

VBA Made Easy

Objects, Properties and Methods

02

www.accessallinone.com

This guide was prepared for AccessAllInOne.com by:
Robert Austin

This is one of a series of guides pertaining to the use of Microsoft Access.

© AXLSolutions 2012

All rights reserved. No part of this work may be reproduced in any form, or by any means, without permission in writing.

Contents

Objects, Properties and Methods.....	3
Objects.....	3
Properties	3
Methods.....	4
The Recordset object	5
Collections	6
Objects, Properties and Methods – An Analogy	7
Programming with Objects	9
Questions.....	20
Answers	21

Objects, Properties and Methods

Objects

VBA is an object-based language and can interact seamlessly with Access objects (along with objects from other Office programs such as Excel and Word). In the physical world objects are things like tables, cars and people; in the VBA world objects are things like Tables, Queries, Forms, Reports, RecordSets, Buttons, Combo-Boxes, List-Boxes, Text-Boxes, Charts, etc.

The secret to programming with an object-based language is to understand how to manipulate these objects by taking advantage of their properties and methods.

Properties

Properties are said to be attributes of the objects they are attached to. As stated, command buttons are an object in VBA and contain various properties including the BackColor property, the Caption property, the Left property, the Top property and the ForeColor property. These are all attributes of the command button and help us to define various aspects of its appearance and behaviour.

With the properties listed above we can:

- Determine the color of the Command button (BackColor property)
- Determine what text is displayed in the Command button (Caption property)
- Determine how far from the left of the form or report the Command Button lies (Left property)
- Determine how far from the top of the form or report the Command Button lies (Top property)
- Determine the color of the text in the Command Button (ForeColor property)

In Figure 2.1 a button has been added to a form. We have manipulated a few of the properties listed above and because of this we can say that:

- The Back-Color property is set to **Accent 2, Darker 25%**.
- The Caption property contains the text ***This is a caption.***
- The Left property contains the value **1.998 cm**
- The Top property contains the value **1.998 cm**
- The ForeColor property contains the value **Background 1**

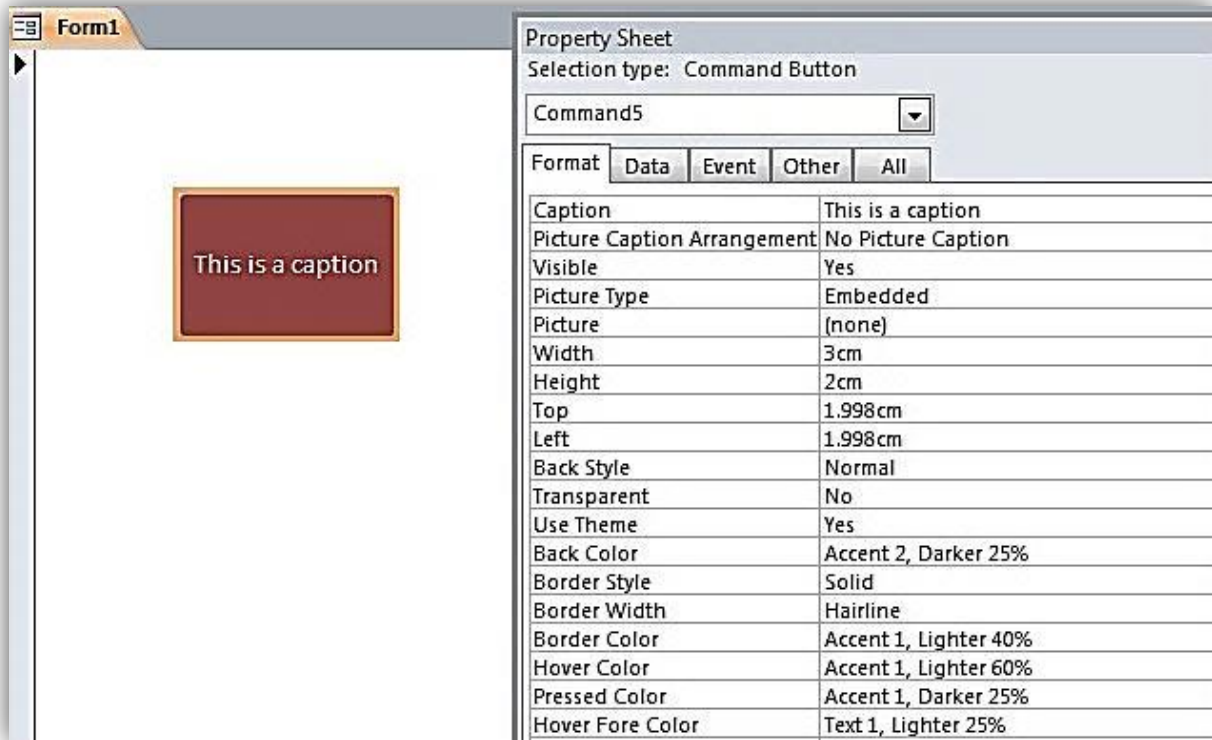


Figure 2.1

Using the property sheet feel free to change the values of certain properties for the Command Button in order to see how they affect the appearance and behaviour.

Note: In Layout or Design view on a form or report, if you can't see the property sheet go to the Design Tab in the Ribbon and then click the Property Sheet Button. (Alternatively, just press F4).



Figure 2.2

Methods

Methods can be described as actions that an object can perform. Objects such as Forms, Reports and Command Buttons don't have a lot of methods that they can perform so we will be introducing something called the recordset object to give you a feel for methods.

