MIT CD!

.NET Everywhere: Gewinnen Sie eine Teilnahme an der BASTA! 2008 ► 20

9.2008 | Deutschland 8,50€ | Österreich 9,60 € | Schweiz 16,50 sFr | www.dotnet-magazin.de S&S

dot.net magazin

dot.net magazin

.NET, Visual Studio & More

SQL Server Security

Was Entwickler wissen müssen ► 63

Traue keinen Daten

Validierung von Mining-Modellen ► 73

Office Open XML

Dokumentenverwaltung mit ASP.NET ► 85

SharePoint und Access im Team

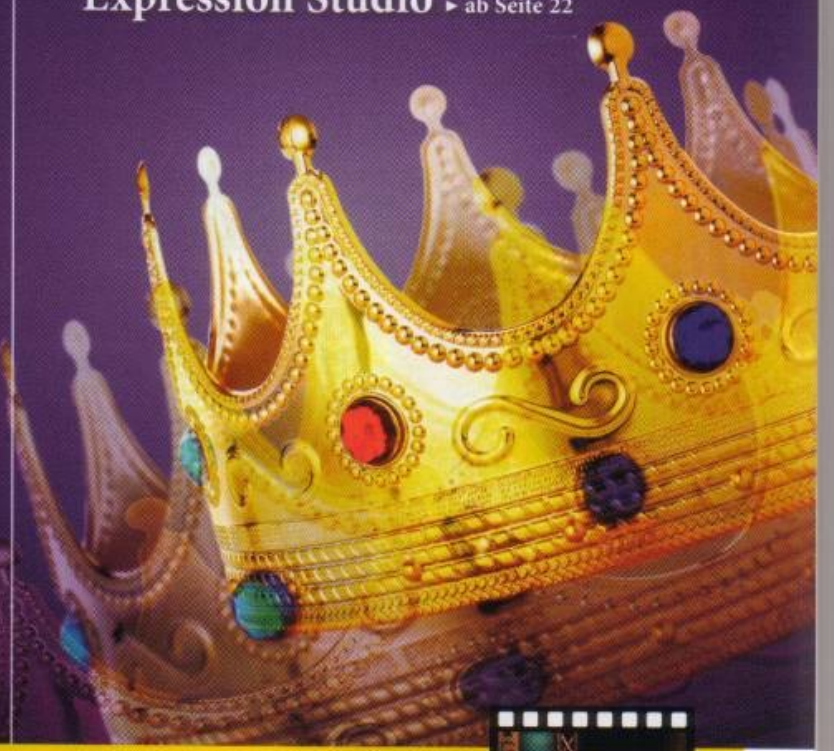
Auf gute Zusammenarbeit ► 102

LINQ to SAP

Geschäftsdaten im Griff? ► 108

RIA.NET

Neues von Silverlight und Expression Studio ► ab Seite 22



AUF CD

Testversionen:

- Telerik Sitefinity CMS 3.1 Community Edition
- GalleryMaker 3.6
- FileView Control 2008
- Virtual Print Engine v4.00 Community
- VS.Php 2.5.2
- DBMoto v6.2

Tools:

- Paint.NET v3.35
- DotNetNuke 4.8.4
- Pidgin 2.4.3
- DataBlock 1.4

...weitere Testversionen und Tools finden Sie auf Seite 3

Alle Beispiele und Quellcodes zu den Artikeln dieser Ausgabe

dot.net-TV

Ethical Hacking für Fortgeschrittene von der BASTA! Spring 2008

Datenträger enthält nur Lehr- oder Infoprogramme

4 195809 808509 09

LINQ to SAP

Mit LINQ auf Geschäftsdaten des SAP-R/3-Systems zugreifen

Mit der Einführung von .NET 3.5 und den Erweiterungen der Programmiersprachen C# und VB.NET steht Entwicklern die neue Datenabfrage-Technologie LINQ zur Verfügung. Neben Standardprovidern wie LINQ to SQL zur klassischen SQL-Datenbankabfrage ist die Entwicklung eigener LINQ-Provider möglich. Dieser Artikel zeigt, wie Daten im Geschäftsumfeld aus einem SAP-R/3-System mithilfe eines speziellen Providers abgefragt werden können.

