DynaWrap - DynaCall Wrapper

Soumis par Gilles LAURENT
27-06-2008
Dernière mise à jour : 23-09-2008

DynaWrap - DynaCall Wrapper

Le composant COM DynaWrap disponible en téléchargement sur le site de Günter Born donne la possibilité de déclarer et d'appeler de manière dynamique des API Win32 à partir d'un langage supportant automation tel que VBScript. Bien que très utile, ce composant se limite toutefois à l'appel d'API simples ne nécessitant ni pointeur ni structure.

Très interressé par ce composant, j'ai donc décidé d'étendre ses capacités en ajoutant quelques méthodes et constantes permettant de combler la plupart de ces limitations. Une librairie de types a également été intégrée dans le but d'obtenir les membres (méthodes, propriétés et constantes) avec par exemple les méthodes natives Shell.GetMembers (alias gm) et Shell.GetConstants (alias gc) disponibles avec la console WSH Shell.Credits:
This is an enhanced version of the original version as found on Günter Born Web Site. So many thanks to Günter Born, Ton Plooy, Jeff Stong, William Epp and Michael Hines.

Historique:
Version 1.0.0.1 - 20080625 - Première Release Publique

Plateformes supportées:
Windows 2000+

Téléchargement:
Guide PDF: -
Archive: DynaWrap.zip

Note : Tous les exemples ci-dessous ont été réalisés en ligne de commandes avec la console WSH Shell dans le but de visualiser de manière interactive le résultat des instructions saisies. Plusieurs alias builtin sont utilisés.

Note : Vous trouverez en suivant ce lien une page contenant quelques exemples d'utilisation de ce composant dans divers langages (PHP, Perl, ...). Cette page sera régulièrement enrichie en fonction des demandes. De ce fait, n'hésitez pas à me contacter via le site pour améliorer ce tutorial.

Ci-dessous un exemple VBScript d'instanciation du composant COM "DynamicWrapper" et la détermination de ses membres. Bien entendu, le composant COM doit avoir été préalablement enregistré dans le registre via l'outil regsvr32.exe

```
WSH D:\Test> Set oDyn=co("DynamicWrapper")
WSH D:\Test> gm(oDyn)

Category  Name
--------  ----
Function  FormatMessage (dwMessageId)
Function  GetBSTRAddr (String)
Function  GetLastError ()
Function  GetMemInBSTRAddr (dwBSTRAddr, dwOffset, nDataType)
Function  GetRegistered ()
Function  GetVariantAddr (Var)
Function  Register (DllName, Win32Api, [TypeOfCall], [ReturnDT], [InputDT])
Function  SetMemInBSTRAddr (dwBSTRAddr, dwOffset, nDataType, Data)
Property  Version

WSH D:\Test> gc(oDyn)

Constant   Value
--------   -----
dw_BSTR    0
dw_BYTE    1
dw_DOUBLE  8
dw_INT     2
dw_LONG    4
```

Description des méthodes et propriétés

Function FormatMessage (dwMessageId)
Cette méthode retourne le message d'erreur d'une fonction Win32 associé au code numérique dwMessageId passé comme argument. Le message d'erreur est affiché en fonction de la locale en cours.

```
WSH D:\Test> for i=0 to 10 {
  >> echo Right("00" & i, 2) & " " & Split(oDyn.FormatMessage(i), VBCrLf)(0)
  >> }
  >>
00 Opération réussie.
01 Fonction incorrecte.
02 Le fichier spécifié est introuvable.
03 Le chemin d'accès spécifié est introuvable.
04 Le système ne peut pas ouvrir le fichier.
05 Accès refusé.
06 Descripteur non valide
07 Les blocs de contrôle de mémoire ont été détruits.
08 Espace insuffisant pour traiter cette commande.
09 Adresse de bloc de contrôle de stockage non valide.
10 Environnement incorrect.
```

Function GetBSTRAddr (String)
Cette méthode retourne l'adresse (pointeur) de la chaîne de caractères de type BSTR contenue dans une variable Variant de type String. Dans le cas d'une utilisation par référence, le Variant de type String doit avoir été préalablement alloué. Vous trouverez dans les exemples ci-dessous quelques références quant à son utilisation.

