Tips & Tricks

Packing dBase Tables

Having just prepared a routine to enable me to pack a dBase table I picked up The Delphi Magazine (Issue 9) to discover that Mike Orriss had already done it! However, his routine is for Paradox tables. When using dBase tables it is necessary to use the `dbiPackTable` routine instead of the `dbiDoRestructure`.

Listing 1 shows a unit which can be called with just the single parameter of the table object which you require to be packed.

Contributed by Richard Smith, CompuServe 100446,327

```
unit Packtbl;
interface
uses
  DBIProcs, DBIErrs, DBITypes, DB, DBTables, sysutils;
procedure PackTable(const tbl : TTable);
implementation
procedure PackTable(const tbl : TTable);
var
  dbResult : DBIResult;
  hDB : hDBIDb;
  hCursor : hDBICur;
  pszTablename : PChar;
  pszDriverType : PChar;
  bRegenIdxs : Boolean;
  StoreExcl : boolean;
  StoreActive : boolean;
begin
  StoreExcl := tbl.exclusive;
  StoreActive := tbl.Active;
  try
    try
      tbl.open;
      hDB := tbl.DBHandle;
      tbl.close;
      tbl.exclusive := true;
    except
      on E : Exception do
        raise Exception.create(
         'Error locking table for exclusive access:'+
          E.message);
    end;
    pszTableName := StrAlloc(25);
    StrPCopy(pszTableName, tbl.tablename);
    pszDriverType := StrAlloc(25);
    StrPCopy(pszDriverType, 'DBase');
    bRegenIdxs := true;
    dbResult := DBiPackTable(hDB, tbl.handle,
      pszTableName, pszDriverType, bRegenIdxs);
    if dbResult <> DBIERR_NONE then
      raise EDBEngineError.create(dbResult);
  finally
    tbl.exclusive := StoreExcl;
    tbl.active := StoreActive;
  end;
end;
end.
```

➤ *Listing 1*

Changing TEdit Text

Usually, in the `OnChange` event of a `TEdit` control, you cannot change the `Text` value: it chains another `OnChange` event recursively until the stack is exhausted. To change the `Text` value, you must assign `OnChange` to `nil`, make the change, and then return the event to its original value. This code shows how:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
  Edit1.OnChange := NIL;
  if Edit1.Text = 'Some text' then
    Edit1.Text := 'New Value';
  Edit1.OnChange := Edit1Change;
end;
```

This trick also works in `OnValidate` events.

Contributed by Bruno Sonnino from Sao Paulo, Brazil (Email: bruno.sonnino@mandic.com.br)

Reading Characters: Text Or Binary?

I have recently ported a map drawing application from Delphi 1 to Delphi 2; it was originally a Turbo Pascal 5.5 application so it is well travelled. One problem I found concerned the use of text files.

The originator of the code (not myself) had written routines for reading binary files that used the `text` file type and read bytes as `char`. Since he is now my boss, I am not passing comment on this practice. This code worked without a problem under Delphi 1. However, when moved to Delphi 2 the application failed to work correctly. We traced the problem to the hex value `$1A` (ASCII 26) occurring in the data, and of course this wonderful number can also be expressed as Control Z, the end of file marker. Once this value has been read, `ReadChar` always returns it. The function `Eof` returns `True` when called and so input ends without error.

Inspection of the library code in READCHAR.ASM confirms the prognosis; a look at TCHR.ASM in the Delphi 1 source shows that Borland has re-written these routines. Once discovered we easily dealt with this problem. Changing the file type to `file` (ie untyped) and using `BlockRead` solved the problem and improved the efficiency.

Contributed by Mike Rogers, Vaisala TMI Ltd (Email: MCR@vatmi.vaisala.com)

Naming Forms And Units

I used to be annoyed by this problem. I create a form and give it a name, say `PartFrm` for *Part Form*. Then I work on it and finally save it. What should its file name be? PARTFRM.PAS for `PartFrm` of course. But no, if you use this, Delphi will tell you there is already something else called `PartFrm`!

Actually there are two potentially conflicting names. The form name is used as the variable name for the form. You need another name for the unit that contains the form. The unit name is also the 8-character file name. Once I was aware of this, I devised a naming

notation. For the variable name, which is referenced frequently, I apply my standard naming notation to *prefix* it with Frm. For the unit name (and file name), I use an alternate notation by *suffixing* it with Frm. So the code looks like this:

```
{ file PARTFRM.PAS }
unit PartFrm;
...
var
  FrmPart: TFrmPart;
```

Contributed by Tung Wai Yip from Singapore (Email: stung@pacific.net.sg)

Unique Values Only In TDBLookupCombo

When using a TDBLookupCombo to allow the user to look up values from another table, you will only want to see one example of each unique value of the lookup field.

Say you have a Paradox table, called STOCK, with a field called Type. In Delphi 1 you can use a Query By Example (QBE) file as the TTable source. First, with the Database Desktop create a query against the TYPE table:

```
Query
STOCK.DB | Type  |
         | Check |
EndQuery
```

This creates a result set with a single field. Save this as TYPE.QBE. Next store TYPE.QBE in the TableName property of the lookup TTable (use the name in full, including QBE) and you will then be able to use Type as the TDBLookupCombo's LookupField property. Delphi 2 supports Paradox unique secondary indices, so this workaround is not needed.

Contributed by Mike Orriss, CompuServe 100570,121

Calling Forms

To call a form using a string variable whose value is the type of the form try the following (assuming that str contains TForm2 etc):

```
Uses Form2, Form3, Form4;
procedure TForm1.Button1Click(Sender: TObject);
begin
  with TFormClass(
    FindClass(str)).Create(Application) do try
    ShowModal;
  finally
    Free;
  end;
end;
initialization
  RegisterClasses([TForm2, TForm3, TForm4]);
end.
```

Contributed by Mike Orriss, CompuServe 100570,121

Form Mode Change

When presenting database data for the user to view, you may well use a modeless form. However, if the user then wishes to *edit* the data (say, having clicked an Edit button), how can you change the form from modeless to modal without closing and re-showing it? This can be achieved by adding code to the TDataSource's OnStateChange event handler to disable or enable all but the current form, like this:

```
procedure TForm2.DataSource1StateChange(
  Sender: TObject);
var ix: integer;
    b: boolean;
begin
  with (Sender as TDataSource).DataSet do
    b := (State = dsBrowse);
  with Screen do
    for ix := 0 to FormCount-1 do
      if Forms[ix] <> ActiveForm then
        Forms[ix].Enabled := b;
end;
```

Note that preventing the form from closing while in Edit mode (via OnCloseQuery code) is also required.

Contributed by Mike Orriss, CompuServe 100570,121

Empty Records?

If you need to find out if any of the fields in a particular table record contain data, try this function:

```
function HasAnyValues(tbl: TTable): boolean;
var ix: integer;
begin
  Result := True;
  for ix:= 0 to tbl.FieldCount-1 do
    if not tbl.Fields[ix].IsNull then exit;
  Result := False;
end;
```

Note that the TTable.Modified property is set if any field value is changed.

Contributed by Mike Orriss, CompuServe 100570,121

Same As Last Time...

In data entry screens, much of the data in a new record is often exactly the same as the previous record, just entered. One way of avoiding the user re-typing it all again is to use a shadow table. You can synchronise via the DataSource.OnDataChange event with code like:

```
if Table1.State = dsBrowse then
  Table2.GotoCurrent(Table1);
```

Then, after a record insert, on all affected controls you can use their OnEnter event to copy the corresponding value from Table2.

Contributed by Mike Orriss, CompuServe 100570,121