Access Forms Masterclass 3
Go to Record – A Custom Record Locator

Published: 14 August 2013
Author: Martin Green
Screenshots: Access 2010, Windows 7
For Access Versions: 2007, 2010, 2013

A Custom Record Locator

A common requirement for database users is to quickly locate a specific record. Whilst Access provides a number of tools for sorting and filtering a form's recordset there isn't a built-in tool to help the user quickly find and display a specific record. If you want one, and you almost certainly will, you will have to build it yourself. I have built several different kinds of *Go to Record* tools to suit my users' requirements and the complexity of the data. This is one of my favourites and is really easy to build (*Fig. 1*).

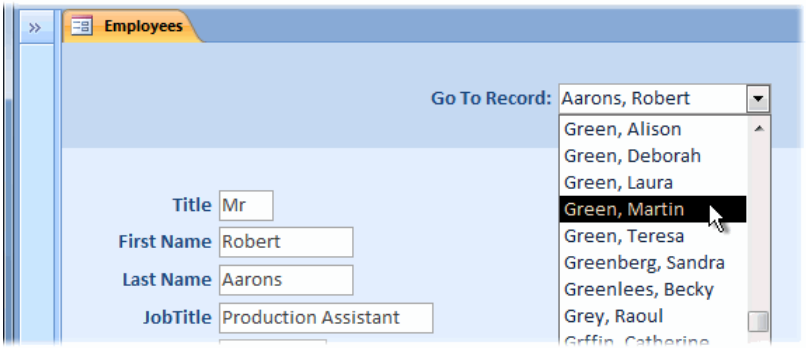

Fig. 1 The Custom Record Locator.

How It Works

This *Go to Record* tool uses a combo box to display a list of items, each one identifying a specific record, from which the user makes a choice. The form then jumps to that specific record.

Because the form simply moves to the specified record in the recordset no filtering is involved so the user does not have to remember to remove the filter to see the full recordset again.

Any field or combination of fields can be used to identify the record so that might be, for example, a company name or a person's first and last names. The process makes use of the record's primary key (which the user does not need to know) to identify it and locate it in the recordset.

The Combo Box Wizard that runs when you add a combo box to a form includes an option to create a tool similar in action to this but, as it stands, it is of limited flexibility. My *Go to Record* tool is powered by VBA and SQL to offer as much flexibility as you could need for this task.

Prerequisites

There are only two prerequisites. First, that the recordset that the form is displaying includes the primary key field. That field need not necessarily be displayed for the user to see as long as it is included in the recordset. And second, that the recordset also includes one or more fields that will allow the user to accurately specify a particular record.

Step 1: Add a Combo Box to the Form

You can place the combo box anywhere on the form. I usually locate it on the form's header or footer section. A header and footer are not displayed by default on an Access form. If they are not visible you will need to add them. In the form's **Design View** right-click on the **Detail** area of the form (the form background) and choose **Form Header/Footer** from the context menu.

Add the Combo Box

In the form's **Design View**, click the **Combo Box** tool on the **Controls** section of the **Design** tab of the ribbon (*Fig. 2*). The cursor changes to indicate that it is ready to place a combo box on the form.

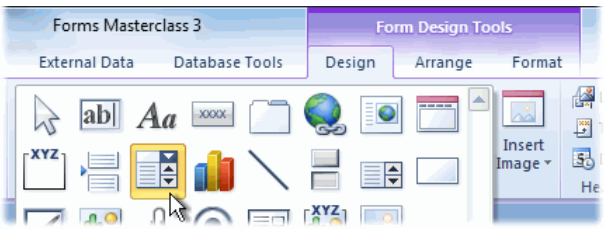

Fig. 2 Select the Combo Box tool.

Click on the form approximately where you want the combo box to appear (*Fig. 3*). You can accurately size and position it later.

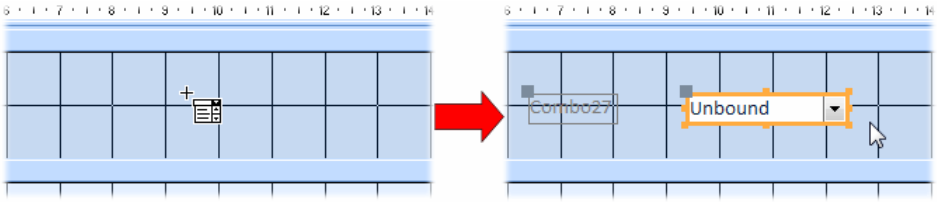

Fig. 3 Add a combo box to the form.

