

How To Access SAP Business Data From Silverlight 4 Clients Using WCF RIA Services

Jürgen Baurle

November 2010

Parago Media GmbH & Co. KG

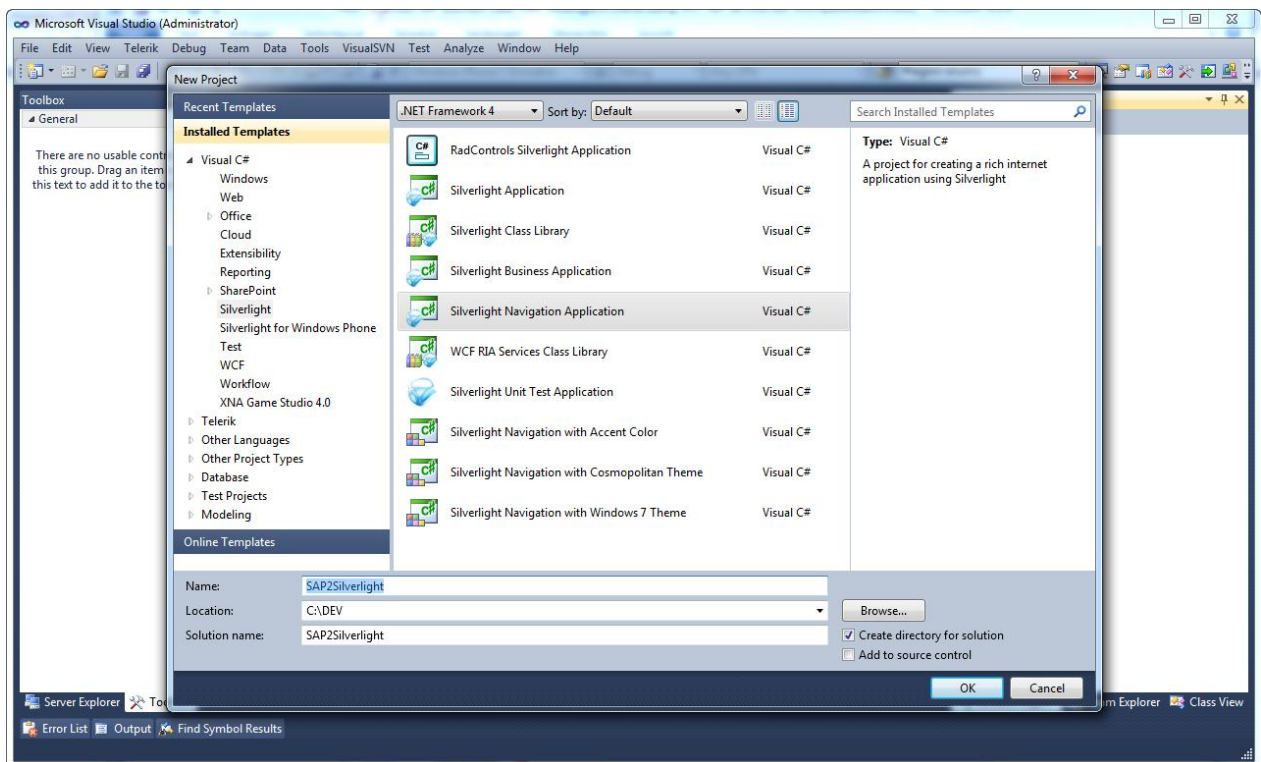
Introduction

The introduction of Microsoft's WCF RIA Services for Silverlight 4 simplified very much the development process of N-tier business applications using Silverlight and ASP.NET. By using this new technology we can also easy access and integrate SAP business data in Silverlight clients.

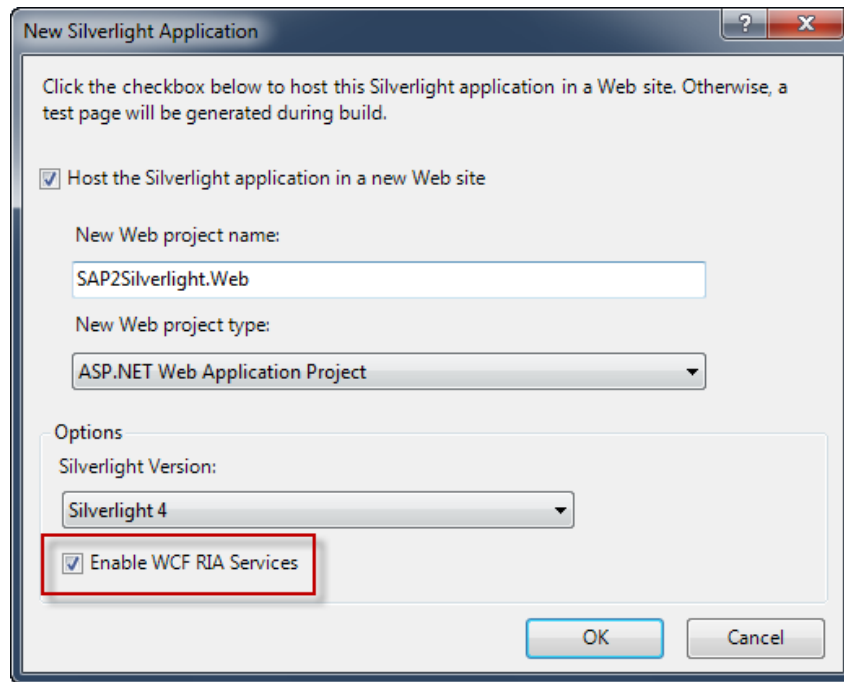
This article shows how to provide a SAP domain service as web service that will be consumed by a Silverlight client. The sample application will allow the user to query customer data. The service uses LINQ to SAP from Theobald Software to connect to a SAP R/3 system.

