Surf's Up! Parsing Web Data

Doug Steele

2000 2002 2003 DOWNLOAD

This month, Doug gets some questions that send him back to an earlier topic: linking your Access application into data on the Internet.

How can I get information from the Internet into my app?

Back in November 2003, I wrote about how to use the XMLHTTP object to get information from a Web site. However, after getting more information from the reader who posed the original question, it turns out that approach isn't really appropriate in his case. Among other issues, my technique assumed that you know the URL that the information will be coming from. The reader who raised the questions is a librarian who wanted to be able to get information about books from the Library of Congress Online Catalog at www.loc.gov/cgi-bin/zgate. However, he needed to be able to interact with the Web page to ensure that the correct book was found. This would have been rather awkward with the technique I showed.

Fortunately, though, there is a solution, as long as you're using Access 2000 or newer. If Internet Explorer is installed on your computer, then you should also have a Microsoft Web Browser control available for you to use on your form. To test this out, create a new form and, while you're in Design mode, select ActiveX Control from the Insert menu.

Now scroll through the list of available controls until you find the entry for Microsoft Web Browser. Select it, click the OK button, and a new control will appear on your form.

As a check that you have the right control, the new control will have a name along the lines of WebBrowser0. In the code that follows, I've renamed the control to ocxWebBrowser.

Using the control is simple: Invoke the Navigate method, passing a URL, and your page appears! In the sample database that accompanies this article, I've put a textbox named txtURL on the form, as well as a button named cmdGo that will navigate to the URL contained in the textbox. The complete code for the Click event of the command button is:

```
If Len(Me.txtURL) > 0 Then
  Me.ocxWebBrowser.Navigate Me.txtURL
End If
```

Assuming that I've got the URL given above in the textbox, clicking on cmdGo takes me to the Z39.50 Gateway to the Library Of Congress Online Catalog.

How do I use the Web Browser?

You use what's in the Web Browser control exactly the same as you would any other Web browser: You fill in textboxes, select controls, and use the Web page's navigation methods.

As I mentioned at the start of this column, the original request was to use the search feature at the Web site above to get information about books into his application. Now, I happen to know that our editor, Peter Vogel, has authored a few books, so I can use them as a test case.

If I change the search type from the default Title to Author, enter Vogel, Peter as the search term, and then click on the Submit Query button, a page returns telling me that a total of 18 books were found. Of course, not all of them are by our Peter, but if I scroll through the list, I find several that he was Editor for, as well as several that he authored. Figure 1 shows the details of two of the books he authored. At this point, you've given your users the ability—without leaving Access—to interact with some Web page that you display to them. Which leads to our reader's next question.

How do I get the information from the Web Browser control into my application?

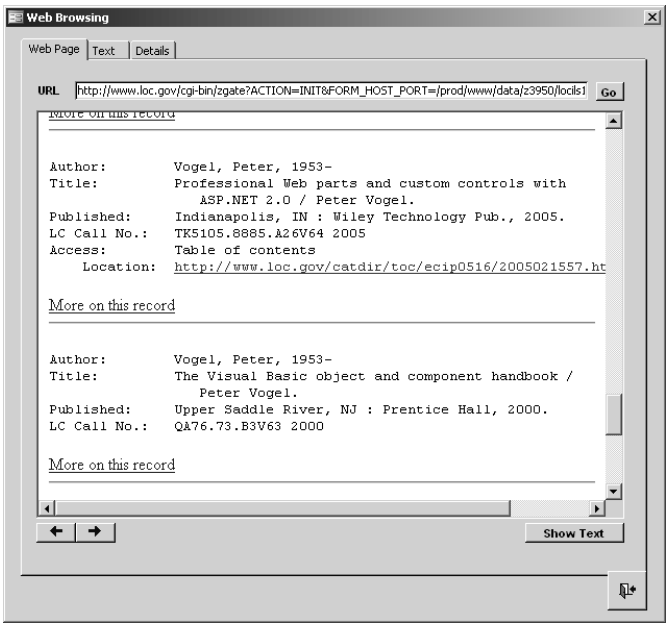

Figure 1. Details of some books authored by Peter Vogel.

Here's where you need to know a little bit about how Web pages are built. In their simplest form, Web pages use HTML, and the browser converts that HTML to what you see. There's a Document Object Model defined that lets you work with the HTML associated with a Web page, but delving into it in any detail is far beyond the scope of this article. (If you want more information, a good place to start is at the World Wide Web Consortium's [W3C's] Web site, specifically at http://www.w3.org/TR/WD-DOM, or in the Web Development section of Microsoft's MSDN, say at http://msdn.microsoft.com/workshop/author/dhtml/dhtml_node_entry.asp.)