Change the Name and Caption

Access has already assigned a default name to the combo box and captioned the accompanying label. With the combo box selected go to the **Property Sheet** (if it is not visible right-click the combo box and choose Properties from the menu) and on the **Other** tab change the **Name** property to *cboGoToRecord*. At the same time you can optionally add some text, such as *Go To Record* to the **Control Tip Text** property. This will cause a helpful Tool Tip to appear when the user points their mouse at the combo box (*Fig. 4*).

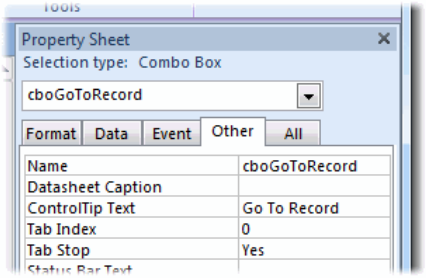

Fig. 4 Edit the Name and Control Tip Text properties.

Select the label that Access placed next to the combo box and on the **Format** tab of the **Property Sheet** change the **Caption** Property to *Go To Record:*. You will probably find that the label is now too small to accommodate the new text. Drag the resizing handles around the perimeter of the selected label to change its size. To move the label, drag the large grey dot in its upper-left corner.

You can also format the label, changing the font, size and colour using the tools on the **Format** tab of the ribbon. Check the finished appearance of the combo box and its label by switching the form into Form View (*Fig. 5*).

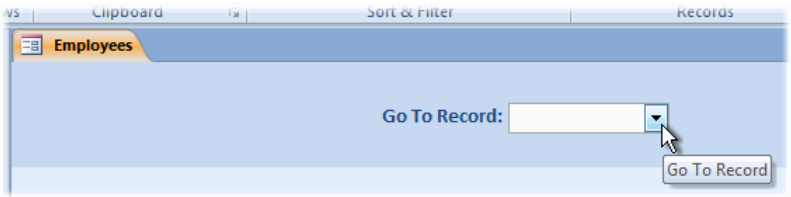

Fig. 5 The combo box and its formatted label.

The combo box does not have a list yet. That is the next task.

Step 2: Create the Combo Box List

The list that the combo box will display will be derived from the recordset that the form is displaying. Decide which field, or combination of fields, the user will use to specify which record

they want to see. This may be the Primary Key field itself or another field or combination of fields. I usually use an Autonumber field for the Primary Key, which will probably have little or no meaning to the user, but if your primary key is meaningful to the user you could use that.

In **Design View** select the new combo box and go to the **Data** tab of its **Property Sheet**. Make sure that the **Row Source Type** property is set to *Table/Query* then click in the **Row Source** property text box and click the **Build** button **[...]** (*Fig. 6*). This opens the *Query Builder* tool and displays the *Show Table* dialog box.

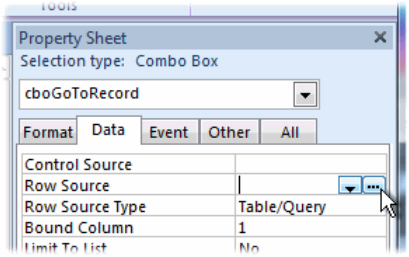

Fig. 6 Click the Build button of the Row Source property.

Select the name of the table that the form's recordset is based upon (*Fig. 7*), click the **Add** button then click **Close** to dismiss the dialog box.

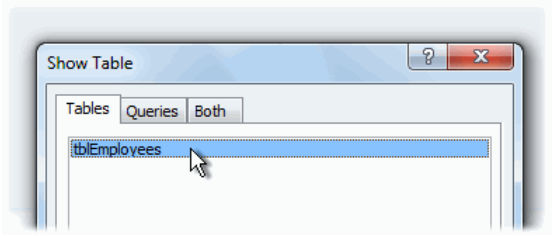

Fig. 7 Choose the table to add.

