Version: FoxPro 7.0
Figures:
File for Subscriber Downloads:

Publishing Your First Web Service

Whil Hentzen

Web Services – The Buzzword of the '02s! It's nothing really new, however, any more than LAN networks were new concepts in the late '80's – just a different implementation and enhanced functionality of an old concept. However, Web Services are different enough that it's a worth a new look – and to consider whether it's time for you to jump in. This series of articles will show you how to create, test, and publish a Web Service using Visual FoxPro 7.0 and the SOAP Toolkit from Microsoft.

The marketing folks have been working overtime with "Web Services" lingo over the last six to twelve months, but it seems lately that every month you run into another real example of a Web Service. The community portal, FoxCentral.net, is built on Web Services, the ProFox listserve just added a Web Service access to its archives, the Universal Thread is being rearchitected on Web Services – and those are just places important to Fox developers. Web Services are…. everywhere!

What's a Web Service?

I was tempted to do a Google search on "Web Service" and just paste a bunch of marketing descriptions that promised the ability to Make Money Fast, Lose 20 Pounds Overnight and Banish Acne Forever! But you've probably already read all of them.

No, Really, What's a Web Service?

A Web Service is merely a DLL sitting on your Web server that can be called by other computers on the Internet (or your Intranet.) The methods in the DLL can do things just like COM servers can now. For example, a Web Service can deliver selected data from the Web server based on parameters that the user provided when calling the DLL. An oft-cited example is a weather service, where you, the user, provide a Zip Code to a Web Service, and the Web Service sends back the forecast for that geographic area.

Another example of a Web Service is the use of an algorithm that receives input, processes that input, and returns a calculated value. In other words, it's a function available over the Internet. An oft-cited example of such a function is Babelfish, where you input a string of text, select a language, and the Web Service returns that text translated into the language of your choice. At Great Lakes last year, Gary DeWitt and Kevin McNeish demonstrated this Web Service, and had the usual amount of fun by sending the translated string back to Babelfish a second time, and compared the resulting English with the original, ensuing in much hilarity.

Designing your Web Service

If you think of a Web Service as providing access to data of yours to others who want it, it's pretty easy to come up with some examples. In this article, I'll use the ability to provide access to news items in a simple table, querying selectively by date, as my example.

The table will look like this:

```
NEWS.DBF
dNews d(8)
cNews c(50)
```

The Web Service will allow the user to get all of the News items, or pass a parameter to specify just those News items after a certain date. The code to do so will look something like this:

```
lparameter ldDate
if pcount() = 0
 ldDate = {^1980/1/1}
endif
```

```
select cNews, dNews from NEWS ;
  where dNews >= ldDate ;
  order by dNews ;
  into array aNews
```

And that's about it.

So much for all those $100 books on software design, eh?

Preparing your Development Box for Creating a Web Service

You'll need to have Visual FoxPro 7.0 and the SOAP Toolkit 2.0 installed on your development machine before you start. I will assume that you have VFP 7.0 (What? You DON'T? Why the hell not? Been living in a cave for the last ten years?) installed, but it's entirely possible that you don't have the SOAP Toolkit installed.

You can install the SOAP Toolkit from the VFP CD (if the screen in Figure 1 doesn't display when you load your VFP 7 CD into your computer, find the VFPSTART.HTA file on the CD and execute it by double-clicking on it in Explorer.)

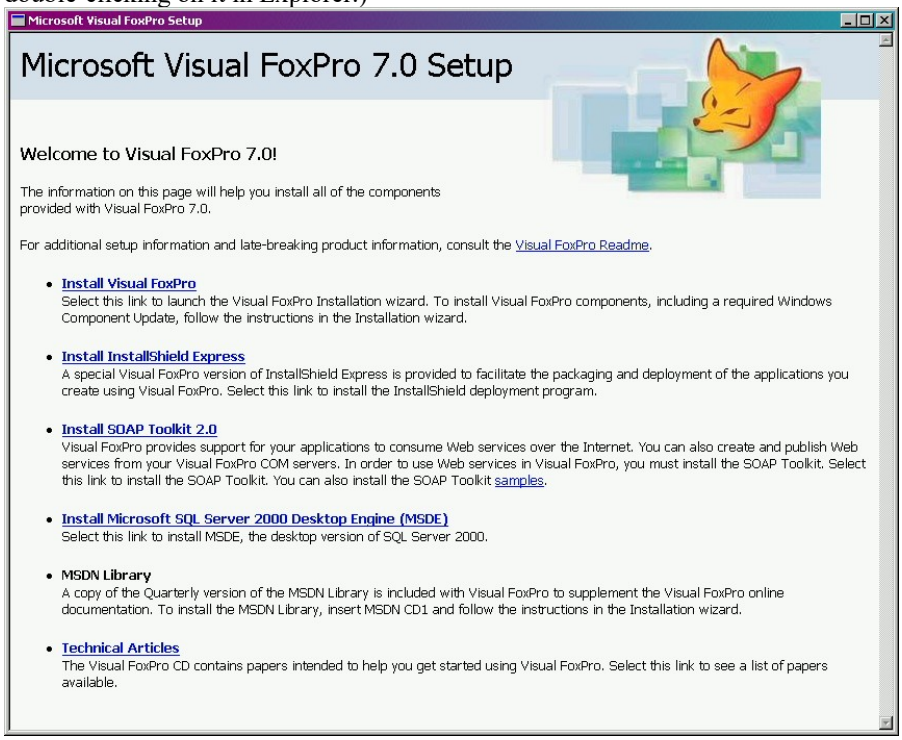

Figure 1. You can install the SOAP Toolkit from the VFP 7.0 setup screen.

You can also download the SOAP Toolkit from the Microsoft website at msdn.microsoft.com. Try msdn.microsoft.com/soap, click on Downloads, and search for SOAP Toolkit 2.0. At the time of this writing, the latest release is SP2. Unlike it's bigger operating system brothers, this SP contains the entire toolkit – it's not just a patch to existing software. So you can download the entire thing and install it from scratch, and be confidence that you have everything you need for development.

