.NET Framework 4

**Consuming a DataSet from an XML Web Service (ADO.NET)**

The [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) was architected with a disconnected design, in part to facilitate the convenient transport of data over the Internet. The **DataSet** is "serializable" in that it can be specified as an input to or output from XML Web services without any additional coding required to stream the contents of the **DataSet** from an XML Web service to a client and back. The **DataSet** is implicitly converted to an XML stream using the DiffGram format, sent over the network, and then reconstructed from the XML stream as a **DataSet** on the receiving end. This gives you a very simple and flexible method for transmitting and returning relational data using XML Web services. For more information about the DiffGram format, see [DiffGrams (ADO.NET)](http://msdn.microsoft.com/en-us/library/ms172088.aspx).

The following example shows how to create an XML Web service and client that use the **DataSet** to transport relational data (including modified data) and resolve any updates back to the original data source.

|  |
| --- |
| **NoteNote** |
| We recommend that you always consider security implications when creating an XML Web service. For information on securing an XML Web service, see [Securing XML Web Services Created Using ASP.NET](http://msdn.microsoft.com/en-us/library/w67h0dw7.aspx). |

**To create an XML Web service that returns and consumes a DataSet**

1. Create the XML Web service.

In the example, an XML Web service is created that returns data, in this case a list of customers from the **Northwind** database, and receives a **DataSet** with updates to the data, which the XML Web service resolves back to the original data source.

The XML Web service exposes two methods: **GetCustomers**, to return the list of customers, and **UpdateCustomers**, to resolve updates back to the data source. The XML Web service is stored in a file on the Web server called DataSetSample.asmx. The following code outlines the contents of DataSetSample.asmx.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl05_code');" \o "Copy Code)

<% @ WebService Language = "vb" Class = "Sample" %>

Imports System

Imports System.Data

Imports System.Data.SqlClient

Imports System.Web.Services

<WebService(Namespace:="http://microsoft.com/webservices/")> \_

Public Class Sample

Public connection As SqlConnection = New SqlConnection("Data Source=(local);Integrated Security=SSPI;Initial Catalog=Northwind")

<WebMethod( Description := "Returns Northwind Customers", EnableSession := False )> \_

Public Function GetCustomers() As DataSet

Dim adapter As SqlDataAdapter = New SqlDataAdapter( \_

"SELECT CustomerID, CompanyName FROM Customers", connection)

Dim custDS As DataSet = New DataSet()

adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey

adapter.Fill(custDS, "Customers")

Return custDS

End Function

<WebMethod( Description := "Updates Northwind Customers", EnableSession := False )> \_

Public Function UpdateCustomers(custDS As DataSet) As DataSet

Dim adapter As SqlDataAdapter = New SqlDataAdapter()

adapter.InsertCommand = New SqlCommand( \_

"INSERT INTO Customers (CustomerID, CompanyName) " & \_

"Values(@CustomerID, @CompanyName)", connection)

adapter.InsertCommand.Parameters.Add( \_

"@CustomerID", SqlDbType.NChar, 5, "CustomerID")

adapter.InsertCommand.Parameters.Add( \_

"@CompanyName", SqlDbType.NChar, 15, "CompanyName")

adapter.UpdateCommand = New SqlCommand( \_

"UPDATE Customers Set CustomerID = @CustomerID, " & \_

"CompanyName = @CompanyName WHERE CustomerID = " & \_

@OldCustomerID", connection)

adapter.UpdateCommand.Parameters.Add( \_

"@CustomerID", SqlDbType.NChar, 5, "CustomerID")

adapter.UpdateCommand.Parameters.Add( \_

"@CompanyName", SqlDbType.NChar, 15, "CompanyName")

