CBA · NAU

College of Business Administration Northern Arizona University
Box 15066
Flagstaff AZ 86011

Creating Web Services Using ASP.NET

Working Paper Series 04-06— April 2004

Craig A. VanLengen
Associate Professor of CIS
craig.vanlengen@nau.edu

and

John D. Haney
Clinical Professor of CIS
john.haney@nau.edu

College of Business Administration
Northern Arizona University
Flagstaff, AZ 86011-5066

CBA·NAU
College of Business Administration Northern Arizona University

Creating Web Services Using ASP.NET

Craig A. VanLengen and John D. Haney

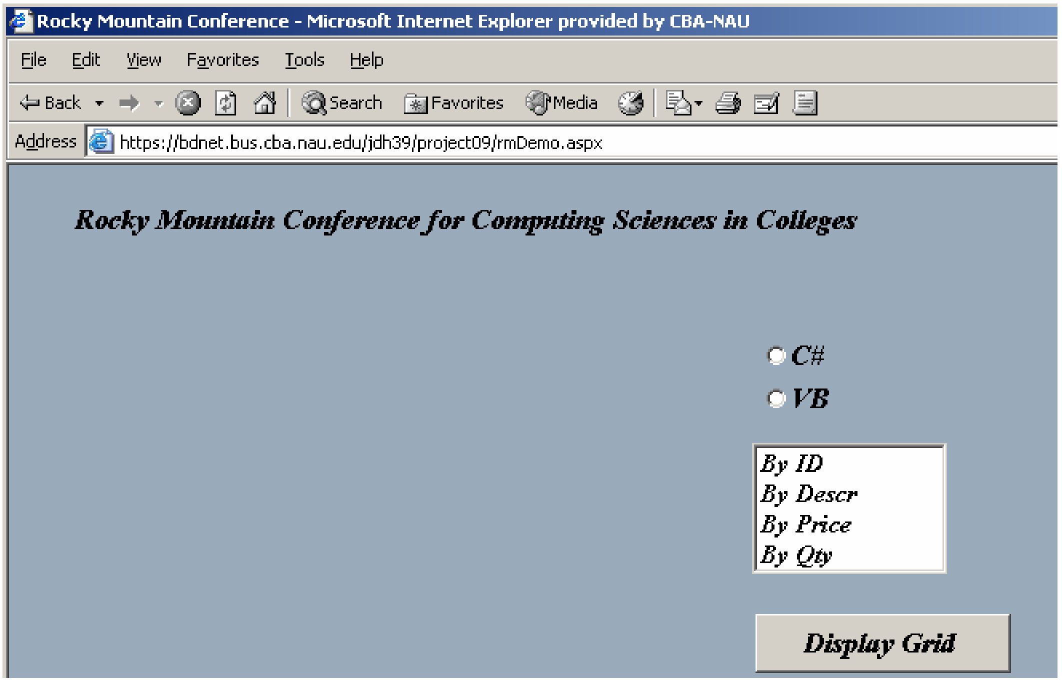
INTRODUCTION

First we will start with some definitions of Web services and the technologies that are used. Let's start with, "What does Web services mean? Web services are a set of standard technologies (W3C) that allow applications to expose and consume functionality independent of operating system, programming language, and hardware. Interfaces are built using open standards. eXtensible Markup Language (XML) is used to describe the data, operations, and the protocols for interaction between the provider of the service and the consumer of the service. The Web service is registered in a Universal Description, Discovery and Integration (UDDI) registry. This registry can be private for intranet access, or exposed to selected business partners over an extranet, or external for the entire Internet. The registration of the Web service is through a discovery file, which is called a Web Service Description Language (WDSL) document, where the namespaces, methods, and URL of the Web service are described. (IBM, MSDN, Kalata). In Microsoft Visual Studio .NET you have a Web discovery service file with a .disco extension and a .wdsl file. This allows the developer to discover Web services from within Visual Studio .NET. This way the objects of the Web service can be used the same as local objects within the consuming application (Kalata).

WEB SERVICE EXAMPLE

The example demonstrates user access to a database that is displayed in sort order by either ID, Description, Price or Quantity. We have added the ability for the user to select either the C# or the Visual Basic service. The user would then select or click on the Display Grid button and the appropriate service would be executed and displayed. The programming language option is provided to demonstrate Web services ability to work with different programming languages. First we present the Web interface for the client application that is consuming the Web service. The following Web address will display the client application:

https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx


Rocky Mountain Conference - Microsoft Internet Explorer provided by CBA-NAU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx
Rocky Mountain Conference for Computing Sciences in Colleges
C#
VB
By ID
By Descr
By Price
By Qty
Display Grid

The example shows a Web service client application that consumes either a Visual Basic.NET or a Visual C#.NET server application. These two service applications are published on Web server and consumed by the client application that is written in Visual Basic.NET. All applications were developed using Microsoft Visual Studio .NET. The .NET development environment provides the necessary support to develop, expose and consume Web services. The Web service application that is exposed is accessible using standard Web protocols.

