بسم الله الرحمان الرحيم

مفاهيم حول :

واجهة برمجة التطبيقات (" Win32 API ")
بالفيجوال بيسيك 6.0

مقدمة:

في بعض من الأحيان تود كتابة إجراء لمهمة ما ويتعذر لك ذالك لسبب ما كعدم دعم الفيجوال بيسيك للطرق "Methods" التي تتعامل مع هذه المهمة أو كون الإجراء جد معقد يحتمل وقوع أخطاء فتكون إجراءات "API" هي ملجأك الأخير.

ماهية إجراءات "API" ؟:

"API" هي اختصار لجملة (Application Programming Interface) أي تلك الطرق (إجراءات ودوال) الموزعة على عدد كبير من مكتبات الربط الديناميكي امتداد "DLL" لنظام التشغيل فهي أساس و روح النظام إن صح التعبير لماذا ؟، لأنها المسئولة على كل أوامر النظام والبرامج وأجهزة الإدخال والإخراج وغير ذالك الكثير؛ فبعد توغلك في هذه الإجراءات ستكتشف مدى قوتها و سحرها، ودعمها للبيسيك تزيد من قدرته و كفاءته.

ميزات إجراءات "API":

– البرمجة على النظام

المقصود هنا هو التعامل مع أوامر النظام كالتحكم على سبيل المثال في أجهزة الإدخال و الإخراج ،وهذا هو الغرض الأسمى من الولوج في هده الإجراءات.

– سرعة الشفرة أو الكود

مهما كان قصر طول الشفرة (الكود) الذي تعتمد عليه في إنجاز غرض ما سريعا في التنفيذ فلن يكون أسرع من إجراءات "API" فما بالك إذا كان طويل**!**؛ هذا إذا كانت السرعة تعني لك الشّيء الكبير.

– تخفيف البرنامج

من ميزات استمالها تخفيف البرنامج أو تخفيف حجمه، فبدل من كتابة أسطر كثيرة لإنشاء مهمة ما، استبدِلها بسطر أو سطرين من الإجراءات لتقليص حجم البرنامج وهذا أولى وأحرى من الاعتماد على مكونات "ActiveX" للإلمام بهذه الميزة .

ملاحظة: المراد بكلمة إجراءات في هذا المقال هو كل الدوال "functions" و الإجراءات "sub's". ☺

مصطلحات:

الشكل العام للإجراء أو الدالة:

[Private | Public] Declare Function | Sub *publicname* **Lib** "*libname*" **[Alias** " *aliasname*"**]** **[([[ByVal]** *variable* **[As** *type***]** **[,[ByRef]** *variable* **[As** *type***]]...])] [As** Type**]**

–الكلمة المفتاح " Declare " تشير إلى أن الدالة أو الإجراء خارجي ليس من عناصر اللغة. وهي إلزامية.

– **Lib** " اختصار لكلمة "Library" أي مكتبة وهي إلزامية لتمييز المكتبة التابع لها الإجراء.

–"*libname*" اسم المكتبة المشار إليها سابقً على سبيل المثال "User32" فهي لا تحتاج لتعريف مسار وصولها كـــ "C:\windows\system\User32.dll" و لا حتى لامتداد ملفها "dll" طالما أنها في مجلد النظام ؛ على العكس إذا لم تكن في هذا المجلد فالاسم الكامل لا مناص منه.

–**"Alias**" تسمح هذه الأخيرة بإعلان الإجراء بكل تغيرات اِسمه ، أي أنها دلالة على الاسم الذي يليها هو الاسم الأصلي للإجراء.

–"*aliasname*" إذن هو الاسم الأصلي للإجراء وهو حساس لحالة الحرف (الحروف الكبيرة و الحروف الصغيرة).

–"**As Any** " أي أن المتغير أو "البارامتر" من نوع غير ثابت ، وفي هذه الحالة الإرسال يكون دائما بالمرجع لإرسال قيمة خالية أو مؤشر "Parameter " لـــ "null" من نوع الصحيح الطويل "long" تستعمل الكلمة المحجوزة "ByVal" تليها القيمة &0.

-" **Private** " ، "**Public**" ، " **Function** " ، " **Sub** " لا أظن قارئ هذه المقال بحاجة لمعرفة معاني هذه العبارات لأنها نفس أوامر اللغة ما عليك التركيز إلا في القيم المرسلة أو الممررة "Parameters".

-"**ByVal** " ،"**ByRef**" أيضا غنية عن التعريف فالأولى تنسخ قيمة المتغير وتضعها في مكان مؤقت في الذاكرة ومن ثمة ترسلها كمؤشر للدالة وهنا إذا غيرة الدالة المؤشر أو قيمة المؤشر فلن يكون التغيير على مستوى المتغير الأصلي بل على النسخة فقط ، عكس الثانية "**ByRef**". إذا لم تذكَر كلاهما فيكون الإرسال افتراضيا بالمرجع أي إرسال مؤشر المتغير أو عنوان المتغير بالذاكرة.

-"**AddressOf**" من أوامر لغة البيسك ، يغنيك هذا المعامل من استعمال مكونات أو مكتبات خاصة لتمرير عنوان إجراء ما إلى دالة "API".

> ملاحظة:
>
> استعن بالبرنامج المدرج مع الفيجوال بيسك "API Viewer" أحسن بكثير من أن تتكبد عناء كتابة الإجراءات بنفسك والأحسن من ذالك تفادي أخطاء الكتابة، لأن بيئة 32 بت تميز بين الحروف الكبيرة و الصغيرة في أسماء الإجراءات.

القيم المرسلة "Parameters":

– السلاسل الحرفية:

تستقبل الدوال السلاسل الحرفية بالقيمة عموما "**ByVal** " وهذه السلسلة يجب أن تكون ذات سعة كافية لتخزين وإرجاع المعطيات المطلوبة من الدالة ، قبل عمل بُعد أو سعة حرفية بدوال اللغة كـــ "*space$*" أو "*string$*" يجب مراعاة طول هذه السعة وإلا قد لارتجع الدالة أي قيمة أو قد تفقد بعضها ؛ (طبعًا هذا في حالة نجاح الدالة).

إذا أردت إرسال مؤشر لقيمة خالية استعمل الثابت "*vbNullString*"، أما السلاسل الحرفية الثابتة الطول "Fixed-length" فلا ترسل إلى الدوال لأنها أي الدوال ترجع القيم الحرفية منتهية إجباريًا بقيمة "*Chr$(0)*"

استعمل التعبير التالي لاستخلاص النتيجة *Left$(ApiRet$, Instr(ApiRet$, Chr$(0))-1)* حيث أن "*ApiRet$*" هي القيمة المُرجعة من الدالة.

