The Delphi CLINIC

Edited by Bob Swart

Bring your problems to our panel of experts!

If there's something puzzling you about an aspect of Delphi, just email the Delphi Clinic Editor, Bob Swart, at Compuserve 100434,2072 or write or fax us at The Delphi Magazine

Q In dBase and Visual Basic I have functions to trim spaces from a string (`TRIM$` for example). Where I can find these functions in Delphi?

A I'm afraid you won't find them, because they aren't there! However, you can easily write them yourself, or just use the `TrimStr` unit shown in Listing 1 instead (put it in the `uses` clause of your units).

Q How do I bring up a secondary form in response to an action in my application's primary form? I want to design several forms as secondary forms and then execute them in response to different actions in the primary form. I also want to move back and forth between the forms when they are on the screen.

A All you need to do is hide the primary form (`Form1.Hide`) and show the second one (`Form2.Show`). Note that you must include `Unit2` in the `uses` clause of `Unit1`. If you want to get rid of a form, just call `Close` (for example `Form1.Close`).

Q I'm using the Borland Delphi Menu Designer to create the menu for my application. Each new menu item's name can be typed in the caption of the `TMenuItem`, so this is no problem. I don't seem to be able to add an old fashioned menu item separator line (like in Resource Workshop).

A If you type a single '-' (dash) as the caption of a menu item, then the menu item will become a separator. It looks a bit awkward in the Menu Designer (since the separator will take the same space as a normal menu item), but at run-time it will be fine!

Q I'm using a `StringGrid` component and I want to resize the columns. No problem occurs when I'm resizing the columns 2, 3, 4 and so on, until I want to resize the first column – it doesn't seem to be possible at all!

A Set the `FixedCols` attribute of the `StringGrid` to 0. Now you can resize the first column. When this is done, reset `FixedCols` to 1.

Q How do I close a form when opening? The main form of my application checks a user login first of all. It creates and shows a modal form with Password and User ID `TEdit` controls. If the `Cancel`

➤ *Listing 1*

```
unit TrimStr;
{$B-}
{ Purpose: routines for removing leading/trailing spaces
  from strings and to take parts of left/right of string.
  LTrim()     - Remove leading spaces from string
  RTrim()     - Remove trailing spaces from string
  Trim()      - Remove leading & trailing spaces
  RightStr()  - Extract substring from right of string
  LeftStr()   - Extract substring from left of string
  MidStr()    - Extract substring from within string }
interface
Const Space = #$20;
function LTrim(Const Str: String): String;
function RTrim(Str: String): String;
function Trim(Str: String):  String;
function RightStr(Const Str: String; Size: Word):
  String;
function LeftStr(Const Str: String; Size: Word): String;
function MidStr(Const Str: String; Size: Word): String;
implementation
function LTrim(Const Str: String): String;
var len: Byte absolute Str;
    i: Integer;
begin
  i := 1;
  while (i <= len) and (Str[i] = Space) do Inc(i);
  LTrim := Copy(Str,i,len)
end {LTrim};
function RTrim(Str: String): String;
var len: Byte absolute Str;
begin
  while (Str[len] = Space) do Dec(len);
  RTrim := Str
end {RTrim};
function Trim(Str: String): String;
begin
  Trim := LTrim(RTrim(Str))
end {Trim};
function RightStr(Const Str: String; Size: Word):
String;
var len: Byte absolute Str;
begin
  if Size > len then Size := len;
  RightStr := Copy(Str,len-Size+1,Size)
end {RightStr};
function LeftStr(Const Str: String; Size: Word): String;
begin
  LeftStr := Copy(Str,1,Size)
end {LeftStr};
function MidStr(Const Str: String; Size: Word): String;
var len: Byte absolute Str;
begin
  if Size > len then Size := len;
  MidStr := Copy(Str,((len - Size) div 2)+1,Size)
end {MidStr};
end.
```

button is pressed (`ShowModal` returns `mrCancel`) I want to prevent the main form from opening. The activation of the login dialog is in the `FormCreate` method of the main form:

```
procedure TFormMain.FormCreate(
  ...)
var
  LoginForm : TUserLogin;
begin
  Application.CreateForm(
    TUserLogin, LoginForm);
  if LoginForm.ShowModal =
    mrCancel then begin
    LoginForm.Free;
    Close;
  end;
end {FormCreate};
```

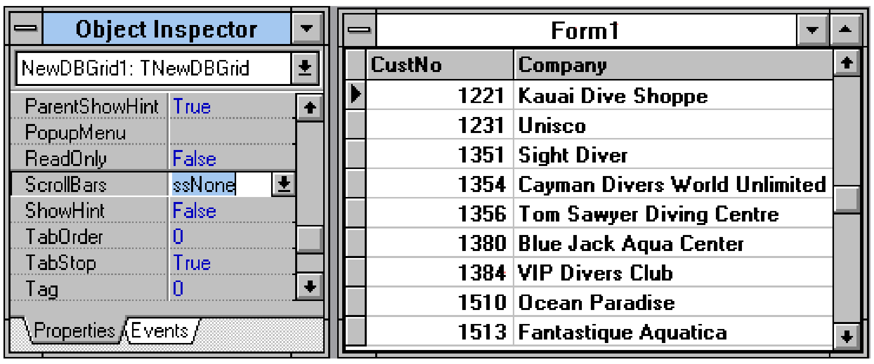
The `Close` call has no effect!

