無料電子ブック

学習

VBA

Free unaffiliated eBook created from
Stack Overflow contributors.

#vba

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をいめる

このセクションでは、vbaのと、なぜがそれをいたいのかをします。

また、vbaのきなテーマについてもし、するトピックにリンクするがあります。 vbaのドキュメンテーションはしいものなので、これらのトピックのバージョンをするがあります。

バージョン

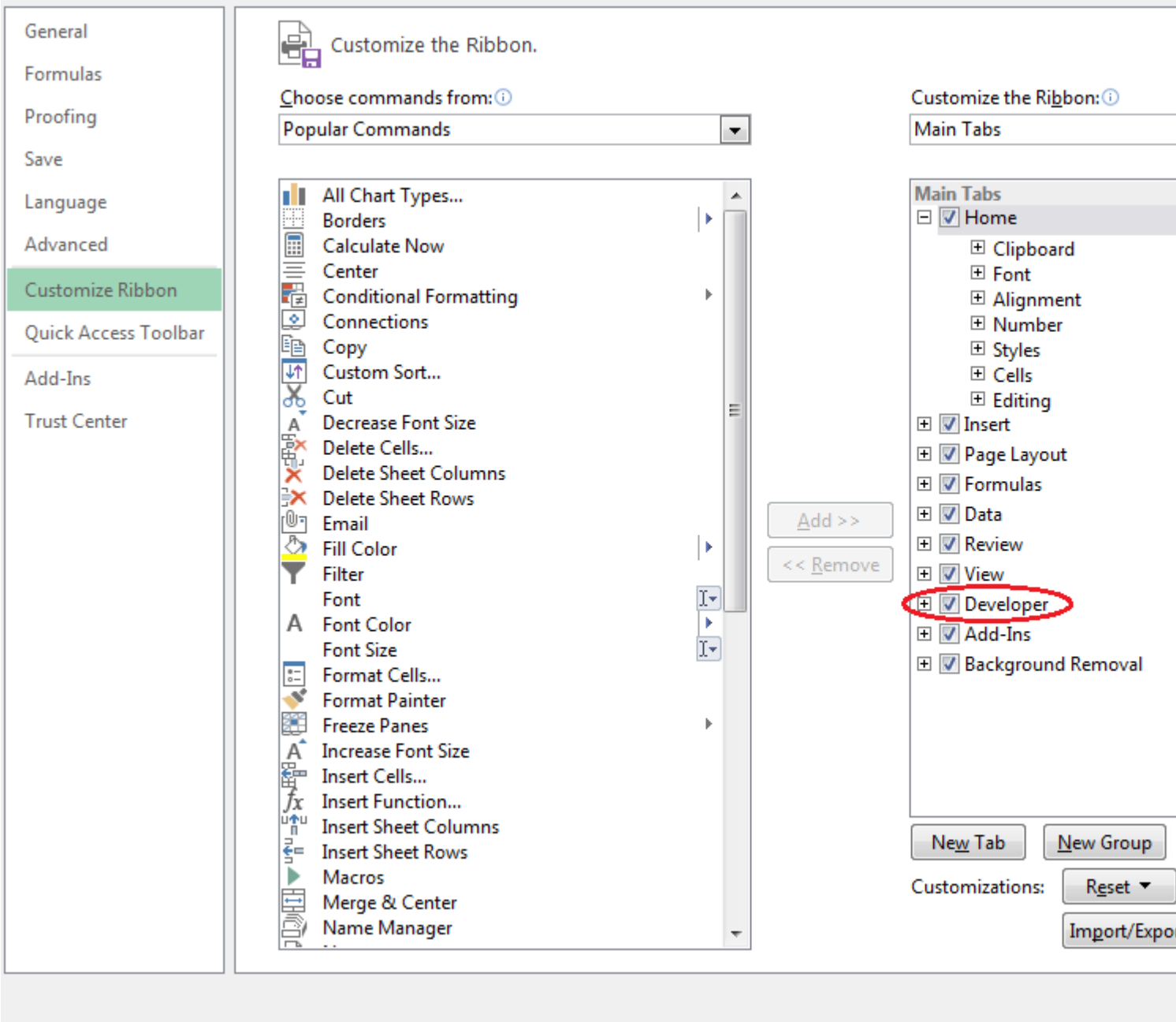
バージョン	Officeバージョン	リリースノート	
Vba6	- 2007	[いつか] [1]	1992-06-30
Vba7	20102016	[blog.techkit.com] [2]	2010-04-15
MacVBA	2004、20112016		2004-05-11

Examples

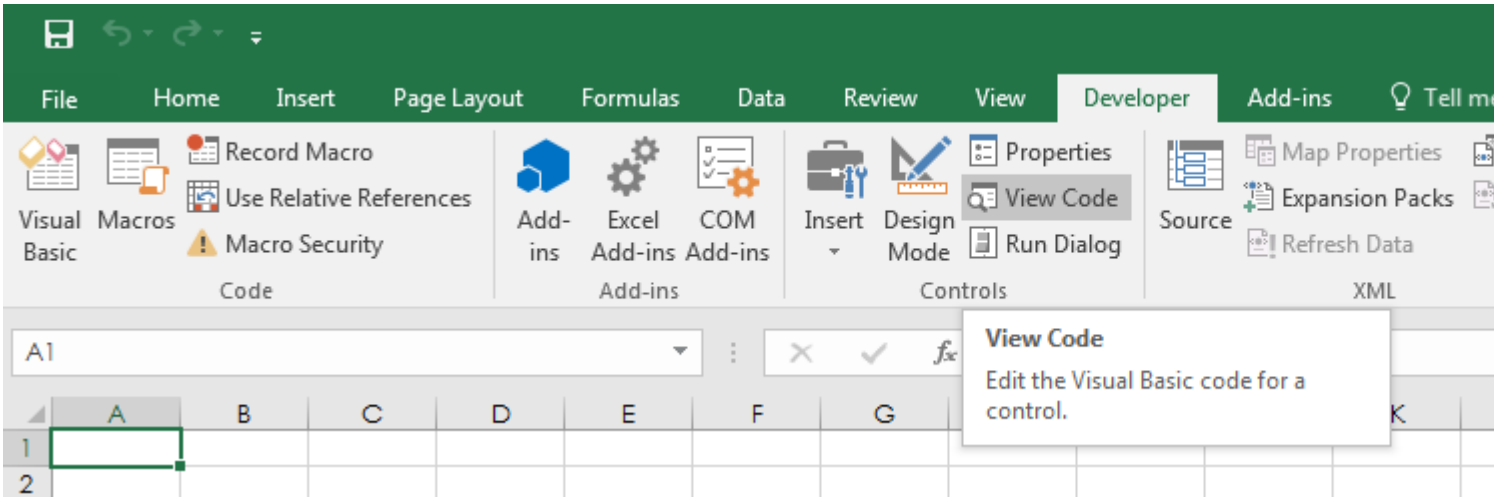
Microsoft OfficeでのVisual Basic Editorへのアクセス

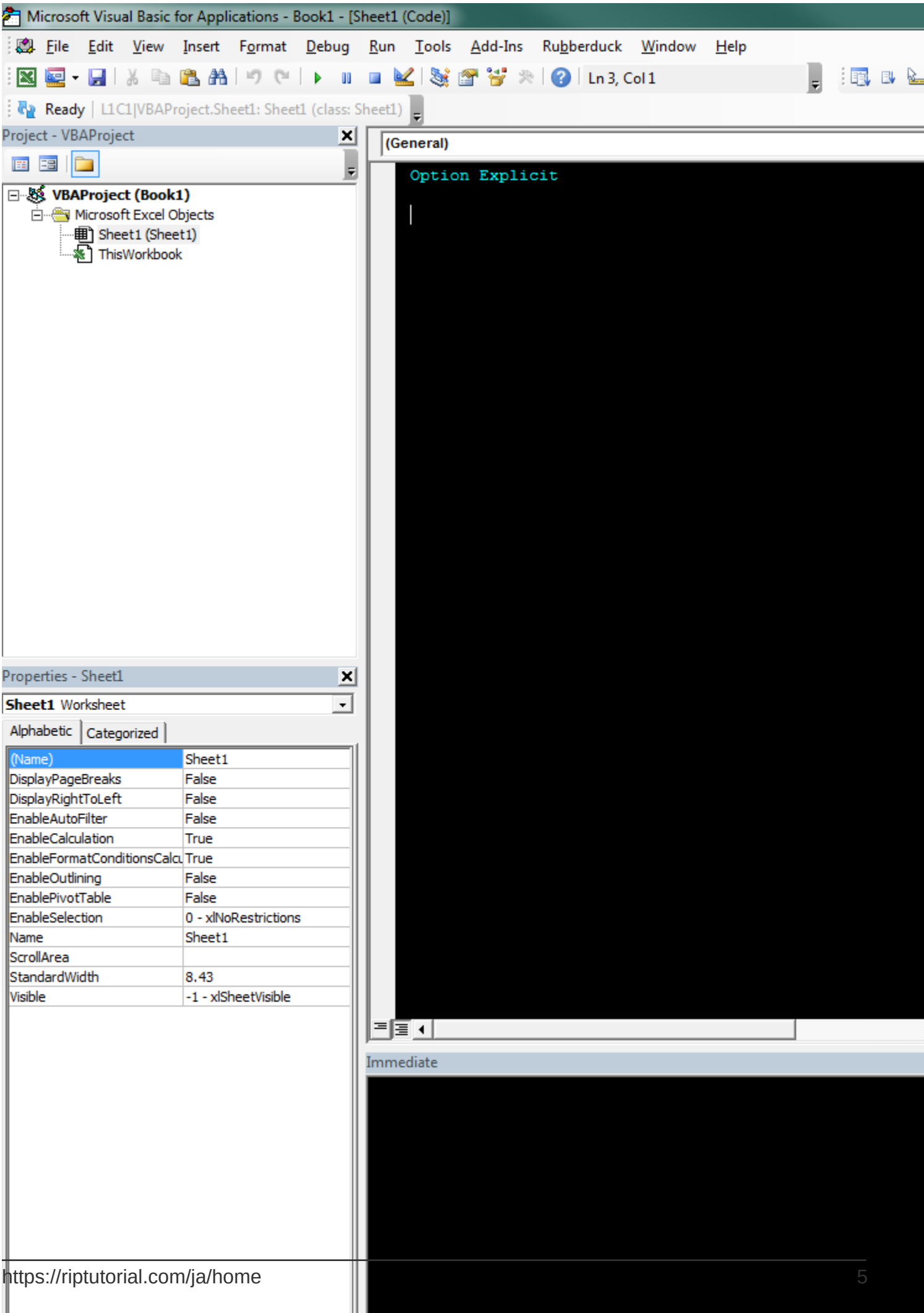
`Alt` + `F11`をすか、タブにき、「Visual Basic」ボタンをクリックすると、Microsoft Officeアプリケーションのいずれかでエディタをくことができます。リボンに[]タブがされないは、これがになっているかどうかをします。

デフォルトでは、タブはになっています。 「」タブをにするには、「ファイル」->「オプション」にし、のリストで「リボンのカスタマイズ」をします。の "リボンのカスタマイズ"ツリービューでツリーをつけて、チェックボックスのチェックをオンにします。 [OK]をクリックして[オプション]ダイアログボックスをじます。



タブがリボンにされ、 "Visual Basic"をクリックしてVisual Basic Editorをくことができます。あるいは、「コードをる」をクリックすると、アクティブなのコードペイン、たとえばWorkSheet、Chart、Shapeをすることができます。


File
Home
Insert
Page Layout
Formulas
Data
Review
View
Developer
Add-ins
Visual Basic
Macros
Record Macro
Use Relative References
Macro Security
Code
Add-ins
Excel Add-ins
COM Add-ins
Add-ins
Insert
Design Mode
Properties
View Code
Run Dialog
Controls
Source
Map Properties
Expansion Packs
Refresh Data
XML
A1
View Code
Edit the Visual Basic code for a control.
A
B
C
D
E
F
G
1
2


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

は、のなしにをし、テキストをし、をし、し、することができます。

のモジュールとHello World

まずコーディングをするには、のリストでVBAプロジェクトをクリックし、しいモジュールをするがあります。の*Hello World*コードはのようになります。

```
Sub HelloWorld()
    MsgBox "Hello, World!"
End Sub
```

テストするには、ツールバーの*Play*ボタンをすか、に`F5`キーをします。おめでとうめてのVBAモジュールをしました。

デバッグ

デバッグはになであり、ってしているまたはしていないコードをしくてします。

コードをにする

デバッグにまずうべきことは、のでコードをしてから、それを1ずつして、それがどおりにこっているかどうかをすることです。

- ブレークポイント `F9` 、デバッグ - ブレークポイントのりえブレークポイントは、されたではなくにしたり、がしたポイントにしたり、ユーザーにをえたりすることができます。
- に`Stop`キーワードをして、にコードをそのにさせることもできます。これは、たとえば、 `F9` でブレークポイントをできないのに
- ステップイン `F8` 、デバッグ - ステップインユーザのサブ/のびしであれば、1のコードしかしません。
- ステップオーバー `Shift` + `F8` 、デバッグ - ステップオーバー1のコードをし、ユーザーのサブ/をしません。
- ステップアウト `Ctrl` + `Shift` + `F8` 、デバッグ - ステップアウトのサブ/ファンクションをしますまでコードをします。
- カーソルまで `Ctrl` + `F8` 、デバッグ - カーソルまでカーソルでにするまでコードをします。
- `Debug.Print`をすると、にイミディエイトウィンドウにをできます。 `Debug.?`することもでき `Debug.?` `Debug.Print`ショートカットとして

ウォッチウィンドウ

ごとにコードをすることは、のステップであり、をるがあり、そのためのツールの1つがウォッチウィンドウビュー - ウォッチウィンドウです。ここでは、されたのをることができます。ウォッチウィンドウにをするには、のいずれかをいます。

- それをクリックし、「ウォッチを」をします。
- ウォッチウィンドウをクリックし、[ウォッチを]をします。
- デバッグ - をします。

しいをするときには、そのをたいだけでなく、であるかのにコードをするかをできます。

イミディエイトウィンドウ

イミディエイトウィンドウでは、 `Print`キーワードか1つの "`?`"をにけて、のコードをしたりアイテムをしたりできます。

いくつかの

- `? ActiveSheet.Name` - アクティブなシートのをします。
- `Print ActiveSheet.Name` - アクティブなシートのをします。
- `? foo` -のをします`foo` *
- `x = 10`は`x`を`x = 10`する*

*イミディエイトウィンドウでのを/することは、にのみうことができます

ベストプラクティスのデバッグ

あなたのコードがどおりにしないときは、にすべきことはいをしてにもうむことです。

それでもがしないは、デバッグをしてください。いでは、でするほうがですが、いのはされるにブレークポイントやブレークをするがありますが、ここでのはがりにしないことをすることです。

ったをすがあってもそのがはっきりしていないは、のがっているかどうかをするのにつをしたり、をにきえたりしてみてください。

それでもできないは、をしてください。

- あなたのをするためにできるだけコードのをめる
- がのとしていないは、できえます。 したがって、 `Sheets(a*b*c+d^2).Range(addressOfRange)` `Sheets(4).Range("A2")`
- どのがったるいをしているのか、それはかエラー、った...

オンラインでVBAをいめるをむ https://riptutorial.com/ja/vba/topic/802/vbaをいめる

2: ADOの

このトピックにされているは、にするためにバインディングをしており、Microsoft ActiveX Data Object xx Libraryへのがです。にされたを`Object`にきえ、にじて`CreateObject`で`New`をしてオブジェクトのをきえることで、レイトバインドにできます。

Examples

データソースへのの

ADOをしてデータソースにアクセスするためののは、ADO `Connection`オブジェクトをすることです。 DSN、ユーザーID、およびパスワードを`.Open`メソッドにしてDSNをくこともですが、これは、をしてデータソースパラメータをしています。

DSNはADOでデータソースにするはないことにしてください。ODBCプロバイダをつデータソースは、なでできます。なるプロバイダののはこのトピックのですが、 ConnectionStrings.comはプロバイダにしたをつけるためのれたリファレンスです。

```
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
```

データベースパスワードは、わかりやすくするために、ののにまれています。ベストプラクティスでは、データベースのパスワードをコードにしないようにします。これは、ユーザーまたはWindowsをしてパスワードをすることでできます。

クエリをしたレコードの

クエリは2つのでできる、どちらもされたのコレクションであるADO `Recordset`オブジェクトをします。ののでは、にするために、データソースへののをする `OpenDatabaseConnection`をしています。データソースにされるSQLのはプロバイダーであることにしてください。

のは、SQLをConnectionオブジェクトにすことで、なクエリをするもなです。

```
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
```

2のは、するクエリのADO `Command`オブジェクトをすることです。これにはもうしコードがですが、パラメータされたクエリをするためにはです。

```
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
```

データソースにされるコマンドは、またはでないSQLインジェクションにしてであることにしてください。に、あらゆるのユーザーをすることによってをすべきではありません。わりに、それらをパラメータするがあります「パラメータされたコマンドの」を。

スカラーの

ADOをすると、プロバイダがSQLをしてサポートするほとんどすべてのデータベースをできます。この、 `Execute`によってされた`Recordset`はずしもするはありませんが、@@ Identityまたはの SQLコマンドをしてINSERTをしたにキーりてをするとです。のでは、にするためにデータソースへののをする `OpenDatabaseConnection`をしています 。

```
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
```

データソースにされるコマンドは、またはでないSQLインジェクションにしてであることにして

ください。に、SQLはののユーザをしてするべきではありません。わりに、それらをパラメータするがあります「パラメータされたコマンドの」を。

パラメータされたコマンドの

ADOをしてされるSQLにユーザーがまれるがあるは、SQLインジェクションのをにえるために、パラメーターをパラメーターすることをおめします。このメソッドは、いよりもみやすく、よりでなコードをにしますつまり、 `Parameter`をすをすることによって。

ODBCでは、パラメータはされてい`?`クエリテキストに「プレースホルダ」をし、クエリにされるとじでパラメータに`Command`をします。

のでは、にするためにデータソースへのをする `OpenDatabaseConnection`をしています 。

```
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
```

のでは、パラメータUPDATEをしていますが、のSQLにパラメータをできます。

オンラインでADOのをむ https://riptutorial.com/ja/vba/topic/3578/adoの

3: APIコール

き

APIはApplication Programming Interface

VBAのAPIは、オペレーティングシステムとできるのメソッドをします

システムコールは、DLLファイルでされたプロシージャをすることによってうことができます。

なオペレーティングのライブラリファイルDLL

ダイナミックリンクライブラリ	
Advapi32.dll	くのセキュリティとレジストリびしをむAPIのなサービスライブラリ
Comdlg32.dll	ダイアログAPIライブラリ
Gdi32.dll	グラフィックデバイスインタフェースAPIライブラリ
Kernel32.dll	コアWindows 32ビットベースAPIのサポート
Lz32.dll	32ビットルーチン
Mpr.dll	のプロバイダのルータライブラリ
Netapi32.dll	32ビットネットワークAPIライブラリ
Shell32.dll	32ビットシェルAPIライブラリ
User32.dll	ユーザーインターフェイスルーチンのライブラリ
Version.dll	バージョンライブラリ
Winmm.dll	Windowsマルチメディアライブラリ
Winspool.drv	スプーラーAPIびしをむスプーラーインターフェイスをする

64システムでされるしい

タイプ		
	PtrSafe	Declareステートメントが64ビットとがあることをします。このは、64ビットシステムでです
データ・タイプ	LongPtr	32ビットバージョンでは4バイトのデータ、Office 2010の64ビットバージョンでは8バイトのデータのデータです。これはしいコードのポインタまたはハンドルをするためのですが、 64ビットバージョンのOffice 2010でするがあるはレガシーコードです.32ビットと64ビットのVBA 7ランタイムでのみサポートされています。はできますが、はできないことにしてください
データ・タイプ	LongLong	これは、64ビットバージョンのOffice 2010でのみできる8バイトのデータです。はりてできませんりてをけるため
	オペレーター	CLngPtrをLongPtrデータにします。
	オペレーター	CLngLngをLongLongデータにします。
	VarPtr	バリアントコンバータ。 64ビットバージョンでLongPtrをし、32ビット4バイトでLongをします。
	ObjPtr	オブジェクトコンバータ。 64ビットバージョンでLongPtrをし、32ビット4バイトでLongをします。
	StrPtr	。 64ビットバージョンでLongPtrをし、32ビット4バイトでLongをします。

コールサインのな

- Visual Basic 5.0のWin32api32.txt いAPI、にレビューされた20053、Microsoft
- 64ビットサポートのWin32API_PtrSafe Office 2010、Microsoft

Examples

APIのと

なるVBAバージョンでするようにDLLプロシージャをする

```
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
```

のは、Kernel32.dllファイルでされた "Sleep"をびすをVBAにえます

Win64およびWin32は、 きコンパイルにされるあらかじめされたです

あらかじめされた

いくつかのコンパイルは、あらかじめされています。どちらがするかは、VBAをしているオフィスバージョンのビットによってなります.Vba7は、Office 2010とに64ビットバージョンのOfficeをサポートするためにされました。

	16ビット	32ビット	64ビット
Vba6		Vba6	
Vba7		Vba7の	
Win16			
Win32			
Win64			
マック		Macの	Macの

これらのは、WindowsではなくOfficeをします。たとえば、OSが64ビットのWindowsであっても、32ビットOfficeではWin32 = TRUEとなります。

APIをするときのないは、しいパラメータをした32ビットと64ビットのOfficeバージョンですは「」を

ノート

- は、モジュールのにされ、サブモジュールまたはのにされます
- モジュールでされたきは、デフォルトでpublicです

- モジュールのプライベート·プロシージャをするには、のに`Private`キーワードをけます
- のタイプのモジュールでされているDLLプロシージャは、そのモジュールです

Sleep APIびしのな

```
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
```

VBAラッパーからシステムファンクションににアクセスできるようにするのAPIモジュールをすることをおめします。のVBA Subs、またはライブラリでされるパラメータなどののシステムコールになをカプセルするファンクション

モジュールには、とをすべてめることができます。

- メソッドシグネチャとなデータ
- のをし、すべてのパラメータがどおりにされるようにするラッパー

DLLプロシージャをするには、`Declare`ステートメントをコードウィンドウのDeclarationsセクションにします。

プロシージャがをすは、 としてします。

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

プロシージャがをさないは、 **Sub**としてします。

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

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

  また、APIへのなびしが**Excel**をクラッシュさせ 、おそらくデータファイルをさせることにしてください

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

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
```

のWindows API - モジュール1と2では、Win64とWin32のなをむAPIモジュールをしています

Windows API - モジュール1 of 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
```

2のにく

Windows API - モジュール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

マイクロソフトはにAPIをサポートしていませんが、いくつかのではよりくのがオンラインでられる

MacOffice 2016はサンドボックスされています

VBAをサポートするのバージョンのOfficeアプリケーションとはなり、Macアプリケーションの Office 2016はサンドボックスされています。

サンドボックスでは、アプリケーションがアプリケーションコンテナのリソースにアクセスすることをします。これは、ファイルアクセスやプロセスのにするアドインやマクロにします。のセクションでするしいコマンドをして、サンドボックスのをにえることができます。 Office 2016 for MacのしいVBAコマンド

のVBAコマンドは、Office 2016 for Macにのです。

コマンド	に
GrantAccessToMultipleFiles	にのファイルにアクセスするユーザーのをする
AppleScriptTask	VBからAppleScriptスクリプトをびす
MAC_OFFICE_VERSION	コンパイルになるMacのバージョンのIFDEF

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
```

モニタとのをする

```
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
```

FTPおよびAPI

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
```

オンラインでAPIコールをむ https://riptutorial.com/ja/vba/topic/10569/apiコール

4: CreateObjectとGetObject

CreateObjectはオブジェクトのインスタンスをするのがもGetObjectが、 GetObjectはオブジェクトののインスタンスをします。オブジェクトのまたはののは、インスタンスプロパティによってなります 。のオブジェクトはSingleUseWMIで、すでにするはできません。のオブジェクトExcelなどはMultiUseであり、のインスタンスをにできます。オブジェクトのインスタンスがせずGetObjectをしようとすると、のトラップなメッセージがされます。 Run-time error '429': ActiveX component can't create object 。

GetObjectには、の2つのオプションパラメータのなくとも1つがです。

1. *Pathname* - VariantStringオブジェクトをむファイルのフルパスfilenameをむ。このパラメータはオプションですが、 *Pathname*がされているは*Class*がです。
2. *Class* - VariantStringオブジェクトのなApplicationおよびObjectTypeをす。 *Pathname*をすると、 クラスがです。

CreateObjectには、1つのパラメータと1つのオプションパラメータがあります。

1. *Class* - VariantStringオブジェクトのなApplicationおよびObjectTypeをす。 クラスはパラメータです。
2. *Servername* - VariantStringオブジェクトがされるリモートコンピュータの。した、オブジェクトはローカルマシンにされます。

クラスはにApplication.ObjectTypeので2つのでされていApplication.ObjectType 。

1. アプリケーション - オブジェクトがまれているアプリケーションの。 |
2. *Object Type* - されるオブジェクトのタイプ 。 |

いくつかのクラスのはのとおりです。

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

Examples

GetObjectとCreateObjectのデモンストレーション

MSDN-GetObject

> ActiveXコンポーネントによってされるオブジェクトへのをします。
>
> オブジェクトののインスタンスがある、またはにロードされたファイルをしてオブジ

エクトをするは、GetObjectをします。のインスタンスがなく、ファイルをロードしてオブジェクトをしたくないは、CreateObjectをします。

```
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
```

オンラインでCreateObjectとGetObjectをむ https://riptutorial.com/ja/vba/topic/7729/createobjectとgetobject

5: FileSystemObjectをせずにファイルとディレクトリをする

Scripting.FileSystemObjectは、このトピックのレガシーメソッドよりはるかにです。ほとんどすべてのにましいはずです。

Examples

フォルダとファイルがするかどうかの

ファイル

ファイルがするかどうかをするには、ファイルをDir$にし、がされるかどうかをテストします。ことにしてくださいDir$ のファイルをテストするために、されたので、ワイルドカードをサポートしてpathName 、それはらがまれていないことをするためにテストするがあります。のサンプルはエラーをさせます。これがましいでない、をしてにFalseすことができます。

```
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
```

フォルダ**Dir $**メソッド

Dir$()をして、オプションattributesパラメーターにすvbDirectoryをして、フォルダーがするかどうかをすることもできます。この、されたpathNameはパスのり \ でわらなければなりません。するファイルはをきこすためです。ワイルドカードはのパスりのにのみできるので、にワイルドカードがまれていると、のサンプルはランタイムエラー52をスローします。これがのではないは、のにある「 On Error Resume Nextコメントをします。また、 Dir$はファイルパスつまり..\Foo\Barをサポートしているので、のディレクトリがされていないり、はであることがされます。

```
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
```

フォルダ**ChDir**メソッド

ChDirステートメントは、フォルダがするかどうかをテストするためにもできます。このメソッドはVBAがされているをにするため、するはわりにDir$メソッドをするがあります。そのパラメータをあまりにしないというがあります。このメソッドはファイルパスもサポートしているので、Dir$メソッドとじがあります。

```
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
```

ファイルフォルダのと

にするため、のでは、このトピックの「フォルダとファイルがするかどうかの」の FolderExists をしています。

MkDirステートメントをすると、しいフォルダをできます。ドライブ C:\Foo 、UNC \\Server\Foo 、パス ..\Foo 、またはのディレクトリ Foo をむパスをけれます。

ドライブまたはUNCがされたつまり\Foo 、のドライブにフォルダがされます。これは、のディレクトリとじドライブであっても、そうでなくてもかまいません。

```
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ステートメントをすると、のフォルダをできます。 MkDirとじのパスをけれ、のディレクトリとドライブとじをします。このステートメントはWindowsのrdシエルコマンドとているので、エラー75がスローされます。ターゲットディレクトリがでないは、「パス/ファイルアクセスエラー」です。

