Broaden your audience

Learn how to improve your accessibility guidance

Communicator

The Institute of Scientific and Technical Communicators
Autumn 2018

Have the people involved in your meetings changed?

Build a hybrid mobile app using Cordova

Communicator is 50!

Manage terminology with the TechScribe term checker

Use SCHEMA ST4 content for augmented reality applications

Creating user forms: part 4

Coding an interface for your VBA code. By Mike Mee.

Introduction

Welcome to the latest article in my series that covers adding a form to your VBA code. In this fourth one, I will be covering combo (aka drop-down) boxes.

Naming convention

As described in my previous articles, each type of control has a three-letter prefix that you can use to help you remember which control each variable is referring to when you interpret the actions.

As we're only using combo boxes in this article, there is only one to remember.

Table 1. Controls, icons and naming conventions

Control	Icon	Leszynski
Combo box		cmb

A clarification

This article will cover combo boxes when used in VBA forms only .It does not cover the use of combo boxes on a document that uses forms on the page: they are very different indeed.

Although the code is very similar in the population side of things, the setting up and initialisation of them are very, very different.

Populating combo boxes

Check boxes have been around for a long time and are easy to use in your own code. You generally use them to limit a user's options.

We will use the form that we started in the previous article, *Communicator* Summer 2018, which had the multiple flags on it. I have placed the combo box next to the image and made it long enough to fit in some example data.

The next step is to rename the combo box to something a bit more useful and, as for previous articles, this is done by the Properties window.

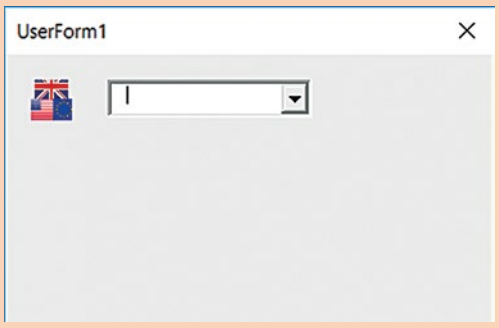

Figure 1. The form from the last article with the image on and now a new check box inserted

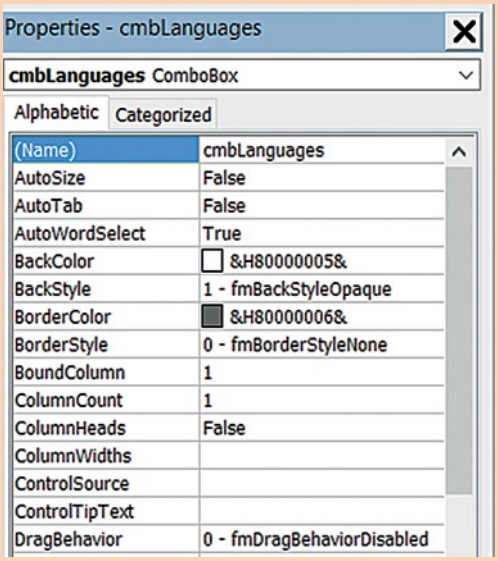

Figure 2. The inserted combo box's VBA properties which I have renamed to 'cmbLanguages'

If you run the form now, you will have the image with the combo box next to it. It may be empty, but it is the first stage completed.

Now we need to populate the combo box so that there is something for the user to choose from.

Populating combo boxes

There are two methods in VBA that are used to populate a combo box.

The usual way is to clear the existing items (even if there are none) via the `Clear` function. Then you can add the `AddItem` command. The following code sets up our new combo box with five new items:

```
Sub UserForm_Initialize()
cmbLanguages.Clear
With cmbLanguages
    .AddItem "English"
    .AddItem "French"
    .AddItem "German"
    .AddItem "Spanish"
    .AddItem "Italian"
End With
End Sub
```

The above code goes into the `UserForm_Initialize` code for the form. This is the same process as we have been using in the previous articles.

However, there is also nothing stopping you altering the contents of the combo box afterwards elsewhere in your code.

Evaluating what the user picks

This is just as easy as setting up the combo box and it requires a single command. You need to define a string variable to hold the result, such as:

```
Dim strResult as String
```

You can then place the result of the combo box into the string like this:

```
strResult = cmbLanguages.Text
```

What you do with the result is entirely up to you. In the following example, the code just displays the answer that the end user picked.

```
Private Sub cmbLanguages_Change()
strResult = cmbLanguages.Text
MsgBox "You picked: " & strResult
End Sub
```

By hooking into the `_Change` event for the combo box, you can interact with the user as soon as they have changed the contents of the combo box.

Populating combo boxes via array

There is another way to set up the contents of your combo box and that is by using an array to populate it.

However, as I have not yet covered the use of arrays in VBA, I will give you some example code to look at and you will have to work your way through it and see if this is easier for you than copious amounts of `.AddItem` in your code.

The following code was taken from my Word Toolbox. The array set up is placed as a single line of code. It uses an array of values to populate the combo box, in this example it is for the conversion of a Section Break from one type to another:

```
cmbConvertSBTo.List = Array("Continuous",
"New column", "New Page", "Even Page", "Odd
Page")
```

The reason for doing it this way was to optimize the size of the final add-in. Some of the arrays are *huge* and creating them via a long list of `.AddItem` commands would bloat the code size. The types of field in a Word document stretches to 95 items (currently!) and it would take up a lot of space.

However, I won't go into too much detail (for now) as trying to cover the use of arrays will take up a whole article itself. Arrays are the easiest way to control list boxes, so I will need to cover arrays sooner rather than later.

A shorter article than normal

My apologies for this shorter article, but that is because I've also written a second article for the 50th *Communicator* special supplement that you are getting with this issue of the journal.

This is also the first article that has been written at my weekly B&B lodgings, which are situated in front of a dairy farm. The smell of over-ripe manure lining the roads as it fell out of a trailer during transit that hot summer day will definitely not be forgotten by my 'city boy' senses!

Probably though, by the time this article appears in print, I guess we will all be reminiscing all about that long hot summer, as we trudge to work in the rain!

TCUK 2018

All being well, I will be attending this year's TCUK conference in Daventry, so I hope to see some of you there. Please bring any of your VBA queries with you and I will see what I can do to help. C

References and further reading

Mee M (2015) 'Variables and screen output/input' *Communicator*, Autumn 2015: 44-47

Mee M (2017) 'Optimising your VBA code' *Communicator*, Spring 2017: 37-39

Mee M (2017) 'Using the Quick Access Toolbar' *Communicator*, Summer 2017: 52-53

Mee M (2017) 'Creating user forms: part 1' *Communicator*, Winter 2017: 44-46

Mee M (2018) 'Creating user forms: part 2' *Communicator*, Spring 2018: 34-37

Mee M (2018) 'Creating user forms: part 3' *Communicator*, Summer 2018: 30-33

Microsoft 'Declaring Variables' https://msdn.microsoft.com/en-us/VBA/Language-Reference-VBA/articles/declaring-variables (accessed August 2018)

Mike Mee MISTC
is a contract technical author.
E: mugukmail@gmail.com
T: @Mug_UK
W: www.mikestoolbox.co.uk
– my toolkit for Word 2007-2018