– التركيبات "Types":

ترسل المتغيرات من نوع "Type" إلى الإجراءات دائما بالمرجع، وإعلان التركيبات يكون عامًا سواء على مستوى النموذج أو على مستوى الوحدة البرمجية، تستطيع نسخها إلى الحافظة من عارض الدوال المشار إليه في الملاحظة و من ثمة إلصاقها إلى البرنامج.

– الثوابت:

كما تتعامل الإجراءات مع الثوابت سواءً الحرفية أو العددية تجدها هي الأخرى في المستكشف ،و تستطيع استبدالها بقيمها إذا أردت ذالك. – **المقابض** "Handles":

الكثير الكثير من الإجراءات التي تستقبل خاصيتي المقبضين سواءً مقبض النافذة "hWnd" أو مقبض سياق الجهاز "hDC" و هما عبارة عن قيمتين من النوع الطويل "long" يميز بهما نظام التشغيل الرقم الخاص بالأداة " Window Handle" أو الخاص بسياق الجهاز "Device Context" ، و هاذين لا مفر منهما خاصة إذا كنت تتعامل مع النوافذ أو الرسومات.

*ملاحظة: ينصح بإرسال خاصية المقبض مباشرة وتجنب إسناد قيمته لمتغير ومن ثمة إرسال المتغير لأن القيمة تتغير أثناء التنفيذ.

الإعلان:

يكون عام سواءً على مستوى النموذج "frm" أو على مستوى الوحدة البرمجية "bas" أو حتى على الفئة "cls" كدوال وإجراءات اللغة (VB).

الأمثلة:

سنتطرق بإذن الله تعالى إلى أمثلة بسيطة عن استخدام إجراءات تابعة لأهم أو من أهم مكتبات النظام بدءًا بمكتبة "gdi32" لسهولة

فهم واستخدام إجراءاتها.

1.مكتبة "gdi32":

اختصار لجملة "Graphics Device Interface" أو واجهة الجهاز الرسومية ، مكتبة زاخرة جدا بأوامر الرسم وأوامر التعديل و تنسيق الصور والخطوط من تكبير و تصغير وإقلاب اللون و دمج .....تغنيك بكل ما تريده من أوامر الرسم و بكل مرونة خاصة أن لغتنا الجميلة معروفة بعجزها في هذا المجال.

التمرين الأول:

أنشئ مشروع جديد "Standard EXE" ثم من قائمة "Add-In" أنقر على خيار "Add-In Manager" ستظهر لك نافذة بهذا الاسم حدد البرنامج "API Viewer" ثم بالنقر المزدوج عليه تلاحظ كتابة أو ظهور كلمة "Loaded" مقابل اسم البرنامج داخل القائمة، ثم زر "OK" .

سنقوم في هذا التمرين بإنشاء إجراء يرسم لنا شكل رباعي الأضلاع مستدير الزوايا، نبدأ أولا بنسخ الدالة "RoundRect" من عارض الدوال ثم لصقها في القسم العام من النموذج على سبيل المثال:

```
Private Declare Function RoundRect Lib "gdi32" (ByVal hdc As Long , ByVal X1 As Long , _
ByVal Y1 As Long , ByVal X2 As Long , ByVal Y2 As Long , ByVal X3 As Long , ByVal Y3 _
As Long) As Long
```

الدالة تستقبل 7 بارامترات الأول مقبض سياق الرسم للأداة المراد رسم الأشكال عليها، الثاني والثالث إحداثيتي بداية الرسم (البداية تكون بأحد الزوايا الأربعة)، الرابع و الخامس إحداثيتا النهاية أما السادس و السابع يشيران إلى درجة استدارة الزاوية إن صح التعبير.

```
Dim   X1 as long     'متغيرات عامة
Dim   Y1  as  long
```

```
Private Sub Picture1_MouseDown(Button As Integer , Shift As Integer , X As Single , Y As Single)
X1 = X                'حفظ إحداثيتا بداية الرسم في المتغيرات العامة
Y1 = Y
End Sub
```

```
Private Sub Picture1_MouseUp(Button As Integer , Shift As Integer , X As Single , Y As Single)
Picture1.ScaleMode = vbPixels                'تغيير وحدة القياس
RoundRect Picture1.hdc, X1, Y1, X, Y, 30, 30    ' القيمة 30 الأولى و الثانية تشير إلى درجة استدارة الزاوية
End Sub
```

نفذ البرنامج واتجه إلى أداة الصورة بمؤشر الفأرة ثم مع السحب و الإلقاء في أي اتجاه سواءً من أعلى إلى أسفل و من جهة اليسار إلى اليمين أو العكس تلاحظ رسم الشكل المطلوب و هذا يوفر عليك عناء كتابة سطور أخرى للتحقق في كل مرة من اتجاه مرور مؤشر الفأرة .

انتباه:

يجب أن تنتبه إلى وحدة القياس لخاصية "ScaleMode" عدلها دائما بالبكسل سواء وقت التصميم أو وقت التنفيذ لأن إجراءات "API" تتعامل مع البكسل وإلا فلن تتوقع منها نتائج صحيحة. و إذا كنت مصمما على إبقاء الوحدة الافتراضية "Twip" فتستطيع بتقسيم القيم سواء الطول و العرض أو الإحداثيات على الخاصيتين "TwipsPerPixelX " و "TwipsPerPixelY "، كلاً بمقابلهِ ،كما ستلاحظ ذالك في المثال القادم بحول الله.

كما توفر لك (API) إمكانيات هائلة في مجال رسم الأشكال والخطوط كالدالة "Rectangle" لرسم المستطيلات والدالة "Ellipse" لرسم الدائرة و المقطع المكافئ و "LineTo" لرسم الخطوط و"Arc" لرسم الأقواس و غيرها الكثير، من اسمها تعرف وضيفتها، أما طريقة عملها فهي تشبه المثال السابق عموما، فالدالة "Ellipse" على سبيل المثال تطلب منك فقط إحداثيتا نقطة بداية قطر الدائرة و إحداثيتا نقطة نهايته، أليست كالمثال السابق تماما؟. (إذا كنت جديدا على إجراءات "API" تستطيع الاستفادة من المثال المدرج مع هذه المقال باسم المثال الأول).

التمرين الثاني:

لنفترض أنك أردت عمل إجراء يقوم بتعبئة شكل ما بلون ما كالأحمر مثلا فماذا تفعل ؟، خاصةً إذا كان الشكل غير هندسي (غشيم)؟، هنا ستكون الدالة "FloodFill" هي الحل:

الدالة تستقبل أربع قيم: الأول مقبض سياق الأداة، الثاني و الثالث إحداثيتي بداية التعبئة و الأخير لون محيط الشكل المراد تعبئته، أما لون التعبئة فهو نفس اللون المعطى للخاصية "FillColor" للأداة الحاضنة.

