Figures:
File for Subscriber Downloads:

Your First Database Web Application – Part II

Whil Hentzen

*\\\ do we want to include a copy of the 3/99 article in the subscriber downloads or just point them to it?/// The URL to the article is http://www.pinnaclepublishing.com/FT/FTMag.nsf/Index/67A2E29A825CCC08852568E60076BAE1?opendocument

In March of 1999, I wrote an article titled "Your First Database Web Application" that walked you through the basics of how to set up a simple Web-based database application. Nearly two years later, a lot of people are just now approaching their first Web-based database application. In the meantime; I've received a lot of questions and comments about that article – either about additional details, more advanced features, or just general queries about the process.

And, since that article, the rate of change in the world of software development has, believe it or not, increased. As a result, it's even more important to become efficient during development – even with new technologies and new types of applications.

The purpose of this article, then, is to address all three issues – to again walk you through your first database application on the Web, building on the information in the March, 1999 article, answer questions raised and fill in gaps noticed since then, and to show you how to structure your development environment, and, finally, to provide a development strategy that will help you build your Web applications quickly and reliably.

The RAD approach

The term "RAD", or Rapid Application Development, has been around since the early 1970's when an IBMer named James Martin coined it to describe a process for accelerated application development. In the three decades since, though, the term has been bastardized in every conceivable way to the point that it's unrecognizable and virtually useless, except as a generic moniker for "cool and fast" by marketing folks. What we're after, though, what you're after is the implicit promise that you're going to be able to build web database applications faster.

You can go "faster" in a number of ways. For example, if you're driving from point A to point B, and you want to get there 'faster', you have several choices. One choice is to use the new-fangled car that the guys in the R&D labs just put together. It's really fast – really, really, fast, in fact, but it breaks down a lot. Thus, there's a trade off – it'd better go fast enough to offset the time you're going to waste while the car is being repaired.

Another choice is to use "old reliable" – the Volvo that may not be able to get out of its own way, but that also never breaks down. Sure, the new rocketcar may theoretically be able to get you there faster - if you never have problems. But you may also get stuck in a 1950's era gas station outside of Nowhere, Nevada for three days, waiting for a spare part.

Similarly, in software development, you can go faster by using the latest and greatest tools, or you can rely on 'ole paint'. The tradeoffs are similar. In our case, "ole paint" is an application framework that's been built and tested by virtue of hundreds or thousands of users banging on it from every conceivable angle.

There are several "application frameworks" for Web development with Visual FoxPro; I'll continue to use Web Connection (www.west-wind.com) to serve as the example in this article because it's popular, solid, and what I've been using for years.

Getting started

If you haven't read the March, 1999 article, now is a good time. The contents of the article are still valid, although I'll now argue that you really don't need three boxes to develop, test and deploy your application

anymore. These days, two boxes – a live Web server and a development machine that doubles as your test server - is plenty. For the remainder of this article, I'll primarily focus on this development box that is doing double duty. So, go ahead and read the article for the background on Internet plumbing and how a Fox application works with TCP/IP. Once you're done with the theory, we'll build our development machine from scratch again.

Installing a Web Server on your development machine

You can do the lion's share of your work on a single development machine running NT 4.0 workstation or Windows 2000 Professional (the equipment of NT4WS.) Both versions of the operating system come with free web server software that emulates the web server software you will have running on your live web server box that's communicating with the rest of the Internet.

By the way, in both cases, whether you're installing PWS or IIS, the installation routine will suggest a home directory for your website, and that suggestion will usually be c:\inetpub\wwwroot. This is a stupid suggestion, and a bad place. Don't use it. Instead, specify a directory on another drive, such as e:\inetpub\wwwroot\. Don't get totally stressed if you don't understand this now, or if you forget to; there are ways to change it later without any repercussions. Doing so now just saves you a step.

