Excel VBA Utility

[Excel VBA]

basUtil

```
Option Explicit

' Win32 API
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Private Declare Function WritePrivateProfileString Lib "kernel32 " Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As String, ByVal lpString As String, ByVal lpFileName As String) As Long

Public Const MAX_INI_LEN    As Long = 256
Public Const INI_EXTENT     As String = ".ini"
Public Const EXCEL_EXTENT   As String = ".xls"
Public Const DIR_SEP        As String = "\"

'
' 設定を保存する
'
' @param book    Excel ブック
' @param strKey  設定キー
' @param strVal  設定値
'
Public Function setAppSetting(ByRef book As Excel.Workbook, _
                              ByVal strKey As String, _
                              ByVal strVal As String) As Long

    setAppSetting = WritePrivateProfileString( _
                        APP_NAME, strKey, strVal, getIniFilename(book))

End Function
'
' 設定を取得する
'
' @param book       Excel ブック
' @param strKey     設定キー
' @param strDefault 対応する設定値が存在しない場合のデフォルト値
'
Public Function getAppSetting(ByRef book As Excel.Workbook, _
                              ByVal strKey As String, _
                              ByVal strDefault As String) As String

    Dim strBuf As String * MAX_INI_LEN
    Dim result As Long

    result = GetPrivateProfileString( _
                APP_NAME, strKey, strDefault, strBuf, MAX_INI_LEN, _
                getIniFilename(book))

    getAppSetting = Left(strBuf, InStr(strBuf, vbNullChar) - 1)

End Function
'
' 初期化ファイル名を取得する
'
' @param book       Excel ブック
'
Private Function getIniFilename(ByRef book As Excel.Workbook) As String

    getIniFilename = getWorkbookRerativeFilename(book, INI_EXTENT)

End Function
'
' EXCEL ファイルのプルパス、拡張子変更文字列を返す
'
' @param book    Excel ブック
' @param extents 拡張子
' @return Excel のファイル名の拡張子を変更した文字列
'
Private Function getWorkbookRerativeFilename(ByRef book As Excel.Workbook, extents As String) As String

    Dim chk As String
    Dim tmp As String

    tmp = book.Name
    chk = String(Len(EXCEL_EXTENT), " ") & tmp

    If Right$(chk, Len(EXCEL_EXTENT)) = EXCEL_EXTENT Then
        tmp = Left$(tmp, Len(tmp) - Len(EXCEL_EXTENT))
    End If

    getWorkbookRerativeFilename = getPath(book.path) & tmp & extents

End Function
'
' パス名を作成 ( 最後が '\' で終わるように整形する )
'
'  @param pathName 整形するパス名
'  @return 整形されたパス名
'
Public Function getPath(ByVal pathName) As String

    Dim result As String

    result = pathName
    If Trim$(result) = "" Then
        result = "."
    End If

    If Right$(result, 1) <> DIR_SEP Then
        result = result & DIR_SEP
    End If
    getPath = result

End Function
'
' ファイルを選択する
' fileFilter eg-"Microsoft Excel ブック ,*.xls, テキストファイル ,*.txt"
'
Public Function chooseFile(Optional fileFilter As String, _
                           Optional rootDir As String, _
                           Optional filterIndex As Integer, _
                           Optional title As String, _
                           Optional buttonText As String, _
                           Optional MultiSelect As Boolean) As Variant

    Dim drv As String

    drv = UCase$(Left$(rootDir, 1))

    If Not isBlank(Trim$(rootDir)) _
    And ("A" <= drv And drv <= "Z") _
    And (Mid$(rootDir, 2, 1) = ":") _
    Then

        Call ChDrive(drv)
        Call ChDir(rootDir)

    End If

        chooseFile  =  Application.GetOpenFilename(fileFilter,  filterIndex,  title,  buttonText,
MultiSelect)

End Function
'
' フォルダ取得ダイアログを表示する
'
'
' @return 選択されたフォルダ名
' @param  title タイトルの文字列
'         options    選択オプションの値
'         rootFolder 既定フォルダの文字列
'
Public Function browseForFolder(title As String, _
                options, _
                Optional rootFolder As String = "") As String

