ASP/COM+ Web Development Using Team Developer

Unify Corporation

Table of Contents

Abstract

ASP (Active Server Pages) based Web solutions have become popular as it is easy to learn, provides good performance, scalability and flexibility and is backed by Microsoft. For this reason, Unify has decided to provide developers a route to Web based solutions that are based upon this very popular server side scripting language.

Web Extensions, an alternative Unify Web based solution already exists and is based upon leveraging existing SAL interface and business logic. This paper is not about Web Extensions; if you need more information on this topic please refer to the online documentation or the Unify Web site at www.unify.com.

Currently, it is possible to build and deploy fully scalable COM servers in Team Developer using the COM wizard. You now have the ability to debug COM Servers that are called by non-Team Developer clients such as VB or ASP. With Team Developer, an individual can quickly start building ASP/COM solutions that will combine the simplicity of ASP with the productivity of Team Developer. This white paper will show how to approach constructing such solutions.

General Overview of ASP

ASP is one of the most commonly used Web technologies around (at least in the Microsoft environment). You don't have to browse too many Web pages to realize that a good number of them end with the .asp extension. A principal reason for ASP's popularity is that ASP allows developers to execute code inline within a Web page using an easily understood scripting language such as VBScript or Jscript (Microsoft's implementation of JavaScript).

ASP is not a standalone technology. It is a server-side scripting technology that allows you to generate dynamic, interactive Web pages. With ASP you can read information from an HTTP request, send an HTTP response back to a Web client, store user information, and detect browser capabilities of the Web client.

In a simple scenario - an ASP page is typically an ASCII file that contains HTML and Script (VBScript or Jscript) although in more complex cases an ASP based solution may hold together Web applications containing HTML, Jscript/VBScript, SQL, ADO and COM/COM+ components amongst others.

It is not the intention of this White Paper to give a detailed overview of VBScript/Jscript. A basic familiarity with these scripting languages along with the fundamentals of HTML forms will be assumed. Also assumed is familiarity with the process of creating/using COM objects within Team Developer. More information on this latter process can be found in the Team Developer online documentation and samples and other White Papers on Unify's Web site.

IIS (Internet Information Server) is Microsoft's Web Server. Web Servers typically 'serve' HTML to browsers (clients) upon receipt of an HTTP request. When IIS receives a request for a file with the extension '.asp' it executes the scripts inside that file on the server, generating a pure HTML page and returning it to the browser (client) which made the request.

ASP supports both Visual Basic Script (VBScript) and Microsoft's own implementation of JavaScript (Jscript). There are also third party tools that allow integration of other popular scripting languages, such as Perl, into ASP solutions.

To make implementation of ASP solutions as simple as possible, ASP contains the following five built-in objects:

- The **Server** object allows individuals to manage server resources. One of the most commonly used methods it supports is the CreateObject( ) method, which lets you create an instance of a COM object.

- The **Application** object stores and manages information at the application level. This means that one application object variable can be used for ALL user sessions throughout a Web application.

- The **Session** object stores and manages information specific to a user session. Typically it allows a Web application to remember information about a particular user's session.
- The **Request** object allows you to collect information that is sent through an HTTP request.

- The **Response** object allows you to send information back to the client through HTTP.

Inside an ASP page the tags '<%' and '%>' are used to indicate the beginning and end of the script which the Web Server should execute on the server instead of merely rendering and returning HTML.

At this point an example of an ASP page might be most illustrative. Listing 2.1 is an ASP page that retrieves product styles and prices from the PRODUCT table of the ISLAND database using ADO recordset object.