Installing Personal Web Server on NT 4.0 Workstation

Windows NT 4.0 Workstation's web server software is Personal Web Server, and is found on the NT Option Pack. Insert the CD, find the Setup program, and run it until you find a list of programs to install. Hunt for PWS and follow the prompts to install. Note that the Option Pack was released around the time of Service Pack 3, and thus if your NT install has later service packs installed, the Option Pack installation will complain. You can ignore the warnings.

Once installed, you'll should have a menu option on the Start|Programs menu called "Windows NT 4.0 Option Pack" which expands into several choices, including Microsoft Personal Web Server. You'll also get an icon in the task bar that looks like, well, a white glob with a couple of colored spots around the edges. The caption is "Personal Web Server is running" (or, if there's a red X over the icon, the caption will read "Personal Web Server is stopped".)

Clicking on the PWS taskbar icon, or selecting the Personal Web Manager menu from the Microsoft Personal Web Server menu option, will open the Personal Web Manager dialog as shown in Figure 1.

Figure 1. The main dialog of the Personal Web Manager in NT 4.0 Workstation.

Clicking on the Advanced button in the navigation bar on the left side of the dialog will bring forward the dialog tab shown in Figure 2.

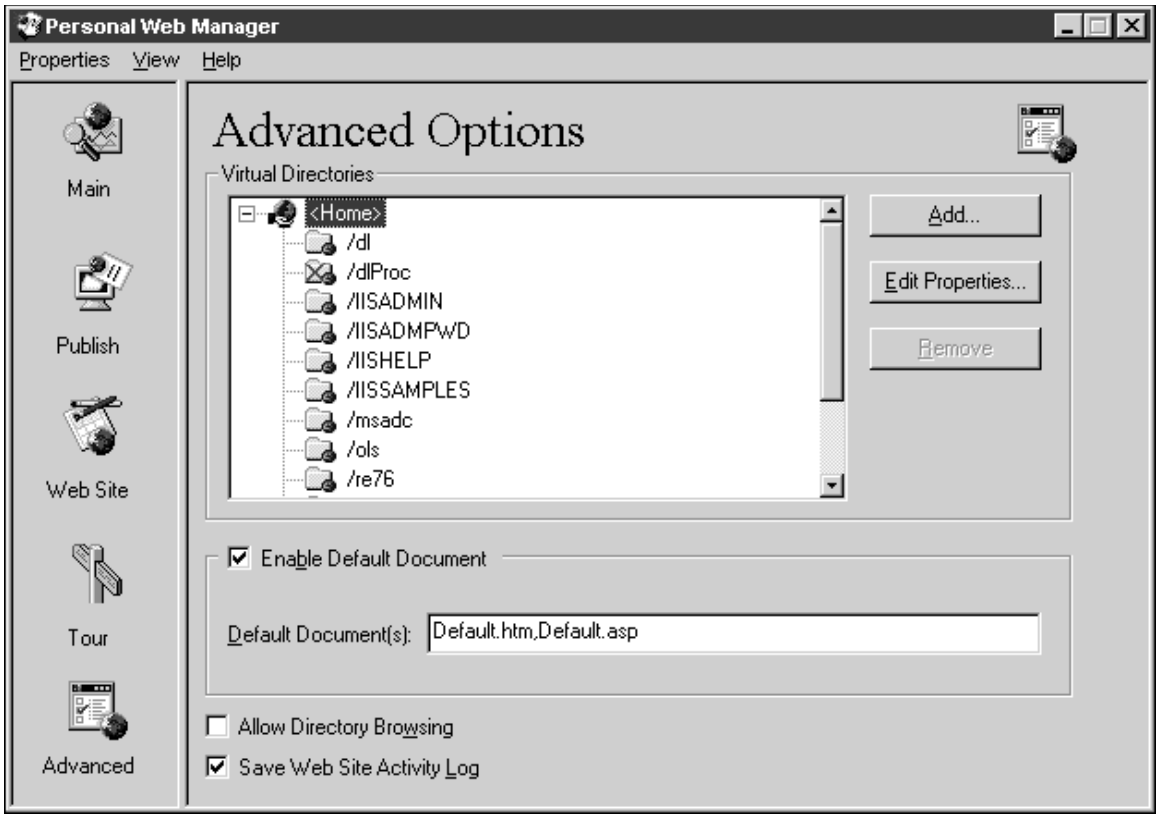

Figure 2. Identifying the default document in PWS.

