Wilfried Koch

Das C++-Builder Rezeptbuch

Teil 2

Inhaltsverzeichnis

8. Eingriffe in den Programmablauf....14
8.1. Abbruch einer laufenden Berechnung zu einem beliebigen Zeitpunkt....14
8.1.1. Aufgabenstellung....14
8.1.2. Erster Lösungsversuch....14
8.1.2.1. Bedienoberfläche....14
8.1.2.2. Berechnungsschleife....14
8.1.2.3. Abbruchmechanismus....15
8.1.2.4. Testergebnis....15
8.1.3. Endgültige Lösung....15
8.1.3.1. Schritthaltende Resultatsanzeige....16
8.1.3.2. Zeitnaher Berechnungsabbruch durch Schaltflächenbetätigung....16
8.1.3.3. Berechnungsabbruch mittels Tastatureingabe....16
8.1.3.4. Zusammenfassung....17
8.1.4. Programmcode: Abbruch von Berechnungen (ProAbbruch)....18
8.1.4.1. Hauptformular FormMainAbbruch....18
8.1.5. Reduzierung der Prozessorbelastung....20
8.2. Abbruch einer laufenden Berechnung nach Ablauf einer vorgegebenen Zeit (Realisierung eines Timeouts)....21
8.2.1. Aufgabenstellung....21
8.2.2. Lösung....21
8.2.2.1. Bedienoberfläche....21
8.2.2.2. Berechnungsschleife....22
8.2.2.3. Zeitlicher Abbruchmechanismus....22
8.2.3. Programmcode: Programmabbruch durch Zeitablauf (ProTimeOut)....22
8.2.3.1. Hauptformular FormMainTimeOut....22
8.2.4. Reduzierung der Prozessorbelastung....24
8.3. Laufende Anzeige von Datenänderungen....24
8.3.1. Aufgabenstellung....24
8.3.2. Elementare Lösung....25
8.3.2.1. Anwendung und Bedienoberfläche in der selben Unit....25
8.3.2.2. Verteilung von Anwendung und Bedienoberfläche auf zwei verschiedene Units....26

8.3.3. Systematische Lösung....28
8.3.3.1. Bedienoberfläche....28
8.3.3.2. Anwendungsmodul....29
8.3.3.3. Ereignisbehandlung im Hauptformular....30
8.3.4. Programmcode: Ereignisgesteuerte Anzeige veränderlicher Daten (ProEreignis)....31
8.3.4.1. Hauptformular FrmMainEreignis....31
8.3.4.2. Unit UAnwendung....33
8.4. Gleiche Fachaufgabe – geändertes Gesicht....34
8.4.1. Lösung zur variierten Aufgabenstellung....34
8.4.2. Programmcode zur variierten Aufgabenstellung....35
8.4.2.1. Hauptformular FrmMainEreignisVar....35

9. Anspruchsvolle Geschäftsgrafiken problemlos erstellen (Einsatz von TChart)....37

9.1. Aufgabenstellung....38
9.1.1. Fachaufgabe....38
9.1.2. Aufbau und Vorgehen....38
9.2. Lösung mit dem TeeChart-Framwork....39
9.2.1. Wichtige Methoden und Eigenschaften im Umfeld der Klasse TChart....39
9.2.1.1. Wichtige Eigenschaften und Methoden des Diagramms (TChart)....39
9.2.1.2. Wichtige Eigenschaften und Methoden von Datenreihen (TChartSeries)....42
9.2.1.3. Wichtige Eigenschaften und Methoden von Liniendiagrammen (TLineSeries)....42
9.2.1.4. Wichtige Eigenschaften und Methoden von Kreisdiagrammen (TPieSeries)....43
9.2.1.5. Wichtige Eigenschaften und Methoden von Blasendiagrammen (TBubbleSeries)....44
9.2.2. Formulare....44
9.2.2.1. Hauptformular FrmChartMain....44
9.2.2.2. Diagrammformulare....44
9.2.3. Interaktive Handhabung der Komponente TChart....46
9.2.4. Elemente der Diagrammgestaltung....46
9.2.5. Darstellung der Wahlergebnisse....47
9.2.5.1. Darstellung der Wahlergebnisse im Dialogverfahren....48
9.2.5.2. Darstellung der Wahlergebnisse durch Programmierung....55
9.2.5.3. Berechnung und Darstellung der Differenz zweier Datenreihen....55
9.2.5.4. Variationsmöglichkeiten für die Darstellung....60

9.2.5.5. Diagrammexport in Dateien....61
9.2.5.6. Diagrammdruck....61
9.2.6. Programmcode: Erstellung von Geschäftsgrafiken....62
9.2.6.1. Hauptformular FrmChartMain....62
9.2.6.2. Formular FrmChartStat für die statische Diagrammerstellung....63
9.2.6.3. Formular FrmChartDyn für die dynamische Diagrammerstellung....70

