Table Management

Excel isn’t designed to be a database manager but many of us use Excel to store and manipulate lists. I’m personally very fond of Excel tables and I use a lot of them. Manipulating Excel tables from VBA can be a challenge. I’ve created some code that lets me convert a table into a VBA dictionary where it’s easier to manipulate and then copy the data back to the table.

The table manipulation code is flexible. It can read/write tables to/from this workbook, other Excel workbooks, or Access.

I’ll describe the table manipulation process and show you how to create your own routines to manipulate your own tables. Each table has a code module and an associated class module. There is a pattern to both modules that I have automated into a TableBuilder code generator. I’ll describe the code generator in a separate write-up.

There is a Table module that handles most of the heavy lifting. There is an interface class module (iTable) that provides an interface specification for each table. If you’re not familiar with interface class modules, I recommend Google; I won’t be explaining them here. Table is written against the interfaces n iTable; that allows Table to be generic and manipulate any table regardless of the number of columns or rows and to read/write to/from any Excel table or Access database.

The Table module has two primary entrance points: TryCopyTableToDictionary and TryCopyDictionaryToTable. These are both Boolean functions that return True if they are successful. If they are unsuccessful, you can use the error handling techniques provided in <https://github.com/BriargateExcel/Error_Handling>. The term “Table” in the function names is general; the table can be an Excel table or a table in an Access database. For example, TryCopyTableToDictionary can copy an Excel table from this or another workbook or it can copy an Access table.

You need to create a class module for each table that implements the 14 interfaces in iTable. Each of these class modules is a pass-through to the code module. The class module stores minimal information about the table. This is the iTable class:

This is going to be quite difficult to describe how all this works together. Maybe cut straight to the TableBuilder and skip the explanations.

**iTable Interface Class:**

Option Explicit

' 2020-04-08

' Added 3 database related routines to support

' the capability to read/write to/from an access database

Public Property Get HeaderWidth() As Long

End Property

Public Property Get LocalDictionary() As Dictionary

End Property

Public Property Get LocalTable() As ListObject

End Property

Public Property Get Headers() As Variant

End Property

Public Property Get Initialized() As Boolean

End Property

Public Sub Initialize()

End Sub

Public Function TryCopyDictionaryToArray(ByVal Dict As Dictionary, ByRef Ary As Variant) As Boolean

End Function

Public Function TryCopyArrayToDictionary(ByVal Ary As Variant, ByRef Dict As Dictionary) As Boolean

End Function

Public Sub FormatArrayAndWorksheet( \_

ByRef Ary As Variant, \_

ByVal Table As ListObject)

End Sub

Public Property Get LocalName() As String

End Property

Public Property Get CreateKey(ByVal Record As iTable) As String

End Property

Public Property Get IsDatabase() As Boolean

End Property

Public Property Get DatabaseName() As String

End Property

Public Property Get DatabaseTableName() As String

End Property