سنتابع المثال السابق للاختصار:

```
Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)

Picture1.FillColor = vbRed                                  ' لون
التعبئة
Picture1.ForeColor = vbYellow                               ' لون محيط
الشكل

If Button And vbLeftButton Then
Picture1.FillStyle = vbFSTransparent '                      لا تنسى
هذه الخاصية
'
'Picture1.ScaleMode = vbPixels                  ' السطرين بالسطر القادم لإبقاء الوحدة الحالية
استبدال هاذين
'RoundRect Picture1.hdc, X1, Y1, X, Y, 30, 30

RoundRect Picture1.hdc, X1 / Screen.TwipsPerPixelX, Y1 /
Screen.TwipsPerPixelY _
,X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY  ,30, 30   'رسم
الأشكال

Else
Picture1.FillStyle = vbFSSolid                              'التعبئة
FloodFill Picture1.hdc, X / Screen.TwipsPerPixelX, _
                        Y / Screen.TwipsPerPixelY, vbYellow
End If
End Sub
```

نفذ البرنامج وارسم عدة أشكال بالزر الأيسر للفأرة ثم انقر داخلها بالزر الآخر،

إذا لم تجد الدالة لون الشكل المراد تعبئته أو كانت إحداثيتا بداية التعبئة خارج نطاقه فقد تملئ الدالة كامل الأداة الحاضنة أو تستثني الشكل.

التمرين الثالث:

الآن نسلط الضوء بإذن الله على دالة من أهم الدوال التي تتعامل مع الرسومات . مهمة هذه دالة هي نسخ صورة تابعة لأداة ما و وضعها في أداة أخرى بعد تحجيمها، الدالة هي: "StretchBlt" وهي تستقبل 11 قيمة، لكنها سهلة جدا للاستيعاب ، الخمسة الأولى خاصة بالأداة المراد نقل الصورة إليها أو الهدف، و الخمسة الثانية خاصة بالأداة المراد نقل الصورة منها أي الأداة المصدر، و

القيمة الأخيرة هي ثابت عددي من بين 11 يشير إلى نوع العملية التي ستجرى (تكبير الصورة كما هي أم هناك دمج و قلب للألوان) وهي أشبه بالقيم المسندة للخاصية "DrawMode"

في (VB).الثابت الذي يهمنا هنا هو"SRCCOPY" لنقل الصورة كما هي لأن الغرض الأساسي من الدالة هو التحجيم، إذا كنت تهتم لأمر الثوابت الأخرى أي طريقة رسم الصورة فعليك بالدالة "BitBlt" لأن هذا غرضها الأساسي أو حتى دالة البيسك "PaintPicture"

فهي على الأقل لا تطلب منك مقبض سياق الأداة سواءً المصدر أو الهدف فتستطيع إذن أن تنقل الصورة من و إلى أداة الصور "image".

نرجع إلى دالة التكبير أو التحجيم و لنبدأ بالقيم الخمسة الأولى:

"hDC" مقبض سياق الأداة الهدف

"X"،"Y" ، "nWidth"،"nHeight" تشير إلى أبعاد الصورة المراد رسمها .

قيم الأداة المصدر:

"hSrcDC" مقبض سياق الأداة المصدر أي التي تحمل الصورة المراد النقل منها.

"XSrc"،"YSrc"،"nSrcWidth"،"nSrcHeight" تشير إلى أبعاد الصورة المصدرية.

المثال:

ضع على النموذج "form" أداة صورة باسم "PctDestination"و لتكن هي الأداة المراد الرسم عليها و أداة باسم "PctSource" تحمل صورة ما ، و زر تحكم.

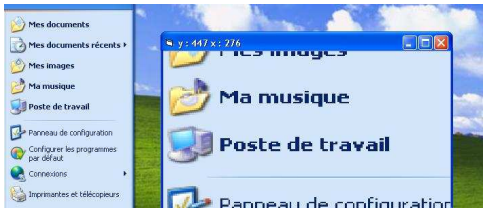
```
Private Const SRCCOPY = &HCC0020

Private Sub Command1_Click()
Dim Ret As Long
Const zoum As Single = 1.5          ' قيمة افتراضية للتكبير

PctSource.ScaleMode = vbPixels      '    ضبط وحدة القياس
لأداتي الصور
PctDestination.ScaleMode = vbPixels

PctDestination.Width = PctSource.Width * zoum          'تغيير حجم
الأداة
PctDestination.Height = PctSource.Height * zoum
PctDestination.AutoRedraw = True            image الخاصية  'لكي تستفيد
من

Ret = StretchBlt(PctDestination.hdc, 0, 0, PctSource.ScaleWidth *
zoum, _   PctSource.ScaleHeight * zoum, PctSource.hdc, 0, 0,
PctSource.ScaleWidth _ ,
                PctSource.ScaleHeight, SRCCOPY )
PctDestination.Refresh
If  Ret = 0 Then: MsgBox "العملية غير ناجحة" ,vbInformation
                              'ترجع الدالة القيمة 0 في حالة فشلها
End Sub
```



لك مني مثال جيد باسم "المكبرة" في مجلد الأمثلة المضغوطة المدْرجة مع هذه المقال يستعمل هذا المثال دالة "CreateDC" التي عملها هو إنشاء سياق الجهاز.

الشكل المقابل : (رسم حتى قائمة ابدأ على واجهة البرنامج)

كما تمكنك "API" من التعامل مع الخطوط "Font" كرسمها إن صح التعبير أو إنشاءها أو تسجيلها في النظام أو حساب المسافات بين الحروف أو طباعتها أو....، فالدالة "TextOut" تكتب سلسلة نصية على سياق مخصص بإحداثيتان تشيران إلى بداية الكتابة و قيمة تمثل طول السلسلة. و الدالة "GetTextExtentPoint32" تعود بحجمي عرض و ارتفاع سلسلة نصية مخصصة تستقبلها الدالة كثاني قيمة و طول هذه السلسلة كثالث قيمة و متغير التركيب الذي ترجع فيه الحجمين :

```
Private Type SIZE
    cx As Long
    cy As Long
End Type

Dim Taille  As SIZE
GetTextExtentPoint32 hdc, "MyText", Len("MyText"), Taille
Print Taille.cx     'عرض النص
Print Taille.cy     'ارتفاع النص
```

الدالة "AddFontResource" لإضافة خط جديد (تسجيله بالنظام) تستقبل فقط اسم ملف الخط المراد إضافته، (امتداد "TTF" أو "FON" "...") طبعا الاسم كامل أي بمسار الوصول ، و نظيرتها "RemoveFontResource" لحذفه.

