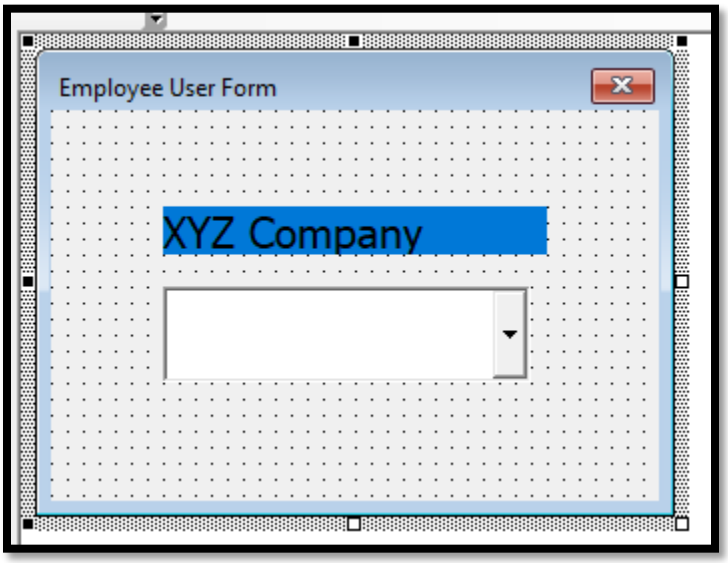
So here is the simple UserForm



So as company name Label I used XYZ Company in design time. So, all the instance will have that if I don't change that. Now Here, is the code for the Create Function

```
Public Function Create(GivenCompanyName As String) As EmployeeForm

    Dim CurrentEmployeeForm As EmployeeForm
    Set CurrentEmployeeForm = New EmployeeForm

    With CurrentEmployeeForm
        .CompanyName = GivenCompanyName
        'Using Ref
        .CompanyNameLabel.Caption = GivenCompanyName
        'Using Me
        SetEmployeeNameUsingMe
    End With
    Set Create = CurrentEmployeeForm

End Function


Private Sub SetEmployeeNameUsingMe()
    Me.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub

Private Sub SetEmployeeNameUsingRef(SetToUF As EmployeeForm)
    SetToUF.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub
```

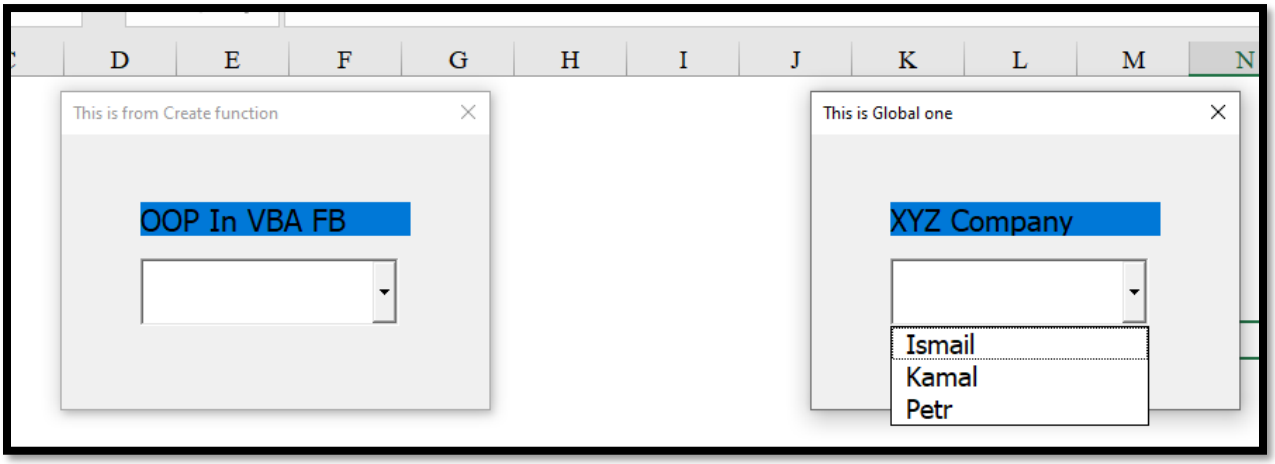
Look Carefully in the comments. So, I am changing the caption for company using Ref(CurrentEmployeeForm) but in case of Employee Name list I am using Me keyword.

So As Userform PredeclaredId is set to true that's why we have one Global object and another is that new up (CurrentEmployeeForm). So Basically, Me is referencing that Global object till the Create function is in the Call stack.