The Recordset object

We will be dedicating a whole unit to the Recordset object later on in the series but for now we will be providing a brief overview of the Recordset object, its properties and crucially its methods.

The Recordset object allows us to open a table or query in memory (which means we can't see it) and add, modify and delete records. Used with a For Loop (explained in a later unit) it is a powerful tool for helping us to manipulate data.

```
Option Compare Database

Public Sub EditRecordset()
    Dim i As Integer
    Dim db As Database
    Dim rs As Recordset

    Set db = CurrentDb
    Set rs = db.OpenRecordset("tblStudents")

    For i = 0 To rs.RecordCount - 1
        If rs.Fields("LastName") = "Ramos" Then

            rs.Edit
            rs.Fields("LastName") = "Dos Santos"
            rs.Update

        End If
        rs.MoveNext
    Next i

    rs.Close
    Set rs = Nothing
    db.Close
End Sub
```

Figure 2.3

In Figure 2.3 we open up a recordset for a table we have stored called “tblStudents”. We now have that table definition in memory and can loop through each record and update it if we wish.

We are asking Access to locate the record for a student whose last name is “Ramos” and then change the last name to “Dos Santos”.

In order to achieve this we have to use certain methods associated with the Recordset object including:

- **Movenext** – Moves to the next record.
- **Edit** – Ensures the record can be modified.
- **Update** – Writes any updates to the record.

These are all methods of the Recordset object and thus are defined as “actions that the recordset can perform”.

Note: We cover *Branching, For Loops and the Recordset object* in great detail later on in the course. Figure 2.3 is merely intended to provide you with an overview of how we use methods in VBA.

Collections

As we have stated forms, reports, queries and tables are all objects in an Access database. But normally we have more than one of each. We may have a table for students as well as a table to store courses, classes, teachers, etc. In the VBA environment we have the ability to refer to sets of objects of the same type using collections.

Collections are literally collections of objects of the same type (for example, there is a Forms collection and a Reports collection but not a Forms and Reports collection) and have their own properties and methods.

An example of a common property in a collection is the Count property. We use this to return the number of items in a collection. The Forms collection is highlighted in red.

```
1 Public Function IsLoaded(strFormName As String) As Boolean
2 Const conFormDesign = 0
3 Dim i As Integer
4 IsLoaded = False
5 For i = 0 To Forms.Count - 1
6     If Forms(i).FormName = strFormName Then
7         If Forms(i).CurrentView <> conFormDesign Then
8             IsLoaded = True
9             Exit Function
10        End If
11    End If
12 Next
13 End Function
```

Figure 2.4

In Figure 2.4 we use the form collection and the count property to help us ascertain whether a particular form is loaded. We refer to the Forms collections as Forms and the Count property as Forms.Count.

Note: We cover *Collections, Functions, For Loops and the If statement* in great detail later on in the course. Figure 2.4 is merely intended to provide you with an overview of how we use collections in VBA.

Objects, Properties and Methods - An Analogy

Most people tend to understand that if you want to change the text in a command button you use the Caption property and if you want to change the back colour, you use the BackColor property. It almost seems like killing a fly with a nuclear bomb to explain objects, properties and methods when, in truth, they can be quite intuitive. There is, alas, an ulterior motive behind these machinations!

The concepts surrounding object based languages and programming mean that it is very important that programmers are not only able to use objects, properties and methods but are also able to understand the thinking behind them.

Note: Taking some time now to really get to grips with these ideas will pay dividends in the future. (You may have to trust us on that one for now!)
So, as with all courses that deal with objects, properties and methods, we present an analogy! And for this particular analogy we shall be using a car!

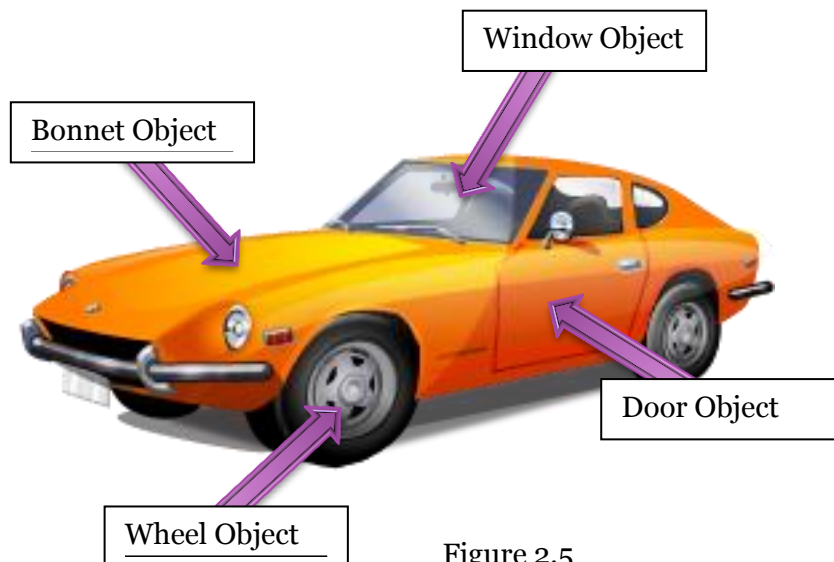


Figure 2.5

Cars are objects. Cars can contain other objects like a Wheel object and a Bonnet object. These objects can contain other objects themselves such as a Tire object for the Wheel object.

