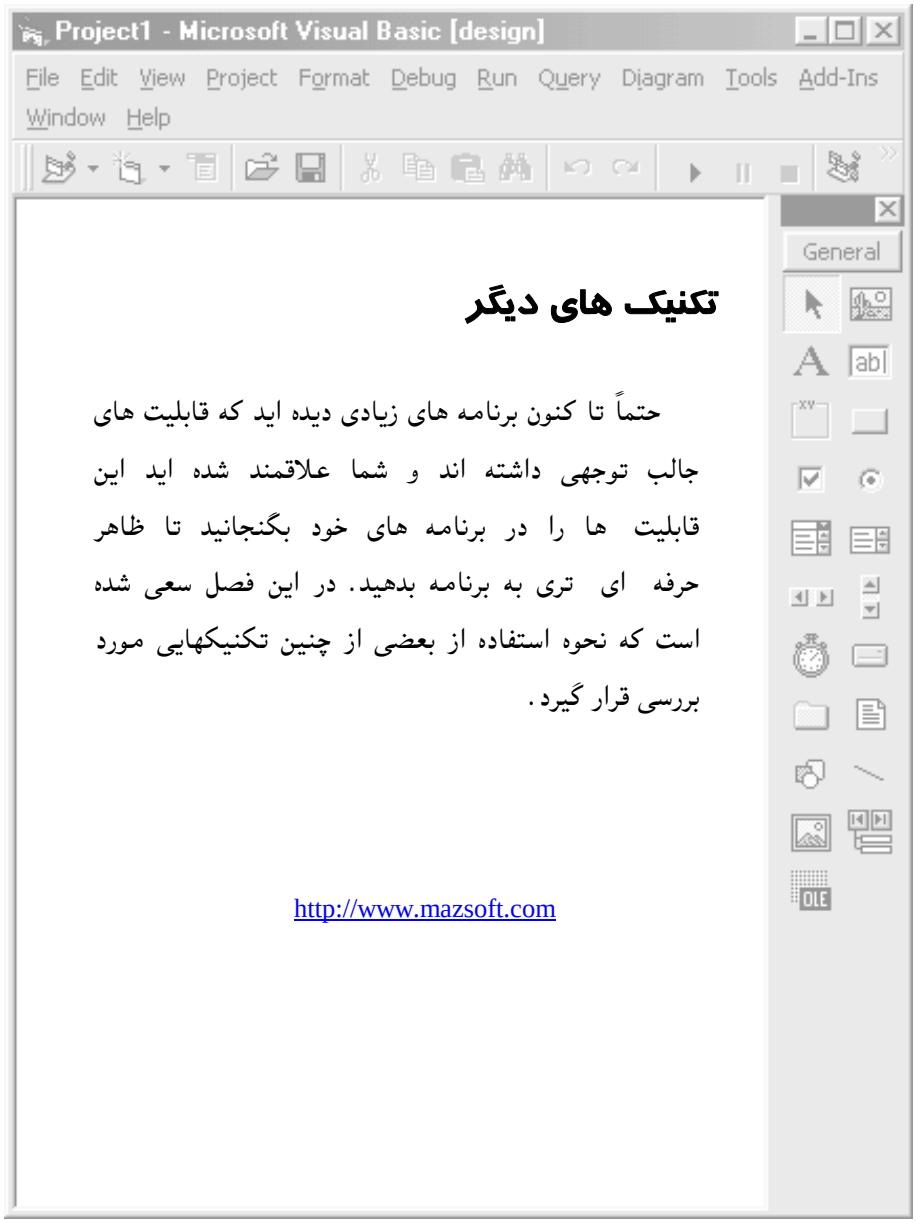
Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins
Window Help
General
تکنیک های دیگر
حتماً تا کنون برنامه های زیادی دیده اید که قابلیت های جالب توجهی داشته اند و شما علاقمند شده اید این قابلیت ها را در برنامه های خود بگنجانید تا ظاهر حرفه ای تری به برنامه بدهید. در این فصل سعی شده است که نحوه استفاده از بعضی از چنین تکنیکهایی مورد بررسی قرار گیرد.
http://www.mazsoft.com

استفاده مستقیم از امکانات System Tray در ویژوال بیسیک

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
176085.TXT	۷ آگوست ۱۹۹۸	5 , 6	95 / 98 / NT

حتماً تا کنون برنامه های زیادی دیده اید که آیکونی را در System Tray (بخشی از Task Bar که در سمت راست صفحه قرار می گیرد و معمولا حاوی ساعت سیستم می باشد) قرار می دهند و از آن برای برخی کارها استفاده می کنند. در این بخش نشان داده خواهد شد که چگونه از طریق ویژوال بیسیک بطور کامل از امکانات System Tray استفاده می شود. در مثال زیر نشان داده می شود که چگونه آیکون دلخواه خود را در System Tray قرار دهید، که این آیکون می تواند هنگامی که ماوس را روی آن قرار می دهید، ToolTip دلخواه شما را نمایش دهد و هنگامی که روی آن کلیک کنید، پنجره اصلی برنامه را ظاهر نماید و اگر روی آن با دکمه راست ماوس کلیک کنید یک منوی popup ظاهر نماید. به دلیل قابلیت ویژوال بیسیک در مدیریت مستقیم فراخوانی های بازگشتی[۱]، تمام این کارها به سهولت قابل انجام است و در نتیجه می توان از تمامی امکانات تابع Shell_NotifyIcon که عمل نمایش آیکون در System Tray را انجام می دهد، استفاده نمود.

