Tutoriales Aitorrio

Crear una DLL normal de Windows con Visual Basic 6.0 explicado paso a paso

Publicado: 25/Ene/2006
Actualizado: 25/Ene/2006

Hace un mes, el día de Navidad, puse un link a una página (en inglés) que explicaba cómo poder crear librerías (DLL) normales de Windows con Visual Basic 6.0, (por si no lo sabes, el VB6 solo puede crear DLL ActiveX o de automatización COM).
Durante ese tiempo he recibido alguna que otra petición de que tradujera ese artículo al castellano.
No lo traduje en su día (ni puse los pasos para hacerlo), porque quería tener autorización del autor, escribí a la gente que lo publicó, (ya que no había ningún link al correo del autor), y no han contestado, ni para decir "candemore", por tanto, y después de haber pasado más de un mes (realmente un mes y un día) sin que reciba respuesta, pues me he decidido a publicarlo en español, no una traducción de dicho artículo, sino la explicación de cómo conseguir crear DLL normales con el VB6 o el VB5.
Por supuesto, sin lo dicho en ese artículo no existiría esta explicación, por tanto el mérito es totalmente de Ron Petrusha, el autor del citado artículo. Si quieres saber dónde está el mencionado artículo "original", pulsa en este link.

Antes de empezar te recomiendo que hagas una copia de el compilador de VB6 y del "enlazador", ya que esos dos programas se van a "sobrescribir" y no es plan de que te quedes sin esas dos piezas valiosas, sin las que no podrías crear aplicaciones de Visual Basic.
Se que muchos no leéis todo lo que escribo, así que... si esto que viene a continuación no lo lees... a mi no me pidas responsabilidades... así que... advertido estás.

En lo que te explicaré a continuación se van a sobrescribir dos programas situados en la carpeta de instalación del Visual Basic, esos dos programas son: **C2.exe** (el compilador de VB) y **LINK.exe** (el enlazador), los dos estarán en la carpeta de instalación de Visual Studio 6.0 (o del VB, según la versión). Yo voy a dar por hecho de que tienes el VB6 instalado a partir del Visual Studio 6.0 y que el sistema operativo está en español, y, por supuesto que has usado la carpeta recomendada por el programa de instalación, si es así, esos dos programas de los que debes hacer una copia, están en:
C:\Archivos de programa\Microsoft Visual Studio\VB98
Cópialos y guárdalos a buen recaudo.

> **Nota:**
> En este artículo NO vamos a crear una versión del compilador C2.EXE, pero te he dicho que hagas una copia de C2.EXE por si te da por probar lo que el autor original de este "truco" te explica en su página, en la que si crea una versión de C2.EXE, pero en realidad no hace falta.

No te voy a explicar todo lo que ese hombre explica en su página, cómo descubrió el tema este, etc., etc. Lo que te voy a explicar es cómo tienes que hacer las cosas para que puedas crear tus DLL normales y corrientes, es decir las librerías (o bibliotecas) al estilo de las del propio sistema operativo y que desde VB las tenemos que usar con la instrucción Declare.

Nota:
Te advierto que aquí explico las cosas muy simples, y no es por ahorrar, es que en realidad no hace falta hacer todo lo que Ron explica en su artículo, realmente la vida de las DLL (no solo desde VB6) es más simple que todo eso, ya verás ;-)))

Empecemos por el principio.

Paso 1. Crear copia de LINK.exe con el nombre LinkLnk.exe

Lo primero que tenemos que hacer antes de crear nuestra propia versión del enlazador (linker), es hacer una copia del programa LINK.exe, para cambiarle el nombre a esa copia, ya que ese nuevo fichero será el que usemos desde nuestra versión de LINK.EXE, esta copia es independiente de la que te comenté antes. Para hacerlo, sigue estos pasos:

1. Sitúate en el directorio de Visual Basic (por ejemplo C:\Archivos de programa\Microsoft Visual Studio\VB98)
2. Selecciona el fichero LINK.EXE
3. Copia y pega
4. A la copia, cámbiale el nombre a LinkLnk.EXE

