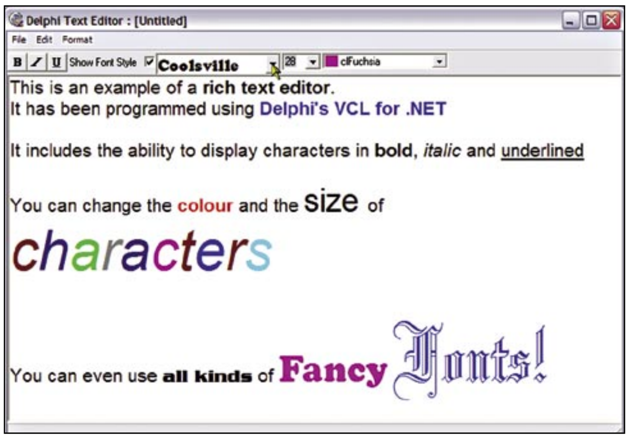
Programming with Delphi 2005

Huw Collingbourne explains how to use Delphi 2005 to build your own text-editing application for .NET

Last month, we used Delphi to create a Win32 application; in other words, a traditional Windows program (we've included a PDF of this article on the cover disc; just click on the Editorial tab). This month, we concentrate on writing applications for the .NET Framework. Delphi gives us the choice of creating two significantly different types of .NET application. We can either create a Windows Forms application or a VCL (Visual Component Library) Forms application. Before going any further, let's consider the differences between Windows Forms, VCL Forms and Win32.

Win32 programs are compiled into machine code, which is executed by the computer hardware. Applications written for .NET, on the other hand, are compiled into an intermediate language, which is run not by the hardware but by a software system known as the Common Language Runtime (CLR). The CLR provides a protective layer between the hardware and any running programs. A Win32 program has no such protection; if it encounters an error, it can easily crash. If it allocates memory and fails to deallocate it later, it may cause memory leaks, which can degrade the performance of your entire system.



This is an example of a rich text editor.
It has been programmed using Delphi's VCL for .NET

It includes the ability to display characters in **bold**, *italic* and underlined

You can change the colour and the size of

characters

You can even use **all kinds** of **Fancy** Fonts!

For an example of the kind of powerful applications possible with Delphi, try out our VCL Editor project on this month's cover disc.

SAFETY FIRST

The CLR looks after object creation and memory management of .NET applications. It performs automatic memory garbage collection to reclaim resources when objects are no longer needed, which reduces the risk of memory leaks. It also handles errors in order to minimise the risk of disastrous program crashes.

The .NET Library contains a hierarchy of classes, which supply various controls such as Forms and Buttons and can help you do maths, save and load files, and perform all kinds of other useful tasks. The .NET Library could be said to supersede other class libraries such as Microsoft's Foundation Classes (MFC) for C++ and the Win32 VCL for Delphi.

This being so, you may wonder why Delphi 2005 includes a new version of the VCL for .NET. One reason is that Delphi programmers are already familiar with the VCL and may prefer to carry on using it. Moreover, existing Win32 Delphi programs can be converted, relatively painlessly, for use with the .NET VCL, as we explained last month. This month, we explore some of the differences between Delphi's two alternative types of .NET project, while creating a simple text editor using both Delphi's VCL Forms and Microsoft's Windows Forms.

A VCL EDITOR

Let's begin by using the VCL. Select File | New | VCL Forms Application – Delphi For .NET. In the Tool palette, find the Win32 category. Drop a TRichEdit control onto the blank form. With this control selected, use the Object Inspector to set its Align property to alClient. This causes it to fill the form. We don't want the rich edit control to display its name, so double-click the Lines property. A String List editor will pop up. Delete the text 'RichEdit1' and click OK.

Now, select TMainMenu from the Standard category of the Tool Palette and drop a menu onto the form. The small box, which represents the menu, won't be visible when the program is run. Double-click the box to load the menu designer. Currently the menu is blank. Let's add a File menu. In the Object Inspector, enter 'File' next to the Caption property. This causes a top-level file menu to be added in the menu editor and moves the selection to its right.

If you wanted to add another top-level item, you'd do so by adding items to the right of the File menu. In fact, we want to add drop-down items. To do this, click the File menu, then select the blank item that appears beneath it. Give it the caption 'Open'. In a similar way, add two more drop-down items with the captions 'Save' and 'Exit'.