The Query Builder is now displayed along with a field list for the table you chose. You are going to choose a selection of fields that will be used to create the list that the combo box will display. One of the fields must be the Primary Key field and this should be added first. In the field list it will have a small icon of a Key against its name. Double-click the Primary Key field name to add it to the query design grid (*Fig. 8*).

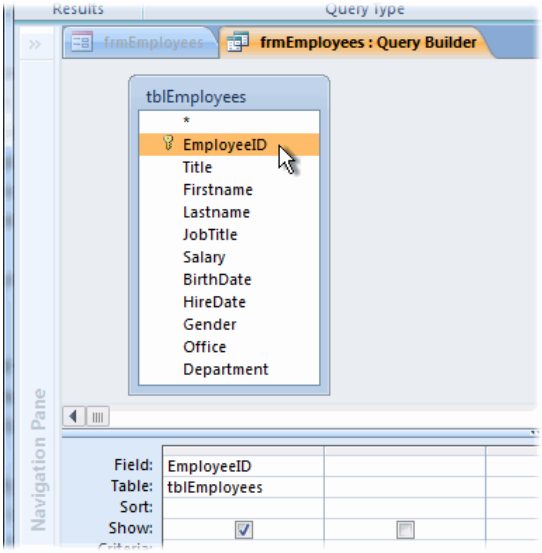

Fig. 8 Add the Primary Key field to the grid.

What you do next depends upon your data and the fields you need to identify specific records. If this were, for example, a database of business customers for which there was a single entry for each customer, you could use a single field such as the Company Name.

In this example the recordset contains a list of employees all of whom can be identified by the combination of their first and last names. (*NOTE: It is possible that there could be two people with*

the same first and last name, in which case an additional field would be necessary to separately identify them.)

I could add these two fields, in the order I wished to see them in the list, simply by double-clicking on each as I did for the Primary Key. But I prefer to combine them into a single field (*Fig. 9*). I want my list to show the person's Last Name followed by a comma and a space and then their First Name e.g. *Green, Martin*.

To do this, instead of adding the fields to the grid, in the **Field** cell at the top of the empty column to the right of the Primary Key in the grid, enter a name for the new field followed by a colon (**:**) and the names of the fields to be combined as follows:

Fullname: [Lastname] & ", " & [Firstname]

You have now defined a new, calculated field. It is also important to sort the necessary fields into ascending order to assist the user in finding the required item in the list. To do this click in the **Sort** row of the appropriate column of the grid and choose **Ascending** from the list.

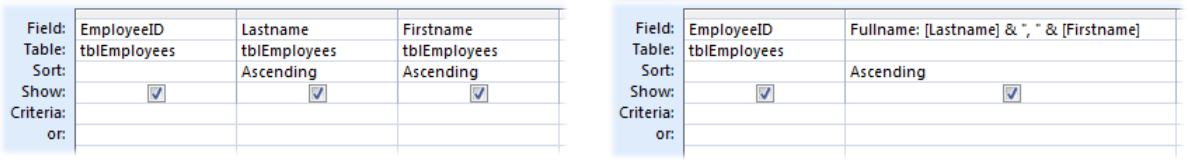

Fig. 9 Multiple fields can be added separately (Left) or combined (Right).

Finally, click the **Run** button on the **Design** tab of the ribbon to run the query and check that the list it creates is correct (*Fig. 10*). If you need to alter anything return to the Query Builder by clicking the **Design View** button on the **Home** tab of the ribbon and make the necessary changes.

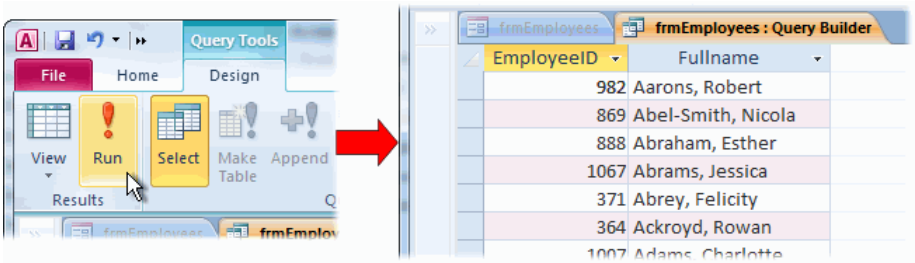

Fig. 10 Run the query and check the list.

