Cor Ligthert is Microsoft MVP Visual Developer

De eenvoud van de ADO.NET designer

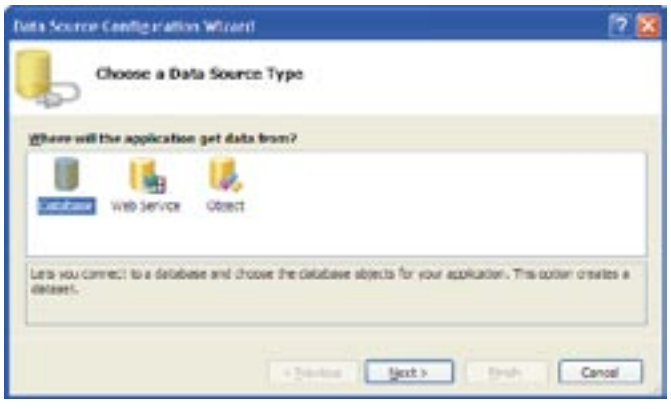
EEN DATAENTRY-PROGRAMMA VIA VISUAL STUDIO MAKEN

Microsoft is voor mij altijd een kampioen geweest in het maken van mooie dingen die vervolgens weinig publiciteit krijgen, omdat het bedrijf zich richt op andere zaken die commercieel veel beter klinken. Iets waar wel eens meer aandacht voor mag worden gevraagd zijn volgens mij de designer-functies van ADO.NET 2.0.

In dit artikel bekijken we de mogelijkheid om op eenvoudige wijze via Visual Studio een dataentry-programma te kunnen maken. Als datatable is gekozen voor de Orders-tabel van de Northwind-database in SQLServer. Dit is niet gedaan omdat hij representatief is als oplossing (hiervoor zou Master Detail beter zijn), maar omdat hij de relaties bevat naar andere tabellen om de basismogelijkheden te presenteren. We blijven in dit artikel bij een enkele tabel om de stappen goed in beeld te kunnen brengen. In een vervolgartikel zal een Master Detail-oplossing worden beschreven.

Een dataentry-programma

Het maken van een dataentry-form bestond al in de klassieke Visual Basic-versies. Het was er ook in de versies 2002/2003, maar dan ergens verborgen in een speciaal dataformulier. Ik heb altijd gedacht dat het ADO.NET-team zich hiervoor schaamde. Kenmerkend van deze versie was dat als je het eenmaal had gemaakt, er niets meer te wijzigen was. De versie 2005 bevat een gelijksoortige DataEntry-maker, maar nu met vele mogelijkheden om dit via de designer steeds weer bij te werken. Jammer genoeg is het nog niet helemaal compleet. Ik heb hierover contact gehad met het ADO.NET-team en er zal serieus aandacht aan worden besteed. Zeer belangrijk voor degenen die sceptisch zijn en de designer van de versies 2002/2003 kennen, is dat aanpassing via code heel goed mogelijk, zonder dat je steeds opnieuw hoeft te beginnen. *Een opmerking voor je begint: maak vooraf een goede veiligheidskopie van de NorthWind-database, want deze kan worden gemuteerd door deze voorbeeldapplicatie.*



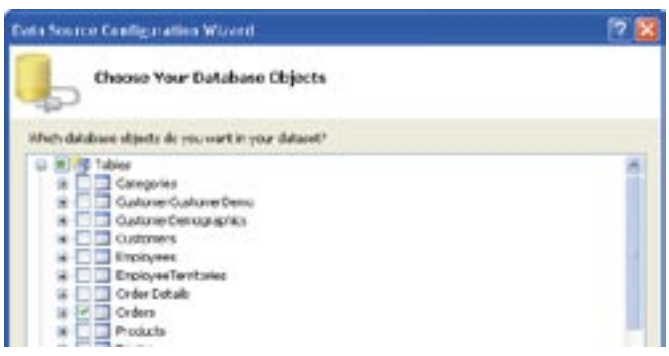
Afbeelding 1. Add New Data Source Wizard

Het begin van deze grotendeels walkthrough

Start als eerste stap Visual Studio en kies een WindowsForm-applicatie die je op de standaardwijze een naam geeft. Er wordt zo weinig code gebruikt dat het niet uitmaakt of je voor C#, C++ of Visual Basic kiest. In dit artikel wordt de code uitsluitend in Visual Basic gegeven, omdat ik daar de voorkeur aangeef. Naast dat het bouwen van een DataEntry-applicatie eenvoudig is, kun je een project bouwen om te zien hoe iets in code moet. De gegenereerde code (en alles wordt gegenereerd) kan volledig worden getoond (mits dat waar nodig in Solution Explorer is vermeld: Show All). Het is dus al heel leuk om te zien hoe de designer het doet in C#, C++ en Visual Basic.

