Use StringBuilder

Contents

Introduction 1

Sample Scenario 1

Create method 2

- Create StringBuilder 3
- Append first string part 4
- Add a repeat action 5
- Fetch index from collection 6
- Add last string part 12
- Return with ToString 12

Execute method 14

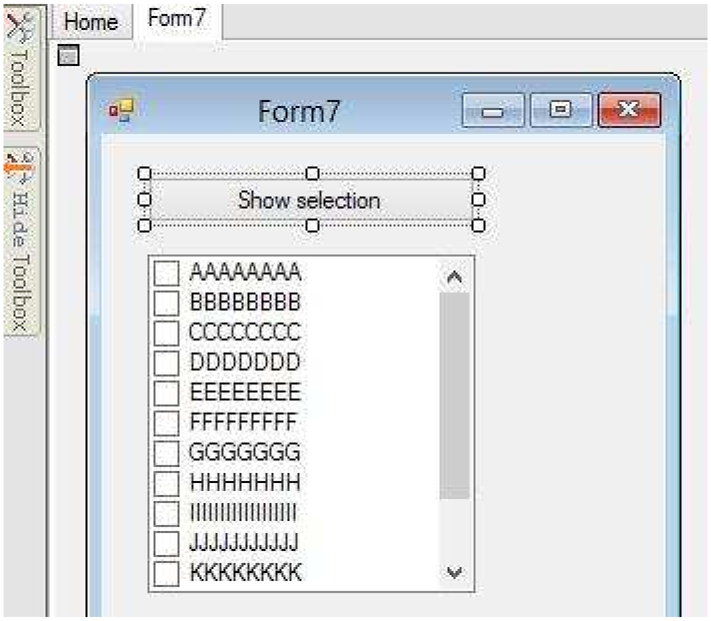
Test 18

Introduction

StringBuilder can be used to programmatically construct a string by concatenations. It has an Append method to add string parts. It has a ToString method to form a string by concatenating all strings added by Append actions. We use a sample scenario to demonstrate the use of Append and ToString.

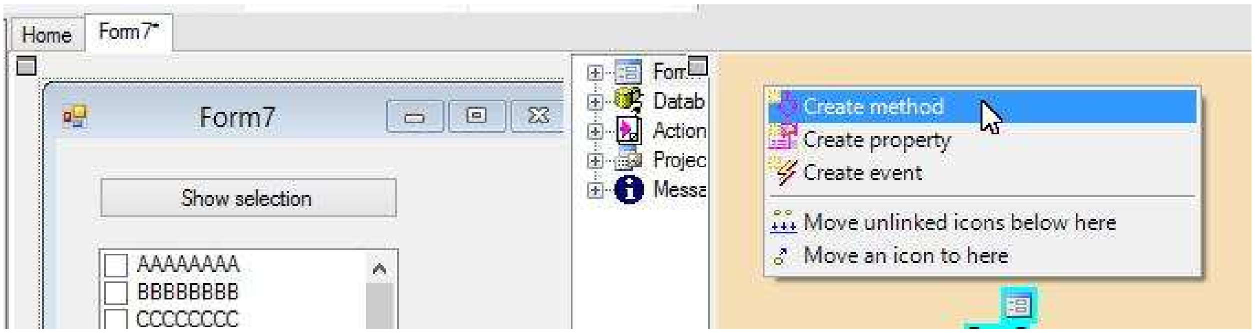
Sample Scenario

Suppose we have a CheckedListBox and a button. When the user clicks the button, a message box appears showing a message in following format: ProductID IN (n1, n2, …), where n1, n2, … are the indexes for checked items of the CheckedListBox. For example, if the first, third and fourth items are checked then the message is ProductID IN (0, 2, 3)

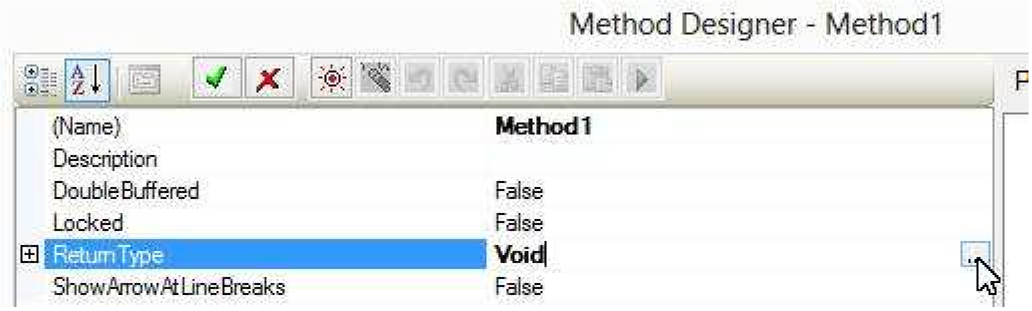


Create method

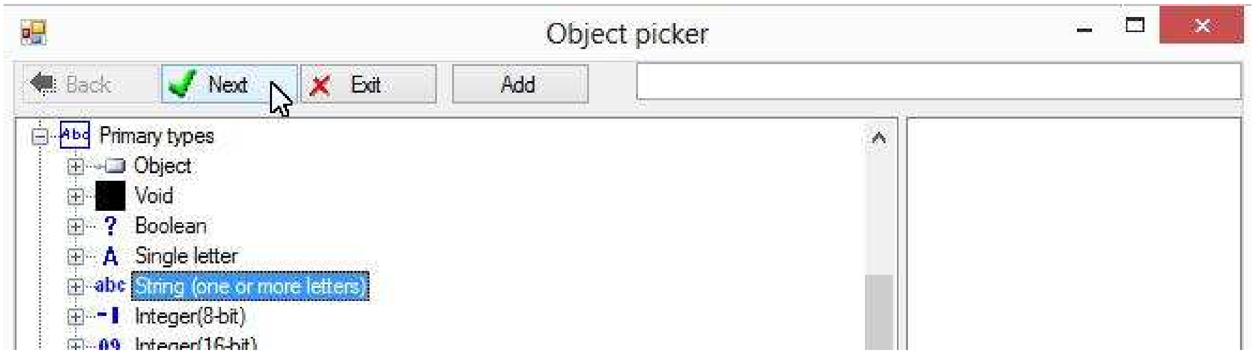
We create a method to form the message described above.



Change its return value to string:

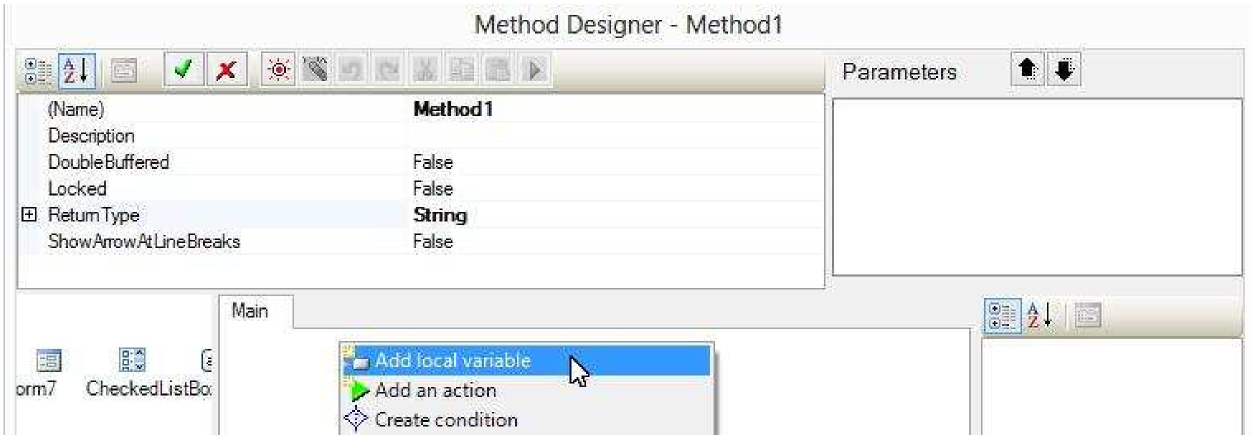


(Name)	**Method1**
Description	
DoubleBuffered	False
Locked	False
⊞ Return Type	**Void**
ShowArrowAtLineBreaks	False



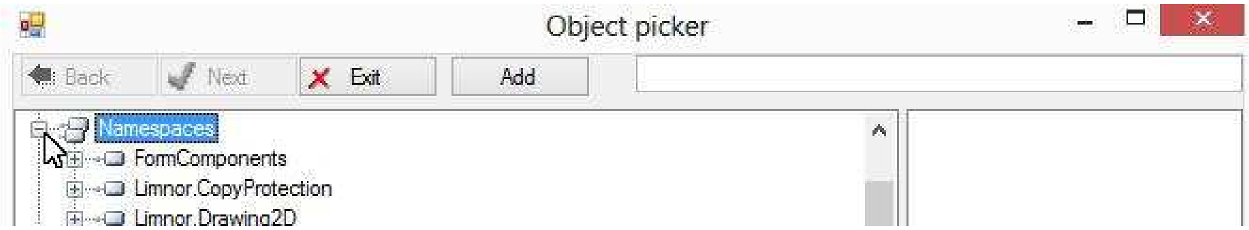
Create StringBuilder

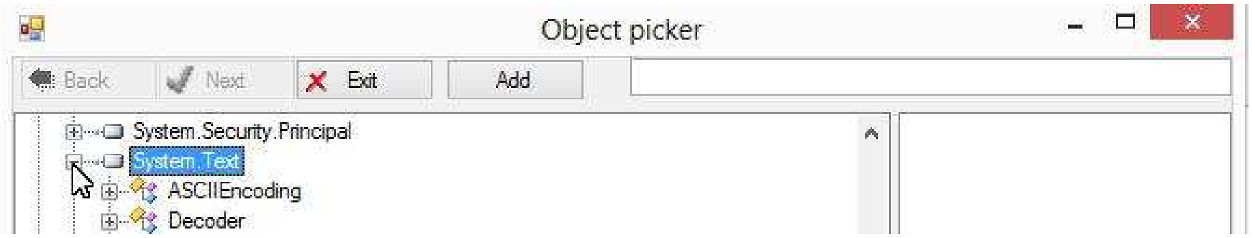
Create a StringBuilder variable.

