Issue Date: FoxTalk June 1996

The Call of the Internet

Rick Strahl
rstrahl@west-wind.com

Many experts are predicting that in the next two years the Internet will cause a software industry paradigm shift as significant as the one that occurred 15 years ago when the IBM PC was born. So how do you get started? A new *FoxTalk* series will focus on the Internet and how you can leverage your existing FoxPro skills with it. Rick Strahl kicks off this series with an introduction to developing database applications for the Internet. He's brought together a lot of information you need to get up to speed quickly.

The Internet and its surrounding technologies are hot. More and more businesses want to hook their existing database applications into the World Wide Web. Whether you'll be hitting the wide open Internet or building applications for a company's internal "intranet," Web-based development is coming at you in a big way! Most software companies -- and especially Microsoft -- are promoting Web-based development, and it won't be long until it drastically changes the way you build applications.

For the last several months I've been developing Internet-related applications using Visual FoxPro exclusively. Since I started, the demand for Internet apps has been tremendous and the results using Visual FoxPro have been excellent. I converted several existing applications to Visual FoxPro for speed reasons -- the move from ODBC to a Visual FoxPro data back end resulted in a three- to five-fold increase in query speeds even while reducing the load on the Web server's CPU. Visual FoxPro makes a great database back end for Web applications in many situations (more in a moment).

In the process of building these Web apps, I developed a front end called Web Connection for Visual FoxPro that I use to connect Visual FoxPro to various Web servers. I'll share some of the insights I gained in developing these applications, as well as the framework that I used, both to stir additional interest in FoxPro-based Internet development and to provide you with a rough roadmap for getting started. I'll follow up in later months by going into greater detail on specific topics.

Can you spell W.E.B.?

If you don't have an Internet account yet, or you haven't looked at what's happening on the Web, do so now! You might think that Web browsing is a waste of valuable time, but the technologies behind the Web are driving the future of desktop computing and application development. There are vast amounts of information on the Internet that can be invaluable, which is why the Web is expanding as fast as it is. New sites are appearing continuously -- sites that will need database capabilities once their owners realize the power of creating dynamic Web content!

As a database developer, you should realize that all of the newest development tools and strategies are focusing on the Web not only as a medium to make data available publicly, but also as a medium to distribute and present data internally within companies and inside your own computer. You've heard a lot about distributed computing over the last few years, but the Web might be the catalyst that brings this paradigm to the masses. The key idea is that the front end -- a Web browser -- can access any available back end -- a Web server -- for information using a common platform-independent interface. Microsoft plans to integrate the entire Windows operating system using a common Web browser-type interface.

While most of the attention today is focused on the World Wide Web and the wide open Internet, this is only part of the Web story. Many analysts, industry experts, and corporate MIS managers view the intranet as an opportunity to both centralize their applications and distribute them more widely. The Web's ability to connect computers with a simple protocol (TCP/IP) and a standard interface (the Web browser) means applications can be made widely available, yet maintained centrally. Instead of installing applications locally on each user's machine, users access the application via a Web browser running a Web-based application stored on an application or data server somewhere on the network. Not only is the application centrally maintained, but it's also immediately accessible across the network, not only to the local workgroup but throughout the company. And since Web browsers are hyperlink-based tools, it's as easy as embedding appropriate links to other related applications or tools either inside of the company or on the external World Wide Web. This is what distributed computing was meant to be.

The message is clear: The development world is going to change in a big way. The new paradigm is still in its infancy and the tools will become easier to use and more powerful. But knowing the basics of how the Internet works will be a critical asset as you enter this new world of development.

Where do we fit in?

Until recently, much of the content on the Web had been static, consisting of pages laid out in advance and linked together statically via hypertext links embedded in HTML (Hyper Text Markup Language, the page description language used to display pages in Web browsers) pages. While it's nice to look at well-designed static pages, they're fairly limited. The information is likely to become out of date, and the user has to browse manually through large amounts of data. Even if the data source is just moderately complex, maintaining this kind of setup is a daunting task because groupings, categories, and so forth have to be constructed manually. Static pages are great for static information or certain types of formats: home pages, individual product information pages that need lots of custom graphic content, personal information pages. However, if you need to display typical database information that involves large amounts of data, the manual update concept falls apart quickly.