When the user clicks on the "Display Grid" button the Web service will execute the methods exposed in either dbService_VB.asmx.vb or dbService_CS.asmx.cs and the results will be displayed in the user's browser.

CREATING THE WEB SERVICE SERVER APPLICATION

To create the Web service in Visual Studio .NET you create a new project and select ASP.NET Web Service as the template. This will create the solution, project, and default files with a default Web service page of Service1.asmx.vb. Code needs to be added for the connection to the database and other processing logic. For purposes of this example we will concentrate on the code for Web services.

A server application that is exposed in ASP.NET must have an .asmx for the file extension. The code behind page will have an .asmx.vb or .asmx.cs extension depending on the language used. The System.Web.Services namespace must be imported in the source code of the code behind file so that all the methods of Web services are available.

```
<WebService(Namespace:="http://tempuri.org/")>
```

In the VB Web service the following code is used:

```
Imports System.Web.Services
<WebService(Namespace:="http://tempuri.org/")> _
Public Class VB_Service
```

The default Web service page includes the following commented code:

```
' WEB SERVICE EXAMPLE
' The HelloWorld() example service returns the string Hello World.
' To build, uncomment the following lines then save and build the project.
' To test this web service, ensure that the .asmx file is the start page
' and press F5.
'
'<WebMethod()> Public Function HelloWorld() As String
'       HelloWorld = "Hello World"
' End Function
```

Using the example given in the default code we create the following function to be exposed as a Web service.

```
<WebMethod()> Public Function getProduct(ByVal srtType As String) As DataSet
        'This Service connects to the database,
        'and based on the argument sent, orders the Product table by
        'ID, Description, Price, or Quantity and then returns the
        'selected dataset.
        Connection = "Provider=MSDAORA;Password=oraPW;User ID=oraUserID;Data Source=oracba"
        Conn = New OleDbConnection(Connection)
        'Select from the database table
        If srtType = "ID" Then
                SQL = "SELECT * FROM Product Order by ID"
        End If
        If srtType = "Descr" Then
                SQL = "SELECT * FROM Product Order by Description"
        End If
```

```
          If srtType = "Price" Then
                    SQL = "SELECT * FROM Product Order by Price Desc"
          End If
          If srtType = "Qty" Then
                    SQL = "SELECT * FROM Product Order by Quantity Desc"
          End If
          Cmd = New OleDbCommand(SQL, Conn)
          'Open the connection and establish the SQL statement
          Conn.Open()
          'Create an instance of the adapter and dataset
          Adapter = New OleDbDataAdapter(Cmd)
          ds = New DataSet()
          'Fill the dataset
          Adapter.Fill(ds, "Product")
          'Return the dataset
          Return ds
End Function
End Class
```