Clicking on the Edit Properties button in the Advanced Options dialog brings forward the dialog shown in Figure 3.

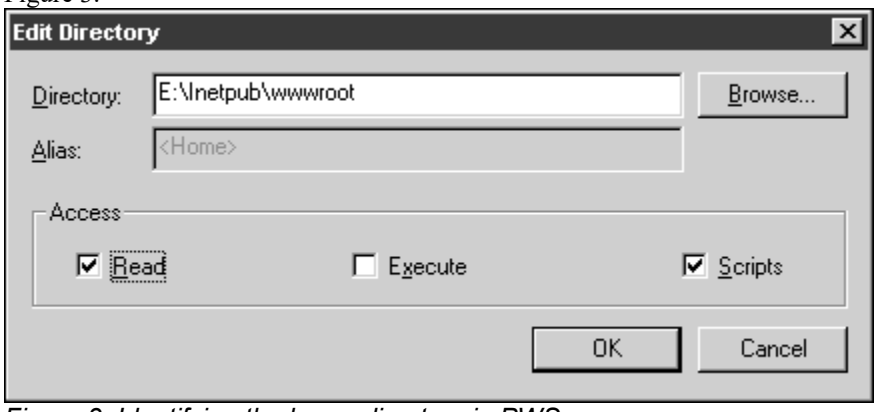

Figure 3. Identifying the home directory in PWS.

Installing Internet Information Services (IIS) 5.0 on Windows 2000 Professional

Installing IIS 5.0 on Windows 2000 Professional is even easier. Select the Add/Remove Programs applet from the Start|Settings|Control Panel menu to bring forward the screen shown in Figure 4.