VBA works much the same way. A Form object can contain a Command Button object or a Combo box object or a Text box object. And just as a Form is part of a collection, so too can controls be part of a collection.

Taking the car analogy a little further we can make some observations about the car in Figure 2.5:

- The Car Object contains a Door Object.
- The Door object is part of a collection (Doors).
- The count property of the Door object is 2 (There are 2 doors).
- The Door object has a Color property.
- The Color Property of the Door object is set to Orange.
- The Door object contains a Window object.
- The Window object has a Broken property.

- The Broken property of any of the Window objects in this Car object is set to False (None of the windows are broken).
- The Car has a Drive Method.

Get the idea? Objects are things that have attributes (properties) and things they can do (Methods).

Programming with Objects

Parent and Children Objects

In Figure 2.6 we have created an object of type form and saved it (this form is in the downloadable Access content). We have added three control objects to the form (2 buttons and a text-box), changed some properties, and saved the form with its new objects. The control objects are *children* of the form, whilst the form is the *parent* of the control objects.

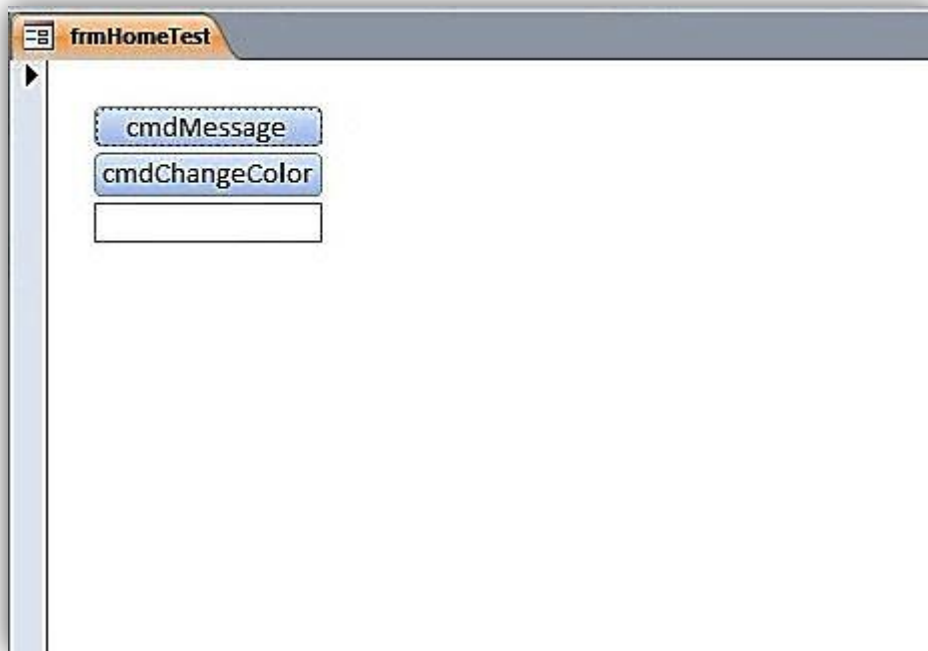


Figure 2.6

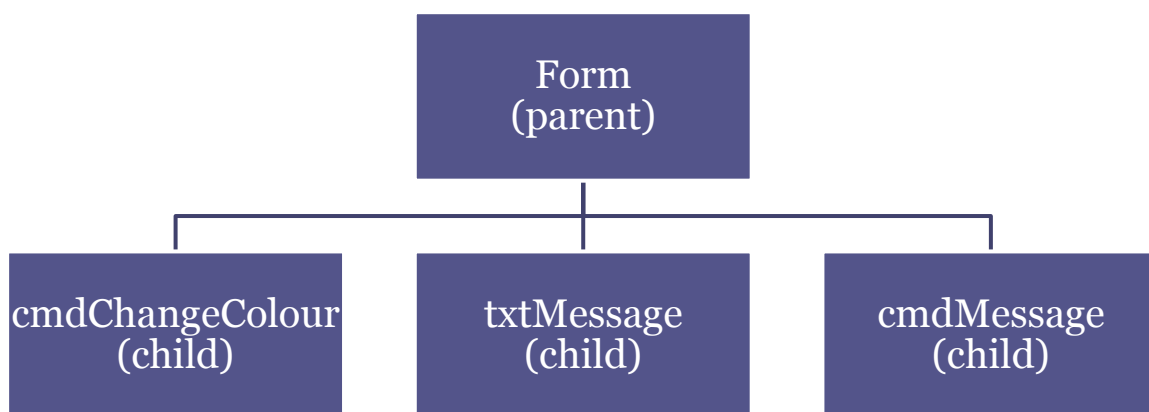


Figure 2.7

Accessing Objects and Properties, Me, the . (dot) Operator and !(exclamation)

We are going to get a message box to appear displaying the text “Hello World!” after clicking on the cmdMessage button. Here’s how we do it:

- Right-Click on frmHomeTest and choose design view from the drop-down box.

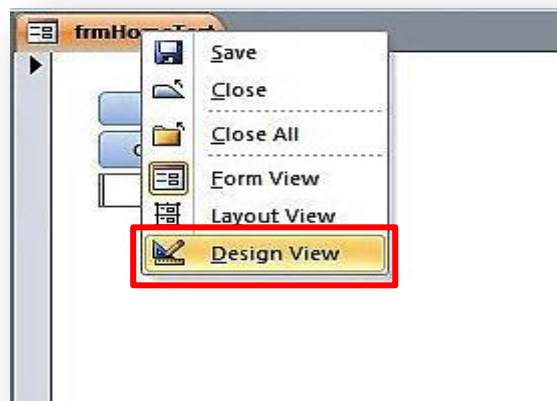


Figure 2.8

- Display the Property Sheet (Press F4).