Una vez hecho esto, el contenido de la carpeta de Visual Basic 6.0 será como el de esta figura:

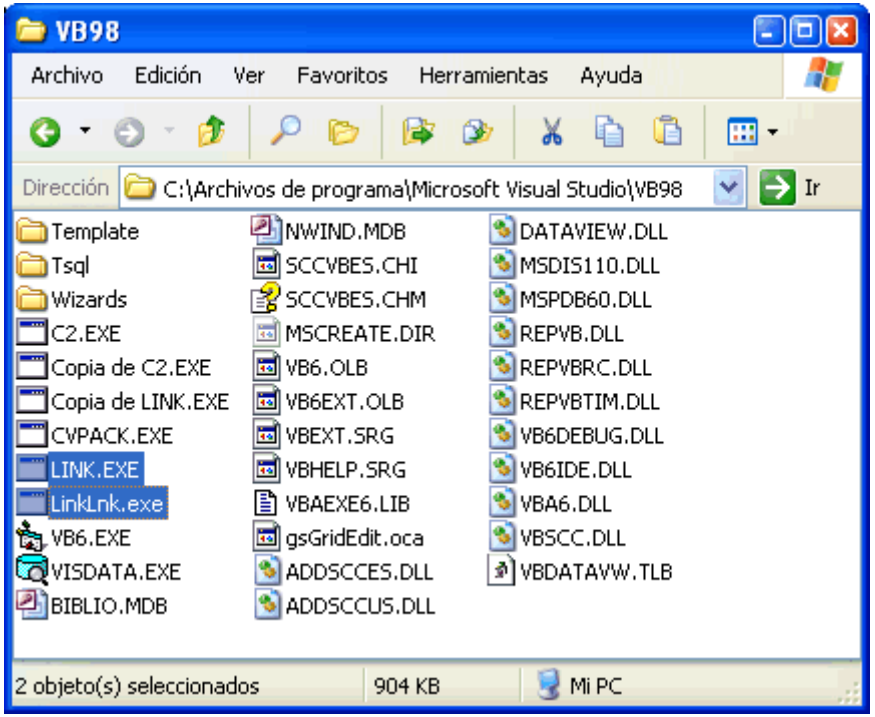

Figura 1. El directorio de VB6 después de copiar los dos ficheros

Paso 2. Crear nuestra propia versión de LINK.exe

Ahora debemos crear nuestra propia versión de LINK.exe

1. Crea un nuevo proyecto en Visual Basic 6.0 de tipo EXE estándar
2. Cambia el nombre del proyecto a LINK
3. Elimina el formulario que se crea de forma predeterminada
4. Añade un módulo BAS y cámbiale el nombre a MLINK.bas
5. En el menú Proyecto, selecciona Propiedades...
6. En el combo "Objeto inicial", selecciona Sub Main
7. Pulsa Aceptar para aceptar el cambio
8. En el menú Proyecto, selecciona Referencias...
9. Marca la casilla de Microsoft Scripting Runtime

