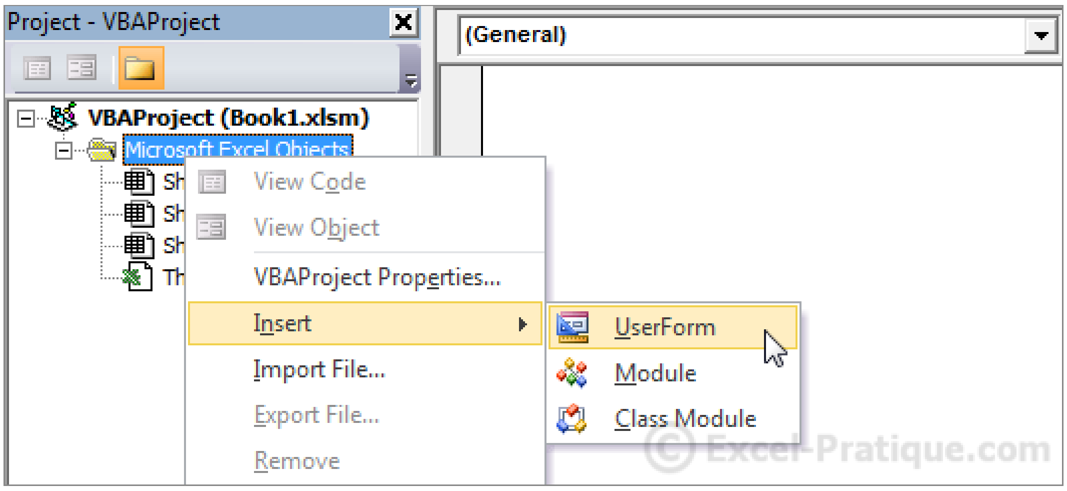
Lesson 9

Sébastien Mathier

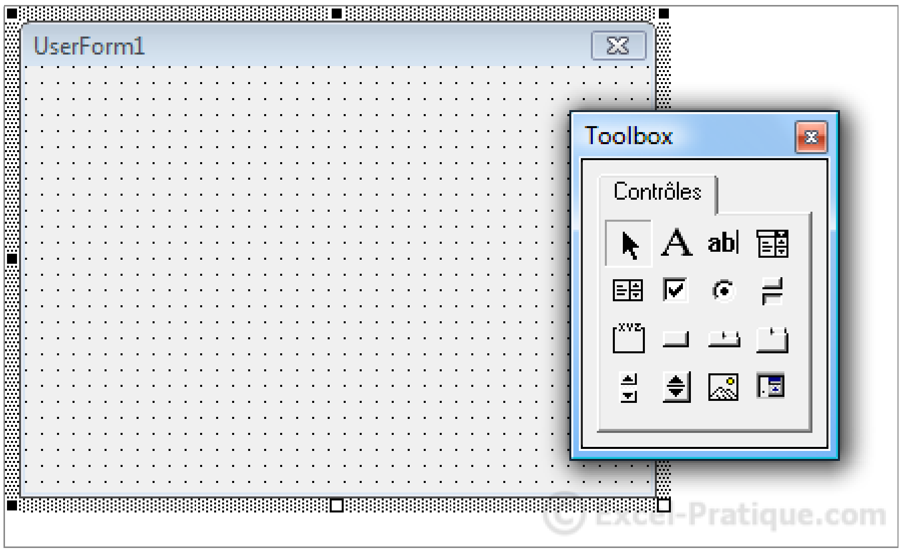
www.excel-pratique.com/en

UserForm :

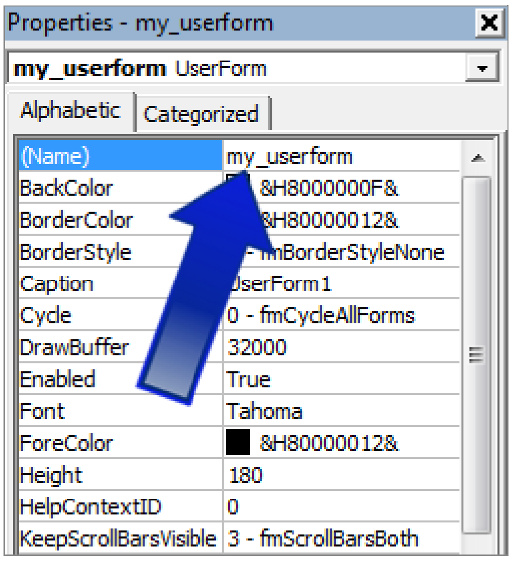
To add a UserForm, do exactly as you would if you were adding a new module :



The UserForm window and "Toolbox" will appear :



If you don't see the Properties window, make sure that it is shown and then start by editing the name of the UserForm (so that you can easily find it later on) :



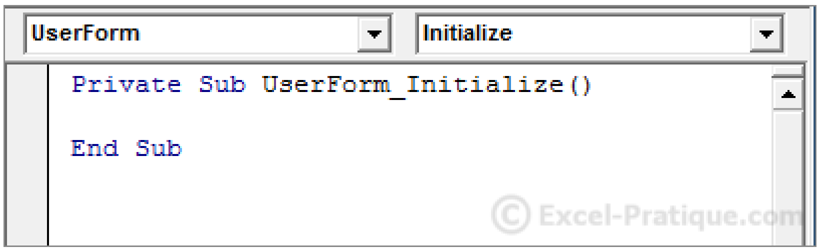
Properties - my_userform

my_userform UserForm

Alphabetic | Categorized

(Name)	my_userform
BackColor	&H8000000F&
BorderColor	&H80000012&
BorderStyle	- mBorderStyleNone
Caption	serForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	180
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth

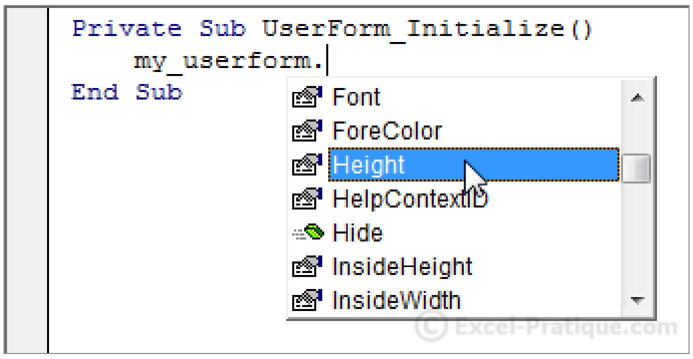
A UserForm has its own events, just like a workbook or a worksheet. To add events, double click on the UserForm window :



```
Private Sub UserForm_Initialize()

End Sub
```

Now let's create two events as an example of how they work. The first event will define the initial dimensions of the UserForm, and the second will increase each of its dimensions by 50 pixels when the user clicks.