Here is the driver code:

```
Sub Test()

    Dim FormUsingFactory As EmployeeForm
    Set FormUsingFactory = EmployeeForm.Create("OOP In VBA FB Group")
    FormUsingFactory.Show vbModeless
    FormUsingFactory.Caption = "This is from Create function"
    FormUsingFactory.Left = 200

    EmployeeForm.Show vbModeless
    EmployeeForm.Caption = "This is Global one"
    EmployeeForm.Left = 600

End Sub
```

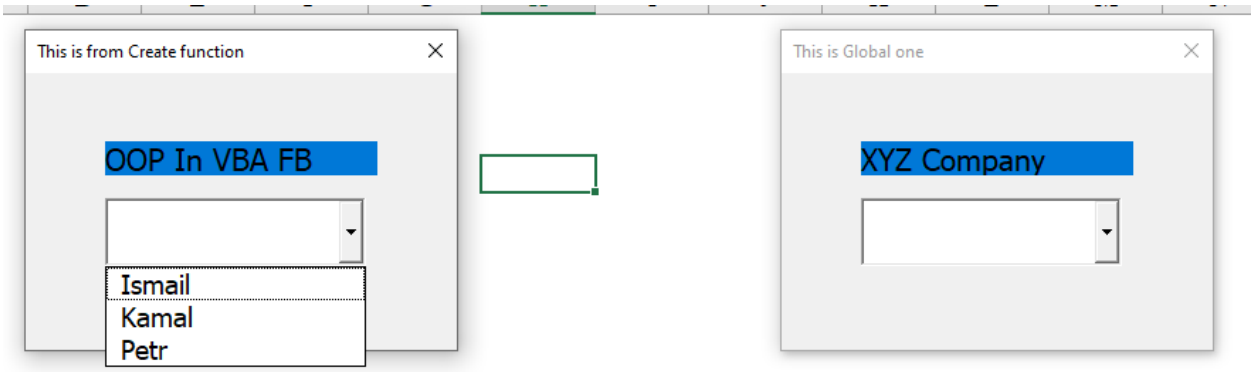
And the output:



See Caption and the Combobox data. The one which is created from the Create function has set the Label properly but it doesn't have the Employee list(Believe me or test it). But the Global one has the Employee List but it doesn't have the updated Company name. So, if you want to use Me keyword in this case then you have to be careful here. So, if you are thinking that you are setting the list for the "CurrentEmployeeForm" using Me keyword then you are doing it wrong way. To set it properly you need to do something like this:

```
Public Function Create(GivenCompanyName As String) As EmployeeForm

    Dim CurrentEmployeeForm As EmployeeForm
    Set CurrentEmployeeForm = New EmployeeForm

    With CurrentEmployeeForm
        .CompanyName = GivenCompanyName
        'Using Ref
        .CompanyNameLabel.Caption = GivenCompanyName
        'Using Ref >> Only Change in this line
        SetEmployeeNameUsingRef CurrentEmployeeForm
    End With
    Set Create = CurrentEmployeeForm

End Function


Private Sub SetEmployeeNameUsingMe()
    Me.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub

Private Sub SetEmployeeNameUsingRef(SetToUF As EmployeeForm)
    SetToUF.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub
```

And Now the UF from Create Function has that list.



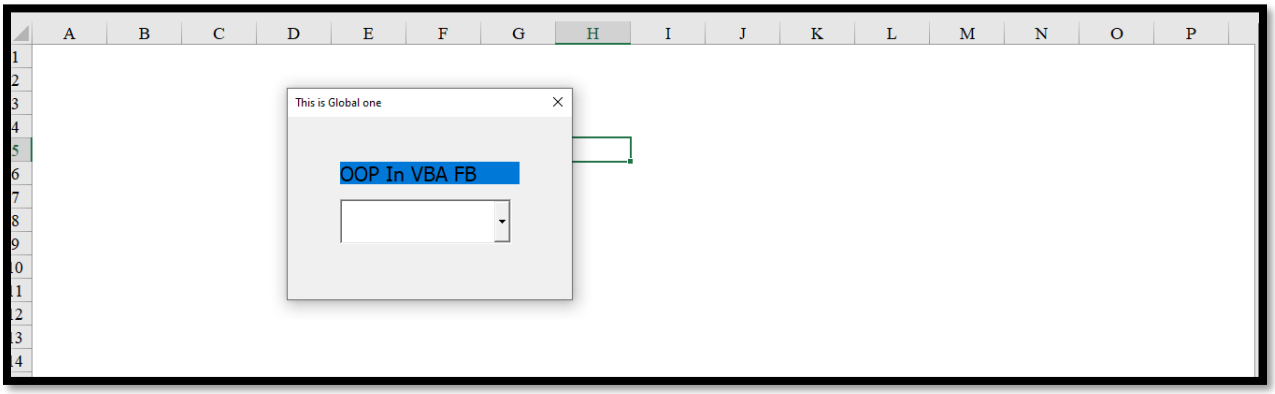
So, if you want to use this way of coding then you have to remember that Me is different while Create is in the call stack.

