AN EXPERIENCE IN MIGRATING FROM ASP TO ASP.NET

Mindfire Solutions
www.mindfiresolutions.com

March 2002

Abstract:

This paper discusses the porting of an existing ASP application to new ASP .NET application. It also talks about issues faced in porting of existing ADO-to-ADO .NET technology and suggests appropriate steps that should be used to make the whole process simpler.

Description of the Application

We have developed an application named as *'BUG TRACKING SYSTEM'* in ASP using ADO as the data access technology. The application is used for keeping track of the bugs/defects encountered in specified project in the system. The tool can handle many projects at one time with many users with their own set of pre-defined rights. It generates bug status reports as and when desired, which helps the client in viewing the progress of the project Open/Closed/Work in progress bugs in the form of different charts/tables.

To enable a rich front-end while maintaining platform independence, we have decided to re-architect the system with Web services. The information from web services will feed existing system to generate platform-independent HTML front-end. At the same time, a

native Windows application can access information using these Web services and display them in a Windows-native rich user interface. This re-architecture requires us to migrate the existing Bug Tracking System in ASP with ADO to new ASP.NET with ADO.NET data access technology using ASP.NET Web Services.

The major steps involved are migrating to ASP .NET, implementing Web services and making the new VB .NET user interface.

Migrating to ASP .NET

In layman's language ASP.NET is nothing but the next version of Active Server Pages (ASP); but practically it is more than just the next version; it is a development platform that helps the developers to build enterprise Web Applications. Most importantly it introduces us to the new world of developing XML Web Services.

Here we will only talk about the practical experience of converting ASP to ASP.NET. It's not mandatory that one should know ASP or ASP.NET to understand this article, but a little bit of knowledge will be helpful.

Language Conversion

Syntactically there isn't much difference between ASP and ASP.NET. One can directly run simple ASP pages under .NET without doing anything. But if you change the extension of the ASP page from .asp to .aspx then it requires some changes.
The example below will make things clearer.

Say you have an ASP page by the name of Test1.asp in which code goes like this.

```
<%
        Response.Write "Hello this a test ASP Page"
%>
```

Now if you run this under .NET environment with the same extension you will face no problem, but if you change its extension to Test1.aspx then minimal change which is required is:

```
<%
        Response.Write ("Hello this a test ASPX Page") 'Brackets are compulsory
                                                       ' For all method calls.
%>
```

By doing this you will be able to run your .aspx page.

But this is the beginning. The major change is in the ADO.NET environment. ASP.NET can use conventional ADO and ADO.NET. Porting a simple ASP page which uses ADO to ASP.NET with ADO technology is not a great deal of work. Take a look at the example below:

```
<%
        Dim objDBConn  'Creates ADO Connection Object.
        Set objDBConn = Server.CreateObject ("ADODB.Connection")
        objDBConn.ConnectionString ="datasource=BTS;uid=abc; pwd=xyz;"
        objDBConn.Open
        Dim Rst 'Created ADO Recordset Object.
        Set Rst = Server.CreateObject ("ADODB.Recordset")
        Rst.Open "select first_name, last_name, member_id from members", objDBConn,
        1, 2
        Do while not Rst.EOF
                Response.Write "Member ID: " & Rst ("member_id") & "<br>"
                Response.Write "Member Name: " & Rst ("first_name") & " " & Rst
                ("last_name")
        Loop
        Rst.Close
        Set Rst = Nothing
        ObjDBConn.Close
        Set ObjDBConn = Nothing
%>
```

The above was a conventional ASP Page, which uses ADO as a data access technology. Let's see what all we need to do to make it run under ASP.NET (.aspx) using same ADO technology.

```
<%@Page aspcompat=true Language = VB%> 'Include this Directive at the top
<%
        Dim objDBConn
        objDBConn = Server.CreateObject ("ADODB.Connection") 'Set Removed
        'Parenthesis Added
        objDBConn.Open ("datasource=BTS;uid=abc; pwd=xyz;")
        Dim Rst
        Rst = Server.CreateObject ("ADODB.Recordset") 'Set Removed
        Rst.Open ("select first_name, last_name, member_id from members",
        objDBConn, 1, 2) 'Parenthesis Added
        Do while not Rst.EOF
                'Value Property Added
                Response.Write "Member ID: " & Rst ("member_id").Value & "<br>"
                Response.Write "Member Name: " & Rst ("first_name").Value & " " &_
                Rst ("last_name").Value
        Loop
        Rst.Close
        Rst = Nothing 'Set Removed
        ObjDBConn.Close
        ObjDBConn = Nothing 'Set Removed
%>
```

