Scan for News

Click & Retrieve Source CODE!

Microsoft's new Internet ActiveX Controls make Internet newsgroup access simple.

by Chris Barlow

If you are getting started with Visual Basic, you are probably finding yourself overwhelmed with all the bits and pieces of information. Although Microsoft has done an excellent job with the help files and sample applications that ship with Visual Basic, to really become proficient in VB you have to keep gathering up-to-date information from a variety of sources. For me, the difficult part is finding the time to study all the information I have gathered. I look for information sources that I can study in my "offline" time away from my desk—airplanes, airports, and the proverbial home "reading room."

While reading *VBPJ* is certainly one of the best tools, I've always enjoyed the give-and-take of information exchange over electronic forums such as the *VBPJ* and MSBASIC forums on CompuServe. It's gratifying to scan the message subjects to see if anyone else is struggling with the same problem as I am and to read suggested solutions from other Visual Basic programmers all over the world. I usually scan the messages quickly, search on some common keywords, and download lots of messages to read and study the next time I'm offline. With the AutoPilot capabilities of WinCIM or some of the other "navigator" applications, I'm able to scan several forums and download interesting messages to read later. If you haven't used these online resources, try them the next time you have a question. I think you'll be amazed how quickly you can get a solution!

In May 1996 Microsoft added another important online resource for Visual Basic programmers—its own Internet news server. This news server, located on the Internet at msnews.microsoft.com, has newsgroups for every category of Microsoft product. Microsoft is always adding newsgroups, but I've listed the Visual Basic-related newsgroups as of mid-May (see Table 1). Microsoft announced that it was going to stop responding to product support requests on CompuServe. Instead, it will provide all support through these newsgroups.

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications including the ObjectBank and the ObjectJob Systems, where he and Ken Henderson hold a software patent related to decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

While Internet newsgroup servers provide much the same functionality as CompuServe forums, you will need to use an application program that supports the Network News Transfer Protocol (NNTP) to access these newsgroups over the Internet. You can view articles (messages) by subject, sender, and date within a newsgroup. You also can read articles, as well as post your own articles with questions and comments. Microsoft even posted a free Internet News Reader that you can download from its Web site. The cool thing about this news reader is that it functions as part of Windows Explorer (see Figure 1).

I like the content of these Microsoft newsgroups, but I really miss being able to download the interesting articles and read them offline. Fortunately, as a Visual Basic programmer, you can write your own application to scan certain newsgroups and save the articles to read later! In this column, we'll develop the Newscan application using Microsoft's new NNTP ActiveX control that is part of the Internet ActiveX Controls.

MICROSOFT INTERNET ACTIVEX CONTROLS

Microsoft's Internet ActiveX Controls include a wide selection of ActiveX controls, from the basic WinSock control to sophisticated HTML controls. Like the other controls, the NNTP control encapsulates the Internet NNTP protocol into an easy-to-use control. By using the methods and properties of this control, you can select newsgroups and read articles. You can use this control to create an application that will scan the article subjects for certain keywords and download the articles.

While the NNTP control will make this easy, you need to become somewhat familiar with newsgroups and articles. The articles posted within each newsgroup are made up of a header, containing several items including Subject, Date, and From, and the body of the article. Microsoft assigned each

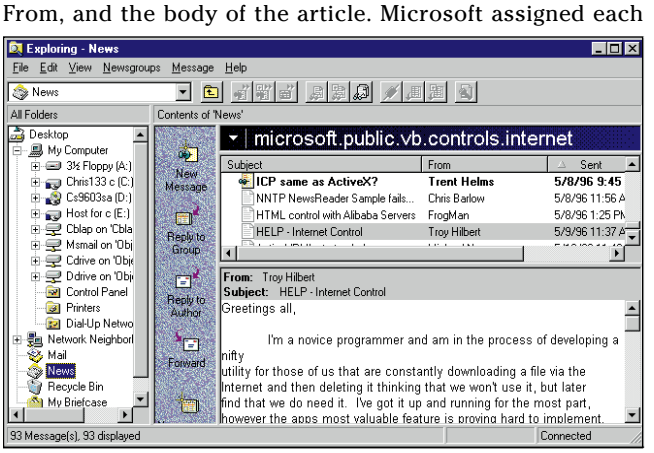

FIGURE 1 ***Microsoft News, at Your Fingertips.*** *Notice how easily the news reader blends right into Windows Explorer and negotiates menus and toolbars. OLE is neat!*

article a number, and when you select a newsgroup the news server responds with the first and last article number in the newsgroup. You can retrieve all the article headers in a range of article numbers and then retrieve an individual article by its number.