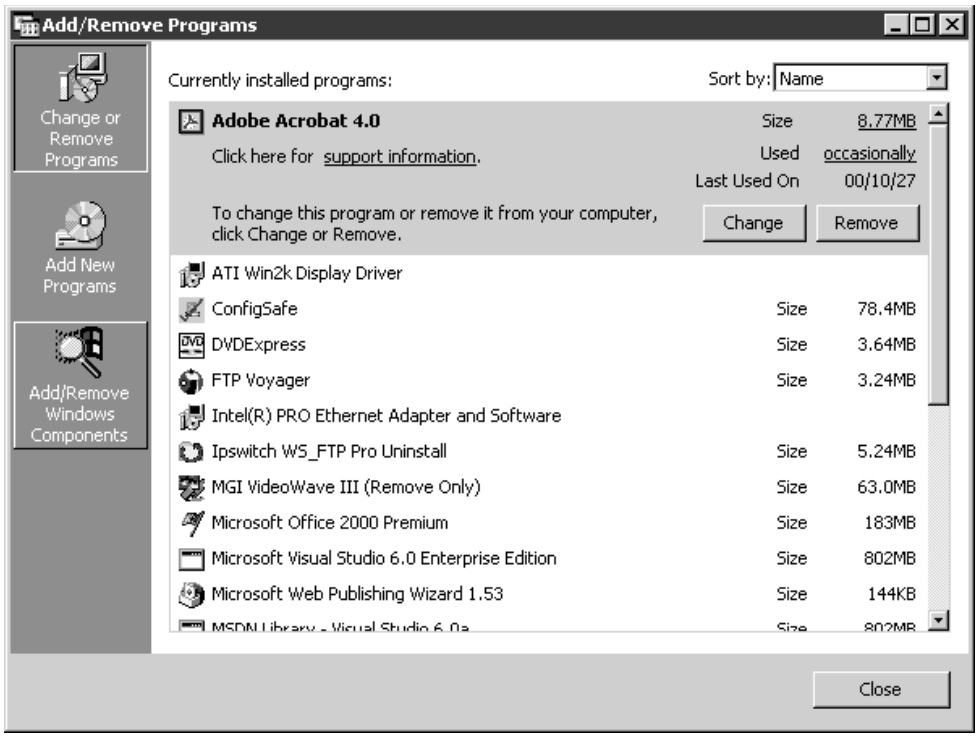

Figure 4. Installing IIS 5.0 in Windows 2000 Professional.

Select the Add/Remove Windows Components, select the IIS 5.0 option in the next dialog, follow the prompts, and you're done. It took me all of 90 seconds and two button clicks to accomplish the whole process. This is truly how computers and software should work.

Getting to IIS once it's installed, however, is another matter. If you're used to NT 4, you could take twenty or thirty minutes trying to find out where IIS has been hidden in W2K. Select the Administrative Tools applet in the Start|Settings|Control Panel and then click on Computer Management to bring forward the dialog shown in Figure 5.

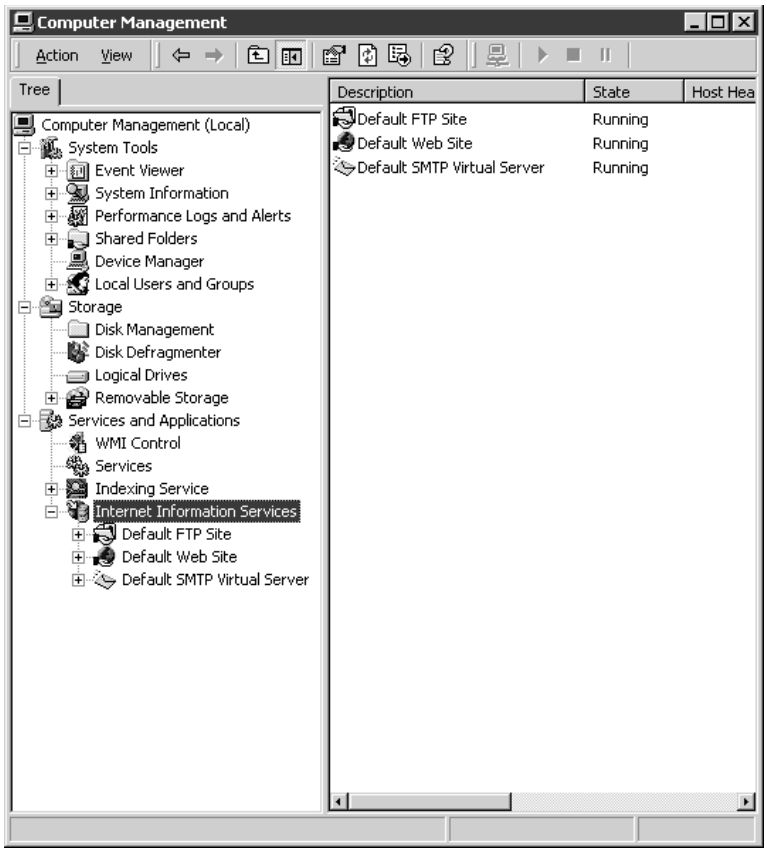

Figure 5. The Computer Management dialog provides access to IIS 5.0 in Windows 2000.

Hint: This is a real pain, so you'll want to make a shortcut to the Computer Management dialog and put it on your desktop, or, even better, your taskbar.

Right-clicking on Default Website and selecting Properties will bring forward the dialog shown in Figure 6.