10. Pulsa en Aceptar
11. En el módulo pega el código mostrado en el listado 1
12. Compílalo (menú Archivo>Generar Link.exe...)
13. Copia el LINK.EXE creado por este proyecto y pégalo en la carpeta de Visual Basic 6.0, te pedirá confirmación de si quieres sobrescribirlo, dile que sí

```
'-----------------------------------------------------------------------------
' Link.exe                                                          (27/Dic/05)
' Wrapper para el Link de VB6
' Basado en el código original de Ron Petrusha
' http://www.windowsdevcenter.com/pub/a/windows/2005/04/26/create_dll.html
'
' Versión reducida (sin escribir en un .LOG)                        (25/Ene/06)
' para publicar en mi sitio
'
' ©Guillermo 'guille' Som, 2005-2006
'-----------------------------------------------------------------------------

Option Explicit

Public Sub Main()

    Dim SpecialLink As Boolean, fCPL As Boolean, fResource As Boolean
    Dim intPos As Integer
    Dim strCmd As String
    Dim strPath As String
    Dim strFileContents As String
    Dim strDefFile As String, strResFile As String

    Dim oFS As New Scripting.FileSystemObject
    Dim fld As Folder
    Dim fil As File
    Dim tsDef As TextStream

    strCmd = Command$

    ' Determine if .DEF file exists
    '
    ' Extract path from first .obj argument
    intPos = InStr(1, strCmd, ".OBJ", vbTextCompare)
    strPath = Mid$(strCmd, 2, intPos + 2)
    ' Esto solo vale para VB6
    intPos = InStrRev(strPath, "\")
    strPath = Left$(strPath, intPos - 1)
    ' Open folder
```

```
Set fld = oFS.GetFolder(strPath)

' Get files in folder
For Each fil In fld.Files
    If UCase$(oFS.GetExtensionName(fil)) = "DEF" Then
        strDefFile = fil
        SpecialLink = True
    End If
    If UCase$(oFS.GetExtensionName(fil)) = "RES" Then
        strResFile = fil
        fResource = True
    End If
    If SpecialLink And fResource Then Exit For
Next

' Change command line arguments if flag set
If SpecialLink Then

    ' Determine contents of .DEF file
    Set tsDef = oFS.OpenTextFile(strDefFile)
    strFileContents = tsDef.ReadAll
    If InStr(1, strFileContents, "CplApplet", vbTextCompare) > 0 Then
        fCPL = True
    End If

    ' Add module definition before /DLL switch
    intPos = InStr(1, strCmd, "/DLL", vbTextCompare)
    If intPos > 0 Then
        strCmd = Left$(strCmd, intPos - 1) & _
              " /DEF:" & Chr$(34) & strDefFile & Chr$(34) & " " & _
              Mid$(strCmd, intPos)
    End If

    ' Include .RES file if one exists
    If fResource Then
        intPos = InStr(1, strCmd, "/ENTRY", vbTextCompare)
        strCmd = Left$(strCmd, intPos - 1) & Chr$(34) & strResFile & _
                 Chr$(34) & " " & Mid$(strCmd, intPos)
    End If

    ' If Control Panel applet, change "DLL" extension to "CPL"
    If fCPL Then
        strCmd = Replace(strCmd, ".dll", ".cpl", 1, , vbTextCompare)
    End If

    strCmd = strCmd & " /LINK50COMPAT"

End If

Shell "linklnk.exe " & strCmd
```

```
    If Err.Number <> 0 Then
       ' Error al llamar al LINKer
       Err.Clear
    End If

End Sub
```

Listado 1. El código de LINK.EXE (versión personalizada de VB6)

> **Nota:**
> Si quieres usar un icono para el ejecutable diferente al creado por el VB6, puedes hacer lo siguiente:
> -Añade un formulario al proyecto.
> -Cambia el icono del formulario por el que quieras usar.
> -En el cuado de propiedades de las propiedades del proyecto (menú Proyecto>Propiedades de Link...), selecciona la ficha Generar, y selecciona ese formulario del combo que hay junto a Icono.
> -**El Objeto inicial (de la ficha General) debe seguir siendo Sub Main**.
> -Compilar el proyecto y el ejecutable tendrá el icono indicado.

Ahora cuando quieras crear una DLL normal (al estilo de las del API de Windows), debes hacer lo siguiente:

Paso 3. Crear una DLL normal con Visual Basic 6.0

1. Crea un nuevo proyecto del tipo DLL ActiveX
2. Añade un módulo BAS
3. Escribe en ese módulo las funciones o procedimientos Sub que quieras "exportar" como parte de la DLL (deben ser Public).
4. Crea un fichero con el mismo nombre del proyecto que tenga la extensión **.DEF**.
5. La estructura (o formato) de ese fichero es el mostrado en el listado 2.
6. Junto a **NAME** escribe el nombre del proyecto (sin extensión).
7. Junto a **LIBRARY** escribe el nombre que quieras (debe ser un nombre válido, no uses espacios, etc.).
8. En **DESCRIPTION** escribe la descripción de la librería (dentro de comillas dobles).