From the above code, you have seen that how easy it is to convert running asp page with ADO into running ASP.NET Page with ADO.

Note: - While porting your ASP applications to ASP.NET, be sure to keep the following points in mind:

- ? Be familiar with the syntactical changes from VBScript to Visual Basic .NET. Guarding against small syntactical errors will make the process that much smoother.
- ? Add the **aspcompat="true"** attribute to the Page Directive if the ASP page you are porting uses COM components that access the ASP-intrinsic objects. Otherwise this directive is not required.

But this is not what we wanted to do; we are here to see how we will use ADO.NET in ASP.NET for porting.

ADO .NET Conversion

Let's analyze the example below to understand the technicality behind using ADO.NET and also the difference between ADO.NET and ADO.

This example is an ASP.NET application that uses ADO.NET to read records from the database (BTS) as the previous example. This code generates output equivalent to that of the previous example, but has been modified to comply with ASP.NET requirements.

```
<%@ Page Language="VB" Debug="true"%>
<%@Import Namespace="System.Data"%>
<%@Import Namespace="System.Data.SqlClient"%>

<!
This example uses ADO.NET to read records from a database and print two
fields from all returned records to an active server page. The database
is located on the local SQL Server.
>
<Html>
<%
        Dim Conn As SqlConnection
        Dim StrConn as String
        Dim Cmd As SqlDataAdapter
        Dim dtSet As DataSet
        Dim dtTable As DataTable
        Dim dtRow as DataRow
        Dim iLoop As Integer
        Dim iNRows as Integer
```

```
        Dim sSQL As String
        StrConn = “datasource=BTS;uid=abc;pwd=xyz;”
        Conn = new SqlConnection (StrConn)
        Conn.Open ()
        sSQL = "SELECT * FROM Members;"

        ' Create a connection to the data source.

        ' Create a Command object with the SQL statement.
        Cmd = New SqlDataAdapter (sSQL, Conn)

        ' Fill a DataSet with data returned from the database.
        dtSet = New DataSet
        Cmd.Fill (dtSet)

        ' Create a new DataTable object and assign to it
        ' the new table in the Tables collection.
        dtTable = New DataTable
        dtTable = dtSet.Tables (0)
        ' Find how many rows are in the Rows Collection
        ' of the new DataTable object.
        iNRows = dtTable.Rows.Count
        If iNRows = 0 then
                Response.Write ("<p>No records.</p>")
        Else
                Response.Write ("<p>" & Cstr (iNRows) & " records found. </p>")
                For iLoop = 0 to iNRows - 1
                        dtRow = dtTable.Rows (iLoop)
                        Response.Write (dtRow ("first_name") _
                        & " " & dtRow ("last_name") & "<br>")
                Next iLoop
        End If
        Response.Write ("<p>End of data.</p>")
%>
</html>
```

Let’s see what actually the example above does.

It creates an ADO.NET DataSet object, which in this case contains one data table that is used in much the same way as an ADO RecordSet. Note that a DataSet can consist of collections of one or more DataTables, so it is not correct to think of an ADO.NET DataSet as the equivalent of an ADO RecordSet.

In order to use ADO.NET, the **System.Data** and **System.Data.SqlClient** namespaces must be imported for SQL Server Database.

By viewing the above example you must have come to know that DataSet object replaces our old Recordset object of ADO fame in ADO.NET.

There is one more important object for reading Data from database table i.e. SqlDataReader object.