2.مكتبة "Kernel32" :

كان من المفترض البدء بهذه المكتبة نظراً لأهميتها بين نظيراتها من المكتبات الأخرى ،لأن أسلوبنا هو البدء بالأهم ثم المهم لكن كما أسلفنا ذكره بدأنا بالمكتبة السابقة نظرا لسهولة دوالها عموماً.

بادئ ببدء سنتعرض لدالتا مسار الوندوز "GetWindowsDirectory" و مسار النظام "GetSystemDirectory" ،هاتان الدالتان تستقبلان متغيرين، الأول حرفي يحمل سعة كافية بالذاكرة "Buffer" ترجع به النتيجة و الثاني صحيح طويل يحمل طول هذه السعة، القيمة التي ترجعها كل منهما تشير إلى طول المسار و إلا فصفر. سنقوم في مثالنا هذا بعمل دالة حرفية ترجع لنا مسار الوندوز باستعمال الدالة الأولى:

```
Private Function PathWindows() As String
Dim Buffer As String, LTemp As Long

Buffer = Space$(255)                    'حجز سعة تخزينية في الذاكرة

LTemp = GetWindowsDirectory(Buffer, Len(Buffer))    'بدل من إرسال متغير آخر بنفس السعة
Len استعل الدالة
PathWindows = Left$(Buffer, LTemp)

End Function
```

استعمل نفس الطريقة مع الدالة الثانية.

المثال الثاني:

يستطيع برنامجك التأكد من المساحة المتوفرة على القرص الصلب قبل تثبيته عليه بالدالة "GetDiskFreeSpace"، كما تزودك هذه الدالة بالمعلومات الهامة عليه: المساحة التخزينية الفارغة، المحجوزة والمساحة الكلية . لكن القيم المرسلة إلينا لا تكون مباشرة بل قيم تشير إلى: رقم القطاعات في العنقود ، البايتات "Bytes" في القطاع ، العناقيد الفارغة، وإجمالي العناقيد.إذن سنحتاج إلى إجراء بسيط يحسب لنا المساحات المطلوبة ،أولا نتعرف على القيم التي تستقبلها الدالة و القيم التي تعود بها:

"lpRootPathName" :تستقبل به الدالة المسار (الدليل الجذري).

"lpSectorsPerCluster": يرجع عدد القطاعات في العنقود.

"lpBytesPerSector": عدد البايتات في القطاع.

"lpNumberOfFreeClusters":عدد العناقيد الفارغة.

"lpTotalNumberOfClusters":يرجع هذه الأخير العدد الإجمالي للعناقيد.

```
Private Sub Sub_Click()

Dim NumSectorsPerCluster    As Long
Dim NumBytesPerSector       As Long
Dim NumberFreeClusters      As Long
Dim TotalNumberClusters     As Long

Dim TotalSpace              As Double
Dim TotalFree               As Double

GetDiskFreeSpace "C:\", NumSectorsPerCluster, NumBytesPerSector_ ,
                        NumberFreeClusters, TotalNumberClusters
TotalSpace = CDbl(TotalNumberClusters) * CDbl(NumSectorsPerCluster) *
CDbl(NumBytesPerSector)
TotalFree = CDbl(NumberFreeClusters) * CDbl(NumSectorsPerCluster) *
CDbl(NumBytesPerSector)
Me.FontSize = 18
    Print "المسار C :"
    Print " رقم القطاعات في العنقود:" + Str$(NumSectorsPerCluster)
    Print " رقم البايتات في القطاع:" + Str$(NumBytesPerSector)
    Print " رقم العناقيد الفارغة:" + Str$(NumberFreeClusters)
    Print " مجموع العناقيد:" + Str$(TotalNumberClusters)
    Print "المساحة الإجمالية بالبايت :" & Str$(TotalSpace)
    Print " المساحة الفارغة بالبايت  :" & Str$(TotalFree)
    Print " المساحة المحجوزة بالبايت:" & Str$(TotalSpace - TotalFree)

End Sub
```

حول القيم المرجعة من الدالة إلى "Double " لتفادي خطأ زيادة القيمة على النوع "OverFlow".

كما أن الدالة "GlobalMemoryStatus" تفي بهذا الغرض لكن طريقة تعامل هذه الأخيرة ليست كالسابقة.

المثال الثالث :

قد يحتاج برنامجك لملف أو لأكثر لحفظ كل التغيرات الحديثة التي طرأت عليه من طرف المستخدم أثناء عمله (كحفظ مسارات لملفات المشاريع الحديثة المنجزة ببرنامجك على سبيل المثال) ثم ينطلق بها أي التغييرات أثناء اشتغاله في المرة القادمة. هذه الملفات تستعمل كقاعدة بيانات أشبه بقاعدة سجل النظام وتسمى ملفات التهيئة أو التأسيس كما يسميها البعض، وهي ذات امتداد "ini".نبدأ مع الدالة الأولى "WritePrivateProfileString " للإنشاء أو التعديل ،تقوم هذه الأخيرة بالبحث عن الملف المحدد إذا وجدته تجري عليه التعديلات المعينة و إلّم تجده تقوم بإنشائه كما تغنيك عن أي إشكال في تفاصيل البحث عن الأقسام أو المفاتيح ثم الكتابة و هنا تكمن قوة و مرونة هذه الدالة .

المتغيرات المرسلة:

"lpApplicationName" يشير إلى اسم القسم .

"lpKeyName" يشير إلى اسم المفتاح ،

"lpString" يشير إلى القيمة المراد إضافتها إلى المفتاح ،

"lpFileName" اسم الملف مع مساره الكامل .

الشكل العام للملف:

```
[lpApplicationName]
lpKeyName = lpString
```

```
Dim Path$
Path = App.Path & IIf(Right$(App.Path, 1) <> "\", "\", vbNullString) &
"IntFile.ini"
WritePrivateProfileString "Section1", "Key1", "Value", Path
```

أما الدالة الثانية "GetPrivateProfileString" تستعمل للقراءة من هذا الملف :

"lpApplicationName" يشير إلى اسم القسم ،

"lpKeyName" يشير إلى اسم المفتاح،

"lpDefault" يعود بقيمة افتراضية إذا لم يحدد المفتاح.

"lpReturnedString" يرجع بقيمة المفتاح المخصص إن وجده ،وإلا يرجع القيمة الافتراضية السابقة، كما يتطلب إرساله حمل قيمة أو حجز سعة ذاكريه.