von Jürgen Bäurle

Nahezu in jeder Applikation wird heutzutage auf die verschiedensten Datenbestände zugegriffen. Nicht selten stammen die Daten aus zahlreichen Datenquellen, wie z.B. SQL-Datenbanken, XML-Dateien oder Auflistungen (engl. *Collections*). Das .NET Framework bietet für all diese Quellen einen Mechanismus und ein API für die Datenabfrage an, jedoch unterscheiden sich die Zugriffstechnologien stark. Mit Bereitstellung von LINQ (Language Integrated Query) beschreitet Microsoft einen neuen Weg hin zur standardisierten Datenabfrage durch eine universelle Sprachsyntax. Die eingeführten Spracherweiterungen ähneln in ihrer Syntax SQL-Abfragebefehlen. Das nachfolgende Beispiel zeigt eine LINQ-Abfrage über den Dateninhalt einer Auflistung:

```
string[] names = {"Jürgen", "Karin", "Frank", "Nicole"}

var res = from n in names where n.Contains("i") select n;

foreach(var name in res)
        Console.WriteLine(name);
```

Die Abfrage selektiert alle Namen im Array *name s*, die den Buchstaben *i* enthalten und gibt diese anschließend auf der Konsole aus (Ausgabe: Karin, Nicole). Die Ähnlichkeit der Spracherweiterungen mit SQL ist von Microsoft bewusst gewollt, da Entwickler häufig relationale Datenbanken mit SQL-Sprachdialekten abfragen und deren Syntax kennen. Die Hoffnung, dass dadurch LINQ schneller als universelle Datenabfragetechnologie angenommen wird, ist nicht unbegründet. Bei genauer Betrachtung fällt auf, dass entgegen der SQL-Syntax das Schlüsselwort *from* vor *select* steht. Notwendig wurde diese Syntaxanpassung, um den Einsatz von IntelliSense in Visual Studio innerhalb der LINQ-Abfrage zu ermöglichen. Da der *from*-Teil vor *where* und *select* steht, erlangt Visual Studio am Anfang der LINQ-Abfrage die Information, welcher Datentyp in den anderen Teilen verwendet wird und kann die zur Verfügung stehenden Eigenschaften zur Auswahl im IntelliSense bereitstellen. Diese Vorgehensweise erleichtert den Umgang mit LINQ deutlich.

Um LINQ zu ermöglichen, mussten einige Spracherweiterungen in C# und VB.NET vollzogen werden, die nicht auf den ersten Blick ins Auge springen, welche aber dem Entwickler nicht nur im Zusammenhang mit LINQ von Nutzen sind. Das sind u.a. Lambda-Ausdrücke, die eine vereinfachte Schreibweise für Delegates bereitstellen, oder so genannte Extension-Methoden, die bestehende Klassendefinitionen wie String oder Integer um zusätzliche Instanzmethoden erweitern.

LINQ-Provider

Zwar werden die meisten Anwender im Zusammenhang mit LINQ weiterhin

kurz & bündig

Inhalt

Der Einsatz der LINQ-Technologie zur Abfrage von Daten aus einem SAP-System

Zusammenfassung

Mit dem auf SAP-R/3-Systeme spezialisierten Provider LINQ to SAP können Daten effizient in einem SAP-System abgefragt und weiterverarbeitet werden

SQL-Datenbanken abfragen und LINQ wurde für solche Abfragen optimiert, dennoch lassen sich mit dieser Technologie beliebige Datenquellen „anzapfen", solange ein LINQ-Provider zur Verfügung steht. Für oft eingesetzte Datenquellen stellt Microsoft Standardprovider bereit: LINQ to SQL für SQL-Datenbanken, LINQ to XML für XML-Dateien, LINQ to Objects für Auflistungen und LINQ to DataSet für *DataSet*-Instanzen. Neben den Standardprovidern können eigene Provider entwickelt werden. Beispiele wie LINQ to EXCEL, LINQ to Amazon oder LINQ to LDAP zeigen die Einsatzmöglichkeiten und die verschiedenen Typen an Datenquellen auf, die über LINQ angesprochen werden können. Eine weiterführende Einführung in LINQ ist in diesem Artikel nicht möglich – Ausgabe 7/8.07 des *dot.net magazin* hat sich dem Thema LINQ bereits ausführlich angenommen.