Of course, you'll probably also want to download the samples and the redistribution kit. More on that stuff in future articles.

You'll also need the Windows Installer on your machine. If you're running any "2000" software – Windows 2000, Office 2000, etc, you've already got the Windows Installer. If you're still using a bare bones NT4 installation, though, you'll need to download the Windows Installer first (again, from msdn.microsoft.com – search for "Windows Installer"), install it, and then download the SOAP Toolkit and install it.

After you have installed the SOAP Toolkit, you'll have a new directory under Program Files called MSSOAP. The help file (.CHM format) for the SOAP Toolkit is in the MSSOAP\Documentation directory.

Creating a Web Service on Your Development Box - I

Since a Web Service is a DLL, it makes sense that you'll have to create a DLL with Visual FoxPro. I'll go through the exact steps here in case you've not had the opportunity to do create a DLL before.

1. Create a directory on your development box called "WSC". (WSC stands for "Web Services version C" – you could call your directory Fred or Ethyl if you wanted.)

2. Load Visual FoxPro 7.0 and CD to that directory.

3. Create the table NEWS.DBF with the structure defined earlier, and put a couple of records in it, like so:

```
dNews      cNews
2001/11/1  This is November 1 News
2001/11/2  This is News from 11/2
2001/11/3  And this is the third New item
```

4. Create a program named WSC.PRG with the following code (or download from …)

```
DEFINE CLASS hwpclass AS session OLEPUBLIC
Name = "hwpclass"
PROCEDURE getnews
LPARAMETERS ldDate
DO case
CASE PCOUNT() >= 1
 * do nothing cuz we have a date
 * well, maybe check that the parm is a date?
CASE PCOUNT() < 1
 ldDate = {^1980/1/1}
ENDCASE
select cNews, dNews from \NEWS ;
  where dNews >= ldDate ;
  order by dNews ;
  into array aNews
RETURN aNews[1,1]
ENDPROC
ENDDEFINE
```

Listing 1: A simple Web Service class.

Note that in this 'production strength' version of the SELECT statement, I've included a slash in front of NEWS; I'll discuss why later in this article.

5. Create a project named WSC.

6. Add the program file to the project. You'll notice that it gets marked as MAIN automatically as shown in Figure 2.

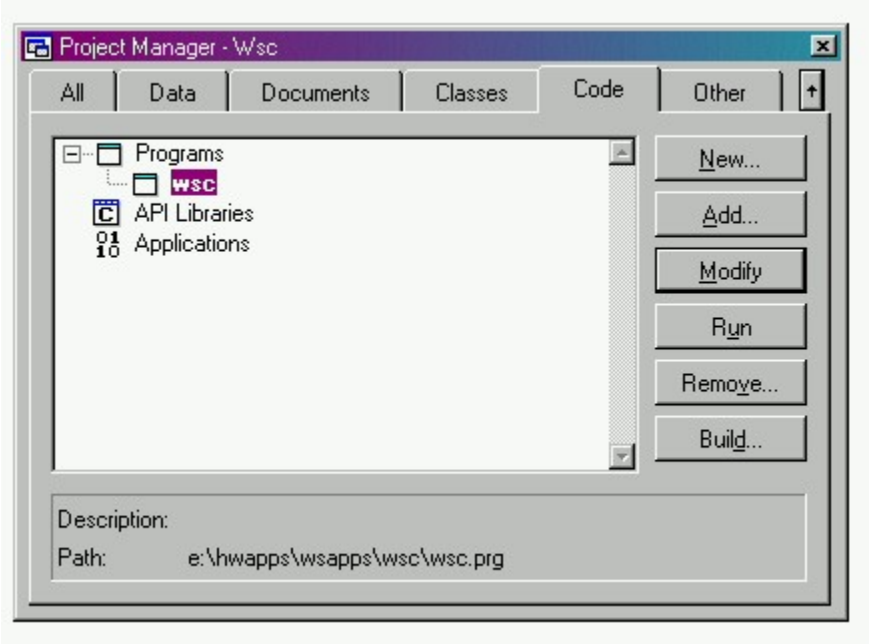

Figure 2. WSC.PRG, when added to a project as the first file, is automatically set as MAIN.

7. Build the project as a multi-threaded DLL, as shown in Figure 3.

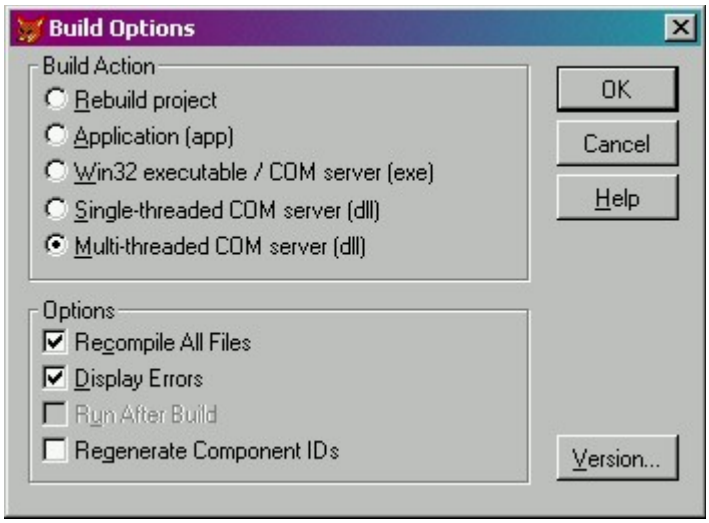

Figure 3. Select Multi-threaded COM server when building your Web Services project.

8. After building the DLL once, open the Project Info dialog via the Project | Project Info menu option, select the Servers tab, and set the Instancing to Multi Use as shown in Figure 4.