Dim parameter As SqlParameter = \_

adapter.UpdateCommand.Parameters.Add( \_

"@OldCustomerID", SqlDbType.NChar, 5, "CustomerID")

parameter.SourceVersion = DataRowVersion.Original

adapter.DeleteCommand = New SqlCommand( \_

"DELETE FROM Customers WHERE CustomerID = @CustomerID", \_

connection)

parameter = adapter.DeleteCommand.Parameters.Add( \_

"@CustomerID", SqlDbType.NChar, 5, "CustomerID")

parameter.SourceVersion = DataRowVersion.Original

adapter.Update(custDS, "Customers")

Return custDS

End Function

End Class

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl06_code');" \o "Copy Code)

<% @ WebService Language = "C#" Class = "Sample" %>

using System;

using System.Data;

using System.Data.SqlClient;

using System.Web.Services;

[WebService(Namespace="http://microsoft.com/webservices/")]

public class Sample

{

public SqlConnection connection = new SqlConnection("Data Source=(local);Integrated Security=SSPI;Initial Catalog=Northwind");

[WebMethod( Description = "Returns Northwind Customers", EnableSession = false )]

public DataSet GetCustomers()

{

SqlDataAdapter adapter = new SqlDataAdapter(

"SELECT CustomerID, CompanyName FROM Customers", connection);

DataSet custDS = new DataSet();

adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;

adapter.Fill(custDS, "Customers");

return custDS;

}

[WebMethod( Description = "Updates Northwind Customers",

EnableSession = false )]

public DataSet UpdateCustomers(DataSet custDS)

{

SqlDataAdapter adapter = new SqlDataAdapter();

adapter.InsertCommand = new SqlCommand(

"INSERT INTO Customers (CustomerID, CompanyName) " +

"Values(@CustomerID, @CompanyName)", connection);

adapter.InsertCommand.Parameters.Add(

"@CustomerID", SqlDbType.NChar, 5, "CustomerID");

adapter.InsertCommand.Parameters.Add(

"@CompanyName", SqlDbType.NChar, 15, "CompanyName");

adapter.UpdateCommand = new SqlCommand(

"UPDATE Customers Set CustomerID = @CustomerID, " +

"CompanyName = @CompanyName WHERE CustomerID = " +

"@OldCustomerID", connection);

adapter.UpdateCommand.Parameters.Add(

"@CustomerID", SqlDbType.NChar, 5, "CustomerID");

adapter.UpdateCommand.Parameters.Add(

"@CompanyName", SqlDbType.NChar, 15, "CompanyName");

SqlParameter parameter = adapter.UpdateCommand.Parameters.Add(

"@OldCustomerID", SqlDbType.NChar, 5, "CustomerID");

parameter.SourceVersion = DataRowVersion.Original;

adapter.DeleteCommand = new SqlCommand(

"DELETE FROM Customers WHERE CustomerID = @CustomerID",

connection);

parameter = adapter.DeleteCommand.Parameters.Add(

"@CustomerID", SqlDbType.NChar, 5, "CustomerID");

parameter.SourceVersion = DataRowVersion.Original;

adapter.Update(custDS, "Customers");

return custDS;

}

}

In a typical scenario, the **UpdateCustomers** method would be written to catch optimistic concurrency violations. For simplicity, the example does not include this. For more information about optimistic concurrency, see [Optimistic Concurrency (ADO.NET)](http://msdn.microsoft.com/en-us/library/aa0416cz.aspx).

1. Create an XML Web service proxy.

Clients of the XML Web service require a SOAP proxy in order to consume the exposed methods. You can have Visual Studio generate this proxy for you. By setting a Web reference to an existing Web service from within Visual Studio, all the behavior described in this step occurs transparently. If you want to create the proxy class yourself, continue with this discussion. In most circumstances, however, using Visual Studio to create the proxy class for the client application is sufficient.

