免费电子书

学习

VBA

Free unaffiliated eBook created from
Stack Overflow contributors.

#vba

1
1: VBA 2
2
2
Examples 2
Microsoft OfficeVisual Basic 2
Hello World 4
4
4
5
5
5
2: API 6
6
6
Examples 7
API 7
Windows API - 1/2 9
Windows API - 2/2 13
Mac API 17
18
FTPAPI 18
3: CreateObjectGetObject 23
23
Examples 23
GetObjectCreateObject 23
4: Scripting.Dictionary 25
25
Examples 25
25
Scripting.Dictionary 26

Scripting.Dictionary ... 28
5: Scripting.FileSystemObject ... 30
Examples ... 30
FileSystemObject ... 30
FileSystemObject ... 30
FileSystemObject ... 31
FileSystemObject ... 31
FileSystemObject ... 31
... 32
... 33
... 33
... 33
FSO.BuildPath ... 34
6: VBA ... 35
... 35
Examples ... 35
'3'GoSub ... 35
... 35
... 35
... 35
... 35
... 35
'6' ... 35
... 35
... 35
... 35
... 36
... 36
'9' ... 36
... 36
... 36
... 36
...

36
36
'13' 36
36
36
36
36
37
'91' 37
37
37
37
37
37
'20' 37
37
38
38
38
38
7: VBA 39
39
39
39
Examples 39
39
{Binary ||} 40
40
41
41
{0 | 1} 41
0 41
1 42

Base 1 42
8: ByRefByVal 44
44
44
44
Examples 44
ByRefByVal 44
ByRef 45
45
45
ByVal 46
BYVAL 46
46
9: ADO 48
48
Examples 48
48
48
49
50
10: 52
52
Examples 52
Stringn 52
StringSpacen 52
11: 53
Examples 53
53
53
53
12: 55

55

Examples....55

....55

....56

....56

13: 58

Examples....58

....58

....58

....60

14: VBA2GB +....62

....62

....62

MICROSOFT....62

MICROSOFT....62

....62

Examples....62

Class“Random”....63

....66

....68

....68

....68

15: FileSystemObject....72

....72

Examples....72

....72

....72

16: 74

....74

Examples....74

InStr....74

InStr 74
InStrRev 74
17: 75
Examples 75
75
75
75
75
76
77
77
77
Const 78
78
79
79
80
81
81
18: 84
84
Examples 84
84
84
84
84
Mid 84
85
19: 86
Examples 86
86
86

87

87

87

88

Class 88

88

20: 90

90

Examples 90

LeftLeft $3 90

RightRight $3 90

MidMid $ 90

Trim 90

21: - 91

91

Examples 91

“ 91

91

VBA 91

22: VBA/ 93

Examples 93

SELFCERT.EXE 93

23: 106

106

Examples 106

CStr 106

Format 106

StrConv 106

106

24: 108

108

Examples 108

VB_Name 108
VB_GlobalNameSpace 108
VB_Createable 108
VB_PredeclaredId 109
..... 109
..... 109
VB_Exposed 109
VB_Description 110
VB_ [] UserMemId 110
..... 110
For Each 111
25: 112
..... 112
Examples 112
- 112
Excel 112
26: 115
..... 115
Examples 115
- 115
- 116
27: 118
Examples 118
..... 118
..... 118
..... 119
..... 119
..... 119
..... 119
..... 120
..... 120
..... 120
.....

120

120

120

121

LongPtr 121

122

28: 123

123

Examples 123

123

124

29: 125

Examples 125

VBA 125

125

125

125

125

Split 125

127

... 127

..... 127

128

128

128

128

128

129

129

129

129

131

131
132
134
30: 137
Examples 137
137
137
137
137
138
IsDate 138
139
DatePart 140
141
DateDiff 141
DateAdd 141
142
CDATE 142
DateSerial 142
31: 144
Examples 144
144
OfficeDeclare Imports 145
32: 147
147
147
Examples 147
147
147
147
148

148

149

33: 151

Examples 151

151

152

152

153

153

154

34: 155

155

Examples 155

Len 155

LenB 155

`LenmyString= 0```myString =“”` 155

35: 156

Examples 156

156

156

156

156

QueryClose 157

157

157

QueryClose 158

158

36: 160

160

160

160

Examples....160

....160

....160

....160

....160

....161

....161

....161

....161

37: 163

....163

Examples....163

....163

38: 165

....165

....165

....165

Examples....165

VBScript....165

....165

....166

....167

Internet Explorer....167

Internet Explorer Objec....167

....168

....169

Microsoft HTMLIE....169

IE....169

39: 171

....171

Examples 171
171
REM 171
40: 173
173
Examples 173
173
173
174
174
\ 175
41: 178
178
Examples 178
178
Join 178
42: 179
179
179
Examples 179
179
179
43: 181
Examples 181
181
181
182
183
183
On Error Resume Next 183
184

184

44: 186

..... 186

..... 186

Examples 186

..... 186

..... 187

..... 187

..... 188

..... 189

..... 189

..... 189

..... 190

45: 191

..... 191

Examples 191

VBA 191

..... 192

46: VBA 193

Examples 193

..... 193

..... 193

..... 193

..... 193

..... 194

..... 196

..... 196

..... 196

..... 198

..... 199

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vba

It is an unofficial and free VBA ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official VBA.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: VBA

vba。

vba。 vba。

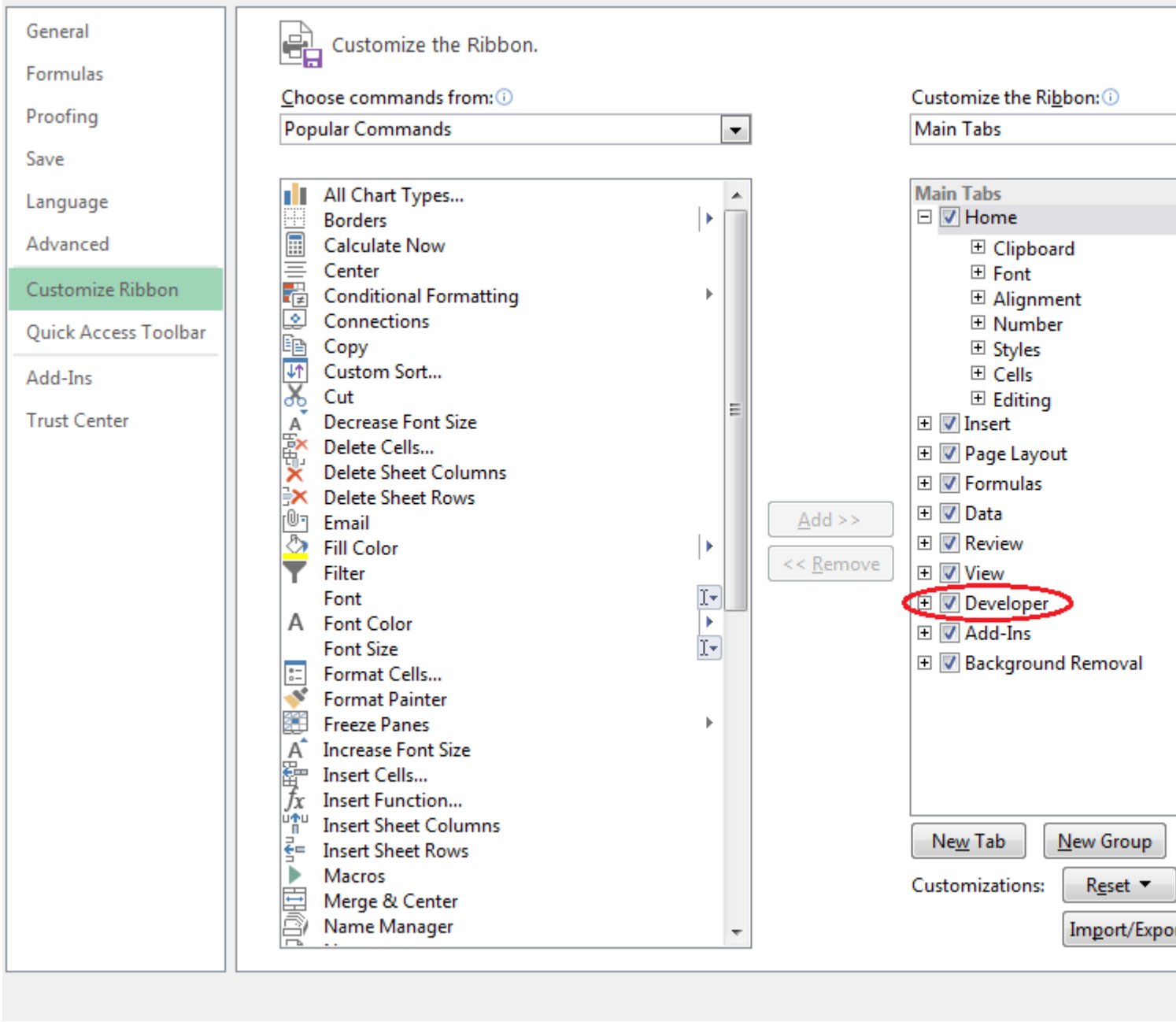
	Office		
VBA6	- 2007	[] [1]	1992630
VBA7	2010 - 2016	[blog.techkit.com] [2]	2010-04-15
MacVBA	20042011 - 2016		2004-05-11

Examples

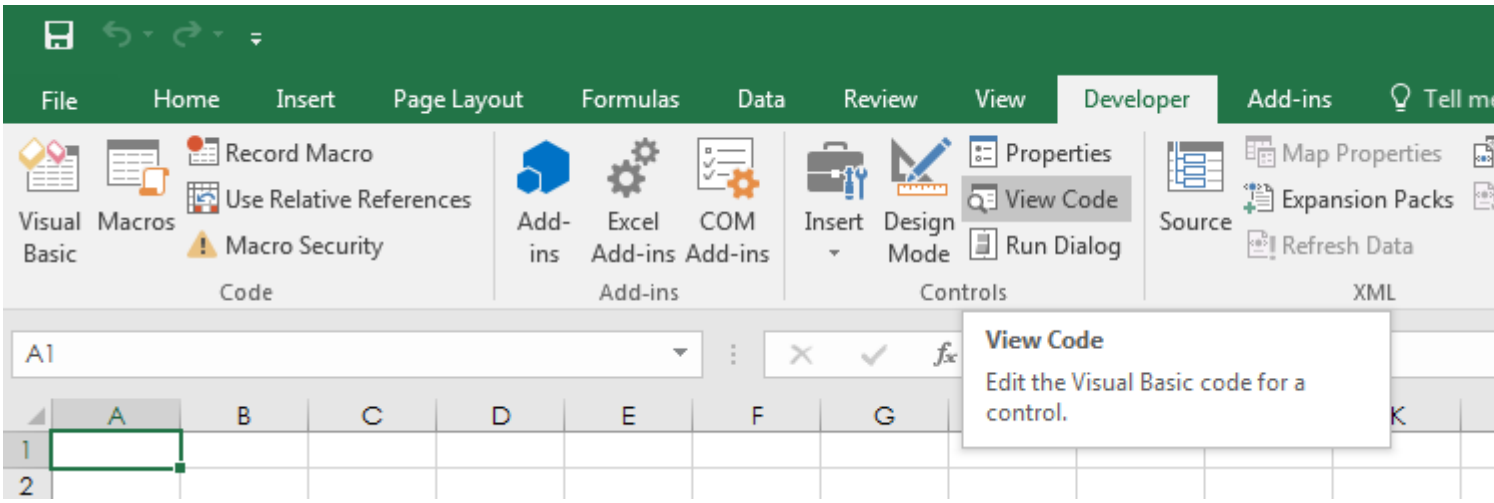
Microsoft OfficeVisual Basic

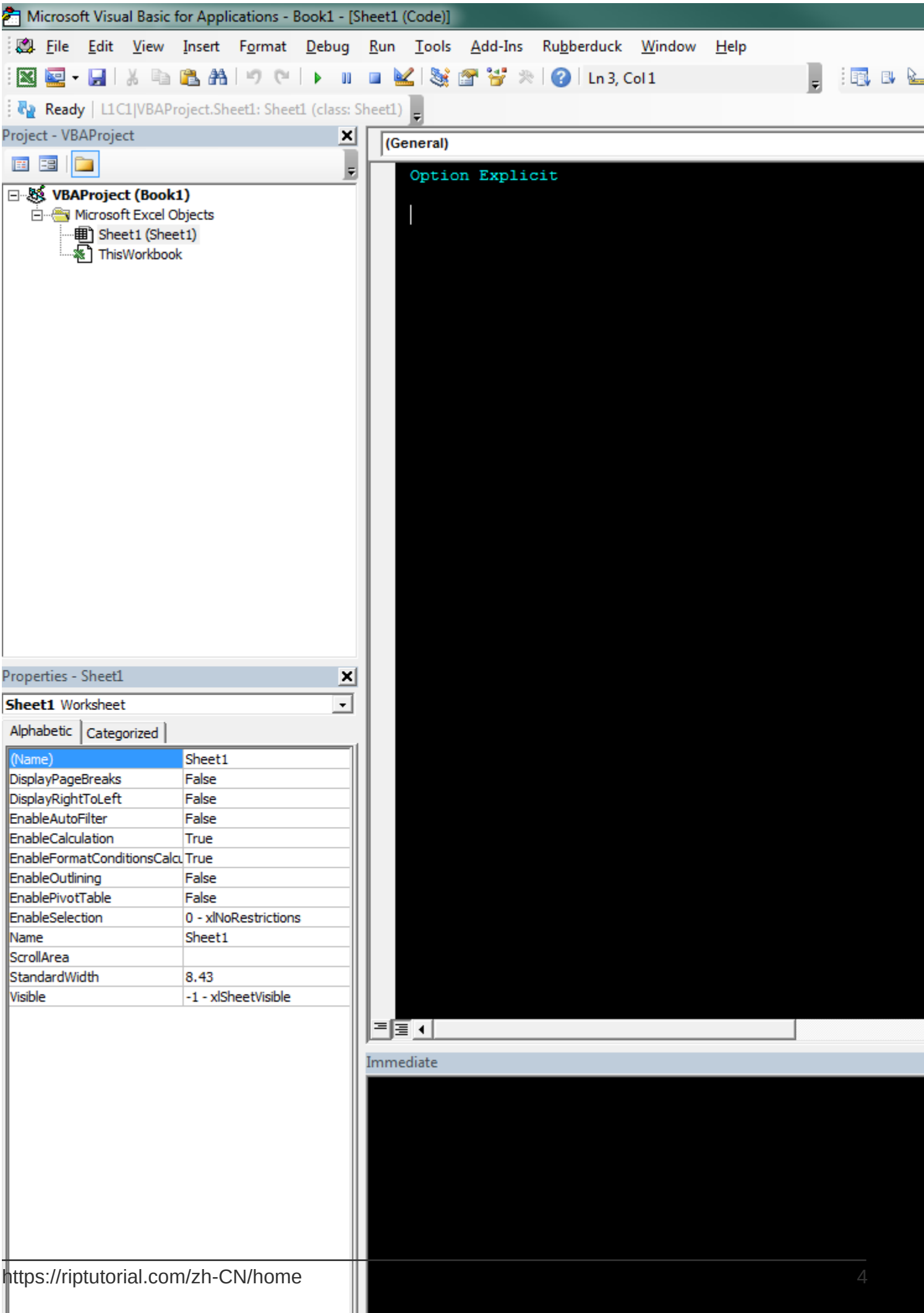
`Alt` + `F11`“”“Visual Basic”Microsoft OfficeVB。“”。

Developer。“”“” - >“”“”。“”“”“”。“”“”。



“”“Visual Basic”Visual Basic。“”WorkSheetChartShape。




Microsoft Visual Basic for Applications - Book1 - [Sheet1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Rubberduck Window Help
Ln 3, Col 1
Ready
L1C1|VBAProject.Sheet1: Sheet1 (class: Sheet1)
Project - VBAProject
VBAProject (Book1)
Microsoft Excel Objects
Sheet1 (Sheet1)
ThisWorkbook
(General)
Option Explicit
Properties - Sheet1
Sheet1 Worksheet
Alphabetic
Categorized
(Name) Sheet1
DisplayPageBreaks False
DisplayRightToLeft False
EnableAutoFilter False
EnableCalculation True
EnableFormatConditionsCalcu True
EnableOutlining False
EnablePivotTable False
EnableSelection 0 - xlNoRestrictions
Name Sheet1
ScrollArea
StandardWidth 8.43
Visible -1 - xlSheetVisible
Immediate

- Stop。 F9
- F8 - /。
- Shift + F8 - /。
- Ctrl + Shift + F8 - /。
- Ctrl + F8 - 。
- Debug.Print。 Debug.?Debug.Print

View - Watch。

- “”。
- “”。
- - 。

。

Print“ ? ”。

- ? ActiveSheet.Name -
- Print ActiveSheet.Name -
- ? foo -foo *
- x = 10x10 *

*/

。

;。

。

-
- 。 Sheets(a*b*c+d^2).Range(addressOfRange)Sheets(4).Range("A2")
- .....

VBA https://riptutorial.com/zh-CN/vba/topic/802/vba

2: API

API

VBAAPI

DLL

DLL

ADVAPI32.DLL	API
COMDLG32.DLL	API
GDI32.DLL	API
KERNEL32.DLL	Windows 32API
Lz32.dll	32
MPR.DLL	
NETAPI32.DLL	32API
SHELL32.DLL	32Shell API
user32.dll	
VERSION.DLL	
WINMM.DLL	Windows
WINSPOOL.DRV	API

64

	PTRSAFE	Declare64。64
	LongPtr	32464Office 20108。64Office 2010。3264VBA 7。
		864Office 2010。
		CLngPtrLongPtr
		CLngLngLongLong
	VarPtr	。64LongPtr32Long4

	ObjPtr	。64LongPtr32Long4
	StrPtr	64LongPtr32Long4

- Visual Basic 5.0Win32api32.txt API20053Microsoft
- 64Win32API_PtrSafe Office 2010Microsoft

Examples

API

DLLVBA

```
Option Explicit

#If Win64 Then

    Private Declare PtrSafe Sub xLib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

#ElseIf Win32 Then

    Private Declare Sub apiSleep Lib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

#End If
```

VBAKernel32.dll“Sleep”

Win64Win32

。VBA。Vba7Office 201064Office。

	16	32	64
VBA6		Vba6	
VBA7		Vba7	
Win16			
Win32			
Win64			
		Mac	Mac

OfficeWindows。32OfficeWin32 = TRUE64Windows。

API3264Office“”

-
-
- `Private`
- DLL

Sleep API

```
Public Sub TestPause()

    Dim start As Double

    start = Timer

    Sleep 9000      'Pause execution for 9 seconds

    Debug.Print "Paused for " & Format(Timer - start, "#,###.000") & " seconds"

    'Immediate window result: Paused for 9.000 seconds

End Sub
```

APIVBA - VBA Subs

-
-

DLL`Declare`Declarations。

Function

```
Declare Function publicname Lib "libname" [Alias "alias"] [([[ByVal] variable [As type]
[,[ByVal] variable [As type]]...])] As Type
```

Sub

```
Declare Sub publicname Lib "libname" [Alias "alias"] [([[ByVal] variable [As type] [,[ByVal]
variable [As type]]...])]
```

- !!! !! !!!!!!!!!!!!!!!!!!!!!!!!!!

  ### APIExcel

- !!! !! !!!!!!!!!!!!!!!!!!!!!!!!!!

Office 2011 for Mac

```
Private Declare Function system Lib "libc.dylib" (ByVal command As String) As Long

Sub RunSafari()
    Dim result As Long
    result = system("open -a Safari --args http://www.google.com")
    Debug.Print Str(result)
End Sub
```

Windows API - 12APIWin64Win32

Windows API - 1/2

```
Option Explicit

#If Win64 Then  'Win64 = True, Win32 = False, Win16 = False
    Private Declare PtrSafe Sub apiCopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (MyDest As
Any, MySource As Any, ByVal MySize As Long)
    Private Declare PtrSafe Sub apiExitProcess Lib "Kernel32" Alias "ExitProcess" (ByVal
uExitCode As Long)
    Private Declare PtrSafe Sub apiSetCursorPos Lib "User32" Alias "SetCursorPos" (ByVal X As
Integer, ByVal Y As Integer)
    Private Declare PtrSafe Sub apiSleep Lib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds As
Long)
    Private Declare PtrSafe Function apiAttachThreadInput Lib "User32" Alias
"AttachThreadInput" (ByVal idAttach As Long, ByVal idAttachTo As Long, ByVal fAttach As Long)
As Long
    Private Declare PtrSafe Function apiBringWindowToTop Lib "User32" Alias "BringWindowToTop"
(ByVal lngHWnd As Long) As Long
    Private Declare PtrSafe Function apiCloseWindow Lib "User32" Alias "CloseWindow" (ByVal
hWnd As Long) As Long
    Private Declare PtrSafe Function apiDestroyWindow Lib "User32" Alias "DestroyWindow"
(ByVal hWnd As Long) As Boolean
    Private Declare PtrSafe Function apiEndDialog Lib "User32" Alias "EndDialog" (ByVal hWnd
As Long, ByVal result As Long) As Boolean
    Private Declare PtrSafe Function apiEnumChildWindows Lib "User32" Alias "EnumChildWindows"
(ByVal hWndParent As Long, ByVal pEnumProc As Long, ByVal lParam As Long) As Long
    Private Declare PtrSafe Function apiExitWindowsEx Lib "User32" Alias "ExitWindowsEx"
(ByVal uFlags As Long, ByVal dwReserved As Long) As Long
    Private Declare PtrSafe Function apiFindExecutable Lib "Shell32" Alias "FindExecutableA"
(ByVal lpFile As String, ByVallpDirectory As String, ByVal lpResult As String) As Long
    Private Declare PtrSafe Function apiFindWindow Lib "User32" Alias "FindWindowA" (ByVal
lpClassName As String, ByVal lpWindowName As String) As Long
    Private Declare PtrSafe Function apiFindWindowEx Lib "User32" Alias "FindWindowExA" (ByVal
hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long
    Private Declare PtrSafe Function apiGetActiveWindow Lib "User32" Alias "GetActiveWindow"
() As Long
    Private Declare PtrSafe Function apiGetClassNameA Lib "User32" Alias "GetClassNameA"
(ByVal hWnd As Long, ByVal szClassName As String, ByVal lLength As Long) As Long
    Private Declare PtrSafe Function apiGetCommandLine Lib "Kernel32" Alias "GetCommandLineW"
() As Long
    Private Declare PtrSafe Function apiGetCommandLineParams Lib "Kernel32" Alias
"GetCommandLineA" () As Long
    Private Declare PtrSafe Function apiGetDiskFreeSpaceEx Lib "Kernel32" Alias
"GetDiskFreeSpaceExA" (ByVal lpDirectoryName As String, lpFreeBytesAvailableToCaller As
Currency, lpTotalNumberOfBytes As Currency, lpTotalNumberOfFreeBytes As Currency) As Long
    Private Declare PtrSafe Function apiGetDriveType Lib "Kernel32" Alias "GetDriveTypeA"
(ByVal nDrive As String) As Long
    Private Declare PtrSafe Function apiGetExitCodeProcess Lib "Kernel32" Alias
```

```
"GetExitCodeProcess" (ByVal hProcess As Long, lpExitCode As Long) As Long
    Private Declare PtrSafe Function apiGetForegroundWindow Lib "User32" Alias
"GetForegroundWindow" () As Long
    Private Declare PtrSafe Function apiGetFrequency Lib "Kernel32" Alias
"QueryPerformanceFrequency" (cyFrequency As Currency) As Long
    Private Declare PtrSafe Function apiGetLastError Lib "Kernel32" Alias "GetLastError" () As
Integer
    Private Declare PtrSafe Function apiGetParent Lib "User32" Alias "GetParent" (ByVal hWnd
As Long) As Long
    Private Declare PtrSafe Function apiGetSystemMetrics Lib "User32" Alias "GetSystemMetrics"
(ByVal nIndex As Long) As Long
    Private Declare PtrSafe Function apiGetSystemMetrics32 Lib "User32" Alias
"GetSystemMetrics" (ByVal nIndex As Long) As Long
    Private Declare PtrSafe Function apiGetTickCount Lib "Kernel32" Alias
"QueryPerformanceCounter" (cyTickCount As Currency) As Long
    Private Declare PtrSafe Function apiGetTickCountMs Lib "Kernel32" Alias "GetTickCount" ()
As Long
    Private Declare PtrSafe Function apiGetUserName Lib "AdvApi32" Alias "GetUserNameA" (ByVal
lpBuffer As String, nSize As Long) As Long
    Private Declare PtrSafe Function apiGetWindow Lib "User32" Alias "GetWindow" (ByVal hWnd
As Long, ByVal wCmd As Long) As Long
    Private Declare PtrSafe Function apiGetWindowRect Lib "User32" Alias "GetWindowRect"
(ByVal hWnd As Long, lpRect As winRect) As Long
    Private Declare PtrSafe Function apiGetWindowText Lib "User32" Alias "GetWindowTextA"
(ByVal hWnd As Long, ByVal szWindowText As String, ByVal lLength As Long) As Long
    Private Declare PtrSafe Function apiGetWindowThreadProcessId Lib "User32" Alias
"GetWindowThreadProcessId" (ByVal hWnd As Long, lpdwProcessId As Long) As Long
    Private Declare PtrSafe Function apiIsCharAlphaNumericA Lib "User32" Alias
"IsCharAlphaNumericA" (ByVal byChar As Byte) As Long
    Private Declare PtrSafe Function apiIsIconic Lib "User32" Alias "IsIconic" (ByVal hWnd As
Long) As Long
    Private Declare PtrSafe Function apiIsWindowVisible Lib "User32" Alias "IsWindowVisible"
(ByVal hWnd As Long) As Long
    Private Declare PtrSafe Function apiIsZoomed Lib "User32" Alias "IsZoomed" (ByVal hWnd As
Long) As Long
    Private Declare PtrSafe Function apiLStrCpynA Lib "Kernel32" Alias "lstrcpynA" (ByVal
pDestination As String, ByVal pSource As Long, ByVal iMaxLength As Integer) As Long
    Private Declare PtrSafe Function apiMessageBox Lib "User32" Alias "MessageBoxA" (ByVal
hWnd As Long, ByVal lpText As String, ByVal lpCaption As String, ByVal wType As Long) As Long
    Private Declare PtrSafe Function apiOpenIcon Lib "User32" Alias "OpenIcon" (ByVal hWnd As
Long) As Long
    Private Declare PtrSafe Function apiOpenProcess Lib "Kernel32" Alias "OpenProcess" (ByVal
dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long
    Private Declare PtrSafe Function apiPathAddBackslashByPointer Lib "ShlwApi" Alias
"PathAddBackslashW" (ByVal lpszPath As Long) As Long
    Private Declare PtrSafe Function apiPathAddBackslashByString Lib "ShlwApi" Alias
"PathAddBackslashW" (ByVal lpszPath As String) As Long 'http://msdn.microsoft.com/en-
us/library/aa155716%28office.10%29.aspx
    Private Declare PtrSafe Function apiPostMessage Lib "User32" Alias "PostMessageA" (ByVal
hWnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
    Private Declare PtrSafe Function apiRegQueryValue Lib "AdvApi32" Alias "RegQueryValue"
(ByVal hKey As Long, ByVal sValueName As String, ByVal dwReserved As Long, ByRef lValueType As
Long, ByVal sValue As String, ByRef lResultLen As Long) As Long
    Private Declare PtrSafe Function apiSendMessage Lib "User32" Alias "SendMessageA" (ByVal
hWnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
    Private Declare PtrSafe Function apiSetActiveWindow Lib "User32" Alias "SetActiveWindow"
(ByVal hWnd As Long) As Long
    Private Declare PtrSafe Function apiSetCurrentDirectoryA Lib "Kernel32" Alias
"SetCurrentDirectoryA" (ByVal lpPathName As String) As Long
    Private Declare PtrSafe Function apiSetFocus Lib "User32" Alias "SetFocus" (ByVal hWnd As
Long) As Long
```

```
    Private Declare PtrSafe Function apiSetForegroundWindow Lib "User32" Alias
"SetForegroundWindow" (ByVal hWnd As Long) As Long
    Private Declare PtrSafe Function apiSetLocalTime Lib "Kernel32" Alias "SetLocalTime"
(lpSystem As SystemTime) As Long
    Private Declare PtrSafe Function apiSetWindowPlacement Lib "User32" Alias
"SetWindowPlacement" (ByVal hWnd As Long, ByRef lpwndpl As winPlacement) As Long
    Private Declare PtrSafe Function apiSetWindowPos Lib "User32" Alias "SetWindowPos" (ByVal
hWnd As Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As
Long, ByVal cy As Long, ByVal wFlags As Long) As Long
    Private Declare PtrSafe Function apiSetWindowText Lib "User32" Alias "SetWindowTextA"
(ByVal hWnd As Long, ByVal lpString As String) As Long
    Private Declare PtrSafe Function apiShellExecute Lib "Shell32" 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 PtrSafe Function apiShowWindow Lib "User32" Alias "ShowWindow" (ByVal hWnd
As Long, ByVal nCmdShow As Long) As Long
    Private Declare PtrSafe Function apiShowWindowAsync Lib "User32" Alias "ShowWindowAsync"
(ByVal hWnd As Long, ByVal nCmdShow As Long) As Long
    Private Declare PtrSafe Function apiStrCpy Lib "Kernel32" Alias "lstrcpynA" (ByVal
pDestination As String, ByVal pSource As String, ByVal iMaxLength As Integer) As Long
    Private Declare PtrSafe Function apiStringLen Lib "Kernel32" Alias "lstrlenW" (ByVal
lpString As Long) As Long
    Private Declare PtrSafe Function apiStrTrimW Lib "ShlwApi" Alias "StrTrimW" () As Boolean
    Private Declare PtrSafe Function apiTerminateProcess Lib "Kernel32" Alias
"TerminateProcess" (ByVal hWnd As Long, ByVal uExitCode As Long) As Long
    Private Declare PtrSafe Function apiTimeGetTime Lib "Winmm" Alias "timeGetTime" () As Long
    Private Declare PtrSafe Function apiVarPtrArray Lib "MsVbVm50" Alias "VarPtr" (Var() As
Any) As Long
    Private Type browseInfo     'used by apiBrowseForFolder
        hOwner As Long
        pidlRoot As Long
        pszDisplayName As String
        lpszTitle As String
        ulFlags As Long
        lpfn As Long
        lParam As Long
        iImage As Long
    End Type
    Private Declare PtrSafe Function apiBrowseForFolder Lib "Shell32" Alias
"SHBrowseForFolderA" (lpBrowseInfo As browseInfo) As Long
    Private Type CHOOSECOLOR    'used by apiChooseColor;
http://support.microsoft.com/kb/153929 and http://www.cpearson.com/Excel/Colors.aspx
        lStructSize As Long
        hWndOwner As Long
        hInstance As Long
        rgbResult As Long
        lpCustColors As String
        flags As Long
        lCustData As Long
        lpfnHook As Long
        lpTemplateName As String
    End Type
    Private Declare PtrSafe Function apiChooseColor Lib "ComDlg32" Alias "ChooseColorA"
(pChoosecolor As CHOOSECOLOR) As Long
    Private Type FindWindowParameters   'Custom structure for passing in the parameters in/out
of the hook enumeration function; could use global variables instead, but this is nicer
        strTitle As String  'INPUT
        hWnd As Long        'OUTPUT
    End Type                            'Find a specific window with dynamic caption from a
list of all open windows: http://www.everythingaccess.com/tutorials.asp?ID=Bring-an-external-
application-window-to-the-foreground
```

```
    Private Declare PtrSafe Function apiEnumWindows Lib "User32" Alias "EnumWindows" (ByVal
lpEnumFunc As LongPtr, ByVal lParam As LongPtr) As Long
    Private Type lastInputInfo  'used by apiGetLastInputInfo, getLastInputTime
        cbSize As Long
        dwTime As Long
    End Type
    Private Declare PtrSafe Function apiGetLastInputInfo Lib "User32" Alias "GetLastInputInfo"
(ByRef plii As lastInputInfo) As Long
    'http://www.pgacon.com/visualbasic.htm#Take%20Advantage%20of%20Conditional%20Compilation
    'Logical and Bitwise Operators in Visual Basic: http://msdn.microsoft.com/en-
us/library/wz3k228a(v=vs.80).aspx and http://stackoverflow.com/questions/1070863/hidden-
features-of-vba
    Private Type SystemTime
          wYear          As Integer
          wMonth         As Integer
          wDayOfWeek     As Integer
          wDay           As Integer
          wHour          As Integer
          wMinute        As Integer
          wSecond        As Integer
          wMilliseconds  As Integer
    End Type
    Private Declare PtrSafe Sub apiGetLocalTime Lib "Kernel32" Alias "GetLocalTime" (lpSystem
As SystemTime)
    Private Type pointAPI       'used by apiSetWindowPlacement
         X As Long
         Y As Long
    End Type
    Private Type rectAPI       'used by apiSetWindowPlacement
        Left_Renamed As Long
        Top_Renamed As Long
        Right_Renamed As Long
        Bottom_Renamed As Long
    End Type
    Private Type winPlacement   'used by apiSetWindowPlacement
        length As Long
        flags As Long
        showCmd As Long
        ptMinPosition As pointAPI
        ptMaxPosition As pointAPI
        rcNormalPosition As rectAPI
    End Type
    Private Declare PtrSafe Function apiGetWindowPlacement Lib "User32" Alias
"GetWindowPlacement" (ByVal hWnd As Long, ByRef lpwndpl As winPlacement) As Long
    Private Type winRect     'used by apiMoveWindow
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
    End Type
    Private Declare PtrSafe Function apiMoveWindow Lib "User32" Alias "MoveWindow" (ByVal hWnd
As Long, xLeft As Long, ByVal yTop As Long, wWidth As Long, ByVal hHeight As Long, ByVal
repaint As Long) As Long

    Private Declare PtrSafe Function apiInternetOpen Lib "WiniNet" Alias "InternetOpenA"
(ByVal sAgent As String, ByVal lAccessType As Long, ByVal sProxyName As String, ByVal
sProxyBypass As String, ByVal lFlags As Long) As Long    'Open the Internet object    'ex:
lngINet = InternetOpen("MyFTP Control", 1, vbNullString, vbNullString, 0)
    Private Declare PtrSafe Function apiInternetConnect Lib "WiniNet" Alias "InternetConnectA"
(ByVal hInternetSession As Long, ByVal sServerName As String, ByVal nServerPort As Integer,
ByVal sUsername As String, ByVal sPassword As String, ByVal lService As Long, ByVal lFlags As
```

```
Long, ByVal lContext As Long) As Long  'Connect to the network  'ex: lngINetConn =
InternetConnect(lngINet, "ftp.microsoft.com", 0, "anonymous", "wally@wallyworld.com", 1, 0, 0)
    Private Declare PtrSafe Function apiFtpGetFile Lib "WiniNet" Alias "FtpGetFileA" (ByVal
hFtpSession As Long, ByVal lpszRemoteFile As String, ByVal lpszNewFile As String, ByVal
fFailIfExists As Boolean, ByVal dwFlagsAndAttributes As Long, ByVal dwFlags As Long, ByVal
dwContext As Long) As Boolean    'Get a file 'ex: blnRC = FtpGetFile(lngINetConn,
"dirmap.txt", "c:\dirmap.txt", 0, 0, 1, 0)
    Private Declare PtrSafe Function apiFtpPutFile Lib "WiniNet" Alias "FtpPutFileA" (ByVal
hFtpSession As Long, ByVal lpszLocalFile As String, ByVal lpszRemoteFile As String, ByVal
dwFlags As Long, ByVal dwContext As Long) As Boolean  'Send a file  'ex: blnRC =
FtpPutFile(lngINetConn, "c:\dirmap.txt", "dirmap.txt", 1, 0)
    Private Declare PtrSafe Function apiFtpDeleteFile Lib "WiniNet" Alias "FtpDeleteFileA"
(ByVal hFtpSession As Long, ByVal lpszFileName As String) As Boolean 'Delete a file 'ex: blnRC
= FtpDeleteFile(lngINetConn, "test.txt")
    Private Declare PtrSafe Function apiInternetCloseHandle Lib "WiniNet" (ByVal hInet As
Long) As Integer  'Close the Internet object  'ex: InternetCloseHandle lngINetConn    'ex:
InternetCloseHandle lngINet
    Private Declare PtrSafe Function apiFtpFindFirstFile Lib "WiniNet" Alias
"FtpFindFirstFileA" (ByVal hFtpSession As Long, ByVal lpszSearchFile As String, lpFindFileData
As WIN32_FIND_DATA, ByVal dwFlags As Long, ByVal dwContent As Long) As Long
    Private Type FILETIME
        dwLowDateTime As Long
        dwHighDateTime As Long
    End Type
    Private Type WIN32_FIND_DATA
        dwFileAttributes As Long
        ftCreationTime As FILETIME
        ftLastAccessTime As FILETIME
        ftLastWriteTime As FILETIME
        nFileSizeHigh As Long
        nFileSizeLow As Long
        dwReserved0 As Long
        dwReserved1 As Long
        cFileName As String * 1 'MAX_FTP_PATH
        cAlternate As String * 14
    End Type    'ex: lngHINet = FtpFindFirstFile(lngINetConn, "*.*", pData, 0, 0)
    Private Declare PtrSafe Function apiInternetFindNextFile Lib "WiniNet" Alias
"InternetFindNextFileA" (ByVal hFind As Long, lpvFindData As WIN32_FIND_DATA) As Long  'ex:
blnRC = InternetFindNextFile(lngHINet, pData)
#ElseIf Win32 Then  'Win32 = True, Win16 = False
```