Project Setup

The first step in setting up a new Silverlight 4 project with WCF RIA Services is to create a solution using the Visual Studio template *Silverlight Navigation Application*:



Visual Studio 2010 is then asking you to create an additional web application, which hosts the Silverlight application. It's important to select the checkbox *Enable WCF RIA Services* (see screenshot below):

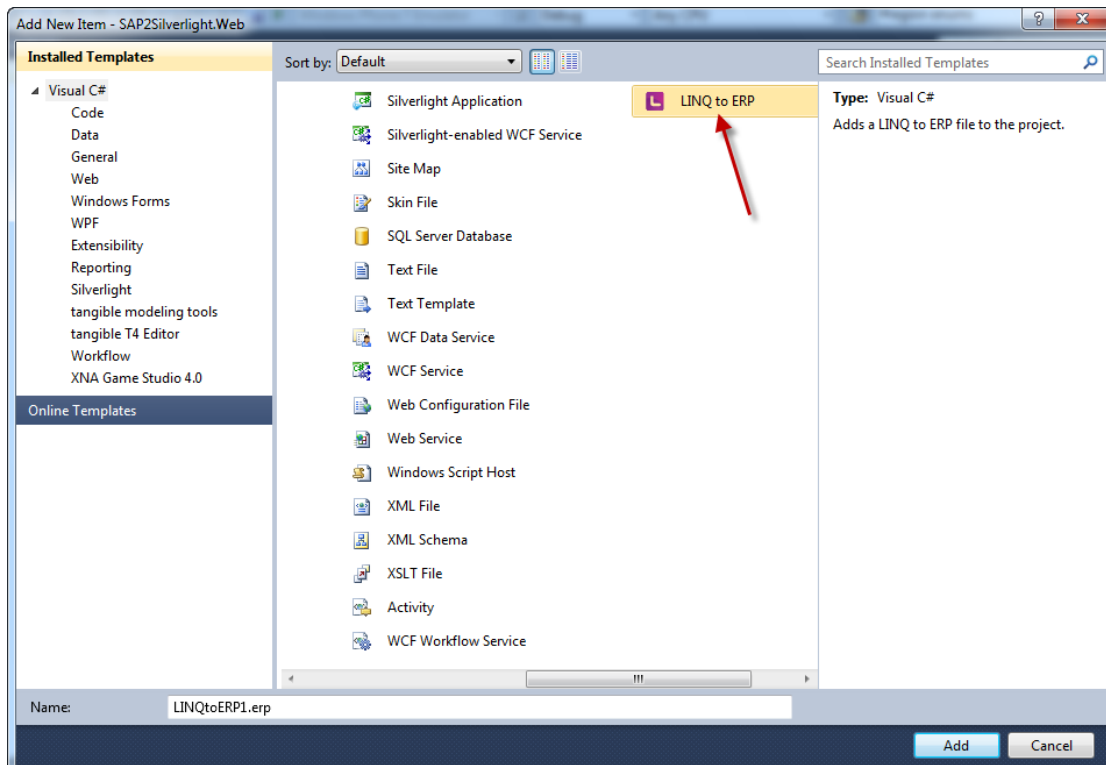


After clicking the Ok button, Visual Studio generates a solution with two projects, one Silverlight 4 project and one ASP.NET project. In the next section we are creating the SAP data access layer using the LINQ to SAP designer.