9. En **EXPORTS** debes indicar los nombres de las funciones o Sub que has creado, cada una de esas funciones o Sub deben llevar seguida @ y un número correlativo a partir del número 1.
10. Guárdalo y ya puedes compilar el proyecto creado como DLL.

```
NAME el_nombre_del_proyecto
LIBRARY el_nombre_que_quieras
DESCRIPTION "La descripción que quieras usar"
EXPORTS Nombre_Función_o_Sub @1
        Otro_nombre_Función_o_Sub @2
```

Listado 2. El formato del fichero .DEF

Paso 4. Usar la DLL recién creada desde otro proyecto de VB6

Ahora vamos a usar desde el propio VB6 la DLL que acabamos de crear.

1. Crea un nuevo proyecto de tipo EXE estándar
2. Escribe los Declare para acceder a las funciones como si de una DLL de Windows se tratara
3. Por ejemplo, si tenemos una DLL llamada Prueba.dll en la que hemos definido una función llamada Saludo que devuelve un String, la debes definir de esta forma:
    - Private Declare Function Saludo Lib "Prueba.dll" () As String
4. La librería Prueba.dll debe estar en el mismo directorio que el ejecutable de este proyecto.
5. Escribe un código que use esa función de la misma forma que lo harías con cualquier otra función, por ejemplo:
    - MsgBox Saludo
6. Al ejecutarlo, verás que se muestra lo que la función Saludo devuelva, (ahora veremos un ejemplo real).
7. En realidad las cadenas debemos "tratarlas" un poco antes de usarlas.

Paso 5. Usar la DLL desde una aplicación de C++

Esa DLL creada con VB6 la podemos usar también desde otros lenguajes, por ejemplo C++, aunque en C++ las funciones que reciban y/o devuelvan cadenas no funcionarán, solo las que trabajen con números. En otros lenguajes no lo he probado.

Por ejemplo, para usar una función llamada Suma que está en la librería Prueba.dll y que recibe un parámetro de tipo Double y devuelve un valor también Double, lo tendríamos que usar como se muestra en el listado 3.

```
//--------------------------------------------------------------------------
// Prueba en C/C++ para usar la DLL Prueba.dll creada con VB6       (25/Ene/06)
//
// No podemos usar las funciones que trabajen con cadenas
// (o al menos yo no se usarlas)
//
// Para compilar con el compilador gratuito de Borland:
// bcc -WC -P tPrueba_c.cpp
//
// Para compilar con el compilador de C++ de VS2003:
// cl /D "WIN32" /D "_CONSOLE" tPrueba_c.cpp
//
//--------------------------------------------------------------------------

// Los include a usar
#include <windows.h>
#include <stdio.h>

// Prototipos de las funciones de la DLL
typedef double (CALLBACK* FARPROC1)(double, double);

void main()
{
    // Handle de la DLL
    HINSTANCE hDLL;
    // Punteros a las funciones
    FARPROC1 Suma;

    // Las variables internas
    double n1, n2;
    LPSTR f;

    // Cargar la DLL
    hDLL = LoadLibrary("Prueba.dll");

    // Si se ha podido cargar
    if( hDLL != NULL ){
        // El nombre de la función
        f = (LPSTR)"Suma";
        // Asignar el puntero a la función de la DLL
        Suma = (FARPROC1)GetProcAddress( hDLL, f);
        // Si no existe esa función, mostrar un error
        if( !Suma ){
```

```
            printf("¡¡¡Error!!!\nAl asignar el puntero de la funcion %s.", f);

        }else{
            n1 = 25.33;
            n2 = 748.125;
            printf("Sumar: %f + %f = %f\n", n1, n2, Suma(n1, n2));

        }

Salida:
        // Librerar la DLL
        FreeLibrary(hDLL);

    }else{
        printf("¡¡¡Error!!!\nNo se encuentra la DLL.");
    }

    return;
}
```

Listado 3. Ejemplo de C++ para usar la DLL de VB6

Veamos ahora esa DLL creada en VB6 y otro proyecto (también en VB6) que use las funciones de la DLL.