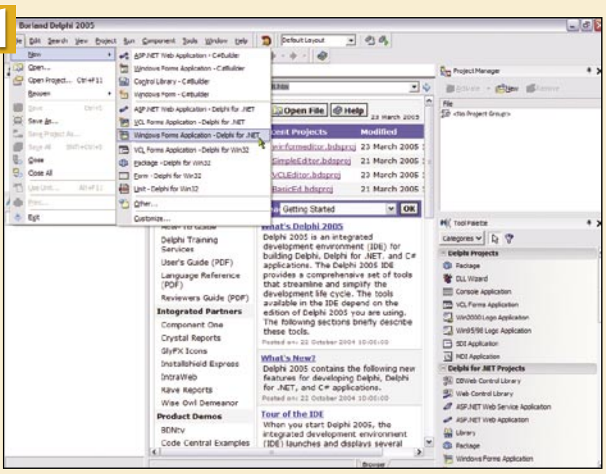
Now we'll add some code to each drop-down menu item. In the menu editor, double-click the Open item. This creates a procedure called Open1Click(). Edit this so that it looks like the following:

```
procedure TForm1.Open1Click(Send
er: TObject);
```

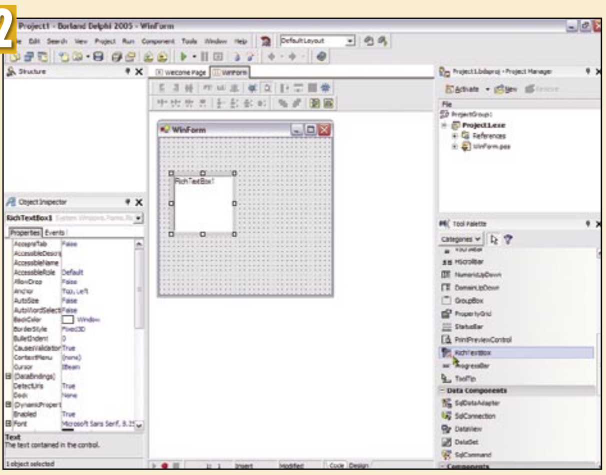
Continued on p172

How to obtain Delphi 2005
We included a copy of Delphi 2005 Personal Edition on last month's cover DVD. You can order a copy for £2.50 (inc P&P) by calling Dennis Direct on 01789 490215

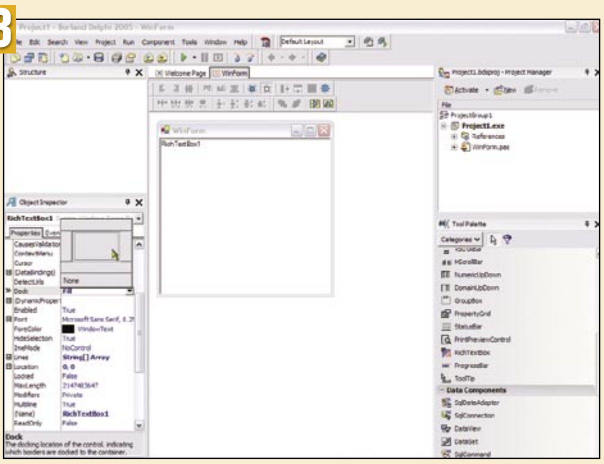
Walkthrough: a text editor in six steps

1


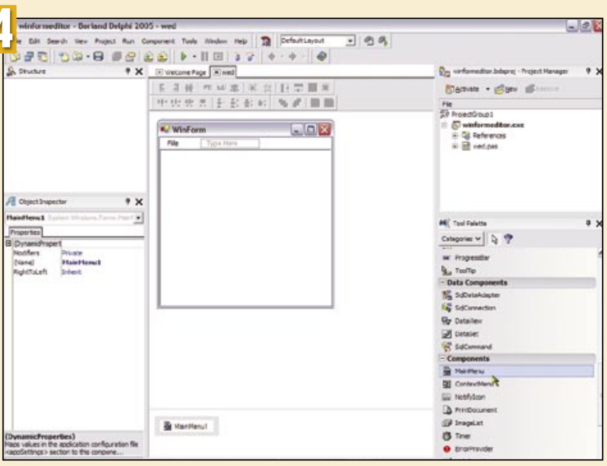
The first thing to do when beginning any new project in Delphi is to select the target platform. Here, we want to build a 'standard' .NET program rather than a program for Delphi's VCL, so you'll need to select 'Windows Forms Application for .NET' from the File | New menu.

