Access Forms Masterclass 5
Create Dynamic Titles for Your Forms

Published: 13 September 2018
Author: Martin Green
Screenshots: Access 2016, Windows 10
For Access Versions: 2007, 2010, 2013, 2016

Add a Dynamic Title to Your Form

Most Access forms need some sort of title. The title can simply tell the user what the form is for, a facility also provided by any text you add to the form's caption and displayed on the tab, but it can also provide a quick and easily visible reminder of which record is currently being displayed.

In this Masterclass I will lead you through the steps of creating a Dynamic Title for your form, one that changes automatically as you move from record to record, and when certain data on the form is changed.

A title should be both visually appealing and useful. This Dynamic Form Title fulfills both those aims (*Fig. 1*). It is very easy to create and in the process of building it you can learn a little VBA code.

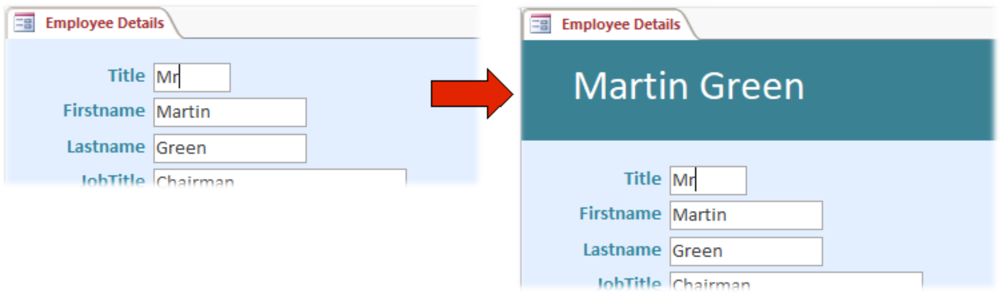


Fig. 1 Add a Dynamic Title to a form.

Step 1: Create a Form Header

We need somewhere to put the title. It can go anywhere but, to make it prominent, I like to place it in the Header section at the top of the form. If your form doesn't already have a header, switch it to Design View then **right-click** anywhere on the Detail area (the form's background) and choose **Form Header/Footer** from the context menu (*Fig. 2*).

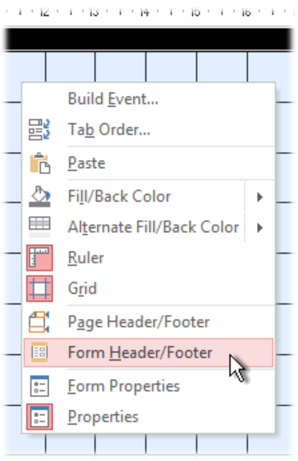


Fig. 2 Open the form header.

NOTE: Don't choose *Page Header/Footer* which is something different and not what we need for this exercise.

