Access Answers: Doing two things at once, and Web Service wannabes

Doug Steele

Doug will address commonly asked questions from Access developers. This month, he looks at how to update and insert in one step, as well as looking at a way to get information from the Internet.

If I've got new data, how can I process it so that matching rows in my master table are updated, and new rows are inserted?

I can remember having to write complicated "balance line" programs to solve this with code in the past. Fortunately, it's possible to do it in Access using a single query without having to write a line of VBA code!

Remember that queries can join tables together. The most common join is probably an Inner Join, which combines records from two tables whenever there are matching values in a field common to both tables. A Left Outer Join also combines records from two table, but it includes all of the records from the first (left) of two tables, even if there are no matching values for records in the second (right) table. That sounds useful in this situation, doesn't it? I have a set of new values, and a set of existing values, and I want to combine the values, whether or not each of the new values correspond to an existing value.

Let me walk you through how to do it in the Query Builder.

For the sake of discussion, I'll assume that there are two tables (ExistingData, and NewData), and that the two tables are identical in structure.

1. Create a new query
2. Add tables NewData and ExistingData to the query.
3. If a relationship line isn't drawn between the two tables, do so now, relating them by their primary key. (Note that if it's a compound primary key, you need to join on each of the fields in the index.)
4. Double-click on the relationship line joining the two tables, and choose the option "Include ALL records from 'NewData' and only those records from 'ExistingData' where the joined fields are equal.", then click OK. (If you're dealing with a compound primary key, you need to do this for each line joining the tables.)
5. Drag all of the fields from table ExistingData into the query grid.
6. From the Query menu, select Update Query to change the Select query to an Update query.
7. Now, for every field in the query, go to the Update To cell and type [NewData].[<name of the field>]. (Don't type the <> I show around "name of the field". Yes, this is time consuming, and there's no automated way to do it, but fortunately you only have to do it once!)
8. Save the query with an appropriate name.

That's it: you now have a query that will update or insert as required.

If you look at the SQL for this query, you should have something like:

```
UPDATE NewData LEFT JOIN ExistingData
ON NewData.ID = ExistingData.ID
SET ExistingData.ID = [NewData].[ID],
ExistingData.Field1 = [NewData].[Field1],
ExistingData.Field2 = [NewData].[Field2],
ExistingData.Field3 = [NewData].[Field3]
and so on
```

When you run the query, it brings back one row for each row in the NewData table, whether or not a corresponding row exists in the ExistingData table. (The values will be Null for each field in the ExistingData table when the value exists in NewData but not in ExistingData.). Each row is updated to include the values from the corresponding row in the NewData table. You're updating either existing values

(when the corresponding row does exist in ExistingData), or Nulls (when the corresponding row doesn't exist in ExistingData).

Hopefully it's obvious that this will only work if you have a Primary Key defined for each table. It doesn't matter whether that Primary Key is a single field, or you've got a compound key as your Primary Key, as long as it's the same in both tables.

There's a web page that has information that changes over time. I want to be able to read that page, and get the most up-to-date information from it. How can I do this?

Realistically, this sounds like a job for Web Services. However, sometimes the provider doesn't provide a Web Service. Depending on the web page, it may be possible to simulate a Web Service using a technique often referred to as "screen scraping".

To do this, I'm going to take advantage of the XMLHTTP object that should already exist on your machine, even if you don't know about it. As long as you've installed at least Internet Version 4.0, you should have the file msxml.dll on your machine. If it's there, you're in business.

What I'm going to describe isn't the prettiest, but I find it works. It does require that you have a bit of knowledge of HTML, since what you're going to be retrieving from the web site is the actual HTML that makes up the page, not the text that appears in your browser. As well, you'll see that this technique isn't really that amenable to sites which change their format frequently. With that being said, here goes.

The following is a module that will return the HTML associated with a given URL and print it to the debug window. Sorry, but I'm going to completely ignore how you initiate communication with the Internet if you don't have a dedicated connection. Since I have cable, I have no way of testing the code. If it's any consolation, I can at least point you to code that you can use to determine whether or not you have a live Internet connection. Microsoft VB MVP Randy Birch has some code at http://www.mvps.org/vbnet/code/network/internetgetconnectedstate.htm at his VBNet site. You could also take a look at his http://www.mvps.org/vbnet/code/network/isdestinationreachable.htm, but that requires some tweaking to use in Access, as it uses a control array, which Access doesn't support. Another bit of code you can check is Michael Kaplan's `connect.bas` module at http://www.trigeminal.com/codes.asp (I found that using `0&,` where that's a zero followed by an ampersand, instead of `StrPtr(App.Title)` in the call to the `InternetOpen` function worked).