مثال زیر را می توان به هر پروژه ویژوال بیسیک که حداقل یک فرم و یک ماژول داشته باشد اضافه نمود:

۱- کد زیر را به بخش declarations از یک ماژول برنامه خود اضافه کنید:

```
'user defined type required by Shell_NotifyIcon API call
Public Type NOTIFYICONDATA
    cbSize As Long
    hwnd As Long
    uId As Long
    uFlags As Long
    uCallBackMessage As Long
    hIcon As Long
```

[۱] Callbacks

```
    szTip As String * 64
End Type

'constants required by Shell_NotifyIcon API call:
Public Const NIM_ADD = &H0
   Public Const NIM_MODIFY = &H1
   Public Const NIM_DELETE = &H2
   Public Const NIF_MESSAGE = &H1
   Public Const NIF_ICON = &H2
   Public Const NIF_TIP = &H4
   Public Const WM_MOUSEMOVE = &H200
   Public Const WM_LBUTTONDOWN = &H201     'Button down
   Public Const WM_LBUTTONUP = &H202       'Button up
   Public Const WM_LBUTTONDBLCLK = &H203   'Double-click
   Public Const WM_RBUTTONDOWN = &H204     'Button down
   Public Const WM_RBUTTONUP = &H205       'Button up
   Public Const WM_RBUTTONDBLCLK = &H206   'Double-click

   Public Declare Function SetForegroundWindow Lib "user32" _
   (ByVal hwnd As Long) As Long
   Public Declare Function Shell_NotifyIcon Lib "shell32" _
   Alias "Shell_NotifyIconA" _
   (ByVal dwMessage As Long, pnid As NOTIFYICONDATA) As Boolean

   Public nid As NOTIFYICONDATA
```

۲- کد زیر را به یکی از فرمهای برنامه خود که می خواهید به آیکون مربوط به برنامه شما در System Tray پاسخ دهد وارد نمایید:

```
Private Sub Form_Load()
     'the form must be fully visible before calling Shell_NotifyIcon
     Me.Show
     Me.Refresh
     With nid
      .cbSize = Len(nid)
      .hwnd = Me.hwnd
      .uId = vbNull
      .uFlags = NIF_ICON Or NIF_TIP Or NIF_MESSAGE
      .uCallBackMessage = WM_MOUSEMOVE
      .hIcon = Me.Icon
      .szTip = "Your ToolTip" & vbNullChar
     End With
     Shell_NotifyIcon NIM_ADD, nid
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As _
       Single, Y As Single)
    'this procedure receives the callbacks from the System Tray icon.
    Dim Result As Long
    Dim msg As Long
    'the value of X will vary depending upon the scalemode setting
    If Me.ScaleMode = vbPixels Then
        msg = X
    Else
        msg = X / Screen.TwipsPerPixelX
    End If
```

```
  Select Case msg
    Case WM_LBUTTONUP        '514 restore form window
     Me.WindowState = vbNormal
     Result = SetForegroundWindow(Me.hwnd)
     Me.Show
    Case WM_LBUTTONDBLCLK    '515 restore form window
     Me.WindowState = vbNormal
     Result = SetForegroundWindow(Me.hwnd)
     Me.Show
    Case WM_RBUTTONUP        '517 display popup menu
     Result = SetForegroundWindow(Me.hwnd)
     Me.PopupMenu Me.mPopupSys
  End Select
End Sub

Private Sub Form_Resize()
   'this is necessary to assure that the minimized window is hidden
   If Me.WindowState = vbMinimized Then Me.Hide
End Sub

Private Sub Form_Unload(Cancel As Integer)
   'this removes the icon from the system tray
   Shell_NotifyIcon NIM_DELETE, nid
End Sub

Private Sub mPopExit_Click()
   'called when user clicks the popup menu Exit command
   Unload Me
End Sub

Private Sub mPopRestore_Click()
   'called when the user clicks the popup menu Restore command
   Me.WindowState = vbNormal
   Result = SetForegroundWindow(Me.hwnd)
   Me.Show
End Sub
```

۳- تنظیمات زیر را برای خصوصیت های همان فرمی که کد بالا را در آن وارد کردید انجام دهید:

```
خصوصیت          تنظیمات لازم برای این مثال
-----------------------------------------------------------------
Icon           = The icon you want to appear in the system tray.
Minbutton      = True
ShownInTaskbar = False
```

۴- به کمک ویرایشگر منوی ویژوال بیسیک، منوهای زیر را در همان فرم وارد کنید:

```
Caption      Name          Enabled   Visible   Position
-----------------------------------------------------------
&SysTray     mPopupSys      True      False    Main Level
&Restore     mPopRestore    True      True     Inset one
&Exit        mPopExit       True      True     Inset one
```

در صورت نیاز می توانید آیتم های دیگری نیز به این منو اضافه کنید.

