Druckwunder

Die Neuerungen von List & Label 11 unter der Lupe

von Patrick Theobald

Seit vielen Jahren gilt *List & Label* von *combit* als der Platzhirsch unter den Reportgeneratoren. Grund genug sich die neu erschienene Version 11 etwas genauer anzuschauen.

Die Konstanzer Softwarefirma *combit* setzte schon zu 16-Bit-Zeiten mit Ihrem Reportgenerator *List & Label* neue Maßstäbe. Nicht nur die klassische Reportgenerierung und der Druck waren immer das Entscheidende, sondern auch die Tatsache, dass der komplette Reportdesigner mit der Endanwendung lizenzkostenfrei an den Kunden ausgeliefert werden konnte. Da ein eingebauter und an die Anwendung angepasster Reportgenerator praktisch jedes Software-Produkt mit relativ wenig Aufwand enorm aufwertet und dies ein mächtiges Verkaufsargument gegenüber dem Endkunden darstellt, hat sich um das Produkt eine treue Fangemeinde entwickelt, die natürlich gespannt auf die Version 11 wartete.

Die Grundidee von *List & Label* und somit die Basis für die mittlerweile weite Verbreitung ist eine Datenbankunabhängigkeit. Anders als bei anderen Reportgeneratoren kann das Beschaffen der Daten und das Aufbereiten zum Report sauber voneinander getrennt werden. Das beschert dem Entwickler zwar einen etwas höheren Aufwand in der Programmierung, zahlt sich aber im Endergebnis mehrfach aus. So kann die Datenversorgung völlig flexibel gestaltet werden und ist nicht auf Datenbankabfragen beschränkt. Genau das war aber auch immer wieder ein Kritikpunkt, denn das Thema *Data Binding* wurde in der Vergangenheit recht stiefmütterlich behandelt. Helfen soll die neue Version 11, in der die .NET-Komponente in Bezug auf die Datenbindung komplett überarbeitet wurde. Die folgenden Beispiele sollen Interessierten wie „Alten Hasen“ neben dem überarbeiteten *Data Binding* natürlich auch die übrigen neuen Features vorstellen.

kurz & bündig

Inhalt

Eine Übersicht und Beispiele zu den neuen Features von List & Label 11

Zusammenfassung

Die wichtigste Neuerung für .NET-Entwickler sind die Verbesserungen an der .NET-Komponente, sodass zum Beispiel Tabellen mit Beziehungen besser unterstützt werden

Quellcode

C#

Quellcode auf CD

Hierarchische Datenbindung

Das im Folgenden vorgestellte Beispielprogramm (Sie finden es auch auf der Heft-CD) zeigt das Anfertigen eines Berichts mit drei unterschiedlichen Arten der Datenbindung. Die Grundlage bildet eine kleine Access-Datenbank, die Daten einer kleinen CD-Sammlung enthält. Abbildung 1 zeigt das zugehörige, typisierte *DataSet* in Visual Studio. Die Tabelle *CDs* enthält pro CD einen Eintrag – die Tabelle *CDTitel* in einer *1:n*-Beziehung alle Titel der CD mit Laufzeit (in Sekunden). Über die *KategorieID* verweist eine CD auf die *Kategorien*-Tabelle. Alles in allem eine schöne, hierarchische Datenbeziehung, mit der sich die neuen Features besonders eindrucksvoll zeigen lassen. Listing 1 enthält der Vollständigkeit halber das Füllen des *DataSet* mit den Daten der Datenbank mithilfe einer *OleDbConnection*-Komponente.