Windows API - 2/2

```
#ElseIf Win32 Then  'Win32 = True, Win16 = False
    Private Declare Sub apiCopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (MyDest As Any,
MySource As Any, ByVal MySize As Long)
    Private Declare Sub apiExitProcess Lib "Kernel32" Alias "ExitProcess" (ByVal uExitCode As
Long)
    'Private Declare Sub apiGetStartupInfo Lib "Kernel32" Alias "GetStartupInfoA"
(lpStartupInfo As STARTUPINFO)
    Private Declare Sub apiSetCursorPos Lib "User32" Alias "SetCursorPos" (ByVal X As Integer,
ByVal Y As Integer)  'Logical and Bitwise Operators in Visual Basic:
http://msdn.microsoft.com/en-us/library/wz3k228a(v=vs.80).aspx and
http://stackoverflow.com/questions/1070863/hidden-features-of-vba
'http://www.pgacon.com/visualbasic.htm#Take%20Advantage%20of%20Conditional%20Compilation
    Private Declare Sub apiSleep Lib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)
    Private Declare Function apiAttachThreadInput Lib "User32" Alias "AttachThreadInput"
(ByVal idAttach As Long, ByVal idAttachTo As Long, ByVal fAttach As Long) As Long
    Private Declare Function apiBringWindowToTop Lib "User32" Alias "BringWindowToTop" (ByVal
```

```
lngHWnd As Long) As Long
    Private Declare Function apiCloseHandle Lib "Kernel32" (ByVal hObject As Long) As Long
    Private Declare Function apiCloseWindow Lib "User32" Alias "CloseWindow" (ByVal hWnd As
Long) As Long
    'Private Declare Function apiCreatePipe Lib "Kernel32" (phReadPipe As Long, phWritePipe As
Long, lpPipeAttributes As SECURITY_ATTRIBUTES, ByVal nSize As Long) As Long
    'Private Declare Function apiCreateProcess Lib "Kernel32" Alias "CreateProcessA" (ByVal
lpApplicationName As Long, ByVal lpCommandLine As String, lpProcessAttributes As Any,
lpThreadAttributes As Any, ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long,
lpEnvironment As Any, ByVal lpCurrentDriectory As String, lpStartupInfo As STARTUPINFO,
lpProcessInformation As PROCESS_INFORMATION) As Long
    Private Declare Function apiDestroyWindow Lib "User32" Alias "DestroyWindow" (ByVal hWnd
As Long) As Boolean
    Private Declare Function apiEndDialog Lib "User32" Alias "EndDialog" (ByVal hWnd As Long,
ByVal result As Long) As Boolean
    Private Declare Function apiEnumChildWindows Lib "User32" Alias "EnumChildWindows" (ByVal
hWndParent As Long, ByVal pEnumProc As Long, ByVal lParam As Long) As Long
    Private Declare Function apiExitWindowsEx Lib "User32" Alias "ExitWindowsEx" (ByVal uFlags
As Long, ByVal dwReserved As Long) As Long
    Private Declare Function apiFindExecutable Lib "Shell32" Alias "FindExecutableA" (ByVal
lpFile As String, ByVallpDirectory As String, ByVal lpResult As String) As Long
    Private Declare Function apiFindWindow Lib "User32" Alias "FindWindowA" (ByVal lpClassName
As String, ByVal lpWindowName As String) As Long
    Private Declare Function apiFindWindowEx Lib "User32" Alias "FindWindowExA" (ByVal hWnd1
As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long
    Private Declare Function apiGetActiveWindow Lib "User32" Alias "GetActiveWindow" () As
Long
    Private Declare Function apiGetClassNameA Lib "User32" Alias "GetClassNameA" (ByVal hWnd
As Long, ByVal szClassName As String, ByVal lLength As Long) As Long
    Private Declare Function apiGetCommandLine Lib "Kernel32" Alias "GetCommandLineW" () As
Long
    Private Declare Function apiGetCommandLineParams Lib "Kernel32" Alias "GetCommandLineA" ()
As Long
    Private Declare Function apiGetDiskFreeSpaceEx Lib "Kernel32" Alias "GetDiskFreeSpaceExA"
(ByVal lpDirectoryName As String, lpFreeBytesAvailableToCaller As Currency,
lpTotalNumberOfBytes As Currency, lpTotalNumberOfFreeBytes As Currency) As Long
    Private Declare Function apiGetDriveType Lib "Kernel32" Alias "GetDriveTypeA" (ByVal
nDrive As String) As Long
    Private Declare Function apiGetExitCodeProcess Lib "Kernel32" (ByVal hProcess As Long,
lpExitCode As Long) As Long
    Private Declare Function apiGetFileSize Lib "Kernel32" (ByVal hFile As Long,
lpFileSizeHigh As Long) As Long
    Private Declare Function apiGetForegroundWindow Lib "User32" Alias "GetForegroundWindow"
() As Long
    Private Declare Function apiGetFrequency Lib "Kernel32" Alias "QueryPerformanceFrequency"
(cyFrequency As Currency) As Long
    Private Declare Function apiGetLastError Lib "Kernel32" Alias "GetLastError" () As Integer
    Private Declare Function apiGetParent Lib "User32" Alias "GetParent" (ByVal hWnd As Long)
As Long
    Private Declare Function apiGetSystemMetrics Lib "User32" Alias "GetSystemMetrics" (ByVal
nIndex As Long) As Long
    Private Declare Function apiGetTickCount Lib "Kernel32" Alias "QueryPerformanceCounter"
(cyTickCount As Currency) As Long
    Private Declare Function apiGetTickCountMs Lib "Kernel32" Alias "GetTickCount" () As Long
    Private Declare Function apiGetUserName Lib "AdvApi32" Alias "GetUserNameA" (ByVal
lpBuffer As String, nSize As Long) As Long
    Private Declare Function apiGetWindow Lib "User32" Alias "GetWindow" (ByVal hWnd As Long,
ByVal wCmd As Long) As Long
    Private Declare Function apiGetWindowRect Lib "User32" Alias "GetWindowRect" (ByVal hWnd
As Long, lpRect As winRect) As Long
    Private Declare Function apiGetWindowText Lib "User32" Alias "GetWindowTextA" (ByVal hWnd
```

```
As Long, ByVal szWindowText As String, ByVal lLength As Long) As Long
    Private Declare Function apiGetWindowThreadProcessId Lib "User32" Alias
"GetWindowThreadProcessId" (ByVal hWnd As Long, lpdwProcessId As Long) As Long
    Private Declare Function apiIsCharAlphaNumericA Lib "User32" Alias "IsCharAlphaNumericA"
(ByVal byChar As Byte) As Long
    Private Declare Function apiIsIconic Lib "User32" Alias "IsIconic" (ByVal hWnd As Long) As
Long
    Private Declare Function apiIsWindowVisible Lib "User32" Alias "IsWindowVisible" (ByVal
hWnd As Long) As Long
    Private Declare Function apiIsZoomed Lib "User32" Alias "IsZoomed" (ByVal hWnd As Long) As
Long
    Private Declare Function apiLStrCpynA Lib "Kernel32" Alias "lstrcpynA" (ByVal pDestination
As String, ByVal pSource As Long, ByVal iMaxLength As Integer) As Long
    Private Declare Function apiMessageBox Lib "User32" Alias "MessageBoxA" (ByVal hWnd As
Long, ByVal lpText As String, ByVal lpCaption As String, ByVal wType As Long) As Long
    Private Declare Function apiOpenIcon Lib "User32" Alias "OpenIcon" (ByVal hWnd As Long) As
Long
    Private Declare Function apiOpenProcess Lib "Kernel32" Alias "OpenProcess" (ByVal
dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long
    Private Declare Function apiPathAddBackslashByPointer Lib "ShlwApi" Alias
"PathAddBackslashW" (ByVal lpszPath As Long) As Long
    Private Declare Function apiPathAddBackslashByString Lib "ShlwApi" Alias
"PathAddBackslashW" (ByVal lpszPath As String) As Long 'http://msdn.microsoft.com/en-
us/library/aa155716%28office.10%29.aspx
    Private Declare Function apiPostMessage Lib "User32" Alias "PostMessageA" (ByVal hWnd As
Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
    Private Declare Function apiReadFile Lib "Kernel32" (ByVal hFile As Long, lpBuffer As Any,
ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long, lpOverlapped As Any) As Long
    Private Declare Function apiRegQueryValue Lib "AdvApi32" Alias "RegQueryValue" (ByVal hKey
As Long, ByVal sValueName As String, ByVal dwReserved As Long, ByRef lValueType As Long, ByVal
sValue As String, ByRef lResultLen As Long) As Long
    Private Declare Function apiSendMessage Lib "User32" Alias "SendMessageA" (ByVal hWnd As
Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
    Private Declare Function apiSetActiveWindow Lib "User32" Alias "SetActiveWindow" (ByVal
hWnd As Long) As Long
    Private Declare Function apiSetCurrentDirectoryA Lib "Kernel32" Alias
"SetCurrentDirectoryA" (ByVal lpPathName As String) As Long
    Private Declare Function apiSetFocus Lib "User32" Alias "SetFocus" (ByVal hWnd As Long) As
Long
    Private Declare Function apiSetForegroundWindow Lib "User32" Alias "SetForegroundWindow"
(ByVal hWnd As Long) As Long
    Private Declare Function apiSetLocalTime Lib "Kernel32" Alias "SetLocalTime" (lpSystem As
SystemTime) As Long
    Private Declare Function apiSetWindowPlacement Lib "User32" Alias "SetWindowPlacement"
(ByVal hWnd As Long, ByRef lpwndpl As winPlacement) As Long
    Private Declare Function apiSetWindowPos Lib "User32" Alias "SetWindowPos" (ByVal hWnd As
Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal
cy As Long, ByVal wFlags As Long) As Long
    Private Declare Function apiSetWindowText Lib "User32" Alias "SetWindowTextA" (ByVal hWnd
As Long, ByVal lpString As String) As Long
    Private Declare Function apiShellExecute Lib "Shell32" 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 apiShowWindow Lib "User32" Alias "ShowWindow" (ByVal hWnd As
Long, ByVal nCmdShow As Long) As Long
    Private Declare Function apiShowWindowAsync Lib "User32" Alias "ShowWindowAsync" (ByVal
hWnd As Long, ByVal nCmdShow As Long) As Long
    Private Declare Function apiStrCpy Lib "Kernel32" Alias "lstrcpynA" (ByVal pDestination As
String, ByVal pSource As String, ByVal iMaxLength As Integer) As Long
    Private Declare Function apiStringLen Lib "Kernel32" Alias "lstrlenW" (ByVal lpString As
Long) As Long
```

```
    Private Declare Function apiStrTrimW Lib "ShlwApi" Alias "StrTrimW" () As Boolean
    Private Declare Function apiTerminateProcess Lib "Kernel32" Alias "TerminateProcess"
(ByVal hWnd As Long, ByVal uExitCode As Long) As Long
    Private Declare Function apiTimeGetTime Lib "Winmm" Alias "timeGetTime" () As Long
    Private Declare Function apiVarPtrArray Lib "MsVbVm50" Alias "VarPtr" (Var() As Any) As
Long
    Private Declare Function apiWaitForSingleObject Lib "Kernel32" (ByVal hHandle As Long,
ByVal dwMilliseconds As Long) As Long
    Private Type browseInfo     'used by apiBrowseForFolder
        hOwner As Long
        pidlRoot As Long
        pszDisplayName As String
        lpszTitle As String
        ulFlags As Long
        lpfn As Long
        lParam As Long
        iImage As Long
    End Type
    Private Declare Function apiBrowseForFolder Lib "Shell32" Alias "SHBrowseForFolderA"
(lpBrowseInfo As browseInfo) As Long
    Private Type CHOOSECOLOR    'used by apiChooseColor;
http://support.microsoft.com/kb/153929 and http://www.cpearson.com/Excel/Colors.aspx
        lStructSize As Long
        hWndOwner As Long
        hInstance As Long
        rgbResult As Long
        lpCustColors As String
        flags As Long
        lCustData As Long
        lpfnHook As Long
        lpTemplateName As String
    End Type
    Private Declare Function apiChooseColor Lib "ComDlg32" Alias "ChooseColorA" (pChoosecolor
As CHOOSECOLOR) As Long
    Private Type FindWindowParameters   'Custom structure for passing in the parameters in/out
of the hook enumeration function; could use global variables instead, but this is nicer
        strTitle As String  'INPUT
        hWnd As Long        'OUTPUT
    End Type                            'Find a specific window with dynamic caption from a
list of all open windows: http://www.everythingaccess.com/tutorials.asp?ID=Bring-an-external-
application-window-to-the-foreground
    Private Declare Function apiEnumWindows Lib "User32" Alias "EnumWindows" (ByVal lpEnumFunc
As Long, ByVal lParam As Long) As Long
    Private Type lastInputInfo  'used by apiGetLastInputInfo, getLastInputTime
        cbSize As Long
        dwTime As Long
    End Type
    Private Declare Function apiGetLastInputInfo Lib "User32" Alias "GetLastInputInfo" (ByRef
plii As lastInputInfo) As Long
    Private Type SystemTime
          wYear          As Integer
          wMonth         As Integer
          wDayOfWeek     As Integer
          wDay           As Integer
          wHour          As Integer
          wMinute        As Integer
          wSecond        As Integer
          wMilliseconds  As Integer
    End Type
    Private Declare Sub apiGetLocalTime Lib "Kernel32" Alias "GetLocalTime" (lpSystem As
SystemTime)
```

```
    Private Type pointAPI
        X As Long
        Y As Long
    End Type
    Private Type rectAPI
        Left_Renamed As Long
        Top_Renamed As Long
        Right_Renamed As Long
        Bottom_Renamed As Long
    End Type
    Private Type winPlacement
        length As Long
        flags As Long
        showCmd As Long
        ptMinPosition As pointAPI
        ptMaxPosition As pointAPI
        rcNormalPosition As rectAPI
    End Type
    Private Declare Function apiGetWindowPlacement Lib "User32" Alias "GetWindowPlacement"
(ByVal hWnd As Long, ByRef lpwndpl As winPlacement) As Long
    Private Type winRect
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
    End Type
    Private Declare Function apiMoveWindow Lib "User32" Alias "MoveWindow" (ByVal hWnd As
Long, xLeft As Long, ByVal yTop As Long, wWidth As Long, ByVal hHeight As Long, ByVal repaint
As Long) As Long
#Else   ' Win16 = True
#End If
```

Mac API

MicrosoftAPI

Office 2016 for Mac

VBAOfficeOffice 2016 for Mac。

。。。 Office 2016 for MacVBA

VBAOffice 2016 for Mac。

GrantAccessToMultipleFiles	
AppleScriptTask	VBAppleScript
MAC_OFFICE_VERSION	Mac OfficeIFDEF

Office 2011 for Mac

```
Private Declare Function system Lib "libc.dylib" (ByVal command As String) As Long
```

```
Private Declare Function popen Lib "libc.dylib" (ByVal command As String, ByVal mode As
String) As Long
Private Declare Function pclose Lib "libc.dylib" (ByVal file As Long) As Long
Private Declare Function fread Lib "libc.dylib" (ByVal outStr As String, ByVal size As Long,
ByVal items As Long, ByVal stream As Long) As Long
Private Declare Function feof Lib "libc.dylib" (ByVal file As Long) As Long
```

-

Office 2016 for Mac

```
Private Declare PtrSafe Function popen Lib "libc.dylib" (ByVal command As String, ByVal mode
As String) As LongPtr
Private Declare PtrSafe Function pclose Lib "libc.dylib" (ByVal file As LongPtr) As Long
Private Declare PtrSafe Function fread Lib "libc.dylib" (ByVal outStr As String, ByVal size As
LongPtr, ByVal items As LongPtr, ByVal stream As LongPtr) As Long
Private Declare PtrSafe Function feof Lib "libc.dylib" (ByVal file As LongPtr) As LongPtr
```

```
Option Explicit

'GetSystemMetrics32 info: http://msdn.microsoft.com/en-us/library/ms724385(VS.85).aspx
#If Win64 Then
    Private Declare PtrSafe Function GetSystemMetrics32 Lib "User32" Alias "GetSystemMetrics"
(ByVal nIndex As Long) As Long
#ElseIf Win32 Then
    Private Declare Function GetSystemMetrics32 Lib "User32" Alias "GetSystemMetrics" (ByVal
nIndex As Long) As Long
#End If

'VBA Wrappers:
Public Function dllGetMonitors() As Long
    Const SM_CMONITORS = 80
    dllGetMonitors = GetSystemMetrics32(SM_CMONITORS)
End Function

Public Function dllGetHorizontalResolution() As Long
    Const SM_CXVIRTUALSCREEN = 78
    dllGetHorizontalResolution = GetSystemMetrics32(SM_CXVIRTUALSCREEN)
End Function

Public Function dllGetVerticalResolution() As Long
    Const SM_CYVIRTUALSCREEN = 79
    dllGetVerticalResolution = GetSystemMetrics32(SM_CYVIRTUALSCREEN)
End Function

Public Sub ShowDisplayInfo()
    Debug.Print "Total monitors: " & vbTab & vbTab & dllGetMonitors
    Debug.Print "Horizontal Resolution: " & vbTab & dllGetHorizontalResolution
    Debug.Print "Vertical Resolution: " & vbTab & dllGetVerticalResolution

    'Total monitors:         1
    'Horizontal Resolution:  1920
    'Vertical Resolution:    1080
End Sub
```

FTPAPI

modFTP

```
Option Explicit
Option Compare Text
Option Private Module

'http://msdn.microsoft.com/en-us/library/aa384180(v=VS.85).aspx
'http://www.dailydoseofexcel.com/archives/2006/01/29/ftp-via-vba/
'http://www.15seconds.com/issue/981203.htm

'Open the Internet object
Private Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA" ( _
    ByVal sAgent As String, _
    ByVal lAccessType As Long, _
    ByVal sProxyName As String, _
    ByVal sProxyBypass As String, _
    ByVal lFlags As Long _
) As Long
'ex: lngINet = InternetOpen("MyFTP Control", 1, vbNullString, vbNullString, 0)

'Connect to the network
Private Declare Function InternetConnect Lib "wininet.dll" Alias "InternetConnectA" ( _
    ByVal hInternetSession As Long, _
    ByVal sServerName As String, _
    ByVal nServerPort As Integer, _
    ByVal sUsername As String, _
    ByVal sPassword As String, _
    ByVal lService As Long, _
    ByVal lFlags As Long, _
    ByVal lContext As Long _
) As Long
'ex: lngINetConn = InternetConnect(lngINet, "ftp.microsoft.com", 0, "anonymous",
"wally@wallyworld.com", 1, 0, 0)

'Get a file
Private Declare Function FtpGetFile Lib "wininet.dll" Alias "FtpGetFileA" ( _
    ByVal hFtpSession As Long, _
    ByVal lpszRemoteFile As String, _
    ByVal lpszNewFile As String, _
    ByVal fFailIfExists As Boolean, _
    ByVal dwFlagsAndAttributes As Long, _
    ByVal dwFlags As Long, _
    ByVal dwContext As Long _
) As Boolean
'ex: blnRC = FtpGetFile(lngINetConn, "dirmap.txt", "c:\dirmap.txt", 0, 0, 1, 0)

'Send a file
Private Declare Function FtpPutFile Lib "wininet.dll" Alias "FtpPutFileA" _
( _
    ByVal hFtpSession As Long, _
    ByVal lpszLocalFile As String, _
    ByVal lpszRemoteFile As String, _
    ByVal dwFlags As Long, ByVal dwContext As Long _
) As Boolean
'ex: blnRC = FtpPutFile(lngINetConn, "c:\dirmap.txt", "dirmap.txt", 1, 0)

'Delete a file
Private Declare Function FtpDeleteFile Lib "wininet.dll" Alias "FtpDeleteFileA" _
( _
    ByVal hFtpSession As Long, _
    ByVal lpszFileName As String _
```

```
) As Boolean
'ex: blnRC = FtpDeleteFile(lngINetConn, “test.txt”)

'Close the Internet object
Private Declare Function InternetCloseHandle Lib "wininet.dll" (ByVal hInet As Long) As
Integer
'ex: InternetCloseHandle lngINetConn
'ex: InternetCloseHandle lngINet



Private Declare Function FtpFindFirstFile Lib "wininet.dll" Alias "FtpFindFirstFileA" _
( _
    ByVal hFtpSession As Long, _
    ByVal lpszSearchFile As String, _
    lpFindFileData As WIN32_FIND_DATA, _
    ByVal dwFlags As Long, _
    ByVal dwContent As Long _
) As Long
Private Type FILETIME
        dwLowDateTime As Long
        dwHighDateTime As Long
End Type
Private Type WIN32_FIND_DATA
        dwFileAttributes As Long
        ftCreationTime As FILETIME
        ftLastAccessTime As FILETIME
        ftLastWriteTime As FILETIME
        nFileSizeHigh As Long
        nFileSizeLow As Long
        dwReserved0 As Long
        dwReserved1 As Long
        cFileName As String * MAX_FTP_PATH
        cAlternate As String * 14
End Type
'ex: lngHINet = FtpFindFirstFile(lngINetConn, "*.*", pData, 0, 0)

Private Declare Function InternetFindNextFile Lib "wininet.dll" Alias "InternetFindNextFileA"
_
( _
    ByVal hFind As Long, _
    lpvFindData As WIN32_FIND_DATA _
) As Long
'ex: blnRC = InternetFindNextFile(lngHINet, pData)


Public Sub showLatestFTPVersion()
    Dim ftpSuccess As Boolean, msg As String, lngFindFirst As Long
    Dim lngINet As Long, lngINetConn As Long
    Dim pData As WIN32_FIND_DATA
    'init the filename buffer
    pData.cFileName = String(260, 0)

    msg = "FTP Error"
    lngINet = InternetOpen("MyFTP Control", 1, vbNullString, vbNullString, 0)
    If lngINet > 0 Then
        lngINetConn = InternetConnect(lngINet, FTP_SERVER_NAME, FTP_SERVER_PORT,
FTP_USER_NAME, FTP_PASSWORD, 1, 0, 0)
        If lngINetConn > 0 Then
                FtpPutFile lngINetConn, "C:\Tmp\ftp.cls", "ftp.cls", FTP_TRANSFER_BINARY, 0
                'lngFindFirst = FtpFindFirstFile(lngINetConn, "ExcelDiff.xlsm", pData, 0, 0)
```

```
            If lngINet = 0 Then
                msg = "DLL error: " & Err.LastDllError & ", Error Number: " & Err.Number & ",
Error Desc: " & Err.Description
            Else
                msg = left(pData.cFileName, InStr(1, pData.cFileName, String(1, 0),
vbBinaryCompare) - 1)
            End If
            InternetCloseHandle lngINetConn
        End If
    InternetCloseHandle lngINet
    End If
        MsgBox msg
End Sub
```

---

modRegional

```
Option Explicit

Private Const LOCALE_SDECIMAL = &HE
Private Const LOCALE_SLIST = &HC

Private Declare Function GetLocaleInfo Lib "Kernel32" Alias "GetLocaleInfoA" (ByVal Locale As
Long, ByVal LCType As Long, ByVal lpLCData As String, ByVal cchData As Long) As Long
Private Declare Function SetLocaleInfo Lib "Kernel32" Alias "SetLocaleInfoA" (ByVal Locale As
Long, ByVal LCType As Long, ByVal lpLCData As String) As Boolean
Private Declare Function GetUserDefaultLCID% Lib "Kernel32" ()

Public Function getTimeSeparator() As String
    getTimeSeparator = Application.International(xlTimeSeparator)
End Function
Public Function getDateSeparator() As String
    getDateSeparator = Application.International(xlDateSeparator)
End Function
Public Function getListSeparator() As String
    Dim ListSeparator As String, iRetVal1 As Long, iRetVal2 As Long, lpLCDataVar As String,
Position As Integer, Locale As Long
    Locale = GetUserDefaultLCID()
    iRetVal1 = GetLocaleInfo(Locale, LOCALE_SLIST, lpLCDataVar, 0)
    ListSeparator = String$(iRetVal1, 0)
    iRetVal2 = GetLocaleInfo(Locale, LOCALE_SLIST, ListSeparator, iRetVal1)
    Position = InStr(ListSeparator, Chr$(0))
    If Position > 0 Then ListSeparator = Left$(ListSeparator, Position - 1) Else ListSeparator
= vbNullString
    getListSeparator = ListSeparator
End Function

Private Sub ChangeSettingExample()  'change the setting of the character displayed as the
decimal separator.
    Call SetLocalSetting(LOCALE_SDECIMAL, ",")  'to change to ","
    Stop                                        'check your control panel to verify or use the
GetLocaleInfo API function
    Call SetLocalSetting(LOCALE_SDECIMAL, ".")  'to back change to "."
End Sub

Private Function SetLocalSetting(LC_CONST As Long, Setting As String) As Boolean
    Call SetLocaleInfo(GetUserDefaultLCID(), LC_CONST, Setting)
End Function
```

API https://riptutorial.com/zh-CN/vba/topic/10569/api

3: CreateObjectGetObject

`CreateObjectGetObject`。 Instancing 。 SingleUseWMI。 ExcelMultiUse。 `GetObject Run-time error '429': ActiveX component can't create object`。

GetObject

1. *Pathname* - VariantString。 *Pathname Class* 。
2. *Class* - VariantStringApplicationObjectType。 *PathnameClass* 。

CreateObject

1. *Class* - VariantStringApplicationObjectType。 。
2. *Servername* - VariantString。 。

`Application.ObjectType`

1. - 。 |
2. - 。 |

1. Word.Application
2. Excel.Sheet
3. Scripting.FileSystemObject

Examples

GetObjectCreateObject

MSDN-GetObject

ActiveX。

GetObject。 CreateObject。

```
Sub CreateVSGet()
    Dim ThisXLApp As Excel.Application 'An example of early binding
    Dim AnotherXLApp As Object 'An example of late binding
    Dim ThisNewWB As Workbook
    Dim AnotherNewWB As Workbook
    Dim wb As Workbook

    'Get this instance of Excel
    Set ThisXLApp = GetObject(ThisWorkbook.Name).Application
    'Create another instance of Excel
    Set AnotherXLApp = CreateObject("Excel.Application")
    'Make the 2nd instance visible
    AnotherXLApp.Visible = True
    'Add a workbook to the 2nd instance
```

```
    Set AnotherNewWB = AnotherXLApp.Workbooks.Add
    'Add a sheet to the 2nd instance
    AnotherNewWB.Sheets.Add

    'You should now have 2 instances of Excel open
    'The 1st instance has 1 workbook: Book1
    'The 2nd instance has 1 workbook: Book2

    'Lets add another workbook to our 1st instance
    Set ThisNewWB = ThisXLApp.Workbooks.Add
    'Now loop through the workbooks and show their names
    For Each wb In ThisXLApp.Workbooks
        Debug.Print wb.Name
    Next
    'Now the 1st instance has 2 workbooks: Book1 and Book3
    'If you close the first instance of Excel,
    'Book1 and Book3 will close, but book2 will still be open

End Sub
```

CreateObjectGetObject https://riptutorial.com/zh-CN/vba/topic/7729/createobjectgetobject

4: Scripting.Dictionary

VBE→Microsoft Scripting RuntimeVBAScripting Dictionary。;VBA。

Examples

Scripting DictionaryKey / Item。 KeysItemsKey。

“” *Key* 。 Keys“”KeyItem。

	/		
CompareMode	/	CompareMode	CompareMode。0vbBinaryCompare1vbTextCompare2 vbDatabaseCompare。
			/。
	/		。
	/		。 。 。

	KeyItem。 Keys。
	。
	。
	。
	。
	。

```
'Populate, enumerate, locate and remove entries in a dictionary that was created
'with late binding
Sub iterateDictionaryLate()
    Dim k As Variant, dict As Object

    Set dict = CreateObject("Scripting.Dictionary")
    dict.CompareMode = vbTextCompare          'non-case sensitive compare model

    'populate the dictionary
    dict.Add Key:="Red", Item:="Balloon"
    dict.Add Key:="Green", Item:="Balloon"
```

```
    dict.Add Key:="Blue", Item:="Balloon"

    'iterate through the keys
    For Each k In dict.Keys
        Debug.Print k & " - " & dict.Item(k)
    Next k

    'locate the Item for Green
    Debug.Print dict.Item("Green")

    'remove key/item pairs from the dictionary
    dict.Remove "blue"      'remove individual key/item pair by key
    dict.RemoveAll          'remove all remaining key/item pairs

End Sub

'Populate, enumerate, locate and remove entries in a dictionary that was created
'with early binding (see Remarks)
Sub iterateDictionaryEarly()
    Dim d As Long, k As Variant
    Dim dict As New Scripting.Dictionary

    dict.CompareMode = vbTextCompare          'non-case sensitive compare model

    'populate the dictionary
    dict.Add Key:="Red", Item:="Balloon"
    dict.Add Key:="Green", Item:="Balloon"
    dict.Add Key:="Blue", Item:="Balloon"
    dict.Add Key:="White", Item:="Balloon"

    'iterate through the keys
    For Each k In dict.Keys
        Debug.Print k & " - " & dict.Item(k)
    Next k

    'iterate through the keys by the count
    For d = 0 To dict.Count - 1
        Debug.Print dict.Keys(d) & " - " & dict.Items(d)
    Next d

    'iterate through the keys by the boundaries of the keys collection
    For d = LBound(dict.Keys) To UBound(dict.Keys)
        Debug.Print dict.Keys(d) & " - " & dict.Items(d)
    Next d

    'locate the Item for Green
    Debug.Print dict.Item("Green")
    'locate the Item for the first key
    Debug.Print dict.Item(dict.Keys(0))
    'locate the Item for the last key
    Debug.Print dict.Item(dict.Keys(UBound(dict.Keys)))

    'remove key/item pairs from the dictionary
    dict.Remove "blue"                        'remove individual key/item pair by key
    dict.Remove dict.Keys(0)                  'remove first key/item by index position
    dict.Remove dict.Keys(UBound(dict.Keys))  'remove last key/item by index position
    dict.RemoveAll                            'remove all remaining key/item pairs

End Sub
```

Scripting.Dictionary

- 。

Log

	10/12/2016 9:00
	2016101313:00
	2016101313:30
	2016101314:00
	2016101413:00

Summary。

1.ActiveWorkbook。

;。

3.Dictionary。

```
Sub LastEdit()
Dim vLog as Variant, vKey as Variant
Dim dict as New Scripting.Dictionary
Dim lastRow As Integer, lastColumn As Integer
Dim i as Long
Dim anchor As Range

With ActiveWorkbook
    With .Sheets("Log")
        'Pull entries in "log" into a variant array
        lastRow = .Range("a" & .Rows.Count).End(xlUp).Row
        vlog = .Range("a1", .Cells(lastRow, 2)).Value2

        'Loop through array
        For i = 1 to lastRow
            Dim username As String
            username = vlog(i, 1)
            Dim editDate As Date
            editDate = vlog(i, 2)

            'If the username is not yet in the dictionary:
            If Not dict.Exists(username) Then
                dict(username) = editDate
            ElseIf dict(username) < editDate Then
                dict(username) = editDate
            End If
        Next
    End With

    With .Sheets("Summary")
        'Loop through keys
```

```
        For Each vKey in dict.Keys
            'Add the key and value at the next available row
            Anchor = .Range("A" & .Rows.Count).End(xlUp).Offset(1,0)
            Anchor = vKey
            Anchor.Offset(0,1) = dict(vKey)
        Next vKey
    End With
End With
End Sub
```

Summary

	2016101313:30
	2016101413:00

For

```
        'Loop through array
        For i = 1 to lastRow
            Dim username As String
            username = vlog(i, 1)

            'If the username is not yet in the dictionary:
            If Not dict.Exists(username) Then
                dict(username) = 1
            Else
                dict(username) = dict(username) + 1
            End If
        Next
```

Summary

	2
	3

Scripting.Dictionary

Dictionary◦

```
Function Unique(values As Variant) As Variant()
    'Put all the values as keys into a dictionary
    Dim dict As New Scripting.Dictionary
    Dim val As Variant
    For Each val In values
        dict(val) = 1 'The value doesn't matter here
    Next
    Unique = dict.Keys
End Function
```

```
Dim duplicates() As Variant
duplicates = Array(1, 2, 3, 1, 2, 3)
Dim uniqueVals() As Variant
uniqueVals = Unique(duplicates)
```

uniqueVals{1,2,3} ◦

◦

Scripting.Dictionary https://riptutorial.com/zh-CN/vba/topic/3667/scripting-dictionary

5: Scripting.FileSystemObject

Examples

FileSystemObject

```
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

Sub FsoExample()
    Dim fso As Object ' declare variable
    Set fso = CreateObject("Scripting.FileSystemObject") ' Set it to be a File System Object

    ' now use it to check if a file exists
    Dim myFilePath As String
    myFilePath = "C:\mypath\to\myfile.txt"
    If fso.FileExists(myFilePath) Then
        ' do something
    Else
        ' file doesn't exist
        MsgBox "File doesn't exist"
    End If
End Sub
```

FileSystemObject

```
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

Sub ReadTextFileExample()
    Dim fso As Object
    Set fso = CreateObject("Scripting.FileSystemObject")

    Dim sourceFile As Object
    Dim myFilePath As String
    Dim myFileText As String

    myFilePath = "C:\mypath\to\myfile.txt"
    Set sourceFile = fso.OpenTextFile(myFilePath, ForReading)
    myFileText = sourceFile.ReadAll ' myFileText now contains the content of the text file
    sourceFile.Close ' close the file
    ' do whatever you might need to do with the text

    ' You can also read it line by line
    Dim line As String
    Set sourceFile = fso.OpenTextFile(myFilePath, ForReading)
    While Not sourceFile.AtEndOfStream ' while we are not finished reading through the file
        line = sourceFile.ReadLine
        ' do something with the line...
    Wend
    sourceFile.Close
End Sub
```

FileSystemObject

```
Sub CreateTextFileExample()
    Dim fso As Object
    Set fso = CreateObject("Scripting.FileSystemObject")

    Dim targetFile As Object
    Dim myFilePath As String
    Dim myFileText As String

    myFilePath = "C:\mypath\to\myfile.txt"
    Set targetFile = fso.CreateTextFile(myFilePath, True)  ' this will overwrite any existing
file
    targetFile.Write "This is some new text"
    targetFile.Write " And this text will appear right after the first bit of text."
    targetFile.WriteLine "This bit of text includes a newline character to ensure each write
takes its own line."
    targetFile.Close ' close the file
End Sub
```

FileSystemObject

```
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8


Sub WriteTextFileExample()
    Dim oFso
    Set oFso = CreateObject("Scripting.FileSystemObject")

    Dim oFile as Object
    Dim myFilePath as String
    Dim myFileText as String

    myFilePath = "C:\mypath\to\myfile.txt"
    ' First check if the file exists
    If oFso.FileExists(myFilePath) Then
        ' this will overwrite any existing filecontent with whatever you send the file
        ' to append data to the end of an existing file, use ForAppending instead
        Set oFile = oFso.OpenTextFile(myFilePath, ForWriting)
    Else
        ' create the file instead
        Set oFile = oFso.CreateTextFile(myFilePath) ' skipping the optional boolean for
overwrite if exists as we already checked that the file doesn't exist.
    End If
    oFile.Write "This is some new text"
    oFile.Write " And this text will appear right after the first bit of text."
    oFile.WriteLine "This bit of text includes a newline character to ensure each write takes
its own line."
    oFile.Close ' close the file
End Sub
```

FileSystemObject

Microsoft Scripting Runtime

```
Public Sub EnumerateDirectory()
    Dim fso As Scripting.FileSystemObject
    Set fso = New Scripting.FileSystemObject

    Dim targetFolder As Folder
    Set targetFolder = fso.GetFolder("C:\")

    Dim foundFile As Variant
    For Each foundFile In targetFolder.Files
        Debug.Print foundFile.Name
    Next
End Sub
```

```
Public Sub EnumerateDirectory()
    Dim fso As Object
    Set fso = CreateObject("Scripting.FileSystemObject")

    Dim targetFolder As Object
    Set targetFolder = fso.GetFolder("C:\")

    Dim foundFile As Variant
    For Each foundFile In targetFolder.Files
        Debug.Print foundFile.Name
    Next
End Sub
```

Microsoft Scripting Runtime

```
Sub EnumerateFilesAndFolders( _
    FolderPath As String, _
    Optional MaxDepth As Long = -1, _
    Optional CurrentDepth As Long = 0, _
    Optional Indentation As Long = 2)

    Dim FSO As Scripting.FileSystemObject
    Set FSO = New Scripting.FileSystemObject

    'Check the folder exists
    If FSO.FolderExists(FolderPath) Then
        Dim fldr As Scripting.Folder
        Set fldr = FSO.GetFolder(FolderPath)

        'Output the starting directory path
        If CurrentDepth = 0 Then
          Debug.Print fldr.Path
        End If

        'Enumerate the subfolders
        Dim subFldr As Scripting.Folder
        For Each subFldr In fldr.SubFolders
            Debug.Print Space$((CurrentDepth + 1) * Indentation) & subFldr.Name
            If CurrentDepth < MaxDepth Or MaxDepth = -1 Then
                'Recursively call EnumerateFilesAndFolders
                EnumerateFilesAndFolders subFldr.Path, MaxDepth, CurrentDepth + 1, Indentation
            End If
        Next subFldr

        'Enumerate the files
        Dim fil As Scripting.File
```

```
        For Each fil In fldr.Files
            Debug.Print Space$((CurrentDepth + 1) * Indentation) & fil.Name
        Next fil
    End If
End Sub
```

```
EnumerateFilesAndFolders "C:\Test"
```

```
C:\Test
  Documents
    Personal
      Budget.xls
      Recipes.doc
    Work
      Planning.doc
  Downloads
    FooBar.exe
  ReadMe.txt
```

```
EnumerateFilesAndFolders "C:\Test", 0
```

```
C:\Test
  Documents
  Downloads
  ReadMe.txt
```

```
EnumerateFilesAndFolders "C:\Test", 1, 4
```

```
C:\Test
    Documents
        Personal
        Work
    Downloads
        FooBar.exe
    ReadMe.txt
```

```
Dim fso As New Scripting.FileSystemObject
Debug.Print fso.GetBaseName("MyFile.something.txt")
```

```
MyFile.something
```

```
GetBaseName()。
```

```
Dim fso As New Scripting.FileSystemObject
Debug.Print fso.GetExtensionName("MyFile.something.txt")
```

```
txt GetExtensionName()。
```

GetParentFolderName。

```
Dim fso As New Scripting.FileSystemObject
Debug.Print fso.GetParentFolderName("C:\Users\Me\My Documents\SomeFile.txt")
```

```
C:\Users\Me\My Documents
```

◦

FSO.BuildPath

\ ◦ FSO.BuildPath

```
  Const sourceFilePath As String = "C:\Temp"  '<-- Without trailing backslash
  Const targetFilePath As String = "C:\Temp\" '<-- With trailing backslash

  Const fileName As String = "Results.txt"

  Dim FSO As FileSystemObject
  Set FSO = New FileSystemObject

  Debug.Print FSO.BuildPath(sourceFilePath, fileName)
  Debug.Print FSO.BuildPath(targetFilePath, fileName)
```

```
C:\Temp\Results.txt
C:\Temp\Results.txt
```

Scripting.FileSystemObject https://riptutorial.com/zh-CN/vba/topic/990/scripting-filesystemobject

6: VBA

◦ ◦

Examples

'3'GoSub

```
Sub DoSomething()
    GoSub DoThis
DoThis:
    Debug.Print "Hi!"
    Return
End Sub
```

DoSomethingDoThis“”GoSub“”Return GoSub◦

```
Sub DoSomething()
    GoSub DoThis
    Exit Sub
DoThis:
    Debug.Print "Hi!"
    Return
End Sub
```

DoThisExit SubDoThis - DoThisGoSub◦

GoSub / Return◦ ◦

'20' ;On Error GoTo ◦

'6'

```
Sub DoSomething()
    Dim row As Integer
    For row = 1 To 100000
        'do stuff
    Next
End Sub
```

Integer1632,767;◦

```
Sub DoSomething()
    Dim row As Long
    For row = 1 To 100000
        'do stuff
    Next
End Sub
```

Long 3232,767。

。

'9'

```
Sub DoSomething()
    Dim foo(1 To 10)
    Dim i As Long
    For i = 1 To 100
        foo(i) = i
    Next
End Sub
```

foo10。 i11 foo(i) 。 。

```
Sub DoSomething()
    Dim foo(1 To 10)
    Dim i As Long
    For i = LBound(foo) To UBound(foo)
        foo(i) = i
    Next
End Sub
```

LBoundUBound。

ThisWorkbook.Worksheets("I don't exist") 。

; Collection5“”

```
Sub RaisesRunTimeError5()
    Dim foo As New Collection
    foo.Add "foo", "foo"
    Debug.Print foo("bar")
End Sub
```

'13'

```
Public Sub DoSomething()
    DoSomethingElse "42?"
End Sub

Private Sub DoSomethingElse(foo As Date)
'    Debug.Print MonthName(Month(foo))
End Sub
```

VBA"42?"Date。 DoSomethingElseVBA13 。

```
Public Sub DoSomething()
    DoSomethingElse Now
```

```
End Sub

Private Sub DoSomethingElse(foo As Date)
'    Debug.Print MonthName(Month(foo))
End Sub
```

DateDate◦

'91'

```
Sub DoSomething()
    Dim foo As Collection
    With foo
        .Add "ABC"
        .Add "XYZ"
    End With
End Sub
```

Set◦ Nothing◦ fooCollectionNothing - Nothing.Add ◦

```
Sub DoSomething()
    Dim foo As Collection
    Set foo = New Collection
    With foo
        .Add "ABC"
        .Add "XYZ"
    End With
End Sub
```

Set .Add◦

- ExcelRange.FindRange

```
Dim resultRow As Long
resultRow = SomeSheet.Cells.Find("Something").Row
```

Nothing .Row◦

If Not xxxx Is Nothing

```
Dim result As Range
Set result = SomeSheet.Cells.Find("Something")

Dim resultRow As Long
If Not result Is Nothing Then resultRow = result.Row
```

'20'

```
Sub DoSomething()
    On Error GoTo CleanFail
    DoSomethingElse
```

```
CleanFail:
    Debug.Print Err.Number
    Resume Next
End Sub
```

DoSomethingElseCleanFail Resume NextDebug.PrintResume Next20。

```
Sub DoSomething()
    On Error GoTo CleanFail
    DoSomethingElse

    Exit Sub
CleanFail:
    Debug.Print Err.Number
    Resume Next
End Sub
```