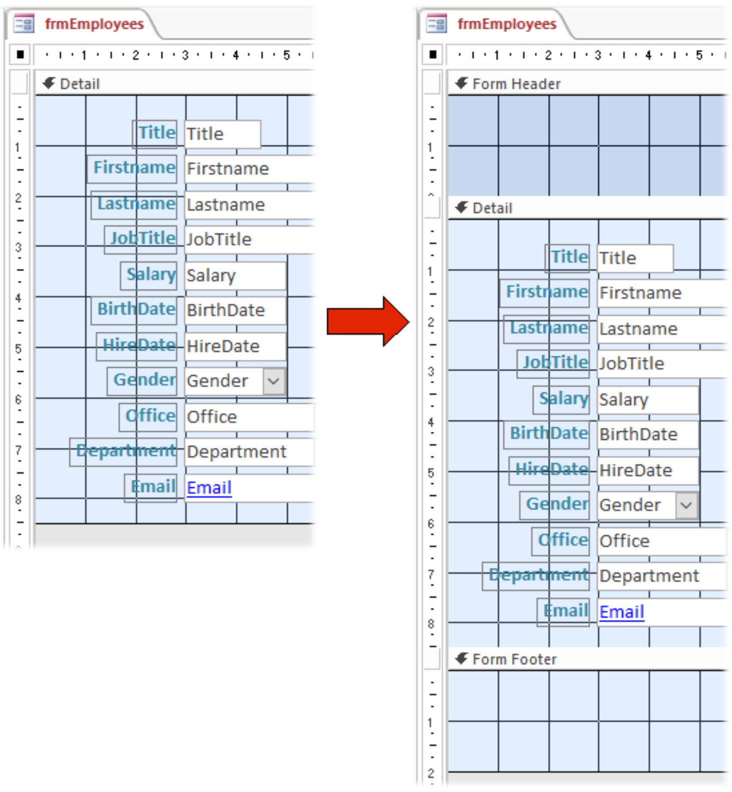


Fig. 3 Access adds a Header and a Footer to the form.

When you do this Access opens both Header and Footer areas on the form (*Fig. 3*). They open at 2cm in height, but you can easily adjust their size by pointing either at the base of the footer or at the lower edge of the header when a double-headed arrow cursor will appear (*Fig. 4*). Use this to drag up or down to change the section's height.

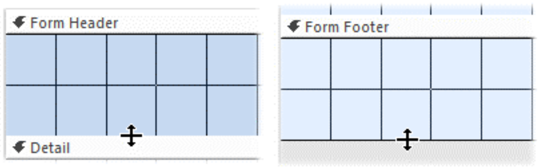


Fig. 4 Use the resize cursor to change the height of the Header or Footer.

I like to make the header section of my forms a contrasting colour. To do this **right-click** on the background of the Header and choose **Fill/Back Color** from the context menu, then pick a suitable colour from the palette (*Fig. 5*). For a greater choice of colours **right-click** on the background of the Header and choose **Properties** to open the Properties Sheet (if it isn't already open) and go to the *Back Color* property on the *Format* tab where you can choose one of the presets on the drop-down list or click the Build button (**[...]**) to open a detailed colour chooser.

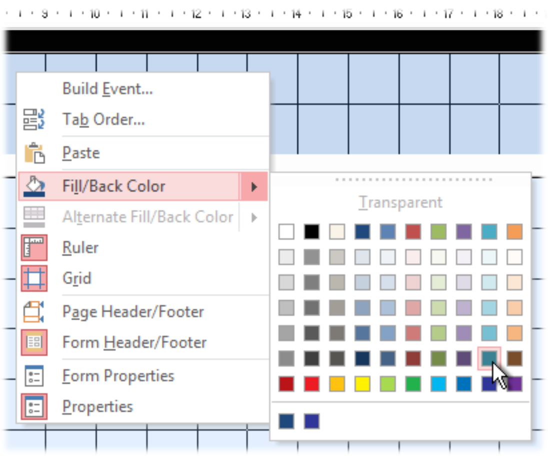


Fig. 5 Choose a background colour for the Header.

Step 2: Add the Title

The purpose of the Dynamic Title is to show the user, at a glance, which record they are looking at, by replicating some of the data on the form. In my example, the form is displaying the details of a company employee so I want the title to clearly display their name by making use of the data in the *Firstname* and *Lastname* fields. There are two ways to do this:

Method 1: Using an Unbound Text Box

This method does not require any coding and is very easy to do (but I prefer Method 2 for reasons I'll explain in a moment). Select the Text Box tool (*Fig. 6*) from the Design tab of the ribbon and use it to draw a rectangle on the form's Header. Usually a Text Box is linked ("bound") to an underlying field but this one will be "unbound" and display data that we will create.

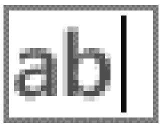
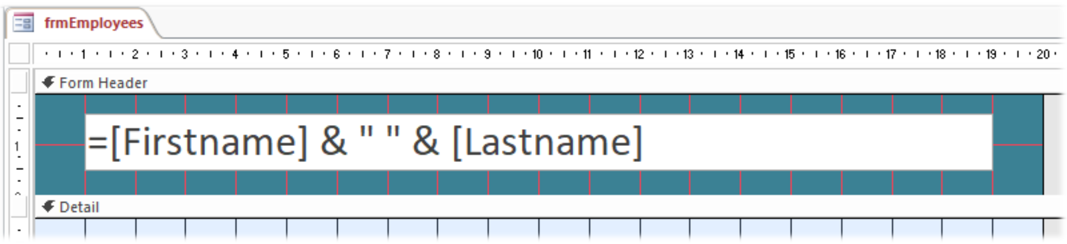


Fig. 6 The Text Box tool.