10. Borland Database Engine (BDE) –
Der schnelle Weg zur Erstellung einfacher Datenbankanwendungen....74

10.1. Die Architektur von Datenbankanwendungen....74
10.1.1. Datenbanklösung und Dateilösung....75
10.1.2. Die Beispielanwendung....76
10.2. Erstellung von Datenbanktabellen....77
10.2.1. Interaktive Erstellung der Datenbanktabelle Kunden.db mit dem Programm Datenbankoberfläche (Database Desktop)....77
10.2.1.1. Erstellen der Datenbanktabelle....77
10.2.1.2. Felder der Datenbanktabelle erstellen....78
10.2.1.3. Bearbeiten / Ändern....80
10.2.2. Erstellen der Datenbanktabelle Waren.db zur Laufzeit mittels eines individuell erstellten Programms....80
10.2.2.1. Aufgabenstellung....81
10.2.2.2. Lösungsweg....81
10.2.2.3. Der Alias....84
10.2.2.4. Definitionen für die gesamte Datenbanktabelle....85
10.2.2.5. Programmcode: Erstellung einer Datenbanktabelle....89
10.3. Programme zur Anzeige und Manipulation von Datenbanktabellen....92
10.3.1. Datenbanktabelle Waren.db....92
10.3.1.1. Aufgabenstellung....92
10.3.1.2. Löungsweg....92
10.3.1.3. Programmcode: Manipulation der Warendatentabelle....96
10.3.2. Datenbanktabelle Rechnungen.db....97
10.3.3. Datenbanktabelle Rechnungspositionen.db....97
10.4. Verknüpfung von Datenbanktabellen....99
10.4.1. Verbindung von Kunden- und Rechnungsdaten....99
10.4.1.1. Einbau der Lookup-ComboBox....99
10.4.2. Verbindung von Waren- und Rechnungsdaten....100
10.4.2.1. Die zu einer bestimmten Rechnung gehörige Rechnungspositionen

finden....101
10.4.2.2. Statt einer Warennummer die kompletten Wareninformationen in die Rechnungsposition einfügen....102
10.4.3. Programmcode: Erstellung und Manipulation von Rechnungen....105
10.4.3.1. Hauptformular FrmRechTabDialog....105
10.5. Berichtserstellung mit Rave Reports....115
10.5.1. Gestaltung des Berichts....117
10.5.1.1. Seiteneinrichtung....117
10.5.1.2. Fest positionierte Elemente....118
10.5.1.3. Flexible Berichtselemente....120
10.5.2. Verbindung von Datenbank (Datenmengen) und Bericht....121
10.5.2.1. Bereitstellung von Datenmengen für das Rave-Projekt....121
10.5.2.2. Nutzung der Datenmengen im Rave-Projekt....122
10.5.2.3. Datenanschluss der datensensitiven Komponenten im Rave-Report....122
10.5.2.4. Der Rave Reports Projektbaum....122
10.5.3. Programmierung der Berichtsausgabe....124
10.5.4. Programmcode: Ausdrucken von Datenbankberichten (Rechnungen)....125
10.5.4.1. Programmaufbau....125
10.5.4.2. Hauptformular FrmRechTabDialog....126
10.5.4.3. Berichtsformular FrmBerichtRech....126
10.6. Datenbanktabellen und Textdateien....129
10.6.1. Aufgabenstellung....129
10.6.2. Lösungsweg....130
10.6.2.1. CSV-Export....131
10.6.2.2. CSV-Import....131
10.6.3. Programmcode: Verbindung von Textdateien (CSV) und Datenbanken....133
10.6.3.1. Hauptformular FrmMainKunCSVImpExp....133

11. Nutzen Sie die Möglichkeiten Dynamischer Link-Bibliotheken (DLLs)....137

11.1. Ein bisschen Theorie....137
11.1.1. Möglichkeiten der DLL....137
11.1.1.1. Nutzung einer DLL durch mehrere Programme....138
11.1.1.2. Realisierung hybrider Programmkonzepte....138
11.1.1.3. Skalierung von Programmen....138
11.1.1.4. Verbesserung der Wartbarkeit....138
11.2. Sie erstellen eine DLL....139
11.2.1. Aufgabenstellung....139