Below is the code snippet that allows you to understand how it works.

```
<%…
…
…
Dim Conn as SqlConnection
Dim StrConn as String
Dim Dr as SqlDataReader
Dim StrSql as String
Dim Cmd as SqlCommand
StrConn = “database=BTS; uid=abc; password=xyz;”
Conn = new SqlConnection (StrConn)
Conn.Open ()
StrSql = “select first_name, last_name from Members”
Cmd = new SqlCommand (StrSql, Conn)
Dr = Cmd.ExecuteReader ()   ‘Return type is SqlDataReader
Do while Dr.Read()
        Response.write (dr (“first_name”) & “ “ & dr (“last_name”) & “<br\>”)
Loop
%>
```

Note: - SqlDataReader object can be used in the same way as that of ADO Recordset object but we cannot use another SqlDataReader object with the same connection object before closing the previous SqlDataReader object. So, while using SqlDataReader object one should be very careful about closing the first object before using the second. Practically, we can use both “ SqlDataReader and DataSet” objects together for the development of ASP.NET application.

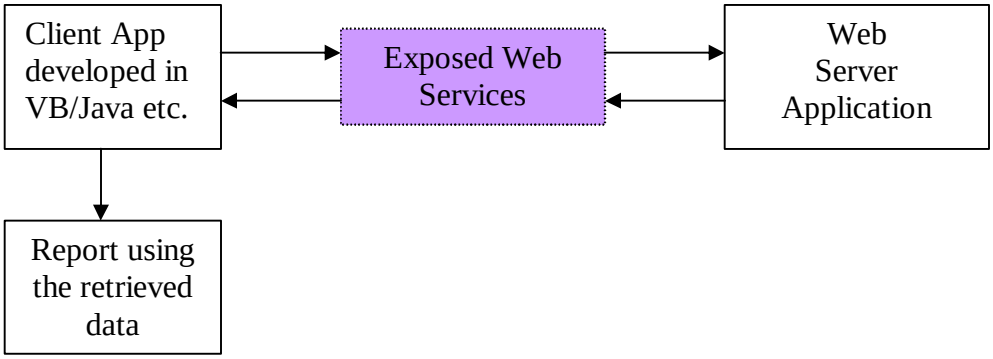
As we have learnt a lot about ASP.NET and ADO.NET, let’s move ahead to one of the most important features of ASP.NET i.e. ASP.NET Web Services or XML Web Services.

Re-architecting the application around Web Service

We will re-design our application around Web Service using ASP.NET Web Services, which exposes some methods to the client application to retrieve data or generate, reports e.g. Bug Status Report. Here client application could be any application developed using any platform such as VB, JAVA, VC or for that matter applications created under UNIX or LINUX OS etc.

Client application will just use the exposed Web Services and will get the results from the same and can manipulate it accordingly whether in the form of reports or any kind of analysis.

This figure will make things much clearer.



In the figure above client application makes a call to Web Service, in turn Web Service calls the web application to retrieve the data and make the results available to the Client application. Client can then use the result to generate the reports

XML Web Services

Technically speaking, a web service is "a component of programmable application logic that can be accessed using standard web protocols". That is, it's quite similar to the components we have been using till date, but lets us access all its functionality via the Internet. In principle, anyone who can browse the Web can see and use a web service.

Let's see what all we need to do to create a Web Service, determine its content and make it available for others.