Click in the Text Box and enter a suitable expression such as:

=[Firstname] & " " & [Lastname]

This expression simply concatenates (joins together) the data in the *Firstname* and *Lastname* fields with a space in between (*Fig. 7*). The space is text so must be enclosed in quotes.



=[Firstname] & " " & [Lastname]

Fig. 7 Add a suitable expression to create the Text Box entry.

The Text Box will need to be wide enough to accommodate the widest entry that it will have to display, and its text should be a suitable colour and large enough to stand out. I made the following changes using the tools on the *Format* tab of the ribbon (*Table 1*).

Table 1: Text Box Properties.

Tool	Value
Font Size:	24 pt
Font Color:	White
Shape Fill:	Transparent
Shape Outline:	Transparent

TIP: When you change the size of the text in your Text Box it probably won't fit the box you drew. To make the box snap to the size of the text it contains, select the Text Box then double-click the dot in its lower-right corner. You can then change the width of the Text Box by dragging the dot at the centre of its right edge.

Switch the form into Form View to view the result (*Fig. 8*). You will see that as you move from record to record the title changes to reflect the values in the fields used in its expression.

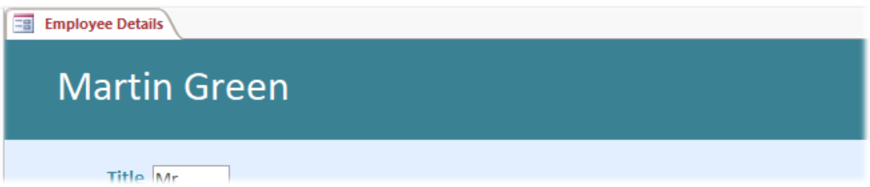


Fig. 8 The Text Box displays the specified data.

Here comes the problem! I mentioned earlier that I preferred not to use this method. To see why, in Form View click on the title. The text box is selected and if, like me you used white text, the title disappears (*Fig. 9*).

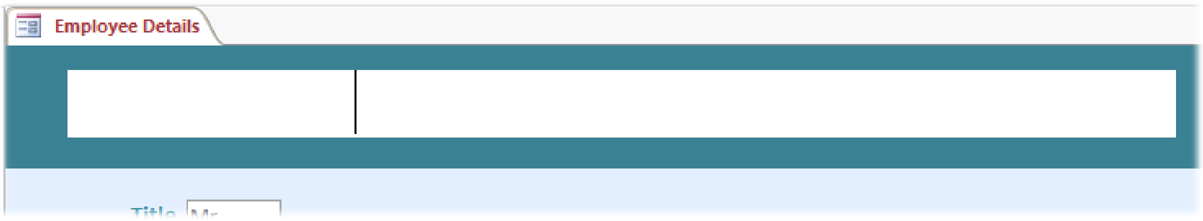


Fig. 9 If the user clicks on the title its contents disappear.

If the user can select the Title they might think that they are supposed to enter its contents but, since it is an unbound textbox containing an expression, Access won't let them type there (nor do we want them to!). Even if the text box is locked it can still be selected. It looks ugly and is guaranteed to confuse the user, hence Method 2…

Method 2: Using a Label

As described in Method 1, but this time using the Label tool (*Fig. 10*) draw and format a rectangle and enter some default text (the text of a Label control is called its *caption*) such as *Employee Details*.

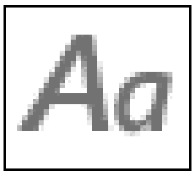


Fig. 10 The Label tool.