Remember, you are communicating with a remote news server over the Internet. After you call one of the control's methods to send a request to the server, the server will respond in a variable amount of time. When the server's response arrives at your computer, the control will trigger an event for your application to respond to. You need to plan your application carefully so you don't get out of sync with the server—for example, you don't want to request another article while the server is still sending you the prior article.

The NNTP control makes this interaction a bit difficult, because all data returned from the server triggers the same event—DocOutput. This event is triggered repeatedly as the server sends blocks of headers and data. The DocOutput object's State property indicates the type of data you are receiving. As you write code in this event, you need to know what data your program requested so you can put the right data in the right place.

OK, time to get started. I'll show you how to write an application called Newscan that scans the subjects of articles in selected newsgroups for a match with a list of keywords, then downloads the matching articles and saves them in a Microsoft Access database (see Figure 2).

The Access database, newscan.mdb, contains three tables (see Figure 3). I've used an Access 2.0 database in the sample code for those of you who don't have the latest version. The Groups table contains the name of the newsgroups with a Boolean field to indicate whether this group should be searched and a LastMessage field to identify the articles that have already been searched. The SearchFor table contains multiple records, each indicating a word or phrase to search for in the article subjects. The Articles table contains the downloaded articles. The sample code includes a copy of this database with

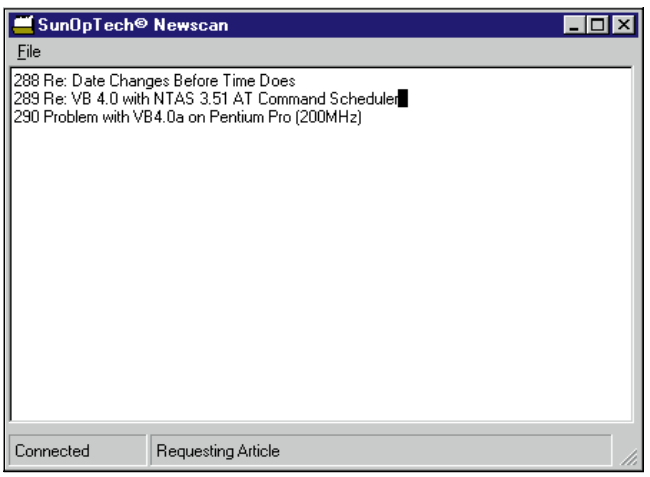

FIGURE 2 ***The Newscan Application in Action.*** *This snapshot was taken as the program was in the requesting-articles state. The Newscan application found a match on a search word in article number 289.*

Visual Basic-Related Newsgroups

WEB SITE	DESCRIPTION
microsoft.public.vb.3rdparty	Third-party products
microsoft.public.vb.addins	VB add-ins / Visual SourceSafe
microsoft.public.vb.bugs	Bug reports
microsoft.public.vb.controls	Control usage and programming
microsoft.public.vb.controls.databound	Data-aware controls
microsoft.public.vb.controls.Internet	Internet-aware controls
microsoft.public.vb.crystal	Crystal report writer
microsoft.public.vb.database	Database issues
microsoft.public.vb.database.dao	DAO programming
microsoft.public.vb.database.odbc	ODBC issues
microsoft.public.vb.database.rdo	RDO programming
microsoft.public.vb.dos	VBDOS issues
microsoft.public.vb.enterprise	VB Enterprise Edition issues
microsoft.public.vb.installation	Installation/setup issues
microsoft.public.vb.ole	General OLE issues
microsoft.public.vb.ole.automation	OLE Automation programming
microsoft.public.vb.ole.cdk	OLE control development
microsoft.public.vb.ole.servers	Creating OLE servers
microsoft.public.vb.setupwiz	Setup wizard/kit
microsoft.public.vb.syntax	General language issues
microsoft.public.vb.winapi	Windows API usage
microsoft.public.vb.winapi.graphics	Windows graphics APIs
microsoft.public.vb.winapi.networks	Windows network APIs, Internet

TABLE 1 ***Visual Basic Newsgroups, On Tap.*** *You can find valuable information from many Visual Basic-related newsgroups. Be sure to check the news server for additional groups.*

VB4

```
Option Explicit

Public ArticleID As String
Public Group As String
Public Subject As String
Public From As String
Public Dated As String
Public Text As String
Public Read As Integer
Public DateAdded As Long
Public ArticleNumber As Long

Public Sub Clear()
ArticleID = ""
Group = ""
Subject = ""
From = ""
Dated = ""
Text = ""
Read = 0
DateAdded = 0
ArticleNumber = 0
End Sub

Public Sub Save(rs As Recordset)
rs.AddNew
rs("ArticleID") = Left(ArticleID, 50)
rs("Group") = Left(Group, 50)
rs("Subject") = Left(Subject, 50)
rs("From") = Left(From, 50)
rs("Dated") = Left(Dated, 50)
rs("Text") = Text
rs("Read") = False
rs("DateAdded") = Now
rs("ArticleNumber") = Left(ArticleNumber, 50)
rs.Update
End Sub
```

LISTING 1 ***The Article Class.*** *This simple class makes your code a bit easier to follow. Take a look at the Save method to see how to save articles to the database.*

the tables preloaded with some newsgroups, search words, and articles.