In a nutshell, you can access the details of what's being displayed in the Web Browser control by looking at its Document property. To refer to the root node of the document, you access its documentElement property. To be able to get the HTML associated with what's in the Web Browser control, you refer to the innerHTML property of the documentElement, like this:

```
ocxWebBrowser.Document.documentElement.innerHTML
```

However, unless you have a real need to work with the HTML (say, to help you find a specific part of the Web page), you're probably better off working with the innerText property with code like this:

```
ocxWebBrowser.Document.documentElement.innerText
```

Unfortunately, I can't give you any hard-and-fast rules for how to get what you want out of the Web page. I'll work through my example, continuing to use the Library of Congress page, and hopefully that will give you a feel for how to use it in your situation. I'd suggest you actually go to the Library of Congress Online Catalog to see what I'm talking about in action.

While it's possible to have users select the text they want on the page, right-click on it, and copy a selection to the clipboard, I didn't feel that gives developers enough control over what data is being transferred to the application. I decided to take the contents of the innerText property for the page and display it in a textbox. I also added a tab control to my form to separate the Web Browser control from the text extracted from the page. Now, once the user has found the correct book, he or she clicks on the Show Text button and I switch to the tab with the text from the Web Browser control.

To illustrate how to take the information that's returned by the Web Browser control and integrate it into an Access application, I added some textboxes onto my form to simulate an application. If you look at the results from the search on the Web, you'll see that there are normally four pieces of information returned by the Library of Congress page: the Author and Title, the Publishing information, and the Library of Congress Call Number. I added four textboxes to my form, as shown in Figure 2. (My sample form is unbound, and I'm not storing the data in a table, but that's a fairly straightforward extension that I won't bother demonstrating.)

The approach I took to the application was to allow the user to highlight the details for the specific book in which he was interested, click a button, and have the highlighted information transferred to the appropriate textboxes. To do this, I took advantage of the fact that textbox controls have three properties associated with what's currently selected in them. The SelStart property indicates the position of the first character selected, the SelLength property indicates how many characters are selected, and the SelText property represents the actual text selected. However, these three properties are only available for use when the textbox has focus. Since I'm making the user click a button when the appropriate text has been selected, I had to cheat a little. First, I declared three module-level variables at the beginning of the module:

```
Dim mlngSelStart As Long
Dim mlngSelLength As Long
Dim mstrSelText As String
```

I populated them in the textbox's LostFocus event, which will have to fire as the user switches from the textbox to the button:

```
Private Sub txtText_LostFocus()

  mlngSelLength = Me.txtText.SelLength
  mlngSelStart = Me.txtText.SelStart
  mstrSelText = Me.txtText.SelText

End Sub
```

As you'll see, I don't need the SelStart and SelLength properties. I included the code to show how to access the properties.

Now, in the Click event of the button, I'm able to refer to the selected text and parse it so that I can populate the individual fields on my form. Unfortunately, the data wasn't perfect for what I wanted to do: As is illustrated here, some of the values returned by the search span more than one line of text:

```
Author:       Vogel, Peter, 1953-
Title:        The Visual Basic object and component
                 handbook / Peter Vogel.
Published:    Upper Saddle River, NJ : Prentice Hall, 2000.
LC Call No.:  QA76.73.B3V63 2000
```

Another thing to point out is that some of the entries

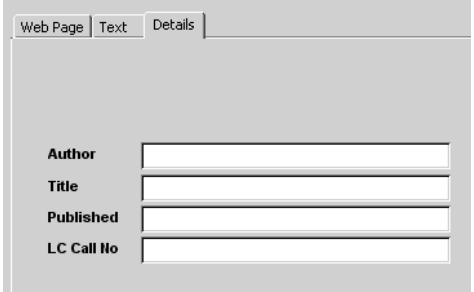

Figure 2. Textboxes to hold information parsed from the Web page.

(such as books Peter edited) don't have all four fields defined—principally, the author entry is missing:

```
Title:          Experts on Access : the best from
                Smart Access / [editor, Peter Vogel.]
Published:      Marietta, GA : Pinnacle Pub., c1998.
LC Call No.:    QA76.9.D3E985 1998
```

And, to make matters more interesting, some entries have more than the four fields:

```
Author:       Vogel, Peter, 1953-
Title:          Professional Web parts and custom
                 controls with ASP.NET 2.0 / Peter Vogel
Published:    Indianapolis, IN : Wiley Technology
              Pub., 2005.
LC Call No.:  TK5105.8885.A26V64 2005
Access:       Table of contents
     Location:  http://www.loc.gov/catdir/toc/ecip0516
                 /2005021557.html
```

To handle these entries, I developed some custom parsing logic. I split the selected text into individual lines and look at the first 15 characters of each line. If there's text there (or, more accurately, if there's a colon there), I save whatever precedes the colon as strEntryType, and save what comes after the colon as strEntryValue. If there's no colon, I assume that the current line is a continuation of the previous line, and append the text to the current strEntryValue. I use strEntryType to decide what kind of data I have in strEntryValue. I use a Select Case statement that recognizes the four strEntryTypes that I'm interested in (Author, Title, Published, and LC Call No.) and just ignore any other entries.

