Going live

Tim Anderson shows how to display live data using ASP and Windows web server

The killer feature of Microsoft's Active Server Page technology is the ability to take advantage of COM objects on the web server, including Advanced Data Objects (ADO), the high-level database API for accessing data programmatically on Windows. This month I explain how to create a website that lets you query a database and view the results. It uses an Access database because it is the most convenient to try out the technology, but on busy websites you should consider at least SQL Server as an alternative.

I have chosen to show a solution using just plain text typed into Notepad, rather than delving into the intricacies of Visual InterDev, Microsoft's ASP development tool. Beginning with the direct route gives you a better understanding of how it works. Another factor is that text editing is free, whereas Visual Studio costs money. If you have not tried this before, you may be surprised how little code is needed. ASP is a great solution for publishing data, even on a small local network. You get good performance even from a dial-in connection; Mac and Linux boxes can easily read the data; and no client installations are needed. You do need at least Personal Web Server 4.0, or better still Internet Information Server 4.0 running on Windows NT.

This particular site is for searching a CD collection, although it could easily be adapted for customers and their orders, books and authors, or any number of other scenarios. The database has three tables: Titles, Artists and Tracks. The idea is that users can search for all the titles by a particular artist, and then drill down to the track listing of individual CDs.

The site begins with a simple form (Fig 1), which calls an ASP page. A server-side VBScript in this second page does the real work (Fig 2). It begins by checking that a search string was entered, and exits with a message if it is an empty string. Note the Response.End method, which stops processing the page. All the code within '<%' delimiters is script that runs on the server, while the other code is ordinary HTML.

(FIG 1)

The search form

```
<html>
<title>Music search</title>
<body>
<h3>Search by artist</h3>
<form method="post" ↙
action="artistsearch.asp">
<p><b>Name:</b></p>
<input type="text" name ↙
="artistname" size=50><p>
<input type=Submit>
</form>
</body>
</html>
```

(*Key:* ↙ *code string continues*)

(FIG 2)

ARTISTSEARCH.ASP

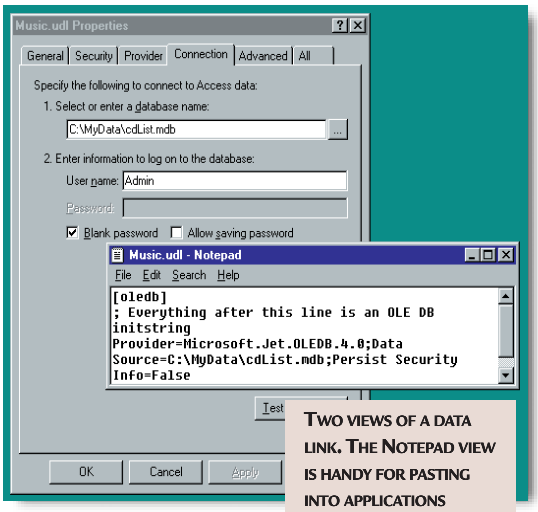
```
<html><body>
<h3>Here are the results of your ↙
search:</h3>
<% on error resume next
sArtistName=request.form("artistname")
if sArtistName = "" then
Response.write "<p>You must enter an ↙
artist name</p>"
Response.write "</body></html>"
Response.end
end if
set conn = Server.CreateObject↙
("ADODB.Connection")
conn.open "Provider=Microsoft.↙
Jet.OLEDB.4.0;Data
Source=C:\MyData\CDlist.mdb;Persist ↙
Security Info=False"
sSql="SELECT
titles.title,titles.id,↙
titles.medium,titles.date,artists.name ↙
from titles,artists WHERE ↙
artistid = artists.id "
sSql=sSql + "AND artists.name LIKE '" + ↙
sArtistName +"%' "
sSql = sSQL + "ORDER BY artists.name, ↙
titles.title"
set rs=conn.Execute(sSql)
rs.Movefirst %>
<table border=1 cols=4>
<tr>
<th> Title </th> <th> Date </th> <th> ↙
Format </th> <th> Artist </th>
<% do while not rs.eof %>
<tr>
<% response.write "<td><a href=" + ↙
chr(34) + "showtracks.asp"
response.write "?titleID="
response.write cstr(rs.fields("ID")) + ↙
chr(34) + ">"
response.write rtrim(rs.fields↙
("Title")) + "</a>"
%>
</td><td>
<% response.write rs.fields("date") %>
</td><td>
<% response.write rs.fields("Medium") %>
</td><td>
<% response.write rs.fields("Name") %>
</td></tr>
<% rs.MoveNext
loop %>
</table>
<% rs.close
conn.close
if err.number <> 0 then
response.write "<p>Error! " + ↙
Err.description +"</p>"
end if %>
<p><a href="musicsearch.asp">Click ↙
here to search again </a></p>
</body></html>
```

The next step is to set up a data connection. The code here assumes the presence of the Microsoft Data Access Components (MDAC) 2.x, which supplies a number of goodies including the JET 4.0 native OLEDB driver. Creating a connection object with Server.CreateObject ('ADODB.Connection') is easy, but the parameters for the connection's Open method are a little obscure. The easy way round this is to use a Universal Data Link (UDL) file to generate the correct code. Right-click the desktop, choose New – Microsoft Data Link, and double-click the new file to open its property sheet. On the Provider tab, choose Microsoft Jet 4.0. Click Next, browse to the MDB file you want to use, and finally click Test Connection to try out the link.