انعطاف پذیری System Tray

با تغییر دادن سطر زیر در روتین Form_Load می توانید متن نمایش داده شده توسط ToolTip مربوط به آیکون System Tray را تغییر دهید:

```
.szTip = "Your ToolTip" & vbNullChar
```

برای این کار متن مورد نظر خود را جایگزین عبارت "Your ToolTip" نمایید.

برای تغییر دادن آیکونی که در System Tray ظاهر خواهد شد می توانید سطر زیر را در روال Form_Load تغییر دهید:

```
.hIcon = Me.Icon
```

به این ترتیب می توانید عبارت Me.Icon را با هر آیکون دیگری در برنامه خود عوض کنید.

پس از اینکه توسط مقدار ثابت NIM_ADD یک آیکون را در System Tray قرار دادید می توانید هر زمان که لازم شد با تغییر دادن مقادیر متغیر nid و استفاده از دستور زیر مشخصات آیکون و نحوه عمل آنرا تغییر دهید:

```
Shell_NotifyIcon NIM_MODIFY, nid.
```

در صورتیکه بخواهید یک فرم دیگر غیر از فرمی که اکنون تحت کنترل System Tray قرار دارد فراخوانی های بازگشتی را دریافت کند لازمست ابتدا آیکون فعلی را توسط فرمان زیر از System Tray حذف نمایید:

```
Shell_NotifyIcon NIM_Delete, nid
```

از آنجا که NIM_Modify یک Hwnd جدید برای یک آیکون موجود قبول نمی کند این کار ضروری است. لازمست برای یک فرم جدید (که Hwnd متفاوتی دارد) مقادیر مربوط به nid مجددا تعریف شده و آیکون توسط دستور "Shell_NotifyIcon NIM_ADD, nid" ایجاد شود. علاوه براین می توانید نسخه های متفاوتی از nid برای پنجره های مختلف ایجاد نموده و در

واقعه Activate هر یک از آن پنجره ها توسط NIM_DELETE و NIM_ADD آیکون ها را جایگزین یکدیگر نمایید.

از کار انداختن CTRL+ALT+DEL و ALT+TAB در ویندوز ۹۵

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
161133.TXT	**۷ جون ۱۹۹۹**	4 , 5 , 6	95

اخطار: مسئولیت هر گونه استفاده از کد ارایه شده در مثال زیر به عهده خود شماست. مایکروسافت کد زیر را به همان گونه که هست ارایه کرده است و در مورد کاربرد آن در مصارف دیگر هیچگونه ضمانتی نمی کند.

با فراخوانی تابع SystemParametersInfo می توان از عملکرد معمولی ترکیب کلید های CTRL+ALT+DEL و ALT+TAB جلوگیری نمود. توجه داشته باشید که این تکنیک فقط در ویندوز ۹۵ قابل استفاده است.

در مواردی لازم می شود که برنامه شما از عملکرد کلید های CTRL+ALT+DEL جلوگیری نماید. فشار هم زمان این کلید ها پنجره Close Program را ظاهر می کند و این امکان را به کاربر می دهد که یک برنامه را خاتمه دهد یا ویندوز را shut down نماید. به همین ترتیب ترکیب کلیدهای ALT+TAB برای جابجا شدن بین برنامه های مختلف در حال اجرا بکار می رود. تکنیک زیر برای جلوگیری از عملکرد این کلیدها برای ویندوز ۹۵ چنین وانمود می کند که یک Screen Saver در حال اجرا است و بدین ترتیب این کلیدها غیر فعال می شوند.

در مستندات Win32 SDK گفته شده است که علامت SPI_SCREENSAVERRUNNING در ویندوز **۹۵** بطور داخلی استفاده می شود، برنامه ها نباید از آن استفاده کنند. در ویندوز NT این علامت وجود ندارد.

توجه داشته باشید که از کار انداختن ترکیب کلیدهای CTRL+ALT+DEL توصیه نمی شود، زیرا پنجره Close Program به این منظور بوجود آمده است که کاربران بتوانند توسط آن برنامه هایی که دچار مشکل شده اند را خاتمه دهند. اگر هنگامی که کلید های CTRL+ALT+DEL از کار افتاده اند برنامه ای متوقف شود، ممکنست به غیر از بوت کردن مجدد ماشین که باعث از دست رفتن داده ها می شود، راه دیگری برای خاتمه دادن به آن وجود نداشته باشد. علاوه بر این، روش بیان شده در این بخش ممکنست در نگارش های آینده ویندوز قابل استفاده نباشد.

