LINQ to SAP

Jürgen Bäurle

July 2008

Parago Media GmbH & Co. KG

Introduction

A lot has been written about Microsoft's new data access technology LINQ (Language Integrated Query) since the first preview version has been published. But there are still some interesting aspects to LINQ and its extensibility. This article will introduce a new provider called LINQ to SAP which offers an easy way to retrieve business data from SAP/R3 systems within a .NET application.

With the introduction of the .NET framework 3.5 and the programming language extensions for C# and VB.NET Microsoft began to redefine the way developers would implement data access. Nearly all applications today are querying data from different data sources. Mainly those data sources are SQL databases, XML files or in-memory collections. LINQ offers an universal and standardized approach to access all those data sources using special data providers. The LINQ language syntax strongly resembles that of SQL. The following sample shows how to query data from a string array with LINQ:

```
string[] names = {"John", "Patrick", "Bill", "Tom"}
var res = from n in names where n.Contains("o") select n;
foreach(var name in res)
   Console.WriteLine(name);
```

This simple LINQ query based on an in-memory collection (array) selects all items in the array "names" that contain the letter "o". Console output: "John, Tom". A LINQ introduction is beyond the scope of this article. A very good introduction article can be found on CodeProject.com.

LINQ Providers

The .NET framework comes with built-in providers for collections and lists (LINQ to Objects), for Microsoft SQL Server databases (LINQ to SQL), for XML files (LINQ to XML) and finally for DataSet object instances (LINQ to DataSet). Beside the standard providers, developers can extend LINQ by creating custom providers to support special data sources. LINQ to LDAP or LINQ to Amazon are examples of such custom providers.

To write a custom LINQ provider one must basically implement two interfaces: IQueryable and IQueryProvider. These interfaces make objects queryable in LINQ expressions. Developing a LINQ provider can be a very complex task, but there a quite some good blog entries on the net explaining the steps in detail.

This article will introduce a new provider called LINQ to SAP from Theobald Software which provides developers with a simple way to access SAP/R3 systems and their data objects. The software also provides a Visual Studio 2008 Designer to define SAP objects interactively and integrate them in .NET applications.

SAP Background

This section will give you a short explanation and background of SAP objects that are queryable by LINQ to SAP. The most important objects are Function Modules, Tables, BW Cubes and Queries.

A Function Module is basically similar to a normal procedure in conventional programming languages. Function Modules are written in ABAP, the SAP programming language, and are accessible from any other programs within a SAP/R3 system. They accept import and export parameters as well as other kind of special parameters. The image below shows an example of a Function Module named BAPI_EQUI_GETLIST within the SAP Workbench:

Classification						
Function Group	ITOB_BAP	I_EQ	PM Equipme	ent: BAPIs		
Short Text BAPI PM: Selection of			Equipment List			
Processing Type /			General Data			
O Normal Function M	odule		Person Responsible	SAP	SAP	
Remote-Enabled N	lodule		Last Changed By	SAP		
O Update Module			Changed on	30.03.2	2006	
Start immed.			Package ITOB			
O Immediate Start, No Restart			Program Name	Name SAPLITOB_BAPI_E		
O Start Delayed			INCLUDE Name	LITOB_E	BAPI_EQU11	
O Coll.run			Original Language	DE		
			Released on	03.02.2	2000	
			Edit Lock			
			Global			

Figure 1: Function Module (SAP Workbench)

In addition BAPIs (Business-API) are special Function Modules that are organized within the SAP Business Object Repository. LINQ to SAP also allows access to SAP tables. Those are basically straightforward relational database tables. Furthermore the LINQ to SAP Designer allows developers to define and access BW Cubes (Business Cube Cubes) and Queries (SAP Query). BW Cubes are also known as OLAP Cubes. Data are organized in a multidimensional way within a cube. SAP Queries work just like other queries. To indentify a SAP Query uniquely there are three pieces of information necessary: user area (global or local); user group and the name of the query. With the concept of Queries SAP provides a very simple to generate reports, without the need of knowing the SAP ABAP programming language.

Visual Studio 2008 Designer for LINQ to SAP

In order to use LINQ to SAP and the associated Visual Studio Designer, the .NET library ERPConnect.net from Theobald Software must be installed first. This software is the basic building block between .NET and a SAP/R3 system and provides an easy API to exchange data between the two systems. The company offers a free trial version to download. After installing ERPConnect.net, LINQ to SAP must be installed separately using a setup program (see manual). The provider and the designer are actually extensions to the ERPConnect.net library. The LINQ to SAP provider itself consists of the Visual Studio 2008 Designer and additional class libraries that are bundled within the namespace ERPConnect.Linq.

The setup adds a new ProjectItem type to Visual Studio 2008 with the file extension .erp and is linking it with the designer. Double-clicking the .erp-file will open the LINQ to SAP Designer. The designer supports application developers with the option to automatically generate source code to integrate with SAP objects. For all defined SAP objects in an .erp file, the provider will create a context class which inherits from the ERPDataContext base class. The generated context class contains methods and sub-classes that represent the defined SAP objects. Beside the .erp file, LINQ to SAP designer will save the associated and automatically generated source code in a file with the extension .Designer.cs.