The true potential of the Web can be realized only from information that is generated dynamically and fed by up-to-date information contained in databases so that users receive only the data they ask for. If this sounds like a typical database application, you're right! This is where we as database developers come in. It will be our job to make the connection between existing data sources and the Web.

Web-based database applications are client/server jobs. If you're using Visual FoxPro as the data back end, you'll be implementing the server side using Visual FoxPro code. The concept is simple (more details later): The Web browser, which provides the front end "application" that the user sees, requests data from a Web server, which sends static pages and dynamically generated output to the browser, which in turn displays the combined page and dynamic data. A Web server is simply a piece of software that runs on a machine connected to the Internet. The Web server in turn calls your Visual FoxPro application to provide that data. This means your Visual FoxPro code has to figure out what the Web server is requesting, run the code to process the relevant queries/updates/inserts or other code, and then return an HTML document that either displays the requested data or a status message indicating that the task was completed successfully.

Keep this server concept in mind as you get started, since FoxPro-based Web server development involves very little user interface programming. The user interface is handled entirely by the Web browser, although you'll probably be building some of those Web browser input pages with data from a database! You'll also be generating a lot of HTML output.

The now and the then

The technologies to make this happen are changing and improving at blinding speed and today's hot technology is yesterday's forgotten relic. Without a doubt, Web application development will get easier. Some exciting technologies are on the horizon from Microsoft, Netscape, and a variety of other vendors. But while a lot has been promised and some tools have been previewed, few of the promised advanced tools are available in production form (as I write this, anyway). If your goal is to create public applications for the Web, you also have to deal with compatibility issues. Microsoft and Netscape are duking it out with new, incompatible standards that make it hard to adopt one or the other's standards for development or page layout.

The newest technologies are coming, but what works today will probably work in the future. Besides, tried and true methods are often better than the hottest new trends, especially on the Internet where buggy beta or even alpha software is becoming the norm.

Microsoft and the Web

Microsoft has announced an incredible amount of information and new tools. First, and this should come as no surprise, they're pushing ODBC in a big way. Microsoft is giving away its new high performance Internet Information Server, which provides Web, FTP, and Gopher Servers to run under Windows NT. IIS comes with built-in database connectivity via the Internet Database Connector (IDC). The IDC is a SQL-based scripting mechanism that allows you to directly access any ODBC data source installed on the Web server. It's a simple interface that can be set up quickly to generate simple, dynamic queries based on user input. You can define SQL statements that include input from HTML forms, the Web server, and browser status information. The output can be formatted using a simple macro language that allows for minimal programming logic to be embedded into a scripted HTML template file. To do more complicated programming with the IDC, you have to dig into SQL stored procedures and triggers, which can get messy.

The other big news is a set of technologies that Microsoft has vaguely termed ActiveX. The idea is that this technology will move us from static to dynamic or *active* Web content. The active portion of this technology is heavily focused on two areas. The first is a set of Web browser enhancements that will allow scripting via VBScript, JavaScript, and other languages that can be hooked into the next generation browsers' open scripting engine. The other is support for "active" OLE controls (OCX controls with some added functionality for Internet compatibility -- downloadability and self-registration, for example) that can be controlled by the scripting language. In addition, the browser can be used as an OLE document container, so viewing any OLE document, including full editing capabilities, can be handled within the familiar browser interface.

These tools will allow you to build more interactive Web applications that can have browser side, interactive data validation and field interaction, real GUI interface operation for tasks such as autofilling fields based on user selections, showing or hiding additional areas of a form, and so on. (Standard HTML forms lack any interactivity -- once displayed, the page has to be changed and reloaded). In addition, support for OLE controls means that the third-party tools the VB community has been working with for years will be enhanced to do Internet-related development; expect to see them sport controls for interface gadgets, multimedia, and even self-contained application objects.

The full ActiveX promise isn't here yet, but Microsoft is pushing hard to get it out -- the browser-side extensions are promised by the middle of the year while the server-side scripting extensions aren't expected until the end of the year.

Visual FoxPro and the Web

So with all of this built-in database connectivity and VBScript as a scripting language, where does Visual FoxPro fit in?

As always, your motto should be, "The right tool for the job!" The current Web tools and even the proposed Microsoft ActiveX extensions lack what developers of modern development languages take for granted: flexibility. Tools like Visual Basic and Visual FoxPro have tremendous functionality that allow you to build complex applications quickly. It will take a while for Web-only tools to catch up in power and flexibility.