Das *List-&-Label*-Steuerelement wurde dem Formular als Komponente hinzugefügt. Listing 2 zeigt den Aufruf des Designers. Zunächst werden wie beim *Data Binding* üblich die Eigenschaften *DataSource* und *DataMember* mit den *List-&-Label*-Objekt „verdrahtet“. Die „spannendste“ Eigenschaft ist *AutoMasterMode*. Sie entscheidet darüber, wie sich *List & Label* gegenüber hierarchischen Datenbeständen verhält. Danach wird der Designer mit der Methode *Design* gestartet. Die Übergabeparameter steuern die Projektart (*LlProject.List* steht für Reports in Listenform) und das Verhalten bei der Auswahl der Datei, die die Reportdefinition enthält (*LlProject.FileAlsoNew* lässt den Benutzer auch neue Report-Definitionsdateien anlegen).

Im Designer lässt sich schnell feststellen, dass nur die Tabelle *CDs* zum Druck zur Verfügung steht. Mit der Übergabe von *None* bei der Eigenschaft *AutoMasterMode* wurde *List & Label* angewiesen, verknüpfte Tabellen zu ignorieren und nur die im *DataMember* angegebene Tabelle an den Report zu binden. Ganz anders verhält es sich, übergibt man *AsVariables*. In diesem Fall werden die Zeilen der angegebenen (Haupt)-Tabelle als Variablen übergeben und alle verknüpften Tabellen stehen als Listenfelder zu Verfügung. Abbildung 2 zeigt einen auf diese Weise generierten Report in der Voransicht. Die Haupttabelle (also die CDs mit Interpret und Albumtitel) erzeugt pro Eintrag eine neue Seite. Die hierarchisch darunter liegenden Daten (nämlich die Einzeltitel) finden sich als Felder in der Liste wieder. Ein analoges Beispiel zu dieser Art der Da-

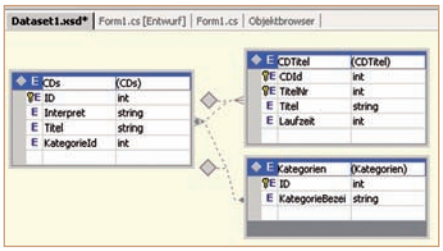

Abb. 1: Das typisierte DataSet für das Beispielprogramm

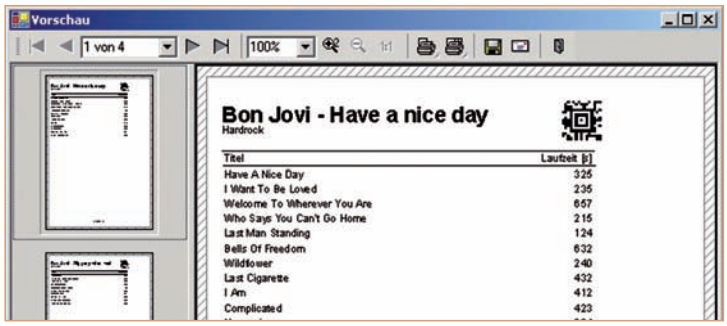

Abb. 2: Beispielreport in der Voransicht mit einer Kombination aus Variablen und Feldern

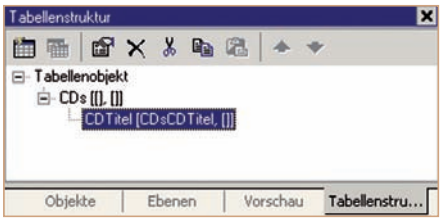

Abb. 3: Neues Fenster: Hierarchiestufen werden durch den Endbenutzer definiert

tenbindung wären Köpfe und Positionen von Rechnungen. Eine Rechnung (Haupttabelle) entspricht einem Blatt, auf dem tabellarisch die Einzelpositionen angeordnet sind. Der 2D-Barcode aus Abbildung 2 ist im Übrigen vom Typ *Aztec* und wird ab der Version 11 ebenfalls unterstützt.