LINQ to SAP

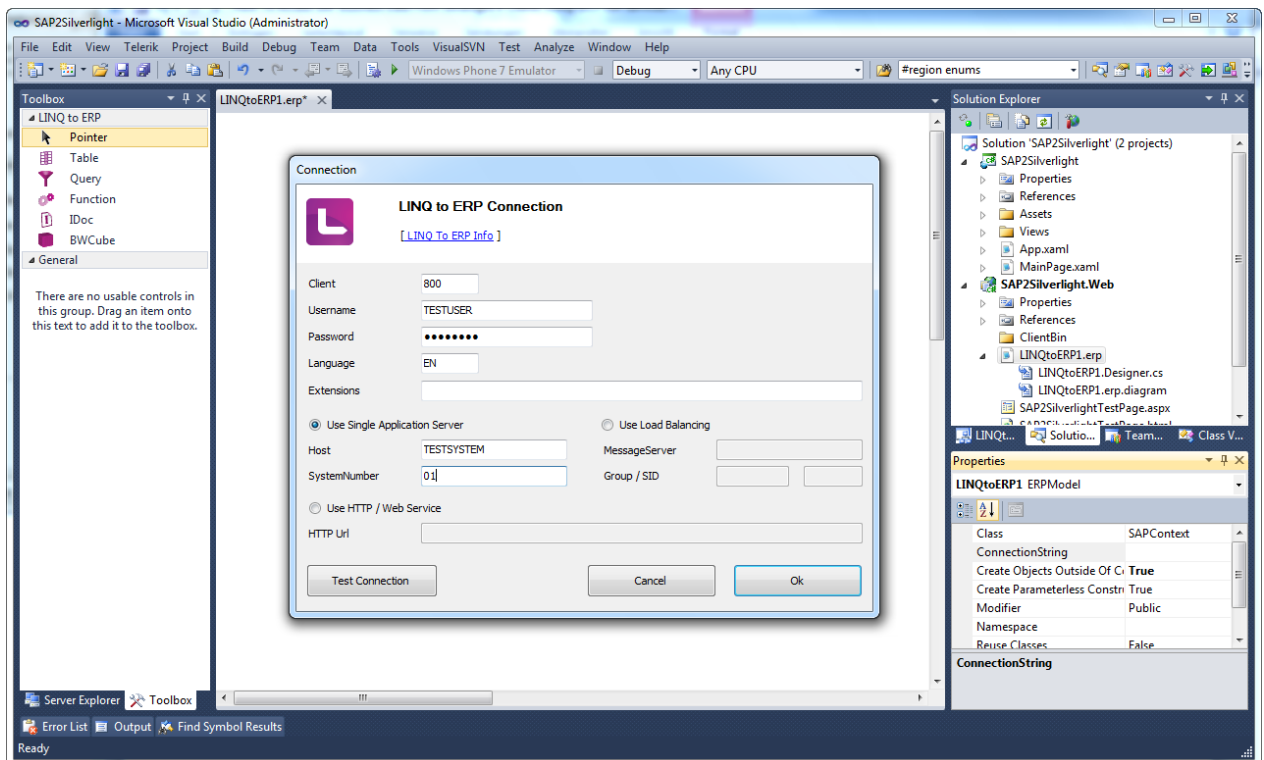
The LINQ to SAP provider and its Visual Studio 2010 designer offers a very handy way to design SAP interfaces visually. The designer will generate the code for the SAP data access layer automatically, similar to LINQ to SQL. The LINQ provider is part of the .NET library ERPConnect.net from Theobald Software. The company offers a demo version for download on its homepage.

The next step is to create the needed LINQ to SAP file by opening the *Add New Item* dialog:



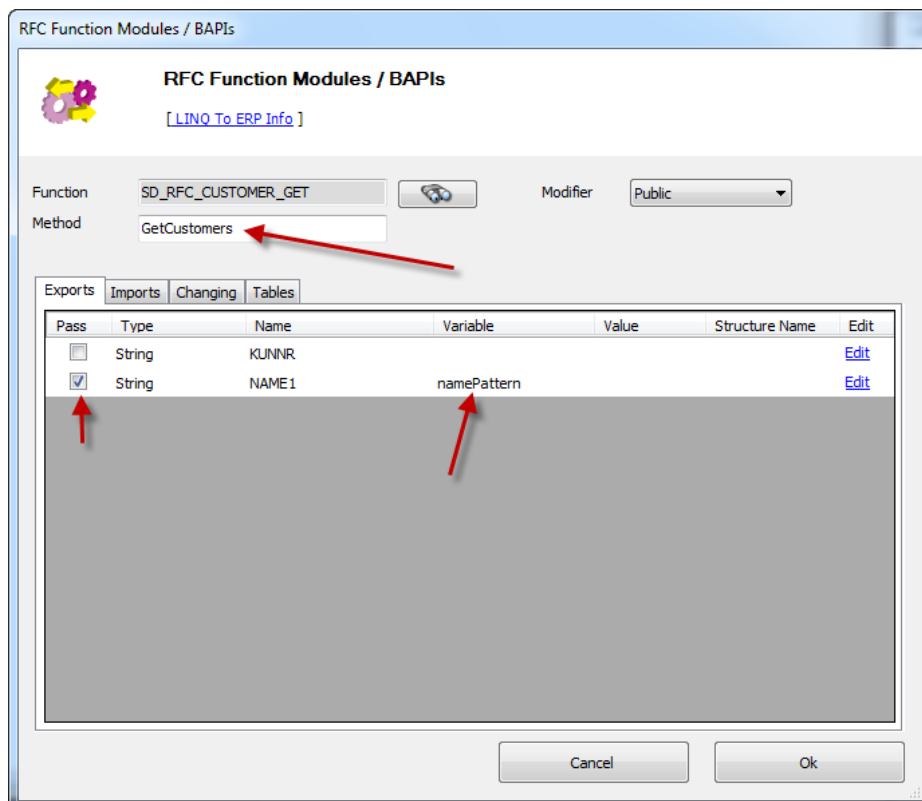
LINQ to SAP is internally called LINQ to ERP.

