Publicado por primera vez en el blog de Juan Cots

Si hay algo que me gusta mucho del paquete Office, es su capacidad de interactuar entre las diferentes aplicaciones que lo componen; mediante Visual Basic para Aplicaciones (VBA) podemos manejar una hoja de cálculo (por ejemplo) desde Access, Word u otras aplicaciones, como si estuviéramos en el propio Excel, de este modo desde Access podremos trabajar con el modelo de objetos de Excel como si fuera el propio, a esto, Microsoft, ha dado en llamarlo **Automatización**.

Hoy en día son muchas las aplicaciones que utilizan VBA como lenguaje de programación, no solo las desarrolladas por Microsoft, sino también otras de terceros: Autocad, Corel, Osisoft PI, …

En Automatización hablaremos del **Cliente** y del **Servidor**: el Cliente sería por ejemplo Access, cuando desde este abrimos una instancia de Excel, que en este caso será el Servidor.

La Automatización puede tener un inconveniente en los cambios de versión. Si referenciamos a versiones concretas de una aplicación, Microsoft Excel 11.0 Object Library por ejemplo, e intentamos hacer trabajar ese código con una versión diferente de Office, podemos tener problemas por no encontrarse esa referencia en concreto.

Por esa razón prefiero realizar un "Late Binding" sin establecer ninguna referencia y dimensionar los objetos como Object, creando el objeto a posteriori. Esto es un poco menos eficiente que hacerlo previamente, pero me evitará, a la larga, problemas con las referencias. Siempre podremos, para aprovecharnos del "IntelliSense" de VBA, declarar las referencias y dimensionar los objetos como tales mientras programamos, para una vez finalizada la aplicación, eliminar las referencias y cambiar el dimensionado de objetos por Object.

Seguramente me explico mejor con un ejemplo, así que veamos este fragmento de código, tomado de uno de los ejemplos de mi Web, que podéis descargar desde aquí http://www.mvp-access.es/emilio/Descarga.asp?IdEjemplo=29

```
'*************************************************************************
'* cmdCalcularExcel_Click
'* Este procedimiento realiza la lectura de datos de una tabla Access, para a
'* continuación escribirlos en Excel, tomando después el resultado de un calculo
'* de otra celda escribiéndolo a de seguido en la tabla Access
'* ha de incluirse una referencia a Microsoft Excel X.0 Object Library
'* también a Microsoft DAO 3.X Object Library
'* ESH 14/12/04 15:16
'*************************************************************************

Private Sub cmdCalcularExcel_Click()
On Error GoTo cmdCalcularExcel_Click_TratamientoErrores

Dim xls As Excel.Application, _
    rst As DAO.Recordset, _
    strSQL As String, _
    strNombreArchivo As String

' construyo la ruta del archivo Excel en que realizar los cálculos
strNombreArchivo = CurrentProject.Path & "\12-ExportImportExcel.xls"

If Dir(strNombreArchivo) = "" Then
   MsgBox "El archivo " & strNombreArchivo & " No existe", vbCritical + vbOKOnly, "ATENCION"
   Exit Sub
End If

' creo una instancia a Excel
```

```
Set xls = CreateObject("Excel.Application")

xls.Workbooks.Open (strNombreArchivo)

' activo la hoja 1
xls.Worksheets("Hoja1").Activate

' aplico recalculo automático
xls.Application.Calculation = xlCalculationAutomatic

' abro un recordset con los campos calculados vacíos
strSQL = "SELECT Radio, Alto, Volumen "
strSQL = strSQL & "FROM Cilindros "
strSQL = strSQL & "WHERE Volumen = 0 "

Set rst = CurrentDb.OpenRecordset(strSQL, dbOpenDynaset)

' si el recordset tiene datos
If Not rst.EOF And Not rst.BOF Then
   ' hago un bucle con todos los registros
   Do While Not rst.EOF
      ' escribo el radio en la celda A1
      xls.ActiveSheet.Range("A1") = rst!Radio
      ' escribo la altura en la celda A10
      xls.ActiveSheet.Range("A10") = rst!Alto
      ' guardo el resultado de la celda B9 en el campo Volumen
      rst.Edit
      rst!Volumen = xls.ActiveSheet.Range("B9")
      rst.Update
      ' salto al siguiente registro
      rst.MoveNext
   Loop
End If

' cierro el recordset
If Not rst Is Nothing Then
   rst.Close
   Set rst = Nothing
End If

' cierro Excel y para evitar que me pregunte si quiero guardar,
' le engaño diciendo que ya está guardado
xls.ActiveWorkbook.Saved = True
xls.Application.Quit
Set xls = Nothing

' refresco el formulario
Me.Secundario1.Requery

cmdCalcularExcel_Click_Salir:
   On Error GoTo 0
   Exit Sub

cmdCalcularExcel_Click_TratamientoErrores:
   MsgBox "Error " & Err.Number & " en proc.: cmdCalcularExcel_Click de " & _
  "Documento VBA: Form_frmCilindros (" & Err.Description & ")"
   GoTo cmdCalcularExcel_Click_Salir

End Sub      ' cmdCalcularExcel_Click
```