When you are satisfied with the list, close the Query Builder. Access displays a message asking you to confirm that you want to return the SQL statement that the Query Builder created and use it to build the combo box list (*Fig. 11*). Click **Yes**.

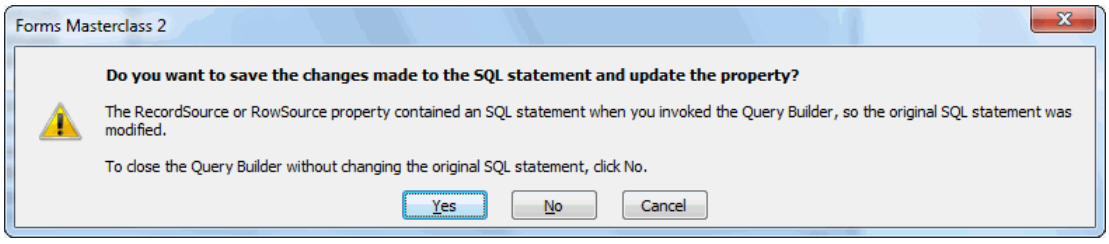

Fig. 11 Access asks for confirmation.

The SQL statement that the Query Builder generated is now written into the *Row Source* property of the combo box.

If you switch the form into Form View and open the combo box list you will see that only the Primary Key field is displayed (*Fig. 12*). This is because, unless you specify otherwise, a combo box displays only the first field in the list's recordset.

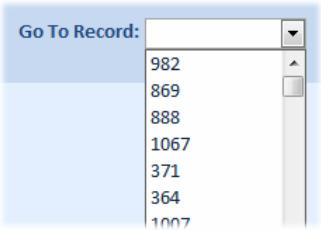

Fig. 12 The combo box displays the first column.

You must tell Access how many columns you want the combo box to display and also define their widths. If, as in this example, you don't want to display the Primary Key column simply set its width to zero. The combo box will then display the first visible column instead, together with any others that you have specified.

In **Design View** select the combo box and on the **Data** tab of the **Property Sheet** check that the **Bound Column** property is set to **1**. The value of the *Bound Column* becomes the *Value* of the combo box when the user makes a selection (even if the Bound Column is not visible). We need the Bound Column to be the Primary Key column.

Then, on the **Format** tab of the **Property Sheet** change the **Column Count** property to the total number of columns that you defined in the Query Builder. In this example that value is **2** but if you used more columns enter the appropriate number.

Next, in the **Column Widths** property enter the required width of each column, starting with a zero (to hide the Primary Key column) and separating each with a semicolon (**;**). You can just type the numbers, there is no need to enter *cm* or *in* (*Fig. 13*).

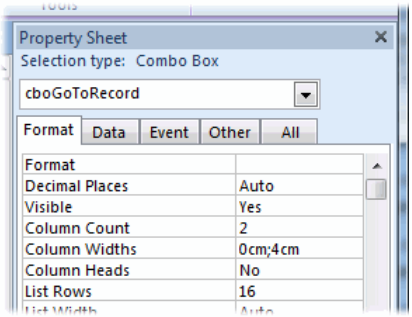

Fig. 13 Specify the Column Count and Column Widths.

This might involve some trial and error to get it right. Check the result in Form View and adjust the measurements until it looks right. You might also need to adjust the **Width** property of the combo box so that it is wide enough to accommodate the list (*Fig. 14*).

Fig. 14 The finished combo box.

Now we have a combo box that displays a list of items each of which can be used to identify a specific record. The next step is to write the VBA code that will turn this combo box into our *Go To Record* tool.

Step 3: Write the VBA Code

The next task is to create the code that will instruct the form to display the record whose Primary Key field has the same value as the hidden column in the item chosen by the user in the combo box. The appropriate place for this code is in the combo box's *After Update* event procedure which fires when the user makes a choice from the combo box list.