The current crop of Web tools provide crude macro languages or plain CGI interfaces using cryptic scripting languages that are ill-suited for heavy-duty data access. There are also problems with some of the new technologies I've just described. The Microsoft Internet Database Connector is very limited: You can supply only a single SQL statement per request and the scripting language lacks many common constructs to build output properly. ODBC-based data sources other than remote SQL servers, like the Access, dBASE, or even the Visual FoxPro ODBC drivers, have to run on the same machine as the Web server, which makes them unscalable if the load on the servers get heavy. In my opinion, Web servers and database servers don't belong on the same machine or both will bog down quickly under load. As you probably know, ODBC is fairly slow at accessing data with local data engines and imposes a heavy processor load while processing queries; both are problematic for the high transaction volumes typical of Web requests.

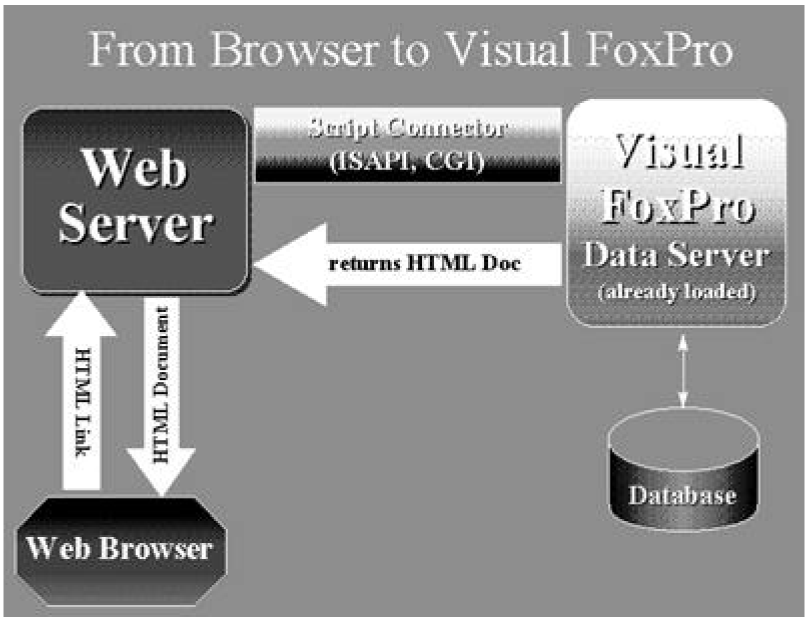
Visual FoxPro provides a great middle ground as a data server back end between a full-scale SQL server installation and small-scale low-volume applications that can get by with Web server local ODBC access. Visual FoxPro's speed is a major asset in this scenario, since it can provide data access three to five times faster than most local ODBC data sources (even the new VFP ODBC driver is much slower than VFP natively), especially if the FoxPro server is offloaded on a separate machine on the network.

Is Visual FoxPro always the right choice? No way! If you're dealing with a very large volume of data and simultaneously experience heavy demand on the Web server, then there's nothing better than a true multi-threaded, secure, SQL server like SQL Server or Oracle because they can scale the load on a single high-powered machine. But for smaller installs, and even moderately heavy loads, a SQL Server or Oracle setup is overkill, and your clients will be much better served by one or more VFP data servers. One of my Visual FoxPro Web installations is serving 20,000 CGI requests daily with more than one request per second at peak times; all are running off a single Visual FoxPro server session that has shown no sign of maxing out yet. And this setup can easily be scaled by adding more Visual FoxPro sessions. Finally, don't forget the licensing fees required for SQL Server -- Microsoft requires a $3,000 license fee to connect SQL Server to a publicly accessed Web Server.

Connecting Visual FoxPro to the Web

The key to making traditional development languages like Visual FoxPro work with Web servers is to use an interface that allows FoxPro to read the information the server provides about itself, the client browser and the HTML form that might have generated a request, and then passing the HTML or HTTP request output back to the Web server. Figure 1 shows the flow of data from a Web browser to Visual FoxPro and back. Note the intermediate "connector" that connects Visual FoxPro to the Web server. Such a connector is required in order to allow Visual FoxPro to act as a data server, rather than being called and loaded directly as an executable from a script call generated by an HTML link. You need the connector because VFP is too large to be loaded and unloaded for each individual request. Instead Visual FoxPro needs to act as a pre-loaded server waiting for incoming requests, instantly springing to life when a request is received.