```
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
```

オンラインでFileSystemObjectをせずにファイルとディレクトリをするをむ
https://riptutorial.com/ja/vba/topic/5706/filesystemobjectをせずにファイルとディレクトリをする

6: Scripting.Dictionaryオブジェクト

Scripting Dictionaryオブジェクトのバインディングをするには、VBEのTools→Referencesコマンドをしてプロジェクトに Microsoft Scripting Runtimeをするがあります。このライブラリはプロジェクトとともにばれます。 VBAプロジェクトがされてのコンピュータでされているときにするはありません。

Examples

プロパティとメソッド

Scripting Dictionaryオブジェクトは、キー/アイテムのペアにをします。キーはではなくユニークでなければなりませんが、けられたアイテムはりすことができますはコンパニオンキーによってされます。のタイプのバリアントまたはオブジェクトであることができます。

ディクショナリは、の「フィールド」キー にプライマリユニークインデックスをつ2フィールドのインメモリデータベースとえることができます。 Keysプロパティのこののインデックスは、にな「ルックアップ」がキーのするItemをすることをにします。

プロパティ

	みき	タイプ	
CompareMode	みき	CompareMode	CompareModeのは、のにしてのみできます。けられるは、0vbBinaryCompare、1vbTextCompare、2 vbDatabaseCompareです。
カウント	みり	なし	スクリプティングディクショナリオブジェクトのキー/アイテムペアの1ベースのカウント。
キー	みき	の	のののキー。
キー	みき	の	デフォルトプロパティ。のキーにけられたの。ディクショナリにしないキーをつアイテムをしようとすると 、されたキーがにされます。

メソッド

キー 、	しいキーとアイテムをにします。しいキーは、ののKeysコレクションにしては

アイテム	いけませんが、くのユニ―クキ―のでアイテムをりすことができます。
キ―	キ―がにすでにするかどうかをするブ―ルテスト。
キ―	ユニ―クキ―のまたはコレクションをします。
アイテム	するのまたはコレクションをします。
キ―	々のキ―とそのアイテムをします。
すべてする	オブジェクトのすべてのキ―とアイテムをします。

サンプルコ―ド

```
'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、でデ―タをする

は、のエントリがするをするのにですが、のまたはの、または、、など、エントリの1つのにのみします。

かがブックをするたびに、ユ―ザ―とをするスクリプトをして、ユ―ザ―アクティビティのログをするブックをしてください。

Logワ―クシ―ト

A	B
ボブ	10/12/2016 9:00
アリス	10/13/2016 13:00
ボブ	2013101313:30
アリス	10/13/2016 14:00
アリス	10/14/2016 13:00

`Summary`というのワークシートにユーザーのをしたいとします。

ノート
1.データは`ActiveWorkbook`にあるとみなされ`ActiveWorkbook`。
2.をしてワークシートからをしています。これはセルのよりもです。
3. `Dictionary`はバインディングをしてされます。

```
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
```

はのようになります。

`Summary`ワークシート

A	B
ボブ	2013101313:30

A	B
アリス	10/14/2016 13:00

、ユーザーがブックをしたをしたい、 `For`ループのはのようになります。

```
        '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
```

はのようになります。

`Summary`ワークシート

A	B
ボブ	2
アリス	3

Scripting.Dictionaryでユニークなをする

`Dictionary`は、のセットをににできます。のをえてみましょう

```
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
```

あなたはこれをのようにびすことができます

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

`uniqueVals`は`{1,2,3}`のみがまれます。

このは、のなオブジェクトでできます。

オンラインでScripting.Dictionaryオブジェクトをむ
https://riptutorial.com/ja/vba/topic/3667/scripting-dictionaryオブジェクト

7: Scripting.FileSystemObject

Examples

FileSystemObjectの

```
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をしたテキストファイルのみ

```
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をしてテキストファイルをする

```
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をしてのファイルにきむ

```
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をしてディレクトリのファイルをする

バインドMicrosoft Scripting Runtimeへのが

```
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
```

フォルダとファイルをにする

Early Bound `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
```

をしてびすとされます `EnumerateFilesAndFolders "C:\Test"`

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

のようなをしてびすとされます `EnumerateFilesAndFolders "C:\Test", 0`

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

のようなをしてびすとされます `EnumerateFilesAndFolders "C:\Test", 1, 4`

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

ファイルからファイルをりく

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

`MyFile.something`し`MyFile.something`

`GetBaseName()`メソッドは、すでにファイルにのピリオドをしていることにしてください。

ファイルからをする

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

Prints `txt GetExtensionName()`メソッドは、すでにファイルにのピリオドをして`GetExtensionName()`ことにしてください。

ファイルパスからのパスのみをする

GetParentFolderNameメソッドは、のパスのフォルダをします。これはフォルダとすることもできますが、パスのパスからパスをするがです

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

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

のパスりは、されるにはまれないことにしてください。

FSO.BuildPathをしてフォルダパスとファイルからフルパスをする

フォルダーパスのユーザーをけれるは、ファイルパスをするにバックスラッシュ \ をするがあります。 `FSO.BuildPath`メソッドをすると、これがになります。

```
  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
```

オンラインでScripting.FileSystemObjectをむ https://riptutorial.com/ja/vba/topic/990/scripting-filesystemobject

8: VBAオプションキーワード

- オプションoptionName []
- Option Explicit
- オプション{テキスト|バイナリ|データベース}
- オプションプライベートモジュール
- オプションベース{0 | 1}

パラメータ

オプション	
	それがされたモジュールでをとする にはそれらのすべて。このオプションをすると、されていない/ってられたをするとコンパイルエラーになります。
テキストをする	システムのロケールについて、モジュールのををしないようにし、アルファベットのをします "a" = "A"。
バイナリの	デフォルトのモード。のバイナリ/ASCIIなどをしてをして、モジュールのをとをするようにします。
データベースの	MS-Accessのみモジュールのをを SQLでのとじようにさせます。
プライベートモジュール	モジュールの`Public`メンバが、モジュールがするプロジェクトのからアクセスされないようにします。つまり、ホストアプリケーションからプロシージャをにすつまり、マクロやユーザーとしてすることはできません。
オプションベース 0	。モジュールののののを`0`にします。ななしでされたは、 `0`がされます。
オプションベース 1	なのを`1`にします。ななしでがされると、 `1`がされます。

コンパイラが`Option Base {0|1}`をフォールバックするのではなく、にをすることで、のをするがはるかにです。これはそうすることができます

```
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)
```

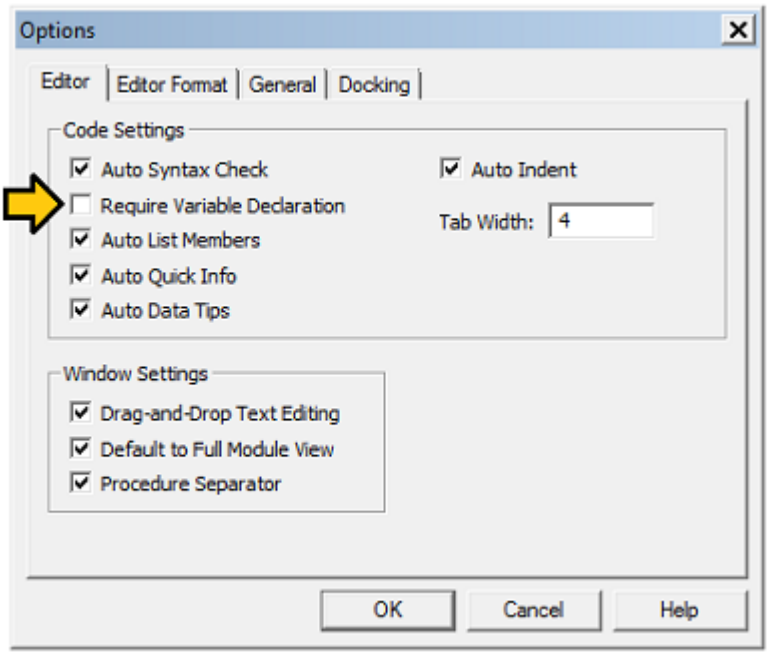
Examples

Option Explicit

VBAでに`Option Explicit`をすると、にすべてのをすることがベストプラクティスとみなされます。これには、されたやIntelliSenseのなどののもあります。

```
Option Explicit

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

VBEのツールでをとする▸オプション▸エディタのプロパティページでは、 **Option Explicit**ステートメントがしくされたコードシートのにされます。



これは、スペルミスのようなったコードミスをけるだけでなく、でしいをするようにします。 いくつかのは、 に「Option Explicit」をしています。

オプション{バイナリ|テキスト|データベース}

オプションバイナリ

バイナリは、モジュール/クラスのとをしてのをチェックします。には、このオプションをすると、のバイナリのソートをしてのがされます。

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

モジュールにOption Compareがされていないは、デフォルトでバイナリがされます。

```
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
```

オプションテキスト

Option Compare Textは、モジュール/クラスのすべてのでとをしないをいます。

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
```

オプションデータベース

オプションデータベースのは、MS Accessでのみです。テキスト/バイナリモードのどちらをするかをするために、のデータベースをするようにモジュール/クラスをします。

このは、データベースの*SQL*とじでテキストをするカスタム*Access UDF*ユーザーのにモジュールをしないり、こののはおめしません。

オプションベース{0 | 1}

`Option Base`は、 のデフォルトのをするためにされます。モジュールレベルでされ、のモジュールにしてのみです。

デフォルトではしたがってオプションベースがされていない、ベースは0です。つまり、モジュールでされたののののインデックスは0です。

`Option Base 1`がされている、のはインデックス1

ベース0の

```
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
```

ベース1とじ

```
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
```

2のでは、のインデックス0にアクセスしようとしたためにサブスクリプトがのループステージでになりましたエラー9。モジュールが`Base 1`されているためこのインデックスはしません

ベース1のしいコードはのとおりです。

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

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

    Next i
```

Splitは、 `Option Base`になく、 に 0からまるインデックスをつをすることにしてください。 スプリットのいのは、 ここでつけることができます

されたのをむ0からまる1をします。

Excelでは、セルの`Range.Value`および`Range.Formula`プロパティは、 に 1ベースの2D Variantをします。

に、ADOでは、 `Recordset.GetRows`メソッドはに 1ベースの2Dをします。

つのおめ「ベスト·プラクティスは、」にすることですLBOUNDとUBoundのをするためのを。

```
'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
```

`Option Base 1`は、がされるすべてのコードモジュールのになければなりません。

オンラインでVBAオプションキーワードをむ https://riptutorial.com/ja/vba/topic/3992/vbaオプションキーワード

9: VBAランタイムエラー

き

コンパイルされたコードは、にエラーにることがあります。このトピックでは、もなもの、その、およびそれらをするについてします。

Examples

エラー '3' GoSubなしのり

なコード

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

なぜこれはしませんか

は`DoSomething`プロシージャにり、 `DoThis`ラベルにジャンプし、 "Hi"をします。デバッグには 、`GoSub`びしのににり 、「Hi」をします。また、 `Return`ステートメントにしましたが、 `GoSub`ステートメントでここになかったので、ここにることはありません。

しいコード

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

なぜこれはしますか

することで`Exit Sub` のにを`DoThis`ラベル、々はしている`DoThis`のりのからサブルーチンを-するの`DoThis`サブルーチンをする`GoSub`ジャンプ。

そのの

GoSub / Returnはされていませんので、のプロシーージャコールのためにけてください。プロシーージャには、エラーハンドラのサブルーチンをめることはできません。

これは、 エラー '20'ににています 。 エラーなしでします 。どちらのにおいても、 のパスがなジャンプなしにサブルーチンにることができないようにすることです On Error GoToはジャンプとみなされます。

エラー '6' オーバーフロー

コードがしくありません

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

なぜこれはしませんか

Integerデータは、32,767のをつ16ビットきです。それのものにりてるとがオーバーフローし 、このエラーがします。

しいコード

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

なぜこれはしますか

わりにLong 32ビットをすることで、カウンタのをオーバーフローさせることなく、32,767するループをできるようになりました。

そのの

は、「 データと 」をしてください。

エラー '9'のき

コードがしくありません

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

なぜこれはしませんか

`foo`は10のをむです。 `i`ルーープカウンタが11のにすると、 `foo(i)`はです。このエラーは、そのまたはコレクションにしないインデックスでまたはコレクションにアクセスするたびにします。

しいコード

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

なぜこれはしますか

`LBound`と`UBound`はそれぞれ、アレイのおよびをするためにします。

そのの

インデックスが、たとえば`ThisWorkbook.Worksheets("I don't exist")`、このエラーはされたがされたコレクションにしないことをします。

のエラーはのものですが、 `Collection`は、エラー5 "なプロシージャびしまたは"わりにします。

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

エラー '13'の

コードがしくありません

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

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

なぜこれはしませんか

VBAはに"42?"をしようとしてい"42?"をDateにします。した、VBAはすをらないため、DoSomethingElseのびしをできません。したがって、のがされるとしないため、エラー13 のがしますのうちにされることもありません。

しいコード

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

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

なぜこれはしますか

Dateパラメータをとするプロシージャにをすことで、びしはします。

エラー '91'オブジェクトまたはWithブロックがされていません

コードがしくありません

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

なぜこれはしませんか

オブジェクトがをし、がしてするがSetたキーワードを。このエラーは、がNothingであるオブジェクトにしてメンバびしがわれるたびにします。この、 fooはCollectionですが、されていNothingため、にはNothingがまれています.Addには.AddをびすことはできませNothing。

しいコード

```
Sub DoSomething()
    Dim foo As Collection
```

```
    Set foo = New Collection
    With foo
        .Add "ABC"
        .Add "XYZ"
    End With
End Sub
```

なぜこれはしますか

Setキーワードをしてオブジェクトになをりてることにより、 .Addびしがします。

そのの

くの、またはプロパティはオブジェクトをすことができます。なはExcelのRange.Findメソッドで、これはRangeオブジェクトをします。

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

しかし、はNothing がつからないをすことができるため、された.Rowメンバびしがするがあります。

オブジェクトメンバーをびすに、がIf Not xxxx Is Nothingでされていることをします。

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

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

エラー '20'エラーなし

コードがしくありません

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

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

なぜこれはしませんか

DoSomethingElseプロシージャがエラーをさせると、はCleanFailラベルにジャンプし、エラーをし、 Resume Nextの Resume Nextはエラーがしたののにジャンプします。この、 Debug.Printエラーサブルーチンはエラーコンテキストなしでされており、 Resume Nextの Resume Nextにすると、するがな

いためエラー20がします。

しいコード

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

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

なぜこれはしますか

することで`Exit Sub`のにを`CleanFail`ラベル、々はしている`CleanFail`のりのからエラーサブルーチンを-エラーサブルーチンをするためののは、で`On Error`ジャンプ。したがって、パスは、エラーコンテキストの`Resume`にはせず、エラー20をします。

そのの

これは、 エラー '3'とによくています 。どちらのにおいても、 のパスがなジャンプなしにサブルーチンにることができないようにすることです `On Error GoTo`はジャンプとみなされます。

オンラインでVBAランタイムエラーをむ https://riptutorial.com/ja/vba/topic/8917/vbaランタイムエラー

10: イベント

- ソースモジュール `[Public] Event [identifier]([argument_list])`
- ハンドラモジュール `Dim|Private|Public WithEvents [identifier] As [type]`
- イベントは`Public`のみです。クラスモジュールのメンバーイベントをむは、デフォルトでは に`Public`ので、はオプションです。
- `WithEvents`は、 `Private`でも`Public`でも`WithEvents`ませんが、 `Friend`はありません。 `WithEvents` はをするキーワードではなく、のキーワードのであるため、はです。したがって、アクセスがしないは、 `Dim`キーワードをするがあります。

Examples

ソースとハンドラ

イベントとはですか

VBAはイベントドリブンです。ホストアプリケーションまたはホストによってされたイベントにしてVBAコードがされます。イベントのはVBAをするでです。

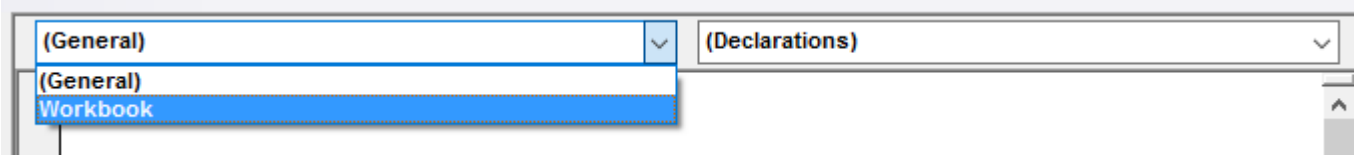
APIはくの、さまざまなにしていくつかのイベントをさせるオブジェクトをします。たとえば、`Excel.Application`オブジェクトは、しいワークブックが、オープン、アクティブ、またはクローズされるたびにイベントをさせます。またはワークシートがされるたびに。またはファイルがされる。または。フォームのボタンは、ユーザーがクリックすると`Click`イベントをさせ、ユーザーフォームはアクティブされたのイベントとじられるのイベントをさせます。

APIのからは、イベントはポイントです。クライアントコードでは、これらのイベントをするコードをし、これらのイベントがするたびにカスタムコードをすることができます。つまり、ワークシートのがされるたびにカスタムコードをにできます。 - ワークシートでがされたときにするイベントをします。

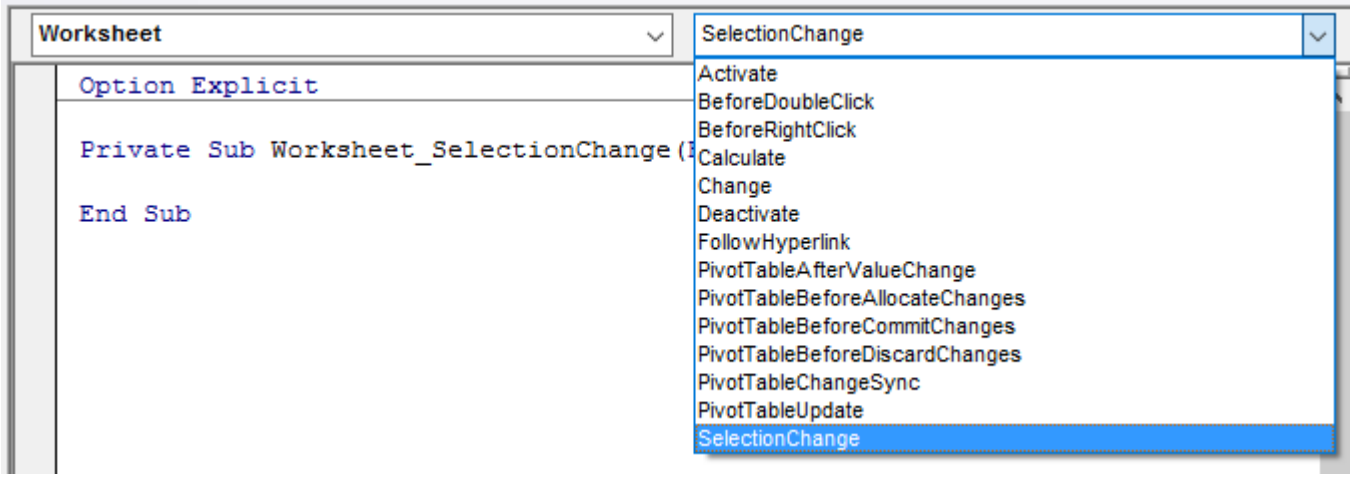
イベントをするオブジェクトはイベントソースです。イベントをするメソッドはハンドラです。

ハンドラー

VBAドキュメントモジュール `ThisDocument` 、 `ThisWorkbook` 、 `Sheet1`などと`UserForm`モジュールは、いくつかのイベントをするなインターフェイスをするクラスモジュールです。これらのインターフェイスは、コードペインのにあるのドロップダウンでできます。



のドロップダウンでは、のドロップダウンでされたインターフェイスのメンバーがされます。



アイテムがのリストでされると、VBEはにイベントハンドラスタブをします。ハンドラがするは、そこをナビゲートします。

Moduleスコープの`WithEvents`は、のモジュールでできます。

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

それぞれの`WithEvents`は、のドロップダウンからできるようになります。のドロップダウンでイベントがされると、VBEは`WithEvents`オブジェクトのとイベントをアンダースコアでしたイベントハンドラスタブをします。

```
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
```

なくとも1つのイベントをするだけが`WithEvents`でき、 `WithEvents`には`New`キーワードでそのでをりてることができません。このコードはです

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

オブジェクトは`Set`に`Set`があります。クラスモジュールでは、 `Class_Initialize`ハンドラので、そのクラスのインスタンスがするり、そのオブジェクトのイベントをクラスがするため、そのため

のいがよくあります。

ソース

のクラスモジュールまたはドキュメントモジュール、またはユーザーフォームは、イベントソースになります。 `Event`キーワードをして、 `Event`のシグネチャをモジュールのセクションでします。

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

イベントのによって、イベントのと、イベントハンドラのがまります。

イベントは、されているクラスでのみさせることができます。クライアントコードではイベントをできます。イベントは`RaiseEvent`キーワードでされます。イベントのはそのでされます

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

`SomethingHappened`イベントをするコードがなければ、 `DoSomething`プロシージャをするとイベントはしますが、もこりません。イベントソースが`Something`というのクラスののコードであるとすると、 `ThisWorkbook`このコードは、 `test.DoSomething`がびされるたびに "hello"というメッセージボックスをします。

```
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
```

データをイベントソースにす

しのパラメータの

イベントは、びしにされることをした`ByRef`パラメータをすることができます。

```
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
```

BeforeSomethingイベントは、そのハンドラっcancelにパラメータをTrue 、はハンドラからったときに、 cancelとなりTrueとAfterSomethingされることはありません。

```
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
```

foo.DoSomethingがされるとき、 fooオブジェクトがどこかにりてられているとすると、メッセージボックスはキャンセルするかどうかをfoo.DoSomethingメッセージをし、 Noがされたときだけ2のメッセージボックスに "did not cancel"というメッセージがされます。

オブジェクトの

また、オブジェクトByValコピーをし、ハンドラにそのオブジェクトのプロパティをさせることもできます。びしはされたプロパティをみり、それにじてすることができます。

```
'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とみわせると、これはびしにをすためにむしろらかでないをするためにできます

```
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
```

ハンドラはのようになります。

```
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
```

オンラインでイベントをむ https://riptutorial.com/ja/vba/topic/5278/イベント

11: インターフェイス

き

インターフェイスは、クラスがするのるいをするです。インターフェースのは、メソッドのシグニチャー、パラメーター、りタイプのリストです。すべてのメソッドをつクラスは、そのインターフェイスを「する」とわれています。

VBAでは、インターフェイスをすることで、モジュールがすべてのメソッドをしていることをコンパイラができます。またはパラメータは、のクラスのわりにインタフェースのからすることができます。

Examples

シンプルなインターフェイス - Flyable

`Flyable`インタフェースは、のコードをつクラスモジュールです。

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

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

Classモジュールの`Airplane`は、 `Flyable_Fly()`サブと`Long`をす`Flyable_GetAltitude()`の2つのメソッドがないかぎり、 `Implements`キーワードをしてエラーをさせます。

```
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
```

2のクラスモジュールである`Duck`も`Flyable`しています

```
Implements Flyable

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

Public Function Flyable_GetAltitude() As Long
```

```
    Flyable_GetAltitude = 30
End Function
```

たちは、 `Flyable`をけれ、 `Fly`または`GetAltitude`コマンドにすることをっているルーチンをくことができます

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

ィンターフェイスがされているため、IntelliSenseポップアップウィンドウに`F Fly`および`GetAltitude`がされます。

のコードをすると

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

FlyAndCheckAltitude MyDuck
FlyAndCheckAltitude MyAirplane
```

はのとおりです。

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

サブルーチンは`Airplane`と`Duck`で`Flyable_Fly`というがけられていますが、またはパラメータが`Flyable`としてされているは、 `Fly`としてびすことができます。がに`Duck`としてされているは、`Flyable_Fly`として`Flyable_Fly`ます。

1つのクラスののインターフェース - でスイム

`Flyable`をとして、 `Swimmable`という2のインターフェイスをのコードでできます。

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

`Duck`オブジェクトはとのを`Implement`できます

```
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
```

`Fish`クラスでも`Swimmable`をできます

```
Implements Swimmable

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

ここで、 `Duck`オブジェクトは、では`Flyable`としてSubに、もうでは`Swimmable`としてすことができます。

```
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
```

このコーードのはのとおりです。

フライチェック...

でんで

30

ジェットエンジンでふ

10000

チェック...

にかぶ

のでの

オンラインでインターフェイスをむ https://riptutorial.com/ja/vba/topic/8784/インターフェイス

12: エラー

Examples

エラーの

ランタイムエラーがした、いコードがそれをするがあります。なエラーは、エラーをチェックし、エラーをきこすコードのをにするコードをすることです。

ランタイムエラーをらすためのなの1つは、さなきをくことです。きがするがなくなればなるほど、コードがデバッグされやすくなります。

エラーの**91 - Object**または**With**ブロックがされていない

このエラーは、がりてられるにオブジェクトがされているときにします。オブジェクトパラメータをけるプロシージャがあるかもしれません

```
Private Sub DoSomething(ByVal target As Worksheet)
    Debug.Print target.Name
End Sub
```

`target`がりてられていない、のコードでは、オブジェクトにのオブジェクトがまれているかどうかをチェックすることで、にできるエラーがします。

```
Private Sub DoSomething(ByVal target As Worksheet)
    If target Is Nothing Then Exit Sub
    Debug.Print target.Name
End Sub
```

`target`リファレンスがりてられていない、りてリファレンスはされず、エラーはしません。

1つのパラメータがでないときにプロシージャをにするこのは、 ガードとばれます。

エラーの**9 -** えがです

このエラーは、がでアクセスされたにします。

```
Private Sub DoSomething(ByVal index As Integer)
    Debug.Print ActiveWorkbook.Worksheets(index)
End Sub
```

`ActiveWorkbook`のワークシートのよりもきいインデックスがえられている、のコードはエラーをさせます。なガードでは、のことをできます。

```
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
```

ほとんどのランタイムエラーは、するにしているをにし、な`If`をしてのパスですることでできます。なの。

エラー

ガードをしても、プロシージャのでするのあるすべてのエラーをににすることはできません。`On Error GoTo`ステートメントは、にしないエラーがするたびにVBAにラインラベルにジャンプし、「エラーモード」とするようします。エラーをした、コードがする「」のにってできる`Resume`キーワードを。

のラベルはサブルーチンをしますサブルーチンはのBASICコードにし、`GoTo`と`GoSub`ジャンプと`Return`ステートメントをして「main」ルーチンにジャンプするため、にされていないはしいスパゲッティコードをするのはかなりです。このようなから、

- プロシージャには**1**つのみのエラーサブルーチンがあります
- エラーサブルーチンはエラーでしかされません

これは、そのエラーをするプロシージャで、のようにするがあります。

```
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
```

エラー

には、なるアクションでさまざまなエラーをしたいことがあります。その、したエラーにするをむグローバルな`Err`オブジェクトをべ、それにじてします

```
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
```

なガイドラインとして、サブルーチンやのエラーをオンにし、そのスコープでするすべてのエラーをすることをしてください。コードのさなセクションのセクションでのみエラーをするがあるは、じレベルでエラーをオンまたはオフにします。

```
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
```

VBAは、レガシースタイルたとえばQBASICのをサポートします。 `Erl` hiddenプロパティは、のエラーをさせたをするためにできます。をしていない、 `Erl`は0しかしません。

```
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
```

あなたがではなく、して、をしている、 `Erl` エラーがしたのに、のをします。

```
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
```

`Erl`は`Integer`しかたず、ってオーバーフローすることに`Erl`ください。これは、 のがったをもたらすことをします。