11.2.2. Die Lösung....139
11.2.2.1. Die DLL – ein C++Builder-Projekt....139
11.2.2.2. Schnittstellen der DLL....141
11.2.3. Monolithische DLL....141
11.2.3.1. Programmcode für die monolithische DLL....142
11.2.4. Modulare DLL....142
11.2.4.1. Programmcode für die modulare DLL....143
11.2.5. Statische Bindung der DLL....144
11.2.5.1. Projekt ProMainStat_D....144
11.2.5.2. Hauptprogramm ProMainStatDLL_D....144
11.2.5.3. Probleme beim Verbinden von EXE-Datei und DLL....144
11.2.5.4. Hauptformular UFrmMain_D - Direkteinbau der Importschnittstelle...145
11.2.5.5. Hauptformular UFrmMain_Z - Zentrale Anordnung der Importschnittstelle in einer Schnittstellendatei....146
11.2.6. Testen von DLL-Funktionen....147
11.2.7. Trennung der Datenbereiche....148
11.2.8. Dynamische Bindung....149
11.2.8.1. Aufbau des Hauptprogramms (DLL-benutzendes Programm)....149
11.2.8.2. Aufbau des Schnittstellenmoduls UDLLImp....149
11.2.8.3. ProgrammCode für die Dynamische Einbindung von DLLs....151
11.3. Auch das geht: DLLs und VCL....152
11.3.1. Aufgabenstellung....152
11.3.1.1. Realisierung eines Formulars mittels einer DLL....152
11.3.1.2. Hauptprogramm ProDLLVCL....153
11.3.2. Lösungsweg....153
11.3.2.1. Die DLL VCLFormDLL zur Realisierung des Formulars....153
11.3.2.2. Hauptprogramm....154
11.3.3. Programmcode: Verwendung von VCL-Komponenten in DLLs....155
11.3.3.1. DLL VCLFormDLL.dll....155
11.3.3.2. Hauptprogramm....158
11.4. Noch ein Beispiel: DLL mit Objektschnittstelle....160
11.4.1. Aufgabenstellung....160
11.4.2. Lösung....160
11.4.2.1. Basisklasse TMWSt0....161
11.4.2.2. DLL ProDLLObjBeispiel....161
11.4.2.3. Hauptprogramm ProMainObjDLL....162
11.4.3. Programmcode: DLL zur Kapselung von Klassen....164

11.4.3.1. Basisklasse TMWSt0....164
11.4.3.2. DLL ProDLLObjBeispiel....164
11.4.3.3. Hauptprogramm ProMainObjDLL....165
11.4.3.4. Hauptformular FrmMainObjDLL....165
11.5. Zugang zu fremden Sprachwelten: Verwendung von Delphi-DLLs in C++Builder-Programmen....167
11.5.1. Aufgabenstellung....168
11.5.2. Lösungsweg....169
11.5.2.1. Funktion....169
11.5.2.2. Schnittstelleninformation....170
11.5.3. Programmcode: Verbindung eines C++-Hauptprogramms mit einer Delphi-DLL....172
11.5.3.1. Hauptformular....172

12. Internetanwendungen....174
12.1. Das (Open-Source-) Projekt Indy....174
12.2. Spezielle Indy-Datentypen....176
12.3. E-Mail aus einer Anwendung senden (Entwicklung eines SMTP-Clients)....179
12.3.1. Aufgabenstellung....179
12.3.2. Lösungsweg....180
12.3.2.1. Sendemethode BtnSMTPSendenClick....183
12.3.3. Programmcode: Senden von E-Mail-Nachrichten (SMTP-Client)....186
12.3.3.1. Hauptprogramm ProSMTPClient....186
12.3.3.2. Formularunit UFrmMainSMTPClient....186
12.4. E-Mail in einer Anwendung empfangen (Entwicklung eines POP3-Clients)....190
12.4.1. Aufgabenstellung....190
12.4.2. Lösungsweg....192
12.4.2.1. Empfangsmethode BtnPOP3EmpfangenClick....193
12.4.2.2. Auflistung der eingegangenen Nachrichten....194
12.4.2.3. Anzeige der Textnachrichten und Auflistung der Anhänge....198
12.4.2.4. Abspeichern der Nachrichten und Anhänge....200
12.4.2.5. Löschen von Nachrichten....201
12.4.2.6. Trennen der Verbindung....202
12.4.3. Programmcode: Empfangen von E-Mail-Nachrichten (POP3-Client)....202
12.4.3.1. Hauptprogramm ProPOP3Client....202
12.4.3.2. Formularunit UFrmMainPOP3Client....202
12.5. Dateiverkehr mit dem Internet (Erstellung eines FTP-Clients)....208

12.5.1. Was ist FTP?....208
12.5.2. Aufgabenstellung....208
12.5.3. Installation eines Testservers....208
12.5.3.1. Installation von Filezilla....209
12.5.3.2. Installation von FreeFTPd....211
12.5.3.3. Installation von ftpsrv110....213
12.5.4. Vorgehen bei der Lösung....213
12.5.5. Entwicklungsstufe 1....214
12.5.5.1. Bedienoberfläche....214
12.5.5.2. Verbinden....214
12.5.5.3. Anmelden (LogIn)....216
12.5.5.4. Trennen....216
12.5.5.5. Ereignismethoden zur Verfolgung des Betriebszustandes des FTP-Clients 217
12.5.6. Entwicklungsstufe 2....217
12.5.6.1. Bedienoberfläche....218
12.5.6.2. Darstellung der Verzeichnisliste....218
12.5.7. Entwicklungsstufe 3....219
12.5.7.1. Navigation im Server-Verzeichnis....220
12.5.7.2. Wechsel auf das übergeordnete Verzeichnis....220
12.5.8. Entwicklungsstufe 4....220
12.5.8.1. Bedienoberfläche....220
12.5.8.2. Download....222
12.5.8.3. Upload....223
12.5.9. Programmcode: Realisierung eines FTP-Clients....223
12.5.9.1. Hauptformular FrmMainFTPClient4....223