Die dritte und letzte Art der Datenbindung kann durch Übergabe von *AsFields* an *AutoMasterMode* erfolgen. Alle Tabellen im *DataSet* werden dann gleichberechtigt, aber unter Wahrung der Datenrelation als Listenfelder an *List & Label* übergeben. Dadurch ist eine Verarbeitung möglich, die dem Endbenutzer nahezu jegliche Freiheit in Sachen Gruppierung und Hierarchiedarstellung bietet. Abbildung 3 zeigt, wie die logische Darstellung im Designer vom Endbenutzer definiert wird. Es ist zwar nicht möglich, mehrere Tabellenobjekte in das Projekt einzupflegen. Dieser Umstand ist durch die selbstdefinierte Hierarchie aber auch überflüssig geworden. Jede zusätzlich eingepflegte Hierarchieebene erzeugt innerhalb des einzigen Tabellenobjekts einen Subreport, der für sich wiederum alle Möglichkeiten von Kopf und Fußzeilen sowie Gruppierungen ermöglicht. In Abbildung 4 ist die Voransicht eines solchen Subreports zu sehen. Die CDs bilden den äußeren Rahmen (zusammen mit der Klarschrift der Kategorie), während die CD-Titel sich als eigenständige Untertabelle dazwischen einfügen. Bei dieser Gelegenheit sei nebenbei noch auf das neue Zoom-Verhalten hingewiesen. In der Preview kann dies nämlich mittlerweile in einer stufenlosen Vergrößerung erfolgen.

Grundsätzlich lässt sich solch eine hierarchische Anwendung auch mit älteren Versionen realisieren – allerdings deutlich umständlicher. Bis Version 10 war es nötig, zu diesem Zweck Gruppierungen und die damit verbundenen Wechselbedingungen zu definieren. Die Fähigkeit, Datenhierarchien abzubilden, macht vieles einfacher – insbesondere für den Endanwender. Auch die Darstellung der Felder und Variablen innerhalb des Designers ist übersichtlicher, denn sie sind nach ihren Tabellen in eigenen Unterordnern gruppiert.

Wer trotz der neuen Möglichkeiten des *Data Binding* wegen der besseren Kontrolle über die Datenbeschaffung lieber bei der klassischen, manuellen Übergabe bleiben möchte, braucht auf hierarchische Strukturen nicht zu verzichten. Tabelle 1 erklärt die neuen Funktionen, um Hierarchien händisch an *List & Label* zu übergeben. Das im Installationsumfang enthaltene Beispielprojekt *Multitab.sln* zeigt die Anwendung dieser Funktionen. Einzige Wermutstropfen sind die Funktionen *LlPrintFieldsEnd* und *LlPrintFields*, die innerhalb der Druckschleife für die Datenübergabe genutzt werden. Sie sind leider nicht mit der Vorgängerversion kompatibel. Hier kann bestehender .NET-Code nicht ohne Änderung übernommen werden, da die beiden Funktionen ab Version 11 keinen booleschen Wert mehr zurückgeben, sondern stattdessen einen Integer. Diese Änderung musste so vom Hersteller vorgenommen werden, weil ansonsten das aufrufende Programm keine Chance