In diesem Artikel wird der Provider LINQ to SAP der Firma *Theobald Software* vorgestellt [1]. Mit diesem SAP R/3-Provider und dem zugehörigen Visual Studio Designer ist es möglich, SAP-Objekte in die eigene .NET-Anwendung einzubinden und abzufragen.

SAP-Objekte

An dieser Stelle soll ein kleiner Exkurs eine kurze Einführung und Definition der wichtigsten Objekte in SAP bringen. Mit LINQ to SAP kann für die folgenden SAP-Objekte automatisch Programmcode zur Datenabfrage mit LINQ erzeugt werden: Funktionsbausteine, Tabellen, BW-Cubes und Queries.

Funktionsbausteine sind Programmroutinen, die systemweit in einer SAP-Umgebung bekannt sind und von SAP-Programmen (ABAP) eingesetzt werden. Im Vergleich mit .NET stellen die Bausteine eine große, global verfügbare Methodenbibliothek im SAP-System dar. Abbildung 1 zeigt den Funktionsbaustein *CSAP_MAT_BOM_READ* in der SAP-eigenen Entwicklungsumgebung ABAP-Workbench. Dieser Baustein ermöglicht die Abfrage von Detailinformationen zu einer Materialstückliste (siehe Abschnitt Funktionsbausteine). In sog. Business-APIs (BAPI) werden alle Funktionsbausteine zusammengefasst, die betriebswirtschaftliche Standardvorgänge abbilden. Verwaltet werden BAPIs über das SAP Business Object Repository (BOR).

Des Weiteren lassen sich über LINQ to SAP so genannte BW-Cubes abfragen, auch als Business Warehouse Cubes oder OLAP-Würfel bezeichnet. Die Daten werden in mehrdimensionalen Würfeln (engl. Cubes) angeordnet.

Zuletzt ist der Zugriff auf Queries (SAP Query) über den Provider möglich. SAP stellt mit der SAP Query und der InfoSet Query Werkzeuge zur Verfügung, mit denen man auf einfache Weise optimierte Auswertungen erstellen kann. Dabei wird vom Anwender kein spezielles SAP-Wissen in der SAP-Programmiersprache ABAP vorausgesetzt.

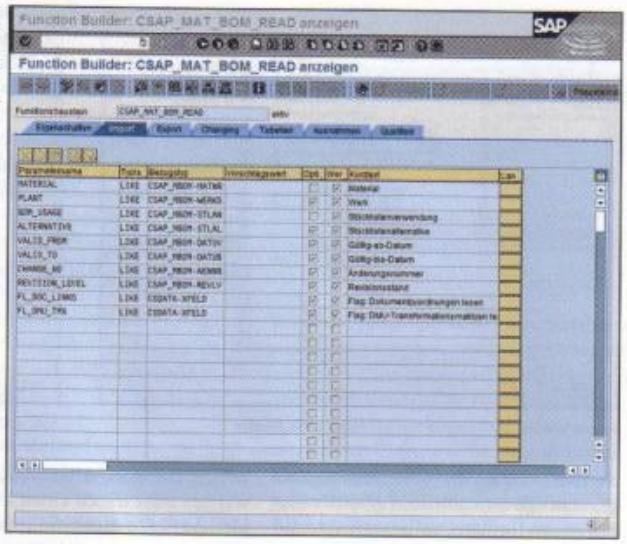


Abb. 1: Funktionsbaustein CSAP_MAT_BOM_READ in der SAP-Workbench

ERPConnect.net und LINQ to SAP

Für den Einsatz von LINQ to SAP muss zunächst die .NET-Bibliothek ERPConnect.net von Theobald Software installiert werden. Der Hersteller bietet auf seiner Homepage eine Testversion kostenlos zum Download an. ERPConnect .net bietet Basisfunktionalität zum Datenaustausch zwischen .NET und SAP R/3-Systemen an.

LINQ to SAP ist eine Erweiterung (Add-on) von ERPConnect.net und muss nach Installation des Basisprodukts nachträglich über eine separate Setup-Routine aktiviert werden. Der Provider besteht aus einem Visual Studio 2008 Designer und den zugehörigen Basisklassen, die im Namespace *ERPConnect.Linq* gebündelt wurden. Das Setup fügt Visual Studio 2008 einen neuen *ProjectItem*-Typ mit der Dateiendung .erp hinzu und verbindet diese mit dem LINQ to SAP Designer (Abbildung 2).