Literaturverzeichnis....230

Stichwortverzeichnis....232

8. Eingriffe in den Programmablauf

8.1. Abbruch einer laufenden Berechnung zu einem beliebigen Zeitpunkt

8.1.1. Aufgabenstellung

In der professionellen Programmierung ist es häufig erforderlich, den Abbruch einer Berechnung durch den Bediener vorzusehen. Dabei ist wichtig, dass zwar die Berechnung aber nicht auch das Programm abgebrochen wird.

Beispielhaft soll hier ein Programm erstellt werden das nacheinander die Quadrate der Zahlen von 0 bis 30000 berechnet und deren Wert sofort nach der Berechnung anzeigt. Ein Abbruch diese Programms muss zu jedem beliebigen Zeitpunkt vorgenommen werden können.

8.1.2. Erster Lösungsversuch

Die Lösung scheint ganz einfach. – Sie ist es aber nicht.

Auf den ersten Blick erscheint Ihnen die Erstellung dieses Programms wahrscheinlich ganz einfach. Wobei Sie vielleicht in folgenden Schritten vorgehen:

- Bedienoberfläche
- Berechnungsschleife
- Abbruchmechanismus

8.1.2.1. Bedienoberfläche

Die Bedienoberfläche (s.a. Abbildung 8.1) besteht aus einem Formular, das zwei Schaltflächen (`BtnStart` und `BtnStop`) und ein Beschriftungsfeld (`LblResultat`) enthält. Die Schaltflächen dienen dem Starten und Stoppen der Berechnung. Im Beschriftungsfeld soll während der Programmlaufs immer das aktuelle Resultat angezeigt werden.

Abbildung 8.1: Im Lösungsversuch verwendete Bedienoberfläche

8.1.2.2. Berechnungsschleife

Die Berechnungsschleife wird in der Ereignismethode des `OnClick`-Ereignisses der Schaltfläche Start untergebracht. Diese Schleife wird entweder durchlaufen bis der Schleifenzähler den vorgesehenen Maximalwert (`MaxZaehl`) von 30000 erreicht oder bis die boolesche Variable `bEnde` den Wert `true` annimmt.

8.1.2.3. Abbruchmechanismus

Der Abbruchmechanismus basiert auf der Ereignismethode für das `OnClick`-Ereignis der Schaltfläche `BtnStop`. Das Betätigen der Stop-Schaltfläche startet die Ereignismethode. Innnerhalb der Ereignismethode wird die Variable `bEnde` auf `true` gesetzt.

Der Code für Berechnungsschleife und (vorgesehenen) Abbruchmechanismus ist nachstehend dargestellt:

```
//---------------------------------------------------------------------------

void __fastcall TFrmMainAbbruch::BtnStartClick(TObject *Sender)
{
  Lbl___1->Caption = "";
  for (int i = 0; i <= MaxZaehl; i++)
  {
    Lbl___1->Caption = IntToStr(i*i);
    if (bEnde)   ///Abbruch falls BtnStop gedrückt wurde.
      {
        break;
      }
  }
}
//---------------------------------------------------------------------------

void __fastcall TFrmMainAbbruch::BtnStopClick(TObject *Sender)
{
  bEnde = true;//
}
```

8.1.2.4. Testergebnis

Weshalb wirkt die Stop-Schaltfläche nicht??!!

Wenn Sie das Programm starteen, erkennen Sie, dass der Text im Beschriftungsfenster zunächst unverändert bleibt und dann nach kurzer Zeit den Wert 900.000.000 annimmt. Dieses Programmverhalten ist unabhängig davon, ob die Stop-Schaltfläche betätigt wird oder nicht!

Was ist passiert hier?

Wenn Sie die Berechnung wie oben dargestellt programmieren, d. h. ohne besondere Maßnahmen, dann wird sie nach Start der Berechnung ohne Unterbrechung abgearbeitet. Der Prozessor ist während dieser Zeit voll ausgelastet. Die Veränderung der Anzeige oder die Ausführung von `BtnStopClick` können erst erfolgen, wenn die Methode `BtnStartClick` beendet wurde.