Listing 2.1 Retrieving Product Styles & Prices & displaying them in an HTML Table

```
<%@ Language=VBScript %>
<html>
<head>
<title>ASP ADO Demo</title>
<h2>Products from ISLAND Database</h2>
<h3>Using ADO recordSet object</h3>
<% dim rsProducts 'ADO recordset for holding authors.%>
<% set rsProducts = server.CreateObject("ADODB.Recordset")%>
<table width =300 border=2>
<tr><td><b>Product Style</b></td><td><b>Price</b></td></tr>
<%
'Set the ActiveConnection property
rsProducts.ActiveConnection = "Provider=SQLBASEOLEDB;" & _
"Data Source=ISLAND; User ID=SYSADM; Password=SYSADM;"
'Open the recordset.
rsProducts.Open "Select style, price from product"
'Loop through the recordset and display the results in a table.
do until rsProducts.EOF
Response.Write "<tr><td>" & rsProducts("style") & _
"</td><td>" & rsProducts("price") & "</td></tr>"
rsProducts.MoveNext
loop
'Close the recordset and release resources.
rsProducts.Close
set rsProducts = nothing
%>
</table>
</body>
</html>
```

The above script could be placed in a file named ‘ADODemo.asp’ on a PC containing IIS. If it is placed in the default Web server directory, (e.g. C:\Inetpub\wwwroot), then the results of its execution can be seen by entering the URL ‘http://localhost/ADODemo.asp’.

This simple script shows how ASP scripting can be placed within an HTML page in order to achieve a dynamic page that is based upon the contents of a database table; changes to the data within the table will be reflected in the HTML sent back to the browser that made the request. If the user who entered the URL views the source they will NOT see the ASP script but will see the HTML which has been generated as a result of the execution of that script. Thus they would see something like the listing shown below.

Listing 2.2 HTML generated by above ASP script

```
<html>
<head>
<title>ASP ADO Demo</title>
<h2>Products from ISLAND Database</h2>
<h3>Using ADO recordSet object</h3>
<table width =300 border=2>
<tr><td><b>Product Style</b></td><td><b>Price</b></td></tr>
<tr><td>Five-O Tourist Shirt</td><td>8.25</td></tr><tr><td>Hang-Ten Surf
Pants</td><td>15.5</td></tr><tr><td>Hawaiian Mu
Mu</td><td>11.25</td></tr><tr><td>Kona Kha-
kis</td><td>14.6</td></tr><tr><td>Lanai Linge-
rie</td><td>9.5</td></tr><tr><td>Maui Tank
Top</td><td>6.5</td></tr><tr><td>North Shore Swimsuit</td><td>12.25</td></tr>
</table>
</body>
</html>
```

ASP and COM

ASP Scripting as shown above is fine for relatively small and straightforward Web pages. In real world scenarios however, solutions that are based wholly on scripting start to become very large and cumbersome. The difficulties of maintaining such code are the very limited debugging capabilities that accompany ASP.

For this reason COM (Microsoft's Component Object Model) fits in very well with ASP based solutions. COM offers the opportunity for implementing 'black box' solutions; the user of the COM object only needs to be aware of its Interfaces, Methods and Properties.

The major reasons to use COM components in conjunction with ASP are as follows.

- **Reusability:** It is difficult to package an ASP script so that it can be used over and over again in many different pages. Components allow more effective management and reuse of certain functionality, e.g. database access.

- **Security:** It may be important that you keep source code hidden from potential threats. Placing such code on a COM component will assist in this requirement.

- **Scalability:** The COM components themselves are not required to reside on the same machine as the Web Server. When Microsoft released ASP, they also released Microsoft Transaction Server (MTS) that provides for transaction management for COM based applications.

- **Interoperability:** Extending the point above, it is possible to even transcend object systems and access objects that adhere to other standards such as CORBA; third party products exist which facilitate this.

- **Ease of development:** COM objects can be written in Team Developer, thus you protect the investment you have made in your existing skill set. Also, if you have ever tried to debug an ASP page you will quickly realize the benefits of being able to use your favorite debugger. By building your COM objects in Team Developer you can easily debug your COM object dynamically even when it is invoked from an ASP page. This brings the power of the Team Developer IDE to ASP based development.