The event **UserForm_Initialize** will fire when the UserForm is launched :



```
Private Sub UserForm_Initialize()
    my_userform.
End Sub
```

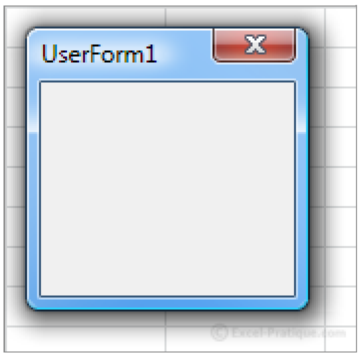
```
Private Sub UserForm_Initialize()
    my_userform.Height = 100
    my_userform.Width = 100
End Sub
```

To simplify the code, we can use **Me** instead of the name of the UserForm (since this code is within the UserForm that we're working with) :

```
Private Sub UserForm_Initialize()
    Me.Height = 100
    Me.Width = 100
End Sub
```

The second event will fire when the user clicks on the UserForm :

```
Private Sub UserForm_Initialize()
    Me.Height = 100
    Me.Width = 100
End Sub

Private Sub UserForm_Click()
    Me.Height = Me.Height + 50
    Me.Width = Me.Width + 50
End Sub
```



Launch a UserForm :

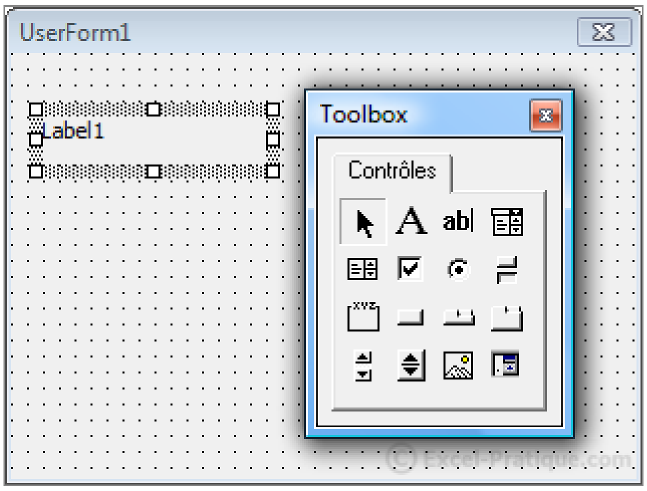
To launch a UserForm in a procedure, use **Show** :

```
Sub show_userform()
    my_userform.Show
End Sub
```

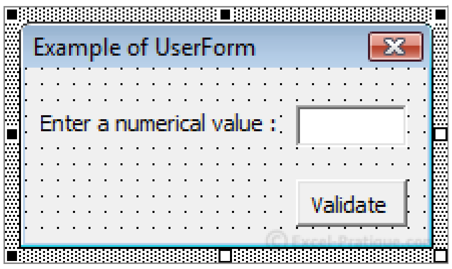
Controls :

Controls have all sorts of properties, and the events associated with them vary, but for now we will only look at a few of the many possible uses of controls in VBA coding.

Let's start by adding the following 3 controls : a **Label**, a **TextBox** and a **CommandButton** :



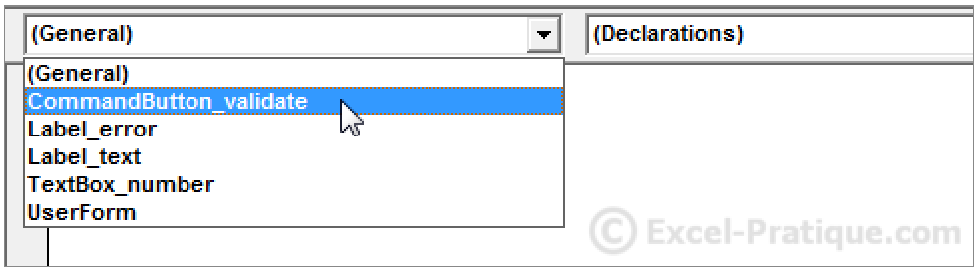
Now let's edit the names and properties of the controls (using the **Caption** property, which contains the text). We want the following result :



For now, when we enter a number and press OK, nothing happens.

To make something happen, we'll start by adding an event that puts the value of the text box into cell A1 and closes the UserForm.

You can access the options that you see immediately below this text by double clicking on a control :



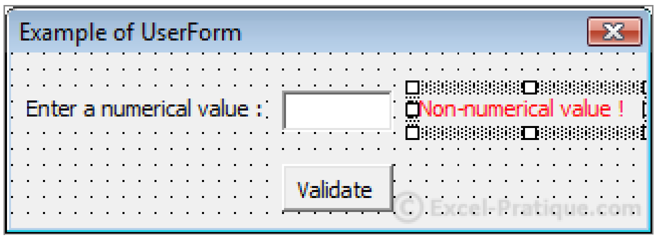
The drop-down list contains different controls and the UserForm.

Select a button and an event **Click** :

```
Private Sub CommandButton_validate_Click()

    Range("A1") = Textbox_number.Value
    'Textbox_number is the name of the text box
    'Value is the property that contains the value of the text box

    Unload Me
    'Unload closes the UserForm
    'We are using Me in place of the name of the UserForm (because this code is within the
UserForm that we want to close)
End Sub
```

The value will now be saved in cell A1 before the closing of the UserForm.