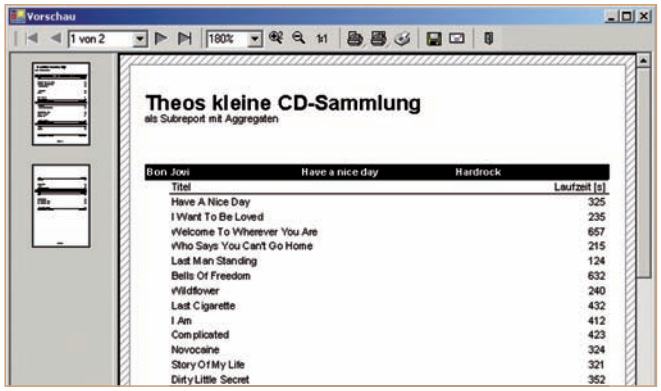

Abb. 4: Voransicht eines Reports mit Subreport

Listing 1

```
public Form1()
{
 InitializeComponent();

 // Kategorien laden
 OleDbDataAdapter da1 = new OleDbDataAdapter(
  "select * from Kategorien",this.oleDbConnection1);
 da1.Fill(this.dsCD,"Kategorien");

 // CDs laden
 OleDbDataAdapter da2 = new OleDbDataAdapter(
  "select * from CDs",this.oleDbConnection1);
 da2.Fill(this.dsCD,"CDs");

 // CDTitel laden
 OleDbDataAdapter da3 = new OleDbDataAdapter(
  "select * from CDTitel",this.oleDbConnection1);
 Da3.Fill(this.dsCD,"CDTitel");
}
```

Listing 2

```
private void btnDesignSimpleTable_Click
                    (object sender, System.EventArgs e)
{
 try
  {
   listLabel1.DataSource = this.dsCD;
   listLabel1.DataMember = "CDs";
   listLabel1.AutoMasterMode =
                                LlAutoMasterMode.None;
   listLabel1.Design(LlProject.List | LlProject.
                                                FileAlsoNew,
    @"..\..\SimpleTable.lst");
  }
 catch (Exception e1)
 {
  MessageBox.Show(e1.Message);
 }
}
```

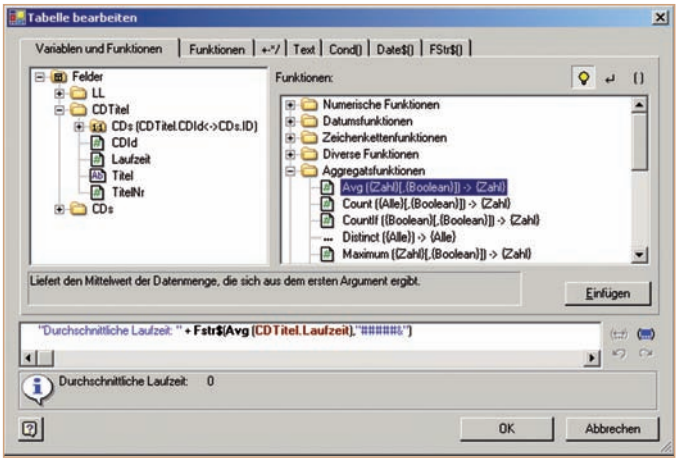

Abb. 5: Neue Aggregatsfunktionen im Formeleditor

hätte, den Wechsel einer Druckseite vom Wechsel einer Tabelle in der Datenbeschaffung mitzubekommen.

Aggregatsfunktionen

Summenvariablen waren unter *List & Label* schon immer das Mittel der Wahl, um nicht nur Summen mitlaufen zu lassen, sondern beispielsweise auch den Durchschnittswert einer kumulierten Datenspalte zu errechnen. In der Praxis hat sich das hin und wieder als sehr umständlich herausgestellt. Manche Anforderungen waren überhaupt nicht umsetzbar (z.B. kleinste oder größte vorkommende Werte). Mit einem Schwung an neuen Aggregatsfunktionen tun sich hier mit der Version 11 vollkommen neue Möglichkeiten auf. Abbildung 5 zeigt die Ansicht des Formeleditors im Designer. Das Beispiel zeigt ein Feld in der Fußzeile des Subreports. Es wird die durchschnittliche Liedlänge berechnet (Aggregatsfunktion *Avg*). Tabelle 2 listet die wichtigsten Aggregatsfunktionen.