For the reasons outlined above, it is highly recommended that COM objects form part of any significant ASP based Web solution. For developers with experience of the SAL language and the basics of COM, Team Developer is perfectly suited to this task.

Let us now look at another simple ASP solution but this time we will use a COM object written in Team Developer to implement our required functionality. To keep this as simple as possible, we start with a Web page containing an HTML form (**ASP_Add.htm**) containing two input fields. When submitted, this form will invoke an ASP page (**ASP_Add.asp**). The ASP page will instantiate a COM object (**ASP_COM.ASPTestObject**) and from this object the method/function **AddStr()** shall be invoked. The method merely concatenates whatever values are placed in the two input fields and returns the result.

In practice, it is often the case that the initial HTML form and the ASP page are combined, but here I am simplifying the matter as much as possible in order to dwell more on the process of how an ASP script can use COM objects. Later examples will do away with the need to have a separate HTML page.

Listing 3.1 HTML Page that will invoke an ASP script upon submission of form (ASP_Add.htm)

```
<html>
<head>
<title>
Unify Team Developer COM Sample : Stage 1 HTML
</title>
</head>
<body>
<form method="POST" action="ASP_Add.asp">
<input type="text" name="value1">
<input type="text" name="value2">
<input type="submit" value="Append Values">
</form>
</body>
</html>
```

Listing 3.2 ASP page invoked upon submission of above <form> (ASP_Add.asp)

```
<%
'----Code within these tags runs at server
'----Recommended that Option Explicit is set
Option Explicit
Dim n1, n2, nRes, objCTDAdd
n1 = Request.Form("value1")
n2 = Request.Form("value2")
%>
<html>
<head>
<title>Process the form in ASP_Add.htm</title>
</head>
<body>
<h1>First field contains <%=n1%></h1>
<h1>Second field contains <%=n2%></h1>
```

```
<p>
<%
'Instantiate object whose methods / properties we want to invoke / set
Set objCTDAdd = Server.CreateObject("ASP_COM.ASPTestObject")
'Call the AddStr() method which will carry out concatenation operation
nRes = objCTDAdd.AddStr (n1, n2)
%>
<!--output result-->
<h1>Have called AddStr method & result is <%=nRes%></h1>
<%
'Clean up
Set objCTDAdd = Nothing
%>
</body>
</html>
```

Using Team Developer, we can quickly create a CoClass and Interface (ASPTestObject and IASPTestObject respectively). To the Interface we add a function / method AddStr( ) the SAL code of which is displayed here.

Listing 3.3 COM Object in Team Developer

```
Function: AddStr
Description:
Attributes
Dispatch Number: 1
Help String:
Help Context: 0
Returns
String:
Parameters
String: p_sValue1
String: p_sValue2
Static Variables
Local variables
String: sRetValue
Actions
Set sRetValue = p_sValue1 || p_sValue2
Return sRetValue
```

NB: ALL source files referred to accompany this white paper.

For any data source/provider that has an OLE DB provider, Team Developer code can provide easily programmable access which would be familiar to anyone with experience of SAL/SQLWindows.

Instead of using SqlConnect( ) to return a Sql Handle to the datasource, the developer would use SqlCreateSession( ) and SqlCreateStatement( ); instead of using SqlDisconnect( ) to disconnect from the datasource they would use SqlFreeSession(

). For most standard SQL operations all the intervening Sql…( ) functions would be the same as are used when connecting natively (i.e. SqlPrepare( ), SqlExecute( ), SqlFetNext( ) etc).

With such a simple migration to using the OLE DB Provider to any data source (remember the data source need not be restricted to relational databases) the Team Developer programmer can rapidly utilize OLE DB within an ASP/COM based Web solution. The simplicity with which this can be achieved is shown below. In this example I have used the COM Class Wizard within Team Developer to create a CoClass (OLEDB) with a single Interface (IOLEDB). The interface contains six methods (Connect, Disconnect, NextProduct, Price, SelectProducts and Style).

