

# Create and Use Interfaces

---

Last update: Wednesday, November 6, 2013

Required Limnor Studio build: [5.6.1.575](#)

## Contents

Introduction .....	2
Sample Scenarios .....	2
Interface Design .....	3
Create interface .....	3
Interface Properties .....	4
Interface Methods .....	6
Implement Interface .....	7
Add Interface .....	7
Implement Properties .....	9
Implement SaveButton – Form2 .....	9
Implement NewRecButton – Form2 .....	10
Implement SaveButton – Form3 .....	12
Implement NewRecButton – Form3 .....	13
Implement Methods .....	14
Implement AddRecord – Form2 .....	14
Implement AddRecord – Form3 .....	16
Implement UpdateDatabase – Form2 .....	17
Implement UpdateDatabase – Form3 .....	18
Use Interface .....	19
Use interface in methods .....	19
Button handler for New Record .....	20
Button handler for Save .....	24
Passing interface .....	28
Passing Form2 .....	29
Passing Form3 .....	33

Test.....	37
Feedback .....	39

## Introduction

In Limnor Studio user forum, a user asked about how to create a method so that this method may work for several different forms to perform same actions. Thus, such a method needs to be created just once, not to be repeatedly created for each form. One way to achieve such programming is by creating and using interfaces. An interface can be defined to represent all different forms. A method may be developed to work on the interface. Thus the method may work on forms implementing the interface. This article demonstrates some samples.

## Sample Scenarios

Suppose we are developing two forms:

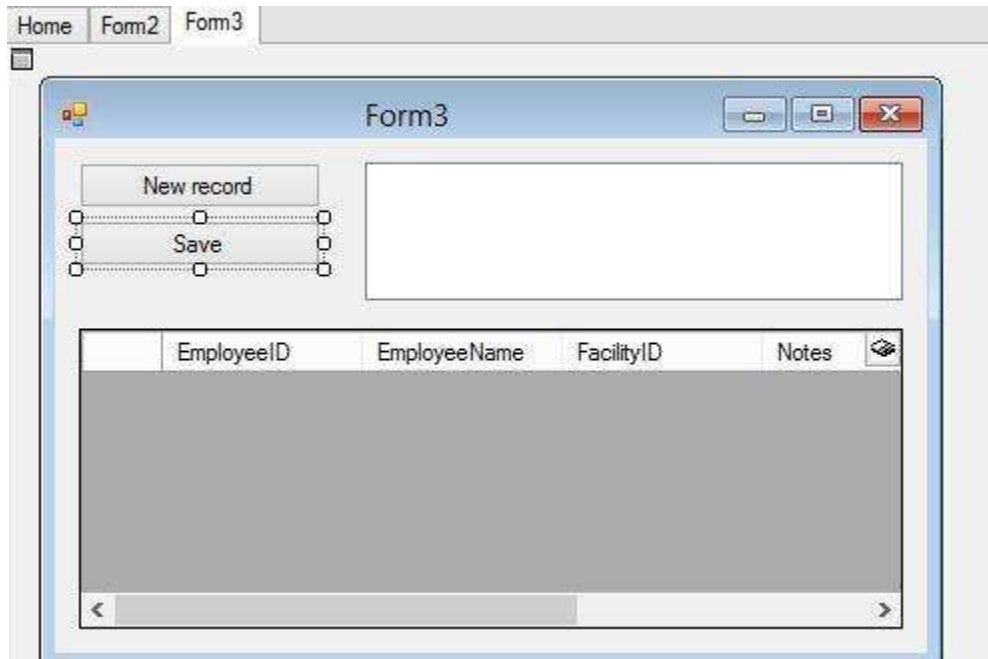
The screenshot displays a software application window with a tabbed interface. The tabs are labeled 'Home', 'Form2', and 'Form3'. The 'Form2' tab is active, showing a form titled 'Form2'. The form contains the following elements:

- A text label 'Product ID:' followed by a text input field containing the value '0'.
- A text label 'Product name:' followed by an empty text input field.
- Two navigation buttons: a left arrow button (<) and a right arrow button (>).
- A button labeled 'New record'.
- A button labeled 'Save'.

At the bottom left of the window, there is a status bar with a small icon and the text 'EasyDataSet1'.

## Create and Use Interfaces

---



These forms look quite differently. But we'll identify some common programming tasks for them and see how we may use interfaces to reduce repeated programming. They both have a "New record" button and a "Save" button. Initially the "Save" button is disabled. On clicking "New record" button, a new record is added to the form; the "Save" button is enabled; and the "New record" button is disabled. On clicking "Save" button, the data on the form are saved to the database; the "Save" button is disabled; and the "New record" button is disabled.

Note that the above sample scenarios are for demonstration purpose only, you do not need to try to make sense of business logics of such programming requirements.

From such programming requirements, we may identify that the handling of "Save" and "New record" buttons are quite same for both forms. It is possible to develop a single method to handle "Save" button on both forms, and a single method to handle "New record" button for both forms.

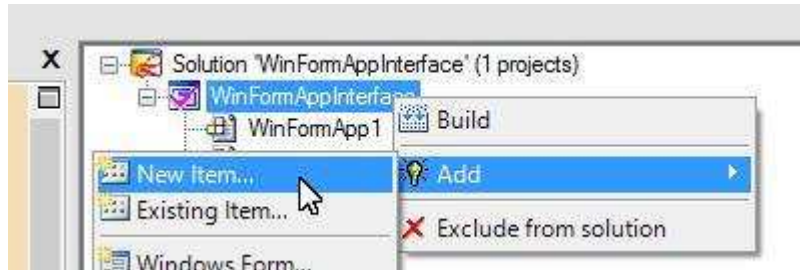
### Interface Design

An interface consists of properties, methods and events, representing commonality of all objects it represents. In our sample scenarios, common programming tasks involve a "Save" button and a "New record" button, a "create record" method and a "save" method. So, our new interface may contain two properties and two methods. Let's create such an interface.

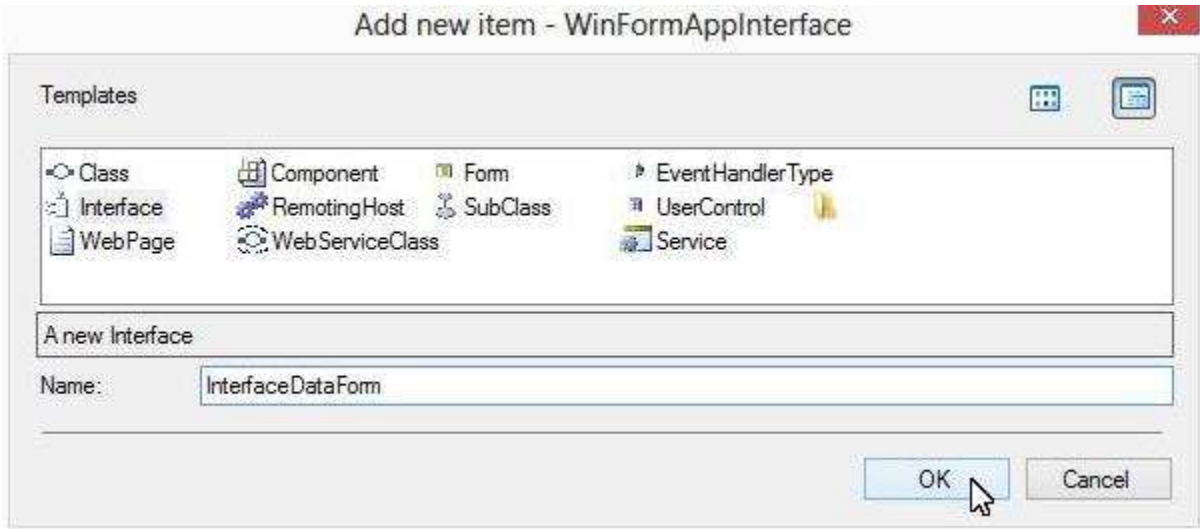
### Create interface

Right-click the project and choose "Add" and "New Item...":

## Create and Use Interfaces



Select "Interface" and give it a name, InterfaceDataForm:

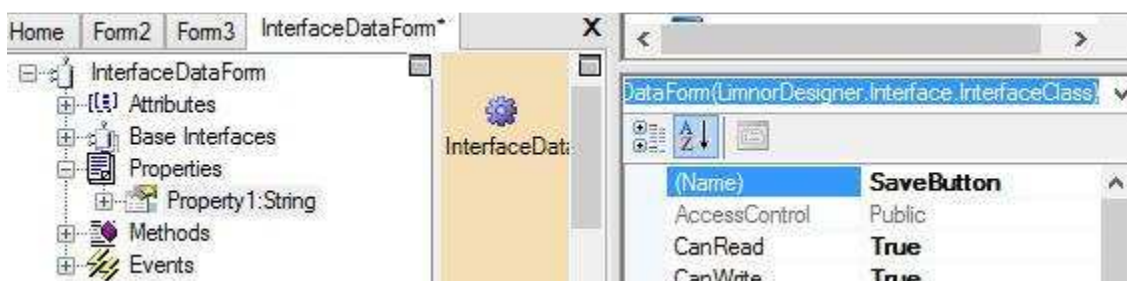


### Interface Properties

Right-click Properties; choose "Add property":

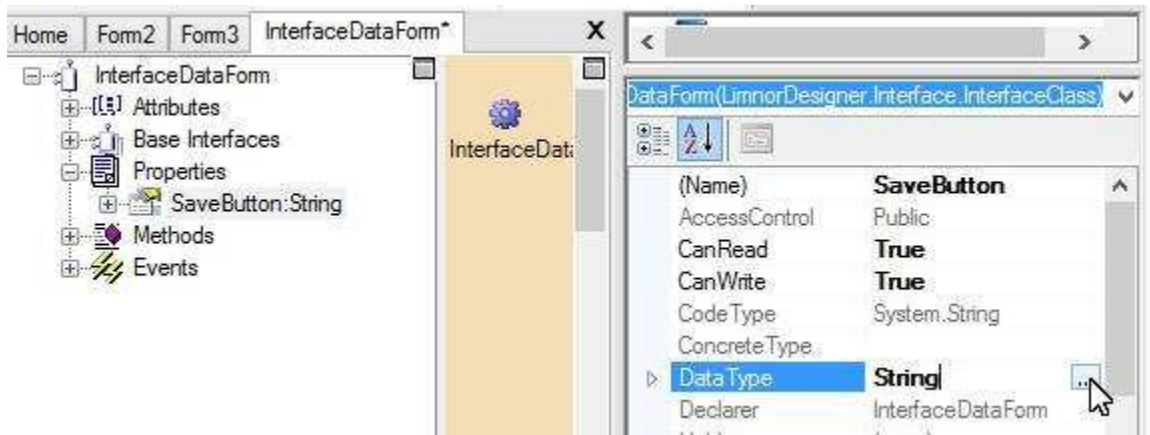


Rename the new property to "SaveButton":



## Create and Use Interfaces

Change the type of the property to Button:



To find and select Button class, expand Namespaces:



Expand System.Windows.Forms:

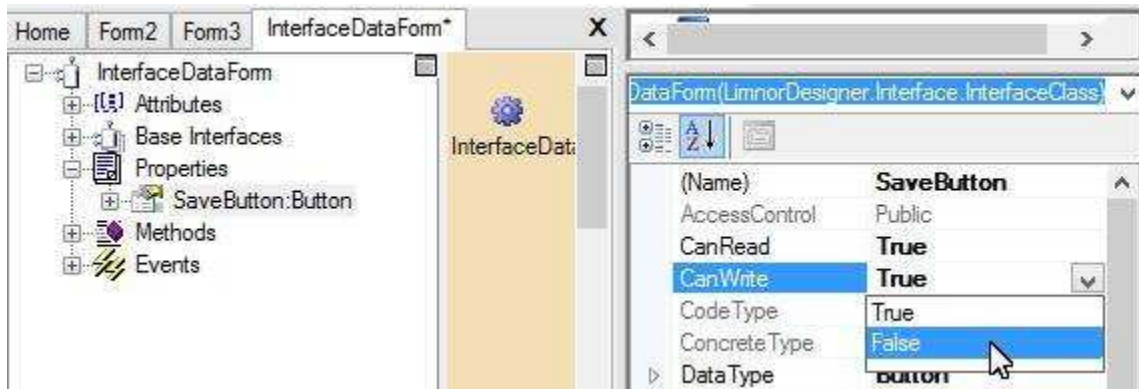


Select Button:

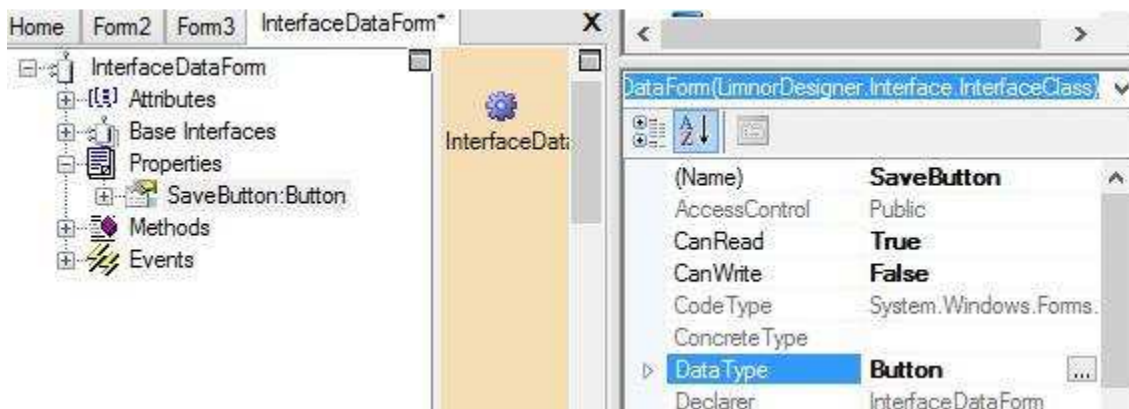


Set CanWrite to False because we do not want to allow changing button instance at runtime:

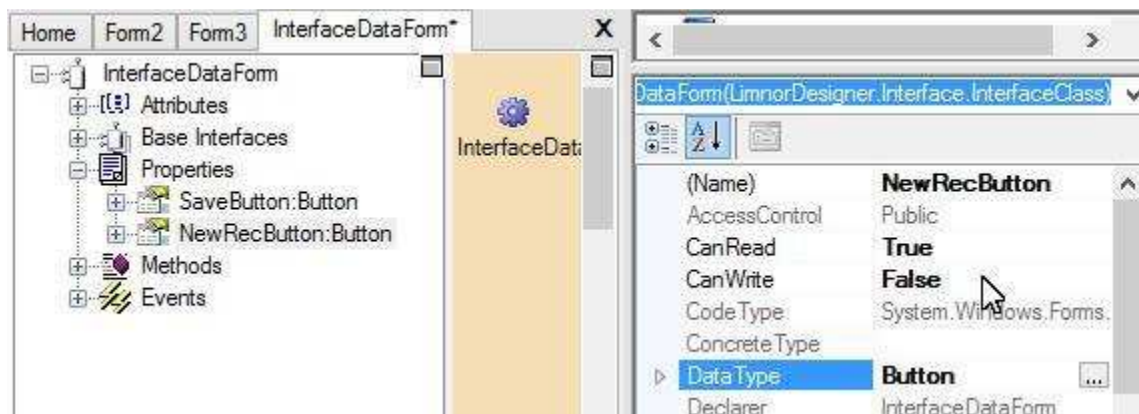
## Create and Use Interfaces



We thus created a property for the interface:



In the same process, we may create another property named NewRecButton:

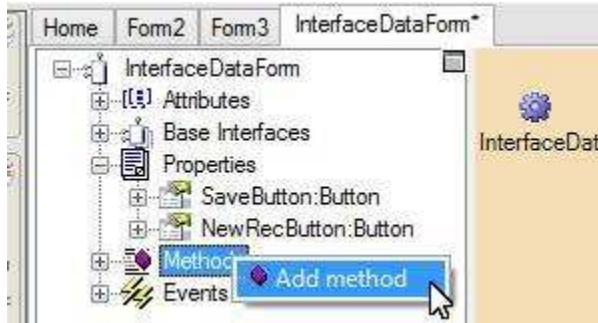


### Interface Methods

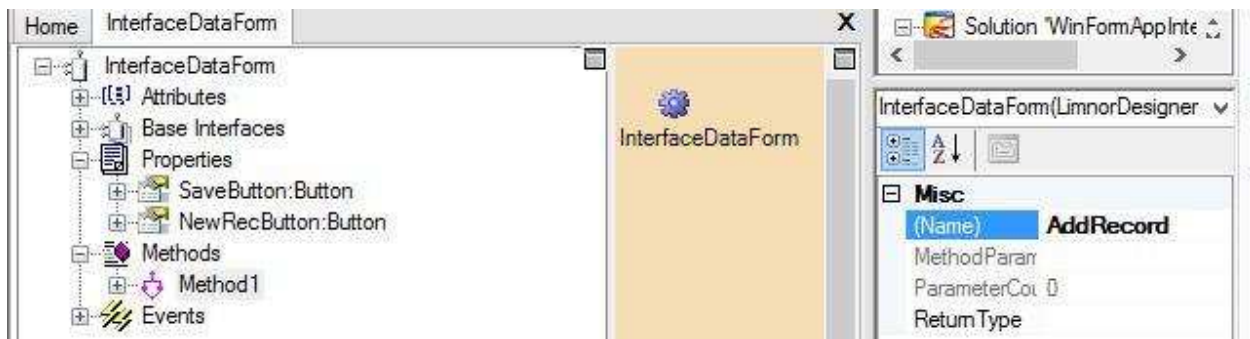
Right-click "Methods"; choose "Add method":



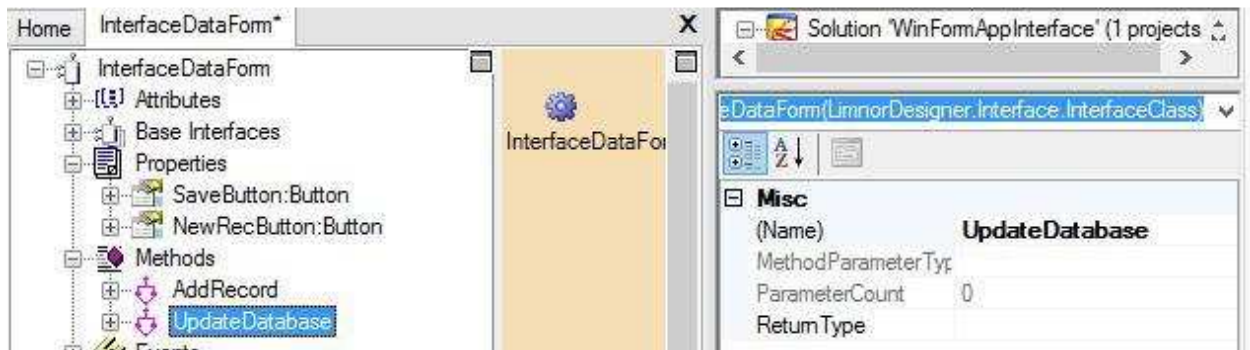
## Create and Use Interfaces



Rename the new method to AddRecord:



Add another method named UpdateDatabase:

