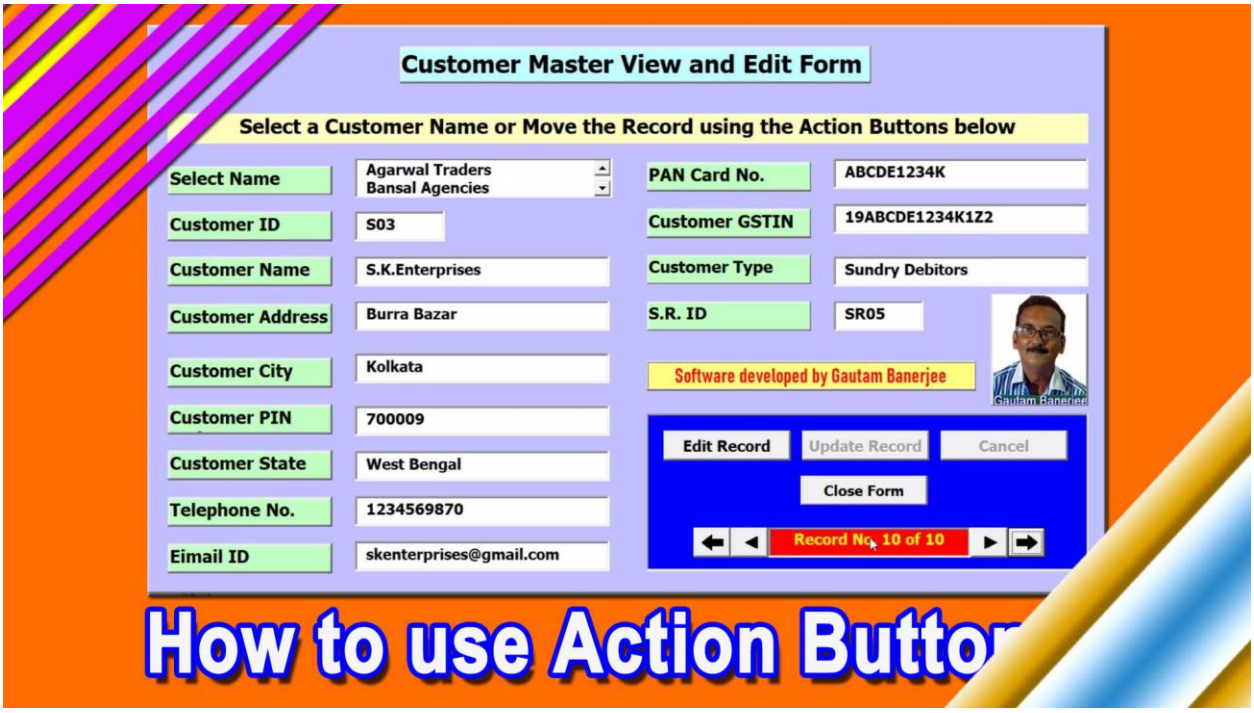

Excel VBA UserForm Tutorial: Create Action Buttons & Clickable List for Customer Data

```
Private Sub PopulateListBox()

    ' Populate the list box with customer names
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    Dim rng As Range
```

Set rng = ws.Range("B2:B" & lastRow)

TxtNameViewList.Clear

TxtNameViewList.List = rng.Value

End Sub

The PopulateListBox subroutine is designed to populate a list box named TxtNameViewList with customer names from a specified range on the "Customer_Master" worksheet. Here's a breakdown of the code:

Worksheet Reference Setup: The code sets up a reference to the "Customer_Master" worksheet using the variable ws.

Determine Last Row: It calculates the last row of data in column A of the worksheet using the lastRow variable. This is achieved by finding the last non-empty cell in column A using the End(xlUp) method.

Range Definition: A range named rng is defined to cover the range of customer names in column B, starting from row 2 to the last row.

Clear List Box: The Clear method is used to remove any existing items in the TxtNameViewList list box, ensuring a clean slate for new data.

Populate List Box: The List property of the TxtNameViewList list box is set to the values from the rng range. This effectively fills the list box with the customer names from the specified range.

In summary, the PopulateListBox subroutine prepares the TxtNameViewList list box by clearing it and then populating it with customer names fetched from the "Customer_Master" worksheet.