Function GetLastError ()
Cette méthode retourne le code d'erreur généré lors du dernier appel d'une API Win32. Cette méthode est particulièrement utile pour obtenir le code d'erreur étendu d'une fonction. Le message d'erreur correspondant pourra être obtenu à l'aide de la méthode FormatMessage.

```
WSH D:\Test> ' AbortSystemShutdown Function
WSH D:\Test> ' The AbortSystemShutdown function stops a system shutdown started
WSH D:\Test> ' by using the InitiateSystemShutdown function
WSH D:\Test> ' BOOL AbortSystemShutdown (LPTSTR lpMachineName)
WSH D:\Test>
WSH D:\Test> ' Parameters :
WSH D:\Test> ' lpMachineName
WSH D:\Test> ' [in] Pointer to the null-terminated string that specifies the
WSH D:\Test> ' network name of the computer where the shutdown is to be stopped
WSH D:\Test> ' If lpMachineName is NULL or an empty string, the function stops
WSH D:\Test> ' the shutdown on the local computer
WSH D:\Test>
WSH D:\Test> ' Return Values :
WSH D:\Test> ' If the function succeeds, the return value is nonzero
WSH D:\Test> ' If the fonction fails, the return value is zero. To get extended
WSH D:\Test> ' error information, call GetLastError
WSH D:\Test>
WSH D:\Test> ' déclaration de l'API Win32
WSH D:\Test> oDyn.Register "advapi32.dll", "AbortSystemShutdownW", "r=b", "i=w"
WSH D:\Test>
WSH D:\Test> ' tentative d'arrêt du redémarrage de la machine distante DEVW2K
WSH D:\Test> echo oDyn.AbortSystemShutdownW("DEVW2K")
0
WSH D:\Test> ' la fonction a retourné une erreur (0)
WSH D:\Test> ' détermination du code d'erreur étendu
WSH D:\Test> echo oDyn.GetLastError()
1116
WSH D:\Test> ' affichage du message d'erreur correspondant au code 1116
WSH D:\Test> echo oDyn.FormatMessage(1116)
Le système n'étant pas en cours d'arrêt, il est impossible d'annuler l'arrêt ...
```