Add a second Label and edit the following properties : **Caption**, **Forecolor** (color : red) and **Visible** (False, to hide the control by default) :



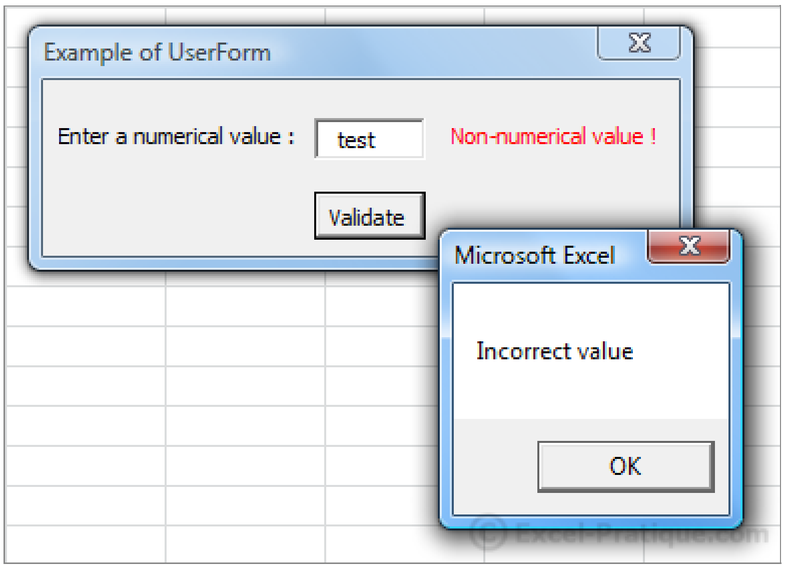
Now let's add an event that will be fired when the value of the text box is changed, which will display an error message if the value is not numerical.

```
Private Sub Textbox_number_Change()
    If IsNumeric(Textbox_number.Value) Then 'IF numerical value ...
        Label_error.Visible = False 'Label hidden
    Else 'OTHERWISE ...
        Label_error.Visible = True 'Label shown
    End If
End Sub
```

The value will be tested each time a character is entered ...

We still need to prevent the validation of the form if the value is not numerical :

```
Private Sub CommandButton_validate_Click()
    If IsNumeric(Textbox_number.Value) Then 'IF numerical value ...
        Range("A1") = Textbox_number.Value 'Copy to A1
        Unload Me 'Closing
    Else 'OTHERWISE ...
        MsgBox "Incorrect value"
    End If
End Sub
```

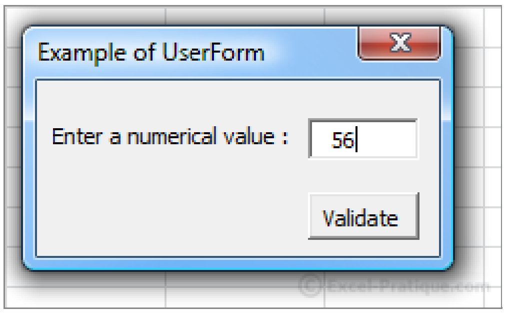


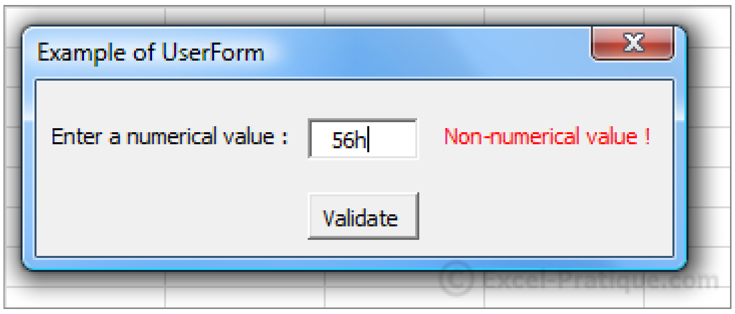
So as not to leave the right-hand side of the UserForm blank when there isn't any error, we can reduce its size by modifying the UserForm's **Width** property :

```
Private Sub Textbox_number_Change()
    If IsNumeric(Textbox_number.Value) Then 'IF numerical value ...
        Label_error.Visible = False 'Label hidden
        Me.Width = 156 'UserForm Width
    Else 'OTHERWISE ...
        Label_error.Visible = True 'Label shown
        Me.Width = 244 'UserForm Width
    End If
End Sub
```

Source file : **userform1.xls**

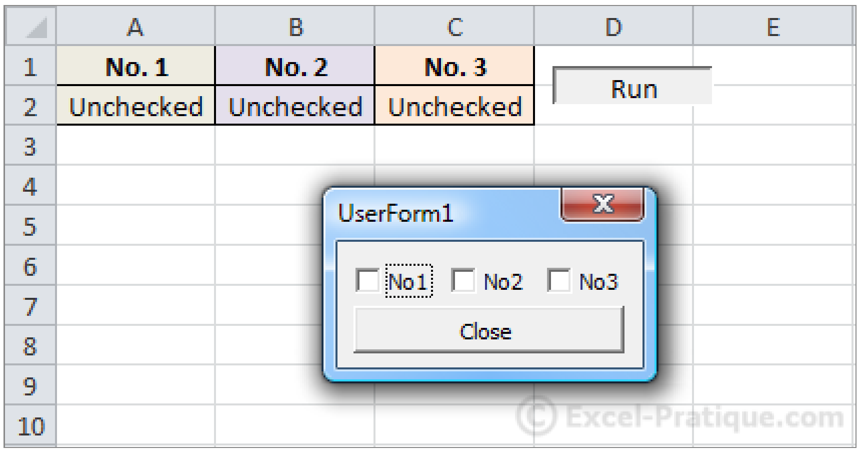
See result below :





Checkboxes :

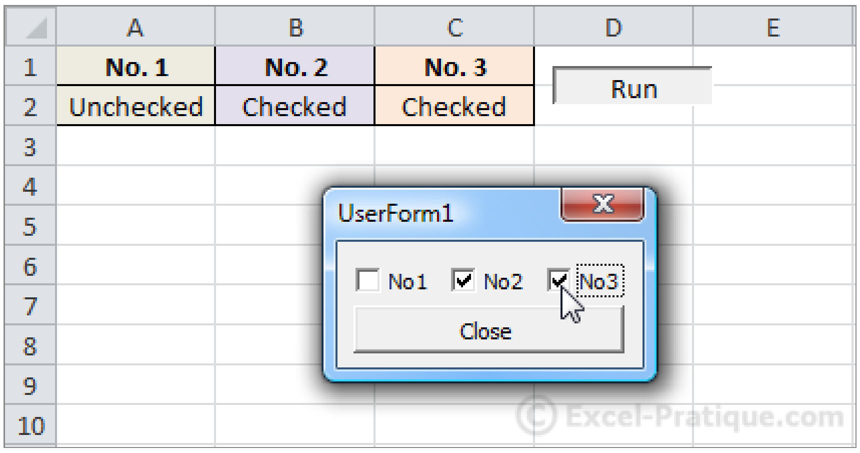
Here is an example of how to use the **CheckBox** :



