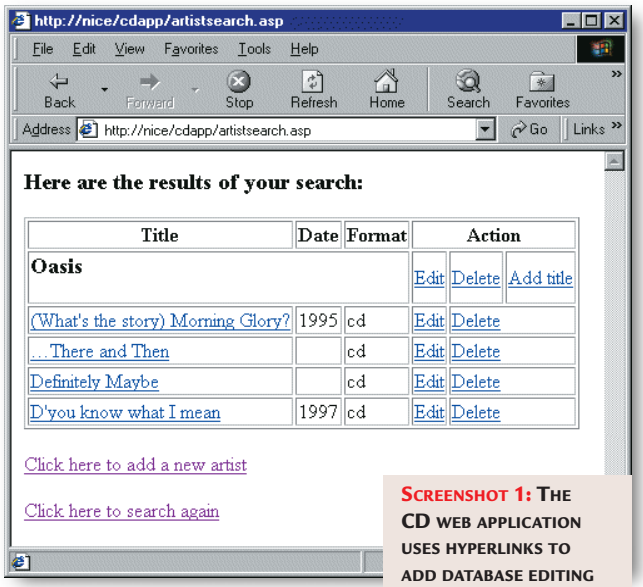
Component moments

Tim Anderson introduces component development with Active Server Pages.

Last month's column included a simple demonstration of how to query a CD database using Active Server Pages (ASP). The application has three pages. The first is a form for searching by Artist, the second displays a list of matching titles, and the third shows the track listing for an individual CD. The next stage is to enhance the application to allow deletions and updates. Another consideration is to improve performance and make the site easier to maintain by shifting code out of the HTML page and into separate components.

As with any application development, the starting point is to work out the design. A web application needs a different kind of design from the traditional Windows style, particularly if you want to support any browser running on any operating system. The key functions of any database are searching, editing, adding and deleting records. A user-friendly way to include these features is to amend the search results page so that each record has hyperlinks for editing and deleting. This application also needs links to add a title by an existing artist, and to add artists to the database. Screenshot 1 shows one approach. The design could be improved, for example, by using bitmaps for the Edit, Delete and Add buttons. It has been kept simple to show the basic approach as clearly as possible.



SCREENSHOT 1: THE CD WEB APPLICATION USES HYPERLINKS TO ADD DATABASE EDITING FUNCTIONS

FIG 1 edittitle.asp

```
<html>
<title>Edit Title</title>
<body>
<h3>Edit title</h3>
<% sTitleID=Request.QueryString↙
("titleid")
set thisCDHTML = server.createobject↙
("cdApp.cdHTML")
iTitleID = Cint(stitleID)
thisCDHTML.GetTitleValues ↙
iTitleID,sTitle,iTitleDate
set thisCDHTML=Nothing %>
<form method="post" action = ↙
"savetitle.asp">
<p><b>Title</b> <input type=↙
"hidden" name="titleid"↙
value="<%= iTitleID %>" >
<input type="text" name=↙
"title" value = "<%= sTitle %>"↙
size=50></p>
<p><b>Date:</b> <input type=↙
"text" name="titledate" value="<%= ↙
iTitleDate %>" size=10></p>
<p><input type=Submit value="Save ↙
title"></p>
</form>
</body>
</html>
```

(Key: ↙ code string continues)

Implementing the new functions

Rather than implementing these new functions as ASP script, here is how it might be done using components. In this context, 'component' means a COM component, also known as an ActiveX DLL, running on the web server. The components will have two tasks. They need to perform database operations in response to calls from ASP, and they may need to generate HTML content to pass back to the browser. It makes sense to use separate components for these tasks as you might want to re-use the database components in a Windows application that has no need of the HTML aspect.

There are several approaches you can take in designing an ASP application with components. This example is intended to give you an idea of what is involved. Here is the sequence of events when the user clicks the Edit hyperlink for a CD title:

- The link calls edittitle.asp, passing the ID of the title required as a parameter.
- Script in edittitle.asp instantiates a COM component built in Visual Basic 6.0. It then calls a method of the component to retrieve the values of the required Title record and builds a form to display the title. The form's Action attribute points at a further ASP page called savetitle.asp
- Savetitle.asp instantiates the same

Note that in order to preserve data integrity, the Delete function may need to delete any related records as well. For example, if an Artist is deleted, then Titles by that artist must also be deleted.

COM component, retrieves the form values from edittitle.asp, and calls a method of the component to save the title.

This approach has the advantage that no client-side scripting is needed, so browser compatibility is excellent. Data access is handled by the component, so Visual Basic's features can be used to validate the data or perform other processing.

Fig 1 shows edittitle.asp. Note how a hidden field is used to preserve the ID of the current title.

The Visual Basic end

This application used two COM components, cdHTML and cdData. The idea is that one component handles the interaction with ASP, and the other does the data access. This is a flexible system, since you might want to write other applications that use the same data access component.

