Link Your App to a Web Server

by Chris Barlow

With VB5, you can not only Web-enable your app, but also make it Web-aware.

Click & Retrieve Source CODE!

A common feature in Web-enabled software applications is the ability to pop up a browser and display a Web page. This is an easy feature to add to your software—would you believe it takes only about four lines of code? In this column, I'll show you not only how to Web-enable your app, but also how to make it Web-aware, so it has direct access to text returned by the Web server (see Figure 1).

You can probably think of many apps that could utilize this direct Web access. How about an app that cycles through a database of UPS package tracking numbers, finds out from www.ups.com whether the package had been delivered, and updates the database with the delivery information? How about an app that downloads future ocean tide information from www.tides.com/cgi-bin/tcweb.exe and puts it into a local database so you can review it when you're on your sailboat? Or how about an app that checks the Date Modified property of all the Web pages in your Favorites folder and notifies you of any change?

First, I'll show you how to Web-enable your software app. To create an app that links to a browser and displays a Web page, you need to set up your computer to work with Web apps. I suggest using Windows 95, Visual Basic 5, Internet Explorer 3.02, Visual InterDev, and Personal Web Server with the Active Server Page (ASP) extensions so you can test your Web apps on your own system. If you don't use Visual InterDev, use FrontPage.

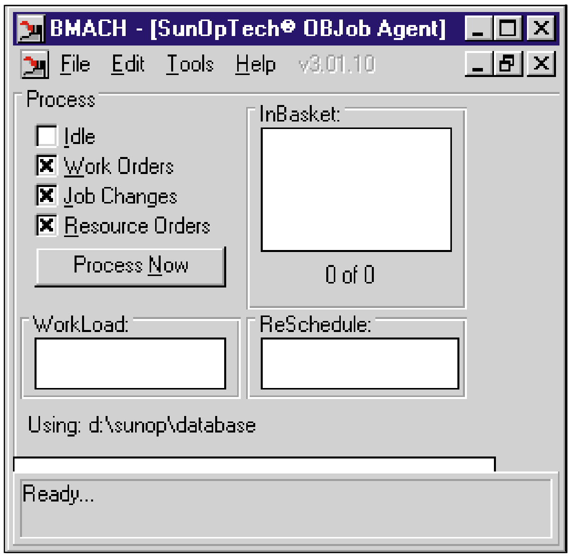


FIGURE 1 ***Make Your App Web-Aware.*** *At one of my company's client's manufacturing plants, each production department running SunOpTech scheduling software "publishes" its production schedule on its Web server. When an app running in one department needs to check when its components will be ready in another department, the app contacts the Web server in the second department and requests the current schedule for that component. Although this interapplication communication between departments uses standard Web protocols, the exchange of information isn't visible to the user and doesn't require user interaction. The application has direct access to the text returned by the Web server.*

Now you're ready to add Web access to an existing VB project or to a new project. Right-click on the project and display the References. Make sure Microsoft Internet Controls is checked; if not, use the Browse button to locate the SHDOCVW DLL in your Windows System folder (see Figure 2). Internet Explorer also uses this DLL. Press the F2 key to display the Object Browser and look at the properties and methods of the Internet Explorer class within this DLL. The Navigate method takes a URL as an argument, pops up Internet Explorer, and jumps to the URL.

Add a form to your project, and place a TextBox control and a CommandButton control. Double-click on the CommandButton control and add this code, which calls another subroutine and passes the URL typed into the TextBox control:

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support and supply chain applications. He's also a frequent speaker at VBITS, Tech•Ed, and DevDays, and has been featured in two Microsoft videos. Reach Chris at ChrisB@SunOpTech.com or through SunOpTech's Web server at www.SunOpTech.com.

```
Private Sub Command1_Click()
GoExplorer txtURL
End Sub
```

Dimension a variable, cWeb, to hold an instance of the InternetExplorer class. Create the GoExplorer subroutine with these three lines of code to instantiate the class. Now call the subroutine's Navigate method with the URL, and make the Internet Explorer window visible:

```
Dim cWeb As SHDocVw.InternetExplorer

Private Sub GoExplorer(sURL$)
Set cWeb = New SHDocVw.InternetExplorer
cWeb.Navigate sURL
cWeb.Visible = True
End Sub
```

Now your application can do the same thing done by many software packages advertised as "Web-enabled" apps.

MAKE YOUR APPLICATION WEB-AWARE

As neat as it is to be able to pop up a browser already navigated to a specific URL, it doesn't make your app a "Web" app. You still rely on the user to point and click to navigate beyond that URL, and the text on the Web pages is not readily available to your app.