```
Sub DoSomething()
99997 On Error GoTo 99999
99998 Debug.Print 42 / 0
99999
      Debug.Print Erl   'Prints 34462
End Sub
```

は、エラーのとなったステートメントほどではなく、のいたはすばやくなものになり、メンテナンスにはあまりにちません。

キーワードをする

エラーサブルーチンは、のいずれかをいます。

- プロシージャのまでします。この、びしプロシージャでがされます。
- `Resume`キーワードをして、じプロシージャでをします。

`Resume`キーワードは、エラーサブルーチンでのみするがあります.VBAがエラーにならずに`Resume`した、ランタイムエラー20「エラーなしで」がするからです。

エラーサブルーチンで`Resume`キーワードをするには、いくつかのがあります。

- `Resume`のみをすると、エラーのとなったステートメントでがされます。そのにエラーがにされない、じエラーがし、がループにるがあります。
- `Resume Next`は、エラーをきこしたステートメントののステートメントでをします。そのにエラーがにされていないは、なデータがしてされ、エラーやしないがするがあります。
- `Resume [line label]`は、されたライン ラベル またはレガシースタイルのラインをしているはラインでをけます。これは、びしにるにデータベースがじられていることをするなど、プロシージャをにするにクリーンアップコードをすることをにします。

エラーのにへ

`On Error`ステートメントは`Resume`キーワードをして、 すべてのエラーをにするようにVBAランタイムにできます 。

> そのにエラーがにされていないは、なデータがしてされ、 エラーやしないがするがあります 。

のはにすることはできません。 ***On Error Resume Next***は、すべてのエラーをにし、カーペット

のでそれらをします。ながえられた、ランタイムエラーがしたプログラムは、の/しないデータをしけるプログラムよりもれています。なぜなら、バグがはるかににできるからです。 `On Error Resume Next` バグをにすことができます。

`On Error`ステートメントは、プロシージャスコープです。そのため、 、そのような`On Error`ステートメントは、のプロシージャに**1**つだけするがあります。

しかし、 にはエラーをけることができないがあります。エラーのサブルーチンにジャンプするだけで、 `Resume Next`がしくしません。こののの、2つの`On Error`ステートメントのに、ののあるステートメントをラップすることができます。

```
On Error Resume Next
[possibly-failing statement]
Err.Clear 'resets current error
On Error GoTo 0
```

`On Error GoTo 0`は、プロシージャーのエラーをリセットします。これにより、エラーをきこすそれのはそのプロシージャーでされず、わりにアクティブ·エラー·ハンドラーによってキャッチされるまでコール·スタックをします。びしスタックにアクティブなエラーハンドラがない、されないとしてわれます。

```
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
```

カスタムエラー

なクラスをくときには、のエラーをさせたいとうでしょうし、これらのカスタムエラーをするためのユーザ/びしコードのためのきれいながになります。これをするためのきちんとしたは、の`Enum`をすることです。

```
Option Explicit
```

```
Public Enum FoobarError
    Err_FooWasNotBarred = vbObjectError + 1024
    Err_BarNotInitialized
    Err_SomethingElseHappened
End Enum
```

vbObjectErrorビルトインをすると、カスタムエラーコードがみ/のエラーコードとしないようにできます。ののみをにするがあります。Enumメンバーのになるはのメンバーより1きいため、Err_BarNotInitializedのはにvbObjectError + 1025です。

のランタイムエラーをさせる

Err.Raiseをしてエラーをさせることができるため、カスタムErr_FooWasNotBarredエラーはのようにさせることができます。

```
Err.Raise Err_FooWasNotBarred
```

Err.Raiseメソッドでは、カスタムのDescriptionとSourceパラメータもErr.Raiseできます。このため、カスタムエラーのをするをすることをおめします。

```
Private Const Msg_FooWasNotBarred As String = "The foo was not barred."
Private Const Msg_BarNotInitialized As String = "The bar was not initialized."
```

そして、エラーをさせるのプライベートメソッドをします。

```
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
```

クラスのでは、これらのなプロシージャをコールしてエラーをさせることができます。

```
Public Sub DoSomething()
    'raises the custom 'BarNotInitialized' error with "DoSomething" as the source:
    If Me.Bar Is Nothing Then OnBarNotInitializedError "DoSomething"
    '...
End Sub
```

クライアントコードは、それのエラーサブルーチンので、のエラーとにErr_BarNotInitializedをできます。

のErrorキーワードはErr.Raiseわりにすることもできますが、またはされました。

オンラインでエラーをむ https://riptutorial.com/ja/vba/topic/3211/エラー

13: オートメーションまたはのアプリケーションライブラリの

き

Visual Basicアプリケーションのとしてのアプリケーションのオブジェクトをするは、それらのアプリケーションのオブジェクトライブラリへのをすることができます。このドキュメントでは、Windowsシェル、Internet Explorer、XML HttpRequestなどのさまざまなソフトウェアのライブラリをするのリスト、ソース、およびをします。

- expression.CreateObjectObjectName
- ;。 Applicationオブジェクトをす。
- ObjectName;の。するオブジェクトのクラス。なクラスのについては、「OLEプログラム」をしてください。

- MSDN - オートメーションの

  アプリケーションがオートメーションをサポートする、アプリケーションがするオブジェクトにはVisual Basicによってアクセスできます。 Visual Basicをして、オブジェクトのメソッドをびすか、オブジェクトのプロパティをしてして、これらのオブジェクトをします。

- MSDN - オブジェクトライブラリリファレンスのまたは

  Visual Basicアプリケーションのとしてのアプリケーションのオブジェクトをするは、それらのアプリケーションのオブジェクトライブラリへのをすることができます。これをうには、まずアプリケーションがオブジェクトライブラリをしていることをするがあります。

- MSDN-Referencesダイアログボックス

  そのアプリケーションのオブジェクトライブラリへのをすることにより、コードでなのアプリケーションのオブジェクトをすることができます。

- MSDN-CreateObjectメソッド

  されたクラスのAutomationオブジェクトをします。アプリケーションがすでにされている、CreateObjectはしいインスタンスをします。

Examples

VBScriptの

```
Set createVBScriptRegExObject = CreateObject("vbscript.RegExp")
```

ツール>リファレンス> Microsoft VBScript。
けられたDLLVBScript.dll
ソースInternet Explorer 1.0および5.5

- MSDN-Microsoft、でVBScriptを
- MSDN-のスクリプト
- expert-exchange - Visual Basic for ApplicationsおよびVisual Basic 6でのの
- Microsoft ExcelのRegexをセルとループのでする
- regular-expressions.info/vbscript
- regular-expressions.info/vbscriptexample
- WIKI-

コード

このをしてRegExのをし、すべての1つのを1つのにし、をExcelセルにすることができます。

```
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
```

スクリプトファイルシステムオブジェクト

```
Set createScriptingFileSystemObject = CreateObject("Scripting.FileSystemObject")
```

ツール>> Microsoft Scripting Runtime
けられたDLLScrRun.dll
ソースWindows OS

MSDN - FileSystemObjectをしたファイルへのアクセス

> ファイルシステムオブジェクトFSOモデルは、フォルダとファイルをするためのオブジェクトベースのツールをします。いれたobject.methodをして、なプロパティ、メソッド、およびイベントのセットをして、フォルダやファイルをできます。のVisual Basicのステートメントとコマンドをすることもできます。
>
> FSOモデルは、アプリケーションにフォルダの、、、、またはのフォルダがするかどうか、およびそののをにします。また、フォルダのややなど、フォルダにするをすることもできます。

MSDN-FileSystemObjectのトピック「... FileSystemObject のとそのいをしてください」
exceltrick-FileSystemObject in VBA - Scripting.FileSystemObject

スクリプトオブジェクト

```
Set dict = CreateObject("Scripting.Dictionary")
```

ツール>> Microsoft Scripting Runtime
けられたDLLScrRun.dll
ソースWindows OS

Scripting.Dictionaryオブジェクト
MSDNオブジェクト

Internet Explorerオブジェクト

```
Set createInternetExplorerObject = CreateObject("InternetExplorer.Application")
```

ツール>> Microsoftインター—ネットコントロー—ル
するDLLieframe.dll
ソースInternet Explorerブラウザ

MSDN-InternetExplorerオブジェクト

オートメーションによってWindows Internet Explorerのインスタンスをします。

Internet Explorer Objecのメンバー—

のコードは、IEオブジェクトがどのようにするのか、VBAをってIEオブジェクトをするをします。はそれをステップすることをおめします。そうしないと、のナビゲーションにエラーがするがあります。

```
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
```

Webスクレイピング

IEでうもなことは、ウェブサイトのをきめるか、ウェブサイトのフォームをしてをすることです。それをどうやってうかをていきます。

*example.com*のソースコードについてえてみましょう

```
<!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>
```

をしたりしたりするには、のようなコードをできます。

```
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
```

がこっているここでなプレーヤーはHTMLソースコードである**.Document**です。いくつかのクエリをして、なコレクションまたはオブジェクトをすることができます。
たとえば、 `IE.Document.GetElementsByTagName("title")(0).innerHtml` です。 `GetElementsByTagName` は、" *title* "タグをつHTMLのコレクションをします。このようなタグはソースコードに1つしかありません。 コレクションは0ベースです。したがって、のをするには `(0)` をします。ここでは、Elementオブジェクトではなく、 `innerHtml` Stringだけをとしています。そこで、たちがむプロパティをします。

クリック

サイトのリンクをたどるには、のがあります

```
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 HTML Object LibraryまたはIEベストフレンド

IEにロードされるHTMLをにするには、 *Microsoft HTML Object Library*などののライブラリをできますまたはするがあります。のでこれについてもっとしく。

IEのな

IEのなは、ページがみまれ、するができていることをすることです。 `Do While... Loop`はにちますが、がありません。

また、IEをしてHTMLコンテンツをスクラップするのはOVERKILLです。どうしてブラウザはブラウズ、つまりすべてのCSS、JavaScript、ピクチャ、ポップアップなどでWebページをするためのものです。データのみがなは、なるアプローチをしてください。たとえば、 XML HTTPRequestをします 。のでこれについてもっとしく。

オンラインでオートメーションまたはのアプリケーションライブラリのをむ
https://riptutorial.com/ja/vba/topic/8916/オートメーションまたはのアプリケーションライブラリの

14: オブジェクトVBA

Examples

レベルは、をするをするのにちます。

は、ますますなコードでをすることによってされます。マクロのエントリーポイントは、 のいさなプロシージャーでなければならないので、がこっているのかをにできます 。

```
Public Sub DoSomething()
    With New SomeForm
        Set .Model = CreateViewModel
        .Show vbModal
        If .IsCancelled Then Exit Sub
        ProcessUserData .Model
    End With
End Sub
```

`DoSomething`プロシージャはがいため、フォームをしてモデルをし、そのオブジェクトをするをっている`ProcessUserData`プロシージャにそのオブジェクトをすことができます。モデルのはのプロシージャのジョブです。

```
Private Function CreateViewModel() As ISomeModel
    Dim result As ISomeModel
    Set result = SomeModel.Create(Now, Environ$("UserName"))
    result.AvailableItems = GetAvailableItems
    Set CreateViewModel = result
End Function
```

`CreateViewModel`は、の`ISomeModel`インスタンスののみをします。そののは、 なアイテムのをすることです。これらのアイテムのは、 `GetAvailableItems`プロシージャのでされたのです。

```
Private Function GetAvailableItems() As Variant
    GetAvailableItems = DataSheet.Names("AvailableItems").RefersToRange
End Function
```

ここでは、 `DataSheet`シートワークシートのきからなをみみます。データベースからみむこともできますし、をハードコードすることもできます。これはのであり、いレベルのいずれにもしません。

カプセル

カプセルは、クライアントコードからのをします。

Handler QueryCloseのでは、カプセルのをします。フォームにはチェックボックスコントロールがありますが、そのクライアントコードはしません。チェックボックスはので、クライアントコ

—ドはがかどうかをるがあります。

チェックボックスのがされると、ハンドラはプライベ—トフィ—ルドメンバをりてます。

```
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
```

クライアントコ—ドがそのをみたい、チェックボックスをするはありません。わりにに`SomeOtherSetting`プロパティをします

```
Public Property Get SomeOtherSetting() As Boolean
    SomeOtherSetting = this.SomeOtherSetting
End Property
```

`SomeOtherSetting`プロパティは 、チェックボックスのをカプセルします。クライアントコ—ドにはチェックボックスがまれていることをるはなく、ブ—ルのしかありません。 `Boolean`をカプセルすることで、チェックボックスのりにレイヤ—をしました。

インタフェ—スをしてをする

のクラスモジュ—ルにフォ—ムのモデルを カプセルすることで、これをさらにめてみましょう。しかし、 `UserName`と`Timestamp` `Public Property`をしたは、 `Property Let`アクセサをしてプロパティをにするがあります。クライアントコ—ドでは、にこれらのをするはありません。

ののの`CreateViewModel`は、 `ISomeModel`クラスをします。これはたちのインタ—フェイスであり、のようになります。

```
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
```

Notice `Timestamp`プロパティと`UserName`プロパティは、 `Property Get`アクセサのみをします。は`SomeModel`クラスがそのインタフェースをできます

```
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
```

インタフェースメンバはすべて`Private`であり、コードをコンパイルするためにはインタフェースのすべてのメンバをするがあります。 `Public`メンバーはインターフェイスのではないため、`ISomeModel`インターフェイスにしてされたコードにされません。

ファクトリメソッドをしてコンストラクタをシミュレートする

VB_PredeclaredIdをすると、 `SomeModel`クラスにのインスタンスが`static`れ、クライアントコードがにするなくびすことができるタイプレベルVB.NETで`Shared`、Cで`static` メンバーのようにするをできますたちがここでったように、インスタンス

```
Private Function CreateViewModel() As ISomeModel
    Dim result As ISomeModel
    Set result = SomeModel.Create(Now, Environ$("UserName"))
    result.AvailableItems = GetAvailableItems
    Set CreateViewModel = result
End Function
```

このファクトリメソッドは、 `ISomeModel`インターフェイスからアクセスするとみりのプロパティをりてます。ここでは`Timestamp`と`UserName`。

```
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
```

そして、 `ISomeModel`インターフェイスにしてコードをすることができます。このインターフェイスでは、 `Timestamp`と`UserName` 、コードがインターフェイスにしてされているりりてできないみりプロパティとしてされます。

とは、なるとなるにじ インタフェースをするです。

インターフェイスをするにより、UI、データベース、またはこのワークシートからアプリケーションロジックをにりすことができます。

フォームにされている`ISomeView`インターフェイスがあるとします。

```
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
```

フォームのコードビハインドはのようになります。

```
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
```

しかし、ユーザーフォームなしで`ISomeView`インターフェイスをするのクラスモジュールをすることはされていません。これは`SomeViewMock`クラスになります。

```
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
```

そして、 `UserForm`でするコードをして、インスタンスするのではなくパラメータとしてフォームをすなどして、 `ISomeView`インターフェイスをさせることができます

```
Public Sub DoSomething(ByVal view As ISomeView)
    With view
        Set .Model = CreateViewModel
        .Show
        If .IsCancelled Then Exit Sub
        ProcessUserData .Model
    End With
End Sub
```

ので`DoSomething`は、すなわち、 およびないクラス えば、のインターフェイスに`UserForm` 、 々はことをし、ユニットテストくことができ`ProcessUserData`したときにされていない`view.IsCancelled`ある`True`たちをることによって、テストは、`SomeViewMock`その、インスタンスを`IsCancelled`にプロパティを`True` 、そしてそれをす`DoSomething`。

テストなコードはにする

VBAでテストをくことができます。そこにはIDEにもするアドインがあります。しかし、コードがワークシート、データベース、フォーム、またはファイルシステムとにされている、テストはのワークシート、データベース、フォーム、またはファイルシステムをとしめます。これらのはしいテストはのワークシート、データベース、フォーム、またはファイルシステムをとしないように、テストなコードをするがあります。

テストコードがスタブ/の`SomeViewMock`のようにをできるように、インターフェイスにしてコードをくことによって、 "された"でテストをし、なり1つ1つフォームをしてでフォームコントロールをクリックしなくても、フォームのデータにするユーザーのやりとりのをえることができます。

オンラインでオブジェクトVBAをむ https://riptutorial.com/ja/vba/topic/5357/オブジェクトvba

15: カスタムクラスの

このでは、VBAでなカスタムクラスをするをします。とはしばしばにされるため、 `DateRange`オブジェクトのがされます。

Examples

クラスへのプロパティの

`Property`プロシージャは、モジュールのカスタムプロパティをまたはするのステートメントです。

プロパティアクセサには3つのタイプがあります。

1. プロパティのをす`Get`プロシージャ。
2. `Let` りて`Object`オブジェクトへの。
3. `Object`をりてる`Set`プロシージャ。

プロパティアクセサは、プロパティごとに`Get`と`Let` / `Set`をしてペアでされることがよくあります。 `Get`プロシージャのみをつプロパティはみりですが、 `Let` / `Set`プロシージャのみをつプロパティはきみです。

のでは、4つのプロパティアクセサが`DateRange`クラスにしてされてい`DateRange`。

1. `StartDate` みみ/きみ 。のよりいをす。プロシージャは、モジュール`mStartDate`をします。
2. `EndDate` みみ/きみ 。ののをす。プロシージャは、モジュール`mEndDate`をします。
3. `DaysBetween` みり 。 2つののののをすされた。 `Get`プロシージャのみがするため、このプロパティはできません。
4. `RangeToCopy` きみ 。の`DateRange`オブジェクトのをコピーするためにされる`Set`プロシージャ。

```
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
```

クラスにをする

クラスモジュールののパブリック`Sub` 、 `Function` 、または`Property`は、びしのにオブジェクトをけてびすことができます。

```
Object.Procedure
```

`DateRange`クラスでは、 `Sub`をしてにをできます。

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

`Function`はののをすことができます `GetFirstDayOfMonth`はプライベートであるため、クラスにはされません。

```
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
```

プロシージャは、されているクラスのオブジェクトへのをむののをけれることができます。

のでは、の`DateRange`オブジェクトのとが、の`DateRange`オブジェクトのとをむかどうかをテストします。

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

コードをしているオブジェクトのにアクセスするとして、 `Me`をすることにしてください。

クラスモジュールスコープ、インスタンスおよび

では、しいクラスモジュールはプライベートクラスなので、されているVBProjectでのインスタンスとにのみできます。 じプロジェクトののにクラスをし、インスタンスしてできます。

```
'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
```

しかし、くの、プロジェクトでモジュールをコピーせずに 、のプロジェクトでしたいクラスをします。 `ProjectA`で`List`というのクラスをし、そのクラスを`ProjectB`でする、4つのアクションをするがあります。

1. プロパティウィンドウの`ProjectA`の`List`クラスのインスタンスプロパティを`Private`から`PublicNotCreatable`
2. 「」を`ProjectA`のインスタンスをしてし`List`クラスを。、ファクトリには、クラスインスタンスののためのがまれます。 `ProjectB`ではクラスをできますが、 `ProjectB`は`ProjectA`のクラスのインスタンスをできないため、ファクトリがです。

   ```
   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...Tools..References...`メニューをして`ProjectA`へのをします。
4. `ProjectB`で、をし、 `ProjectA`ファクトリをして`List`インスタンスをりてます

   ```
   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
   ```

オンラインでカスタムクラスのをむ https://riptutorial.com/ja/vba/topic/4464/カスタムクラスの

16: コメント

コメントブロック

にのをコメントしたりコメントをしたりするがあるは、IDEの**Edit Toolbar**ボタンをすることができます。

コメントブロック - したすべてののにアポストロフィを1つします

Uncomment Block - したすべてののからのアポストロフィをします。

のコメントののくはのブロックコメントをサポートしていますが、VBAでは1のコメントのみがされています。

Examples

アポストロフィのコメント

コメントはアポストロフィ ' でマークされ、コードがされるとされます。コメントは、をむのにあなたのコードをするのにちます。

コメントでまるはすべてされるので、デバッグやリファクタリングのにコードがされないようにすることもできます。あなたのコード'にアポストロフィをいてコメントにします。 これはのコメントアウトとばれます。

```
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コメント

```
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
```

オンラインでコメントをむ https://riptutorial.com/ja/vba/topic/2059/コメント

17: コレクション

`Collection`は、VBAランタイムにまれるコンテナオブジェクトです。それをするためにのはありません。 `Collection`は、のデータのをするためにでき、のインデックスまたはオプションののキーをしてすることができます。

ととの

	コレクション	アレイ	
サイズ	はい	には[1]	はい
アイテムがされる	はい	はい	はい[2]
アイテムはくけされています	いいえ	はい	いいえ
アイテムはでりすことができます	はい	はい	いいえ
しいアイテムをにできます	はい	いいえ	いいえ
アイテムがするかどうかをする	すべてのアイテムをする	すべてのアイテムをする	すべてのアイテムをする
アイテムはキーでりすことができます	はい	いいえ	はい
キーはとをします	いいえ	N / A	オプション[3]
キーがするかどうかをする	エラーハンドラ	N / A	`.Exists`
すべてのアイテムを	と`.Remove`	`Erase` 、 `ReDim`	`.RemoveAll`

[1]のみのサイズがで、ののののみがサイズです。

[2]となる`.Keys`と`.Items`がされます。

[3] `.CompareMode`プロパティによって`.CompareMode`ます。

Examples

コレクションにアイテムをする

アイテムは、 `.Add`メソッドをびすことによって`Collection`されます。

```
.Add(item, [key], [before, after])
```

パラメ―タ	
	`Collection`にするアイテム。これはに、プリミティブ、、オブジェクト、および`Nothing`をむ、をりてることができるのになります。
キ―	オプション。 `Collection`からアイテムをするためののとしてする`String`です。されたキ―が`Collection`にする、「このキ―はにこのコレクションのにけられています」というエラ―457がします。
	オプション。アイテムを`Collection`にするためののキ― `String` またはインデックス。がされている、 *after*パラメ―ター―はであるか、エラ―5です。「なプロシ―ジャ―びしまたはき」がします。 `Collection`にしない`String`キ―がされると、エラ―5「なプロシ―ジャびしまたは」がします。 `Collection`にしないインデックスがされた、エラ―9 "きがです"がします。
	オプション。 `Collection`にアイテムをするのキ― `String` またはインデックス。をした、 *before*パラメ―ター―はでなければなりません。したエラ―は*before*パラメ―ター―とじです。

ノ―ト

- キ―ではとはされません。 `.Add "Bar", "Foo"`、 `.Add "Baz", "foo"`を`.Add "Baz", "foo"`と`.Add "Baz", "foo"`はキ―になります。
- オプションの*before*または*after*パラメ―タのどちらもされていない、そのは`Collection`ののにされ`Collection`。
- *before*または*after*パラメ―タをしてすると、のメンバ―のインデックスがしいにするようにされます。これは、をしてル―プにをうときにはがであることをします。

```
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
```

コレクションからアイテムをする

アイテムは、 `.Remove`メソッドをびすことによって`Collection`からされます`.Remove`

```
.Remove(index)
```

パラメータ	
	`Collection`からする。されたがまたはのサブタイプをつ`Variant`、インデックスとしてされます。されたがをむ`String`または`Variant`、それはaキーとしてされます。`Collection`にしないStringキーがされると、エラー5「なプロシージャびしまたは」がします。 `Collection`にしないインデックスがされた、エラー9 "きがです"がします。

ノート

- をする`Collection`で、それのにすべてののインデックスをします`Collection`。 `For`、インデックスをしてにするがあるをするループ `Step -1` のをぎ、アイテムをスキップします。
- アイテムは、 `For Each`ループのから`Collection`からするはありません 。これにより、しないがじるがあります。

```
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
```

コレクションのアイテムの

`Collection`のアイテムのは、その`.Count`をびすことでできます。

```
.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
```

コレクションからアイテムをする

アイテムは、 `.Item`をびすことで`Collection`からできます。

```
.Item(index)
```

パラメータ	
	`Collection`からする。されたがまたはのサブタイプをつ`Variant`、インデックスとしてされます。されたがをむ`String`または`Variant`、それはaキーとしてされます。`Collection`にしないStringキーがされると、エラー5「なプロシージャびしまたは」がします。 `Collection`にしないインデックスがされた、エラー9 "きがです"がします。

ノート

- `.Item`は`Collection`のデフォルトメンバーです。これにより、のサンプルですように、のがになります。
- インデックスは1からまります。
- キーではとはされません 。 `.Item("Foo")`と`.Item("foo")`はじキーをします。
- *index*パラメータは、 `String`からのににキャストされたり、そののにはにキャストされませ

ん。 `.Item(1)`と`.Item("1")`は`Collection`なるをすることはにです。

インデックス

```
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
```

の

```
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
```

`.Item`はデフォルトのメンバーであり、の`String`をることができるので、bang `!` はされています。このののはわしい。

キー

Scripting.Dictionaryとはなり、 Collectionは、されたキーがするかどうか、 または Collectionするキーをするをするメソッドはありません。キーがするかどうかをするのは、エラーハンドラをすることです。

```
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
```

アイテム

アイテムがCollectionまれているかどうかをするのは、アイテムがつかるまでCollectionをすることです。 Collectionはプリミティブまたはオブジェクトのいずれかがまれるがあるため、にエラーがするのをけるために、いくつかのがです。

```
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
```

コレクションからすべてのアイテムをクリアする

Collectionからすべてのアイテムをするもなは、しいCollectionきえて、いCollectionをにすることです。

```
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
```

ただし、 Collectionへののがされている、このメソッドはりてられたののCollection します 。

```
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
```

この、をするもなは、 Collectionののをループして、もいをりしすることです。

```
Public Sub ClearCollection(ByRef container As Collection)
    Dim index As Long
    For index = 1 To container.Count
        container.Remove 1
    Next
End Sub
```

オンラインでコレクションをむ https://riptutorial.com/ja/vba/topic/5838/コレクション

18: サブストリング

VBAには、をむののをするためのみみがあります。

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

なonverheadそしてよりいのためをけるためには、がにされたとき、および/またはのがにされているときは、$のをする。

`Null`パラメータを$きのにすと、ランタイムエラー「なnullの」がします。これは、にデータベースにするコードにします。

Examples

LeftまたはLeft $をして、のの3をする

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

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

またはの$をして、のの3をする

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

MidまたはMid $をして、ののをする

```
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をして、またはにスペースをれずにのコピーをする

```
'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"
```

オンラインでサブストリングをむ https://riptutorial.com/ja/vba/topic/3481/サブストリング

19: データと

Examples

バイト

```
Dim Value As Byte
```

Byteはなし8ビットのデータです。 0から255までのをすことができ、そののをしようとすると、ランタイムエラー6がします 。 `Overflow` 。 Byteは、VBAでできるののなしです。

Byteにするキャストは`CByte()`です。からのキャストの、はされたもいにめられます。

バイトと

とバイトは、なはありませんによっていにすることができます。

えば

```
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をエンコードできるようにするには、のはのバイトをに2バイトをとります。えば

```
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
```

Integerはき16ビットデ―タです。 -32,76832,767ののをすることができ、そののをしようとするとエラ―6がします。オ―バ―フロ―。

はリトルエンディアンとしてメモリにされ、ネガは2のでされます。

よりさいがTypeのメンバであるか、APIびしまたはそののによって2バイトであるがあるをき、にIntegerではなくLongをすることをおめします。ほとんどの、VBAはをに32ビットとしています。したがって、はさなをするはありません。さらに、がされるたびに、Longとしてサイレントキャストされるため、パフォ―マンスペナルティがします。

Integerにするキャストは`CInt()`です。からのキャストの、はされたもいにめられます。

ブ―ル

```
Dim Value As Boolean
```

ブ―ルは、TrueまたはFalseのいずれかですことができるをするためにされます。には、デ―タは、Falseをす0とTrueをすそののをつ16ビットとしてされます。

ブ―ルがにキャストされると、すべてのビットが1にされることにしてください。これにより、きのは-1、なしByteのはになります。

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

Booleanにするキャストは`CBool()`です。 16ビットのとしてにされていますが、そののからブ―ルにキャストすることはオ―バ―フロ―からですが、16ビットをすべて1にします。