Implementation

The following is one of the services, which we have used to retrieve the User Details who has just logged into the system from SQL Server Database table using ADO.NET

```
<%@ WebService Language="VB" Class="UserDetails"%>
Imports System
Imports System.Xml.Serialization
Imports System.Web.Services
Imports System.Diagnostics
Imports System.Data
Imports System.Data.SqlClient

Public Class UserProperties
        Public FirstName as String
```

```
        Public SecondName as String
        Public UserID as String
        Public ErrorOccured as String
End Class

Public Class UserDetails:Inherits WebService
        <WebMethod ()> Public Function GetUserDetails (ByVal uname as String,
ByVal password as String) As UserProperties
                Return IsUserExist (uname, password)
        End Function

        Public Function IsUserExist (ByVal username as String, ByVal pwd as String) as
UserProperties
                Dim strConn as String
                Dim Conn as SqlConnection
                Dim cmd as SqlCommand
                Dim dr As SqlDataReader
                Dim strSql as String
                Dim userDet as new UserProperties

                strConn = "server=Rocky; database=Bms; uid=abc; password=xyz;
Connect Timeout=300"
                Conn = New SqlConnection (strConn)
                Conn.Open ()

                strSql = "select member_id, first_name, last_name from Members where
email = "'& username &"' and password = "'& pwd &""""
                cmd = New SqlCommand (strSql, Conn)
                dr = cmd. ExecuteReader ()
                If dr.Read () then
                        userDet.FirstName = dr ("first_name")
                        userDet.SecondName = dr ("last_name")
                        userDet.UserID = dr ("member_id")
                        dr.Close ()
                        conn.Close ()
                        Return userDet
                Else
                        userDet.ErrorOccured = "User Does Not Exists"
                        Return userDet
                End if
        End Function
End Class
```

Note: - To determine the file to be Web Service File save it with an extension of .ASMX

Save the above file as UserDetails.asmx in your working web folder and open it in your web browser. You will see the following figure:

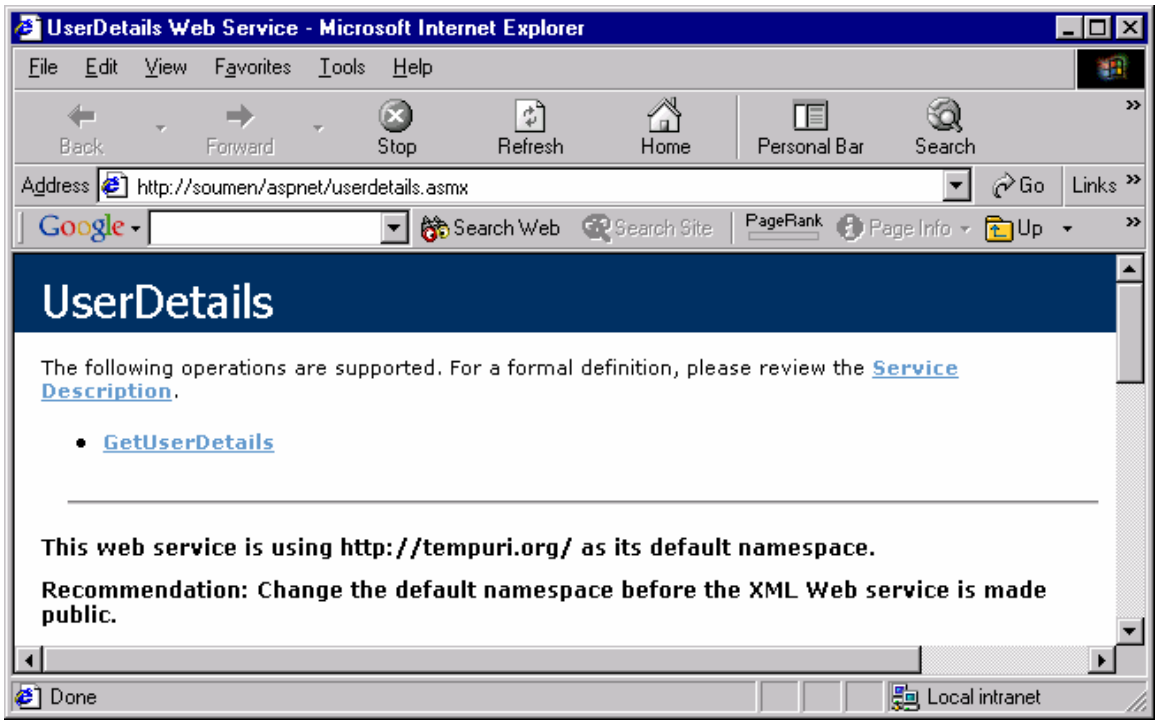


Fig (a)

Now click on the exposed Method GetUserDetails in your browser.