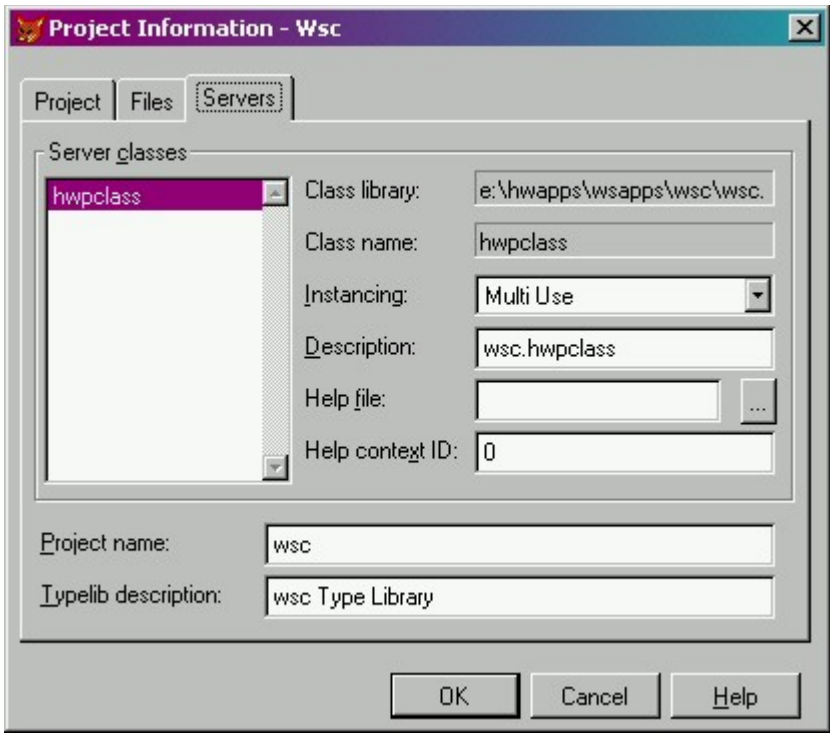

Figure 4. Be sure to set your Web Services project as Multi-Use.

9. Rebuild the project.

You now have a DLL ready for testing on your development computer.

Note that your Web Service DLL will not have a visual interface, so you want to use a base class that is as lightweight and compact as possible. The Session class is the best choice because not only is it lightweight and compact, but it has it's own data session. You can't create a class based on the VFP Session base class using the Class Designer. You have to do it in code.

Also note that the code in Listing 1 defines the class as OLEPUBLIC. Thus, if you define your DLL using the class designer (based on a class other than the session class), you'll have to make your class OLEPUBLIC using the visual tools. Suppose you use the Custom class. Once you've got the Class Designer open, open the Class Info dialog by selecting the Class | Class Info menu option, and then check the OLE Public check box in the Class tab, as shown in Figure 5.

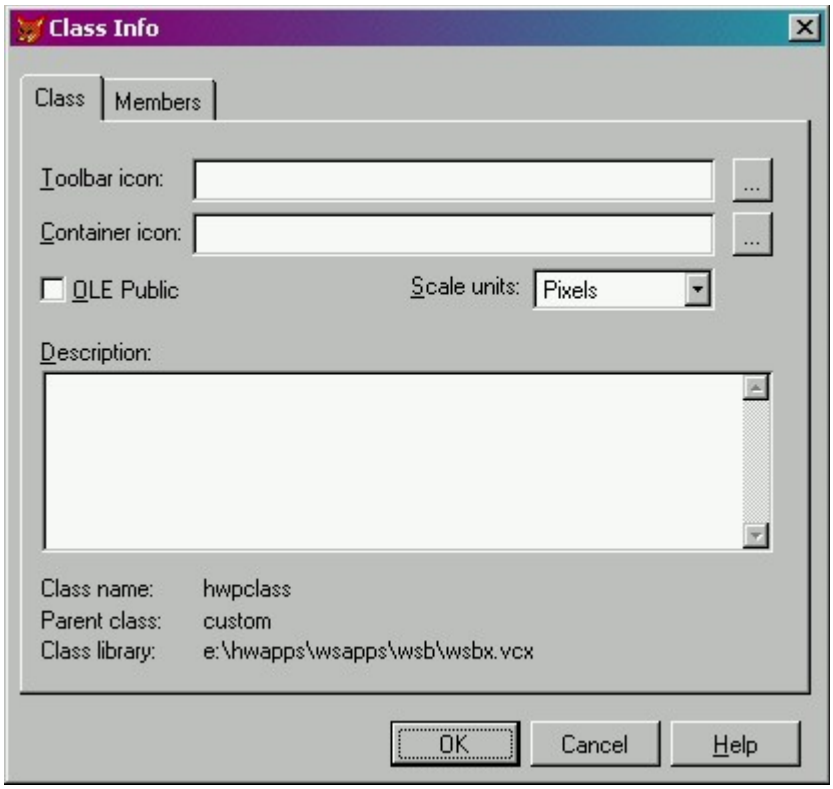

Figure 5. Setting the OLE Public checkbox in the visual Class Designer.

Testing your Web Service DLL on Your Development Box

Now that you have a DLL, it's time to test it to make sure you've written your VFP code properly. Assuming you've built WSC.DLL and it's in the current directory, just enter the following in your Command Window:

```
o=createobject("wsc.hwpclass")
? o.getnews()
```

and see the following value display

```
This is November 1 News
```

Entering the following in the Command Window

```
? o.getnews({^2001/11/2})
```

results in the value

```
This is News from 11/2
```

appearing on the screen.

Creating a Web Service on Your Development Box - II

Now it's time to convert this reg'lar ol' DLL into a Web Service.

In the Project Manager, highlight the WSC file, and right click on it to open the context menu. Select the Builder menu option as shown in Figure 6.

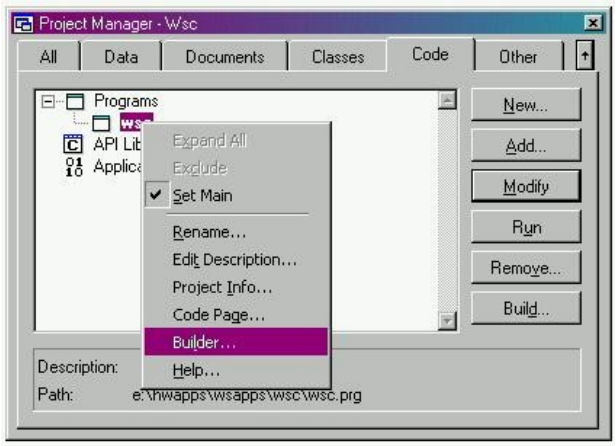

Figure 6. Select the Builder option from the context menu to run the Wizard Selection dialog.