Here is how to create the components using Visual Basic 6.0. Start a new project, choosing ActiveX DLL. Call the project cdApp, rename Class1 to cdHTML, and add a second class, cdData. Open up Project – References, and set references to the ActiveX Data Objects 2.1 library and the Active Server Pages object library. This second reference opens communication between your component and the current ASP session. It includes a ScriptingContext object with Application, Request, Response, Server and Session properties that match those available in an ASP script. You just need to activate the ScriptingContext, like this:

```
Private thisContext As ↙
ScriptingContext

Private Sub OnStartPage↙
(sc As ScriptingContext)
Set thisContext = sc
End Sub
```

OnStartPage is a method that gets called by the web server whenever this component is instantiated via ASP.

If you are running Internet Information Server 4.0 or higher, you can use a different technique. Set a reference to Microsoft Transaction Server in your project and call GetObjectContext() to retrieve an ObjectContext object. You can then get at the ASP objects through the Item property of the ObjectContext, for example:

```
thisObjectContext.Item↙
("Response").Write("<p>Some ↙
text</p>")
```

Whichever method you use, the result is you get full access to ASP from within your compiled Visual Basic component.

Implementing the components

What happens in the VB components is mostly standard VB programming. In this example, cdData creates an ADO connection in its Class_Initialize method. It also defines a CDTitle type for convenience in working with the data. For editing a title, there are two relevant methods. GetTitle takes a titleID parameter and returns the data for that title. SaveTitle takes a CDTitle parameter and saves it to the database.

FIG 2 GetTitleValues method

```
Public Sub GetTitleValues↙
(titleID As Variant, title As ↙
Variant, titledate As Variant)
Dim thisTitle As CDTitle
thisTitle = thisData.↙
GetTitle(titleID)
title = thisTitle.title
titledate = thisTitle.date
End Sub
```

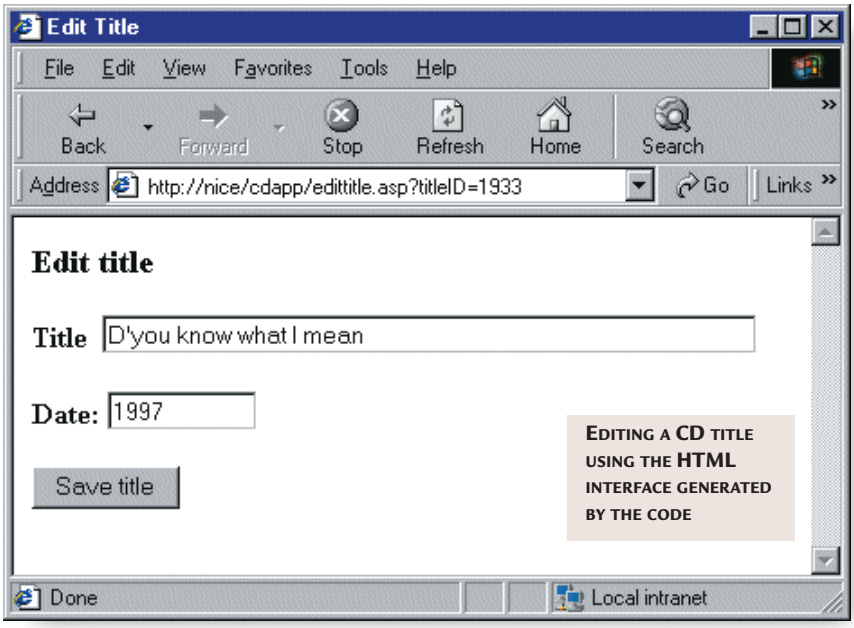
The other component is cdHTML. In its OnStartPage event handler, it obtains the current ScriptingContext or ObjectContext, and also instantiates a cdData object. The GetTitleValues method is listed in Fig 2. Note VBScript only passes variants as arguments. This method exploits the fact they are passed by reference to give them new values, that are then available in the calling ASP page. In the same way, the SaveTitle method accepts arguments from ASP that are stored back to the database.

A disadvantage of the ASP approach, inherent in many web applications, is that connections and recordsets are constantly opened and closed. It isn't as bad as it appears, particularly when you go a step further and host components in Microsoft Transaction Server, that]\ can cache objects and connections so they are not always recreated from scratch.

What about Unix?

You can't please all the people all the time. David Marty writes: 'As a web-master with a Unix host, I find it frustrating that you've concentrated on Windows web servers. Please include more on Unix Apache and especially an idiot's guide to database system building on the web.'

My view is both operating systems have a place, so next month there will be a look at using MySQL on Linux.



EDITING A CD TITLE USING THE HTML INTERFACE GENERATED BY THE CODE

PCW CONTACTS

Tim Anderson welcomes your web development queries, at webdev@pcw.co.uk. You can find this ASP example online at www.onlyconnect.co.uk/pcw/pcw_index.html. For further information, see the newsgroups microsoft.public.inetserver.asp.components and microsoft.public.inetserver.asp.db. Two recommended books are Beginning components for ASP *(Wrox Press, ISBN 1-861002-88-2, £28.99) and* Professional Active Server Pages 3.0 *(Wrox Press, ISBN 1-861002-61-0, £43.99).*