Service-based Database	*
 Settings File Style Sheet Text File User Control User Control (WPF) WCF Service Windows Form Windows Script Host Windows Service XML File XML Schema XSLT File LINQ to ERP 	1
	 Style Sheet Text File User Control User Control (WPF) WCF Service Windows Form Windows Script Host Windows Service XML File XML Schema XSLT File LINQ to ERP

Figure 2: Add new project item



Figure 3: SAP objects in LINQ to SAP Designer

File Edit View Project Build Debug Profiler Data Tools Test Window Help Image: Connection Pointer Connection Image: Connection Client BWCut Client Busername TESTUSER Password Busername Debug Client Busername Debug Use Single Application Server Use Load Balancing	🥠 LINQtoSAP - N	/licrosoft Visual Studio			
Toolbox - 1 × LINQ to Connection Pointer Image Table Image Query Image Semeral Client BWCut Client Username TESTUSER Password	File Edit Vie	w Project Build	Debug Profiler Data To	ols Test Window Help Debug + Any CPU	- D80 - 10
LINQ to Connection Pointer Query Query Query Function Eline t BWCut Client 800 Username TESTUSER Password •••••• Language de Use Single Application Server Use Load Balancing	Toolbox	× LIN	QtoERP1.erp Program.cs St	art Page - × Solution Expl	orer - LINQtoSAP + 4 ×
Language de Solution Server Use Load Balancing	Center Explorer Conter C	Connection	NQ to ERP Connection		as m
Host SAPMACHINE MessageServer SystemNumber 07 Group / SID er.LINQtc +		Use Single Applica Host SystemNumber Use HTTP / Web S	de iton Server SAPMACHINE 07	 Use Load Balancing MessageServer Group / SID 	əlorer • 4 × er.LINQtc •
HTTP UH Test Connection Cancel Ok Show outout from: Output 2 Find Symbol Results Firor List	Error List	HTTP Url Test Connection Show	outout from: tput A Find Symbol Results	Cancel	Ok .

Figure 4: Connection dialog in LINQ to SAP Designer

Function Modules

This section shows how to access and obtain data using the function module BAPI_EQUI_GETLIST by creating a LINQ to SAP object. The module is returning an equipment list for pre-defined plants. First of all one must add a new LINQ to SAP file (.erp) to a new or existing Visual Studio 2008 project. By opening the .erp file the LINQ to SAP Designer will start. By double-clicking on the Function item in the toolbox of Visual Studio will add a new SAP object Function Module. In the next step the object search dialog opens and the developer can search for function modules.

Unction DAP1_EQUI	00
Name	Description
BAPI_EQUIPMENT_SAVERE	BAPI for Mass Maintenance of Equipment
BAPI_EQUI_CHANGE	PM BAPI: Change Equipment
BAPI_EQUI_CREATE	PM BAPI: Create Equipment
BAPI_EQUI_CREATE_BY_R.	
BAPI_EQUI_DISMANTLE	PM BAPI: Install Equipment (Functional Location, Superior Equipment)
BAPI_EQUI_GETDETAIL	PM BAPI: Read Equipment
BAPI_EQUI_GETLIST	BAPI PM: Selection of Equipment List
BAPI_EQUI_GETSTATUS	PM BAPI: Read (System-/User-)Status Equi
BAPI_EQUI_INSTALL	PM BAPI: Install Equipment (Functional Location, Superior Equipment)

Figure 5: Search dialog in LINQ to SAP Designer

Once the selection is made, the LINQ to SAP Designer will show up the Function Module dialog box with all data, properties and parameter definitions of the selected module BAPI_EQUI_GETLIST. The user can now change the naming of the auto-generated method as well as all used parameters.

FC Function	Modules	/ BAPIs				
8 2	R	FC Function Mod	ules <mark>/ BAPI</mark> s			
Function Method	BAPI_ BAPI_	EQUI_GETLIST	Ø	Modifier [F	Public	
Return	Pass	Name	Variable	Table Name	e Structure Name	Edit
	17	DESCRIPT_RA				Edit
V		EQUIPMENT_LIST				Edit
		EQUIPMENT_RA				Edit
100	1	MAINTPLANT_RA				Edit
		PLANGROUP_RA				Edit
		PLANPLANT_RA				Edit
		SORTFIELD_RA				Edit
				Cancel	Ok	

Figure 6: Function Module dialog in LINQ to SAP Designer

For each function module the LINQ to SAP Designer will generate a context class method with all additional object classes and structures. If for instance the user defines a method name called GetEquipmentList for the function module BAPI_EQUI_GETLIST, the designer will generate a context class method with that name and the defined method signature. The user can also specify the parameters to exchange. The lower area of the dialog is displaying the SAP typical parameters, like IMPORT, EXPORT, CHANGING and TABLES parameters. LINQ to SAP allows to define default values for SAP parameters. Those parameters can also be used as parameters for the auto-generated context class method as well as for return values. The names for the parameters and the associated structures can also be renamed.