Doing so will bring forward the Wizard Selection dialog, as shown in Figure 7. Select the Web Services Publisher and click OK.

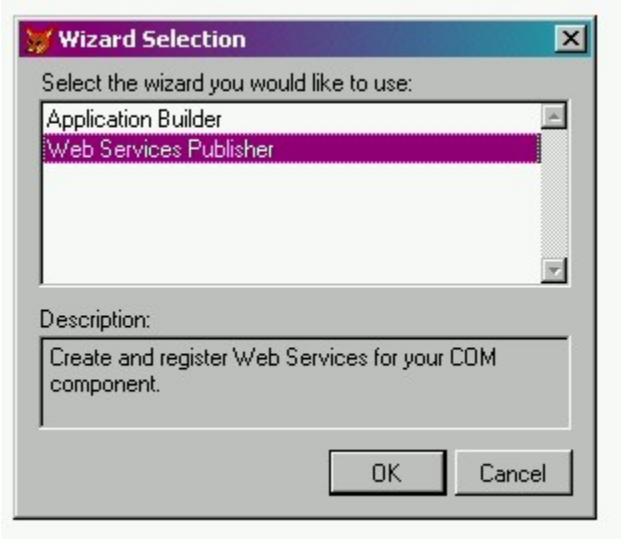

Figure 7. Selecting the Web Services Publisher Wizard.

The Visual FoxPro Web Services Publisher dialog displays, as shown in Figure 8.

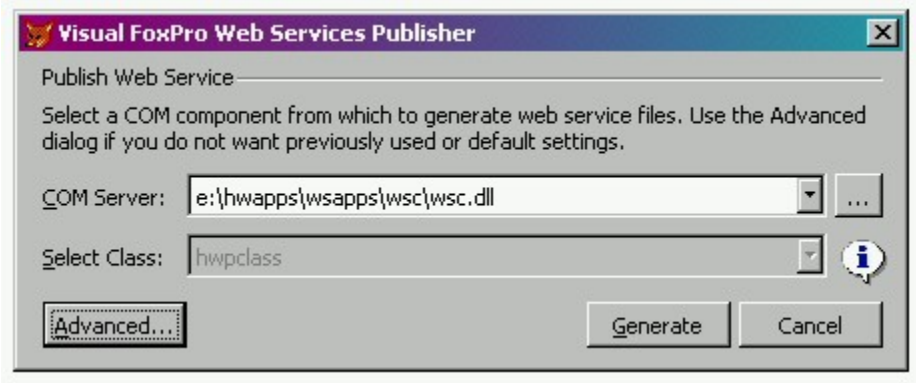

Figure 8. Selecting the COM server that contains your Web Services class.

The Advanced button is used for changing settings, as shown in Figure 9. You'll probably want to open it up and verify that the settings are correct, particularly for the location of the WSDL files. If your web server software is pointing to c:\inetpub, you can leave the entry in the WSDL: File: text box alone as shown in Figure 9. If your web server software points somewhere else, like e:\inetpub\wwwroot, then change the file to the appropriate directory.

You'll also want to change the Listener from ISAPI (the default used by Visual FoxPro) to ASP for the time being. (In a future article, I'll discuss the difference between these two choices.)

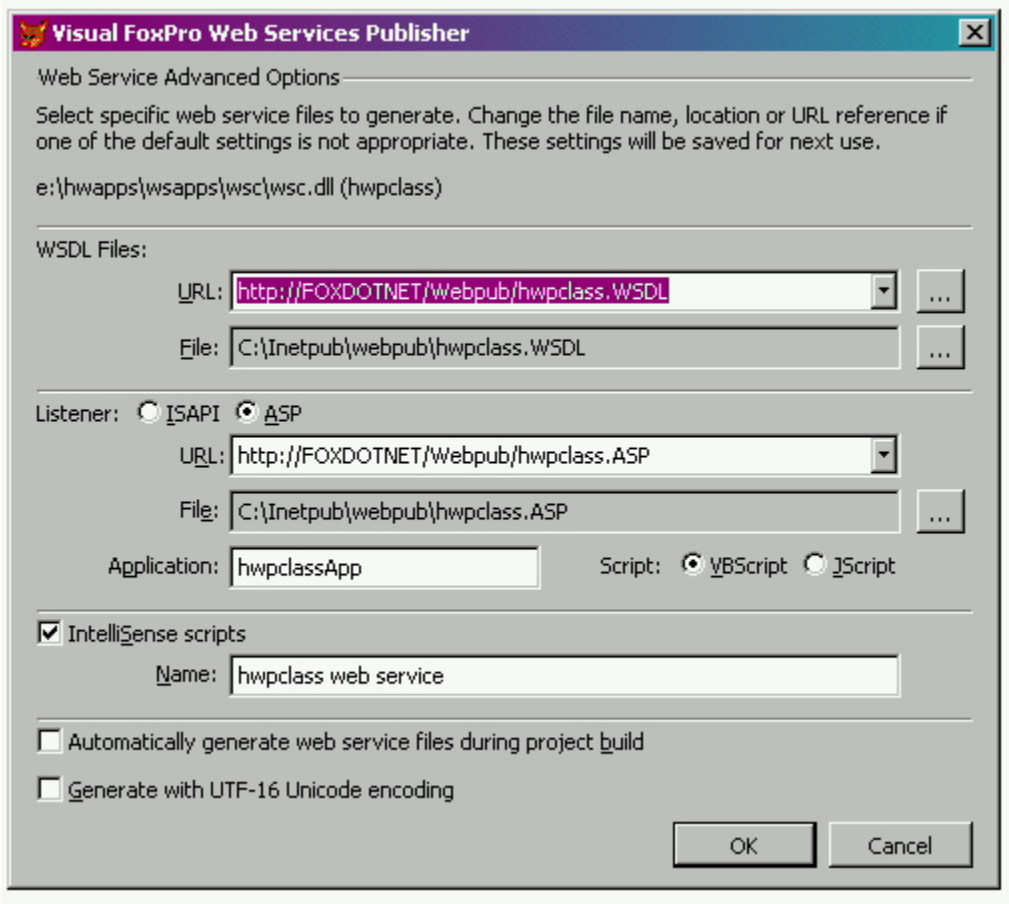

Figure 9. The Advanced Settings dialog.