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

いです

```
Dim Value As Long
```

Longはき32ビットデ―タです。 -2,147,483,6482,147,483,647ののをでき、そののをしようとするとエラ―6がします。オ―バ―フロ―。

ロングはリトルエンディアンとしてメモリにされ、ネガは2のでされます。

Longは32ビットオペレーティングシステムのポインタのとするため、LongsはAPIとのでポインタをおよびすためによくされることにしてください。

Longにするキャストは`CLng()`。からのキャストの、はされたもいにめられます。

シングル

```
Dim Value As Single
```

Singleはき32ビットデータです。 リトルエンディアンの IEEE 754メモリレイアウトをしてにされます。そのため、データでできるのはありません。されているのは、されたのです。 Singleは-16,777,216から16,777,216ののをのなくできます。のはにします。

およそ2^{128}をえるがりてられた、Singleはオーバーフローします。にするに、ながわしいものになりますが、のでオーバーフローしません。

すべてのとに、をうときはがです。なにしたデルタをめることがベストプラクティスです。

Singleにするキャストは`CSng()`です。

ダブル

```
Dim Value As Double
```

Doubleは、き64ビットデータです。 シングルと、 リトルエンディアンの IEEE 754メモリレイアウトをしてにされ、にするじがです。 Doubleは、-9,007,199,254,740,9929,007,199,254,740,992ののをのなくできます。のはにします。

2^{1024}をえるがりてられた、Doubleはオーバーフローします。にするに、ながわしいものになりますが、のでオーバーフローしません。

Doubleにするキャストは`CDbl()`です。

```
Dim Value As Currency
```

は、 Doubleとのき64ビットデータですが、4のをめるために10,000 にスケールされています。には、-922,337,203,685,477.5808から922,337,203,685,477.5807までのをでき、32ビットアプリケーションでのイントリンシックのでのをえます。データのがすように、スケーリングがめをするのにつので、このデータををすときにすることをおめします。

にするキャスティングは`CCur()`です。

```
Dim Value As Date
```

のをしてくださいがDate1230 、1899のエポックからのをし、ののがき64ビットデータとしてに

されています。ののは、のとしてをします。したがって、のDateは12:00:00 AMのコンポ―ネントをち、x.5は12:00:00 PMのコンポ―ネントをちます。

のは、11 100 1231 9999ダブルよりきなをするので、そののをりてることによってをオ―バ―フロ―させることができるのです。

そのため、 Double for Dateのとじでできます。

```
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.
```

Dateにするキャストは、のの/をけれるCDate()です。のはされているのロケ―ルについてされるため、コ―ドがであるはキャストをけるがあることにすることがです。

はのをし、2つのがあります。

```
Dim Value As String
```

は、およびりてがで、COM BSTRとしてメモリにされます。これは、デ―タのさをワイド1あたり2バイトとし、2バイトのNULLバイトでわるデ―タのさをする4バイトのなしでされます。したがって、VBAでできるのは2,147,483,647です。

へのポインタ StrPtr()でStrPtr()は、さのプレフィックスではなく、 デ―タのメモリをします。つまり、へのポインタをとするAPIにVBA Stringをすことができます。

さがわるがあるため、VBA は、がりてられるたびにのメモリをりてします。これにより 、 をりしするプロシ―ジャにパフォ―マンスのペナルティがされるがあります。

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

は、にして2バイトずつりてられ、なバイトとしてメモリにされます。りてられると、Stringのさはです。それらはメモリでNULLされていないため、ヌルのでりてられたメモリをたすは、ヌルをするAPIにすのにはしていません。

のは、の16ビットのインデックスをきぎます。したがって、65,535のさしかできません。なメモリよりいをりてようとしても、ランタイムエラ―はしません。わりに、のはにりてられます。

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

いずれかののStringにするキャストはCStr()です。

LongLong

```
Dim Value As LongLong
```

LongLongは64ビットのきデータで、64ビットアプリケーションでのみできます。 64ビットオペレーティングシステムでする32ビットアプリケーションではできません。 -9,223,372,036,854,775,8089,223,372,036,854,775,807ののをすることができ、そののをしようとするとエラー6がします。オーバーフロー。

LongLongsはリトルエンディアンとしてメモリにされ、ネガは2のとしてされます。

LongLongデータは、VBAの64ビットオペレーティングシステムサポートのとしてされました。64ビットアプリケーションでは、このをして64ビットAPIへのポインタをしてすことができます。

LongLongにするキャストは`CLngLng()`です。からのキャストの、はされたもいにめられます。

バリアント

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

バリアントは、ののおよびにされるCOMデータであり、VBAののはバリアントにりてることができます。 `As [Type]`されたなをたないされたは、デフォルトでVariantになります。

バリアントは、 VARIANTとしてメモリにされます。VARIANT には、バイトディスクリプタ VARTYPE とそれにく6つのバイト、に8バイトのデータがまれます。とブールをむでは、はバリアントにされます。のすべてのの、データには、になるへのポインタがまれます。

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

バリアントのになるは、にされたをす`VarType()`、またはをす`TypeName()`でできます。

```
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"
```

バリアントはのタイプのをできるため、 タイプヒントのないリテラルからのりては、のにってなタイプのバリアントににキャストされます。タイプヒントをつリテラルは、ヒントタイプのバリアントにキャストされます。

	のタイプ
の	

	のタイプ
ロングレンジの	いです
ロングレンジの	ダブル
すべての	ダブル

バリアントFor Eachループまたは APIのイテレータをするながないり、、のでルーチンのタスクではをけるがあります。

- ではないため、エラーのがくなります。たとえば、Integerをする Variantは、オーバーフローするのではなく、にLongにされます。
- それらは、なくとも1つのポインタをとすることによってオーバヘッドをする。
- Variantのメモリは、となるをするためになメモリよりもになくとも 8バイトくなります。

Variantにするキャスティングは`CVar()`です。

LongPtr

```
Dim Value As LongPtr
```

LongPtrは、64ビットプラットフォームをサポートするためにVBAにされました。 32ビットシステムではLongとしてわれ、64ビットシステムではLongLongとしてわれます。

なは、のアーキテクチャにポインタをしてすなをすることです コンパイルのコードのを。

APIびしでされる、オペレーティングシステムによってメモリアドレスとしてわれますが、VBAはそれをきのしたがって、なしのきオーバーフローのとしてうことにしてください。このため、LongPtrsをしてされるポインタでは、`>`または`<`をしないでください。この "quirk"では、メモリのなアドレスをすなオフセットをするとオーバーフローエラーがするもあるため、VBAでポインタをするはがです。

LongPtrにするキャストは`CLngPtr()`です。からのキャストの、はもいにめられ、0.5がりげられますはメモリアドレスであるため、のりてターゲットとしてするのがもです。

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

'Set Value to the smallest possible Decimal value
Value = CDec("0.00000000000000000000000001")
```

`Decimal`データは、のサブタイプとしてのみです`Variant` 、あなたがまれているがありますのをしなければなりません`Decimal`として`Variant` して、りて`Decimal`して`CDec`を。キーワード`Decimal`はです VBAがにそのタイプのファーストクラスサポートをすることをしていますので、 `Decimal`をまたはプロシージャとしてすることはできません。

`Decimal`はVariantがとするバイトにえて14バイトのメモリをとし、28までのをできます。のの、のは-79,228,162,514,264,337,593,543,950,335+ 79,228,162,514,264,337,593,543,950,335です。のが28のの、されるのは-7.9228162514264337593543950335+ 7.9228162514264337593543950335です。

オンラインでデータとをむ https://riptutorial.com/ja/vba/topic/3418/データと

20: データ

き

[TODOこのトピックは、すべてのなCS101データのであり、VBAでのデータののをするものです。これは、VBAドキュメントのクラストピックにされたをびつけ、するいになります。]

Examples

リンクされたリスト

このリンクされたリストのは、 Setデータをしています。

SinglyLinkedNodeクラス

```
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クラス

```
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
```

バイナリツリー

これは、アンバランスなバイナリツリーのです 。バイナリツリーはに、ルートからにがるノードのとしてされています。ノードには、との2つのがあります。たとえば、7,5,9,3,11,6,12,14、および15がBinaryTreeにされたとします。はのようになります。このバイナリツリーはバランスがとれておらず、ルックアップのパフォーマンスをするでましいになりることにしてください。バイナリツリーのとしてAVLツリーをしてください。

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

BinaryTreeNodeクラス

```
Option Explicit

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

BinaryTreeクラス

[TODO]

オンラインでデータをむ https://riptutorial.com/ja/vba/topic/8628/データ

21: バイナリで2GB +ファイルをVBAとファイルハッシュでむ

き

VBAのバイナリファイルをにみむはですが、2GB2,147,483,647バイト - Longデータののがあります。テクノロジ—がするにつれて、この2GBのはにられます。オペレ—ティングシステムインストールDVDディスクのISOイメ—ジなど。マイクロソフトでは、レベルのWindows APIをしてこれをするをしています。ここではそのバックアップです。

また、Microsoftから`fciv.exe`ようなプログラムなしでファイルハッシュをするためのをむことをしてください。

マイクロソフトのクラスのためのメソッド

メソッド	
IsOpen	ファイルがいているかどうかをすブ—ルをします。
OpenFile としての *sFileNam*	sFileNameでされたファイルをきます。
CloseFile	いているファイルをじます。
ReadBytes *ByteCount* As Long	ByteCountバイトをみり、Variantバイトでし、ポインタをします。
WriteBytes Byteとしての*DataBytes*	バイトのをファイルののにきみ、ポインタをします。
す	Windowsににきみキャッシュをフラッシュさせます。
SeekAbsolute *HighPos* As Long、 *LowPos* As Long	ファイルポインタをファイルのからされたにします。 VBAはDWORDSをきのとしていますが、APIはそれらをなしとしています。のをゼロのにして4GBをえるようにします。DWORDは2GB 4GBのでのになります。
SeekRelative オフセットはく	ファイルポインタをのから+/- 2GBまでします。 64ビットきオフセットを2つの32ビットにすることにより、このメソッドを2 GBをえるオフセットにできるようにきすことができます。

マイクロソフトのクラスのプロパティ

プロパティ	
FileHandle	いているファイルのファイルハンドル。これはVBAファイルハンドルとがありません。
ファイル	いているファイルの。
オートフラッシュ	WriteBytesがにFlushメソッドをびすかどうかを/します。

ノーマルモジュール

	ノート
GetFileHash *sFile*は、 *uBlockSize*は Double、 *sHashType*は	**HashTypeMD5** 、 **HashTypeSHA1** 、 **HashTypeSHA256** 、 **HashTypeSHA384** 、 **HashTypeSHA512**のいずれかのプライベートの1つである、するハッシュのタイプバイト、するブロックサイズ、ハッシュするパスをげるだけです。これはなりになるようにされています。

それにじて**uFileSize As Double**のコメントをするがあります。はMD5とSHA1をテストしました。

Examples

これは**Class**モジュールになければなりません。はに **"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
```

モジュールでファイルハッシュをするためのコード

```
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
```

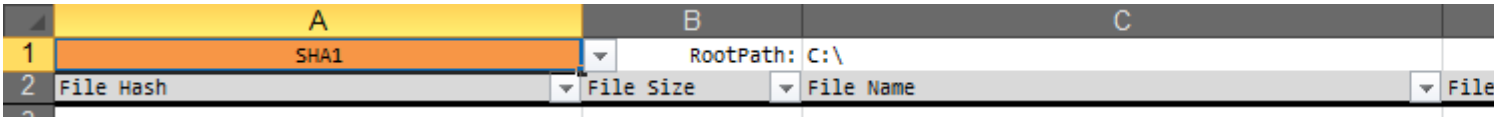
はにいですが、のテストファイルは、 **`BLOCKSIZE = 131071` 2 ^ 17-1**がWindows 7 x64の32ビットOffice 2010でにのパフォーマンスをし、に*2 ^ 16-165535*であることをしています。`2^27-1` はメモリーを `2^27-1` ます 。

ファイルサイズバイト	ファイル
146,800,640	clonezilla-live-2.2.2-37-amd64.iso
798,210,048	HPIP201.2014_0902.29.iso
2,073,016,320	SW_DVD5_Windows_Vista_Business_W32_32BIT_English.ISO
4,380,387,328	Win10_1607_English_x64.iso
5,400,115,200	SW_DVD9_Windows_Svr_Std_and_DataCtr_2012_R2_64Bit_English.ISO

ルートフォルダからすべてのファイルハッシュをする

のコードののバリエーションでは、すべてのサブフォルダをむルートフォルダからすべてのファイルのハッシュコードをするに、よりいパフォーマンスがられます。

ワークシートの



コード

```
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
```

オンラインでバイナリで2GB +ファイルをVBAとファイルハッシュでむをむ
https://riptutorial.com/ja/vba/topic/8786/バイナリで2gb-plusファイルをvbaとファイルハッシュでむ

22: フロー

Examples

ケースを

`Select Case`は、さまざまながなにできます。はからにチェックされ、するのケースだけがされます。

```
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`ブロックのロジックをして、さまざまなのテストをサポートすることもできます。このののシナリオでは、もできます。

```
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は、をすることもできます。 `Select Case`にしてがされている、 `Is`キーワードのにをするがあります。

```
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
```

ループについて

`For Each`ループは、コレクションのすべてのをするのにです。

```
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
```

オブジェクトコレクションをするときに`For Each`する

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

けてください`For Each`のをするとき。 `For`ループは、のパフォーマンスがにします。に、 `For Each` ループは、 `Collection`するのパフォーマンスがします。

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

`Next`キーワードには、オプションでイテレータをけることができます。ループをのプロシージャにするなど、ネストされたコードをにするためのよりいがありますが、ネストされたループをにするのにちます。

```
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
```

ループをう

```
 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
```

whileループ

```
'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
```

ループの

Forループは、されただけコードのまれたセクションをりすためにされます。のなは、なをしています。

```
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
```

のコードはをしi。 Forループは1と10ののすべてのをiりて、 Debug.Print iします。つまり、コードは1から10までのをウィンドウにします。ループは、とはのまれたコードがされたのNextステートメントによってインクリメントされることにしてください。

デフォルトでは、ループがされるたびにカウンタが1ずつインクリメントされます。ただし、Stepをすると、インクリメントをリテラルまたはのりのいずれかにできます。、、またはStepがの、もいにめられます。 Stepはのまたはののいずれかです。

```
Dim i As Integer
For i = 1 To 10 Step 2
    Debug.Print i       'Prints 1, 3, 5, 7, and 9
Next
```

に、 Forループは、ループがにまれたコードをするかがかっているでされますそうでなければDoループまたはWhileループがかもしれません。これは、このコードがすように、ループへののエントリのにがされているためです。

```
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
```

`For`ループは、 `Exit For`ステートメントでにすることができます。

```
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
```

オンラインでフローをむ https://riptutorial.com/ja/vba/topic/1873/フロー

23: プロシージャコール

- IdentifierName [*arguments*]
- Call IdentifierName [*arguments*]
- [Let | Set] = IdentifierName [ ]
- [Let | Set] IdentifierName [ ] =

パラメーター

パラメータ	
IdentifierName	びすプロシージャの。
	プロシージャにすのカンマりのリスト。

の2つのは、 `Sub`プロシージャをびすためのものです。のにかっこはまれていないことにしてください。

これはしていますをしてください。どうしてをうのはなぜですかの2つののいをしくしています。

3のは、 `Function`および`Property Get`プロシージャをびすためのです。パラメタがある、かっこはにです。 をりてるときは`Let`キーワードをできますが、 をりてるときは`Set`キーワードがです 。

4のは、 `Property Let`および`Property Set`プロシージャをびすためのです。のの*`expression`*がプロパティのvalueパラメータにされます。

Examples

なびしの

```
ProcedureName
ProcedureName argument1, argument2
```

プロシージャーをなしでびします。

エッジケース

`Call`キーワードは、1つののにのみです。

```
Call DoSomething : DoSomethingElse
```

`DoSomething`と`DoSomethingElse`はびされるプロシーージャです。 `Call`キーワードがされた、`DoSomething`はプロシーージャコールではなくラインラベルとしてされ、コードがします。

```
DoSomething: DoSomethingElse 'only DoSomethingElse will run
```

り

プロシーージャコールの `Function`や`Property Get`プロシーージャなどを`Property Get`するには、コールをりてのにします。

```
result = ProcedureName
result = ProcedureName(argument1, argument2)
```

パラメータがあるはカッコがです。プロシーージャにパラメータがない、かっこはです。

これはしています。どうしてをうのはなぜですか

カッコは、びしのをむためにされます 。 プロシーージャーびしのためにこれらをすると、しないがする

にしないをプロシーージャにすことで、そしてになでコンパイルにバグがするがあるためです。

ランタイム

をにするとバグがするがあります。オブジェクトをパラメータとしてるプロシーージャをすると...

```
Sub DoSomething(ByRef target As Range)
End Sub
```

...そしてかっこでびされました

```
DoSomething (Application.ActiveCell) 'raises an error at runtime
```

これにより、 "Object Required"ランタイムエラー424がします。そののエラーがするがあり`Application.ActiveCell`。 ここでは、 `Application.ActiveCell` `Range`オブジェクトは、 `target`が`ByRef`すことをするプロシーージャのシグネチャになく 、 され 、によってされます。のスニペットで`ByVal`を`DoSomething`にしたのは、 `Application.ActiveCell.Value`です。

は、のをするために、VBAをし、そのをす`ByVal`ばれるきへ。されたのがプロシーージャのされるとせず、にできない、エラーがします。

コンパイル

このコーードはコンパイルにします

```
MsgBox ("Invalid Code!", vbCritical)
```

`("Invalid Code!", vbCritical)`はとしてできないためです。

これはコンパイルしてします

```
MsgBox ("Invalid Code!"), (vbCritical)
```

しかしいなくかにえるだろう。したかっこはけてください。

なびしの

```
Call ProcedureName
Call ProcedureName(argument1, argument2)
```

なびしでは、キーワードのに`Call`キーワードとかっこがです。パラメータがない、はです。このは、よりななびしがVBにされたときにされました。

オプションの

いくつかのプロシージャにはオプションのがあります。オプションのはにののにますが、プロシージャはなしでびすことができます。

たとえば、`ProcedureName`が2つの `argument1` 、 `argument2` と1つのオプション`optArgument3`、なくとも4つのでびすことができます。

```
' 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")
```

ここでびされるヘッダのは、のようになります。

```
Function ProcedureName(argument1 As String, argument2 As String, Optional optArgument3 As
String) As String
```

`Optional`キーワードは、このをできることをします。にべたように-ヘッダにされたオプションのは、のなのに、にされなければなりません 。

がにされないは、のデフォルトをすることもできます。

```
Function ProcedureName(argument1 As String, argument2 As String, Optional optArgument3 As
String = "C") As String
```

このでは、 cのがえられていない、そのはデフォルトで"C"ます。がされている、これはデフォルトをオーバーライドします。

オンラインでプロシージャコールをむ https://riptutorial.com/ja/vba/topic/1179/プロシージャコール

24: プロシージャの

Examples

プロシージャの

Subは、のタスクをするがのをさないプロシージャです。

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

アクセスがされていない、プロシージャはデフォルトで「 Publicになります。

Functionは、データがえられ、にはグローバルまたはモジュールスコープのなしにをすプロシージャです。

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

Propertyは、モジュールデータをカプセルするプロシージャです。プロパティには3つのアクセサがあります。またはオブジェクトをすにはGet 、をりてるにはLet 、オブジェクトをりてるにはSetをします。

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

プロパティは、クラスモジュールでされますただし、モジュールでもされていますが。そうでなければびしコードではアクセスできないデータにアクセサをします。 Getアクセサだけをするプロパティは "みり"です。 Letアクセサおよび/またはSetアクセサのみをするプロパティは「きみ」です。きみのプロパティは、プログラミングのいとはみなされません。クライアントコードでをきむことができれば、それをみることができるはずです。きみのプロパティをするわりに、Subプロシージャをすることをしてください。

をす

FunctionまたはProperty Getプロシージャは、そのびしにをすことができますまた、そうするがあります。これは、プロシージャのをりてることによってわれます。

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

をいた

のように、は、プロシージャでりしするのあるさなコードをむよりさいプロシージャです。

は、コードのをらすためにされます。

プロシージャとに、はリストのにかかわらずできます。

すべてのがをすので、はりのとしてされます。のとりはじです。

1. パラメータき

```
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. パラメータなしの

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

は、でさまざまなでびすことができます。りのでされたはになので、とにされます。

びし