Function GetMemInBSTRAddr (dwBSTRAddr, dwOffset, nDataType)
Cette méthode retourne une valeur de type nDataType à l'offset dwOffset contenue dans une chaine de caractères de type BSTR dont l'adresse dwBSTRAddr est passée comme argument. Cette méthode est principalement utile pour extraire des valeurs présentes dans une structure.

```
WSH D:\Test> ' GetProfileSection Function
WSH D:\Test> ' Retrieves all the keys and values for the specified section of
WSH D:\Test> ' the Win.ini file
WSH D:\Test> ' DWORD WINAPI GetProfileSection (
WSH D:\Test> '    LPCTSTR lpAppName,
WSH D:\Test> '    LPTSTR  lpReturnedString,
WSH D:\Test> '    DWORD   nSize
WSH D:\Test> ' )
WSH D:\Test>
WSH D:\Test> ' Parameters :
WSH D:\Test> ' lpAppName
WSH D:\Test> ' [in] The name of the section in the Win.ini file
WSH D:\Test> '
WSH D:\Test> ' lpReturnedString
WSH D:\Test> ' [out] A pointer to a buffer that receives the keys and values
WSH D:\Test> ' associated with the named section. The buffer is filled with one
WSH D:\Test> ' or more null-terminated strings; la last string is followed by a
WSH D:\Test> ' second null character
WSH D:\Test> '
WSH D:\Test> ' nSize
WSH D:\Test> ' [in] The size of the buffer pointed to by the lpReturnedString
WSH D:\Test> ' parameter, in characters. The maximum profile section size is
WSH D:\Test> ' 32,767 characters
WSH D:\Test>
WSH D:\Test> ' Return Value :
WSH D:\Test> ' The return value specifies the number of characters copied to
WSH D:\Test> ' the specified buffer, not including the terminating null
WSH D:\Test> ' character. If the buffer is not large enough to contain all the
WSH D:\Test> ' keys and values associated with the named section, the return
WSH D:\Test> ' value is equal to the size specified by nSize minus two
WSH D:\Test>
WSH D:\Test> ' déclaration de l'API Win32
WSH D:\Test> oDyn.Register "kernel32.dll","GetProfileSection","r=l","i=sll"
WSH D:\Test>
WSH D:\Test> ' allocation du buffer contenant les couples (clé, valeur)
WSH D:\Test> sReturnedString=String(1024, Chr(0))
WSH D:\Test>
WSH D:\Test> ' détermination de l'adresse (pointeur) du buffer
WSH D:\Test> dwBSTRAddr=oDyn.GetBSTRAddr(sReturnedString)
WSH D:\Test>
WSH D:\Test> ' lecture de la section Mail du fichier de configuration Win.ini
WSH D:\Test> echo oDyn.GetProfileSection("Mail", dwBSTRAddr, 1024)
97
WSH D:\Test> ' le buffer contient 97 caractères
WSH D:\Test> ' lecture des couples (clé, valeur) à l'aide de la fonction
WSH D:\Test> ' builtin GetMemInBSTRAddr. Celle-ci va permettre d'extraire les
WSH D:\Test> ' chaines de caractères présentes dans le buffer
WSH D:\Test> nOffset=0
WSH D:\Test> Do {
  >> sKeyVal=oDyn.GetMemInBSTRAddr(dwBSTRAddr, nOffset, dw_BSTR)
  >> If sKeyVal<>"" Then
  >>   echo sKeyVal
  >>   nOffset=nOffset+Len(sKeyVal)+1
  >> End If
  >> Loop Until sKeyVal=""
  >>
MAPI=1
CMCDLLNAME32=mapi32.dll
CMCDLLNAME=mapi.dll
CMC=1
MAPIX=1
```

MAPIXVER=1.0.0.1
OLEMessaging=1

Function GetRegistered ()
Cette méthode retourne la liste des API Win32 déclarées via la méthode Register. La liste des méthodes déclarées est retournée dans une variable de type Array.

```
WSH D:\Test> arrAPI=oDyn.GetRegistered()
WSH D:\Test> echo UBound(arrAPI)
6
WSH D:\Test> echo $(arrAPI)
NetWkstaGetInfo
NetUnjoinDomain
NetJoinDomain
NetGetJoinInformation
NetApiBufferFree
InitiateSystemShutdownExW
AbortSystemShutdownW
```

Function GetVariantAddr (Var)
Cette méthode retourne l'adresse (pointeur) de la variable Variant Var. Pour des raisons de structure interne, cette méthode ne doit être utilisée que pour les variables Variant qui ne sont pas de type String. Pour les variables Variant de type String il est nécessaire d'utiliser la méthode dédiée GetBSTRAddr.

```
WSH D:\Test> ' GetWindowThreadProcessId Function
WSH D:\Test> ' The GetWindowThreadProcessId function retrieves the identifier
WSH D:\Test> ' of the thread that created the specified window and, optionally,
WSH D:\Test> ' the identifier of the process that created the window
WSH D:\Test> ' DWORD GetWindowThreadProcessId(HWND hWnd, LPDWORD lpdwProcessId)
WSH D:\Test>
WSH D:\Test> ' Parameters :
WSH D:\Test> ' hWnd
WSH D:\Test> ' [in] Handle to the window
WSH D:\Test> '
WSH D:\Test> ' lpdwProcessId
WSH D:\Test> ' [out] Pointer to a variable that receives the process identifier
WSH D:\Test> ' If this parameter is not NULL, GetWindowThreadProcessId copies
WSH D:\Test> ' the identifier of the process to the variable; otherwise, it
WSH D:\Test> ' does not
WSH D:\Test>
WSH D:\Test> ' Return Value :
WSH D:\Test> ' The return value is the identifier of the thread that created
WSH D:\Test> ' the window
WSH D:\Test>
WSH D:\Test> ' déclaration de l'API Win32
WSH D:\Test> oDyn.Register "user32.dll","GetWindowThreadProcessId","r=l","i=hl"
WSH D:\Test>
WSH D:\Test> ' allocation d'une variable de type Long
WSH D:\Test> dwProcessId=CLng(0)
WSH D:\Test>
WSH D:\Test> ' détermination du PID de la console WSH Shell
WSH D:\Test> echo oDyn.GetWindowThreadProcessId(hWnd, oDyn.GetVariantAddr(dwProc
essId))
2084
WSH D:\Test> ' Le Thread identifier = 2084
WSH D:\Test> ' Lecture du PID
WSH D:\Test> ' Le Variant dwProcessId a été transmis par référence (pointeur)
WSH D:\Test> echo dwProcessId
760
```