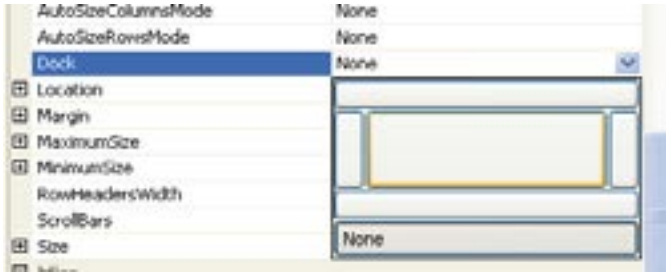
Het maken van de datasource van de orders-tabel

Het probleem van het volgende voorbeeld is dat er meer tekst en plaatjes zijn dan handelingen. Alles is in een minuut te doen, terwijl het lezen (en schrijven) veel meer tijd in beslag neemt. Als Visual Studio is gestart, maken we het formulier zo breed mogelijk en kiezen we 'Data'. Uit dit menu kiezen we nu Add New Data Source. De voor velen waarschijnlijk bekende wizard verschijnt in beeld; zie afbeelding 1.
Normaliter is de database default, zodat we direct op Next klikken. Vervolgens klik je op New Connection en selecteer je de gewenste database-server. Laat wel de authentication op Windows Authentication (of zet hem op Authentication) en selecteer vervolgens de Northwind-database en druk op Test Connection. Als het is gelukt, klik je op OK. In het volgende scherm laten we de defaults staan en kiezen wederom voor Next. In beeld verschijnt nu in het scherm van de Data Source Connector Wizard de keuze voor Database Objects. We selecteren Tables (via de checkbox) Orders (zie afbeelding 2) en drukken op Finish.



Afbeelding 2. Keuze voor Database Objects

Afbeelding 3. Overzicht met beschikbare datasources in het project



Afbeelding 5. De Dock-property instellen op Fill

De datasource is nu aangemaakt. Dit resulteert in een complete Strongly Generated Dataset class. Deze is te vinden in de Solution Explorer. De Strongly Typed DataSet in versie 2.0 is volledig ongelijk aan de versie van 1.x . In 1.x waren er gedeelten in de UI geplaatst. De nu in 2.0 gegenereerde dataset behoort als zelfstandige class tot de applicatie. Een onderdeel van deze class zijn een of meer TableAdapters, deze overerven de DataAdapter.

Het maken van het DataEntry-inputformulier

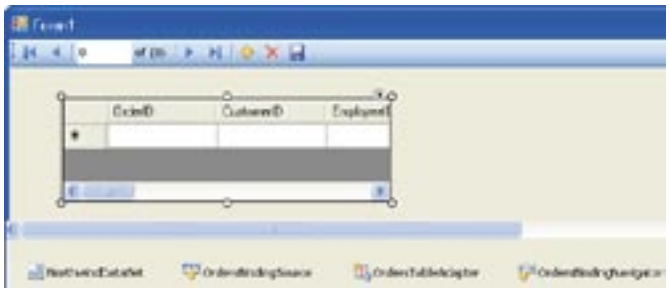
Dit gaat volledig generiek. We kiezen hiervoor weer het menu Data en daarna kiezen we dit keer voor Show Data Sources. Als dit is gebeurd, opent links de box Data Sources; zie afbeelding 3. We trekken hieruit de ordertabel op het ontwerpformulier. Een verassing, Visual Studio creëert nu een complete applicatie. We zien een keurig scherm met een complete navigatiebalk. Tevens worden er allerlei objecten gecreëerd voor de verwerking; zie afbeelding 4.
Het is wel aardig om nu even naar de gegenereerde code te kijken in Solution Explorer. Klik op de Show All Files.
De code voor het schermdesign staat in de Form1.designer.vb.
De code voor de dataset staat in NorthWindDataset.desinger.vb na openen van het plusteken voor de XSD. In de app.config staat onder andere de connection string. De code voor het lezen en updaten staat in Form1.vb. De gegenereerde code ziet er uit als in codevoorbeeld 1 (We hebben dit nodig voor de rest van dit artikel).

