Are you being served?

Tim Anderson explains how to use Active Server Pages – web development the Microsoft way.

Active Server Pages (ASP) is the Microsoft method of making web pages dynamic. ASP is an extension to Internet Information Server (IIS) that works like Server Side Includes (SSI) on steroids. When a browser requests a page with a .asp extension from IIS, the page is not returned directly. It is parsed to let any embedded scripts run on the server. These scripts can insert new content into the page, typically by querying a database.

The other key aspect of ASP is it wraps a COM object model around your web application, making it more comfortable for developers used to traditional Visual Basic applications. Tricky issues such as keeping track of users as they navigate hither and thither around a website are handled for you. It is also easy to create and program COM automation objects from ASP scripts, so you can deploy web components written in any language that can create COM servers, including Delphi, Visual Basic and Visual C++. ASP is worth exploring, the drawback being that you need to use Internet Information Server, which means Windows NT.

However, all is not lost if you want to use another web server or operating system. Chili!Soft <www.chilisoft.com> does ASP extensions for other servers and platforms, although that is not much help if you are using an ISP via dial-up and the ISP doesn't have these extensions installed. The other problem is that on non-Windows platforms, ASP scripts may run, but the COM objects that are everywhere on Windows aren't likely to be present. Finally, ASP runs on Personal Web Server for Windows 95 and NT, so you don't have to have NT Server to get started.

[FIG 1]

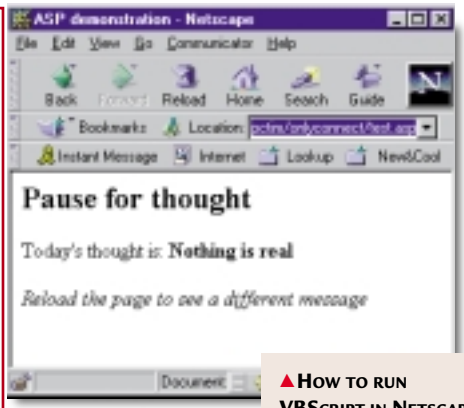
A simple Active Server Page

```
<html>
<title>ASP demonstration</title>
<body>
<script runat=server language=VBScript>
function randommessage
dim iMessage
Randomize
iMessage = Int((3*Rnd)+1)

select case iMessage
case 1
randommessage="Beware the Bandersnatch"
case 2
randommessage="Only connect"
case 3
randommessage="Nothing is real"
case else
randommessage="You what?"
end select
end function
</script>

<% sThought=randommessage %>
<h2>Pause for thought</h2>
<p>Today's thought is: <b><%= sthought↙
 %></b></p>
<p><i>Reload the page to see a ↙
different message</i></p>
</body>
</html>
```

(*Key:* ↙ *code string continues*)



▲ **How to run VBScript in Netscape: use an ASP**

First steps with ASP

You only need a Microsoft web server and Notepad to run an ASP. An ASP is simply an HTML page with a .asp extension. The clever bits are inside the <% and %> delimiters. Anything within these delimiters is script that runs on the server. You can also use <%= and %> (note the equals operator) which means: 'Evaluate this expression and send the result to the browser'.

Finally, the <SCRIPT> tag in an ASP accepts the RUNAT=SERVER attribute, to ask the web server to run the script. All three features are demonstrated in Fig 1.

When the page loads, the web server sets the value of a variable by calling a VBScript function. A couple of lines later, the variable is referenced within the <%= delimiter, which inserts its value into the page. Although this is VBScript, the page opens fine in Netscape Navigator as it runs only on the server. If you view the source in the browser, it is plain HTML.

ASP objects and methods

Server-side scripting is useful, but there is more to ASP than that. The ASP object model provides an easy way to deal with many routine web application tasks. There are five built-in objects available:

Request

This represents the current HTTP request from the browser. You can read the query string, the request body (including form elements), cookie values and a ServerVariables collection that gives comprehensive access to what is in the HTTP request along with key information about the web server.

Response

This object sends data back to the browser through the Write method. You can buffer the output by setting the Buffer property to true, which is handy if you want to build up a complete page before deciding whether to send it or cancel for some reason. If you cancel, you might use the Redirect method to display another page, perhaps with an error message or an alternative form.

Server

The key method of the Server object is CreateObject, letting you instantiate any available COM object on the server. There is also a handy HTMLEncode method which is great for creating web tutorials, since it converts what would be tags into literal strings that appear in the browser.