TIP: if you have already created the title in the form of a Text Box you can save time re-drawing and formatting a new object by simply changing the existing Text Box to a Label. To do this **right-click** on the Text Box and from the context menu choose **Change To** then **Label** (*Fig. 11*).

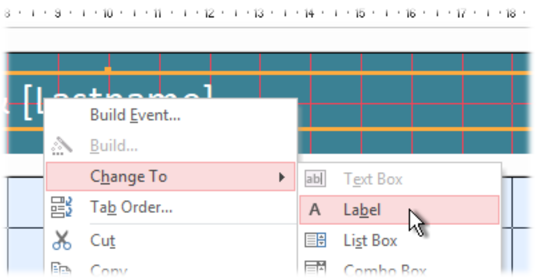


Fig. 11 Change an existing Text Box control to a Label control.

After entering some default text check the result in Form View. It looks just as it did before (*Fig. 12*) except the text does not change as you move through the records and, importantly, it can't be selected.

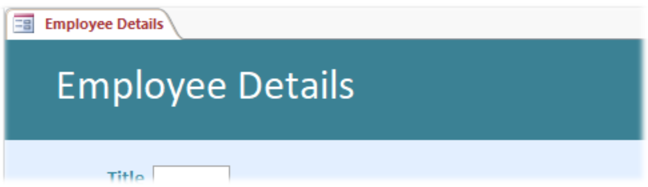


Fig. 12 The Form Title created using a Label control.

We are going to be referring to our new Label control in VBA code so we need to give it a meaningful name. In Form Design view select the Label control and go to the *Other* tab of its Property Sheet and change its *Name* property to, for example, *lblFormTitle.*

Step 3: Program the Title

The next task is to make the caption of the Title Label change automatically to reflect the values in the relevant fields when the user moves from record to record. To achieve this we need to write some VBA code.

The VBA we are going to use is known as Event Driven programming. This means that we create macros that run automatically when a particular *Event* happens. We need the Label's caption to update when the form opens and when it moves from record to record. Both of these cause the *On Current* event to fire ("fire" is propellorhead language for a programmable event happening). We also need the label's caption to update when the Text of the appropriate fields changes (in my example the *Firstname* and *Lastname* fields). To achieve this we can make use of the *After Update* event of each of these fields.

With the form in Design View click the **View Code** button (*Fig. 13*) on the **Design** tab of the ribbon.

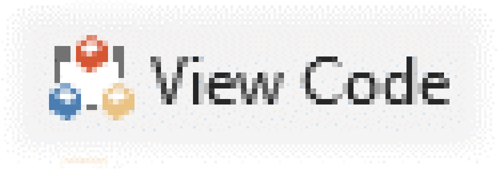


Fig. 13 The View Code button.

This takes you directly to the form's code module in the Visual Basic Editor. We will start by creating a macro that, when run, will update the Label's caption. Type the following code statements (*Listing 1*):

Listing 1:

```
Private Sub UpdateTitle()
    On Error Resume Next
    Me.lblFormTitle.Caption = Me.Firstname.Value & " " & Me.Lastname.Value
End Sub
```

In addition to the instruction to write the Label's caption, I have added a simple error handler to prevent problems in the unlikely event that something should go wrong. I have also made the macro a *Private Sub* rather than just a *Sub*. Making it Private means it can only be run from within this module.

Next, we need to write our *Event Procedures* that will instruct the *UpdateTitle* macro to run. This is known as *Calling* the macro.

At the to of the code module window there are two drop-down lists. We can generate our empty Event Procedures using these lists rather than having to return to the form's design view each time.

Open the left-hand list (currently showing **(General)**) and choose **Form**. The Visual Basic Editor immediately creates an empty *Form_Load* event procedure, which happens to be the default, but it isn't what we need. Instead, open the right-hand list and choose **Current** (*Fig. 14*) to create an empty *Form_Current* event procedure.

