



whitepaper

Creating Web Apps The Hard Way

By Bob Cusick

© 2009 Clickware, Inc. All Rights Reserved.

Duplication of any part of the document in whole or in part is prohibited without prior permission.

Getting permission to reprint this content or include it on your site is easy – just send us a request via email at sales@clickware.com.

Clickware and the Clickware logo are trademarks or registered trademarks of Clickware, Inc. in the US and other countries. All other trademarks are the property of their respective owners.

Creating Web Applications The Hard Way

I was going through some of my old archives – and I found this whitepaper that I had written about 7 years ago for a training class I was doing on ASP (Active Server Pages). It's actually not too bad, for what it is.

But, I have to tell you – after you look at it (**GEEK ALERT:** It's VERY technical and includes a bunch of HTML source code) – scroll down to the last page and check out part two – which is entitled “Creating Web applications The Easy Way.” If you've never used ASP (or JSP or PHP) before – you should have a look at the code and try to understand it – just for fun.

Using ASP and ADO To Create A Data-Driven Website

Today's web users expect a data-driven, dynamic site that provides at least some “personalization” features. The purpose of this article is to help you understand to how use ADO in the context of ASP to create a data-driven website. Since we have to walk before we can run, we'll take a look at some ADO basics like the connection object, how an ASP implementation differs from a VB implementation and then we'll create some code to produce a search form and hitlist.

ADO Overview

Microsoft introduced ADO and ASP in 1997 as the primary way of building data-driven web applications. Whether you're using ADO from VB or from ASP, you can connect to any datasource via ODBC or through native OLE DB drivers.

If you've ever used ADO as part of a VB project, you know that since ADO is a set of COM objects you need to use CreateObject (or Object) to instantiate the object before using them. It's the same in ASP – you must use CreateObject to intantiate the object.

However, in VB you create a project reference to the ADO library and so you have access to all the constant definitions (i.e. adOpenForwardOnly) that are part of the library.

Because VB is referencing the constants library, it can utilize early binding to the ADO object. Since ASP is parsed and interpreted at runtime – ASP must use late binding and you need to reference the equivalent of the constant library via an INCLUDE file (see

“Creating A Recordset” below) whenever you use an ADO constant in your code. Before we get to using INCLUDES, we’ll take a look at the basic syntax of ADO with ASP.

Creating a Connection

In order to create a connection to the database you need to use CreateObject:

```
Dim objConn  
Set objConn = Server.CreateObject("ADODB.Connection")
```

Once the object is instantiated you then use the “Open” method to create an explicit connection (you can also use an implicit connection when opening the recordset – see “Creating A Recordset” below). The only argument you need to supply is the method of connection you’re using. You can choose between using an ODBC DSN connection, an ODBC DSNless connection, or a native OLEDB connection.

The fastest connection with the least overhead is the OLEDB connection, because OLEDB bypasses ODBC altogether can do the data access itself. All of your code will work the same regardless of the type of connection method, but if your ASP application needs to be highly available and scalable, OLEDB is the best choice. Choosing an OLEDB connection means that you have to have ADO 2.0 or higher installed on the server (you can download the latest version of ADO FREE from: <http://www.microsoft.com/data>).

Creating a connection with an ODBC DSN:

For any datasource using a DSN:

```
ObjConn.Open "DSN=MyDSN;UID=Admin;PWD=;
```

Creating an ODBC DSNless connection:

For Access:

```
ObjConn.Open "DRIVER={Microsoft Access Driver  
(* .mdb) };DBQ=\somepath\mydb.mdb;UID=admin;PWD=;"
```

For SQL Server:

```
ObjConn.Open "DRIVER={SQL  
Server};SERVER=myServer;DATABASE=pubs;UID=sa;PWD=;"
```

Creating an OLEDB connection:

For Access:

```
ObjConn.Open "PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA  
SOURCE=\somepath\mydb.mdb;UID=admin;PWD=;"
```

For SQL Server:

```
ObjConn.Open "PROVIDER=SQLOLEDB;DATA  
SOURCE=MySQLServer;DATABASE=pubs;UID=sa;PWD=;"
```

General Tips for Using Connections in ASP:

Make only ONE connection per page (you can have multiple recordsets access the same connection);

Keep the connection open for only as long as you need it – and close it explicitly and and soon as possible.