CleanFailExit SubCleanFail - On Error;Resume20。

“3”GoSub ;On Error GoTo 。

VBA https://riptutorial.com/zh-CN/vba/topic/8917/vba

7: VBA

- optionName [value]
-
- {||}
-
- {0 | 1}

	;/ mispelled。
	“a”=“A”。
	。 /ASCII。
	MS-AccessSQL。
	Public。
0	。 0 。 0 。
1	1 。 1 。

Option Base {0|1}。

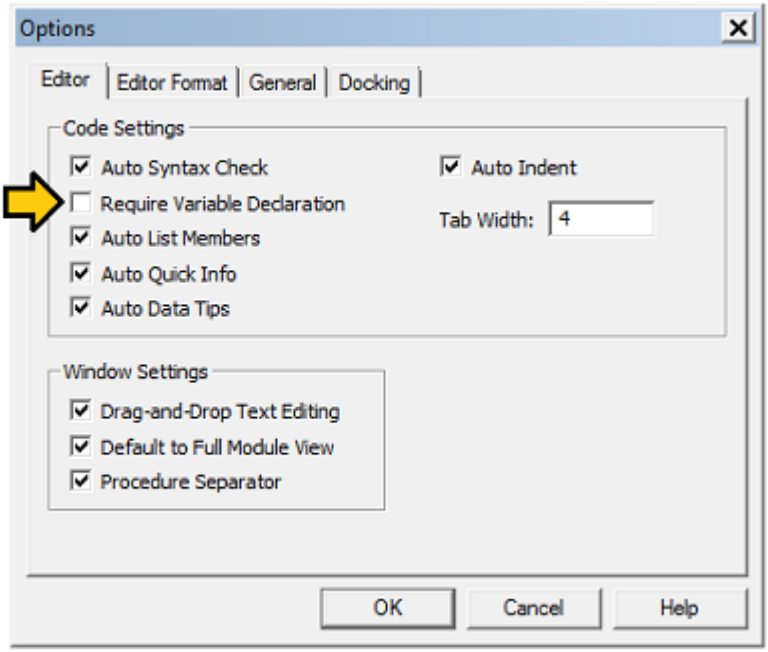
```
Dim myStringsA(0 To 5) As String '// This has 6 elements (0 - 5)
Dim myStringsB(1 To 5) As String '// This has 5 elements (1 - 5)
Dim myStringsC(6 To 9) As String '// This has 3 elements (6 - 9)
```

Examples

VBAOption Explicit 。 IntelliSense。

```
Option Explicit

Sub OptionExplicit()
    Dim a As Integer
    a = 5
    b = 10 '// Causes compile error as 'b' is not declared
End Sub
```

VBE ►►**Option Explicit**。



。 ALWAYS“Option Explicit”。

{Binary ||}

/ 。 。

A <B <E <Z <a <b <e <z

。

```
Option Compare Binary

Sub CompareBinary()

    Dim foo As String
    Dim bar As String

    '// Case sensitive
    foo = "abc"
    bar = "ABC"

    Debug.Print (foo = bar) '// Prints "False"

    '// Still differentiates accented characters
    foo = "ábc"
    bar = "abc"

    Debug.Print (foo = bar) '// Prints "False"

    '// "b" (Chr 98) is greater than "a" (Chr 97)
    foo = "a"
    bar = "b"

    Debug.Print (bar > foo) '// Prints "True"

    '// "b" (Chr 98) is NOT greater than "á" (Chr 225)
    foo = "á"
```

```
    bar = "b"

    Debug.Print (bar > foo) '// Prints "False"

End Sub
```

/。

A | a<B | b<Z | z

```
Option Compare Text

Sub CompareText()

    Dim foo As String
    Dim bar As String

    '// Case insensitivity
    foo = "abc"
    bar = "ABC"

    Debug.Print (foo = bar) '// Prints "True"

    '// Still differentiates accented characters
    foo = "ábc"
    bar = "abc"

    Debug.Print (foo = bar) '// Prints "False"

    '// "b" still comes after "a" or "á"
    foo = "á"
    bar = "b"

    Debug.Print (bar > foo) '// Prints "True"

End Sub
```

MS Access。/。

Access UDFSQL。

{0 | 1}

`Option Base`。。

Option BaseBase0.0。

`Option Base 1` 1

0

```
Option Base 0

Sub BaseZero()
```

```
    Dim myStrings As Variant

    ' Create an array out of the Variant, having 3 fruits elements
    myStrings = Array("Apple", "Orange", "Peach")

    Debug.Print LBound(myStrings) ' This Prints "0"
    Debug.Print UBound(myStrings) ' This print "2", because we have 3 elements beginning at 0
-> 0,1,2

    For i = 0 To UBound(myStrings)

        Debug.Print myStrings(i) ' This will print "Apple", then "Orange", then "Peach"

    Next i

End Sub
```

1

```
Option Base 1

Sub BaseOne()

    Dim myStrings As Variant

    ' Create an array out of the Variant, having 3 fruits elements
    myStrings = Array("Apple", "Orange", "Peach")

    Debug.Print LBound(myStrings) ' This Prints "1"
    Debug.Print UBound(myStrings) ' This print "3", because we have 3 elements beginning at 1
-> 1,2,3

    For i = 0 To UBound(myStrings)

        Debug.Print myStrings(i) ' This triggers an error 9 "Subscript out of range"

    Next i

End Sub
```

90`Base 1`

Base 1

```
    For i = 1 To UBound(myStrings)

        Debug.Print myStrings(i) ' This will print "Apple", then "Orange", then "Peach"

    Next i
```

`Option Base` Split ◦ **Split**

◦

Excel`Range.Value`Range`.Formula`12D Variant◦

ADO `Recordset.GetRows`12D。

“”LBoundUBound。

```
'for single dimensioned array
Debug.Print LBound(arr) & ":" & UBound(arr)
Dim i As Long
For i = LBound(arr) To UBound(arr)
    Debug.Print arr(i)
Next i

'for two dimensioned array
Debug.Print LBound(arr, 1) & ":" & UBound(arr, 1)
Debug.Print LBound(arr, 2) & ":" & UBound(arr, 2)
Dim i As long, j As Long
For i = LBound(arr, 1) To UBound(arr, 1)
    For j = LBound(arr, 2) To UBound(arr, 2)
         Debug.Print arr(i, j)
    Next j
Next i
```

1`Option Base 1`。

VBA https://riptutorial.com/zh-CN/vba/topic/3992/vba

8: ByRefByVal

ByRefByVal。 VBAByRef ByRef。

VB.NETByRef。

。 424“Object Required”

```
Public Sub Test()
    DoSomething Array(1, 2, 3)
End Sub

Private Sub DoSomething(ByVal foo As Variant)
    foo.Add 42
End Sub
```

```
Private Sub DoSomething(ByVal foo() As Variant) 'ByVal is illegal for arrays
    foo.Add 42
End Sub
```

Examples

ByRefByVal

ByRefByValCallingProcedureCalledProcedure CalledProcedure 。

ByRef CalledProcedureCalledProcedureCallingProcedure 。

ByValCalledProcedure 。

```
Sub CalledProcedure(ByRef X As Long, ByVal Y As Long)
    X = 321
    Y = 654
End Sub

Sub CallingProcedure()
    Dim A As Long
    Dim B As Long
    A = 123
    B = 456

    Debug.Print "BEFORE CALL => A: " & CStr(A), "B: " & CStr(B)
    ''Result : BEFORE CALL => A: 123 B: 456

    CalledProcedure X:=A, Y:=B

    Debug.Print "AFTER CALL =  A: " & CStr(A), "B: " & CStr(B)
    ''Result : AFTER CALL => A: 321 B: 456
End Sub
```

```
Sub Main()
```

```
    Dim IntVarByVal As Integer
    Dim IntVarByRef As Integer

    IntVarByVal = 5
    IntVarByRef = 10

    SubChangeArguments IntVarByVal, IntVarByRef '5 goes in as a "copy". 10 goes in as a
reference
    Debug.Print "IntVarByVal: " & IntVarByVal 'prints 5 (no change made by SubChangeArguments)
    Debug.Print "IntVarByRef: " & IntVarByRef 'prints 99 (the variable was changed in
SubChangeArguments)
End Sub

Sub SubChangeArguments(ByVal ParameterByVal As Integer, ByRef ParameterByRef As Integer)
    ParameterByVal = ParameterByVal + 2 ' 5 + 2 = 7 (changed only inside this Sub)
    ParameterByRef = ParameterByRef + 89 ' 10 + 89 = 99 (changes the IntVarByRef itself - in
the Main Sub)
End Sub
```

ByRef

。

```
Public Sub DoSomething1(foo As Long)
End Sub
```

```
Public Sub DoSomething2(ByRef foo As Long)
End Sub
```

fooDoSomething1DoSomething2ByRef。

VBA。VB.NET/。

- ByRef 。

  ```
  Public Sub Test()
      Dim foo As Long
      foo = 42
      DoSomething foo
      Debug.Print foo
  End Sub

  Private Sub DoSomething(ByRef foo As Long)
      foo = foo * 2
  End Sub
  ```

  Test84. DoSomethingfoo。

- ByRef 。

  ```
  Public Sub Test()
  ```

```
    Dim foo As Collection
    Set foo = New Collection
    DoSomething foo
    Debug.Print foo.Count
End Sub

Private Sub DoSomething(ByRef foo As Collection)
    foo.Add 42
    Set foo = Nothing
End Sub
```

91 CountDoSomethingNothing ◦

ByVal

ByRefByVal

```
Public Sub Test()
    Dim foo As Long
    foo = 42
    DoSomething (foo)
    Debug.Print foo
End Sub

Private Sub DoSomething(ByRef foo As Long)
    foo = foo * 2
End Sub
```

ByRef 42◦

◦

```
bar = DoSomething(foo) 'function call, no whitespace; parens are part of args list
DoSomething (foo) 'procedure call, notice whitespace; parens are NOT part of args
list
DoSomething foo 'procedure call does not force the foo parameter to be ByVal
```

BYVAL

- ByVal ◦

```
Public Sub Test()
    Dim foo As Long
    foo = 42
    DoSomething foo
    Debug.Print foo
End Sub

Private Sub DoSomething(ByVal foo As Long)
    foo = foo * 2
End Sub
```

Test42. DoSomethingfoo ◦ 2;◦

- ByVal ◦

```
Public Sub Test()
    Dim foo As Collection
    Set foo = New Collection
    DoSomething foo
    Debug.Print foo.Count
End Sub

Private Sub DoSomething(ByVal foo As Collection)
    foo.Add 42
    Set foo = Nothing
End Sub
```

Test1. DoSomethingfoo Collection ◦ TestfooDoSomething◦ Nothing◦

ByRefByVal https://riptutorial.com/zh-CN/vba/topic/7363/byrefbyval

9: ADO

Microsoft ActiveXxx。 ObjectNewCreateObject。

Examples

ADOADO Connection。 DSNID.OpenDSN。

DSNADO - ODBC。 ConnectionStrings.com。

```
Const SomeDSN As String = "DSN=SomeDSN;Uid=UserName;Pwd=MyPassword;"

Public Sub Example()
    Dim database As ADODB.Connection
    Set database = OpenDatabaseConnection(SomeDSN)
    If Not database Is Nothing Then
        '... Do work.
        database.Close          'Make sure to close all database connections.
    End If
End Sub

Public Function OpenDatabaseConnection(ConnString As String) As ADODB.Connection
    On Error GoTo Handler
    Dim database As ADODB.Connection
    Set database = New ADODB.Connection

    With database
        .ConnectionString = ConnString
        .ConnectionTimeout = 10             'Value is given in seconds.
        .Open
    End With

    OpenDatabaseConnection = database

    Exit Function
Handler:
    Debug.Print "Database connection failed. Check your connection string."
End Function
```

。 。 Windows。

ADO Recordset。 “ ”OpenDatabaseConnection。 SQL。

SQLConnection

```
Public Sub DisplayDistinctItems()
    On Error GoTo Handler
    Dim database As ADODB.Connection
    Set database = OpenDatabaseConnection(SomeDSN)

    If Not database Is Nothing Then
        Dim records As ADODB.Recordset
        Set records = database.Execute("SELECT DISTINCT Item FROM Table")
        'Loop through the returned Recordset.
```

```
        Do While Not records.EOF      'EOF is false when there are more records.
            'Individual fields are indexed either by name or 0 based ordinal.
            'Note that this is using the default .Fields member of the Recordset.
            Debug.Print records("Item")
            'Move to the next record.
            records.MoveNext
        Loop
    End If
CleanExit:
    If Not records Is Nothing Then records.Close
    If Not database Is Nothing And database.State = adStateOpen Then
        database.Close
    End If
    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & ": " & Err.Description
    Resume CleanExit
End Sub
```

ADO `Command`。

```
Public Sub DisplayDistinctItems()
    On Error GoTo Handler
    Dim database As ADODB.Connection
    Set database = OpenDatabaseConnection(SomeDSN)

    If Not database Is Nothing Then
        Dim query As ADODB.Command
        Set query = New ADODB.Command
        'Build the command to pass to the data source.
        With query
            .ActiveConnection = database
            .CommandText = "SELECT DISTINCT Item FROM Table"
            .CommandType = adCmdText
        End With
        Dim records As ADODB.Recordset
        'Execute the command to retrieve the recordset.
        Set records = query.Execute()

        Do While Not records.EOF
            Debug.Print records("Item")
            records.MoveNext
        Loop
    End If
CleanExit:
    If Not records Is Nothing Then records.Close
    If Not database Is Nothing And database.State = adStateOpen Then
        database.Close
    End If
    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & ": " & Err.Description
    Resume CleanExit
End Sub
```

SQL 。 。 。

ADOSQL。 `ExecuteRecordset` @@ IdentitySQLINSERT。 “ ”`OpenDatabaseConnection`。

```
Public Sub UpdateTheFoos()
    On Error GoTo Handler
    Dim database As ADODB.Connection
    Set database = OpenDatabaseConnection(SomeDSN)

    If Not database Is Nothing Then
        Dim update As ADODB.Command
        Set update = New ADODB.Command
        'Build the command to pass to the data source.
        With update
            .ActiveConnection = database
            .CommandText = "UPDATE Table SET Foo = 42 WHERE Bar IS NULL"
            .CommandType = adCmdText
            .Execute        'We don't need the return from the DB, so ignore it.
        End With
    End If
CleanExit:
    If Not database Is Nothing And database.State = adStateOpen Then
        database.Close
    End If
    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & ": " & Err.Description
    Resume CleanExit
End Sub
```

SQL 。 SQL。 。

ADOSQLSQL。 Parameter。

ODBC?“”Command 。

“ ”OpenDatabaseConnection。

```
Public Sub UpdateTheFoos()
    On Error GoTo Handler
    Dim database As ADODB.Connection
    Set database = OpenDatabaseConnection(SomeDSN)

    If Not database Is Nothing Then
        Dim update As ADODB.Command
        Set update = New ADODB.Command
        'Build the command to pass to the data source.
        With update
            .ActiveConnection = database
            .CommandText = "UPDATE Table SET Foo = ? WHERE Bar = ?"
            .CommandType = adCmdText

            'Create the parameters.
            Dim fooValue As ADODB.Parameter
            Set fooValue = .CreateParameter("FooValue", adNumeric, adParamInput)
            fooValue.Value = 42

            Dim condition As ADODB.Parameter
            Set condition = .CreateParameter("Condition", adBSTR, adParamInput)
            condition.Value = "Bar"

            'Add the parameters to the Command
```

```
            .Parameters.Append fooValue
            .Parameters.Append condition
            .Execute
        End With
    End If
CleanExit:
    If Not database Is Nothing And database.State = adStateOpen Then
        database.Close
    End If
    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & ": " & Err.Description
    Resume CleanExit
End Sub
```

UPDATESQL。

ADO https://riptutorial.com/zh-CN/vba/topic/3578/ado

10:

- VBA
  - `String` / `String$`
  - `Space` / `Space$` ◦

Examples

Stringn

```
Dim lineOfHyphens As String
'Assign a string with 80 repeated hyphens
lineOfHyphens = String$(80, "-")
```

StringSpacen

```
Dim stringOfSpaces As String

'Assign a string with 255 repeated spaces using Space$
stringOfSpaces = Space$(255)

'Assign a string with 255 repeated spaces using String$
stringOfSpaces = String$(255, " ")
```

https://riptutorial.com/zh-CN/vba/topic/3581/

11:

Examples

Sub◦

```
Sub ProcedureName ([argument_list])
    [statements]
End Sub
```

“ Public ◦

Function◦

```
Function ProcedureName ([argument_list]) [As ReturnType]
    [statements]
End Function
```

Property◦ 3 Get Let/Set◦

```
Property Get|Let|Set PropertyName([argument_list]) [As ReturnType]
    [statements]
End Property
```

◦ Get“”;Let/Set“”◦ - ◦ Sub◦

FunctionProperty Get◦

```
Property Get Foo() As Integer
    Foo = 42
End Property
```

◦

◦

◦

◦ NameReturn Variable◦

1.
```
Function check_even(i as integer) as boolean
if (i mod  2) = 0 then
check_even = True
else
check_even=False
end if
end Function
```

2.
```
Function greet() as String
greet= "Hello Coder!"
end Function
```

◦ Function◦ ◦

```
call greet() 'Similar to a Procedural call just allows the Procedure to use the
             'variable greet
string_1=greet() 'The Return value of the function is used for variable
                 'assignment
```

if◦

```
for i = 1 to 10
if check_even(i) then
msgbox i & " is Even"
else
msgbox i & " is Odd"
end if
next i
```

By refBy val◦

https://riptutorial.com/zh-CN/vba/topic/1474/

12:

VBA◦ DateRange◦

Examples

Property◦

1. Get◦
2. Let Object ◦
3. ObjectSet◦

GetLet / Set ◦ GetLet / Set◦

DateRange

1. StartDate /◦ ◦ mStartDate◦
2. EndDate /◦ ◦ mEndDate◦
3. DaysBetween ◦ ◦ Get◦
4. RangeToCopy ◦ DateRangeSet◦

```
Private mStartDate As Date                ' Module variable to hold the starting date
Private mEndDate As Date                  ' Module variable to hold the ending date

' Return the current value of the starting date
Public Property Get StartDate() As Date
    StartDate = mStartDate
End Property

' Set the starting date value. Note that two methods have the name StartDate
Public Property Let StartDate(ByVal NewValue As Date)
    mStartDate = NewValue
End Property

' Same thing, but for the ending date
Public Property Get EndDate() As Date
    EndDate = mEndDate
End Property

Public Property Let EndDate(ByVal NewValue As Date)
    mEndDate = NewValue
End Property

' Read-only property that returns the number of days between the two dates
Public Property Get DaysBetween() As Integer
    DaysBetween = DateDiff("d", mStartDate, mEndDate)
End Function

' Write-only property that passes an object reference of a range to clone
Public Property Set RangeToCopy(ByRef ExistingRange As DateRange)

Me.StartDate = ExistingRange.StartDate
Me.EndDate = ExistingRange.EndDate
```

```
End Property
```

Sub FunctionProperty

```
Object.Procedure
```

DateRangeSub

```
Public Sub AddDays(ByVal NoDays As Integer)
    mEndDate = mEndDate + NoDays
End Sub
```

Function GetFirstDayOfMonth

```
Public Function GetNextMonthEndDate() As Date
    GetNextMonthEndDate = DateAdd("m", 1, GetFirstDayOfMonth())
End Function

Private Function GetFirstDayOfMonth() As Date
    GetFirstDayOfMonth = DateAdd("d", -DatePart("d", mEndDate), mEndDate)
End Function
```

◦

DateRangeDateRange◦

```
Public Function ContainsRange(ByRef TheRange As DateRange) As Boolean
    ContainsRange = TheRange.StartDate >= Me.StartDate And TheRange.EndDate <= Me.EndDate
End Function
```

Me◦

PrivateVBProject◦

```
'Class List has Instancing set to Private
'In any other module in the SAME project, you can use:

Dim items As List
Set items = New List
```

◦ ProjectAListProjectB4

1. “”ProjectAListPrivatePublicNotCreatable
2. ProjectA“”List◦ ◦ ProjectB ProjectBProjectA◦

```
Public Function CreateList(ParamArray values() As Variant) As List
    Dim tempList As List
    Dim itemCounter As Long
    Set tempList = New List
```

```
    For itemCounter = LBound(values) to UBound(values)
        tempList.Add values(itemCounter)
    Next itemCounter
    Set CreateList = tempList
End Function
```

3. ProjectB Tools..References...ProjectA◦

4. ProjectB ProjectAList

```
Dim items As ProjectA.List
Set items = ProjectA.CreateList("foo","bar")

'Use the items list methods and properties
items.Add "fizz"
Debug.Print items.ToString()
'Destroy the items object
Set items = Nothing
```

https://riptutorial.com/zh-CN/vba/topic/4464/

13:

Examples

◦ ◦ thing1, thing2, thing3 CollectionDictionary ◦

Collection DictionaryRange ◦

VBA

---

camelCase

```
Public Sub ExampleNaming(ByVal inputValue As Long, ByRef inputVariable As Long)

    Dim procedureVariable As Long
    Dim someOtherVariable As String

End Sub
```

PascalCase

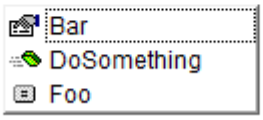
```
Public GlobalVariable As Long
Private ModuleVariable As String
```

SHOUTY_SNAKE_CASE

```
Public Const GLOBAL_CONSTANT As String = "Project Version #1.000.000.001"
Private Const MODULE_CONSTANT As String = "Something relevant to this Module"

Public Sub SomeProcedure()

    Const PROCEDURE_CONSTANT As Long = 10

End Sub
```

IntelliSensePascalCase

```
Option Explicit
Public Const Foo As String = "foo"
Public Bar As String

Sub DoSomething()
    Module1.
End Sub
```



。

“”

；。 VBECtrl + i。

VBA。

iFilestrFile

```
Function bReadFile(ByVal strFile As String, ByRef strData As String) As Boolean
    Dim bRetVal As Boolean
    Dim iFile As Integer

    On Error GoTo CleanFail

    iFile = FreeFile
    Open strFile For Input As #iFile
    Input #iFile, strData

    bRetVal = True

CleanExit:
    Close #iFile
    bReadFile = bRetVal
    Exit Function
CleanFail:
    bRetVal = False
    Resume CleanExit
End Function
```

```
Function CanReadFile(ByVal path As String, ByRef outContent As String) As Boolean
    On Error GoTo CleanFail

    Dim handle As Integer
    handle = FreeFile

    Open path For Input As #handle
    Input #handle, outContent

    Dim result As Boolean
    result = True

CleanExit:
    Close #handle
    CanReadFile = result
    Exit Function
CleanFail:
    result = False
    Resume CleanExit
End Function
```

strDataByRef strData 。

outContent ；out “out”。

IntelliSenseByRef

```
Public Sub DoSomething()
    if CanReadFile(path,
End Sub
```
CanReadFile(*ByVal path As String*, ***outContent As String***) As Boolean

...

Hungarian Notation ◦ ◦ `ByValAs Integer`brevety

```
Public Sub Copy(iX1, iY1, iX2, iY2)
End Sub
```

```
Public Sub Copy(srcColumn, srcRow, dstColumn, dstRow)
End Sub
```

`srcdst`IntelliSense◦

-

```
Type Coordinate
    RowIndex As Long
    ColumnIndex As Long
End Type

Sub Copy(source As Coordinate, destination As Coordinate)
End Sub
```

◦ ◦ ◦

VBA

`PascalCase`

```
Public Sub DoThing()

End Sub

Private Function ReturnSomeValue() As [DataType]

End Function
```

`ObjectName_EventName`

```
Public Sub Workbook_Open()

End Sub

Public Sub Button1_Click()

End Sub
```

VBE;/ - ◦

```
Function bReadFile(ByVal strFile As String, ByRef strData As String) As Boolean
End Function
```

```
Function CanReadFile(ByVal path As String, ByRef outContent As String) As Boolean
End Function
```

CanbBoolean ◦ Canb

```
If CanReadFile(path, content) Then
```

```
If bReadFile(strFile, strData) Then
```

Can IsHas◦ Microsoft ◦

https://riptutorial.com/zh-CN/vba/topic/1184/

14: VBA2GB +

VBA2GB2,147,483,647 - 。 2GB。 ISODVD。 MicrosoftWindows API。

`fciv.exe` 。

MICROSOFT

	。
OpenFile *sFileNam* e As String	sFileName。
CloseFile	。
ReadBytes *ByteCount* As Long	ByteCountVariant。
WriteBytes *DataBytes* As Byte	。
	Windows。
SeekAbsolute *HighPos* As Long *LowPos* As Long	。 VBADWORDSAPI。 4GB。 2GB4GB DWORD。
SeekRelative	+/- 2GB。 64322GB。

MICROSOFT

。 VBA。	
	。
/WriteBytesFlush。	

GetFileHash *sFile*String *uBlockSize*Double *sHashType* String	Blocksize - **HashTypeMD5** **HashTypeSHA1** **HashTypeSHA256** **HashTypeSHA384** **HashTypeSHA512** 。 。

/uFileSize As Double 。 MD5SHA1。

Examples

Class“Random”

```
' How To Seek Past VBA's 2GB File Limit
' Source: https://support.microsoft.com/en-us/kb/189981 (Archived)
' This must be in a Class Module

Option Explicit

Public Enum W32F_Errors
    W32F_UNKNOWN_ERROR = 45600
    W32F_FILE_ALREADY_OPEN
    W32F_PROBLEM_OPENING_FILE
    W32F_FILE_ALREADY_CLOSED
    W32F_Problem_seeking
End Enum

Private Const W32F_SOURCE = "Win32File Object"
Private Const GENERIC_WRITE = &H40000000
Private Const GENERIC_READ = &H80000000
Private Const FILE_ATTRIBUTE_NORMAL = &H80
Private Const CREATE_ALWAYS = 2
Private Const OPEN_ALWAYS = 4
Private Const INVALID_HANDLE_VALUE = -1

Private Const FILE_BEGIN = 0, FILE_CURRENT = 1, FILE_END = 2

Private Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000

Private Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA" ( _
    ByVal dwFlags As Long, _
    lpSource As Long, _
    ByVal dwMessageId As Long, _
    ByVal dwLanguageId As Long, _
    ByVal lpBuffer As String, _
    ByVal nSize As Long, _
    Arguments As Any) As Long

Private Declare Function ReadFile Lib "kernel32" ( _
    ByVal hFile As Long, _
    lpBuffer As Any, _
    ByVal nNumberOfBytesToRead As Long, _
    lpNumberOfBytesRead As Long, _
    ByVal lpOverlapped As Long) As Long

Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

Private Declare Function WriteFile Lib "kernel32" ( _
    ByVal hFile As Long, _
    lpBuffer As Any, _
    ByVal nNumberOfBytesToWrite As Long, _
    lpNumberOfBytesWritten As Long, _
    ByVal lpOverlapped As Long) As Long

Private Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" ( _
    ByVal lpFileName As String, _
    ByVal dwDesiredAccess As Long, _
    ByVal dwShareMode As Long, _
    ByVal lpSecurityAttributes As Long, _
    ByVal dwCreationDisposition As Long, _
    ByVal dwFlagsAndAttributes As Long, _
```

```
    ByVal hTemplateFile As Long) As Long

Private Declare Function SetFilePointer Lib "kernel32" ( _
    ByVal hFile As Long, _
    ByVal lDistanceToMove As Long, _
    lpDistanceToMoveHigh As Long, _
    ByVal dwMoveMethod As Long) As Long

Private Declare Function FlushFileBuffers Lib "kernel32" (ByVal hFile As Long) As Long

Private hFile As Long, sFName As String, fAutoFlush As Boolean

Public Property Get FileHandle() As Long
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    FileHandle = hFile
End Property

Public Property Get FileName() As String
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    FileName = sFName
End Property

Public Property Get IsOpen() As Boolean
    IsOpen = hFile <> INVALID_HANDLE_VALUE
End Property

Public Property Get AutoFlush() As Boolean
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    AutoFlush = fAutoFlush
End Property

Public Property Let AutoFlush(ByVal NewVal As Boolean)
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    fAutoFlush = NewVal
End Property

Public Sub OpenFile(ByVal sFileName As String)
    If hFile <> INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_OPEN, sFName
    End If
    hFile = CreateFile(sFileName, GENERIC_WRITE Or GENERIC_READ, 0, 0, OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, 0)
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_PROBLEM_OPENING_FILE, sFileName
    End If
    sFName = sFileName
End Sub

Public Sub CloseFile()
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    CloseHandle hFile
```

```
    sFName = ""
    fAutoFlush = False
    hFile = INVALID_HANDLE_VALUE
End Sub

Public Function ReadBytes(ByVal ByteCount As Long) As Variant
    Dim BytesRead As Long, Bytes() As Byte
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    ReDim Bytes(0 To ByteCount - 1) As Byte
    ReadFile hFile, Bytes(0), ByteCount, BytesRead, 0
    ReadBytes = Bytes
End Function

Public Sub WriteBytes(DataBytes() As Byte)
    Dim fSuccess As Long, BytesToWrite As Long, BytesWritten As Long
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    BytesToWrite = UBound(DataBytes) - LBound(DataBytes) + 1
    fSuccess = WriteFile(hFile, DataBytes(LBound(DataBytes)), BytesToWrite, BytesWritten, 0)
    If fAutoFlush Then Flush
End Sub

Public Sub Flush()
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    FlushFileBuffers hFile
End Sub

Public Sub SeekAbsolute(ByVal HighPos As Long, ByVal LowPos As Long)
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    LowPos = SetFilePointer(hFile, LowPos, HighPos, FILE_BEGIN)
End Sub

Public Sub SeekRelative(ByVal Offset As Long)
    Dim TempLow As Long, TempErr As Long
    If hFile = INVALID_HANDLE_VALUE Then
        RaiseError W32F_FILE_ALREADY_CLOSED
    End If
    TempLow = SetFilePointer(hFile, Offset, ByVal 0&, FILE_CURRENT)
    If TempLow = -1 Then
        TempErr = Err.LastDllError
        If TempErr Then
            RaiseError W32F_Problem_seeking, "Error " & TempErr & "." & vbCrLf & CStr(TempErr)
        End If
    End If
End Sub

Private Sub Class_Initialize()
    hFile = INVALID_HANDLE_VALUE
End Sub

Private Sub Class_Terminate()
    If hFile <> INVALID_HANDLE_VALUE Then CloseHandle hFile
End Sub
```

```
Private Sub RaiseError(ByVal ErrorCode As W32F_Errors, Optional sExtra)
    Dim Win32Err As Long, Win32Text As String
    Win32Err = Err.LastDllError
    If Win32Err Then
        Win32Text = vbCrLf & "Error " & Win32Err & vbCrLf & _
        DecodeAPIErrors(Win32Err)
    End If
    Select Case ErrorCode
        Case W32F_FILE_ALREADY_OPEN
            Err.Raise W32F_FILE_ALREADY_OPEN, W32F_SOURCE, "The file '" & sExtra & "' is
already open." & Win32Text
        Case W32F_PROBLEM_OPENING_FILE
            Err.Raise W32F_PROBLEM_OPENING_FILE, W32F_SOURCE, "Error opening '" & sExtra &
"'." & Win32Text
        Case W32F_FILE_ALREADY_CLOSED
            Err.Raise W32F_FILE_ALREADY_CLOSED, W32F_SOURCE, "There is no open file."
        Case W32F_Problem_seeking
            Err.Raise W32F_Problem_seeking, W32F_SOURCE, "Seek Error." & vbCrLf & sExtra
        Case Else
            Err.Raise W32F_UNKNOWN_ERROR, W32F_SOURCE, "Unknown error." & Win32Text
    End Select
End Sub

Private Function DecodeAPIErrors(ByVal ErrorCode As Long) As String
    Dim sMessage As String, MessageLength As Long
    sMessage = Space$(256)
    MessageLength = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, 0&, ErrorCode, 0&, sMessage,
256&, 0&)
    If MessageLength > 0 Then
        DecodeAPIErrors = Left(sMessage, MessageLength)
    Else
        DecodeAPIErrors = "Unknown Error."
    End If
End Function
```

```
Private Const HashTypeMD5 As String = "MD5" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.md5cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA1 As String = "SHA1" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha1cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA256 As String = "SHA256" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha256cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA384 As String = "SHA384" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha384cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA512 As String = "SHA512" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha512cryptoserviceprovider(v=vs.110).aspx

Private uFileSize As Double ' Comment out if not testing performance by FileHashes()

Sub FileHashes()
    Dim tStart As Date, tFinish As Date, sHash As String, aTestFiles As Variant, oTestFile As
Variant, aBlockSizes As Variant, oBlockSize As Variant
    Dim BLOCKSIZE As Double

    ' This performs performance testing on different file sizes and block sizes
    aBlockSizes = Array("2^12-1", "2^13-1", "2^14-1", "2^15-1", "2^16-1", "2^17-1", "2^18-1",
"2^19-1", "2^20-1", "2^21-1", "2^22-1", "2^23-1", "2^24-1", "2^25-1", "2^26-1")
    aTestFiles = Array("C:\ISO\clonezilla-live-2.2.2-37-amd64.iso",
"C:\ISO\HPIP201.2014_0902.29.iso",
"C:\ISO\SW_DVD5_Windows_Vista_Business_W32_32BIT_English.ISO",
"C:\ISO\Win10_1607_English_x64.iso",
```

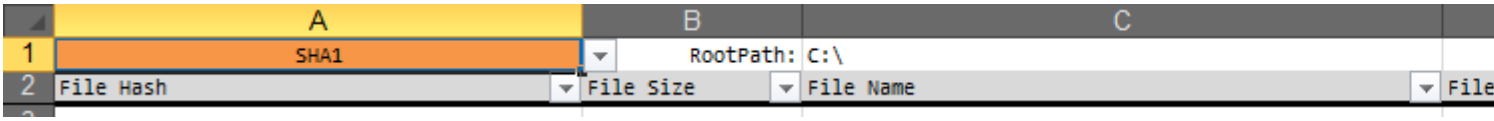
```
"C:\ISO\SW_DVD9_Windows_Svr_Std_and_DataCtr_2012_R2_64Bit_English.ISO")
    Debug.Print "Test files: " & Join(aTestFiles, " | ")
    Debug.Print "BlockSizes: " & Join(aBlockSizes, " | ")
    For Each oTestFile In aTestFiles
        Debug.Print oTestFile
        For Each oBlockSize In aBlockSizes
            BLOCKSIZE = Evaluate(oBlockSize)
            tStart = Now
            sHash = GetFileHash(CStr(oTestFile), BLOCKSIZE, HashTypeMD5)
            tFinish = Now
            Debug.Print sHash, uFileSize, Format(tFinish - tStart, "hh:mm:ss"), oBlockSize & "
(" & BLOCKSIZE & ")"
        Next
    Next
End Sub

Private Function GetFileHash(ByVal sFile As String, ByVal uBlockSize As Double, ByVal
sHashType As String) As String
    Dim oFSO As Object ' "Scripting.FileSystemObject"
    Dim oCSP As Object ' One of the "CryptoServiceProvider"
    Dim oRnd As Random ' "Random" Class by Microsoft, must be in the same file
    Dim uBytesRead As Double, uBytesToRead As Double, bDone As Boolean
    Dim aBlock() As Byte, aBytes As Variant ' Arrays to store bytes
    Dim aHash() As Byte, sHash As String, i As Long
    'Dim uFileSize As Double ' Un-Comment if GetFileHash() is to be used individually

    Set oRnd = New Random ' Class by Microsoft: Random
    Set oFSO = CreateObject("Scripting.FileSystemObject")
    Set oCSP = CreateObject("System.Security.Cryptography." & sHashType &
"CryptoServiceProvider")

    If oFSO Is Nothing Or oRnd Is Nothing Or oCSP Is Nothing Then
        MsgBox "One or more required objects cannot be created"
        GoTo CleanUp
    End If

    uFileSize = oFSO.GetFile(sFile).Size ' FILELEN() has 2GB max!
    uBytesRead = 0
    bDone = False
    sHash = String(oCSP.HashSize / 4, "0") ' Each hexadecimal has 4 bits

    Application.ScreenUpdating = False
    ' Process the file in chunks of uBlockSize or less
    If uFileSize = 0 Then
        ReDim aBlock(0)
        oCSP.TransformFinalBlock aBlock, 0, 0
        bDone = True
    Else
        With oRnd
            .OpenFile sFile
            Do
                If uBytesRead + uBlockSize < uFileSize Then
                    uBytesToRead = uBlockSize
                Else
                    uBytesToRead = uFileSize - uBytesRead
                    bDone = True
                End If
                ' Read in some bytes
                aBytes = .ReadBytes(uBytesToRead)
                aBlock = aBytes
                If bDone Then
```

```
                    oCSP.TransformFinalBlock aBlock, 0, uBytesToRead
                    uBytesRead = uBytesRead + uBytesToRead
                Else
                    uBytesRead = uBytesRead + oCSP.TransformBlock(aBlock, 0, uBytesToRead,
aBlock, 0)
                End If
                DoEvents
            Loop Until bDone
            .CloseFile
        End With
    End If
    If bDone Then
        ' convert Hash byte array to an hexadecimal string
        aHash = oCSP.hash
        For i = 0 To UBound(aHash)
            Mid$(sHash, i * 2 + (aHash(i) > 15) + 2) = Hex(aHash(i))
        Next
    End If
    Application.ScreenUpdating = True
    ' Clean up
    oCSP.Clear
CleanUp:
    Set oFSO = Nothing
    Set oRnd = Nothing
    Set oCSP = Nothing
    GetFileHash = sHash
End Function
```

BLOCKSIZE = 131071 **2 ^ 17-1**Windows 7 x6432Office 2010*2 ^ 16-165535* 。 `2^27-1` 。

146800640	Clonezilla-2.2.2-37-amd64.iso
798210048	HPIP201.2014_0902.29.iso
2073016320	SW_DVD5_Windows_Vista_Business_W32_32BIT_English.ISO
4380387328	Win10_1607_English_x64.iso
5400115200	SW_DVD9_Windows_Svr_Std_and_DataCtr_2012_R2_64Bit_English.ISO