	A	B	C	D	E
1	No. 1	No. 2	No. 3		
2	Unchecked	Unchecked	Unchecked		

When a checkbox is checked/unchecked, the value of the corresponding cell can be modified by using the **Click** event :

```
Private Sub CheckBox1_Click() 'Number 1
    If CheckBox1.Value = True Then 'If checked ...
       Range("A2") = "Checked"
    Else 'If not checked ...
       Range("A2") = "Unchecked"
    End If
End Sub

Private Sub CheckBox2_Click() 'Number 2
    If CheckBox2.Value = True Then 'If checked ...
       Range("B2") = "Checked"
    Else 'If not checked ...
       Range("B2") = "Unchecked"
    End If
End Sub

Private Sub CheckBox3_Click() 'Number 3
    If CheckBox3.Value = True Then 'If checked ...
       Range("C2") = "Checked"
    Else 'If not checked ...
       Range("C2") = "Unchecked"
    End If
End Sub
```



	A	B	C	D	E
1	No. 1	No. 2	No. 3		
2	Unchecked	Checked	Checked		

In this example, the checkboxes start out unchecked when the UserForm is first opened.
To check the boxes when the value of the corresponding cell is "Checked", we'll run a test when the UserForm is activated, using **UserForm_Initialize** :

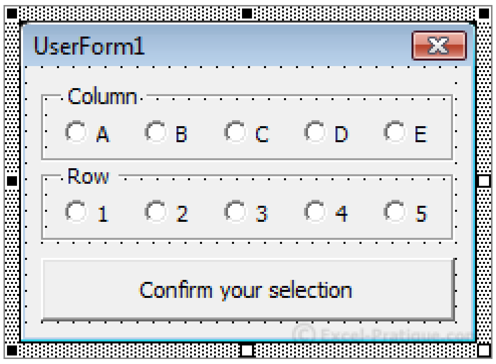
```
Private Sub UserForm_Initialize() 'Check box if "Checked"
    If Range("A2") = "Checked" Then
        CheckBox1.Value = True
    End If

    If Range("B2") = "Checked" Then
        CheckBox2.Value = True
    End If

    If Range("C2") = "Checked" Then
        CheckBox3.Value = True
    End If
End Sub
```

Source file : **userform2.xls**

The Option Buttons :

The user may only select one Option Button per "group", which is not the case with checkboxes.

To create a group, first insert a **Frame** and then **OptionButton** :



Source file : **userform3.xls**

Once the form has been submitted, we will enter data into the cell that corresponds to the column numbers and row_value chosen.

In order to know which option button was chosen, we could do the same as in the previous example (with the checkboxes) but we will do it with a loop to reduce the length of the code.

We're going to use a **For Each** loop this time, a kind of loop that we haven't yet introduced. This kind of loop makes it possible to execute instructions for each object in an "object group" :

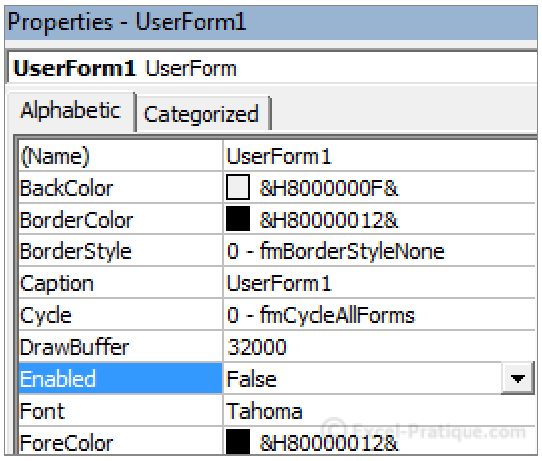
```
Private Sub CommandButton1_Click()
    Dim column_value As String, row_value As String

    'Loop for each Frame_column control
    For Each column_button In Frame_column.Controls
        'If the value of the control  = True (then, if checked) ...
        If column_button.Value Then
           'The variable "column_value" takes the value of the button text
           column_value = column_button.Caption
        End If
    Next

    'Loop for the other frame
    For Each row_button In Frame_row.Controls
        If row_button.Value Then
            row_value = row_button.Caption
        End If
    Next

    Range(column_value & row_value) = "Cell chosen !"
    Unload Me
End Sub
```

Now this form enters the value "Cell chosen !" into the cell that has been chosen (provided that the form is complete).

To avoid a bug, we need to check to make sure that the user has chosen correctly from the two sets of option buttons.

In this example, when the form is incomplete, the "Confirm" button will appear in gray (deactived). This is not the simplest solution, but it's a good example of why functions/procedures are useful within a UserForm.

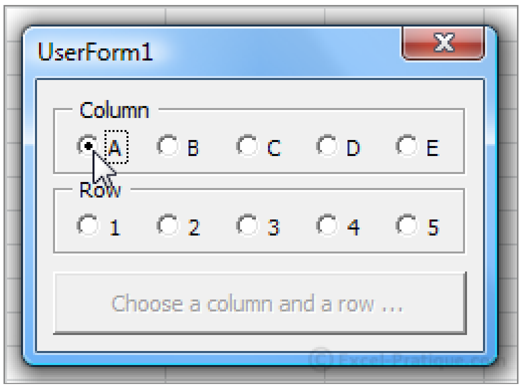
Edit the text as well as the **Enabled** property to deactivate the button :


Properties - UserForm1

UserForm1 UserForm

(Name)	UserForm1
BackColor	⬜ &H8000000F&
BorderColor	⬛ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	UserForm1
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	False
Font	Tahoma
ForeColor	⬛ &H80000012&

The result will be :



In the preceding code, we used 2 **For Each** loops to retrieve the values of the option buttons. We will now need to use these same values for the "Confirm" button and the **Click** events for the ten option buttons.

So that we don't have to copy the loops for each event, we'll call them using a function.