"nSize" تستقبل به الدالة قيمة طول المتغير السابق "lpReturnedString".

"lpFileName" يمثل اسم الملف بمساره.

سنجري مثال أو إجراء يظهر لنا اسم آخر ملف أو مشروع يعرضه في خاصية العنوان "Caption" للعنصر "MenuFileRecent" من قائمة "File" و يكون في حدث التحميل للنموذج.إذن بعد أن تنشأ قائمة تدرج فيها العنصر "MenuFileRecent" أكتر الإجراء التالي :

```
Private Sub Form_Load()
Dim Path$, buffer$, Ret&

Path = App.Path & IIf(Right$(App.Path, 1) <> "\", "\", vbNullString) &
"IntFile.ini"
buffer = Space(255)

Ret = GetPrivateProfileString("Menu", "MenuFileRecent", "", buffer,
Len(buffer), Path)
MenuFileRecent.Caption = Left$(buffer, Ret)

End Sub
```

تعود الدالة بقيمة طول الحروف المعادة بالمتغير الرابع عموما و إلا صفر في حالة فشلها.

3. مكتبة "User32" :

تهتم هذه المكتبة بكل ما هو نافذة ، ما هي النافذة ؟ ،هي كل أداة تحمل خاصية المقبض و هذا الأخير كما قلنا عبارة عن رقم طويل يميز به نظام التشغيل هذه الأداة عن سائر الأدوات فقد تحتاج إلى تغيير شكل النموذج أو إلى إضافة صورة إلى قائمة أو إلى قلب الأدوات إلى جهة اليمين (تقنية المرآة) أو تغيير مؤشر الفأرة الساكن بآخر متحرك و ملون.....

من الدوال الأساسية في هذه المكتبة الدالة "GetWindowLong" و نظيرتها "SetWindowLong" ،تزودك الأولى بالمعلومات الخاصة بالنافذة المحددة حسب الثابت المرسل لها عن طريق المتغير "nIndex" من هذه الثوابت :

"GWL_WNDPROC" تُرجع به الدالة مقبض عنوان إجراء النافذة.

"GWL_STYLE" تعيد به أسلوب أو شكل النافذة.

"GWL_EXSTYLE" تعيد به أسلوب النافذة الموسع "Extended Style".

أما الدالة الثانية فهي عكس الأولى إن صح التعبير أي تغير خاصية من خصائص نافذة معينة بحسب القيم المرسلة :

"nIndex" هو تقريبا نفسه المرسل للدالة السابقة (لإرسال ثابت العملية المراد إجراءها على النافذة).

"dwNewLong" يشير إلى القيمة الجديدة المراد الاستبدال بها.

المثال الأول: (تقنية المرآة)

سميت بالمرآة لأنها تعكس شكل النافذة عكس شبه تمامي فتصبح كأنها صورة طيفية لنافذة أخرى متجهة نحو المرآة (مع العلم أن "VB6" لا يدعم هذه التقنية بشكل تام).

سنقوم إن شاء الله بعمل إجراء بسيط يؤدي لنا هذه المهمة ،يستقبل هذا الأخير متغير يمثل مقبض نافذة و ينقسم عمله إلى مهمتين :

المهمة الأولى هي الحصول على قيمة أسلوب (نمط) النافذة المرسلة للإجراء بالدالة الأولى "GetWin..."،

و المهمة الثانية هي تغيير نمط النافذة الموسع بالدالة الثانية "SetWin..." وبمساعدة الثابت "WS_EX_LAYOUTRTL" :

```
Private Declare Function GetWindowLong Lib "user32" Alias
"GetWindowLongA"…...
Private Declare Function SetWindowLong Lib "user32" Alias
"SetWindowLongA"…...
Private Const GWL_EXSTYLE = (-20)
Private Const WS_EX_LAYOUTRTL = &H400000
```

```
Sub Mirror(Obj_hWnd As Long)
Dim Style As Long
 Style = GetWindowLong(Obj_hWnd, GWL_EXSTYLE)
 SetWindowLong Obj_hWnd, GWL_EXSTYLE, Style Or WS_EX_LAYOUTRTL
End Sub
```

أحسن حدث لنداء هذا الإجراء هو "Form_Load ()" :

```
Private Sub Form_Load()
   Mirror Me.hWnd
   Mirror Text1.hWnd
   Mirror Drive1.hWnd
End Sub
```

المثال الثاني :(دالة شفافية النافذة ذات الطبقات)

لا شيء مستحيل مع "API" ، أكيد أنك تساءلت يوما عما إذا كنت تستطيع أن تغير شكل النموذج "Form" من الشكل القياسي المستطيل أو المربع إلى شكل آخر سواءً هندسي أو غير هندسي كما تلاحظ ذالك في بعض البرامج كـــ "ErrorDoctor" على سبيل المثال ،

فقد ترغب في جعل واجهة برنامجك عبارة عن صورة ما تعبر عن عمله و لا يظهر من النموذج سوى الصورة و لا حتى خلفية هذه الأخيرة .

المهمة تمكنك منها الدالة الشهيرة "SetLayeredWindowAttributes" التي عملها ليس مقتصراً في جعل النموذج شفاف فقط ، بل تتعدى ذالك إلى إخفاء جزء منه (محدد بلون مرسل للدالة) دون آخر وهذا هو محور موضوعنا.

"crKey" مؤشر للون المناطق المراد إخفاءها من النموذج.

"bAlpha" يحدد درجة الشفافية، القيمة بين 0 (شفافية قصوى) و 255 (معتمة).

"dwFlags" يحدد نوع العملية بأحد الثابتين:

"LWA_COLORKEY" يستعين هذا الثابت بالمتغير "crKey" لعملية الشفافية المنطقية (للمناطق التي بهذا اللون).

"LWA_ALPHA" يستعين هذا الثابت بالمتغير "bAlpha" لعملية شفافية النافذة.(الشفافية العامة) .

أنشئ مشروع جديد و أضف للنموذج صورة بالخاصية "Picture" لتكون هي كل ما سيظهر من النموذج بشرط أن تكون خلفية هذه الصورة ذات لون واحد (لكل النقاط"Pixels" المراد إخفاءها) ، عدِّل خاصية "BorderStyle" إلى " 0 - None"،(لتجميل النموذج)

أضف إلى مشروعنا الدالتين السابقتين "GetWindowLong" و "SetWindowLong" ثم الدالة الأخيرة "SetLayeredWind...."

و الثابتين "GWL_EXSTYLE" و "WS_EX_LAYERED" .