Figure 1. Flow from the browser to Visual FoxPro and back.


From Browser to Visual FoxPro
Web Server
Script Connector (ISAPI, CGI)
Visual FoxPro Data Server (already loaded)
returns HTML Doc
HTML Link
HTML Document
Database
Web Browser

A request starts on an HTML page as either a hypertext link or an ACTION command generated from an input form. The link or form contains the name of the script (an EXE or DLL that references the connector application) that is to be executed when the link is clicked or the Submit button is pressed on the form. A typical script link might look like this (all on one line):

```
<A HREF="http://west-wind.com/wwcgi.dll?
   ShowItems~Hardware">Show Hardware Items</A>
```

In essence a call to a script is treated in the same fashion as a link to another HTML page, except that instead of simply opening and displaying a static page, a script executes and returns an HTML document. You can pass "parameters" as part of the link following the ? on the URL portion of the HREF link. These "parameters" can be used by Visual FoxPro to decide what it needs to do when it receives this request.

The connector application that passes this request to VFP needs to provide a high-speed interface without demanding a lot of Web server resources. Therefore they're usually written in C or another low-level language so they have a small executable. Traditionally these connector applications are EXE files written using the Common Gateway Interface (CGI), a standard

Internet protocol that describes how input and output are to be handled. A newer, Microsoft-sponsored standard called ISAPI (Internet Server API) provides similar functionality, uses resources more efficiently, and is better integrated into the Win32 API. ISAPI works by loading in-process DLLs that are direct extensions to the Web server. ISAPI DLLs are more resource friendly, remain resident once loaded, and operate in the Web server's address space, all of which translates into very fast operation.

Several connector applications for Visual FoxPro have been implemented by third-party tool providers and by Microsoft for their Web page wizard. Internally, these connectors are signaled that the Web server has a request waiting to be processed and send a message to the Visual FoxPro server to let Visual FoxPro know that it needs to get ready to process this request. A variety of methods can be used to pass this message to the application:

- File-based messages that are polled for by the VFP server (which is what my Web Connection tool and the VFP WWWPage Wizard use).
- OLE messages that are sent to an OLE Automation server (this is the preferred method used to connect VB to a Web server).
- DDE message sent to Visual FoxPro acting as a DDE server.
- Memory-based messages that are placed in reserved areas of memory and picked up by Visual FoxPro via a FLL/DLL call.

As you might expect, each approach has its pros and cons, and you need to find the one that best matches your needs.

Once Visual FoxPro has control, your user code or, in case of the non-programmatic servers, the server's internal processing, sets out to retrieve the Web server information and return an HTML page to the Web server. The Web server receives the finished HTML page and passes it back to the Web browser, which displays the output on the user's screen.

What's a Visual FoxPro data server?

I used the term, Visual FoxPro data server, to make it clear that Visual FoxPro is running in server mode -- continuously loaded in memory and waiting for incoming requests from the Web server. Contrast this with typical CGI applications that are run as executables whenever the Web servers needs them. By running as a server, VFP doesn't need to start up each time, which would be unacceptable due to VFP's runtime size and load time. The term is also meant to describe the server's main purpose: To serve data or, more to the point, HTML page output that is made up of data. What actually transpires as part of the data server processing is up to you and depends on the connector product you work with. The term *server* often conjures up an image of a fixed static process that the developer has little control over. It doesn't need to be this way, however.

For example, Web Connection allows you to run *any* Visual FoxPro code you see fit by letting you set up a user-defined procedure or class method that handles an incoming request. The Web Connection Server will call this code with a single parameter containing a CGI object that provides all the information that the Web server makes available for use in your program. Once your code gets control, you can use FoxPro code to its fullest extent. Web Connection also provides some additional classes to help you create the dynamic HTML output that your code should respond with to complete a request, as well as scripting methods that allow FoxPro expressions and even entire code blocks to be embedded directly into HTML pages without having to write code inside of Visual FoxPro.