To use the sample code as it was written, you should be running Visual Basic 4.0 Professional in Windows 95 and you should download Microsoft's Internet ActiveX Controls from www.microsoft.com/intdev.

To accommodate the asynchronous nature of accessing data over the Internet, we'll write the program in the form of a "state machine." Don't get nervous, this is not as complicated as it sounds. Basically, you'll be using a Select Case statement in the Timer control's Timer event to determine what action your program should take based on its current "state." For more detail on state machines, see Daniel Appleman's excellent article, "Design True Event-Driven Code," in the August/September 1994 issue of *VBPJ*.

The Newscan application will move through several different states and perform different functions based on the current state. Before you write a program like this, examine each of the program states and list the functions you want the program to perform:

• Connecting to News Server: the first step is to open the database and connect to the news server at msnews.microsoft.com. If you are using Windows 95 and you are not presently connected to the Internet, this method will dial your Internet provider and establish a connection. When the StateChanged event of the NNTP control fires and indicates that you are connected, you'll change the program's state to *SelectGroup*.

• Selecting a newsgroup: next, you will read a record from the Groups table of the database and call the SelectGroup method of the NNTP control with the newsgroup name. When the SelectGroup event of the NNTP control fires, you'll know you are connected to that newsgroup and you can change the program's state to *RequestingHeaders*.

• Requesting headers: when you call the GetArticleHeaders method of the NNTP control and request the article subjects for the new articles that have not been scanned, you can change the program's state to *GettingHeaders*.

• Getting headers: it might take a while for the news server to send all the article headers. The DocOutput event of the NNTP control will be fired several times as the headers are received. When all headers have been received, the DocOutput object's State property will change to icDocEnd and you can change the program's state to *RequestArticle*.

• Requesting an article: now you're ready to scan each article header to see if any of the keywords from the SearchFor table are in the header. When you find a matching header you can call the GetArticleByArticleNumber method of the NNTP control and you can change the program's state to *GettingArticle*.

• Getting an article: again, it may take a while for the article to arrive over the Internet. As it arrives the DocOutput event of the NNTP control will be fired and your program can capture the header fields and the body of the article and save it in the Articles table of the database. When all of the article has arrived you can change the program's state back to *RequestArticle,* so that it retrieves the next article. When there are no more articles, you can change the program's state back to *SelectGroup* to repeat this process for the next newsgroup.

• Done: when there are no more newsgroups to select, you can change the program's state to *Done* to disconnect from the news server and close the database.

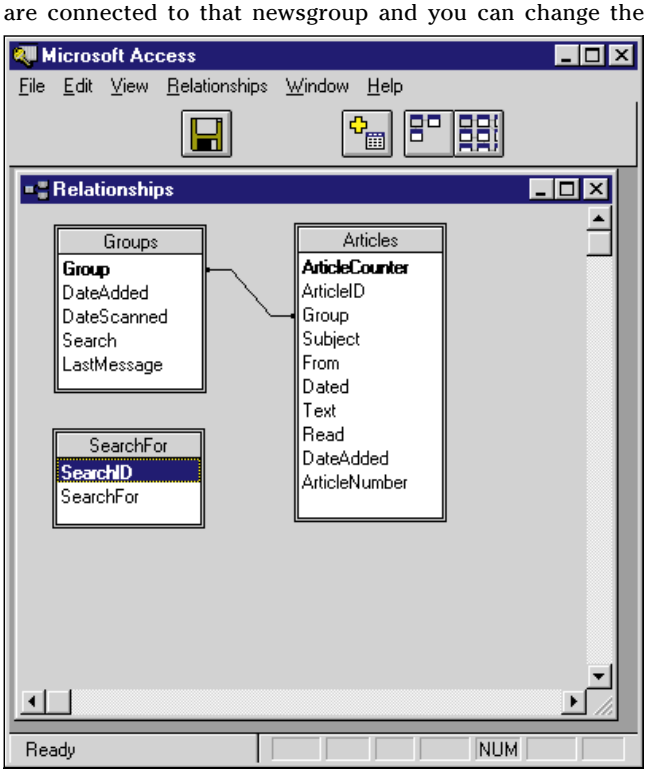

FIGURE 3 ***An Aerial View of the Newscan Database.*** *This is the Relationship view in Access. It provides a quick view of the tables, fields, and relationships.*

WRITING THE CODE

Now that you have a clear idea of the program logic, it is time to write the code. Start a new Visual Basic project and place a RichTextBox and StatusBar on the form. Add references to the Microsoft NNTP Client Control and the Microsoft Internet Support Objects to your project and place an NNTP control and a Timer control on the form.