De presentatie van het DataEntry-formulier

Uit designoogpunt is de presentatie op het scherm zeer summier, maar alle elementen zijn aanwezig. Daarnaast is de winform-designer in mijn ogen zo verbeterd, dat iedereen naar zijn eigen wensen een mooi design kan maken. We moeten overigens niet vergeten om de Dock-property minimaal op Fill te zetten voor deze lange datagridview; zie afbeelding 5.

De eerste proef

We lopen nu even door het programma om te bekijken hoe het er uit ziet; zie afbeelding 6. Als er op het floppypictogram wordt gedrukt, verandert daadwerkelijk de database, dus denk om de veiligheidskopie van de NorthWind-database. Het getoonde resultaat ziet er leuk uit, maar eigenlijk behoren de waarden voor de Customers en de Employees uit de bijbehorende aanwezige gerelateerde bestanden te komen die via een combobox zijn te kiezen.



Afbeelding 4. Objecten die automatisch gegenereerd zijn.

Functionaliteit toevoegen aan onze dataentry-applicatie

Als we in de applicatie iets toevoegen of wijzigen, moeten we volledig voldoen aan de eisen (constraints), die hiervoor in de database zijn gesteld. Dit geldt ook voor het aanwezig zijn van een bestaande employee en customer. Wat in de vorige versie een ramp was, is in Visual Studio 2005 heel eenvoudig.

Datatables toevoegen aan NorthWind-dataset

Vergeet niet eerst de debugging te stoppen. We openen weer de box DataSources (als hij niet meer in beeld is, selecteer je hem in View) en zien bovenin vier pictogrammen. We kiezen de derde, de Configure Dataset with Wizard (zie ook afbeelding 3 waar dit pictogram reeds geselecteerd is). Door hier op te klikken verschijnt de Data Source Configuration Wizard, die we al eerder hebben gezien. We willen twee tabellen aan onze dataset toevoegen, maar hebben uitsluitend voor de combobox (en de relaties) de Id en de naam nodig. Alle andere extra's zijn ballast voor de databaseserver en voor een eventueel netwerk. Dit kan allemaal heel eenvoudig. We klikken op het plusteken op Employees en selecteren EmployeeID en LastName, bij de Customers selecteren we CustomerID en de CompanyName en klikken dan op finish, zie afbeelding 7). De dataset, de tabellen en de data-adapters worden nu bijgewerkt in de strongly typed Northwind-dataset.

Comboboxen met relaties toevoegen in de DataGridView

We selecteren de datagridview en selecteren met een rechtermuisklik 'edit columns'. Dit kan ook via de property-window van een geselecteerde datagridview. Het staat dan onderaan in het blauw. We kiezen in de linkerkolom 'Customer'; zie afbeelding 8. Daarna selecteren we bij kolomtype de DataGridViewComboBox. De propertytext wordt 'Customer'. Vervolgens klikken we op DataSource en kiezen (Let op dit is de eerste keer verwarrend) 'Other DataSources' en daarna voor NorthwindDataSet en dan voor Customers. Er wordt automatisch weer een CustomerBinding-

```
Public Class Form1
  Private Sub OrdersBindingNavigatorSaveItem_Klik _
  (ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles OrdersBindingNavigatorSaveItem.Klik
    Me.Validate()
    Me.OrdersBindingSource.EndEdit()
    Me.OrdersTableAdapter.Update(Me.NorthwindDataSet.Orders)
  End Sub

  Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    ' TODO: This line of code loads data into the
    ' NorthwindDataSet.Orders table. You can move,
    ' or remove it, as needed. NorthwindDataSet.Orders' table.
    ' You can move, or remove it, as needed.
    Me.OrdersTableAdapter.Fill(Me.NorthwindDataSet.Orders)
  End Sub
End Class
```

Codevoorbeeld 1

```
Private Sub OrdersBindingNavigatorSaveItem_Klik _
   (ByVal sender As System.Object, ByVal e As System.EventArgs) _
   Handles OrdersBindingNavigatorSaveItem.Klik
      Me.Validate()
      Me.OrdersBindingSource.EndEdit()
      Try
      Me.OrdersTableAdapter.Update(Me.NorthwindDataSet.Orders)
      Catch ex As Exception
      messagebox.Show(ex.tostring)
      End Try
End Sub
```

Codevoorbeeld 2.