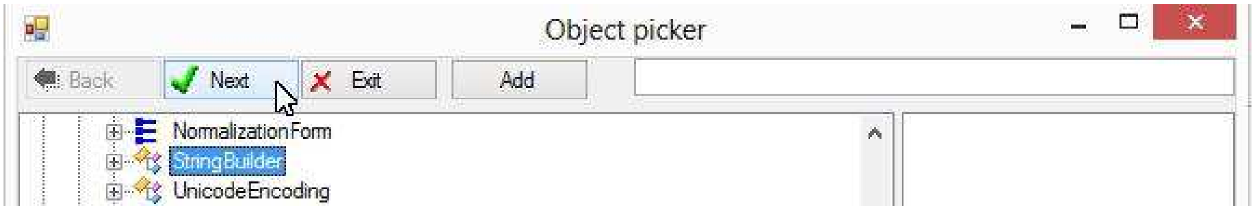


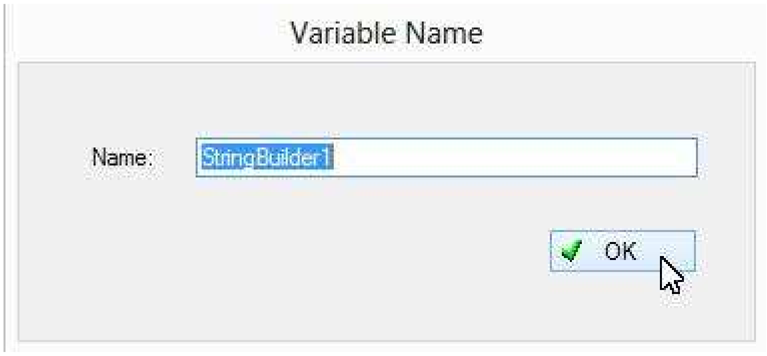
(Name)	**Method1**
Description	
DoubleBuffered	False
Locked	False
⊞ Return Type	**String**
ShowArrowAtLineBreaks	False

Select StringBuilder:

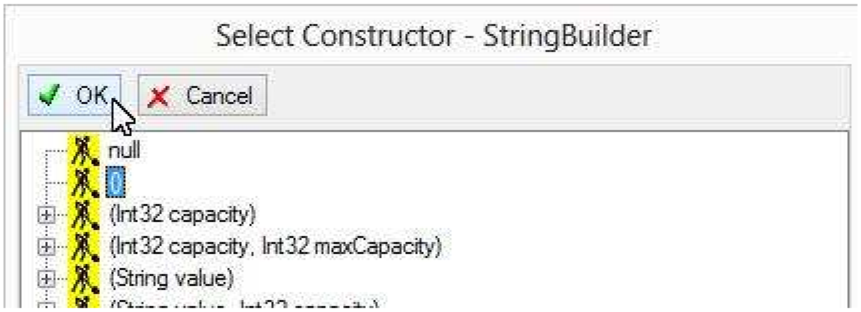




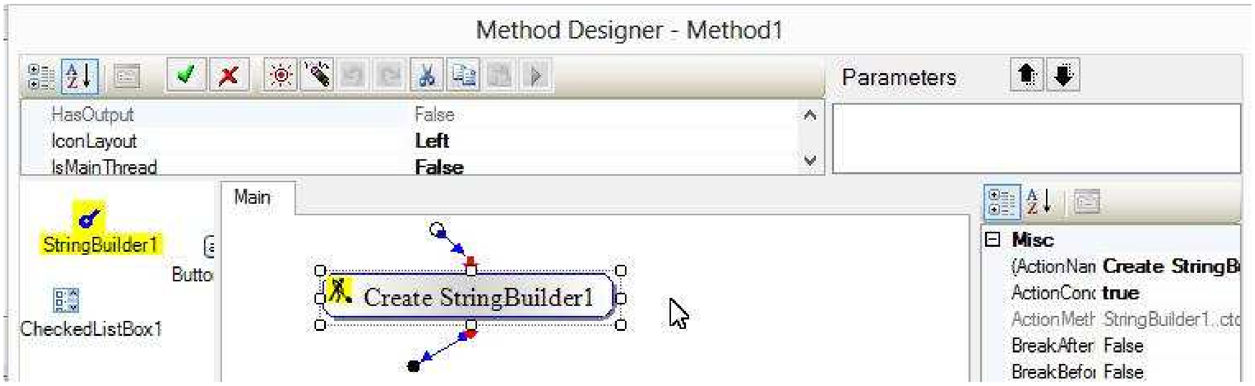




Name: StringBuilder1


Select Constructor - StringBuilder
OK
Cancel
null
()
(Int32 capacity)
(Int32 capacity, Int32 maxCapacity)
(String value)

A StringBuilder variable appears; an action for creating the variable appears.

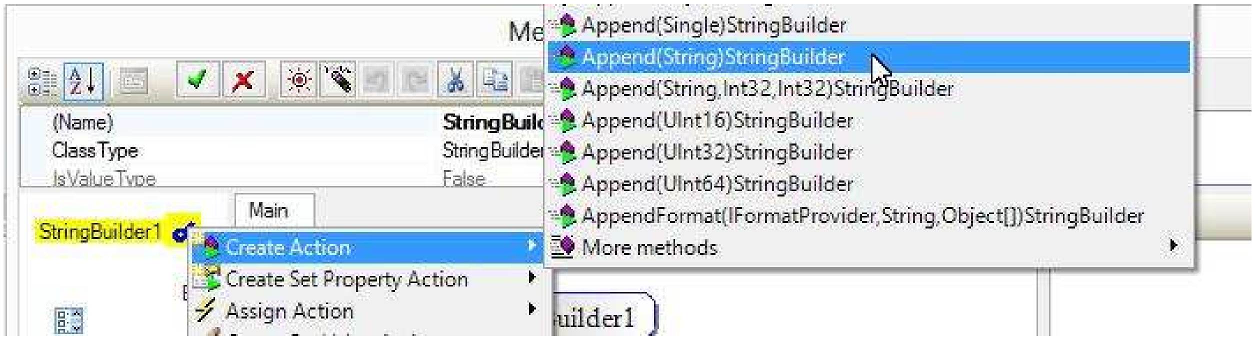


HasOutput False
IconLayout Left
IsMainThread False

(ActionNan Create StringB
ActionConc true
ActionMeth StringBuilder1..ctc
BreakAfter False
BreakBefor False

Append first string part

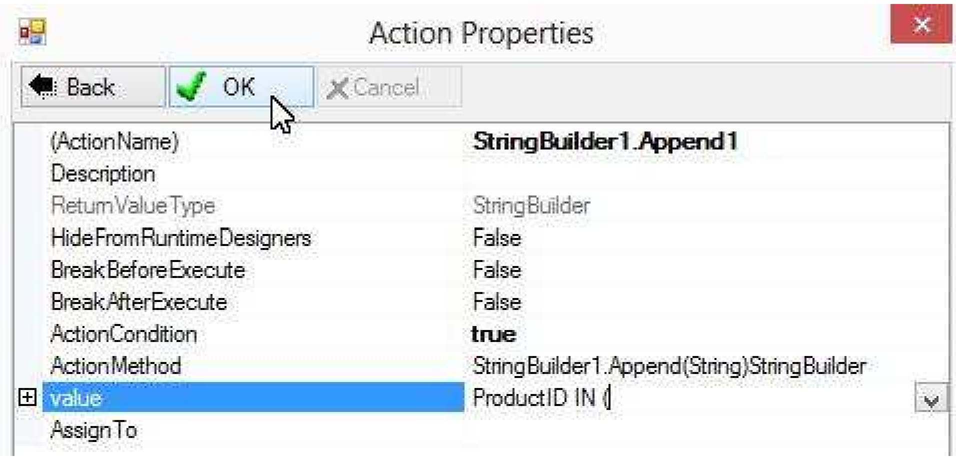
In our scenario, the first part of the string is “Product In (“. We may use an Append action to add it. You can see that there are many forms of Append method. We choose Append (String):



Me
Main
StringBuilder1
Create Action
Create Set Property Action
Assign Action
Append(Single)StringBuilder
Append(String)StringBuilder
Append(String,Int32,Int32)StringBuilder
Append(UInt16)StringBuilder
Append(UInt32)StringBuilder
Append(UInt64)StringBuilder
AppendFormat(IFormatProvider,String,Object[])StringBuilder
More methods
uilder1

(Name)	StringBuild
ClassType	StringBuilder
IsValueType	False

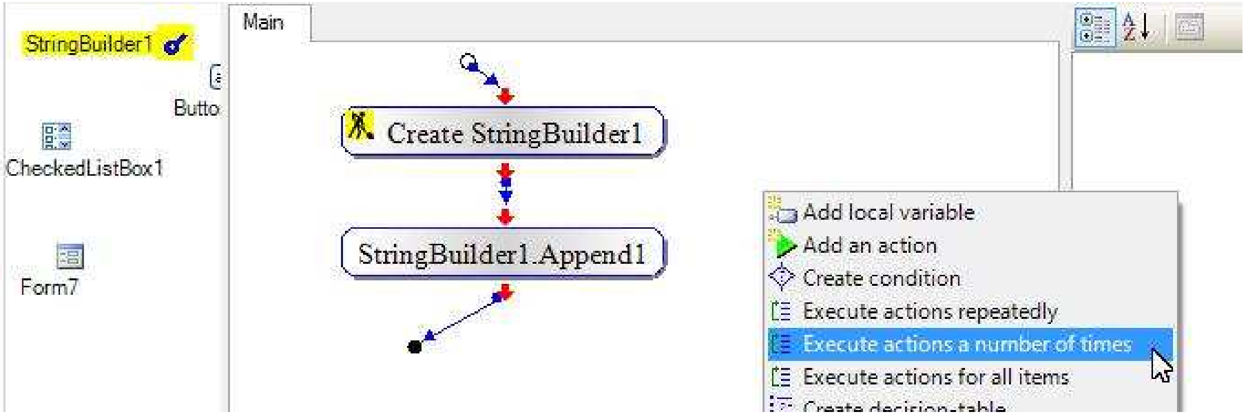
Set the first string part to “value” of the action:



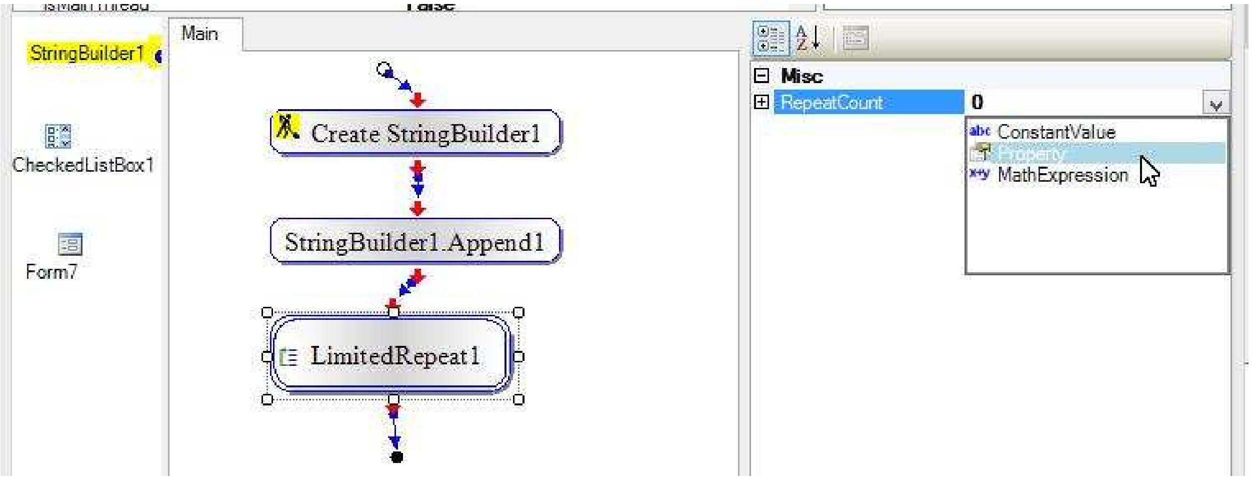
(ActionName)	**StringBuilder1.Append1**
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	StringBuilder1.Append(String)StringBuilder
⊞ value	ProductID IN (
AssignTo	