8.1.3. Endgültige Lösung

Mit ein paar kleinen Veränderungen können Sie das in 8.1.2.4. geschilderte Problem rasch beheben.

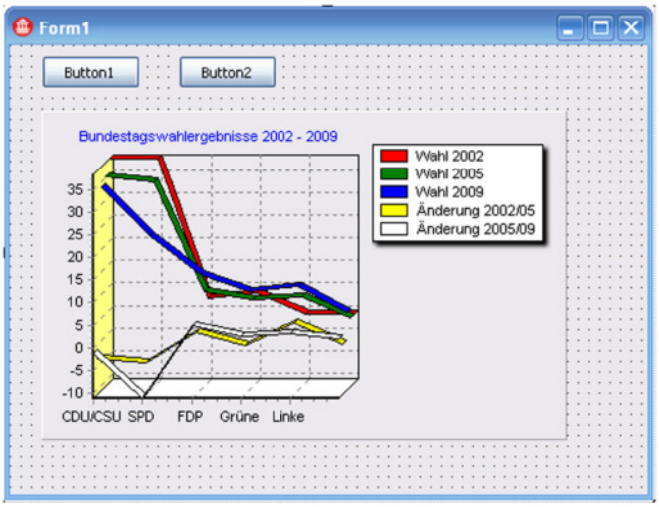

Abbildung 9.16: Diagrammdarstellung mit 3D-Liniendiagrammen (Ergebnisse und Änderungen)

Abbildung 9.16 zeigt die gesamte Grafik. Die erste der ausgewählten Reihen bildet den Minuend, jede weitere repräsentiert einen Subtrahenden. In unserem Fall ist das Ergebnis von 2005 der Minuend und das von 2002 der Subtrahend (Abbildung 9.15).

Programmgestützte Berechnung und Darstellung der Differenzen

Elementare Programmierung

Die Differenz zwischen zwei Wertereihen z. B. zwischen dem Wahlergebnis von 2003 und 1998 können Sie natürlich durch elementare Programmierung, d. h. Programmierung einer Subtraktionsoperation ermitteln. Hierzu legen Sie eine Datenreihe – in unserem Falle *WahlenDiffSeries* vom Typ `TLineSeries` - zur Aufnahme des Resultats neu an und verbinden Sie mit dem Diagramm (*WahlenDiffSeries->ParentChart = Chart1*).

Dann bestimmen Sie aus den bereits existierenden Datenreihen die beiden, die Sie voneinander subtrahieren möchten. Das seien beispielsweise *Wahlen1998Series* und *Wahlen2003Series* (die Wahlergebnisse von 1998 und 2003). *Wahlen1998Series* sei die erste und *Wahlen2003Series* die zweite Datenreihe, die an das Grundelement des Diagramms (Variable *Chart1*, Typ `TChart`) angehängt wurde. Diese Datenreihen können auch mit `Chart1->Series[0]` (für 1998) und `Chart1->Series[1]` (für 2003) angesprochen werden.

Über wieviele Werte Sie innerhalb der Datenreihen iterieren müssen geht aus `Chart1->Series[i]->Count()` hervor. Der Einfachheit halber nehmen wir an,

```
Table1->DatabaseName = "BUCH2DB";
Table1->TableType = ttParadox;
Table1->TableName = "Waren.db";
```

Danach legen Sie nacheinander die einzelnen Felder an und ergänzen im darauf folgenden Schritt die Indizes. Das Indexfeld würde man am einfachsten als selbstinkrementierenden Zähler anlegen (`ftAutoInc`). Man kann jedoch einige Probleme beim Anlegen der Testdatenwerte vermeiden, wenn man in diesem Zusammenhang auf den Typ Integer(lang) (`ftInteger`) ausgeweicht.

Zum Schluss werden Testdatenwerte erzeugt und angezeigt. Damit die erzeugten Werte persistent in der Datenbanktabelle auf dem Datenspeicher abgelegt werden, müssen Sie die Datenbanktabelle schließen. Die Methode `Table1->Close` schreibt die im Programm vorhandenen Werte physikalisch in die Datenbank.

10.2.2.3. Der Alias

Alias = Abkürzung für Pfad zu den Danbanktabellen

Der Alias ist ein Symbol (= eine Abkürzung), das für den zu den Datenbanktabellen führenden Dateipfad steht. Die Einführung eines Alias erleichtert die Migration von Datenbankanwendungen zwischen verschiedenen Rechnern.