Cómo podéis ver, en este caso he hecho un Early Binding, la declaración de variables no la hago como object sino como cada tipo concreto

```
     Dim xls As Excel.Application, _
         rst As DAO.Recordset, _
```

```
    strSQL As String, _
    strNombreArchivo As String
```

ello me obliga, tal como indico en el encabezado del procedimiento, a hacer referencia, en este caso a Excel y a DAO. Pudiera haber hecho, y repito, es mas recomendable para el caso de que la aplicación se ejecute en diferentes entornos con versiones diferentes, de la siguiente forma:

```
Dim xls As Object, _
    rst As Object, _
    strSQL As String, _
    strNombreArchivo As String
```

todo ello sin referencias extra.

Cómo veis, en el código de ejemplo, empiezo creando una instancia a Excel:

```
' creo una instancia a Excel
Set xls = CreateObject("Excel.Application")
```

A partir de este momento la variable xls, declarada como objeto genérico, se convierte en un objeto concreto y disponemos de un objeto llamado xls que no es otra cosa que una instancia Excel, que mientras no le digamos lo contrario estará oculta y a la que me referiré siempre anteponiendo xls como prefijo en todas las llamadas al código a ejecutar en el Servidor (en este caso Excel), de este modo podré ejecutar sentencias de este desde el Cliente (en este caso Access).

Por ejemplo, las siguientes líneas abren un libro Excel, activan la Hoja1 del mismo y aplican el recalculo automático

```
xls.Workbooks.Open (strNombreArchivo)

' activo la hoja 1
xls.Worksheets("Hoja1").Activate

' aplico recalculo automático
xls.Application.Calculation = xlCalculationAutomatic
```

A partir de aquí el proceso no tiene mas complicación y, como todo en este mundo de la programación, es cuestión de echarle imaginación y de experimentar, sin miedo, que no muerde, siempre, claro, que se tomen las oportunas precauciones de hacer copias de seguridad antes de comenzar con los experimentos y de guardar a buen recaudo aquello que ya funciona tal como nosotros deseamos.

Indiscutiblemente se ha de tener un mínimo conocimiento del modelo de objetos de la aplicación cliente, aunque, en la mayoría de los casos (Excel, Word, Power Point,…) disponen de algo que hecho de menos en Access, la nunca bien ponderada “Grabadora de Macros”, que nos ayuda a construir el código que con unas pequeñas adaptaciones podemos reutilizar en nuestros proyectos de Automatización.

La ayuda de Automatización está disponible para descarga en http://support.microsoft.com/kb/302460, desgraciadamente está en inglés, y al menos que yo sepa, no se ha actualizado para las últimas versiones de Office.

Emilio Sancha 20/01/10