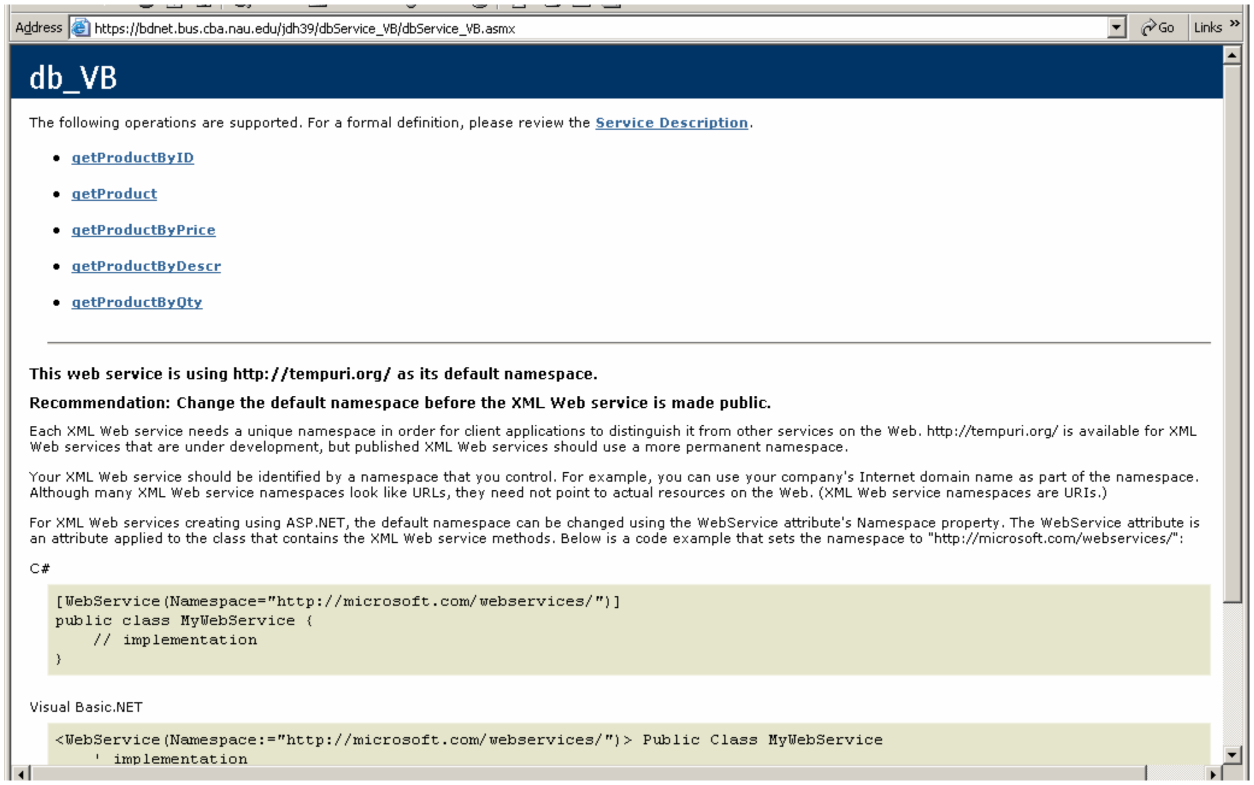
Notice that the function is prefaced by the keyword "<WebMethod()>" that exposes the function as a Web service. At the end of this function we return a DataSet object to the client application with a

```
                Return ds
```

statement.

When we build (compile) the Web service a discovery file, Web Service Description Language (WDSL) document is created. This file is an XML document that describes how to interact with the service and how to format and send data to the service. If we enter the address for this Web service in the browser we get the following:

https://bdnet.bus.cba.nau.edu/jdh39/dbService_VB/dbService_VB.asmx


Address https://bdnet.bus.cba.nau.edu/jdh39/dbService_VB/dbService_VB.asmx
Go
Links

db_VB

The following operations are supported. For a formal definition, please review the **Service Description**.

- **getProductByID**
- **getProduct**
- **getProductByPrice**
- **getProductByDescr**
- **getProductByQty**

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic.NET

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
```

If you click on the Service Description link you will be able to see the WSDL contract, which is an xml file. The top part of the WSDL file is shown below.

```
    <?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
      xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
  <s:import namespace="http://www.w3.org/2001/XMLSchema" />
```

Below the Service Description you see the methods that are exposed:

- getProductByID
- getProduct
- getProductByPrice
- getProductByDescr
- getProductByQty

Clicking on a method allows you to examine a sample SOAP request and response.

```
POST /jdh39/dbService_VB/dbService_VB.asmx HTTP/1.1
Host: bdnet.bus.cba.nau.edu
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/getProductByID"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductByID xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

We also created this same functionality with C#. Some sample code from dbService_CS.asmx.cs is shown below:

```
using System.Web.Services;
namespace rmWSCS
{
        public class CS_Service : System.Web.Services.WebService
        {
                private System.Data.OleDb.OleDbConnection Conn;
                private System.Data.OleDb.OleDbCommand Cmd;
                private System.Data.OleDb.OleDbDataAdapter Adapter;
                private System.Data.DataSet ds;
                private string Connection;
                private string SQL;
                public CS_Service()
                {
                        //CODEGEN: This call is required by the ASP.NET Web Services Designer
                        InitializeComponent();
                }

                [WebMethod]
                public DataSet getProduct(string srtType)
                {
                        /*
                         This Service connects to the database,
                         and based on the argument sent, orders the Product table by
                         ID, Description, Price, or Quantity and then returns the
                         selected dataset.
                        */
                        // Connect to the database
                        Connection = "Provider=MSDAORA;Password=oraPW;User ID=oraUserID;
                        Data Source=oracba";
                        Conn = new System.Data.OleDb.OleDbConnection(Connection);
                        // Select from the database table
                        if (srtType.Equals("ID"))
                        {
                                SQL = "SELECT * FROM Product Order by ID";
                        }
                        if (srtType.Equals("Descr"))
                        {
                                SQL = "SELECT * FROM Product Order by Description";
                        }
                        if (srtType.Equals("Price"))
                        {
                                SQL = "SELECT * FROM Product Order by Price Desc";
                        }
                        if (srtType.Equals("Qty"))
                        {
                                SQL = "SELECT * FROM Product Order by Quantity Desc";
                        }
                        Cmd = new System.Data.OleDb.OleDbCommand(SQL, Conn);
                        // Open the connection and establish the SQL statement
                        Conn.Open();
                        // Create an instance of the adapter and dataset
                        Adapter = new System.Data.OleDb.OleDbDataAdapter(Cmd);
                        ds = new DataSet();
                        // Fill the dataset
                        Adapter.Fill( ds, "Product" );
                        // Return the dataset
                        return ds;
                } // end of function
        } // end of class
} // end of namespace
```