In the form's **Design View** select the combo box and go to the **Event** tab of the **Property Sheet**. Click on **After Update** then click the **Build** button (**[...]**) to open the **Choose Builder** dialog box, and choose **Code Builder** (*Fig. 15*) then click **OK**.

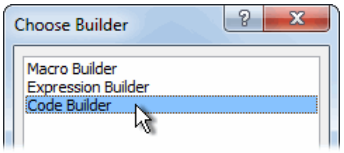

Fig. 15 Open the Code Builder.

This takes you into the Visual Basic Editor where Access has created an empty event procedure and placed your cursor inside, ready for you to enter the code (*Listing 1*).

Listing 1:

```
Private Sub cboGoToRecord_AfterUpdate()

End Sub
```

To start, we need an error handler just in case something goes wrong. A simple one will do. Press **[Tab]** to indent your code then type:

On Error Resume Next

The code will make use of the form's *RecordsetClone* property. This is a copy of the form's underlying recordset that we can manipulate with code. Since this is a VBA object it must be declared as an object variable (which I shall call "rst") and defined by having a value assigned to it. To do this, enter the following two code statements:

Dim rst As Object
Set rst = Me.RecordsetClone

The next statement instructs Access to find the record in the *RecordsetClone* that has a value in its Primary Key field that matches the value of the hidden Primary Key column that was chosen by the user in the combo box. This is done using the *FindFirst* method and a SQL expression that supplies the appropriate field name and value. Enter the following code statement:

rst.FindFirst "EmployeeID = " & Me.cboGoToRecord.Value

Note: I have used *EmployeeID* which is the name of the Primary Key field in this example. Make sure you enter the correct field name if yours is different.

When the above statement is executed the *RecordsetClone* moves to the specified record. Now the code needs to synchronise the form's recordset with the *RecordsetClone*. This is done with using the *Bookmark* property of each recordset. Enter the following:

Me.Bookmark = rst.Bookmark

Here is the completed procedure (*Listing 2*).

Listing 2:

```
Private Sub cboGoToRecord_AfterUpdate()
    On Error Resume Next
    Dim rst As Object
    Set rst = Me.RecordsetClone
    rst.FindFirst "EmployeeID = " & Me.cboGoToRecord.Value
    Me.Bookmark = rst.Bookmark
End Sub
```

In the Visual Basic Editor open the **Debug** menu and choose **Compile...** to check the code. If any errors are shown, correct them and compile again, then click the **Save** button. It is good practice to compile and save your new code in this way before testing it in case an unforeseen error should cause Access to crash.

Move back to Access, switch the form into Form View, and test the *Go To Record* tool. Each time you select an item from the combo box the form should jump to that record.

There is one final refinement to add. Notice that if you navigate to a record by another method, such as with the Navigation Buttons or by using the **[Page Up]** or **[Page Down]** keys, The combo

box still displays the last item you chose. This can be fixed by synchronising the combo box with the form.

To do this a line of code added to the form's *Current* event procedure can be used to set the value of the combo box to the value of the Primary Key of the record being displayed. Since the form's *Current* event fires each time a record is displayed this will make sure that the item shown in the combo box will always be correct for the record being displayed.

In **Design View** go to the **Property Sheet** and choose **Form** from the dropdown list at the top. On the **Event** tab click on **On Current** and go to the **Code Builder** as before. If you have followed earlier Masterclasses in this series your form will already have code in its *On Current* event procedure, in which case simply add the following code statement to it:

```
Me.cboGoToRecord.Value = Me.EmployeeID.Value
```

I have used the name of the Primary Key field employed in this example. Remember to use the name of the Primary Key field in your form's recordset if it is different.

After a final **Debug**, **Compile** and **Save**, test your finished *GoToRecord* tool. Job done!

Summary

In this Masterclass you created a simple *GoToRecord* tool that helps the user find and jump quickly to any record in the forms recordset. You added and formatted a combo box control and used the Query Builder to create an SQL statement that defined its list. You then created a VBA event procedure for the *AfterUpdate* event of the combo box that made use of the form's *RecordsetClone* and *Bookmarks* to find and a specific record and instruct the form to display it. Finally you added an instruction to the form's *Current* event procedure to keep the combo box synchronised with the form itself.