Clicking the Add button will create a new ERP file and opens the LINQ designer. Now, drag the Function object from the toolbox and drop it onto the designer surface. If you have not entered the SAP connection data so far, you are now asked to do so:



Enter the connection data for your SAP R/3 system and then click the Ok button. Next, search for and select the SAP function module named *SD_RFC_CUSTOMER_GET*. The function module provides a list of customer data.

The *RFC Function modules* dialog opens and let you define the necessary parameters:



In the above function dialog, change the method name to *GetCustomers* and mark the *Pass* checkbox for the *NAME1* parameter in the *Exports* tab. Also set the variable name to *namePattern*. On the *Tables* tab mark the *Return* checkbox for the table parameter *CUSTOMER_T* and set the table and structure name to *CustomerTable* and *CustomerRow*.

RFC Function Modules / BAPIs

[LINK To ERP Info]

Function: SD_RFC_CUSTOMER_GET Modifier: Public

Method: GetCustomers

Exports Imports Changing Tables

Return	Pass	Name	Variable	Table Name	Structure Name	Edit
<input checked="" type="checkbox"/>	<input type="checkbox"/>	CUSTOMER_T		CustomerTable	CustomerRow	Edit

Cancel Ok

After clicking the Ok button and saving the ERP file, the LINQ designer will generate a *SAPContext* class which contains a method called *GetCustomers* with an input parameter named *namePattern*. This method executes a search for SAP customer data allowing the user to enter a wildcard pattern. The method returns a table of customer data:

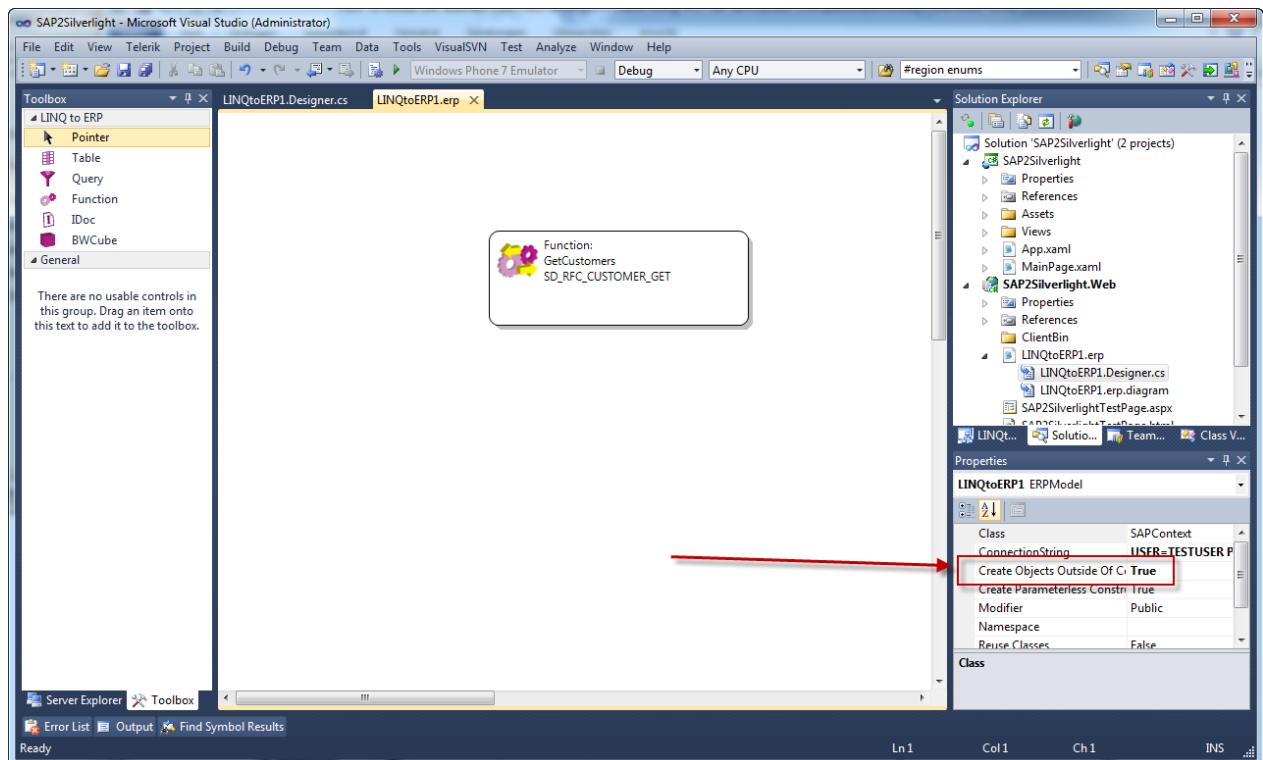
```

LINQtoERP1.Designer.cs    LINQtoERP1.erp
SAP2Silverlight.Web.SAPContext    SAPContext()

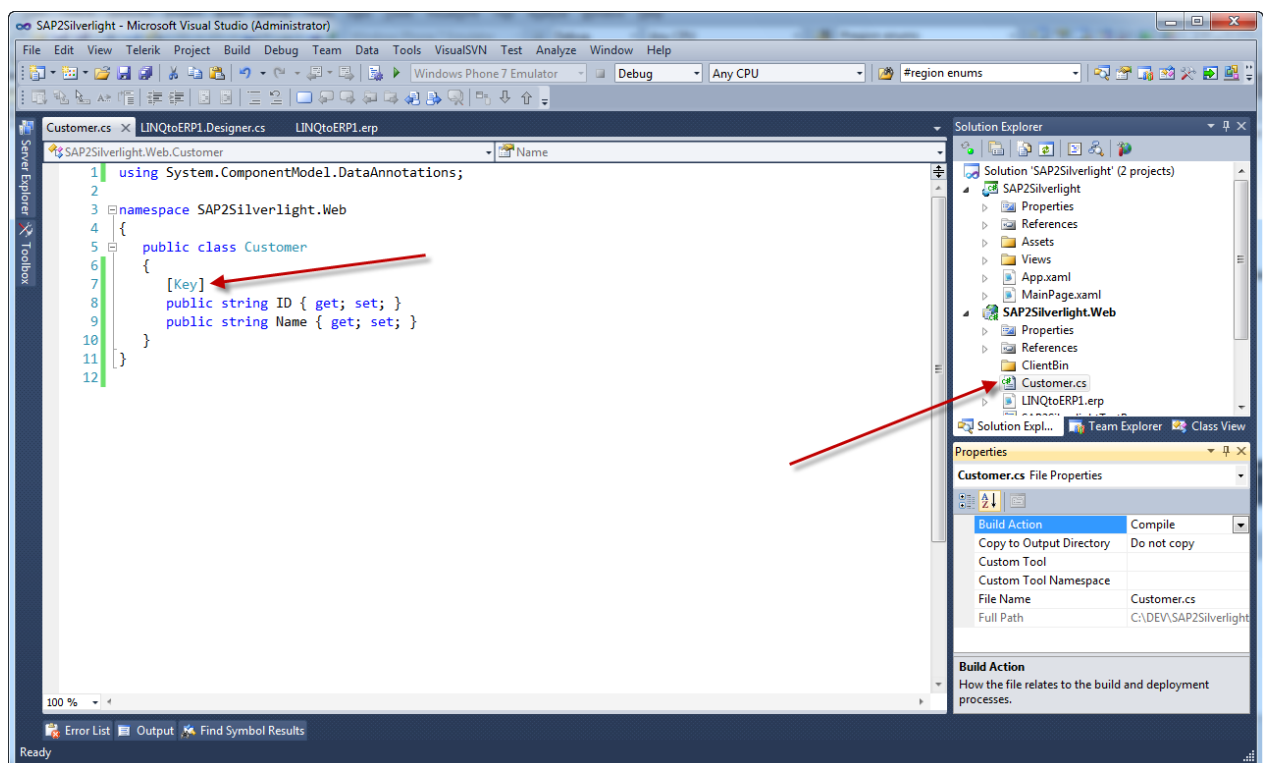
46    public SAPContext(string httpUrl, string username, string password, string
47       : base(httpUrl, username, password, language, client)
48       {
49       }
50
51    public SAPContext(string messageServer, string SID, string logonGroup, string
52       : base(string.Format(@"MSHOST=""{0}"" R3NAME=""{1}"" GROUP=""{2}"" USER=""
53       {
54       }
55
56    public CustomerTable GetCustomers(string namePattern)
57    {
58       if(!Connection.Ping()) Connection.Open();
59
60       RFCFunction function = Connection.CreateFunction("SD_RFC_CUSTOMER_GET");
61
62       function.Exports["NAME1"].ParamValue = namePattern;
63
64       CustomerTable customertable = new CustomerTable();
65       customertable.Table.Name = "CUSTOMER_T";
66       function.Tables["CUSTOMER_T"] = customertable.Table;
67
68       function.Execute();
69
70       return customertable;
71    }
72    }
73
74    public partial class CustomerTable : ITable, ITableSource

```

On the LINQ designer level (click on the free part of the LINQ designer surface) the property *Create Object Outside Of Context Class* must be set to *True*.