Private Sub ShowRecord(recordIndex As Long)

Dim ws As Worksheet

```
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    If recordIndex < 1 Then recordIndex = 1
    If recordIndex > lastRow - 1 Then recordIndex = lastRow - 1

    currentRow = recordIndex
    LblRecordNo.Caption = "Record No. " & currentRow & " of " & lastRow -
1

    TxtViewID.Value = ws.Cells(recordIndex + 1, 1).Value
    TxtViewName.Value = ws.Cells(recordIndex + 1, 2).Value
    TxtViewAdd.Value = ws.Cells(recordIndex + 1, 3).Value
    TxtViewCity.Value = ws.Cells(recordIndex + 1, 4).Value
    TxtViewPIN.Value = ws.Cells(recordIndex + 1, 5).Value
    TxtViewState.Value = ws.Cells(recordIndex + 1, 6).Value
    TxtViewPhone.Value = ws.Cells(recordIndex + 1, 7).Value
    TxtViewEmail.Value = ws.Cells(recordIndex + 1, 8).Value
    TxtViewPAN.Value = ws.Cells(recordIndex + 1, 9).Value

    If ws.Cells(recordIndex, 12).Value = "SD" Then
        TxtViewType.Value = "Sundry Debitors"
    Else
        TxtViewType.Value = "Sundry Creditors"
```

```
    End If

    TXtViewGSTIN.Value = ws.Cells(recordIndex + 1, 11).Value
    TxtViewSRID.Value = ws.Cells(recordIndex + 1, 13).Value

    CmdUpdate.Enabled = False
    CmdViewCancel.Enabled = False

End Sub
```

This code is part of a VBA user form that is used to display customer records. The purpose of the ShowRecord subroutine is to show the details of a specific customer record based on the recordIndex parameter.

Here's how the code works step by step:

It sets up a reference to the "Customer_Master" worksheet in the workbook.

It determines the last row with data in column "A" of the worksheet. This helps in finding out the total number of records in the dataset.

It performs bounds checking on the recordIndex parameter to ensure that it falls within a valid range. If recordIndex is less than 1, it's set to 1 (to ensure it doesn't go below the first record). If recordIndex is greater than the last valid record index, it's set to the last valid index (to prevent going beyond the last record).

The currentRow variable is updated with the corrected recordIndex. This variable might be used elsewhere in the code to keep track of the current record being displayed.

The label LblRecordNo is updated with a caption indicating the current record number and the total number of records, e.g., "Record No. 2 of 10". This provides the user with context about the displayed record within the dataset.

The rest of the code in this snippet likely involves populating various text boxes and controls on the user form with the details of the customer record specified by the recordIndex. This way, the user can navigate through the records using buttons or other controls and see the corresponding record details.

These two lines of code are disabling certain command buttons (CmdUpdate and CmdViewCancel) on the user form. Disabling a button means that it becomes unclickable and visually appears in a disabled state, indicating that the associated action is not currently available.

Here's what each line does:

CmdUpdate.Enabled = False: This line sets the Enabled property of the CmdUpdate command button to False. This means that the "Update" button (CmdUpdate) is being disabled, and users won't be able to click on it to perform an update action.

CmdViewCancel.Enabled = False: Similarly, this line sets the Enabled property of the CmdViewCancel command button to False. It disables the "Cancel" button (CmdViewCancel), preventing users from clicking on it to cancel a view or operation.

Disabling buttons can be useful in scenarios where certain actions are not applicable or allowed under certain conditions. In this case, it seems that these buttons are being disabled, likely in response to some logic in your user form that determines when these actions should be available and when they should be disabled.

```
Private Sub TxtNameViewList_Click()

    Dim selectedName As String
    selectedName = TxtNameViewList.Value

    Dim ws As Worksheet
```

```
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim rng As Range
    Set rng = ws.Columns("B:B").Find(What:=selectedName, LookIn:=xlValues, LookAt:=xlWhole)

    If Not rng Is Nothing Then
        Dim recordIndex As Long
        recordIndex = rng.Row - 1 ' Adjust for header row

        ShowRecord recordIndex
    End If

End Sub
```

This code is associated with the Click event of a control named TxtNameViewList, which seems to be a text box control on your user form. This code is executed when the user clicks on the text box, presumably to select a customer name from the list.

Here's what this code does:

Dim selectedName As String: This line declares a variable selectedName to store the name selected by the user from the text box.

selectedName = TxtNameViewList.Value: This line assigns the value of the selected text from the text box TxtNameViewList to the selectedName variable.

Dim ws As Worksheet...: This section of code sets up a reference to the "Customer_Master" worksheet in the workbook. It's used to look up the selected customer name in the list.

Dim rng As Range...: This line sets up a Range variable rng that will be used to search for the selected customer name within the column of customer names in the worksheet.