Function Register (DllName, Win32Api, [TypeOfCall], [ReturnDT], [InputDT])

Cette méthode permet de déclarer dynamiquement une API Win32 Win32Api disponible au sein de la librairie dynamique DllName. Les paramètres optionnels TypeOfCall, ReturnDT et InputDT permettent de spécifier le type d'appel, le type de la valeur de retour ainsi que les arguments nécessaires à l'exécution de l'API Win32. Le type de valeur BOOL (b) est maintenant supporté.

Code	Variant	Description
a	VT_DISPATCH	IDispatch*
b	VT_BOOL	BOOL
c	VT_I4	unsigned char
d	VT_R8	8 byte real
f	VT_R4	4 byte real
h	VT_I4	HANDLE
k	VT_UNKNOWN	IUnknown*
l	VT_I4	LONG
p	VT_PTR	pointer
r	VT_LPSTR	string by reference
s	VT_LPSTR	string
t	VT_I2	SHORT
u	VT_UINT	UINT
w	VT_LPWSTR	wide string

Function SetMemInBSTRAddr (dwBSTRAddr, dwOffset, nDataType, Data)
Cette méthode permet d'écrire la valeur Data de type nDataType à l'offset dwOffset dans une chaine de caractères de type BSTR dont l'adresse dwBSTRAddr est passée comme argument. Cette méthode est principalement utile pour écrire des valeurs initiales dans une structure.

```
WSH D:\Test> ' GetVersionEx Function
WSH D:\Test> ' Retrieves information about the current operating system
WSH D:\Test> ' BOOL WINAPI GetVersionEx (LPOSVERSIONINFO lpVersionInfo)
WSH D:\Test>
WSH D:\Test> ' Parameters :
WSH D:\Test> ' lpVersionInfo
WSH D:\Test> ' [in, out] An OSVERSIONINFO or OSVERSIONINFOEX structure that
WSH D:\Test> ' receives the operating system information. Before calling the
WSH D:\Test> ' GetVersionEx function, set the dwOSVersionInfoSize member of
WSH D:\Test> ' this structure as appropriate
WSH D:\Test>
WSH D:\Test> ' Return Value :
WSH D:\Test> ' If the function succeeds, the return value is a nonzero value
WSH D:\Test> ' If the function fails, the return value is zero. To get extended
WSH D:\Test> ' error information, call GetLastError. The function fails if you
WSH D:\Test> ' specify an invalid value for the dwOSVersionInfoSize member of
WSH D:\Test> ' the OSVERSIONINFO or OSVERSIONINFOEX structure
WSH D:\Test>
WSH D:\Test> ' OSVERSIONINFO Structure :
WSH D:\Test> ' typedef struct _OSVERSIONINFO {
  >> ' DWORD dwOSVersionInfoSize ;
  >> ' DWORD dwMajorVersion ;
  >> ' DWORD dwMinorVersion ;
  >> ' DWORD dwBuildNumber ;
  >> ' DWORD dwPlatformId ;
  >> ' TCHAR szCSDVersion[128] ;
  >> ' } OSVERSIONINFO ;
  >>
WSH D:\Test> ' déclaration de l'API Win32
WSH D:\Test> oDyn.Register "kernel32.dll", "GetVersionEx", "r=b", "i=l"
WSH D:\Test>
WSH D:\Test> ' allocation du buffer représentant la structure OSVERSIONINFO
WSH D:\Test> ' détermination de la taille de la structure à allouer
WSH D:\Test> echo dw_LONG*5+128
148
WSH D:\Test> ' la fonction VBScript String alloue une chaine de caractères de
WSH D:\Test> ' type Unicode (un caractère est codé sur deux octets). La taille