Código de ejemplo de una DLL en VB6

Para crear la DLL seguiremos los pasos indicados en Paso 4, que son:
-Crear un nuevo proyecto del tipo DLL ActiveX, (cambia el nombre a Prueba).
-Añade un módulo BAS, (cambia el nombre a MPrueba).
-En ese módulo pega el código del listado 4.
-Con un editor de textos crea un fichero llamado Prueba.DEF y añade el texto mostrado en el listado 5.
-El fichero .DEF debe estar en el mismo directorio del proyecto Prueba.vbp.
-Compila el proyecto, si todo ha ido bien, tendrás un fichero llamado Prueba.dll
-Esta DLL la usaremos en nuestro proyecto de prueba desde VB6 y es la misma DLL usada en el código de ejemplo de C/C++ del listado 3.

```
'-------------------------------------------------------------------------------
' DLL de prueba para crearla desde VB6                              (25/Ene/06)
' Esta DLL se usará como una "normal" no de ActiveX DLL
'
' ©Guillermo 'guille' Som, 2006
'-------------------------------------------------------------------------------
```

```
Option Explicit

Public Function Saludo() As String
    Saludo = "Hola desde la DLL Prueba"
End Function

Public Function Suma(ByVal n1 As Double, ByVal n2 As Double) As Double
    Suma = n1 + n2
End Function
```

Listado 4. El código de la DLL a crear con VB6

```
NAME Prueba
LIBRARY elGuille
DESCRIPTION "Prueba de DLL creada con VB6"
EXPORTS Saludo @1
        Suma @2
```

Listado 5. El contenido del fichero Prueba.def

Y ya tenemos nuestra DLL creada, ahora vamos a escribir un proyecto en VB6 para usar esa DLL, lo haremos de la misma forma que usaríamos cualquier DLL "normal" de Windows.

Código del proyecto de prueba hecho con VB6

- Crea un nuevo proyecto de tipo EXE estándar.
- En el formulario añade 2 botones, tres etiquetas y tres cajas de textos para que tenga el aspecto de la figura 2.
- El botón de Saludar se llama cmdSaludo
- El botón Sumar se llama cmdSumar
- Las cajas de texto tendrán los nombres: txtNum1, txtNum2 y txtRes (que es la caja de abajo)
- En el formulario añade el código del listado 6
- Pulsa F5 y verás que todo funciona bien, siempre que la DLL esté en el mismo directorio que este proyecto o bien, la DLL puede estar en cualquier directorio incluido en el PATH del sistema, por ejemplo en el directorio de Windows.