。




```
Option Explicit

Private Const HashTypeMD5 As String = "MD5" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.md5cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA1 As String = "SHA1" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha1cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA256 As String = "SHA256" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha256cryptoserviceprovider(v=vs.110).aspx
```

```
Private Const HashTypeSHA384 As String = "SHA384" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha384cryptoserviceprovider(v=vs.110).aspx
Private Const HashTypeSHA512 As String = "SHA512" ' https://msdn.microsoft.com/en-
us/library/system.security.cryptography.sha512cryptoserviceprovider(v=vs.110).aspx

Private Const BLOCKSIZE As Double = 131071 ' 2^17-1

Private oFSO As Object
Private oCSP As Object
Private oRnd As Random ' Requires the Class from Microsoft https://support.microsoft.com/en-
us/kb/189981
Private sHashType As String
Private sRootFDR As String
Private oRng As Range
Private uFileCount As Double

Sub AllFileHashes() ' Active-X button calls this
    Dim oWS As Worksheet
    ' | A: FileHash | B: FileSize | C: FileName | D: FilaName and Path | E: File Last
Modification Time | F: Time required to calculate has code (seconds)
    With ThisWorkbook
        ' Clear All old entries on all worksheets
        For Each oWS In .Worksheets
            Set oRng = Intersect(oWS.UsedRange, oWS.UsedRange.Offset(2))
            If Not oRng Is Nothing Then oRng.ClearContents
        Next
        With .Worksheets(1)
            sHashType = Trim(.Range("A1").Value) ' Range(A1)
            sRootFDR = Trim(.Range("C1").Value) ' Range(C1) Column B for file size
            If Len(sHashType) = 0 Or Len(sRootFDR) = 0 Then Exit Sub
            Set oRng = .Range("A3") ' First entry on First Page
        End With
    End With

    uFileCount = 0
    If oRnd Is Nothing Then Set oRnd = New Random ' Class by Microsoft: Random
    If oFSO Is Nothing Then Set oFSO = CreateObject("Scripting.FileSystemObject") ' Just to
get correct FileSize
    If oCSP Is Nothing Then Set oCSP = CreateObject("System.Security.Cryptography." &
sHashType & "CryptoServiceProvider")

    ProcessFolder oFSO.GetFolder(sRootFDR)

    Application.StatusBar = False
    Application.ScreenUpdating = True
    oCSP.Clear
    Set oCSP = Nothing
    Set oRng = Nothing
    Set oFSO = Nothing
    Set oRnd = Nothing
    Debug.Print "Total file count: " & uFileCount
End Sub

Private Sub ProcessFolder(ByRef oFDR As Object)
    Dim oFile As Object, oSubFDR As Object, sHash As String, dStart As Date, dFinish As Date
    Application.ScreenUpdating = False
    For Each oFile In oFDR.Files
        uFileCount = uFileCount + 1
        Application.StatusBar = uFileCount & ": " & Right(oFile.Path, 255 - Len(uFileCount) -
2)
        oCSP.Initialize ' Reinitialize the CryptoServiceProvider
```

```
        dStart = Now
        sHash = GetFileHash(oFile, BLOCKSIZE, sHashType)
        dFinish = Now
        With oRng
            .Value = sHash
            .Offset(0, 1).Value = oFile.Size ' File Size in bytes
            .Offset(0, 2).Value = oFile.Name ' File name with extension
            .Offset(0, 3).Value = oFile.Path ' Full File name and Path
            .Offset(0, 4).Value = FileDateTime(oFile.Path) ' Last modification timestamp of
file
            .Offset(0, 5).Value = dFinish - dStart ' Time required to calculate hash code
        End With
        If oRng.Row = Rows.Count Then
            ' Max rows reached, start on Next sheet
            If oRng.Worksheet.Index + 1 > ThisWorkbook.Worksheets.Count Then
                MsgBox "All rows in all worksheets have been used, please create more sheets"
                End
            End If
            Set oRng = ThisWorkbook.Sheets(oRng.Worksheet.Index + 1).Range("A3")
            oRng.Worksheet.Activate
        Else
            ' Move to next row otherwise
            Set oRng = oRng.Offset(1)
        End If
    Next
    'Application.StatusBar = False
    Application.ScreenUpdating = True
    oRng.Activate
    For Each oSubFDR In oFDR.SubFolders
        ProcessFolder oSubFDR
    Next
End Sub

Private Function GetFileHash(ByVal sFile As String, ByVal uBlockSize As Double, ByVal
sHashType As String) As String
    Dim uBytesRead As Double, uBytesToRead As Double, bDone As Boolean
    Dim aBlock() As Byte, aBytes As Variant ' Arrays to store bytes
    Dim aHash() As Byte, sHash As String, i As Long, oTmp As Variant
    Dim uFileSize As Double ' Un-Comment if GetFileHash() is to be used individually

    If oRnd Is Nothing Then Set oRnd = New Random ' Class by Microsoft: Random
    If oFSO Is Nothing Then Set oFSO = CreateObject("Scripting.FileSystemObject") ' Just to
get correct FileSize
    If oCSP Is Nothing Then Set oCSP = CreateObject("System.Security.Cryptography." &
sHashType & "CryptoServiceProvider")

    If oFSO Is Nothing Or oRnd Is Nothing Or oCSP Is Nothing Then
        MsgBox "One or more required objects cannot be created"
        Exit Function
    End If

    uFileSize = oFSO.GetFile(sFile).Size ' FILELEN() has 2GB max
    uBytesRead = 0
    bDone = False
    sHash = String(oCSP.HashSize / 4, "0") ' Each hexadecimal is 4 bits

    ' Process the file in chunks of uBlockSize or less
    If uFileSize = 0 Then
        ReDim aBlock(0)
        oCSP.TransformFinalBlock aBlock, 0, 0
        bDone = True
```

```
    Else
        With oRnd
            On Error GoTo CannotOpenFile
            .OpenFile sFile
            Do
                If uBytesRead + uBlockSize < uFileSize Then
                    uBytesToRead = uBlockSize
                Else
                    uBytesToRead = uFileSize - uBytesRead
                    bDone = True
                End If
                ' Read in some bytes
                aBytes = .ReadBytes(uBytesToRead)
                aBlock = aBytes
                If bDone Then
                    oCSP.TransformFinalBlock aBlock, 0, uBytesToRead
                    uBytesRead = uBytesRead + uBytesToRead
                Else
                    uBytesRead = uBytesRead + oCSP.TransformBlock(aBlock, 0, uBytesToRead,
aBlock, 0)
                End If
                DoEvents
            Loop Until bDone
            .CloseFile
CannotOpenFile:
            If Err.Number <> 0 Then ' Change the hash code to the Error description
                oTmp = Split(Err.Description, vbCrLf)
                sHash = oTmp(1) & ":" & oTmp(2)
            End If
        End With
    End If
    If bDone Then
        ' convert Hash byte array to an hexadecimal string
        aHash = oCSP.hash
        For i = 0 To UBound(aHash)
            Mid$(sHash, i * 2 + (aHash(i) > 15) + 2) = Hex(aHash(i))
        Next
    End If
    GetFileHash = sHash
End Function
```

VBA2GB + https://riptutorial.com/zh-CN/vba/topic/8786/vba2gb-plus

15: FileSystemObject

Scripting.FileSystemObject。 。

Examples

Dir$。 Dir$pathName。 - False 。

```
Public Function FileExists(pathName As String) As Boolean
    If InStr(1, pathName, "*") Or InStr(1, pathName, "?") Then
        'Exit Function   'Return False on wild-cards.
        Err.Raise 52    'Raise error on wild-cards.
    End If
    FileExists = Dir$(pathName) <> vbNullString
End Function
```

Dir $

Dir$()attributesvbDirectory。 pathName \ 。 52 - “”。 On Error Resume Next 。 Dir$..\Foo\Bar 。

```
Public Function FolderExists(ByVal pathName As String) As Boolean
    'Uncomment the "On Error" line if paths with wild-cards should return False
    'instead of raising an error.
    'On Error Resume Next
    If pathName = vbNullString Or Right$(pathName, 1) <> "\" Then
        Exit Function
    End If
    FolderExists = Dir$(pathName, vbDirectory) <> vbNullString
End Function
```

ChDir

ChDir。 VBADir$。 。 Dir$。

```
Public Function FolderExists(ByVal pathName As String) As Boolean
    'Cache the current working directory
    Dim cached As String
    cached = CurDir$

    On Error Resume Next
    ChDir pathName
    FolderExists = Err.Number = 0
    On Error GoTo 0
    'Change back to the cached working directory.
    ChDir cached
End Function
```

“ ”FolderExists。

MkDir。 C:\Foo UNC \\Server\Foo ..\Foo Foo。

UNC\Foo。 。

```
Public Sub MakeNewDirectory(ByVal pathName As String)
    'MkDir will fail if the directory already exists.
    If FolderExists(pathName) Then Exit Sub
    'This may still fail due to permissions, etc.
    MkDir pathName
End Sub
```

RmDir。 MkDir。 Windows rd shell75“/”。

```
Public Sub DeleteDirectory(ByVal pathName As String)
    If Right$(pathName, 1) <> "\" Then
        pathName = pathName & "\"
    End If
    'Rmdir will fail if the directory doesn't exist.
    If Not FolderExists(pathName) Then Exit Sub
    'Rmdir will fail if the directory contains files.
    If Dir$(pathName & "*") <> vbNullString Then Exit Sub

    'Rmdir will fail if the directory contains directories.
    Dim subDir As String
    subDir = Dir$(pathName & "*", vbDirectory)
    Do
        If subDir <> "." And subDir <> ".." Then Exit Sub
        subDir = Dir$(, vbDirectory)
    Loop While subDir <> vbNullString

    'This may still fail due to permissions, etc.
    RmDir pathName
End Sub
```

FileSystemObject https://riptutorial.com/zh-CN/vba/topic/5706/filesystemobject

16:

InStrVBAInStrInStrRev。

Examples

InStr

```
Const baseString As String = "Foo Bar"
Dim containsBar As Boolean

'Check if baseString contains "bar" (case insensitive)
containsBar = InStr(1, baseString, "bar", vbTextCompare) > 0
'containsBar = True

'Check if baseString contains bar (case insensitive)
containsBar = InStr(1, baseString, "bar", vbBinaryCompare) > 0
'containsBar = False
```

InStr

```
Const baseString As String = "Foo Bar"
Dim containsBar As Boolean

Dim posB As Long
posB = InStr(1, baseString, "B", vbBinaryCompare)
'posB = 5
```

InStrRev

```
Const baseString As String = "Foo Bar"
Dim containsBar As Boolean

'Find the position of the last "B"
Dim posX As Long
'Note the different number and order of the paramters for InStrRev
posX = InStrRev(baseString, "X", -1, vbBinaryCompare)
'posX = 0
```

https://riptutorial.com/zh-CN/vba/topic/3480/

17:

Examples

Option Explicit “”◦ Variant ◦

```
Public Sub ExampleDeclaration()

    someVariable = 10                   '
    someOtherVariable = "Hello World"
    'Both of these variables are of the Variant type.

End Sub
```

Option Explicit someVariablesomeOtherVariableDim◦

```
Option Explicit

Public Sub ExampleDeclaration()

    Dim someVariable As Long
    someVariable = 10

    Dim someOtherVariable As String
    someOtherVariable = "Hello World"

End Sub
```

Option Explicit◦

VBA◦

- Dim; ◦
- Private; ◦
- Friend; ◦
- Public; ◦
- Public; ◦

◦

◦

Dim

```
Dim identifierName [As Type][, identifierName [As Type], ...]
```

[As Type]◦ ◦ String

```
Dim identifierName As String
```

Variant

```
Dim identifierName 'As Variant is implicit
```

VBA

```
Dim someString As String, someVariant, someValue As Long
```

[As Type] 'Variant'。

```
Dim integer1, integer2, integer3 As Integer 'Only integer3 is an Integer.
                                            'The rest are Variant.
```

Static。 VBA Static“”

```
Private Sub DoSomething()
    Static values As Collection
    If values Is Nothing Then
        Set values = New Collection
        values.Add "foo"
        values.Add "bar"
    End If
    DoSomethingElse values
End Sub
```

valuesStatic local; Nothing。 Set -。 DoSomethingElseDoSomething。

VBAStatic -。 static...。 static。 VBA Static。

StaticPrivate - ; - .....Static。

Dim;Private

```
Option Explicit
Dim privateField1 As Long 'same as Private privateField2 as Long
Private privateField2 As Long 'same as Dim privateField2 as Long
```

Private;DimPrivatePublic。 Dim -

“”

- Private。
- Dim。
- Dim。

“”

-

Dim/◦

- Private◦
- Public◦ *

*PublicGlobal◦

◦ Public

```
Public PublicField As Long
```

VBA◦

global / publicFriend

```
Friend FriendField As Long
```

VBAAPI◦

```
Friend FriendField As Long 'public within the project, aka for "friend" code
Public PublicField As Long 'public within and beyond the project
```

◦

ThisWorkbook ThisDocument Worksheet UserForm ◦

```
'> Class1
Option Explicit
Public PublicField As Long
```

```
'> Module1
Option Explicit
Public Sub DoSomething()
    'Class1.PublicField means nothing here
    With New Class1
        .PublicField = 42
    End With
    'Class1.PublicField means nothing here
End Sub
```

Private ◦ Property◦ Property Get

```
Option Explicit
Private encapsulated As Long

Public Property Get SomeValue() As Long
    SomeValue = encapsulated
End Property
```

```
Public Sub DoSomething()
    encapsulated = 42
End Sub
```

PublicFriend◦

- Property Let◦
- Property Set◦

Const

◦

Const◦ ◦

```
Public Const GLOBAL_CONSTANT As String = "Project Version #1.000.000.001"
Private Const MODULE_CONSTANT As String = "Something relevant to this Module"

Public Sub ExampleDeclaration()

    Const SOME_CONSTANT As String = "Hello World"

    Const PI As Double = 3.141592653

End Sub
```

◦

```
Public Const GLOBAL_CONSTANT = "Project Version #1.000.000.001" 'Still a string
Public Sub ExampleDeclaration()

    Const SOME_CONSTANT = "Hello World"           'Still a string
    Const DERIVED_CONSTANT = SOME_CONSTANT        'DERIVED_CONSTANT is also a string
    Const VAR_CONSTANT As Variant = SOME_CONSTANT 'VAR_CONSTANT is Variant/String

    Const PI = 3.141592653        'Still a double
    Const DERIVED_PI = PI         'DERIVED_PI is also a double
    Const VAR_PI As Variant = PI  'VAR_PI is Variant/Double

End Sub
```

Variant◦

String

```
'This is a valid 5 character string constant
Const FOO As String = "ABCDE"

'This is not valid syntax for a 5 character string constant
Const FOO As String * 5 = "ABCDE"
```

Dim◦

- Private◦
- Public◦
- Friend VBA
- GlobalPublic - Public◦ ◦

◦

```
Private ModuleVariable As String
Public GlobalVariable As String

Private Sub ModuleProcedure()

    ModuleVariable = "This can only be done from within the same Module"

End Sub

Public Sub GlobalProcedure()

    GlobalVariable = "This can be done from any Module within this Project"

End Sub
```

Sub◦

FunctionUDF◦

Option Private ModuleUDF◦

◦ ◦ As [DataType]◦

```
Public Sub ExampleDeclaration()

        Dim someInteger% '% Equivalent to "As Integer"
        Dim someLong&    '& Equivalent to "As Long"
        Dim someDecimal@ '@ Equivalent to "As Currency"
        Dim someSingle!  '! Equivalent to "As Single"
        Dim someDouble#  '# Equivalent to "As Double"
        Dim someString$  '$ Equivalent to "As String"

        Dim someLongLong^  '^ Equivalent to "As LongLong" in 64-bit VBA hosts
End Sub
```

```
Dim strFile$
Dim iFile%
```

```
Dim path As String
Dim handle As Integer
```

◦ 32,768Integer

```
Dim foo 'implicit Variant
foo = 42& ' foo is now a Long
foo = 42# ' foo is now a Double
Debug.Print TypeName(42!) ' prints "Single"
```

。

```
'Calls procedure DoSomething and passes a literal 42 as a Long using a type hint
DoSomething 42&

'Calls procedure DoSomething and passes a literal 42 explicitly converted to a Long
DoSomething CLng(42)
```

VariantString$。 Variant String - 。

```
Debug.Print Left(foo, 2)  'Left returns a Variant
Debug.Print Left$(foo, 2) 'Left$ returns a String
```

- VBA.Conversion.Error - > VBA.Conversion.Error $
- VBA.Conversion.Hex - > VBA.Conversion.Hex $
- VBA.Conversion.Oct - > VBA.Conversion.Oct $
- VBA.Conversion.Str - > VBA.Conversion.Str $
- VBA.FileSystem.CurDir - > VBA.FileSystem.CurDir $
- VBA。[_ HiddenModule] .Input - > VBA。[_ HiddenModule] .Input $
- VBA。[_ HiddenModule] .InputB - > VBA。[_ HiddenModule] .InputB $
- VBA.Interaction.Command - > VBA.Interaction.Command $
- VBA.Interaction.Environ - > VBA.Interaction.Environ $
- VBA.Strings.Chr - > VBA.Strings.Chr $
- VBA.Strings.ChrB - > VBA.Strings.ChrB $
- VBA.Strings.ChrW - > VBA.Strings.ChrW $
- VBA.Strings.Format - > VBA.Strings.Format $
- VBA.Strings.LCase - > VBA.Strings.LCase $
- VBA.Strings.Left - > VBA.Strings.Left $
- VBA.Strings.LeftB - > VBA.Strings.LeftB $
- VBA.Strings.LTtrim - > VBA.Strings.LTrim $
- VBA.Strings.Mid - > VBA.Strings.Mid $
- VBA.Strings.MidB - > VBA.Strings.MidB $
- VBA.Strings.Right - > VBA.Strings.Right $
- VBA.Strings.RightB - > VBA.Strings.RightB $
- VBA.Strings.RTrim - > VBA.Strings.RTrim $
- VBA.Strings.Space - > VBA.Strings.Space $
- VBA.Strings.Str - > VBA.Strings.Str $
- VBA.Strings.String - > VBA.Strings.String $
- VBA.Strings.Trim - > VBA.Strings.Trim $
- VBA.Strings.UCase - > VBA.Strings.UCase $

◦ LeftB_Var_LeftLeft$B_Str_Left◦

VBA $◦ Word Basic$◦

VBA;◦

```
Public Sub TwoTypesOfStrings()

    Dim FixedLengthString As String * 5 ' declares a string of 5 characters
    Dim NormalString As String

    Debug.Print FixedLengthString       ' Prints "     "
    Debug.Print NormalString            ' Prints ""

    FixedLengthString = "123"           ' FixedLengthString now equals "123  "
    NormalString = "456"                ' NormalString now equals "456"

    FixedLengthString = "123456"        ' FixedLengthString now equals "12345"
    NormalString = "456789"             ' NormalString now equals "456789"

End Sub
```

Sub◦ “”◦

'helper'◦

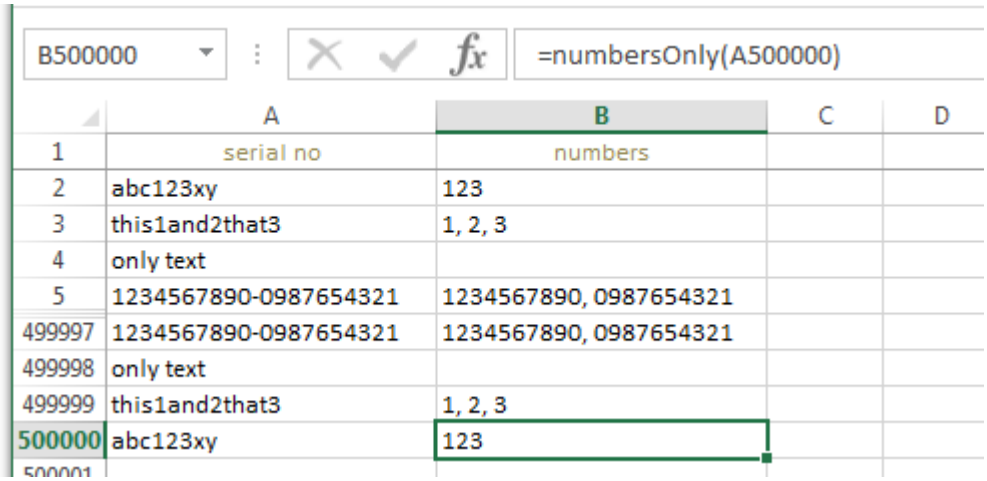
1Scripting.Dictionary

```
Option Explicit

Sub main()
    Dim w As Long

    For w = 1 To Worksheets.Count
        processDictionary ws:=Worksheets(w)
    Next w
End Sub

Sub processDictionary(ws As Worksheet)
    Dim i As Long, rng As Range
    Static dict As Object

    If dict Is Nothing Then
        'initialize and set the dictionary object
        Set dict = CreateObject("Scripting.Dictionary")
        dict.CompareMode = vbTextCompare
    Else
        'remove all pre-existing dictionary entries
        ' this may or may not be desired if a single dictionary of entries
        ' from all worksheets is preferred
        dict.RemoveAll
    End If

    With ws

        'work with a fresh dictionary object for each worksheet
        ' without constructing/destructing a new object each time
```

```
        ' or do not clear the dictionary upon subsequent uses and
        ' build a dictionary containing entries from all worksheets

    End With
End Sub
```

2VBScript.RegExpUDF

```
Option Explicit

Function numbersOnly(str As String, _
                     Optional delim As String = ", ")
    Dim n As Long, nums() As Variant
    Static rgx As Object, cmat As Object

    'with rgx as static, it only has to be created once
    'this is beneficial when filling a long column with this UDF
    If rgx Is Nothing Then
        Set rgx = CreateObject("VBScript.RegExp")
    Else
        Set cmat = Nothing
    End If

    With rgx
        .Global = True
        .MultiLine = True
        .Pattern = "[0-9]{1,999}"
        If .Test(str) Then
            Set cmat = .Execute(str)
            'resize the nums array to accept the matches
            ReDim nums(cmat.Count - 1)
            'populate the nums array with the matches
            For n = LBound(nums) To UBound(nums)
                nums(n) = cmat.Item(n)
            Next n
            'convert the nums array to a delimited string
            numbersOnly = Join(nums, delim)
        Else
            numbersOnly = vbNullString
        End If
    End With
End Function
```



UDF

* UDF500K
    - **Dim rgx As Object** 148.74
    - **Static rgx As Object** 26.07
* 。
。

UDF。UDF。。

- 。
- 。
- 。

Visual Basic

https://riptutorial.com/zh-CN/vba/topic/877/

18:

。。2GB。VBAUnicode。

Examples

```
Const appName As String = "The App For That"
```

```
Dim surname As String 'surname can accept strings of variable length
surname = "Smith"
surname = "Johnson"
```

```
'Declare and assign a 1-character fixed-width string
Dim middleInitial As String * 1 'middleInitial must be 1 character in length
middleInitial = "M"

'Declare and assign a 2-character fixed-width string `stateCode`,
'must be 2 characters in length
Dim stateCode As String * 2
stateCode = "TX"
```

```
'Declare, dimension and assign a string array with 3 elements
Dim departments(2) As String
departments(0) = "Engineering"
departments(1) = "Finance"
departments(2) = "Marketing"

'Declare an undimensioned string array and then dynamically assign with
'the results of a function that returns a string array
Dim stateNames() As String
stateNames = VBA.Strings.Split("Texas;California;New York", ";")

'Declare, dimension and assign a fixed-width string array
Dim stateCodes(2) As String * 2
stateCodes(0) = "TX"
stateCodes(1) = "CA"
stateCodes(2) = "NY"
```

Mid

VBAMidMid 。

MidMid。

```
Dim surname As String
surname = "Smith"

'Use the Mid statement to change the 3rd character in a string
Mid(surname, 3, 1) = "y"
Debug.Print surname
```

```
'Output:
'Smyth
```

MidB◦ MidB1MidB(surname, 5, 2) = "y" ◦

◦ ASCII◦

```
Dim bytes() As Byte
Dim example As String

example = "Testing."
bytes = example             'Direct assignment.

'Loop through the characters. Step 2 is used due to wide encoding.
Dim i As Long
For i = LBound(bytes) To UBound(bytes) Step 2
    Debug.Print Chr$(bytes(i))  'Prints T, e, s, t, i, n, g, .
Next

Dim reverted As String
reverted = bytes            'Direct assignment.
Debug.Print reverted        'Prints "Testing."
```

https://riptutorial.com/zh-CN/vba/topic/3446/

19:

Examples

=VBA。“”。

```
Dim source(0 to 2) As Long
Dim destinationLong() As Long
Dim destinationDouble() As Double

destinationLong = source     ' copies contents of source into destinationLong
destinationDouble = source   ' does not compile
```

。“”。 。

```
Dim source() As Long
ReDim source(0 To 2)

Dim fixed(0 To 2) As Long
Dim dynamic() As Long

fixed = source   ' does not compile
dynamic = source ' does compile

Dim dynamic2() As Long
ReDim dynamic2(0 to 6, 3 to 99)

dynamic2 = source ' dynamic2 now has dimension (0 to 2)
```

。

```
Dim source(0 To 2) As Long
Dim destination() As Long

source(0) = 3
source(1) = 1
source(2) = 4

destination = source
destination(0) = 2

Debug.Print source(0); source(1); source(2)                ' outputs: 3 1 4
Debug.Print destination(0); destination(1); destination(2) ' outputs: 2 1 4
```

。 - 。 。

```
Dim source(0 To 2) As Range
Dim destination() As Range

Set source(0) = Range("A1"): source(0).Value = 3
Set source(1) = Range("A2"): source(1).Value = 1
Set source(2) = Range("A3"): source(2).Value = 4
```

```
destination = source

Set destination(0) = Range("A4")   'reference changed in destination but not source

destination(0).Value = 2           'affects an object only in destination
destination(1).Value = 5           'affects an object in both source and destination

Debug.Print source(0); source(1); source(2)                  ' outputs 3 5 4
Debug.Print destination(0); destination(1); destination(2)   ' outputs 2 5 4
```

◦ “”◦

```
Dim var As Variant
Dim source(0 To 2) As Range
Dim destination() As Range

var = source
destination = var

var = 5
destination = var  ' throws runtime error
```

()◦

```
Function arrayOfPiDigits() As Long()
    Dim outputArray(0 To 2) As Long

    outputArray(0) = 3
    outputArray(1) = 1
    outputArray(2) = 4

    arrayOfPiDigits = outputArray
End Function
```

◦

```
Sub arrayExample()

    Dim destination() As Long
    Dim var As Variant

    destination = arrayOfPiDigits()
    var = arrayOfPiDigits

    Debug.Print destination(0)          ' outputs 3
    Debug.Print var(1)                  ' outputs 1
    Debug.Print arrayOfPiDigits()(2)    ' outputs 4

End Sub
```

◦ ◦

◦ VBA`ByRef`◦

```
Sub threePiDigits(ByRef destination() As Long)
    destination(0) = 3
    destination(1) = 1
    destination(2) = 4
End Sub

Sub printPiDigits()
    Dim digits(0 To 2) As Long

    threePiDigits digits
    Debug.Print digits(0); digits(1); digits(2) ' outputs 3 1 4
End Sub
```

Class

/

```
' Class Module 'MathConstants'
Sub threePiDigits(ByRef destination() As Long)
    ReDim destination(0 To 2)

    destination(0) = 3
    destination(1) = 1
    destination(2) = 4
End Sub

' Standard Code Module
Sub printPiDigits()
    Dim digits() As Long
    Dim mathConsts As New MathConstants

    mathConsts.threePiDigits digits
    Debug.Print digits(0); digits(1); digits(2) ' outputs 3 1 4
End Sub
```

()◦

```
Function countElements(ByRef arr() As Double) As Long
    countElements = UBound(arr) - LBound(arr) + 1
End Function
```

◦ myFunction(arr()) VBAByRef ◦ myFunction(ByVal arr())“ArrayByRef”VBEAuto Syntax Check“” ◦

◦

```
Sub testArrayPassing()
    Dim source(0 To 1) As Long
    source(0) = 3
    source(1) = 1

    Debug.Print doubleAndSum(source)  ' outputs 8
    Debug.Print source(0); source(1)  ' outputs 6 2
End Sub
```

```
Function doubleAndSum(ByRef arr() As Long)
    arr(0) = arr(0) * 2
    arr(1) = arr(1) * 2
    doubleAndSum = arr(0) + arr(1)
End Function
```

。

```
Function doubleAndSum(ByRef arr() As Long)
    doubleAndSum = arr(0) * 2 + arr(1) * 2
End Function
```

。

```
Function doubleAndSum(ByRef arr() As Long)
    Dim copyOfArr() As Long
    copyOfArr = arr

    copyOfArr(0) = copyOfArr(0) * 2
    copyOfArr(1) = copyOfArr(1) * 2

    doubleAndSum = copyOfArr(0) + copyOfArr(1)
End Function
```

https://riptutorial.com/zh-CN/vba/topic/9069/-

20:

VBA

- `Left` / `Left$`
- `Right` / `Right$`
- `Mid` / `Mid$`
- `Trim` / `Trim$`

onverhead$ -suffixed/。

`Null`$ -suffixed“null” - 。

Examples

LeftLeft $3

```
Const baseString As String = "Foo Bar"

Dim leftText As String
leftText = Left$(baseString, 3)
'leftText = "Foo"
```

RightRight $3

```
Const baseString As String = "Foo Bar"
Dim rightText As String
rightText = Right$(baseString, 3)
'rightText = "Bar"
```

MidMid $

```
Const baseString As String = "Foo Bar"

'Get the string starting at character 2 and ending at character 6
Dim midText As String
midText = Mid$(baseString, 2, 5)
'midText = "oo Ba"
```

Trim

```
'Trim the leading and trailing spaces in a string
Const paddedText As String = "    Foo Bar    "
Dim trimmedText As String
trimmedText = Trim$(paddedText)
'trimmedText = "Foo Bar"
```

https://riptutorial.com/zh-CN/vba/topic/3481/

21: -

VBAIDE。。

unicode。

Examples

“

VBA"/""VBA"""。

```
'The following 2 lines produce the same output
Debug.Print "The man said, ""Never use air-quotes"""
Debug.Print "The man said, " & """ & "Never use air-quotes" & """

'Output:
'The man said, "Never use air-quotes"
'The man said, "Never use air-quotes"
```

VBA1023100-150。。

```
Debug.Print "Lorem ipsum dolor sit amet, consectetur adipiscing elit. " & _
            "Integer hendrerit maximus arcu, ut elementum odio varius " & _
            "nec. Integer ipsum enim, iaculis et egestas ac, condiment" & _
            "um ut tellus."
'Output:
'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer hendrerit maximus arcu, ut
elementum odio varius nec. Integer ipsum enim, iaculis et egestas ac, condimentum ut tellus.
```

VBA。

```
Dim loremIpsum As String

'Assign the first part of the string
loremIpsum = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. " & _
             "Integer hendrerit maximus arcu, ut elementum odio varius "
'Re-assign with the previous value AND the next section of the string
loremIpsum = loremIpsum & _
            "nec. Integer ipsum enim, iaculis et egestas ac, condiment" & _
            "um ut tellus."

Debug.Print loremIpsum

'Output:
'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer hendrerit maximus arcu, ut
elementum odio varius nec. Integer ipsum enim, iaculis et egestas ac, condimentum ut tellus.
```

VBA

VBA

- vbCrCarriage-Return'C“\ r”。
- vbLf'C“\ n”。
- vbCrLfWindows
- vbTab
- vbNullString“”

。

```
Debug.Print "Hello " & vbCrLf & "World"
'Output:
'Hello
'World

Debug.Print vbTab & "Hello" & vbTab & "World"
'Output:
'    Hello    World

Dim EmptyString As String
EmptyString = vbNullString
Debug.Print EmptyString = ""
'Output:
'True
```

vbNullString""。 VBAvbNullString。 ""StringVariant

```
Debug.Print StrPtr(vbNullString)    'Prints 0.
Debug.Print StrPtr("")              'Prints a memory address.
```

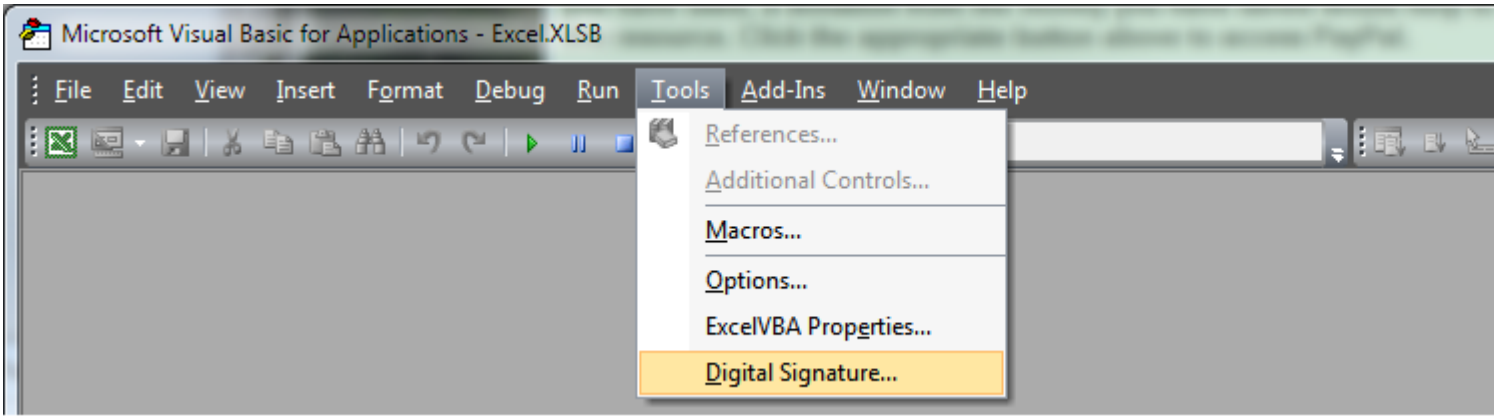
- https://riptutorial.com/zh-CN/vba/topic/3445/----

22: VBA/

Examples

SELFCERT.EXE

Office*VBA*>>VBAProject.OTM ◦

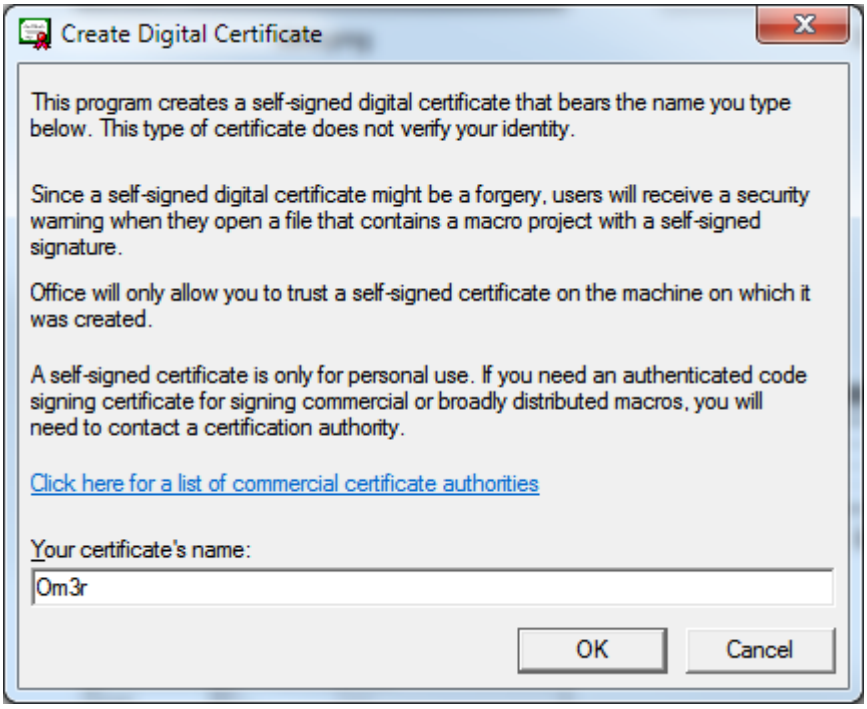


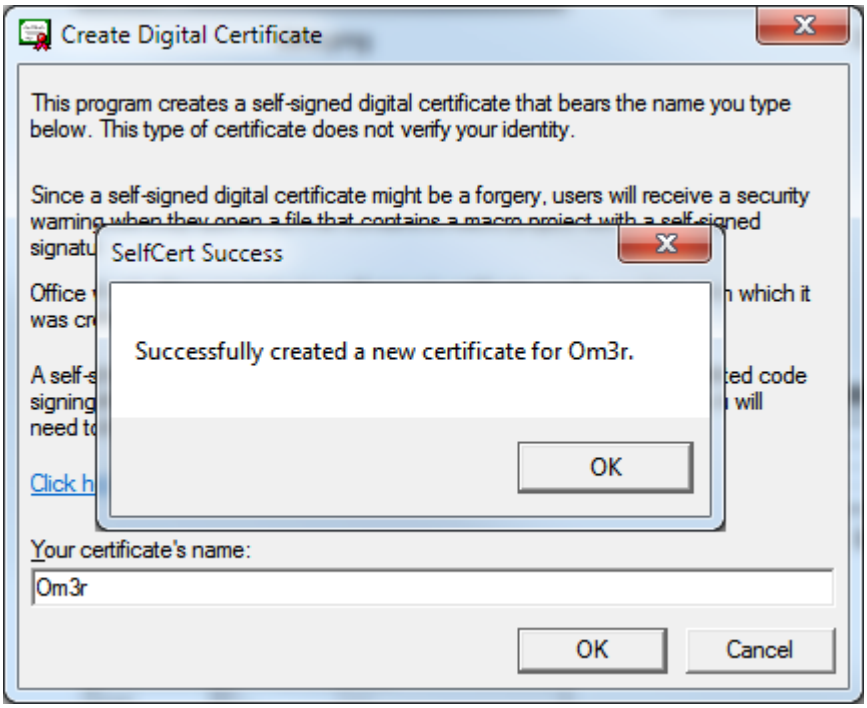
OfficePC◦

SELFCERT.EXE Office

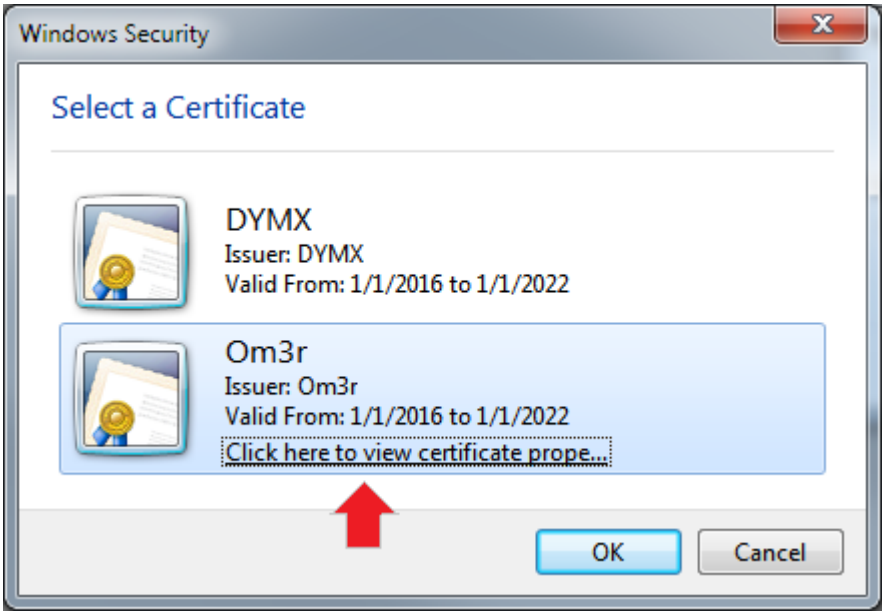
VBA ◦

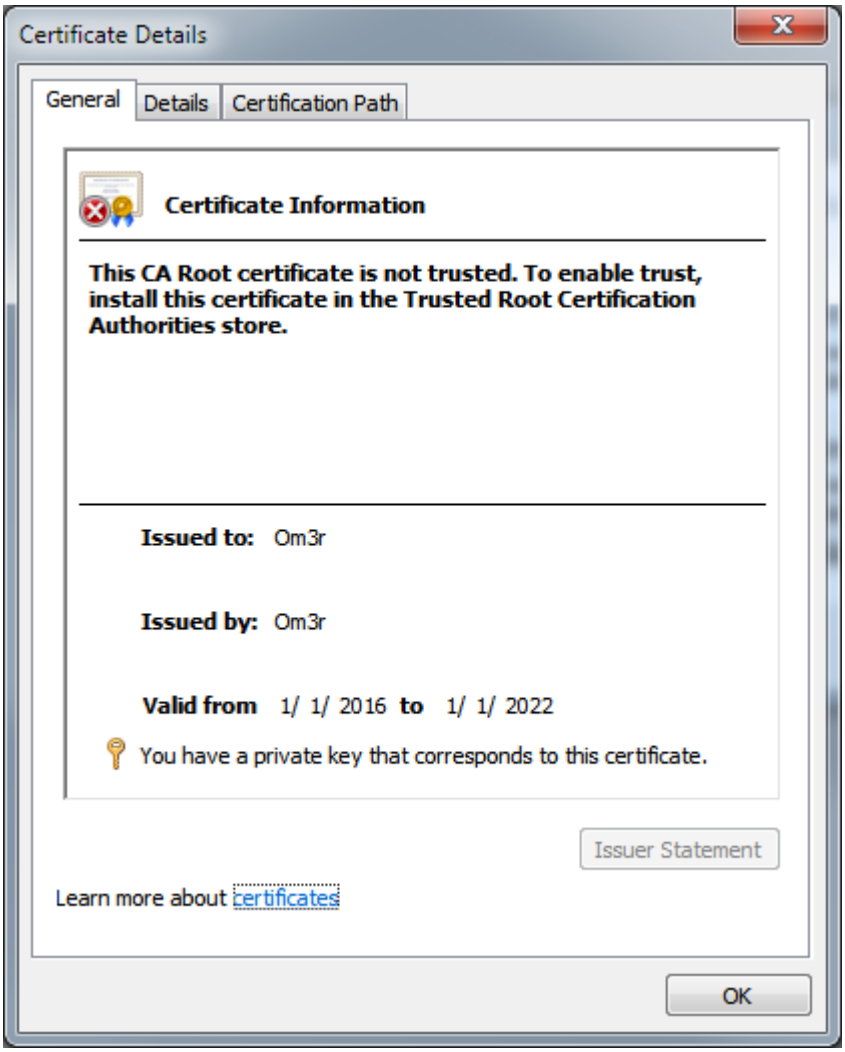
“”。




Create Digital Certificate
This program creates a self-signed digital certificate that bears the name you type below. This type of certificate does not verify your identity.
Since a self-signed digital certificate might be a forgery, users will receive a security
SelfCert Success
Successfully created a new certificate for Om3r.
OK
Your certificate's name:
Om3r
OK
Cancel