WSH D:\Test> ' transmise comme argument à cette fonction sera donc divisée par
```

```
WSH D:\Test> ' deux :
WSH D:\Test> sVersionInfo=String((dw_LONG*5+128)/2, Chr(0))
WSH D:\Test> echo Lenb(sVersionInfo)
148
WSH D:\Test> ' détermination de l'adresse (pointeur) du buffer
WSH D:\Test> dwBSTRAddr=oDyn.GetBSTRAddr(sVersionInfo)
WSH D:\Test>
WSH D:\Test> ' il est nécessaire d'initialiser le membre dwOSVersionInfoSize de
WSH D:\Test> ' la structure OSVERSIONINFO avant d'appeler la fonction Win32
WSH D:\Test> ' GetVersionEx. Cette opération est réalisée avec la fonction
WSH D:\Test> ' builtin SetMemInBSTRAddr. Le membre dwOSVersionInfoSize de type
WSH D:\Test> ' DWORD (dw_LONG) étant situé en première position, sa position
WSH D:\Test> ' relative (offset) est donc zéro
WSH D:\Test> oDyn.SetMemInBSTRAddr dwBSTRAddr, 0, dw_LONG, dw_LONG*5+128
WSH D:\Test>
WSH D:\Test> ' récupération des informations sur la version Windows
WSH D:\Test> echo oDyn.GetVersionEx(dwBSTRAddr)
-1
WSH D:\Test> ' le code de retour -1 (True) indique que la fonction a réussi
WSH D:\Test> ' lecture des membres de la structure OSVERSIONINFO à l'aide de la
WSH D:\Test> ' fonction builtin GetMemInBSTRAddr. Celle-ci va permettre de lire
WSH D:\Test> ' les membres de la structure OSVERSIONINFO. L'offset transmis
WSH D:\Test> ' comme argument doit impérativement respecter le typage
WSH D:\Test>
WSH D:\Test> ' major version
WSH D:\Test> echo oDyn.GetMemInBSTRAddr(dwBSTRAddr,4,dw_LONG)
5
WSH D:\Test> ' minor version
WSH D:\Test> echo oDyn.GetMemInBSTRAddr(dwBSTRAddr,8,dw_LONG)
1
WSH D:\Test> ' build number
WSH D:\Test> echo oDyn.GetMemInBSTRAddr(dwBSTRAddr,12,dw_LONG)
2600
WSH D:\Test> ' platformId
WSH D:\Test> echo oDyn.GetMemInBSTRAddr(dwBSTRAddr,16,dw_LONG)
2
WSH D:\Test> ' service pack
WSH D:\Test> echo oDyn.GetMemInBSTRAddr(dwBSTRAddr,20,dw_BSTR)
Service Pack 2
WSH D:\Test>
WSH D:\Test> ' donc pour résumer :
WSH D:\Test> ' Microsoft Windows XP [version 5.1.2600] Service Pack 2 !
```

Property Version
Cette propriété en lecture seule permet de déterminer la version du composant COM DynaWrap.dll chargé en mémoire. La version est de la forme w.x.y.z.

Description des constantes

Les constantes définies par le composant COM permettent de spécifier le type nDataType pour les méthodes GetMemInBSTRAddr et SetMemInBSTRAddr. Ces constantes sont définies dans la librairie de types et sont automatiquement disponibles lors de l'instanciation de l'objet COM via la méthode native Shell.CreateObjectWithConstants (alias co) de la console WSH Shell.

dw_BSTR (0) permet de lire ou d'écrire un Variant de type String
dw_BYTE (1) permet de lire ou d'écrire un Variant de type Byte
dw_DOUBLE (8) permet de lire ou d'écrire un Variant de type Double
dw_INT (2) permet de lire ou d'écrire un Variant de type Integer
dw_LONG (4) permet de lire ou d'écrire un Variant de type Long