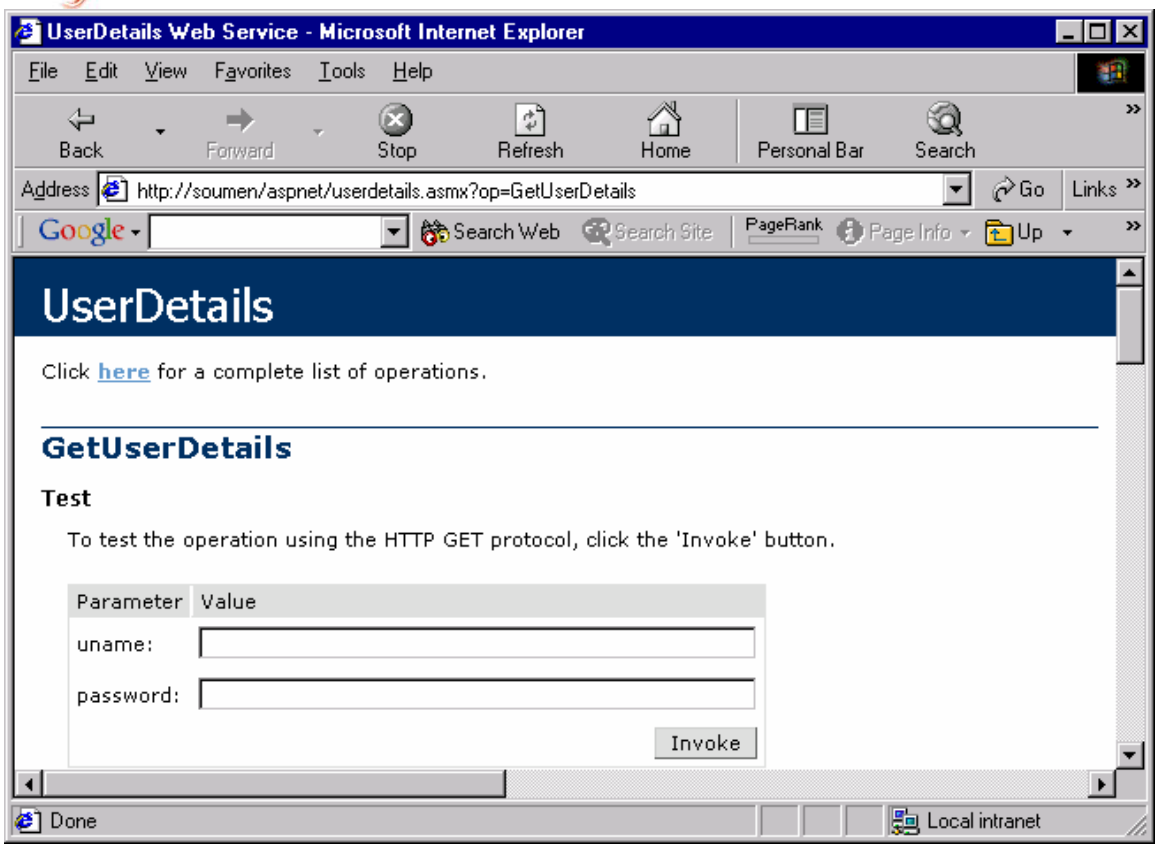


Fig (b)

You will see that in above figure it will ask you for the username and password (because of the parameters in the exposed web method GetUserDetails i.e. uname and password). Enter the required user name and password and click on invoke button, you will see the following screen.