۱- یک پروژه EXE جدید ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- دو CommandButton به نامهای Command1 و Command2 به Form1 اضافه کنید.

۳- کد زیر را به بخش Declarations از Form1 اضافه نمایید.

```
Private Const SPI_SCREENSAVERRUNNING = 97&
Private Declare Function SystemParametersInfo Lib "User32" _
        Alias "SystemParametersInfoA" _
        (ByVal uAction As Long, _
         ByVal uParam As Long, _
         lpvParam As Any, _
         ByVal fuWinIni As Long) As Long

Private Sub Form_Load()
   Command1.Caption = "Disabled"
   Command2.Caption = "Enabled"
End Sub

Private Sub Form_Unload(Cancel As Integer)
   'Re-enable CTRL+ALT+DEL and ALT+TAB before the program terminates.
    Command2_Click
End Sub

Private Sub Command1_Click()
    Dim lngRet As Long
    Dim blnOld As Boolean
    lngRet = SystemParametersInfo(SPI_SCREENSAVERRUNNING, True, _
       blnOld,0&)
End Sub

Private Sub Command2_Click()
    Dim lngRet As Long
```

```
    Dim blnOld As Boolean
    lngRet = SystemParametersInfo(SPI_SCREENSAVERRUNNING, False, _
    blnOld, 0&)
End Sub
```

۴- برای اجرای برنامه کلید F5 را فشار دهید، و روی دکمه Disabled کلیک نمایید. اکنون ترکیب کلیدهای CTRL+ALT+DEL و ALT+TAB غیر فعال شده اند. روی دکمه Enabled کلیک کنید تا این ترکیب کلیدها مجددا فعال شوند.

اجرای مرورگر اینترنت از درون برنامه

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
174156.TXT	**۲۵** می **۱۹۹۹**	4 , 5 , 6	95 / 98 / NT

برای اجرای مرورگر اینترنت پیش فرض روی سیستم می توان از توابع API استفاده نمود. برای انجام این کار ابتدا باید تعیین نمایید کدام برنامه به عنوان مرورگر اینترنت پیش فرض روی سیستم شما تعریف شده است و سپس آن را اجرا نمایید.

مثال ویژوال بیسیک زیر، مرورگر اینترنت پیش فرض موجود روی سیستم شما را اجرا می کند. بدیهی است که برای اجرای این مثال باید حداقل یک مرورگر اینترنت بدرستی روی سیستم شما نصب شده باشد. توجه داشته باشید که توابع مورد استفاده در این مثال در صورتی که بطور موفقیت آمیز عمل نکنند، مقداری کوچکتر یا مساوی با **۳۲** برمی گردانند، اما در این مثال به ثابت های نشان دهنده این پیام های خطا اشاره ای نمی شود. توصیه می شود در هنگام استفاده از این تکنیک، از روشهای کنترل خطا برای نظارت بر درست اجرا شدن برنامه استفاده شود.

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- یک Command Button به Form1 اضافه کنید. Command1 بطور پیش فرض ایجاد می شود.

۳- کد زیر را در Form1 وارد نمایید:

```
Private Const SW_SHOW = 5       ' Displays Window in its current size
                                ' and position
Private Const SW_SHOWNORMAL = 1 ' Restores Window if Minimized or
                                ' Maximized

Private Declare Function ShellExecute Lib "shell32.dll" Alias _
    "ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As _
    String, ByVal lpFile As String, ByVal lpParameters As String, _
    ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Private Declare Function FindExecutable Lib "shell32.dll" Alias _
    "FindExecutableA" (ByVal lpFile As String, ByVal lpDirectory As _
    String, ByVal lpResult As String) As Long

Private Sub Command1_Click()
    Dim FileName, Dummy As String
    Dim BrowserExec As String * 255
    Dim RetVal As Long
    Dim FileNumber As Integer

    ' First, create a known, temporary HTML file
    BrowserExec = Space(255)
    FileName = "C:\temphtm.HTM"
    FileNumber = FreeFile                   ' Get unused file number
    Open FileName For Output As #FileNumber  ' Create temp HTML file
    Write #FileNumber, "<HTML> <\HTML>"  ' Output text
    Close #FileNumber                       ' Close file
    ' Then find the application associated with it
    RetVal = FindExecutable(FileName, Dummy, BrowserExec)
    BrowserExec = Trim(BrowserExec)
    ' If an application is found, launch it!
    If RetVal <= 32 Or IsEmpty(BrowserExec) Then ' Error
        MsgBox "Could not find associated Browser", vbExclamation, _
           "Browser Not Found"
    Else
        RetVal = ShellExecute(Me.hwnd, "open", BrowserExec, _
           "www.microsoft.com", Dummy, SW_SHOWNORMAL)
        If RetVal <= 32 Then        ' Error
            MsgBox "Web Page not Opened", vbExclamation, "URL Failed"
        End If
    End If
    Kill FileName                   ' delete temp HTML file
End Sub
```

۴- برای اجرای برنامه کلید F5 را فشار دهید. روی Command1 کلیک کنید تا مرورگر اینترنت پیش فرض شما اجرا شود.