The first step is to define a variable, CurrentFunction, which will hold the program state and the constants for the different states:

```
Private CurrentFunction As Integer
Const CFConnecting = 0
Const CFSelectGroup = 1
Const CFRequestHeaders = 2
Const CFGettingHeaders = 3
Const CFRequestArticle = 4
Const CFGettingArticle = 5
Const CFDone = 6
```

You also might want to define an array to hold a text version of these program states to display on the StatusBar control:

```
Private CFString
Private Sub InitVar()
CFString = Array("Connecting", "Selecting Group", _
   "Requesting Headers", "Getting Headers", _
   "Requesting Article", _
   "Getting Article", "Done")
End Sub
```

You'll need to define some other global variables as well. Please refer to the complete code, available from MSN, CompuServe, and the Registered Level of The Development Exchange (for details, see the "Code Online" box at the end of this column). You'll need a procedure to open the database

and create record sets for the three tables. You can use a SQL statement to select only the newsgroups that are supposed to be searched:

```
Private Sub OpenDB()
Set dbNews = OpenDatabase(App.Path & _
   "\newscan.mdb")
Set rsGroups = dbNews.Open_
   Recordset("Select * from Groups _
   where Search = True")
Set rsSearch = _
   dbNews.OpenRecordset("SearchFor")
Set rsArticles = _
   dbNews.OpenRecordset("Articles")
End Sub
```

When the program begins, you can use the Form_Load event to initialize the status array, open the database, and establish the connection to the news server with the NNTP control's Connect method:

```
Private Sub Form_Load()
InitVar
OpenDB
NNTP.Connect "msnews.microsoft.com"
End Sub
```

When the StateChanged event of the NNTP control fires, you can check if the State is connected and set the CurrentFunction to CFSelectGroup:

```
Private Sub NNTP_StateChanged(ByVal _
   State As Integer)
If State = prcConnected Then _
   CurrentFunction = CFSelectGroup
Status.Panels(1).Text = _
   NNTP.StateString
End Sub
```

Now you are ready to start writing the Timer event code. Remember, this is the main control point of your application. Every time the Timer event fires, this code will check for the current program state and execute the proper code. The first few lines of code will display the current program state in the second panel of the StatusBar control and begin the Select Case statement on the program state. If the state is Connecting then the code will fall through and nothing will happen during this cycle:

```
Private Sub Timer1_Timer()
Status.Panels(2).Text = _
   CFString(CurrentFunction)
Select Case CurrentFunction
Case CFConnecting
   'just wait...
```

When the state is SelectGroup you should get the next record from the Groups table, clear the RichTextBox, and call the SelectGroup method of the NNTP control:

```
Case CFSelectGroup
   'find next group
   If rsGroups.EOF Then
      CurrentFunction = CFDone
   Else
      txtNews = ""
      sPos = 0
      NNTP.SelectGroup _
         rsGroups("Group")
   End If
```

When the state is RequestHeaders you should check whether any headers have not yet been scanned and call the GetArticleHeaders method of the NNTP control to retrieve these headers:

```
Case CFRequestHeaders
   'get headers from last one
   'retrieved to last in group
   If (rsGroups("LastMessage") + 1) < _
      LastArticle Then
      NNTP.GetArticleHeaders _
         "Subject", CStr(rsGroups_
         ("LastMessage") + 1), _
         CStr(LastArticle)
      CurrentFunction = _
         CFGettingHeaders
   Else
      CurrentFunction = _
         CFRequestArticle
   End If
```

While you are getting headers, the code in the DocOutput event of the NNTP control is getting fired. So, if this code is the program state, you can just fall through the Select and take no action:

```
Case CFGettingHeaders
   'just wait...
```