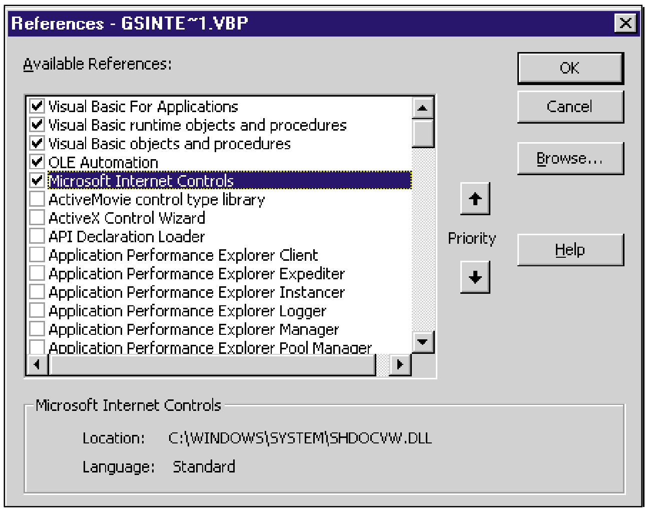
At my company, SunOpTech, we develop apps that go beyond "Web-enabled" to become "Web-aware." These apps rely on Web servers to provide much of the data they need to operate. For example, in one of our client's manufacturing plants, each production department running SunOpTech scheduling software "publishes" its production schedule on its Web server (see Figure 1). When an app running in one department needs to check when its components will be ready in another department, the app contacts the Web server in the second department and requests the current schedule for that component. Although this interapp communication between departments uses standard Web protocols, the exchange of information isn't visible to the user and doesn't require user interaction. The app has direct access to the text returned by the Web server. The SHDOCVW method doesn't provide this functionality. Fortunately, you can accomplish this nearly as easily with the Win32 Internet functions built into the WinInet DLL.

The WinInet DLL gives complete Internet functionality to any VB app. When you use this DLL, you don't need to use the SHDOCVW DLL or any other Internet controls on the market. To keep it simple, use only four of the functions from this DLL. For more detailed information, look at the Microsoft Win32 Internet Programmer's Reference at www.microsoft.com/intdev/sdk/docs/wininet.

Add a module to your project and insert the function declarations for the WinInet DLL (see Listing 1). You'll use these four functions:

- **InternetOpen** to establish an Internet session for your application.
- **InternetOpenURL** to connect to a Web server and make sure the file exists on the Web server.
- **InternetReadFile** to read the HTML text from the file on the Web server.
- **InternetCloseHandle** to close the file after you read it.

Let's step through the function you'll write to retrieve data from a Web server. Create a function called GoWinInet that receives a string argument of the URL and returns the text from that URL as a string. Dimension a 4K buffer, sBuffer, to hold the text from each read of the file, and a string, sReturn, to hold the entire text from the file. Dimension a long variable to hold the number of bytes read from the file, and a Boolean, bReadOK, to hold the return value from reading the file. Finally, dimension two long variables, lSession and lFile, for the session and file handles returned by the function calls:



Available References:

- ☑ Visual Basic For Applications
- ☑ Visual Basic runtime objects and procedures
- ☑ Visual Basic objects and procedures
- ☑ OLE Automation
- ☑ Microsoft Internet Controls
- ☐ ActiveMovie control type library
- ☐ ActiveX Control Wizard
- ☐ API Declaration Loader
- ☐ Application Performance Explorer Client
- ☐ Application Performance Explorer Expediter
- ☐ Application Performance Explorer Instancer
- ☐ Application Performance Explorer Logger
- ☐ Application Performance Explorer Manager
- ☐ Application Performance Explorer Pool Manager

Microsoft Internet Controls

Location: C:\WINDOWS\SYSTEM\SHDOCVW.DLL

Language: Standard

FIGURE 2 ***Project References.*** *This dialog lets you set the references that the Visual Basic editor will recognize as you type code. Adding a reference here lets you easily view the properties and methods in the Object Browser. A reference also lets VB provide proper syntax checking as you enter code using this object's properties and methods.*

VB4 32-bit VB5

```
Option Explicit

Public Const INTERNET_OPEN_TYPE_PRECONFIG = 0
' indicates to use config info from registry
Public Const INTERNET_FLAG_EXISITING_CONNECT = _
  &H20000000

Public Declare Function InternetOpenUrl Lib _
  "wininet.dll" Alias "InternetOpenUrlA" _
  (ByVal hInternetSession As Long, _
  ByVal lpszUrl As String, _
  ByVal lpszHeaders As String, _
  ByVal dwHeadersLength As Long, _
  ByVal dwFlags As Long, ByVal dwContext As Long) _
  As Long

Public Declare Function InternetOpen Lib "wininet.dll" _
  Alias "InternetOpenA" (ByVal sAgent As String, _
  ByVal lAccessType As Long, _
  ByVal sProxyName As String, _
  ByVal sProxyBypass As String, ByVal lFlags As Long) _
  As Long

Public Declare Function InternetReadFile Lib _
  "wininet.dll" (ByVal hFile As Long, _
  ByVal sBuffer As String, _
  ByVal lNumBytesToRead As Long, _
  lNumberOfBytesRead As Long) As Integer

Public Declare Function InternetCloseHandle Lib _
  "wininet.dll" (ByVal hInet As Long) As Integer
```