برای کسب اطلاعات بیشتر در این زمینه می توانید در مرجع برنامه نویسان Win32 یا شبکه برنامه نویسان مایکروسافت MSDN موضوعات زیر را جستجو نمایید:

- ShellExecute
- FindExecutable

استفاده از راهنماهای با فرمت HTML در برنامه های ویژوال بیسیک

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
183434.TXT	۱۷ اکتبر ۱۹۹۸	4 , 5 , 6	95 / 98 / NT

مراحل لازم برای استفاده از توابع API مربوط به راهنماهای HTML در برنامه های ویژوال بیسیک عبارتند از:

۱- ثابت های مربوط به راهنماهای HTML را تعریف کنید.

۲- تابع HtmlHelp() را معرفی کنید.

۳- تابع HtmlHelp() را به همراه پارامترهای مناسب فرابخوانید.

قطعه برنامه های زیر مراحل فوق را نشان می دهند:

۱- ثابت های مربوط به راهنمای HTML بصورت زیر تعریف می شوند:

```
Const HH_DISPLAY_TOPIC = &H0
Const HH_SET_WIN_TYPE = &H4
Const HH_GET_WIN_TYPE = &H5
Const HH_GET_WIN_HANDLE = &H6
Const HH_DISPLAY_TEXT_POPUP = &HE   ' Display string resource ID or
                                    ' text in a pop-up window.
Const HH_HELP_CONTEXT = &HF         ' Display mapped numeric value in
                                      ' dwData.
Const HH_TP_HELP_CONTEXTMENU = &H10 ' Text pop-up help, similar to
                                    ' WinHelp's HELP_CONTEXTMENU.
Const HH_TP_HELP_WM_HELP = &H11     ' text pop-up help, similar to
                                    ' WinHelp's HELP_WM_HELP.
```

۲- تابع HtmlHelp() بصورت زیر تعریف می شود:

```
Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
   (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As Long, ByVal dwData As Long) As Long
```

۳- موارد زیر مثالهایی از نحوه فراخوانی تابع HtmlHelp() در ویژوال بیسیک هستند :

```
   ' HTML Help file launched in response to a button click:
   Private Sub HH_DISPLAY_Click()
      'hWnd is a Long defined elsewhere to be the window handle
      'that will be the parent to the help window.
      Dim hwndHelp As Long
   'The return value is the window handle of the created help window.
   hwndHelp = HtmlHelp(hWnd, "myfile.chm", HH_DISPLAY_TOPIC, 0)
End Sub

' A specific topic identified by the variable ContextID is launched
' in response to this button click:
Private Sub HH_HELP_Click()
   Dim hwndHelp As Long
   'The return value is the window handle of the created help window.
   hwndHelp = HtmlHelp(hWnd, "myfile.chm", HH_HELP_CONTEXT, ContextID)
End Sub
```

مراجع :

راهنمای HTML Help Workshop

سایت اینترنت HTML Help با آدرس

```
http://www.microsoft.com/workshop/author/htmlhelp
```

پیوند دادن یک پسوند فایل با یک برنامه اجرایی

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
185453.TXT	۲۰ می ۱۹۹۹	4 , 5 , 6	95 / 98 / NT

اگر برنامه ای نوشته اید که از فایلهای داده ورودی استفاده می کند و امکان پردازش پارامترهای سطر فرمان را دارد، شاید بخواهید پسوند فایلهای داده برنامه خود را با نام برنامه اجرایی خود مرتبط نمایید. این کار به کمک دستکاری در رجیستری ویندوز قابل انجام است.

برای اعمال تغییر در فایل Reg.dat می توانید از توابع &RegCreateKey و &RegSetValue استفاده کنید.

