

Einleitung

Folgende Aufgabe soll gelöst werden:

Eine Delphi-Anwendung will mittels ADO auf eine Access-Datenbank zugreifen. Diese kann sich an einem beliebigen Ort, z.B. auch innerhalb eines lokalen Netzwerks befinden. Der Benutzer soll die Datenbank frei auswählen und bei Bedarf auch wechseln können.

Wir legen erst mal ein neues Delphi-Projekt an und speichern es in einem neuen Ordner. Bei mir ist das "Projekt1.dpr" mit dem Hauptformular "TForm1" und der Unit "MainUnit".

Es wäre sinnvoll ein Datenmodul anzulegen. Das muss zwar für kleine Anwendungen nicht sein, aber sauberer ist es allemal. Klicken Sie dazu auf den Menüpunkt Datei/Neu/Andere und wählen Sie das Datenmodul aus. Speichern unter Unit: "DatenmodulUnit", Name des Datenmoduls: "DataModule2".

TADOConnection

Zuerst fügen wir eine TADOConnection ins Datenmodul ein. Mit Doppelklick können wir eine Verbindung zur Datenbank aufbauen.



Für Access nehmen wir den *Microsoft Jet 4.0 OLE DB Provider*. Auf dem Reiter *Verbindungen: Datenbank auswählen...* und mit "OK" bestätigen. Jetzt sollte auch das gelbe Fragezeichen vor der ADOConnection weg sein.

Verbindung aufbauen

Es geht darum, dem Benutzer zu ermöglichen, diese Verbindung zu ändern, z.B. weil er die .mdb verschoben hat oder zwischen mehreren Datenbanken wechseln will und, evtl. auch beim ersten Start, wenn wir den DSN nicht über die Installationsroutine erzeugen wollen.

Als erstes fügen wir in der *MainUnit* ganz oben unter uses unser Datenmodul ein:

```
uses ..., DatenmodulUnit;
```

Dann sollten wir mal das Öffnen und Schließen der Datenbank testen: Dazu werden wir zwei Funktionen im Datenmodul unter public einfügen:

```
public
{ Public declarations }
function OpenDatabase: Boolean;
function CloseDatabase: Boolean;
...

function TDataModule2.OpenDatabase: Boolean;
begin
  OpenDatabase:=true;
  try
    ADOConnection1.Open;
  except
    OpenDatabase:=false;
  end;
end;

function TDataModule2.CloseDatabase: Boolean;
begin
  CloseDatabase:=true;
  try
    ADOConnection1.Close;
  except
    CloseDatabase:=false;
  end;
end;
```

Die beiden Prozeduren können wir jetzt über das Menü im Hauptformular aufrufen. Dazu sollten wir erst einmal eine TMainMenu-Komponente in das Formular einfügen und die Items OpenDatabase und CloseDatabase anlegen. Das dürfte aber niemanden vor ein Problem stellen.

```
procedure TForm1.OpenDatabaseClick(Sender: TObject);
begin
  if DataModule2.OpenDatabase=true then
    MessageDlg('Die Datenbank konnte erfolgreich geöffnet werden',
      mtInformation, [mbOK], 0)
  else
    MessageDlg('Die Datenbank konnte nicht geöffnet werden',
      mtError, [mbOK], 0);
end;

procedure TForm1.CloseDatabaseClick(Sender: TObject);
begin
  if DataModule2.CloseDatabase=true then
    MessageDlg('Die Datenbank konnte erfolgreich geschlossen
werden',
      mtInformation, [mbOK], 0)
  else
    MessageDlg('Die Datenbank konnte nicht geschlossen werden',
      mtError, [mbOK], 0);
end;
```

Das sollte funktionieren. Einfach mal testen.

Einstellungen speichern

Nun wäre es nett, den *ConnectionString* beim Beenden der Anwendung in die Registry zu schreiben, damit der User ihn nicht immer neu eingeben muss. Beim Programmstart soll der DSN dann natürlich ausgelesen werden - wenn möglich und vorhanden.

Dazu brauchen wir im Datenmodul drei Funktionen, hinter dem Schlüsselwort *public*. Vorher müssen wir aber noch unter *uses* im Datenmodul *ADOConED* einfügen. Dann geht's an die Funktionen:

```
unit DatenmodulUnit;

public
function EditMeineAnwendungDSN(var DSN: string): Boolean;
function GetMeineAnwendungDSN: string;
function SetMeineAnwendungDSN(DSN: string): Boolean;
...

function TDataModule2.EditMeineAnwendungDSN(var DSN: string):
Boolean;
begin
    EditMeineAnwendungDSN:=true;
    if ADOConnection1.Connected=true then
        if CloseDatabase=false then begin
            EditMeineAnwendungDSN:=false;
            exit;
        end;
    if EditConnectionString(ADOConnection1)=true then
        DSN:=ADOConnection1.ConnectionString
    else
        EditMeineAnwendungDSN:=false;
end;

function TDataModule2.GetMeineAnwendungDSN: string;
begin
    GetMeineAnwendungDSN:=ADOConnection1.ConnectionString;
end;

function TDataModule2.SetMeineAnwendungDSN(DSN: string): Boolean;
begin
    SetMeineAnwendungDSN:=true;
    if ADOConnection1.Connected=true then
        if CloseDatabase=false then begin
            SetMeineAnwendungDSN:=false;
            Exit;
        end;
    ADOConnection1.ConnectionString:=DSN;
```

end;