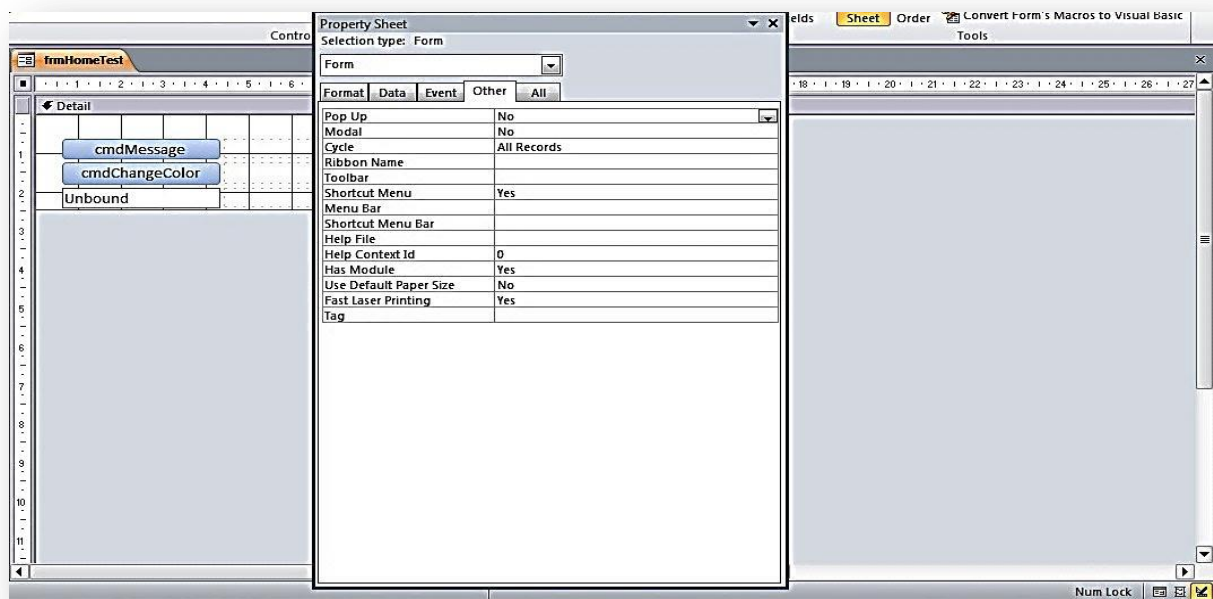


Figure 2.9

- Click on cmdMessage in the combo-box.

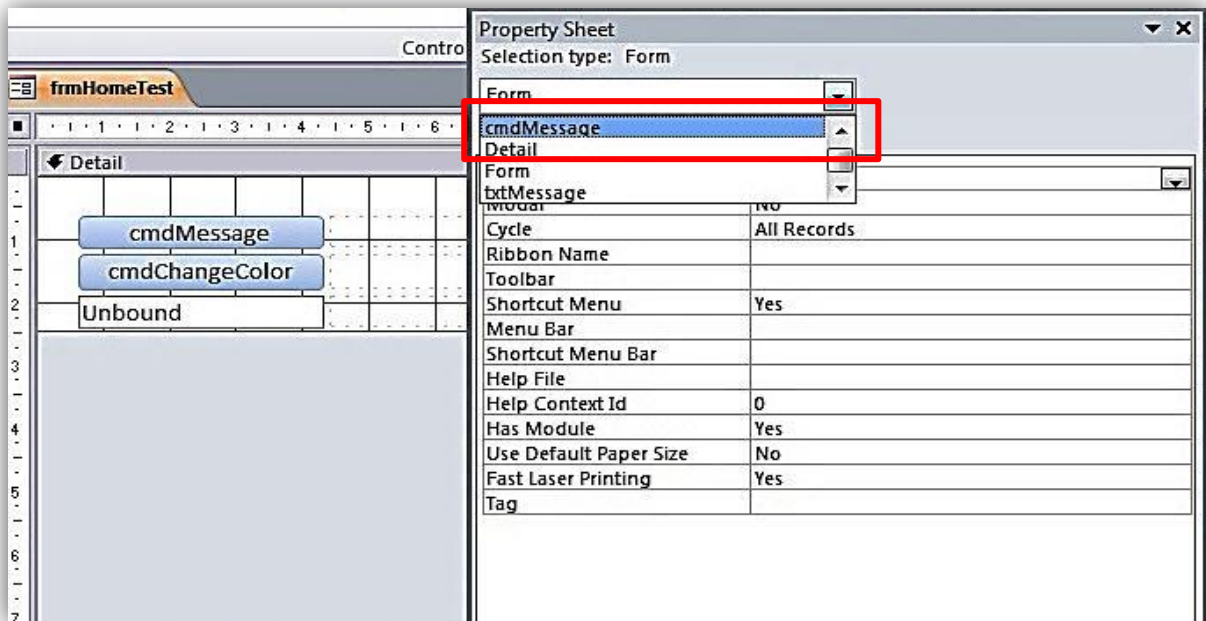


Figure 2.10

- Go to the Event Tab on the Property Sheet.