    Dim cmdShell  As Object
    Dim folder As Object
    On Error GoTo errHandler

    Set cmdShell = CreateObject("Shell.Application")

    'syntax
    '  object.BrowseForFolder Hwnd, Title, Options, [RootFolder]
    '
    Set folder = cmdShell.browseForFolder(0, title, options, rootFolder)
    If Not folder Is Nothing Then
        browseForFolder = folder.Items.Item.path
    Else
        browseForFolder = ""
    End If

closer:
    Set folder = Nothing
    Set cmdShell = Nothing

    Exit Function

errHandler:
    MsgBox Err.Description, vbCritical
    GoTo closer

End Function
'
' 文字列配列を、区切文字で区切られたテキストに展開する
'
' @param fields    展開する文字列配列
' @param separator 区切文字
' @return 区切文字で区切られたテキスト
'
Public Function arrayToSeparetedValueText(ByRef fields() As String, separator As String) As String

    Dim ret As String
    Dim i As Integer

    For i = 0 To UBound(fields)
        If i = 0 Then
            ret = fields(i)
        Else
            ret = ret & separator & fields(i)
        End If
    Next

    arrayToSeparetedValueText = ret

End Function
'
' 文字列に対する簡易ソート
'
' @param values ソート対象の文字列配列
'
Public Sub bubbleSort(ByRef values() As String)

    Dim tmp As String
    Dim i As Integer
    Dim j As Integer

    tmp = ""
    For i = LBound(values) To UBound(values)
        For j = i + 1 To UBound(values)
            If StrComp(values(i), values(j), vbTextCompare) > 0 Then
                tmp = values(i)
                values(i) = values(j)
                values(j) = tmp
            End If
        Next
    Next

End Sub
'
' 指定された文字列が配列に含まれるか検査する
'
' @param str    検査文字列
' @param strAry 検査対象配列
' @return 配列に値が含まれていればTrue
'
Public Function isContainArray(str As String, strAry() As String) As Boolean
    Dim i As Integer

    For i = LBound(strAry) To UBound(strAry)
        If str = strAry(i) Then
            isContainArray = True
            Exit Function
        End If
    Next
    isContainArray = False

End Function
'
' 文字列配列に、重複項目があるかをチェックする
'
' @param values 検査対象配列
' @param duplicatedItems 重複項目を文字列配列にセットする
' @return 重複がある場合に true を返す
'
 Public Function duplicatedCheck(ByRef values() As String, ByRef duplicatedItems() As String) As
Boolean
    Dim result As Boolean
    Dim i As Integer
    Dim tmp() As String
    Dim dupIdx As Integer

    result = False

    dupIdx = 0
    ReDim tmp(UBound(values))
    For i = LBound(values) To UBound(values)
        tmp(i) = values(i)
    Next

    Call Util.bubbleSort(tmp)

    Dim curVal As String
    For i = LBound(tmp) To UBound(tmp)
        If i > LBound(tmp) Then
            If curVal = tmp(i) Then
                ReDim Preserve duplicatedItems(dupIdx)
                duplicatedItems(dupIdx) = curVal
                dupIdx = dupIdx + 1
                result = True
            End If
        End If
        curVal = tmp(i)
    Next
    duplicatedCheck = result

End Function

'
' ブランク判定
'
' @param str 判定する文字列
' @return 文字列が空なら True
'
Public Function isBlank(str As String) As Boolean

    isBlank = ((Trim$(str)) = "")

End Function

'
' Boolean を "1":True、"0":False に変換
'
' @param blv 真偽値 (Boolean)
' @return True なら "1"、False なら "0"
'
Public Function booleanToFlag(blv As Boolean) As String

    booleanToFlag = IIf(blv, "1", "0")

End Function
'
' Boolean を "1":True、" ":False に変換
'
' @param blv 真偽値 (Boolean)
' @return True なら "1"、False なら " "
'
Public Function booleanToFlag2(blv As Boolean) As String

    booleanToFlag2 = IIf(blv, "1", " ")

End Function
'
' フラグ を "1":True、"0":False に変換
'
' @param flag フラグ
' @return "1" なら True、それ以外は False
```

```
'
Public Function flagToBoolean(flag As String) As String
    flagToBoolean = IIf((Trim$(flag) = "1"), True, False)
End Function
```