Ordering Controls, Fixing Bugs, and Speeding Up Remote Databases

Christopher Weber

95 97 2000 2002 DOWNLOAD

Chris Weber returns to his list of most asked questions to address some pressing issues for Access developers. This month he looks at managing your control's tab order, provides an easy solution for an annoying Access bug, and shows an easy way to improve performance when retrieving large recordsets from remote computers.

I have numerous forms to build that will display various fields from my tables in columns. I break them down into logical groups using pages on tab controls as I build the forms, but setting the tab order on each tab is a pain. First, I have to select the tab page that they're on, and then I select Tab Order... from the View menu. There are too many controls on each page to view through the Tab Order dialog, which won't expand. If I click the Auto Order button it orders them left to right, and then top to bottom. Is there an easier way?

Whenever I find myself spending inordinate amounts of time on a repetitive task like you've described, I consider building my own tool to do the job. In this case, I just happen to have built what you need for a project last year. My client's changing requirements were driving me crazy as he kept tinkering with the form colors and moving the order of the controls on the screen and forgetting to fix the tab order.

You can use my SetTabsInPhoneBookOrder() routine in the sample database (in the Download file that's available at www.smartaccessnewsletter.com) to set the tab order to "phonebook column order" for all the controls on a form or just a page's controls. You should be able to use the procedure as is, but for those who like to look under the hood, I'll give a brief explanation.

The procedure begins by accepting the name of the form you want to work with and, optionally, the name of a page on a tab control. It opens the form whose name you've passed in and then re-dimensions a variant array, avarCtls, giving it three columns and no rows. If the optional varPageName parameter is missing, the code uses a For Each loop to examine all of the controls on the form. I use my HasProperty function to determine whether each control has a TabIndex property. If it does, I retrieve the second dimension of avarCtls (the row dimension) and then resize the array to create a new row. I then load the control's name, left, and top properties into the three columns of the array.

If the page name parameter is supplied, the procedure is the same, but instead of using the form's default Controls collection, I use the Controls collection of the page, which itself appears in the Controls collection of the form. Here's the code:

```
Sub SetTabsInPhoneBookOrder(frmName As String, _
          Optional varPageName As Variant)
'by CRW @ DSW 092302
'uses 3 column array of |name|left|top| to set
'tab order of controls on a form
'or a tab on form
'sorts controls by left then by top
'and assigns control's tabstop to Nth position in array
Dim ctl As Control, i As Integer, j As Integer, _
          fChanged As Boolean
Dim strName As String, dblLeft As Double, _
          dblTop As Double

  ReDim avarCtls(3, 0)
  DoCmd.OpenForm frmName, acDesign
  'load all tab-able ctls to array
  If IsMissing(varPageName) Then
    For Each ctl In Forms(frmName)
      If HasProperty(ctl, "TabIndex") Then
        i = UBound(avarCtls, 2)
        ReDim Preserve avarCtls(3, i + 1)
        avarCtls(1, i + 1) = ctl.Name
        avarCtls(2, i + 1) = ctl.Left
        avarCtls(3, i + 1) = ctl.Top
      End If
    Next ctl
  Else
    'page name provided
    For Each ctl In Forms(frmName).
        Controls(varPageName).Controls
      If HasProperty(ctl, "TabIndex") Then
        i = UBound(avarCtls, 2)
        ReDim Preserve avarCtls(3, i + 1)
        avarCtls(1, i + 1) = ctl.Name
        avarCtls(2, i + 1) = ctl.Left
        avarCtls(3, i + 1) = ctl.Top
      End If
    Next ctl
  End If
```

The next step is to sort the array in left/top order. I use a simple bubble sort technique—comparing the left value of ith row in the array with the following row

(avarCtls(2, i) and avarCtls(2, i + 1)). If the left value of the preceding row is greater than that in the next row, I switch the values in the rows. If that isn't the case, I then check to see if the left values are equal. If they are, then the two controls represented by these rows are in the same column on the form or page. I then compare the top properties (avarCtls(3, i) and avarCtls(3, i + 1)). If the top value of the preceding row is greater than that in the next row, again, I switch the values in the rows.