Now, we finally add a *Customer* class which we use in our SAP domain service later on. This class and its values will be transmitted to the Silverlight client by the WCF RIA Services. It's important to set the Key attribute on the identifier fields for WCF RIA Services, otherwise the project will not compile:

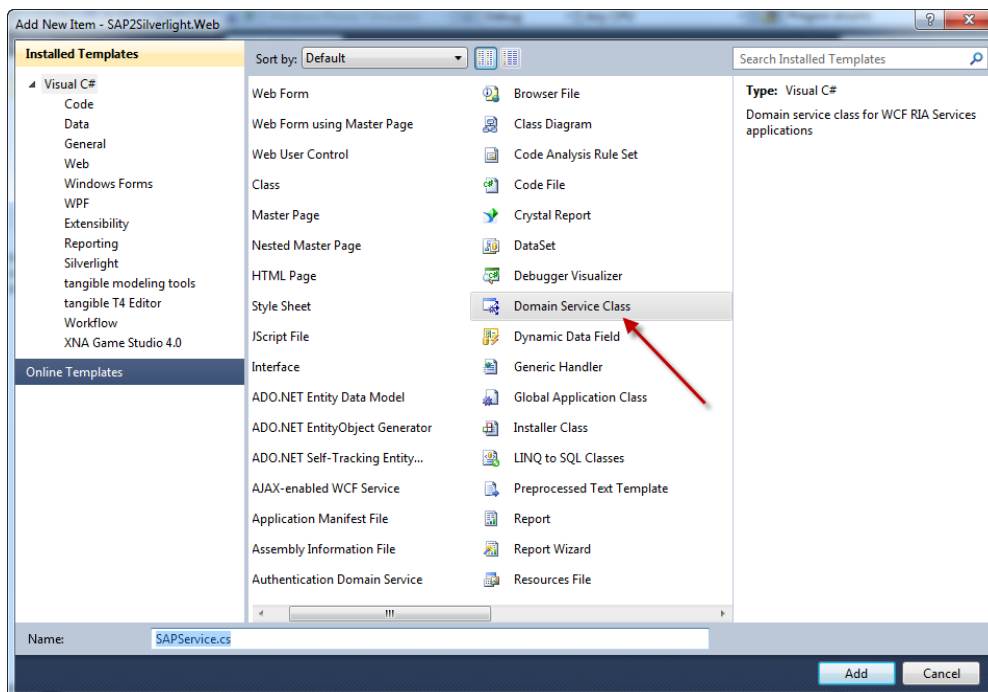


That's it! We now have our SAP data access layer ready to use and can start adding the domain service in the next section.

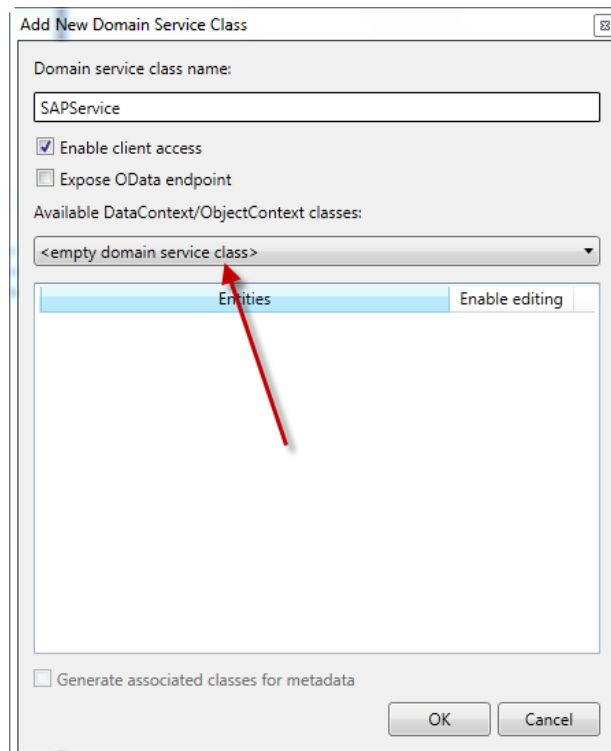
SAP Domain Service

The next step is to add the SAP domain service to our web project. A domain service is a specialized WCF service and is one of the core constructs of WCF RIA Services. The service exposes operations that can be called from the client generated code. On the client side we use the domain context to access the domain service on the server side.