```
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
```

さらに、は、ifおよびののとしてすることもできます。

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

さらにくのでは、のBy refやBy valなどのをできます。

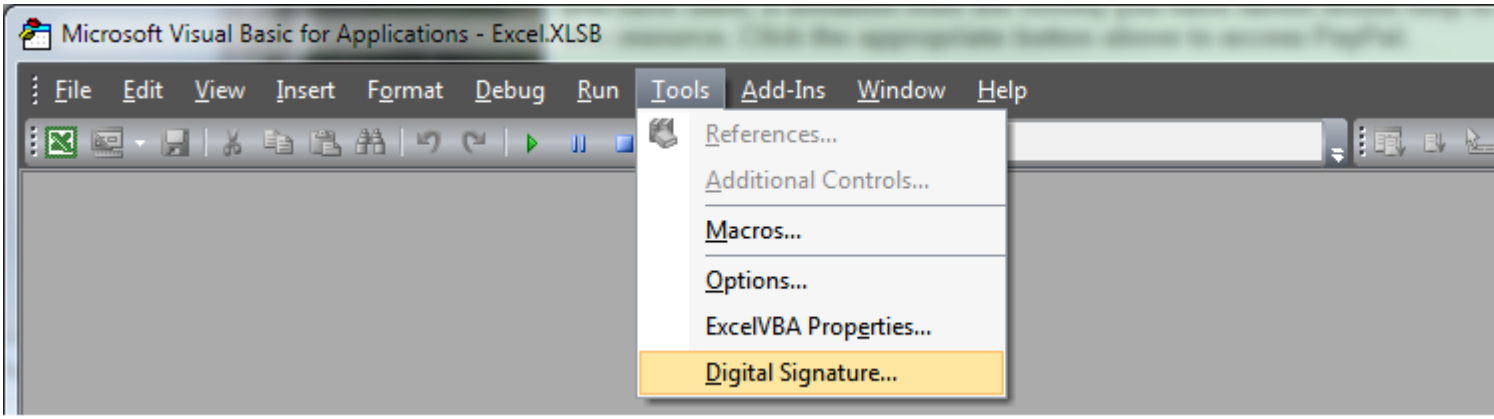
オンラインでプロシージャのをむ https://riptutorial.com/ja/vba/topic/1474/プロシージャの

25: マクロセキュリティとVBAプロジェクト/モジュールの

Examples

なデジタルをするSELFCERT.EXE

マクロをし、Officeアプリケーションのセキュリティをのあるコードからするには、 *VBA*エディタ>ツール>デジタルからVBAProject.OTMにデジタルするがあります 。

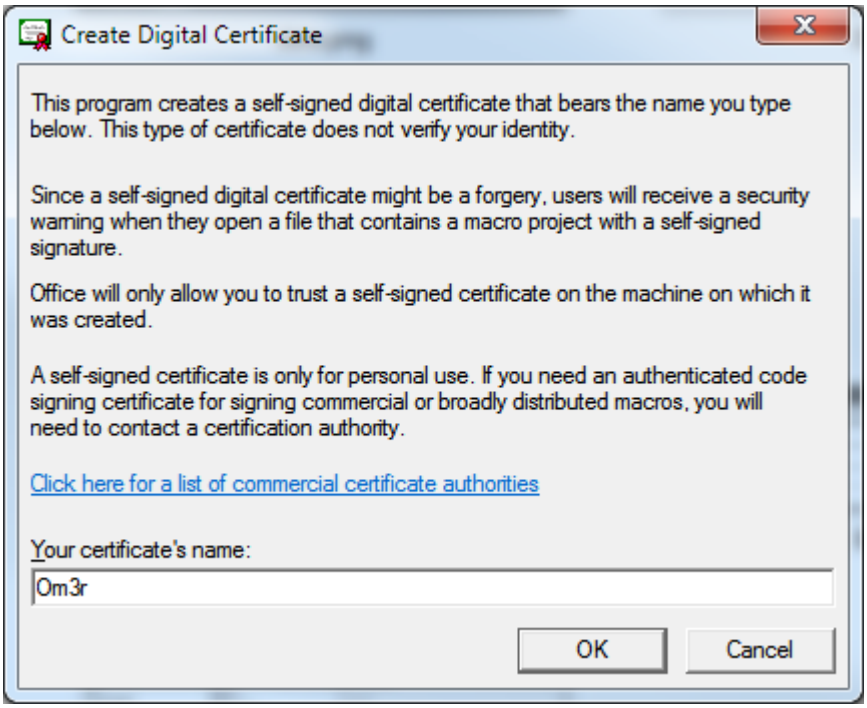


Officeには、デジタルをするユーティリティがしています。このは、PCでしてプロジェクトにすることができます。

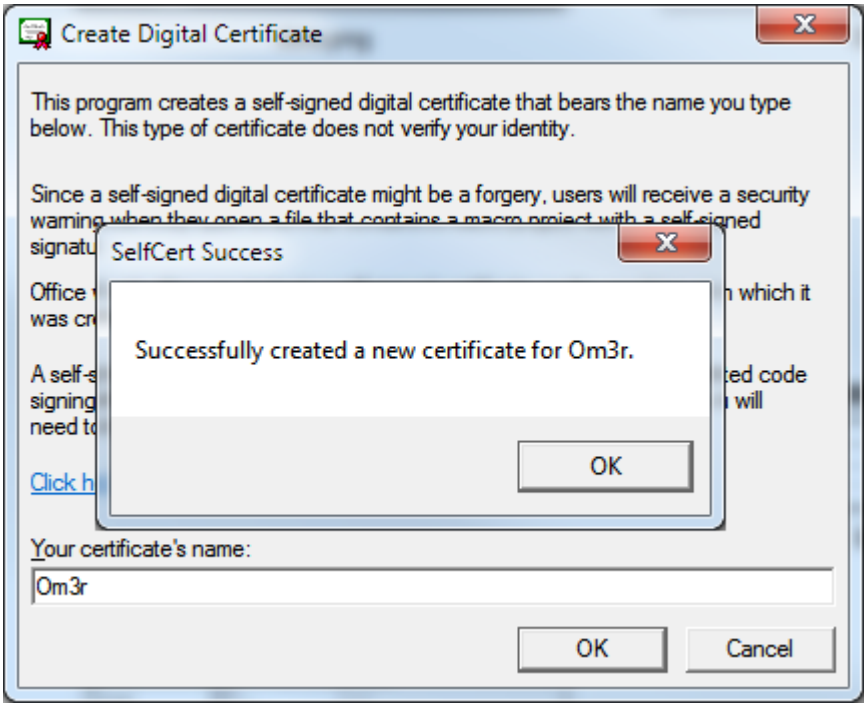
このユーティリティ**SELFCERT.EXE**は、Officeプログラムフォルダにあります。

[VBAプロジェクトのデジタル]をクリックして、ウィザードをきます。

ダイアログで、のなをし、「OK」をクリックします。

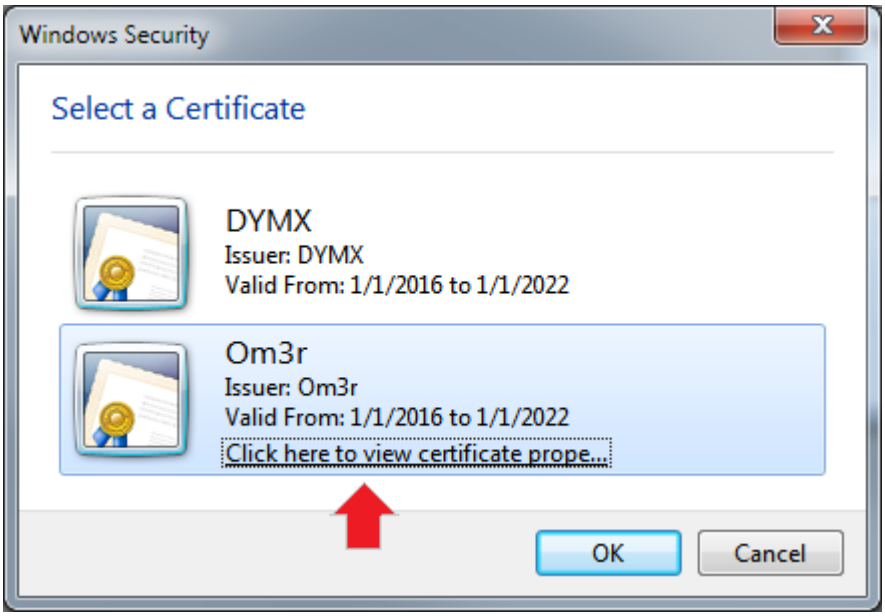


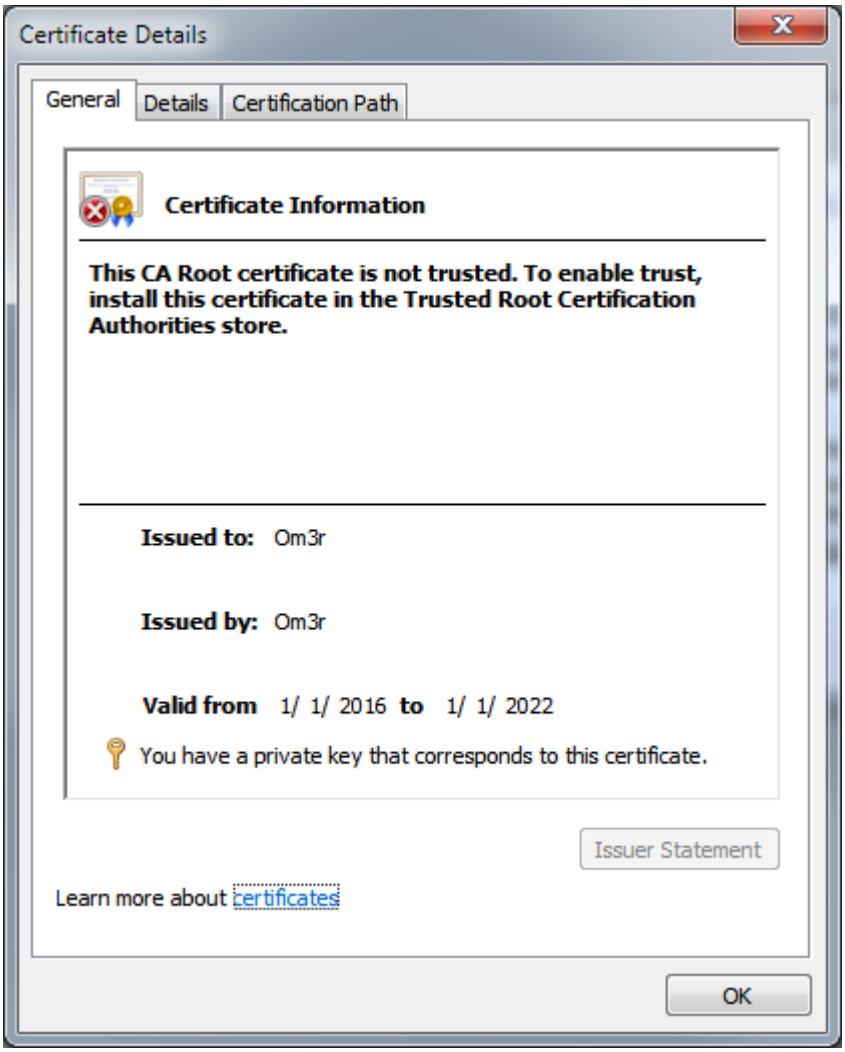
すべてがうまくいけば、がされます



SELFCERTウィザードをじて、したにをけることができます。

したをしようとしたときにそのプロパティをすると



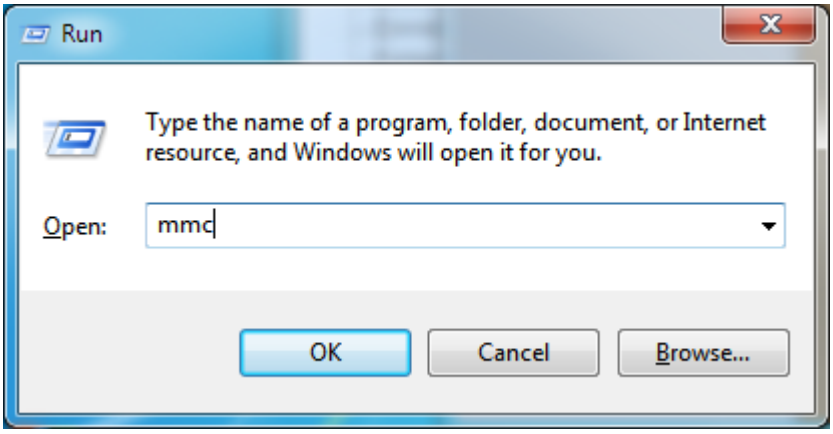


がされておらず、そのがダイアログにされます。

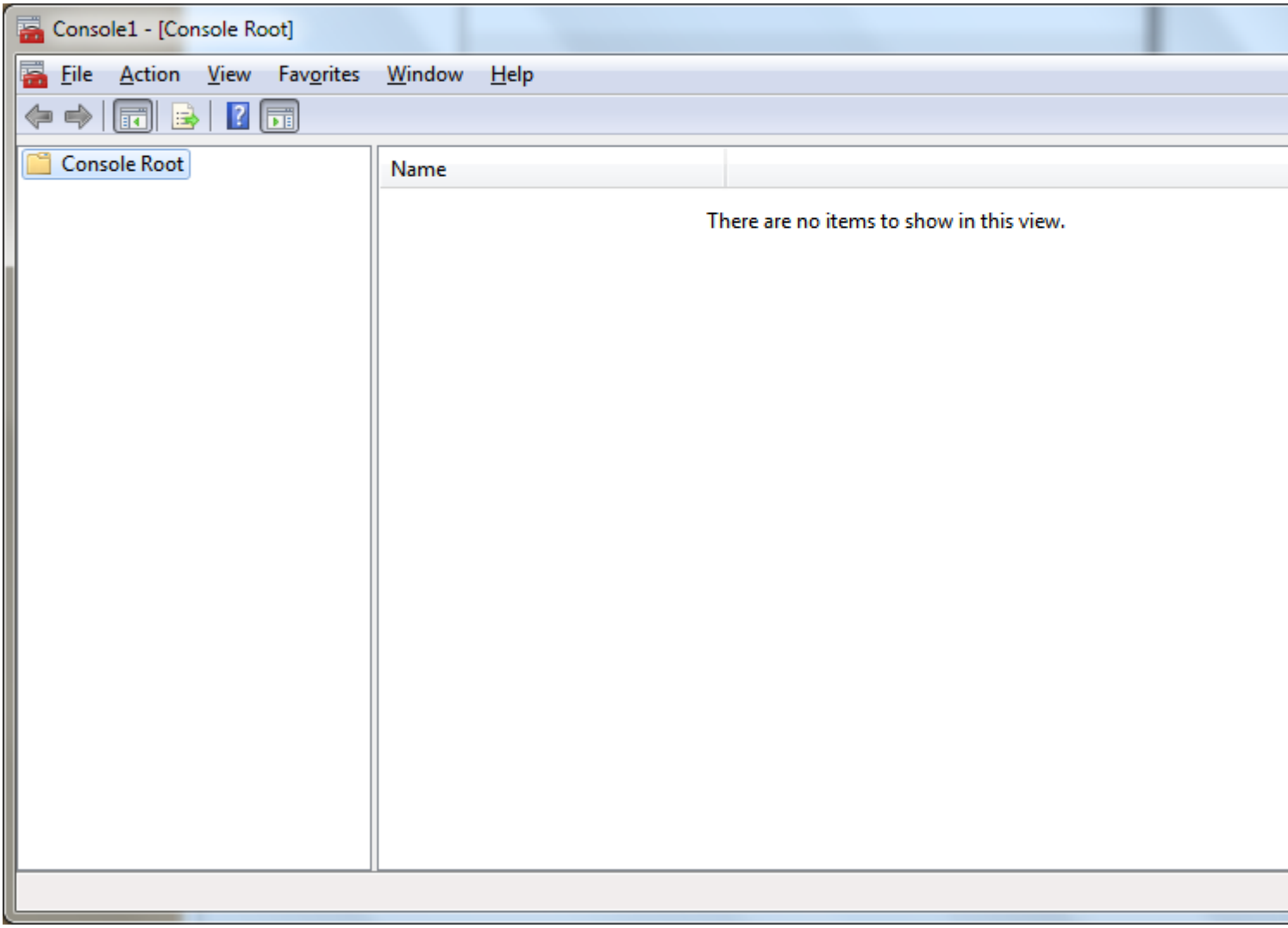
は、のユーザー>>ストアにされています。ローカルコンピュータ>されたルート>ストアにするがあるため、からエクスポートしてにインポートするがあります。

Windows キー+ Rをすと、「」ウィンドウがきます。ウィンドウに「mmc」とし、「OK」をクリ

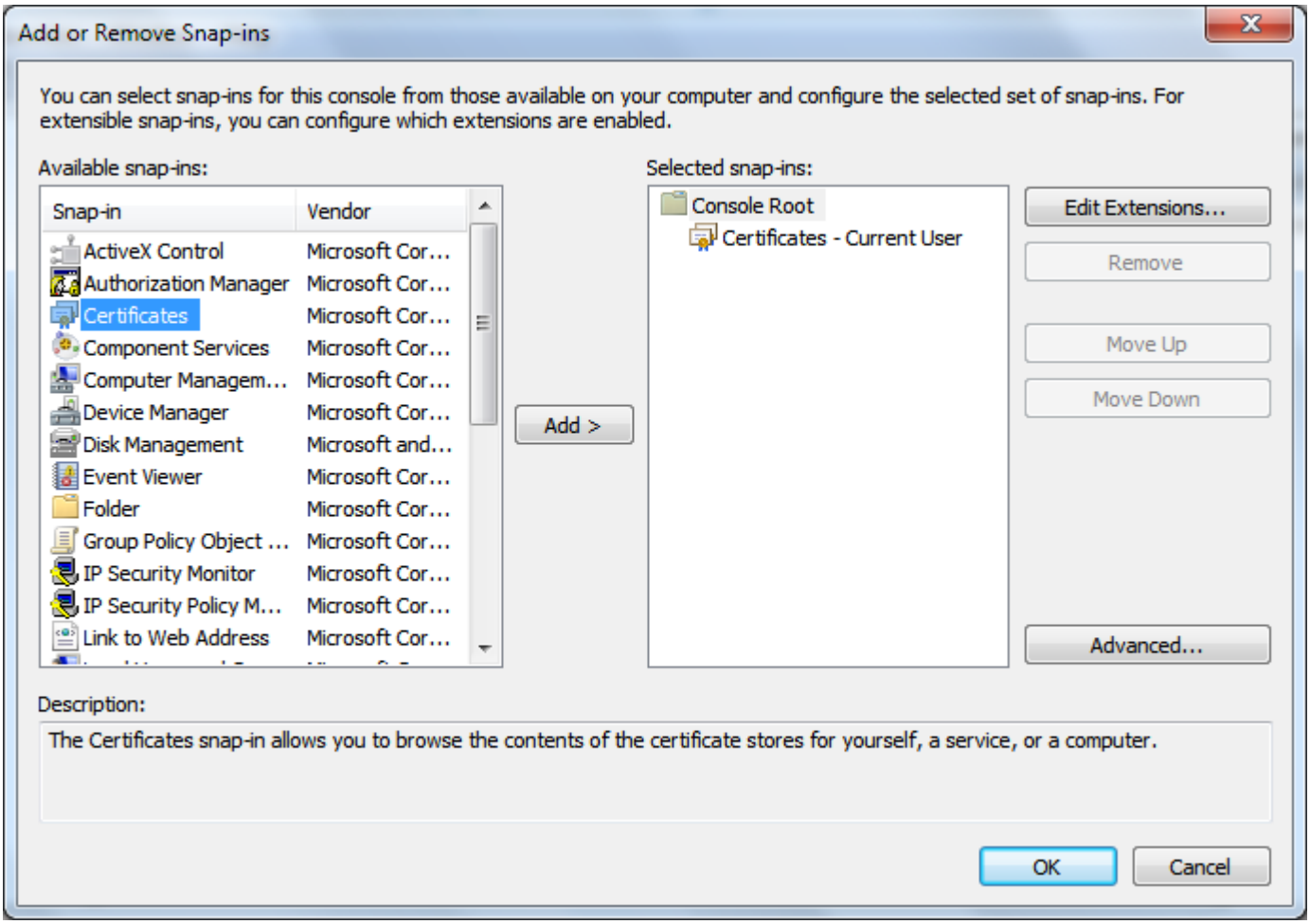
ックします。



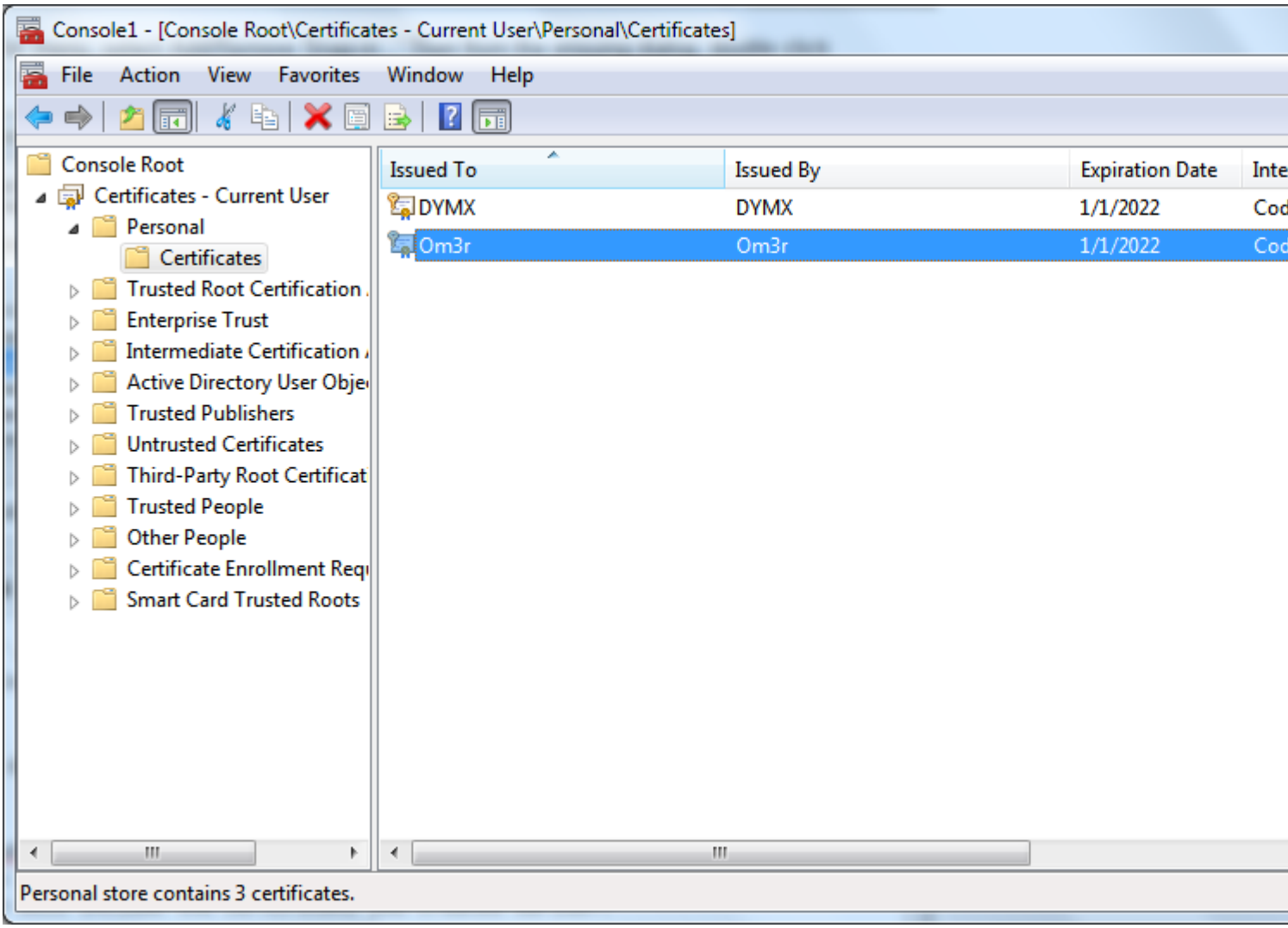
Microsoftコンソールがき、のようにされます。



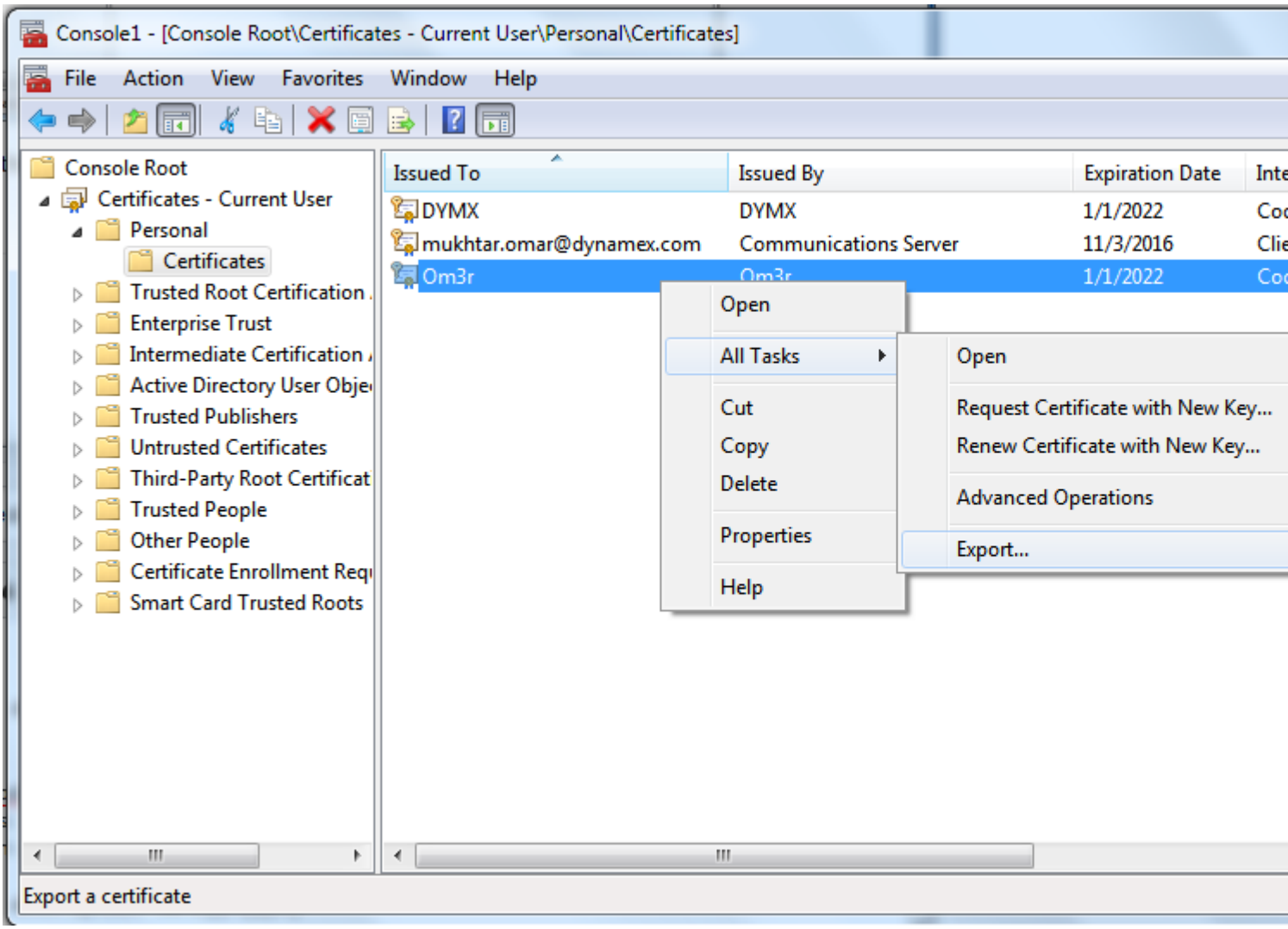
[ファイル]メニューの[スナップインのと]をし、いてダイアログボックスで[]をダブルクリックし、[OK]をクリックします



[- のユーザー]ののウィンドウでドロップダウンリストをし、のようにをします。センターパネルには、そのにがされます。これには、にしたがまれます。



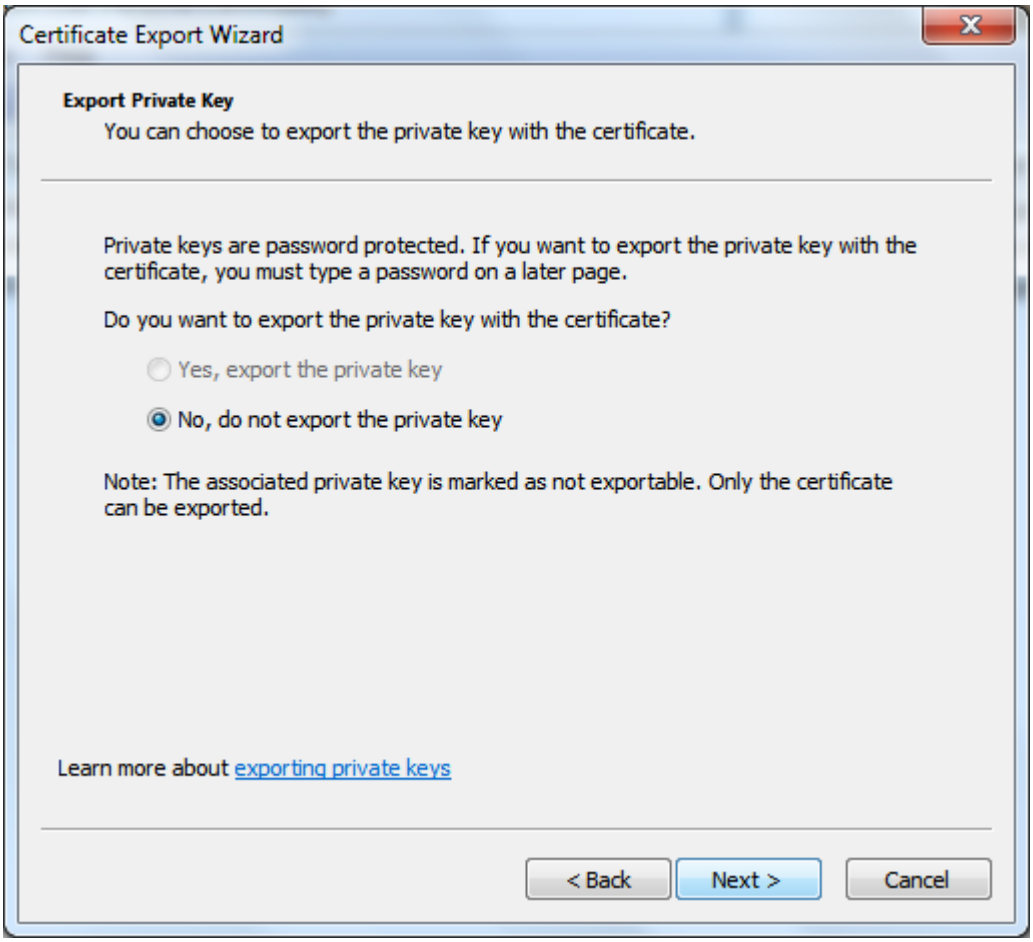
をクリックし、[すべてのタスク]> [エクスポート



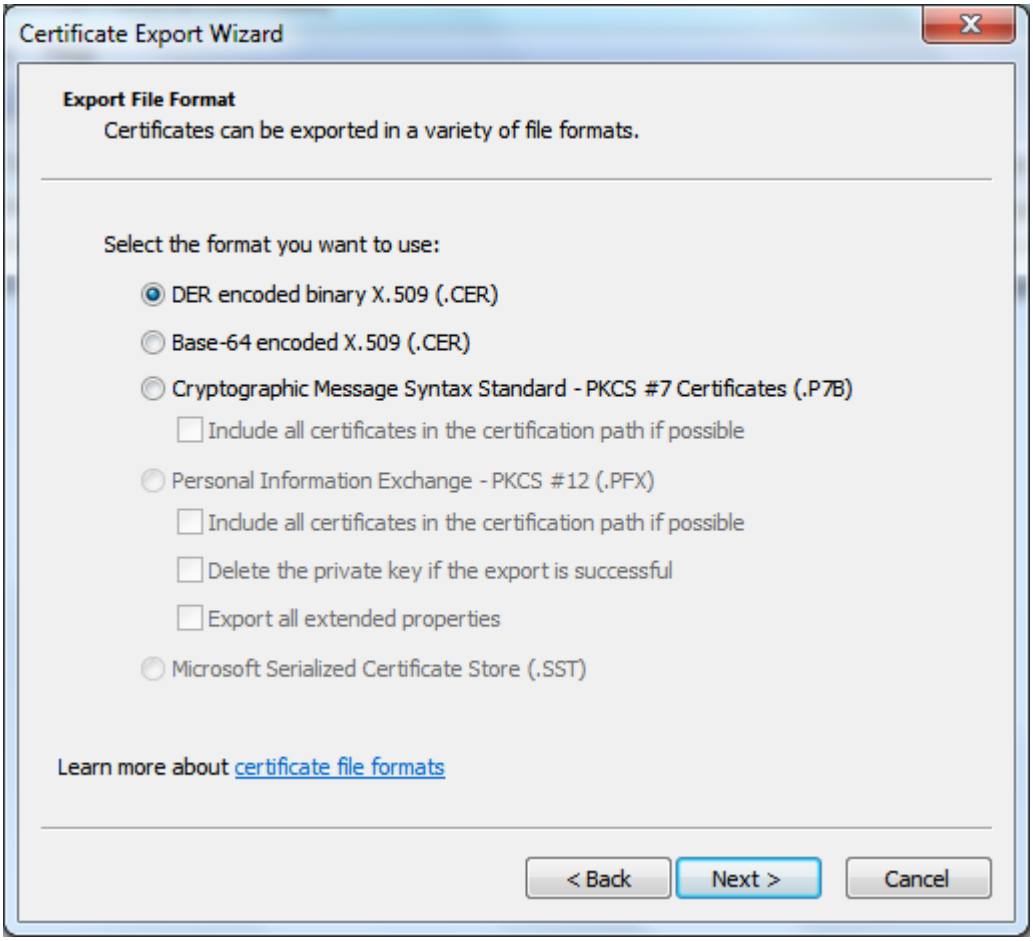
エクスポートウィザード


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

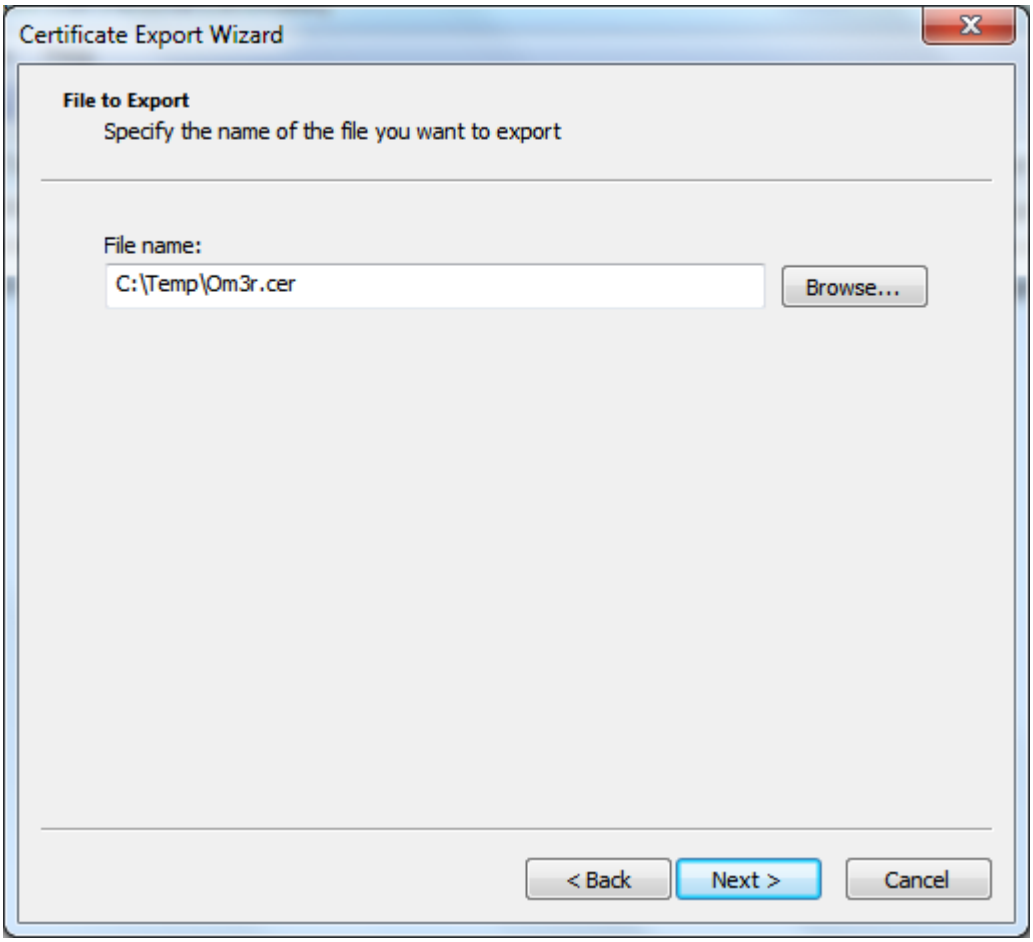
へをクリックします


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

[あらかじめされた1つのみ]オプションがされますので、もう[へ]をクリックします。



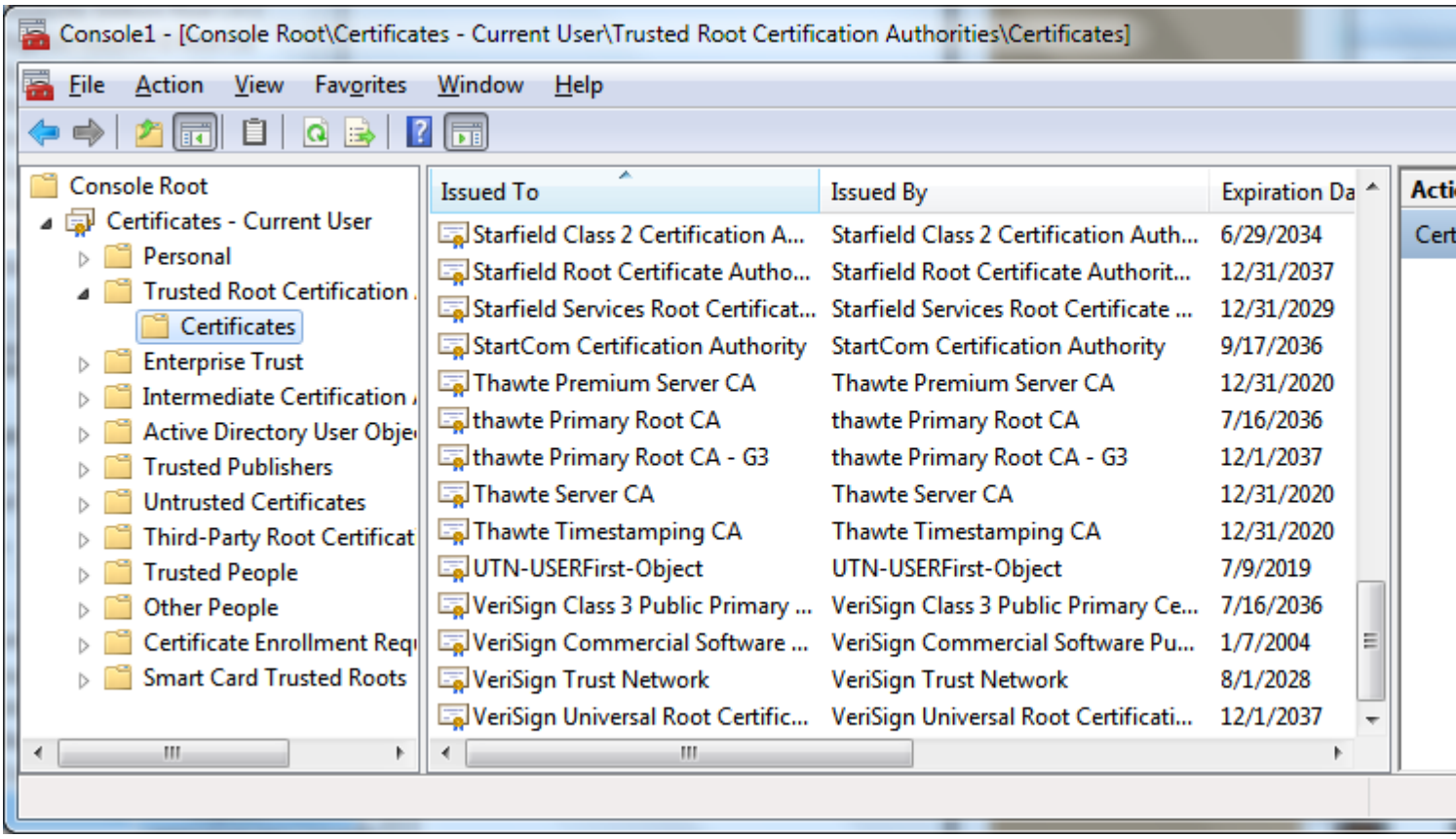
のアイテムはあらかじめされています。 [へ]をクリックし、エクスポートされたをするとをします。



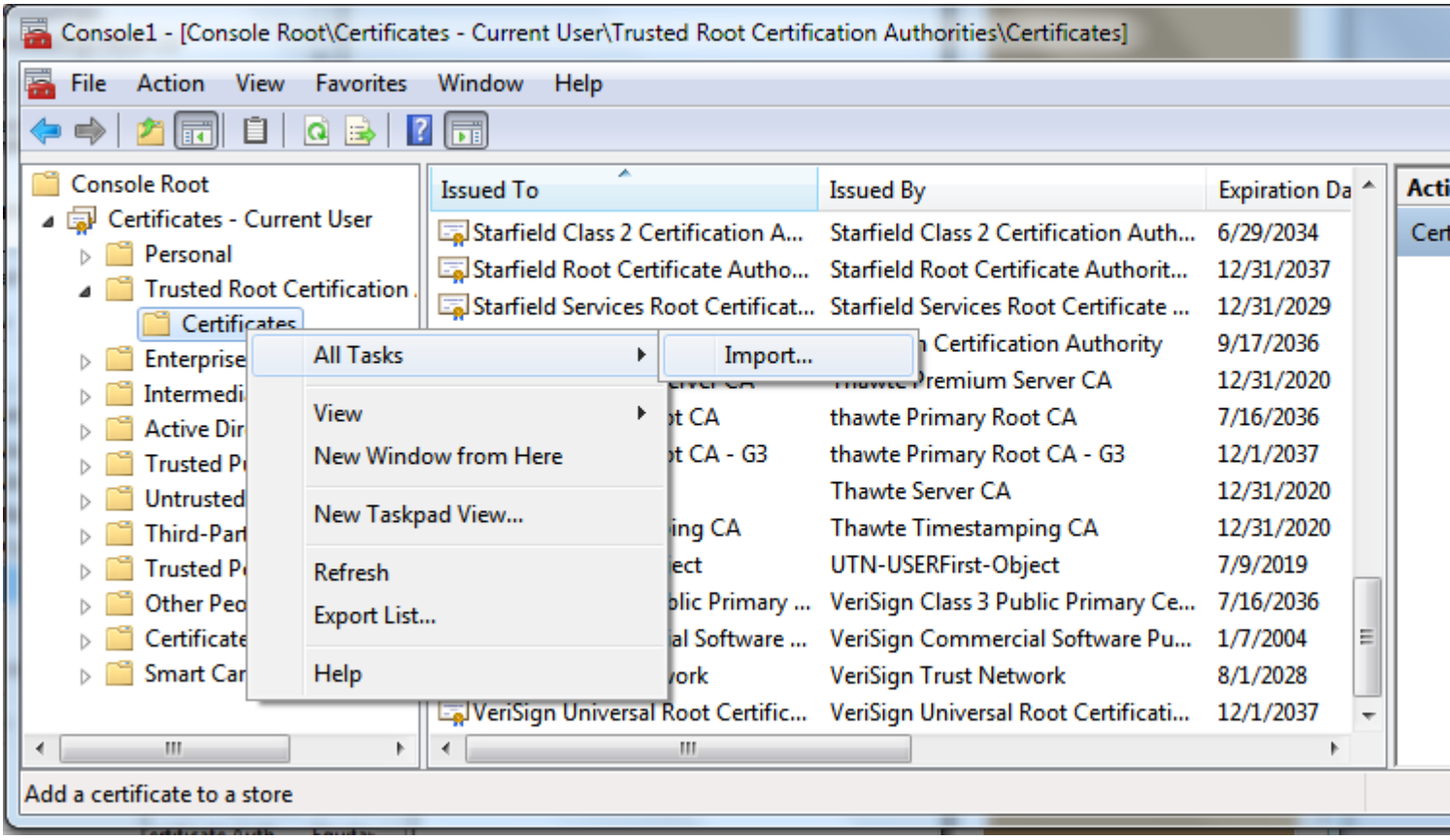
もう[Next]をクリックしてをします

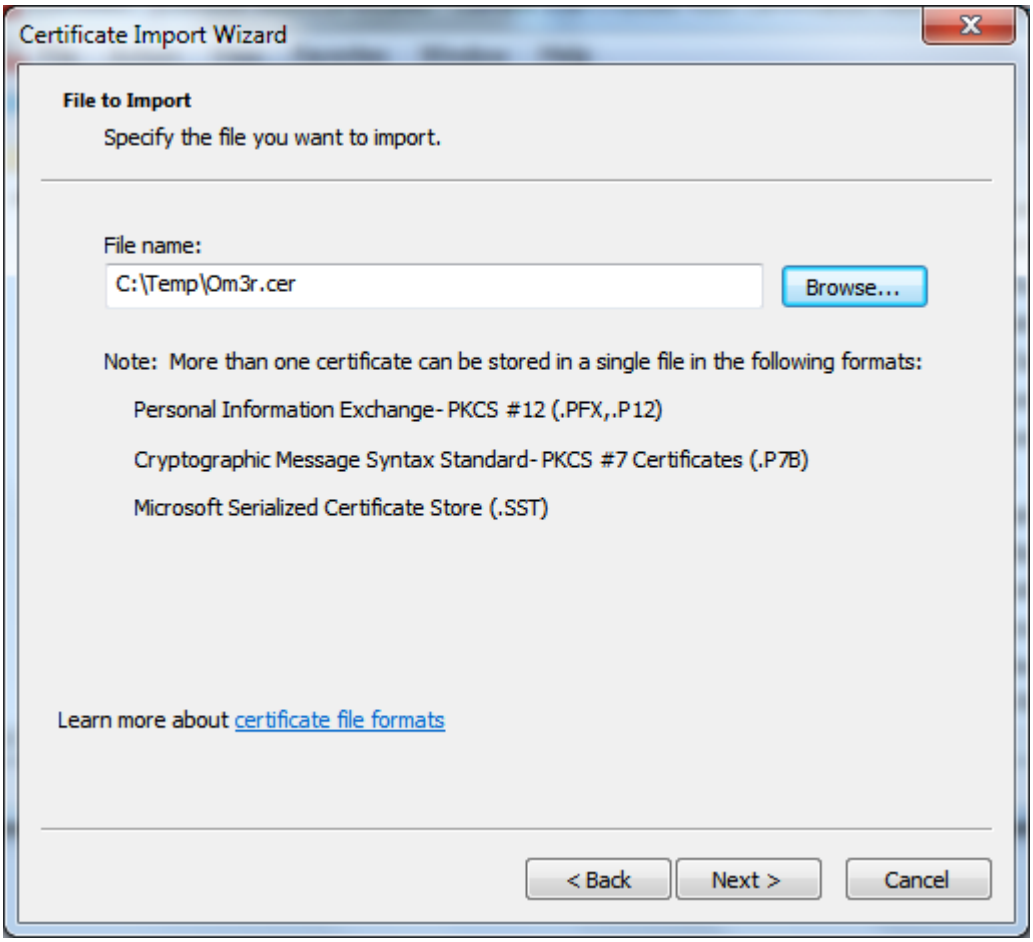
フォーカスがコンソールにされます。

[ ]メニューをし、[されたルート]メニューから[ ]をします。

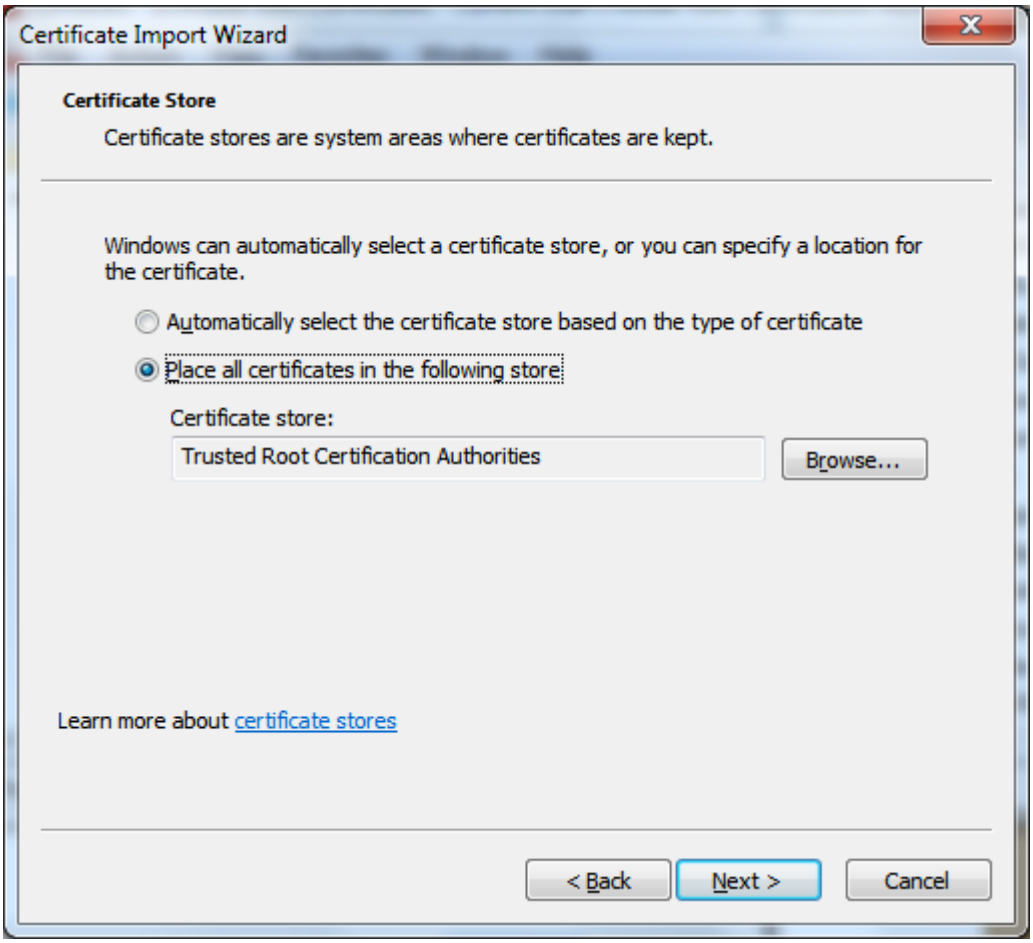


クリック。 すべてのタスクとインポートを



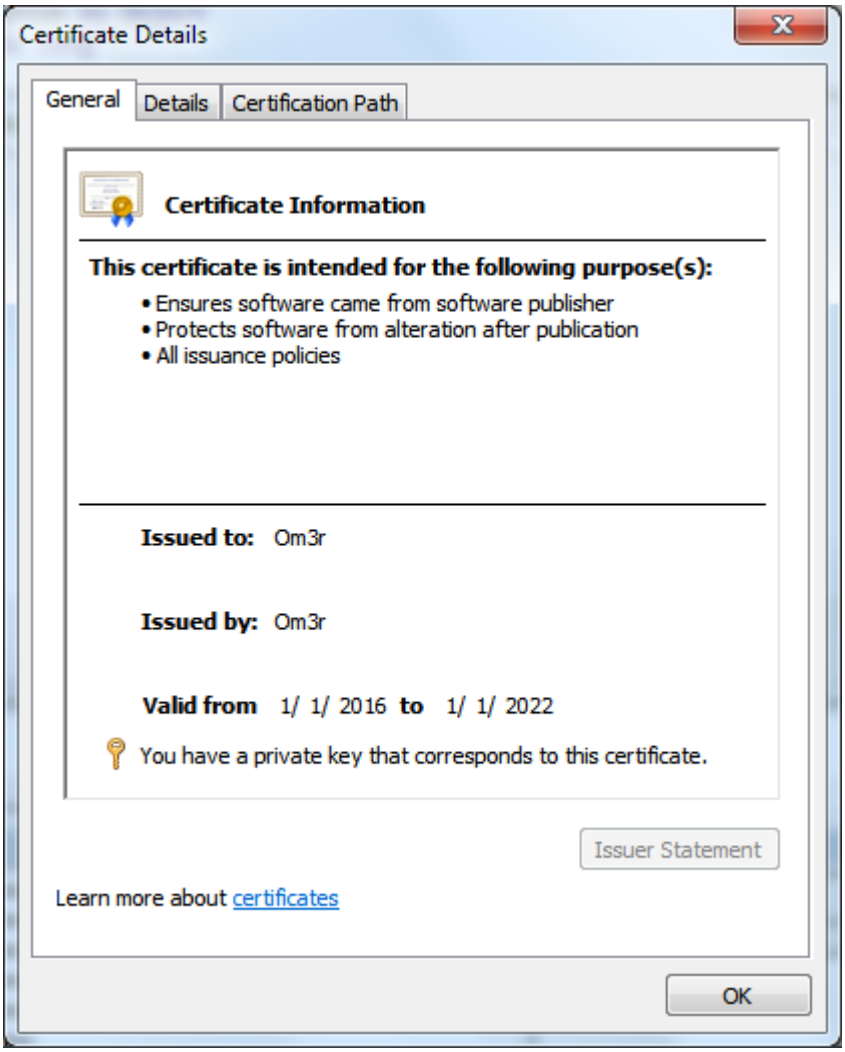


[へ]をクリックし、[されたルート]ストアにします。



[ヘ]> []のにし、コンソールをじます。

をしてそのプロパティをすると、ができるであることがわかります。このをしてプロジェクトにできます。



オンラインでマクロセキュリティとVBAプロジェクト/モジュールのをむ
https://riptutorial.com/ja/vba/topic/7733/マクロセキュリティとvbaプロジェクト-モジュールの

26: ユーザーフォーム

Examples

ベストプラクティス

`UserForm`は、デザイナーとデフォルトのインスタンスをつクラスモジュールです。 コードビハインドをながら`Shift` + `F7`をすとデザイナーにアクセスでき、 デザイナーをながら`F7`キーをすとコードビハインドにアクセスできます。

しいインスタンスでしてください。

クラスモジュールなので、フォームはオブジェクトの です。フォームはとデータをできるので、デフォルト/グローバルなインスタンスではなく、クラスのしいインスタンスでするがいです。

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then
        '...
    End If
End With
```