Starting from the preceding code and modifying it, we will achieve this result :

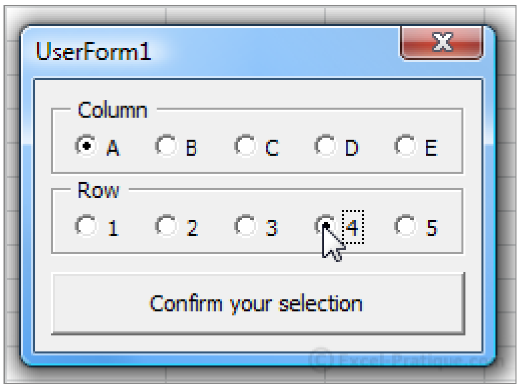
```
Private Function column_value()
'The function returns the value of the text for the button chosen (column_value)
    For Each column_button In Frame_column.Controls
        If column_button.Value Then
            column_value = column_button.Caption
        End If
    Next
End Function

Private Function row_value()
'The function returns the value of the text for the button chosen (row_value)
    For Each row_button In Frame_row.Controls
        If row_button.Value Then
            row_value = row_button.Caption
        End If
    Next
End Function
```

```
Private Sub CommandButton1_Click() 'Action that is taken when you click "Confirm your
selection"
    Range(column_value & row_value) = "Cell chosen !"
    'column_value and row_value are the values returned by the functions
    Unload Me
End Sub
```

All we have left to do is to create a procedure that verifies that the buttons have been checked correctly (by calling the two functions), and which will activate the button if necessary.

Here again, the test is performed in a separate procedure to avoid copying the code 10x for each of the option button events :

```
Private Sub activate_button()
'Activating the button if the condition is verified
    If column_value <> "" And row_value <> "" Then
    'column_value and row_value are the values returned by the functions
        CommandButton1.Enabled = True
        CommandButton1.Caption = "Confirm your selection"
    End If
End Sub

Private Sub OptionButton11_Click()
    activate_button 'Run the "activate_button" procedure
End Sub
Private Sub OptionButton12_Click()
    activate_button
End Sub
Private Sub OptionButton13_Click()
    activate_button
End Sub
Private Sub OptionButton14_Click()
    activate_button
End Sub
Private Sub OptionButton15_Click()
    activate_button
End Sub
Private Sub OptionButton16_Click()
    activate_button
End Sub
Private Sub OptionButton17_Click()
    activate_button
End Sub
Private Sub OptionButton18_Click()
    activate_button
End Sub
Private Sub OptionButton19_Click()
    activate_button
End Sub
Private Sub OptionButton20_Click()
    activate_button
End Sub
```

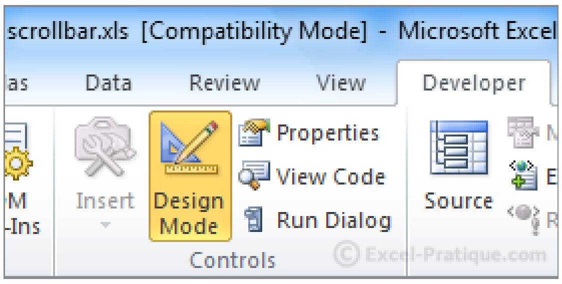


Source file : **userform3b.xls**

ScrollBar :

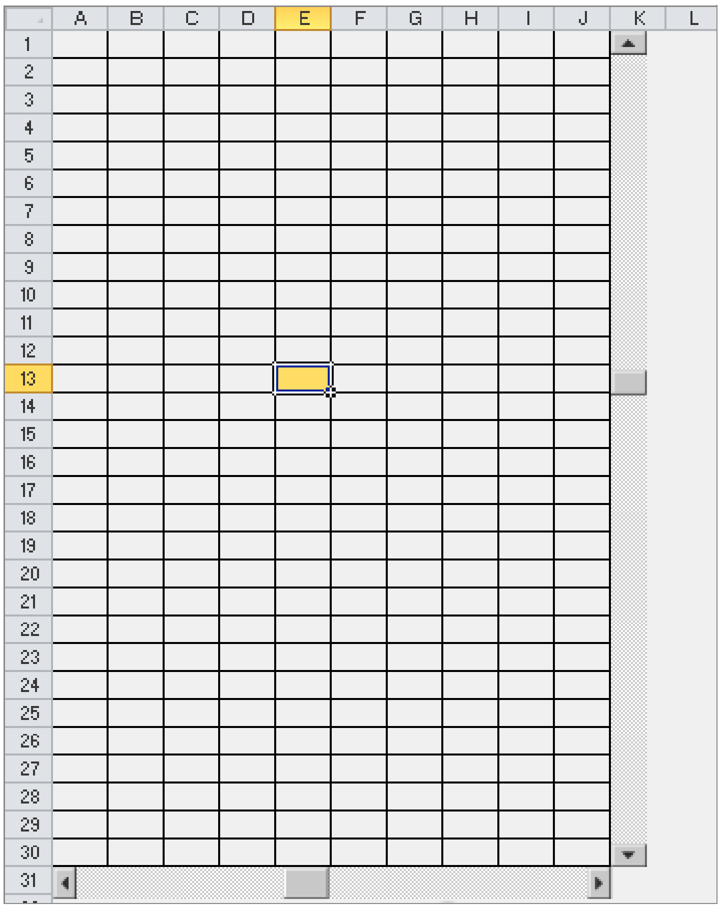
Controls can be used outside of UserForms. In this example, we'll use controls right on a worksheet.

Please note that "Creation Mode" must be activated in order to modify a control that is placed on a worksheet (and likewise deactivated in order to use the control).



In versions d'Excel lower than 2007 : the button is on the "Controls Toolbox" toolbar.

Before we go into detail with this example, please have a look at this :



Our goal is to add a background color to a cell and select it based on the position of the scroll bars in the defined 30 row x 10 column area.

The properties of the vertical scroll bar :

- **Min** : 1
- **Max** : 30 (because there are 30 rows)
- **Value** : the position of the bar (in this case, between 1 and 30)

The horizontal bar is exactly the same except that it has a **Max** of 10 ...

Here is the code that will run each time there is a change to the **Value** of the vertical scrollbar :

```
'Gray background color in the cells
Cells.Interior.Color = RGB(240, 240, 240)

'Applying color and selecting the cell
With Cells(ScrollBar_vertical.Value, ActiveCell.Column) 'Identifying the cell using Value
    .Interior.Color = RGB(255, 220, 100) 'Applying Orange Color
    .Select 'Selecting the cell
End With
```

This code will run when the **Change** and **Scroll** events are fired and will execute the instructions no matter which part of the scrollbar is clicked.