2


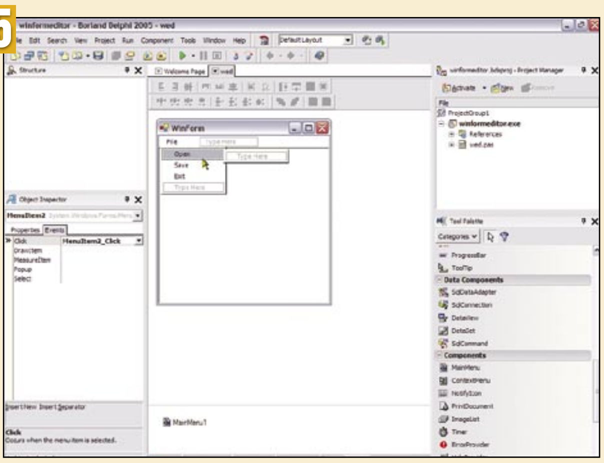
Delphi automatically creates an empty Windows Forms application with a blank form at the centre of the design workspace. Click the RichTextBox component in the Tool Palette, then click the blank form to drop a rich text box onto it. This supplies the main editing features to the application.

3


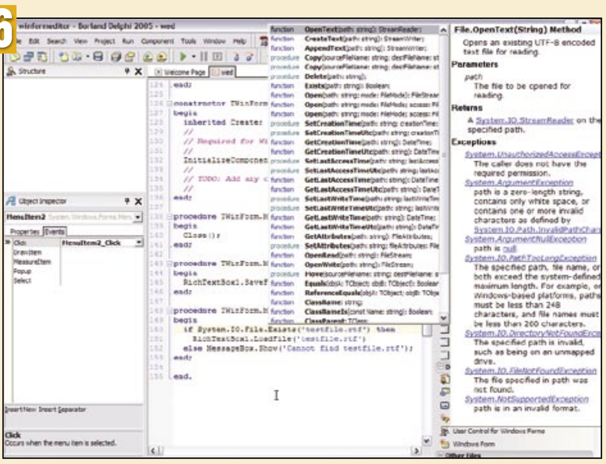
In order to make the RichTextBox control fill the entire form, you need to set its Dock property. Do this by clicking Dock in the Properties page of the Object Inspector. A box pops up to let you pick the kind of docking you need. Click the centre of this box.

4


Drop a MainMenu item from the Components category of the Tool Palette onto the form. Notice that this is a non-visible control so it's automatically positioned beneath the form. To create a labelled menu item, select the first item on the form itself and enter the text 'File'.

5


Continue to add drop-down menu items by entering 'Open', 'Save' and 'Exit' into the text areas of each menu item. When you want to add code to a menu item, double-click it. Alternatively, select Click from the Events page of the Object inspector. This takes you into the code editor.

6


```
end;

constructor TWinForm
begin
  inherited Create;
  //
  // Required for Wi
  //
  InitializeComponen
  //
  // TODO: Add any c
  //
end;

procedure TWinForm.M
begin
  Close();
end;

procedure TWinForm.M
begin
  RichTextBox1.SaveF
end;

procedure TWinForm.M
begin
  if System.IO.File.Exists('testfile.rtf') then
    RichTextBox1.LoadFile('testfile.rtf')
  else MessageBox.Show('Cannot find testfile.rtf');
end;

end.
```

You'll now need to enter the code for each click-handling method, as explained in this month's tutorial. Notice that, when you enter a full stop after the name of a .NET class or namespace, a selection list of available functions and properties appears. Just click the item you need.

```
    begin
        if FileExists('testfile.txt') then
            RichEdit1.Lines.LoadFromFile('testfile.txt')
        else ShowMessage('Cannot find testfile.txt!');
    end;
```

Here FileExists() is a function provided by the VCL. It returns a true value if the named file exists on disk, otherwise it returns false. RichEdit1 is an instance of the TRichEdit class. If you scroll up the code, you can see that Delphi has automatically added this declaration:

```
    RichEdit1: TRichEdit;
```

TRichEdit is the VCL's 'wrapper' around the rich edit control provided by Windows. Its Lines property represents the individual lines of text, and LoadFromFile() is a VCL method that loads a file from disk into the Lines of the rich edit control. Finally, ShowMessage() is another VCL procedure that simply displays the specified text.

Now double-click the Save item in the menu editor and edit the Save1Click() method as follows:

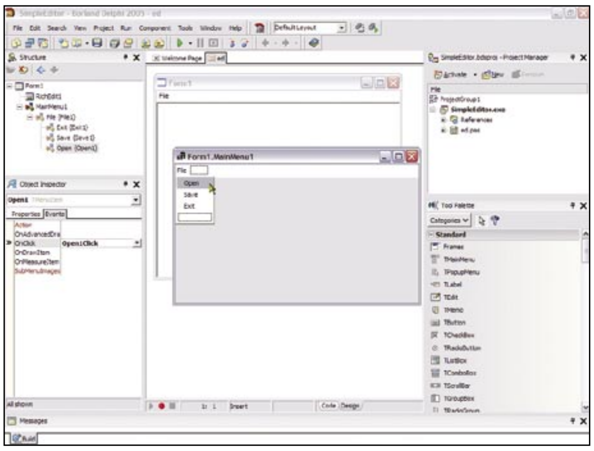
```
    procedure TForm1.Save1Click(Sender: TObject);
    begin
        RichEdit1.Lines.SaveToFile('testfile.txt');
    end;
```