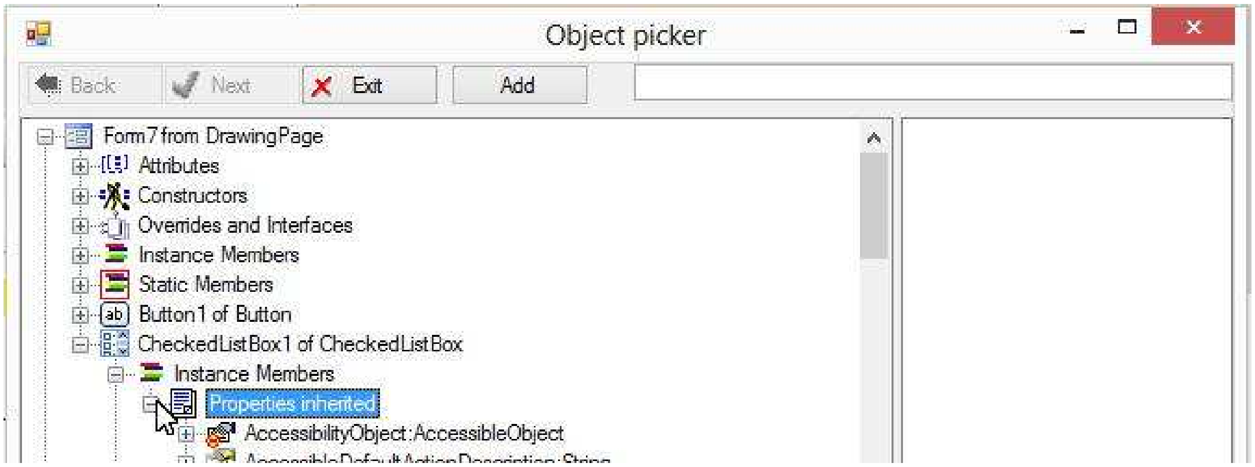
Add a repeat action

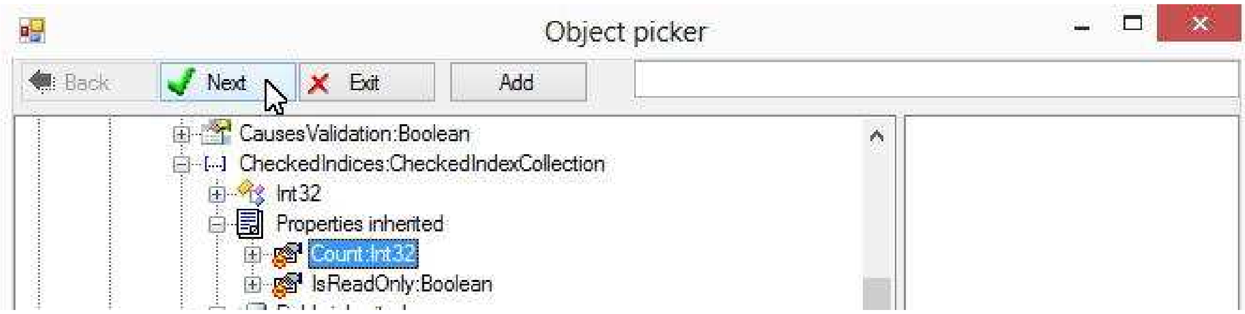
The CheckedListBox has a property, CheckedIndexes, which is a collection of the indexes of all checked items. We may use a repeat action to fetch every index from it:



Set the RepeatCount to the Count of CheckedIndexes:

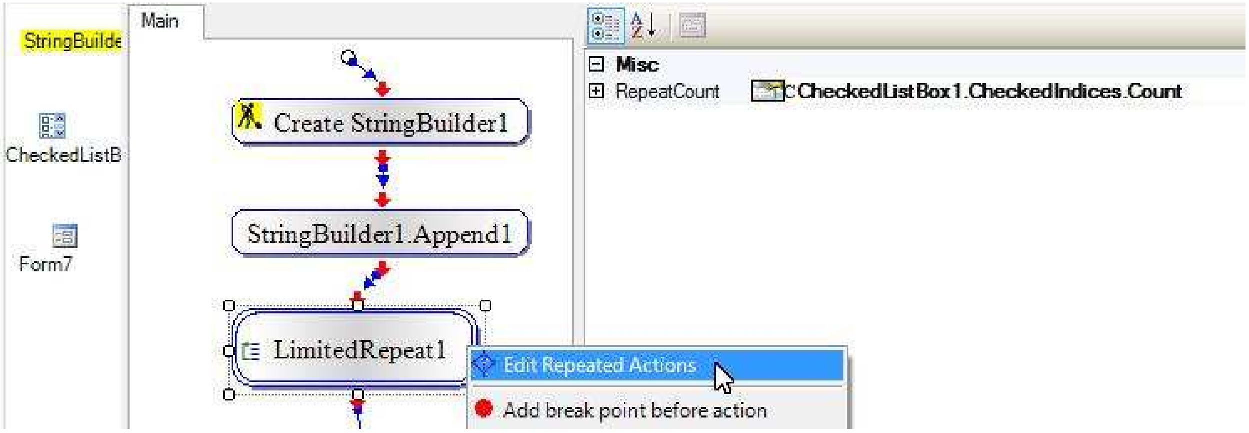






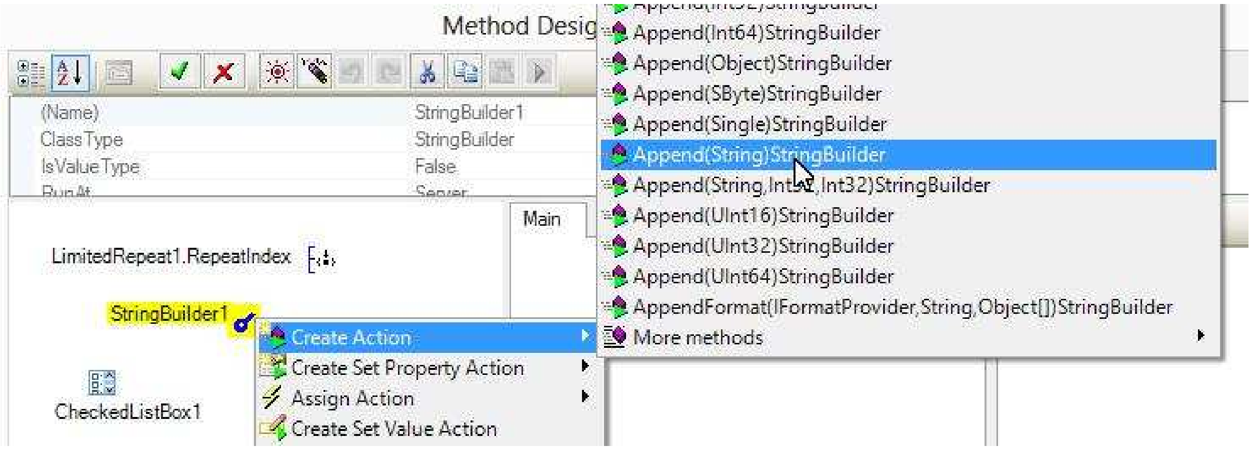
Fetch index from collection

Edit the repeat action to fetch each checked index from CheckedIndexes:



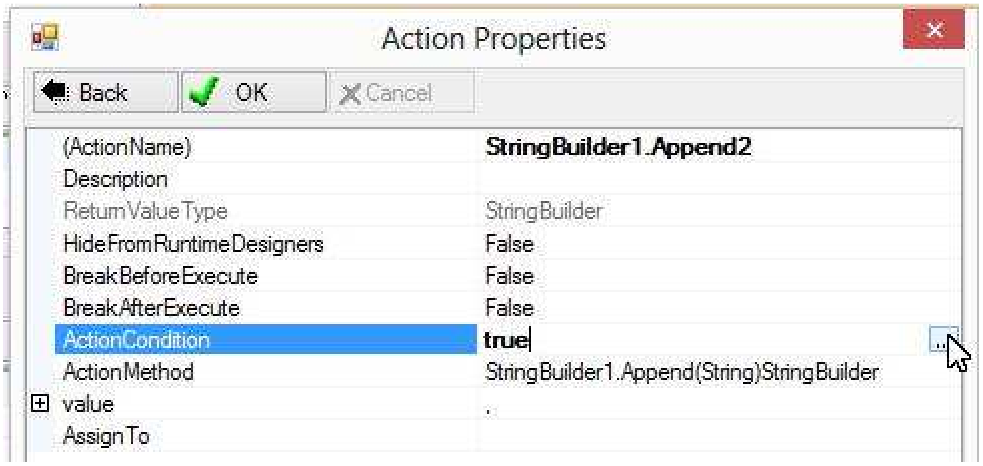
Misc
RepeatCount CheckedListBox1.CheckedIndices.Count

A new method editor appears for appending string part for each checked index. If the repeat index is greater than 0 then we need to append a comma:

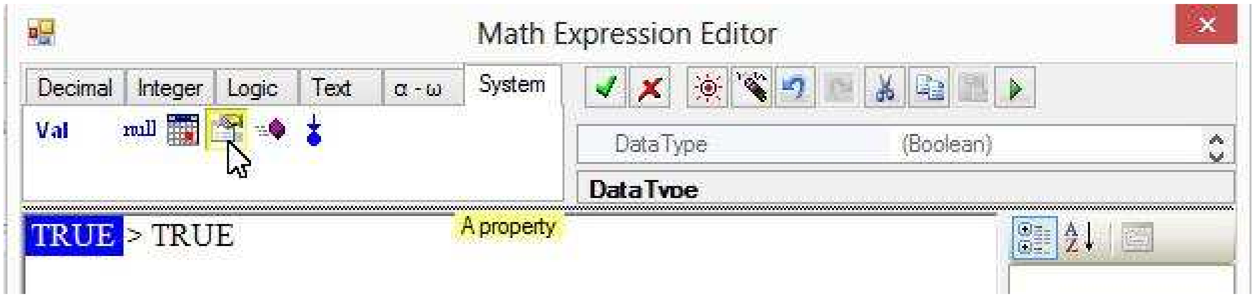

Method Desig
Main
LimitedRepeat1.RepeatIndex
StringBuilder1
CheckedListBox1
Create Action
Create Set Property Action
Assign Action
Create Set Value Action
Append(Int64)StringBuilder
Append(Object)StringBuilder
Append(SByte)StringBuilder
Append(Single)StringBuilder
Append(String)StringBuilder
Append(String,Int32,Int32)StringBuilder
Append(UInt16)StringBuilder
Append(UInt32)StringBuilder
Append(UInt64)StringBuilder
AppendFormat(IFormatProvider,String,Object[])StringBuilder
More methods

(Name)	StringBuilder1
ClassType	StringBuilder
IsValueType	False
RunAt	Server

Set ", " for the "value" of the action; set ActionCondition to that the repeat index is greater than 0:



(ActionName)	StringBuilder1.Append2
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	StringBuilder1.Append(String)StringBuilder
⊞ value	,
AssignTo	

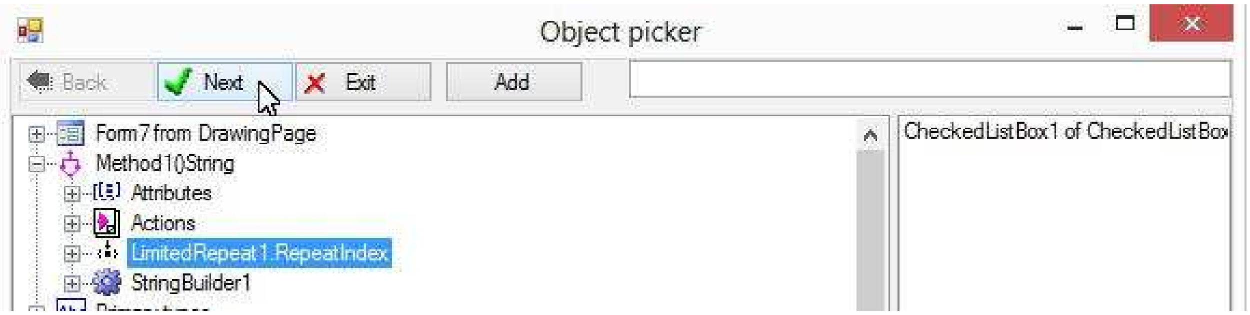


Data Type (Boolean)

Data Type

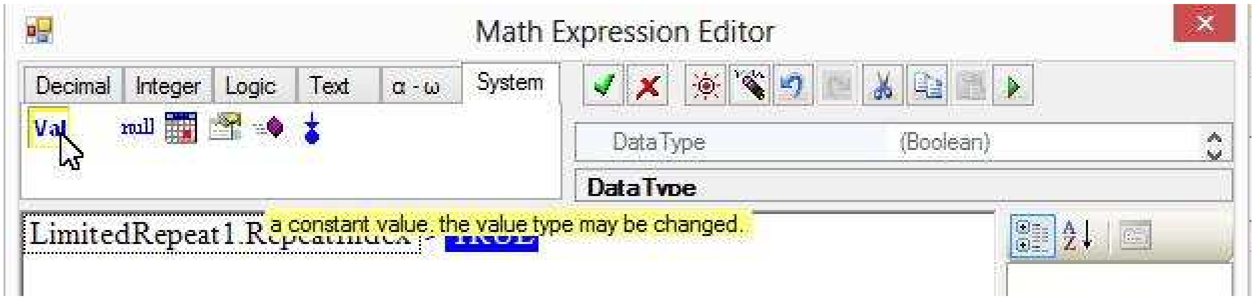
TRUE > TRUE

A property



- Form7 from DrawingPage
- Method1()String
  - Attributes
  - Actions
  - LimitedRepeat1.RepeatIndex
  - StringBuilder1