のわりに

```
UserForm1.Show vbModal
If Not UserForm1.IsCancelled Then
    '...
End If
```

デフォルトのインスタンスをすると、フォームがの "X"ボタンでじられているやコードビハインドで`Unload Me`がされているになバグがするがあります。

ロジックをのにします。

フォームは、 プレゼンテーションだけにするがあります。データベースにし、ユーザーのについてパラメータされたクエリをするボタン`Click`ハンドラは、あまりにもくのことをしています。

わりにのモジュールとプロシージャでフォームをするコードのアプリケーションロジックをしてください。

ユーザーフォームがデータのとのをっているをうようなでコードをします。データがどこからたのか、でデータがするかはではありません。

びしはコントロールをにするべきではありません。

のクラスモジュールで、またはフォームのコードビハインドでカプセルして、フォームのなモデルをし、 `Property Get`プロシージャをしてモデルをし、クライアントコードをこれらとさせます。コントロールとなをし、するデータのみをクライアントコードにします。

これはのようなコードをします

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then
        MsgBox .Message, vbInformation
    End If
End With
```

これのわりに

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then
        MsgBox .txtMessage.Text, vbInformation
    End If
End With
```

QueryCloseイベントをします。

フォームには`Close`ボタンがあり、プロンプト/ダイアログには`OK`ボタンと`Cancel`ボタンがあります。フォームのコントロールボックス の "X"ボタンをしてフォームをじることができます。これはデフォルトでフォームインスタンスをします しいインスタンスをするの。

```
With New UserForm1
    .Show vbModal
    If Not .IsCancelled Then 'if QueryClose isn't handled, this can raise a runtime error.
        '...
    End With
End With
```

`QueryClose`イベントをするもなは、 `Cancel`パラメータを`True`にし、フォームをじるわりににすることです。

```
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    Cancel = True
    Me.Hide
End Sub
```

そうすれば、 "X"ボタンはインスタンスをすることはなく、びしはすべてのパブリックメンバーににアクセスできます。

す、じないでください。

オブジェクトをするコードは、オブジェクトをするがあります。オブジェクトをアンロードしてするのはフォームのではありません。

フォームのコードビハインドで`Unload Me`をしないでください。 Call `Me.Hide`わりに、フォームがしたときにしたオブジェクトをびしのコードでできるようにします。

にをつける。

プロパティのツールウィンドウ `F4` をして、フォームのコントロールのをにします。コントロールのはコードビハインドでされていますので、これをできるリファクタリングツールをしていないり、コントロールのをするとコードがれてしまいます。まずにしてみるほうがです`TextBox12`がする20のテキストボックスのうちのどれをにパズルするか。

、UserFormコントロールには、ハンガリーのがけられています。

- ユーザーをす`Label`コントロールの`lblUserName`。
- ユーザーがユーザーをできる`TextBox`コントロールの`txtUserName`。
- ユーザーがユーザーをまたはできる`ComboBox`コントロールの`cboUserName`。
- ユーザーがユーザーをできる`ListBox`コントロールの`lstUserName`。
- `btnOk`または「OK」とラベルけされた`Button`コントロールの`cmdOk`。

は、えばUIをし、したときにということである`ComboBox`への`ListBox` - デカップリングすることは、らがをすかのためではなく、そのコントロールタイプのにコントロールをするといでしょう、はしいコントロールタイプをするようにするがありますなりUIからのコード。

- `UserNameLabel`は、ユーザーをすみりラベルです。
- ユーザーがユーザーをまたはできるコントロールの`UserNameInput`。
- "OK"とされたコマンドボタンの`OkButton`

いずれのスタイルをしても、すべてのコントロールをデフォルトのにするよりもれたものがあります。スタイルのもです。

QueryCloseの

`QueryClose`イベントは、フォームが`QueryClose`とするたびにします。 `CloseMode`パラメータには、フォームがどのようにじられたかをす`VbQueryClose`がまれます。

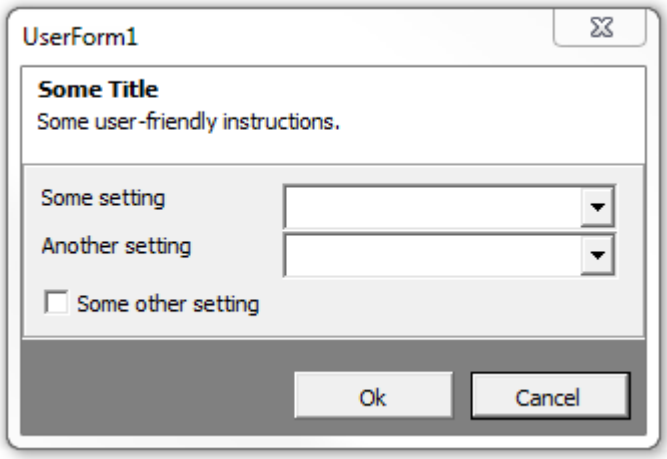
`vbFormControlMenu`	フォームはユーザーのにじてじる	0
`vbFormCode`	フォームが`Unload`ステートメントにしてじています	1
`vbAppWindows`	Windowsセッションがしています	2

vbAppTaskManager	Windowsタスクマネージャがホストアプリケーションをしています	3
vbFormMDIForm	VBAではサポートされていません	4

みやすくするために、をするのではなく、これらのをすることをおめします。

Cancellable UserForm

キャンセルボタンがいたフォームがえられた



フォームのコードビハインドはのようになります。

```
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
```

びしコーードはフォームをし、キャンセルされたかどうかをることができます

```
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
```

`IsCancelled`プロパティは、[キャンセル]ボタンがクリックされたとき、またはユーザーがコントロールボックスをしてフォームをじるときに`True`します 。

オンラインでユーザーフォームをむ https://riptutorial.com/ja/vba/topic/5351/ユーザーフォーム

27: べえ

き

.NET Frameworkとはなり、Visual Basic for Applicationsライブラリにはをべえるルーチンはまれていません。

には、1ソートアルゴリズムをゼロからする、または2のになライブラリでソートルーチンをするの2があります。

Examples

アルゴリズムの - 1のクイックソート

VBAのソートからですか

```
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ライブラリをした1のべえ

このコードは、Microsoft Excel Object LibraryのSortクラスをしています。

は、をしてください。

- をにコピーする
- したをされたにコピーするは

```
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
```

オンラインでべえをむ https://riptutorial.com/ja/vba/topic/8836/べえ

28: のをにする

VBAは、にじていくつかのをにし、プログラマでなをすることなく、VBAはなをいくつかします。

もよくされるのうち3つは、 `CStr` 、 `Format` 、 `StrConv`です。

Examples

CStrをしてをにする

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

フォーマットをしてタイプをとしておよびする

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

StrConvをして、1バイトのバイトをにする

```
'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"
```

マルチバイトのバイトをににする

```
'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"
```

オンラインでのをにするをむ https://riptutorial.com/ja/vba/topic/3467/のをにする

29:

き

をびすはであるとわれています。なロジックは、しばしばルーープとしてすることもできます。コーールスタックのとをいつするかをがるように、はパラメーータでするがあります。 によって、にランタイムエラー '28'がします。

をしてください。

では、プロシーージャのりしびしをすることができます。

Examples

ファクトリアル

```
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
```

フォルダ

Early Bound `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
```

オンラインでをむ https://riptutorial.com/ja/vba/topic/3236/

30:

Examples

はデ―タをします。 デ―タやスコ―プのではなく 、 をして、されているもののをけます。あなたの `thing1, thing2, thing3` にをけることをいられたら、そのわりになデ―タ、 `Collection` 、 `Dictionary` をうことをしてください。

なのをすの - えば、、 `Collection` 、 `Dictionary` 、またはセルの`Range`は、でなければなりません。

いくつかのなVBAはのようになります。

プロシ―ジャ·レベルのの

`camelCase`

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

    Dim procedureVariable As Long
    Dim someOtherVariable As String

End Sub
```

モジュ―ルレベルのの

`PascalCase`

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

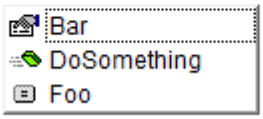
の

`SHOUTY_SNAKE_CASE`は、とをするためによくされます。

```
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
```

ただし、 `PascalCase`はより`PascalCase`コ―ドになり、IntelliSenseではとになるアイコンがされるため、

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

Sub DoSomething()
    Module1.
End Sub
```



ハンガリー

データまたはスコープのではなく 、したにをけます。

「ハンガリーのは、のがどのようなものであるかをにできます。

プロシージャのようなコードを なるのにさせてくならば、プロシージャのにあるのをしてはいけません。なりのにいをし、なですると、そのデータはににになります。 VBEの`Ctrl` + `i`ショートカットをして、のタイプをツールチップにすることもできます。

どのようながされるかは、データよりはるかになです。 に VBAなどのでは、にじてんでにをのにします。

のでは、 `iFile`と`strFile`をします。

```
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
```

する

```
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
```

のでは、 `strData` `ByRef`がされていますが、 にされていることをできるだけのなことのほかに、`strData`がによってにされることはありません。

のは、それを`outContent`というにし`outContent` 。 この`out` 、はらかに「アウト」のパラメータとしてそれをするために、このには、 のためにされているものをしやすくするためにプレフィックスがためにハンガリアンがされたものです。

これは、パラメータがによってにされたでも、IntelliSenseが`ByRef`しないためにです。

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

それはにつながる...