Notice the importing or for C#, using System.Web.Services line and the System.Web.Services.WebService declaration for the public class CS_Service. This inherits all the appropriate methods for Web services. Once again the keyword [WebMethod] prefaces the methods that are being exposed.

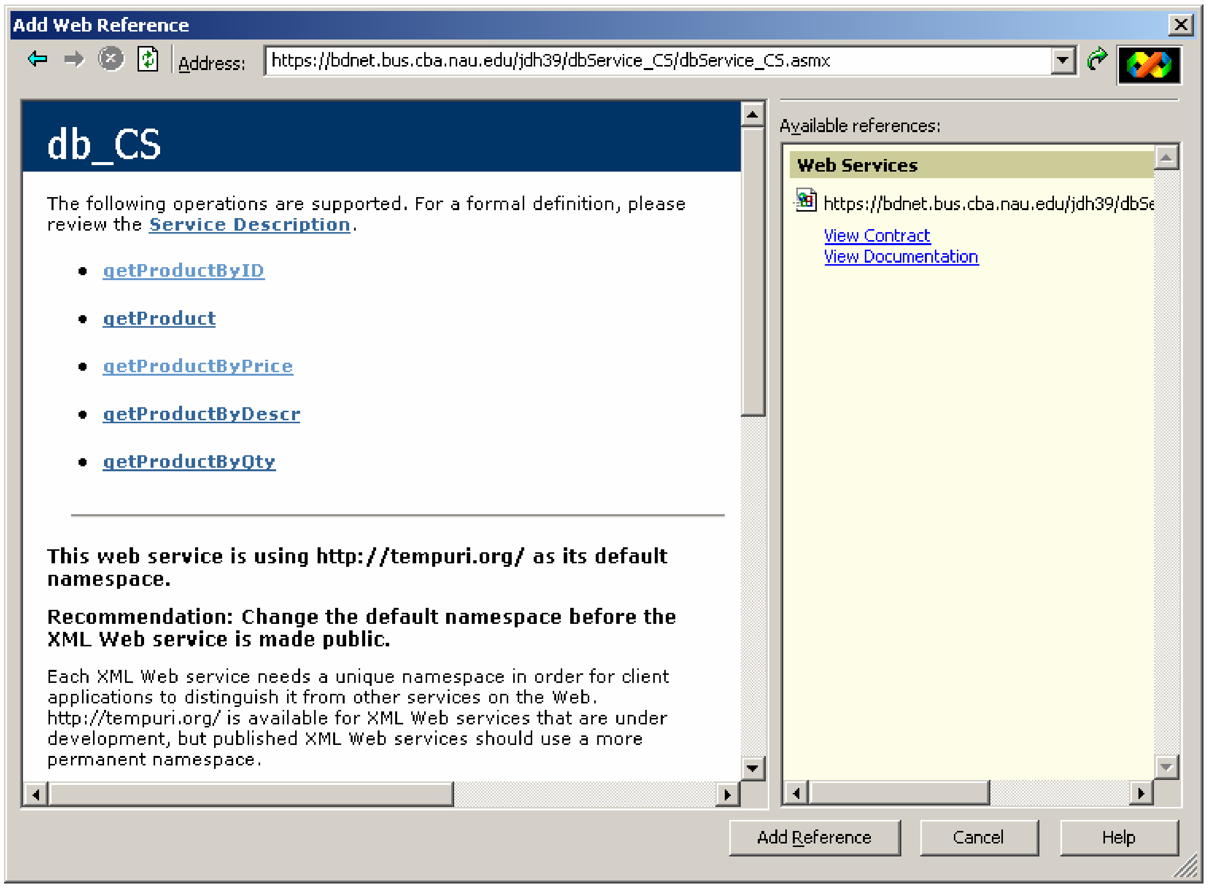
```
        public DataSet getProduct(string srtType)
```

Consuming the Web Service

Now we create the "client" Web application to consume the "server" Web service. This time we create a new project in Visual Studio .NET as an ASP.NET Web Application, not a Web service. The consuming application will be developed in Visual Basic.NET so we have the following files:

rmDemo.aspx
rmDemo.aspx.vb

In the Solution Explorer window add a Web Reference. The Add Web Reference will search or discover the Web Services. We enter the URL of our Web Service "server" application. This provides the following dialog. From this dialog the developer can view the WDSL and other documentation of the Web service.



https://bdnet.bus.cba.nau.edu/jdh39/dbService_CS/dbService_CS.asmx

db_CS

The following operations are supported. For a formal definition, please review the Service Description.