LINQ to SAP Designer

Mit dem Öffnen einer *erp*-Datei wird der LINQ to SAP Designer in Visual Studio gestartet. Der Designer bietet dem Anwender die Option, sämtliche vom Provi-

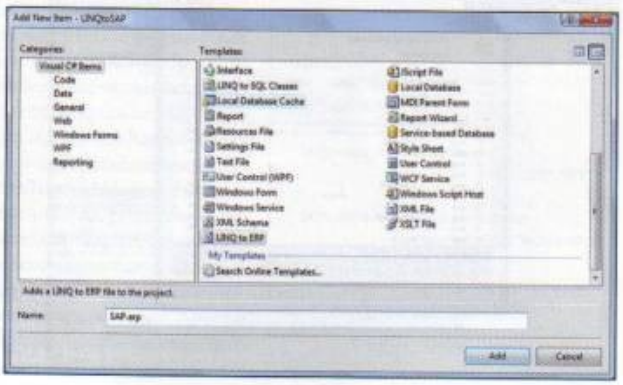


Abb. 2: Hinzufügen einer LINQ to SAP-Datei (.erp)

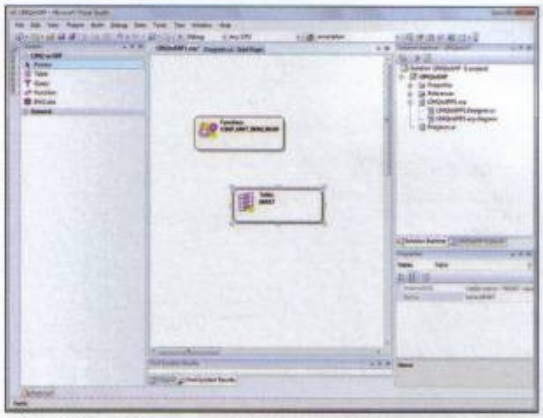


Abb. 3: LINQ to SAP Designer mit definierten SAP-Objekten

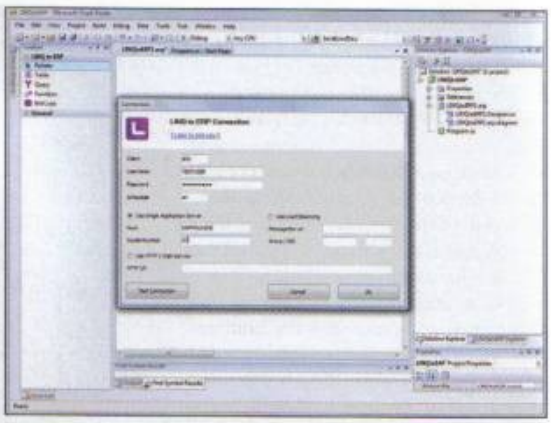


Abb. 4: Dialog zur Angabe von SAP-Verbindungsdaten

der unterstützten SAP-Objekte in den Programmcode automatisiert einzubinden. Für die im Designer definierten Objekte (Abbildung 3) wie Funktionsbausteine oder Tabellen, wird beim Speichern automatisch eine von der Klasse *ERPConnect.Linq.ERPDataContext* abgeleitete Kontextklasse generiert, die Methoden und zusätzliche Klassen für die Objekte beinhaltet. Der durch den Designer erzeugte Code wird in einer zur *.erp*-Datei zugehörigen Datei mit der Endung *.Designer.cs* (*.vb*) gespeichert. Auf diese Art und Weise lässt sich ohne weitere Programmierung über die Kontextklasse auf einzelne SAP-Objekte in .NET-Anwendungen zugreifen und weiterverarbeiten.

Auf der linken Seite in Abbildung 3 zeigt die Toolbox mit der Bezeichnung LINQ to ERP (interne Bezeichnung von LINQ to SAP) alle durch den Provider unterstützten SAP-Objekte an: *Table, Query, Function* sowie *BW-Cube*. Diese Objekte können einfach mit der Maus auf die Designer-Arbeitsfläche gezogen werden oder durch einen Doppelklick hinzugefügt werden. Sollten beim Hinzufügen des ersten Objektes noch keine Verbindungsdaten für den Zugriff auf ein SAP-System existieren, werden diese Daten zunächst im so genannten Connection-Dialog abgefragt (Abbildung 4). Zu diesem Zeitpunkt muss eine Verbindung zum SAP-System vorhanden sein. Für eine genaue Beschreibung der notwendigen Angaben für einen Verbindungsaufbau muss auf das Produkthandbuch des Herstellers verwiesen werden. Je nach Auswahl des SAP-Objekts wird ein spezieller Dialog dargestellt. Abbildung 5 zeigt den Dialog zur Verwaltung des Funktionsbausteins *CSAP_MAT_BOM_READ*.