Add a new *Domain Service Class* and name it *SAPService*:



In the upcoming dialog create an empty domain service class by just clicking the Ok button:

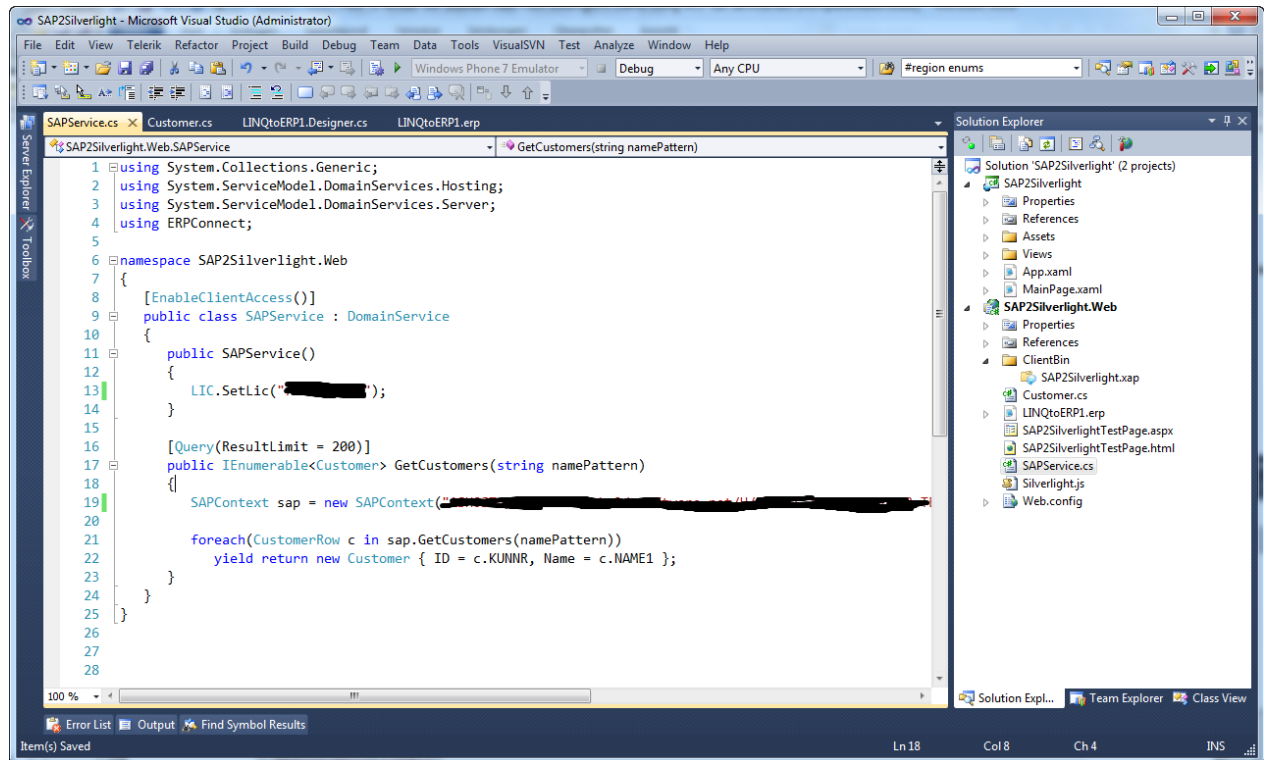


Next, we add the service operation *GetCustomers* to the SAP service with a name pattern parameter. The operation then returns a list of *Customer* objects. The Query attribute limits the result set to 200 entries.

The operation uses the visually designed SAP data access logic to retrieve the SAP customer data. First of all, an instance of the *SAPContext* class will be created using a connection string (see sample in code). For more details regarding the SAP connection string see the ERPConnect.net manual.

The LINQ to SAP context class contains the *GetCustomers* method which we will call using the given namePattern parameter. Next, the operation creates an instance of the *Customer* class for each customer record returned by SAP.

The license code for the ERPConnect.net library is set in the constructor of our domain service class.



That's all we need on the server side.

In the next section we are implementing the Silverlight client.

Silverlight Client

The implementation of the client side is straightforward. The home view contains a DataGrid control to display the list of customer data as well as a search area with TextBox and Button controls to allow users to enter name search pattern.

The click event handler of the load button, called *OnLoadButtonClick*, will execute the SAP service. The boilerplate code to access the web service was generated by WCF RIA Services in the subfolder *Generated_Code* in the Silverlight project.

First of all, an instance of the *SAPContext* will be created. Then we load the query *GetCustomersQuery* and execute the service operation on the server side using WCF RIA Services. If the domain service returns an error, the callback anonymous method will mark the error as handled and display the error message.

