

# Printing in .NET with PrintForm, a Simple Invoice Printing Application

The sample code for this article is available at <http://www.winformreports.co.uk/invoiceprinter.zip>  
You will also need the trial version of PrintForm which is available for download from <http://www.winformreports.co.uk/downloads.htm>

## Introduction

In this white paper we will explore the process of adding printing features to an application, discussing both the use of the standard .NET framework printing components (PrintDocument, PrintPreviewDialog etc) and the use of PrintForm .NET from TMG Development Ltd.

PrintForm .NET helps the software developer add printing support to an application quickly and easily with practically no coding - by rendering a printed page whose layout and content is based on the Windows .NET Forms they have already designed and coded using Visual Studio .NET.

## Background

Typically the developer has two paths open to them when adding printing support to an application: 1) Use one of the 'established' reporting tools for .NET, such as Crystal Reports or ActiveReports, or 2) write a custom printing solution using the basic printing framework provided in the .NET runtime. Later on in this article we will discuss a third option based on printing Windows Forms using GDI+ and the PrintForm .NET component.

Reporting tools are typically designed for producing reports based on repeating lines of data, and coercing them into producing a custom printing solution can be a difficult task as they all have their own quirks and a definite 'learning curve'. Add to this the inflexibility of their page layout options and developers are often left with the task of writing a custom printing solution from scratch.

The .NET runtime provides a printing framework based on a group of classes in the System.Drawing.Printing namespace. The most commonly used of these are **PrintDocument**, **PrintPreviewDialog**, **PageSetupDialog** and **PrintDialog** which provide the bulk of the core functionality required in a typical application.

The framework **PrintDocument** component can be used in two main ways:

- n For simple printing requirements you would add the **PrintDocument** component to a Windows Form, then add programming logic in your application that renders the printed page using GDI+ graphics functions in the **PrintPage** event handler.
- n For more complex printing you would derive a new component from the **PrintDocument** component and override the **PrintPage** event. This new component could then form the basis of your own custom printing solution, which you could reuse in future applications.

These features are discussed in detail in the online help, and in the Microsoft Quickstart Samples at <http://www.gotdotnet.com>.

While the custom printing solution is without doubt the most flexible, it can be a time consuming process producing an aesthetically pleasing layout from low level GDI+ calls of the form 'draw a line from A to B', 'draw text at C', 'draw a circle to simulate a radio button' etc. In some cases it can be very difficult to persuade controls you may be using on your Form e.g. a Chart, BarCode, Map etc to draw themselves at all in your printout.

At this point we introduce PrintForm..! PrintForm .NET is derived from PrintDocument - this means it can be dropped onto the design surface of a Windows Form in Visual Studio .NET and linked seamlessly with the core .NET printing framework. PrintForm is able to directly render any GDI+ based control onto the Graphics surface provided by either the **Printer** or the **PrintPreviewControl**. It also includes support for the majority of controls based on legacy Win32 technology (including the '.NET' controls like TextBox, ComboBox etc which are thin wrappers onto their Win32 counterparts).

## Sample Application

To demonstrate the process of adding printing support to an application using PrintForm we will produce a simple invoice printing application. We will assume that the data to be printed would be supplied by a business logic layer component and will omit this for the purpose of this example.

**Step 1:** Create a new Windows Forms application in Visual Studio .NET. In this example we'll use VB.NET, but PrintForm .NET works with any .NET compatible language.

**Step 2:** Add a Panel called Panel1 to the Form. This Panel will hold the controls we wish to appear on the printed page. To allow us to make it more like the size of a page set the Panel's **AutoScroll** property true and set it's **AutoScrollMinSize** to 500, 1000 - this gives us a viewport onto a scrolling region approximately the size of a standard piece of printer paper.