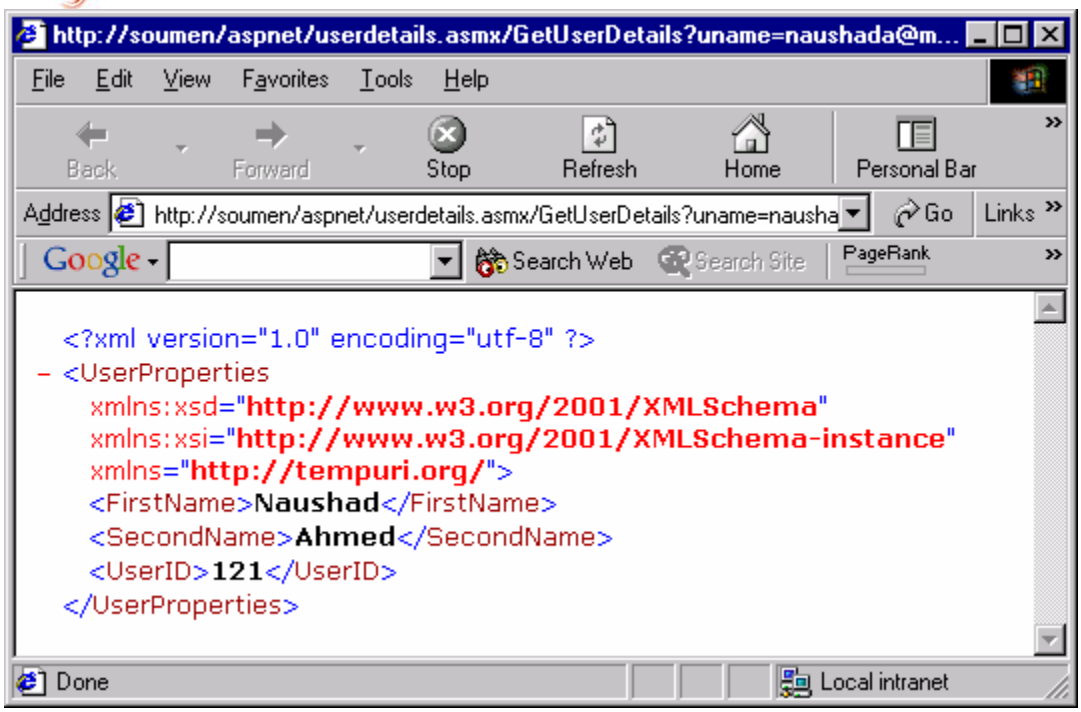


Fig (c)

This screen shows you the required user details such as First name, Second name and user id in an XML Format, where each of the User Details i.e. First name, Second name and user id represents one of the XML Tags, like <Firstname>John</Firstname> and so on.

Now, you have developed a Web service the next step is how to use this web service in your client application or how your clients/partners will come to know which all Web Services you are exposing and what all they do.

Let's see what all is required in Consuming or using this web service.

Making a proxy

To consume a web service, we must first create an interface that will allow the consumer to see all of the web-callable methods and properties exposed by the Web Service. We refer to this interface as a Web Service proxy.

In order to make use of a Web Service from an ASP.NET Page or VB.NET Application, our proxy must be created and compiled appropriately. Creating a proxy to a Web Service using the .NET Framework tools is very straightforward. These tools make use of WSDL (Web Services Description Language) to create a proxy, built in the language of your choice.

Building a proxy is a two-step process:

- ? Generate the proxy source code
- ? Compile the proxy into a run-time lib

Navigate to the directory in which your UserDetails.asmx file resides, and execute the following statement at command prompt in a single line:

> wsdl /l:vb /o:UserDetailsProxy.vb http://localhost/5040/wsFolder/UserDetails.asmx?WSDL /n:UserDetailsService

This will create a UserDetailsProxy.vb

Now let's compile our new proxy code – note that this statement must be entered in full on a single line:

> vbc /out:UserDetailsProxy.dll /t:library /r:system.web.dll, system.dll,system.xml.dll,system.web.services.dll,system.data.dll UserDetailsProxy.vb

This will create a UserDetailsProxy.Dll file. Save this file under bin directory of your application root.