Wichtig ist, dass das Auswählen und Hinzufügen neuer SAP-Objekte zum Designer eine Verbindung zum SAP-System erfordert. Die Objekt-Metadaten werden allerdings nach dem Einfügen für zukünftige schnelle Zugriffe gespeichert, ohne dass eine Verbindung zu SAP vorhanden sein muss.

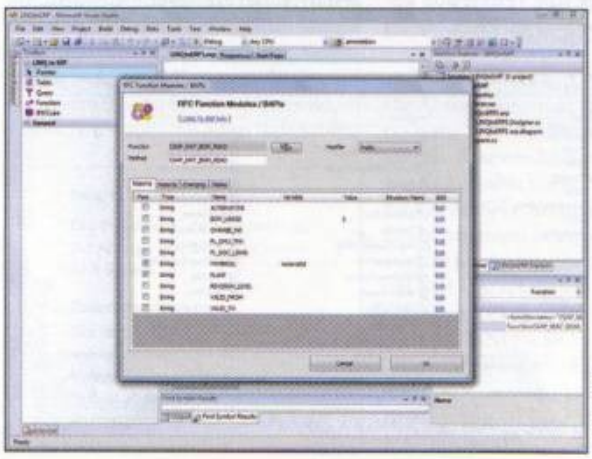


Abb. 5: Dialog zur Bearbeitung des Funktionsbausteins CSAP_MAT_BOM_READ

Funktionsbausteine

Dieser Abschnitt zeigt den Zugriff auf den Funktionsbaustein CSAP_MAT_BOM_READ mit LINQ to SAP. Der Baustein liefert für ein Material, welches aus mehreren separaten Materialen zusammengesetzt ist, dessen Stück- oder Komponentenliste zurück. Das hier beschriebene Beispiel liest die Materialstückliste für ein Fahrrad mit der Materialnummer „M1973100" aus dem SAP-System und gibt diese Liste anschließend auf der Konsole aus. Das Fahrrad, ein Mountainbike, besteht aus den Komponenten Rahmen, Sattel, Rad und Lenker.

Zunächst muss einem neuen oder bestehenden Visual Studio 2008-Projekt eine LINQ to SAP-Datei über den Menüpunkt PROJEKT | HINZUFÜGEN... hinzugefügt werden. Beim Öffnen der neu erstellten *.erp*-Datei wird der LINQ to SAP Designer gestartet. Durch einen Doppelklick auf das SAP-Objekt *Function* in der Toolbox LINQ TO ERP kann ein Funktionsbaustein dem Designer hinzugefügt werden. Es erscheint zunächst ein Suchdialog, über den ein Baustein gesucht und selektiert werden kann. Dazu wird im Textfeld der Bausteinname angegeben oder mit „*" ein Suchmuster definiert (Abbildung 6).

Nach erfolgreicher Selektion wird dem Anwender ein Dialog mit allen den Funktionsbaustein beschreibenden Attributen sowie zusätzlichen Informationen, die zum Erzeugen von Methoden sowie weiteren Objekten für die Kontextklasse notwendig sind, angezeigt (Abbildung 5). Im oberen Abschnitt des Dialogs werden

der Name des Bausteins, der zu erzeugende Methodenname sowie deren Zugriffsmodifizierer angezeigt.

Für jeden über den LINQ to SAP Designer eingebundenen Baustein wird eine Klassenmethode für die Kontextklasse erzeugt. Standardmäßig benennt der Designer die Kontextklasse *SAPContext*, ähnlich der *DataContext*-Klasse bei LINQ to SQL. Der Name der Kontextklasse, der zu erzeugende Namespace, die SAP-Verbindungsdaten und die Angabe, ob die Verbindungseinstellungen in der Kontextklasse gespeichert werden sollen, können über das Eigenschaftsfenster des Designers festgelegt werden. Definiert der Anwender für den Funktionsbausstein *CSAP_MAT_BOM_READ* beispielsweise den Methodenname *GetMaterialPartList*, so erzeugt der Designer eine Methode mit diesem Namen, den im Dialog selektierten Parametern als Signatur und weitere Strukturen (z.B. LINQ-fähige Auflistungen) in der Kontextklasse.