سيناريو مثالنا الآتي يقتضي بالحصول على نمط النافذة الموسع ثم تغييره كما فعلنا في المثال السابق لكن بالثابت "WS_EX_LAYERED"، ثم أخيرًا إخفاء المناطق المخصصة:

```
Private Declare Function SetLayeredWindowAttributes Lib "user32"
(ByVal hWnd As ......
Const GWL_EXSTYLE = (-20)
Const WS_EX_LAYERED = &H80000
Const LWA_COLORKEY = &H1

_____________________________________________________

Private Sub Form_Load()
  Dim CurStyle As Long
  CurStyle = GetWindowLong(Me.hWnd, GWL_EXSTYLE)
  SetWindowLong Me.hWnd, GWL_EXSTYLE, CurStyle Or WS_EX_LAYERED
  SetLayeredWindowAttributes Me.hWnd, vbWhite, 0, LWA_COLORKEY
End Sub
```

ملاحظة: إذا لم تجد الدالة في عارض الدوال تستطيع إنزال نسخة أحدث منه و مجانية على هذا الرابط :

http://www.activevb.de/rubriken/apiviewer/index-apiviewereng.html

الشكلين التاليين يوضحان شكل النموذج قبل و بعد عملية التشكيل:

تلاحظ إخفاء المناطق البيضاء فقط لأننا أرسلنا ثابت هذا اللون "vbWhite" للدالة، كما أن قوتها لا تتجلى فقط في إخفاء هذه المناطق من النموذج التي بهذا اللون بل تخفي أي أداة "Control" مادامت تحمل نفس اللون، إذن تستطيع أن تحدث ثقب على سبيل المثال بالأداة "Shape" .
حتى أنها تخفي مناطق من أزرار شريط النافذة !! .

تمرير مؤشرات دوال إلى إجراءات "DLL"

تستطيع بمصطلح مؤشر الدالة تمرير عنوان دالة ما (معينة من طرف المستخدم) كمعامل"parameter" لدالة أخرى أعلنت عنها للاستخدام في برنامجك.

باستخدام مؤشرات الدوال تستطيع الإستدعاء و الإستفادة من دوال كـــ" EnumWindows" لإنشاء قائمة بالنوافذ المفتوحة في النظام، أو الدالة "EnumFontFamilies" لتعداد و وضع قائمة بالخطوط المتوفرة في النظام، كما تمكنك من الوصول إلى دوال أخرى من دوال "Win32 API" التي لم تكن ممكنة مع الإصدارات الأقدم من "VB".

من أجل فهم استخدامات مؤشرات الدوال من الأفضل الاستعانة بالأمثلة

أدرج على سبيل المثال الدالة الشهيرة "EnumWindows" من عارض الدوال:

```
Declare Function EnumWindows Lib "user32" (ByVal lpEnumFunc As _
 Long, ByVal lParam As Long) As Long
```

هذه دالة حساب، وهذا يعني أنها تقوم بإعطائك مقابض كل النوافذ المفتوحة في النظام ، أما طريقة عملها فهي تعمل بالاستدعاء و بأسلوب تكراري للدالة التي عينتها كأول معامل لها ("lpEnumFunc") ، ففي كل مرة تنادي 'EnumWindows' للدالة تقوم فيها بتمرير مقبض لنافذة مفتوحة ، بعدها ما يبقى لديك إلا المعالجة لهذه المجموعة من القيم . على سبيل المثال ، تستطيع إنشاء دالة لسرد هذه القيم ،

"hWnd" ثم تبديلها بأسماء نوافذها و إدراج هذه الأسماء داخل أداة نص ، أو إلى غير ذالك .

لتخصيص الدالة (المعينة من طرف المستخدم) التي تريد تمريرها إلى دالة "API" ، قم بتسبيق اسم هذه الدالة بالكلمة المفتاح " *AdressOf*"، أما "lParam" يمكن أن يمرر أي قيمة صحيحة ، مثال على ذالك : لتمرير دالة باسم "MyProc" تستطيع استدعاء الدالة "EnumWindows" كما يلي :

```
Ret = EnumWindows( AddressOf MyProc, 5)
```

مثال :

أنشئ مشروع جديد و أدرج به أداة القائمة باسم "List1" و أضف ملف برمجة "BAS" وأعلن فيه عن الدالة "EnumWindows" والدالة "GetWindowText" . هذه الأخيرة تمكننا من معرفة النص ("Text" أو "Caption") المرتبط مع النافذة المرسل مقبضها لهذه الدالة ، وتعود بقيمة تمثل طول هذا النص. بعد هذا أعلن عن إجراءك :

```
Option Explicit

Public Declare Function EnumWindows Lib "user32" (ByVal ......
Public Declare Function GetWindowText Lib "user32" Alias ....

Public Function My_Proc(ByVal lgHwnd As Long, ByVal lgParam As _
Long) As  Long
 Dim Buffer$, Wind_Caption$
     Buffer = Space(100)

Wind_Caption = Left$(Buffer, GetWindowText(lgHwnd, Buffer, 100))
Form1!List1.AddItem Trim$("&H" & Hex$(lgHwnd)) & " : " & Wind_Caption

 My_Proc = 1
End Function
```

ثم توجه إلى حدث النقر للنموذج و استدعي دالة موضوعنا اليوم .

```
Option Explicit


Private Sub Form_Click()
 Dim Ret&

 Ret = EnumWindows(AddressOf My_Proc, 0&)

End Sub
```

هام جدًا :

لا تنسى أبدًا أن تقوم بحفظ مشروعك قبل التنفيذ و لا تحاول أبدًا القيام بعملية التنقيح (التنفيذ خطوة بخطوة)، أو حتى وضع نقاط توقف عند سطر ما و إلاّ قد يتوقف "VB" عن العمل و تضيع برنامجك .

الرسائل و التصنيف الفرعي :

1-الرسائل :

- مفاهيم :

من المعروف أن نظام (ويندوز) مرتكز على إطلاق الأحداث (الرسائل)، فهناك تبادل دائم لرسائل مختلفة بين نظام التشغيل و البرامج النشطة أو العاملة "Process" ، أغلب هذه الرسائل تنطلق بتدخل من المستخدم .

ماهية الرسائل ؟، في أي مرة يقوم فيها المستخدم بفعل أمر ما كـــ (تحريك الفأرة، الضغط على زر من لوح المفاتيح ...) ينشىء النظام رسالة، وهي عبارة عن قيمة عددية يوجهُهَا إلى نموذجك ، يقوم "VB" بالمعالجة و التنفيذ لهذه الرسائل ثم يطلق أحداث "Events" مقابل البعض من هذه الرسائل.على سبيل المثال ، حدث النقر "Click" ينشأ و ينطلق بواسطة الـــ "VB" و هذا عندما يستقبل من النظام رسالة تخبره بأن المستخدم قام بالنقر على النموذج .