Nun werden wir im Hauptformular auf die Funktionen zugreifen. Dazu allerdings zuerst die Unit Registry ganz oben unter uses im Hauptformular einbinden. Die DSN zur Datenbankverbindung wird mit folgendem Code in die Registry geschrieben und wieder ausgelesen:

```

unit MainUnit;
...

private
{ Private declarations }
function WriteDSNToRegistry(DSN: string): Boolean;
function ReadDSNFromRegistry(var DSN: string): Boolean;
...

function TForm1.WriteDSNToRegistry(DSN: String): Boolean;
var Registry: TRegistry;
begin
  WriteDSNToRegistry:=true;
  Registry:=TRegistry.Create;
  Registry.RootKey:=HKEY_LOCAL_MACHINE;
  {Der Registry-Pfad für Ihren Eintrag}
  if Registry.KeyExists('\Software\MeinProg\MeineAnw')=false then
    try
      Registry.CreateKey('\Software\MeinProg\MeineAnw');
    except
      Registry.Free;
      WriteDSNToRegistry:=false;
      Exit;
    end;
    try
      if Registry.OpenKey('\Software\MeinProg\MeineAnw',
true)=true then
        Registry.WriteString('DSN', DSN)
      else begin
        MessageDlg('Es konnte nicht in die Registry geschrieben '+
'werden', mtError, [mbOK], 0);
        WriteDSNToRegistry:=false;
      end;
    finally
      Registry.CloseKey;
      Registry.Free;
    end;
end;

function TForm1.ReadDSNFromRegistry(var DSN: String): Boolean;
var Registry: TRegistry;
begin
  ReadDSNFromRegistry:=true;

```



```
Registry:=TRegistry.Create;
try
  Registry.RootKey:=HKEY_LOCAL_MACHINE;
  if Registry.KeyExists('\Software\MeinProg\MeineAnw') then begin
    if Registry.OpenKey('\Software\MeinProg\MeineAnw\', true) then
      DSN:=Registry.ReadString('DSN')
    else
      ReadDSNFromRegistry:=false;
  end
else
  ReadDSNFromRegistry:=false;
finally
  Registry.Free;
end;
end;
```

Konfiguration

Nun kann man das schon mal zur Laufzeit ausprobieren. Dazu im Menü des Hauptformulars den Eintrag "Datenbank Konfiguration" hinzufügen mit dem Ereignis:

```

procedure TForm1.DatenbankKonfiguration1Click(Sender: TObject);
var DSN: string;
begin
    if DataModule2.EditMeineAnwendungDSN(DSN)=true then begin
        MessageDlg('Der DSN konnte erfolgreich geändert werden',
            mtInformation, [mbOK], 0);
        WriteDSNToRegistry(DSN);
    end
    else
        MessageDlg('Der DSN konnte nicht geändert werden',
            mtError, [mbOK], 0);
end;

```

Der Benutzer kann also jetzt bereits zur Laufzeit die Datenbank neu auswählen. Nun müssen wir nur noch beim Öffnen der Anwendung dafür sorgen, dass die DSN aus der Registry geholt wird, und beim Schließen wieder zurückgeschrieben wird; bzw. falls die Datenbank nicht geöffnet werden kann, soll sich der Dialog zur Eingabe durch den Benutzer öffnen.

Dazu legen wir die Ereignisse OnShow bzw. OnClose des Hautformulars an:

```

procedure TForm1.FormShow(Sender: TObject);
var DSN: string;
begin
    if ReadDSNFromRegistry(DSN)=true then begin
        if DataModule2.SetMeineAnwendungDSN(DSN)=true then
            if DataModule2.OpenDatabase=false then begin
                MessageDlg('Es konnte keine Verbindung zur Datenbank '+
                    'aufgebaut werden.', mtError, [mbOK], 0);
                DatenbankKonfiguration1Click(Sender);
            end;
        end
    else if DataModule2.OpenDatabase=false then begin
        MessageDlg('Es konnte keine Verbindung zur Datenbank '+
            'aufgebaut werden.', mtError, [mbOK], 0);
        DatenbankKonfiguration1Click(Sender);
    end
    else begin
        MessageDlg('Es konnte keine DSN ausgelesen werden.',
            mtError, [mbOK], 0);
        DatenbankKonfiguration1Click(Sender);
    end;

```



```
end;  
  
procedure TForm1.FormClose(Sender: TObject; var Action:  
TCloseAction);  
begin  
    WriteDSNToRegistry(DataModule2.GetMeineAnwendungDSN);  
    DataModule2.CloseDatabase;  
end;
```

Ausführung

Wenn wir jetzt unser Projekt kompilieren wird sich zuerst der Hinweis "Es konnte keine Verbindung zur Datenbank aufgebaut werden" öffnen und anschließend der Dialog `ADOConnectionString`.

Wir wählen unsere Datenbank aus. Die Anwendung schließen und noch mal starten. Jetzt sollte alles reibungslos funktionieren. Der DSN ist erfolgreich durch unser `OnClose`-Ereignis in die Registry geschrieben und wieder ausgelesen worden.

Wenn Sie Ihr Programm weitergeben wollen, dann kompilieren Sie es am Schluss ohne DSN in der `ADOConnection`. Dann wird auf jeden Fall erst einmal der Dialog zum Auswählen der Datenbank gestartet.

Sie können jetzt mal die verschiedenen Varianten testen. Z.B. die Windows-Registry öffnen mit `Start/Ausführen/regedit` und den Eintrag `HKEY_LOCAL_MACHINE\Software\MeinProg\MeineAnw` löschen und dann das Programm neu starten. Oder mal versuchen Ihre Datenbank-Datei in ein anderes Verzeichnis zu verschieben usw.

Die Anwendung kann zwar noch nicht Vernünftiges, aber da wird Ihnen schon was einfallen.