
Me.pnlFormParent.Controls.SetChildIndex(theForm, 0)
```

Now we have a main application with a number of data entry 'pages' which we can make visible inside a Panel on the main form. Because each of the child forms has been set `AutoScroll = True`, they are scrollable when placed inside the smaller containing Panel.

Adding Printing Support

Having completed the design of the data entry forms we now need to make them printable. This is actually the simplest part of the whole process:

1. Add a `PrintForm` component to each of the child forms
2. Add a `PrintChainManager` to the main form
3. Add the following code to the main form's `Load` event handler, to create the child forms, reparent them to our display Panel (as described earlier) and add their `PrintForm` reference to the `PrintChainManager.Documents` collection. (Note: the calls to `_forms.Add` are to enable us to relate a `PrintForm` reference to its containing Form for use in the toolbar code which switches the Z-Order of the child forms):

```
Dim page0 As frmInstructions = New frmInstructions()  
Me.SetupTaxFormPage(page0, Me.pnlFormParent)  
_forms.Add(page0.PrintForm1, page0)  
PrintChainManager1.Documents.Add(page0.PrintForm1)
```

```
Dim page1 As frmTaxForm1 = New frmTaxForm1()  
Me.SetupTaxFormPage(page1, Me.pnlFormParent)  
_forms.Add(page1.PrintForm1, page1)  
PrintChainManager1.Documents.Add(page1.PrintForm1)
```

```
Dim page2 As frmTaxForm2 = New frmTaxForm2()  
Me.SetupTaxFormPage(page2, Me.pnlFormParent)  
_forms.Add(page2.PrintForm1, page2)  
PrintChainManager1.Documents.Add(page2.PrintForm1)
```

4. Add a standard .NET `PrintPreviewDialog` component to the main form. Set its `Document` property to the `PrintChainManager` reference
5. Now add a menu item click handler to call `PrintPreviewDialog.ShowDialog`:

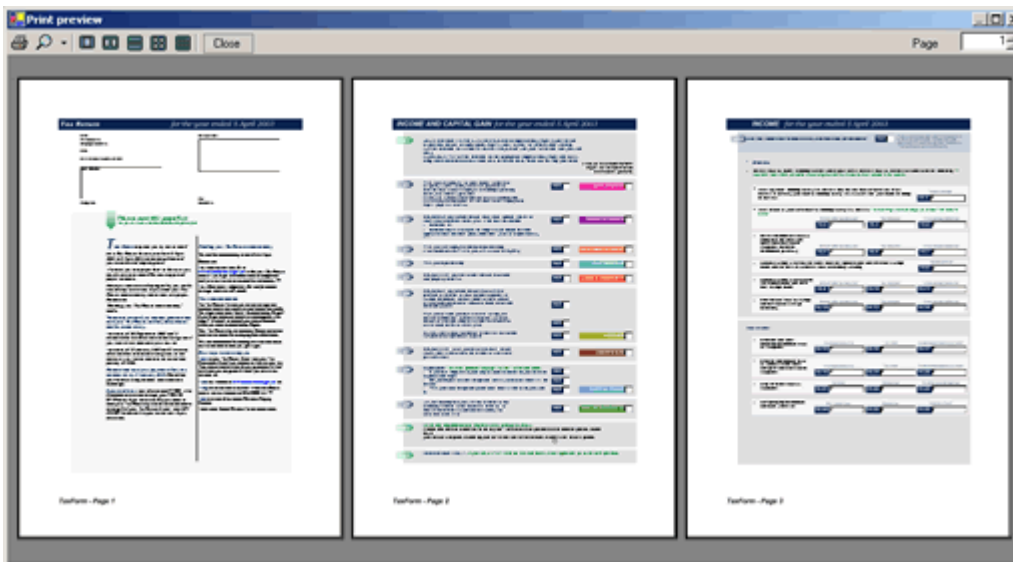
```
Me.PrintPreviewDialog1.ShowDialog()
```

The Finished Product

The main application:



And the corresponding print preview... because PrintForm renders the form contents in a resolution independent manner the result is a high quality printout, without any of the effort of writing all the printing code yourself:



For more information about PrintForm, and other .NET solutions, please visit the WinformReports website at <http://www.winformreports.co.uk>

June 6, 2003

© 2002-2003 TMG Development Ltd