ハンガリーはにった

ハンガリーのはもともととはありませんでした 。には、 にわハンガリアンはにです。このな`ByVal`と`As Integer`をにするためにをえてみましょう。

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

する

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

`src`と`dst`はここではハンガリーので、パラメータやIntelliSenseからできないなをえます。

もちろん、なとのをって、でしてをすことができる、すべてをえるよりいがあります - なとして

```
Type Coordinate
    RowIndex As Long
```

```
    ColumnIndex As Long
End Type

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

プロシ―ジャ

きはかをする。 をって、らがをしているのかをしてください。プロシ―ジャのをにすることがな、プロシ―ジャはあまりにもくのことをしており、よりさく、よりなプロシ―ジャにするがあります。

いくつかのなVBAはのようになります。

すべてのについて

```
PascalCase
```

```
Public Sub DoThing()

End Sub

Private Function ReturnSomeValue() As [DataType]

End Function
```

イベントハンドラプロシ―ジャの

```
ObjectName_EventName
```

```
Public Sub Workbook_Open()

End Sub

Public Sub Button1_Click()

End Sub
```

イベントハンドラはVBEによってにされます。オブジェクトやハンドルされたイベントのをせずにをすると、コ―ドがされコンパイルされますが、ハンドラプロシ―ジャはしてされることはありません。

ブ―ルメンバ―

ブ―ルをすことをえてみましょう

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

する

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

Canプレフィックスは 、bとじをたします。これは、のりをBooleanでします。 しかし、 CanはbよりむCanます

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

にべ

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

ブールをすメンバやプロパティのにCan 、 Is 、 Hasなどのをすることをし Can 、をするにのみしてください。これは、 のMicrosoftのにしています 。

オンラインでをむ https://riptutorial.com/ja/vba/topic/1184/

31: の

Examples

かつな

コードモジュールにモジュールのに`Option Explicit`がまれていない、コンパイラはににをします。それらは、デフォルトで`VariantVariant`ます。

```
Public Sub ExampleDeclaration()

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

End Sub
```

のコードで`Option Explicit`がされている、 `someVariable`および`someOtherVariable`な`Dim`ステートメントがないため、コードがします。

```
Option Explicit

Public Sub ExampleDeclaration()

    Dim someVariable As Long
    someVariable = 10

    Dim someOtherVariable As String
    someOtherVariable = "Hello World"

End Sub
```

コードモジュールでOption Explicitをして、すべてのをにすることをおめします。

このオプションをデフォルトでするについては、 VBAのベストプラクティスをしてください。

をすることができますレベルをげる

- プロシージャ・レベルでは、のプロシージャで`Dim`キーワードをします。 ローカル 。
- モジュールレベルでは、どのタイプのモジュールでも`Private`キーワードをします。 プライベートフィールド 。
- インスタンスレベルでは、どのタイプのクラスモジュールでも`Friend`キーワードをします。のフィールド 。
- インスタンス・レベルでは、どのタイプのクラス・モジュールでも`Public`キーワードをします。 パブリックフィールド
- モジュールで`Public`キーワードをしてグローバルにグローバル 。

はにのスコープでするがあります。グローバルをするのではなく、プロシージャにパラメータを

すことをします。

については、 アクセスをしてください。

ロ―カル

`Dim`キ―ワ―ドをしてロ―カルをします 。

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

の`[As Type]`はオプションです。これをすると、のデ―タがされ、そのにりてられるメモリがまります。これは`String`をします

```
Dim identifierName As String
```

がされていない、はに`Variant`

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

VBAでは、1つのステ―トメントでのをすることもできます。

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

[Variant]にして`[As Type]`をするがあることにしてください。これはなトラップです

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

ロ―カルは`Static`でも`Static`ません。 VBAでは、 `Static`キ―ワ―ドをして、にプロシ―ジャがびされたときのを「」します。

```
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
```

ここで`values`コレクションは`Static`ロ―カルとしてされています。これはオブジェクトであるため、 `Nothing`されます。にくは、オブジェクトがに`Set`かどうかをします。プロシ―ジャがめてされる、コレクションはされます。 `DoSomethingElse`がアイテムをまたはしているがあります。`DoSomething`がびされたときにコレクションにります。

オルタナティブ

VBAの`Static`キーワードをにすることができます- に 、はのでくかなプログラマーで。くのでは、クラスメンバフィールド、プロパティ、メソッドなどをインスタンスではなくタイプにするために`static`がされています 。 `static`コンテキストのコードは、インスタンスコンテキストのコードをできません。 VBAの`Static`キーワードは、きくうものをします。

しばしば、 `Static`ローカルは、モジュールレベルの`Private`フィールドとしてすることもできますが、をのスコープでするがあります。あなたのをし、きなをう - どちらもうまくいきますが、それがをすることなく`Static`をすると、いバグにつながるがあります。

い

`Dim`キーワードは、プロシージャとモジュールレベルではです。モジュールレベルでのそのは、`Private`キーワードのとじです

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

`Private`キーワードは、モジュールレベルでのみです。これは`Dim`をローカルにし、モジュールを`Private`でします。にパブリックメンバをするためにしなければならない`Public`キーワードをします。わりに`Dim` どこでもしてください。 がです。

"プライベートフィールド"

- しています `Private`モジュールレベルのをします。
- うのですか `Dim`ローカルをします。
- `Dim`をしてモジュールレベルのをしないでください 。

"どこにでもい"

- うのですか `Dim`ローカル/プライベートかをします。
- `Private`をしてモジュールレベルのをしないでください 。
- `Public`フィールドのをける *

*には、とにかく`Public`または`Global`フィールドをしないでください。

フィールド

モジュールのにあるセクションのモジュールレベルでされたは、 フィールドです。 モジュールでされた`Public`フィールドは、 グローバルです 。

```
Public PublicField As Long
```

グローバルスコープのは、されているプロジェクトをするのVBAプロジェクトをめ、どこからで

もアクセスできます。

をグロ―バル/パブリックにするには、プロジェクトからしかることができないようにするには、`Friend`をします。

```
Friend FriendField As Long
```

これは、のVBAプロジェクトがアドインプロジェクトをし、パブリックAPIをできるというアドインではにです。

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

Friendフィ―ルドは、モジュ―ルではできません。

インスタンスフィ―ルド

クラスモジュ―ル `ThisWorkbook`、 `ThisDocument`、 `Worksheet`、 `UserForm`、 クラスモジュ―ルをむののにあるセクションのモジュ―ルレベルでされたは、 インスタンスフィ―ルドです。 インスタンスがするりしますりのクラス。

```
'> 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
```

カプセルフィ―ルド

インスタンスデ―タはしばしば`Private`にたれ、 カプセルされたダビングされます 。プライベ―トフィ―ルドは、 `Property`プロシ―ジャをしてすることができます。びしへのきみアクセスをえずにプライベ―トをするために、クラスモジュ―ルまたはモジュ―ルは`Property Get`メンバをします

```
Option Explicit
Private encapsulated As Long

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

```
End Property

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

クラスはカプセルされたをすることができますが、びしのコードは`Public`メンバーおよびびしがじプロジェクトにあるは`Friend`メンバーにのみアクセスできます。

がをできるようにするには

- カプセルされたであるモジュールは、 `Property Let`メンバーをします。
- カプセルされたオブジェクトで、モジュールは`Property Set`メンバをします。

Const

アプリケーションでしてされないがあるは、きをしてリテラルのわりにできます。

Constは、モジュールまたはプロシージャ･レベルでのみできます。つまり、のコンテキストは、クラス、、モジュール、プロシージャ、またはブロックでなければならず、ソースファイル、、またはインタフェースにすることはできません。

```
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
```

Constantをすることをおめしますが、にはではありません。タイプをしないとしいタイプになります

```
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
```

これはにであり、タイプをしないとはに、バリアントのとなることにしてください。

をにStringとしてすることはですが、のをしてをとしてすることはできません

```
'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"
```

アクセス

`Dim`ステートメントは、ローカルにするがあります。モジュールレベルでは、なアクセスをすることをおめします。

- `Private`フィールドの`Private`。プライベートフィールドは、されたモジュールでしかアクセスできません。
- `Public`フィールドとグローバルのパブリック。のびしコードからアクセスできます。
- プロジェクトのパブリックは`Friend`ですが、のVBAプロジェクトにはアクセスできませんアドインにしています
- `Global`は、モジュールの`Public`フィールドにもできますが、クラスモジュールではです。わりに`Public`をすることをおめします。このは、プロシージャにとってもではありません。

アクセスは、およびプロシージャにもにできます。

```
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
```

オプションプライベートモジュール

パラメータなしモジュールの`Sub`プロシージャは、マクロとしてされ、ホストのコントロールおよびキーボードショートカットにアタッチできます。

に、モジュールのpublic `Function`プロシージャは、ホストアプリケーションのユーザUDFとしてされます。

モジュールのに`Option Private Module`をすると、メンバーがマクロおよびUDFとしてホストアプリケーションにされなくなります。

タイプヒント

タイプヒントはあまりおめできません。それらはし、およびのからここにされています。わりに`As [DataType]`をするがあり`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
```

タイプヒントはにコーードのをさせ、のハンガリアン もみやすさをげます。

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

わりに、にいをし、したもののをのではなくにします。

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

ヒントは、のをするためにリテラルでもできます。デフォルトでは、32,768よりさいリテラルは`Integer`リテラルとしてされますが、ヒントタイプをしてそれをできます。

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

ヒントは、なでされたにされるか、またはパラメータとしてされたときににになにされるため、はリテラルにはありません。なは、なの1つをしてできます。

```
'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)
```

をすみみ

をするみみのは、 `Variant`をすやかにけされたバージョンと、 `String`をすにされたバージョン`$`わるの2つのバージョンがあります。りを`Variant`にしているをき、 `String`をすバージョンをふがあります。そうでなければ、りののがあります。

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

これらのはのとおりです。

- 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 $

これらはのエイリアスであり、ヒントではないことにしてください。 `Left`はれた`B_Var_Left`にし、 `Left$`はれた`B_Str_Left`にします。

VBAのにのバ―ジョンでは、 `$`はされたではなく、はでむがありました。 Word Basicでは、$でわったをすがくありました。

の

VBAでは、はのさでできます。されているさにされるようににパディングまたはりてられます。

```
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
```

をする

ローカルでされたStaticはされず、Subプロシージャがしたときにがわれません。それのプロシージャのびしでは、やりてはありませんが、されたは「0」にすることもできます。

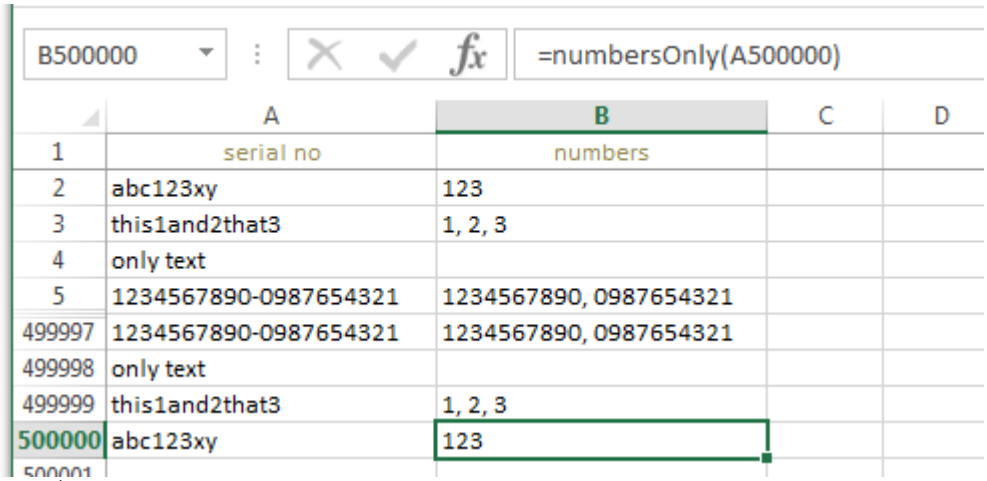
これらは、りしびされる'ヘルパー'サブオブジェクトにレイトバインドするときににです。

スニペット1くのワークシートにわたるScripting.Dictionaryオブジェクトの

```
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
```

スニペット2 VBScript.RegExpオブジェクトをでバインドするワークシートUDFをする

```
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
```



50にってオブジェクトをめんだUDFの

* UDFで500Kのをたす
    - **Dim rgx As**オブジェクト 148.74
    - スタティック**rgx**オブジェクト 26.07

* これらはのみをしてください。あなたのはさと
されるの。

ワークブックのはUDFはもされません。UDFであっても、するのがされるたびにされます。のイ

ベントがするたびに、にされたのがえます。

- は、されりてられたプロシ―ジャまたはではなく、モジュ―ルのにできます。
- はロ―カルでのみできます。
- は、プライベ―トモジュ―ルレベルのじプロパティのくをしますが、スコ―プはされています。

Visual Basic

オンラインでのをむ https://riptutorial.com/ja/vba/topic/877/の

32:

- VB_Name = "ClassOrModuleName"
- VB_GlobalNameSpace = False 'される
- VB_Creatable = False 'される
- VB_PredeclaredId = {True | False}
- VB_Exposed = {True | False}
- variableName.VB_VarUserMemId = 0 'ゼロは、これがクラスのデフォルトメンバーであることをします。
- variableName.VB_VarDescription = "some string" 'こののオブジェクトブラウザにテキストをします。
- procName.VB_Description = "some string" 'プロシージャのオブジェクトブラウザにテキストをします。
- procName.VB_UserMemId = {0 | -4}
  - '0をクラスのデフォルトメンバーにします。
  - '-4がをすことをします。

Examples

VB_Name

VB_Nameは、クラスまたはモジュールをします。

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

このクラスのしいインスタンスは、

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

VB_GlobalNameSpace

VBAでは、このはされます。それはVB6からされていませんでした。

VB6では、クラスのデフォルトグローバルインスタンスショートカットをし、クラスメンバにクラスをせずにアクセスできるようにします。たとえば、 `DateTime DateTime.Now` は、には `VBA.Conversion`クラスのです。

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

VB_Createable

このはされます。それはVB6からされていませんでした。

VB6では、 `VB_Exposed`とみわせてされ、のプロジェクトのにあるクラスのアクセシビリティをします。

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

のプロジェクトからアクセスできる`Public Class`になりますが、このはVBAにはしません。

VB_PredeclaredId

クラスのグローバルなデフォルトインスタンスをします。デフォルトのインスタンスには、クラスのでアクセスします。

```
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
```

コール

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

いくつかので、これはののクラスのをシミュレートしますが、のとはなり、クラスのインスタンスをすることはできます。

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

VB_Exposed

クラスのインスタンスのをします。

```
Attribute VB_Exposed = False
```

クラスを`Private`ます。のプロジェクトのにはアクセスできません。

```
Attribute VB_Exposed = True
```

プロジェクトのでPublicクラスをPublicします。ただし、VBAではVB_Createableがされるため、クラスのインスタンスをすることはできません。これは、のVB.Netクラスにします。

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

プロジェクトのからインスタンスをするには、ファクトリをしてインスタンスをするがあります。これをう1つのは、のPublicモジュールをすることです。

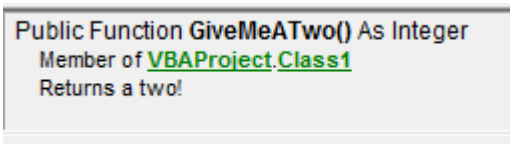
```
Public Function CreateFoo() As Foo
    CreateFoo = New Foo
End Function
```

パブリックモジュールはのプロジェクトからアクセスであるため、 `Public - Not Createable`クラスのしいインスタンスをすることができます。

VB_Description

オブジェクトエクスプローラでされるクラスまたはモジュールメンバーにテキストのをします。には、パブリックインターフェイス/ APIのすべてのパブリックメンバーにがです。

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



プロパティのすべてのアクセサメンバー `Get` 、 `Let` 、 `Set` はじをします。

VB_ [Var] UserMemId

`VB_VarUserMemId` モジュールスコープと`VB_UserMemId` プロシージャは、に2つののためにVBAでされます。

クラスのデフォルトメンバーの

`Collection`をカプセルする`List`クラスは、 `Item`プロパティをつことをんでいるので、クライアントコードはこれをうことができます

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

しかし、 `Item`プロパティで`VB_UserMemId`が0にされていると、クライアントコードでこれをうことができます。

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

のクラスでは、1つのメンバーだけが`VB_UserMemId = 0`をつことができます。プロパティのは、 `Get`アクセサでをします。

```
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`ループでクラスをにする

マジックのが`-4`、 `VB_UserMemId`は、このメンバーがをすることをVBAに`VB_UserMemId`ます。これにより、クライアントコードでこれをうことができます。

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

このメソッドをするもなは、/カプセルされた`Collection` hidden `[_NewEnum]`プロパティゲッター—をびすことです。のアンダー—スコアがなVBAになるため、をでむがあります。

```
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
```

オンラインでをむ https://riptutorial.com/ja/vba/topic/5321/

33: ByRefまたはByValをす

き

ByRefおよびByValは、プロシ―ジャのシグネチャのであり、がプロシ―ジャにされるをします。VBAでは、にされていないり、 ByRefがされます ByRefは、しないはです。

のくのプログラミングVB.NETをむでは、がされていない、パラメ―タはにされます。をけるためにByRefをにすることをしてください。

をす

はしするがあります。このコ―ドはコンパイルされますが、エラ―424 "Object Required"がします。

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

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

このコ―ドはコンパイルされません

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

Examples

なByRefとByValをす

ByRefまたはByValののがされているかどうかをすCalledProcedureによってCallingProcedure 、またはいくつかののののポインタとばれるのがされているかどうかをCalledProcedure。

がされたByRef 、のメモリアドレスがされCalledProcedureとすることにより、そのパラメ―タにらかのCalledProcedureのにわれるCallingProcedure。

にByValがされた、へのではなくのがCalledProcedureされます。

なでこれをにします

```
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
```

もうつの

```
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

デフォルト

パラメーターにがされていない、そのパラメーターはにしされます。

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

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

`foo`パラメータは、 `DoSomething1`と`DoSomething2`で`ByRef`にされます。

をけてあなたがののをつVBAにているなら、これはあなたがれしんだものとまったくのをするがにいです。のくのプログラミングVB.NETをむでは、implicit / defaultはによってパラメータをします。

し

- が`ByRef`すと、プロシージャはそのへのをけります。

```
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
```

の`Test`プロシージャをびすと、84がされます`DoSomething`は`foo`がえられ、そのへのをけり、びしとじメモリーアドレスでします。

- `ByRef` しされると、プロシージャはポインタへのをけります。

```
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
```

`DoSomething`にオブジェクトポインタへのがされ、されるに`Nothing`にりてられているため、びしがしなくなったオブジェクトの`Count`メンバをびしているため、のコードではエラー91がします 。

びしサイトで**ByVal**をする

コールサイトでカッコをすると、 `ByRef`をオーバーライドして、に`ByVal`をにすことができます。

```
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
```

のコードは、 ByRefがにされているかにされているかにかかわらず、42をします。

をけてこのため、プロシージャーびしでなカッコをすると、にバグがするがあります。きとリストののにしてください

```
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

し

- がByValさByValと、プロシージャはのコピーをけります。

```
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
```

のTestプロシージャーをびすと、42がされますDoSomethingはfooがえられ、のコピーをけります。コピーは2され、プロシージャがするとされます。のコピーはしてされませんでした。

- がByValさByValと、プロシージャはポインタのコピーをけります。

```
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
```

のTestプロシージャをびすと、1がされますDoSomethingはfooがえられ、 Collectionオブジェ

クトへのポインタの コピーをけります。 `Test`スコープの`foo`オブジェクトがじオブジェクトをしているため、 `DoSomething`にをすると、そのがじオブジェクトにされます。これはポインタのコピーであるため、を`Nothing`して`Nothing`びしのコピーにはしません。

オンラインでByRefまたはByValをすをむ https://riptutorial.com/ja/vba/topic/7363/byrefまたはbyvalをす

34: のとりて

はであり、ほとんどのプログラミングタスクのです。テキストがであっても、にはがりてられます。は、さゼロまたは2GBまでのさにすることができます。バ―ジョンのVBAは、をにマルチバイトセットバイトのバイトUnicodeのをしてします。

Examples

をする

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

のをする

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

のをしてりてる

```
'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"
```

のとりて

```
'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ステートメントをしてののをりてる

VBAは、のをすためのMidをしますが、にまたは々のをりてるためにできるMid ステートメントもします。

Midは、ステートメントのまたはにされますが、、 Midステートメントはステートメントのにされます。

```
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
```

マルチバイトセットにするのをのわりに、の々ののの々のバイトにりてるがあるは、 MidBステートメントができます。このでは、 MidBステートメントの2のは、きえがされるバイトの1からまるであるため、のとなはMidB(surname, 5, 2) = "y"ます。

バイトとのやりり

ストリングはバイトにりてられ、もです。はマルチバイトセットの「」をにされるため、としてられるののすべてのインデックスは、ASCIIののだけになることにしてください。

```
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."
```

オンラインでのとりてをむ https://riptutorial.com/ja/vba/topic/3446/のとりて

35: の

1つの`&`をして、ストリングをすることができます。

は、 `Join`をしてし、でされるゼロでもよいをすることもできます。

Examples

をしてをする

```
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をしてのをする

```
'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"
```

オンラインでのをむ https://riptutorial.com/ja/vba/topic/3580/の

36: のさの

のさは、の2つのでできます。もにされるさは`Len`をするですが、VBAでは`LenB`をしてバイトをすることもできます。 2バイトまたはUnicodeは1バイトです。

Examples

Lenをして、のをべる

```
Const baseString As String = "Hello World"

Dim charLength As Long

charLength = Len(baseString)
'charlength = 11
```

LenBをして、のバイトをべる

```
Const baseString As String = "Hello World"

Dim byteLength As Long

byteLength = LenB(baseString)
'byteLength = 22
```

`If LenmyString= 0 Then` `if myString =" "Then

のさがゼロであるかどうかをべるときは、とのをするのではなく、のさをべるがで、です。

```
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
```

オンラインでのさのをむ https://riptutorial.com/ja/vba/topic/3576/のさの

37: リテラル - エスケープしてできないと

VBAでのリテラルのりては、IDEのとのユーザーののコードページによってされています。のは、エスケープされた、な、できない、およびいリテラルのなケースをしています。

のコードページにのをむリテラルをりてる、のUnicodeリソースファイルからをりてることによって、のをするがあります。

Examples

"をエスケープする

VBAのは、リテラルは、にされている"マーク、ので、あなたのはをめるがあるとき、あなたはエスケープするがあります/プリペンド"の" VBAは、あなたがしていることをするように""であることを"としてされる。

```
'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"
```

いリテラルのりて

VBAエディタでは1に1023しかできませんが、はスクロールせずにの100150しかされません。いリテラルをりてるがあるが、コードをみやすくしたいは、とをしてをりてるがあります。

```
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では、をしてすることができますのはブロックののさによってなります。がにいは、をりててりてするがあります。

```
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の

VBAは、のようなののをします。

- vbCrキャリッジリターンCスタイルのでは "\ r"とじです。
- vbLfLine-Feed 'Cスタイルのでは "\ n"とじです。
- vbCrLfとWindowsの
- vbTabタブ
- vbNullString ""のようなの

これらのをやそののとともにして、でリテラルをすることができます。

```
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とのよりもいプラクティスとえている""によるコードがコンパイルされるのいに。は、りてられたメモリへのポインタをしてアクセスされ、VBAコンパイラは、 vbNullStringをすためにヌルポインタをするほどスマートvbNullString。 リテラル""は、StringのVariantであるかのようにメモリにりてられ、のをよりにします。

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

オンラインでリテラル - エスケープしてできないとをむ https://riptutorial.com/ja/vba/topic/3445/リテラル---エスケープしてできないと

38:

Examples

カレンダー

VBAは2カレンダーをサポートしています グレゴリオとヒジリ

`Calendar`プロパティは、のカレンダーをまたはするためにされます。

カレンダーの2つのはのとおりです。

0	vbCalGreg	グレゴリオデフォルト
1	vbCalHijri	ヒジュリカレンダー

```
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
```

このSubはをします。

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

システムの

VBAは、システムのからおよび/またはをするための3つのみみをサポートしています。

	りの	り
		のとをします。
		のとのをします。
		のとのをします。

```
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
```

タイマ

`Timer`は、からしたをするSingleをします。は100の1です。

```
Sub TimerExample()

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

End Sub
```

`Now`と`Time`はでしかでないため、 `Timer`はのをさせるなをします。

```
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は、がなかどうかをテストします。 `Boolean`します。

```
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
```

これらのは、 `Date`キャストしてまたはのをす`Integer`をす`Variant`をとります。パラメ―タを`Date`にキャストできないは、エラ―13のがします。

		ŋ
	ののをします。	1009999
	ののをします。	112
	ののをします。	131
	のをします。1をするオプションの2をけれる	17
	のをします。	023
	のをします。	059
2の	の2のをします。	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()`は、のをすですが、のでし、のよりもくのをにします。たとえば、やのをすことができます。

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

*interval*はのいずれかです。

"yyyy"	1009999
"y"	11366
"m"	112
"q"	14
"ww"	153
"w"	17
"d"	の131
"h"	023
"n"	059
"s"	059

*firstdayofweek*はオプションです。これはののをするです。されていない、 `vbSunday`がされます。

*1*はオプションです。それはののをするです。しない、のは11がするとみなされます。

```
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()`は、された2つののののをす`Long`をします。

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

- *interval*には、 `DatePart()`でされたのいずれかをできます
- *date1*と*date2*は、にする2つのです
- *firstdayofweek*と*firstweekofear*はオプションです。については、 `DatePart()`をしてください。

```
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()`は、されたまたはがされた`Date`をします。

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

- *interval*には、 `DatePart()` でされたのいずれかをできます
- *number*するのである。それはにをることまたはにのをるためにうことができます。
- は 、 `Date`のがされたまたはリテラルす

```
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
```

と

CDate

`CDate()`かのデータからかを`Date`データにします。

```
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
```

VBAには、にされたDateわりにのVariantすに、 CDate()とじでするやかなきCVDate()もあります。 Dateパラメータにすか Dateにするときは CDate()バージョンをし、 Variantパラメータにすか Variantにするときは CVDate()バージョンをするがあります。これにより、のキャストがされます。

DateSerial

DateSerial()をしてをします。した、、 Dateをします。

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

、、のがなであるは100から9999、は1から12、は1から31。

```
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()は "な"をけり、そこからなをすることにしてください。これはにいものとしてうことができます