- getProductByID
- getProduct
- getProductByPrice
- getProductByDescr
- getProductByQty

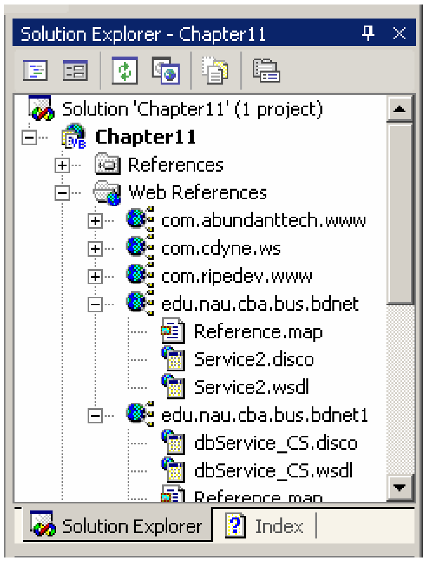
This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

https://bdnet.bus.cba.nau.edu/jdh39/dbSe

Clicking on the Add Reference button in the lower right-hand corner will add a reference to the Web service that allows the developer to treat it like any other object within a page. Visual Studio IntelliSense will provide popup access to the properties and methods of the Web service class. The Web application solution explorer window will now include a Web Reference for the dbService_CS. Note the .disco and .wsdl file under edu.nau.cba.bus.bdnet1.



When you build or compile your consuming application a proxy class is created using the WSDL document. When the Web service application is executed the proxy class reads the WDSL to verify the descriptions of the Web service. When using SOAP for the request and response messages, the proxy creates a SOAP envelope that is passed to the Web service. After the Web service executes the return value is also passed back to the proxy in a SOAP envelope over the HTTP connection. The client Web application must then convert the SOAP XML response into the format required by the client Web application.

In the code behind page .aspx.vb we make the following declarations:

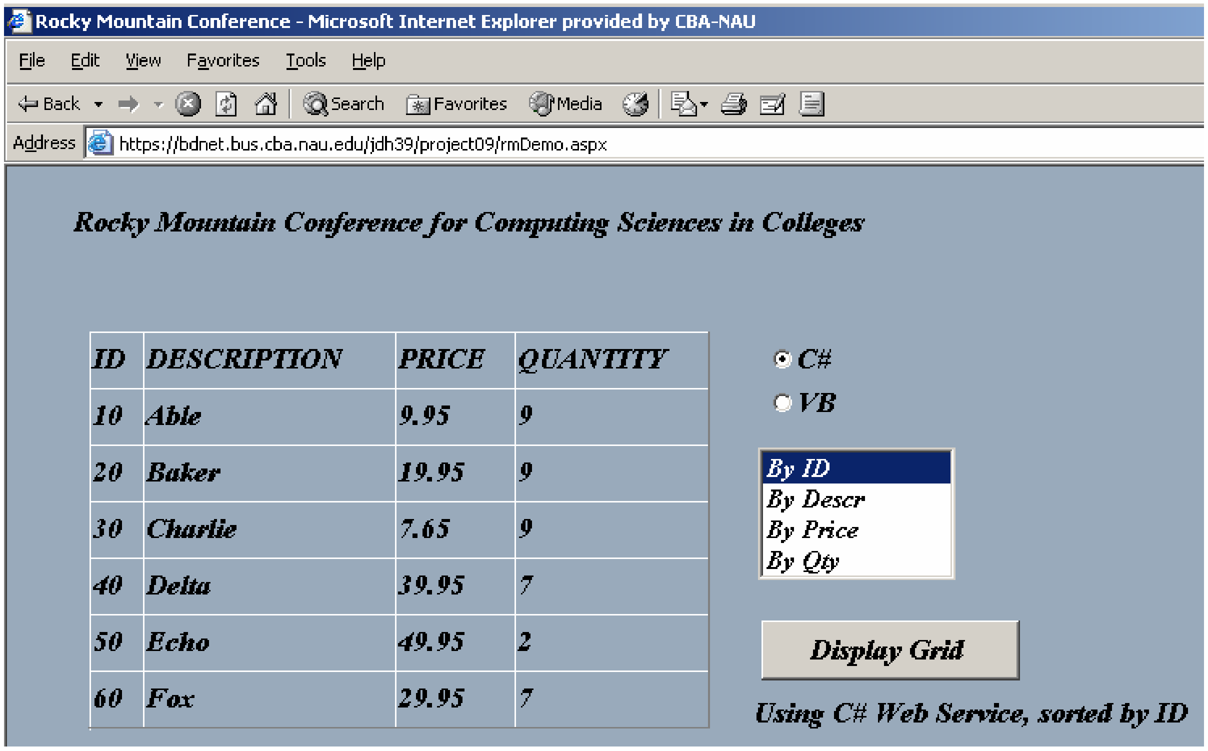
```
Dim dbServiceVB As vbService.VB_Service = New vbService.VB_Service()
Dim dbServiceCS As csService.CS_Service = New csService.CS_Service()
```

