aprenderaprogramar.com

Captura y gestión de errores. Instrucción on error. Objeto Err, Err.Number y Err.Description. Método clear en Visual Basic. (CU00353A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº52 del Curso Visual Basic Nivel I

29

© 2006-2029 aprenderaprogramar.com

CAPTURA Y GESTIÓN DE ERRORES. INSTRUCCIÓN ON ERROR. OBJETO ERR. MÉTODO CLEAR.

Introduciremos la gestión de errores o protocolo para el tratamiento de los errores detectados durante la ejecución del programa a través de la instrucción **On Error**.

On Error activa una rutina de control de errores y especifica la ubicación de la misma. Vamos a ver algunas de las posibles sintaxis:

a) `On Error GoTo [Línea]`

donde la línea puede ser especificada como un número o como una etiqueta.

b) `On Error Resume Next`

c) `On Error GoTo 0`

En definitiva se trata de indicarle al programa qué hacer en caso de que se produzca un error durante la ejecución (recuerda que los errores en compilación van a impedir que se ejecute el programa).

En el caso a) hacemos que el programa salte a la línea especificada, que tendrá que estar en el mismo procedimiento en que se ha producido el error.

En el caso b) Resume Next indica que el programa salta a la siguiente instrucción después de la que ha generado el error (se intenta continuar con la ejecución ignorando el error).

Por último c) desactiva el control de errores, con lo que en caso de producirse uno el programa se detendrá o será impredecible.

Cuando se produce un error en tiempo de ejecución (error durante la ejecución del código), las propiedades del objeto Err se llenan con información que identifica al error de forma única. Antes del error o después de ser tratado las propiedades son las de defecto (en general cero, cadenas vacías o mensajes estándar). Las propiedades del objeto Err son:

a) **Number:** número de error. Vale cero (falso) si no se ha producido un error.
b) **Source:** nombre del programa (proyecto) en que se generó el error.
c) **Description:** texto informativo sobre el error.
d) **Helpfile:** unidad, ruta y nombre del archivo de ayuda de Visual Basic, si existe. En caso de no existir contiene una cadena vacía o espacio en blanco o cero.
e) **Helpcontext:** identificador de la parte de la ayuda asociada al número de error, si existe. En caso de no existir contiene una cadena vacía, espacio en blanco o cero.

© 2006-2029 aprenderaprogramar.com

Por ejemplo podemos hacer que sobre un Label se muestre la información Err.Description, que devuelve un mensaje informativo sobre el error. Por ejemplo "Nombre o número de archivo incorrecto", “La operación aritmética ha provocado un desbordamiento”, etc.

Prueba el siguiente código:

Código versiones menos recientes VB:

```
‘Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
Dim i As Integer
Label1 = ""
On Error GoTo 88    '[Si ponemos On Error Resume Next sí se mostraría el texto]
i = Rnd * 10 ^ 16    '[Esta línea genera el error, i demasiado grande]
Label1 = Label1 & "Esta instrucción no se llega a ejecutar"
88 If Err Then MsgBox ("Se ha producido un error. Tipo de error = " & _
Err & " Descripción: " & Err.Description)
Label1 = Label1 & "La ejecución continúa"
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim i As Integer
        Label1.Text = ""
        On Error GoTo 88    '[Si ponemos On Error Resume Next sí se mostraría el
texto en el label]
        i = Rnd() * 10 ^ 16    '[Esta línea genera el error, i demasiado grande]
        Label1.Text = ("Esta instrucción no se llega a ejecutar")

88:     If Err.Number Then MsgBox("Se ha producido un error. Tipo de error = " & _
           Err.Number & " Descripción: " & Err.Description)
        Label1.Text = ("La ejecución continúa")
    End Sub
End Class
```

Con este programa generamos un error de desbordamiento (i excede los valores admisibles para Integer) y nos aparece un MsgBox que dice: “Se ha producido un error. Tipo de error = 6. Descripción: Desbordamiento” o “Descripción: la operación aritmética ha provocado un desbordamiento”.

Tras el tratamiento Err vuelve a establecerse a valores de defecto.

Podemos borrar los contenidos del objeto Err haciendo uso de la sintaxis Err.Clear (en las versiones más recientes Err.Clear()). A través de Clear se borran todas las propiedades establecidas del objeto Err. La invocación de Clear puede ser hecha por el programador, o bien ser automática tras ejecutarse un Resume, Exit Sub, Exit Function, Exit Property ó una instrucción On Error.

© 2006-2029 aprenderaprogramar.com

En el programa anterior, si después de la línea 88 mostramos Err.Number sobre un Label nos devuelve un 6. En cambio, si escribimos Err.Clear y luego mostramos Err.Number sobre un Label, nos devuelve un cero, pues hemos restablecido Err a sus valores iniciales.

Prueba este otro código:

Código versiones menos recientes VB:

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit

Private Sub Form_Load()
On Error GoTo Gestionaerror
Dim i As Integer
i = Rnd * 10 ^ 16   '[Esta línea genera el error]
Label1 = "La ejecución continúa aquí debido al Resume Next" & vbCrLf
Label1 = Label1 & i   '[Devuelve cero ya que fue imposible asignarle valor tipo integer]
Gestionaerror:
If Err.Number <> 0 Then
   GestiónError
   Resume Next
End If
End Sub

Private Sub GestiónError()
MsgBox("Se ha producido un error. Tipo de error = " & Err.Number & ". Descripción: " & Err.Description)
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        On Error GoTo Gestionaerror
        Dim i As Integer
        i = Rnd() * 10 ^ 16    '[Esta línea genera el error]
        Label1.Text = "La ejecución continúa aquí debido al Resume Next" & vbCrLf
        Label1.Text = Label1.Text & i    '[Devuelve cero ya que fue imposible asignarle valor
tipo integer]
Gestionaerror:
        If Err.Number <> 0 Then
            GestiónError()
            Resume Next
        End If
    End Sub

    Private Sub GestiónError()
    MsgBox("Se ha producido un error. Tipo de error = " & Err.Number & ". Descripción: " &
Err.Description)
    End Sub
End Class
```

© 2006-2029 aprenderaprogramar.com

En esta ocasión tenemos una etiqueta de línea a donde se remite el flujo en caso de error, y a través de esta línea se redirecciona nuevamente el flujo a un procedimiento denominado GestiónError. Ten en cuenta que al usar Resume Next se continúa la ejecución en la instrucción inmediatamente posterior a la que originó el error, y que también ocurre que las propiedades del objeto Err se restablecen. En caso contrario, se volvería a entrar en GestiónError una segunda vez.

Visual Basic dispone de más instrucciones y posibilidades relacionadas con la captura y gestión de errores, que es una parte importante cuando se desarrollan programas a nivel profesional. Nosotros en este curso nos limitamos a exponer algunas ideas básicas sobre qué son los errores y sobre las posibilidades para tratarlos.

Próxima entrega: CU00354A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

© 2006-2029 aprenderaprogramar.com