If the program state is RequestArticle, then you'll need to get the number of the next matching article from the ScanArticleHeader procedure (for more information on this procedure, download the complete listing from FTP's online services described at the end of this column). Then, call the GetArticleByArticleNumber method of the NNTP control to retrieve that article. If no article is found, update the message number in the Groups table and change the program state to select the next group:

```
Case CFRequestArticle
   'scan selected headers for search
   'criteria and retrieve
   CurrArticle = ScanArticleHeader()
   If CurrArticle Then
      MyArticle.Clear
      MyArticle.ArticleNumber = _
         CurrArticle
      NNTP.GetArticleByArticleNumber _
         CStr(CurrArticle)
      CurrentFunction = _
         CFGettingArticle
   Else
      CurrentFunction = _
         CFSelectGroup
      rsGroups.Edit
      rsGroups("LastMessage") = _
         CStr(LastArticle)
      rsGroups.Update
      rsGroups.MoveNext
   End If
```

The last two program states are easy. You don't need to take any action while you are getting articles, because the DocOutput event of the NNTP control will be fired each time article data is received. Finally, when the program state is Done, you can exit the program:

```
Case CFGettingArticle
   'just wait...
Case CFDone
   MsgBox "Newscan Complete!"
   mExit_Click
End Select
End Sub
```

CAPTURING DATA FROM THE INTERNET

The last major piece of code you'll need to write is in the DocOutput event of the NNTP control to capture the data that is sent from the news server in response to your requests. The DocOutput event

hands your program a DocOutput object that contains data retrieved from the Internet. The DocOutput object has a State property that indicates whether the retrieval transaction is just beginning, has received headers, has received data, has ended, or has generated an error. In a manner similar to the code you wrote in the Timer event to differentiate between the states of your program, you can write a Select statement to distinguish between the states of the DocOutput object. You don't need to take any action when the receive transaction begins. When article headers are received, you need to select by the Name property of the Header object and save the header in the proper field of the article class:

```
Private Sub NNTP_DocOutput(ByVal _
   DocOutput As DocOutput)
Dim Data As String
Dim Hdr As Object
Select Case DocOutput.State
Case icDocBegin
Case icDocHeaders
   For Each Hdr In DocOutput.Headers
   ' For each header in headers
   ' collection
      Select Case LCase(Hdr.Name)
       ' Determine type of header...
      Case "subject"
         MyArticle.Subject = Hdr.Value
      Case "from"
         MyArticle.From = Hdr.Value
      Case "newsgroups"
         MyArticle.Group = Hdr.Value
      Case "date"
         MyArticle.Dated = Hdr.Value
      Case "message-id"
         MyArticle.ArticleID = _
            Hdr.Value
      Case Else
      End Select
   Next
```

When data is received in the DocOutput object, you need to take two different actions depending on your program state variable, CurrentFunction. If you are getting article subject headers, add them to the RichTextBox control. If you are getting an article, save the article text:

```
Case icDocData
   DocOutput.GetData Data
   Select Case CurrentFunction
   Case CFGettingArticle
      MyArticle.Text = Data
   Case CFGettingHeaders
      txtNews.Text = txtNews.Text & _
         Data
   Case Else
      txtNews.Text = txtNews.Text & _
         Data
   End Select
```

Finally, if the receive transaction is ending and you have been getting articles successfully, you need to save the complete article to the database (see Listing 1). Then you need to change your program state to request the next article:

```
Case icDocEnd
   Select Case CurrentFunction
   Case CFGettingHeaders
      CurrentFunction = _
         CFRequestArticle
      Case CFGettingArticle
       MyArticle.Save rsArticles
       rsGroups.Edit
       rsGroups("LastMessage") = _
         MyArticle.ArticleNumber
       rsGroups.Update
       CurrentFunction = _
         CFRequestArticle
   End Select
Case icDocError
Case icDocNone
End Select
End Sub
```

You can see how easy Visual Basic makes it to create your own Internet news reader. Once you've created this Newscan application, you'll never be at a loss for offline reading material. ×

Code Online

All the source code for this issue and extensive extra files, including complete running applications and utilities, are available online. Basic listings for this issue are available as one compressed file on the Registered Level of The Development Exchange (http://www.windx.com). Use Locator Code VBPJ0896 to find the zipped file. Each issue's listings are posted for free access for one month. Extra files and utilities are available to paid subscribers of the Premier Level of the Development Exchange Web site, along with complete archives of all code ever published in VBPJ *and* Microsoft Interactive Developer *magazines. Files from this article that are available only to Premier Members include the CLS, VBP, MDB, FRM, LDB, and FRX files associated with Chris Barlow's Newscan application. Search for Locator Code GS0896P. Core listings are also available at no extra charge from FTP, beyond the service provider charges, on The Microsoft Network (GO WINDX), and the Magazine Library of the* VBPJ *Forum on CompuServe (GO VBPJ).*