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- کد زیر را به بخش Declarations از Form1 اضافه کنید:

```
Option Explicit

Private Declare Function RegCreateKey Lib "advapi32.dll" Alias _
"RegCreateKeyA" (ByVal hKey As Long, _
                 ByVal lpSubKey As String, _
                 phkResult As Long) As Long

Private Declare Function RegSetValue Lib "advapi32.dll" Alias _
"RegSetValueA" (ByVal hKey As Long, _
           ByVal lpSubKey As String, _
           ByVal dwType As Long, _
           ByVal lpData As String, _
           ByVal cbData As Long) As Long

' Return codes from Registration functions.
Const ERROR_SUCCESS = 0&
Const ERROR_BADDB = 1&
Const ERROR_BADKEY = 2&
Const ERROR_CANTOPEN = 3&
Const ERROR_CANTREAD = 4&
Const ERROR_CANTWRITE = 5&
Const ERROR_OUTOFMEMORY = 6&
Const ERROR_INVALID_PARAMETER = 7&
Const ERROR_ACCESS_DENIED = 8&

Private Const HKEY_CLASSES_ROOT = &H80000000
Private Const MAX_PATH = 260&
Private Const REG_SZ = 1

Private Sub Form_Click()

   Dim sKeyName As String   'Holds Key Name in registry.
   Dim sKeyValue As String  'Holds Key Value in registry.
   Dim ret&           'Holds error status if any from API calls.
   Dim lphKey&        'Holds created key handle from RegCreateKey.

   'This creates a Root entry called "MyApp".
      sKeyName = "MyApp"
      sKeyValue = "My Application"
      ret& = RegCreateKey&(HKEY_CLASSES_ROOT, sKeyName, lphKey&)
      ret& = RegSetValue&(lphKey&, "", REG_SZ, sKeyValue, 0&)

   'This creates a Root entry called .BAR associated with "MyApp".
      sKeyName = ".BAR"
      sKeyValue = "MyApp"
      ret& = RegCreateKey&(HKEY_CLASSES_ROOT, sKeyName, lphKey&)
      ret& = RegSetValue&(lphKey&, "", REG_SZ, sKeyValue, 0&)
```

```
    'This sets the command line for "MyApp".
       sKeyName = "MyApp"
       sKeyValue = "c:\mydir\my.exe %1"
       ret& = RegCreateKey&(HKEY_CLASSES_ROOT, sKeyName, lphKey&)
       ret& = RegSetValue&(lphKey&, "shell\open\command", REG_SZ, _
                           sKeyValue, MAX_PATH)
End Sub
```

۳- برنامه را ذخیره کرده و اجرا نمایید. یکبار روی فرم کلیک کرده، سپس از برنامه خارج شوید.

۴- از طریق منوی Start و گزینه Run برنامه REGEDIT را اجرا کنید. می توانید در HKEY_CLASSES_ROOT مقادیر .bar و MyApp را مشاهده نمایید (محل قرار گیری این مقادیر در دیاگرام زیر نشان داده شده است).

```
.bar = MyApp
MyApp = My Application
  |
   -- Shell
        |
         -- open
              |
               -- command = c:\mydir\my.exe %1
```

برای پاک کردن این مقادیر می توانید از گزینه Delete از منوی Edit برنامه RegEdit استفاده کنید.

مراجع :

برای کسب اطلاعات بیشتر در مورد نسبت دادن پسوند فایل به برنامه ویژوال بیسیک ۱۶ بیتی خود (ویژوال بیسیک ۴) می توانید به بخش شماره 147805 مراجعه کنید:

```
Q147805 : HOWTO: Associate a File Extension with Your Application
```

تعیین نوع ویندوز ۳۲ بیتی در حال اجرا

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
189249.TXT	۱۴ می ۱۹۹۹	4 , 5 , 6	95 / 98 / NT

در بعضی موارد لازمست برنامه نوع سیستم عامل در حال اجرا روی کامپیوتر را تشخیص دهد تا برحسب آن بتواند اعمال خاصی را انجام دهد. در این بخش به کمک یک مثال نشان داده می شود که چگونه می توان فرق بین ویندوز **۹۵** و **۹۸** و NT 3.51 و NT 4.0 را تشخیص داد.

تابع GetVersionEx اطلاعاتی را برمی گرداند که به کمک آن می توان سیستم عامل را شناسایی کرد. در میان این اطلاعات می توان شماره نگارش و مشخصه پلاتفرم را یافت. اکنون با ارایه ویندوز **۹۸** کار تشخیص نوع سیستم عامل کمی پیچیده تر شده است. لیست زیر نشان دهنده اطلاعات لازم برای ارزیابی ساختمان داده OSVERSIONINFO است که توسط تابع GetVersionEx برگردانده می شود:

	Win95	Win98	WinNT 3.51	WinNT 4.0
dwPlatFormID	1	1	2	2
dwMajorVersion	4	4	3	4
dwMinorVersion	0	10	51	0

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- از طریق منوی Project یک ماژول به پروژه اضافه کنید.