Here is the code for the vertical scrollbar :

```
Private Sub vertical_bar()
    'Applying gray background color to the cells
    Cells.Interior.Color = RGB(240, 240, 240)

    'Applying background color and selecting the cell
    With Cells(ScrollBar_vertical.Value, ActiveCell.Column)
        .Interior.Color = RGB(255, 220, 100) 'Orange
        .Select 'Selecting the cell
    End With
End Sub

Private Sub ScrollBar_vertical_Change()
    vertical_bar
End Sub

Private Sub ScrollBar_vertical_Scroll()
    vertical_bar
End Sub
```

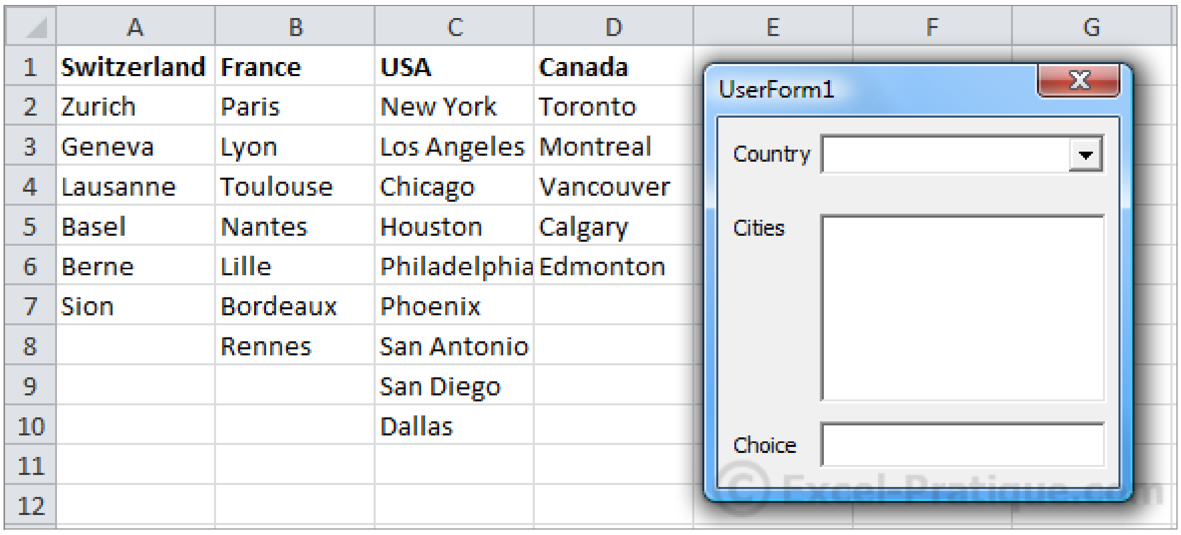
And here is the code for the horizontal scrollbar :

```
Private Sub horizontal_bar()
    'Applying gray background color to the cells
    Cells.Interior.Color = RGB(240, 240, 240)

    'Applying background color and selecting cell
    With Cells(ActiveCell.Row, ScrollBar_horizontal.Value)
        .Interior.Color = RGB(255, 220, 100) 'Orange
        .Select 'Selecting the cell
    End With
End Sub

Private Sub ScrollBar_horizontal_Change()
    horizontal_bar
End Sub

Private Sub ScrollBar_horizontal_Scroll()
    horizontal_bar
End Sub
```

Source file : **scrollbar.xls**

ComboBox and **ListBox** :

This is the starting point for our next example :



	A	B	C	D
1	**Switzerland**	**France**	**USA**	**Canada**
2	Zurich	Paris	New York	Toronto
3	Geneva	Lyon	Los Angeles	Montreal
4	Lausanne	Toulouse	Chicago	Vancouver
5	Basel	Nantes	Houston	Calgary
6	Berne	Lille	Philadelphia	Edmonton
7	Sion	Bordeaux	Phoenix	
8		Rennes	San Antonio	
9			San Diego	
10			Dallas	
11				
12				

Source file : **userform4.xls**

When the UserForm is launched, we want the 4 countries to be loaded into the drop-down list (using the **AddItem** method) :

```
Private Sub UserForm_Initialize()
    For i = 1 To 4 ' => to list the 4 countries
       ComboBox_Country.AddItem Cells(1, i) 'Add the values of cells A1 through A4 using the
loop
   Next
End Sub
```

When the value of the drop-down list changes, we want to add the cities in the chosen country using a loop similar to the previous one.

In order to do this, we need the column number and the row number.

The property **ListIndex** contains the number of the selection in the drop-down list (unlike **Value**, which contains the value of the list item). Please note that **ListIndex** begins with the number 0.

So the column number is given by :

```
column_number = ComboBox_Country.ListIndex + 1
```

To obtain the number of rows in the chosen country's column, we can search for the row number of the last in a sequence of non-empty cells :

```
rows_number = Cells(1, column_number).End(xlDown).Row
```

Using this information, it is now possible to create a loop to add the cities to the list area :

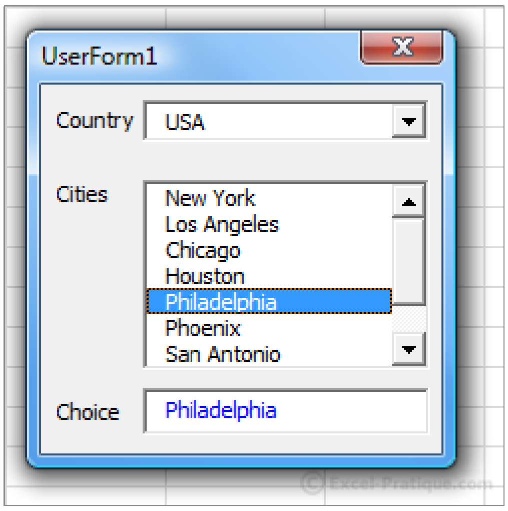
```
Private Sub ComboBox_Country_Change()
    'Emptied list area (otherwise the cities are added immediately)
    ListBox_Cities.Clear

    Dim column_number As Integer, rows_number As Integer

    'The number of the selection (ListIndex starts with 0) :
    column_number = ComboBox_Country.ListIndex + 1
    'Number of rows in the chosen country's column :
    rows_number = Cells(1, column_number).End(xlDown).Row

    For i = 2 To rows_number ' => to list cities
       ListBox_Cities.AddItem Cells(i, column_number)
    Next
End Sub
```