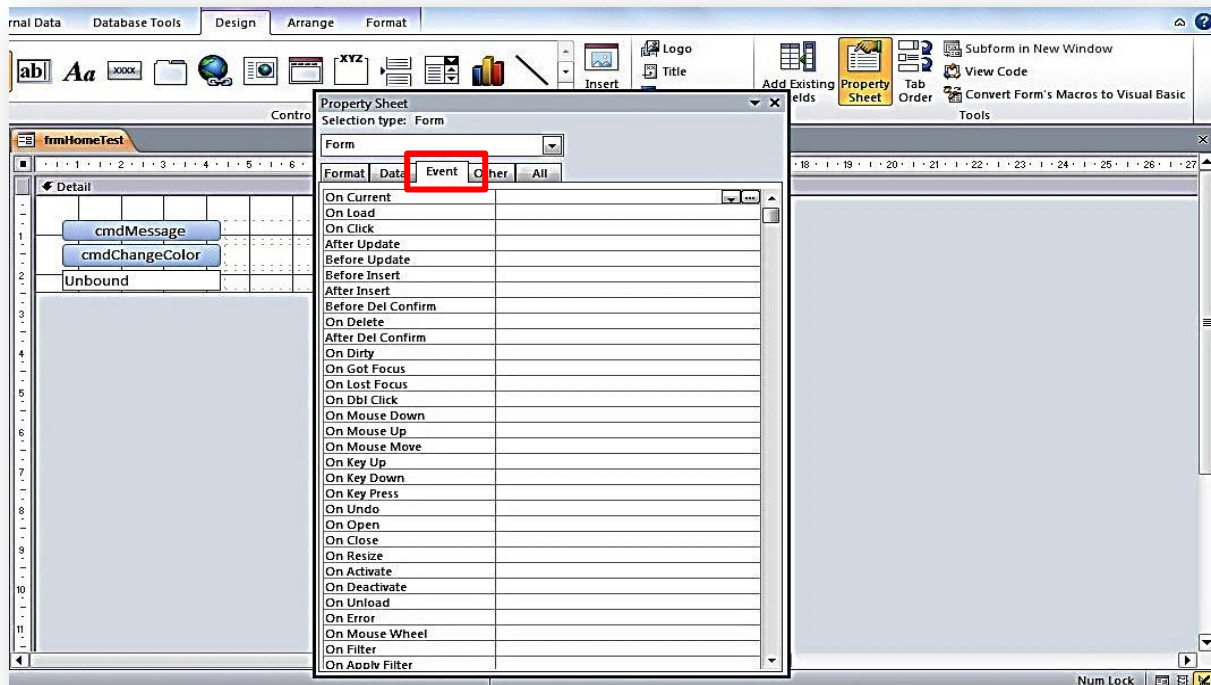


Figure 2.11

- Click on the ellipsis (the 3 dots on the far right) of the On Click Event.

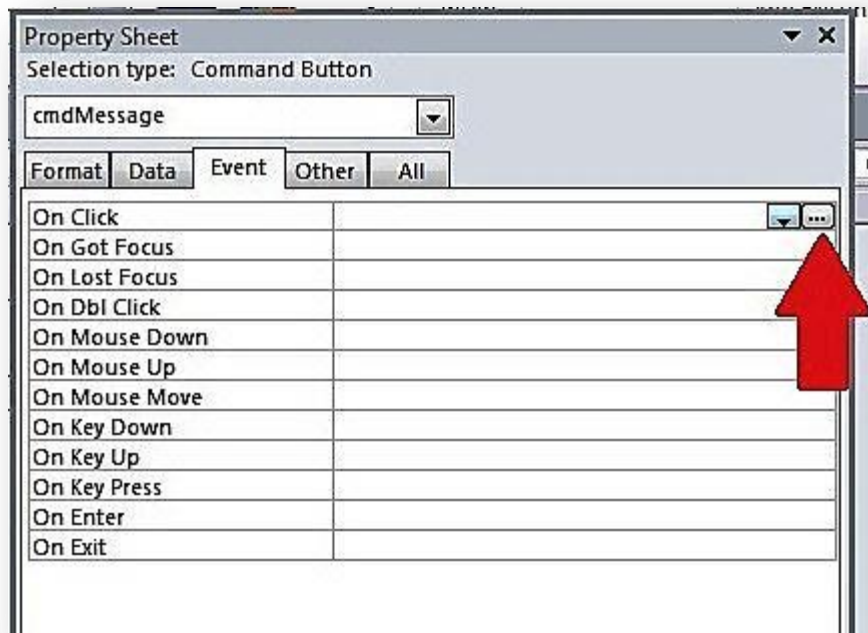


Figure 2.12

- Choose Code Builder.

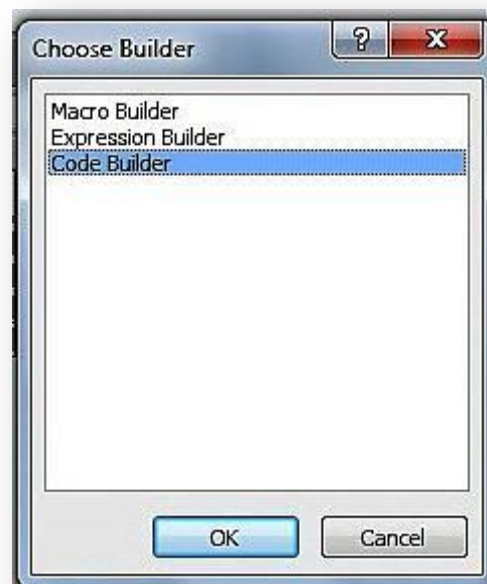


Figure 2.13

- The shell of a sub-procedure called cmdMessage_Click() will have appeared.