CheckedListBox1 of CheckedListBox

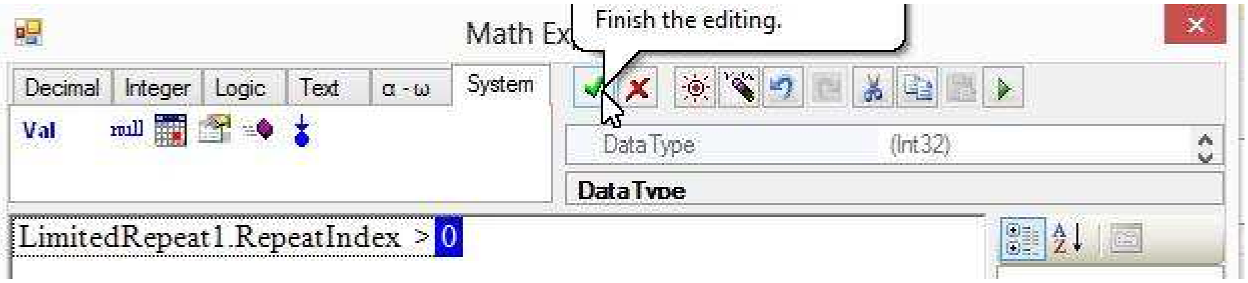


Data Type (Boolean)

Data Type

LimitedRepeat1.RepeatIndex > TRUE

a constant value. the value type may be changed.

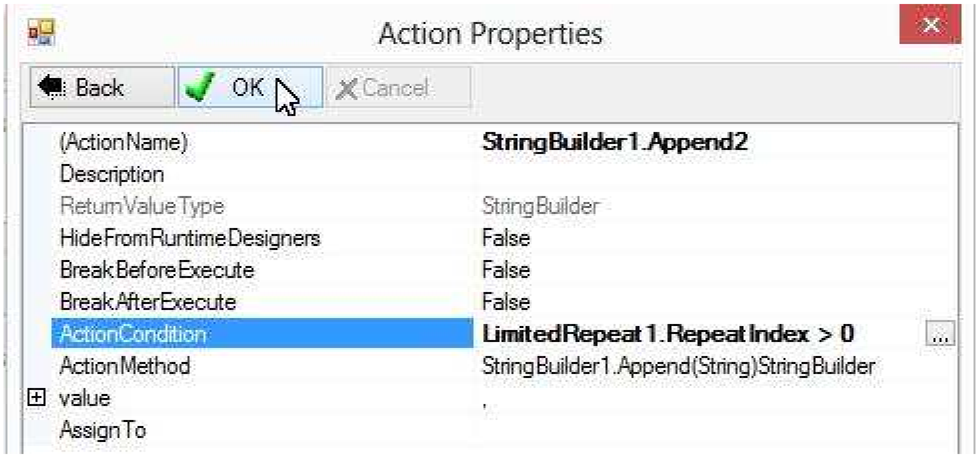


Finish the editing.

Data Type (Int32)

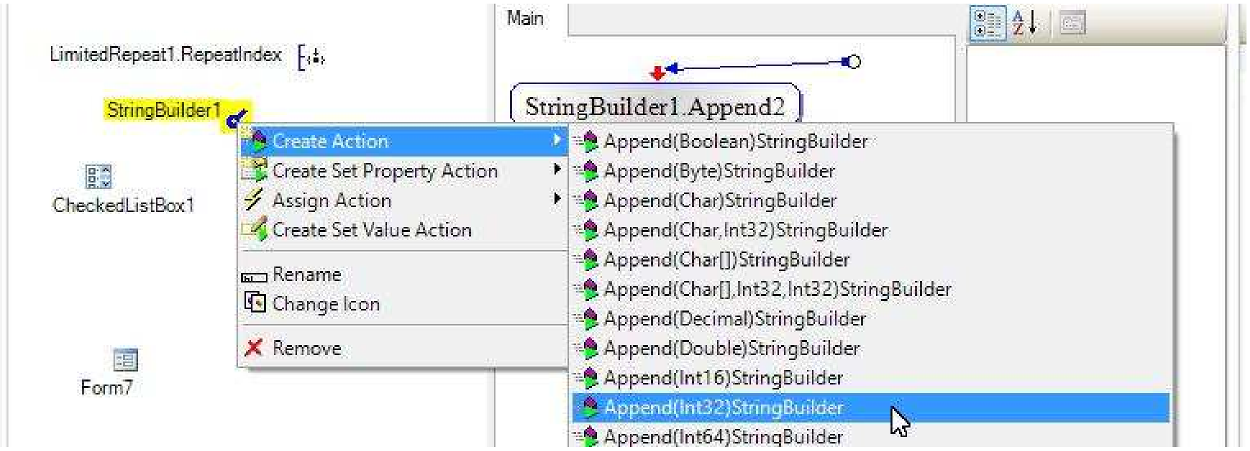
Data Type

LimitedRepeat1.RepeatIndex > 0



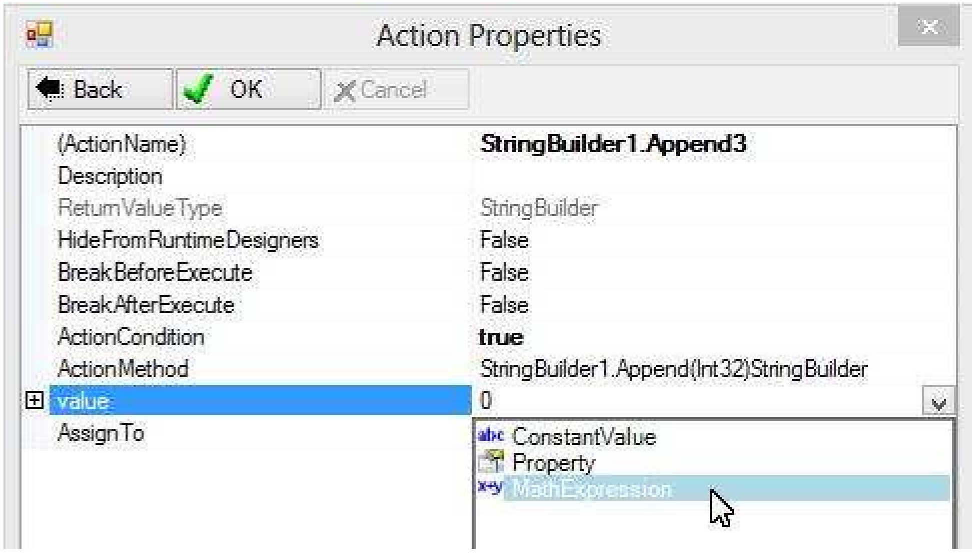
(ActionName)	**StringBuilder1.Append2**
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**LimitedRepeat1.RepeatIndex > 0**
ActionMethod	StringBuilder1.Append(String)StringBuilder
⊞ value	,
AssignTo	

Add an action to append the checked index to the StringBuilder:

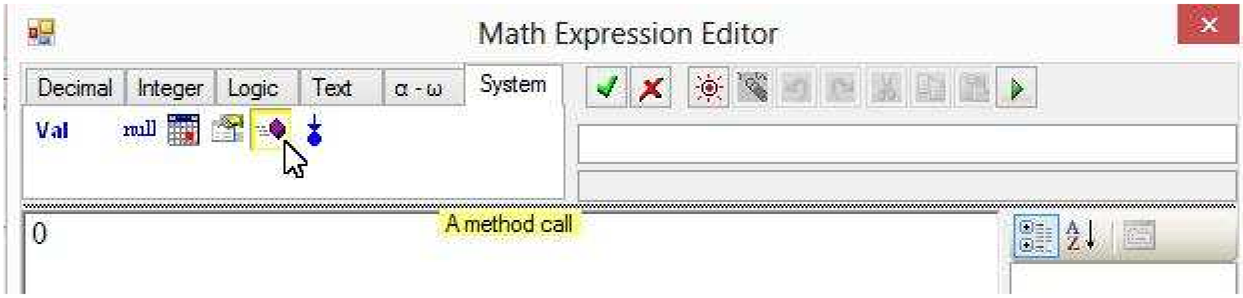


Main
LimitedRepeat1.RepeatIndex
StringBuilder1
StringBuilder1.Append2
CheckedListBox1
Form7
Create Action
Create Set Property Action
Assign Action
Create Set Value Action
Rename
Change Icon
Remove
Append(Boolean)StringBuilder
Append(Byte)StringBuilder
Append(Char)StringBuilder
Append(Char,Int32)StringBuilder
Append(Char[])StringBuilder
Append(Char[],Int32,Int32)StringBuilder
Append(Decimal)StringBuilder
Append(Double)StringBuilder
Append(Int16)StringBuilder
Append(Int32)StringBuilder
Append(Int64)StringBuilder

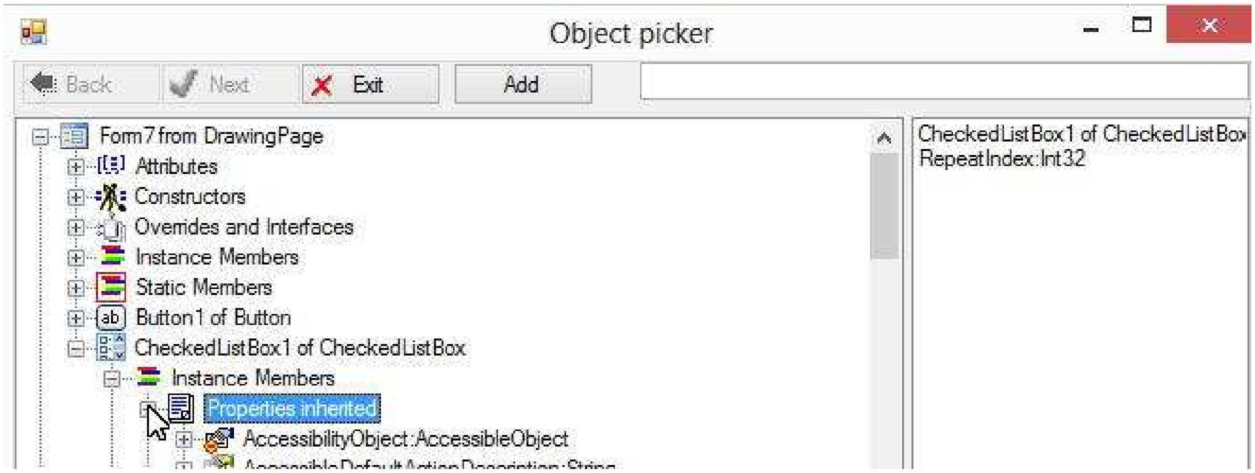
Use an expression to fetch checked index:



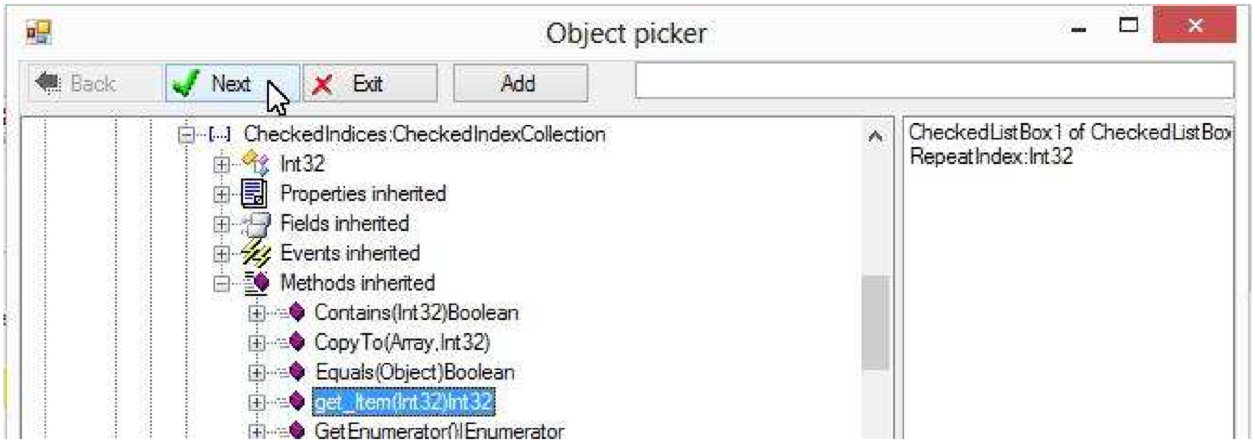
(ActionName)	**StringBuilder1.Append3**
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	StringBuilder1.Append(Int32)StringBuilder
⊞ value	0
AssignTo	



0

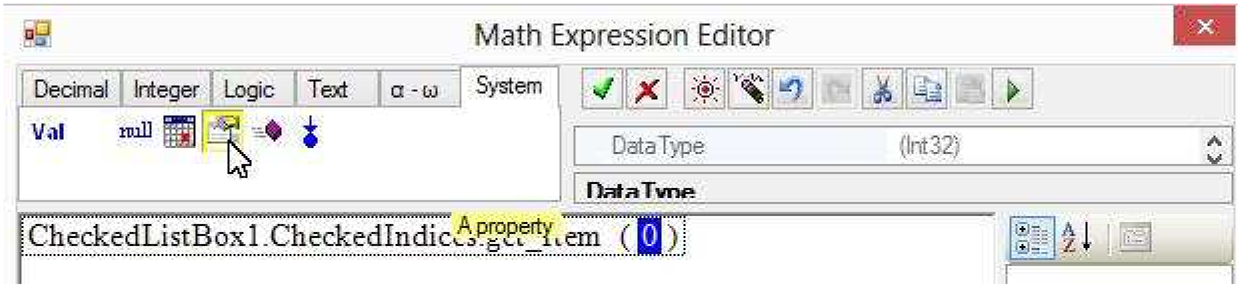


CheckedListBox1 of CheckedListBox
RepeatIndex:Int32



CheckedListBox1 of CheckedListBox
RepeatIndex:Int32

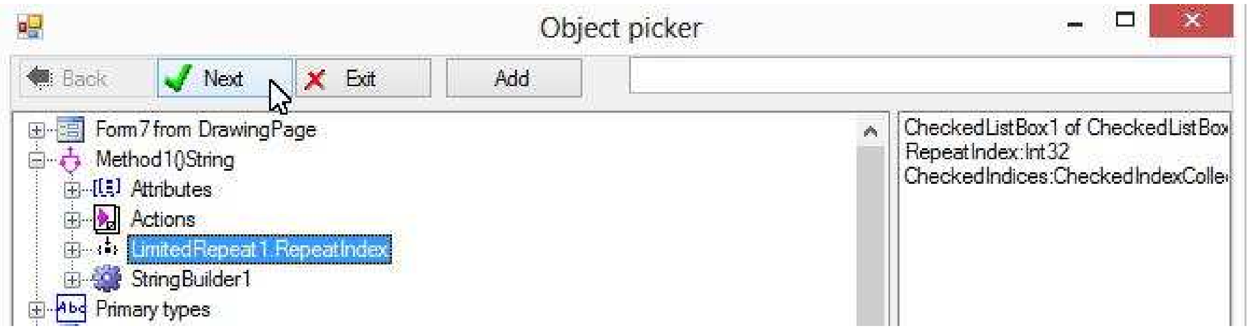
Use the repeat index to fetch checked index:



DataType (Int32)

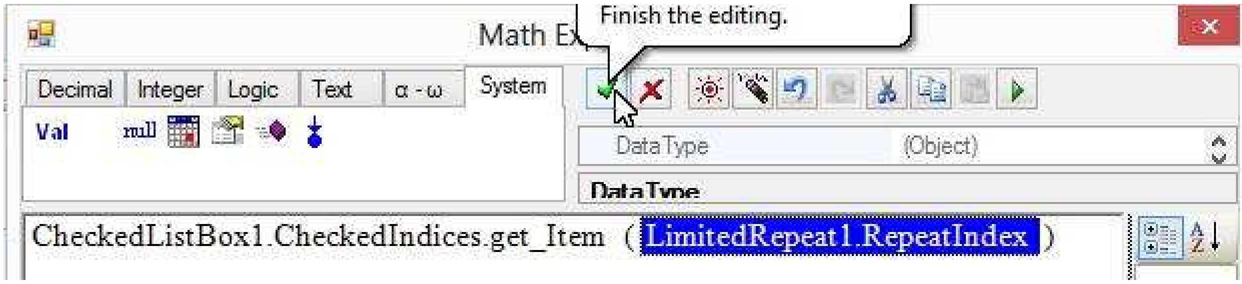
DataType

CheckedListBox1.CheckedIndices.get_Item (0)



- Form7 from DrawingPage
- Method1()String
  - Attributes
  - Actions
  - LimitedRepeat1.RepeatIndex
  - StringBuilder1
- Primary types

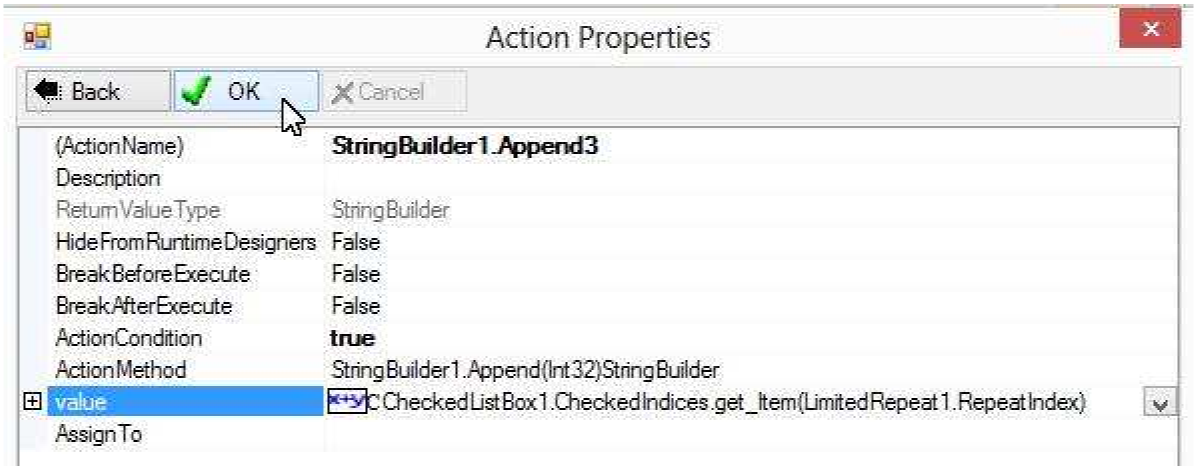
CheckedListBox1 of CheckedListBox
RepeatIndex:Int32
CheckedIndices:CheckedIndexColle


Finish the editing.

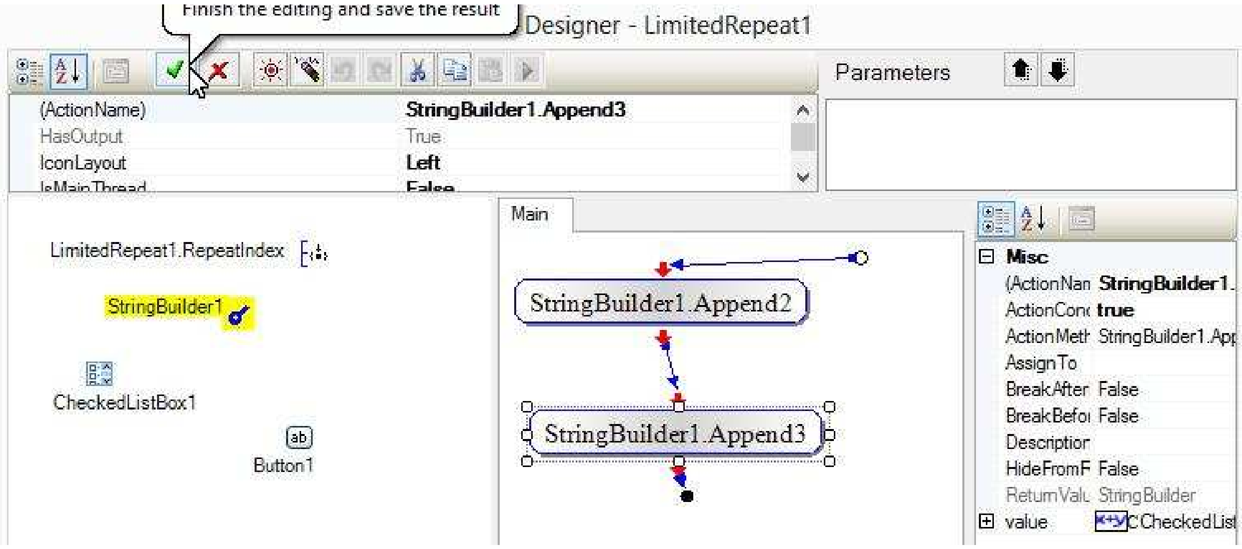
DataType	(Object)

DataType

```
CheckedListBox1.CheckedIndices.get_Item ( LimitedRepeat1.RepeatIndex )
```



(ActionName)	**StringBuilder1.Append3**
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	StringBuilder1.Append(Int32)StringBuilder
⊞ value	CCheckedListBox1.CheckedIndices.get_Item(LimitedRepeat1.RepeatIndex)
AssignTo	

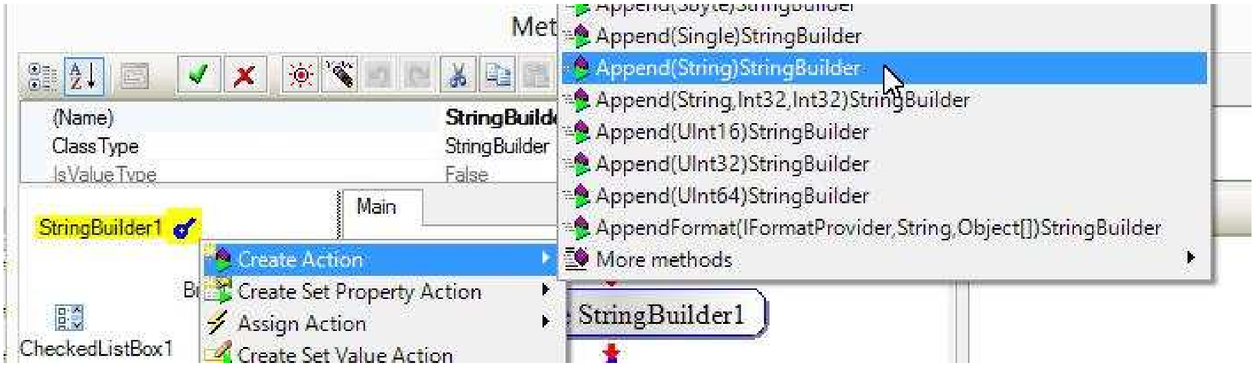

Finish the editing and save the result

(ActionName)	**StringBuilder1.Append3**
HasOutput	True
IconLayout	**Left**
IsMainThread	**False**

⊟ **Misc**	
(ActionNan	**StringBuilder1.**
ActionCond	**true**
ActionMeth	StringBuilder1.App
AssignTo	
BreakAfter	False
BreakBefor	False
Description	
HideFromF	False
ReturnValu	StringBuilder
⊞ value	CCheckedList

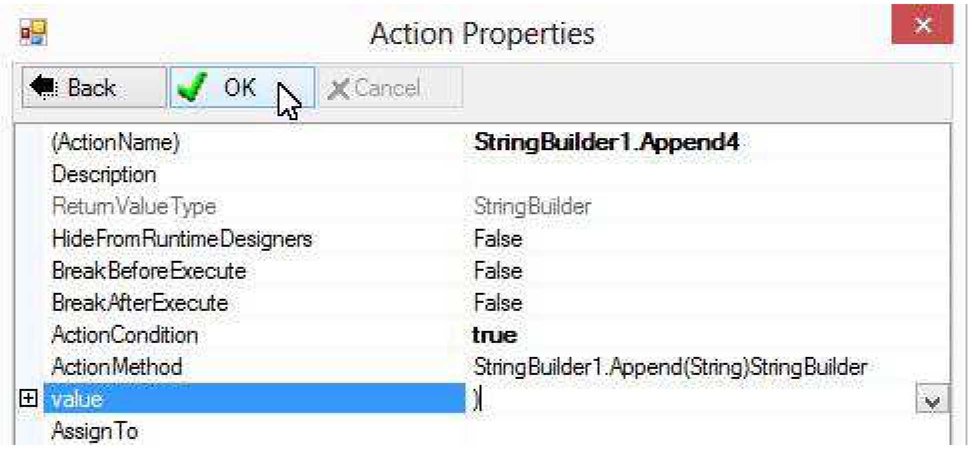
Add last string part

The last part is “)”.