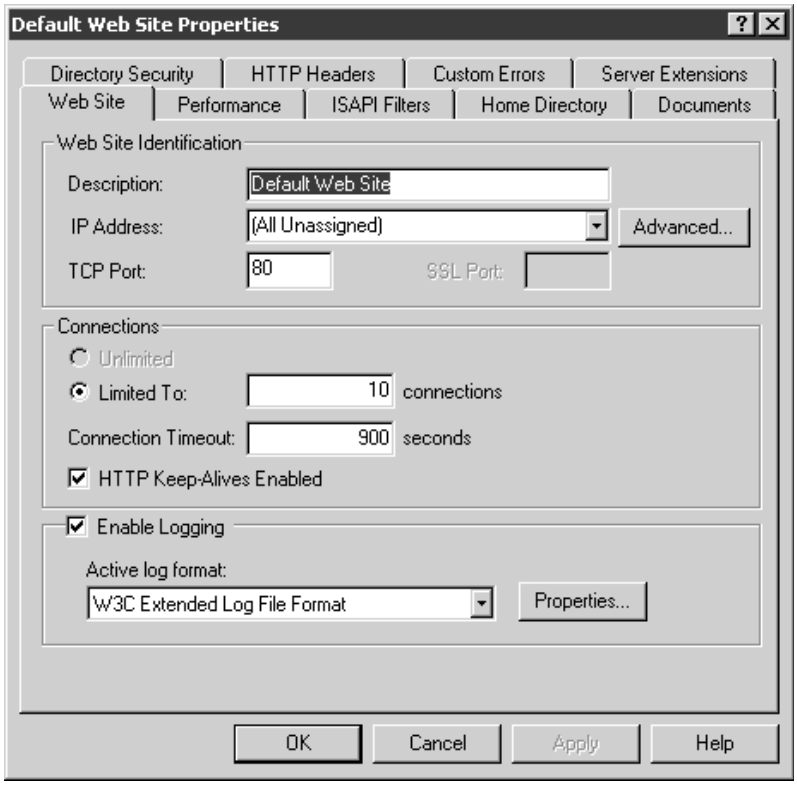

Figure 6. Clicking on the Properties menu from the Default Web Site produces the Default Web Site Properties dialog.

Clicking on the Home Directory tab in the Default Web Site Properties dialog will bring forward the dialog tab shown in Figure 7.

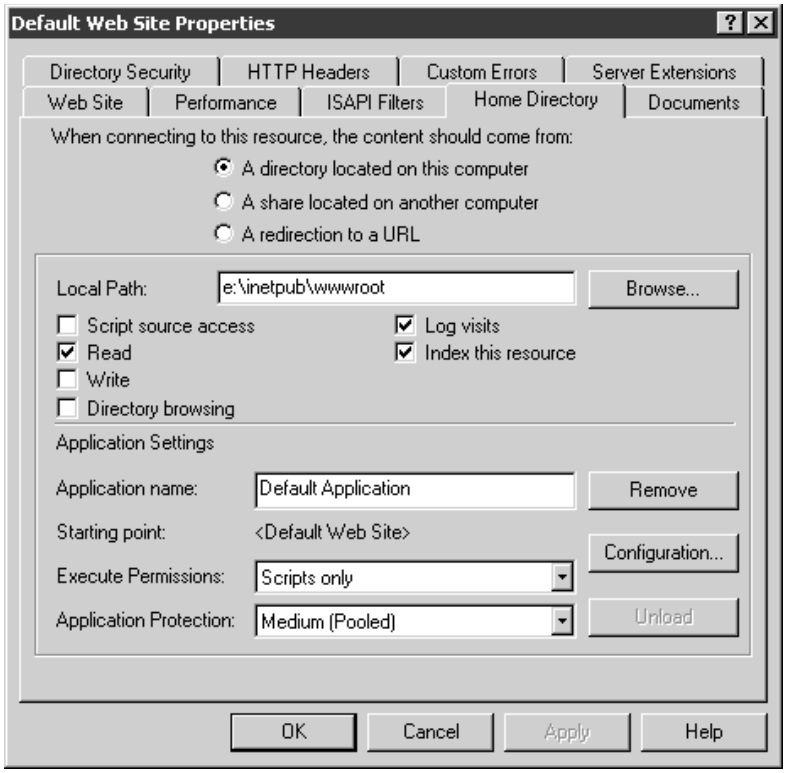

Figure 7. The Home Directory dialog is used to specify the home directory for IIS 5.0 in Windows 2000.