LISTING 1 ***Win32 Internet Module.*** *This module contains the function declarations from the WinInet DLL used by the GSInternet application.*

```
Private Function GoWinInet(sURL$) _
   As String
Dim sBuffer As String * 4096
Dim sReturn As String
Dim lNumBytes As Long
Dim bReadOK As Boolean
Dim lSession As Long
Dim lFile As Long
```

Call the InternetOpen function to return a handle to an Internet session. Pass this handle to the next function call to connect to a URL:

```
lSession = InternetOpen("GSInternet", _
   INTERNET_OPEN_TYPE_PRECONFIG, _
   vbNullString, vbNullString, 0)
```

Call the InternetOpenUrl function to pass the URL and the handle to the session and to save the returned handle that points to the file on the Web server:

```
lFile = InternetOpenUrl(lSession, _
   sURL, vbNullString, 0, _
   INTERNET_FLAG_EXISITING_CONNECT, 0)
```

If a nonzero file handle is returned, begin a Do loop to read text from the file into the buffer using the InternetReadFile function. This function fills in the fourth argument with the actual number of bytes read from the file, and returns a nonzero value if the function was successful. If any bytes are returned, concatenate them to the sReturn string from the buffer:

```
If lFile Then
   Do
```

VB4 32-bit VB5

```
Option Explicit
Dim cWeb As SHDocVw.InternetExplorer
Private Sub Command1_Click()
GoExplorer txtURL
End Sub
Private Sub GoExplorer(sURL$)
Set cWeb = New SHDocVw.InternetExplorer
cWeb.Navigate sURL
cWeb.Visible = True
End Sub
Private Sub Command2_Click()
MsgBox GoWinInet(txtURL)
End Sub
Private Function GoWinInet(sURL$) _
  As String
Dim sBuffer As String * 4096
Dim sReturn As String
Dim lNumBytes As Long
Dim lSession As Long
Dim lFile As Long
Dim bReadOK As Boolean
lSession = _
  InternetOpen("GSInternet", _
  INTERNET_OPEN_TYPE_PRECONFIG, _
  vbNullString, vbNullString, 0)
lFile = InternetOpenUrl(lSession, _
  sURL, vbNullString, 0, _
  INTERNET_FLAG_EXISITING_CONNECT, 0)
If lFile Then
  Do
     bReadOK = _
        InternetReadFile(lFile, _
        sBuffer, Len(sBuffer), _
        lNumBytes)
     If lNumBytes Then
        sReturn = sReturn & _
           Left$(sBuffer, lNumBytes)
     End If
  Loop While bReadOK And lNumBytes > 0
  InternetCloseHandle (lFile)
  GoWinInet = sReturn
Else
  MsgBox "Cannot open URL"
End If
End Function
```

LISTING 2 ***GSInternet Form.*** *This module contains the code for both the SHDOCVW and WinInet functions that let you access the Web from any Visual Basic application.*

```
        bReadOK = InternetReadFile_
          (lFile, sBuffer, _
          Len(sBuffer), lNumBytes)
        If lNumBytes Then
          sReturn = sReturn & _
            Left$(sBuffer, lNumBytes)
        End If
```

If the prior read is successful and bytes were read, loop back and execute another read. Continue until the file is completely read. Call the InternetCloseHandle function to close the file and return the complete text to the function:

```
  Loop While bReadOK And lNumBytes > 0
  InternetCloseHandle (lFile)
  GoWinInet = sReturn
Else
  MsgBox "Cannot open URL"
End If
End Function
```

That's all you need to do to get the raw text from any URL onto a Web server. Test this within your app by adding a second CommandButton to your form and this code. This code passes the URL to your new GoWinInet function and displays the returned text in a message box:

```
Private Sub Command2_Click()
MsgBox GoWinInet(txtURL)
End Sub
```

Give this a try and you should see the actual HTML text from the URL displayed in the message box. Now that you have the text, you can parse it to find the particular values your app needs. When SunOpTech writes similar routines for interapplication communication, we signal the Web server to return minimal HTML text so little parsing is needed to get the required data. We do this by adding an extra argument to the query arguments embedded in the URL to indicate an app rather than a user will read the resulting HTML file.

I've included the code that lets your app access the Web, using both the SHDOCVW version and the WinInet version (see Listing 2). Download the VB files from the Premier Level of The Development Exchange Web site (see the Code Online box for details). An obvious next step would be to encapsulate these WinInet functions in your own ActiveX control. Let me know if you're interested in learning how, and I'll consider it for a future column.

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Link Your App to a Web Server

Locator+ Codes

Listings ZIP file (free Registered Level): VBPJ1097

✪ *Listings for this article plus the VB files that let your app access the Web, including both the SHDOCVW version and the WinInet version (subscriber Premier Level): GS1097P*