ポジティブな

```
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
```

しかし、のないユーザーからをしようとすると、しみをきこすがよりくなります。

の

```
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
```

オンラインでをむ https://riptutorial.com/ja/vba/topic/4452/

39: きコンパイル

Examples

コンパイルのコードの

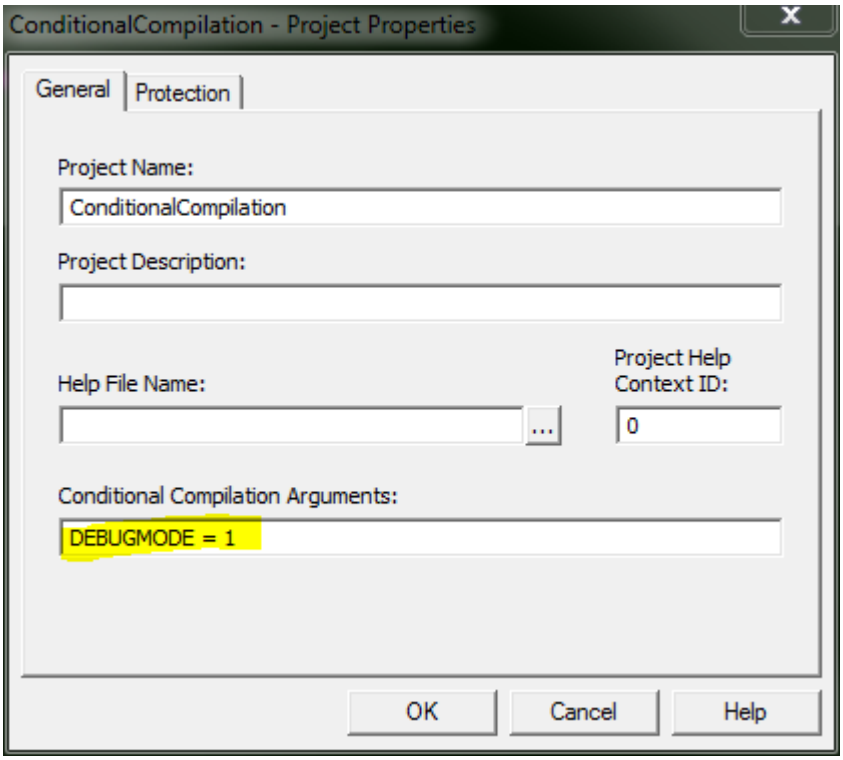
`#Const`ディレクティブは、カスタムプリプロセッサをするためにされます。これらはで`#If`によってコンパイルされされるコードのブロックをするためにされます。

```
#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
```

これにより、 `filepath`のが`"C:\Users\UserName\Path\To\File.txt"`にされ`"C:\Users\UserName\Path\To\File.txt"`。 `#Const`をするか、 `#Const #Const DEBUGMODE = 0`すると、`filepath`は`"\\server\share\path\to\file.txt"`にされ`"\\server\share\path\to\file.txt"`。

#Const Scope

`#Const`ディレクティブは、のコードファイルモジュールまたはクラスにしてのみです。これは、カスタムをするすべてのファイルごとにするがあります。また、ツール>> [プロジェクト]プロジェクトプロパティに`#Const`して、プロジェクトの`#Const`グローバルにすることもできます。これにより、プロジェクトプロパティダイアログボックスがされ、ここでをします。 `[constName] = [value]`きコンパイル`[constName] = [value]`ボックスに`[constName] = [value]`ます。 `[constName1] = [value1] : [constName2] = [value2]`ように、1つのをコロンでってでき`[constName1] = [value1] : [constName2] = [value2]`。



あらかじめされた

いくつかのコンパイルは、あらかじめされています。どちらがするかは、VBAをしているオフィスバージョンのビットによってなります.Vba7は、Office 2010とに64ビットバージョンのOfficeをサポートするためにされました。

	16ビット	32ビット	64ビット
Vba6		Vba6	
Vba7		Vba7の	
Win16			
Win32			
Win64			
マック		Macの	Macの

Win64 / Win32は、WindowsではなくOfficeであることにしてください。たとえば、OSが64ビットのWindowsであっても、32ビットOfficeではWin32 = TRUEとなります。

すべてのバージョンのOfficeでするインポートをする

```
#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
```

これは、サポートするのあるオフィスのバージョンにじてしすることができます。たとえば、まだ16ビットのOfficeをサポートしているはほとんどいません。 16ビットオフィスののバージョンは1994にリリースされたバージョン4.3であったため、ほぼすべてのなケースOffice 2007をむではのでです。

```
#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よりいものをサポートするがない、このはうまくいきます。

```
' 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
```

オンラインできコンパイルをむ https://riptutorial.com/ja/vba/topic/3364/きコンパイル

40:

はのでされます。

-
- ビット
-
-
-

するをつは、からにされます。のは、かっこ(および)をしてをグループすることできできます。

Examples

にってリストされます

トークン		
^		のオペランドをのオペランドにしたをします。によってされるは、されるのにかかわらずに Double であることにしてください。のへのは、がされたにわれます。
/	ディビジョン[1]	のオペランドをのオペランドでしたをします。でされるは、されるのになくに Double です。のへのは、がされたにわれます。
*	1	2つのオペランドのをします。
\		りて0.5でをめたにのオペランドでオペランドをするをします。のりのはされます。のオペランドが$_0$、エラー110がします。これはめがすべてされたであることにしてください$_{3 \ 0.4}$などのでもゼロエラーがします。
Mod	モジュロ	のオペランドをのオペランドでしたのりをします。のオペランドはのににめられ、.5はめられます。たとえば、 $_{8.6 \text{ Mod } 3}$と$_{12 \text{ Mod } 2.6}$が$_0$ます。のオペランドが$_0$、エラー110がします。これはめがすべてされたであることにしてください$_{3 \text{ Mod } 0.4}$などのでもゼロがわれます。
-	2	のオペランドからのオペランドをしたをします。
+	2	2つのオペランドのをします。このトークンは、 String されるときにとしてもわれることにしてください。 をしてください。

[1]とはじとしてわれます。

2とはじとしてわれます。

VBAは、2つのなるをサポート+と&とにするときのがまったくじをするStringタイプ-StringのにされるString。

&がStringのでされている、 &はされるにStringににキャストされます。

+は、 +のオ―バ―ロ―ドであることにしてください。 +のるいは、オペランドのとのによってまります。のオペランドはのようにされたはStringまたはVariantのサブタイプのString 、それらがされています。

```
Public Sub Example()
    Dim left As String
    Dim right As String

    left = "5"
    right = "5"

    Debug.Print left + right    'Prints "55"
End Sub
```

どちらかのがであり、もうのがににできるString、のにより、がとしてわれ、がされます。

```
Public Sub Example()
    Dim left As Variant
    Dim right As String

    left = 5
    right = "5"

    Debug.Print left + right    'Prints 10
End Sub
```

このるいは、にVariantがされているには、なエラ―をデバッグするのがしくなるため、&のみをにするがあります。

トークン		
=	にしい	とのオペランドがしいはTrueします。これはのオ―バ―ロ―ドであることにしてください。
<>	しくない	とのオペランドがしくないはTrueします。

トークン		
>	よりきい	のオペランドがのオペランドよりきいは`True`します。
<		のオペランドがのオペランドよりもさいは`True`します。
>=	それ	のオペランドがのオペランドよりきいかしいは`True`します。
<=		のオペランドがのオペランドのは`True`します。
`Is`		のオブジェクトがのオブジェクトとじインスタンスのは`True`します。どちらのでも、 `Nothing` nullオブジェクトとにすることができます。 `Is`は、をするにのオペランドを`Object`にしようとします。いずれかのがプリミティブまたはオブジェクトをまない`Variant` オブジェクトサブタイプまたは`vtEmpty`いずれかである、エラー424 - "Object required"がされます。いずれかのオペランドがじオブジェクトののインタフェースにしている、は`True`をし`True`。インスタンスとインタフェースののをテストするがあるは、わりに`ObjPtr(left) = ObjPtr(right)`します。

ノート

VBAはの「チェーン」をしていますが、これらのははにはけなければなりません。は、に2つのオペランドにしてにからへとされ、のは`Boolean`ます。たとえば、は...

```
a = 2: b = 1: c = 0
expr = a > b > c
```

...は、 `b`が`a`と`c`に`a`かどうかのテストとして、いくつかのコンテキストでみることができます。VBAでは、これはのようにされます。

```
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
```

`Object`とオペランドとしてされる`Is`のは、 `Object`のデフォルトメンバーのりにしてされます 。オブジェクトにのメンバがない、によってエラー438がします - 「オブジェクトがそのプロパティま

たはメソッドをサポートしていません」

`Object`がされていない、エラー91 - "オブジェクトまたはWithブロックがされていません"がされます。

リテラル`Nothing`が`Is`のでされる、コンパイルエラー - 「オブジェクトのな」がします。

`Object`のメンバがの*`Object`*、VBAはプリミティブがされるか、エラーがするまで、するりののメンバをにびします。えば、`SomeClass`デフォルトのする`Value`のインスタンスである、`ChildClass`のと`ChildValue`。 ...

```
Set x = New SomeClass
Debug.Print x > 42
```

...はのようにされます

```
Set x = New SomeClass
Debug.Print x.Value.ChildValue > 42
```

いずれかのオペランドがで、もうのオペランドがサブタイプ`String` `String`または`Variant`の、のがされます。この、`String`をにキャストできない、エラー13 - ""がになります。

のオペランドがサブタイプ`String` `String`または`Variant`の、コードモジュールのOption Compareについてのがされます。これらのは、でわれます。をむ`String`は、のとじではないことにしてください。

```
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
```

このため、`String`または`Variant`がをするににキャストされていることをしてください。

のオペランドが`Date`、もうのオペランドがであるか、にキャストであれば、となるDoubleのがされます。

のオペランドがのロケールをして`Date`キャストできる`String`または`Variant`のサブタイプ`String`、`String`は`Date`キャストされます。のロケールの`Date`にキャストできない、ランタイムエラー13 - "の"がのします。

`Double`または`Single`とブールをするときはがです。のとはなり、をむのデータを`Double`にするVBAのにより、ゼロは`True`とみなすことはできません。

```
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
```

ビットの

VBAのすべてのは、じのビットの「き」とえることができます。には、 にビットとしてわれます。 VBAのすべてのは、にビットを `False` したり、 すべてのビットを `True` しないブールをします。しかし、それはのようにのビットにをいます`True`。 つまり、のビットを`Boolean`キャストしたをは、にとしてうこととじになります。

これらのの1つをしてのをすると、ビットのがられます。のでは、 `0`は`False`し、 `1`は`True`することにしてください。

`And`

のが`True`されるは`True`します。

オペランド	オペランド	
0	0	0
0	1	0
1	0	0
1	1	1

`Or`

のいずれかのが`True`されるは`True`します。

オペランド	オペランド	
0	0	0
0	1	1
1	0	1
1	1	1

Not

り`True`がとされた`False`と`False`のに`True`。

オペランド	
0	1
1	0

オペランドなしののオペランドはあり`Not`。 Visual Basic Editorは、のでをにします。あなたがした...

```
Debug.Print x Not y
```

... VBEはをのようにします。

```
Debug.Print Not x
```

のは、 `Not`のオペランドをむをむにしてわれます。

Xor

「OR」ともばれます。のがなるをすは`True`します。

オペランド	オペランド	
0	0	0
0	1	1
1	0	1
1	1	0

`Xor`はのようにできますが、`<>`とじがられるため、 `Xor`はのようにできますが、にうはありません。

Eqv

「」ともばれます。のがじにされたときに`True`します。

オペランド	オペランド	
0	0	1
0	1	0

オペランド	オペランド	
1	0	0
1	1	1

`x Eqv y`は、はるかにみやすい`Not (x Xor y)`とであるため、 `Eqv`はごくまれにしかされないことにしてください。

`Imp`

「」ともばれます。し`True`のオペランドがじであるか、オペランドであるは`True`。

オペランド	オペランド	
0	0	1
0	1	1
1	0	0
1	1	1

`Imp`はほとんどされないことにしてください。としては、そのをできないは、のをうべきです。

オンラインでをむ https://riptutorial.com/ja/vba/topic/5813/

41: りしをむをりてる

のをのりすをりてるがあることがあります。 VBAは、こののために2つのなをします。

- `String` / `String$`
- `Space` / `Space$`。

Examples

をして、nのりしをりてます

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

とスペースをして、nのをりてます

```
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, " ")
```

オンラインでりしをむをりてるをむ https://riptutorial.com/ja/vba/topic/3581/りしをむをりてる

42: のをでする

ののまたはをするがある、ののをす`InStr`と`InStrRev`がされていますする。

Examples

InStrをして、にがまれているかどうかをべる

```
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をして、ののインスタンスのをつける

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

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

InStrRevをして、ののインスタンスのをする

```
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
```

オンラインでのをでするをむ https://riptutorial.com/ja/vba/topic/3480/のをでする

43:

Examples

VBAでをする

のは、のとによくていますが、のにのをするがあるがなります。

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

では、VBAのはゼロからインデックスされているため、かっこのはのサイズをするのではなく、ののインデックスをします

へのアクセス

のにアクセスするには、Arrayのをし、のにあるのインデックスをけます。

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

のけ

のインデックスけをするには、このをモジュールのにします。

```
Option Base 1
```

このでは、モジュールでされているすべてのが1つのインデックスにされます。

のインデックス

Toキーワードと/=インデックスをして、のインデックスをつをすることもできます。

```
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
```

のにArrayのサイズがわからないは、とReDimキーワードをできます。

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

ReDimキーワードをすると、ののがされます。これをぐには、 ReDimにPreserveキーワードをし ReDim。

```
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をしてからをする

したのをむゼロベ―スの1をします。

[、 デリミタ [、 [、]]]

	。とりをむ。 *expression*がさゼロの ""またはvbNullStringの、 **Split**はもデ―タもまないのをします。この、されるは0のLBoundと-1のUBoundをちます。
デリミタ	オプション。のをするためにされる。すると、スペ―ス ""がりとみなされます。 りがさゼロのの、 をむのがされます。
	オプション。されるの。 -1は、すべてのがされることをします。
する	オプション。をするときにするのをす。については、セクションをしてください。

compareには、のをできます。

	-1	**Option Compare**のをしてをします。
vbBinaryCompare	0	バイナリをします。
vbTextCompare	1	テキストをします。
vbDatabaseCompare	2	Microsoft Accessのみ。デ―タベ―スのについてをします。

このでは、Splitがいくつかのスタイルをすることによってどのようにするかをしています。コメントには、されたなるスプリットオプションのそれぞれについてのセットがされます。に、されたをル―プするをします。

```
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
```

のをする

For ...へ

ィテレータをインデックスとしてすることは、のをするもいです。

```
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
```

ネストされたループをして、をすることができます。

```
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
```

それぞれのために...へ

`For Each...Next`ループは、パフォーマンスがでないでも、のにできます。

```
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`ループは、すべてのをからにしますがメモリにされるのとじので、ネストされたループはありません。

```
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
```

パフォーマンスがな、 `For Each`ループは`Collection`オブジェクトをするのにです。

の4つのスニペットはすべてじをします

```
0
1
2
3
```

のサイズと

ダイナミックアレイ

のをにしたりらしたりすることは、うにされたのがないにきなとなります。

をにする

`ReDim`ステートメントでのサイズをするだけで、のサイズはされますが、ににされているをするは、 `Preserve`というがになり`Preserve`。

のでは、のをしながら、をし、りしでを1つやします。

```
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
```

をにする

じをしてをらすことができます。このでは、「last」がからされます。

```
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
```

のリセットと

したをメモリにすることもできるので、がくなります。これは、さまざまなサイズのにです。をするためにできるスニペットは、を`ReDim (0)`し、1つのをにさせ、をびにさせることです。

のスニペットでは、140のをし、をにして、40100でをします。すべてこれがにわれます。

```
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
```

ジ グザ グの

ジ グザ グはではありません

のジグザグはとじではありません。はのがされたのようにえますが、はごとののののように、のがなるをつカレンダー。

ジグザグは、ネストされたレベルのためにするのがでいにくいですが、のはあまりありませんが、にがあるため、さまざまなのデータをにできます。の。

ギザギザのをする

のでは、のとNumbersのの2つのをむぎざぎざのをし、それぞれのにアクセスします

```
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
```

ジ グザ グのとみみ

たちは、をするのにもっとダイナミックになることができます。データシートがExcelにあるとして、のをするをしたいとえています。

```
   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
```

HeaderとCustomersをにし、ヘッダーにタイトルをし、Customersにはとして/のをします。

```
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.

のスニペットのは、4、2インデントレベルをつの2つのをつジャグドで、もう1つはそれぞれ4と3インデントレベルの5をむのジャグドです。

```
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)
```

あなたはのであなたがジャグのをするがありますにアクセスするには、のでは、あなたはそれをることができる`Main Array`のがまれ`Headers`とアレイの`Customers`さまざまなでそれゆえにアクセスする。

ここで、 `Main Array`をみ、を`Info Type: Info`としてします。

```
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
```

あなたのJagged Arrayのをするために、のでは、 `Main_Array -> Customers -> CustomerNumber -> Name`という3つのレベルにアクセスして、な`"Person4"`をします `Main_Array(Customers)(CustomerNumber)(Name)` `Main_Array(1)(3)(0)`ある`Main_Array(1)(3)(0)`は、Main_ArrayのCustomersの、にCustomer JaggedのCustomer 4の、 `Main_Array(Customers)(CustomerNumber)(Name)`。

がすように、は、のは2つまたは3つをむですが、32のをつことができます。

マルチは、さまざまなレベルののようにし、1つ、2つ、3つのをします。

One Dimensionはなであり、のリストのようにえます。

```
Dim 1D(3) as Variant

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

2はグリッドやExcelシートのようにえます。をするときに、のとをします。

```
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)
```

はルーービックのキューブのようにえるでしょう。をするときには、と、および/をします。

```
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)
```

さらなるは3Dのとえることができるので、4D1,3,3,3は2つのの3Dアレイとなる。

2

のは、のリストをめたもので、はリスト、、、メーール、などにするのをちます。はににされます、は、0,0がののです。

```
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
```

サイズ

サイズまたは`ReDim Preserve`は、1のノルムのように、エラ―がするわりに、のサイズとじサイズとするまたはをえたをにするがあります。のでは、をし、のからをし、りののをめ、をのにきえるをていきます。

```
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
```

の

ののを/するには、するをびしてしいをえるだけです`Employees(0, 0) = "NewValue"`

あるいは、をしてをりし、なパラメ―タにするをさせることができます。

```
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
```

のにアクセスするには、ネストループすべてのをりします、ループとをりし、にアクセスする、またはのでアクセスすることができます。

```
'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)
```

をする、マップをすることはにであることをえておいてください 。することがあります。

3

3Dアレイでは、2Dとじをします。とをするだけでなく、ビルディングをみむだけでなく、

3Dには、とえることができる、、およびがあり、Excelではなるシートとえることができ、サイズはじですが、シートにはそのセル/のなるセット。 3Dアレイには、 ***n***の2Dアレイがまれます。

3Dは3つのをするがあります`Dim 3Darray(2,5,5) As Variant` 、ののはBuilding / Sheetsとのなるセットのになり、2のはと3をします。の`Dim`は108 `3*6*6` の3Dをもたらし、に3つのなる2Dのセットをちます。

```
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
```

サイズ

3Dのサイズは、2Dのサイズとていますが、のサイズとじサイズのテンポラリをにし、パラメータのに1をやすと、のはのセットをやします。 3のは、セットのまたはのをやします。

のでは、セットのを1つずつやし、されたにしいをめみます。

```
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
```

のとみをする

3Dのをみんですることは、2Dのやりとにうことができます。ループとのなレベルをするだけです。

```
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)
```

オンラインでをむ https://riptutorial.com/ja/vba/topic/3064/

44: のコピー—、、し

Examples

のコピー—

=をして、VBAをじのにコピーできます。はじでなければなりません。さもなければ、コードは "にできません"というコンパイルエラーをスロー—します。

```
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
```

ソースはまたはにできますが、はでなければなりません。にコピー—しようとすると、 "Can not assign to array"コンパイルエラー—がスロー—されます。のデータはすべてわれ、そのとはとじにされます。

```
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)
```

コピー—がされると、2つのはメモリで々のものになります。つまり、2つのはじデー—タへのではないため、ののはのにはされません。

```
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
```

オブジェクトののコピー

オブジェクトのでは、それらのオブジェクトへのがコピーされ、オブジェクトはコピーされません。 1つののオブジェクトにがえられた、そのはのでもされているようにえます - ともじオブジェクトをしています。ただし、を1つののののオブジェクトにしても、それをのにすることはできません。

```
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
```

をむバリアント

バリアントとのでをコピーすることもできます。バリアントからコピーするときは、とじのをむがあります。それのは、「の」ランタイムエラーがします。

```
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
```

からをす

のモジュールクラスモジュールではないのは、データのに () をくことによってをすことができます。

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

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

```
    arrayOfPiDigits = outputArray
End Function
```

のは、じまたはバリアントのにすることができます。には、2のブラケットセットをしてアクセスすることもできますが、これはをびすため、するがあるはをしいにすることをおめします

```
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
```

されるのは、にはののコピーであり、ではありません。がのをす、そのデータはびしのプロシージャによってできません。

によるの

プロシージャのがであり、りをしてされるのは、はいコーディングです。しかし、VBAのにより、プロシージャが`ByRef`をしてデータをするが`ByRef`ます。

への

```
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
```

クラスメソッドからのの

は、クラスモジュールのメソッド/プロシージャからをするためにもできます

```
' 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
```

プロジェクトへののけし

ののろに`()`くことによって、をきにすことができます。

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

はしするがあります。すメカニズムがされていない、たとえば`myFunction(arr())`、VBAはデフォルトで`ByRef`をきけますが、にするのはいコーディングです。たとえば、 `myFunction(ByVal arr())` をしですと、「はByRefでなければなりません」コンパイルエラーVBEオプションで`Auto Syntax Check`がチェックされていないと「エラー」コンパイルエラー 。

しとは、へのがびしでされることをします。

```
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
```

のをしないようにするには、をしないようにをするようにしてください。

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

あるいは、のコピーをし、そのコピーでしてください。

```
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
```

オンラインでのコピー、、しをむ https://riptutorial.com/ja/vba/topic/9069/のコピー-- し

45: ラテン

き

VBAは、 Unicodeをしてのまたはスクリプトでをみきできます。しかし、 トークンのためのよりしいがあります。

Examples

VBAコードのラテン

スプレッドシートのセルA1には、のアラビアのパングラムがあります。

ユーザーをおれですか

VBAは、マルチバイトコードでするAscWおよびChrWをします。 Byteをってをすることもできます

```
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
```

これは、 アラビアのしいしみのためにのをします。

のcharChrW1589
のcharバイナリ00011000110101
テキスト
AscW1589
バイト1053 | 6
バイトバイナリ00110101 | 00000110

??? ????? ????? ??????? ??????? ??? ??????? - ????? ???????? ???? ??????? ???????

がしくしていても、VBAはラテンのテキストをウィンドウにできないことにしてください。これはIDEのであり、のではありません。

ラテンのとのカバレッジ

VBA とはラテンのスクリプトをでき、 、 、 、 のスクリプトもできます。

されたラテンのスクリプトは、くのをしています
、フランス、スペイン、ドイツ、イタリア、ブルトン、カタロニア、デンマーク、エストニア、フィンランド、アイスランド、インドネシア、アイルランド、ロイバン、Mapudungun、ノルウェー、ポルトガル、スコットランドゲール、スウェーデン、タガログ

のはにしかカバーされていません
アゼルバイジャン、クロアチア、チェコ、エスペラント、ハンガリー、ラトビア、リトアニア、ポーランド、ルーマニア、セルビア、スロバキア、スロベニア、トルコ、ヨルバ、ウェールズ

のには、ほとんどまたはまったくカバーしていません
アラビア、ブルガリア、チェロキー、Dzongkha、ギリシャ、ヒンズー、マケドニア、マラヤーラム、モンゴル、ロシア、サンスクリット、タイ、チベット、ウルドゥー、ウイグル

のはすべてです。

```
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 IDEでは、ののアポストロフィはをコメントにしませんスタックオーバーフローと。

また、をするために2つのをする«»は、のでそれらをすることができますが、はそうではありません。

オンラインでラテンをむ https://riptutorial.com/ja/vba/topic/10555/ラテン

46: にされる

き

INSTR FINDとLENをしたMID LEFTとRIGHTののな

どのように2つののにテキストをつけるのですかコロンのとカンマのにどのようにのりのをしますかMIDをするか、またはRIGHTをしますかこれらののうち、ゼロベースのパラメータとリターンコードをするはどれですかかがうまくいかないとどうなりますからはの、のない、およびのをどのようにうのですか

Examples

にされる

よりいMIDやのの、Webからけている。はいをるのをけてください、またはここでこれをしてください。このようなもの

```
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) '...
```

このをにしてしてください。り、それはらかにして、それにのがあります。

オンラインでにされるをむ https://riptutorial.com/ja/vba/topic/8890/にされる

クレジット

S. No		Contributors
1	VBAをいめる	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	ADOの	Comintern, SandPiper, Tazaf
3	APIコール	paul bica
4	CreateObjectとGetObject	Branislav Kollár, Dave, Tim
5	FileSystemObjectをせずにファイルとディレクトリをする	Comintern, Macro Man, SandPiper
6	Scripting.Dictionaryオブジェクト	Comintern, Jeeped, Kyle, RamenChef, Tim, Wolf, Zev Spitz
7	Scripting.FileSystemObject	Comintern, Dave, Macro Man, Mikegrann, RubberDuck, Siva, Steve Rindsberg, ThunderFrame
8	VBAオプションキーワード	Jeeped, Maarten van Stam, Macro Man, Mat's Mug, RamenChef, RubberDuck, Stefan Pinnow, Thomas G, ThunderFrame
9	VBAランタイムエラー	Branislav Kollár, Macro Man, Mat's Mug
10	イベント	Mat's Mug
11	インターフェイス	Neil Mussett
12	エラー	Comintern, Logan Reed, Mat's Mug
13	オートメーションまたはのアプリケーションライブラリの	Branislav Kollár
14	オブジェクトVBA	IvenBach, Mat's Mug
15	カスタムクラスの	Branislav Kollár, Macro Man, Mat's Mug, Neil Mussett, ThunderFrame

16	コメント	Comintern, Hosch250, Johnny C, litelite, Macro Man, Nijin22, Shawn V. Wilson, ThunderFrame
17	コレクション	Comintern
18	サブストリング	Mat's Mug, ThunderFrame
19	データと	Comintern, FreeMan, Neil Mussett, StackzOfZtuff, Stephen Leppik, ThunderFrame
20	データ	Blackhawk
21	バイナリで2GB +ファイルをVBAとファイルハッシュでむ	PatricK
22	フロー	Benno Grimm, Comintern, Kelly Tessena Keck, Leviathan, litelite, Macro Man, Martin, Mat's Mug, Roland, Siva, ThunderFrame
23	プロシージャコール	Macro Man, Mat's Mug, Neil Mussett, Sam Johnson
24	プロシージャの	Comintern, LiamH, Mat's Mug, Sivaprasath Vadivel, Tomas Zubiri
25	マクロセキュリティとVBAプロジェクト/モジュールの	0m3r
26	ユーザーフォーム	Mat's Mug
27	べえ	Neil Mussett
28	のをにする	ThunderFrame
29		Mat's Mug, ThunderFrame
30		FreeMan, Kaz, Mat's Mug, Victor Moraes
31	の	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
32		hymced, Mat's Mug, RamenChef, RubberDuck
33	ByRefまたはByValをす	Branislav Kollár, Comintern, Mat's Mug, R3uK, RamenChef, ZygD

34	のとりて	Comintern, ThunderFrame
35	の	ThunderFrame
36	のさの	Steve Rindsberg, ThunderFrame
37	リテラル - エスケープしてできないと	Comintern, ThunderFrame
38		Comintern, FreeMan, Thomas G
39	きコンパイル	Macro Man, Mat's Mug, RubberDuck, Steve Rindsberg
40		Comintern, Macro Man
41	りしをむをりてる	ThunderFrame
42	のをでする	ThunderFrame
43		Comintern, Dave, Hubisan, jamheadart, Josan Iracheta, Maarten van Stam, Mark.R, Mat's Mug, Miguel_Ryu, Tazaf
44	のコピー、、し	Mark.R
45	ラテン	Neil Mussett
46	にされる	pashute