Im unteren Bereich des Dialogs werden Registerkarten mit den SAP-typischen Parametern wie *Export, Import, Tables* und *Changing* angezeigt. Der Anwender kann wählen, welche Parameter der final erzeugten Methode übergeben werden und wie die entsprechenden Bezeichner benannt sein sollen. Zusätzlich wird definiert, ob und welchen Rückgabewert für eine Baustein-Methode erzeugt werden muss.

Innerhalb von SAP stellen die Export-Einträge die Übergabeparameter, die Import-Einträge die Rückgabewerte für einen Funktionsbaustein dar. Changing-Einträge sind Parameter, die in .NET mit Referenzparameter vergleichbar sind, also Methodenparameter, dessen Werte an den Baustein übergeben und von diesem verändert zurückgegeben werden können. Tables-Definitionen erlauben den Austausch und die Modifikation ganzer Tabellen und Listen mit dem Funktionsbaustein.

LINQ to SAP erlaubt dem Anwender für die entsprechenden Parameter fixe Werte festzulegen oder diese als Methodenparameter zu definieren. Jeweils ein *Import*- oder *Tables*-Eintrag kann als Rückgabewerte für die zu erzeugende Methode festgelegt werden (Abbildung 5). Im Beispiel der Materialstückliste wird die Materialnummer *MATERIAL* und das Werk *PLANT* in der Registerkarte Export als Übergabeparameter (Checkbox in Spalte *Pass* markiert) und die Tabelle *T_STPO* in der Registerkarte Tables als Rückgabewert (Checkbox in Spalte *Return* markiert) festgelegt. Die Tabelle enthält die Komponentenliste für die übergebene Materialnummer. Für die LINQ-fähigen und vom Designer zusätzlich erzeugten Auflistungen kann der Anwender, wie im Fall der Tabelle *T_STPO*, selbsterklärende Bezeichner (z.B. *partListTable* als Tabellenname) wählen. Für alle anderen Methodenparameter können ebenso sprachkompatible und besser verständliche Bezeichner angegeben werden. Als fixen Wert für den Import-Eintrag *BOM_USAGE* wird der Wert 3 für die Verwendung gesetzt. Mit diesen Angaben erzeugt der Designer beim Speichern der *.erp*-Datei die folgende Methode in der Kontextklasse:

```
public PartListTable GetMaterialPartList
                    (string materialId, string PLANT)
```

Listing 1 zeigt, wie die durch den LINQ to SAP Designer automatisch generierte Kontextklasse *SAPContext* und die Methode *GetMaterialPartList* in einer .NET-Anwendung verwendet werden können. Zuerst wird eine Instanz der Kontextklasse mit dem Namen *sap* erzeugt. Den Konstruktoren von *SAPContext* werden die Verbindungsdaten zu einem SAP-System übergeben. Anschließend können die definierten SAP-Objekte als Methoden aufgerufen werden. Sämtliche SAP-Zugriffslogik, basierend auf ERPConnect.net, wird in der Kontextklasse verborgen.

Tabellen

Die Vorgehensweise für das Hinzufügen von SAP-Tabellen zum Designer ist prinzipiell dieselbe wie bei Funktionsbausteinen. Zunächst wird nach der Auswahl von *Table* in der Toolbox (durch Doppelklick) über den Suchdialog ein Tabellenobjekt zur Weiterverarbeitung ermittelt. Der LINQ to SAP Designer liest daraufhin die Metadaten der Tabelle ein und zeigt den Table-Dialog an. Im oberen Dialog-Bereich muss ein Name für die zu erzeugende Tabellenklasse sowie der Zugriffsmodifizierer angegeben werden. Als Standardname wird der oft kryptische SAP-Name vorgeschlagen. Zusätzlich kann noch ein Funktionsbaustein für das Einlesen der Tabellendaten im Textfeld *Custom Function* definiert werden. Diese Option wird nur in wenigen Fällen benötigt, auf die an dieser Stelle nicht weiter eingegangen wird.