Whenever a switch takes place, I set fChanged to True, which causes the Do loop to repeat. When I finally do a pass through the loop without making an exchange, the loop ends:

```
Do
    fChanged = False
    For i = 1 To (UBound(avarCtls, 2) - 1)
      If avarCtls(2, i) > avarCtls(2, i + 1) Then
        strName = avarCtls(1, i)
        dblLeft = avarCtls(2, i)
        dblTop = avarCtls(3, i)
        avarCtls(1, i) = avarCtls(1, i + 1)
        avarCtls(2, i) = avarCtls(2, i + 1)
        avarCtls(3, i) = avarCtls(3, i + 1)
        avarCtls(1, i + 1) = strName
        avarCtls(2, i + 1) = dblLeft
        avarCtls(3, i + 1) = dblTop
        'a bubble moved
        fChanged = True
      ElseIf avarCtls(2, i) = avarCtls(2, i + 1) Then
        If avarCtls(3, i) > avarCtls(3, i + 1) Then
          'control(i) is lower than control(i+1) -
          'switch the values in the rows
          strName = avarCtls(1, i)
          dblLeft = avarCtls(2, i)
          dblTop = avarCtls(3, i)
          avarCtls(1, i) = avarCtls(1, i + 1)
          avarCtls(2, i) = avarCtls(2, i + 1)
          avarCtls(3, i) = avarCtls(3, i + 1)
          avarCtls(1, i + 1) = strName
          avarCtls(2, i + 1) = dblLeft
          avarCtls(3, i + 1) = dblTop
          'a bubble moved
          fChanged = True
        End If
      End If
    Next i
  Loop Until fChanged = False
```

The procedure finishes by assigning the Tab Stop property of each control on the form I've opened using the control name found in the first column of the array. I do the tab assignments in reverse order. This way, the form doesn't reassign values when conflicts arise. If you want to view the end result before applying the changes, simply comment out the assignment line inside the loop and uncomment the Debug statement:

```
For i = UBound(avarCtls, 2) To 1 Step -1
  'Debug.Print avarCtls(1, i) & Space(8) & _
  '  avarCtls(2, i) & Space(8) & avarCtls(3, i)
  Forms(frmName).Controls(avarCtls(1, i))._
                    TabIndex = i - 1
Next i

End Sub
```

Before you use SetTabsInPhoneBookOrder(), be sure to align all of your controls in each column from the Format menu, using Format | Align | Left. Also, the code leaves your form open in design view with the changes unsaved. This is so that you can review what's taken place and make any adjustment for misaligned controls before you save. Now, if Microsoft would add this as an option to the Tab Order dialog in Access 11, we might actually get a new feature worth paying for!

Sometimes when I'm working in the VBA editor, a line of code that worked one day won't work the next. In particular, if I use the name of a field with the Me keyword (Me.fieldname), my code compiles properly and runs. A day later, an unrelated change in the form module causes the Me.fieldname reference to fail. The field still appears in the properties list when I type Me., but the compiler won't recognize it. Have you heard of this problem?

This particular "feature" used to make me so crazy that, for a while, I stopped using Me.fieldname references in my form code, and put hidden controls on my forms for any fields that I needed to reference. Then one day I discovered that I could make it consistently happen if I removed the recordsource from the form. This made sense, as the set of fields returned by the recordsource would no longer be available to the VBA editor. So, the next time it happened to me, I opened the recordsource for the form, ran the query, and closed it again. Magically, the fieldname bug disappeared. Rerunning the query apparently refreshed the fields list for the form and the editor was happy. I think this may work for you, too.

When working in design view, the boundaries of my controls disappear when I change their Special Effect from raised to flat, or whatever. I've tried reassigning the line color, but the boundary won't appear on the screen. If I save and close the form and then re-open it, all is well. What am I doing wrong?