Create the UserDetails.aspx file in your test directory, and enter the following code:

```
<@% Page Language="vb" Debug="true" %>
<%@ Import namespace="UserDetailsService" %>
<script language="vb" runat="server" >
        Private Sub RetreiveInfo (ByVal sender as System.Object, ByVal e as
        System.EventArgs)
                Dim ws as new UserDetailsService.UserDetails ()
                Dim userdet As New soumen.UserProperties ()
                userdet = ws.GetUserDetails (textUname.Text, textPwd.Text)
                lblUsername.Text = userdet.FirstName
                lblUsername2.Text = userdet.SecondName
                lblUserID.Text = userdet.Userid
        End sub
        </script>
        <Html>
                <Body>
                <Form id="Form1" method="post" runat="server" >
                        <asp:TextBox id="textUname" runat="server"></asp:TextBox>
                        <asp:TextBox id="textPwd" runat="server"></asp:TextBox>
                        <asp:Button id="Button1" runat="server" Text="Submit"
                        Onclick="RetrieveInfo"></asp:Button><br \>
                        <asp:Label id="lblUsername" runat="server"></asp:Label>
                        <asp:Label id="lblUsername2" runat="server"></asp:Label>
                        <asp:Label id="lblUserID" runat="server"></asp:Label>
```

```
        </Form>
        </Body>
</Html>
```

Save the above file and you are through in using the Web Service you created, inside your ASP.NET page.

Consuming the Web Service

For consuming the same Web Service in your VB.NET Application, just add the UserDetailsProxy.dll in your references from Solution Explorer and you will be able to use that DLL like any other DLL.

Or if you know which site is providing you this Web Service than you can simply add the Web reference of that site from your Solution Explorer in VB.NET e.g. http://www.testsite.com/UserDetails.asmx?WSDL. Add this web reference from Web reference dialog box in VB.NET. By doing this you will be able to use the Web Service in the same way as you did above.

Last but not least, how to make your partner/client know about the Web Service(s) you are exposing.

Web Service Discovery

As you begin to build Web Service-integrated applications, it will become increasingly important to locate services that provide the functions you need. Universal Description, Discovery and Integration (UDDI) services allow you to do this.

For more information about UDDI visit the site www.uddi.org.

The UDDI service (http://uddi.microsoft.com) lets businesses register themselves and list their existing Web Services at no charge. Anyone can browse and search the UDDI database for a service that may suit his or her needs. WSDL is a key component of the UDDI project. Using http://uddi.microsoft.com/visualstudio/ , you can search for Businesses that provide Web Services, select the WSDL appropriately, and build your proxies.

For more information about how to register at http://uddi.microsoft.com please visit the following link: Registering yourself with UDDI

There is one more site which does the same like Microsoft UDDI site i.e. www.quddi.com.

Registering your web service through this site is much simpler than registering it from Microsoft's UDDI site. This site only takes the location of your Web Service (e.g.

http://your-ip-address/testFolder/UserDetails.asmx?WSDL) with some short description about the Web Service you are exposing and the Name by which you will recognize your Web Service.

After registering your Web Service on this site just open your VB.NET Application, go to Solution Explorer, click on References, Click on add Web Reference you will Add Web Reference Dialog Box, with Business Name Text Box just write ***EraServer.NET*** in this text box and click Search. You will see all the Web Services, which are registered on this site, scroll down and click on the Web Service you've just created, click on the Add Web reference button at the bottom of the dialog box. Your Web Reference is ready now you can use it as you had used it earlier.

Conclusion

Porting of ASP-to-ASP .NET may be easier than one may believe, but proper understanding of why this porting is required makes the port not only possible but also highly recommended.

In this article we have seen a simple example of porting an ASP application to ASP .NET application, ADO-to-ADO .NET data access technology and most important the use of ASP .NET Web Services.
For further study, look into the following resources:

1) Microsoft ASP .NET Quick Start Tutorial
2) UDDI
3) Registering yourself with UDDI
4) ADO .NET for ADO Programmer
5) Web Service Description using UDDI

Mindfire Solutions is an off-shore software services company in India. Mindfire possesses expertise in multiple platforms, and has built a strong track record of delivery. Mindfire passionately believes in the science of porting.

We have developed specialized techniques to make porting efficient and smooth, and to solve the issues specific to porting. We offer core development and QA/testing services for your porting requirements, as well as complete life-cycle support for porting.

If you want to explore the potential of porting, please drop us an email at info@mindfiresolutions.com. We will be glad to help you.

To know more about Mindfire Solutions, please visit us on www.mindfiresolutions.com