A Rather than putting this logic into `FormCreate`, instead put it into the main form's `OnCreate` handler. Also, instead of calling `Close` use `Application.Terminate;`

Q How can I remove the scroll bars from a `TDBGrid`? There's no scroll bars property to set in the Object Inspector.

A If you try to remove the scoll bars from a standard `TDBGrid` object, you are guaranteed only partial success. There's a couple of reasons for this. Firstly, the `ScrollBars` property, which allows you to conveniently turn on or off either or both of the horizontal and vertical scroll bars in `TMemo`, `TDBMemo`, `TOutline` and `TDrawGrid` components is a protected data member in a `TDBGrid`. This of course restricts anyone from using it unless they are deriving a new component from `TDBGrid`.

The second problem would show itself if you tried to achieve the goal by publishing the `ScrollBars` property in a descendant `TDBGrid` (as shown in Listing 2, in the `TNewDBGrid` component). Figure 1 shows the result of installing such a descendant component in the component palette, using it in an application and setting the `ScrollBars` property to `ssNone`. The



Object Inspector — NewDBGrid1: TNewDBGrid

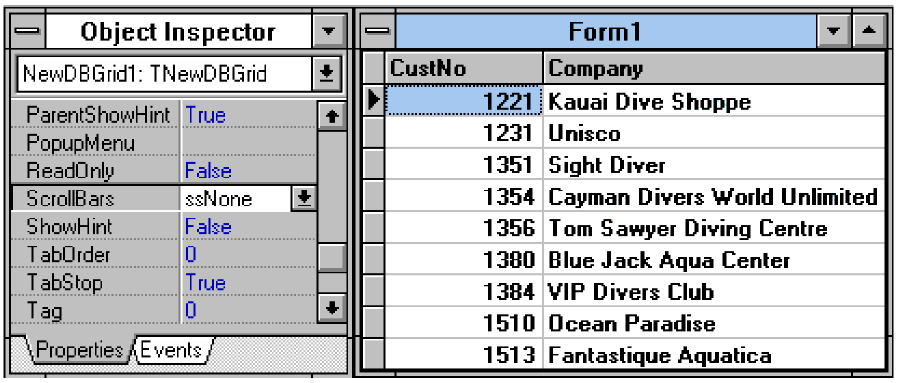
ParentShowHint	True
PopupMenu	
ReadOnly	False
ScrollBars	ssNone
ShowHint	False
TabOrder	0
TabStop	True
Tag	0

Properties / Events

Form1

CustNo	Company
1221	Kauai Dive Shoppe
1231	Unisco
1351	Sight Diver
1354	Cayman Divers World Unlimited
1356	Tom Sawyer Diving Centre
1380	Blue Jack Aqua Center
1384	VIP Divers Club
1510	Ocean Paradise
1513	Fantastique Aquatica

➤ *Figure 1 Removing scrollbars from a TDBGrid: partial success*



Object Inspector — NewDBGrid1: TNewDBGrid

ParentShowHint	True
PopupMenu	
ReadOnly	False
ScrollBars	ssNone
ShowHint	False
TabOrder	0
TabStop	True
Tag	0

Properties / Events

Form1

CustNo	Company
1221	Kauai Dive Shoppe
1231	Unisco
1351	Sight Diver
1354	Cayman Divers World Unlimited
1356	Tom Sawyer Diving Centre
1380	Blue Jack Aqua Center
1384	VIP Divers Club
1510	Ocean Paradise
1513	Fantastique Aquatica

➤ *Figure 2 This time complete success!*

vertical scrollbar doesn't disappear when requested, although the horizontal one does as we ask.

The problem is a bug in the `DBGRIDS` unit in the VCL. If you watch the vertical scrollbar in a grid carefully during use, you'll see that it doesn't work as scrollbars normally do. The thumb bar is either at the top, bang in the centre, or right down at the bottom. The grid doesn't find out how many records are in the table with each movement (for efficiency) and so can't do proper gradations. This abnormal (for want of a better word) scrollbar behaviour is implemented in a non-virtual method of `TCustomDBGrid`, the ancestor of `TDBGrid`. It is called from a number of other `TCustomDBGrid` methods, and forces a particular thumb bar position of the vertical scroll bar.