Having said that, here's one way to retrieve the content of a given web page:

```
Function GetFromWebpage( _
  URL As String_
) As String
On Error GoTo Err_GetFromWebpage

Dim objWeb As Object
Dim strXML As String

' Instantiate an instance of the web object

  Set objWeb = CreateObject("Microsoft.XMLHTTP")

' Pass the URL to the web object, and send the request

  objWeb.Open "GET", URL, False
  objWeb.Send

' Look at the HTML string returned

  strXML = objWeb.responseText
  GetFromWebpage = strXML

End_GetFromWebpage:
' Clean up after ourselves!
  Set objWeb = Nothing
  Exit Sub

Err_GetFromWebpage:
```

```
' Just in case there's an error!
  MsgBox Err.Description & " (" & Err.Number & ")"
  Resume End_GetFromWebpage

End Sub
```

As the comments state, you're instantiating an instance of the XMLHTTP object, giving it a URL to retrieve and looking at the responseText property of the object once the call has completed. The problem with this code is that there's a lot of stuff returned that you probably don't care about: you get all of the HTML required to render the page in your browser, not just the data for which you're looking. If there's a specific piece of information on the page that you want, you have to parse the page (using functions like InStr, Left, Mid and Right) to get only that part of the HTML of interest to you.

I'll try to present a simple example. Let's say you want to be able to get stock quotes. There are lots of web sites that'll give you stock quotes. For no particular reason, I'll choose to use http://finance.yahoo.com. Looking at that site, you can go directly to a given stock's metrics if you know the symbol for that stock. Picking everyone's favourite <g>, you can look for Microsoft's stock price by going to http://finance.yahoo.com/q?s=msft (Warning: Here's where writing these articles weeks or months in advance becomes a problem! When I wrote this at the beginning of August, the site appeared as shown below. However, there was a warning on the page that they were in the process of testing a new interface. Hopefully they haven't implemented it before you read this!)

Symbol	Last Trade		Change		Volume
MSFT	12:22pm	25.62	-0.09	-0.35%	17,721,532

Image 1: Sample Webpage data

When I look at the HTML associated with that table, I see something like:

```
<table cellspacing=1 cellpadding=4 border=0 width=100%>
<tr bgcolor=dcdcdc><th nowrap><font face=arial size=-1>Symbol</font></th>
<th nowrap colspan=2><font face=arial size=-1>Last Trade</font></th>
<th nowrap colspan=2><font face=arial size=-1>Change</font></th>
<th nowrap><font face=arial size=-1>Volume</font></th>
</tr>
<tr align=right bgcolor=white>
<td nowrap align=left><font face=arial size=-1><a
href="/q?s=MSFT&d=t">MSFT</a></font></td>
<td nowrap align=center><font face=arial size=-1>12:22pm</font></td>
<td nowrap><font face=arial size=-1><b>25.62</b></font></td>
<td nowrap><font face=arial size=-1><font color=ff0020>-0.09</font></font></td>
<td nowrap><font face=arial size=-1><font color=ff0020>-0.35%</font></font></td>
<td nowrap><font face=arial size=-1>17,721,532</font></td>
</tr>
and so on
```