Set rng = ws.Columns("B:B").Find...: This line searches the entire column "B" (the second column) for the value of selectedName. It uses the Find method to locate the cell containing the selected name.

If Not rng Is Nothing Then: This line checks if a matching cell was found using the Find method. If a match is found, the code inside the If block will be executed.

Dim recordIndex As Long...: This line declares a variable recordIndex to store the row index of the found cell. It adjusts the index by subtracting 1 to account for the header row.

ShowRecord recordIndex: This line calls the ShowRecord subroutine and passes the recordIndex as an argument. This will display the details of the selected customer record on the user form.

In summary, this code handles the event when a customer name is selected from the text box. It searches for the selected name in the worksheet, and if found, it displays the corresponding customer record using the ShowRecord subroutine.

```
Private Sub UserForm_Initialize()

    ' Initialize the user form
    PopulateListBox
    ShowRecord 1 ' Display the first record initially

End Sub
```

This code is part of the initialization process of a user form in Excel VBA. It's executed automatically when the user form is loaded. Here's what this code does:

PopulateListBox: This line calls the PopulateListBox subroutine. This subroutine is responsible for populating a list box (presumably named TxtNameViewList) with customer names.

ShowRecord 1: This line calls the ShowRecord subroutine and passes the value 1 as an argument. This will display the details of the first customer record in the form.

In summary, when the user form is initialized, the PopulateListBox subroutine is called to populate the list box with customer names, and the ShowRecord subroutine is called to display the details of the first customer record in the form. This provides an initial view of the user form when it is opened.

```
Private Sub BtnNext_Click()
    If currentRow > lastRow - 1 Then
      currentRow = currentRow + 1
      ShowRecord (currentRow)
    End If
End Sub

Private Sub BtnPrev_Click()
    If currentRow > 0 Then
      currentRow = currentRow - 1
      ShowRecord (currentRow)
   End If
End Sub
```

These event handler procedures are associated with the "Next" and "Previous" buttons in your user form. They allow the user to navigate through the records displayed in the form. Here's what each of these procedures does:

BtnNext_Click:

Checks if the current row index (currentRow) is less than the last row index (lastRow - 1). This check ensures that you don't move beyond the last record.

If the check is true, it increments the currentRow index by 1 to move to the next record.

Calls the ShowRecord subroutine and passes the updated currentRow index as an argument. This will update the form to display the details of the next record.

BtnPrev_Click:

Checks if the current row index (currentRow) is greater than 0. This check ensures that you don't move before the first record.

If the check is true, it decrements the currentRow index by 1 to move to the previous record.

Calls the ShowRecord subroutine and passes the updated currentRow index as an argument. This will update the form to display the details of the previous record.

In summary, these procedures allow the user to navigate through the records in the user form by clicking the "Next" and "Previous" buttons. The ShowRecord subroutine is called to update the displayed record based on the new currentRow index.

```
Private Sub BtnFirst_Click()
    ShowRecord 1
End Sub

Private Sub BtnLast_Click()
    Dim ws As Worksheet
```

```
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    ShowRecord lastRow - 1

    LblRecordNo.Caption = "Record No. " & lastRow - 1 & " of " & lastRow - 1
End Sub
```

These event handler procedures are associated with the "First" and "Last" buttons in your user form. They allow the user to quickly navigate to the first and last records in the displayed records. Here's what each of these procedures does:

BtnFirst_Click:

Calls the ShowRecord subroutine and passes 1 as an argument. This will update the form to display the details of the first record.

It also updates the label LblRecordNo.Caption to display "Record No. 1 of [Total Records - 1]". This informs the user that they are viewing the first record.

BtnLast_Click:

Retrieves the total number of rows (records) in the worksheet using the lastRow variable.

Calls the ShowRecord subroutine and passes lastRow - 1 as an argument. This will update the form to display the details of the last record.

Updates the label LblRecordNo.Caption to display "Record No. [Total Records - 1] of [Total Records - 1]". This informs the user that they are viewing the last record.

In summary, these procedures provide the user with the ability to jump to the first and last records in the user form. The ShowRecord subroutine is called to update

the displayed record details accordingly, and the label is updated to provide context to the user about the record they are viewing.

Gautam Banerjee

“Helping beginners learn something new is a great way to share your knowledge and make a positive impact”.

Email: gincom1@yahoo.com

If you have any queries, please visit our "Contact Us" page.

Thank You. See you again!

PLEASE DONATE

Gautam Banerjee

Age: 63

Pay by UPI

9748327614