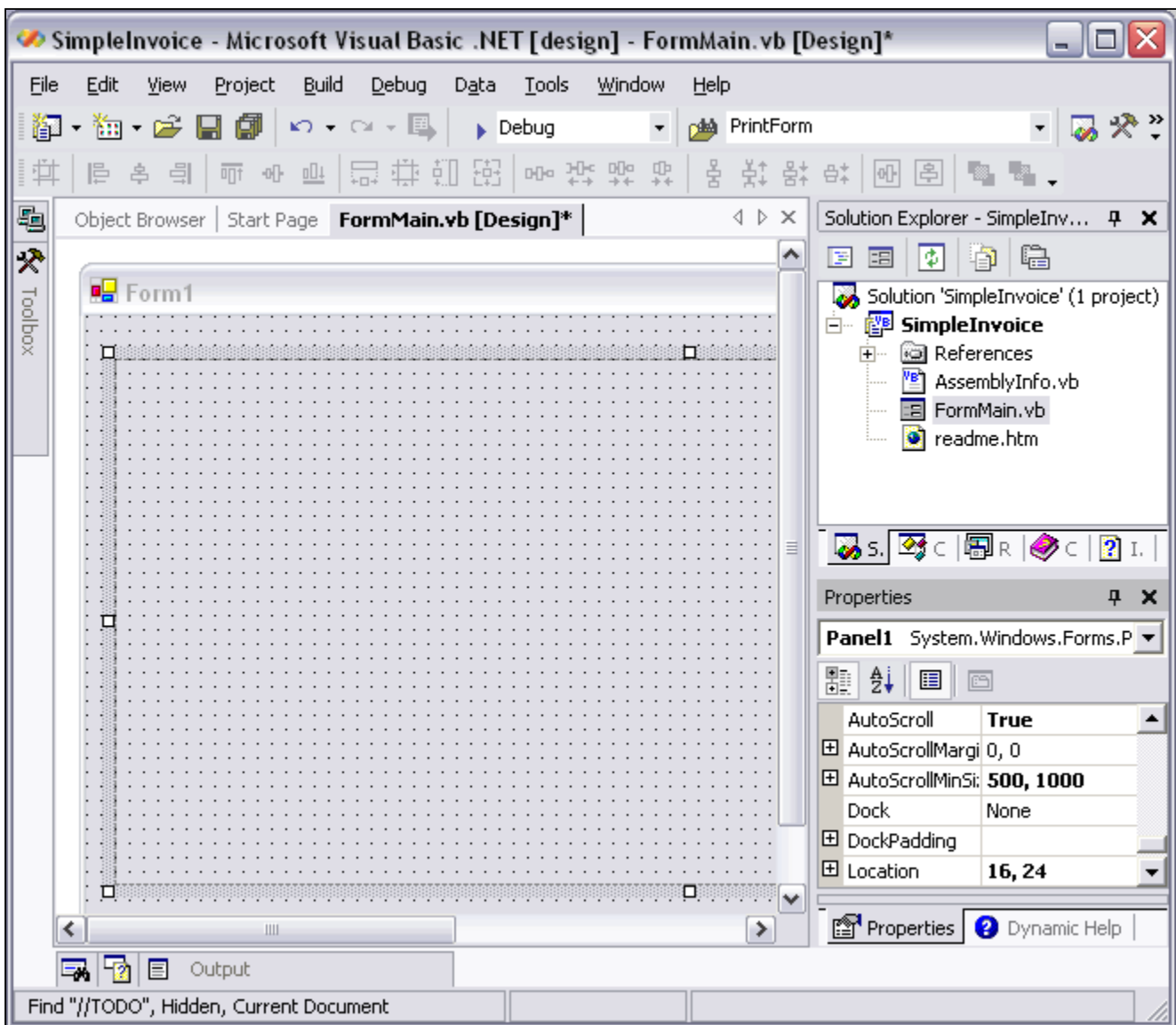


Figure 1: Adding an AutoScrolling Panel to Represent the Invoice

**Step 3:** Now we add Windows Forms controls to the scrolling Panel1 as required to make up the layout of the invoice. Note that where we intend to customise the content of the control at runtime we use the default text of the control to hold the format string that will be used in a call to **String.Format()**.

At this point we also add the necessary printing controls - the standard .NET **PrintPreviewControl**, and two printing components from the **TMGDevelopment.Windows.Forms** namespace: **PrintForm** and **PrintChainManager**, by dragging them into the form from the Toolbox.