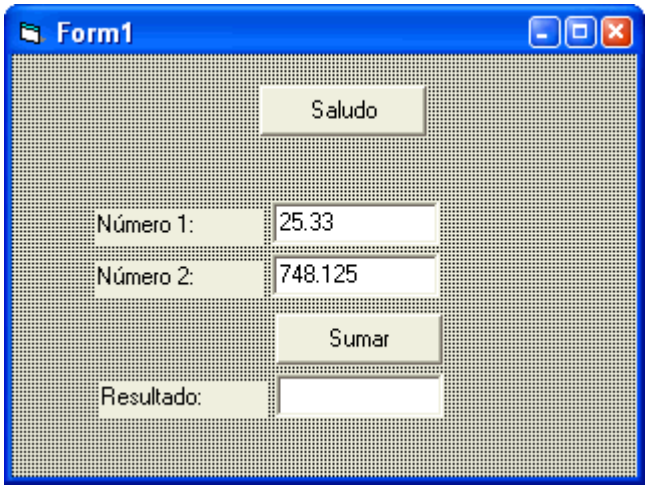

Figura 2. El formulario de prueba de la la librería

```
'--------------------------------------------------------------------------------
' Proyecto de prueba de la librería Prueba.dll creada con VB6       (25/Ene/06)
'
' ©Guillermo 'guille' Som, 2006
'--------------------------------------------------------------------------------

Option Explicit

' Las declaraciones de las funciones de la DLL
Private Declare Function Saludo Lib "Prueba.dll" () As String
Private Declare Function Suma Lib "Prueba.dll" (ByVal n1 As Double, ByVal n2 As Double) As Double

Private Sub cmdSaludo_Click()
    Dim s As String

    s = Saludo

    ' Las cadenas de VB6 son Unicode y al usarla desde una DLL
    ' se hace un follón... así que debemos quitarles los Chr$(0)
    ' que tenga en medio
    s = MTrim(s)

    MsgBox s
End Sub

Private Sub cmdSumar_Click()
    Dim n1 As Double
    Dim n2 As Double
```

```
    n1 = Val(txtNum1.Text)
    n2 = Val(txtNum2.Text)

    txtRes.Text = Suma(n1, n2)
End Sub

Private Function MTrim(ByVal s As String) As String
    Dim i As Long
    Dim res As String

    For i = 1 To Len(s)
        If Mid$(s, i, 1) <> Chr$(0) Then
            res = res & Mid$(s, i, 1)
        End If
    Next

    MTrim = res
End Function
```

Listado 6. El código del formulario de prueba

Fíjate todo el *follón* que hay que hay que hacer para usar la función Saludo que hemos definido en la DLL.
Esto es necesario porque el VB6 maneja cadenas Unicode y cuando esas cadenas provienen de una DLL hace una conversión al formato "normal" de C, y como las cadenas de Unicode son de doble bytes, pues el segundo será un Chr$(0), por eso debemos usar una función que elimine esos caracteres, si no lo hiciéramos, el VB detectaría que la cadena termina en el primero Chr$(0) que se encuentre, por tanto, si no hacemos esa "limpieza" solo veríamos la primera letra de la cadena devuelta.

Usar la DLL creada con VB6 desde .NET (VB y C#)

Sin embargo, si esta misma DLL la usamos desde Visual Basic .NET o desde C#, las cadenas se manejarán bien, aunque debemos indicar en la declaración de la DLL que esa cadena debe tratarse como una cadena de Unicode.
Esto se consigue por medio del atributo DllImport y la propiedad CharSet:

```
<DllImport("Prueba.dll", CharSet := CharSet.Unicode)> _
Private Shared Function Saludo() As String
End Function
```

Para hacer el mismo ejemplo que la prueba de VB6, aunque usando la consola, el código de VB.NET (válido también para VB2005, realmente para cualquier versión de Visual Basic para .NET) sería el mostrado en el listado 7.
Y en el listado 8 está el código de C#.

```
'-------------------------------------------------------------------------
' Prueba de uso de la DLL de VB6 desde VB.NET                    (25/Ene/06)
'
' ©Guillermo 'guille' Som, 2006
'
' Para crear un EXE:
' vbc /out:tPrueba_vb.exe tPrueba_vb.vb /r:System.dll
'-------------------------------------------------------------------------
Option Strict On
Option Explicit On

Imports System
Imports Microsoft.VisualBasic
Imports System.Reflection
Imports System.Runtime.InteropServices

<Assembly: AssemblyTitle("Prueba acceso a DLL de VB6")>
<Assembly: AssemblyDescription("Prueba de acceso a funciones de una DLL creada con VB6")>
<Assembly: AssemblyCompany("elGuille -Nerja")>
<Assembly: AssemblyProduct("revisión del 25/Ene/2006")>
<Assembly: AssemblyCopyright("©Guillermo 'guille' Som, 2006")>
<Assembly: AssemblyTrademark("Microsoft Visual Basic 2003 (.NET 1.1)")>
<Assembly: CLSCompliant(True)>
' Versión
<Assembly: AssemblyVersion("1.0.0.0")>

Public Class Prueba
    '
    <DllImport("Prueba.dll")> _
    Private Shared Function Suma(n1 As Double, n2 As Double) As Double
    End Function

    ' Si indicamos el Charset Unicode, manejará bien la cadena
    ' sin "historias" raras como en VB6
    <DllImport("Prueba.dll", CharSet := CharSet.Unicode)> _
    Private Shared Function Saludo() As String
    End Function


    <STAThread> _
    Shared Sub Main()
        Dim n1, n2 As Double
        n1 = 23.45
```

```
        n2 = 748.125
        Dim n3 As Double = Suma(n1, n2)
        Console.WriteLine("La suma de {0} + {1} = {2}", n1, n2, n3)

        Dim s As String = Saludo()
        Console.WriteLine("s.Length = {0}", s.Length)
        Console.WriteLine("'{0}'", s)

        Console.Write("{0}Pulsa INTRO para terminar ", vbCrLf)
        Console.ReadLine()
    End Sub
End Class
```