(Name)	StringBuild
ClassType	StringBuilder
IsValueType	False

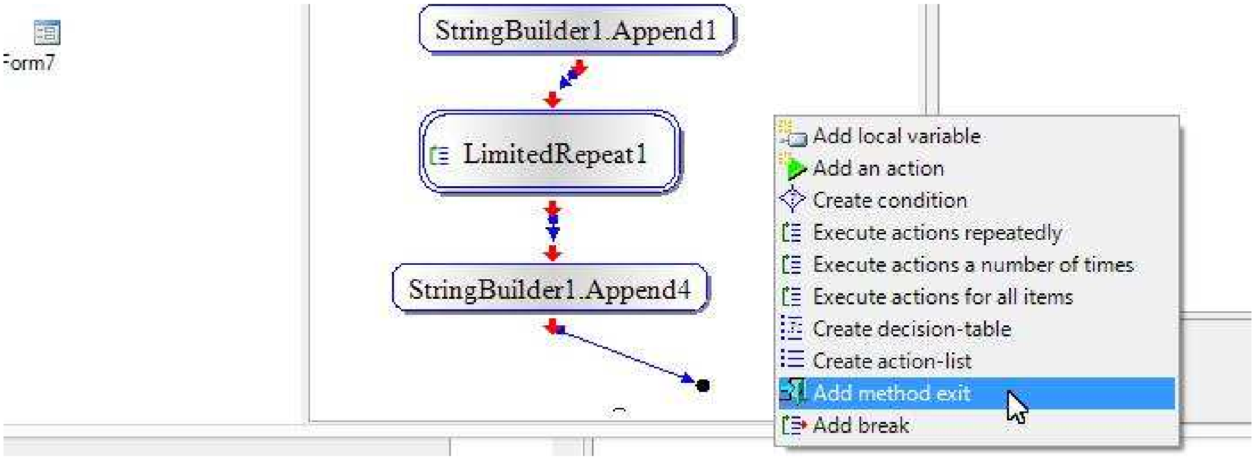
Met
Main
StringBuilder1
Create Action
Create Set Property Action
Assign Action
Create Set Value Action
CheckedListBox1
Append(SByte)StringBuilder
Append(Single)StringBuilder
Append(String)StringBuilder
Append(String,Int32,Int32)StringBuilder
Append(UInt16)StringBuilder
Append(UInt32)StringBuilder
Append(UInt64)StringBuilder
AppendFormat(IFormatProvider,String,Object[])StringBuilder
More methods
StringBuilder1



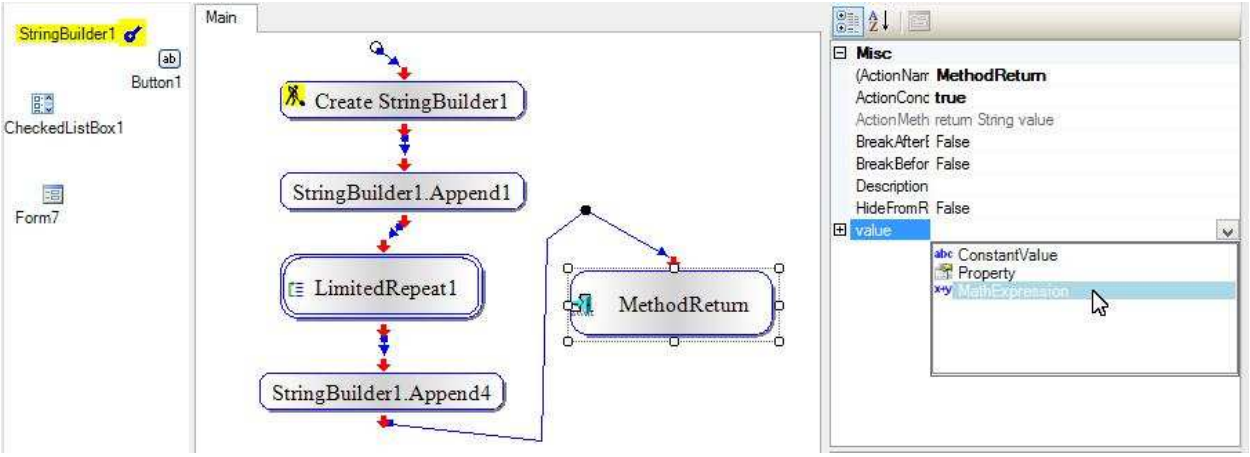
(ActionName)	**StringBuilder1.Append4**
Description	
ReturnValueType	StringBuilder
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	StringBuilder1.Append(String)StringBuilder
⊞ value	)
AssignTo	

Return with ToString

Add an exit action to finish the method and return desired value:

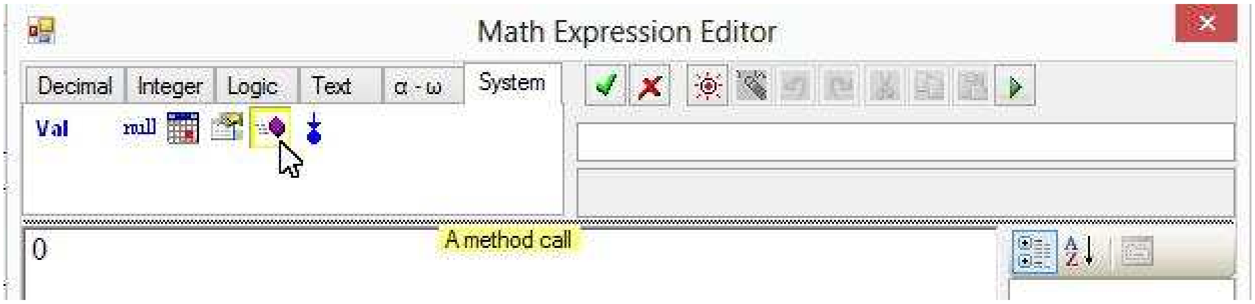


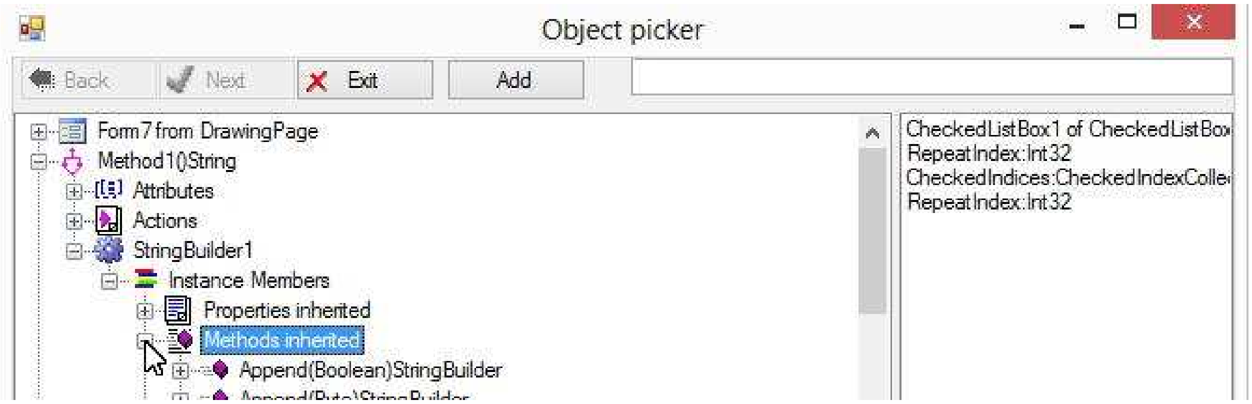
Use an expression so that we may execute ToString method of the StringBuilder:



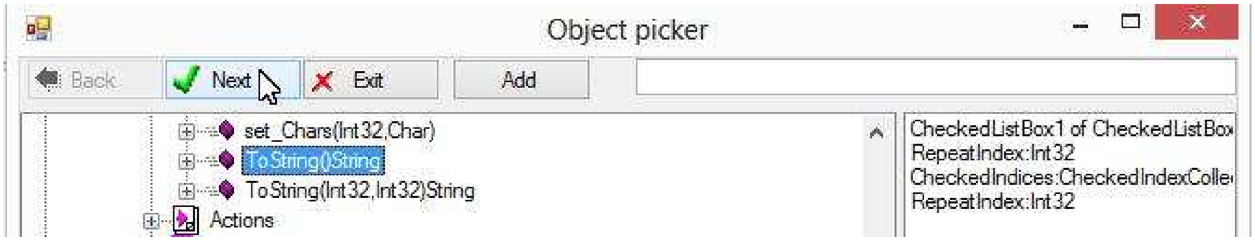
(ActionNam MethodReturn
ActionCond true
ActionMeth return String value
BreakAfterI False
BreakBefor False
Description
HideFromR False
value

Select ToString method of the StringBuilder:

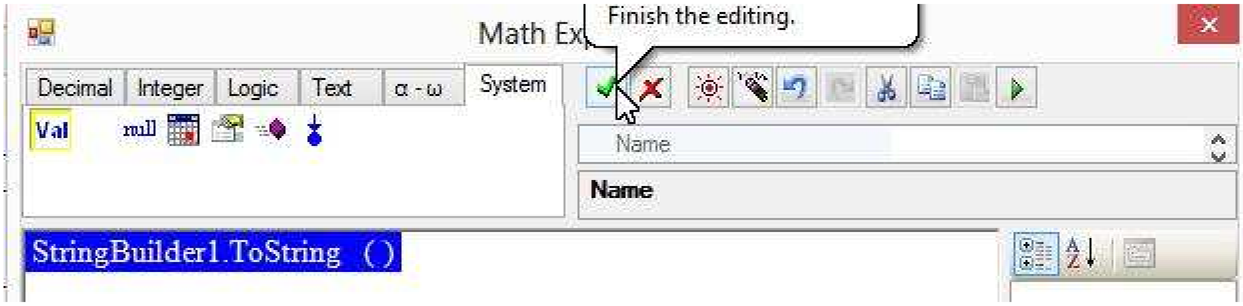




CheckedListBox1 of CheckedListBox
RepeatIndex:Int32
CheckedIndices:CheckedIndexColle
RepeatIndex:Int32



CheckedListBox1 of CheckedListBox
RepeatIndex:Int32
CheckedIndices:CheckedIndexColle
RepeatIndex:Int32