۳- کد زیر را در Module1 وارد کنید:

```
Public Declare Function GetVersionExA Lib "kernel32" _
   (lpVersionInformation As OSVERSIONINFO) As Integer

Public Type OSVERSIONINFO
   dwOSVersionInfoSize As Long
   dwMajorVersion As Long
   dwMinorVersion As Long
   dwBuildNumber As Long
   dwPlatformId As Long
   szCSDVersion As String * 128
End Type
```

```
Public Function getVersion() As String
   Dim osinfo As OSVERSIONINFO
   Dim retvalue As Integer

   osinfo.dwOSVersionInfoSize = 148
   osinfo.szCSDVersion = Space$(128)
   retvalue = GetVersionExA(osinfo)

   With osinfo
   Select Case .dwPlatformId
      Case 1
         If .dwMinorVersion = 0 Then
            getVersion = "Windows 95"
         ElseIf .dwMinorVersion = 10 Then
            getVersion = "Windows 98"
         End If
      Case 2
         If .dwMajorVersion = 3 Then
            getVersion = "Windows NT 3.51"
         Elseif .dwMajorVersion = 4 Then
            getVersion = "Windows NT 4.0"
         End If
      Case Else
         getVersion = "Failed"
   End Select
   End With
End Function
```

۴- سطر زیر را در واقعه Load از Form1 وارد کنید:

```
MsgBox GetVersion()
```

۵- برنامه را اجرا کنید و به پنجره پیامی که نشان دهنده نگارش ویندوز است توجه کنید.

مراجع :

برای کسب اطلاعات بیشتر به بخش شماره ۹۲۹۳۶ از بانک اطلاعات مایکروسافت مراجعه کنید:

```
Q92936 : HOWTO: Get Windows 3.1 Version Number in VB with GetVersion
```

ایجاد یک کنترل کننده عمومی خطا[۲] برای برنامه

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
191474.TXT	۲۴ می ۱۹۹۹	4 , 5 , 6	95 / 98 / NT

دستور ON ERROR در ویژوال بیسیک به منظور فراهم آوردن امکان کنترل خطا در روالهای مختلف برنامه طراحی شده است. این نحوه عملکرد مطابق با قواعد برنامه سازی شئ گرا است. هنگامی که برنامه ای می نویسید، شاید مایل باشید یک کنترل کننده عمومی خطا داشته باشید تا تمامی خطاهای برنامه را به تله بیاندازد و در صورت لزوم بتوانید در موارد خاص، خطاهایی را بطور انفرادی کنترل نمایید.

از دستور ON ERROR نمی توان بگونه ای استفاده کرد که خطاهای تمامی قسمتهای برنامه را یکباره کنترل نماید. در عوض می توان روتینی برای کنترل خطا ایجاد کرد که قابل فراخوانی از تمامی بخشهای یک برنامه باشد.

در صورتیکه روتین کنترل خطا در یک ماژول قرار داشته باشد، می توان آنرا از هر نقطه ای از برنامه فراخواند. مثال زیر نشان می دهد که چگونه یک روتین کنترل عمومی خطا ایجاد کنید و ضمناً در موارد خاص، کنترل خطا را بطور اختصاصی در داخل یک روال انجام دهید.

این مثال نشان دهنده پروژه ای است که دارای دو فرم و یک ماژول می باشد. هر یک از فرم ها دارای دکمه ای است که باعث ایجاد خطا می شود. بعضی از خطاها در درون واقعه Click مربوط به دکمه، کنترل می شوند. بقیه خطاها برای کنترل کننده عمومی خطا ارسال می شوند.

۱- یک پروژه جدید EXE در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- کد زیر را به واقعه Load از Form1 اضافه کنید. این کد باعث می شود که در هنگام بازیابی فرم دو خطا بوجود بیاید. خطای اول توسط خود روتین Load کنترل می شود. خطای دوم برای کنترل کننده عمومی خطا ارسال می شود:

---

[۲] Generic Error Handler

```
Private Sub Form_Load()
    On Error GoTo FormLoadErr
    Err.Raise 76
    Err.Raise 70
    Exit Sub

    FormLoadErr:
    Select Case Err.Number  'Evaluate Error Number
       Case 76
          MsgBox "Form_Load Error Handler. Form Does Not Exist"
       Case Else
          AppWideErr (Err.Number) 'Pass Error to generic module
    End Select
End Sub
```

۳- یک CommandButton به فرم اضافه کنید. این کد باعث می شود دو خطا بوجود بیاید. خطای اول توسط خود روتین Click کنترل می شود و خطای دوم برای کنترل کننده عمومی خطا ارسال می شود. خصوصیت Caption این دکمه را به Cause Error 53 and 70 تغییر دهید. کد زیر را به واقعه Click مربوط به این دکمه اضافه کنید:

```
Private Sub Command1_Click()
    On Error GoTo Cmd1Err
       Err.Raise 53 'Handled locally
       Err.Raise 70 'Handled by generic module
    Exit Sub

    Cmd1Err:
    Select Case Err.Number
       Case 53
          MsgBox "Command 1 Error Handler"
       Case Else
          AppWideErr (Err.Number)
       End Select
       Resume Next  'Process the next (70) error
End Sub
```

۴- یک CommandButton دیگر به این فرم اضافه کنید. در صورتیکه یک خطا در واقعه Click این دکمه رخ دهد، هیچ کنترل کننده ای برای آن وجود ندارد. خصوصیت Caption این دکمه را به Show Form 2 تغییر دهید. کد زیر را به واقعه Click این دکمه اضافه کنید:

```
Private Sub Command2_Click()
    'No error handling is coded in this method
    'AppWideErr would not be called
    Form2.Show 'Use VB's default error handling only
End Sub
```