The first line of code creates an object dbServiceVB of vbService.VB_Service() type. The second line creates an object dbServiceCS of csService.CS_Service() type. Since the Web Reference was added to the project the referencing of the objects when creating the code is the same as any other object within Visual Studio .NET.

The code example below shows the creation of a DataSet object and then using the getProduct(srtType) method of the server Web service to fill the dataset.

```
Dim ds As New DataSet()
If langType = "C#" Then
   ds = dbServiceCS.getProduct(srtType)
Else
   ds = dbServiceVB.getProduct(srtType)
```

Following are the screen shots from executing a few of the different options. The first example shows a selection of C# for the programming language and to sort by ID.



Rocky Mountain Conference - Microsoft Internet Explorer provided by CBA-NAU

Address https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx

Rocky Mountain Conference for Computing Sciences in Colleges

ID	DESCRIPTION	PRICE	QUANTITY
10	Able	9.95	9
20	Baker	19.95	9
30	Charlie	7.65	9
40	Delta	39.95	7
50	Echo	49.95	2
60	Fox	29.95	7

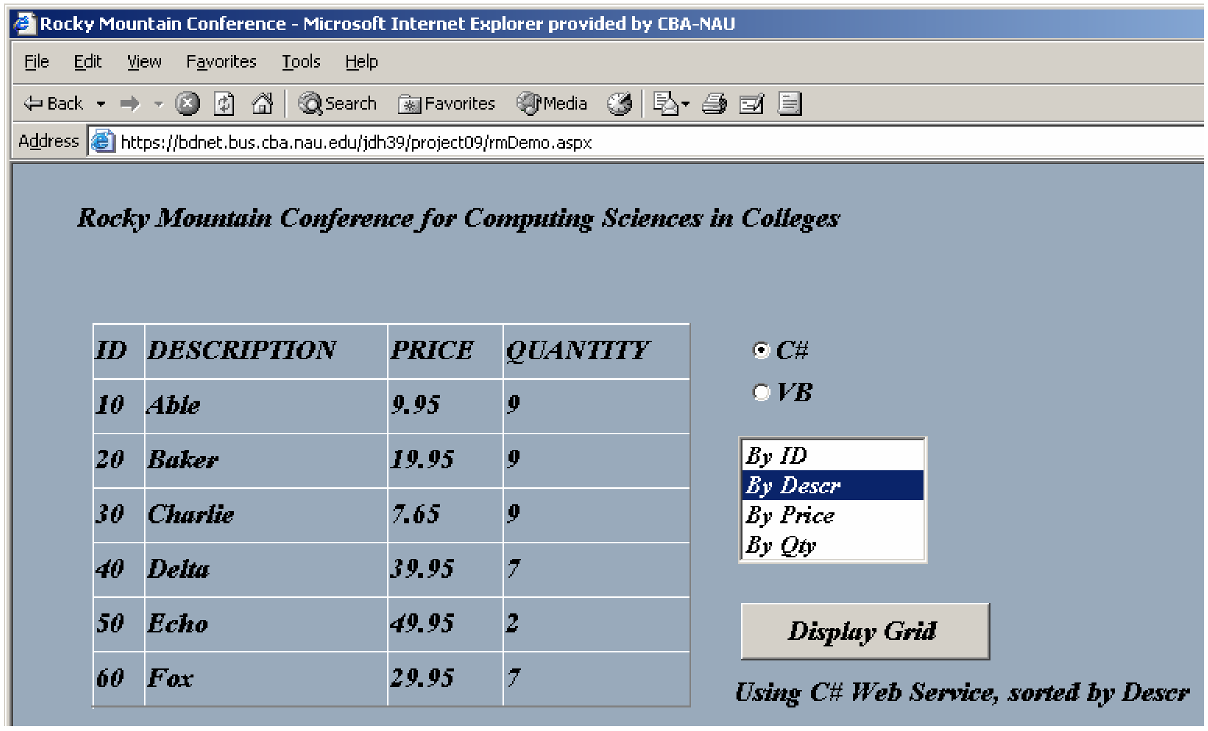
[✓ C#] VB

[✓ By ID] By Descr By Price By Qty

Display Grid

Using C# Web Service, sorted by ID

The next example shows the selection of C# for the language and to sort by description.