If you click Advanced and make changes, click OK to return to the dialog in Figure 8. Then click Generate.

If you're regenerating WSDL and WSML files, you'll get the dialog in Figure 10. Otherwise, proceed to the next step.

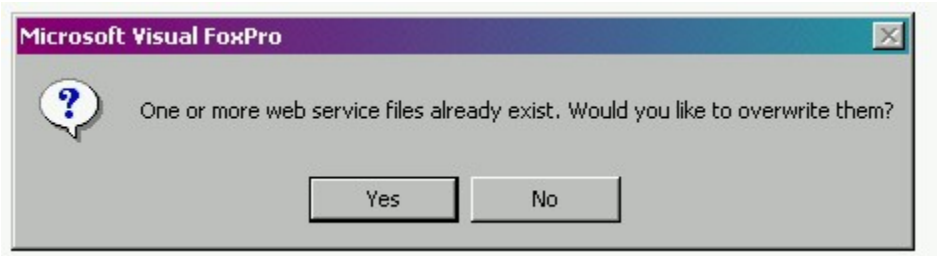

Figure 10. Visual FoxPro warns you if you're about to overwrite Web Service files.

The next step, as shown in Figure 11, shows you what happened. A COM server was created, WSDL was created, the listener was identified, and an entry in IntelliSense was created.

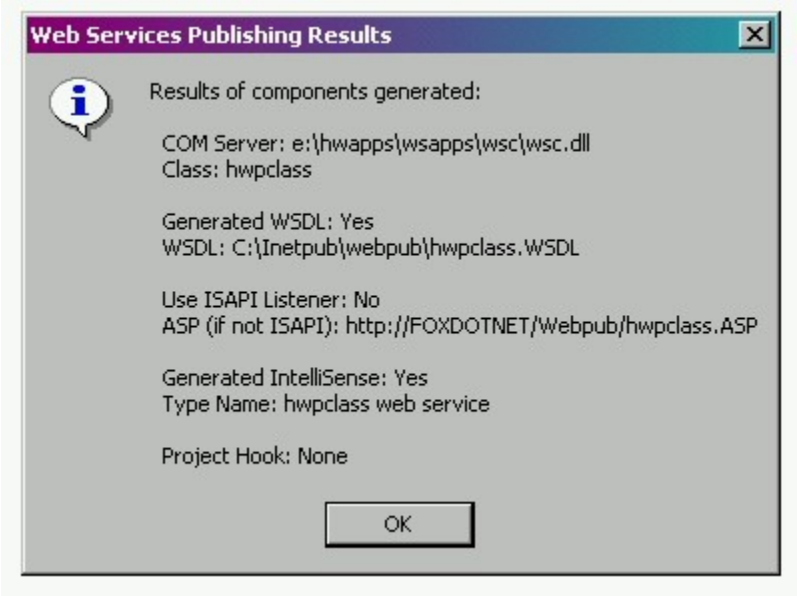

Figure 11. The results of generated the Web Services components.

And you'll see two new files, HWPCLASS.WSDL and HWPCLASS.WSML, in c:\Inetpub\Webpub.

If you checked the 'Automatically generate web service files during project build' check box in Figure 8, the builder will add a project hook to the WSB project, using the class Wsphook in the _webservices class library of the Fox Foundation Classes. Check out the Home text box in the Project tab of the Project Information dialog as shown in Figure 12.

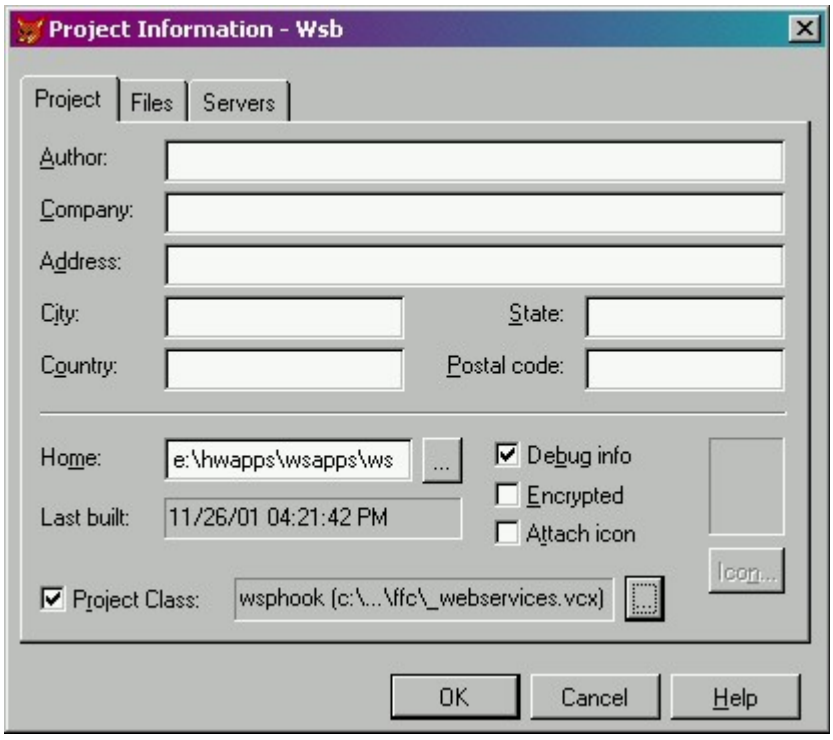

Figure 12. Checking the 'Automatically generate web service files during project build' check box results in a project hook being added to your project.