Source aangemaakt. (Als je dit ooit wilt verwijderen, doe het dan onderaan het designformulier). Het display-member moet worden aangepast (dit kan pas nadat de vorige stap is gedaan). We kiezen voor CompanyName, en vervolgens kiezen we op dezelfde wijze bij ValueMember voor CompanyId. Hetzelfde doen we met de Employees, maar dan met de hiervoor juiste velden. Onderaan het designformulier zien we hierna de automatisch gegenereerde table-adapters en bindingsources verschijnen. Als we het nu even proberen zien we (als alles goed is gegaan) dit keurig functioneren. Het kan gebeuren dat er een pagina komt met allemaal fouten. Even het form sluiten in de designer en opnieuw openen. Ik weet niet of dit bugje verholpen is met de volgende SP.

De dataset editen en presenteren

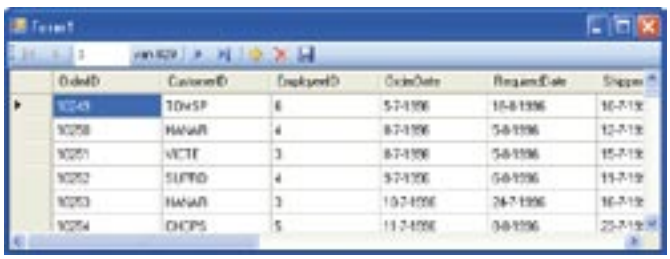
Een aardig hulpmiddeltje is de mogelijkheid de dataset te editen met de designer. Na het openen van de DataSource-box, klikken we op de tweede icoon die we al hebben gezien in afbeelding 3. In beeld verschijnt dan een mooie weergave van de dataset; zie afbeelding 9. In dit artikel gaan we hier niet verder op in.

Foutenbehandeling toevoegen

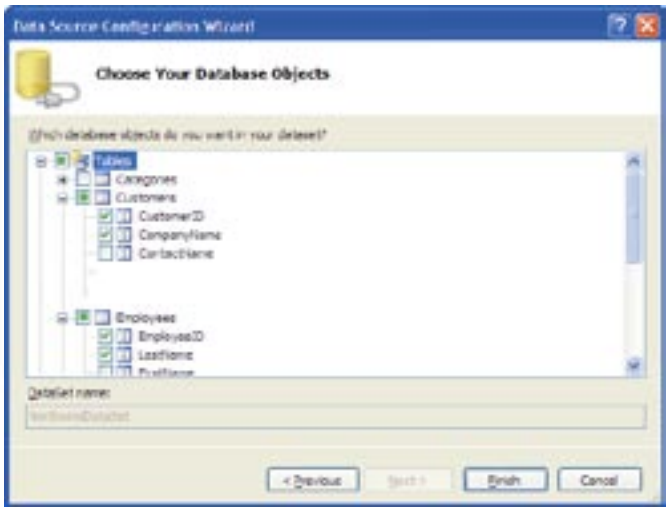
Het onderdeel dat nog niet compleet is - en waarover ik heb gecommuniceerd met de ADO.NET-team van Microsoft - is de controle en afhandeling van fouten. Hiervoor moeten twee zaken gebeuren, het afvangen van fouten en eventuele acties hiermee. Voor de duidelijkheid herhaal ik de code steeds volledig, Een voorbeeld van veelvuldig voorkomende fouten, is de gelijktijdige bewerking van meer gebruikers (Concurrency). Hiervoor een error-afhandeling maken is eenvoudig door het toevoegen van een Catch; zie codevoorbeeld 2.