Now I want you to focus in other way:

So, If you want to change the code like this:

```
Public Function Create(GivenCompanyName As String) As EmployeeForm

    'Changes Here
    With Me
        .CompanyName = GivenCompanyName
        'Using Ref
        .CompanyNameLabel.Caption = GivenCompanyName
        'Using Me
        SetEmployeeNameUsingMe
    End With
    Set Create = Me

End Function


Private Sub SetEmployeeNameUsingMe()
    Me.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub

Private Sub SetEmployeeNameUsingRef(SetToUF As EmployeeForm)
    SetToUF.EmployeeComboBox.List = Array("Ismail", "Kamal", "Petr")
End Sub
```

And This will be the output:



Check we only have one form. Why is that? Because we are not creating any new object of that form type and it is editing in the original form no matter which variable you set that to.

In case of Userform normally we don't use multiple instances but in case of class it will be catastrophic. Let me add another class and example for clarify it.

So, this is the Constructor code:

```
Public Function Create(GivenName As String, GivenAge As Integer) As Person
    With Me
        .Name = GivenName
        .Age = GivenAge
        Set Create = Me
    End With
End Function
```

And here is the driver code and Output

```
(General)                    AnotherTest
Sub AnotherTest()
    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)
    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)
    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name
End Sub
```

```
Immediate
First Person Name : Khadiza
Second Person Name : Khadiza
```

So, both FirstPerson and SecondPerson refer to the same pointer and that's why both are same thing.

I do have another version with "With New Person" way and in that case, here is the code;

```
Public Function Create(GivenName As String, GivenAge As Integer) As Person
    'Only change in this part
    With New Person
        .Name = GivenName
        .Age = GivenAge
        Set Create = Me
    End With
End Function
```

Although we are creating new Object but we are still setting to the Me(Global Instance here) and that's why it is still the same object (Why blank >> Because we are not setting anything for the global one) .

```
(General)                    AnotherTest
Sub AnotherTest()

    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)

    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)

    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name

End Sub
```

```
Immediate
First Person Name :
Second Person Name :
```

Now check this with the GetMe property added and being used:

```
Public Property Get GetMe() As Person
    Set GetMe = Me
End Property

Public Function Create(GivenName As String, GivenAge As Integer) As Person
    'Only change in this part
    With New Person
        .Name = GivenName
        .Age = GivenAge
        Set Create = GetMe
    End With
End Function
```

Now the output is

```
(General)                    AnotherTest
Sub AnotherTest()

    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)

    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)

    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name

End Sub
```

```
Immediate
First Person Name :
Second Person Name :
```

And Maybe You are thinking what the hell? Okay we did a small mistake here. We Set Create = GetMe (This one is being called on the Global instance) and that's why still it is being referred by two different variables here. But if you change that to Set Create =. GetMe then you will get different result. Let me show you.

Here is the code and output.

```
(General)                    Create
End Property

Public Property Get Age() As Integer
    Age = This.Age
End Property

Public Property Let Age(ByVal RHS As Integer)
    This.Age = RHS
End Property

Public Property Get GetMe() As Person
    Set GetMe = Me
End Property

Public Function Create(GivenName As String, GivenAge As Integer) As Perso
    With New Person
        .Name = GivenName
        .Age = GivenAge
        "Here is the change
        Set Create = .GetMe
    End With
End Function
```

```
Immediate
First Person Name : Md Ismail Hosen
Second Person Name : Khadiza
```

Now probably you are thinking why the difference? It lies with the dot

When we are using dot then we are calling GetMe property on the With New Person (On the fly object>> With is holding object reference but you can't assign that to a variable except in this property way). So, it is returning the new object instead of that Me Global Object. So, although we are using Set GetMe=Me and Directly Set Create = Me. This two me is entirely different object. So, while we are not using the property of the New Person till it is still referencing to the global one. While we are using the dot then we are using the new up one.

