Consuming a SQL 2005 Native Web Service from an ASP.NET application

SQL Native Web Services Part 2

In the first article in this series I showed how to create and configure a basic web service in SQL Server 2005 using SQL Native Web Services. In this follow-up I'll describe the steps necessary to call this data and display it in a web browser using ASP.NET.

If you haven't already done so, follow all the steps in part one to create the web service as we'll be using it presently.

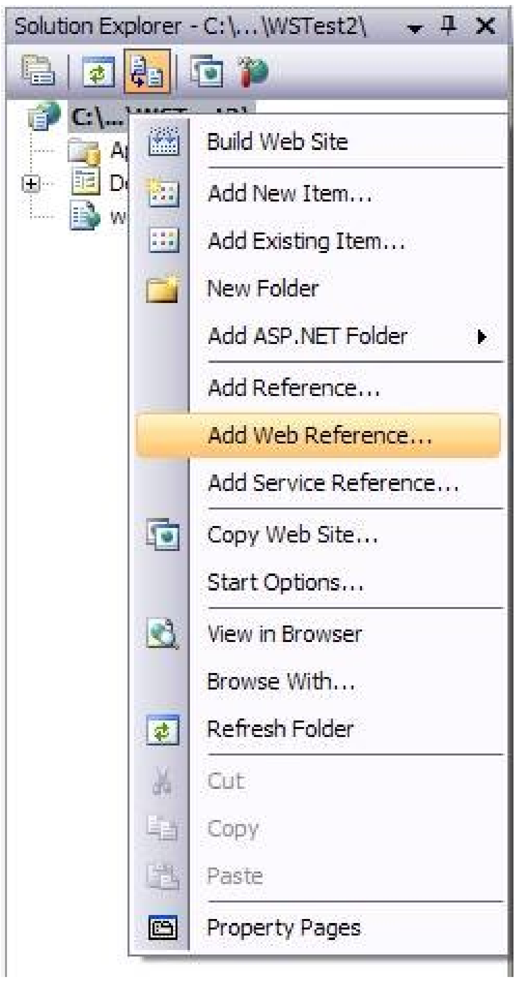
Getting started

All the ASP.NET functionality you'll need for this example is available in the free Visual Web Developer 2008 IDE, so that's what I'll be describing in this article. The steps are all but identical using Visual Studio 2005 or later, however, with the .NET Framework 3.5 installed.

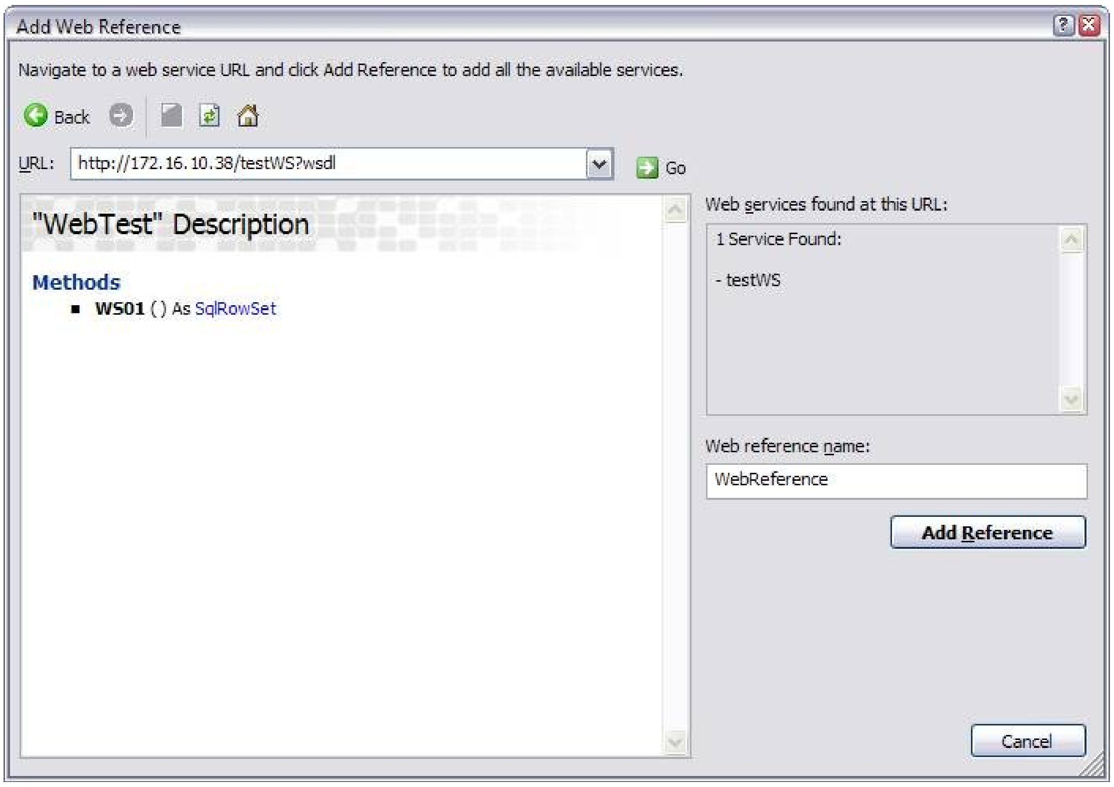
Start a new "ASP.NET Web Site" project and choose **Visual Basic** as the project language.