Creating a Recordset

Once you've explicitly created your connection object and successfully opened the connection, you will (in most cases) issue a query against the datasource and return a recordset that contains the data you want to display. In order to create a recordset you again need to use `CreateObject`:

```
Dim objRS  
Set objRS = Server.CreateObject("ADODB.Recordset")
```

As with the connection object, once the recordset object has been instantiated, you can then use the "Execute" method of the recordset "fill it" with the data from your query:

```
Set objRS = objConn.Execute("SELECT * FROM Authors")
```

You would use the "Execute" method of the recordset object only if you previously opened a valid connection explicitly. You can, however, open a recordset and the connection at the same time (i.e. using an "implicit" connection):

```
ObjRS.Open "SELECT * FROM Authors", "PROVIDER=SQLOLEDB;DATA  
SOURCE=MySQLServer;DATABASE=pubs;UID=sa;PWD=;"
```

The only problem with creating these “implicit” connections (although a bit easier to code) is the fact that you’re opening one new server connection for EACH recordset. This means that you cannot take advantage of connection pooling or MTS.

The best (fastest and most reliable) method of opening a recordset is to use the open method and explicitly telling ADO what type of recordset you’re looking for:

```
ObjRS.Open "SELECT * FROM Authors", objConn, adOpenStatic,  
adLockReadOnly, adCmdText
```

The syntax for the open method is:

```
RecordSet.Open Source, Connection, CursorType, LockType, Options
```

If you use this method opening your recordsets, you have complete control over which connection object to use, the recordset type you want (adOpenForwardOnly [default – i.e. “fire hose”], adOpenStatic, adOpenKeyset, adOpenDynamic), the recordset lock type, and the type of parameter being passed (adCmdText, adCmdTable, adStoredProc).

When using ADO in the creation of websites – it’s often nice to be able to retrieve the number of records in the recordset. There are a number of ways to achieve the record count, but if you want to do it using only the recordset object, you should use the adOpenStatic cursor type. It is the least expensive (in terms of overhead) and yet support paging, record count, and will allow you to move backward and forward within the recordset.

Cursor Type Overview:

adOpenStatic cursor

It is like a snapshot of the data. If you perform a query at 10:00am to retrieve 100,000 records and it takes 5 minutes to read the data, at the 5th minute you will NOT receive data added or deleted since 10:00am. Recordsets opened this way WILL contain an accurate recordcount property.

adOpenForwardOnly cursor (the default)

The data is alive but you can only move forward. Attempts to move backward or to specific record numbers will fail. Recordsets opened this way WILL NOT contain an accurate recordcount property, instead they will always return -1.

adOpenKeySet cursor

The data is alive and any record read will be the most recent data. If you perform a query at 10:00am to retrieve 100,000 records and it takes 5 minutes to read the data, at the 5th minute you will still be retrieving fresh data but NOT data added or deleted since 10:00am. Recordsets opened this way WILL NOT contain an accurate recordcount property, instead they will always return -1.

adOpenDynamic cursor

The data is alive and additions will be noticed. If you retrieve a 100,000 records for example at 10:00am and it takes 5 minutes to read the data, at the 5th minute you will still be retrieving fresh data and records added to the end of the data. Recordsets opened this way WILL contain an accurate recordcount property.

In order to use the ADO constants in your ASP code you can INCLUDE the adovbs.asp file in your individual ASP pages like this:

```
<!-- #INCLUDE file="adovbs.asp" ->
```

As a relative link (without a path), you must adjust the link based upon the configuration of your application directories in relation to the location of the adovbs.asp file. However, rather than having to put the INCLUDE page on all of your pages that use the ADO constants, it's much easier to put the following code at the top of your global.asa file:

```
<!--METADATA  
TYPE="TypeLib"  
NAME="Microsoft ActiveX Data Objects 2.6 Library"  
UUID="{00000206-0000-0010-8000-00AA006D2EA4}"  
VERSION="2.6"  
-->
```

By using the library reference in your global.asa you can avoid those includes on individual pages – and avoid loading and parsing adovbs.asp each time the page loads. **IMPORTANT NOTE:** if you are converting an existing site to this method, be sure to remove all adovbs.asp includes as you will get an error because you're trying to reassign a variable that is in the global.asa file.

Getting Down To Business

Our sample application is a simple search and results page from the Northwind database (we're using the SQL Server Northwind database – but the schema is the same for Access). The application will allow the user to search by company name and or city and return a listing of matching entries from the database.