Now Let's do some more test:

```
Sub AnotherTest2()

    Person.Name = "Test Name"  'Set to the global one

    Debug.Print
    Debug.Print "Global one name : " & Person.Name

    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)

    Debug.Print "Global one name : " & Person.Name

    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)

    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name

    "Let's be little crazy and then get the global one
    Dim ThirdPerson As Person
    Set ThirdPerson = Person.GetMe

    Debug.Print "Third Person Name : " & ThirdPerson.Name
End Sub
```

```
Global one name : Test Name
Global one name : Test Name
First Person Name : Md Ismail Hosen
Second Person Name : Khadiza
Third Person Name : Test Name
```

See here I set the value on the global object to Test Name And we get that throughout the session.

And in the next call delete the first Name setup line and run the code again:

(General) | AnotherTest2

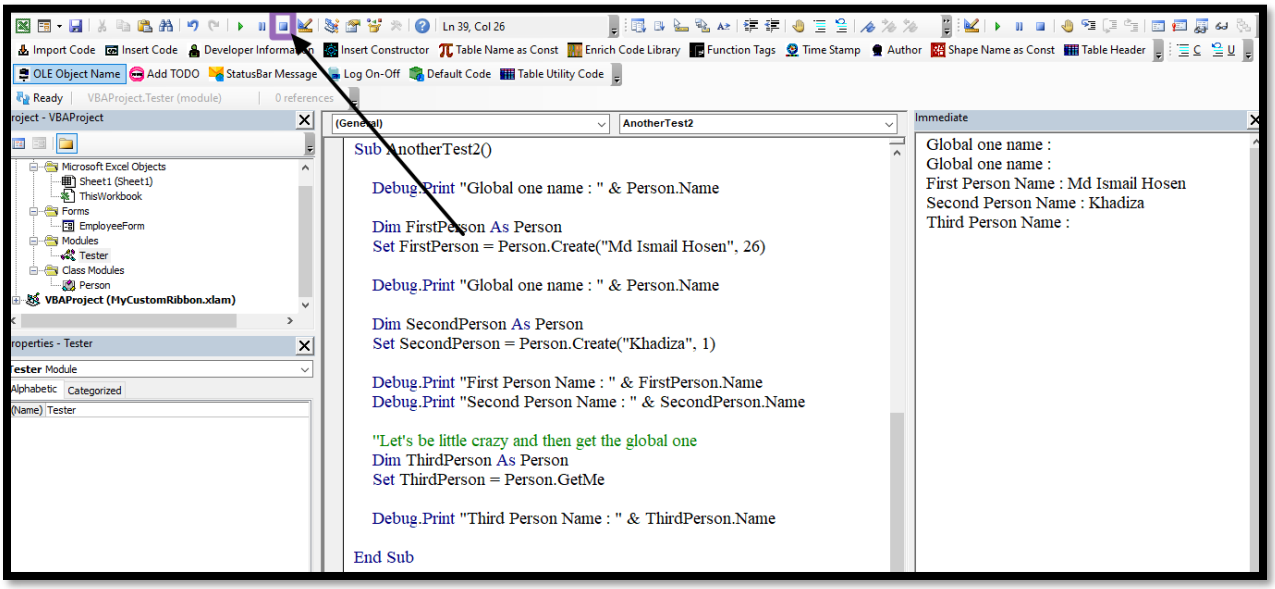
```
Sub AnotherTest2()

    Debug.Print "Global one name : " & Person.Name

    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)

    Debug.Print "Global one name : " & Person.Name

    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)

    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name

    "Let's be little crazy and then get the global one
    Dim ThirdPerson As Person
    Set ThirdPerson = Person.GetMe

    Debug.Print "Third Person Name : " & ThirdPerson.Name

End Sub
```

Immediate

```
Global one name : Test Name
Global one name : Test Name
First Person Name : Md Ismail Hosen
Second Person Name : Khadiza
Third Person Name : Test Name
```

And still that global one is holding its value due to variable life.

If you want to clear then you can either click on the Reset Button or you have to do it in the code.



This will not reset the global one.

```
Sub AnotherTest2()

'    Person.Name = "Test Name"
    Debug.Print "Global one name : " & Person.Name

    Dim FirstPerson As Person
    Set FirstPerson = Person.Create("Md Ismail Hosen", 26)

    Debug.Print "Global one name : " & Person.Name

    Dim SecondPerson As Person
    Set SecondPerson = Person.Create("Khadiza", 1)

    Debug.Print "First Person Name : " & FirstPerson.Name
    Debug.Print "Second Person Name : " & SecondPerson.Name

    "Let's be little crazy and then get the global one
    Dim ThirdPerson As Person
    Set ThirdPerson = Person.GetMe

    Debug.Print "Third Person Name : " & ThirdPerson.Name
    Set ThirdPerson = Nothing

End Sub
```

GitHub Repo: [1504168/Command-Pattern-Clearly (github.com)]