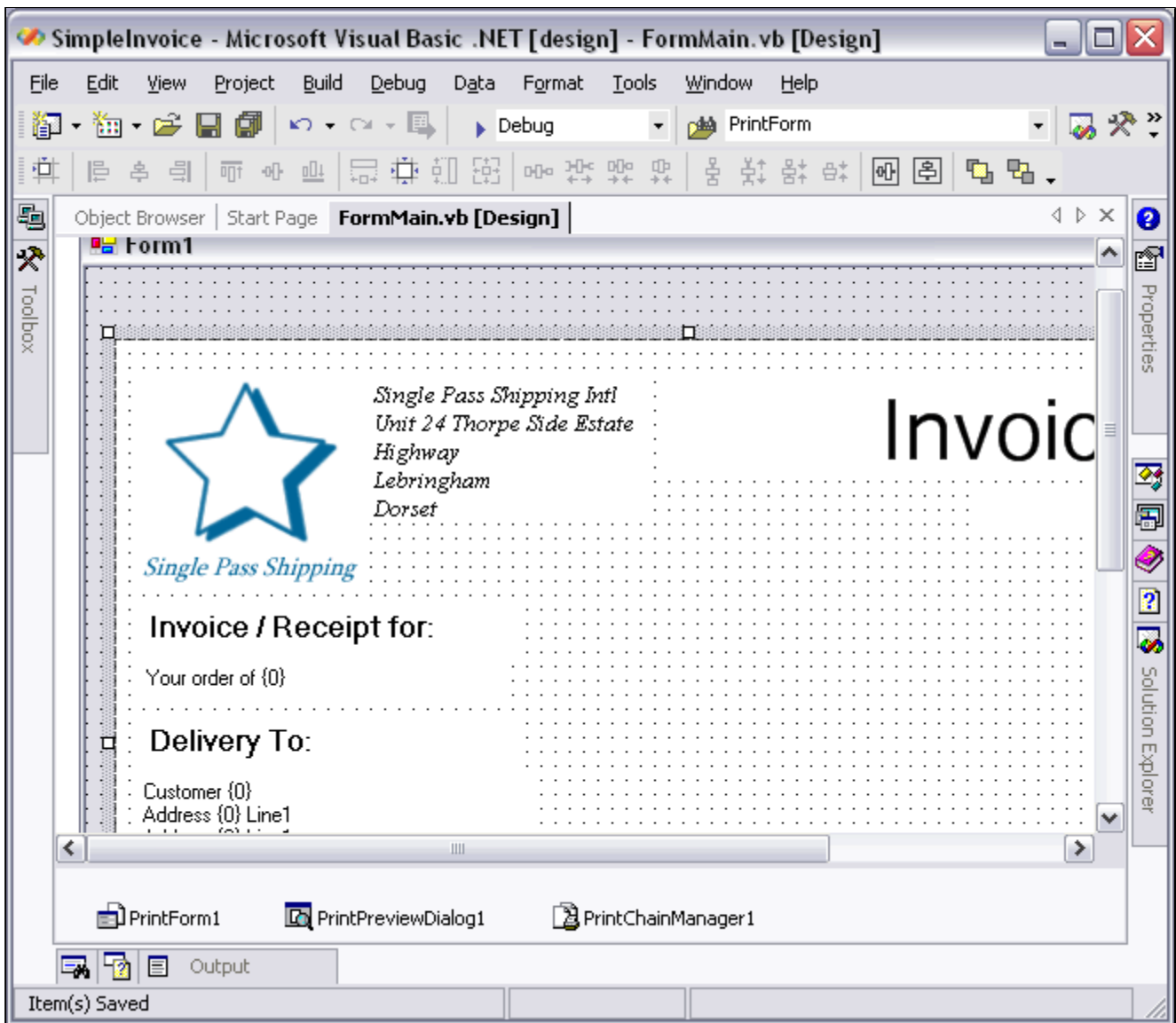


Figure 2: Designing the Invoice Layout

**Step 4:** Now we tell `PrintForm` which controls it is going to print by setting its `BodyContainer` property to the `Panel1`, and then tell the `PrintPreviewDialog` which `PrintDocument` it is going to render by setting its `Document` property to `PrintForm1`. Remember from the Background discussion above that `PrintForm` is derived from `PrintDocument` so the `PrintPreviewDialog` is quite happy to have a reference to `PrintForm` instead of `PrintDocument`. In fact the Visual Studio IDE will even prompt you with all the components on the form which are derived from `PrintDocument` - in this case it will list both `PrintForm1` and `PrintChainManager1`.

**Step 5:** Test a single page printout. Add a button to the form to trigger a call to `Me.PrintPreviewDialog1.ShowDialog()` and run the app. Here is the resulting print preview:

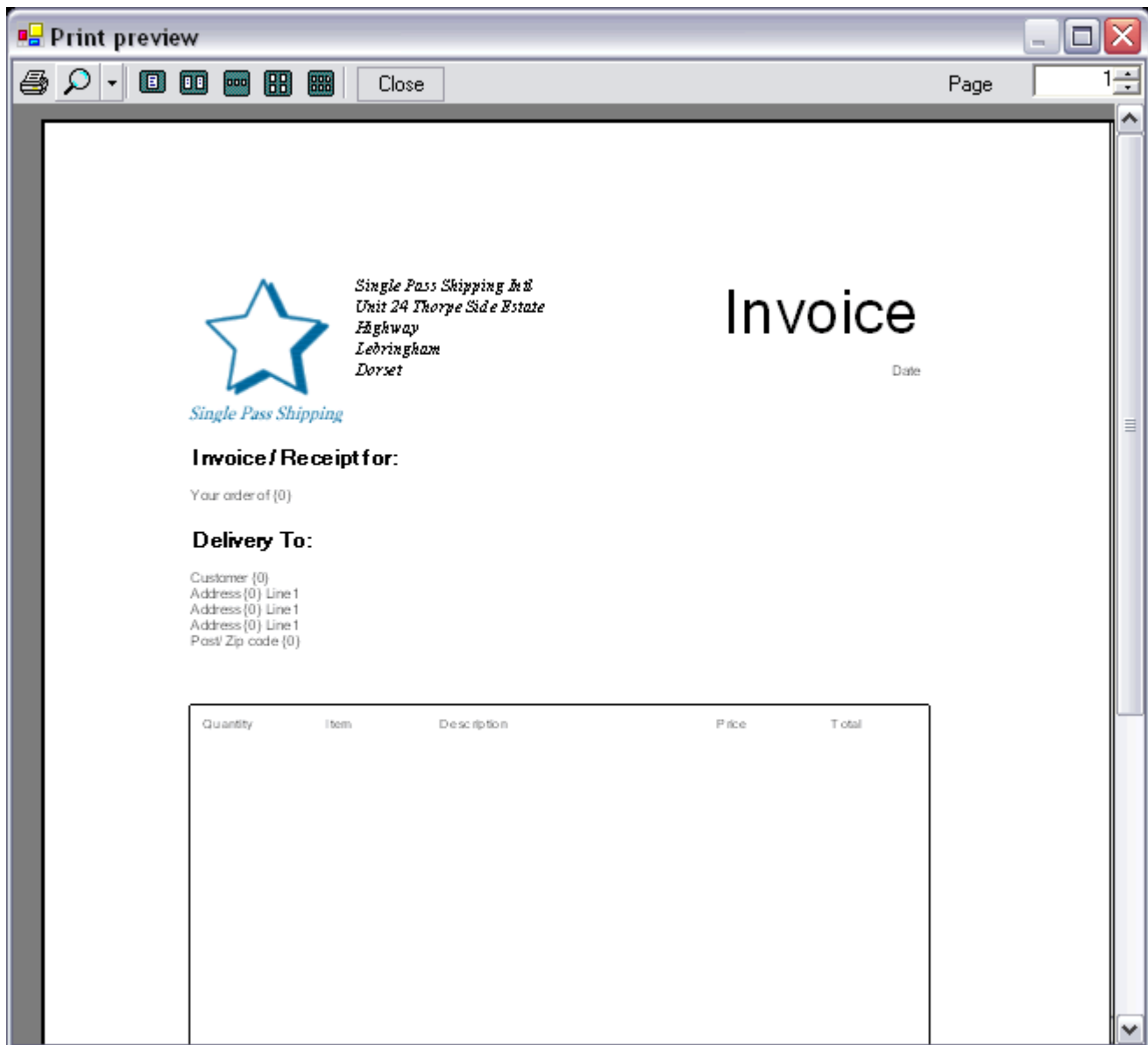


Figure 3: Single Page Print Preview

**Step 6:** Now we will add code using the PrintChainManager to print multiple invoices in the same print run. As mentioned earlier PrintChainManager is also derived from PrintDocument and manages the process of composing the output of multiple PrintDocuments into a single printout. All that is required for an individual PrintDocument to participate in this process is for it to implement the interface TMGDevelopment.Printing.IChainedDocument. All of the TMGDevelopment printing components (PrintForm, PrintAdapters, PrintControls) do and you can easily add support to your own PrintDocuments by implementing the interface as documented in the help files. Add the following code to the Print button click handler:

```
Dim i, count As Integer
count = 5
' add the required number of documents in
For i = 1 To count
    Me.PrintChainManager1.Documents.Add(Me.PrintForm1)
Next
' setup the print preview dialog
Me.PrintPreviewDialog1.Document = Me.PrintChainManager1
Me.PrintPreviewDialog1.ShowDialog()
```

Finally, we add some code to populate the controls with the contents of each invoice as they are being printed. We won't


go into the detail of this as it is a simple matter of creating some random data for customer name, address and items purchased, you can download the accompanying sample code if you want to see the details.