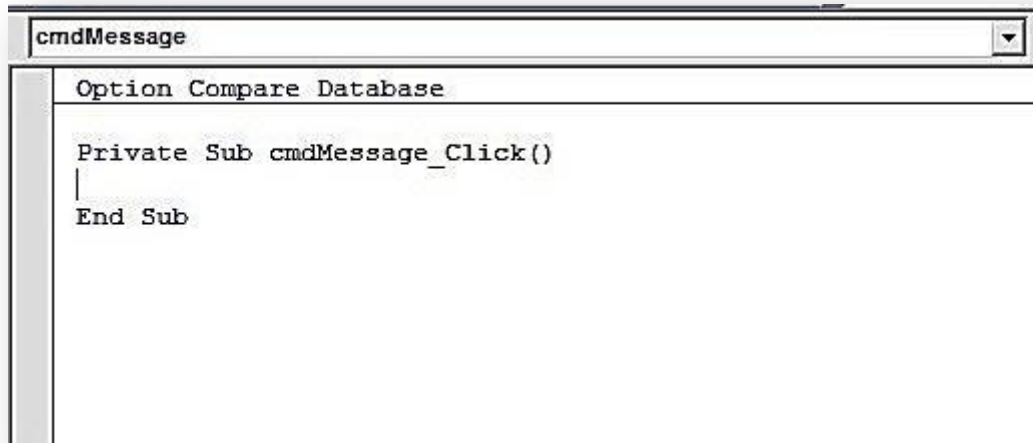


Figure 2.14

- Write *Msgbox "Hello World!"* on the line below `cmdMessage_Click()`.

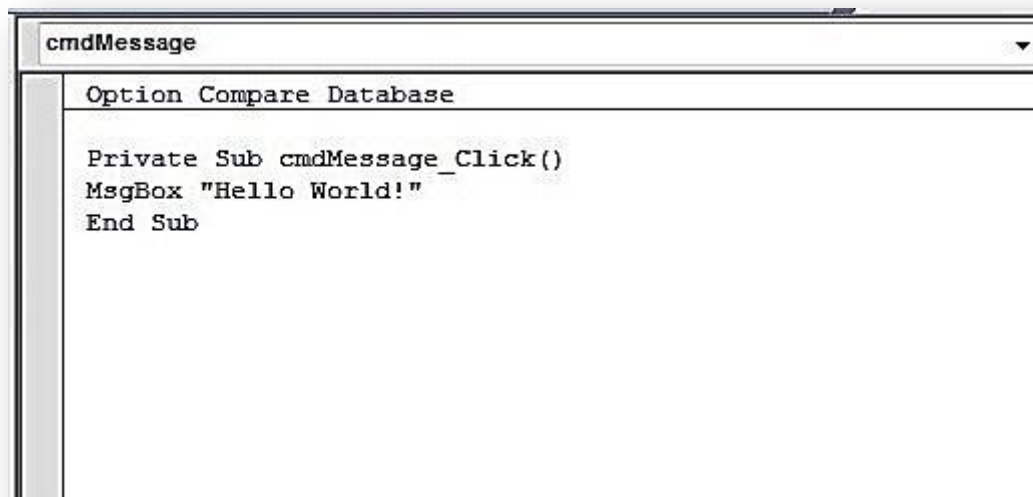


Figure 2.15

If we go back to `frmHomeTest` and change it to Form View, we can then click on `cmdMessage` and the following picture will be displayed:



Figure 2.16

We have created an OnClick event for the cmdMessage button and when we press it, a dialog box appears that contains the text string “Hello World!”. This text string has been **hard-coded**. This means that in order to change the value “Hello World!” we need to alter the code that was written. Sometimes hard-coding is a good thing (as end users don’t have access to the code and cannot change certain things) but other times you will want the end users to be able to change certain values. We are going to show how you can get the dialog box to display whatever you write in the txtMessage text box and how these objects can be referenced. But first a quick explanation of how the MsgBox object works.

When you write “MsgBox” and press the space bar once, a yellow box appears listing all the arguments that you can use. This “guide” that VBA provides you with is called intellisense and is a very useful feature.

Note: We will be covering arguments in greater detail later on in the course. For now, think of arguments as things that VBA needs to know to perform certain actions or would like to know (if they are optional). With regards to the msgbox object, VBA needs the Prompt argument (the actual message) otherwise it won’t have anything to display!

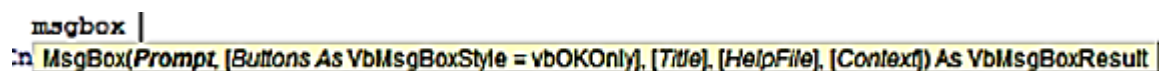


Figure 2.17

- MsgBox(- the open bracket indicates that this object takes arguments.
- Prompt is a text String to be displayed.
- [Buttons As VbMsgBoxStyle = vbOkOnly] – We use this optional argument to let VBA know what kinds of buttons we want. The default is the OK button.
- [Title], [HelpFile] and [Context] As VbMsgBoxResult – are [optional] and are Strings which we will not concern ourselves with for now.