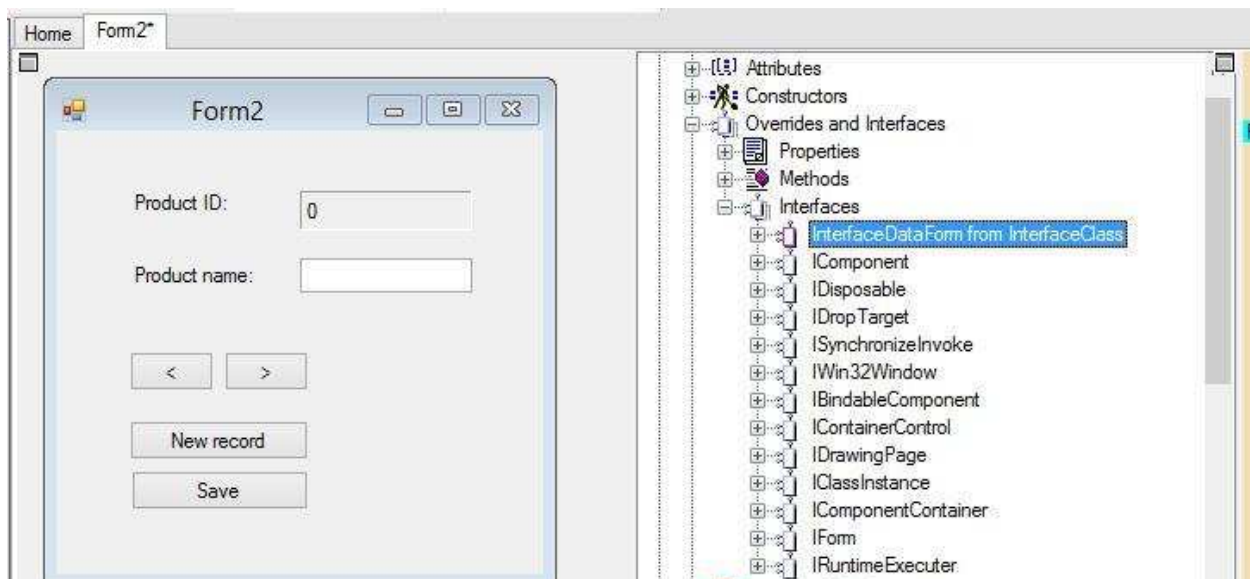
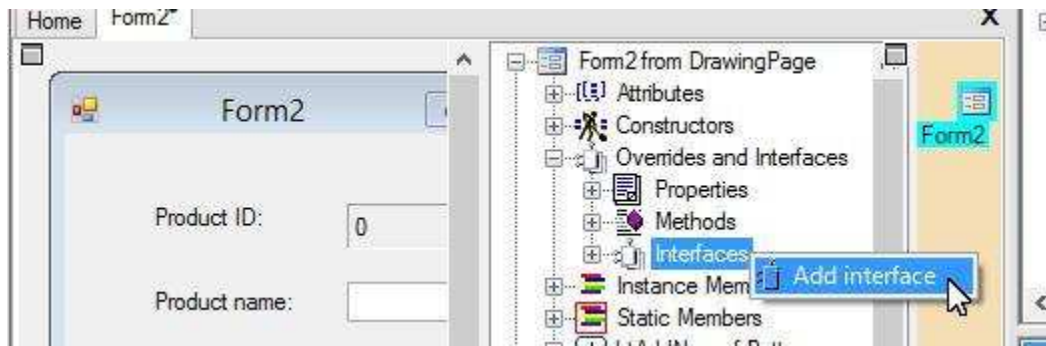


## Implement Interface

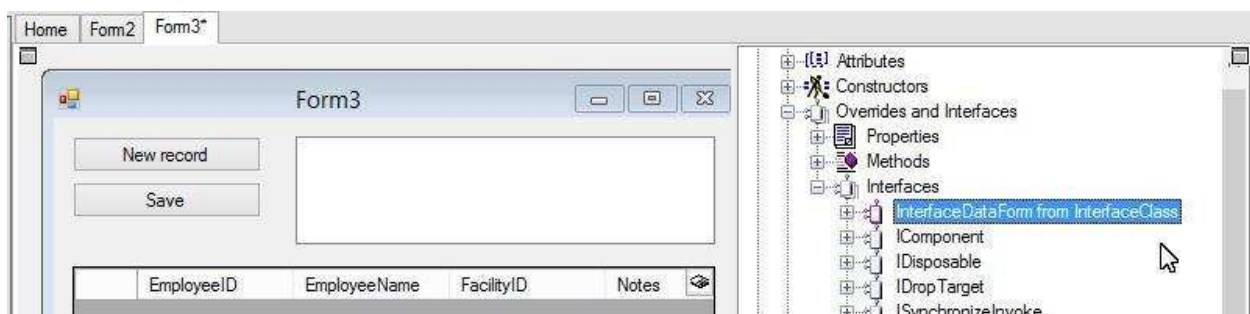
### Add Interface

We want Form2 and Form3 become interface InterfaceDataForm. This process is referred to as “implementing InterfaceDataForm”. This is done by adding the interface to interface list:

## Create and Use Interfaces



We also want to add InterfaceDataForm to Form3:





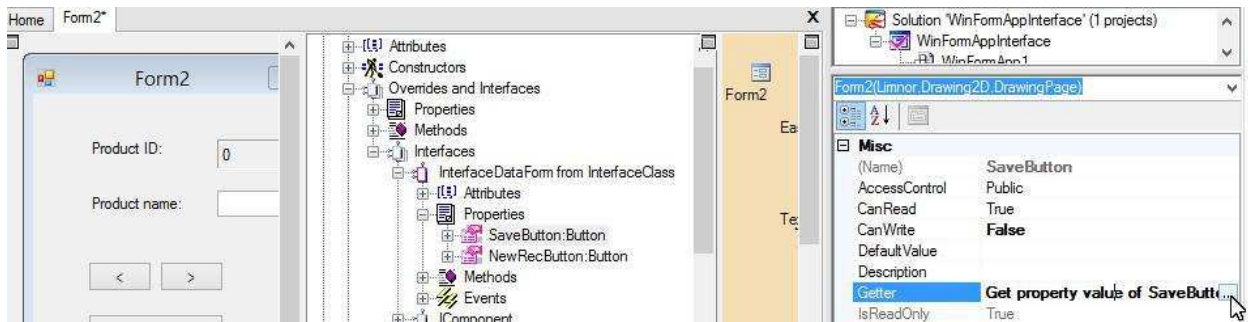
## Create and Use Interfaces

We can see that the interface `InterfaceDataForm` appears under “Interfaces”. Note that there are many other interfaces listed. Those interfaces are implemented by Microsoft .Net Framework to create basic form functionality. Now it is our turn to implement `InterfaceDataForm`, as shown below.

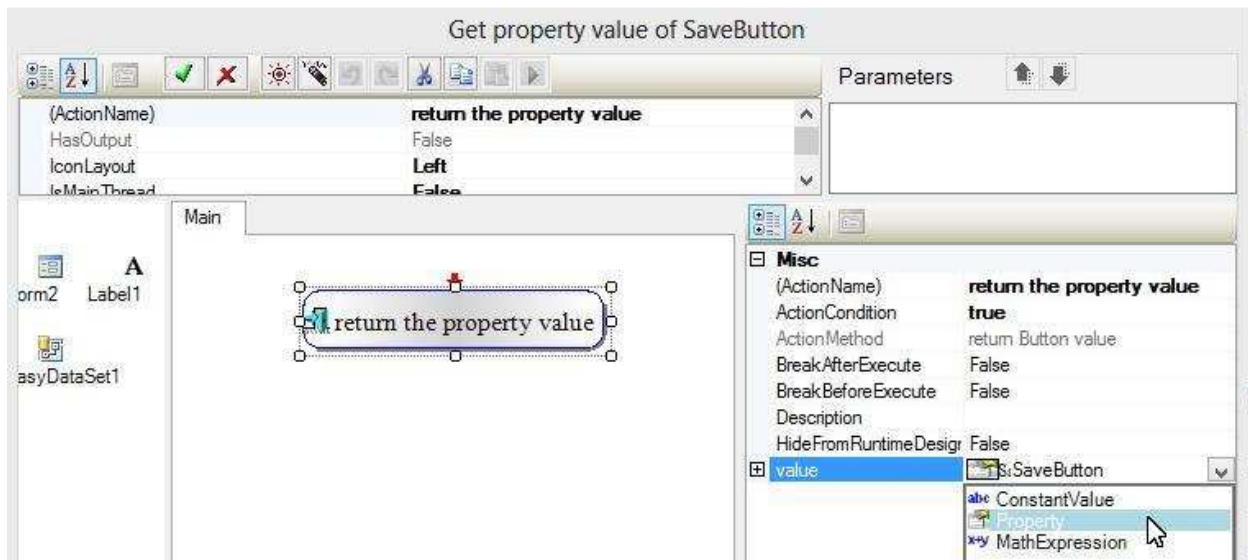
### Implement Properties

#### Implement SaveButton – Form2

To implement an interface property is to edit its Getter:



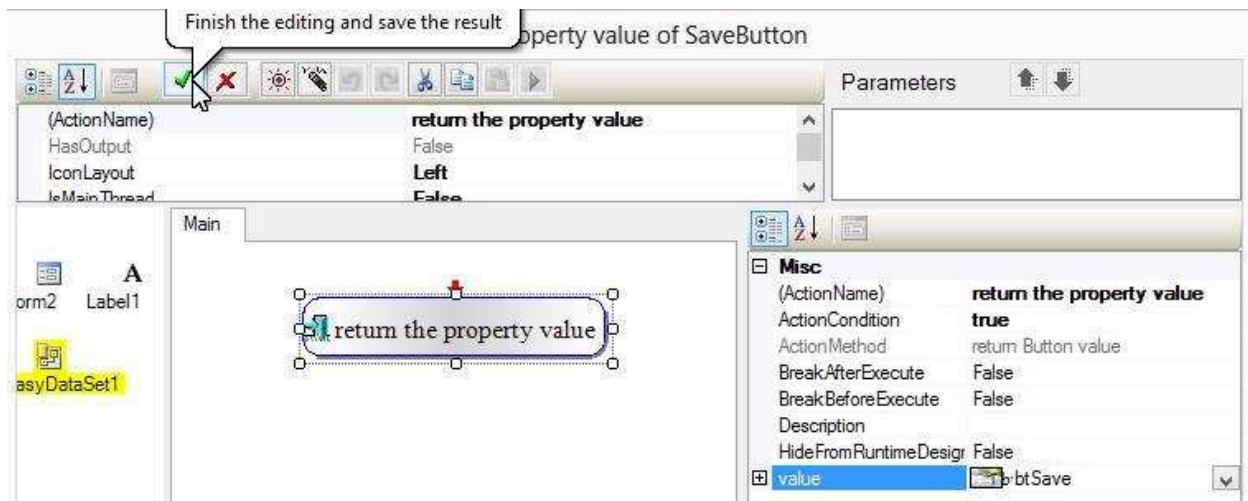
By default, a property Getter simply returns an internal value. We want to change it to return the “Save” button of the form:



## Create and Use Interfaces

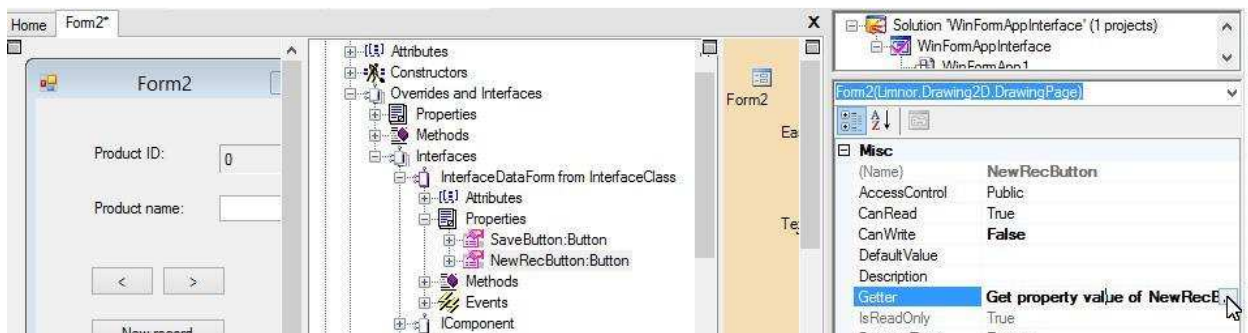


That is all for the Getter:

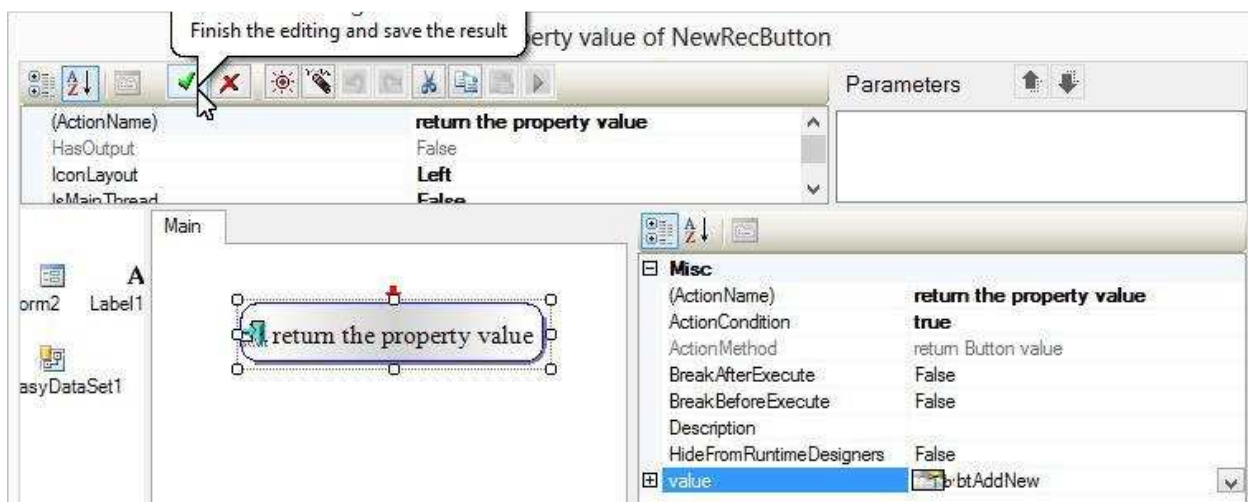
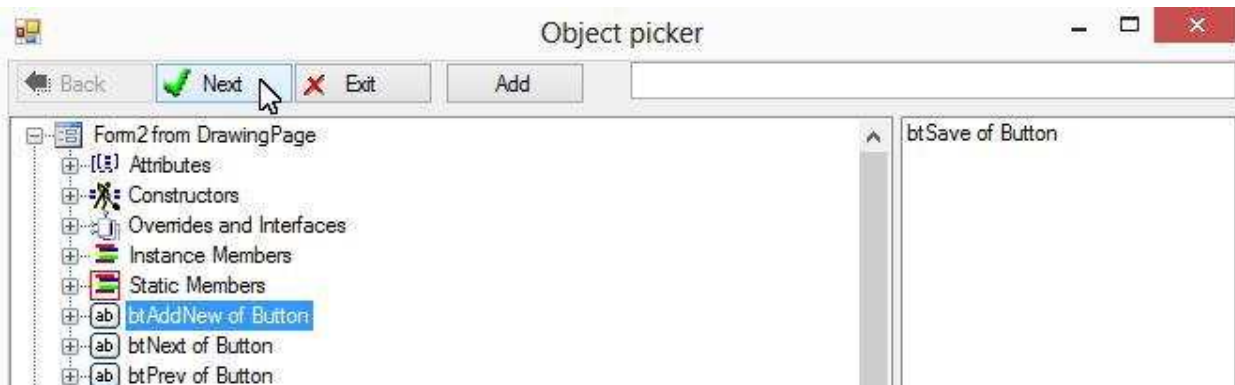
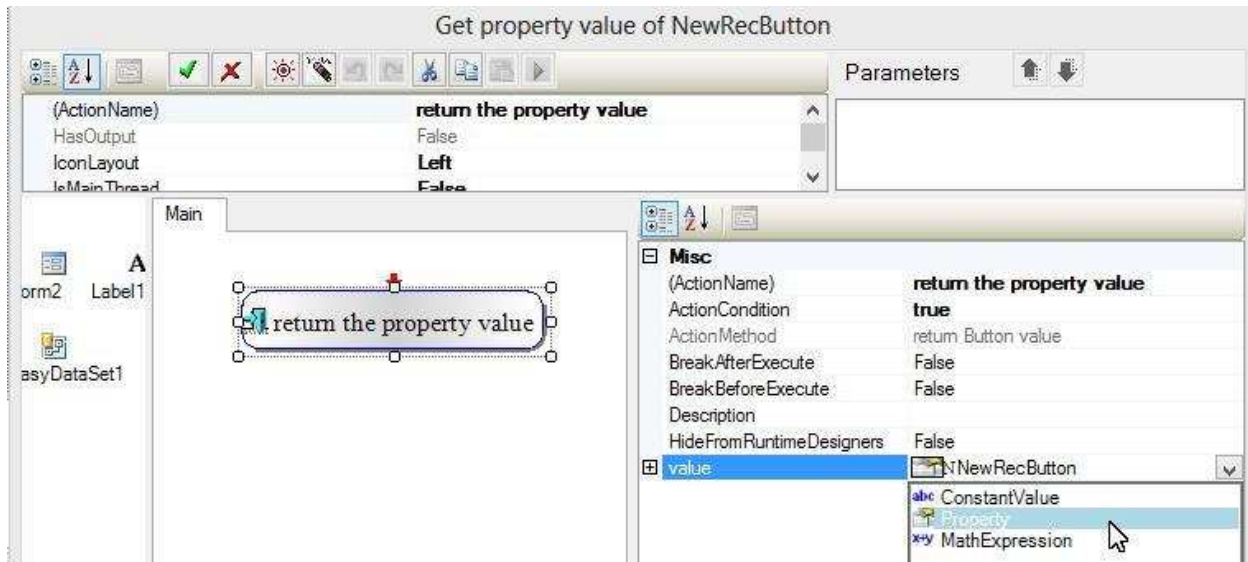


### Implement NewRecButton - Form2

In the same way, we need to implement interface property NewRecButton:

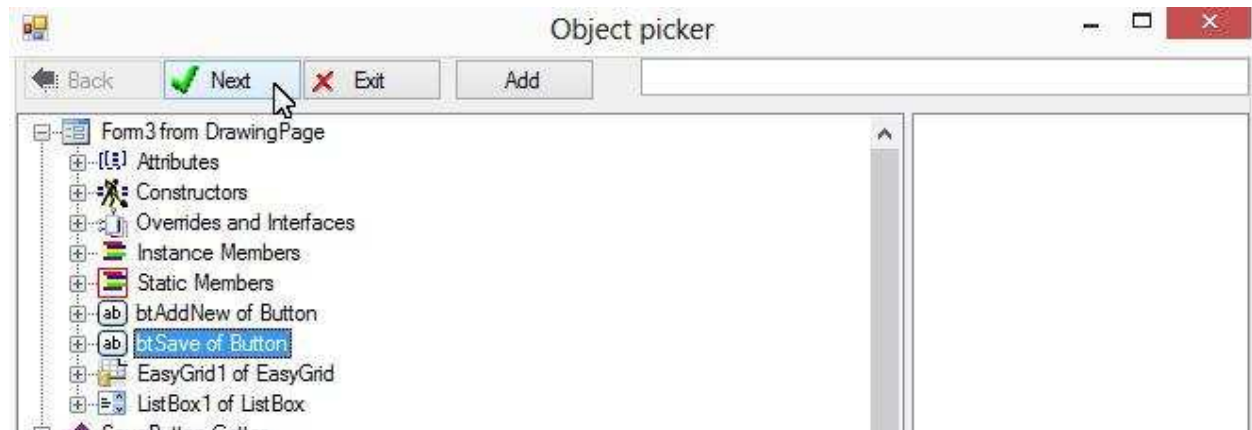
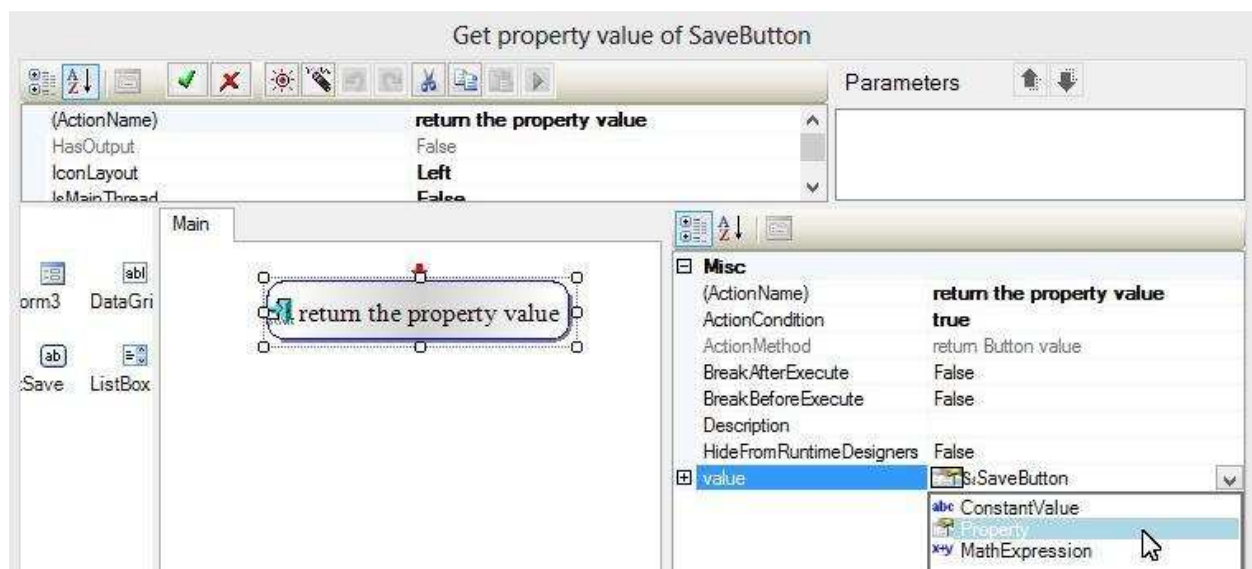
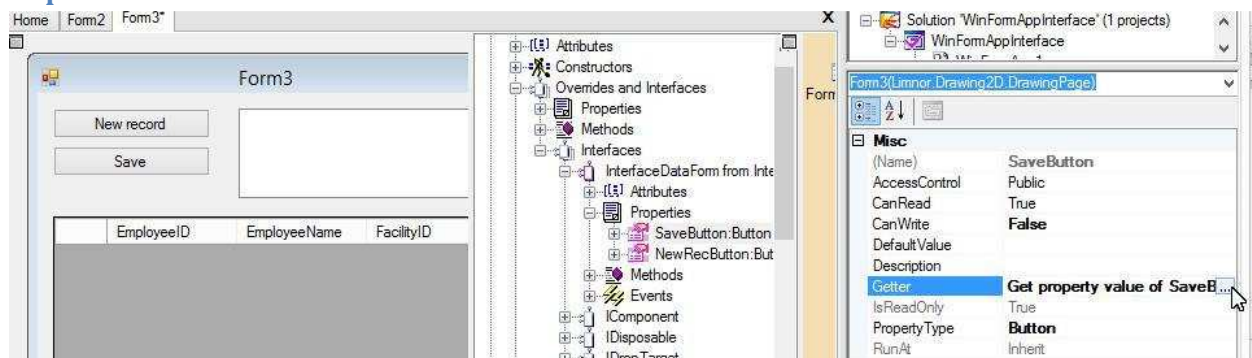