Absolutely nothing. Can you say "graphics glitch?" For whatever reason, the form designer sometimes loses the graphical representation of controls when their appearance properties are changed. If this happens again, just use the vertical scrollbar to move the control out of sight and then back again. The boundary will reappear, as it should.

While working with DAO recordsets from SQL Server, I've seen degradation in speed as Jet retrieves records. The larger the set, the worse the problem becomes. I thought Jet was supposed to be smart about managing the number of records it retrieves!

I'm the first person to stand up for Jet—there are times

when running the database engine on the client machine is the right thing to do. And, like you, I still use DAO. I'm a stubborn dinosaur, and I've watched RDO and ODBC Direct go by the wayside. And now ADO will become obsolete, as it barely resembles ADO.NET. There are things you can't do with DAO. But the same holds for ADO, and I haven't needed its features to complete my clients' requirements.

And yes, Jet is very smart about managing the number of records it retrieves—when it knows how many records there are. Unfortunately, Jet doesn't keep statistics on server tables; it only has that information for the tables in the MDB file. So, as you retrieve thousands of records to process from a server, Jet keeps background fetching more records whenever it gets the chance, causing the RAM on your local machine to max out, and producing the degradation in processing performance.

You could try controlling the retrieval through an MSysConf table on the server. If Jet finds that table, Jet allows settings within it to control how often and how many records it retrieves at a time. I don't think this is a very good solution, though, because MSysConf is static, separate from your application, and you'd have to take your best guess at delay times and record chunking to have any success.

There's a simple solution, however, and tests at our offices have shown that recordset processing times can be cut 33 to 66 percent by using it. Use the .CacheSize, .CacheStart, and .FillCache methods of the recordset object inside your processing loop to optimize record retrieval. These properties are designed to work with server data *only*. They have no effect on Jet tables. Typical code for using the properties looks like this:

```
Option Compare Database
Option Explicit
Const CACHE_SIZE = 50

Sub Whatever()
Dim rst As Recordset, lngCacheSize As Long

  With rst
    .CacheSize = CACHE_SIZE
    .FillCache
    Do While Not .EOF
      'processing here
      '
      '
      .MoveNext
      lngCacheSize = lngCacheSize + 1
      If lngCacheSize Mod CACHE_SIZE = 0 Then
        .CacheStart = .Bookmark
        .FillCache
      End If
    Loop
  End With

End Sub
```

When you begin using a large set of records from a

server, set the CacheSize property to a comfortable level. In this example, I've created a module-level constant called CACHE_SIZE and set it to 50 records. I then load the cache using the .FillCache method and begin looping. I use lngCacheSize to track how many records I've looped through. After moving to the next record, I test whether the cache is used up (lngCacheSize Mod CACHE_SIZE = 0). If it is, I set the CacheStart property to the bookmark of the current record (which still hasn't been processed) and refill the cache with the next 50 records.

By chunking through the records on the server, you save resources on your local machine and processing speeds improve. You'll have to experiment with the size of the cache, but any reasonable size will improve performance dramatically. And remember, this *only* works against servers, *not* for Jet tables in an .mdb file. ▲

AA0703.ZIP at www.smartaccessnewsletter.com

Christopher Weber is a consultant with the DSW Group (www.thedswgroup.com), a software training and development firm in Atlanta, GA, specializing in Access, Java/J-Builder, C#, and Delphi database applications. Chris has authored Access courseware for Application Developers Training Company, Softbite International, Borland Europe, and the DSW Group. He has also co-authored *Developing with Delphi—Object Oriented Techniques* (Prentice Hall), *Human Factors in Geographic Information Systems* (Belhaven Press), *Visualization in Modern Cartography* (Pergamon Press), and *Spatial and Temporal Reasoning in Geographic Information Systems* (Oxford University Press) and has been a frequent reviewer for *Cartography* and *GIS* magazine. An Access developer since its first release, Chris enjoys working with clients, and heads the DSW Group's Access development and training team, traveling throughout the country teaching Access development and programming seminars. www.access-training.com,