The code is a little convoluted, primarily because I can't be sure that I've found a complete entry until I find a colon in the first 15 positions of the next line. In other words, once I find a colon, I know that I've already found the end of my current set of data and have just found the strEntryType for the next set of data that I'll read. I need to move my strEntryValue into the textbox specified by the *previous* strEntryType that I read and then set strEntryType to the data that I just read.

The code begins by declaring the variables that I'll need:

```
Private Sub cmdCopy_Click()

Dim intChar As Integer
Dim intLoop As Integer
Dim intColon As Integer
Dim strEntryType As String
Dim strEntryValue As String
Dim varSelect As Variant
```

I then Initialize my four textboxes to nothing:

```
  Me.txtAuthor = vbNullString
  Me.txtTitle = vbNullString
  Me.txtPublished = vbNullString
  Me.txtLCCallNumber = vbNullString
```

I use the Split function (with the delimiter set to vbCrLf) to divide the selected text into separate lines and place them in an array:

```
  varSelect = Split(mstrSelText, vbCrLf)
```

One warning: This code works because the text is split using the standard Carriage Return/Line Feed combination. You need to check the specific Web page you're dealing with to make sure this is the case for you.

I need to ensure that strEntryType is empty before I start so that I don't place the first line of text into a textbox before determining whether the text continues on the next line. My logic checks to see if there's a colon in the first 15 positions and, if there is, writes what's currently in strEntryValue to the appropriate textbox. If there isn't a colon, I append what's in the line to whatever's already in strEntryValue:

```
  strEntryType = vbNullString
  For intLoop = LBound(varSelect) To _
    UBound(varSelect)
    intColon = InStr(varSelect(intLoop), ":")
    If intColon > 1 And intColon < 16 Then
      Select Case strEntryType
        Case "Author"
          Me.txtAuthor = strEntryValue
        Case "Title"
          Me.txtTitle = strEntryValue
        Case "Published"
          Me.txtPublished = strEntryValue
        Case "LC Call No."
          Me.txtLCCallNumber = strEntryValue
        Case Else
      End Select
      strEntryType = _
        Trim$(Left$(varSelect(intLoop), _
        intColon - 1))
      strEntryValue = _
        Trim$(Mid$(varSelect(intLoop), _
        intColon + 1))
    Else
      strEntryValue = strEntryValue & " " & _
        Trim$(varSelect(intLoop))
    End If
  Next intLoop
```

Now that I've read all the lines of data, I may still have the last line of data to write (depending on what the value of strEntryType is). This code handles that case:

```
  Select Case strEntryType
    Case "Author"
      Me.txtAuthor = strEntryValue
    Case "Title"
      Me.txtTitle = strEntryValue
    Case "Published"
      Me.txtPublished = strEntryValue
    Case "LC Call No."
      Me.txtLCCallNumber = strEntryValue
    Case Else
  End Select
```

Finally, once I've populated the textboxes, I set focus to the tab in my tab control that actually holds the textboxes:

```
  Me.pagDetails.SetFocus

End Sub
```

As always, parsing text files (or HTML) leads to non-

generic solutions and lots of custom code. There's also the possibility that the Web site from which you're getting your information will change its layout, forcing you to rewrite your parsing routine. However, if you're willing to live with those limitations, parsing a Web site's innerText property can be a useful technique. For more information about the Web Browser control, see http://msdn.microsoft.com/workshop/browser/webbrowser/browser_control_node_entry.asp.

My reader had one final question, though.

Why is this solution limited to Access 2000 and newer?

Unfortunately, Internet Explorer 4.0 broke Access 97's ability to use the Web Browser control. If you try to use it in Access 97 (even if it's a converted database from Access 2000 or newer), you'll get a "There Is No Object in This Control" error. There's a KB article documenting this at http://support.microsoft.com/?id=177105, but unfortunately there's no workaround. ▲

604STEELE.ZIP at www.pinnaclepublishing.com

Doug Steele has worked with databases, both mainframe and PC, for many years. Microsoft has recognized him as an Access MVP for his contributions to the Microsoft-sponsored newsgroups. Check http://I.Am/DougSteele for some Access information, as well as Access-related links. He enjoys hearing from readers who have ideas for future columns, though personal replies aren't guaranteed.