Excel-Ausgabe und weitere Features

Neben dem reinen Ausdruck unterstützt *List & Label* natürlich auch eine ganze Reihe von weiteren Ausgabemedien, darunter auch den Export nach Excel. Im Fall von Reports, die etwas komplizierte Tabellenlayouts hatten, waren aber die exportierten Excel-Sheets entsprechend filigran und bestanden aus einer ganzen Reihe von zusammengeschobenen Zellen, um die korrekte Wiedergabe des ursprünglichen Layouts in Excel sicherzustellen. Optisch war diese Sache zwar sauber, führte aber beim Endanwender oft zur Frustration, weil dieser das Gewirr unterschiedlichster Layoutzellen erst einmal entwirren musste, um ergänzende Berechnungen oder Formeln einzutragen. Hier hat *combit* in der neuen Version viel getan. Laut Verkaufsprospekt konnte die Zahl der benötigten Layoutzellen um bis zu 50 Prozent optimiert werden. Ob es nun genau 50 Prozent sind, vermag der Autor dieses Artikels nicht nachvollziehen, aber die Verbesserung ist deutlich zu spüren und wird den Endanwendern zu Gute kommen. Als ein weiteres neues Feature sei zum Schluss noch der überarbeitete Rahmeneditor genannt: Es können mehrere Feldrahmen auf einmal geändert werden. Außerdem ist es möglich, die Spaltenüberschriften einer Tabelle um 90 Grad zu drehen, um den Platzbedarf zu optimieren.

Funktion	Erklärung
LIDbAddTable()	Fügt dem Projekt eine neue, tabellarische Einheit hinzu.
LIDbAddTableRelation()	Definiert eine Beziehung zwischen zwei Tabellen.
LIDbAddTableSortOrder()	Definiert eine Sortierspalte für eine Tabelle, die im Designer ausgewählt werden kann.
LIDbSetMasterTable()	Setzt diejenige Tabelle, die als äußerer Rahmen für einen Report mit *1:n*-Beziehung gelten soll.

Tabelle 1: Zusätzliche Funktionen zur hierarchischen Datenübergabe „von Hand“

Funktion	Erklärung
Avg(...)	Bildet den Durchschnitt von numerischen Datenspalten.
Count(...)	Ermittelt die Anzahl der Zeilen.
CountIf(...)	Zählt die Anzahl der Zeilen, für die eine logische Bedingung zutrifft.
Maximum(...), Minimum(...)	Ermittelt den kleinsten / größten numerischen Wert einer Datenspalte.
Sum(...)	Ermittelt die Summe einer numerischen Datenspalte.
Variance(...)	Ermittelt die Varianz einer Datenspalte.

Tabelle 2: Die wichtigsten Aggregatsfunktionen im Überblick

Fazit

List & Label wurde in den letzten Jahren kontinuierlich optimiert. Auch wenn man der Meinung sein könnte, dass sich die letzte Generation der Reportgeneratoren praktisch nicht mehr verbessern ließe, wartet Version 11 noch einmal mit einer Fülle praktischer Features auf. Die wichtigste Neuerung für .NET-Entwickler ist das hierarchische *Data Binding*, auf das viele sehnsüchtig gewartet haben dürften. In diesem Punkt haben die Entwickler bei *combit* ihre Hausaufgaben mit Bravour erledigt. Aber auch die neuen Aggregatsfunktionen bieten die Möglichkeit, hässliche und komplizierte Formelkonstrukte elegant zu ersetzen. Nehmen Sie die neuen Möglichkeiten zum Anlass, vorhandene Reports auf diese „Altlasten“ hin zu optimieren. Unter [2] finden Sie einen Link zum Download der .NET-2.0-DLL, die seit Erscheinen des Final Release des .NET Framework 2.0 zur Verfügung steht. Sie ist zum jetzigen Zeitpunkt noch nicht im Standardinstallationsumfang enthalten und muss deshalb bei Bedarf separat heruntergeladen werden. Version 11 hinterlässt einen hervorragenden Eindruck – Report unter .NET auf einem neuen Level. ●

Patrick Theobald ist Geschäftsführer der Theobald Software GmbH und Fachbuchautor. Sie erreichen ihn unter patrick.theobald@theobald-software.com.

● Links & Literatur

[1] www.combit.de

[2] www.combit.net/page674.aspx?action=display&articleid=KBTE000662