SELFCERT。

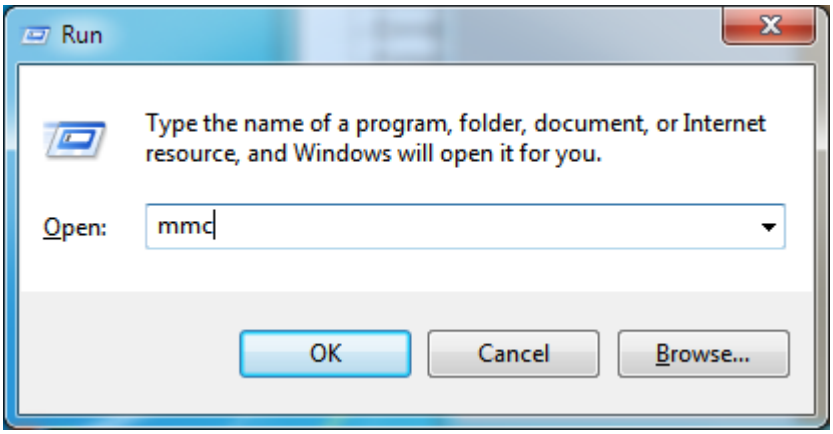

Windows Security
Select a Certificate
DYMX
Issuer: DYMX
Valid From: 1/1/2016 to 1/1/2022
Om3r
Issuer: Om3r
Valid From: 1/1/2016 to 1/1/2022
Click here to view certificate prope...
OK
Cancel



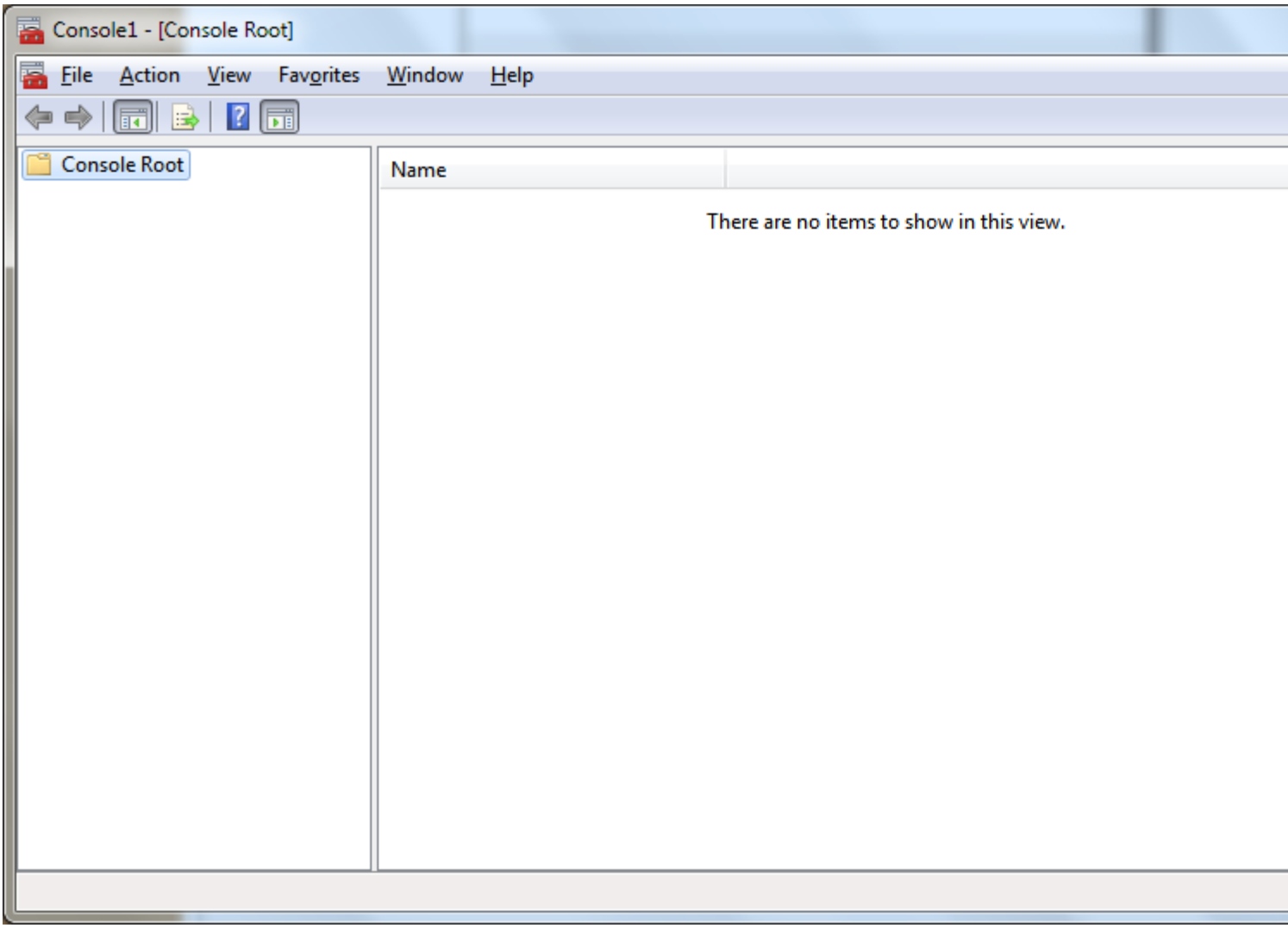
。

“”>“”>“”。 >>。

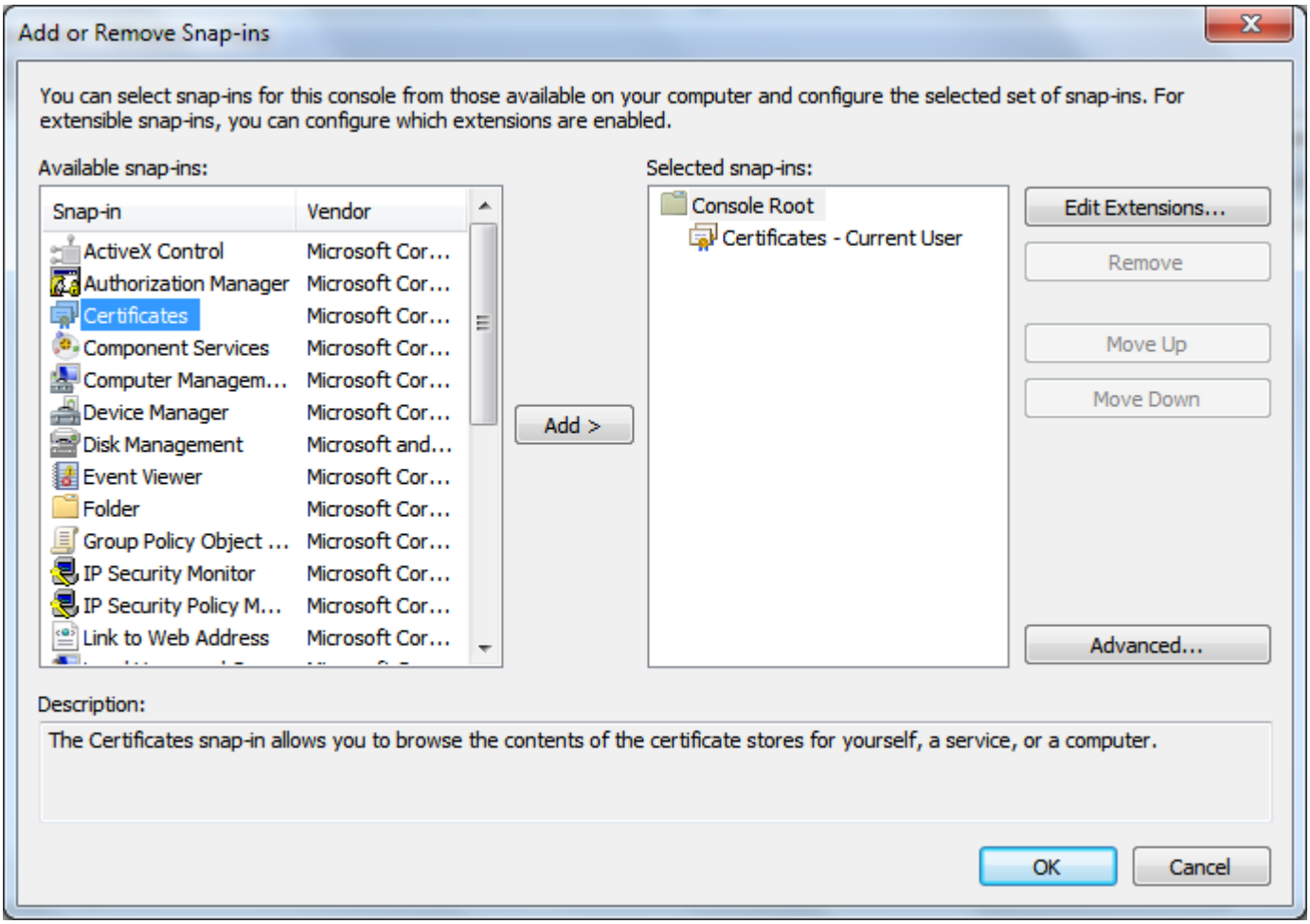
Windows + r“”。 “mmc”“”。



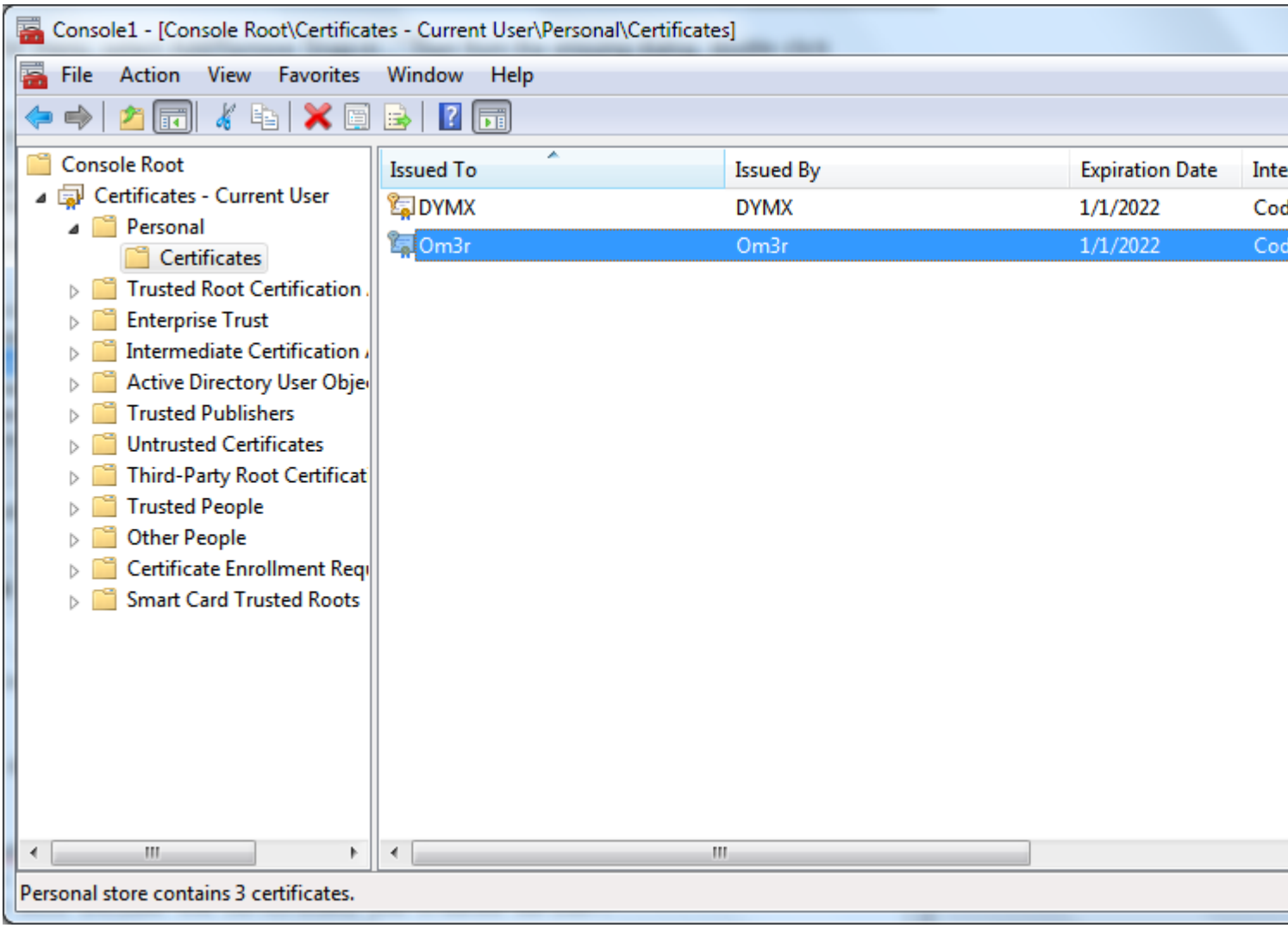
Microsoft。


Console1 - [Console Root]
File Action View Favorites Window Help
Console Root
Name
There are no items to show in this view.

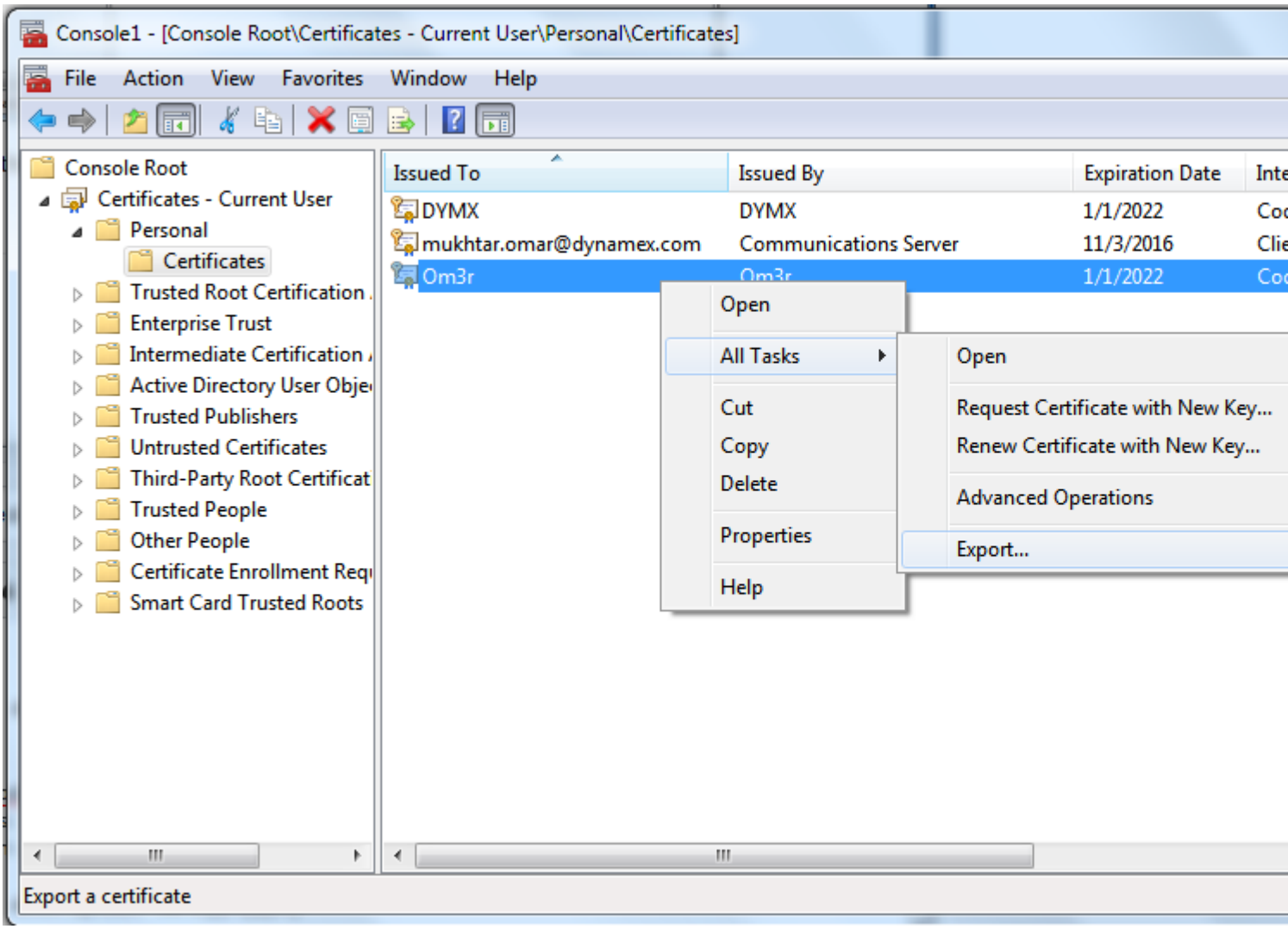
“”“/”...“”“”


Add or Remove Snap-ins
You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.
Available snap-ins:
Snap-in
Vendor
ActiveX Control
Microsoft Cor...
Authorization Manager
Microsoft Cor...
Certificates
Microsoft Cor...
Component Services
Microsoft Cor...
Computer Managem...
Microsoft Cor...
Device Manager
Microsoft Cor...
Disk Management
Microsoft and...
Event Viewer
Microsoft Cor...
Folder
Microsoft Cor...
Group Policy Object ...
Microsoft Cor...
IP Security Monitor
Microsoft Cor...
IP Security Policy M...
Microsoft Cor...
Link to Web Address
Microsoft Cor...
Add >
Selected snap-ins:
Console Root
Certificates - Current User
Edit Extensions...
Remove
Move Up
Move Down
Advanced...
Description:
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.
OK
Cancel

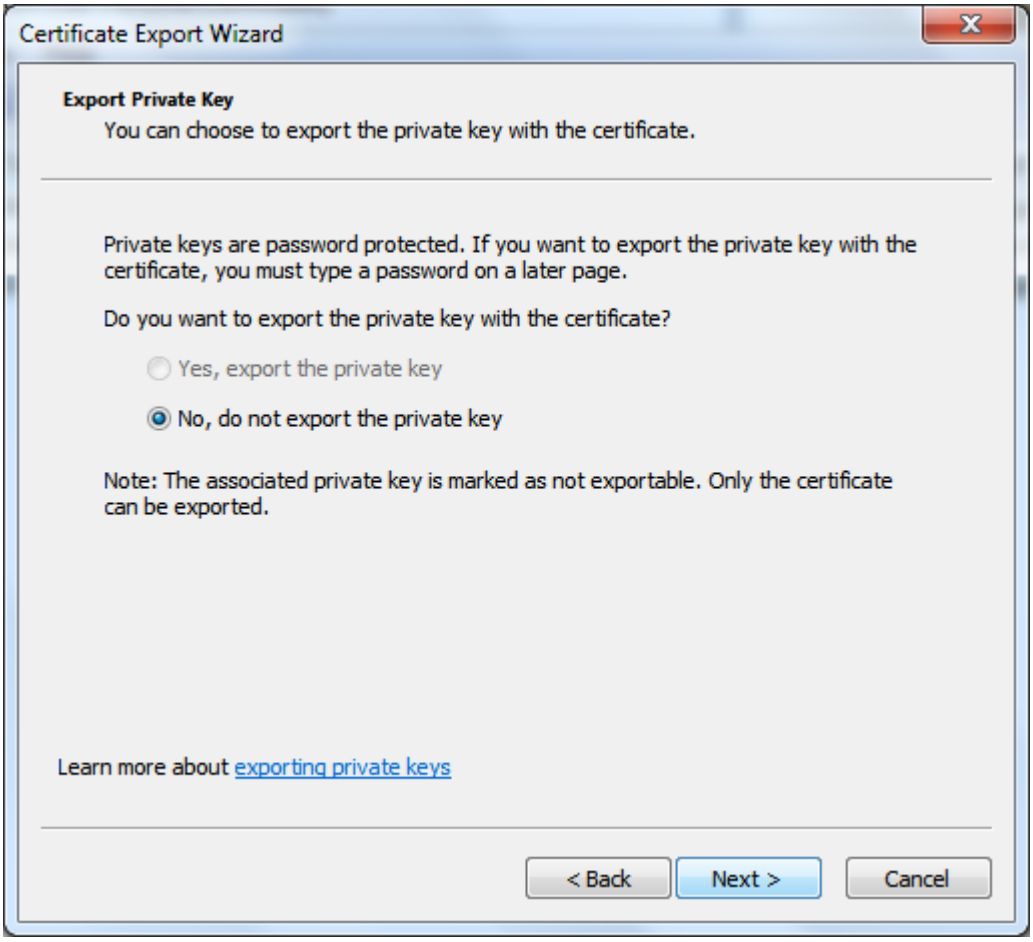
“ - ”。


Console1 - [Console Root\Certificates - Current User\Personal\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Certificates
Trusted Root Certification
Enterprise Trust
Intermediate Certification
Active Directory User Obje
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificat
Trusted People
Other People
Certificate Enrollment Requ
Smart Card Trusted Roots
Issued To
Issued By
Expiration Date
Inte
DYMX
DYMX
1/1/2022
Cod
Om3r
Om3r
1/1/2022
Cod
Personal store contains 3 certificates.

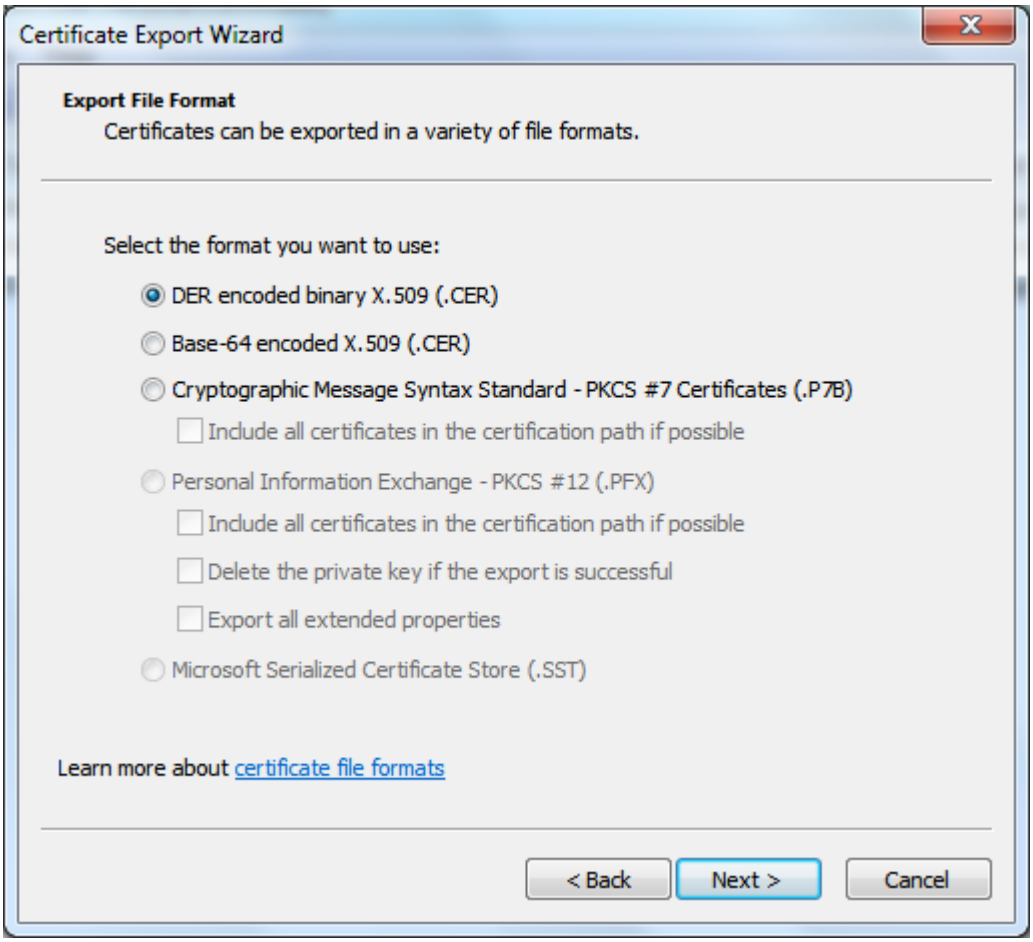
“”>“”


Console1 - [Console Root\Certificates - Current User\Personal\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Certificates
Trusted Root Certification
Enterprise Trust
Intermediate Certification
Active Directory User Obje
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificat
Trusted People
Other People
Certificate Enrollment Req
Smart Card Trusted Roots
Issued To
Issued By
Expiration Date
DYMX
DYMX
1/1/2022
mukhtar.omar@dynamex.com
Communications Server
11/3/2016
Om3r
1/1/2022
Open
All Tasks
Cut
Copy
Delete
Properties
Help
Open
Request Certificate with New Key...
Renew Certificate with New Key...
Advanced Operations
Export...
Export a certificate

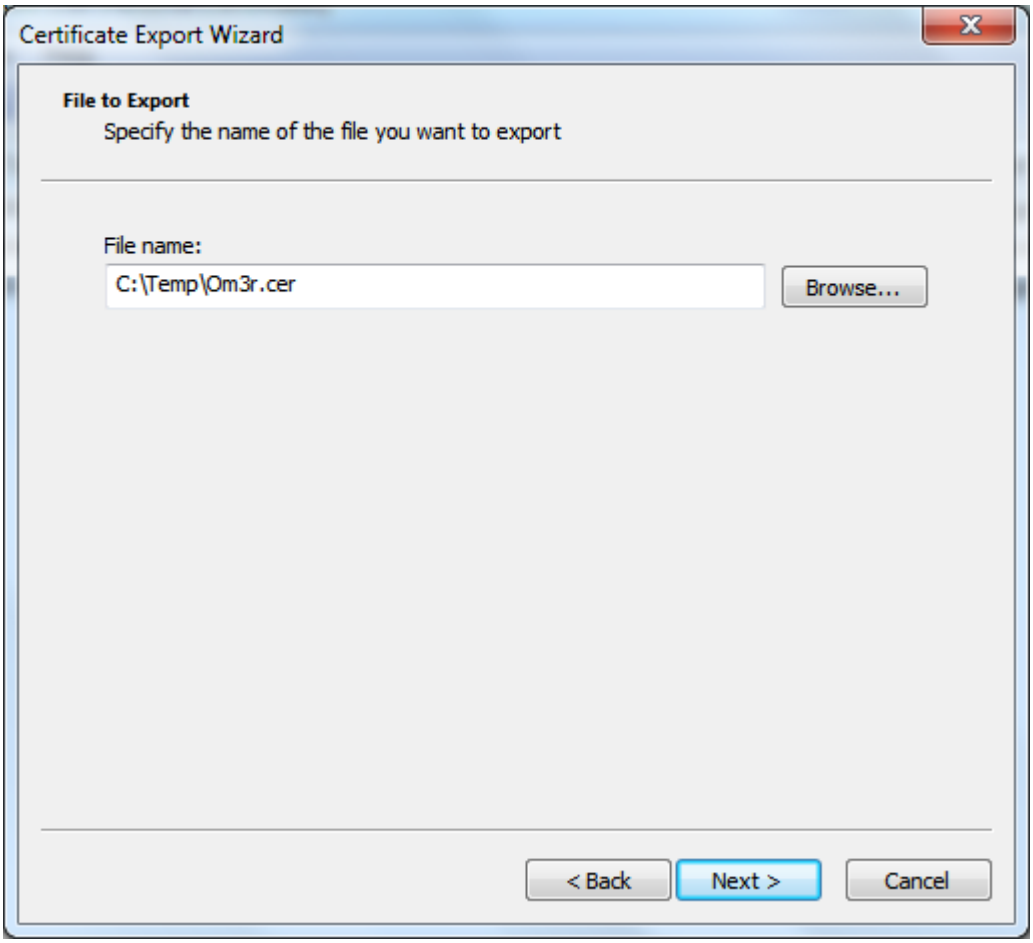

Certificate Export Wizard
Welcome to the Certificate Export Wizard
This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.
A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.
To continue, click Next.
< Back
Next >
Cancel


Certificate Export Wizard
Export Private Key
You can choose to export the private key with the certificate.
Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.
Do you want to export the private key with the certificate?
Yes, export the private key
No, do not export the private key
Note: The associated private key is marked as not exportable. Only the certificate can be exported.
Learn more about exporting private keys
< Back
Next >
Cancel

“”


Certificate Export Wizard
Export File Format
Certificates can be exported in a variety of file formats.
Select the format you want to use:
DER encoded binary X.509 (.CER)
Base-64 encoded X.509 (.CER)
Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
Include all certificates in the certification path if possible
Personal Information Exchange - PKCS #12 (.PFX)
Include all certificates in the certification path if possible
Delete the private key if the export is successful
Export all extended properties
Microsoft Serialized Certificate Store (.SST)
Learn more about certificate file formats
< Back
Next >
Cancel

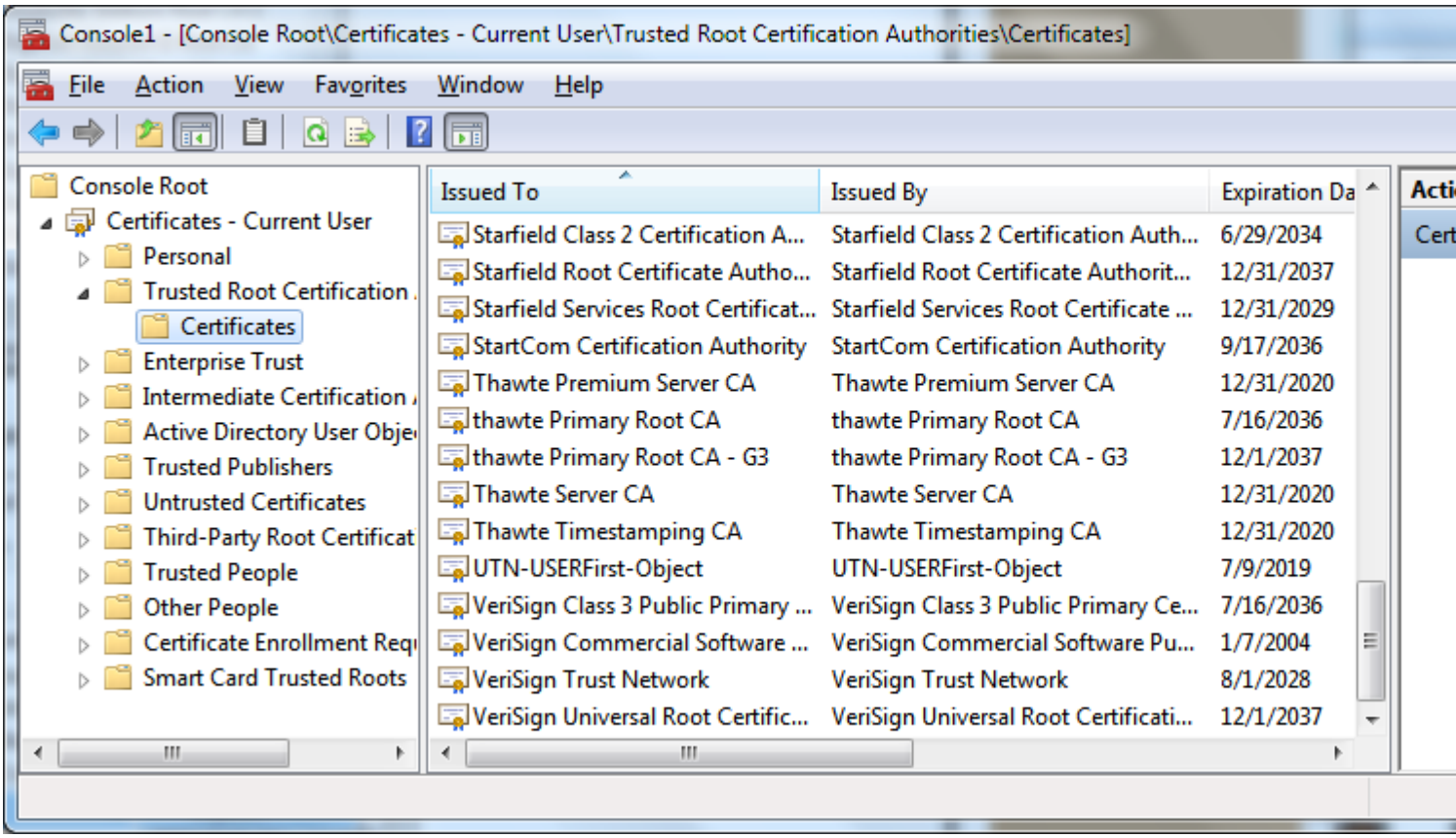
。 “”。


Certificate Export Wizard
File to Export
Specify the name of the file you want to export
File name:
C:\Temp\Om3r.cer
Browse...
< Back
Next >
Cancel

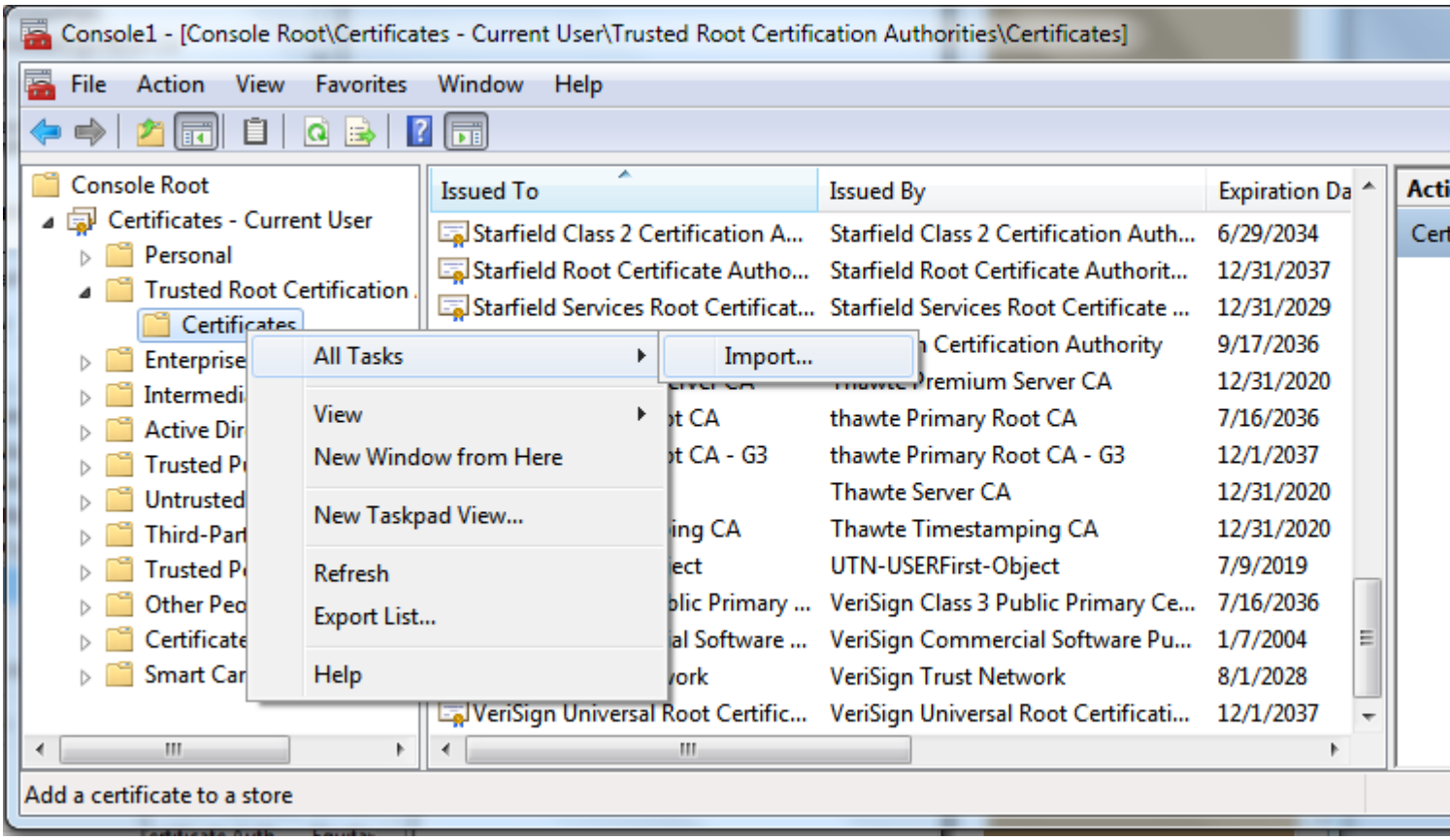
“”