So the msgbox object will display a text string as the message. In our case we have added a string called “Hello World!” But we don’t have to provide the MsgBox object with a *literal* string. We can *reference* an object or variable that contains a string and use that. In other words, we can write something in txtMessage and get the msgbox object to display that. Here’s how we do it:

- Delete the line *MsgBox "Hello World!"*
- Write *MsgBox* and then press the space bar.
- Write *me*. (not in quotation marks – make sure to include the dot).
- When the yellow box appears write *txt*.
- You should immediately jump to *txtMessage* (as indicated below).
- Choose *txtMessage*.
- Write *.Value* (don’t forget the dot).

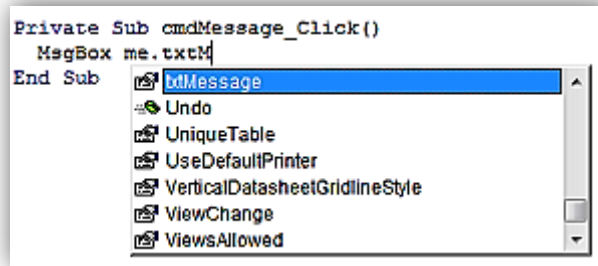


Figure 2.18

A couple of points on Figure 2.18.

- “me” – Refers to frmHome1. We are actually writing inside frmHome1 which is why it is highlighted in the Project Explorer.
- . (dot) – to get the controls we’ve added to the form. We can use the . (dot) operator to bring up a list of Properties of the “me” object. When we dragged the command buttons and textbox onto the form they were added to the Form’s list of Properties, and we set the Property’s name to txtMessage by changing its name in the Properties window.

Now to test it.

Navigate back to the Form Designer, change the mode to Form View.

Enter “Hello World Again!” into the textbox and click on txtMessage.

There it is!



Figure 2.19

Change the text in txtMessage and that is what will appear in the dialog box everytime you press cmdMessage.

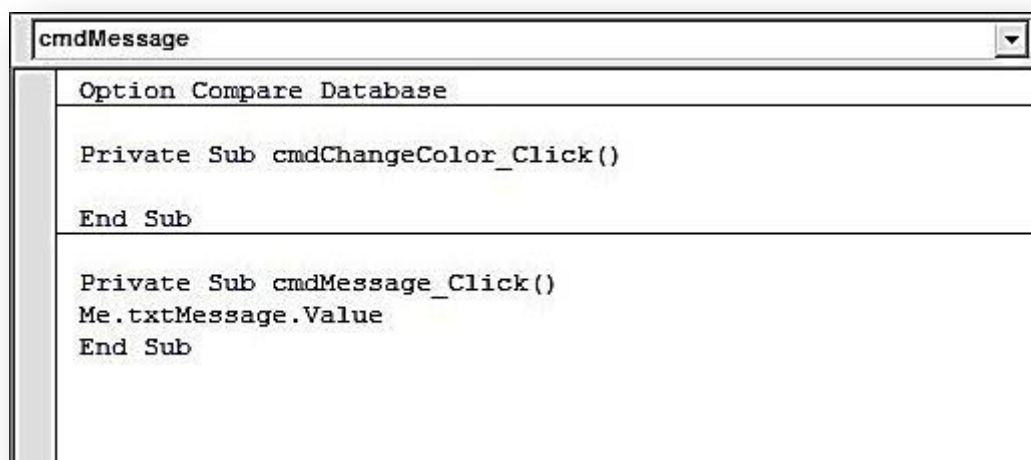
Note: Although you can use the dot operator as in Me.txtMessage.Value strictly speaking this should be Me!txtMessage.Value (notice the use of the exclamation mark instead of the dot operator). It is good practice to refer to objects with the exclamation mark and properties and methods with the dot operator.

However, if you don’t do that, the world will keep on spinning.

Changing an Object's Properties - textbox background colour

To demonstrate the changing / mutating of an object's properties we are going to do two extra tasks.

1. Change the text on the other button.
 2. Change the colour of the textbox background colour.
- Navigate back to the frmHome1 and change it to design view.
 - Click on the other command button we called cmdChangeColour.
 - On the Property Sheet click events and click the ellipses of the On Click event.
 - You should have something that looks like this:



```
cmdMessage
Option Compare Database

Private Sub cmdChangeColor_Click()

End Sub

Private Sub cmdMessage_Click()
Me.txtMessage.Value
End Sub
```

Figure 2.20

In the cmdChangeColour_Click() sub do the following:

- Type Me.cmd and find cmdChangeColour
- Put the .(dot) operator
- And find the property Caption

```
cmdChangeColor
Option Compare Database

Private Sub cmdChangeColor_Click()
Me.cmdChangeColor.Caption
End Sub

Private Sub cmdMessage_Click()
Me.txtMessage.Value
End Sub
```

Figure 2.21

Caption is the text that appears in the command button and it's this property we want to set.