Adding the web reference

In the Solution Explorer, right-click on the project name and select 'Add Web Reference…' from the context menu.



Enter the URL for the WSDL file (as described in Part 1) and enter your domain credentials when prompted for a user name and password. You should now see the “WebTest” web reference with “WS01” listed as its only method.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://172.16.10.38/testWS?wsdl

"WebTest" Description

Methods

- WS01 ( ) As SqlRowSet

Web services found at this URL:

1 Service Found:

- testWS

Web reference name: WebReference

(If you have problems accessing this, you may want to check/disable your Internet Explorer/Windows proxy settings).

Click “Add Reference”. Note the name you give to the web reference – by default this has the imaginative name “WebReference” (and the listed examples) but if you change it you’ll need to change the code listed below too. In the Solution Explorer you’ll now see a new folder “App_WebReferences” with a number of subfolders and the lowest level the WSDL file for the **testWS** web service.

The data table

Open the Default.aspx page and add a basic GridView control and enable the AutoGenerateColumns property.

```
<form id="form1" runat="server">
…
        <asp:GridView ID="gv1" runat="server"
        AutoGenerateColumns="true"></asp:GridView>
…
</form>
```

Adding the code to populate the grid

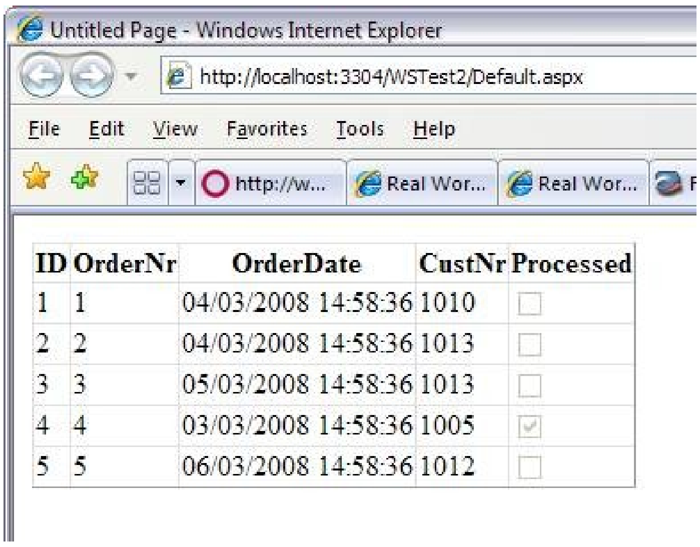
Open the code-behind page Default.aspx.vb. Add the namespace references as follows:

```
Imports WebReference.WebTest  ' Change "WebReference" as appropriate
Imports System.Net
Imports Microsoft.VisualBasic
Imports System.Data.SqlTypes
```

Now add a Page_Load event:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles Me.Load

    Dim WSRequest = New WebReference.WebTest()
    Dim ResultSet As Object

    ' Pass Windows login to web service
    WSRequest.Credentials = System.Net.CredentialCache.DefaultCredentials

    ' Call web method
    ResultSet = WSRequest.WS01()

    ' Bind resulting data to control
    gv1.DataSource = ResultSet
    gv1.DataBind()

End Sub
```

The "WSRequest.Credentials" line is where we define the use of the Windows login. In the previous article we've given the login/user BEDROCK\Fred access to the web service. Here, we instruct the web app to pass the current user's Windows login details to the web service, so if BEDROCK\Fred is logged in on the client machine he shouldn't meet with any problems.


Untitled Page - Windows Internet Explorer
http://localhost:3304/WSTest2/Default.aspx
File Edit View Favorites Tools Help
http://w... Real Wor... Real Wor...

ID	OrderNr	OrderDate	CustNr	Processed
1	1	04/03/2008 14:58:36	1010	☐
2	2	04/03/2008 14:58:36	1013	☐
3	3	05/03/2008 14:58:36	1013	☐
4	4	03/03/2008 14:58:36	1005	☑
5	5	06/03/2008 14:58:36	1012	☐

Hit F5 to run your app in debug mode *et voilà*…your SQL list is displayed in a (somewhat untidy looking) table in your web browser.

Try changing the SQL data in the Management Studio and then refreshing your browser; the data changes are immediately reflected in the browser table.

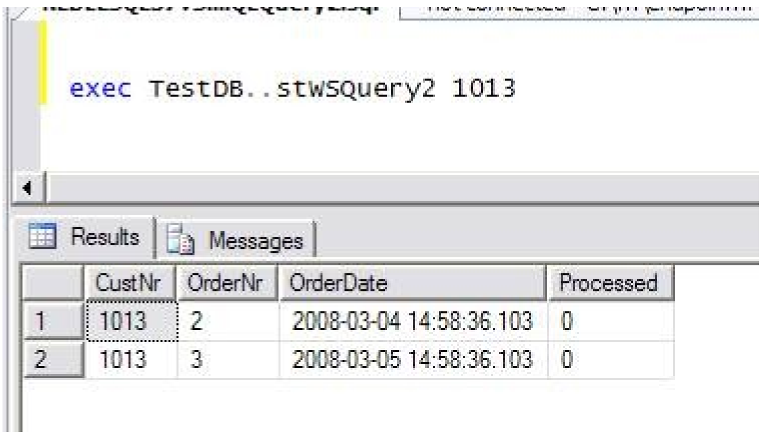
Making things interesting

What we've now got is a static display of data. It's useful up to a point, perhaps, but limited. What if we want to filter the data and control the filter from the browser?

The first thing we'll need is a new SQL stored procedure that accepts a parameter in order to filter the data.