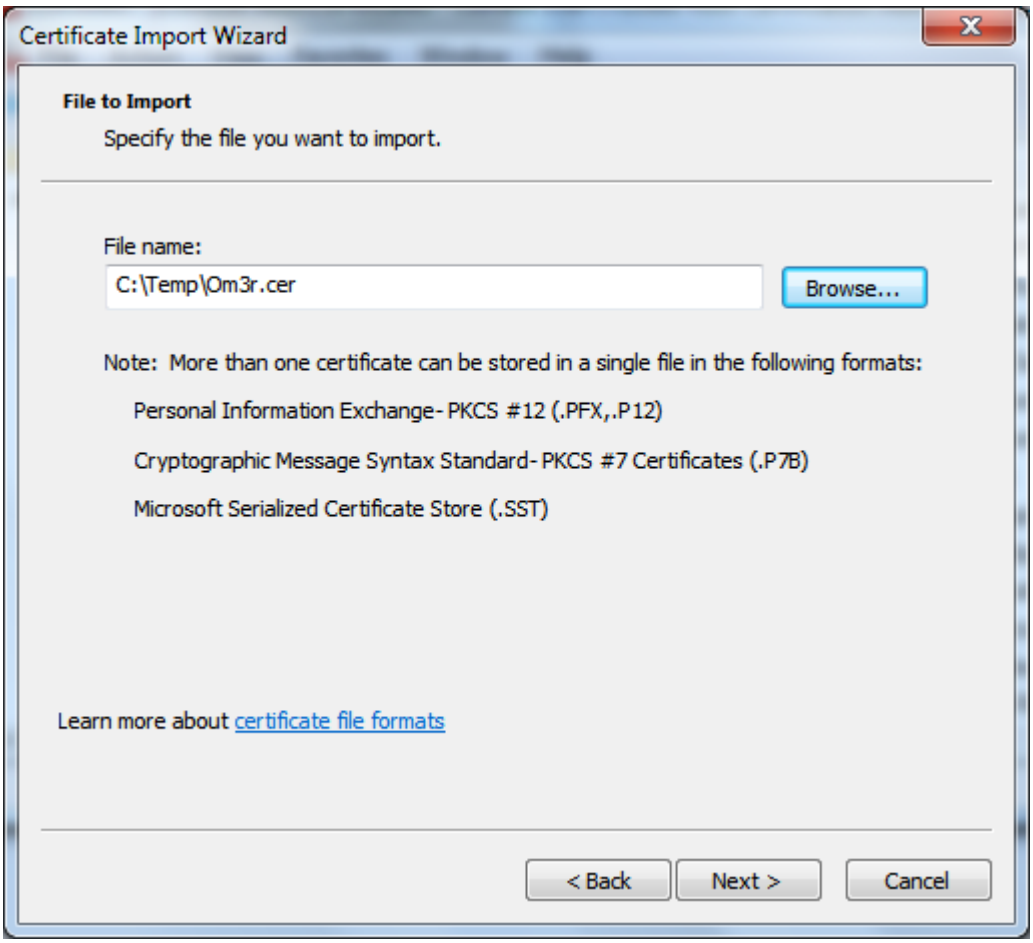
。

“ ”“”“ ”。

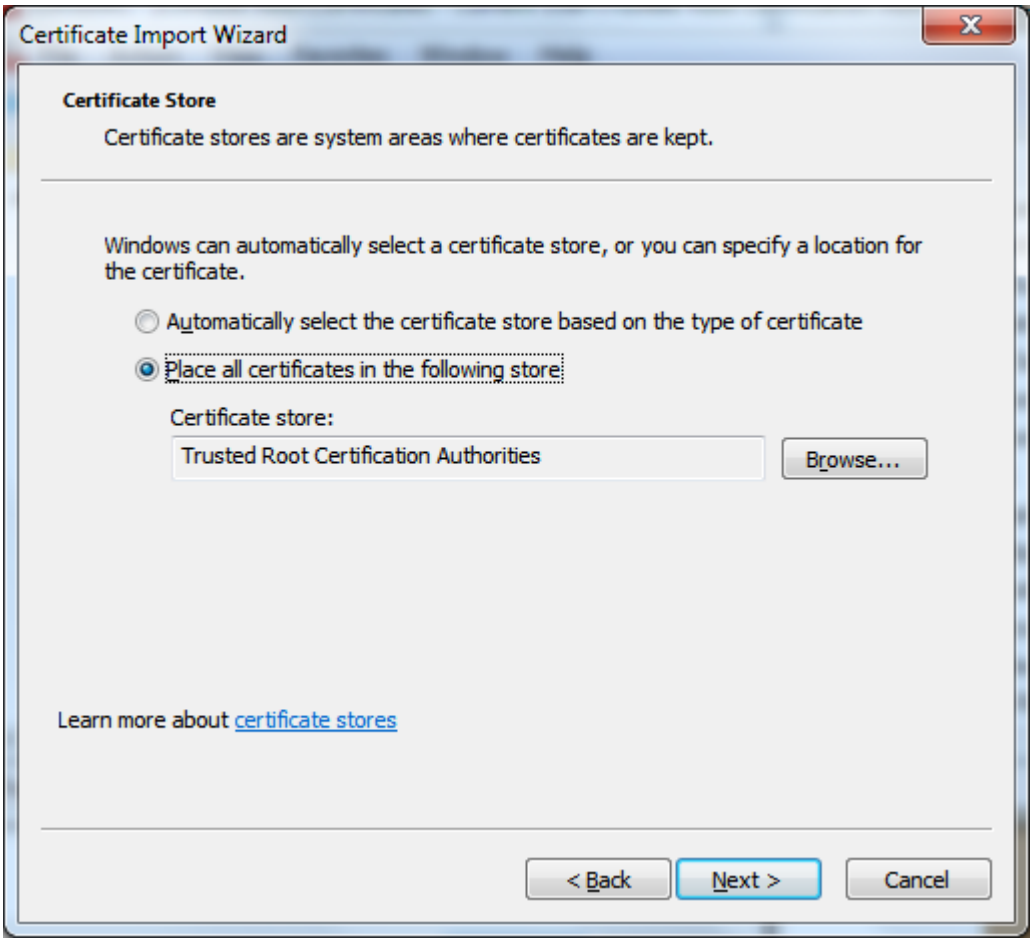

Console1 - [Console Root\Certificates - Current User\Trusted Root Certification Authorities\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Trusted Root Certification
Certificates
Enterprise Trust
Intermediate Certification
Active Directory User Obje
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificat
Trusted People
Other People
Certificate Enrollment Req
Smart Card Trusted Roots
Issued To
Issued By
Expiration Da
Starfield Class 2 Certification A... Starfield Class 2 Certification Auth... 6/29/2034
Starfield Root Certificate Autho... Starfield Root Certificate Authorit... 12/31/2037
Starfield Services Root Certificat... Starfield Services Root Certificate ... 12/31/2029
StartCom Certification Authority StartCom Certification Authority 9/17/2036
Thawte Premium Server CA Thawte Premium Server CA 12/31/2020
thawte Primary Root CA thawte Primary Root CA 7/16/2036
thawte Primary Root CA - G3 thawte Primary Root CA - G3 12/1/2037
Thawte Server CA Thawte Server CA 12/31/2020
Thawte Timestamping CA Thawte Timestamping CA 12/31/2020
UTN-USERFirst-Object UTN-USERFirst-Object 7/9/2019
VeriSign Class 3 Public Primary ... VeriSign Class 3 Public Primary Ce... 7/16/2036
VeriSign Commercial Software ... VeriSign Commercial Software Pu... 1/7/2004
VeriSign Trust Network VeriSign Trust Network 8/1/2028
VeriSign Universal Root Certific... VeriSign Universal Root Certificati... 12/1/2037

。

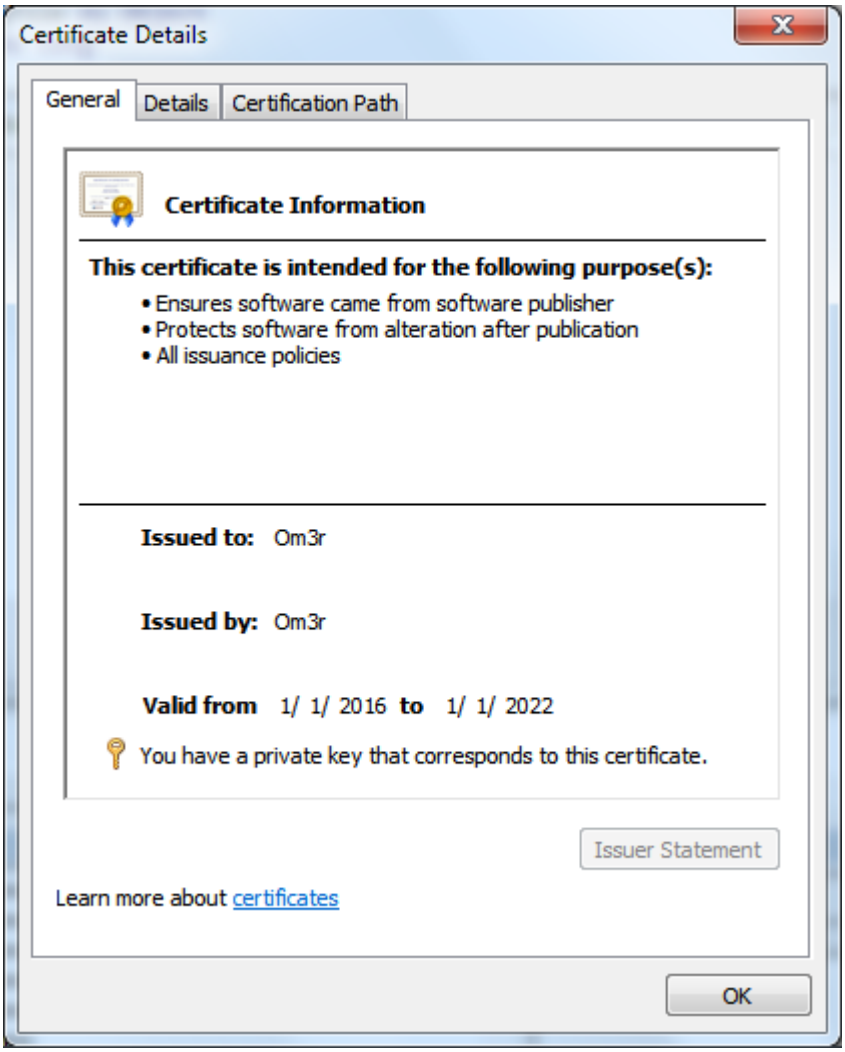

Console1 - [Console Root\Certificates - Current User\Trusted Root Certification Authorities\Certificates]
File Action View Favorites Window Help
Console Root
Certificates - Current User
Personal
Trusted Root Certification
Certificates
Enterprise
Intermedi
Active Dir
Trusted P
Untrusted
Third-Part
Trusted P
Other Peo
Certificate
Smart Car
All Tasks
View
New Window from Here
New Taskpad View...
Refresh
Export List...
Help
Import...
Issued To
Issued By
Expiration Da
Starfield Class 2 Certification A...
Starfield Class 2 Certification Auth...
6/29/2034
Starfield Root Certificate Autho...
Starfield Root Certificate Authorit...
12/31/2037
Starfield Services Root Certificat...
Starfield Services Root Certificate ...
12/31/2029
Certification Authority
9/17/2036
Premium Server CA
12/31/2020
t CA
thawte Primary Root CA
7/16/2036
t CA - G3
thawte Primary Root CA - G3
12/1/2037
Thawte Server CA
12/31/2020
ing CA
Thawte Timestamping CA
12/31/2020
ect
UTN-USERFirst-Object
7/9/2019
blic Primary ...
VeriSign Class 3 Public Primary Ce...
7/16/2036
al Software ...
VeriSign Commercial Software Pu...
1/7/2004
ork
VeriSign Trust Network
8/1/2028
VeriSign Universal Root Certific...
VeriSign Universal Root Certificati...
12/1/2037
Add a certificate to a store


Certificate Import Wizard
File to Import
Specify the file you want to import.
File name:
C:\Temp\Om3r.cer
Browse...
Note: More than one certificate can be stored in a single file in the following formats:
Personal Information Exchange- PKCS #12 (.PFX,.P12)
Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)
Microsoft Serialized Certificate Store (.SST)
Learn more about certificate file formats
< Back
Next >
Cancel

“”“ ”


Certificate Import Wizard
Certificate Store
Certificate stores are system areas where certificates are kept.
Windows can automatically select a certificate store, or you can specify a location for the certificate.
Automatically select the certificate store based on the type of certificate
Place all certificates in the following store
Certificate store:
Trusted Root Certification Authorities
Browse...
Learn more about certificate stores
< Back
Next >
Cancel

>。



VBA/ https://riptutorial.com/zh-CN/vba/topic/7733/vba-

23:

VBAVBA。

`CStr` `Format` `StrConv` 。

Examples

CStr

```
Const zipCode As Long = 10012
Dim zipCodeText As String
'Convert the zipCode number to a string of digit characters
zipCodeText = CStr(zipCode)
'zipCodeText = "10012"
```

Format

```
Const zipCode As long = 10012
Dim zeroPaddedNumber As String
zeroPaddedZipCode = Format(zipCode, "00000000")
'zeroPaddedNumber = "00010012"
```

StrConv

```
'Declare an array of bytes, assign single-byte character codes, and convert to a string
Dim singleByteChars(4) As Byte
singleByteChars(0) = 72
singleByteChars(1) = 101
singleByteChars(2) = 108
singleByteChars(3) = 108
singleByteChars(4) = 111
Dim stringFromSingleByteChars As String
stringFromSingleByteChars = StrConv(singleByteChars, vbUnicode)
'stringFromSingleByteChars = "Hello"
```

```
'Declare an array of bytes, assign multi-byte character codes, and convert to a string
Dim multiByteChars(9) As Byte
multiByteChars(0) = 87
multiByteChars(1) = 0
multiByteChars(2) = 111
multiByteChars(3) = 0
multiByteChars(4) = 114
multiByteChars(5) = 0
multiByteChars(6) = 108
multiByteChars(7) = 0
multiByteChars(8) = 100
multiByteChars(9) = 0

Dim stringFromMultiByteChars As String
stringFromMultiByteChars = multiByteChars
```

```
'stringFromMultiByteChars = "World"
```

https://riptutorial.com/zh-CN/vba/topic/3467/

24:

- VB_Name =“ClassOrModuleName”
- VB_GlobalNameSpace = False'
- VB_Creatable = False'
- VB_PredeclaredId = {True |}
- VB_Exposed = {True |}
- variableName.VB_VarUserMemId = 0'Zon。
- variableName.VB_VarDescription =“some string”'。
- procName.VB_Description =“some string”'。
- procName.VB_UserMemId = {0 | -4}
    - '0。
    - '-4Enumerator。

Examples

VB_Name

VB_Name。

```
Attribute VB_Name = "Class1"
```

```
Dim myClass As Class1
myClass = new Class1
```

VB_GlobalNameSpace

VBA。VB6。

VB6“”。 `DateTime` `DateTime.Now` `VBA.Conversion`。

```
Debug.Print VBA.Conversion.DateTime.Now
Debug.Print DateTime.Now
```

VB_Createable

。VB6。

VB6`VB_Exposed`。

```
VB_Exposed=True
VB_Creatable=True
```

`Public Class` VBA。

VB_PredeclaredId

◦ ◦

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Class1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Public Function GiveMeATwo() As Integer
    GiveMeATwo = 2
End Function
```

```
Debug.Print Class1.GiveMeATwo
```

◦

```
Dim cls As Class1
Set cls = New Class1
Debug.Print cls.GiveMeATwo
```

VB_Exposed

◦

```
Attribute VB_Exposed = False
```

Private ◦ ◦

```
Attribute VB_Exposed = True
```

Public LY◦ VBAVB_Createable ◦ VB.Net◦

```
Public Class Foo
    Friend Sub New()
    End Sub
End Class
```

◦ Public◦

```
Public Function CreateFoo() As Foo
    CreateFoo = New Foo
End Function
```

Public - Not Createable。

VB_Description

。 / API。

```
Public Function GiveMeATwo() As Integer
    Attribute GiveMeATwo.VB_Description = "Returns a two!"
    GiveMeATwo = 2
End Property
```

Get Let Set 。

VB_ [] UserMemId

VB_VarUserMemId VB_UserMemId VBA。

Collection ListItem

```
For i = 1 To myList.Count 'VBA Collection Objects are 1-based
    Debug.Print myList.Item(i)
Next
```

ItemVB_UserMemId0

```
For i = 1 To myList.Count 'VBA Collection Objects are 1-based
    Debug.Print myList(i)
Next
```

VB_UserMemId = 0 。 Get

```
Option Explicit
Private internal As New Collection

Public Property Get Count() As Long
    Count = internal.Count
End Property

Public Property Get Item(ByVal index As Long) As Variant
Attribute Item.VB_Description = "Gets or sets the element at the specified index."
Attribute Item.VB_UserMemId = 0
'Gets the element at the specified index.
    Item = internal(index)
End Property

Public Property Let Item(ByVal index As Long, ByVal value As Variant)
```

```
'Sets the element at the specified index.
    With internal
        If index = .Count + 1 Then
            .Add item:=value
        ElseIf index = .Count Then
            .Remove index
            .Add item:=value
        ElseIf index < .Count Then
            .Remove index
            .Add item:=value, before:=index
        End If
    End With
End Property
```

For Each

`-4 VB_UserMemId`VBA -

```
Dim item As Variant
For Each item In myList
    Debug.Print item
Next
```

/`Collection[_NewEnum]`getter;VBA

```
Public Property Get NewEnum() As IUnknown
Attribute NewEnum.VB_Description = "Gets an enumerator that iterates through the List."
Attribute NewEnum.VB_UserMemId = -4
Attribute NewEnum.VB_MemberFlags = "40" 'would hide the member in VB6. not supported in VBA.
'Gets an enumerator that iterates through the List.
    Set NewEnum = internal.[_NewEnum]
End Property
```

https://riptutorial.com/zh-CN/vba/topic/5321/

25:

.NETVisual Basic for Applications。

12。

Examples

-

VBA

```
Public Sub QuickSort(vArray As Variant, inLow As Long, inHi As Long)

  Dim pivot   As Variant
  Dim tmpSwap As Variant
  Dim tmpLow  As Long
  Dim tmpHi   As Long

  tmpLow = inLow
  tmpHi = inHi

  pivot = vArray((inLow + inHi) \ 2)

  While (tmpLow <= tmpHi)

     While (vArray(tmpLow) < pivot And tmpLow < inHi)
        tmpLow = tmpLow + 1
     Wend

     While (pivot < vArray(tmpHi) And tmpHi > inLow)
        tmpHi = tmpHi - 1
     Wend

     If (tmpLow <= tmpHi) Then
        tmpSwap = vArray(tmpLow)
        vArray(tmpLow) = vArray(tmpHi)
        vArray(tmpHi) = tmpSwap
        tmpLow = tmpLow + 1
        tmpHi = tmpHi - 1
     End If

  Wend

  If (inLow < tmpHi) Then QuickSort vArray, inLow, tmpHi
  If (tmpLow < inHi) Then QuickSort vArray, tmpLow, inHi

End Sub
```

Excel

Microsoft Excel`Sort`。

-

-

```
Sub testExcelSort()

Dim arr As Variant

InitArray arr
ExcelSort arr

End Sub

Private Sub InitArray(arr As Variant)

Const size = 10
ReDim arr(size)

Dim i As Integer

' Add descending numbers to the array to start
For i = 0 To size
    arr(i) = size - i
Next i

End Sub

Private Sub ExcelSort(arr As Variant)

' Ininitialize the Excel objects (required)
Dim xl As New Excel.Application
Dim wbk As Workbook
Set wbk = xl.Workbooks.Add
Dim sht As Worksheet
Set sht = wbk.ActiveSheet

' Copy the array to the Range object
Dim rng As Range
Set rng = sht.Range("A1")
Set rng = rng.Resize(UBound(arr, 1), 1)
rng.Value = xl.WorksheetFunction.Transpose(arr)

' Run the worksheet's sort routine on the Range
Dim MySort As Sort
Set MySort = sht.Sort

With MySort
    .SortFields.Clear
    .SortFields.Add rng, xlSortOnValues, xlAscending, xlSortNormal
    .SetRange rng
    .Header = xlNo
    .Apply
End With

' Copy the results back to the array
CopyRangeToArray rng, arr

' Clear the objects
Set rng = Nothing
wbk.Close False
xl.Quit

End Sub
```

```
Private Sub CopyRangeToArray(rng As Range, arr)

Dim i As Long
Dim c As Range

' Can't just set the array to Range.value (adds a dimension)
For Each c In rng.Cells
    arr(i) = c.Value
    i = i + 1
Next c

End Sub
```

https://riptutorial.com/zh-CN/vba/topic/8836/

26:

。 。 “”。

VBA。 。

Examples

-

Flyable

```
Public Sub Fly()
    ' No code.
End Sub

Public Function GetAltitude() As Long
    ' No code.
End Function
```

AirplaneImplements Flyable_Fly()LongFlyable_GetAltitude()。

```
Implements Flyable

Public Sub Flyable_Fly()
    Debug.Print "Flying With Jet Engines!"
End Sub

Public Function Flyable_GetAltitude() As Long
    Flyable_GetAltitude = 10000
End Function
```

DuckFlyable

```
Implements Flyable

Public Sub Flyable_Fly()
    Debug.Print "Flying With Wings!"
End Sub

Public Function Flyable_GetAltitude() As Long
    Flyable_GetAltitude = 30
End Function
```

FlyableFlyGetAltitude

```
Public Sub FlyAndCheckAltitude(F As Flyable)
    F.Fly
    Debug.Print F.GetAltitude
End Sub
```

IntelliSenseF FlyGetAltitude ◦

```
Dim MyDuck As New Duck
Dim MyAirplane As New Airplane

FlyAndCheckAltitude MyDuck
FlyAndCheckAltitude MyAirplane
```

```
Flying With Wings!
30
Flying With Jet Engines!
10000
```

AirplaneDuckFlyable_Fly FlyableFly ◦ Duck Flyable_Fly ◦

-

FlyableSwimmable

```
Sub Swim()
    ' No code
End Sub
```

DuckImplement

```
Implements Flyable
Implements Swimmable

Public Sub Flyable_Fly()
    Debug.Print "Flying With Wings!"
End Sub

Public Function Flyable_GetAltitude() As Long
    Flyable_GetAltitude = 30
End Function

Public Sub Swimmable_Swim()
    Debug.Print "Floating on the water"
End Sub
```

FishSwimmable

```
Implements Swimmable

Public Sub Swimmable_Swim()
    Debug.Print "Swimming under the water"
End Sub
```

DuckFlyableSubSwimmable

```
Sub InterfaceTest()

Dim MyDuck As New Duck
```

```
Dim MyAirplane As New Airplane
Dim MyFish As New Fish

Debug.Print "Fly Check..."

FlyAndCheckAltitude MyDuck
FlyAndCheckAltitude MyAirplane

Debug.Print "Swim Check..."

TrySwimming MyDuck
TrySwimming MyFish

End Sub

Public Sub FlyAndCheckAltitude(F As Flyable)
    F.Fly
    Debug.Print F.GetAltitude
End Sub

Public Sub TrySwimming(S As Swimmable)
    S.Swim
End Sub
```

...

10000

.....

https://riptutorial.com/zh-CN/vba/topic/8784/

27:

Examples

```
Dim Value As Byte
```

8。02556 Overflow。VBA。

CByte()。.5。

。

```
Sub ByteToStringAndBack()

Dim str As String
str = "Hello, World!"

Dim byt() As Byte
byt = str

Debug.Print byt(0)  ' 72

Dim str2 As String
str2 = byt

Debug.Print str2    ' Hello, World!

End Sub
```

Unicode。

```
Sub UnicodeExample()

Dim str As String
str = ChrW(&H2123) & "."  ' Versicle character and a dot

Dim byt() As Byte
byt = str

Debug.Print byt(0), byt(1), byt(2), byt(3)  ' Prints: 35,33,46,0

End Sub
```

```
Dim Value As Integer
```

16。-32,76832,7676。

little-endian 。

LongIntegerTypeAPI2。VBAIntegers32。IntegerLong。

Integer`CInt()` 。 .5。

```
Dim Value As Boolean
```

TrueFalse。 160FalseTrue。

1.-1。

```
Dim Example As Boolean
Example = True
Debug.Print CInt(Example)  'Prints -1
Debug.Print CBool(42)      'Prints True
Debug.Print CByte(True)    'Prints 255
```

`CBool()` 。 16161

```
Dim Example As Boolean
Example = CBool(2 ^ 17)
Debug.Print CInt(Example)   'Prints -1
Debug.Print CByte(Example)  'Prints 255
```

```
Dim Value As Long
```

Long32。 -2,147,483,6482,147,483,6476。

Longslittle-endian 。

Long32LongsAPI。

Long`CLng()` 。 .5。

```
Dim Value As Single
```

Single32。 little-endian IEEE 754。 - 。 -16,777,21616,777,216。 。

2 128。 。

。 。

Single`CSng()` 。

```
Dim Value As Double
```

Double64。 Singlelittle-endian IEEE 754。 Double-9,007,199,254,740,9929,007,199,254,740,992 。 。

2 1024Double。 。

DoubleCDbl()。

```
Dim Value As Currency
```

64Double 10,0004。 Currency-922,337,203,685,477.5808922,337,203,685,477.580732。。

CCur()。

```
Dim Value As Date
```

Date6418991230。。 Date12:00:00 AMx.512:00:00 PM。

11 1001231 9999。DoubleDate。

Double for Date

```
Dim MyDate As Double
MyDate = 0                                      'Epoch date.
Debug.Print Format$(MyDate, "yyyy-mm-dd")    'Prints 1899-12-30.
MyDate = MyDate + 365
Debug.Print Format$(MyDate, "yyyy-mm-dd")    'Prints 1900-12-30.
```

DateCDate() /。。

String

```
Dim Value As String
```

StringCOM BSTR。422。VBA2,147,483,647。

StrPtr()。VBA StringAPI。

VBA String。

```
Dim Value As String * 1024    'Declares a fixed length string of 1024 characters.
```

2。String。API。

1665,535。 -

```
Dim Foobar As String * 5
Foobar = "Foo" & "bar"
Debug.Print Foobar          'Prints "Fooba"
```

StringCStr()。

```
Dim Value As LongLong
```

LongLong6464。6432。-9,223,372,036,854,775,8089,223,372,036,854,775,8076。

LongLongslittle-endian 。

LongLongVBA 64。6464API。

LongLong`CLngLng()` 。.5。

```
Dim Value As Variant    'Explicit
Dim Value               'Implicit
```

VariantCOMVBAVariant。`As [Type]`Variant。

VARIANT VARTYPE 68。DateBooleanVariant。 。

VARTYPE		Reserved						Data area			
0	1	2	3	4	5	6	7	8	9	10	11

`VarType()TypeName()`Variant

```
Dim Example As Variant
Example = 42
Debug.Print VarType(Example)     'Prints 2 (VT_I2)
Debug.Print TypeName(Example)    'Prints "Integer"
Example = "Some text"
Debug.Print VarType(Example)     'Prints 8 (VT_BSTR)
Debug.Print TypeName(Example)    'Prints "String"
```

VariantsVariant。 。

VariantFor EachAPI

- 。IntegerVariantLong。
- 。
- Variant 8。

Variant`CVar()` 。

LongPtr

```
Dim Value As LongPtr
```

LongPtrVBA64。32Long 64LongLong 。

。

APIVBA。 LongPtrs><。 “”VBA。

LongPtrCLngPtr() 。 .5。

```
Dim Value As Variant
Value = CDec(1.234)

'Set Value to the smallest possible Decimal value
Value = CDec("0.0000000000000000000000000001")
```

DecimalVariantDecimalVariant CDecDecimal。 DecimalVBADecimal。

Decimal14Variant28。 -79,228,162,514,264,337,593,543,950,335 +79,228,162,514,264,337,593,543,950,335。 28-7.9228162514264337593543950335 +7.9228162514264337593543950335。

https://riptutorial.com/zh-CN/vba/topic/3418/

28:

[TODOCS 101VBA。VBA。

Examples

Set。

SinglyLinkedNode

```
Option Explicit

Private Value As Variant
Private NextNode As SinglyLinkedNode '"Next" is a keyword in VBA and therefore is not a valid
variable name
```

LinkedList

```
Option Explicit

Private head As SinglyLinkedNode

'Set type operations

Public Sub Add(value As Variant)
    Dim node As SinglyLinkedNode

    Set node = New SinglyLinkedNode
    node.value = value
    Set node.nextNode = head

    Set head = node
End Sub

Public Sub Remove(value As Variant)
    Dim node As SinglyLinkedNode
    Dim prev As SinglyLinkedNode

    Set node = head

    While Not node Is Nothing
        If node.value = value Then
            'remove node
            If node Is head Then
                Set head = node.nextNode
            Else
                Set prev.nextNode = node.nextNode
            End If
            Exit Sub
        End If
        Set prev = node
        Set node = node.nextNode
    Wend

End Sub
```

```
Public Function Exists(value As Variant) As Boolean
    Dim node As SinglyLinkedNode

    Set node = head
    While Not node Is Nothing
        If node.value = value Then
            Exists = True
            Exit Function
        End If
        Set node = node.nextNode
    Wend
End Function

Public Function Count() As Long
    Dim node As SinglyLinkedNode

    Set node = head

    While Not node Is Nothing
        Count = Count + 1
        Set node = node.nextNode
    Wend

End Function
```

。。7,5,9,3,11,6,12,1415BinaryTree。。 - AVL。

```
             7
            / \
           5   9
          / \   \
         3   6   11
                   \
                    12
                      \
                       14
                        \
                         15
```

BinaryTreeNode

```
Option Explicit

Public left As BinaryTreeNode
Public right As BinaryTreeNode
Public key As Variant
Public value As Variant
```

BinaryTree

[]

https://riptutorial.com/zh-CN/vba/topic/8628/

29:

Examples

VBA

Array

```
Dim myArray(9) As String 'Declaring an array that will contain up to 10 strings
```

VBA**ZERO**

ArrayArray

```
myArray(0) = "first element"
myArray(5) = "sixth element"
myArray(9) = "last element"
```

Arrays

```
Option Base 1
```

ONE 。

To= indexArray

```
Dim mySecondArray(1 To 12) As String 'Array of 12 strings indexed from 1 to 12
Dim myThirdArray(13 To 24) As String 'Array of 12 strings indexed from 13 to 24
```

ReDim

```
Dim myDynamicArray() As Strings 'Creates an Array of an unknown number of strings
ReDim myDynamicArray(5) 'This resets the array to 6 elements
```

ReDim◦ ReDimPreserve

```
Dim myDynamicArray(5) As String
myDynamicArray(0) = "Something I want to keep"

ReDim Preserve myDynamicArray(8) 'Expand the size to up to 9 strings
Debug.Print myDynamicArray(0) ' still prints the element
```

Split

◦

[[[ ]]]

	。 。 *expression*“”vbNullString**Split**。 LBound0UBound-1。
	。 。 “”。 ***delimiter***。
	。 ; -1。
	。 。 。

compare

	-1	**Option Compare** 。
vbBinaryCompare	0	。
vbTextCompare	1	。
vbDatabaseCompare	2	Microsoft Access。 。

Split。 Split。 。

```
Sub Test

    Dim textArray() as String

    textArray = Split("Tech on the Net")
    'Result: {"Tech", "on", "the", "Net"}

    textArray = Split("172.23.56.4", ".")
    'Result: {"172", "23", "56", "4"}

    textArray = Split("A;B;C;D", ";")
    'Result: {"A", "B", "C", "D"}

    textArray = Split("A;B;C;D", ";", 1)
    'Result: {"A;B;C;D"}

    textArray = Split("A;B;C;D", ";", 2)
    'Result: {"A", "B;C;D"}

    textArray = Split("A;B;C;D", ";", 3)
    'Result: {"A", "B", "C;D"}

    textArray = Split("A;B;C;D", ";", 4)
    'Result: {"A", "B", "C", "D"}

    'You can iterate over the created array
    Dim counter As Long

    For counter = LBound(textArray) To UBound(textArray)
        Debug.Print textArray(counter)
```

```
    Next
 End Sub
```

...

```
Dim items As Variant
items = Array(0, 1, 2, 3)

Dim index As Integer
For index = LBound(items) To UBound(items)
    'assumes value can be implicitly converted to a String:
    Debug.Print items(index)
Next
```

```
Dim items(0 To 1, 0 To 1) As Integer
items(0, 0) = 0
items(0, 1) = 1
items(1, 0) = 2
items(1, 1) = 3

Dim outer As Integer
Dim inner As Integer
For outer = LBound(items, 1) To UBound(items, 1)
    For inner = LBound(items, 2) To UBound(items, 2)
        'assumes value can be implicitly converted to a String:
        Debug.Print items(outer, inner)
    Next
Next
```

......

For Each...Next

```
Dim items As Variant
items = Array(0, 1, 2, 3)

Dim item As Variant 'must be variant
For Each item In items
    'assumes value can be implicitly converted to a String:
    Debug.Print item
Next
```

For Each

```
Dim items(0 To 1, 0 To 1) As Integer
items(0, 0) = 0
items(1, 0) = 1
items(0, 1) = 2
items(1, 1) = 3

Dim item As Variant 'must be Variant
For Each item In items
    'assumes value can be implicitly converted to a String:
    Debug.Print item
```

```
Next
```

For EachCollection。

4

```
 0
 1
 2
 3
```

。

ReDimPreserve。

。

```
Dim Dynamic_array As Variant
' first we set Dynamic_array as variant

For n = 1 To 100

    If IsEmpty(Dynamic_array) Then
        'isempty() will check if we need to add the first value to the array or subsequent
ones

        ReDim Dynamic_array(0)
        'ReDim Dynamic_array(0) will resize the array to one variable only
        Dynamic_array(0) = n

    Else
        ReDim Preserve Dynamic_array(0 To UBound(Dynamic_array) + 1)
        'in the line above we resize the array from variable 0 to the UBound() = last
variable, plus one effectivelly increeasing the size of the array by one
        Dynamic_array(UBound(Dynamic_array)) = n
        'attribute a value to the last variable of Dynamic_array
    End If

Next
```

。 “last”。

```
Dim Dynamic_array As Variant
Dynamic_array = Array("first", "middle", "last")

ReDim Preserve Dynamic_array(0 To UBound(Dynamic_array) - 1)
' Resize Preserve while dropping the last value
```

。 。 ReDim(0) 。

14040100。

```
Dim Dynamic_array As Variant

For n = 1 To 100

    If IsEmpty(Dynamic_array) Then
        ReDim Dynamic_array(0)
        Dynamic_array(0) = n

    ElseIf Dynamic_array(0) = "" Then
        'if first variant is empty ( = "") then give it the value of n
        Dynamic_array(0) = n
    Else
        ReDim Preserve Dynamic_array(0 To UBound(Dynamic_array) + 1)
        Dynamic_array(UBound(Dynamic_array)) = n
    End If
    If n = 40 Then
        ReDim Dynamic_array(0)
        'Resizing the array back to one variable without Preserving,
        'leaving the first value of the array empty
    End If

Next
```

Jagged ArraysJagged。

Jagged Arrays。

NamesNumbers。

```
Dim OuterArray() As Variant
Dim Names() As Variant
Dim Numbers() As Variant
'arrays are declared variant so we can access attribute any data type to its elements

Names = Array("Person1", "Person2", "Person3")
Numbers = Array("001", "002", "003")

OuterArray = Array(Names, Numbers)
'Directly giving OuterArray an array containing both Names and Numbers arrays inside

Debug.Print OuterArray(0)(1)
Debug.Print OuterArray(1)(1)
'accessing elements inside the jagged by giving the coordenades of the element
```

excel。

```
   Name -   Phone   -  Email  - Customer Number
Person1 - 153486231 - 1@STACK - 001
Person2 - 153486242 - 2@STACK - 002
Person3 - 153486253 - 3@STACK - 003
Person4 - 153486264 - 4@STACK - 004
Person5 - 153486275 - 5@STACK - 005
```

HeaderCustomersHeaderCustomerscustomer / row。

```
Dim Headers As Variant
' headers array with the top section of the customer data sheet
    For c = 1 To 4
        If IsEmpty(Headers) Then
            ReDim Headers(0)
            Headers(0) = Cells(1, c).Value
        Else
            ReDim Preserve Headers(0 To UBound(Headers) + 1)
            Headers(UBound(Headers)) = Cells(1, c).Value
        End If
    Next

Dim Customers As Variant
'Customers array will contain arrays of customer values
Dim Customer_Values As Variant
'Customer_Values will be an array of the customer in its elements (Name-Phone-Email-CustNum)

    For r = 2 To 6
    'iterate through the customers/rows
        For c = 1 To 4
        'iterate through the values/columns

            'build array containing customer values
            If IsEmpty(Customer_Values) Then
                ReDim Customer_Values(0)
                Customer_Values(0) = Cells(r, c).Value
            ElseIf Customer_Values(0) = "" Then
                Customer_Values(0) = Cells(r, c).Value
            Else
                ReDim Preserve Customer_Values(0 To UBound(Customer_Values) + 1)
                Customer_Values(UBound(Customer_Values)) = Cells(r, c).Value
            End If
        Next

        'add customer_values array to Customers Array
        If IsEmpty(Customers) Then
            ReDim Customers(0)
            Customers(0) = Customer_Values
        Else
            ReDim Preserve Customers(0 To UBound(Customers) + 1)
            Customers(UBound(Customers)) = Customer_Values
        End If

        'reset Custumer_Values to rebuild a new array if needed
        ReDim Customer_Values(0)
    Next

    Dim Main_Array(0 To 1) As Variant
    'main array will contain both the Headers and Customers

    Main_Array(0) = Headers
    Main_Array(1) = Customers
```

To better understand the way to Dynamically construct a one dimensional array please check Dynamic Arrays (Array Resizing and Dynamic Handling) on the Arrays documentation.

Jagged42Jagged543

```
Main_Array(0) - Headers - Array("Name","Phone","Email","Customer Number")
          (1) - Customers(0) - Array("Person1",153486231,"1@STACK",001)
```

```
                Customers(1) - Array("Person2",153486242,"2@STACK",002)
                ...
                Customers(4) - Array("Person5",153486275,"5@STACK",005)
```

Jagged`Main ArrayHeaders Customers`。

`Main ArrayInfo Type: Info`。

```
For n = 0 To UBound(Main_Array(1))
    'n to iterate from fisrt to last array in Main_Array(1)

    For j = 0 To UBound(Main_Array(1)(n))
        'j will iterate from first to last element in each array of Main_Array(1)

        Debug.Print Main_Array(0)(j) & ": " & Main_Array(1)(n)(j)
        'print Main_Array(0)(j) which is the header and Main_Array(0)(n)(j) which is the
element in the customer array
        'we can call the header with j as the header array has the same structure as the
customer array
    Next
Next
```

Jagged`Main_Array -> Customers -> CustomerNumber -> Name"Person4"`Main_ArrayCustomers Jagged4 `Main_Array(1)(3)(0)Main_Array(Customers)(CustomerNumber)(Name)`。

32。

。

One Dimension。

```
Dim 1D(3) as Variant

*1D - Visually*
(0)
(1)
(2)
```

Two DimensionsSudoku GridExcel。

```
Dim 2D(3,3) as Variant
'this would result in a 3x3 grid

*2D - Visually*
(0,0) (0,1) (0,2)
(1,0) (1,1) (1,2)
(2,0) (2,1) (2,2)
```

RubikCube/。

```
Dim 3D(3,3,2) as Variant
```

```
'this would result in a 3x3x3 grid

*3D - Visually*
       1st layer                 2nd layer                   3rd layer
         front                     middle                      back
(0,0,0) (0,0,1) (0,0,2) ¦ (1,0,0) (1,0,1) (1,0,2) ¦ (2,0,0) (2,0,1) (2,0,2)
(0,1,0) (0,1,1) (0,1,2) ¦ (1,1,0) (1,1,1) (1,1,2) ¦ (2,1,0) (2,1,1) (2,1,2)
(0,2,0) (0,2,1) (0,2,2) ¦ (1,2,0) (1,2,1) (1,2,2) ¦ (2,2,0) (2,2,1) (2,2,2)
```

3D4D1,3,3,33D。

.....0,0。

```
Dim Bosses As Variant
'set bosses as Variant, so we can input any data type we want

Bosses = [{"Jonh","Snow","President";"Ygritte","Wild","Vice-President"}]
'initialise a 2D array directly by filling it with information, the redult wil be a array(1,2)
size 2x3 = 6 elements

Dim Employees As Variant
'initialize your Employees array as variant
'initialize and ReDim the Employee array so it is a dynamic array instead of a static one,
hence treated differently by the VBA Compiler
ReDim Employees(100, 5)
'declaring an 2D array that can store 100 employees with 6 elements of information each, but
starts empty
'the array size is 101 x 6 and contains 606 elements

For employee = 0 To UBound(Employees, 1)
'for each employee/row in the array, UBound for 2D arrays, which will get the last element on
the array
'needs two parameters 1st the array you which to check and 2nd the dimension, in this case 1 =
employee and 2 = information
    For information_e = 0 To UBound(Employees, 2)
    'for each information element/column in the array

        Employees(employee, information_e) = InformationNeeded ' InformationNeeded would be
the data to fill the array
        'iterating the full array will allow for direct attribution of information into the
element coordinates
    Next
Next
```

`ReDim Preserve`One-Dimension/。Temptemp。

```
Dim TempEmp As Variant
'initialise your temp array as variant
ReDim TempEmp(UBound(Employees, 1) + 1, UBound(Employees, 2))
'ReDim/Resize Temp array as a 2D array with size UBound(Employees)+1 = (last element in
Employees 1st dimension) + 1,
```

```
'the 2nd dimension remains the same as the original array. we effectively add 1 row in the
Employee array

'transfer
For emp = LBound(Employees, 1) To UBound(Employees, 1)
    For info = LBound(Employees, 2) To UBound(Employees, 2)
        'to transfer Employees into TempEmp we iterate both arrays and fill TempEmp with the
corresponding element value in Employees
        TempEmp(emp, info) = Employees(emp, info)

    Next
Next

'fill remaining
'after the transfers the Temp array still has unused elements at the end, being that it was
increased
'to fill the remaining elements iterate from the last "row" with values to the last row in the
array
'in this case the last row in Temp will be the size of the Employees array rows + 1, as the
last row of Employees array is already filled in the TempArray

For emp = UBound(Employees, 1) + 1 To UBound(TempEmp, 1)
    For info = LBound(TempEmp, 2) To UBound(TempEmp, 2)

        TempEmp(emp, info) = InformationNeeded & "NewRow"