يقوم كل برنامج بالاستعلام عن الرسائل التي قد تصله ، وهذا بما يسمى بحلقة الرسائل "Message Loop" .الكثير من الرسائل تأتي متعاقبة و سريعة ولهذا وجد ما يسمى بطابور الرسائل "Message Queue" ، فكل رسالة تصل تعالج و توضع حسب وقت وصولها ، أي من أقدم رسالة إلى أحدث رسالة.

يرسل النظام رسالة إلى إجراء النافذة مع مجموعة أربع قيم "parameters" : مقبض النافذة ، معرف الرسالة " message identifier"، وقيمتان تمثلان خصائص الرسالة تسمى "message parameters".

لكل رسالة معرِّف أو ما يسمى بثابت الرسالة ، يميز الغرض من هذه الرسالة. فعندما يستلم إجراء النافذة رسالة يقوم باستعمال معرِّفها لتحديد كيفية المعالجة، فالمعرف " WM_PAINT" يخبر الإجراء بأن واجهة النافذة قد تغيرت ويجب إعادة رسمها .

قبل المضي أكثر دعنا نجري فاصل قصير نلقي فيه نضرة بسيطة على **إجراء النافذة** ثم نعود إلى موضوعنا :

إجراء النافذة عبارة عن دالة مهمتها استقبال و معالجة الرسائل المبعوثة إلى النافذة المرتبطة مع هذه الدالة . أي كل فئة نافذة لها إجراء نافذة وكل نافذة أنشأة بتلك الفئة تستعمل ذاك الإجراء نفسه للرد على الرسائل، فكل سمات ظهور النافذة و سلوكها يعتمدان على رد إجراء النافذة على هذه الرسائل.

حروف أو بمعنى أدق رموز معرِّف الرسالة ترمز إلى شيئين تفصلهما شرِطة سفلية "_" :الأول بادئة المعرف و تدل على صنف النافذة ، والثاني نص الرسالة ، أشبه بالأحداث في الـــ "VB" .

يوجد بمكتبة "MSDN" جدول بمختلف بادئات معرفات الرسائل و معناها . وفي ما يلي البعض من هذا الجدول :

Prefix	Message category
BM	Button control
CB	Combo box control
CDM	Common dialog box
DBT	Device
EM	Edit control
LB	List box control
LVM	List view control
PBM	Progress bar
SB	Status bar window
SBM	Scroll bar control
TB	Toolbar
TCM	Tab control

TVM	Tree-view control
UDM	Up-down control
WM	General window

أما باقي الجدول لمن أراد فهو على هذا الرابط:

http://msdn2.microsoft.com/en-us/library/ms644927.aspx#system_defined

- إرسال الرسائل :

ألم يجرك التفكير يوما إلى إرسال رسالة ما إلى إحداهن ؟؟؟، (أقصد النوافذ) أكيد فكرت ، لكن كيف ؟،

يمكنك أو برنامجك إرسال رسائل إلى نافذة تابعة إلى برنامجك أو برنامج آخر كالنظام ، و بطريقتين . ما المقصود بطريقتين ؟،

الذي أقصده هنا هو طريقة الإرسال المباشر إلى النافذة "*Sending*" وطريقة الإرسال الغير مباشر "*Posting*" أي وضع الرسالة في طابور الرسائل .

من أهم الدوال المسئولة عن الإرسال ، الدالة "SendMessage" و "PostMessage" ،ما الفرق بينهما ؟ :

الأولى ترسل الرسالة مباشرةً إلى إجراء النافذة المناسب أو المقابل للنافذة المحددة للقيام بمهمة فورية ، تنتظر الدالة حتى يكمل إجراء النافذة المعالجة ، بعد ذالك ترجع النتيجة ، فهي إذاً متزامنة .

الثانية غير متزامنة ، أي أن عملها مقتصر على وضع رسالة في نهاية طابور الرسائل و العودة بالنتيجة فورًا ، بدون انتظار المعالجة.

2- **التصنيف الفرعي** :

التصنيف الفرعي هو تقنية تسمح باستقبال رسائل النظام المرسلة إلى النموذج أو إلى أداة "`Control`"، فباستقبالك لهذه الرسائل تستطيع استبدال المعالجة الافتراضية لها من النظام بإجراءاتك الخاصة.

ما المشكلة إذن ما دامت الرسائل التي هي عبارة عن أحداث ، متوفرة في الفيجوال بيسك ؟، المشكلة تكمن في أن الـــ "VB" لا تنشئ كل الأحداث مع كل الرسائل .أيضا بعض الأحداث ليست سهلة الوصول، تعال و غيّر حجم النموذج بالضغط على زر تكبير من شريط العنوان للنموذج ، سيقوم البرنامج بإنشاء حدث "`Resizing-event`" ، لكن هذا الأخير يأتي بعد إنهاءك للتحجيم، لكن بهذه التقنية تستطيع استقبال رسالة "Resize" قبل إنهاءك التحجيم و هذا يعطيك فرصة لوضع أبعاد و تحديدات خاصة ببرنامجك ؛ كما أن هذا المبدأ قابل للتطبيق على أغلب الرسائل.

التصنيف الفرعي متوفر منذ الإصدارة الخامسة " VB5" التي تدرج الكلمة المفتاح " *AdressOf*" . لكن من قبل كانت تستعمل أداة خارجية لاسترجاع عنوان الإجراء.

استعمل الدالة " *SetWindowLong*" مع الثابت الأول "`GWL_WNDPROC`" و المعامل السابق لاستبدال إجراء "VB" بالإجراء الذي ترغب.

بعدما توجه رسائل النظام إلى إجراءك ما يبقى لك إلاّ المعالجة اليدوية .

هذه مختلف استعمالات التصنيف الفرعي:

- ✍ خطف الرسائل : قد ترغب في معرفة ما إذا كانت قد وصلت رسالة من النظام لكن، تترك معالجتها بدون تغيير .وفائدة ذلك على سبيل المثال عندما تحتاج إلى معرفة ما إذا كان النموذج قد بدّل من مكانه أم لا، وهذا باستقبال الرسالة "`WM_MOVE`".
- ✍ تستطيع استقبال رسالة و تغير من معالمها "`Parameters`" قبل تمريرها إلى النظام.
- ✍ تستطيع ترك النظام يعالج الرسالة ثم تقوم بتعديل النتيجة.

ملاحظة : عند إطلاقك رسالة ما باستعمال دالة كـــ " *SendMessage* " أو "*PostMessage* " يقوم النظام باستدعاء إجراء النافذة الهدف (لأنه مرتبط بها) الذي يعالج كل الأنواع الممكنة لرسائل النظام .