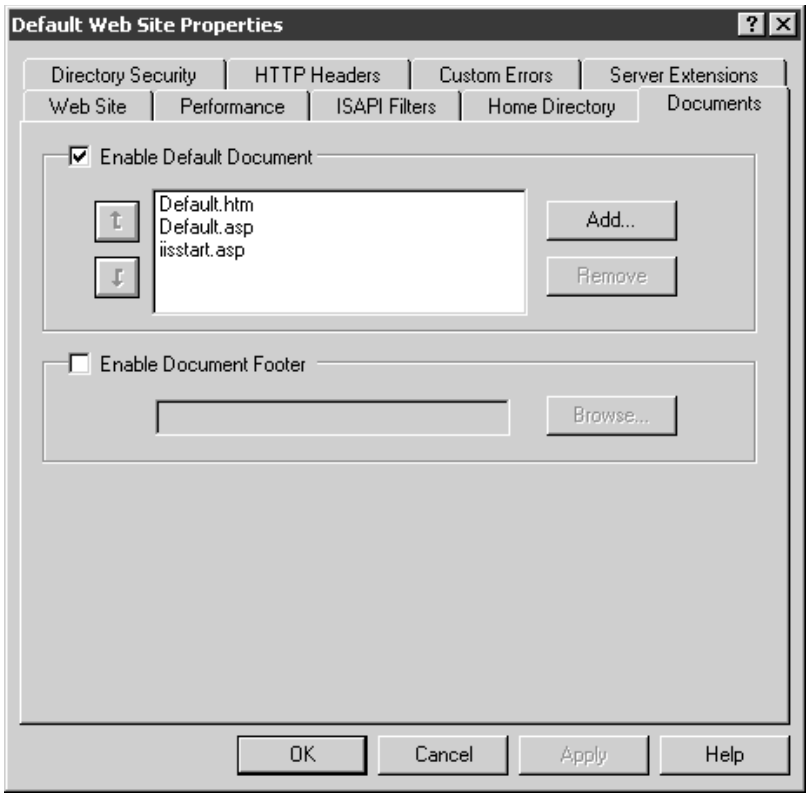

Figure 8. The Documents tab is used to specify the default document for IIS 5.0 in Windows 2000.

The dialogs in Figures 2, 3, 7 and 8 do essentially the same things – allow you to access and configure the home directory of the web server you're going to have running on your development box. I'll come back to these two dialogs shortly.

Configuring your web server

Once you've got your web server software installed on your development box, you'll want to configure and test it to make sure it's running. It's pretty frustrating to skip this step, go through the development of an application, and then try to troubleshoot why the app isn't running – not realizing that without an operational web server, all else is moot. This is the equivalent of not having the printer plugged in.

As you recall from the first article, you can enter the name of a URL that either ends in a directory, or that ends with a file name, in a browser address window, like so:

www.FoxRocks.com
www.FoxRocks.com\OpenSesame.htm

In both cases, the web server software at the FoxRocks.com website has been configured to point to a specific directory on the FoxRocks.com web server box. This specific directory is the home directory (such as c:\inetpub\wwwroot) that you identified during installation of the web server software.

In the first case, the web server software will automatically deliver up the file that has been identified as the 'default' document; in the second case, the web server software will look for a specific file, named OpenSesame.htm, and if found, deliver that file to the user via their browser.

Since you are, for the purposes of this article, the webmaster of the FoxRocks.com website, you're responsible for doing this configuration. You do this through the dialogs shown in Figures 3 and 7. But first

a word of advice for setting up the directory of your website on your development box (as well as your live web server box.)