De Update-methode in een DataAdapter

Tijdens het updaten door de DataAdapter kan er een fout worden geconstateerd. Het kan dan zijn dat de helft van de mutaties wel is verwerkt en de andere helft niet. Om dat te begrijpen leg ik in het kort de werking van de DataAdapter uit. Een DataTable is een tabel met DataRows. Zo gauw als hierop updates zijn, staan er twee versies van deze datarows in de datatable, namelijk de originele rows (slechts zichtbaar via bijvoorbeeld writeXML met de toevoeging diffgram) en de gewijzigde rows. Van nieuwe DataRows staan uitsluitend de nieuwe (Added), en bij verwijderen via 'Delete' uitsluitend de originele met een rowstate 'Deleted'. Als korte hint, let op het verschil tussen de methoden 'Remove' en 'Delete' die u veelvuldig in ADO.NET tegenkomt. 'Remove' betekent: verwijder direct compleet alles van deze DataRow uit de dataset, 'Delete' betekent: zet de rowstate op 'Delete'. Dit laatste weer als de Row niet is toegevoegd na de Fill (of een acceptchanges), want dan wordt een delete behandeld als een remove. DataRows die zijn 'removed', worden dus nooit bijgewerkt in de database. Voor belangstellenden is er een link naar meer informatie over de WriteXML-methode opgenomen in de lijst met referenties.

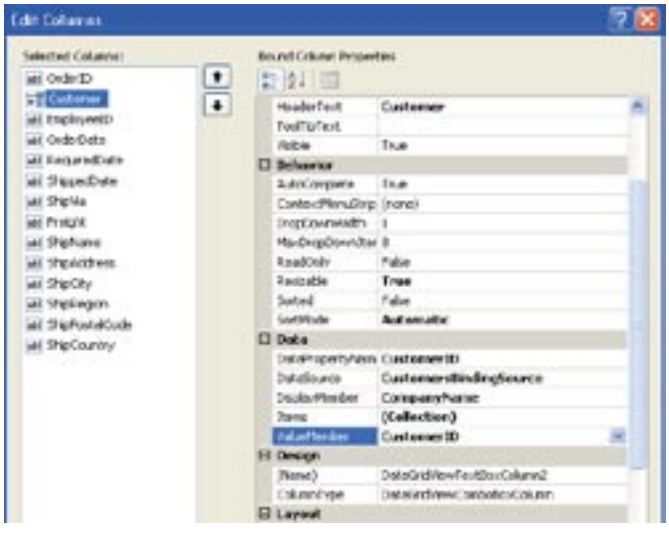


Afbeelding 6. Het resultaat

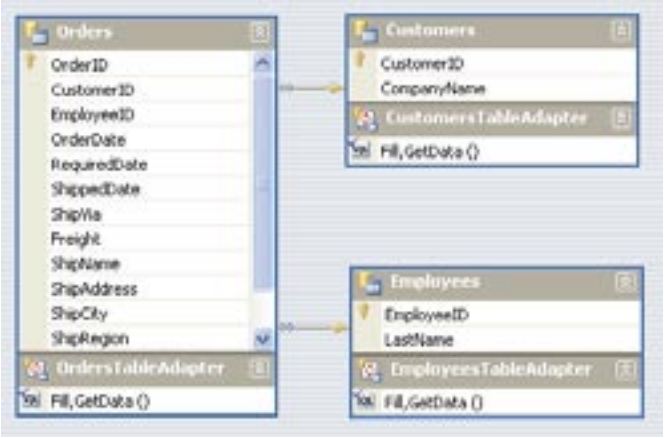


Afbeelding 7. Extra tabellen en velden toevoegen.

Voor de volgende tekst zijn in versie 2.0 alternatieve methoden. Ik beschrijf hierbij uitsluitend een van deze methoden (de standaard). Bij het updaten verwerkt de DataAdapter.Update de DataRows een voor een in de datatable. Als er wijzigingen zijn (als vermeld in de RowState) zal hij deze bijwerken in de database. Vooraf doet hij in de database een select van de Row om te kijken of alle originele waarden nog gelijk zijn. Is dit niet het geval, dan is er een fout door gelijktijdige verwerking en zal de verwerking worden gestopt. Er is een alternatieve methode, maar ook die vermeld ik niet in dit artikel, om deze afhandeling te besturen, is in .NET Framework 2.0 de nieuwe 'transactions' class opgenomen. Wat we doen om fouten te behandelen, is geheel conform onze eigen wil. In dit voorbeeld laat ik het aan de eindgebruiker over. Hij/zij kan, wat is gedaan, accepteren en de rest ignoren, maar hij/zij kan ook besluiten een rolback te doen.



Afbeelding 8. Properties per kolom bepalen.