## Create and Use Interfaces



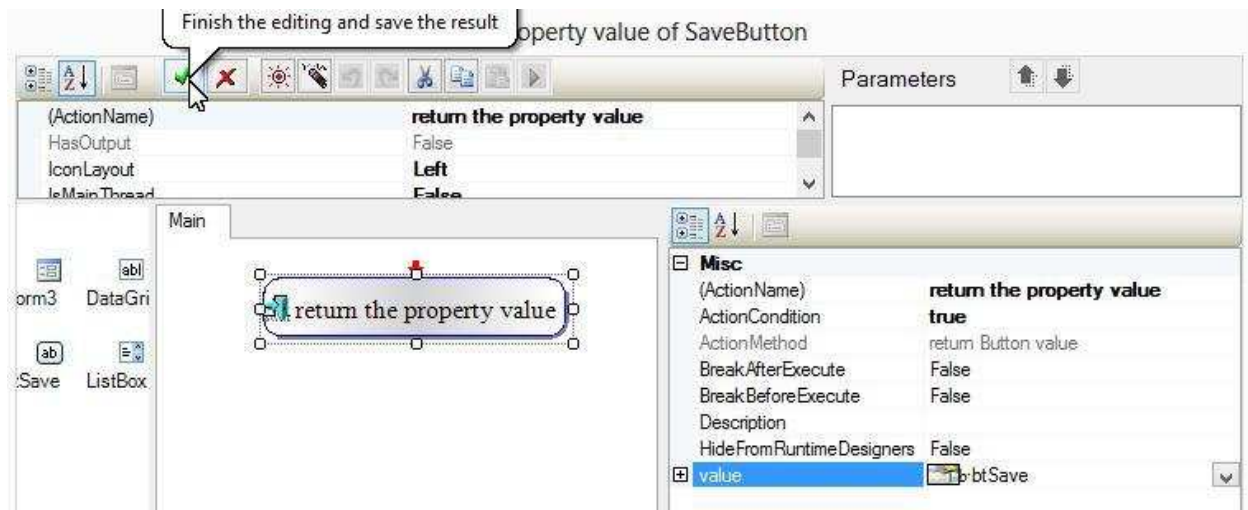
# Create and Use Interfaces

## Implement SaveButton – Form3

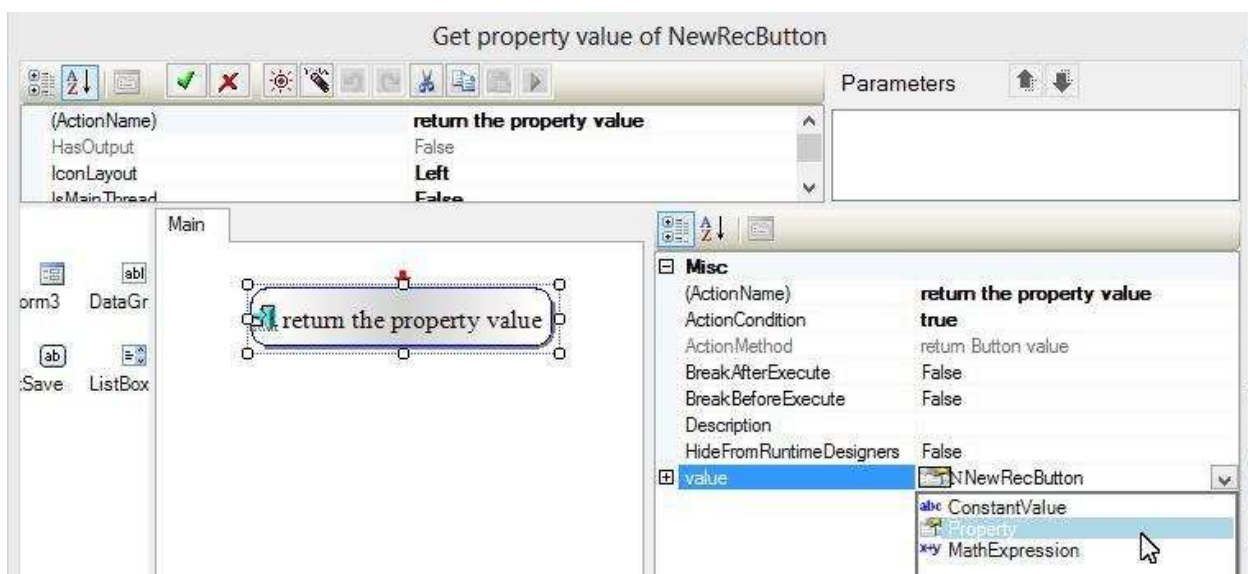
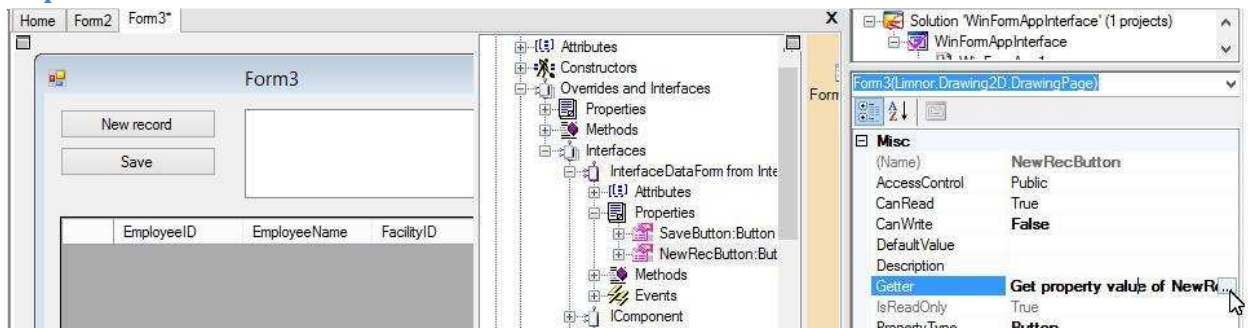




## Create and Use Interfaces

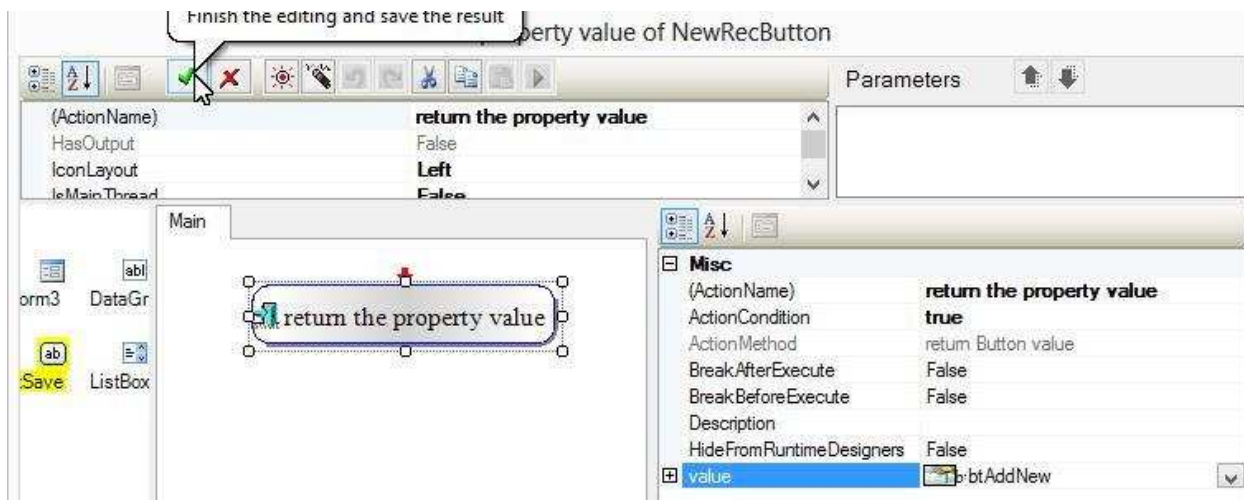
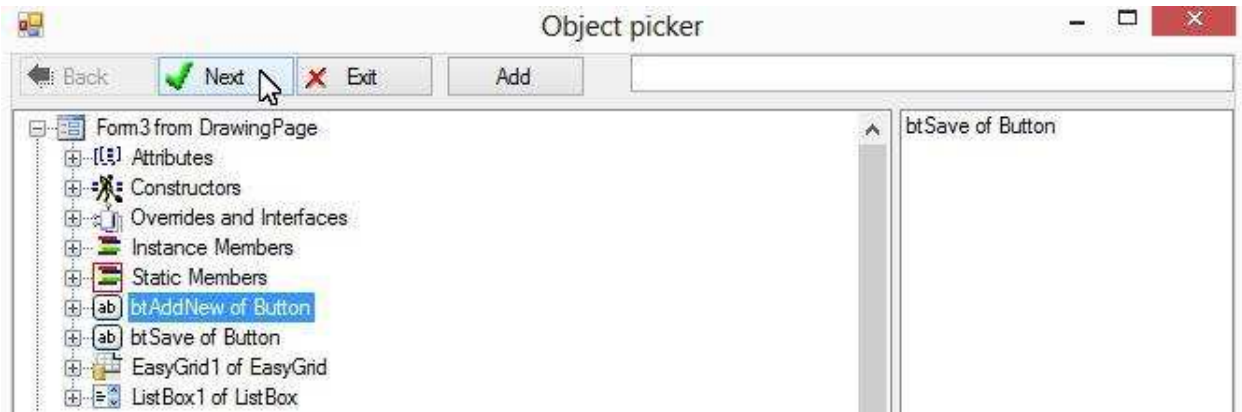


### Implement NewRecButton - Form3





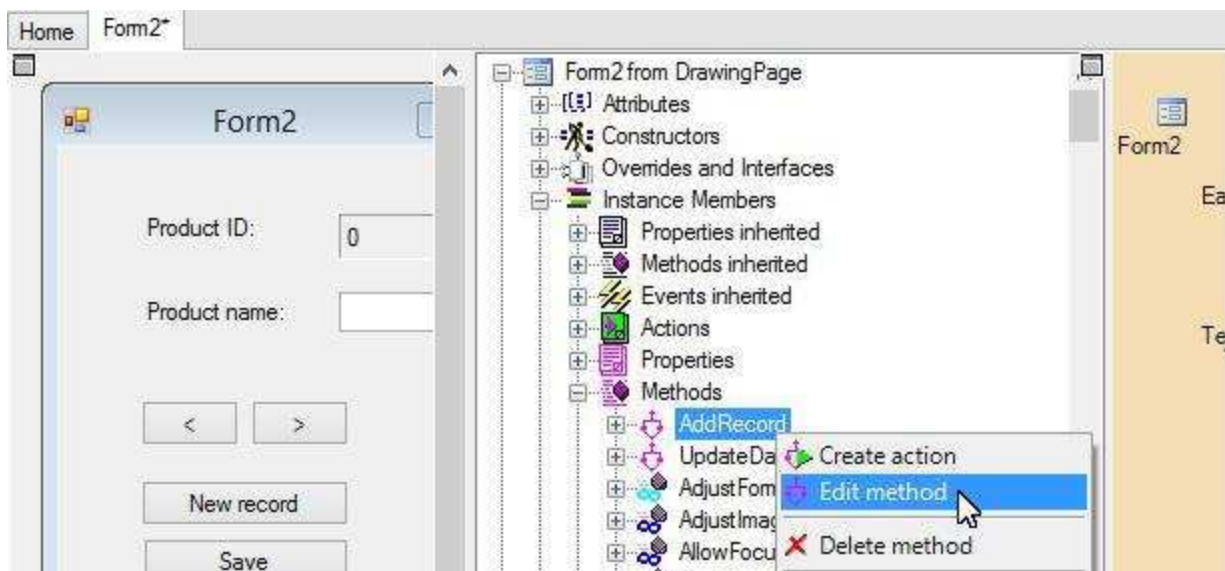
## Create and Use Interfaces



## Implement Methods

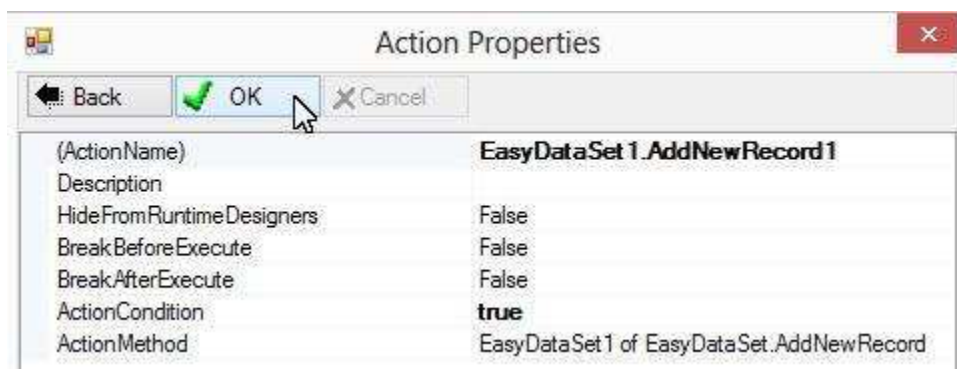
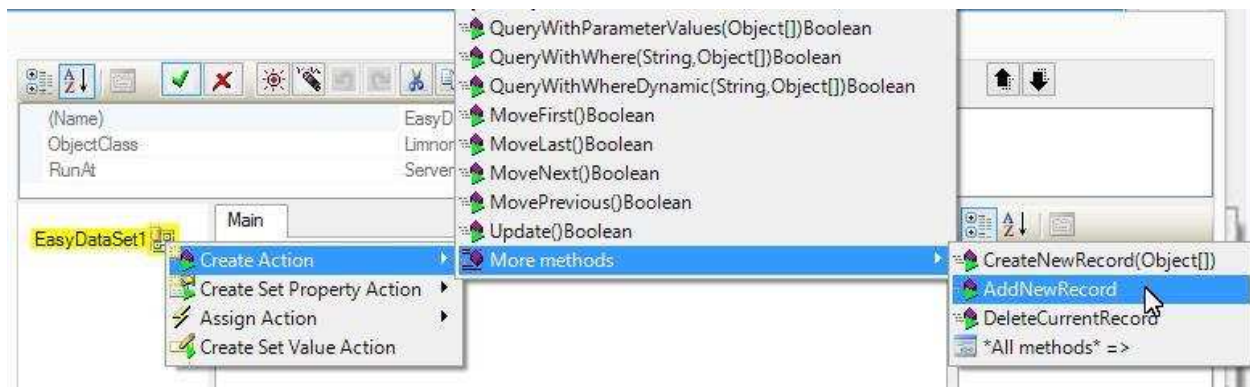
### Implement AddRecord – Form2

To implement an interface method is to edit it:

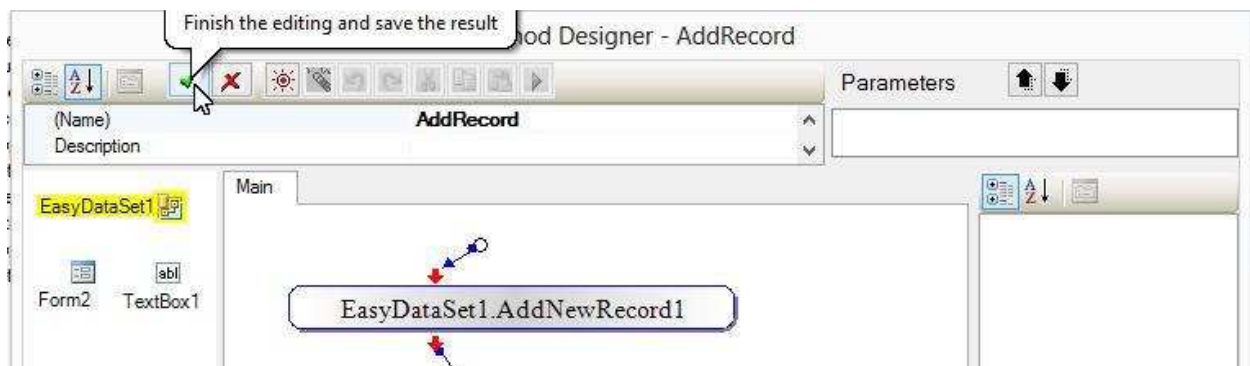


## Create and Use Interfaces

For implementing AddRecord for Form2, we execute an AddNewRecord action of the EasyDataSet:

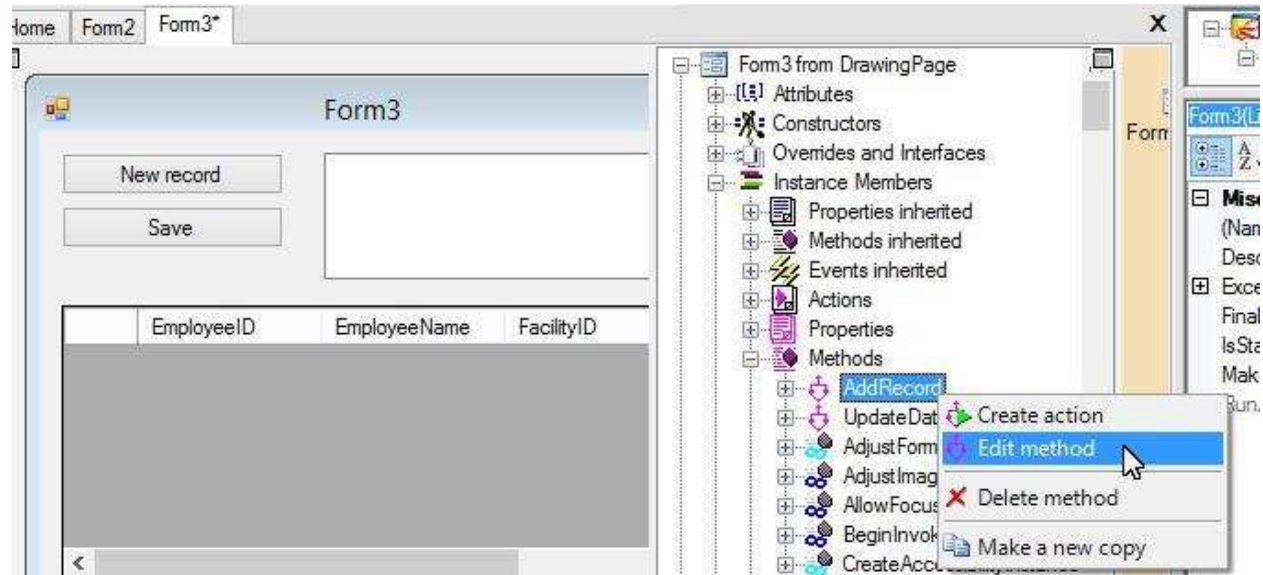


For this sample, that is all for this method implementation:

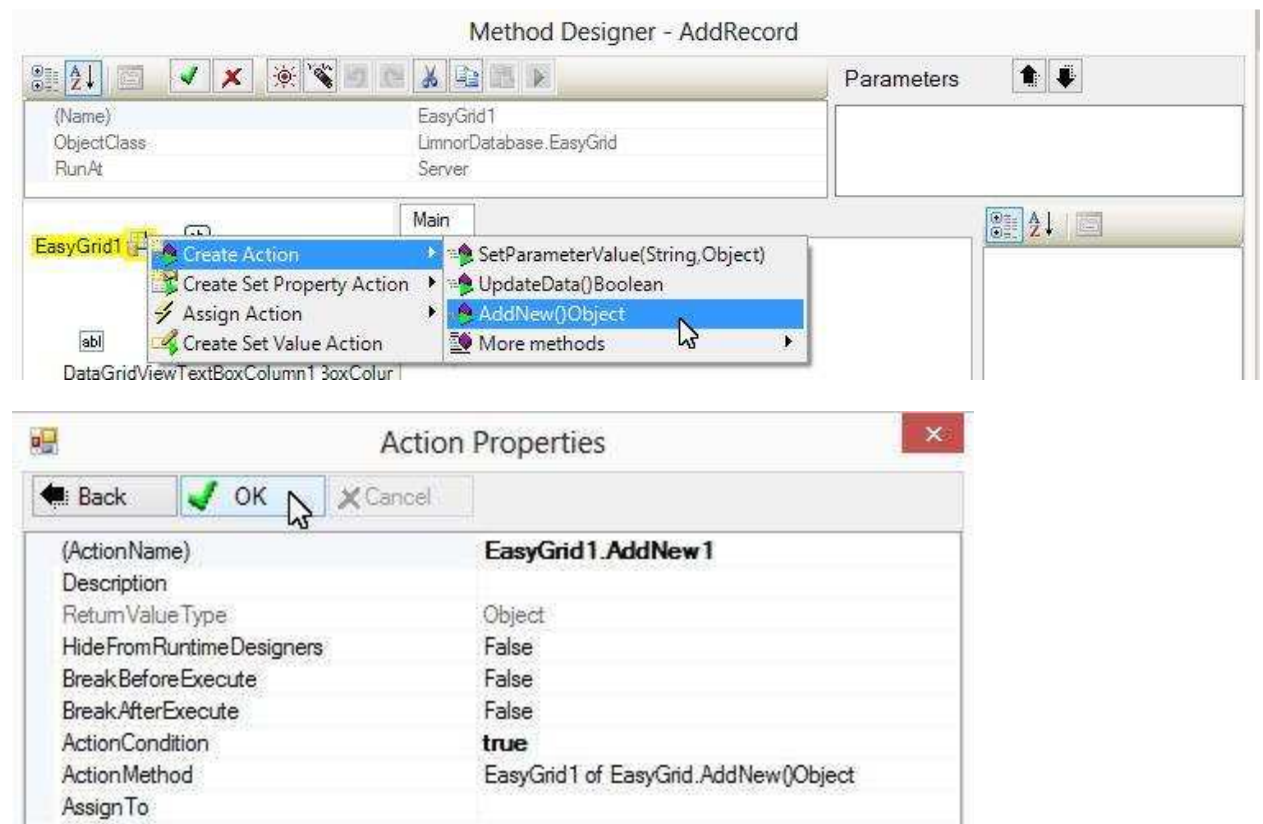


## Create and Use Interfaces

### Implement AddRecord – Form3



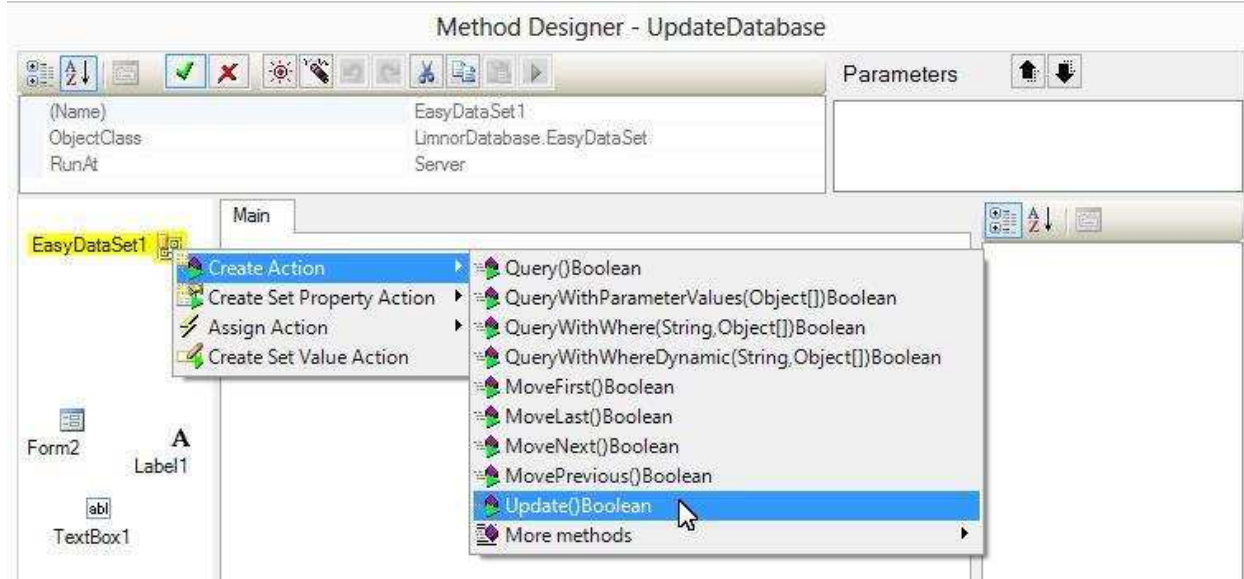
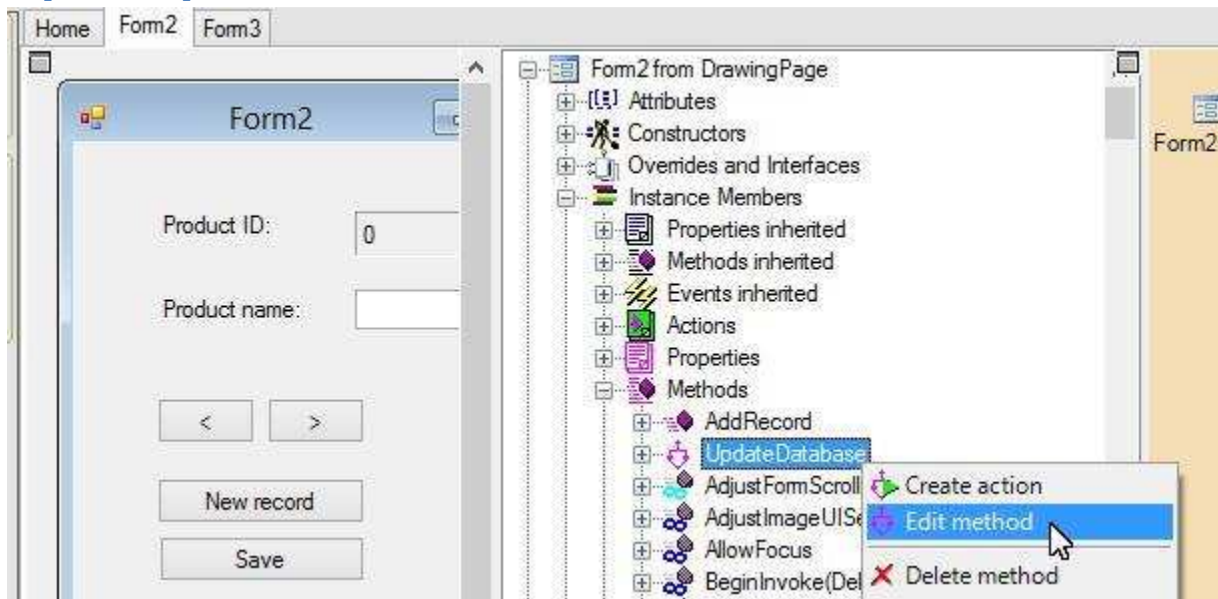
For this sample, we execute an AddNew action in this implementation:



## Create and Use Interfaces

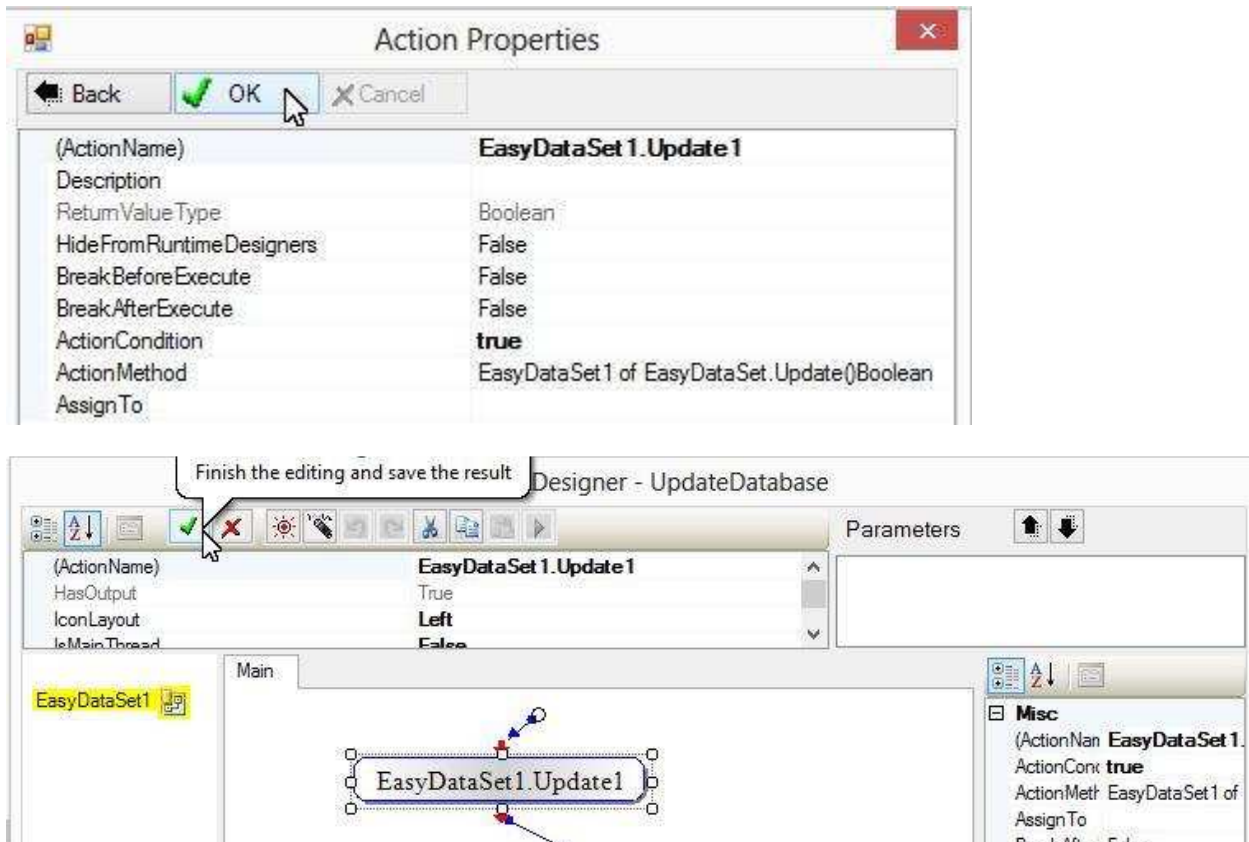


### Implement UpdateDatabase - Form2

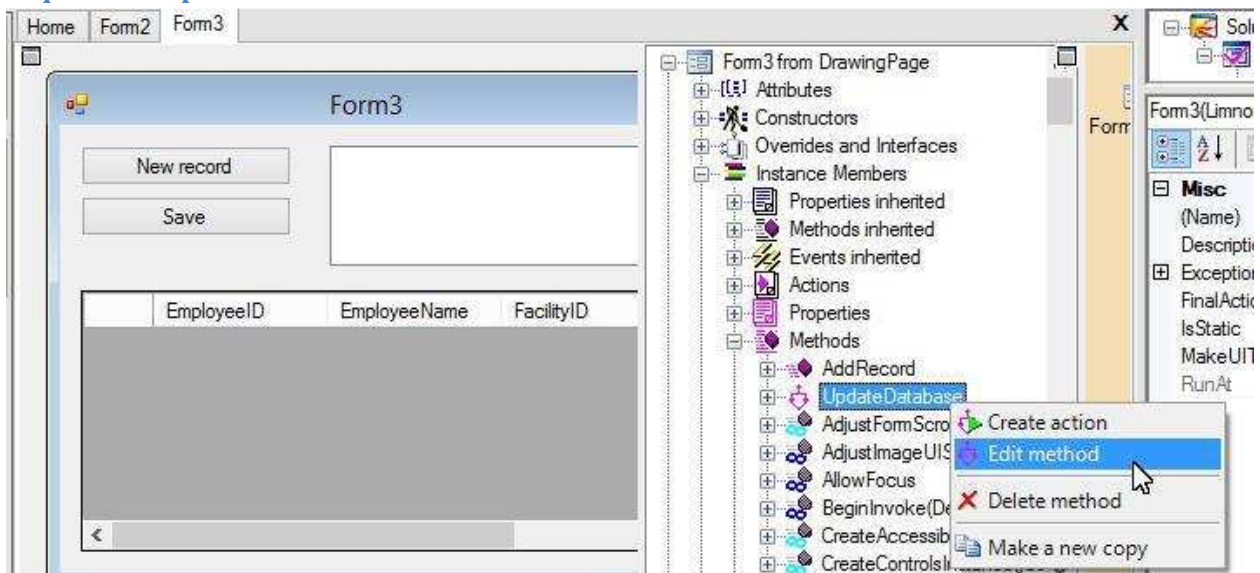




## Create and Use Interfaces

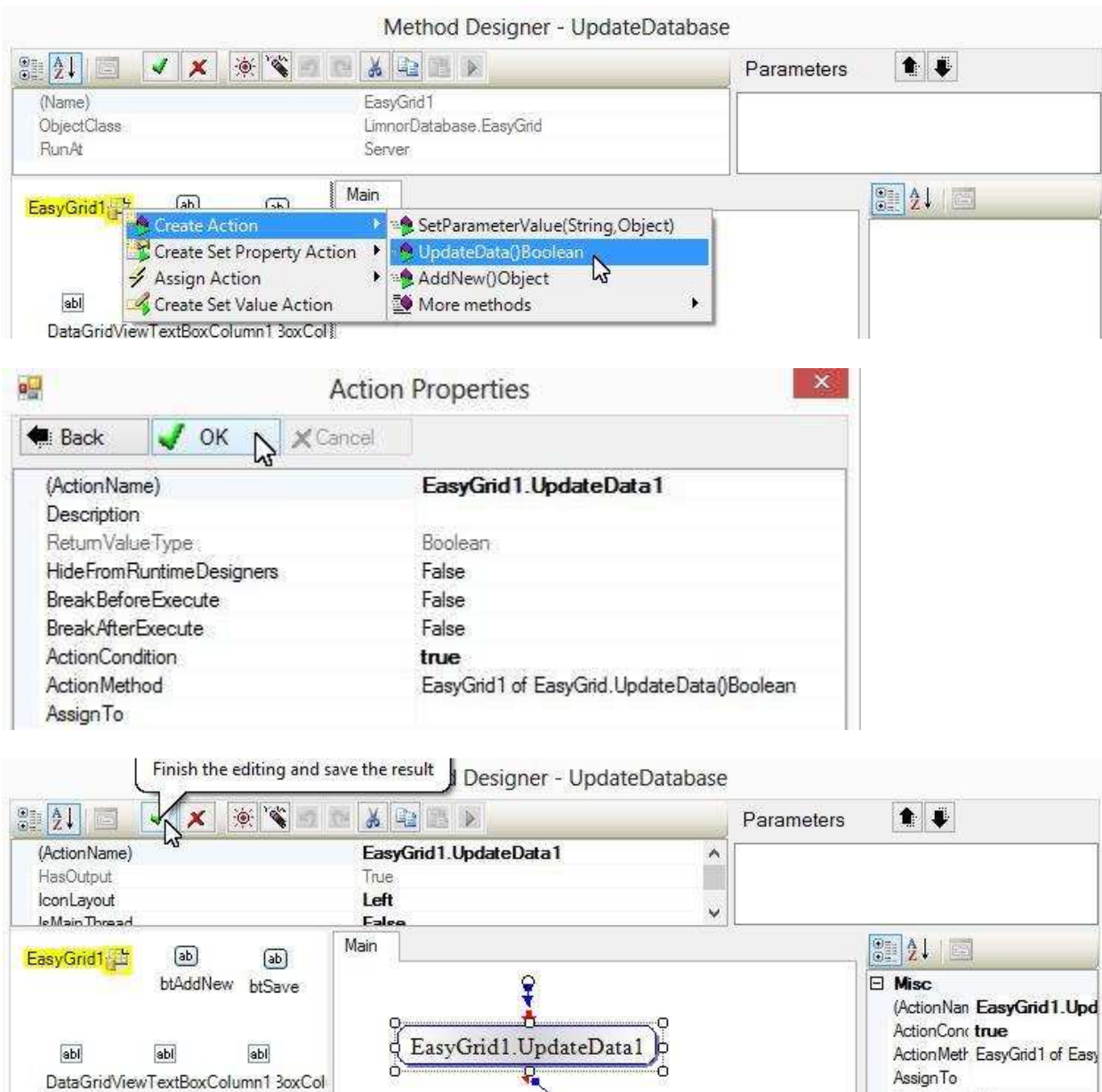


### Implement UpdateDatabase - Form3





## Create and Use Interfaces



### Use Interface

Now Form2 and Form3 may appear as InterfaceDataForm. We may do programming based on InterfaceDataForm, instead of Form2, Form3, or other forms implementing InterfaceDataForm. There are endless ways of programming on interfaces. Here we show a few simple examples.

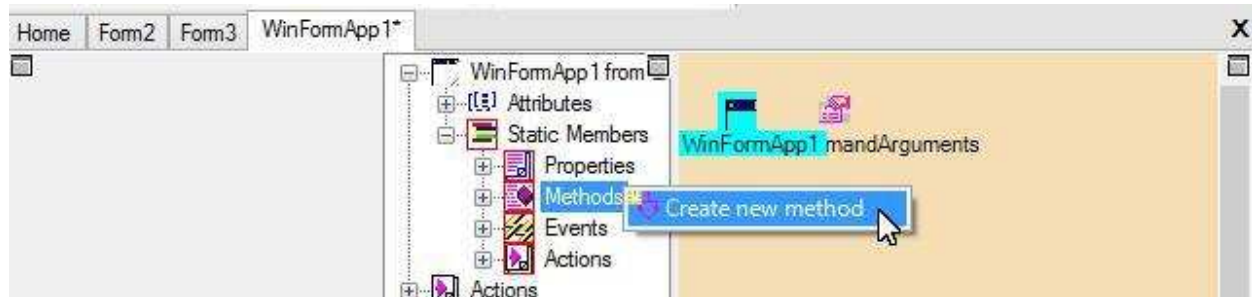
### Use interface in methods

We may create methods and pass interfaces as parameters. Such methods may be executed for all different kinds of objects as long as the objects implement required interface.