إجراء النافذة يستقبل بالتحديد نفس البارامترات التي عينتها أو عرفتها عند إطلاق الرسالة .

إذن تقوم هذه التقنية باستبدال إجراء النافذة بإجرائك الذي سيقوم بدلاً منه باستقبال كل الرسائل ، ثم لك أن تقرر ماذا تفعل مع أيٍّ منها، بديهيا ستقوم باختبار نوع كل رسالة تصلك ثم هل ستعالج كل الرسائل أم البعض منها ؟؟، المعالجة ستكون للبعض فقط أي للرسائل التي من أجلها توجهنا إلى هذه التقنية ، ثم إحالة الأمر لـــ "VB" كي يتولى البقية.

قبل إنهاء البرنامج من الواجب إرجاع المؤشرات إلى ما كانت اتقاء شر سوء إنهاء الـــ "VB" لبرنامجك أو في بعض الأحيان قد يحدث كبح حتى للنظام .

ملاحظة : إجراءك يجب أن يعرّف في ملف البرمجة "BAS" لأن المعامل "*AdressOf*" لا يعمل إلاّ في هذا النوع من الملفات البرمجية، كما ينصح باستعمال الطريقة "Unload Me" عند الخروج من البرنامج أولى من "End" التي لا تنهي البرنامج تماما.

قبل الشروع في كتابة شفرات المثال سنقوم بوضع مهمة تلخص لنا الخطوات الأساسية في التعامل مع هذه التقنية :

المهمة ( )

- Hook: تثبيت عنوان إجراءك مكان عنوان الأجراء القياسي للنظام.
- المعالجة : للرسائل المعنونة لإجرائك.
- UnHook: إعادة المؤشرات إلى حالتها لأجل الإنهاء الصحيح للبرنامج .

نهاية المهمة ( )

تفاصيل هذه الخطوات :

1- Hook :

```
Address_WndProc = SetWindowLong ( hWnd , GWL_WNDPROC , AddressOf My_WndProc )
```

بعدما تستبدل الدالة " *SetWindowLong* " الإجراء القياسي "WndProc" بإجرائك " My_WndProc " تمنحك عند عودتها قيمة تمثل عنوان الإجراء الأصلي "WndProc" لأجل حفظه ثم إرجاعه أثناء نهاية البرنامج.

2- معالجة الرسائل :

يقوم إجراءك باختبار الرسائل الموجهة إليه و المعالجة للمقصود منها ثم تمرير التحكم للإجراء الأصلي أو القياسي .

```
Function My_WndProc ( ByVal hwnd as Long _                    ' مقبض النافذة الهدف
  ByVal uMsg As Long _                                        ' الرسالة المعنونة
  ByVal wParam As Long _                ' يحملان معلومات خاصة بالرسالة wParam و lParam
  ByVal lParam As Long ) As Long

Select case uMsg
Case WM_MOVE
  ' النافذة قد انتقلت من مكانها
  ....
     Case WM_GETMINMAXINFO      ' رسالة التحجيم


     Case Else
My_WndProc = CallWindowProc ( Address_WndProc , hWnd , uMsg , wParam , lParam )     ' نداء
الإجراء القياسي ( الأصلي) لتولي باقي الرسائل

End Select
End Function
```

3- UnHook :

إعادة عنوان الإجراء القياسي قبل إنهاء البرنامج .

SetWindowLong (hWnd , GWL_WNDPROC , Address_WndProc)

أما باقي المثال فهو مدرج مع هذه المقال كملف مضغوط باسم "minmaxinfo".

ولمن يريد أمثلة أخرى لاستعمالات التصنيف الفرعي فهذا الموقع :

http://www.vbexplorer.com/VBExplorer/Downloads/VB_API_Subclassing.asp

(المثال السابق مأخوذ منه) ☺

كما يوجد على الرابط التالي كتاب ضخم لمن أراد الاستزادة (يقارب 700 صفحة) بعنوان " Subclassing and Hooking with Visual Basic " طبعا إنجليزي اللغة :

http://rapidshare.com/files/14620338/0596001185.rar

تمرير مؤشر دالة إلى دالة أخرى فيه مخاطرة كبيرة والأخطر من ذالك تقنية التصنيف الفرعي و التي تستوجب من المبرمج الخبرة الكبيرة قبل اللعب بأسلاك التيار العالي!!. ما أردته فقط من هذا الشرح البسيط حول هذه التقنية هو كشف بعض الغموض ليس إلاّ .

كلمة أخيرة :

كانت و لازالت إجراءات "API" مدعومة في الـــ "VB" إلى يومنا هذا رغم التطور الرهيب للإصدارات الأحدث من هذا الأخير ، إلاّ أن اللجوء إليها قل بشكل كبير .

أما طريقة التعامل معها لازالت تقريبا كما هي مع بعض الاختلاف البسيط فالمثال التالي مأخوذ من مكتبة "MSDN" يعرض طريقة استخدام دالة "getUserName" التي ترجع اسم المستخدم وهذا بالإصدارة الثامنة من الـــ "VB" :

```
Declare Function getUserName Lib"advapi32.dll"Alias"GetUserNameA"ByVal _
                 lpBuffer As String ,ByRef  nSize As Integer)As Integer
Sub getUser()
 Dim buffer As String = New String(CChar(" "), 25)
 Dim retVal As Integer = getUserName(buffer, 25)
 Dim userName As String = Strings.Left(buffer, InStr(buffer, Chr(0))- 1)
    MsgBox(userName)
End Sub
```

ما أوصيك به قارئي العزيز فقط هو إمعان النظر في الدالة التي تريد استعمالها و التركيز في القيم المرسلة قبل المباشرة بالاستخدام و التحقق بعد ذالك من القيمة المعادة لاتخاذ الإجراء المناسب ووضع في عين الاعتبار اِحتمالية فشل الدالة لا قدّر الله .

إذن لا تأخذ دالة أو مثال عن استعمال مجموعة دوال لمهمة ما و تقوم بالاستعمال الجدِّي دون معرفة مسبقة بأي منها.

كانت هذه جولة بسيطة حول إجراءات "API" وطرق استعمال البعض منها ، فإن كان منها نصيب للخطأ فمني و من الشيطان فمعذرة لذالك ،خاصةً للأسلوب اللغوي ، و ما كان منها صواب فمن ربي جلّ و على ، ما أتمناه هو إيصال فكرة ولو بسيطة في هذا المجال ؛

و الله أعلم .

الدعاء الصالح لآخوكم في الله فيصل قسميه .