Afbeelding 9. Dataset Designer

```
Private Sub OrdersBindingNavigatorSaveItem_Klik _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles OrdersBindingNavigatorSaveItem.Klik
   Me.Validate()
   Me.OrdersBindingSource.EndEdit()
   Dim conn As SqlClient.SqlConnection = _
      e.OrdersTableAdapter.Connection
   conn.Open()
   Dim tx As New System.Transactions.CommittableTransaction
   conn.EnlistTransaction(tx)
   Try
      Me.OrdersTableAdapter.Update(Me.NorthwindDataSet.Orders)
   Catch ex As Exception
      If MessageBox.Show("Er is een fout opgetreden: " _
      & ex.Message & vbCrLf _
      & "Alles accepteren tot deze fout " _
      & "en dan de nieuwe standen tonen" & vbCrLf _
      & "(Nee zet de standen zo als ze voor de laatste " _
      & "update waren)", "Databasefout", _
   MessageBoxButtons.YesNo, MessageBoxIcon.Error) = _
   Windows.Forms.DialogResult.Yes Then
         tx.Commit()
      Else
         tx.Rollback()
      End If
      conn.Close()
      conn.Open()
      NorthwindDataSet.Orders.Clear()
      Me.OrdersTableAdapter.Fill(Me.NorthwindDataSet.Orders)
   Finally
      conn.Close()
      tx.Dispose()
   End Try
End Sub
```

Codevoorbeeld 3.

De reference naar de transaction-class

De transaction-class zit zelfs niet in de standaardset van references van Visual Basic, we moeten dus een reference toevoegen. We doen dit door ergens (dit kan op meer plaatsen) voor 'Add reference' te kiezen. Bijvoorbeeld, door bovenin Solution Explorer op ons project met een rechtermuisklik en vervolgens op Add reference te klikken. We kiezen voor de System.Transactions. Voor degenen die deze box nog van versie 1.x kennen; zie eens hoe mooi resizable hij nu in 2.0 is? De grip is nog vergeten, maar dat is waarschijnlijk iets voor de volgende versie.

De commit- en rollback-procedure afmaken

Om dit te doen moeten we nog wat code toevoegen. Aan de code is als tekst niet zoveel toe te voegen. Gaat het goed, dan wordt de transaction goedgekeurd en uitgevoerd. Gaat het fout, dan vind er al dan niet een rollback plaats, of wordt het reeds uitgevoerde gedaan. Zie codevoorbeeld 3.

Completer

Hoewel er nog best wat zaken kunnen worden toegevoegd en veranderd (bijvoorbeeld business-rules onafhankelijk van de database), is de vernieuwing in ADO.NET van een niveau zoals we het waarschijnlijk al in 2002 wensten. Ik verwacht dat velen die op grond van eerdere ervaringen weigerden om op de vanuit de designer gecreëerde Strongly Typed Dataset over te gaan, toch ook spoedig deze stap zullen maken. Het kunnen maken van dit soort simpele DataEntry-schermen is een extra stimulans. Even iets doen dat zelfs eenvoudiger gaat dan met het vroegere dBASE, verwacht je natuurlijk niet in een echt complete ontwikkelomgeving. Die hierdoor dus nog completer is geworden.

Cor Ligthert (MVP) is waarschijnlijk een van de in Nederland langst actieve personen in de ICT. Dit laatste geldt zowel voor technische als voor organisatorische aspecten. Cor is wars van luchtfietserij. De tijd heeft hem geleerd dat wat vandaag alleen voor de sier wordt toegevoegd, morgen belachelijk is. Het goed doordacht zijn van .NET en de mogelijkheden om hiermee een stap in de toekomst te maken, heeft zijn bijzondere interesse voor .NET gewekt. Samen met zijn 'fellow' MVP Ken Tucker uit Florida onderhoudt Cor een website. www.vb-tips.com, die is gericht op het gebruik van Visual Basic .NET (ADO.NET, ASP.NET).

Referenties

De transaction-class:
http://msdn2.microsoft.com/library/system.transactions.aspx
WriteXML-methode: http://msdn2.microsoft.com/en-us/library/system.data.xmlwritemode.aspx
Data access development center: http://msdn.microsoft.com/data/
ADO.NET informatie: http://msdn.microsoft.com/library/en-us/dnanchor/html/adonetanchor.asp