The method signature for the function module defined above looks like this:

public EquipmentTable GetEquipmentList(PlantTable plants)

The context class itself is named SAPContext by default. The context class name, the namespace, the connection settings as well as other flags can be defined in the properties window of the LINQ to SAP Designer. The following code shows how to use the context class SAPContext:

```
class Program
{
  static void Main()
  {
    SAPContext dc = new SAPContext("TESTUSER", "XYZ");
    SAPContext.PlantTable plants = new SAPContext.PlantTable();
    SAPContext.PlantStructure ps = plants.Rows.Add();
    ps.SIGN = "I";
    ps.OPTION = "EQ";
    ps.LOW = "3000";
    SAPContext.EquipmentTable equipList = dc.GetEquipmentList(plants);
    }
}
```

Tables

The procedure for adding a SAP Table is basically the same as for function modules (see above). After adding the SAP Table object from the toolbox in Visual Studio and finding the table with the search dialog, the Table dialog will show up:

		ables .INO To ERP Info]				
able lass	T001W T001W		Ø	Modifier Custom Function	Public 💌	
•	Select All	Select None				
Out	Name	Member	Type	Descript	tion	
	MANDI	MANDI	String	Client		
	WERKS	WERKS	String	Plant		
	NAME1	NAME1	String	Name		-
	BWKEY	BWKEY	String	Valuation	n Area	
	KUNNR	KUNNR	String	Custome	r number of plant	
	LIFNR	LIFNR	String	Vendor n	number of plant	
	FABKL	FABKL	String	Factory	calendar key	
	NAME2	NAME2	String	Name 2		
~	STRAS	STRAS	String	House nu	umber and street	
V	PFACH	PFACH	String	PO Box		
1	PSTLZ	PSTLZ	String	Postal Co	ode	
1	ORT01	ORT01	String	City		
2	EKORG	EKORG	String	Purchasir	ng Organization	
1	VKORG	VKORG	String	Sales org	ganization for intercompany b	illing
	Draviow				Canad	ok

Figure 7: Tables Module dialog in LINQ to SAP Designer

In the upper part of the table dialog the user must define the class name for table object for auto-generation. The default name is the name of the table. The lower part shows a data grid with all table fields and their definitions. For each field a class property name can be defined for the auto-generated context class code. The checkbox in the first column selects if the field will be part of the table class.

The figure above shows the definition of the SAP Table object T001W. This tables stores plant information. The class has not been changed, so the designer will create a C# class with the name T001W. In addition the context class will contain a property T001WList. The type of this property is ERPTable<T001W>, which is LINQ queryable data type.

The following code shows how to query the table T001W using the context class:

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

```
var res = from p in dc.T001WList
    where p.WERKS == "3000"
    select p;
foreach (var item in res)
    Console.WriteLine(item.NAME1);
}
```

SAP Context Class and Logging

To access objects using LINQ to SQL, the provider will generate a context class named DataContext. Accordingly LINQ to SAP also creates a context class called SAPContext. This class is defined as a partial class. A partial class is a type declaration that can be split across multiple source files and therefore allows developers to easily extend auto-generated classes like the context class of LINQ to SAP.

The code sample below shows how to add a partial class (file SAPContext.cs) which adds a new custom method GetEquipmentListForMainPlant to extend the context class generated by the LINQ to SAP designer. This new method calls internally the auto-generated method GetEquipmentList with a pre-defined parameter value. The C# compiler will internally merge the auto-generated LINQtoERP1.Designer.cs with the SAPContext.cs source file.

```
using System;
namespace LINQtoSAP
{
  partial class SAPContext
  {
    public EquipmentTable GetEquipmentListForMainPlant()
    {
       SAPContext.PlantTable plants = new SAPContext.PlantTable();
       SAPContext.PlantStructure ps = plants.Rows.Add();
       ps.SIGN = "I";
       ps.OPTION = "EQ";
       ps.LOW = "3000";
       return GetEquipmentList(plants);
    }
}
```

LINQ to SAP also provides the capability to log LINQ query translations. In order to log data the LOG property of the context class must be set with a TextWriter instance, e.g. the console output Console.Out. All LINQ to SAP does is a very rudimentary logging which is restricted to table objects. But it helps developers to get a feeling about what the translated where part looks like.

Summary

In overall LINQ to SAP is very simple but yet powerful LINQ data provider and Visual Studio 2008 Designer to use. You also get a feeling on how to develop against a SAP/R3 system using .NET. For more information about the product please check out the homepage of the vendor, http://www.theobald-software.com.

Contact Information

If you have any feedback or suggestions, please feel free to contact me:

Jürgen Bäurle http://www.parago.de