It's always the same: the stock symbol appears after the HTML tags `<font face=arial size=-1><a href="/q?s=`, the Last Trade information follows the next `<font face=arial size=-1>` tag, and the actual stock price follows the `<font face=arial size=-1>` tag after that (enclosed in `<b></b>` tags). It isn't that difficult to write a function that retrieved the HTML for the entire page, and then searched for those specific bits of text in the HTML. The code above retrieves the entire HTML, so all I need to do is write code to parse all of the ugly parts and find the important stuff buried in it:

```
Function GetStockQuote( _
    CompanySymbol As String _
) As String

On Error GoTo Err_GetStockQuote

Dim objWeb As Object
Dim lngDateEnd As Long
Dim lngDateLen As Long
Dim lngDateStart As Long
Dim lngPointer As Long
```

```
Dim lngQuoteEnd As Long
Dim lngQuoteLen As Long
Dim lngQuoteStart As Long
Dim strDate As String
Dim strHTML As String
Dim strQuote As String
Dim strReturn As String
Dim strSearchFor As String
Dim strURL As String

' Initialize the URL to search

  strURL = "http://finance.yahoo.com/q?s=" & _
           LCase$(CompanySymbol)

' Use our GetFromWebpage function to
' retrieve the HTML from the site

  strHTML = GetFromWebpage(strURL)

' Set a default message that will (hopefully)
' get changed when we find the quote.

  strReturn = "Sorry: Couldn't get the quote for " & _
              UCase$(CompanySymbol)

' Set the string to look for in that mess of HTML

  strSearchFor = "<font face=arial size=-1>" & _
                 "<a href=""/q?s=" & _
                 CompanySymbol & "&d=t"
  lngPointer = InStr(1, strHTML, _
                 strSearchFor, vbTextCompare)

' If lngPointer is greater than 0, that means that
' the string was found.

  If lngPointer > 0 Then

' Set the next string to search for, and change where
' in the string to search to after what we just found

    strSearchFor = "<font face=arial size=-1>"
    lngPointer = lngPointer + Len(strSearchFor)
    lngDateStart = InStr(lngPointer, strHTML, _
                   strSearchFor, vbTextCompare)
    If lngDateStart > 0 Then

' If strSearchFor was found, we assume that the
' date starts at the end of the string for which we
' were searching

      lngDateStart = lngDateStart + Len(strSearchFor)
      lngDateEnd = InStr(lngDateStart, strHTML, _
                   "</font>", vbTextCompare)
      If lngDateEnd > 0 Then

' We assume that the date ends where </font> was found

        lngDateLen = lngDateEnd - lngDateStart
        strDate = Mid$(strHTML, lngDateStart, lngDateLen)
        lngQuoteStart = InStr(lngDateEnd, strHTML, _
                        strSearchFor, vbTextCompare)
        If lngQuoteStart > 0 Then

' If strSearchFor was found, we assume that the
' quote starts at the end of the string for which we
' were searching

          lngQuoteStart = lngQuoteStart + Len(strSearchFor)
          lngQuoteEnd = InStr(lngQuoteStart, strHTML, _
```

```
                        "</font>", vbTextCompare)
          If lngQuoteEnd > 0 Then

' We assume that the quote ends where </font> was found

            lngQuoteLen = lngQuoteEnd - lngQuoteStart
            strQuote = Mid$(strHTML, lngQuoteStart, lngQuoteLen)

' We know know everything for which we were looking.
' Concatenate it into a string that the function can return.

            strReturn = UCase$(CompanySymbol) & _
                        " last traded for " & _
                        RemoveBold(Trim$(strQuote)) & _
                        " at " & Trim$(strDate)
          End If
        End If
      End If
    End If
  End If

End_GetStockQuote:
' Clean up after ourselves!
  Set objWeb = Nothing
  GetStockQuote = strReturn
  Exit Function

Err_GetStockQuote:
' Just in case there's an error!
  MsgBox Err.Description & " (" & Err.Number & ")"
  Resume End_GetStockQuote

End Function
```

The RemoveBold function I use when concatenating everything for output is simply:

```
Function RemoveBold(StringToEdit As String) As String

  RemoveBold = Replace( _
               Replace(StringToEdit, "<b>", vbNullString), _
               "</b>", vbNullString)

End Function
```

(In other words, replace all occurrences of either or with "")

As you can see, it isn't particularly pretty code, but it does what it's supposed to. The problem, of course, is that this code will fail to work if the web page changes (including errors in the HTML generated to produce the page). And as you can see, a reasonable understanding of HTML code is required to be able to pinpoint the exact location in the file of the information you want.

Doug Steele has worked with databases, both mainframe and PC, for many years. He has taught introductory computer programming at the University of Waterloo. Microsoft has recognized him as an Access MVP for his contributions to the Microsoft-sponsored newsgroups over the years. Check http://I.Am/DougSteele for some Access-related links. You can reach him at AccessHelp@rogers.com, but please note that personal replies are not guaranteed. He does, however, welcome suggestions for topics for future columns. (Don't make him beg…)