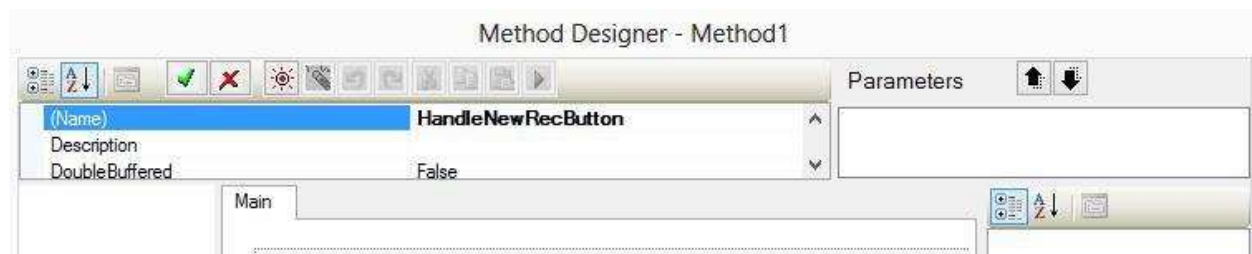
## Create and Use Interfaces

### Button handler for New Record

As a sample, we create a static method for handling “New record” button. Suppose we create this method in the WinFormApp1 class:



Rename the new method to HandleNewRecButton:



Add a method parameter:



Select InterfaceDataForm as the parameter type:

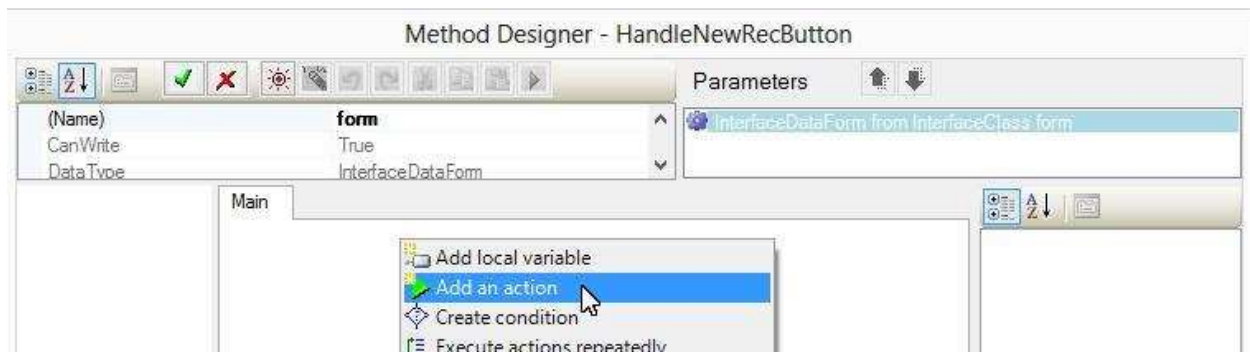


Rename the parameter to “form”:

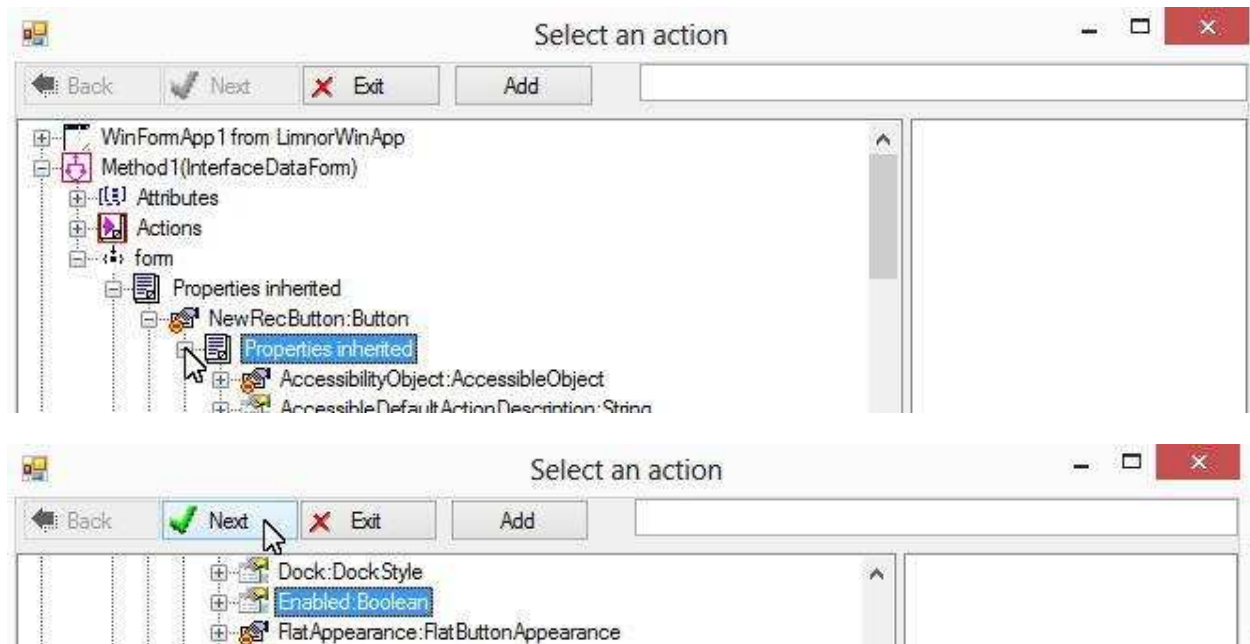
## Create and Use Interfaces



Add an action to disable the new record button:

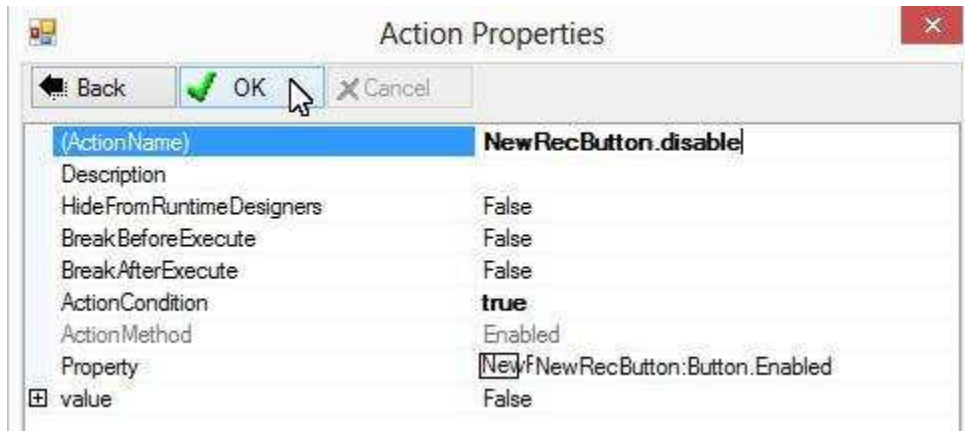


Select "Enabled" property of the new record button of the "form" parameter:

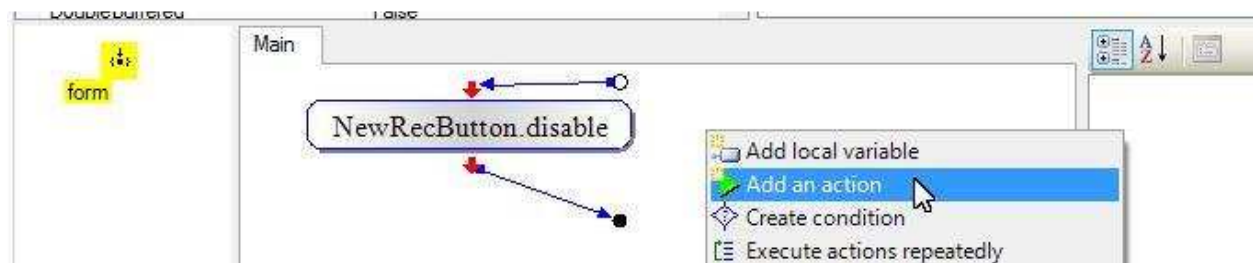


Note that the "value" of the action is "False", indicating that the button will be disabled. Rename the action and click OK:

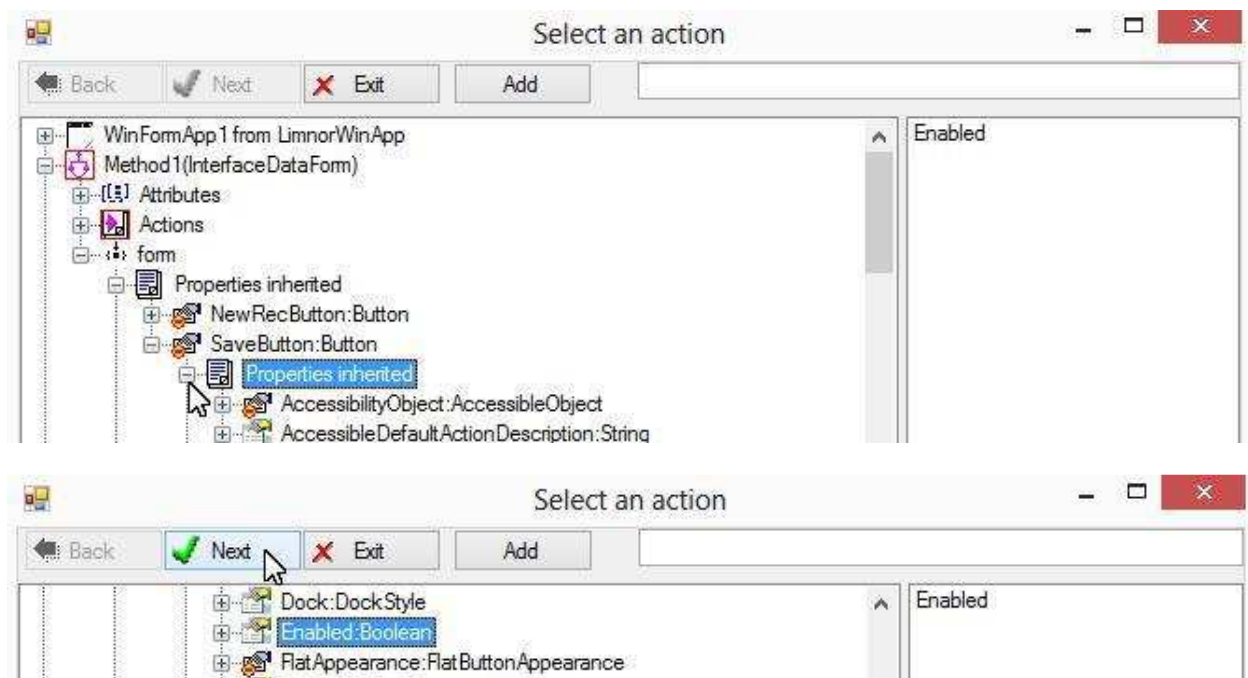
## Create and Use Interfaces



Add another action to enable the save button:



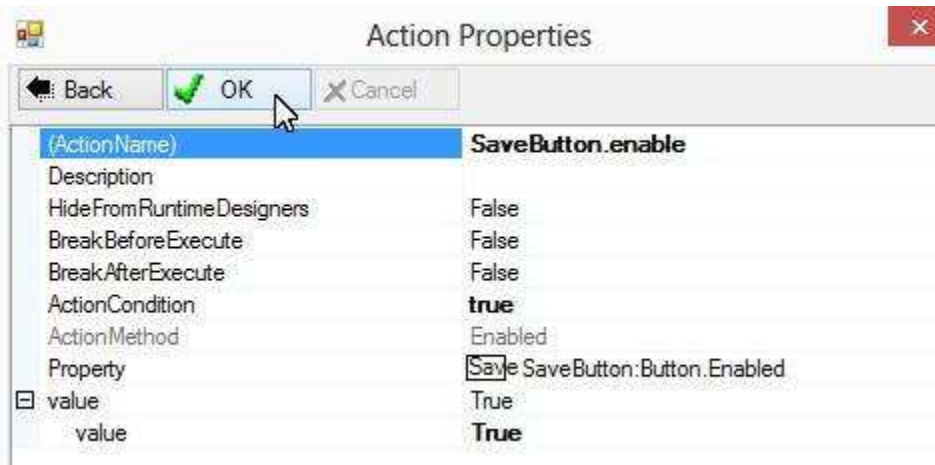
Select "Enabled" property of the save button of the "form" parameter:



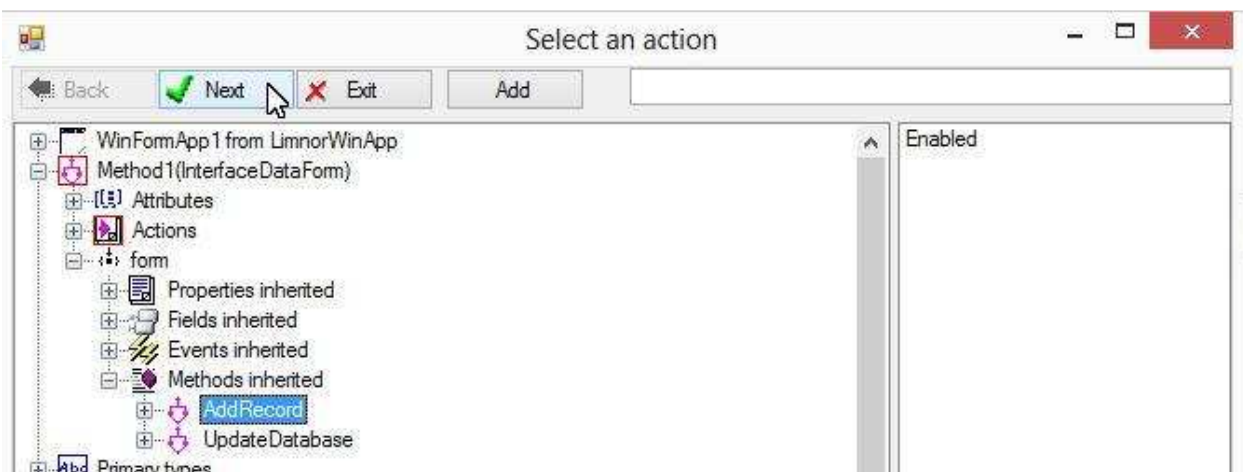
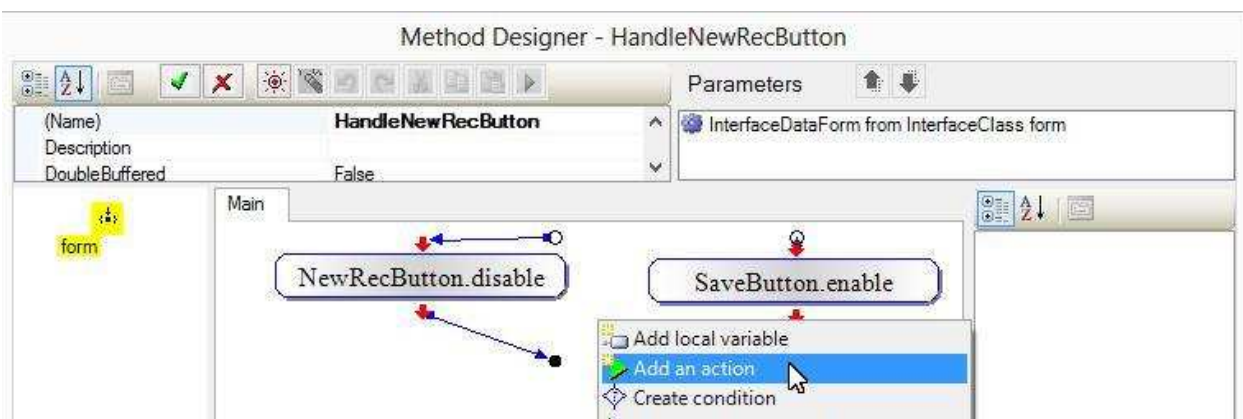
Set "value" of the action to True to enable the button. Rename the action and click OK:



## Create and Use Interfaces

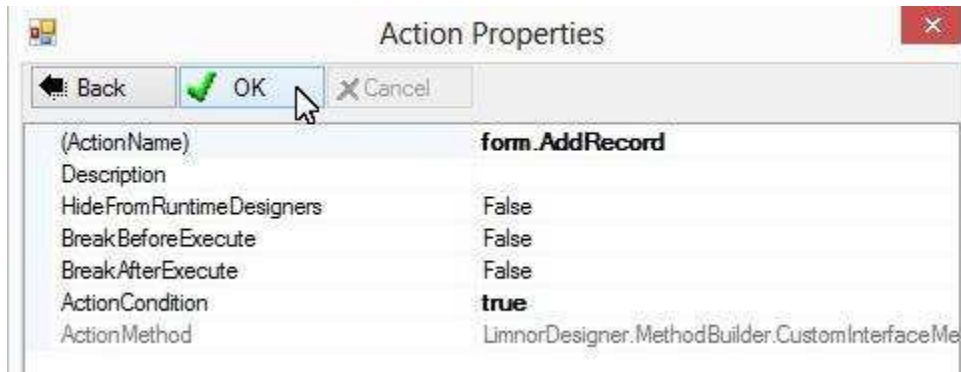


Add an action to execute AddRecord method of "form":

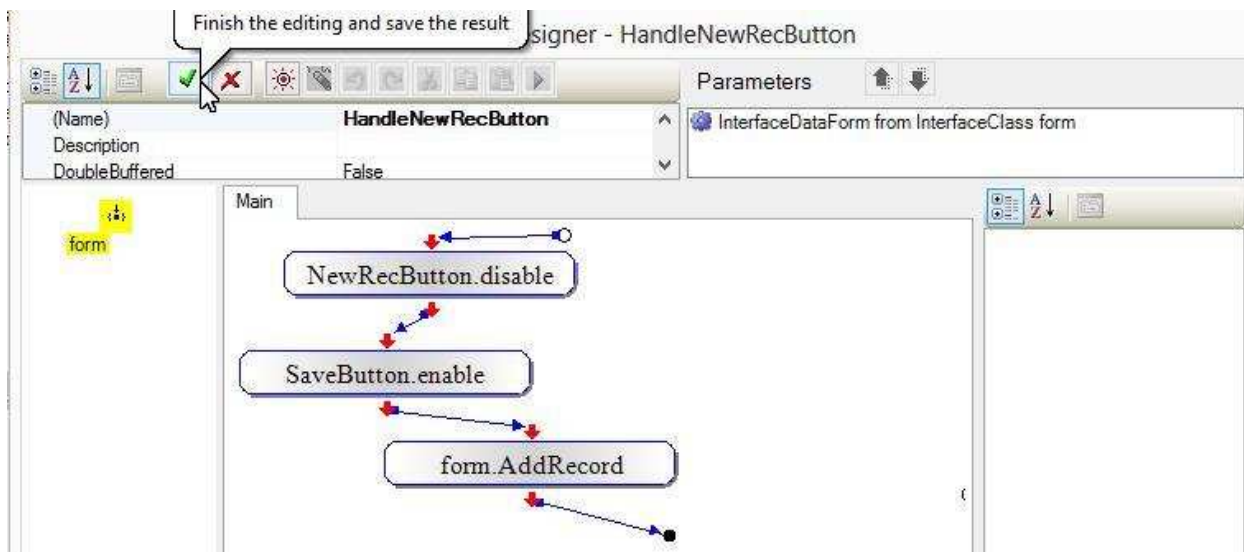




## Create and Use Interfaces

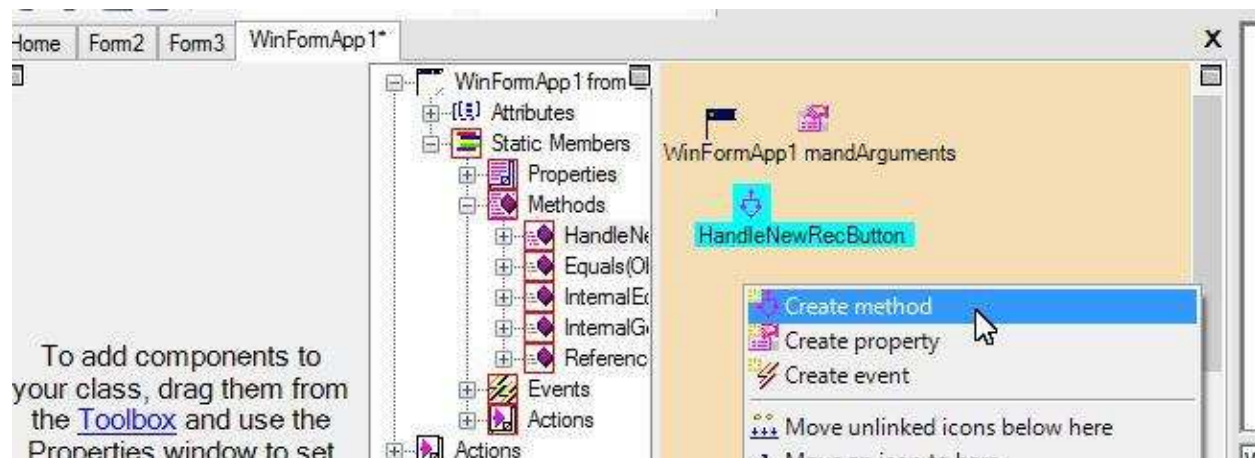


Link all actions together:



### Button handler for Save

As a sample, we create a static method for handling “Save” button. Suppose we create this method in the WinFormApp1 class:

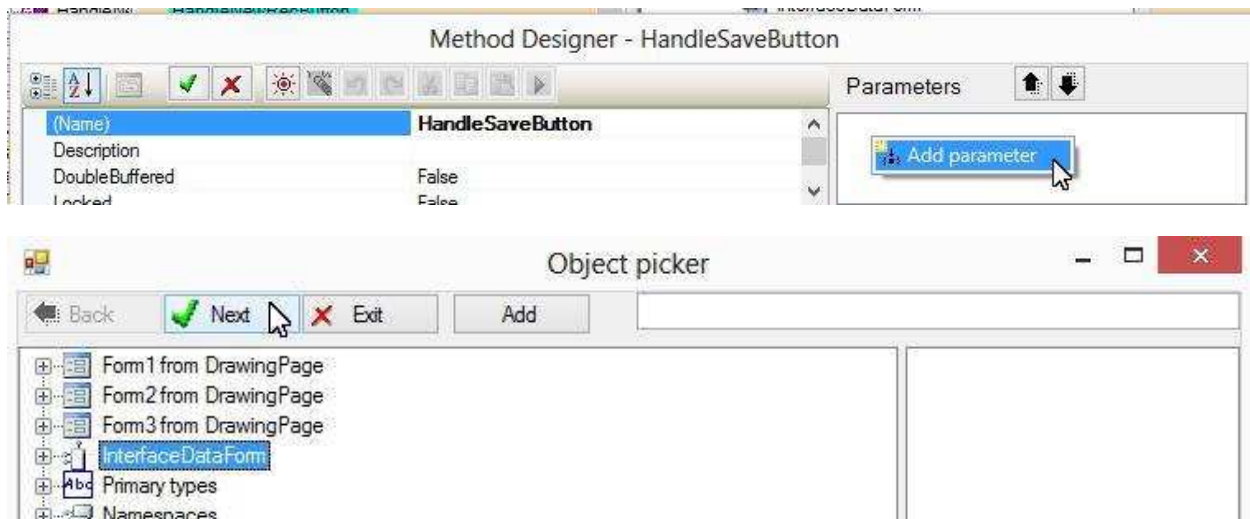


Rename the new method to HandleSaveButton:

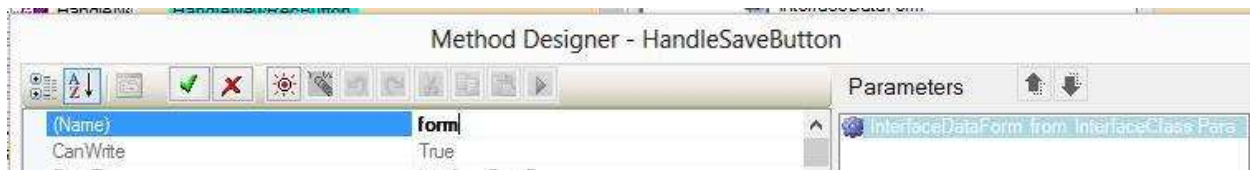
## Create and Use Interfaces



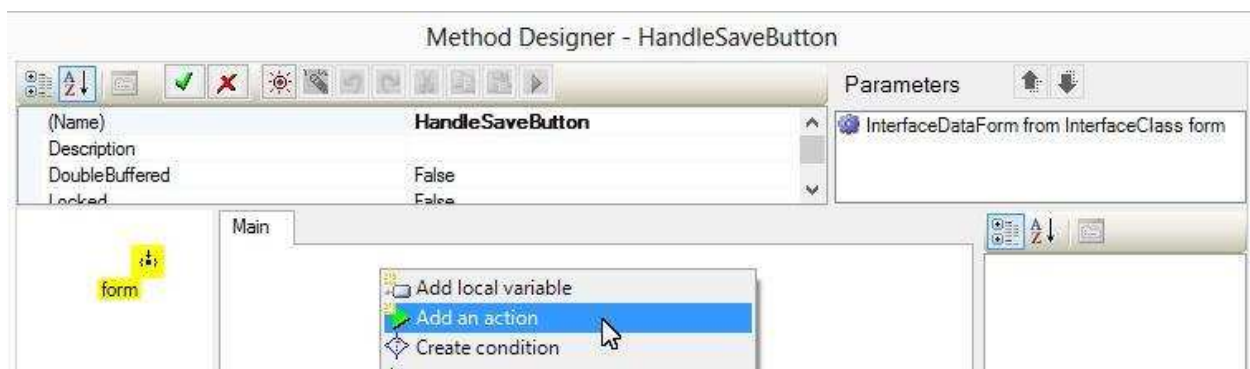
Add an InterfaceDataForm parameter:



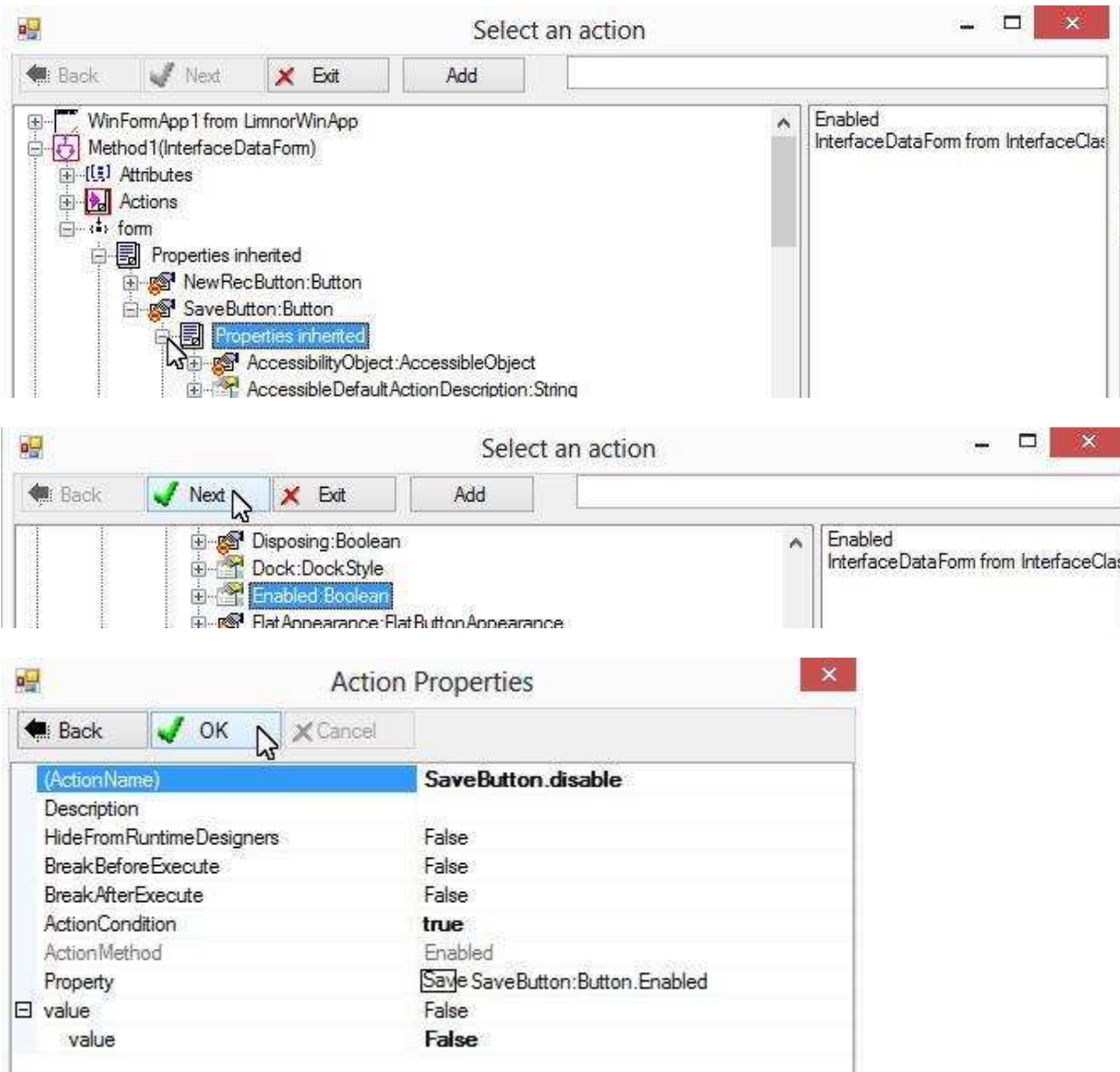
Rename the parameter to “form”:



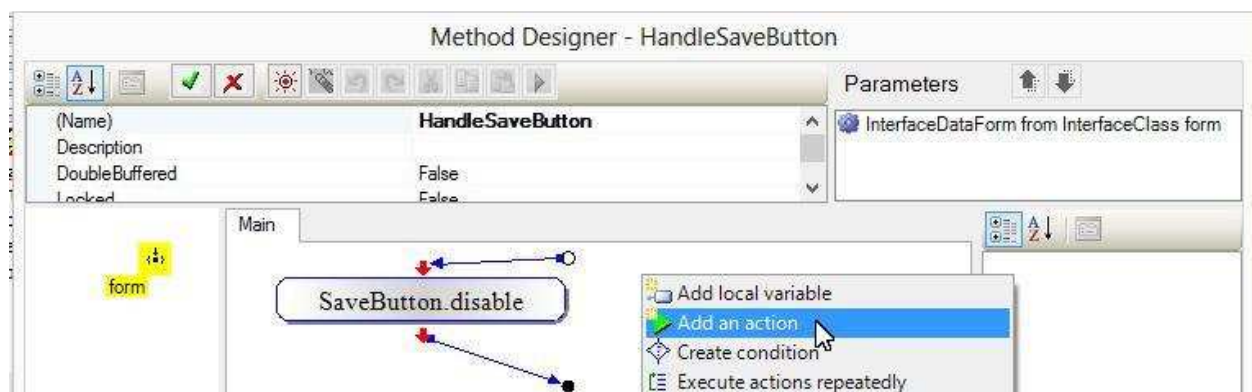
Add an action to disable the save button of “form”:



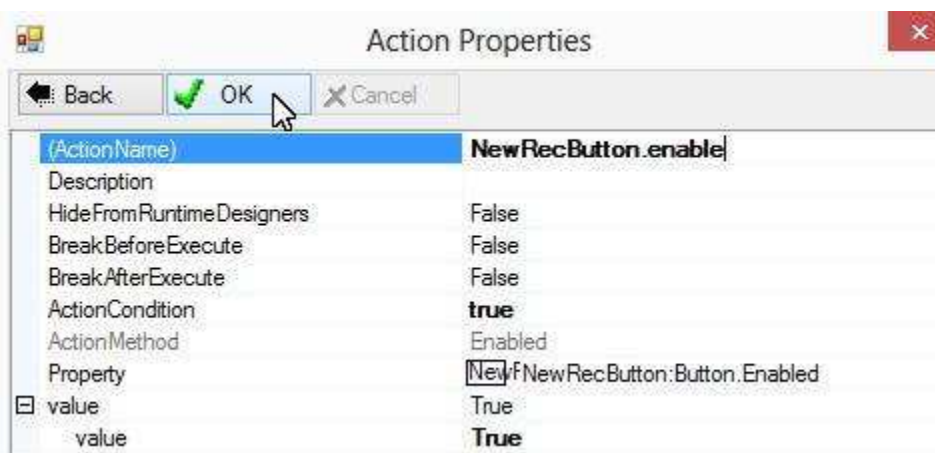
## Create and Use Interfaces



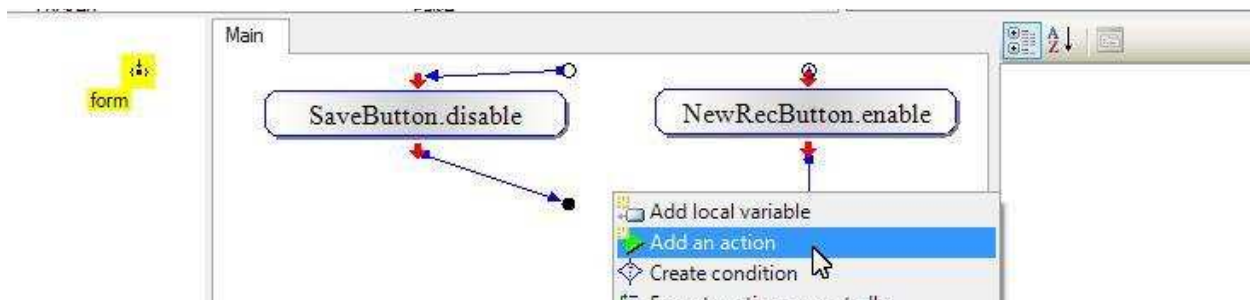
Add an action to enable “New record” button of “form”:



## Create and Use Interfaces

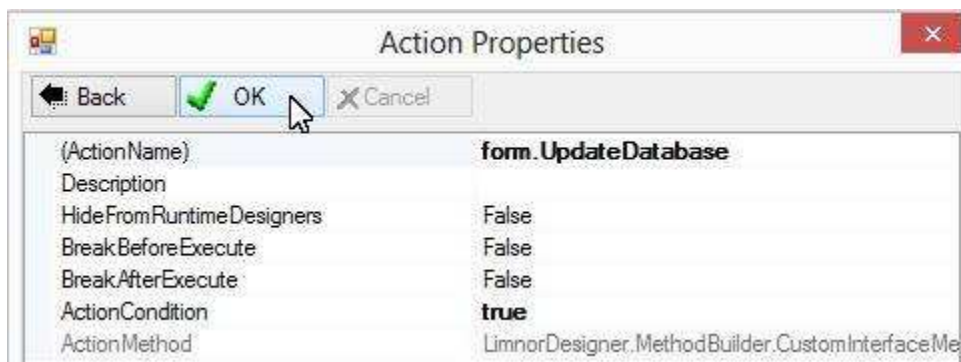
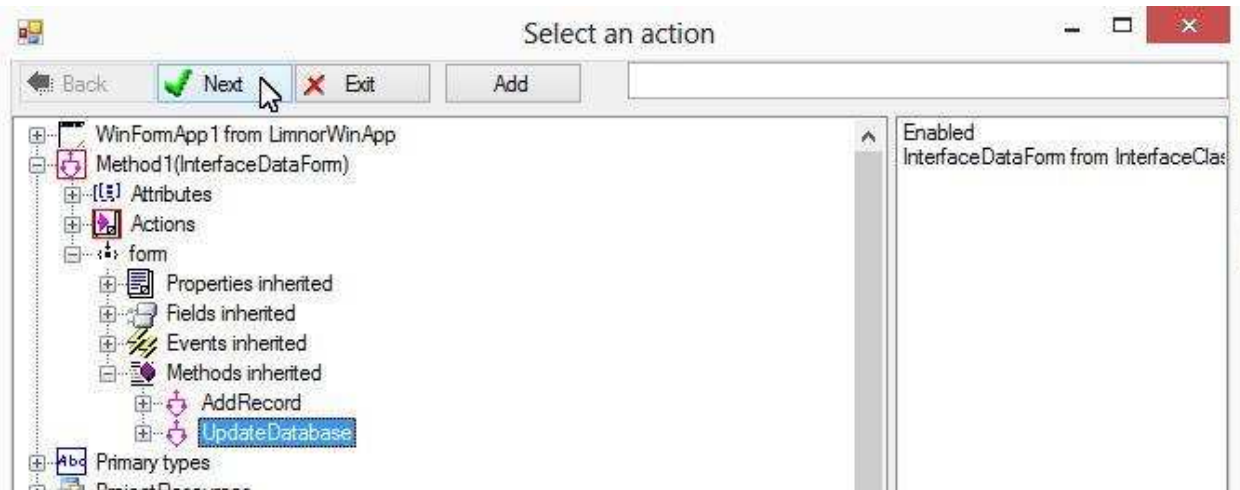


Add an action to execute UpdateDatabase method of “form”:

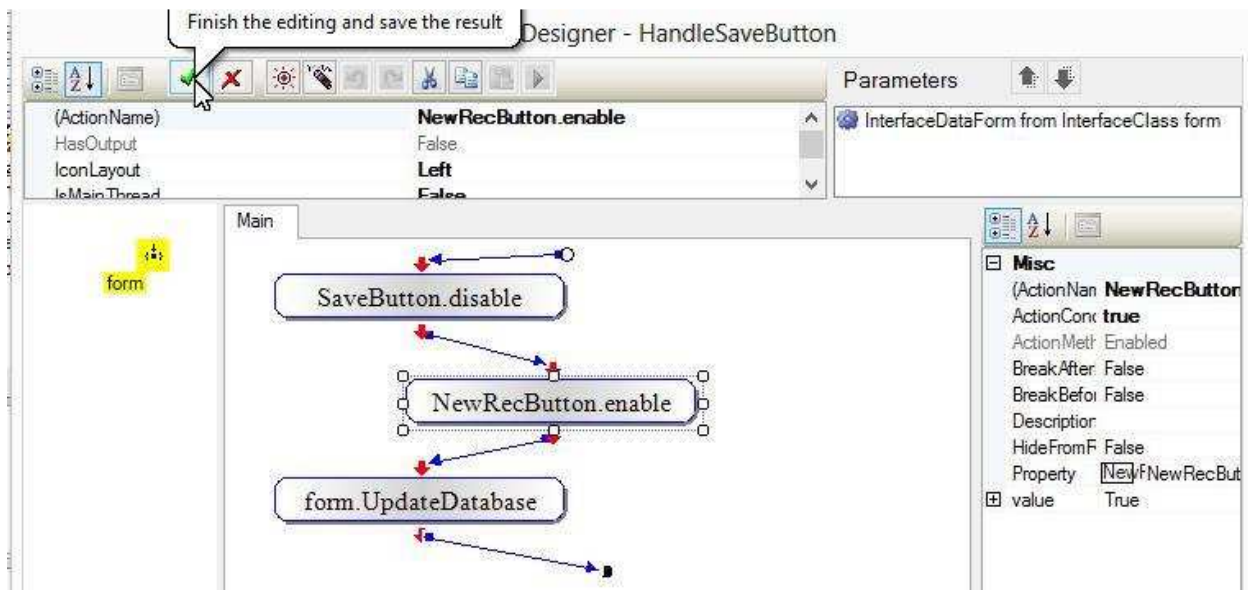




## Create and Use Interfaces



Link all actions together:



### Passing interface

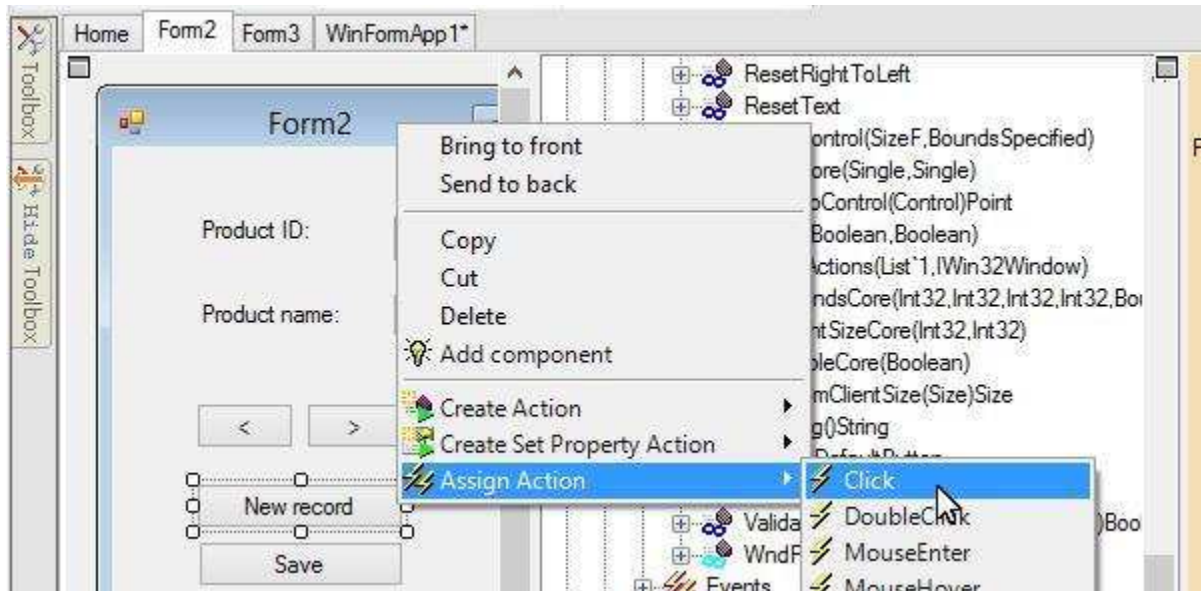
In Form2 and Form3, when “new record” button is clicked we want to execute method HandleNewRecButton, when “save” button is clicked we want to execute method HandleSaveButton.