Die Anzeige im unteren Bereich des Dialogs listet alle Tabellenspalten auf. Über die Spalte *Out* lassen sich die Tabellenspalten angeben, für die der Designer Eigenschaften in der Klassendefinition generiert und Teil der Ergebnismenge von Abfragen sein sollen. Die Spalte *Name* zeigt den ursprünglichen SAP-Namen einer Tabellenspalte an. Die Spalte *Member* ermöglicht einen freien Bezeichner für die zugehörige Klassen-Eigenschaft festzulegen. Eine Vorschauansicht der Ergebnismenge wird bei Klick auf *Preview* dargestellt. Dabei ist eine Verbindung zum SAP-System notwendig.

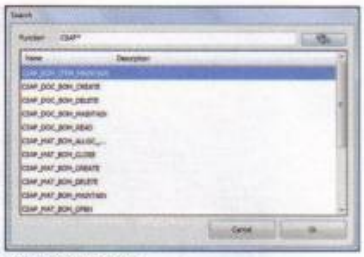


Abb. 6: Such-Dialog

Abbildung 7 zeigt den Table-Dialog mit der Definition des SAP-Tabellenobjekts *MAKT* (MAKT steht für Materialkurztexte). Als Klassennamen wurde *MaterialShortDesc* festgelegt. Der Designer generiert für die Kontextklasse eine *MaterialShortDesc*-Klasse und eine LINQ-fähige Eigenschaft mit dem Namen *MaterialShortDescList* vom Typ *ERPTable<MaterialShortDesc>*.

Listing 2 zeigt den Zugriff auf die SAP-Tabelle. Nach Erzeugung einer Objektinstanz der SAPContext-Kontextklasse mit den SAP-Zugangsdaten wird zunächst die Konsole als Logausgabe gesetzt (siehe Abschnitt Logging) und anschließend eine typische LINQ-Abfrage definiert. Die Abfrage selektiert aus der Ergebnismenge *MaterialShortDescList* genau die Datensätze (vom Typ *MaterialShortDesc*), deren Materialnummer mit

Listing 1

```
class Program
{
  static void Main()
  {
    SAPContext sap = new SAPContext
                        ("TESTUSER", "XYZ");

    SAPContext.PartListTable partList = sap.
                GetMaterialPartList("M1973100");

    foreach (var item in partList)
      Console.WriteLine(item.COMPONENT);
  }
}
```

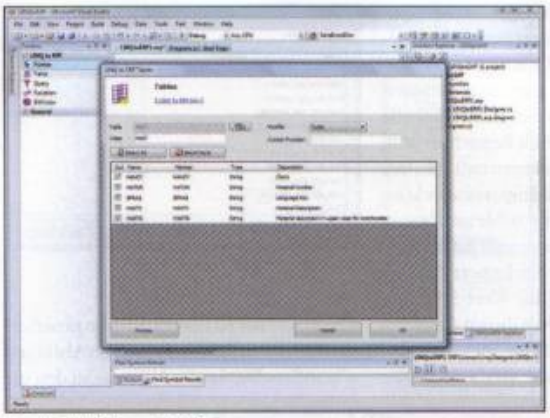


Abb. 7: Table-Dialog mit MAKT

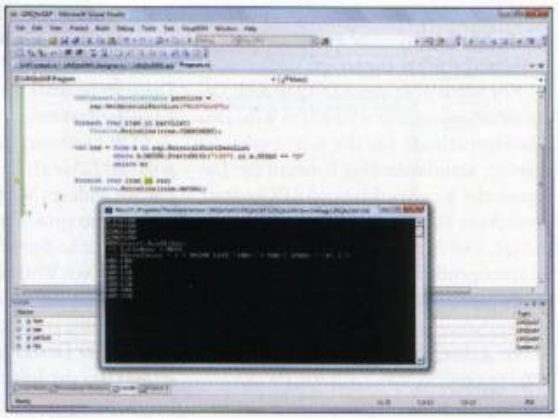


Abb. 8: Logging mit Kontextklasse

dem Wert 100 beginnt und für die die Kurztexte in Deutsch vorliegen. Die Ausgabe erfolgt auf der Konsole.