Alias Manager = Funktion der Datenbankoberfläche

Dabei wird in der Datenbankanwendung immer mit demselben Bezeichner (Alias) gearbeitet, dem auf dem jeweiligen Rechner ein bestimmter Pfad entspricht. Die Zurodnung zwischen Pfad und Alias erfolgt mit dem Alias-Manager. Der Alias-Manager ist eine Funktion des Programms Datenbankoberfläche (Tools|AliasManager...). Seine Bedienoberfläche ist in Abbildung 10.8 dargestellt. Die wichtigsten Eingaben im Alias-Manager sind:

- Der Aliasname (Datenbankalias) unter dem die Datenbank projektweit (lokaler Alias) oder systemweit (globaler Alias) angesprochen werden kann.
- Der Treibertyp, der angibt um welche Datenbankart (Datenbankfabrikat) es sich handelt. Für Paradox und Access-Datenbanken gilt der Treibertyp STANDARD.
- Der Pfad, der angibt, wo die Datenbanktabellen die dem Alias zugeordnet sind abgespeichert werden.

Wenn Sie eine Datenbank mit Alias X auf einen anderen Rechner (Zielrechner) übertragen, dann müssen Sie lediglich am Zielrechner mittels des Alias-Managers festlegen welcher Dateipfad dort diesem Alias entspricht.

Beispiel:

Der Alias lautet auf Quell- und Zielrechner `MeinAlias`.

Auf dem Quellrechner befinden sich die Datenbanktabellen im Pfad **E:\Quelldatenbank**, auf dem Zielrechner in **F:\Zieldatenbank**. Im Programmcode tauchen diese Pfade nicht auf, sondern nur der Aliasname `MeinAlias`. Damit wird

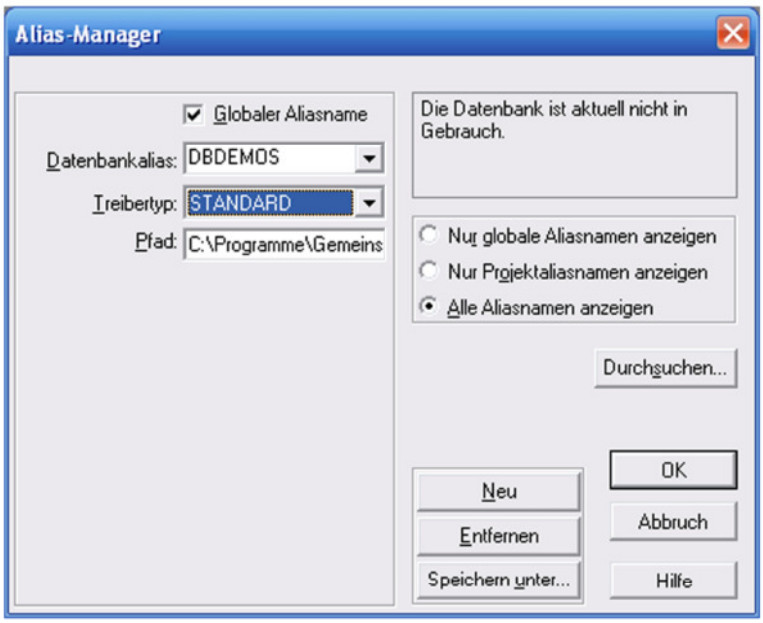

Abbildung 10.8: Bedienoberfläche des Alias-Managers

der Programmcode von der Platzierung der Datenbank auf dem jeweiligen Rechner unabhängig.

Für den Betrieb des Programms muss sowohl auf dem Entwicklungs- als auch auf dem Zielrechner z. B. mittels des Programms Datenbankoberfläche (**dbd32.exe**) die Zuordnung zwischen Alias und Dateipfad festgelegt werden.

Alias muss auf Entwicklungs- **und** Zielrechner festgelegt werden.

Programmgesteuert kann ein lokaler Alias mit der `TDatabase`-Komponente und ein globaler Alias mit der `TSession`-Komponente erstellt werden. Details zu diesem Thema werden in einem Folgeband behandelt.

10.2.2.4. Definitionen für die gesamte Datenbanktabelle

Die wichtigsten Eigenschaften und Methoden von TTable haben Sie in den Tabellen 10.3 und 10.4 kennen gelernt.

Definition von Datenbankfeldern

Die Gesamtheit der Felddefinitionen ist in der Eigenschaft *`FieldDefs`* der Komponente `TTable` untergebracht. Die einzelnen Definitionen (Typ `TFieldDef` (Singular!)) sind dabei in *`FieldDefs->Items`* untergebracht.

11. Nutzen Sie die Möglichkeiten Dynamischer Link-Bibliotheken (DLLs)

11.1. Ein bisschen Theorie

Eine DLL (Dynamic Link Library, Dynamische Link Bibliothek) ist eine Bibliothek, die dynamisch an das jeweilige Hauptmodul (z. B. ein Hauptprogramm) angebunden werden kann. Sie wird ihrem Nutzer also erst zur Laufzeit zugeordnet.