A proxy can be created using the Web Services Description Language Tool. For example, if the XML Web service is exposed at the URL http://myserver/data/DataSetSample.asmx, issue a command such as the following to create a Visual Basic .NET proxy with a namespace of **WebData.DSSample** and store it in the file sample.vb.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

wsdl /l:VB /out:sample.vb http://myserver/data/DataSetSample.asmx /n:WebData.DSSample

To create a C# proxy in the file sample.cs, issue the following command.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl09_code');" \o "Copy Code)

wsdl /l:CS /out:sample.cs http://myserver/data/DataSetSample.asmx /n:WebData.DSSample

The proxy can then be compiled as a library and imported into the XML Web service client. To compile the Visual Basic .NET proxy code stored in sample.vb as sample.dll, issue the following command.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl10_code');" \o "Copy Code)

vbc /t:library /out:sample.dll sample.vb /r:System.dll /r:System.Web.Services.dll /r:System.Data.dll /r:System.Xml.dll

To compile the C# proxy code stored in sample.cs as sample.dll, issue the following command.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

csc /t:library /out:sample.dll sample.cs /r:System.dll /r:System.Web.Services.dll /r:System.Data.dll /r:System.Xml.dll

1. Create an XML Web service client.

If you want to have Visual Studio generate the Web service proxy class for you, simply create the client project, and, in the Solution Explorer window, right-click the project, click **Add Web Reference**, and select the Web service from the list of available Web services (this may require supplying the address of the Web service endpoint, if the Web service isn't available within the current solution, or on the current computer.) If you create the XML Web service proxy yourself (as described in the previous step), you can import it into your client code and consume the XML Web service methods. The following sample code imports the proxy library, calls **GetCustomers** to get a list of customers, adds a new customer, and then returns a **DataSet** with the updates to **UpdateCustomers**.

Notice that the example passes the **DataSet** returned by **DataSet.GetChanges** to **UpdateCustomers** because only modified rows need to be passed to **UpdateCustomers**. **UpdateCustomers** returns the resolved **DataSet**, which you can then **Merge** into the existing **DataSet** to incorporate the resolved changes and any row error information from the update. The following code assumes that you have used Visual Studio to create the Web reference, and that you have renamed the Web reference to DsSample in the **Add Web Reference** dialog box.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

Imports System

Imports System.Data

Public Class Client

Public Shared Sub Main()

Dim proxySample As New DsSample.Sample () ' Proxy object.

Dim customersDataSet As DataSet = proxySample.GetCustomers()

Dim customersTable As DataTable = \_

customersDataSet.Tables("Customers")

Dim rowAs DataRow = customersTable.NewRow()

row("CustomerID") = "ABCDE"

row("CompanyName") = "New Company Name"

customersTable.Rows.Add(row)

Dim updateDataSet As DataSet = \_

proxySample.UpdateCustomers(customersDataSet.GetChanges())

customersDataSet.Merge(updateDataSet)

customersDataSet.AcceptChanges()

End Sub

End Class

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl13_code');" \o "Copy Code)

using System;

using System.Data;

public class Client

{

public static void Main()

{

Sample proxySample = new DsSample.Sample(); // Proxy object.

DataSet customersDataSet = proxySample.GetCustomers();

DataTable customersTable = customersDataSet.Tables["Customers"];

DataRow row = customersTable.NewRow();

row["CustomerID"] = "ABCDE";

row["CompanyName"] = "New Company Name";

customersTable.Rows.Add(row);

DataSet updateDataSet = new DataSet();

updateDataSet =

proxySample.UpdateCustomers(customersDataSet.GetChanges());

customersDataSet.Merge(updateDataSet);

customersDataSet.AcceptChanges();

}

}

If you decide to create the proxy class yourself, you must take the following extra steps. To compile the sample, supply the proxy library that was created (sample.dll) and the related .NET libraries. To compile the Visual Basic .NET version of the sample, stored in the file client.vb, issue the following command.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl14_code');" \o "Copy Code)

vbc client.vb /r:sample.dll /r:System.dll /r:System.Data.dll /r:System.Xml.dll /r:System.Web.Services.dll

To compile the C# version of the sample, stored in the file client.cs, issue the following command.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl15_code');" \o "Copy Code)

csc client.cs /r:sample.dll /r:System.dll /r:System.Data.dll /r:System.Xml.dll /r:System.Web.Services.dll