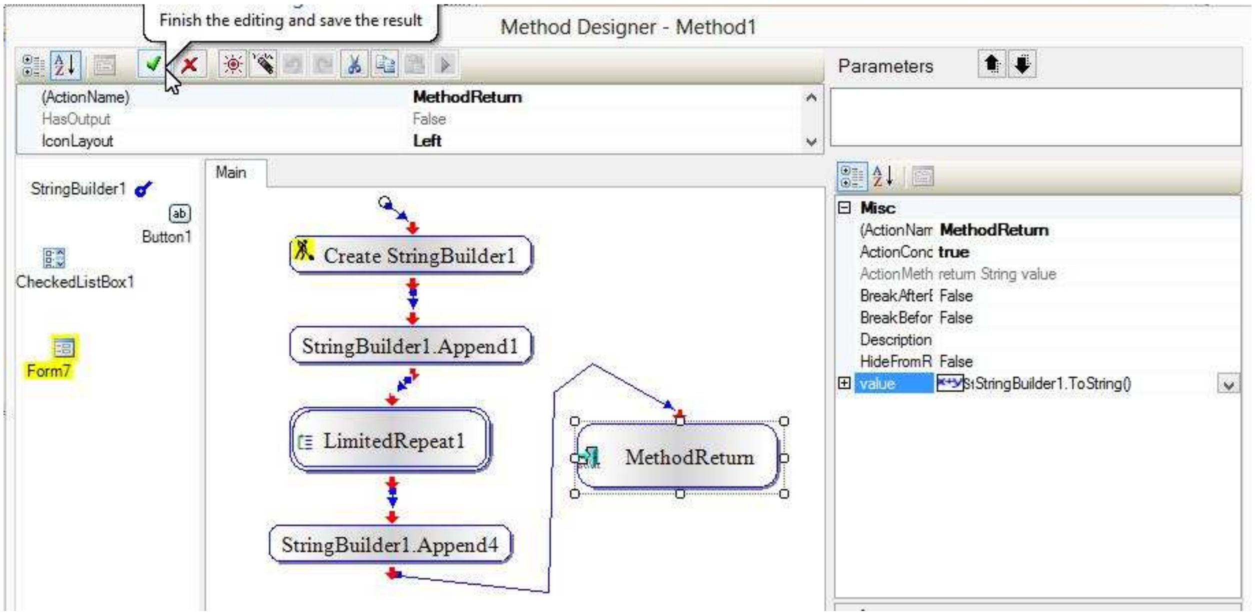


Finish the editing.

Math Ex

StringBuilder1.ToString ( )

That is all the actions we need:



Finish the editing and save the result

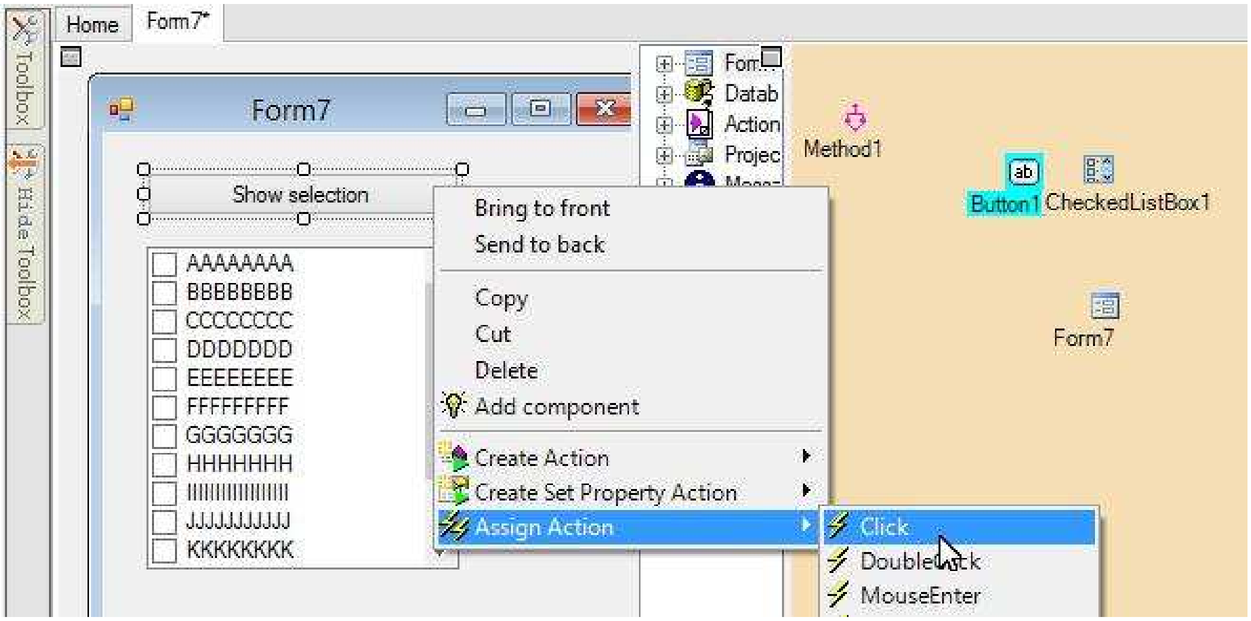
Method Designer - Method1

(ActionName) MethodReturn
HasOutput False
IconLayout Left

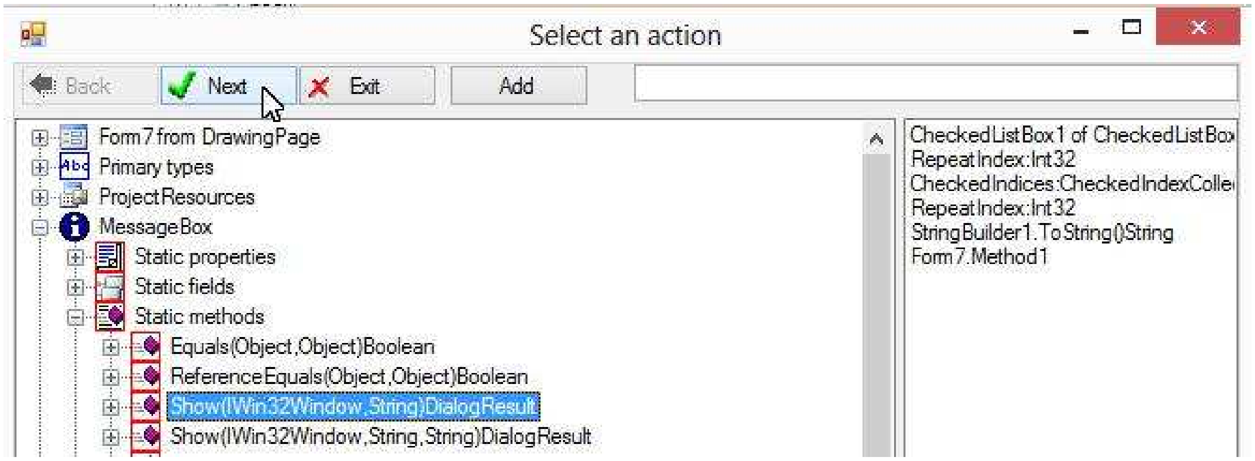
Misc
(ActionNam MethodReturn
ActionConc true
ActionMeth return String value
BreakAfterI False
BreakBefor False
Description
HideFromR False
value StringBuilder1.ToString()

Execute method

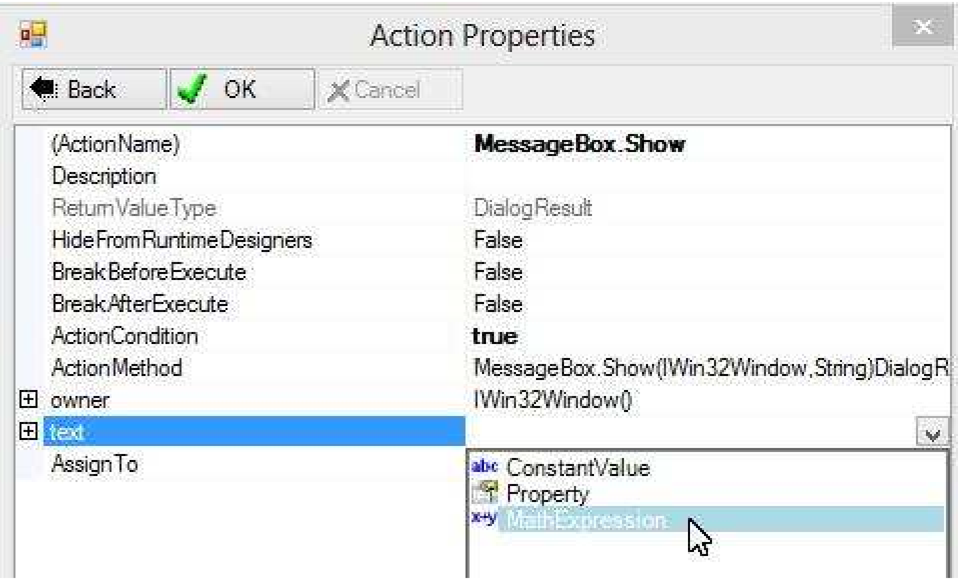
Assign a message box action to the button:



Select a message box action:

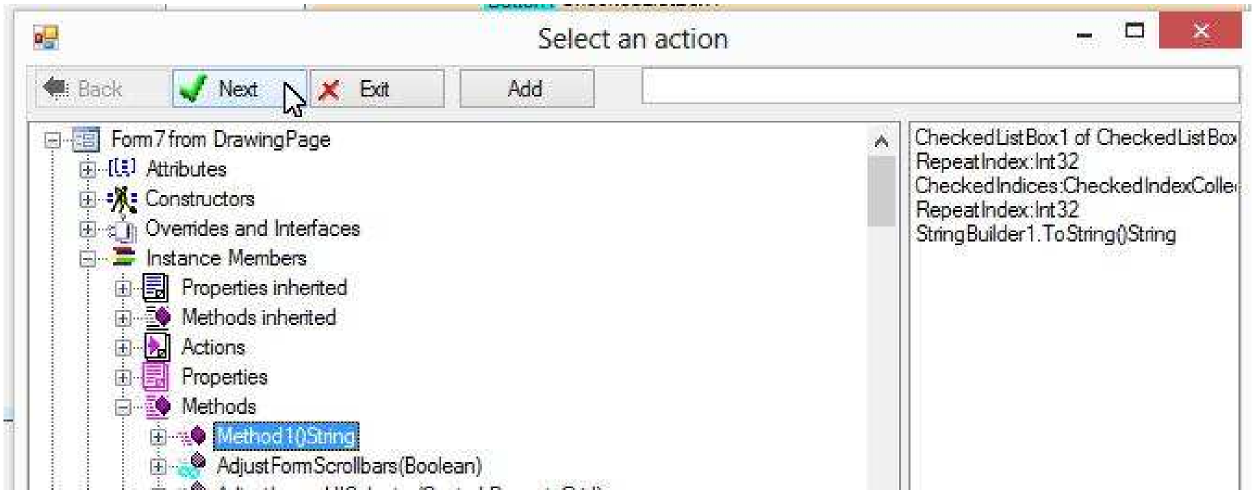

Select an action
Back
Next
Exit
Add
Form7 from DrawingPage
Primary types
ProjectResources
MessageBox
Static properties
Static fields
Static methods
Equals(Object,Object)Boolean
ReferenceEquals(Object,Object)Boolean
Show(IWin32Window,String)DialogResult
Show(IWin32Window,String,String)DialogResult
CheckedListBox1 of CheckedListBox
RepeatIndex:Int32
CheckedIndices:CheckedIndexColle
RepeatIndex:Int32
StringBuilder1.ToString()String
Form7.Method1

Use an expression to call the method we created previously:



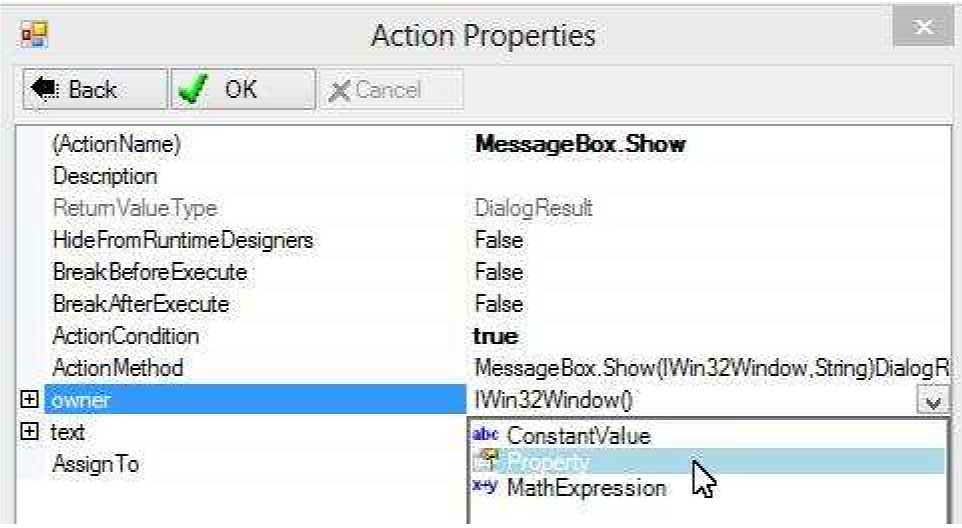
(ActionName)	**MessageBox.Show**
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	MessageBox.Show(IWin32Window,String)DialogR
⊞ owner	IWin32Window()
⊞ text	
AssignTo	

Select the method:

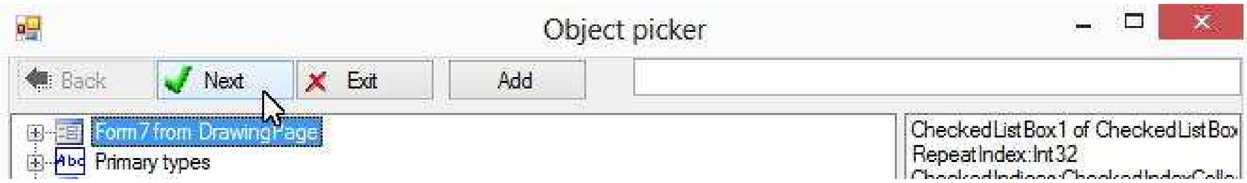


CheckedListBox1 of CheckedListBox
RepeatIndex:Int32
CheckedIndices:CheckedIndexColle
RepeatIndex:Int32
StringBuilder1.ToString()String

Use the current form for the message owner:

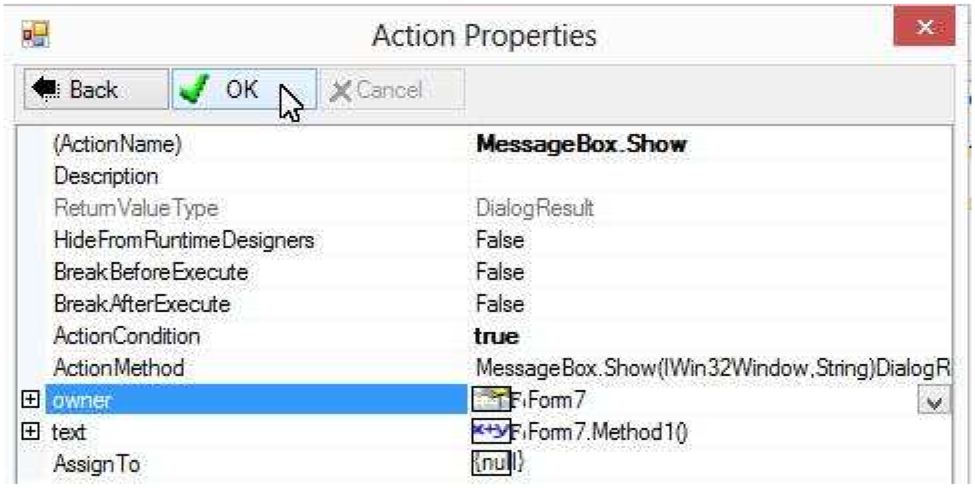


(ActionName)	**MessageBox.Show**
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	MessageBox.Show(IWin32Window,String)DialogR
owner	IWin32Window()
text	
AssignTo	



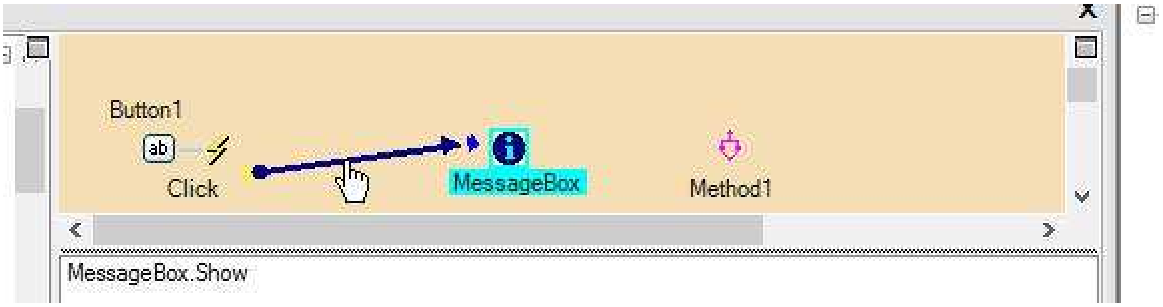
CheckedListBox1 of CheckedListBox
RepeatIndex:Int32

Click OK:



(ActionName)	**MessageBox.Show**
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	MessageBox.Show(IWin32Window,String)DialogR
owner	F.Form7
text	F.Form7.Method1()
AssignTo	{null}

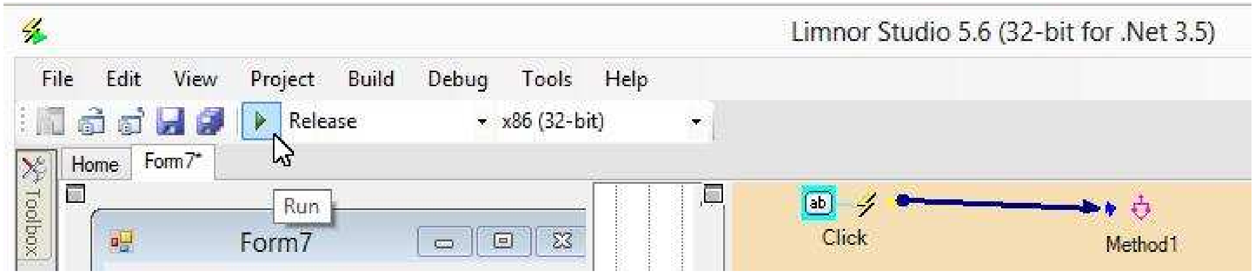
The action the created and assigned to the button:



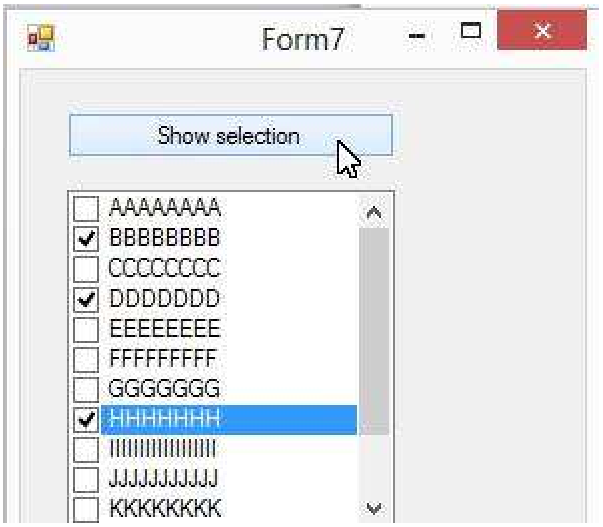
MessageBox.Show

Test

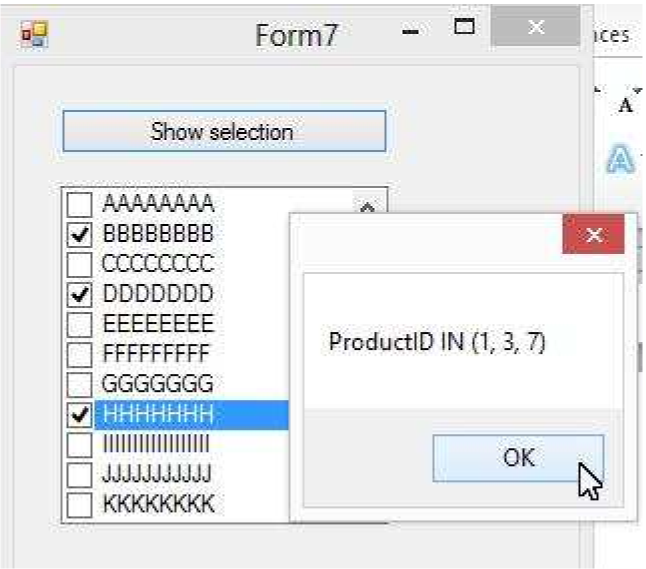
Now we may test the project:



The form appears. Checked some items and click the button:

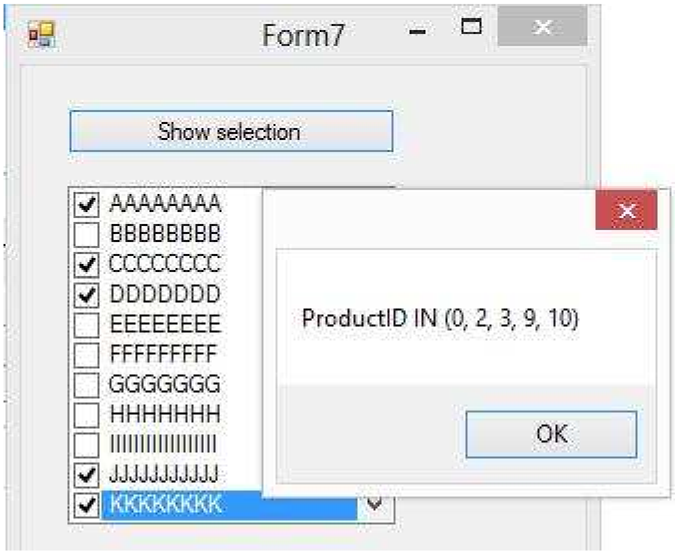


A message box appears, showing selected indexes:

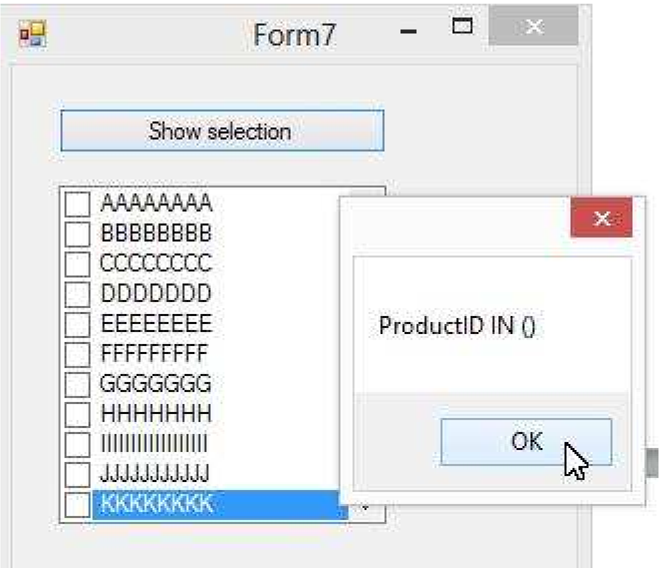


ProductID IN (1, 3, 7)

Make other selections and click the button:



ProductID IN (0, 2, 3, 9, 10)



ProductID IN ()