Logging

Die Kontextklasse bietet, ähnlich LINQ to SQL, die Eigenschaft *Log* zum Protokollieren der durch das LINQ-Konstrukt erzeugten Abfrage. Die Ausgabe ist im Fall von LINQ to SAP auf Tabellenobjekte und den *where*-Teil der SAP-internen Abfrage beschränkt. Die Eigenschaft Log ist vom Typ *TextWriter*. Damit eignet sich die Konsolenausgabe als Log-Ziel. *Console.Out* kann der Log-Eigenschaft zugewiesen werden, um sich Logeinträge auf der Konsole anzeigen zu lassen. Das ist äußerst hilfreich um ein Gefühl dafür zu bekommen, wie eine LINQ-Abfrage in eine Abfrage des Zielsystems „übersetzt" wird. Abbildung 8 zeigt die Ausgabe des Loggers.

Listing 2

```
class Program
{
  static void Main()
  {
    SAPContext sap = new SAPContext
                                    ("TESTUSER", "XYZ");
    sap.Log = Console.Out;

    var res = from m in sap.MaterialShortDescList
        where m.MATNR.StartsWith("100") && m.
                                          SPRAS == "D"
        select m;

    foreach (var item in res)
      Console.WriteLine(item.MATNR);
  }
}
```

Listing 3

```
using System;

namespace LINQtoSAP
{
  partial class SAPContext
  {
    public PartListTable GetMaterialPartListForMain
                                  Plant(string materialId)
    {
      return GetMaterialPartList(materialId, "1000");
    }
  }
}
```

Erweiterung der Kontextklasse

Die durch den LINQ to SAP Designer erzeugte Kontextklasse (SAPContext) wurde als „partial" markiert. Partielle Klassen in C# und VB.NET ermöglichen es dem Anwender, Klassen in separaten Dateien zu definieren und damit zu erweitern. Der Einsatz partieller Klassendefinitionen ist speziell im Kontext automatisch generierter Quellcodedateien wichtig, damit eigene Erweiterungen nicht überschrieben werden. Listing 3 zeigt ein Beispiel, wie die SAPContext-Klasse durch das Hinzufügen einer weiteren partiellen Klassendefinition erweitert wurde. Der Compiler fügt beim Kompilieren die Dateien *LINQtoERP1.Designer.cs* und *SAPContext.cs* (siehe Abbildungen) intern zusammen und übersetzt die so neu entstandene Quellcodedatei als Ganzes.

Die Datei *SAPContext.cs* erweitert die durch den LINQ to SAP Designer definierte Methode *GetMaterialPartList* um eine eigene Methode *GetMaterialPartListForMainPlant* (Listing 3), die intern die Methode *GetMaterialPartList* mit festgelegten Werk aufruft.

Fazit

LINQ ist eine faszinierende Technologie und durch die Erweiterbarkeit bzgl. der Unterstützung eigener Provider wird LINQ zukünftig ein alltäglich genutztes Werkzeug eines jeden .NET-Entwicklers werden. Der Einsatz von LINQ to SAP ist ein solcher Provider. Er erleichtert dem Anwender den Zugriff auf das SAP-System und die Nutzung seiner Objekte in einer .NET-Anwendung enorm. Die SAP-Logik für den Zugriff auf die SAP-Objekte ist in der Kontextklasse versteckt und ermöglicht SAP-unerfahrenen .NET-Entwicklern die schnelle Datenabfrage. Der Visual Studio Designer ist klar strukturiert, und die erzeugten Kontextklassen sind einfach zu verwenden. Rückgabewerte einzelner Methoden vom Typ „Auflistung" sind LINQ-fähig und können gut weiterverarbeitet werden. Insgesamt betrachtet stellt LINQ to SAP einen sehr sinnvoll einsetzbaren LINQ-Provider für .NET/SAP-Integrationsaufgaben bereit. ●

Jürgen Bäurle ist seit vielen Jahren selbständig als Software-Architekt und Entwickler tätig. Sein Schwerpunkt liegt in der .NET-Entwicklung. Sie erreichen ihn per E-Mail unter jbaurle@parago.de. Zusätzliche Informationen erhalten Sie auf der Webseite www.parago.de/jbaurle.

Links & Literatur

1. ERPConnect.net (LINQ to SAP): www.theobald-software.com
2. Joseph Rattz Jr.: Pro LINQ, apress 2007