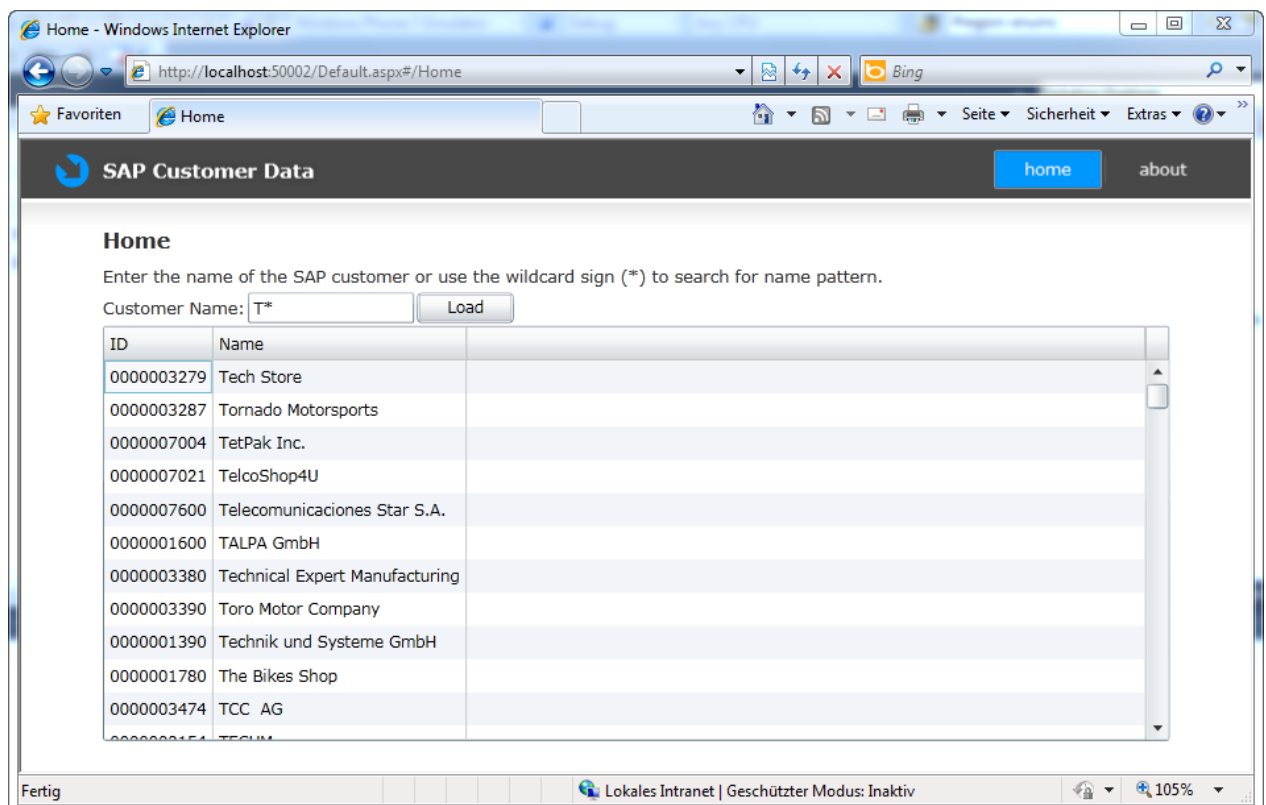
If the execution of the service operation succeeded the result set gets displayed in the DataGrid control.


```

9      {
10         public Home()
11         {
12             InitializeComponent();
13         }
14     }
15     void OnLoadButtonClick(object sender, RoutedEventArgs e)
16     {
17         LoadButton.IsEnabled = false;
18
19         string namePattern = string.IsNullOrEmpty(NamePatternTextBox.Text) ? "*" : NamePatternText
20
21         SAPContext context = new SAPContext();
22
23         context.Load(context.GetCustomersQuery(namePattern), delegate(LoadOperation<Customer> operation
24
25             if(operation.HasError)
26             {
27                 operation.MarkErrorAsHandled();
28                 MessageBox.Show(operation.Error.Message);
29             }
30             else
31                 CustomerList.ItemsSource = operation.Entities;
32
33             LoadButton.IsEnabled = true;
34
35         }, null);
36     }

```

The next screenshot shows the final result:



That's it.

Summary

This article has shown how easy SAP customer data can be integrate within Silverlight clients using tools like WCF RIA Services and LINQ to SAP. It is quite simple to extend the SAP service to integrate all kind of operations.

Contact Information

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

Jürgen Bäurle

jbaurle@parago.de

<http://www.parago.de/jbaurle>

Parago Media GmbH & Co. KG

Im Wengert 3 | 71336 Waiblingen, Germany | Phone +49.7146.861803 | Internet <http://www.parago.de>