Listado 7. Prueba de la DLL creada con VB6 desde Visual Basic .NET

```
//---------------------------------------------------------------------------
// Prueba de uso de la DLL de VB6 desde C#                          (25/Ene/06)
//
// ©Guillermo 'guille' Som, 2006
//
// Para crear un EXE:
// csc /out:tPrueba_cs.exe tPrueba_cs.cs /r:System.dll
//---------------------------------------------------------------------------

using System;
using System.Reflection;
using System.Runtime.InteropServices;

[assembly: AssemblyTitle("Prueba acceso a DLL de VB6")]
[assembly: AssemblyDescription("Prueba de acceso a funciones de una DLL creada con VB6")]
[assembly: AssemblyCompany("elGuille -Nerja")]
[assembly: AssemblyProduct("revisión del 25/Ene/2006")]
[assembly: AssemblyCopyright("©Guillermo 'guille' Som, 2006")]
[assembly: AssemblyTrademark("Microsoft Visual C# 2003 (.NET 1.1)")]
[assembly: CLSCompliant(true)]
// Versión
[assembly: AssemblyVersion("1.0.0.0")]

public class Prueba{
    //
    [DllImport("Prueba.dll")]
    private static extern double Suma(double n1, double n2);

    // Si indicamos el Charset Unicode, manejará bien la cadena
    // sin "historias" raras como en VB6
    [DllImport("Prueba.dll", CharSet = CharSet.Unicode)]
```

```
    private static extern string Saludo();


    [STAThread]
    static void Main()
    {
        double n1, n2;
        n1 = 23.45;
        n2 = 748.125;
        double n3 = Suma(n1, n2);
        Console.WriteLine("La suma de {0} + {1} = {2}", n1, n2, n3);

        string s = Saludo();
        Console.WriteLine("s.Length = {0}", s.Length);
        Console.WriteLine("'{0}'", s);

        Console.Write("\nPulsa INTRO para terminar ");
        Console.ReadLine();
    }
}
```

Listado 8. Prueba de la DLL creada con VB6 desde Visual C#

Y esto es todo... creo que está claro, si no lo tienes claro... ¡enjuágate los ojos! que seguro que estás dormido...

> Ah, y NO me hago responsable de las "trastadas" que puedas hacer al usar esto que te he explicado, así que... si haces esto que te he dicho es EXCLUSIVAMENTE BAJO TU RESPONSABILIDAD, por tanto, las quejas... ¡al maestro armero! que decían en la mili.

De cualquier forma, espero que te sea de utilidad, al menos para "fardar" con los colegas de otros lenguajes que alguna vez te dijeron que el VB6 no puede hacer DLL normales.

Nos vemos.
Guillermo

El código completo

Tanto de la DLL como de los proyectos de prueba de VB6 y C++, y el del LINK.VBP

- Los proyectos de prueba: **crear_DLL_con_VB6.zip** 28.1 KB
    - (**MD5 checksum**: 26C19F581BF6C3C07CE2E9064D137F8E)

- El proyecto para sustituir LINK.EXE: **Link_para_vb6.zip** 6.65 KB
    - (MD5 checksum: F2352C10BC202384C9419AE0A0269CB3)

- Los ficheros de prueba para .NET (VB y C#): **DLL_vb6_NET.zip** 1.73 KB
    - (MD5 checksum: 9FD53268425E914CA6A2551E912DAD16)

Volver a Visual Basic Avanzado