Rocky Mountain Conference - Microsoft Internet Explorer provided by CBA-NAU

Address https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx

Rocky Mountain Conference for Computing Sciences in Colleges

ID	DESCRIPTION	PRICE	QUANTITY
10	Able	9.95	9
20	Baker	19.95	9
30	Charlie	7.65	9
40	Delta	39.95	7
50	Echo	49.95	2
60	Fox	29.95	7

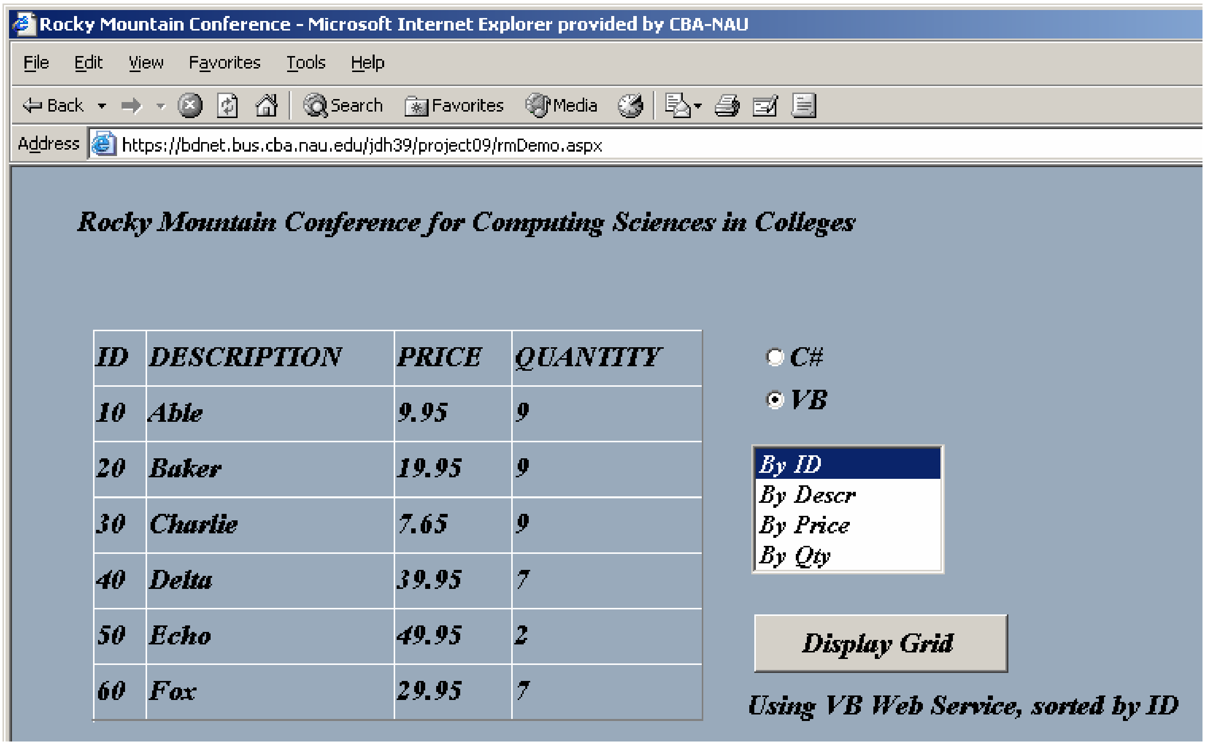
[✓ C#] VB

By ID [✓ By Descr] By Price By Qty

Display Grid

Using C# Web Service, sorted by Descr

The next example shows a selection of VB for the language and to sort the data by ID.