```
USE [TestDB]
GO


CREATE procedure [dbo].[stWSQuery2]

       @p_CustNr int

as
begin

       --     Parameter validations

       if not exists (select 1
                      from   tbTestData
                      where  CustNr = @p_CustNr)
       begin
              raiserror ('ERROR: No matching customer number found.', 10, 1)
              return @@error
       end


       select   CustNr, OrderNr, OrderDate, Processed
       from   TestDB.dbo.tbTestData
       where  CustNr = @p_CustNr

end
```

If we try this out in the SQL Management Studio we get two records:



```
exec TestDB..stWSQuery2 1013
```

	CustNr	OrderNr	OrderDate	Processed
1	1013	2	2008-03-04 14:58:36.103	0
2	1013	3	2008-03-05 14:58:36.103	0

The next step is to add this SP as a method to our existing web service:

```
ALTER ENDPOINT WebTest
FOR SOAP
        (add webmethod 'WS02'
                (name='TestDB.dbo.stWSQuery2',
                schema = standard,
                format=rowsets_only),
        wsdl = default,
        schema = standard,
        database= 'TestDB')
```

Grant permissions as before and the SQL preparations are completed. Now go back to Visual Web Developer. Right-click on 'WebReference' in the Solution Explorer and choose 'Update Web/Service References' from the context menu. We need to do this to be able to call the new method we've just added.
Add this code to your Default.aspx, just above the GridView code:

```
        <div id="dd">
                <asp:Label ID="lab1" runat="server"
                        Text="Customer Number:">
                </asp:Label>
                <asp:DropDownList ID="dd1" runat="server" AutoPostBack="true">
                        <asp:ListItem Text="1005" Value="1005"></asp:ListItem>
                        <asp:ListItem Text="1010" Value="1010"></asp:ListItem>
                        <asp:ListItem Text="1012" Value="1012"></asp:ListItem>
                        <asp:ListItem Text="1013" Value="1013"></asp:ListItem>
                </asp:DropDownList>
        </div>
```

This gives us a four-item drop-down box with one item per Customer Number. In this example it's hard-coded, just to keep things short and simple, but you could populate this list in turn from a web service or an ASP.NET SQLDataSource doing a SELECT DISTINCT on the tbTestData table. For now, let's just leave it like this.

We also need to make a few changes to our old Page_Load event handler:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As _
        System.EventArgs) Handles Me.Load

        Dim WSRequest = New WebReference.WebTest()
        Dim ResultSet As Object
        Dim p_CustNr As SqlInt32

        ' Default range
        p_CustNr = 1013

        ' Pass Windows login to web service
        WSRequest.Credentials = System.Net.CredentialCache.DefaultCredentials

        ' Call web method
        ResultSet = WSRequest.WS02(p_CustNr)

        ' Bind resulting data to control
        gv1.DataSource = ResultSet
        gv1.DataBind()

End Sub
```

…and lastly, an event handler is needed for the drop-down update:

```
Protected Sub dd1_SelectedIndexChanged(ByVal sender As Object, ByVal e As_
        System.EventArgs) Handles dd1.SelectedIndexChanged

        Dim WSRequest = New WebReference.WebTest()
        Dim ResultSet As Object
        Dim p_CustNr As SqlInt32

        p_CustNr = dd1.SelectedValue

        WSRequest.Credentials = System.Net.CredentialCache.DefaultCredentials

        ResultSet = WSRequest.WS02(p_CustNr)

        gv1.DataSource = ResultSet
        gv1.DataBind()

End Sub
```

Running this app now displays a table filtered on the value selected in the drop-down. Selecting a different value from the drop-down updates the table.

We've now got a web service running natively in SQL exposing a parameterized stored procedure, and an application that passes that parameter to the web service and displays the resulting data.

Paul Clancy
360Data
http://www.360data.nl

Links

Providing SQL data as a web service using SQL Server 2005 Native Web Services
http://www.360data.nl/EN/Docs/080215_NativeWS01.aspx

Visual Web Developer 2008 Express Edition
http://www.microsoft.com/express/vwd/

.NET Framework 3.5
http://msdn2.microsoft.com/en-us/netframework/default.aspx