The code for the search page is very straight-forward:

```
<html>
<head>
<title>Clickware - ADO & ASP - Search Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF">
<form method="post" action="search_results.asp">
  <table width="400" border="1" cellspacing="0" cellpadding="3"
align="center" bgcolor="#CCCCCC">
    <tr valign="top">
      <td colspan="2">
        <table width="400" border="0" cellspacing="0" cellpadding="3"
align="center">
          <tr>
            <td width="151" valign="middle" align="center"
bgcolor="#003399"><font face="Verdana, Arial, Helvetica, sans-serif"
size="-1"></font></td>
            <td align="center" width="237" bgcolor="#003399"><font
face="Verdana, Arial, Helvetica, sans-serif" size="-1"><b><font
color="#FFFFFF">Using
              ADO and ASP for <br>
              Data-Driven Websites</font></b></font></td>
          </tr>
          <tr align="left" bgcolor="#333333">
            <td colspan="2">
              <div align="left"><font size="-1"><b><font face="Verdana,
Arial, Helvetica, sans-serif" color="#FFFFFF">Search
                Page </font></b></font></div>
            </td>
          </tr>
          <tr>
            <td width="151" align="right"><font face="Verdana, Arial,
Helvetica, sans-serif" size="-1">Company
              Name: </font></td>
```

```

        <td width="237"><font face="Verdana, Arial, Helvetica,
sans-serif" size="-1">
            <input type="text" name="companyName">
        </font></td>
    </tr>
    <tr>
        <td width="151" align="right"><font face="Verdana, Arial,
Helvetica, sans-serif" size="-1">City:
        </font></td>
        <td width="237"><font face="Verdana, Arial, Helvetica,
sans-serif" size="-1">
            <input type="text" name="city">
        </font></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><font face="Verdana, Arial,
Helvetica, sans-serif" size="-1"></font><font face="Verdana, Arial,
Helvetica, sans-serif" size="-1">
            <input type="reset" name="Reset" value="Reset">
            <input type="submit" name="SubmitBtn" value="Search">
            <input type="submit" name="ShowAll" value="Show All
Records">
        </font></td>
    </tr>
    <tr align="center">
        <td colspan="2"><font size="-2" face="Verdana, Arial,
Helvetica, sans-serif"></font></td>
    </tr>
    <tr align="center">
        <td colspan="2"><font face="Verdana, Arial, Helvetica,
sans-serif" size="-2">Written
            by <a href="mailto:bobcusick@clickware.com?
subject=Regarding%20Your%20SQL%20ADO%20Article...">Bob
            Cusick</a>, Clickware</font></td>
    </tr>
</table>
</td>
</tr>
</table>
</form>
</body>
</html>

```

Our search <form> points to our search_results.asp page – where the form variables are retrieved, the ADO connection is made and the search results are displayed. The first bit of code on the search_results.asp page is where all the “heavy lifting” is done:

```

<%
Dim objConn
Dim objRS
Dim strSQL
Dim strError
Dim altRowColor
Dim foundCount

Set objConn = Server.CreateObject("ADODB.Connection")
Set objRS = Server.CreateObject("ADODB.Recordset")

objConn.Open "PROVIDER=SQLOLEDB;DATA
SOURCE=myServer;DATABASE=Northwind;UID=sa;PWD="

strSQL = "SELECT * FROM Customers"

If Len(Request("SubmitBtn")) > 0 AND Len(Request("CompanyName")) > 0
Then
    strSQL = strSQL & " WHERE CompanyName LIKE '%" &
        Replace(Request("CompanyName"), "'", "'') & "%'"
End If

If Len(Request("SubmitBtn")) > 0 AND Len(Request("City")) > 0 Then
    If Instr(strSQL, " WHERE") Then
        strSQL = strSQL & " AND City LIKE '%" &
            Replace(Request("City"), "'", "'') & "%'"
    Else
        strSQL = strSQL & " WHERE City LIKE '%" &
            Replace(Request("City"), "'", "'') & "%'"
    End If
End If

objRS.Open strSQL, objConn, adOpenStatic, adLockReadOnly, adCmdText

If objRS.EOF Then
    strError = "No matching records were found."
    foundCount = 0
Else
    strError = ""
    foundCount = objRS.RecordCount
End If
%>

```

Notice that we used the adOpenStatic cursor so we could set our foundCount variable to the number of records returned from the query. If we had not specified adOpenStatic or

adOpenDynamic the objRS.RecordCount would ALWAYS return -1.

We then use the foundCount variable to display the number of records found in the heading of the table:

```
<tr align="left">
  <td colspan="8" bgcolor="#333333">
    <font size="-1"><b><font face="Verdana, Arial, Helvetica,
sans-serif" color="#FFFFFF">Search
      Results Page - <%=foundCount%> Matching Records
Found</font></b></font>
    </td>
</tr>
```

Also notice that we set the strSQL string to

```
SELECT * FROM Customers
```

Then we use two different conditionals to check if the user clicked the “Search” button or the “Show All Records” button:

```
If Len(Request("SubmitBtn")) > 0 AND Len(Request("CompanyName")) > 0
Then
```

AND

```
If Len(Request("SubmitBtn")) > 0 AND Len(Request("City")) > 0 Then
```

If the user clicked the “Show All Records” button (the “name” property of the “Show All Records” button on the search.asp page is “ShowAll” and the “name” of the “Search” button is “SubmitBtn”) then the value of “SubmitBtn” will be empty so the page will disregard anything entered in the search fields and will return all records from the Customers table because the strSQL variable will be unchanged. If the user clicks the “Search” button, the strSQL variable is updated with a WHERE clause substituting the value from the search field on the previous page.