Listing 4.1 Methods in Interface IOLEDB (ASP_COM_OLEDB.app)

```
Function: Connect
Description:
Attributes
Dispatch Number: 0
Help String:
Help Context: 0
Returns
Boolean:
Parameters
String: psDataSource
String: psUser
String: psPassword
Static Variables
Local variables
Boolean: bRet
Actions
Set SqlDatabase = psDataSource
Set SqlUser = psUser
Set SqlPassword = psPassword
Set bRet = SqlCreateSession( iv_hSession, “Provider = SQLBASEOLEDB;” )
Set bRet = SqlCreateStatement( iv_hSession, iv_hSql )
Return bRet
Function: Disconnect
Description:
Attributes
Dispatch Number: 1
Help String:
Help Context: 0
Returns
Boolean:
Parameters
Static Variables
Local variables
Boolean: bRet
Actions
```

Set bRet = SqlFreeSession(iv_hSession)
Function: NextProduct
Description:
Attributes
Dispatch Number: 3
Help String:
Help Context: 0
Returns
Boolean:
Parameters
Static Variables
Local variables
Boolean: bRet
Number: nInd
Actions
When SqlError
Set bRet = FALSE
Set bRet = SqlFetchNext(iv_hSql, nInd)
If bRet
If nInd != FETCH_EOF
Set iv_sPrice = SalNumberToStrX(iv_nPrice, 2)
Return bRet
Function: Price
Description:
Attributes
Dispatch Number: 5
Help String:
Help Context: 0
Returns
String:
Parameters
Static Variables
Local variables
Actions
Return iv_sPrice
Function: SelectProducts
Description:
Attributes
Dispatch Number: 2
Help String:
Help Context: 0
Returns
Boolean:
Parameters
Static Variables
Local variables
Boolean: bRet
String: sSql
Actions
Set sSql = “SELECT style, price INTO :iv_sStyle, :iv_nPrice FROM product”
When SqlError

```
Set bRet = FALSE
Set bRet = SqlPrepareAndExecute( iv_hSql, sSql )
Return bRet
Function: Style
Description:
Attributes
Dispatch Number: 4
Help String:
Help Context: 0
Returns
String:
Parameters
Static Variables
Local variables
Actions
Return iv_sStyle
```

The Connect method illustrates how easy it is to connect to a data source using OLE DB in Team Developer. In this instance we are using a SQLBase OLEDB Provider, but by changing the Connect String parameter passed to the function SqlCreateSession(…) it could be any other OLE DB provider. The Disconnect method is worth noting for the presence of SqlFreeSession(…). Apart from these two methods - the functionality used should be instantly well-known to any developer familiar with using Sql… functions in Team Developer/SQLWindows.

Listing 4.2 Instance Variables in Interface IOLEDB (ASP_COM_OLEDB.app)

```
Interface: IOLEDB
        Description:
        Attributes
        Derived From
        Class Variables
        Instance Variables
                Boolean: iv_bConnected
                String: iv_sDataSource
                Session Handle: iv_hSession
                Sql Handle: iv_hSql
                String: iv_sStyle
                Number: iv_nPrice
                String: iv_sPrice
```

The simplicity with which the above COM Server can be invoked from ASP is displayed here.