Application

The Application object lets you store global data. An application is all the documents in a virtual directory on the web server. Set a variable such as this:

```
Application("myvariable") ↙
= myvalue
```

Subsequently you can call Application

[FIG 2]

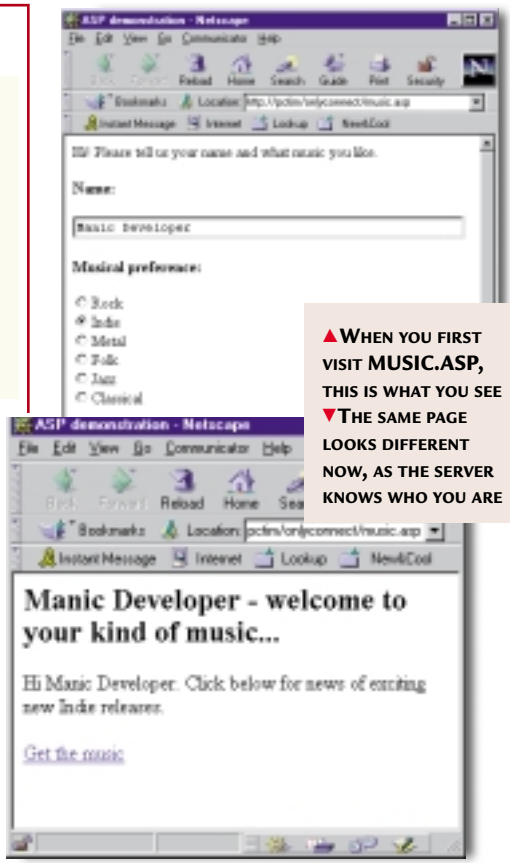
This is MUSIC.ASP

```
<html>
<title>ASP Music application</title>
<body>
<% if session("username") = "" then %>
<p>Hi! Please tell us your name and what music you like.</p>

<form method="post" action="processform.asp">
<p><b>Name:</b></p>
<input type="text" name ="username" size=50><p>
<p><b>Musical preference:</b></p>
<input type="radio" name="musictype" value="Rock" checked>Rock<br>
<input type="radio" name="musictype" value="Indie" >Indie<br>
<input type="radio" name="musictype" value="Metal" >Metal<br>
<input type="radio" name="musictype" value = "Folk">Folk<br>
<input type="radio" name="musictype" value = "Jazz">Jazz<br>
<input type="radio" name="musictype" value = ↙
"Classical">Classical<p>
<input type=Submit>
</form>
<% else %>
<h2><%= session("username")%> - welcome to your kind of ↙
music...</h2>
<p>Hi <%= session("username") %>. Click below for news of exciting ↙
new <%= session("musictype") %> releases.</p>
<a href="getthemusic.asp">Get the music</a>
<% end if %>
</body>
</html>
```

and here is the form handler, PROCESSFORM.ASP

```
<% if request.form("username")="" then %>
<html>
<title>Processing the form...</title>
<body>
You must enter a name. Click Back and do the job properly.
<% else
session("username")=request.form("username")
session("musictype")=request.form("musictype")
response.redirect "music.asp"
end if
%>
</body>
</html>
```



▲ WHEN YOU FIRST VISIT MUSIC.ASP, THIS IS WHAT YOU SEE
▼ THE SAME PAGE LOOKS DIFFERENT NOW, AS THE SERVER KNOWS WHO YOU ARE

("myvariable") to retrieve the value.

The Application object also has two events, OnStart and OnEnd. To handle these, create a file called global.asa and place it in the application directory. Name two procedures Application_OnStart and Application_OnEnd to handle the events. OnStart fires when the first client requests a document. OnEnd fires when the web server shuts down.

Session

The Session object is a critical piece in most ASP applications. It gives your web application state and lets you keep track of users as they navigate from page to page. When a user logs on to the site, the Session_OnStart event fires, and again you can write code in global.asa to handle it. The SessionID uniquely identifies the user while the session is active. Session_OnEnd fires when the user stops requesting pages for an interval determined by the Timeout property. You can also terminate a session with the Abandon method.

Using ASP

Fig 2 shows how you can use some of these objects. When the user first hits the page, their name and favourite music is requested. The web application now knows these details, and refreshing the page does not re-present the form, but shows a customised welcome.

Snags

If you can't get ASP to run, check you have a compatible web server, which for most means IIS 4.0 or Personal Web Server (both are in the NT4 Option Pack, downloadable from Microsoft's site and distributed on CD). Bizarrely, there is a Windows 95 version of this. Next, check the permissions which must be set to at least Script for ASP to run.

You can make browser-independent ASP applications, but browsers with cookies disabled can cause problems.

ASP runs scripts fairly slowly, and mixing code and HTML content can get messy. The fix is to minimise code in ASP pages and use server-side COM components for the real work. This makes applications more scalable and allows the use of Microsoft Transaction Server to manage components.

PCW CONTACTS

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