۵- یک فرم دیگر به برنامه اضافه کنید و یک CommandButton روی آن قرار دهید. این CommandButton تمامی خطاها را برای کنترل کننده عمومی خطا ارسال می کند و هیچ کنترل خطای ویژه ای در آن صورت نمی گیرد. خصوصیت Caption این دکمه را به Cause Error 17 تغییر دهید. کد زیر را به واقعه Click آن اضافه کنید:

```
Private Sub Command1_Click()
    On Error GoTo ThisSubErr
       Err.Raise 17
       Exit Sub
ThisSubErr:
       AppWideErr (Err.Number)
End Sub
```

٦- یک ماژول به پروژه اضافه کنید. کد زیر را در ماژول وارد نمایید:

```
Public Sub AppWideErr(lnErrNumber)
   Select Case lnErrNumber  'Evaluate error passed to routine
      Case 70
        'The following two lines of code should be typed on one line.
        'Added title to MsgBox to make it clearer where error message
        'came from.
        MsgBox "Generic Routine. Access Denied. See Net Administrator." _
         , , "AppWideErr"
         Exit Sub
      Case Else
         'The following two lines of code should be typed on one line.
         'Added title to MsgBox to make it clearer where error message
         'came from.
         MsgBox "Generic Routine. Unhandled Error: " + Err.Description +_
                " # " & lnErrNumber , , "AppWideErr"
         Exit Sub
    End Select
End Sub
```

۷- پروژه را ذخیره کرده و اجرا کنید. توجه کنید که بعضی از خطاها توسط کنترل کننده هایی در داخل روتین ها کنترل می شوند. در تمامی روتین ها مشخص شده است که بجز خطاهای خاصی که در همان روتین کنترل می شوند ، بقیه خطاها برای روتین AppWideErr ارسال شوند. شماره خطا بصورت یک پارامتر عددی برای روتین AppWideErr ارسال می شود.

مراجع :

Visual Basic 5.0 Books Online - Error Handling

Visual Basic 6.0 MSDN - Error Handling

استخراج آیکون از برنامه های ویندوز

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
185883.TXT	**۲۵** می **۱۹۹۹**	4 , 5 , 6	95 / 98 / NT

مثال زیر نشان می دهد که چگونه می توان آیکون یک برنامه یا فایل DLL **۳۲** بیتی را از آن استخراج کرد.

۱- یک پروژه جدید در ویژوال بیسیک ایجاد نمایید. Form1 بطور پیش فرض بوجود می آید.

۲- یک Command Button و یک Picture Box روی Form1 قرار دهید. اندازه Picture1 را طوری تنظیم کنید که بتواند یک آیکون **۳۲×۳۲** را نمایش دهد.

۳- کد زیر را در بخش Declarations از Form1 وارد نمایید:

```
Option Explicit
Private Const MAX_PATH = 260

Private Declare Function GetSystemDirectory Lib "kernel32" Alias _
    "GetSystemDirectoryA" (ByVal lpBuffer As String, _
                           ByVal nSize As Long) As Long

Private Declare Function ExtractIcon Lib "shell32.dll" Alias _
    "ExtractIconA" (ByVal hInst As Long, _
                    ByVal lpszExeFileName As String, _
                    ByVal nIconIndex As Long) As Long

Private Declare Function DrawIcon Lib "user32" (ByVal hdc As Long, _
    ByVal x As Long, ByVal y As Long, ByVal hIcon As Long) As Long

Dim path$, nIcon As Long

Private Sub Command1_Click()
     Dim hIcon As Long
     hIcon = ExtractIcon(App.hInstance, path$, nIcon)
     Set Picture1.Picture = LoadPicture("") ' Clear the picture box
     Picture1.AutoRedraw = True
     Call DrawIcon(Picture1.hdc, 0, 0, hIcon)
     Picture1.AutoRedraw = False
     Picture1.Refresh
     nIcon = nIcon + 1
End Sub

Private Sub Form_Load()
  ' Store the full path to the file containing the icon(s).
     path$ = Space$(MAX_PATH)
     Call GetSystemDirectory(path$, MAX_PATH)
     path$ = Trim$(path$) ' Trim trailing blanks & Null terminator
     path$ = Left$(path$, Len(path$) - 1) & "\Shell32.dll"
```

```
        nIcon = 0
End Sub
```

برای اجرای برنامه کلید F5 را فشار دهید. روی Command1 کلیک کنید تا اولین آیکون ذخیره شده در فایل Shell32.dll را توسط Picture1 نمایش دهد. هر بار کلیک مجدد روی Command1 باعث می شود یکی دیگر از آیکونهای این فایل نمایش داده شود.