Listing 4.3 ASP page using COM Server ASP_COM_OLEDB.OLEDB (ASP_OLEDB.asp)

```
<%
'----Code within these tags runs at server
```

```
‘----Recommended that Option Explicit is set
Option Explicit
Dim DS, Usr, Pwd, bRet, objOLEDB
Dim sStyle, nPrice
DS = Request.Form(“DataSource”)
Usr = Request.Form(“User”)
Pwd = Request.Form(“Password”)
%>
<html>
<head>
<title>Process the form in ASP_OLEDB.htm</title>
</head>
<body>
‘Below we merely confirm parameters passed from form in ASP_OLEDB.htm
<h1>DataSource to connect to <%=DS%></h1>
<h1>User name used <%=Usr%></h1>
        <h1>Password used <%=Pwd%></h1>
<p>
<%
‘Instantiate object whose methods / properties we want to invoke / set
Set objOLEDB = Server.CreateObject(“ASP_COM_OLEDB.OLEDB”)
‘Connect
bRet = objOLEDB.Connect( DS, Usr, Pwd )
‘Select Style & Price from TABLE Product
bRet = objOLEDB.SelectProducts
%>
<table width=300 border=2>
<tr><th>Style</th><th>Price</th></tr>
<%
‘Draw table row for EVERY row of data returned from COM Server :
‘using NextProduct(), Style() & Price() methods
do until bRet = FALSE
bRet = objOLEDB.NextProduct
If bRet = TRUE Then
sStyle = objOLEDB.Style
nPrice = objOLEDB.Price
Response.Write “<tr><td>” & sStyle & “</td><td>” & nPrice & “</td></tr>”
End If
loop
%>
</table>
<%
‘Clean up
Set objOLEDB = Nothing
%>
</body>
</html>
```

It has to be pointed out that the developer could utilize the ADO object model directly from their ASP script as was done in listing 2.1 above, but this requires familiarity with the ADO object model. The above example shows how easy it is for a Team Developer programmer to implement ASP, OLEDB based solutions using their existing skill set.

Finally, if the developer really wishes to persist with using the ADO object model - there is nothing stopping them doing so within their COM object as this would give the advantages of encapsulating as much logic as possible in COM objects which were outlined in Section 3 above.

To really increase the scalability of ASP based solutions, Microsoft provides a tool/API service to provide component and transaction management called Microsoft Transaction Server (MTS). MTS provides for transaction management for COM-based components and is an effective component management tool that provides server and client migration mechanism as well as component registration services. The use of ASP and MTS is greatly enriched by the inclusion of OLE DB/ADO that provides for generic data access regardless of structure.

COM+ Debugging and ASP

With Team Developer it is possible to actively debug a Team Developer written COM server no matter from where it has been called. Thus, a Team Developer COM server called from ASP or VB can be as easily debugged as one called from another Team Developer client.

To invoke debug mode all one has to do is open the source code for a COM object which has already been built and registered and then invoke the Debug / Go menu (alternatively use function key F7). Set a break point at the appropriate point in any of the Interface functions and then return to the client application that will use the COM object and run it; if it is an ASP script invoke the script using http in browser.

If the client invokes a server method which has a breakpoint set then by returning to Team Developer the developer can step through the SAL code and access all the existing Debug tools. The major point to remember is that running the server (Debug / Go or F7) is a prerequisite to being able to debug; this step is often forgotten.

Requirements

- Team Developer 3.1 or above
- Microsoft Internet Information Server

About Unify

Unify is a global provider of software development technology and solutions that helps IT customers participate in Service-Oriented Architecture (SOA). Unify's productive and re-liable development tools, migration solutions and databases enable organizations to build and modernize business essential applications for SOA. Composer for Lotus Notes offers a complete, like for like, production to production migration solution for Lotus Notes applications. Unify's award-winning NXJ Developer enables IT teams to be extremely productive, learn new technologies fast and deliver Web services-based applications on time and on budget. The Team Developer, SQLBase, DataServer, ACCELL and VISION product families enable cross-platform rapid development on Java/J2EE, Linux or Windows. Unify has a rich heritage in delivering rich, cost-effective technologies to its thousands of IT customers and ISV, VAR and distributor partners. Unify is headquartered in Sacramento, Calif., and can be reached at (916) 928-6400 or by visiting www.unify.com.

Unify Corporation
2101 Arena Blvd., Suite 100
Sacramento, CA 95834
USA
Phone: 1.916.928.6400
Toll Free: 1.800.468.6439
Fax: 1.916.928.6404
Munich: +49 8 115 55430
United Kingdom: +44 (0)1753 245 510
France: +33 (0)1 34 58 28 30