You've now created your WSDL and WSML files. These are similar to type libraries for COM – they describe methods and data types that are use in the Web Service.

Testing your Web Service on Your Development Box - I

In order to test your web service on your development box, you're using your development box both as a publisher (server) and a consumer (client.) That means two things. First, since you're running a Web Service on your development box, you'd better have web server software running on your development box as well. Describing how to do so is beyond the scope of this article, but I'll assume that you've got IIS running on Windows NT or Windows 2000, either workstation or server versions.

Testing your Web Service on Your Development Box - II

But since you're also a consumer, you'll have to get ready to consume it just as you would consume any other web service.

Thus, the second step is for you to have your web service ready - that means you have to register it. Here's you how register your own web service.

1. Load Visual FoxPro 7.

2. Select Tools | IntelliSense Manager

3. Select the Types tab as shown in Figure 13.

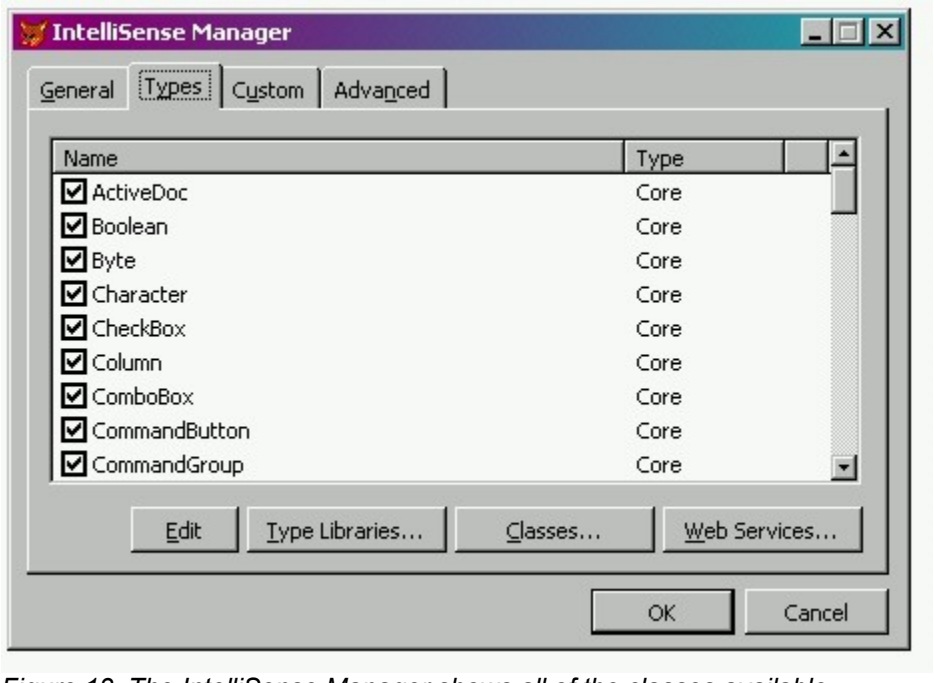

Figure 13. The IntelliSense Manager shows all of the classes available.

4. Click the Web Services button in the lower right corner to display the Visual FoxPro Web Services Registration dialog as shown in Figure 14.

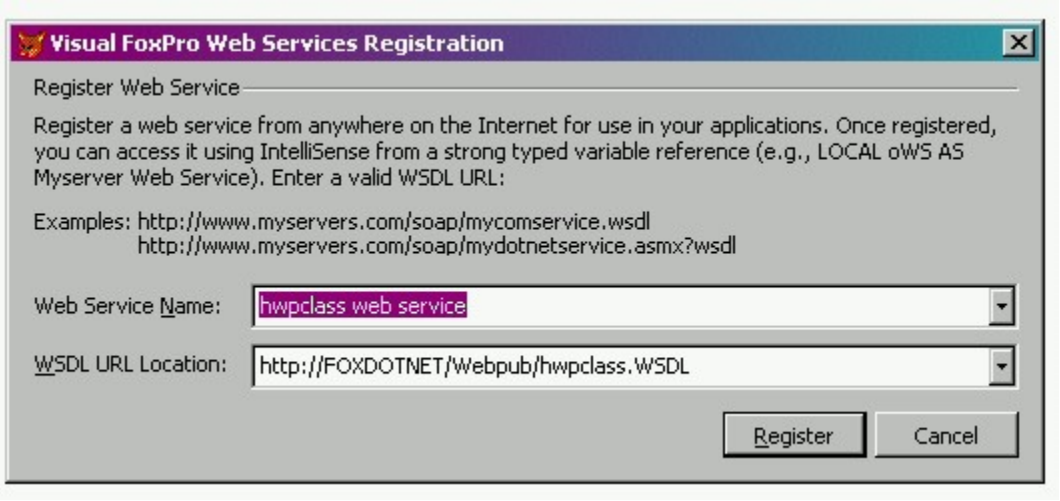

Figure 14. Registering a Web Service.

Change the values in the text boxes as needed (more on this later), and then click on the Register button. If everything works properly, you'll see the confirmation dialog shown in Figure 15.

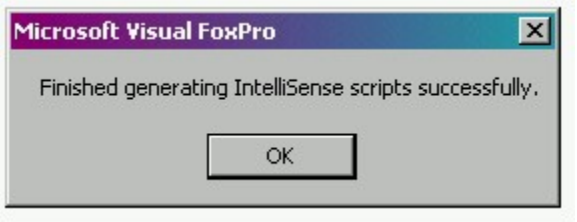

Figure 15. The IntelliSense confirmation dialog.

Once you've finished registering your Web Service, you can go back to the IntelliSense Manager and scroll through the list box and see the new entry, as shown in Figure 15.