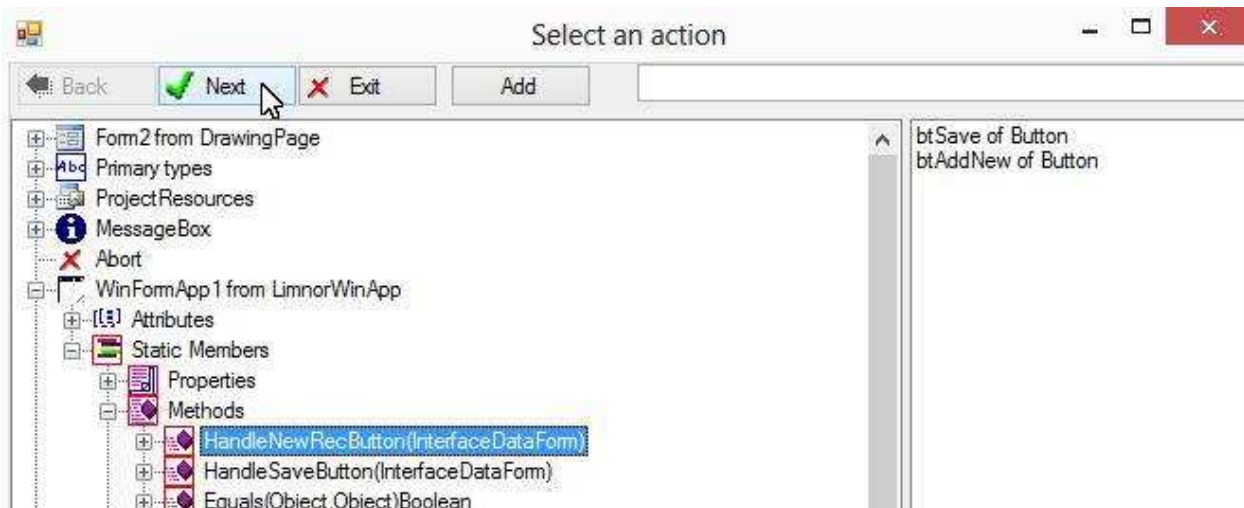
## Create and Use Interfaces

### Passing Form2

Right-click “New record” button; choose “Assign action”; choose Click:

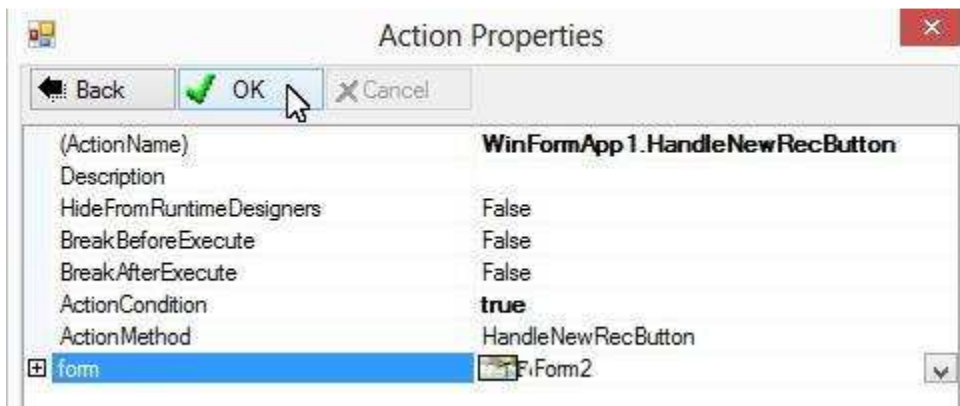
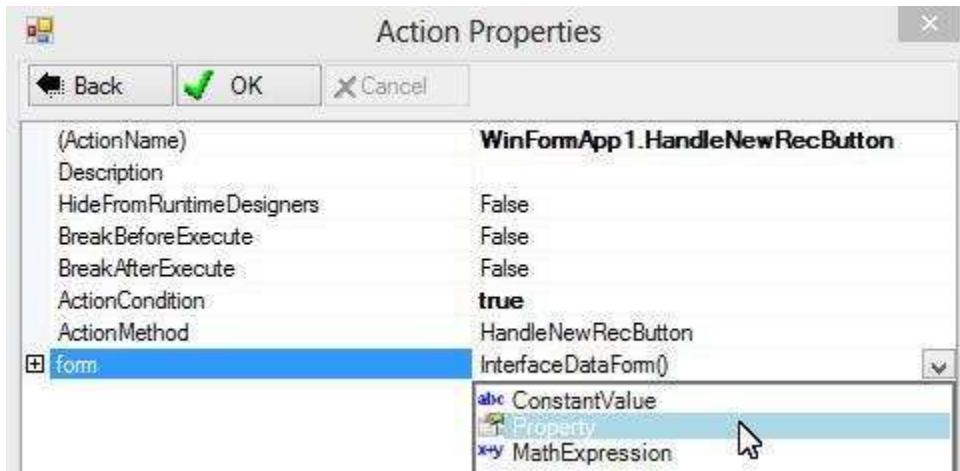


Select HandleNewRecButton:

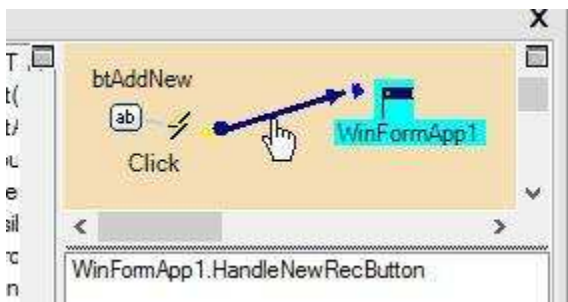


For “form” parameter, select Form2:

## Create and Use Interfaces

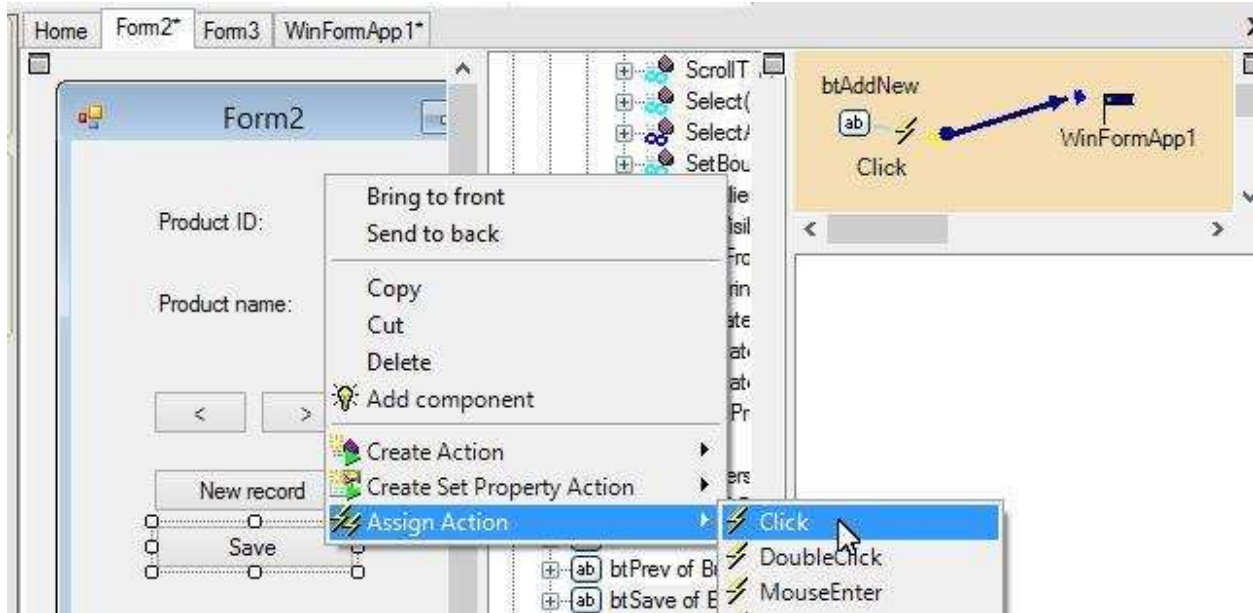


The action is created and assigned to the button:

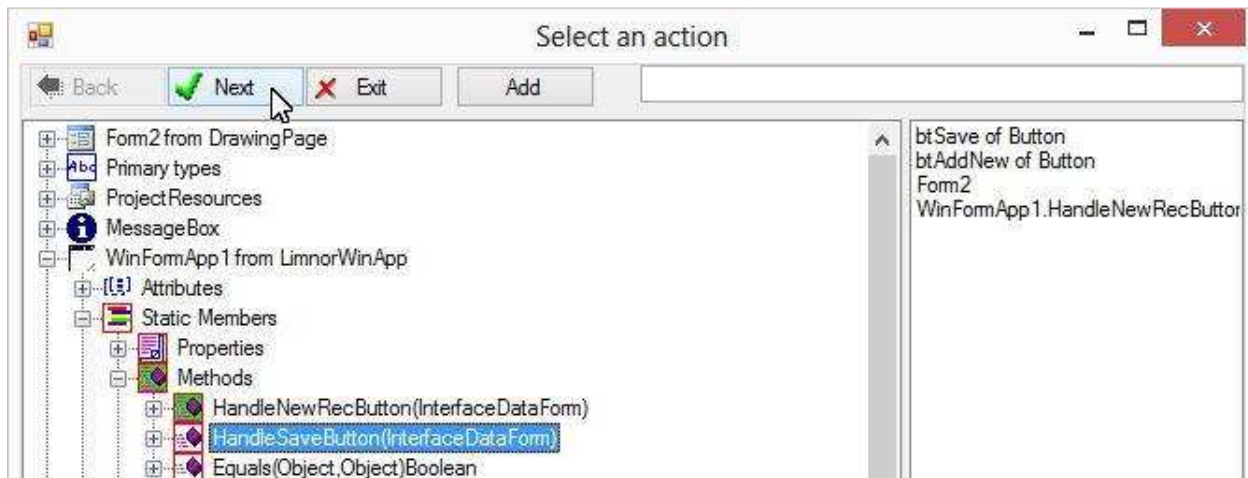


Right-click "Save" button; choose "Assign action"; choose Click:

## Create and Use Interfaces

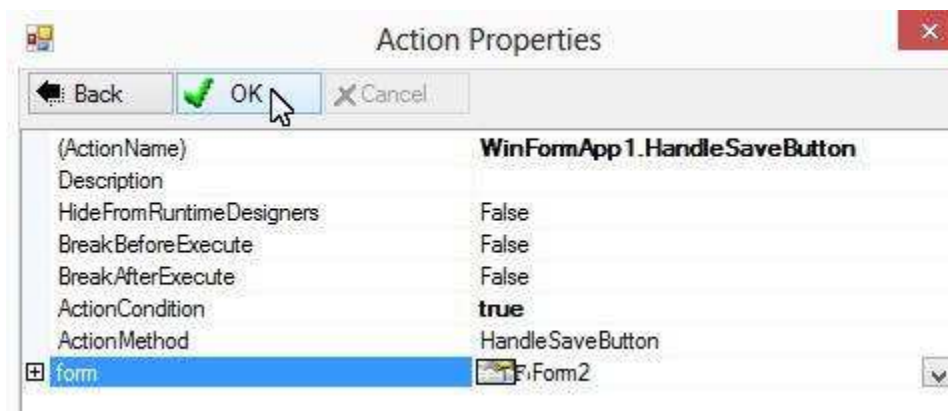
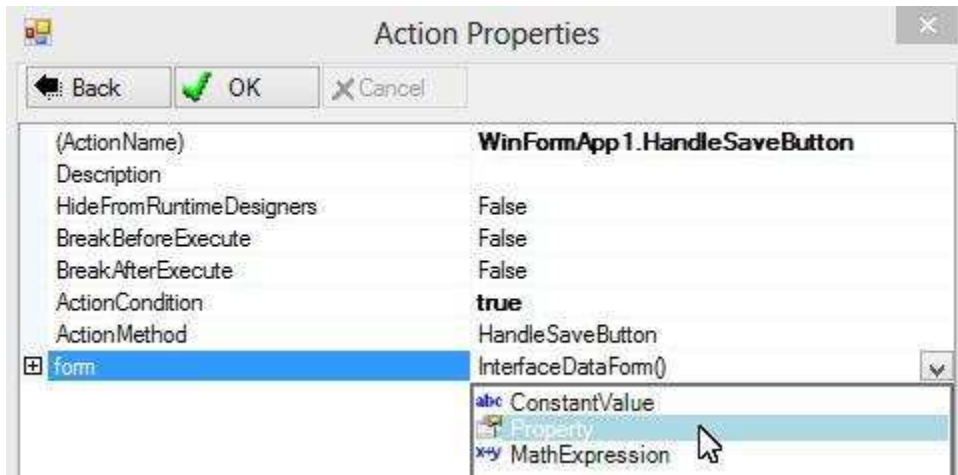


Select method HandleSaveButton:



Pass Form2 to the action via parameter "form":

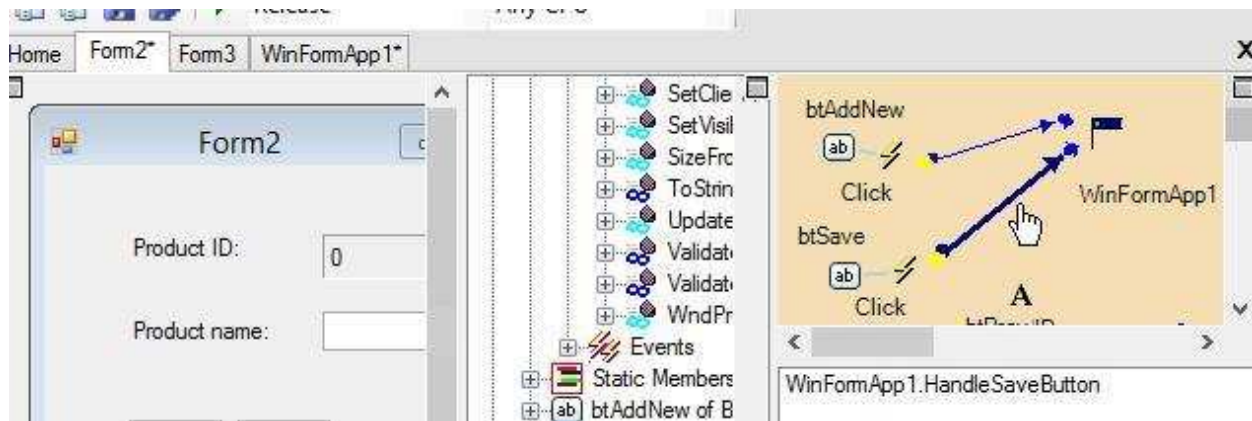
## Create and Use Interfaces



The action is created and assigned to the button:

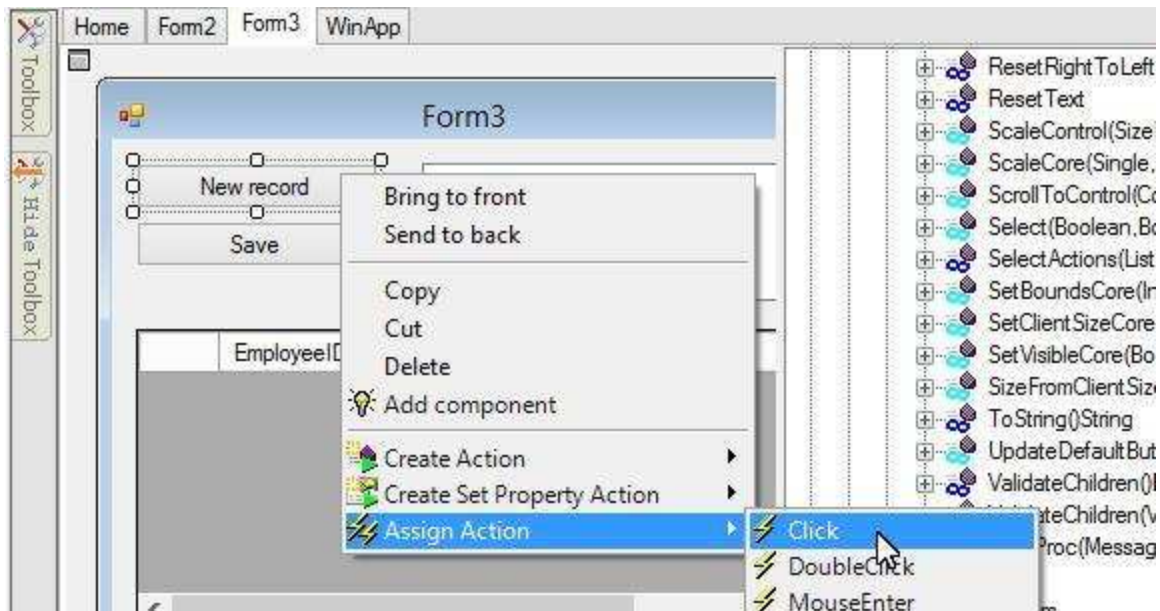


## Create and Use Interfaces



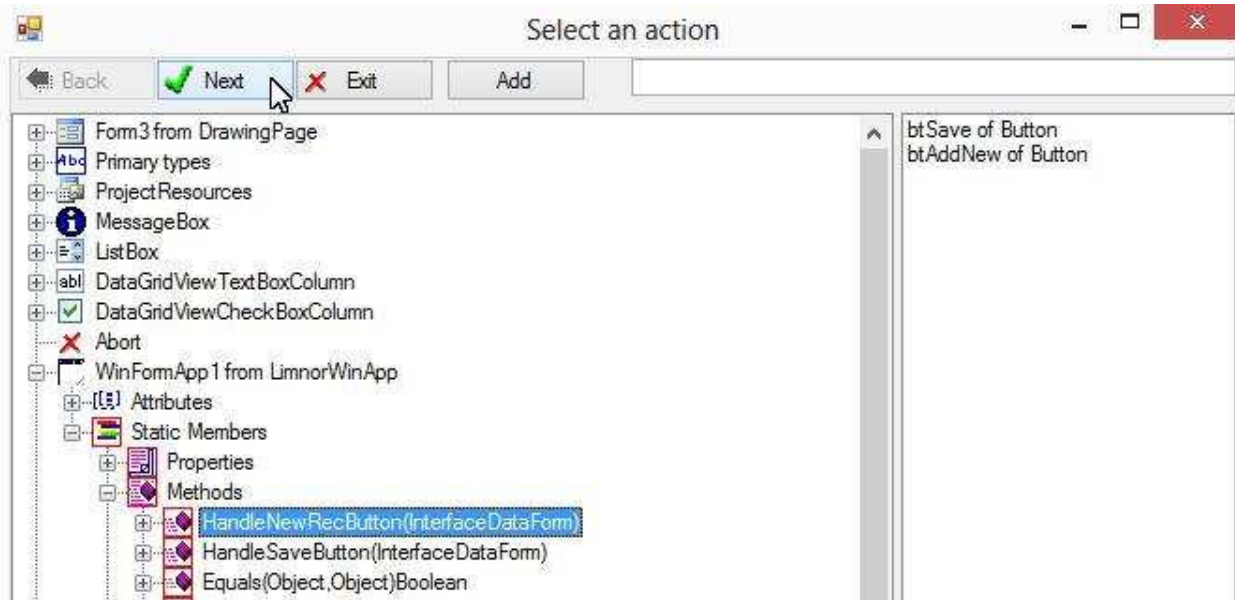
### Passing Form3

Right-click "New record" button; choose "Assign action"; choose Click:

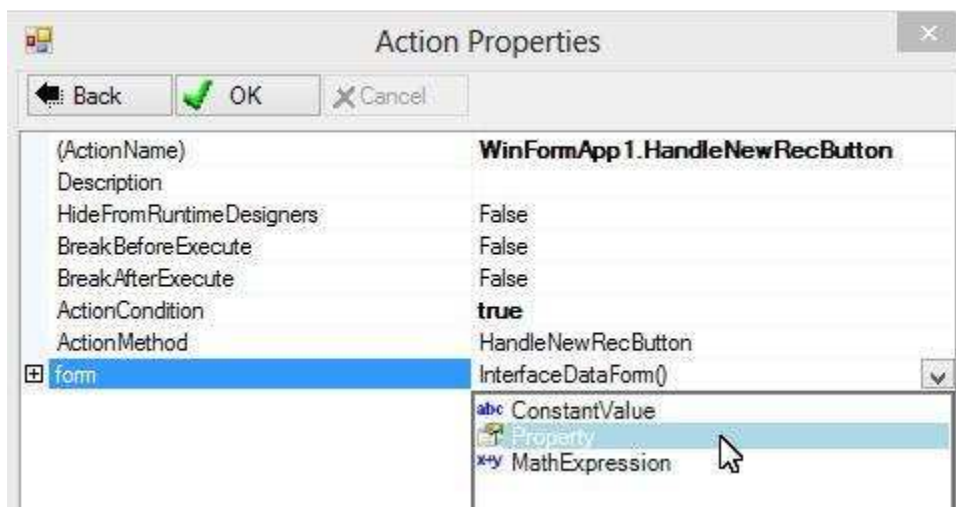


Select method HandleNewRecButton:

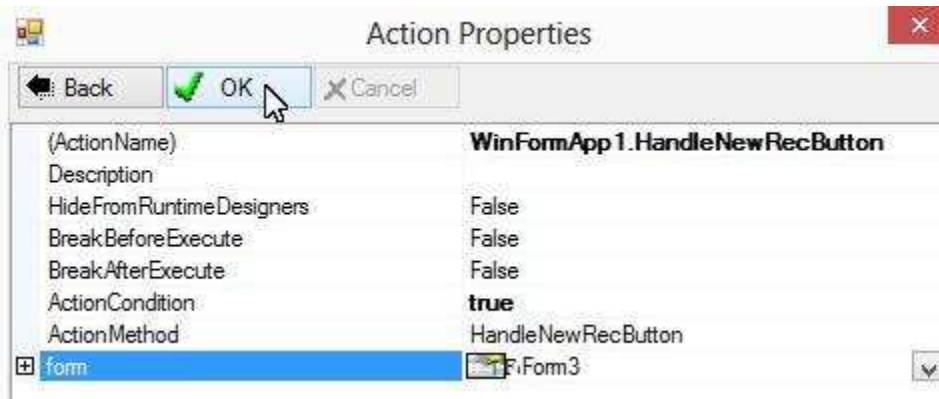
## Create and Use Interfaces



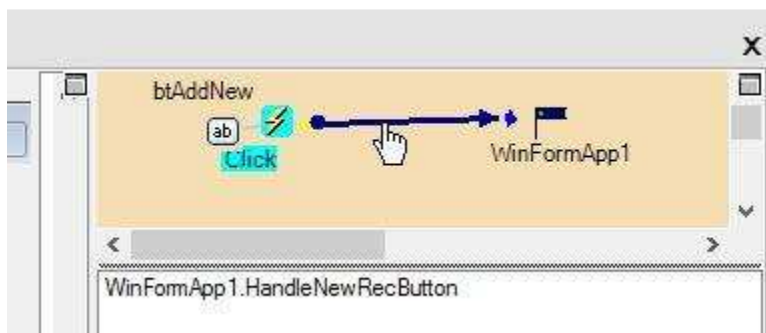
Passing Form3 to the action via parameter "form":



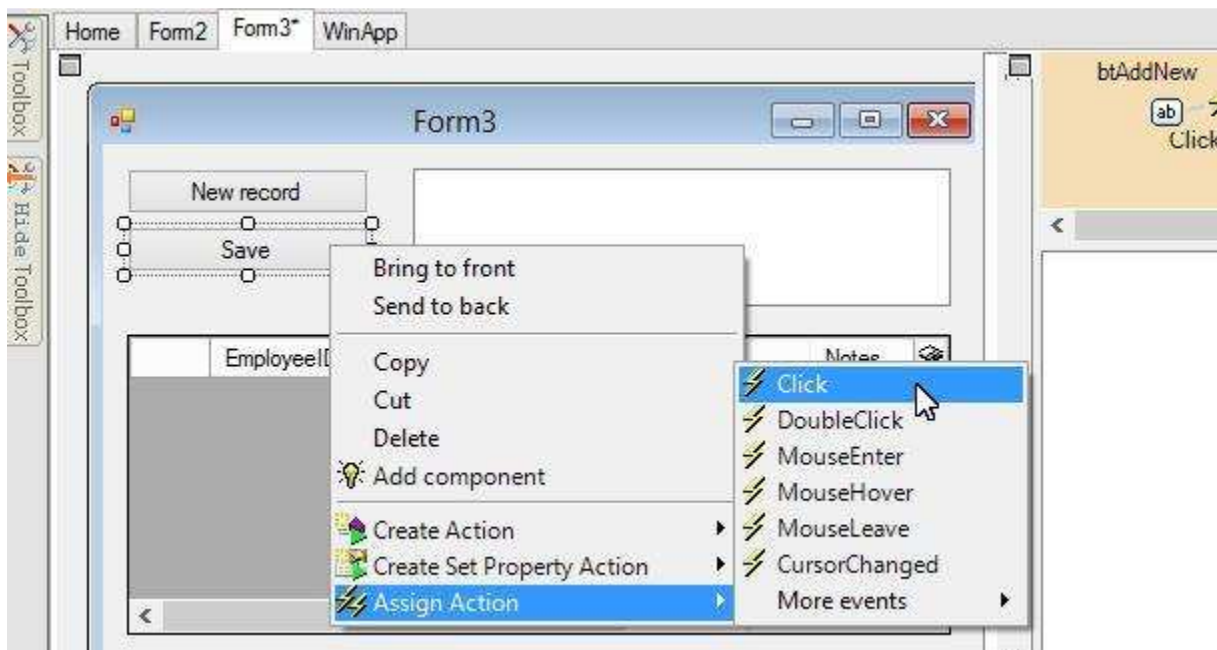
## Create and Use Interfaces



The action is created and assigned to the button:

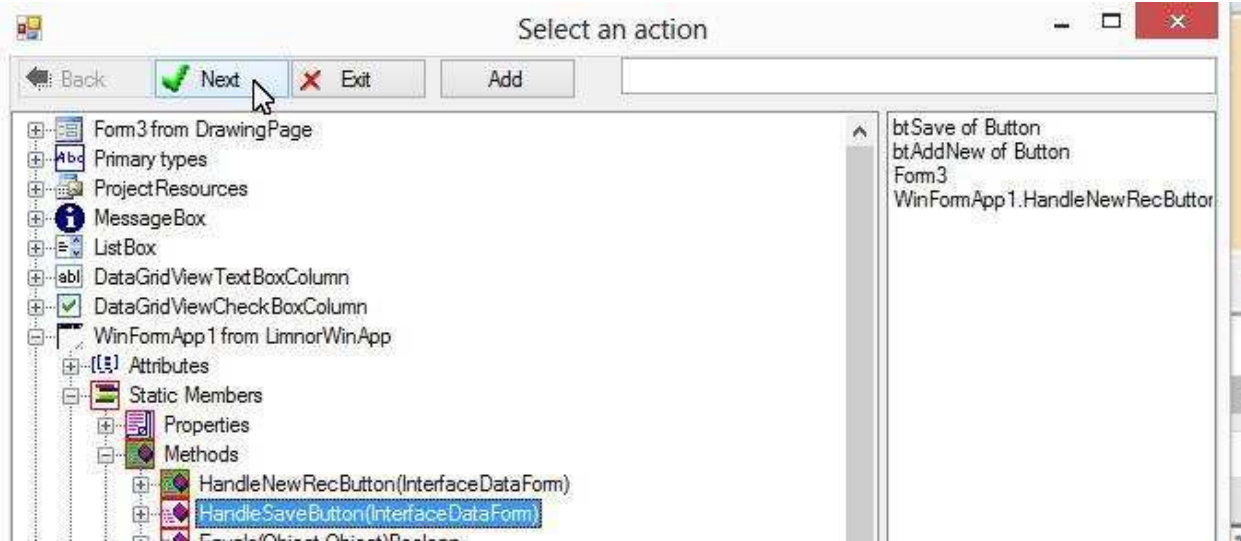


Right-click "Save" button; choose "Assign action"; choose Click event:

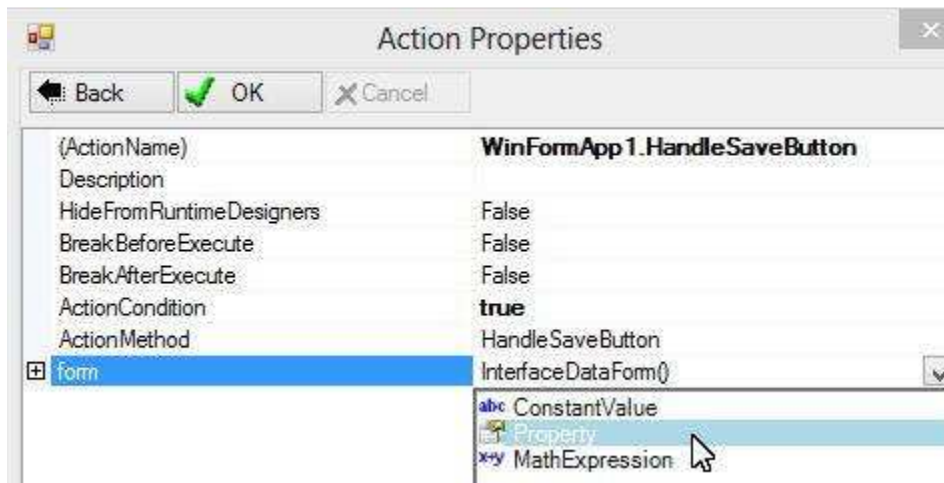


Select method HandleSaveButton:

## Create and Use Interfaces

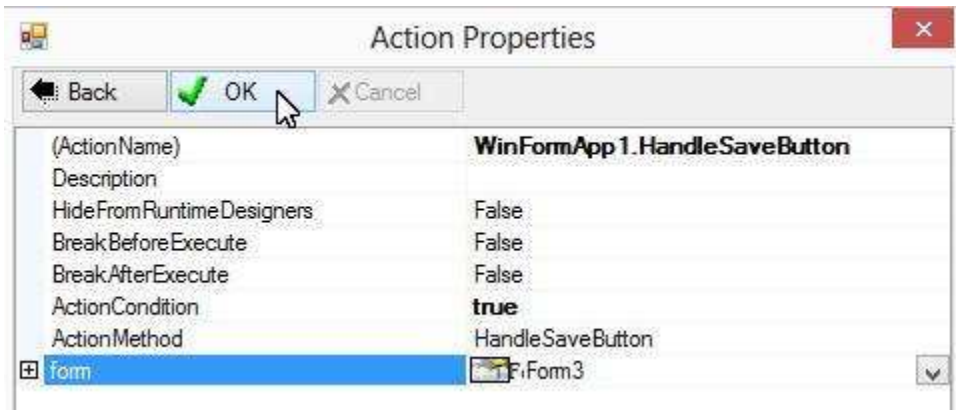


Pass Form3 to the action via parameter “form”:

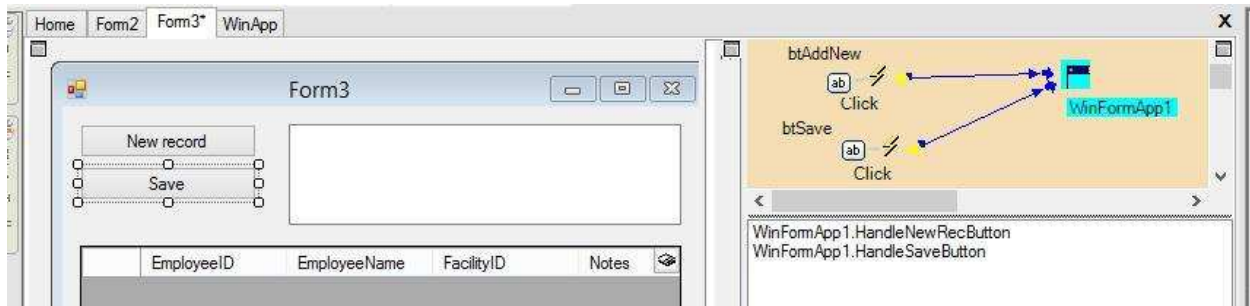




## Create and Use Interfaces

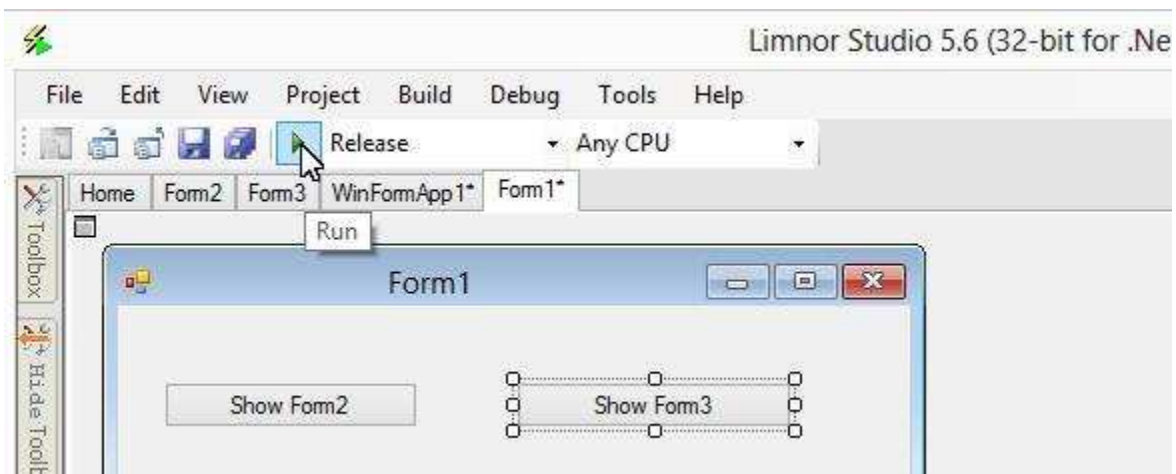


The action is created and assigned to the button:



### Test

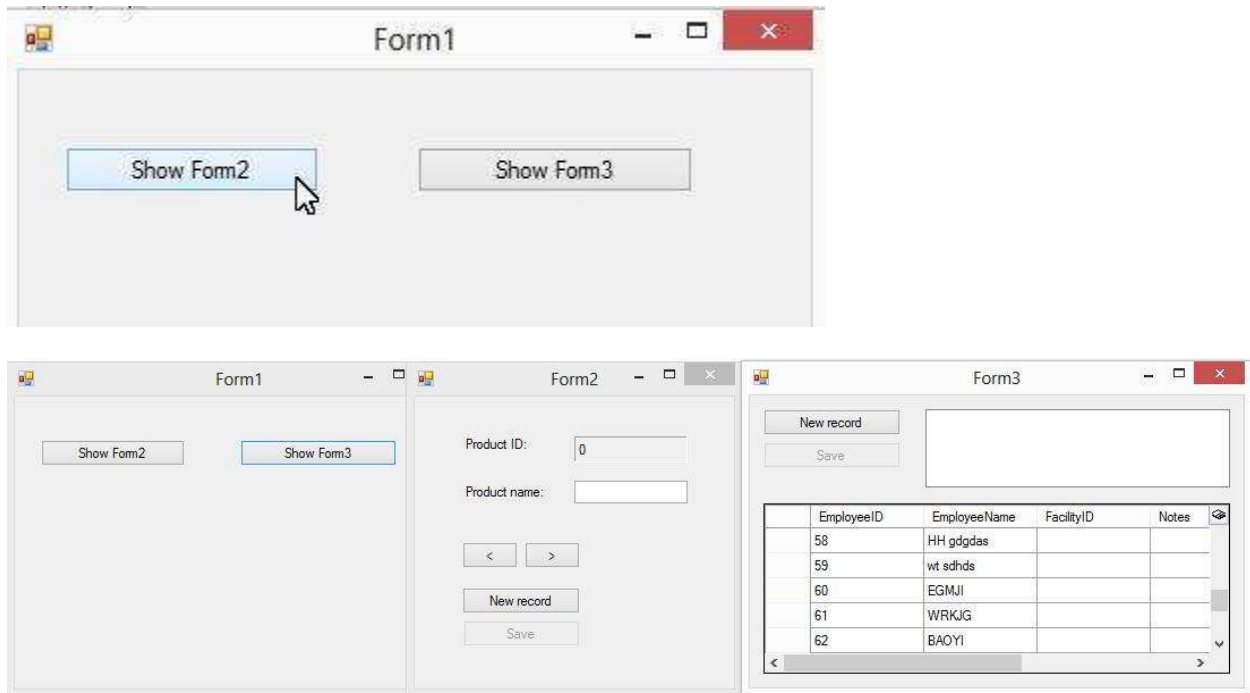
Suppose we have a Form1 to show Form2 and Form3. Form1 is the Start Form. Run the project:



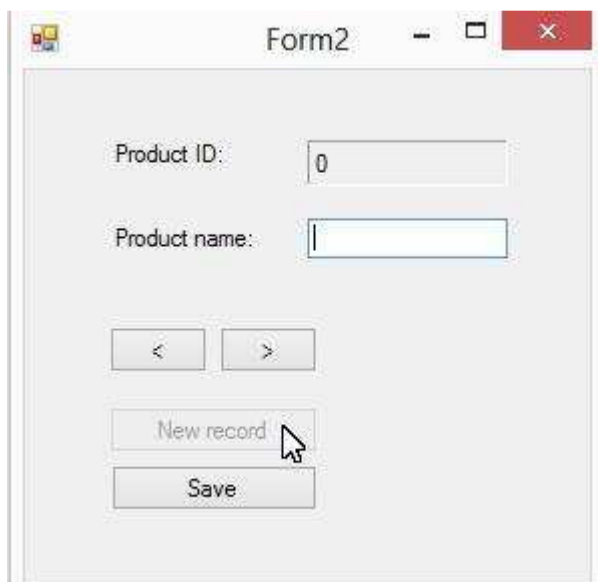
Form1 appears. Click Form2 and Form3 to show Form2 and Form3:

## Create and Use Interfaces

---



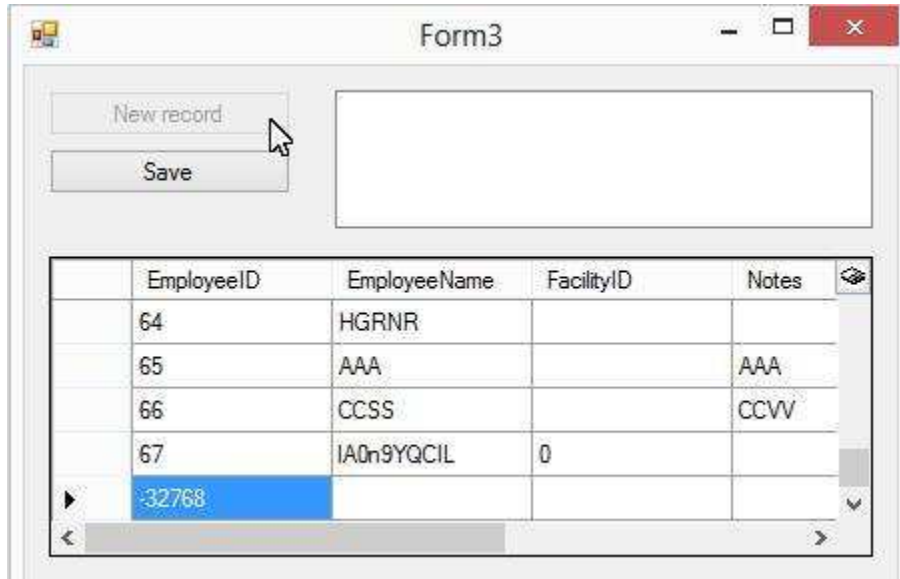
Click "New record" button on Form2. We can see that on Form2 "New record" is disabled and "Save" is enabled:



Click "New record" button on Form3. We can see that on Form3 "New record" is disabled and "Save" is enabled:

## Create and Use Interfaces

---



EmployeeID	EmployeeName	FacilityID	Notes	
64	HGRNR			
65	AAA		AAA	
66	CCSS		CCW	
67	IA0n9YQCIL	0		
32768				

We see that the same method `HandleNewRecButton` works for both `Form2` and `Form3`.

Creating and using interfaces require planning and some work. The work reduced by sharing programming of `HandleNewRecButton` may not justify the efforts. It is an example to show some basic concepts.

### Feedback

Please send your feedback to [support@limnor.com](mailto:support@limnor.com)