Remember when I told you not to accept the suggested home directory of c:\inetpub\wwwroot during the installation of your web server software? If you glossed over that instruction back then, you can change it now. Here's how.

With PWS, you point to the home directory of your web site using the Edit Directory dialog shown in Figure 3. Either manually enter the correct path in the Directory text box, or use the Browse button to navigate to the directory. With IIS, you edit the contents of the Local Path text box (you can point to it via the Browse button) in the Home Directory tab of the Default Web Site Properties dialog.

You'll most likely want a your web server to display a default page in the event that the user doesn't specifically enter the name of a file. You can create an HTML file and put it in your home directory. You can leave its name as "default.html" or use your own name. If you use your own name, use the dialogs shown in Figure 2 and 8 to identify that file as a default document.

Finally you're ready to test your web server. With PWS, you should see the icon in the task bar, and the caption should indicate it's running. With IIS, you can see your web site identified in the Computer Management dialog in Figure 5, and see that its status is "running" in the right pane. In both cases, you can right-click and choose between Start, Pause and Stop Service options.

Testing your web server

Now, to test, bring up your browser, and enter the name of a file in the web site directory structure. The default file is probably the best choice for your first test. First of all, don't try to be clever and bypass the browser by double-clicking on the file name in Windows Explorer. By doing so, you're simply opening the file via the operating system – not via the HTTP protocol that is executed by the browser. And you don't want to enter the name of the file via the directory structure of your machine, like e:\wwwroot\inetpub\default.htm either. Again, you're not accessing the file via HTTP.

So what do you do? How do you do this on your own machine? You know how to do this on the Internet – you enter the URL and then the file name. But, your own machine doesn't have a URL, does it? Well, actually, it does. It just doesn't end in ".com" like most other URLs. You can either use your own machine name, like so:

http://YourMachineName/default.htm

or you can use the generic "localhost" designation, like so:

http://localhost/default.htm

In both cases, the default.htm file must be located in the home directory as identified in your web server configuration. Now, to make sure you've got this down, suppose your web server home directory is e:\inetpub\wwwroot. Further suppose you've got a subdirectory under wwwroot called badger, and there's a file called customerlist.html. To render that file through your web server software, the following links would work:

http://YourMachineName/badger/customerlist.html
http://localhost/badger/customerlist.html

Organizing your development machine

So far, you've got web server software running on your development machine, and a directory identified as your web site. You'll eventually publish your entire web site in that directory (along with whatever subdirectories you need to hold all the myriad files your website consists of.) Actually, you don't need to put your entire website there – only the ones that are going to be referenced by your database application.

If you've got a main page that has a hyperlink to a second page, and on that second page is a link that calls your database application, then you don't need all of the other files that support your main page, such as images or files referenced elsewhere. However, if your database application will reference other parts of your website, then you'll need to have those files available as well.

If you plan on having more than one application running from your website, it would be a good idea to create separate application directories underneath wwwroot. For example,

e:\inetpub\wwwroot\MyAppOne
e:\inetpub\wwwroot\MyAppTwo

This way you can keep each application's files separate, and reduce the risk of one application stomping on another application's files. You can keep each application running separately – if you have to reconfigure or bring down one app, you're less likely to bring down another app if it's located in a different directory.

But there's a lot more than just your web site directory on your development box – you're developing an app, right? Well, let's look at what else you need.

First, you'll need to set up your application framework, if you have one. If you don't, and you're writing your own, you'll still need a place for your Web development class libraries. Second, you'll have to have a place where you keep the source code for your projects – obviously, different directories for each application. And, finally, you'll need to store your data somewhere. Unlike LAN applications, where you're probably used to keeping your data in a subdirectory under your main application, you'll want to specify a data directory elsewhere.

Setting up Web Connection

Setting up Web Connection is generally very easy – run the installation program and follow the prompts. There are a couple places where you might find additional information useful.