The most important thing to note about this page contents setup step is that it is performed in response to the `PrintChainManager ChainPrintingPage` event, which is called immediately before `PrintChainManager` prints a page. There is also a corresponding event, `PrintPageComplete`, which is fired after the page is complete and is a useful place to do any custom drawing on top of the page contents.

**Step 7** : Now run the application again, and click to Print Preview. Here's a single completed invoice:

Print preview

Close Page 1



*Single Pass Shipping*

*Single Pass Shipping Ltd  
Unit 24 Thorpe Side Estate  
Hagkway  
Ledbringham  
Dorset*

# Invoice

20/10/2005

**Invoice / Receipt for:**

Your order of 02/08/2005 12:20:37

**Delivery To:**

Customer 1  
Address 1 Line 1  
Address 1 Line 1  
Address 1 Line 1  
Post/Zip code 1

Quantity	Item	Description	Price	Total
4	Product 1	Product 1 Description	£28.83	£115.32
8	Product 2	Product 2 Description	£74.21	£593.68
6	Product 3	Product 3 Description	£81.69	£490.14
1	Product 4	Product 4 Description	£38.64	£38.64
5	Product 5	Product 5 Description	£18.17	£90.85
<b>Total Price</b>				<b>£1,328.63</b>

Figure 5: The Final Invoice

and viewing all five invoices in the final print document:

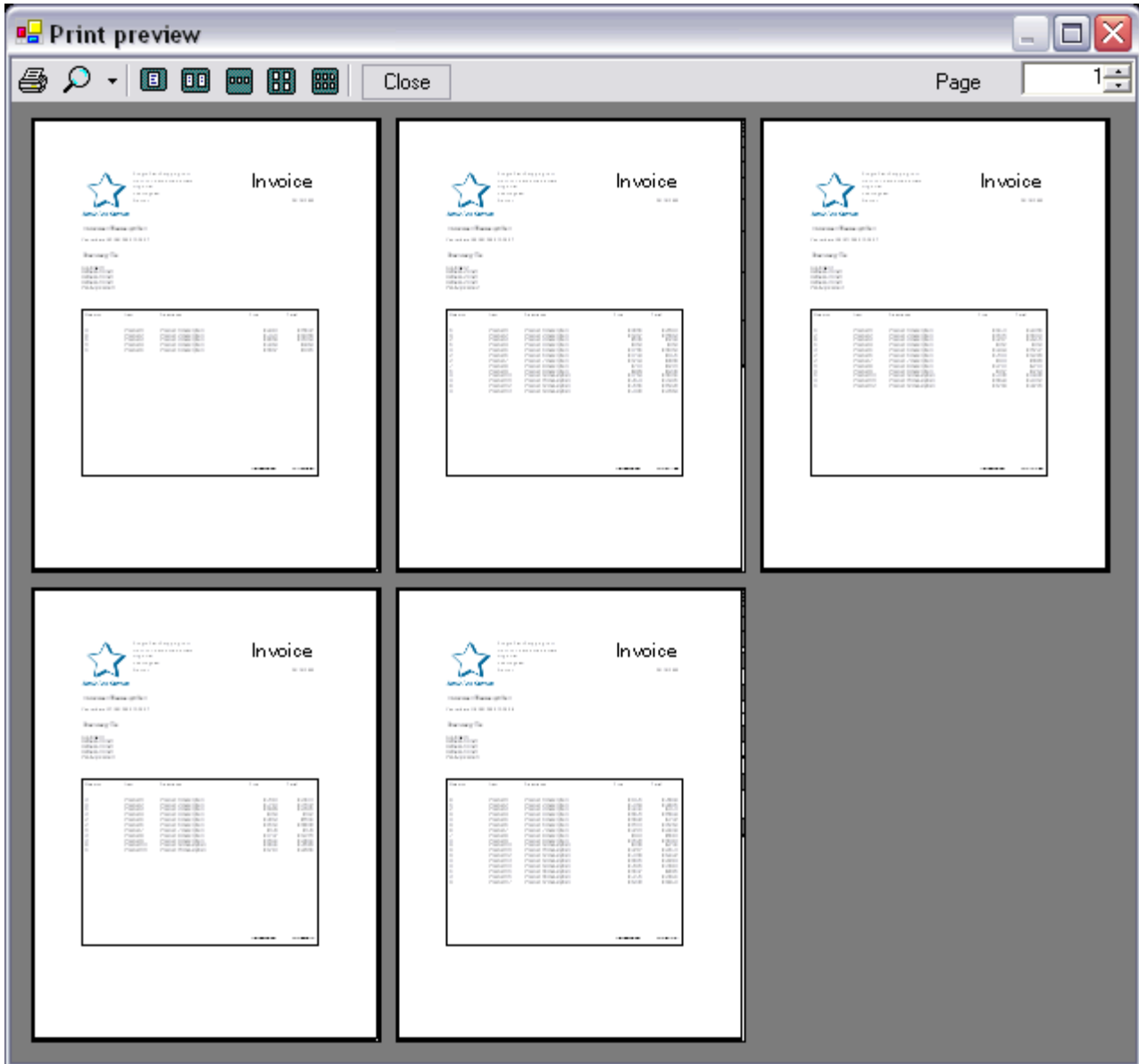


Figure 6: Showing Five Invoices In the Same Print Document Using PrintForm and PrintChainManager

## Summary

That's about it for this sample. We've added printing support to an application using PrintForm, a process involving much less effort than would have been required if we'd coded the printing mechanism by hand.

If you'd like to try out this invoice application you'll need the sample code for this article and the free trial version of PrintForm. The download links for these are listed at the top of this article.

The trial version of PrintForm also contains further samples illustrating how to add headers and footers to each page, and how to produce multipage reports with different layouts on each page with PrintForm. We also produce libraries of controls (barcode, line, rounded rectangle etc) with enhanced printing capabilities, see the PrintControls .NET download, and components for printing TreeView, ListView, RichTextBox etc. - see PrintAdapters .NET.

Further information about PrintForm and other controls by TMG Development Ltd can be found at the Winform Reports website at <http://www.winformreports.co.uk>

This article and sample code © TMG Development Ltd 2005  
Version 1.1