Man unterscheidet - und das klingt fast ein bisschen paradox - die statische Anbindung und die dynamische Anbindung. Die statische Anbindung erfolgt zu Beginn des Prozesses, also zur Beginn der Laufzeit. In diesem Fall muss die benötigte DLL bereits zu Beginn der Laufzeit vorhanden sein. Der Arbeitsspeicherplatz zur Aufnahme der DLL muss während der gesamten Dauer des Prozesses bereit gestellt werden.

Wird die DLL dynamisch angebunden, so kann dies vollkommen bedarfsorientiert geschehen. Die DLL wird dann ausschließlich für den Zeitraum, in dem sie benötigt wird angebunden, unmittelbar danach kann sie wieder abgetrennt werden. Der Ressourcenbedarf für die DLL muss also nur in dem Zeitraum befriedigt werden, wo er tatsächlich auch benötigt wird. Mehrere nicht zeitgleich benötigte DLLs können sich denselben Speicherplatz teilen.

Eine DLL spart oft Resourcen

11.1.1. Möglichkeiten der DLL

DLLs sind ein wesentliches Element des professionellen Softwareengineering. Sie bieten die folgenden Vorteile:

- Es gibt ein extrem breites Angebot vorgefertigter professioneller Lösungen.
- Da die Bibliotheksmodule erst zur Laufzeit gebunden werden, ist die EXE-Datei selbst meist wesentlich kleiner als bei statischer Bindung.
- Das Hauptprogramm (EXE-Datei) und die einzelnen DLLs können in unterschiedlichen Programmiersprachen erstellt werden.

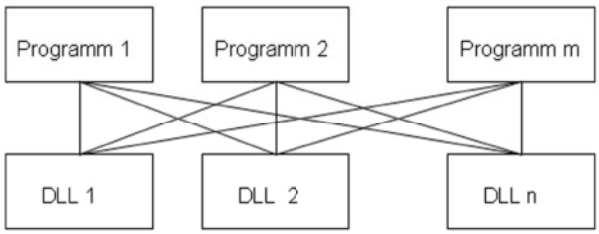

Abbildung 11.1: Eine DLL kann von mehreren Programmen genutzt werden.

- Wartungsarbeiten reduzieren sich auf einfaches Kopieren an Stelle von Binden und Installieren.

11.1.1.1. Nutzung einer DLL durch mehrere Programme

Wenn eine Aufgabe von mehreren Programmen auszuführen ist, dann kann diese Aufgabe von ein und derselben DLL erledigt werden (Abbildung 11.1). Das kann den Umfang von EXE-Dateien extrem reduzieren. Üblich ist dies z. B. bei der Implementation von Betriebssystem-Funktionen.

11.1.1.2. Realisierung hybrider Programmkonzepte

DLLs können in den unterschiedlichsten Programmiersprachen erstellt werden. Bekanntlich ist es so, dass für eine gegebene Aufgabenstellung nicht jede Programmiersprache gleich gut geeignet ist. Verwendet man für grafische Bedienoberflächen oder arithmetische Berechnungen vorteilhaft C++, Pascal oder auch Java, so wird man für die Lösung logischer Aufgabenstellungen eher Prolog verwenden. Mit dem DLL-Konzept kann jeder Programmteil in der für ihn angemessenen Programmiersprache erstellt werden. Hiervon ist eine deutliche Rationalisierung der Programmentwicklung zu erwarten.

Im Abschnitt 11.5.3.1. wird gezeigt, wie Delphi-DLLs durch C++Builder-Programme genutzt werden können. Weitere Sprachkombinationen werden in einem Folgeband behandelt.

11.1.1.3. Skalierung von Programmen

Versionierung mit DLLs: Ein Name verschiedene Inhalte

Unterschiedliche Programmversionen können durch unterschiedliche DLLs bzw. unterschiedliche Ausführungen gleichnamiger DLLs realisiert werden. Dabei werden einfach die erforderlichen Dateien zusammengestellt. Ein Bindevorgang (Linkvorgang) ist nicht erforderlich. So können durch einfaches Zusammenstellen von Dateien verschiedene Programmversionen (Light, Standard, Professional, …) generiert werden.

11.1.1.4. Verbesserung der Wartbarkeit

Updates ohne Bindevorgang

Im Gegensatz zur Verwendung statischer Bibliotheken, kann bei Verwendung von DLLs das Programm dadurch geändert werden, dass einzelne DLLs ausgewechselt werden. Ein Binde- (Link-) Vorgang, der in der Regel nur vom Entwickler durchgeführt werden kann, entfällt damit und die EXE-Datei bleibt unverändert. Dieser Wartungsvorgang kann in verschiedenen Graden automatisiert werden. Das reicht von der Information des Benutzers über Neuerungen mit anschließendem manuellem Dateiersatz bis zum vollautomatischen Ersatz veralteter DLLs durch das Programm selbst (z. B. beim Programmstart). Letzteres setzt voraus, dass die DLLs dynamisch angebunden werden (11.2.8.).