    Next
Next

'erase Employees, attribute Temp array to Employees and erase Temp array
Erase Employees
Employees = TempEmp
Erase TempEmp
```

/ `Employees(0, 0) = "NewValue"`

```
For emp = 0 To UBound(Employees)
    If Employees(emp, 0) = "Gloria" And Employees(emp, 1) = "Stephan" Then
    'if value found
        Employees(emp, 1) = "Married, Last Name Change"
        Exit For
        'don't iterate through a full array unless necessary
    End If
Next
```

。

```
'nested loop, will iterate through all elements
For emp = LBound(Employees, 1) To UBound(Employees, 1)
    For info = LBound(Employees, 2) To UBound(Employees, 2)
        Debug.Print Employees(emp, info)
    Next
```

```
Next

'loop and coordinate, iteration through all rows and in each row accessing all columns
directly
For emp = LBound(Employees, 1) To UBound(Employees, 1)
    Debug.Print Employees(emp, 0)
    Debug.Print Employees(emp, 1)
    Debug.Print Employees(emp, 2)
    Debug.Print Employees(emp, 3)
    Debug.Print Employees(emp, 4)
    Debug.Print Employees(emp, 5)
Next

'directly accessing element with coordinates
Debug.Print Employees(5, 5)
```

。

3D2D。

3DEmployeesexcel/。 3D***n***2D。

3D3`Dim 3Darray(2,5,5) As Variant` /。 `Dim`108 `3*6*6` 3D32D。

```
Dim ThreeDArray As Variant
'initialise your ThreeDArray array as variant
ReDim ThreeDArray(1, 50, 5)
'declaring an 3D array that can store two sets of 51 employees with 6 elements of information
each, but starts empty
'the array size is 2 x 51 x 6 and contains 612 elements

For building = 0 To UBound(ThreeDArray, 1)
    'for each building/set in the array
    For employee = 0 To UBound(ThreeDArray, 2)
    'for each employee/row in the array
        For information_e = 0 To UBound(ThreeDArray, 3)
        'for each information element/column in the array

            ThreeDArray(building, employee, information_e) = InformationNeeded '
InformationNeeded would be the data to fill the array
        'iterating the full array will allow for direct attribution of information into the
element coordinates
        Next
    Next
Next
```

3D2D。

1。

```
Dim TempEmp As Variant
'initialise your temp array as variant
ReDim TempEmp(UBound(ThreeDArray, 1), UBound(ThreeDArray, 2) + 1, UBound(ThreeDArray, 3))
'ReDim/Resize Temp array as a 3D array with size UBound(ThreeDArray)+1 = (last element in
Employees 2nd dimension) + 1,
'the other dimension remains the same as the original array. we effectively add 1 row in the
for each set of the 3D array

'transfer
For building = LBound(ThreeDArray, 1) To UBound(ThreeDArray, 1)
    For emp = LBound(ThreeDArray, 2) To UBound(ThreeDArray, 2)
        For info = LBound(ThreeDArray, 3) To UBound(ThreeDArray, 3)
            'to transfer ThreeDArray into TempEmp by iterating all sets in the 3D array and
fill TempEmp with the corresponding element value in each set of each row
            TempEmp(building, emp, info) = ThreeDArray(building, emp, info)

        Next
    Next
Next

'fill remaining
'to fill the remaining elements we need to iterate from the last "row" with values to the last
row in the array in each set, remember that the first empty element is the original array
Ubound() plus 1
For building = LBound(TempEmp, 1) To UBound(TempEmp, 1)
    For emp = UBound(ThreeDArray, 2) + 1 To UBound(TempEmp, 2)
        For info = LBound(TempEmp, 3) To UBound(TempEmp, 3)

            TempEmp(building, emp, info) = InformationNeeded & "NewRow"

        Next
    Next
Next

'erase Employees, attribute Temp array to Employees and erase Temp array
Erase ThreeDArray
ThreeDArray = TempEmp
Erase TempEmp
```

3D2D。

```
Do
' using Do ... While for early exit
    For building = 0 To UBound(ThreeDArray, 1)
        For emp = 0 To UBound(ThreeDArray, 2)
            If ThreeDArray(building, emp, 0) = "Gloria" And ThreeDArray(building, emp, 1) =
"Stephan" Then
            'if value found
                ThreeDArray(building, emp, 1) = "Married, Last Name Change"
                Exit Do
                'don't iterate through all the array unless necessary
            End If
        Next
    Next
Loop While False

'nested loop, will iterate through all elements
```

```
For building = LBound(ThreeDArray, 1) To UBound(ThreeDArray, 1)
    For emp = LBound(ThreeDArray, 2) To UBound(ThreeDArray, 2)
        For info = LBound(ThreeDArray, 3) To UBound(ThreeDArray, 3)
            Debug.Print ThreeDArray(building, emp, info)
        Next
    Next
Next

'loop and coordinate, will iterate through all set of rows and ask for the row plus the value
we choose for the columns
For building = LBound(ThreeDArray, 1) To UBound(ThreeDArray, 1)
    For emp = LBound(ThreeDArray, 2) To UBound(ThreeDArray, 2)
        Debug.Print ThreeDArray(building, emp, 0)
        Debug.Print ThreeDArray(building, emp, 1)
        Debug.Print ThreeDArray(building, emp, 2)
        Debug.Print ThreeDArray(building, emp, 3)
        Debug.Print ThreeDArray(building, emp, 4)
        Debug.Print ThreeDArray(building, emp, 5)
    Next
Next

'directly accessing element with coordinates
Debug.Print Employees(0, 5, 5)
```

https://riptutorial.com/zh-CN/vba/topic/3064/

30:

Examples

VBA2 GregorianHijri

`Calendar`。

2

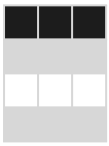
0	vbCalGreg	
1	vbCalHijri	Hijri

```
Sub CalendarExample()
    'Cache the current setting.
    Dim Cached As Integer
    Cached = Calendar

    ' Dates in Gregorian Calendar
    Calendar = vbCalGreg
    Dim Sample As Date
    'Create sample date of 2016-07-28
    Sample = DateSerial(2016, 7, 28)

    Debug.Print "Current Calendar : " & Calendar
    Debug.Print "SampleDate = " & Format$(Sample, "yyyy-mm-dd")

    ' Date in Hijri Calendar
    Calendar = vbCalHijri
    Debug.Print "Current Calendar : " & Calendar
    Debug.Print "SampleDate = " & Format$(Sample, "yyyy-mm-dd")

    'Reset VBA to cached value.
    Cached = Calendar
End Sub
```

Sub;

```
Current Calendar : 0
SampleDate = 2016-07-28
Current Calendar : 1
SampleDate = 1437-10-23
```

VBA3/。



```
Sub DateTimeExample()

    ' -----------------------------------------------------
    ' Note : EU system with default date format DD/MM/YYYY
    ' -----------------------------------------------------

    Debug.Print Now   ' prints 28/07/2016 10:16:01 (output below assumes this date and time)
    Debug.Print Date  ' prints 28/07/2016
    Debug.Print Time  ' prints 10:16:01

    ' Apply a custom format to the current date or time
    Debug.Print Format$(Now, "dd mmmm yyyy hh:nn")  ' prints 28 July 2016 10:16
    Debug.Print Format$(Date, "yyyy-mm-dd")         ' prints 2016-07-28
    Debug.Print Format$(Time, "hh") & " hour " & _
                Format$(Time, "nn") & " min " & _
                Format$(Time, "ss") & " sec "       ' prints 10 hour 16 min 01 sec

End Sub
```

`Timer`Single。 。

```
Sub TimerExample()

    Debug.Print Time    ' prints 10:36:31  (time at execution)
    Debug.Print Timer   ' prints 38191,13  (seconds since midnight)

End Sub
```

`NowTimeTimer`

```
Sub GetBenchmark()

    Dim StartTime  As Single
    StartTime = Timer       'Store the current Time

    Dim i As Long
    Dim temp As String
    For i = 1 To 1000000    'See how long it takes Left$ to execute 1,000,000 times
        temp = Left$("Text", 2)
    Next i

    Dim Elapsed As Single
    Elapsed = Timer - StartTime
    Debug.Print "Code completed in " & CInt(Elapsed * 1000) & " ms"

End Sub
```

IsDate

IsDate。 `Boolean` 。

```
Sub IsDateExamples()

    Dim anything As Variant
```

```
    anything = "September 11, 2001"

    Debug.Print IsDate(anything)    'Prints True

    anything = #9/11/2001#

    Debug.Print IsDate(anything)    'Prints True

    anything = "just a string"

    Debug.Print IsDate(anything)    'Prints False

    anything = vbNull

    Debug.Print IsDate(anything)    'Prints False

End Sub
```

Variant Date Integer ◦ Date 13◦

dateyear◦	1009999
datemonth◦	112
dateday◦	131
date◦	17
date◦	023
date◦	059
date◦	059

```
Sub ExtractionExamples()

    Dim MyDate As Date

    MyDate = DateSerial(2016, 7, 28) + TimeSerial(12, 34, 56)

    Debug.Print Format$(MyDate, "yyyy-mm-dd hh:nn:ss") ' prints 2016-07-28 12:34:56

    Debug.Print Year(MyDate)                           ' prints 2016
    Debug.Print Month(MyDate)                          ' prints 7
    Debug.Print Day(MyDate)                            ' prints 28
    Debug.Print Hour(MyDate)                           ' prints 12
    Debug.Print Minute(MyDate)                         ' prints 34
    Debug.Print Second(MyDate)                         ' prints 56

    Debug.Print Weekday(MyDate)                        ' prints 5
    'Varies by locale - i.e. will print 4 in the EU and 5 in the US
    Debug.Print Weekday(MyDate, vbUseSystemDayOfWeek)
    Debug.Print Weekday(MyDate, vbMonday)              ' prints 4
```

```
    Debug.Print Weekday(MyDate, vbSunday)              ' prints 5

End Sub
```

DatePart

`DatePart()`。 。

```
DatePart ( interval, date  [, firstdayofweek] [, firstweekofyear] )
```

interval

“YYYY”	1009999
“Y”	1366
“m”	112
“Q”	14
“WW”	153
“W”	17
“d”	131
“H”	023
“N”	059
“S”	059

firstdayofweek。 。 `vbSunday` 。

firstweekofyear。 。 11。

```
Sub DatePartExample()

    Dim MyDate As Date

    MyDate = DateSerial(2016, 7, 28) + TimeSerial(12, 34, 56)

    Debug.Print Format$(MyDate, "yyyy-mm-dd hh:nn:ss") ' prints 2016-07-28 12:34:56

    Debug.Print DatePart("yyyy", MyDate)              ' prints 2016
    Debug.Print DatePart("y", MyDate)                 ' prints 210
    Debug.Print DatePart("h", MyDate)                 ' prints 12
    Debug.Print DatePart("Q", MyDate)                 ' prints 3
    Debug.Print DatePart("w", MyDate)                 ' prints 5
```

```
    Debug.Print DatePart("ww", MyDate)                ' prints 31

End Sub
```

DateDiff

DateDiff()Long。

```
DateDiff ( interval, date1, date2  [, firstdayofweek] [, firstweekofyear] )
```

- *interval*DatePart()
- *date1date2*
- *Firstdayofweekfirstweekofyear*。 DatePart()

```
Sub DateDiffExamples()

    ' Check to see if 2016 is a leap year.
    Dim NumberOfDays As Long
    NumberOfDays = DateDiff("d", #1/1/2016#, #1/1/2017#)

    If NumberOfDays = 366 Then
        Debug.Print "2016 is a leap year."              'This will output.
    End If

    ' Number of seconds in a day
    Dim StartTime As Date
    Dim EndTime As Date
    StartTime = TimeSerial(0, 0, 0)
    EndTime = TimeSerial(24, 0, 0)
    Debug.Print DateDiff("s", StartTime, EndTime)       'prints 86400

End Sub
```

DateAdd

DateAdd()Date。

```
DateAdd ( interval, number, date  )
```

- *interval*DatePart()
- *number*。 。
- *date*Date

```
Sub DateAddExamples()

    Dim Sample As Date
    'Create sample date and time of 2016-07-28 12:34:56
    Sample = DateSerial(2016, 7, 28) + TimeSerial(12, 34, 56)

    ' Date 5 months previously (prints 2016-02-28):
```

```
    Debug.Print Format$(DateAdd("m", -5, Sample), "yyyy-mm-dd")

    ' Date 10 months previously (prints 2015-09-28):
    Debug.Print Format$(DateAdd("m", -10, Sample), "yyyy-mm-dd")

    ' Date in 8 months (prints 2017-03-28):
    Debug.Print Format$(DateAdd("m", 8, Sample), "yyyy-mm-dd")

    ' Date/Time 18 hours previously (prints 2016-07-27 18:34:56):
    Debug.Print Format$(DateAdd("h", -18, Sample), "yyyy-mm-dd hh:nn:ss")

    ' Date/Time in 36 hours (prints 2016-07-30 00:34:56):
    Debug.Print Format$(DateAdd("h", 36, Sample), "yyyy-mm-dd hh:nn:ss")

End Sub
```

CDATE

CDate()Date

```
Sub CDateExamples()

    Dim sample As Date

    ' Converts a String representing a date and time to a Date
    sample = CDate("September 11, 2001 12:34")
    Debug.Print Format$(sample, "yyyy-mm-dd hh:nn:ss")      ' prints 2001-09-11 12:34:00

    ' Converts a String containing a date to a Date
    sample = CDate("September 11, 2001")
    Debug.Print Format$(sample, "yyyy-mm-dd hh:nn:ss")      ' prints 2001-09-11 00:00:00

    ' Converts a String containing a time to a Date
    sample = CDate("12:34:56")
    Debug.Print Hour(sample)                        ' prints 12
    Debug.Print Minute(sample)                      ' prints 34
    Debug.Print Second(sample)                      ' prints 56

    ' Find the 10000th day from the epoch date of 1899-12-31
    sample = CDate(10000)
    Debug.Print Format$(sample, "yyyy-mm-dd")       ' prints 1927-05-18

End Sub
```

VBACVDate() CDate()VariantDate◦ DateDateCDate()VariantVariantCVDate()◦ ◦

DateSerial

DateSerial()◦ Date◦

```
DateSerial ( year, month, day )
```

1009999112131◦

```
Sub DateSerialExamples()

    ' Build a specific date
    Dim sample As Date
    sample = DateSerial(2001, 9, 11)
    Debug.Print Format$(sample, "yyyy-mm-dd")                    ' prints 2001-09-11

    ' Find the first day of the month for a date.
    sample = DateSerial(Year(sample), Month(sample), 1)
    Debug.Print Format$(sample, "yyyy-mm-dd")                    ' prints 2001-09-11

    ' Find the last day of the previous month.
    sample = DateSerial(Year(sample), Month(sample), 1) - 1
    Debug.Print Format$(sample, "yyyy-mm-dd")                    ' prints 2001-09-11

End Sub
```

DateSerial()“”。

```
Sub GoodDateSerialExample()

    'Calculate 45 days from today
    Dim today As Date
    today = DateSerial (2001, 9, 11)
    Dim futureDate As Date
    futureDate = DateSerial(Year(today), Month(today), Day(today) + 45)
    Debug.Print Format$(futureDate, "yyyy-mm-dd")             'prints 2009-10-26

End Sub
```

```
Sub BadDateSerialExample()

    'Allow user to enter unvalidate date information
    Dim myYear As Long
    myYear = InputBox("Enter Year")
            'Assume user enters 2009
    Dim myMonth As Long
    myMonth = InputBox("Enter Month")
            'Assume user enters 2
    Dim myDay As Long
    myDay = InputBox("Enter Day")
            'Assume user enters 31
    Debug.Print Format$(DateSerial(myYear, myMonth, myDay), "yyyy-mm-dd")
            'prints  2009-03-03

End Sub
```

https://riptutorial.com/zh-CN/vba/topic/4452/

31:

Examples

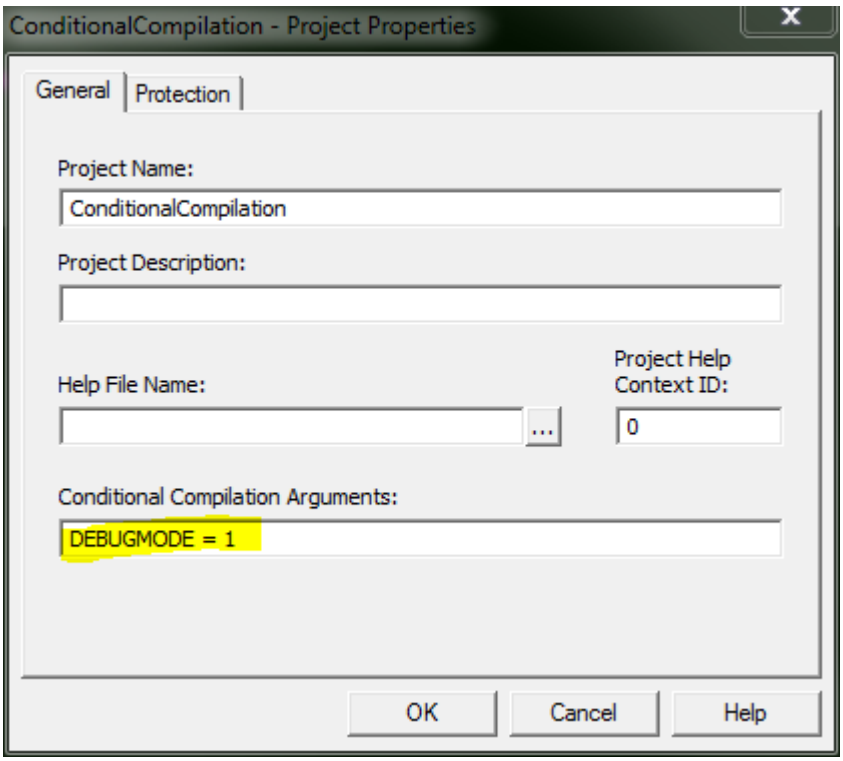
#Const◦ #If◦

```
#Const DEBUGMODE = 1

#If DEBUGMODE Then
    Const filepath As String = "C:\Users\UserName\Path\To\File.txt"
#Else
    Const filepath As String = "\\server\share\path\to\file.txt"
#End If
```

filepath"C:\Users\UserName\Path\To\File.txt"◦ #Const#Const DEBUGMODE = 0filepath "\\server\share\path\to\file.txt"◦

#Const

#Const◦ ◦ >> []#Const ◦ ◦ “”[constName] = [value]◦ 1[constName1] = [value1] : [constName2] = [value2]◦



◦ VBA◦ Vba7Office 201064Office◦

	16	32	64
VBA6		Vba6	

	16	32	64
VBA7		Vba7	
Win16			
Win32			
Win64			
		Mac	Mac

Win64 / Win32OfficeWindows。32OfficeWin32 = TRUE64Windows。

OfficeDeclare Imports

```
#If Vba7 Then
    ' It's important to check for Win64 first,
    ' because Win32 will also return true when Win64 does.

    #If Win64 Then
        Declare PtrSafe Function GetFoo64 Lib "exampleLib32" () As LongLong
    #Else
        Declare PtrSafe Function GetFoo Lib "exampleLib32" () As Long
    #End If
#Else
    ' Must be Vba6, the PtrSafe keyword didn't exist back then,
    ' so we need to declare Win32 imports a bit differently than above.

    #If Win32 Then
        Declare Function GetFoo Lib "exampleLib32"() As Long
    #Else
        Declare Function GetFoo Lib "exampleLib"() As Integer
    #End If
#End If
```

。16Office。 1619944.3 Office 2007。

```
#If Vba7 Then
    ' It's important to check for Win64 first,
    ' because Win32 will also return true when Win64 does.

    #If Win64 Then
        Declare PtrSafe Function GetFoo64 Lib "exampleLib32" () As LongLong
    #Else
        Declare PtrSafe Function GetFoo Lib "exampleLib32" () As Long
    #End If
#Else
    ' Must be Vba6. We don't support 16 bit office, so must be Win32.

    Declare Function GetFoo Lib "exampleLib32"() As Long
#End If
```

Office 2010。

```
' We only have 2010 installs, so we already know we have Vba7.

#If Win64 Then
    Declare PtrSafe Function GetFoo64 Lib "exampleLib32" () As LongLong
#Else
    Declare PtrSafe Function GetFoo Lib "exampleLib32" () As Long
#End If
```

https://riptutorial.com/zh-CN/vba/topic/3364/

32:

- `[Public] Event [identifier]([argument_list])`
- `Dim|Private|Public WithEvents [identifier] As [type]`
- `Public ◦ Public◦`
- `WithEventsPrivatePublic Friend ◦ WithEvents◦ Dim◦`

Examples

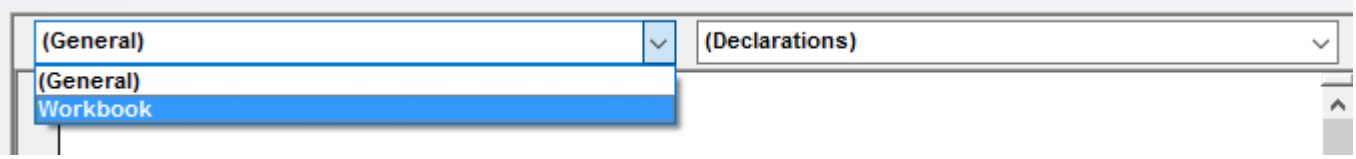
VBA VBA - VBA◦

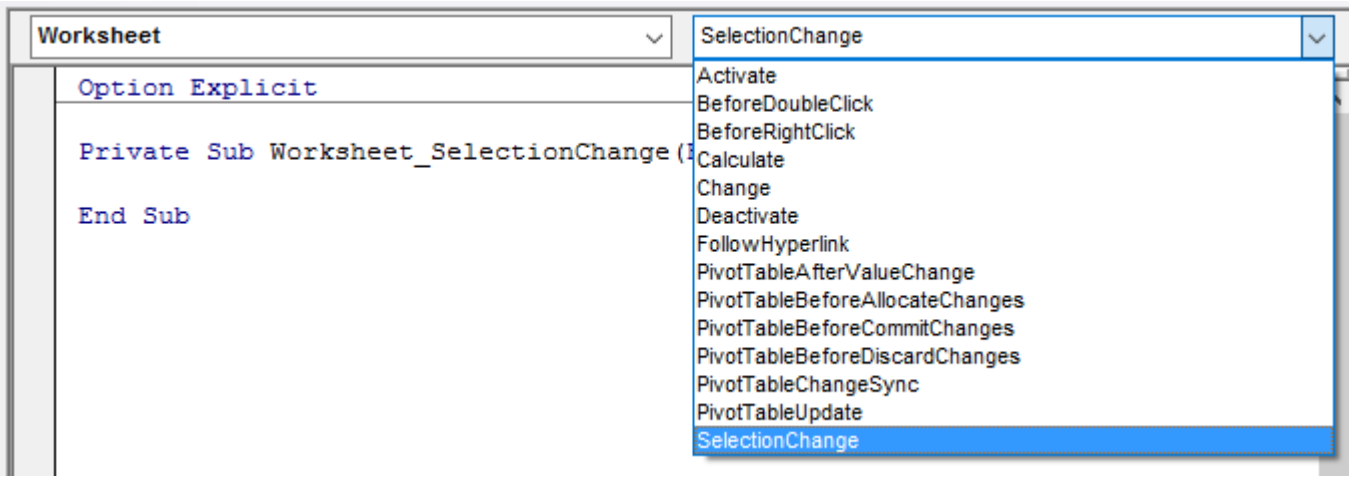
API◦ `Excel.Application◦ ◦ ◦ ◦ Click◦`

API - ◦

◦ ◦

VBA`ThisDocument ThisWorkbook Sheet1UserForm ◦`





VBE◦

`WithEvents`

```
Private WithEvents Foo As Workbook
```

```
Private WithEvents Bar As Worksheet
```

WithEvents◦ VBEWithEvents

```
Private WithEvents Foo As Workbook
Private WithEvents Bar As Worksheet

Private Sub Foo_Open()

End Sub

Private Sub Bar_SelectionChange(ByVal Target As Range)

End Sub
```

WithEventsWithEventsNew◦

```
Private WithEvents Foo As New Workbook 'illegal
```

Set ;Class_Initialize◦

◦ EventEvent

```
Public Event SomethingHappened(ByVal something As String)
```

◦

- ◦ RaiseEvent;

```
Public Sub DoSomething()
    RaiseEvent SomethingHappened("hello")
End Sub
```

SomethingHappenedDoSomething◦ Somethingtest.DoSomething ThisWorkbook“hello”

```
Private WithEvents test As Something

Private Sub Workbook_Open()
    Set test = New Something
    test.DoSomething
End Sub

Private Sub test_SomethingHappened(ByVal bar As String)
'this procedure runs whenever 'test' raises the 'SomethingHappened' event
    MsgBox bar
End Sub
```

ByRef

```
Public Event BeforeSomething(ByRef cancel As Boolean)
Public Event AfterSomething()

Public Sub DoSomething()
    Dim cancel As Boolean
    RaiseEvent BeforeSomething(cancel)
    If cancel Then Exit Sub

    'todo: actually do something

    RaiseEvent AfterSomething
End Sub
```

BeforeSomethingcancelTrue cancelTrueAfterSomething。

```
Private WithEvents foo As Something

Private Sub foo_BeforeSomething(ByRef cancel As Boolean)
    cancel = MsgBox("Cancel?", vbYesNo) = vbYes
End Sub

Private Sub foo_AfterSomething()
    MsgBox "Didn't cancel!"
End Sub
```

foofoo.DoSomethingNo“”。

ByVal;。

```
'class module ReturnBoolean
Option Explicit
Private encapsulated As Boolean

Public Property Get ReturnValue() As Boolean
'Attribute ReturnValue.VB_UserMemId = 0
    ReturnValue = encapsulated
End Property

Public Property Let ReturnValue(ByVal value As Boolean)
    encapsulated = value
End Property
```

Variant

```
Public Event SomeEvent(ByVal foo As Variant)

Public Sub DoSomething()
    Dim result As ReturnBoolean
    result = New ReturnBoolean

    RaiseEvent SomeEvent(result)

    If result Then ' If result.ReturnValue Then
        'handler changed the value to True
```

```
    Else
        'handler didn't modify the value
    End If
End Sub
```

```
Private Sub source_SomeEvent(ByVal foo As Variant) 'foo is actually a ReturnBoolean object
    foo = True 'True is actually assigned to foo.ReturnValue, the class' default member
End Sub
```

https://riptutorial.com/zh-CN/vba/topic/5278/

33:

Examples

Select Case ◦ ◦

```
Sub TestCase()
    Dim MyVar As String

    Select Case MyVar    'We Select the Variable MyVar to Work with
        Case "Hello"     'Now we simply check the cases we want to check
            MsgBox "This Case"
        Case "World"
            MsgBox "Important"
        Case "How"
            MsgBox "Stuff"
        Case "Are"
            MsgBox "I'm running out of ideas"
        Case "You?", "Today"  'You can separate several conditions with a comma
            MsgBox "Uuuhm..." 'if any is matched it will go into the case
        Case Else             'If none of the other cases is hit
            MsgBox "All of the other cases failed"
    End Select

    Dim i As Integer
    Select Case i
        Case Is > 2 '"Is" can be used instead of the variable in conditions.
            MsgBox "i is greater than 2"
        'Case 2 < Is '"Is" can only be used at the beginning of the condition.
        'Case Else is optional
    End Select
End Sub
```

Select Case

```
Dim x As Integer
Dim y As Integer

x = 2
y = 5

Select Case True
    Case x > 3
        MsgBox "x is greater than 3"
    Case y < 2
        MsgBox "y is less than 2"
    Case x = 1
        MsgBox "x is equal to 1"
    Case x = 2 Xor y = 3
        MsgBox "Go read about ""Xor"""
    Case Not y = 5
        MsgBox "y is not 5"
    Case x = 3 Or x = 10
        MsgBox "x = 3 or 10"
    Case y < 10 And x < 10
        MsgBox "x and y are less than 10"
```

```
    Case Else
        MsgBox "No match found"
End Select
```

Case◦ Select CaseSelect CaseIs

```
Dim x As Integer

x = 5

Select Case x
    Case 1
        MsgBox "x equals 1"
    Case 2, 3, 4
        MsgBox "x is 2, 3 or 4"
    Case 7 To 10
        MsgBox "x is between 7 and 10 (inclusive)"
    Case Is < 2
        MsgBox "x is less than one"
    Case Is >= 7
        MsgBox "x is greater than or equal to 7"
    Case Else
        MsgBox "no match found"
End Select
```

For Each◦

```
Public Sub IterateCollection(ByVal items As Collection)

    'For Each iterator must always be variant
    Dim element As Variant

    For Each element In items
        'assumes element can be converted to a string
        Debug.Print element
    Next

End Sub
```

For Each

```
Dim sheet As Worksheet
For Each sheet In ActiveWorkbook.Worksheets
    Debug.Print sheet.Name
Next
```

For Each；For◦ For EachCollection◦

```
For Each [item] In [collection]
    [statements]
Next [item]
```

Next;◦

```
Dim book As Workbook
For Each book In Application.Workbooks

    Debug.Print book.FullName

    Dim sheet As Worksheet
    For Each sheet In ActiveWorkbook.Worksheets
        Debug.Print sheet.Name
    Next sheet
Next book
```

```
 Public Sub DoLoop()
    Dim entry As String
    entry = ""
    'Equivalent to a While loop will ask for strings until "Stop" in given
    'Prefer using a While loop instead of this form of Do loop
    Do While entry <> "Stop"
        entry = InputBox("Enter a string, Stop to end")
        Debug.Print entry
    Loop

    'Equivalent to the above loop, but the condition is only checked AFTER the
    'first iteration of the loop, so it will execute even at least once even
    'if entry is equal to "Stop" before entering the loop (like in this case)
    Do
        entry = InputBox("Enter a string, Stop to end")
        Debug.Print entry
    Loop While entry <> "Stop"


    'Equivalent to writing Do While Not entry="Stop"
    '
    'Because the Until is at the top of the loop, it will
    'not execute because entry is still equal to "Stop"
    'when evaluating the condition
    Do Until entry = "Stop"
        entry = InputBox("Enter a string, Stop to end")
        Debug.Print entry
    Loop

    'Equivalent to writing Do ... Loop While Not i >= 100
    Do
        entry = InputBox("Enter a string, Stop to end")
        Debug.Print entry
    Loop Until entry = "Stop"
End Sub
```

```
'Will return whether an element is present in the array
Public Function IsInArray(values() As String, ByVal whatToFind As String) As Boolean
    Dim i As Integer
    i = 0

    While i < UBound(values) And values(i) <> whatToFind
        i = i + 1
    Wend

    IsInArray = values(i) = whatToFind
End Function
```

For◦

```
Dim i as Integer           'Declaration of i
For i = 1 to 10            'Declare how many times the loop shall be executed
    Debug.Print i          'The piece of code which is repeated
Next i                     'The end of the loop
```

Integer i ◦ For110i Debug.Print i - 110◦ Next◦

1◦ Step◦ Step◦ Step◦

```
Dim i As Integer
For i = 1 To 10 Step 2
    Debug.Print i       'Prints 1, 3, 5, 7, and 9
Next
```

ForDoWhile◦

```
Private Iterations As Long              'Module scope

Public Sub Example()
    Dim i As Long
    Iterations = 10
    For i = 1 To Iterations
        Debug.Print Iterations     'Prints 10 through 1, descending.
        Iterations = Iterations - 1
    Next
End Sub
```

Exit ForFor

```
Dim i As Integer

For i = 1 To 10
    If i > 5 Then
        Exit For
    End If
    Debug.Print i       'Prints 1, 2, 3, 4, 5 before loop exits early.
Next
```

https://riptutorial.com/zh-CN/vba/topic/1873/

34:

LenVBALenB。 Unicode。

Examples

Len

```
Const baseString As String = "Hello World"

Dim charLength As Long

charLength = Len(baseString)
'charlength = 11
```

LenB

```
Const baseString As String = "Hello World"

Dim byteLength As Long

byteLength = LenB(baseString)
'byteLength = 22
```

`LenmyString= 0```myString =“”`

。

```
Const myString As String = vbNullString

'Prefer this method when checking if myString is a zero-length string
If Len(myString) = 0 Then
    Debug.Print "myString is zero-length"
End If

'Avoid using this method when checking if myString is a zero-length string
If myString = vbNullString Then
    Debug.Print "myString is zero-length"
End If
```

https://riptutorial.com/zh-CN/vba/topic/3576/

35:

Examples

UserForm◦ Shift + F7F7◦

◦

◦ /

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then
        '...
    End If
End With
```

```
UserForm1.Show vbModal
If Not UserForm1.IsCancelled Then
    '...
End If
```

“X”/Unload Me ◦

◦

Click ◦

◦

UserForm◦

◦

- Property Get ◦

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then
        MsgBox .Message, vbInformation
    End If
End With
```

```
With New UserForm1
    .Show vbModal
```

```
    If Not .IsCancelled Then
        MsgBox .txtMessage.Text, vbInformation
    End If
End With
```

QueryClose。

/OkCancel; “X” 。

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then 'if QueryClose isn't handled, this can raise a runtime error.
        '...
    End With
End With
```

QueryCloseCancelTrue

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    Cancel = True
    Me.Hide
End Sub
```

“X”。

。

。

Unload Me。 Me.Hide。

。

F4。 - TextBox1220。

UserForm

- lblUserNameLabel。
- TextBoxtxtUserName。
- ComboBoxcboUserName。
- ListBoxlstUserName。
- btnOkcmdOk“Ok”Button。

UIComboBoxListBox - UI。

- UserNameLabel。
- UserNameInput

。

- OkButton“Ok”。

。 。

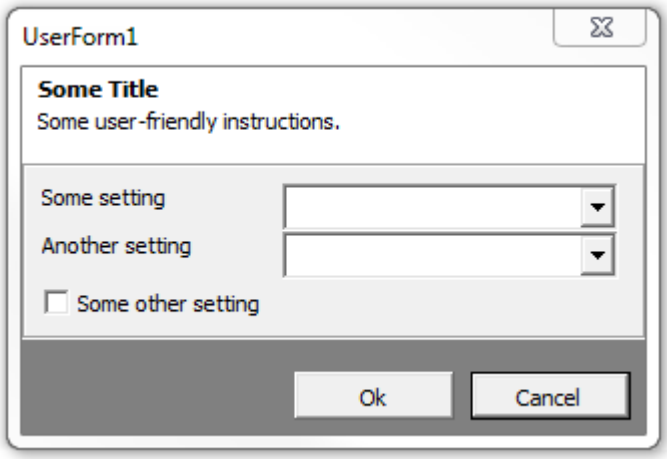
QueryClose

QueryClose。 CloseModeVbQueryClose

vbFormControlMenu		0
vbFormCode	Unload	1
vbAppWindows	Windows	2
vbAppTaskManager	Windows	3
vbFormMDIForm	VBA	4

。

“ ”




```
Option Explicit
Private Type TView
    IsCancelled As Boolean
    SomeOtherSetting As Boolean
    'other properties skipped for brievety
End Type
Private this As TView

Public Property Get IsCancelled() As Boolean
    IsCancelled = this.IsCancelled
End Property
```

```
Public Property Get SomeOtherSetting() As Boolean
    SomeOtherSetting = this.SomeOtherSetting
End Property

'...more properties...

Private Sub SomeOtherSettingInput_Change()
    this.SomeOtherSetting = CBool(SomeOtherSettingInput.Value)
End Sub

Private Sub OkButton_Click()
    Me.Hide
End Sub

Private Sub CancelButton_Click()
    this.IsCancelled = True
    Me.Hide
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    If CloseMode = VbQueryClose.vbFormControlMenu Then
        Cancel = True
        this.IsCancelled = True
        Me.Hide
    End If
End Sub
```

```
Public Sub DoSomething()
    With New UserForm1
        .Show vbModal
        If .IsCancelled Then Exit Sub
        If .SomeOtherSetting Then
            'setting is enabled
        Else
            'setting is disabled
        End If
    End With
End Sub
```

“ ”IsCancelledTrue。

https://riptutorial.com/zh-CN/vba/topic/5351/

36:

- IdentifierName [ ]
- IdentifierName [*arguments*]
- [Let | Set] *expression* = IdentifierName [*arguments*]
- [Let | Set] IdentifierName [*arguments*] = *expression*

IdentifierName	◦
	◦

Sub;◦

◦ ◦

FunctionProperty Get;◦ Let Set◦

Property LetProperty Set;*expression*value◦

Examples

```
ProcedureName
ProcedureName argument1, argument2
```

◦

Call

```
Call DoSomething : DoSomethingElse
```

DoSomethingDoSomethingElse◦ CallDoSomething

```
DoSomething: DoSomethingElse 'only DoSomethingElse will run
```

FunctionProperty Get

```
result = ProcedureName
result = ProcedureName(argument1, argument2)
```

◦ ◦

◦

◦ ◦

◦

◦ ...

```
Sub DoSomething(ByRef target As Range)
End Sub
```

...

```
DoSomething (Application.ActiveCell) 'raises an error at runtime
```

“Object Required”424◦ Application.ActiveCell Range targetByRef◦ ByValDoSomething Application.ActiveCell.Value ◦

VBAByVal◦ ◦

```
MsgBox ("Invalid Code!", vbCritical)
```

("Invalid Code!", vbCritical)◦

```
MsgBox ("Invalid Code!"), (vbCritical)
```

◦ ◦

```
Call ProcedureName
Call ProcedureName(argument1, argument2)
```

Call;◦ VB◦

◦ ◦

ProcedureName argument1 argument2 optArgument3

```
' Without optional argument
result = ProcedureName("A", "B")

' With optional argument
result = ProcedureName("A", "B", "C")

' Using named arguments (allows a different order)
result = ProcedureName(optArgument3:="C", argument1:="A", argument2:="B")

' Mixing named and unnamed arguments
result = ProcedureName("A", "B", optArgument3:="C")
```

```
Function ProcedureName(argument1 As String, argument2 As String, Optional optArgument3 As
String) As String
```

Optional◦ - ◦

```
Function ProcedureName(argument1 As String, argument2 As String, Optional optArgument3 As
String = "C") As String
```

c"C" ◦ ◦

https://riptutorial.com/zh-CN/vba/topic/1179/

37:

INSTR FINDLENMID LEFTRIGHT。

MIDRIGHT

Examples

MID。 。

```
DIM strEmpty as String, strNull as String, theText as String
DIM idx as Integer
DIM letterCount as Integer
DIM result as String

strNull = NOTHING
strEmpty = ""
theText = "1234, 78910"

' _________________
' Extract the word after the comma ", "  and before "910"  result: "78" ***
' _________________

' Get index (place) of comma using INSTR
idx = ...   ' some explanation here
if idx < ... ' check if no comma found in text

' or get index of comma using FIND
idx = ...   ' some explanation here... Note: The difference is...
if idx < ...  ' check if no comma found in text

result = MID(theText, ..., LEN(...

' Retrieve remaining word after the comma
result = MID(theText, idx+1, LEN(theText) - idx+1)

' Get word until the comma using LEFT
result = LEFT(theText, idx - 1)

' Get remaining text after the comma-and-space using RIGHT
result = ...

