Debugging Web Applications

Working with ISAPI, ASP and similar creatures within Microsoft's Internet Information Server

by Brandon Smith

In an overview of tools for building web apps in Delphi in the February issue of *Developers Review* (Issue 8), I made the statement that 'ISAPI ... cannot be unloaded without unloading the server itself.' *[So that you can get the most out of this article, we've included Brandon's overview on this month's disk as an Acrobat file and also put it on the Developers Review website in our recently introduced Reviews Online section. Ed].*

One of the hazards of living and working on the bleeding edge is the blood dripping off the keyboard after one runs into situations such as the one that caused me to make that statement. I didn't go into much detail about the situation, since my focus was on the different kinds of web tools available to a Delphi developer. Nor did I go into the fact that my edit-compile-test cycle had become edit-compile-test-free-from-the-clutches-of-IIS.

Why the extra steps? Well, once you fire up your web application as an ISAPI application, it is a DLL owned by IIS. You test it by firing up your browser and clicking on whatever it is that fires your DLL. You spot that a word is misspelled, or perhaps something equally obvious jumps out at you, and you jump into Delphi, go right to where the correction needs to be made, type in the fixes and hit `Alt-F9`. Bang! Error! Cannot create output file xyz.dll! Because, of course, the DLL is still loaded. There are two easy workarounds which I will get to later, and a series of other potential workarounds, but at that time I felt that the statement I'd made in the article was reasonably accurate.

An Inprise staffer (who has asked for anonymity) promptly informed me that I was very much mistaken, that 'You can debug and do everything with ISAPI you can do with CGI if you are using IIS and virtual directories. You can tell it not to cache ISAPI DLLs or ASPs so there is no reason to develop using CGI since NT option pack came out with IIS4 and MTS.' He went on to point out 'The article mentions that you cannot debug ISAPI and that you have to close down the web server to unload and rebuild the DLL. This is contrasted with CGI which allows you to kill the process and avoid these two problems. This is false.'

Quite true, it is false that you cannot debug ISAPI applications, but it is not false that you have to close down the server: at least it wasn't false for me at the time, since these techniques are not mentioned in the Delphi help files, and the few references which I had come across hinting that such a technique might be possible gave no details. In the article I did discuss a simple and obvious workaround: develop the application first as a CGI, then, once things are working right, convert it to an ISAPI. This step involves changing a few lines in the project file: not exactly a major operation, though I'd be surprised if it always works as smoothly as it did for me. This technique works so well because the various web application classes have a common ancestor.

You can debug a DLL in Delphi 4 and an ISAPI web application is a DLL, as is an ASP object. The specifics for doing so are well covered by the help files that come with the web module, but there is this warning: *'Note: Before launching the web server using your application's run parameters, make sure that the server is not already running.'*

In other words, you have to close down the web server before you can start a debug session. This turns out to be trivial, though it was several weeks before I found someone who knew the technique.

Stephane Grobety responded to my plea for help in getting started with ASP object construction in Delphi. If you go to his website (which is at www.fulgan.com) and select Delphi, you will find a lot of good advice and some techniques dealing with this area. You can download a helpful example of how to do an ASP object in Delphi 3 that works perfectly well with Delphi 4. Part of that zip file was a small .cmd file, which is shown in Listing 1.

I soon had this working for me in a separate DOS session so that my normal edit-compile-test cycle became edit-compile-test-free with very little extra effort or time.

You'll find in the sidebar on page 64 some information that deals with how to turn off the in-memory caching of IIS when it runs an ISAPI. This is information which I found but have not yet used, since the above method seems to work just fine for me.

I'm not really sure how virtual files (which are really handy) can help this situation, nor am I really impressed with MTS at this time, though I am thrilled by the possibilities.

➤ *Listing 1*

```
@echo off
echo stopping services
net stop "World Wide Web Publishing Service"
net stop "FTP Publishing Service"
net stop "IIS Admin Service"
echo starting services
net start "World Wide Web Publishing Service"
net start "FTP Publishing Service"
```

When I complained to the anonymous Inprise insider that I hadn't been able to find any documentation to support what he said could be done, he sent me a document explaining how to convert the entire IIS service to a process. You'll find it on this month's disk as the file HowToDebug.txt, with the supporting .reg files. It probably won't hurt to remind you, *caveat* user (anyone know the Latin for 'user'?): back things up before you start! In fact, that's the reason for the requested anonymity: he is probably rightly worried that his email will get flooded with requests for support and complaints when things don't work right.

You *can* email me, but in this case I make no promise that I can help.

Brandon Smith works in Jefferson City, Missouri for Rose International, on Delphi object infrastructure building. You can email him at Brandon@synature.com or visit www.synature.com

Turning Off IIS In-Memory Caching

Apparently one can turn off the in-memory caching which Microsoft Internet Information Server does on the first occasion a DLL is activated. This is, however, something that should only be done on a development system, as it is a permanent change to IIS and reduces the performance of ISAPI applications by forcing a reload each time, instead of keeping them in memory. In other words, your ISAPI application behaves just like a CGI.

Mutilating your server, at least for ASP cache settings, is probably done with one of these registry keys. I have not tried them, so be sure to do some experimentation, back up your registry and don't come running to me if it all goes wrong!

```
HKEY_LOCAL_MACHINE\SYSTEM\
  CurrentControlSet\
    Services\
      W3SVC\
        ASP\
          Parameters
```

plus `\ScriptFileCacheSize` or `\ScriptFileCacheTTL` or `\SessionTimeout`. It would appear that a zero on `...\ScriptFileCacheSize` means nothing will be cached, whilst a zero on `...\ScriptFileCacheTTL` will mean zero seconds will pass before it is unloaded.

However, the HTML page where I found this (somewhere on Microsoft's site at www.microsoft.com, but since they often don't record how to find a page once you have saved it, all I have is cryptic Java Script commands that don't tell me where I might go to get back to it), is talking about pre-compiled scripts, which might be the same as a compiled ASP object or it might not...

There also seems to be a method which one can call, `Session.Timeout`, which will apparently let one kill a running script without killing IIS. However, now we are dealing with the scripts and the `Session` object, and, for my original purposes, this does not really help, since the `Session` is defined by a cookie exchange with the browser, and I had already ruled out cookies as a viable way of saving state for my purposes. Besides, this probably will not unload any ASP objects loaded by the script.

And then finally there is some further information which I found at Article ID Q166279:

'To release the DLL from Internet Information Server 4.0's cache you must run the following command file:

```
net stop iisadmin /y
mtxstop
net start w3svc'
```

This same article also informs us:

'If you want to make restarting the WWW service considerably faster on a development IIS server, create the following value in the registry, of type REG_DWORD, and set it to zero:

```
HKEY_LOCAL_MACHINE\SYSTEM\
  CurrentControlSet\
    Services\
      W3SVC\
        Parameters\
          EnableSvcLoc
```

Do the same for `MSFTPSVC` *and* `GOPHERSVC` *if you are running them. Note: on a production server the service locator should be enabled by setting the value to one.'*