12. Internetanwendungen

12.1. Das (Open-Source-) Projekt Indy

Indy[23] (Internet Direct, früher Winshoes) ist ein Open Source Projekt das von Freiwilligen mit Hilfe industrieller Sponsoren bearbeitet wird [INDY]. Seit den ersten Anfängen von Chad Z. Hower im Jahre 1993 hat sich Indy stark entwickelt und ist heute eine umfassende Socket-Bibliothek, die auf verschiedenen Plattform verfügbar ist. Anwendungsprogrammierern stellt Indy einen einfachen und zuverlässigen Zugriff auf die Dienste des Internets zur Verfügung zu stellen. Mehr als 100 Hochebenen-Protokolle sind leicht zugänglich implementiert, darunter SMTP, POP3, HTTP und NNTP

Schon 2001 hat Borland Indy im Rahmen seiner Softwareentwicklungsumgebungen C++Builder und Delphi lizenziert. Heute sind die Indy-Komponenten fester Bestandteil des Embarcadero RAD-Studios. Darüber hinaus werden Kylix, C#, Visual Basic .NET und Delphi .NET unterstützt. Die Unterstützung von Free Pascal und Lazarus[24] ist in Vorbereitung.

Das Indy-Handbuch umfasst mehr als 4500 Seiten! Bisher ist es nur für Delphi verfügbar.

Das Indy-Rahmenwerk implementiert ca. 600 Klassen. Das Handbuch umfasst mehr als 4500 Seiten. Es liegt derzeit ebenso wie auch die Hilfeseiten leider nur für Delphi vor. Die Vorgehen für C++ muss aus den existierenden Unterlagen durch Analogschluss abgeleitet werden. Die Klasse `TIdSMTP` besitzt 32 Methoden, bei den anderen verwendeten Klassen sind die Verhältnisse ähnlich.

Nur Lösungsprinzip, keine Details!

Schon aus diesen Zahlen geht hervor, dass weder das gesamte Indy-Projekt noch einzelne wichtige Klassen desselben im Rahmen dieses Buches erschöpfend behandelt werden können. Dementsprechend können die Beispiele auch nur das Prinzip nicht aber universelle Lösungen aufzeigen.

Wer tiefer in Indy einsteigen will oder muss, dem bleibt das Studium der o. e. Dokumentation nicht erspart. Hierfür gilt folgende Empfehlung:

Die grundsätzlichen Details zu Indy finden Sie in der (Delphi-) Dokumentation und in der Online-Hilfe. Die dortigen Angaben gelten sinngemäß auch für die Programmierung in C++. Was evtl. Schwierigkeiten bereitet sind die Einzelheiten der Metho-

---

23 Der Einfachheit halber wird hier allein das Wort Indy sowohl als Bezeichnung für das Projekt wie auch des Rahmenwerks verwendet.

24 Lazarus (www.lazarus.freepascal.org) ist ein Open Source Project. Es ergänzt Free Pascal um eine grafische Entwicklungsumgebung, die der von älteren Delphi-Versionen stark ähnelt. Ähnliches gilt für die verwendeten Klassenbibliotheken. In der Reihe Informatik-ganz-einfach ist für Sommer 2011 als Band 3 der Titel *Lazarus - professionell programmieren von Anfang an* geplant.

denschnittstellen. Für die Methode *Get* der Klasse TIdFTP (FTP-Client) finden Sie z. B. in der Delphi-Dokumentation die folgende Schnittstelle:

```
procedure Get(
  const ASourceFile: string;
  const ADestFile: string;
  const ACanOverwrite: boolean = false;
  AResume: Boolean = false
); overload;
```

Abbildung 12.1: Hilfe-Seite mit Informationen zu Klassenhierarchie und (Header-) Datei

Schnittstellenbestimmung im Zweifelsfall über die Header-Datei.

Die zu gehörige C++-Schnittstelle finden Sie in der Schnittstellen- (Header-) Datei zur Klasse TIdFTP. Den Namen der Schnittstellendatei wiederum können Sie aus Angaben der Hilfeseite herleiten. Wenn Sie die Hilfe-Seite zu TIdFTP öffnen (Abbildung 12.1) finden Sie dort zu Datei (File) die Angabe *IdFTP*. Das ist der Namensstamm der Schnittestellendatei. Da diese aus einem Pascal- (Delphi-) Programm generiert wurde lautet ihr voller Name **IdFTP.hpp**. Dort finden Sie die Methodenschnittstelle

```
void __fastcall Get (
    const System::UnicodeString AsourceFile,
    const System::UnicodeString AdestFile,
    const bool ACanOverwrite = false,
    bool AResume = false);
```

Finden Sie eine Methode oder auch Eigenschaft nicht in dieser Schnittstellendatei so können Sie annehmen, dass sie vererbt wurde. Somit muss die Suche in der Schnitt-