Let's look at another example. The Visual FoxPro Web Page Wizard interprets a script file, the filename of which is passed to it as part of the URL by an HTML form that contains the input form for the query. The Wizard's data server then processes the query and sends back the output -- it's all automated for you, but you have no control over the processing other than what the Wizard set up.

Other FoxPro connector tools also provide mixtures between Visual FoxPro programming interfaces and script-like capabilities that can be built into HTML pages to automate output page creation. Whichever tool you choose, you'll be able to leverage your existing FoxPro skills to create dynamic Web-based content.

Speed, load, and scalability

How well suited is Visual FoxPro to being a database server for a Web server? There are several issues to consider. As we all know, Visual FoxPro is very fast as a data engine, which is one immediate advantage. Compared to the local ODBC engines, Visual FoxPro will finish in a fraction of the time.

Speed, however, isn't the only criterion when dealing with a Web server interface. Visual FoxPro isn't a multi-threaded application, so a single Visual FoxPro application can process only one Web request at a time. This means that one request must finish before the next can be executed. For some applications that might not be a problem -- if requests take a second or less and the number of requests is less than the total time the server spends on processing them, users will probably receive a timely response. In my example that cited 20,000 CGI hits, I was referring to these kinds of sub-second response times.

But things get tricky when you have long requests that take 10, 20, or more seconds to run. A couple of those can pile up and cause users on the Web server to time out their connections, which would definitely be a problem. To process requests simultaneously you need multiple Visual FoxPro sessions. The other sessions can run either on the same machine or (for some products anyway) on another machine over a network.

While running multiple sessions on the same (single processor) machine will allow requests to run simultaneously, both of them will slow down drastically while processing, so this may not be a good idea. Often the slowdown is so bad that a single session provides better results than multiple sessions. Still, two or more sessions on a single CPU can work if query times vary widely between short and long queries, so that a short query can be squeezed through while a long one runs.

Sounds bad, right? However, local ODBC data sources suffer the same fate even though ODBC is multi-threaded. The problem is processor load -- one CPU can handle only one task at a time. Simultaneous queries will only split the CPU delegation but won't perform any better. Thus, for heavy duty query situations, a SQL server on multi-processor hardware is probably going to be a better data server.

There's a way around this load problem, however, by adding machines to process data across the network. Web Connection allows you to run multiple sessions of Visual FoxPro over the network and access the request messages from the Web server over the network. Requests are generated as message files on the machine the Web server is running on, but requests can be picked up by different machines running across the network and accessing the Web server's drives via the network. This is a double win situation: Moving the database load completely off the Web server gives the Web server both the breathing room it needs to serve Web pages and the scalability to allow multiple sessions to pick up and process requests simultaneously.

Using this approach makes a Visual FoxPro solution very scalable.

What you need to get started
Now that you have a general idea of how you can use FoxPro for Web service, here's an overview of the pieces you need to start developing Web applications: a Web server, the required connector application, a FoxPro front end that understands the connector's interface, and a Web browser to test your creations (see **Figure 1**).

You'll also want to set up network support for the TCP/IP network protocol. If you're running Windows 95 or NT, this is a no-brainer since the TCP/IP protocol is built in and can be installed with just a few clicks from your Network Control Panel applet. A network isn't actually required for development, though. You can develop TCP/IP applications locally by accessing a Web server running on your development machine via the local machine name or IP address (the local machine can always be accessed as *Localhost* for a domain name or by the machine's local, reserved IP address, which can always be accessed with 127.0.0.1. For example, http://localhost/default.htm will access a local Web server's homepage, as will http://127.0.0.1/default.htm). While a true network isn't required for testing, it's nice to have your test Web server set up on another machine across the network. Web servers are fairly resource intensive beasts that will slow your development machine down significantly while testing applications.

I've mentioned the Internet Information Server as Microsoft's entry into the Web server world. IIS is a good choice since it's extremely fast, tightly integrated with Windows NT, and can be downloaded for free. However, a variety of Web servers are available from other companies and many of them offer many more features for additional configuration options, internal support for scripting, security, and so forth. I've been using Commerce Builder from the Internet Factory. I started using it prior to the release of IIS and continue to use it because it includes support to run under Windows 95 (for testing). It also offers a fairly powerful macro language that makes it possible to create simple dynamic pages that contain logic flow and HTML parameter passing without having to call an external script. Built-in macro features like counters, browser type identification, HTML form variable expansion, and conditional HTML output make it possible to do most non-data related tasks right inside of HTML pages. Commerce Builder also supports many of the advanced technologies that IIS supports such as ISAPI and ISAPI Web server filter extensions. Other Web servers also support macro languages as well as ODBC support via HTML macro expansion. My point is that you should look closely at other servers as well as IIS before blindly "buying" into the Microsoft line. Microsoft's strategy looks good for the future, but right now they're still behind in features that might make your development of Web applications easier. Most servers can be checked out free of charge on a trial basis and prices are coming down drastically now that Microsoft has made IIS available for free.