Finally, double-click the Exit item and edit its method as follows:

```
    procedure TForm1.Exit1Click(Sender: TObject);
    begin
        Close;
    end;
```

And that's it. You can now run the program by pressing F9. If you have any problems, load the completed project, SimpleEditor.bdsproj, from the cover DVD. When it runs, select File | Open. You'll see an error message because the specified file doesn't yet exist. Try entering a few lines of text. The rich edit control provides basic editing commands such as <Control-Left> and <Control-Right> to move backward and forward one word at a time, and <Control-Z> to undo the last sequence of deletions.

Once you've entered text, select File | Save. To verify that the text has been saved to a file, delete the text and select File | Open. Your saved text will be reloaded. Of course, a lot more work needs to be done in order to make this a really useful editor. But for now, let's see how to create a similar application using the .NET Windows Forms library instead of the Delphi VCL.


Notice that, when you design a VCL application, the menu is edited in a separate window rather than on the form itself.

A WINDOWS FORMS EDITOR

Start a Windows Forms Application Delphi For .NET project. Notice that the Tool Palette displays fewer categories than previously and many of the components have different names. The rich editing control, for example, isn't called RichEdit but RichTextBox. Drop a RichTextBox onto the form. To make it fill the form, we no longer set the Align property. Instead, we set its Dock property. When you press the down arrow next to Dock in the Object Inspector, a diagram pops up. Click the centre box in the diagram to cause the RichTextBox to dock against all four sides of the form. To delete the text from the control, select its Text property and delete 'RichTextBox1'.

Now add a MainMenu from the Components category of the Toolbox. Notice that in the Windows Forms designer, all non-visible controls such as menus are placed beneath the design workspace rather than on the form itself. To add menu items, select the MainMenu1 control. A blank menu appears on the form with the first item containing the text, 'Type Here'. To add a file menu, enter the text 'File' onto the blank menu. Now carry on adding the three other labels to the drop-down menu: 'Open', 'Save' and 'Exit'. Double-click the Open menu item to create an empty event-handling method and add this code between the BEGIN and END keywords:

```
    if System.IO.File.Exists('testfile.rtf') then
        RichTextBox1.LoadFile('testfile.rtf')
        else MessageBox.Show('Cannot find testfile.rtf');
```

In a similar way, create an event handler for the Save item and add this code:

```
        RichTextBox1.SaveFile('testfile.rtf');
```

Finally, create an event handler for the Exit item and add this:

```
       Close();
```

Although broadly similar to the code we added in the previous VCL project, you'll see some obvious differences. For example, MessageBox is a .NET class and Show() is one of its methods, while System.IO is a namespace (a named group of .NET classes), File is a class inside this namespace, and Exists() is a method of the File class. We're obliged to put a fully qualified 'path' to the File class, since 'File' is a Pascal keyword. We also need to add the System.IO namespace to the 'uses' section of the code unit. You can do that scrolling to the top, adding a comma between the final item in the 'uses' section and the terminating semi-colon, and entering System.IO. You can now press F9 to compile and run this program. If you have problems, load up our project, WinFormEdit.bdsproj.

By default, the richtextbox control saves rich text format (RTF) files. You can verify this by formatting some text in a word processor and copying this into the editing window of the running WinFormEditor program. The formatting of the copied text will be retained. Select File | Save. Now delete all the text and select File | Open. You should see that the reloaded file still contains the formatting.

BEYOND THE BASICS

Obviously, the two text editors that we've programmed this month are very basic; they don't even let us choose a name when saving a document, or select a new file to load from disk. Nor do they provide any options to put text in italics or bold, change its colour or use multiple font styles. These features aren't difficult to implement in Delphi. To prove this point, we've included a more complete editor on the cover CD and DVD. Load up the VCLEditor.bdsproj project. This is a VCL for .NET project, which includes the ability to apply a broad range of text formatting. It even displays the format of the currently selected text in the buttons and combo boxes in its toolbar. For good measure, we've also given it cut, copy, paste and undo options plus a find-and-replace dialog.

Experienced programmers can examine the source code to see how all these features have been implemented. If you find the code overwhelming, don't panic: next month, we'll explain in detail how to add formatting options to a Windows Forms text editor using a mix of Delphi's Object Pascal and Microsoft's C# languages.

Next month: We'll get to grips with the C# language and find out how we can use it to add features to our text editor.