Note : we could have abbreviated the code above, but we haven't because that would make it much less readable :

```
Private Sub ComboBox_Country_Change()
    ListBox_Cities.Clear
    For i = 2 To Cells(1, ComboBox_Country.ListIndex + 1).End(xlDown).Row
        ListBox_Cities.AddItem Cells(i, ComboBox_Country.ListIndex + 1)
    Next
End Sub
```

The city that is chosen will then be entered into the text area :

```
Private Sub ListBox_Cities_Click()
    TextBox_Choice.Value = ListBox_Cities.Value
End Sub
```


UserForm1

Country: USA

Cities:
New York
Los Angeles
Chicago
Houston
Philadelphia
Phoenix
San Antonio

Choice: Philadelphia

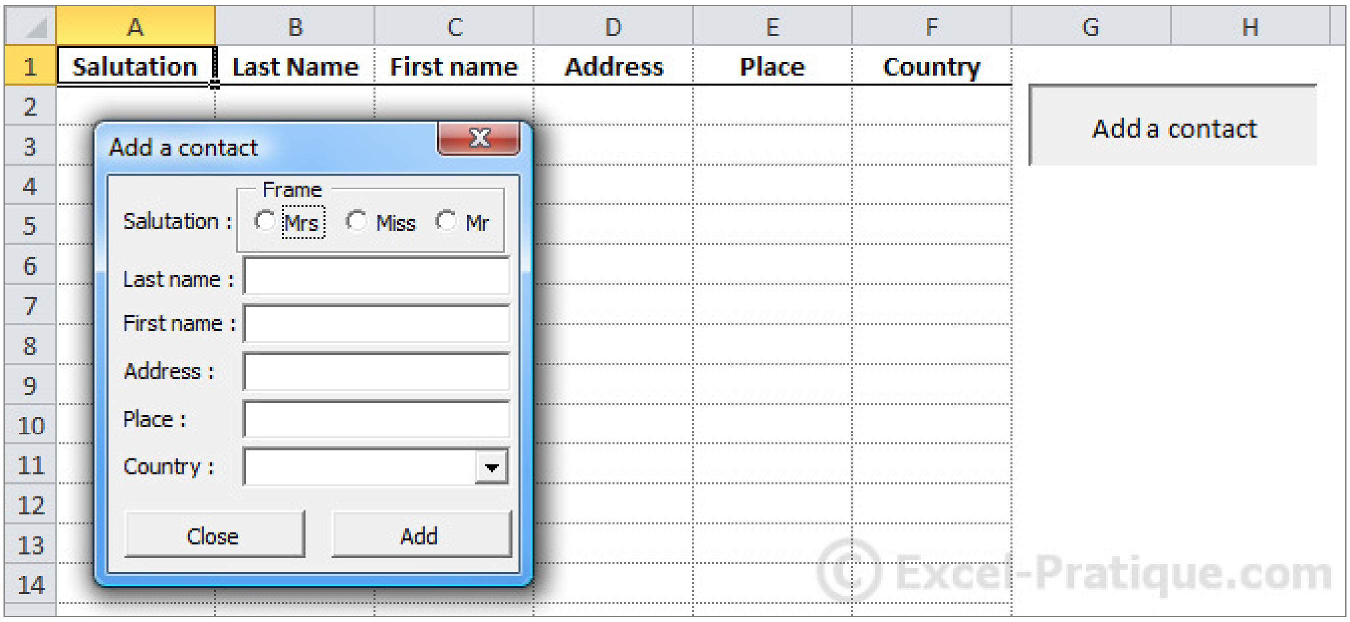
Source file : **userform4b.xls**

Exercise :

Now let's try a little exercise to practice using controls ...

This is how we start the exercise :

Source file : **controls_exercise.xls**



Salutation	Last Name	First name	Address	Place	Country

By now it should be clear that the goal here is to fill the table using the form.

A few thinks to keep in mind :

- List the countries based on the list on the second worksheet
- Verify the contents of the controls before adding a new contact
- After inserting a value, reinitialize the values of the controls without closing the form

Take a moment to work through this exercise before looking at the solution ...

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

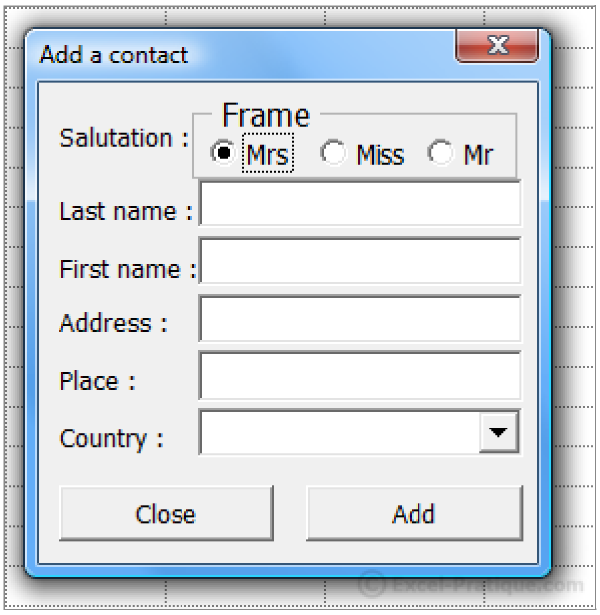
.

.

.
.
.
.
.
.
.
.
.
.
.
.

This is one way that you could solve this problem :

Our first action should be to increase the **Zoom** property of the UserForm to 120 to make it easier to use the form :



We have already covered option button tests (in the first controls page), so we are using a simpler solution here.

"Mrs" is chosen by default (**Value** property : True), which means that we don't have to verify that a salutation has been chosen.