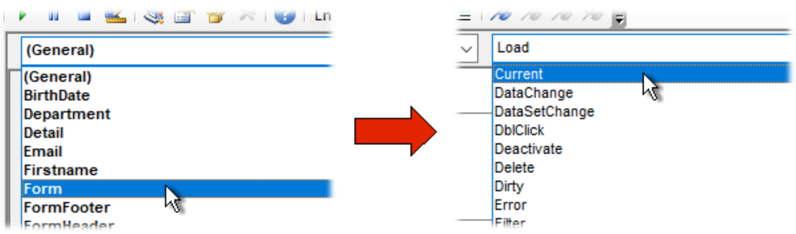


Fig. 14 Choose items from the drop-downs to create an empty Event Procedure.

We don't need to use the *Form_Load* event so select and delete the text that the Visual Basic Editor created for it.

Edit the Form_Current code as follows (*Listing 2*):

Listing 2:

```
Private Sub Form_Current()
    On Error Resume Next
    Call UpdateTitle
End Sub
```

Repeat the process choosing **Firstname** and **AfterUpdate** then **Lastname** and **Afterupdate** (if you have different field name use whatever is appropriate for your needs). Again the Visual Basic Editor, trying to be helpful, supplies the *BeforeUpdate* event procedures so simply delete the unwanted text as you did earlier.

You should now have three event procedures, one for when the form opens and moves through the records, and one each for the fields used to create the Label Caption. Each one *Calling* the same macro.

Before testing your code, open the Visual Basic Editor's **Debug** menu and choose **Compile...** (*Fig. 15*). This causes the Visual Basic Editor to check the code for any errors that you might have missed when writing it.

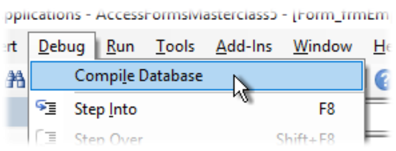


Fig. 15 Compile the code before testing.

Assuming everything is OK (if not then check your code and correct any mistakes) click the Visual Basic Editor's **Save** button or press **[Ctrl]+S** to save your changes.

Switch your form into Form View and see how the text of the Title changes when you move through the recordset and when the value of the fields used to create it change.

Your form's title is now dynamic.

Some More Ideas for Dynamic Titles

My VBA method (*Method 2*) for defining a Dynamic Form Title builds a string that concatenates field data with regular text. The original example:

```
Me.Firstname.Value & " " & Me.Lastname.Value
```

Uses the *Firstname* and *Lastname* fields separated by a space and concatenated with the ampersand (**&**) character to create, for example:

Martin Green

You can add as many fields as the available space permits and include any text you want, such as:

```
    Me.Firstname.Value & " " & Me.Lastname.Value & " (" & Me.JobTitle.Value & ")"
```

… which adds the *JobTitle* field enclosed in parentheses:

Martin Green (Chairman)

You can even include a calculation such as the next example which makes use of the *Birthdate* field to calculate the person's age*:

```
Me.Firstname.Value & " " & Me.Lastname.Value & " - Age: " & Int((Date -
Me.BirthDate.Value) / 365.25)
```

… resulting in something like:

Martin Green – Age: 67

Here is another example, this time using four of the form's fields (*Lastname*, *Firstname*, *Department* and *Office*) along with some text*:

```
Me.Lastname.Value & ", " & Me.Firstname.Value & " : " & Me.Department.Value & "
Dept. " & Me.Office.Value
```

This generates a title in the format:

Green, Martin : Management Dept. London

*NOTE: In the third and fourth examples above the width of this document has forced the code to flow on to a second line. Remember, when writing your code in the Visual Basic Editor that you should keep each expression on a single line. If you choose to break a line of code you must use the line break character (a space followed by an underscore) to tell the Visual Basic Editor that your code statement continues on the next line, for example:

```
Me.Lastname.Value & ", " & Me.Firstname.Value & " : " & Me.Department.Value _
& " Dept. " & Me.Office.Value
```

Download Example Database

You can download a sample Access database containing completed examples of the form shown in this Masterclass. The database contains three copies of the Form: one using an unbound Text Box control to create the Title, another using a Label control driven by VBA code, and a third showing the alternative examples described above.

Download the sample database file from:

http://www.fontstuff.com/access/files/AccessFormsMasterclass5.zip