- After caption put an “=” sign followed by the string “I Changed the Color!”

```
cmdChangeColor
Option Compare Database

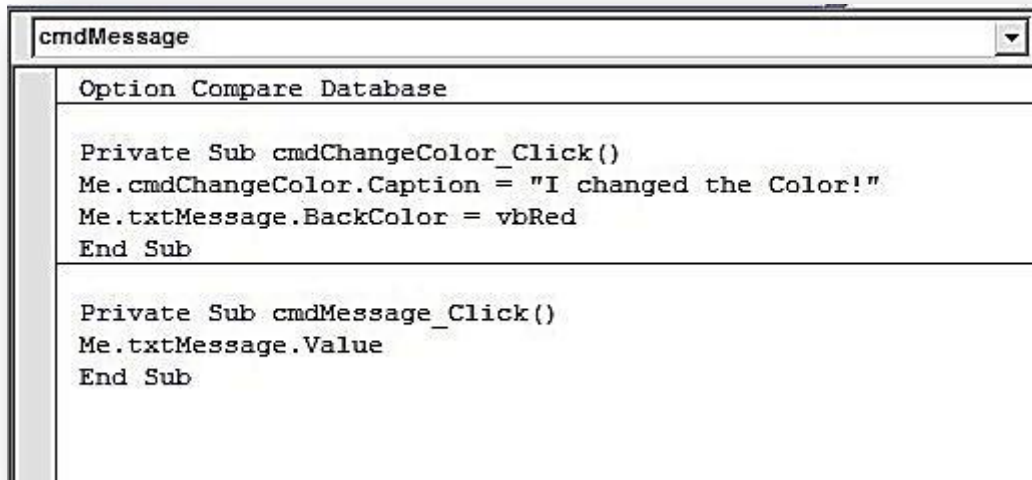
Private Sub cmdChangeColor_Click()
Me.cmdChangeColor.Caption="I changed the Color!"
End Sub

Private Sub cmdMessage_Click()
Me.txtMessage.Value
End Sub
```

Figure 2.22

On the next line, we will update the background colour of the textbox object.

- Write `Me.txtMessage.Value = VbRed`



```
cmdMessage

Option Compare Database

Private Sub cmdChangeColor_Click()
Me.cmdChangeColor.Caption = "I changed the Color!"
Me.txtMessage.BackColor = vbRed
End Sub

Private Sub cmdMessage_Click()
Me.txtMessage.Value
End Sub
```

Figure 2.23

Now go back to the form in Form View, click the button and it should look like this:

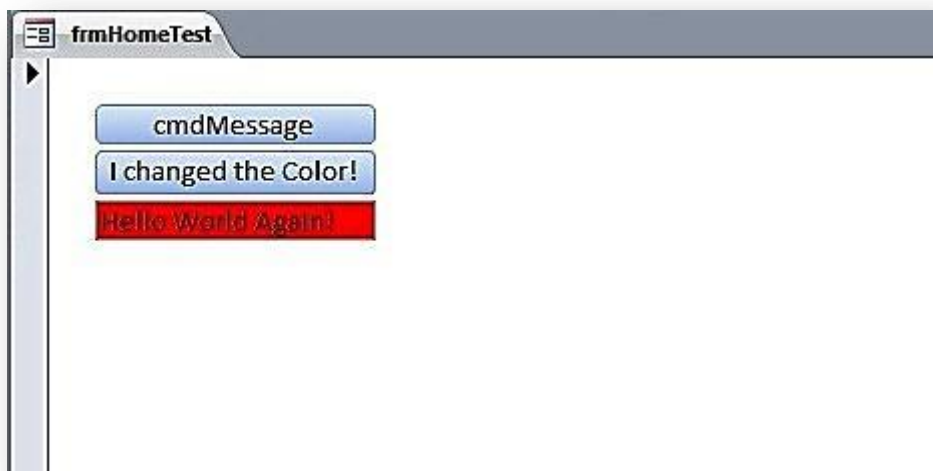


Figure 2.24

Questions

1. Which of these are objects and which are properties:
 - a. Command Button
 - b. BackColor
 - c. Form
 - d. Combo-Box
 - e. Left
 - f. Caption
2. Under what tab is the property sheet on the Ribbon?
3. What shortcut key do you press to bring up the property sheet?
4. Which property determines the color of the text in a command button?
5. Is Movenext a property or method of the recordset object?
6. What does the Count property of a collection tell us?
7. Using the car analogy for objects, properties and methods what category would the color of a tire fit into? Object or property. What object would own that color of a tire.
8. Create a blank form and name it frmObjects. Create a button for the form and name it cmdObjects. What is the relationship between frmObjects and cmdObjects?
9. What property do you change to display the text "This is a button" in the button?
10. How do you create a sub called *Private Sub cmdObjects_Click()*?
11. Add a text box and name it txtObjects. What code do you need to write in Private Sub cmdObjects_Click() to change the Back Color of the text box to vbBlue?
12. When can you use an exclamation mark after the keyword *Me*?

Answers

1. The objects are: a,c and d. The properties are b, e and f.
2. The Design Tab.
3. F4.
4. ForeColor.
5. Method.
6. The number of objects within that collection.
7. The color of a tire would be the Color property of the Tire object.
8. frmObjects is the parent object and cmdObjects is the Child Object.
9. The Caption property.
10. The steps to create *Private Sub cmdObjects_Click()*
 - a. Make sure the form is in design view.
 - b. Bring up the property sheet.
 - c. Click on cmdObjects or choose cmdObjects from the drop-down box in the property sheet.
 - d. Go to the event tab.
 - e. Click on the ellipsis on the far right of the On Click event.
 - f. Choose Code Builder.
11. Me.txtObjects.BackColor = vbBlue or Me!txtObjects.BackColor = vbBlue
12. When you wish to refer to an object that is the child of the form or report *me* refers to. E.g. Me!txtObjects (txtObjects is an object).