The "Close" button :

```
Private Sub CommandButton_Close_Click()
    Unload Me
End Sub
```

The contents of the drop-down list :

```
Private Sub UserForm_Initialize() 'Loading the list when the UserForm is opened
   For i = 1 To 252 'List of 252 countries froim the "Country" worksheet
      ComboBox_Country.AddItem Sheets("Country").Cells(i, 1)
   Next
End Sub
```

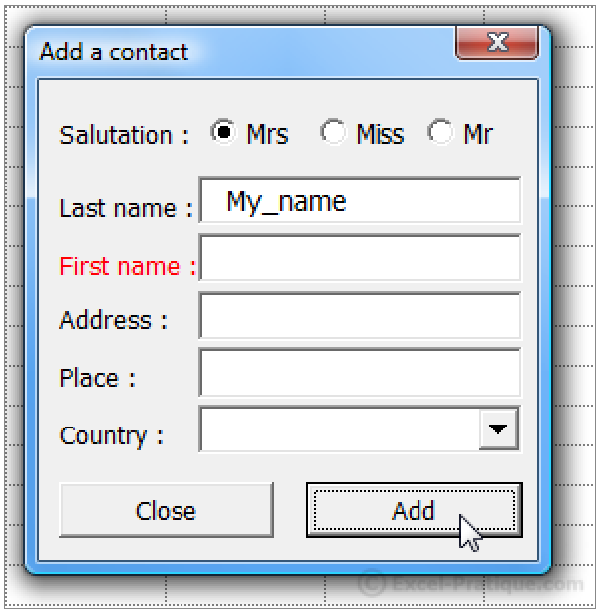
Verification of controls :

One simple solution would be to display a dialog box if any of the controls is empty :

```
Private Sub CommandButton_Add_Click()
    If TextBox_Last_Name.Value = "" Or TextBox_First_Name.Value = "" Or TextBox_Address.Value
= "" Or TextBox_Place.Value = "" Or ComboBox_Country.Value = "" Then
        MsgBox "Form incomplete"
    Else
        'Instructions for inserting a contact here ...
    End If
End Sub
```

But to make things a bit more complicated, each control should be tested individually, and if any of them is empty, its **Label** text color should be changed to red :

```
Private Sub CommandButton_Add_Click()
    'Setting Label text color to black
    Label_Last_Name.ForeColor = RGB(0, 0, 0)
    Label_First_Name.ForeColor = RGB(0, 0, 0)
    Label_Address.ForeColor = RGB(0, 0, 0)
    Label_Place.ForeColor = RGB(0, 0, 0)
    Label_Country.ForeColor = RGB(0, 0, 0)

    'Content controls
    If TextBox_Last_Name.Value = "" Then 'IF no "name" provided ...
        Label_Last_Name.ForeColor = RGB(255, 0, 0) 'Set Label "name" color to red
    ElseIf TextBox_First_Name.Value = "" Then
        Label_First_Name.ForeColor = RGB(255, 0, 0)
    ElseIf TextBox_Address.Value = "" Then
        Label_Address.ForeColor = RGB(255, 0, 0)
    ElseIf TextBox_Place.Value = "" Then
        Label_Place.ForeColor = RGB(255, 0, 0)
    ElseIf ComboBox_Country.Value = "" Then
        Label_Country.ForeColor = RGB(255, 0, 0)
    Else
        'Instructions for inserting a contact here ...
    End If
End Sub
```



Inserting data :

The following code should be inserted in the place indicated in the code above (see commentary) :

```
Dim row_number As Integer, salutation As String

'Choice of Salutation
For Each salutation_button In Frame_Salutation.Controls
    If salutation_button.Value Then
        salutation = salutation_button.Caption 'Salutation chosen
    End If
Next

'row_number = row number of the last non-empty cell in the column +1
row_number = Range("A65536").End(xlUp).Row + 1

'Inserting values into the worksheet
Cells(row_number, 1) = salutation
Cells(row_number, 2) = TextBox_Last_Name.Value
Cells(row_number, 3) = TextBox_First_Name.Value
Cells(row_number, 4) = TextBox_Address.Value
Cells(row_number, 5) = TextBox_Place.Value
Cells(row_number, 6) = ComboBox_Country.Value

'After insertion, the initial values are replaced
OptionButton1.Value = True
TextBox_Last_Name.Value = ""
TextBox_First_Name.Value = ""
TextBox_Address.Value = ""
TextBox_Place.Value = ""
ComboBox_Country.ListIndex = -1
```

Overall view :

That's all, and here you have the complete code for the exercise and the downloadable Excel file :

```vba
Private Sub CommandButton_Close_Click()
    Unload Me
End Sub

Private Sub UserForm_Initialize() 'List of 252 countries on the "Country" worksheet
   For i = 1 To 252
       ComboBox_Country.AddItem Sheets("Country").Cells(i, 1)
   Next
End Sub

Private Sub CommandButton_Add_Click()
    'Setting Label color to black
    Label_Last_Name.ForeColor = RGB(0, 0, 0)
    Label_First_Name.ForeColor = RGB(0, 0, 0)
    Label_Address.ForeColor = RGB(0, 0, 0)
    Label_Place.ForeColor = RGB(0, 0, 0)
    Label_Country.ForeColor = RGB(0, 0, 0)

    'Content controls
    If TextBox_Last_Name.Value = "" Then 'If no "name" provided ...
        Label_Last_Name.ForeColor = RGB(255, 0, 0) 'Set Label "name" color to red
    ElseIf TextBox_First_Name.Value = "" Then
        Label_First_Name.ForeColor = RGB(255, 0, 0)
    ElseIf TextBox_Address.Value = "" Then
        Label_Address.ForeColor = RGB(255, 0, 0)
    ElseIf TextBox_Place.Value = "" Then
        Label_Place.ForeColor = RGB(255, 0, 0)
    ElseIf ComboBox_Country.Value = "" Then
        Label_Country.ForeColor = RGB(255, 0, 0)
    Else
        'If the form is complete, the values will be inserted onto the worksheet
        Dim row_number As Integer, salutation As String

        'Choice of salutation
        For Each salutation_button In Frame_Salutation.Controls
            If salutation_button.Value Then
                salutation = salutation_button.Caption
            End If
        Next

        'row_number = row number of the last non-empty cell in column +1
        row_number = Range("A65536").End(xlUp).Row + 1

        'Inserting values onto the worksheet
        Cells(row_number, 1) = salutation
        Cells(row_number, 2) = TextBox_Last_Name.Value
        Cells(row_number, 3) = TextBox_First_Name.Value
        Cells(row_number, 4) = TextBox_Address.Value
        Cells(row_number, 5) = TextBox_Place.Value
        Cells(row_number, 6) = ComboBox_Country.Value

        'After insert, we replace the initial values
        OptionButton1.Value = True
        TextBox_Last_Name.Value = ""
        TextBox_First_Name.Value = ""
        TextBox_Address.Value = ""
        TextBox_Place.Value = ""
        ComboBox_Country.ListIndex = -1
    End If
End Sub
```

Source file : **controls_exercise2.xls**