Once armed with a working UDL, there are two ways you can use this in an ASP. You can connect using the file itself, by using the File Name parameter of the Connection's Open method. Alternatively, you can open the UDL in notepad and copy the parameters from there. Either way, it saves you working out what parameters particular OLEDB providers expect.

After making a connection, the ASP code has to build an SQL query matching the search string entered by the user. This process will be familiar to anyone who has worked with VB data access. For more complex queries, Microsoft Access can be handy since its query builder has a two-way View SQL feature, so that you can copy and paste working SQL code. Our example uses a wild card. A single letter of the alphabet retrieves all the artists whose names begin with that letter, while longer strings narrow the search. Finally, a Recordset object is created using the Connection.Execute method.

The remainder of the code iterates through the recordset to generate an HTML table. The neat trick here is that the code generates a hyperlink for each title. This will provide a hot link to another page displaying the tracks for that title. The code generates HTML links, which means that although the user only sees the friendly title, a key value for that title is also stored in the HTML table. When the user clicks this link, the key value gets passed to another ASP that has the job of displaying the tracks and perhaps other detailed information about the selected title. To generate this link, it is necessary to embed double quotation characters in the string returned to the browser. This and other awkward characters can be produced using VB's Chr function. Finally, the script closes the Recordset and the connection.



Specify the following to connect to Access data:
1. Select or enter a database name: C:\MyData\cdList.mdb
2. Enter information to log on to the database:
User name: Admin
Password:
[✓] Blank password [ ] Allow saving password

```
[oledb]
; Everything after this line is an OLE DB
initstring
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\MyData\cdList.mdb;Persist Security
Info=False
```

TWO VIEWS OF A DATA LINK. THE NOTEPAD VIEW IS HANDY FOR PASTING INTO APPLICATIONS

(FIG 3)

SHOWTRACKS.ASP

```
<html><body>
<h3>Here are the tracks:</h3>
<% on error resume next
set conn = Server.CreateObject("ADODB.Connection")
conn.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\↙
MyData\cdList.mdb;Persist Security Info=False"
sTitleID=Request.QueryString("titleid")
sSql="Select * from titles,artists where titles.artistid = ↙
artists.id "
sSql=sSql + "AND titles.id = " + sTitleID
set rs=conn.Execute(sSql)
response.write "<p><b>" + rs.fields("title") + "</b></p>"
response.write "<p><i>" + "by: " + rs.fields("name") + "</i></p>"
rs.close
sSql="Select * from titles,tracks where titles.id = ↙
tracks.titleid "
sSql=sSql + "AND tracks.titleid = " + sTitleID
set rs=conn.Execute(sSql)
if rs.bof and rs.eof then
response.write "<p>No tracks available</p>"
else
rs.Movefirst
%>
<table border=1 cols=1>
<% do while not rs.eof %>
<tr><td align=left>
<% response.write rs.fields("Track") %>
</td></tr>
<% rs.MoveNext
loop %>
</table>
<% end if
rs.close
conn.close
if err.number <> 0 then
response.write "<p>Error! " + Err.description +"</p>"
end if
%>
</body></html>
```

With a little effort you could generate a better-looking HTML page than the one shown here, exploiting the power of Cascading Style Sheets and using images as well as text. This code is stripped down so that you can see how it works. The concept of using an ASP to generate links to other ASPs is powerful, since it gives users access to detailed information while keeping each individual page small and manageable.

The showtracks.asp (see Fig 3) is similar. It illustrates another technique, using the phrase:

```
if rs.bof and rs.eof then
```

to check for an empty recordset.

■ Error handling

One ugly aspect of ASP is error handling. There is no nice exception handling, not even On Error Goto. Instead, the usual ploy is to stick On Error Resume Next at the top of your code, and inspect the built-in Err object when you need to detect errors.

PCW CONTACTS

Tim Anderson welcomes your web development queries and tips, via the usual PCW address or at ***webdev@pcw.co.uk***

Get Microsoft's data access components from *www.microsoft.com/data/mdac2.htm.*

You can find the example application online at *www.onlyconnect.co.uk/pcw/asp.html.*