First of all, this should be obvious, but in case it isn't, you'll need the Visual FoxPro development environment on the machine that you're installing Web Connection on. You are building VFP apps with WC, and, more directly, WC will run some VFP programs upon conclusion of the its installation.

Second, it'll ask you where you want to install WebConnect, but the wording isn't clear. Web Connection is an application framework, much like FoxExpress, MaxFrame or Mere Mortals – and thus comprises both a set of framework classes from which you'll build your apps, and some wizards and programs that help you build your apps.

In this step, you're selecting where you want to install the Web Connection framework and wizards. It will default to drive C, but I would argue that it doesn't belong on drive C anymore than any other data or development programs belong on C. I suggest that you install Web Connection on your development drive, such as drive E. The default directory is \wconnect.

Note that when you create a new project in Web Connection, the new project wizard will put that new project in this Web Connection installation directory. I'll show you how to change that to a better place shortly.

Third, you'll be asked to select the location of your web server's home directory (e:\InetPub\wwwroot); select the directory you made on drive D or wherever – not the default on drive C.

Setting up development project directories

Every developer has a different way of setting up their projects.

Suppose you've got your website located in e:\InetPub\wwwroot, and your development drive, including the Web Connection framework classes and wizards in e:\wconnect. (They don't have to be on the same drive.) Now you'll need to set up a directory for web application development.

I create top level directories on my development drive for each customer. Under that top level directory I create subdirectories for each application. You could follow this structure, or you could create a separate top level directory for all of your web applications. For sake of convenience during this article, I'll go with the second choice. Under the top level web applications directory, create separate subdirectories for each separate web application, like so:

e:\WebApps\MyWebAppOne
e:\WebApps\MyWebAppTwo
e:\WebApps\MyWebAppThree

Under each directory, create the normal directories that you use in your day to day development life. For example, I put the project and the final executable in the root of the application directory, and the source code underneath the application directory in a subdirectory called SOURCE. So the contents and structure for AppOne would look like this:

e:\WebApps\MyWebAppOne\AppOne.PJX
e:\WebApps\MyWebAppOne\AppOne.PJT
e:\WebApps\MyWebAppOne\AppOne.EXE
e:\WebApps\MyWebAppOne\Source\<lots of source code files>

Setting up data directories

For simplicities' sake, consider a set of directories that will hold the data for your application that are located in the root of your development drive. There's still enough complex stuff going on that you don't need to be messing with pathing problems at this juncture.

For example, you would have a top level directory named E:\WebData. Under that directory, you would have subdirectories for each application, like so:

E:\WebData\AppOne
E:\WebData\AppTwo
E:\WebData\AppThree

The cardinal rule is to never put your application data in the directory structure under your web site's home directory. Locating your data elsewhere will greatly reduce security problems.

Mapping your development machine to your live machine

Just as with LAN applications, you want your development environment and the live production environment to match as much as possible. However, there's not a one-to-one mapping of development files and live files.

First of all, you won't have your application framework running on your live machine, so there's no need for a WCONNECT (or similar) directory on your live machine. Similarly, you won't have your source code or development files on your live server.

However, you will have a web site directory – duh! – and it should look like your development box's web site directory. It'll be easier on you if you've got them structured them same – both in iNetPub\wwwroot, for example.

You may also want to set up your data directories in the same way on the live box as they are on your development box. At some point, you may be pointing to data somewhere else – for example, on a different box, or even in a different application, such as SQL Server or Oracle. Alternatively, you could use a configuration file that points to the data source – and use different config files for development and live machines. Having a "secret" switch in code that points to one relative data location during development and another during production, though, is just going to cause you problems.

Conclusion

You've now got your machines setup, your Web server installed, and Web Connection ready to go. In next month's article, I'll discuss setting up your project, and introduce a strategy for developing Web application programs quickly and reliably. Stay tuned!