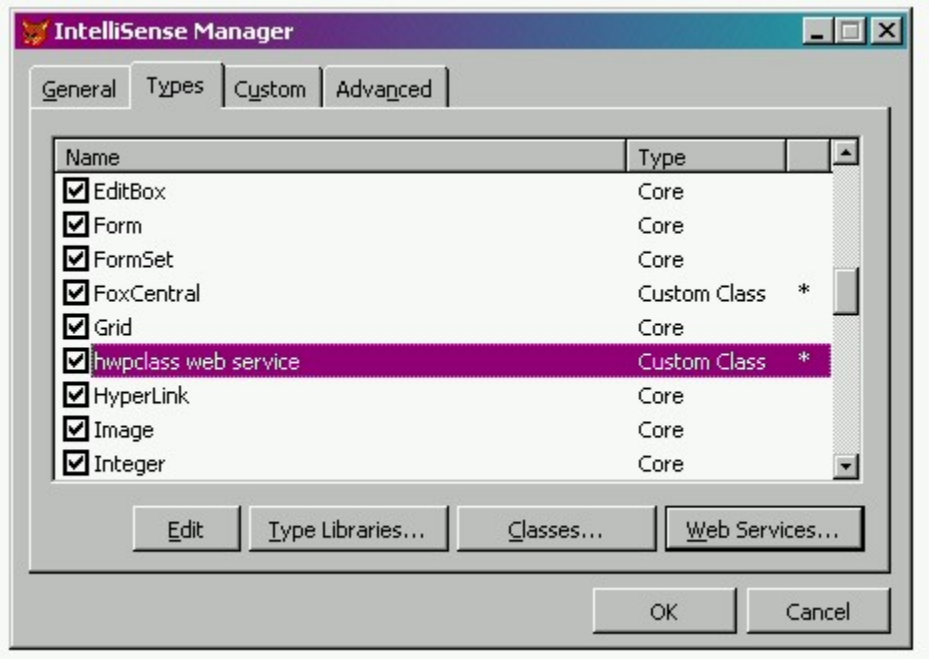

Figure 16. The IntelliSense Manager shows that your new Web Service has been registered.

Testing your Web Service on Your Development Box - III

Now that your development box has published the Web Service and your development box is also set up to consume, it's time to actually consume the service.

1. Go into VFP, switch to a new directory, and create a program called WSCTEST.
2. In the program, start typing

```
LOCAL ows as
```

As you do so, IntelliSense will display a list of available classes, including… YES! The new Web Service that you just registered, as shown in Figure 17.

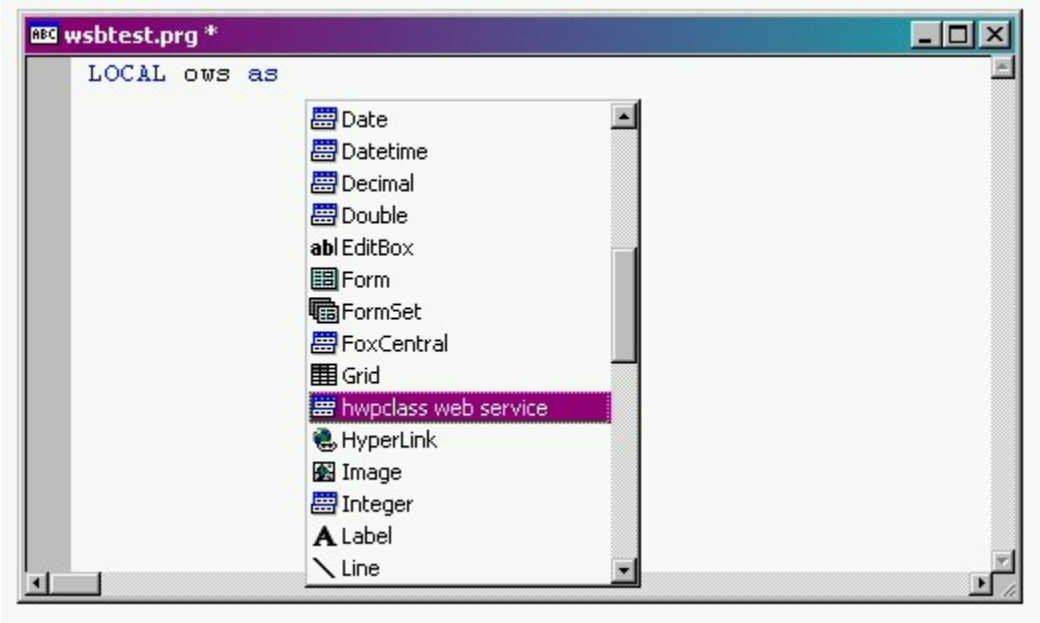

Figure 17. Visual FoxPro IntelliSense now includes your Web Service.

Press Enter to select the Web Service class and IntelliSense will fill in a bunch of stuff for you, as shown in Figure 18.

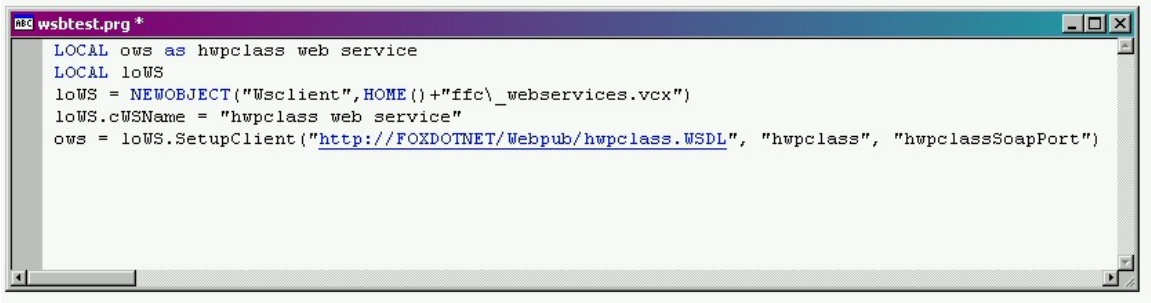

Figure 18. IntelliSense completes the Web Service entry.