Today's Web browsers are more like a dBASE II application than the slick GUI applications we've been building in recent years. This will change as script languages and applet or OLE control plug-ins provide interactive browser support. You'll want to pick up copies of both Netscape and Microsoft's Internet Explorer to take full advantage of their competing functionality. Each of these browsers is pushing the HTML standard into new directions that make it easier to create flashy output pages, but unfortunately the two standards are diverging. You might find it useful to take advantage of some of the browsers' advanced features by conditionally supporting these features for specific browser types, (kind of like you can conditionally take advantage of Windows features in a cross-platform FoxPro application with IF _WINDOWS.) This gets messy, but sometimes the usability results are worth the extra effort. We'll see enhancements to the static form interface as the browser scripting languages allow developers to build more GUI-like interfaces. Keep in mind that these new features still need to use the methods I discussed earlier to talk to the Web server; in other words, VBScript or JavaScript won't make a Visual FoxPro back end obsolete since these scripting languages won't be able to access data directly. They'll make HTML page creation easier, though, by building conditional HTML logic right into Web pages.

Finally, there are the Visual FoxPro connector applications. Since I'm not impartial on this topic, I've included a sidebar (**"Tools to Connect Visual FoxPro to the Web"**) that lists some of the tools available and where you can find them. Their respective Web sites provide additional information on their architecture**.**

Getting it online
Once you've decided to create a Web-based application, you have to deal with another important issue: How will the application be deployed? If you're building an Intranet application this will be a fairly easy proposition as your local or wide area network will host the application. Remember, though, that the network must be running the TCP/IP protocol in order to support access to the Web server.

If you're going online to the Internet, the issues become more complex. In a high-level sense, the Internet is a series of networks (the Internet "backbone") to which access is provided through the local phone companies. A company can access the Internet backbone directly, or they can use an Internet Service Provider (ISP), which is hooked up to the backbone. If your application goes to a large corporate client, it's quite possible that their Web server is linked directly to the Internet backbone. If possible, this is the ideal situation, as the Web server and your FoxPro application are sitting at the client site where it's easy to control and debug. But this is an expensive proposition and probably not an option for a small to medium-sized business unless volume (sales or referrals) justifies the expense.

The other option is to have an ISP host the application for you. Usually this means you get to install a machine on the ISP's network and hook into their existing TCP/IP net to host your application and possibly your own Web server. Not all ISPs allow such an arrangement, so you might have to dig deep to find one that will allow co-location at a reasonable price. Preferably you want to find an ISP that runs a Windows-based system so you can hook into their existing Web servers rather than running your own. Your machine can then act as a data server only across the network, while the ISP handles the Web server administration. This is the setup that I currently have with my ISP -- they're running about 20 sites (five of which are data sites powered by my Visual FoxPro stuff.) The ISP handles Web server setup and management for all sites; all I have to worry about is my data server machines and the Visual FoxPro code that runs on them.

Be aware, though, that co-location requires a good relationship with your ISP since you'll need both physical and remote access to your machine to update code, recover the occasional crashed session, and so on. You'll also need full network access to your machine, so you can update files remotely, a security risk that ISPs often worry about. Finding an ISP for co-location isn't easy. A good selling point on your part might be to let the ISP know that helping you set up shop brings them business as well -- your clients will need business accounts and a domain name on the ISP's system in order to go online and both of these will generate consistent income for ISPs.

Administration
Using Visual FoxPro as a Web data back end requires some care. Since VFP is running in server mode, any code crashes or lockups will essentially stop the server from responding to further requests. It's crucial to make your server application bullet proof. Web Connection deals with this with a set of error handlers that catch most errors, log them, and recycle the server when not running in DebugMode. But even with this scheme it's possible to crash the code with successive errors, invalid file paths (a file Open box can hang the server for example), or an infinite loop. There's also the issue of updating code. The data server needs to be shut down in order for the application code to be updated.