Finally, we write the names of the columns:

```
<tr align="center" valign="bottom" class="tblHead">
  <td width="13" bgcolor="#666666" class="tblCell">ID</td>
  <td width="63" bgcolor="#666666" class="tblCell">Company
  Name</td>
  <td width="52" bgcolor="#666666" class="tblCell">Contact
```

```

        Name</td>
        <td width="51" bgcolor="#666666" class="tblCell">Address</td>
        <td width="25" bgcolor="#666666" class="tblCell">City</td>
        <td width="42" bgcolor="#666666" class="tblCell">Region</td>
        <td width="57" bgcolor="#666666" class="tblCell">Postcode</td>
        <td width="49" bgcolor="#666666" class="tblCell">Country</td>
    </tr>

```

And use a loop to write out the data to the table, and explicitly close the recordset and connection:

```

<%
altRowColor = False
Do While NOT objRS.EOF
    If altRowColor Then
%>
        <tr align="center" valign="top" bgcolor="#FFFFCC">
<%    Else %>
        <tr align="center" valign="top" class="tblCell"> <% End If %>
            <td width="13" class="tblCell"><%=
objRS.Fields("CustomerId").Value %></td>
            <td width="63" class="tblCell"><
%=objRS.Fields("CompanyName").Value %></td>
            <td width="52" class="tblCell"><
%=objRS.Fields("ContactName").Value %></td>
            <td width="51" class="tblCell"><%=objRS.Fields("Address").Value
%></td>
            <td width="25" class="tblCell"><%=objRS.Fields("City").Value %></
td>
            <td width="42" class="tblCell"><%=objRS.Fields("Region").Value
%></td>
            <td width="57" class="tblCell"><
%=objRS.Fields("PostalCode").Value %></td>
            <td width="49" class="tblCell"><%=objRS.Fields("Country").Value
%></td>
        </tr>
<%    objRs.MoveNext
        altRowColor = NOT(altRowColor)
    Loop
objRS.Close
Set objRS = Nothing
objConn.Close
Set objConn = Nothing
%>

```

Summing Up

Although this is a fairly simple example – there's lots of room for improvement. You can (and should) add error checking for both the database connection and data validation (so that the user enters some information if they've clicked the "Search" button). You can go even farther by creating a user login and having the user specify their choice of colors, column order, etc. Use your imagination to harness the power of ASP and ADO via ODBC or OLE DB and create a dynamic experience for your web viewers – it'll keep them coming back for more.

OK – BACK TO THE POINT OF THIS WHITEPAPER:

If you made it this far – and it makes total sense to you – and you love to write code like the above ALL DAY – then you can just stop reading here. If, however, you are human – read on to the next section **Creating Web Applications The Easy Way**.

Creating Web Applications The Easy Way

OK – so all that code above did 2 things - allows the user to search by company name and or city and return a listing of matching entries from the database. THAT'S IT. I didn't even go into the fact that you have to have Microsoft's web server running, and that you have to enable the master session, or that you have to FTP the files, etc.

That's just the SIMPLE code necessary to make a database connection, enter data in to a field and have it return data.

Now let's do it the easy way – with Servoy. OK – here's how to create the exact same application in Servoy from start to finish, step-by-step:

1. Open Servoy Developer and create a new solution;
2. Create a new form based on the customers table;
3. Add the company name and city fields by clicking the “field” tool and specifying the fields you want to add;
4. Add 2 buttons – one called “Enter Search Mode” and one called “Perform Search”
5. Now the “hard” part – we have attach ALL the code to the buttons – for the “Enter Search Mode” button – click on the button and double-click the “onAction” property. Click “New Form Method” and enter this code:

```
controller.find();
```
6. Do the same thing for the “Perform Search” button – and add this code:

```
controller.search();
```
7. Save the form (Ctrl + S);
8. Choose “Activate Web Client” from the “Actions” menu;
- 9. Done.**

Now, I don't know about you – but I find that to be just a TAD easier than the other way – yet a lot of web applications built on ASP .NET, PHP, Ruby, Ruby on Rails, Coldfusion and other technologies – you have to do exactly the same thing... in 2009! That's nuts.