' What happens when things go wrong
result = MID(strNothing, 1, 2)    ' this causes ...
result = MID(strEmpty, 1, 2) ' which causes...
result = MID(theText, 30, 2) ' and now...
result = MID(theText, 2, 999) ' no worries...
result = MID(theText, 0, 2)
result = MID(theText, 2, 0)
result = MID(theText -1, 2)
result = MID(theText 2, -1)
idx = INSTR(strNothing, "123")
idx = INSTR(theText, strNothing)
idx = INSTR(theText, strEmpty)
i = LEN(strEmpty)
i = LEN(strNothing) '...
```

◦ ◦

https://riptutorial.com/zh-CN/vba/topic/8890/

38:

Visual Basic。Windows ShellInternet ExplorerXML HttpRequest。

- expression.CreateObject
- ;。Application。
- ;。。OLE。
- MSDN-

  Visual Basic。Visual Basic。
- MSDN-

  Visual Basic。。
- MSDN

  。
- MSDN-CreateObject

  Automation。CreateObject。

Examples

VBScript

```
Set createVBScriptRegExObject = CreateObject("vbscript.RegExp")
```

>> Microsoft VBScript。
DLLVBScript.dll
Internet Explorer 1.05.5

- MSDN-MicrosoftVBScript
- MSDN-
- expert-exchange - Visual Basic for ApplicationsVisual Basic6
- Microsoft Excel SO。
- regular-expressions.info/vbscript
- regular-expressions.info/vbscriptexample
- WIKI-

RegEx11excel。

```
Public Function getRegExResult(ByVal SourceString As String, Optional ByVal RegExPattern As
String = "\d+", _
```

```
    Optional ByVal isGlobalSearch As Boolean = True, Optional ByVal isCaseSensitive As Boolean
= False, Optional ByVal Delimiter As String = ";") As String

    Static RegExObject As Object
    If RegExObject Is Nothing Then
        Set RegExObject = createVBScriptRegExObject
    End If

    getRegExResult = removeLeadingDelimiter(concatObjectItems(getRegExMatches(RegExObject,
SourceString, RegExPattern, isGlobalSearch, isCaseSensitive), Delimiter), Delimiter)

End Function

Private Function getRegExMatches(ByRef RegExObj As Object, _
    ByVal SourceString As String, ByVal RegExPattern As String, ByVal isGlobalSearch As
Boolean, ByVal isCaseSensitive As Boolean) As Object

    With RegExObj
        .Global = isGlobalSearch
        .IgnoreCase = Not (isCaseSensitive) 'it is more user friendly to use positive meaning
of argument, like isCaseSensitive, than to use negative IgnoreCase
        .Pattern = RegExPattern
        Set getRegExMatches = .Execute(SourceString)
    End With

End Function

Private Function concatObjectItems(ByRef Obj As Object, Optional ByVal DelimiterCustom As
String = ";") As String
    Dim ObjElement As Variant
    For Each ObjElement In Obj
        concatObjectItems = concatObjectItems & DelimiterCustom & ObjElement.Value
    Next
End Function

Public Function removeLeadingDelimiter(ByVal SourceString As String, ByVal Delimiter As
String) As String
    If Left$(SourceString, Len(Delimiter)) = Delimiter Then
        removeLeadingDelimiter = Mid$(SourceString, Len(Delimiter) + 1)
    End If
End Function

Private Function createVBScriptRegExObject() As Object
    Set createVBScriptRegExObject = CreateObject("vbscript.RegExp") 'ex.:
createVBScriptRegExObject.Pattern
End Function
```

```
Set createScriptingFileSystemObject = CreateObject("Scripting.FileSystemObject")
```

>> Microsoft Scripting Runtime
DLLScrRun.dll
Windows

MSDN-FileSystemObject

FSO。object.method。Visual Basic。

FSO。。

MSDN-FileSystemObject “ *.....FileSystemObject。* ” VBAexceltrick-FileSystemObject - Scripting.FileSystemObject

```
Set dict = CreateObject("Scripting.Dictionary")
```

>> Microsoft Scripting Runtime
DLLScrRun.dll
Windows

Scripting.Dictionary
MSDN-

Internet Explorer

```
Set createInternetExplorerObject = CreateObject("InternetExplorer.Application")
```

>> Microsoft Internet Controls
DLLieframe.dll
Internet Explorer

MSDN-InternetExplorer

Windows Internet Explorer。

Internet Explorer Objec

IEVBA。 。

```
Sub IEGetToKnow()
    Dim IE As InternetExplorer 'Reference to Microsoft Internet Controls
    Set IE = New InternetExplorer

    With IE
        .Visible = True 'Sets or gets a value that indicates whether the object is visible or
hidden.

        'Navigation
        .Navigate2 "http://www.example.com" 'Navigates the browser to a location that might
not be expressed as a URL, such as a PIDL for an entity in the Windows Shell namespace.
        Debug.Print .Busy 'Gets a value that indicates whether the object is engaged in a
navigation or downloading operation.
        Debug.Print .ReadyState 'Gets the ready state of the object.
        .Navigate2 "http://www.example.com/2"
        .GoBack 'Navigates backward one item in the history list
        .GoForward 'Navigates forward one item in the history list.
        .GoHome 'Navigates to the current home or start page.
        .Stop 'Cancels a pending navigation or download, and stops dynamic page elements, such
as background sounds and animations.
        .Refresh 'Reloads the file that is currently displayed in the object.

        Debug.Print .Silent 'Sets or gets a value that indicates whether the object can
display dialog boxes.
```

```
        Debug.Print .Type 'Gets the user type name of the contained document object.

        Debug.Print .Top 'Sets or gets the coordinate of the top edge of the object.
        Debug.Print .Left 'Sets or gets the coordinate of the left edge of the object.
        Debug.Print .Height 'Sets or gets the height of the object.
        Debug.Print .Width 'Sets or gets the width of the object.
    End With

    IE.Quit 'close the application window
End Sub
```

IE。。

example.com

```
<!doctype html>
<html>
    <head>
        <title>Example Domain</title>
        <meta charset="utf-8" />
        <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <style ... </style>
    </head>

    <body>
        <div>
            <h1>Example Domain</h1>
            <p>This domain is established to be used for illustrative examples in documents.
You may use this
            domain in examples without prior coordination or asking for permission.</p>
            <p><a href="http://www.iana.org/domains/example">More information...</a></p>
        </div>
    </body>
</html>
```

```
Sub IEWebScrape1()
    Dim IE As InternetExplorer 'Reference to Microsoft Internet Controls
    Set IE = New InternetExplorer

    With IE
        .Visible = True
        .Navigate2 "http://www.example.com"

        'we add a loop to be sure the website is loaded and ready.
        'Does not work consistently. Cannot be relied upon.
        Do While .Busy = True Or .ReadyState <> READYSTATE_COMPLETE 'Equivalent = .ReadyState
<> 4
            ' DoEvents - worth considering. Know implications before you use it.
            Application.Wait (Now + TimeValue("00:00:01")) 'Wait 1 second, then check again.
        Loop

        'Print info in immediate window
        With .Document 'the source code HTML "below" the displayed page.
            Stop 'VBE Stop. Continue line by line to see what happens.
            Debug.Print .GetElementsByTagName("title")(0).innerHtml 'prints "Example Domain"
            Debug.Print .GetElementsByTagName("h1")(0).innerHtml 'prints "Example Domain"
            Debug.Print .GetElementsByTagName("p")(0).innerHtml 'prints "This domain is
```

```
established..."
            Debug.Print .GetElementsByTagName("p")(1).innerHtml 'prints "<a
href="http://www.iana.org/domains/example">More information...</a>"
            Debug.Print .GetElementsByTagName("p")(1).innerText 'prints "More information..."
            Debug.Print .GetElementsByTagName("a")(0).innerText 'prints "More information..."

            'We can change the localy displayed website. Don't worry about breaking the site.
            .GetElementsByTagName("title")(0).innerHtml = "Psst, scraping..."
            .GetElementsByTagName("h1")(0).innerHtml = "Let me try something fishy." 'You have
just changed the local HTML of the site.
            .GetElementsByTagName("p")(0).innerHtml = "Lorem ipsum.......... The End"
            .GetElementsByTagName("a")(0).innerText = "iana.org"
        End With '.document

        .Quit 'close the application window
    End With 'ie

End Sub
```

.Document HTML。。

`IE.Document.GetElementsByTagName("title")(0).innerHtml`。 `GetElementsByTagName`“ *title* ”HTML。。0。`(0)`。`innerHtml` StringElement Object。。

```
Sub IEGoToPlaces()
    Dim IE As InternetExplorer 'Reference to Microsoft Internet Controls
    Set IE = New InternetExplorer

    With IE
        .Visible = True
        .Navigate2 "http://www.example.com"
        Stop 'VBE Stop. Continue line by line to see what happens.

        'Click
        .Document.GetElementsByTagName("a")(0).Click
        Stop 'VBE Stop.

        'Return Back
        .GoBack
        Stop 'VBE Stop.

        'Navigate using the href attribute in the <a> tag, or "link"
        .Navigate2 .Document.GetElementsByTagName("a")(0).href
        Stop 'VBE Stop.

        .Quit 'close the application window
    End With
End Sub
```

Microsoft HTMLIE

IEHTML*Microsoft HTML Object Library*。。

IE

IE。`Do While... Loop`。

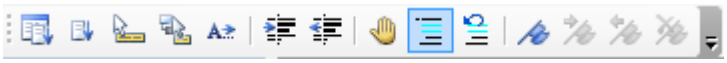
IEHTMLOVERKILL。CSSJavaScripts。。XML HTTPRequest。。

https://riptutorial.com/zh-CN/vba/topic/8916/

39:

IDE“ ”

-



-

VBA。

Examples

' 。 。

。 '。 。

```
Sub InlineDocumentation()
  'Comments start with an "'"

  'They can be place before a line of code, which prevents the line from executing
  'Debug.Print "Hello World"

  'They can also be placed after a statement
  'The statement still executes, until the compiler arrives at the comment
  Debug.Print "Hello World"  'Prints a welcome message

'Comments can have 0 indention....
     '... or as much as needed

  '''' Comments can contain multiple apostrophes ''''

  'Comments can span lines (using line continuations) _
    but this can make for hard to read code

  'If you need to have mult-line comments, it is often easier to
  'use an apostrophe on each line

  'The continued statement syntax (:) is treated as part of the comment, so
  'it is not possible to place an executable statement after a comment
  'This won't run : Debug.Print "Hello World"
End Sub

'Comments can appear inside or outside a procedure
```

REM

```
Sub RemComments()
  Rem Comments start with "Rem" (VBA will change any alternate casing to "Rem")
```

```
  Rem is an abbreviation of Remark, and similar to DOS syntax
  Rem Is a legacy approach to adding comments, and apostrophes should be preferred

  Rem Comments CANNOT appear after a statement, use the apostrophe syntax instead
  Rem Unless they are preceded by the instruction separator token
  Debug.Print "Hello World": Rem prints a welcome message
  Debug.Print "Hello World" 'Prints a welcome message

  'Rem cannot be immediately followed by the following characters "!,@,#,$,%,&"
  'Whereas the apostrophe syntax can be followed by any printable character.

End Sub

Rem Comments can appear inside or outside a procedure
```

https://riptutorial.com/zh-CN/vba/topic/2059/

40:

-
-
-
-
-

◦ ()◦

Examples

^		◦ Double ◦ ◦
/	1	◦ Double ◦ ◦
*	1	2◦
\		.5◦ ◦ 0 11◦ - 3 \ 0.4◦
Mod		◦ .5◦ 8.6 Mod 312 Mod 2.60 ◦ 0 11◦ - 3 Mod 0.4◦
_	2	◦
+	2	2◦ String◦ ◦

1◦

2◦

VBA2 +& String - StringString ◦

&StringString ◦

++◦ +◦ StringStringVariant

```
Public Sub Example()
    Dim left As String
    Dim right As String

    left = "5"
    right = "5"

    Debug.Print left + right    'Prints "55"
End Sub
```

String

```
Public Sub Example()
    Dim left As Variant
    Dim right As String

    left = 5
    right = "5"

    Debug.Print left + right    'Prints 10
End Sub
```

- Variant&。

=		True。 。
<>		True。
>	...	True。
<		True。
>=		True。
<=		True。
Is		True。 NothingNothing。 IsObject。 Variant vtEmpty 424 - “”。 True。 ObjPtr(left) = ObjPtr(right) 。

VBA“”。 Boolean。 .....

```
a = 2: b = 1: c = 0
expr = a > b > c
```

...baca。 VBA

```
a = 2: b = 1: c = 0
expr = a > b > c
expr = (2 > 1) > 0
expr = True > 0
expr = -1 > 0 'CInt(True) = -1
expr = False
```

IsObjectObject。 438 - “”。

Object91 - “”。

NothingIs - “”。

Object*Object* VBA。 SomeClassValue ChildClassChildValue。 .....

```
Set x = New SomeClass
Debug.Print x > 42
```

.....

```
Set x = New SomeClass
Debug.Print x.Value.ChildValue > 42
```

StringString Variant ◦ String13 - “”◦

StringStringVariant Option Compare◦ ◦ String

```
Public Sub Example()
    Dim left As Variant
    Dim right As Variant

    left = "42"
    right = "5"
    Debug.Print left > right              'Prints False
    Debug.Print Val(left) > Val(right)    'Prints True
End Sub
```

StringVariant◦

Date Double◦

StringDateStringVariant StringDate ◦ Date 13 - “”◦

DoubleSingle◦ VBADoubleTrue

```
Public Sub Example()
    Dim Test As Double

    Test = 42        Debug.Print CBool(Test)             'Prints True.
    'True is promoted to Double - Test is not cast to Boolean
    Debug.Print Test = True             'Prints False

    'With explicit casts:
    Debug.Print CBool(Test) = True      'Prints True
    Debug.Print CDbl(-1) = CDbl(True)   'Prints True
End Sub
```

\

VBA“”◦ ◦ VBA False True ◦ True◦ Boolean ◦

◦ 0False 1True ◦

And

TrueTrue ◦

0	0	0
0	1	0
1	0	0
1	1	1

Or

TrueTrue ◦

0	0	0
0	1	1
1	0	1
1	1	1

Not

TrueFalseFalse True ◦

0	1
1	0

Not◦ Visual Basic◦ .....

```
Debug.Print x Not y
```

..... VBE

```
Debug.Print Not x
```

Not◦

Xor

“”◦ True ◦

0	0	0
0	1	1
1	0	1
1	1	0

Xor <>

Eqv

“”◦ True ◦

0	0	1
0	1	0
1	0	0
1	1	1

Eqvx Eqv yNot (x Xor y)

Imp

“”◦ TrueTrue ◦

0	0	1
0	1	1
1	0	0
1	1	1

Imp◦ ◦

https://riptutorial.com/zh-CN/vba/topic/5813/

41:

&◦

Join◦

Examples

```
Const string1 As String = "foo"
Const string2 As String = "bar"
Const string3 As String = "fizz"
Dim concatenatedString As String

'Concatenate two strings
concatenatedString = string1 & string2
'concatenatedString = "foobar"

'Concatenate three strings
concatenatedString = string1 & string2 & string3
'concatenatedString = "foobarfizz"
```

Join

```
'Declare and assign a string array
Dim widgetNames(2) As String
widgetNames(0) = "foo"
widgetNames(1) = "bar"
widgetNames(2) = "fizz"

'Concatenate with Join and separate each element with a 3-character string
concatenatedString = VBA.Strings.Join(widgetNames, " > ")
'concatenatedString = "foo > bar > fizz"

'Concatenate with Join and separate each element with a zero-width string
concatenatedString = VBA.Strings.Join(widgetNames, vbNullString)
'concatenatedString = "foobarfizz"
```

https://riptutorial.com/zh-CN/vba/topic/3580/

42:

。 。 。 '28""。

。

。

Examples

```
Function Factorial(Value As Long) As Long
    If Value = 0 Or Value = 1 Then
         Factorial = 1
    Else
       Factorial = Factorial(Value - 1) * Value
    End If
End Function
```

Microsoft Scripting Runtime

```
    Sub EnumerateFilesAndFolders( _
        FolderPath As String, _
        Optional MaxDepth As Long = -1, _
        Optional CurrentDepth As Long = 0, _
        Optional Indentation As Long = 2)

        Dim FSO As Scripting.FileSystemObject
        Set FSO = New Scripting.FileSystemObject

        'Check the folder exists
        If FSO.FolderExists(FolderPath) Then
            Dim fldr As Scripting.Folder
            Set fldr = FSO.GetFolder(FolderPath)

            'Output the starting directory path
            If CurrentDepth = 0 Then
              Debug.Print fldr.Path
            End If

            'Enumerate the subfolders
            Dim subFldr As Scripting.Folder
            For Each subFldr In fldr.SubFolders
                Debug.Print Space$((CurrentDepth + 1) * Indentation) & subFldr.Name
                If CurrentDepth < MaxDepth Or MaxDepth = -1 Then
                    'Recursively call EnumerateFilesAndFolders
                    EnumerateFilesAndFolders subFldr.Path, MaxDepth, CurrentDepth + 1,
Indentation
                End If
            Next subFldr

            'Enumerate the files
            Dim fil As Scripting.File
            For Each fil In fldr.Files
                Debug.Print Space$((CurrentDepth + 1) * Indentation) & fil.Name
```

```
            Next fil
        End If
    End Sub
```

https://riptutorial.com/zh-CN/vba/topic/3236/

43:

Examples

◦ ◦

◦ ◦

---

91 - With

◦

```
Private Sub DoSomething(ByVal target As Worksheet)
    Debug.Print target.Name
End Sub
```

target

```
Private Sub DoSomething(ByVal target As Worksheet)
    If target Is Nothing Then Exit Sub
    Debug.Print target.Name
End Sub
```

target◦

◦

---

9 -

◦

```
Private Sub DoSomething(ByVal index As Integer)
    Debug.Print ActiveWorkbook.Worksheets(index)
End Sub
```

ActiveWorkbook◦

```
Private Sub DoSomething(ByVal index As Integer)
    If index > ActiveWorkbook.Worksheets.Count Or index <= 0 Then Exit Sub
    Debug.Print ActiveWorkbook.Worksheets(index)
End Sub
```

If - ◦

◦ On Error GoToVBA“”◦ Resume “”◦

BASICGoToGoSubReturn“”。

-
-

```
Private Sub DoSomething()
    On Error GoTo CleanFail

    'procedure code here

CleanExit:
    'cleanup code here
    Exit Sub

CleanFail:
    'error-handling code here
    Resume CleanExit
End Sub
```

◦ Err -

```
CleanExit:
    Exit Sub

CleanFail:
    Select Case Err.Number
        Case 9
            MsgBox "Specified number doesn't exist. Please try again.", vbExclamation
            Resume
        Case 91
            'woah there, this shouldn't be happening.
            Stop 'execution will break here
            Resume 'hit F8 to jump to the line that raised the error
        Case Else
            MsgBox "An unexpected error has occurred:" & vbNewLine & Err.Description,
vbCritical
            Resume CleanExit
    End Select
End Sub
```

◦ -

```
Private Sub DoSomething(CheckValue as Long)

    If CheckValue = 0 Then
        On Error GoTo ErrorHandler   ' turn error handling on
        ' code that may result in error
        On Error GoTo 0              ' turn error handling off - same level
    End If

CleanExit:
    Exit Sub

ErrorHandler:
    ' error handling code here
```

```
    ' do not turn off error handling here
    Resume

End Sub
```

VBAQBASIC。 Erl hidden。 Erl0。

```
Sub DoSomething()
10 On Error GoTo 50
20 Debug.Print 42 / 0
30 Exit Sub
40
50 Debug.Print "Error raised on line " & Erl ' returns 20
End Sub
```

Erl 。

```
Sub DoSomething()
10 On Error GoTo 50
   Debug.Print 42 / 0
30 Exit Sub

50 Debug.Print "Error raised on line " & Erl 'returns 10
End Sub
```

ErlInteger。

```
Sub DoSomething()
99997 On Error GoTo 99999
99998 Debug.Print 42 / 0
99999
      Debug.Print Erl   'Prints 34462
End Sub
```

。

- 。
- Resume。

ResumeVBAResume20“Resume without error”。

Resume

- Resume 。 。
- Resume Next 。 。
- Resume [line label] 。 。

On Error Resume Next

On ErrorResumeVBA。

。

。 ***On Error Resume Next*** 。 / - 。 On Error Resume Next 。

On Error - On Error。

Resume Next。 On Error

```
On Error Resume Next
[possibly-failing statement]
Err.Clear 'resets current error
On Error GoTo 0
```

On Error GoTo 0 。 。

```
Public Sub Caller()
    On Error GoTo Handler

    Callee

    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & " in Caller."
End Sub

Public Sub Callee()
    On Error GoTo Handler

    Err.Raise 1     'This will be handled by the Callee handler.
    On Error GoTo 0 'After this statement, errors are passed up the stack.
    Err.Raise 2     'This will be handled by the Caller handler.

    Exit Sub
Handler:
    Debug.Print "Error " & Err.Number & " in Callee."
    Resume Next
End Sub
```

/。 Enum

```
Option Explicit
Public Enum FoobarError
    Err_FooWasNotBarred = vbObjectError + 1024
    Err_BarNotInitialized
    Err_SomethingElseHappened
End Enum
```

vbObjectError/。 Enum1Err_BarNotInitializedvbObjectError + 1025 。

Err.RaiseErr_FooWasNotBarred

```
Err.Raise Err_FooWasNotBarred
```

Err.RaiseDescriptionSource -

```
Private Const Msg_FooWasNotBarred As String = "The foo was not barred."
Private Const Msg_BarNotInitialized As String = "The bar was not initialized."
```

```
Private Sub OnFooWasNotBarredError(ByVal source As String)
    Err.Raise Err_FooWasNotBarred, source, Msg_FooWasNotBarred
End Sub

Private Sub OnBarNotInitializedError(ByVal source As String)
    Err.Raise Err_BarNotInitialized, source, Msg_BarNotInitialized
End Sub
```

```
Public Sub DoSomething()
    'raises the custom 'BarNotInitialized' error with "DoSomething" as the source:
    If Me.Bar Is Nothing Then OnBarNotInitializedError "DoSomething"
    '...
End Sub
```

Err_BarNotInitialized◦

ErrorErr.Raise /◦

https://riptutorial.com/zh-CN/vba/topic/3211/

44:

Collection VBA。 。 Collection。

		1	
			2
		N / A	3
		N / A	.Exists
	。 .Remove	Erase ReDim	.RemoveAll

1。

2 .Keys .Items。

3 .CompareMode。

Examples

.Add Collection

```
.Add(item, [key], [before, after])
```

	Collection 。 Nothing 。
	。 String Collection。 Collection 457“”。
	。 Collection String 。 *after* 5“”。 CollectionString 5“”。 Collection 9“”。
	。 Collection String 。 *before*。 *before*。

- 。 .Add "Bar", "Foo".Add "Baz", "foo"。
- *beforeafter* Collection。
- *beforeafter*。 。

```
Public Sub Example()
    Dim foo As New Collection

    With foo
        .Add "One"            'No key. This item can only be retrieved by index.
        .Add "Two", "Second"  'Key given. Can be retrieved by key or index.
        .Add "Three", , 1     'Inserted at the start of the collection.
        .Add "Four", , , 1    'Inserted at index 2.
    End With

    Dim member As Variant
    For Each member In foo
        Debug.Print member    'Prints "Three, Four, One, Two"
    Next
End Sub
```

.RemoveCollection

```
.Remove(index)
```

Collection◦ Variant ◦ StringVariant ◦ CollectionString5“”◦ Collection9“”◦

- CollectionCollection ◦ For Step -1 ◦
- For EachCollection◦

```
Public Sub Example()
    Dim foo As New Collection

    With foo
        .Add "One"
        .Add "Two", "Second"
        .Add "Three"
        .Add "Four"
    End With

    foo.Remove 1            'Removes the first item.
    foo.Remove "Second"     'Removes the item with key "Second".
    foo.Remove foo.Count    'Removes the last item.

    Dim member As Variant
    For Each member In foo
        Debug.Print member  'Prints "Three"
    Next
End Sub
```

.CountCollection

```
.Count()
```

```
Public Sub Example()
    Dim foo As New Collection
```

```
    With foo
        .Add "One"
        .Add "Two"
        .Add "Three"
        .Add "Four"
    End With

    Debug.Print foo.Count   'Prints 4
End Sub
```

.ItemCollection◦

```
.Item(index)
```

Collection◦ Variant ◦ StringVariant ◦ CollectionString5“”◦ Collection9“”◦

- .ItemCollection◦ ◦
- 1◦
- ◦ .Item("Foo").Item("foo")◦
- *index*String◦ .Item(1).Item("1")Collection◦

```
Public Sub Example()
    Dim foo As New Collection

    With foo
        .Add "One"
        .Add "Two"
        .Add "Three"
        .Add "Four"
    End With

    Dim index As Long
    For index = 1 To foo.Count
        Debug.Print foo.Item(index) 'Prints One, Two, Three, Four
    Next
End Sub
```

```
Public Sub Example()
    Dim keys() As String
    keys = Split("Foo,Bar,Baz", ",")
    Dim values() As String
    values = Split("One,Two,Three", ",")

    Dim foo As New Collection
    Dim index As Long
    For index = LBound(values) To UBound(values)
        foo.Add values(index), keys(index)
    Next

    Debug.Print foo.Item("Bar") 'Prints "Two"
```

```
End Sub
```

```
Public Sub Example()
    Dim foo As New Collection

    With foo
        .Add "One", "Foo"
        .Add "Two", "Bar"
        .Add "Three", "Baz"
    End With

    'All lines below print "Two"
    Debug.Print foo.Item("Bar")     'Explicit call syntax.
    Debug.Print foo("Bar")          'Default member call syntax.
    Debug.Print foo!Bar             'Bang syntax.
End Sub
```

bang `!` `.ItemString` 。

Scripting.Dictionary `CollectionCollection`。

```
Public Function KeyExistsInCollection(ByVal key As String, _
                                      ByRef container As Collection) As Boolean
    With Err
        If container Is Nothing Then .Raise 91
        On Error Resume Next
        Dim temp As Variant
        temp = container.Item(key)
        On Error GoTo 0

        If .Number = 0 Then
            KeyExistsInCollection = True
        ElseIf .Number <> 5 Then
            .Raise .Number
        End If
    End With
End Function
```

`CollectionCollection`。 Collection

```
Public Function ItemExistsInCollection(ByRef target As Variant, _
                                       ByRef container As Collection) As Boolean
    Dim candidate As Variant
    Dim found As Boolean

    For Each candidate In container
        Select Case True
            Case IsObject(candidate) And IsObject(target)
                found = candidate Is target
            Case IsObject(candidate), IsObject(target)
                found = False
            Case Else
                found = (candidate = target)
        End Select
        If found Then
```

```
            ItemExistsInCollection = True
            Exit Function
        End If
    Next
End Function
```

CollectionCollectionCollection

```
Public Sub Example()
    Dim foo As New Collection

    With foo
        .Add "One"
        .Add "Two"
        .Add "Three"
    End With

    Debug.Print foo.Count   'Prints 3
    Set foo = New Collection
    Debug.Print foo.Count   'Prints 0
End Sub
```

CollectionCollection ◦

```
Public Sub Example()
    Dim foo As New Collection
    Dim bar As Collection

    With foo
        .Add "One"
        .Add "Two"
        .Add "Three"
    End With

    Set bar = foo
    Set foo = New Collection

    Debug.Print foo.Count   'Prints 0
    Debug.Print bar.Count   'Prints 3
End Sub
```

Collection

```
Public Sub ClearCollection(ByRef container As Collection)
    Dim index As Long
    For index = 1 To container.Count
        container.Remove 1
    Next
End Sub
```

https://riptutorial.com/zh-CN/vba/topic/5838/

45:

VBAUnicode。 。

Examples

VBA

A1pangram

صفخلقخودكمثلالشمسإذبزغت - يحظىالضجيعبهانجلاءمعطار

VBA`AscW` `ChrW`。 `Byte`

```
Sub NonLatinStrings()

Dim rng As Range
Set rng = Range("A1")
Do Until rng = ""
    Dim MyString As String
    MyString = rng.Value

    ' AscW functions
    Dim char As String
    char = AscW(Left(MyString, 1))
    Debug.Print "First char (ChrW): " & char
    Debug.Print "First char (binary): " & BinaryFormat(char, 12)

    ' ChrW functions
    Dim uString As String
    uString = ChrW(char)
    Debug.Print "String value (text): " & uString        ' Fails! Appears as '?'
    Debug.Print "String value (AscW): " & AscW(uString)

    ' Using a Byte string
    Dim StringAsByt() As Byte
    StringAsByt = MyString
    Dim i As Long
    For i = 0 To 1 Step 2
        Debug.Print "Byte values (in decimal): " & _
            StringAsByt(i) & "|" & StringAsByt(i + 1)
        Debug.Print "Byte values (binary): " & _
            BinaryFormat(StringAsByt(i)) & "|" & BinaryFormat(StringAsByt(i + 1))
    Next i
    Debug.Print ""

    ' Printing the entire string to the immediate window fails (all '?'s)
    Debug.Print "Whole String" & vbNewLine & rng.Value
    Set rng = rng.Offset(1)
Loop

End Sub
```

Sad

ChrW1589
00011000110101
:?
AscW1589
53 | 6
00110101 | 00000110

??? ????? ????? ??????? ??????? ??? ??????? - ????? ???????? ???? ??????? ???????

VBA。IDE。

VBA 。

LojbanMapudungun

Dzongkha

```
Dim Yec’hed As String 'Breton
Dim «Dóna» As String 'Catalan
Dim fræk As String 'Danish
Dim tšellomängija As String 'Estonian
Dim Törkylempijävongahdus As String 'Finnish
Dim j’examine As String 'French
Dim Paß As String 'German
Dim þjófum As String 'Icelandic
Dim hÓighe As String 'Irish
Dim sofybakni As String 'Lojban (.o’i does not work)
Dim ñizol As String 'Mapudungun
Dim Vår As String 'Norwegian
Dim «brações» As String 'Portuguese
Dim d’fhàg As String 'Scottish Gaelic
```

VBA IDEStack Overflow。

«»“”。

https://riptutorial.com/zh-CN/vba/topic/10555/

46: VBA

Examples

◦

◦

```
Public Sub DoSomething()
    With New SomeForm
        Set .Model = CreateViewModel
        .Show vbModal
        If .IsCancelled Then Exit Sub
        ProcessUserData .Model
    End With
End Sub
```

DoSomething ProcessUserData -

```
Private Function CreateViewModel() As ISomeModel
    Dim result As ISomeModel
    Set result = SomeModel.Create(Now, Environ$("UserName"))
    result.AvailableItems = GetAvailableItems
    Set CreateViewModel = result
End Function
```

CreateViewModelISomeModel◦ - GetAvailableItems

```
Private Function GetAvailableItems() As Variant
    GetAvailableItems = DataSheet.Names("AvailableItems").RefersToRange
End Function
```

DataSheet◦ ◦

◦

QueryClose - ◦

```
Private Type TView
    IsCancelled As Boolean
    SomeOtherSetting As Boolean
    'other properties skipped for brievety
End Type
Private this As TView

'...

Private Sub SomeOtherSettingInput_Change()
    this.SomeOtherSetting = CBool(SomeOtherSettingInput.Value)
End Sub
```

- SomeOtherSetting

```
Public Property Get SomeOtherSetting() As Boolean
    SomeOtherSetting = this.SomeOtherSetting
End Property
```

SomeOtherSetting'state;◦ Boolean ◦

◦ UserNameTimestampPublic Property Property Let◦

AbstractionCreateViewModelISomeModel

```
Option Explicit

Public Property Get Timestamp() As Date
End Property

Public Property Get UserName() As String
End Property

Public Property Get AvailableItems() As Variant
End Property

Public Property Let AvailableItems(ByRef value As Variant)
End Property

Public Property Get SomeSetting() As String
End Property

Public Property Let SomeSetting(ByVal value As String)
End Property

Public Property Get SomeOtherSetting() As Boolean
End Property

Public Property Let SomeOtherSetting(ByVal value As Boolean)
End Property
```

TimestampUserNameProperty Get◦ SomeModel

```
Option Explicit
Implements ISomeModel

Private Type TModel
    Timestamp As Date
    UserName As String
    SomeSetting As String
    SomeOtherSetting As Boolean
    AvailableItems As Variant
End Type
Private this As TModel

Private Property Get ISomeModel_Timestamp() As Date
    ISomeModel_Timestamp = this.Timestamp
End Property
```

```
Private Property Get ISomeModel_UserName() As String
    ISomeModel_UserName = this.UserName
End Property

Private Property Get ISomeModel_AvailableItems() As Variant
    ISomeModel_AvailableItems = this.AvailableItems
End Property

Private Property Let ISomeModel_AvailableItems(ByRef value As Variant)
    this.AvailableItems = value
End Property

Private Property Get ISomeModel_SomeSetting() As String
    ISomeModel_SomeSetting = this.SomeSetting
End Property

Private Property Let ISomeModel_SomeSetting(ByVal value As String)
    this.SomeSetting = value
End Property

Private Property Get ISomeModel_SomeOtherSetting() As Boolean
    ISomeModel_SomeOtherSetting = this.SomeOtherSetting
End Property

Private Property Let ISomeModel_SomeOtherSetting(ByVal value As Boolean)
    this.SomeOtherSetting = value
End Property

Public Property Get Timestamp() As Date
    Timestamp = this.Timestamp
End Property

Public Property Let Timestamp(ByVal value As Date)
    this.Timestamp = value
End Property

Public Property Get UserName() As String
    UserName = this.UserName
End Property

Public Property Let UserName(ByVal value As String)
    this.UserName = value
End Property

Public Property Get AvailableItems() As Variant
    AvailableItems = this.AvailableItems
End Property

Public Property Let AvailableItems(ByRef value As Variant)
    this.AvailableItems = value
End Property

Public Property Get SomeSetting() As String
    SomeSetting = this.SomeSetting
End Property

Public Property Let SomeSetting(ByVal value As String)
    this.SomeSetting = value
End Property

Public Property Get SomeOtherSetting() As Boolean
```

```
    SomeOtherSetting = this.SomeOtherSetting
End Property

Public Property Let SomeOtherSetting(ByVal value As Boolean)
    this.SomeOtherSetting = value
End Property
```

Private◦ PublicISomeModel◦

VB_PredeclaredIdSomeModel VB.NETShared Cstatic

```
Private Function CreateViewModel() As ISomeModel
    Dim result As ISomeModel
    Set result = SomeModel.Create(Now, Environ$("UserName"))
    result.AvailableItems = GetAvailableItems
    Set CreateViewModel = result
End Function
```

ISomeModelTimestampUserName

```
Public Function Create(ByVal pTimeStamp As Date, ByVal pUserName As String) As ISomeModel
    With New SomeModel
        .Timestamp = pTimeStamp
        .UserName = pUserName
        Set Create = .Self
    End With
End Function

Public Property Get Self() As ISomeModel
    Set Self = Me
End Property
```

ISomeModelTimestampUserName◦

◦

UI◦

ISomeView

```
Option Explicit

Public Property Get IsCancelled() As Boolean
End Property

Public Property Get Model() As ISomeModel
End Property

Public Property Set Model(ByVal value As ISomeModel)
End Property

Public Sub Show()
End Sub
```

```
Option Explicit
Implements ISomeView

Private Type TView
    IsCancelled As Boolean
    Model As ISomeModel
End Type
Private this As TView

Private Property Get ISomeView_IsCancelled() As Boolean
    ISomeView_IsCancelled = this.IsCancelled
End Property

Private Property Get ISomeView_Model() As ISomeModel
    Set ISomeView_Model = this.Model
End Property

Private Property Set ISomeView_Model(ByVal value As ISomeModel)
    Set this.Model = value
End Property

Private Sub ISomeView_Show()
    Me.Show vbModal
End Sub

Private Sub SomeOtherSettingInput_Change()
    this.Model.SomeOtherSetting = CBool(SomeOtherSettingInput.Value)
End Sub

'...other event handlers...

Private Sub OkButton_Click()
    Me.Hide
End Sub

Private Sub CancelButton_Click()
    this.IsCancelled = True
    Me.Hide
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    If CloseMode = VbQueryClose.vbFormControlMenu Then
        Cancel = True
        this.IsCancelled = True
        Me.Hide
    End If
End Sub
```

`ISomeView` - `SomeViewMock`

```
Option Explicit
Implements ISomeView

Private Type TView
    IsCancelled As Boolean
    Model As ISomeModel
End Type
Private this As TView

Public Property Get IsCancelled() As Boolean
```

```
    IsCancelled = this.IsCancelled
End Property

Public Property Let IsCancelled(ByVal value As Boolean)
    this.IsCancelled = value
End Property

Private Property Get ISomeView_IsCancelled() As Boolean
    ISomeView_IsCancelled = this.IsCancelled
End Property

Private Property Get ISomeView_Model() As ISomeModel
    Set ISomeView_Model = this.Model
End Property

Private Property Set ISomeView_Model(ByVal value As ISomeModel)
    Set this.Model = value
End Property

Private Sub ISomeView_Show()
    'do nothing
End Sub
```

`UserFormISomeView`

```
Public Sub DoSomething(ByVal view As ISomeView)
    With view
        Set .Model = CreateViewModel
        .Show
        If .IsCancelled Then Exit Sub
        ProcessUserData .Model
    End With
End Sub
```

`DoSomething` `UserForm` `ProcessUserDataview.IsCancelledTrue` test`SomeViewMockIsCancelledTrue` `DoSomething`。

VBAIDE。 - 。

/`SomeViewMock`“”42。 。

VBA https://riptutorial.com/zh-CN/vba/topic/5357/vba

S. No		Contributors
1	VBA	0m3r, Andre Terra, Benno Grimm, Bookeater, Comintern, Community, Derpcode, Kaz, lfrandom, litelite, Maarten van Stam, Macro Man, Máté Juhász, Nick Dewitt, PankajKushwaha, RubberDuck, Stefan Pinnow
2	API	paul bica
3	CreateObjectGetObject	Branislav Kollár, Dave, Tim
4	Scripting.Dictionary	Comintern, Jeeped, Kyle, RamenChef, Tim, Wolf, Zev Spitz
5	Scripting.FileSystemObject	Comintern, Dave, Macro Man, Mikegrann, RubberDuck, Siva, Steve Rindsberg, ThunderFrame
6	VBA	Branislav Kollár, Macro Man, Mat's Mug
7	VBA	Jeeped, Maarten van Stam, Macro Man, Mat's Mug, RamenChef, RubberDuck, Stefan Pinnow, Thomas G, ThunderFrame
8	ByRefByVal	Branislav Kollár, Comintern, Mat's Mug, R3uK, RamenChef, ZygD
9	ADO	Comintern, SandPiper, Tazaf
10		ThunderFrame
11		Comintern, LiamH, Mat's Mug, Sivaprasath Vadivel, Tomas Zubiri
12		Branislav Kollár, Macro Man, Mat's Mug, Neil Mussett, ThunderFrame
13		FreeMan, Kaz, Mat's Mug, Victor Moraes
14	VBA2GB +	PatricK
15	FileSystemObject	Comintern, Macro Man, SandPiper
16		ThunderFrame
17		Comintern, dadde, Dave, Franck Dernoncourt, Jeeped, Kaz, lfrandom, litelite, Macro Man, Mark.R, Mat's Mug, Neil Mussett, RubberDuck, Shawn V. Wilson, SWa,

		Thierry Dalon, ThunderFrame, Tom, Victor Moraes, Zaider
18		Comintern, ThunderFrame
19		Mark.R
20		Mat's Mug, ThunderFrame
21	-	Comintern, ThunderFrame
22	VBA/	0m3r
23		ThunderFrame
24		hymced, Mat's Mug, RamenChef, RubberDuck
25		Neil Mussett
26		Neil Mussett
27		Comintern, FreeMan, Neil Mussett, StackzOfZtuff, Stephen Leppik, ThunderFrame
28		Blackhawk
29		Comintern, Dave, Hubisan, jamheadart, Josan Iracheta, Maarten van Stam, Mark.R, Mat's Mug, Miguel_Ryu, Tazaf
30		Comintern, FreeMan, Thomas G
31		Macro Man, Mat's Mug, RubberDuck, Steve Rindsberg
32		Mat's Mug
33		Benno Grimm, Comintern, Kelly Tessena Keck, Leviathan, litelite, Macro Man, Martin, Mat's Mug, Roland, Siva, ThunderFrame
34		Steve Rindsberg, ThunderFrame
35		Mat's Mug
36		Macro Man, Mat's Mug, Neil Mussett, Sam Johnson
37		pashute
38		Branislav Kollár
39		Comintern, Hosch250, Johnny C, litelite, Macro Man, Nijin22, Shawn V. Wilson, ThunderFrame
40		Comintern, Macro Man

41		ThunderFrame
42		Mat's Mug, ThunderFrame
43		Comintern, Logan Reed, Mat's Mug
44		Comintern
45		Neil Mussett
46	VBA	IvenBach, Mat's Mug