Unfortunately, forcing a thumb bar position causes the scroll bar to display, whether or not it is wanted, and the code does not check the `ScrollBars` property. My suggested fix is to add a check for the `ScrollBars` property into the source code and recompile it. Of course this requires the VCL

```
unit NewGrid;
interface
uses
  Classes, Grids, StdCtrls,
  DBGrids;
type
  TNewDBGrid = class(TDBGrid)
  published
    property ScrollBars;
  end;
procedure Register;
implementation
procedure Register;
begin
  RegisterComponents(
    'Data Controls',
    [TNewDBGrid]);
end;
end.
```

➤ *Listing 2*

source, so you'll need the Client/Server version of Delphi, or you'll need to get the VCL Source Code Disk add-on. If you don't have the source you will be stuck with this approach. *[Note: make sure you install the Delphi patch files from the disk with this issue BEFORE you make this patch to DBDRIDS, or the Borland patch files won't work! Editor].*

A Borland R&D guy confirmed this would be the right approach, although he implied a complete solution probably requires more than just this one test. With that warning in mind, here is my solution (assuming Delphi is installed into C:\DELPHI):

1. Make a new project in Delphi and drop a DBGrid onto the form.
2. Choose File | Save Project, and cancel the Save Unit dialog that comes up (this adds the DBGrids unit to the uses list of the unit that sits behind your form).
3. Select Options | Project, go to the Directories/Conditionals tab.
4. Add C:\DELPHI\SOURCE\VCL to the Search Path.
5. Press the OK button.
6. Scroll to the top of the unit file in the project and click on DBGrids in the uses statement.
7. Right-click on the code window and choose Open File at Cursor (this uses that path we added to find the DBGRIDS.PAS VCL source file and load it into the editor).
8. Use Search | Find to locate the TCustomDBGrid.UpdateScrollBar implementation around line 811.
9. Immediately after the begin, insert the line:

```
 if ScrollBars in
   [ssVertical, ssBoth] then
```

10. Close and save DBGRIDS.PAS.
11. Ensure that Options | Environment | Preferences | Autosave Editor files is unchecked (allowing you to compile this temporary project without saving it first) and compile using Compile | Compile, causing this VCL module, amongst others, to get recompiled.
12. Using File Manager, rename file C:\DELPHI\LIB\DBGRIDS.DCU to C:\DELPHI\LIB\DBGRIDS.SAV, copy C:\DELPHI\SOURCE\VCL\DBGRIDS.DCU to C:\DELPHI\LIB\DBGRIDS.DCU (this replaces the original compiled version of DBGrids that Delphi uses when making the component library with our new version).
13. Back in Delphi choose Options | Rebuild Library to generate a patched component library.

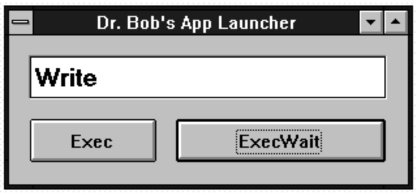
Having gone through the rigmarole of modifying the component library source code and regenerating the library, the TNewDBGrid mentioned earlier now works as expected, and the ScrollBars property works. Figure2 shows a grid with no scrollbars at all.

Q How do I "shell" out with a call to another executable that I want to call, which returns the user to my Delphi application when exited?

A We can modify the old WinExecAndWait function, and replace the while PeekMessage loop with Application.HandleMessage, as shown in Listing 3. Using this function, we can even write a little application launcher that will hide itself when executing (and waiting for) another application. An example is shown in Figure 3, the code is in Listing 4 (the full application source is on the free disk with this issue of course).

Thanks to Arjan Jensen for the menu separator advice, 'Buggy' Bazz Zuidwijk for the TStringGrid hint and Brian Long for his TDBGrid fix. Plus, of course, thanks to all those who submitted queries: keep them coming!

➤ *Figure 3*
Example application launcher



➤ *Listing 3*

```
function WinExecAndWait(CmdLine: PChar; CmdShow: Word): Word;
var InstID: THandle;
    Terminate: Boolean;
begin
  InstID := WinExec(CmdLine, CmdShow);
  if InstID < 32 then
    WinExecAndWait := InstID
  else
    repeat
      Application.HandleMessage(Terminate);
    until (GetModuleUsage(InstID) = 0) or Terminate;
end;
```

➤ *Listing 4*

```
unit Unit1;
interface
uses
  SysUtils, WinTypes, WinProcs,
  Forms, Classes, Controls, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Exec: TButton;
    ExecWait: TButton;
    procedure ExecClick(Sender: TObject);
    procedure ExecWaitClick(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
uses ExecWait;
{$R *.DFM}
procedure TForm1.ExecClick(Sender: TObject);
var Str: String;
    Len : Byte absolute Str;
begin
  Str := Edit1.Text;
  Str[len+1] := #0;
  WinExec(@Str[1], SW_SHOW);
end;
procedure TForm1.ExecWaitClick(Sender: TObject);
var Str: String;
    Len : Byte absolute Str;
begin
  Form1.Hide;
  Str := Edit1.Text;
  Str[len+1] := #0;
  WinExecAndWait(@Str[1], SW_SHOW);
  Form1.Show;
end;
end.
```