When that time comes, you'll need to make sure you can access the machine remotely via a remote access tool to pick up the pieces from the comfort of your office. I've been using LapLink95, which is an extremely well integrated remote communications package and a joy to use. LapLink works very well for controlling my main Windows 95 box that hosts the Visual FoxPro data server, but unfortunately it doesn't work under Windows NT. For NT I'm using pcAnyWhere32, which works but isn't nearly as well done as LapLink and lacks some handy features like the ability to start up before the Windows logon so you can actually shut down and restart the machine remotely. One of these packages is a must for remote server administration. We'll cover these issues in more detail in future articles.

Where to next?
In coming months, I'll fill in many of the details that I glossed over to fit this article space. I hope that this overview has given you a better idea of how the Web fits into our FoxPro-based future. But keep in mind that things are changing fast and that other technologies aside from FoxPro are going to jockey for position. VBScript and HTML-based OLE control integration will likely bring Web development out into the open. It'd be foolish to overlook these tools just because they're not FoxPro-based, especially since they can co-exist very nicely.

Sidebar: Tools to Connect Visual FoxPro to the Web
In addition to the author's Web Connection, which is described in the main article, and Microsoft's Wizard, several other tools are available to hook Visual FoxPro to the Web. You can find information about these products on their respective Web sites.

FoxWeb
Aegis Group
1685 Leroy Ave
Berkeley, CA 94709
Contact: Pandelis Tiritas
Fax 510-523-6794

HotLink
Laguna Groupware Inc.
471 1/2 Blumont St.
Laguna Beach, CA 92651
714-497-0947
http://www.laguna1.com/

Microsoft's Internet Wizard
http://www.microsoft.com/vfoxpro/default.htm

WebLink for Visual FoxPro
Computer Intelligence Inc.
7200 Falls of Neuse Road, Suite 202
Raleigh, NC 27615
Contact: Roger Ko
919-676-8855, ext. 158
Fax 919-676-8484

West Wind Web Connection
West Wind Technologies
400 Morton Road
Hood River, OR 97031
541-386-2087
http://www.west-wind.com/wwcgi.htm

Sidebar: Glossary of Internet Terms
Web browser
A software-based tool used to display pages and dynamic output on the Web. Think of the Web browser as the front end when you're checking out pages on the Web. All user interaction happens here. Browser standards are evolving fast and

compatibility between competitors isn't complete. You should be using the Microsoft Internet Explorer or Netscape when developing applications since these are going to be the dominant browsers and provide the most complete functionality.

HTML
The Hypertext Markup Language is the language of the Web browser used to display output. It's a tag-based language that embeds special characters to specify output formatting.

Web server
A piece of software that's responsible for serving Web content. Primarily this means serving HTML pages, but Web servers are expanding their role as the generic interface that connects to back-end applications. Web servers can serve content via the Internet or using a local, secured network. It's also possible to set up and access a Web server locally for development purposes.

CGI
The Common Gateway Interface is the standard protocol used for scripting Web server extension functionality. This protocol defines standard variable names and a scheme for passing input form data from the Web server to back-end applications. Keep in mind that CGI is a protocol, not an implementation.

ISAPI
The Internet Server Application Programming Interface is a newer, more efficient variation of standard CGI that can be used to build Web server extensions and scripts. Microsoft and several other vendors have embraced this standard, which is DLL based and uses many CGI conventions while providing a more Windows-like API for Web developers.

URL
Uniform Resource Locators are what makes the Web hyperlinked. URLs allow you to jump from one spot in an HTML document to another. URLs can also be used as action directives to start and run CGI/ISAPI scripts or display or play multimedia output.

ISP
Internet Service Providers provide access to the Internet for the masses as well as businesses. Business accounts usually provide additional features such as server space for a set of home pages (or a Web site) as well as customized mail boxes and domain names if required.

TCP/IP
TCP/IP is the protocol on the Internet to connect computers together. In order to access the Web you need to have TCP/IP installed and running on your computer. Windows 95 and NT have this protocol built-in, while Windows 3.1 requires extra software that can be downloaded from Microsoft or other third parties.