Rocky Mountain Conference - Microsoft Internet Explorer provided by CBA-NAU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx

Rocky Mountain Conference for Computing Sciences in Colleges

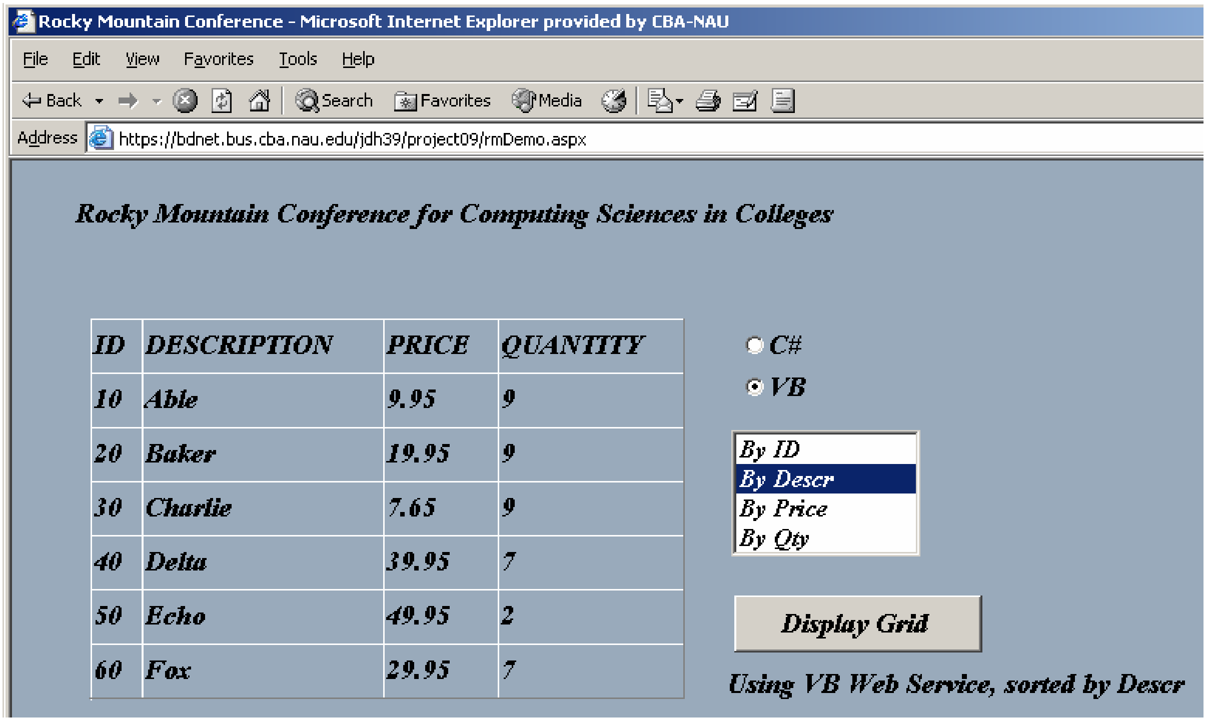
ID	DESCRIPTION	PRICE	QUANTITY
10	Able	9.95	9
20	Baker	19.95	9
30	Charlie	7.65	9
40	Delta	39.95	7
50	Echo	49.95	2
60	Fox	29.95	7

C# [✓ VB]

[✓ By ID]
By Descr
By Price
By Qty

Using VB Web Service, sorted by ID

The last example shows a selection of VB for the language and to sort the data by description.


Rocky Mountain Conference - Microsoft Internet Explorer provided by CBA-NAU
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://bdnet.bus.cba.nau.edu/jdh39/project09/rmDemo.aspx

Rocky Mountain Conference for Computing Sciences in Colleges

ID	DESCRIPTION	PRICE	QUANTITY
10	Able	9.95	9
20	Baker	19.95	9
30	Charlie	7.65	9
40	Delta	39.95	7
50	Echo	49.95	2
60	Fox	29.95	7

C# [✓ VB]

By ID
[✓ By Descr]
By Price
By Qty

Using VB Web Service, sorted by Descr

CONCLUSION

Developing Web Services can be confusing because of the numerous technologies (XML, SOAP, UDDI , and WSDL). Using Visual Studio .NET to create and consume the Web Service makes the process less difficult, but can also hide some of the details. When the Web Service is compiled Visual Studio .NET creates the WDSL document, the SOAP request and response messages for the exposed methods. Visual Studio .NET also creates the Web Discovery Service file that is used to discover Web Services on your server. Developers should study each of these technologies to gain a through understanding of Web Services development.

References

IBM *developerWorks*. New to Web services. March 25, 2004.<http://www-106.ibm.com/developerworks/webservices/newto/>.

Kalata, K., 2003. *Introduction to ASP.NET.* Boston: Thomson: Course Technology.

MSDN Library. XML Web Services Basics. December 2001. March 25, 2004. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/webservbasics.asp>.

W3C. Web Service Activity. March 24, 2004. March 25, 2004. <http://www.w3.org/2002/ws/>.