In the code in Figure 18, the object reference ows is an object like any other – and you can access methods of this object. Since it's registered in IntelliSense, the methods of the object will be available. Our Web Service has one method, GetNews, that returns a string of text. You can get that text with a line of code like so:

```
m.lcX = oWS.GetNews()
```

As you see in Figure 19, you don't even have to remember the exact method – just start typing "ows." And the available methods will be listed in a list per IntelliSense's usual interface. The list in Figure 19 isn't very interesting – just one entry – cuz that's all there is in this Web Service class.

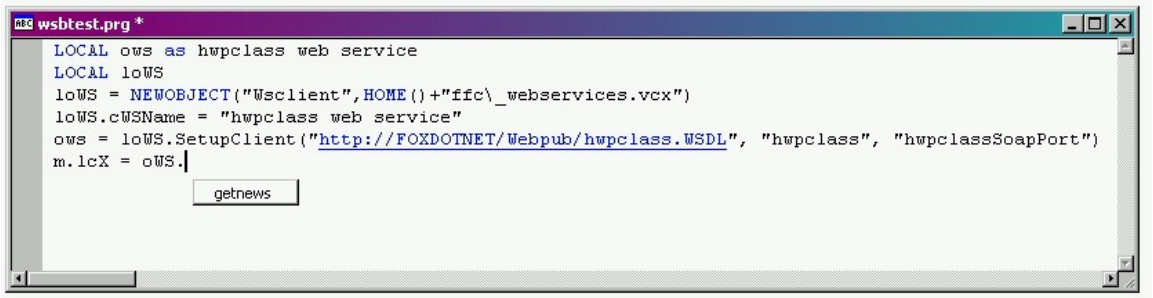

Figure 19. IntelliSense knows which properties and methods belong to the hwpclass Web Service.

For this first quick and dirty demo, I completed the program by just displaying the text string in a messagebox while running in development mode on my development machine:

```
messagebox(m.lcX)
```

and as shown in Figure 20.

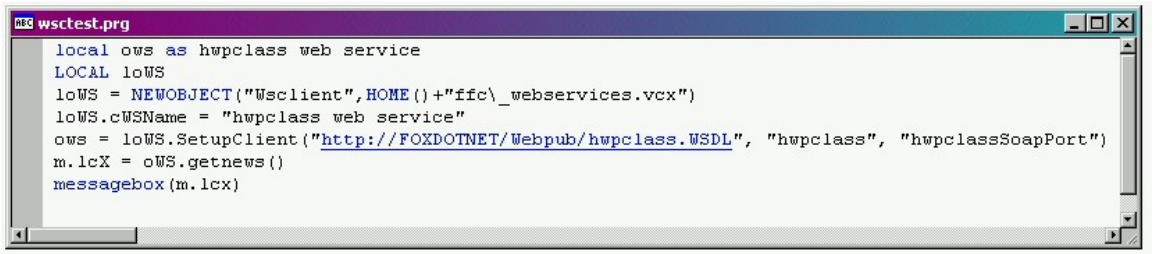

Figure 20. The completed test program for your development box.

Right click in the program window and select DO WSCTEST.PRG. If everything works properly, you'll see the results as shown in Figure 21.

Figure 21. The result of consuming the Web Service on your development machine.

Sure, it works for YOU, but when I try…

Well, actually it doesn't work for me either. Not the first time, at least. Or the second time, either. Why do you think we're on version "C" in this article?

The first error you might run into is shown in Figure 22.

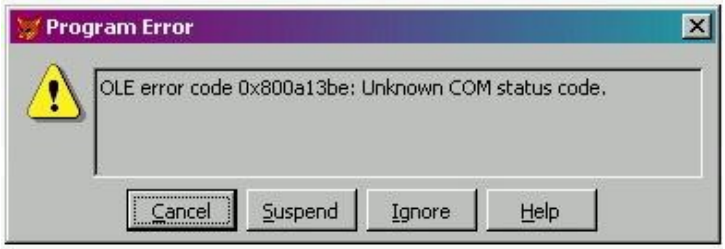

Figure 22. Your classic helpful error message.

Gary DeWitt, while helping me out with this article, said that in his experience, this type of message was often generated if the wrong listener was used (see Figure 9.) Visual FoxPro, by default, tries to register Web Services using an ISAPI listener, but the SOAPISAP.DLL listener is typically not configured in IIS. So you can either go into IIS and set up SOAPISAP.DLL, or just select the ASP listener as I did in Figure 9.

Another error message that I ran into occurred when running the WSCTEST program as shown in Figure 23. This error message is much more helpful – it indicates that it can't find NEWS.DBF. That's easy to track down. I'll show you the details next month; suffice it to say right now that the Web Service's default directory is the Windows System directory. For purposes of this article, I pointed the SELECT statement to look in the root of the local drive – once I copied NEWS.DBF to the root of C, all was fine and good.

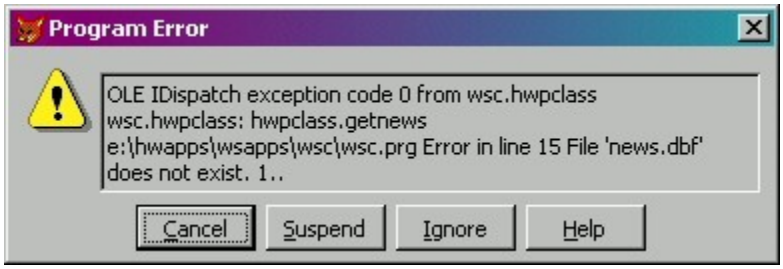

Figure 23. The Web Service can't find the table.

The Next Step

Now that you have your Web Service up and running on your development box, it's time to… deploy! Yes, that magical word that never seems to occur to the folks who create this stuff. Next month I'll discuss deployment of this simple Web Service on a live Web Server. After that, we'll discuss what's happening under the hood, so you understand what each of these choices means, and why you might want to make different choices based on your needs. Then we'll schmaltz up the service so that it's actually useful, and describe how XMLTOCURSOR and CURSORTOXML can be used to handle return values more interesting than a few characters. And I'll discuss a few more common errors as well. Stay tuned!

Whil Hentzen is editor of FoxTalk.