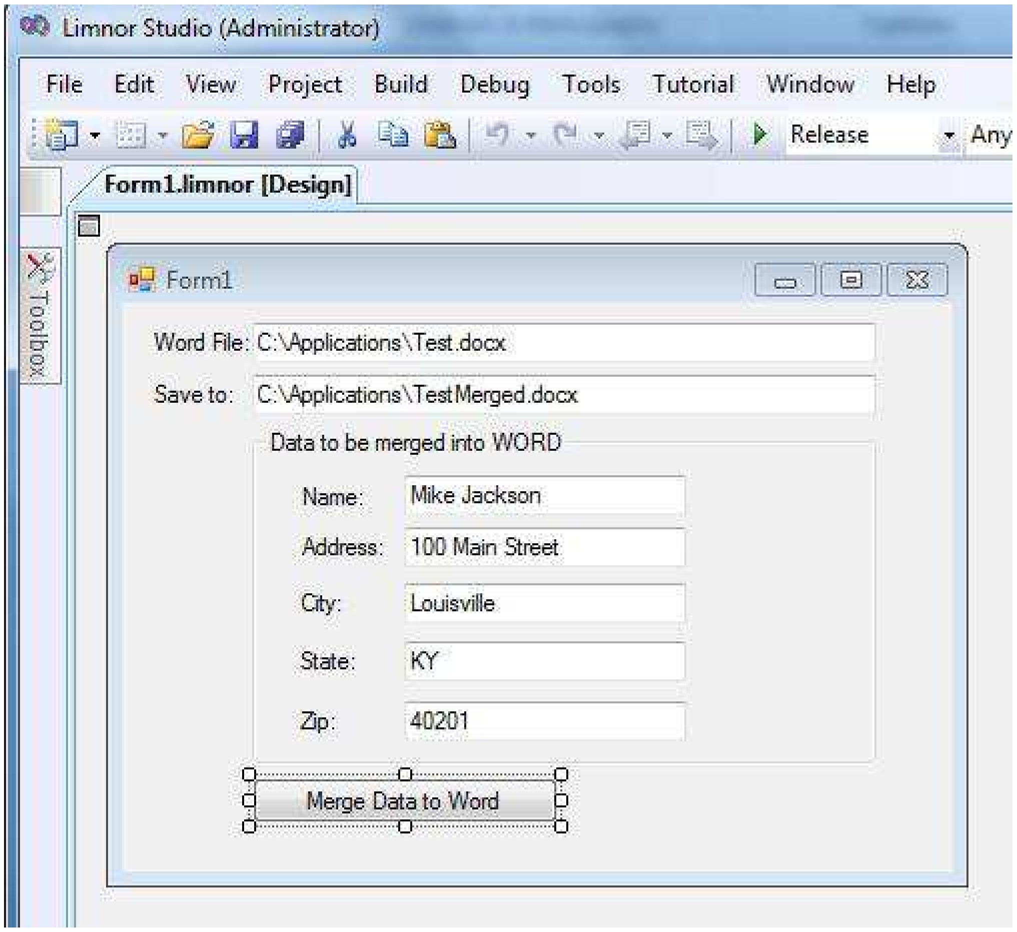
Manipulate Microsoft Office Word

Contents

Introduction

All Microsoft Office applications can be manipulated programmatically using .Net Framework interfaces provided by Microsoft. This sample demonstrates merging data into a Word document and save the document to a new file.

The sample is a Windows Form application. The data to be merged into the Word document are entered into text boxes. A button is used to initiate the data merging action.


Limnor Studio (Administrator)
File Edit View Project Build Debug Tools Tutorial Window Help
Release
Any
Form1.limnor [Design]
Toolbox
Form1

Word File: C:\Applications\Test.docx

Save to: C:\Applications\TestMerged.docx

Data to be merged into WORD

Name: Mike Jackson

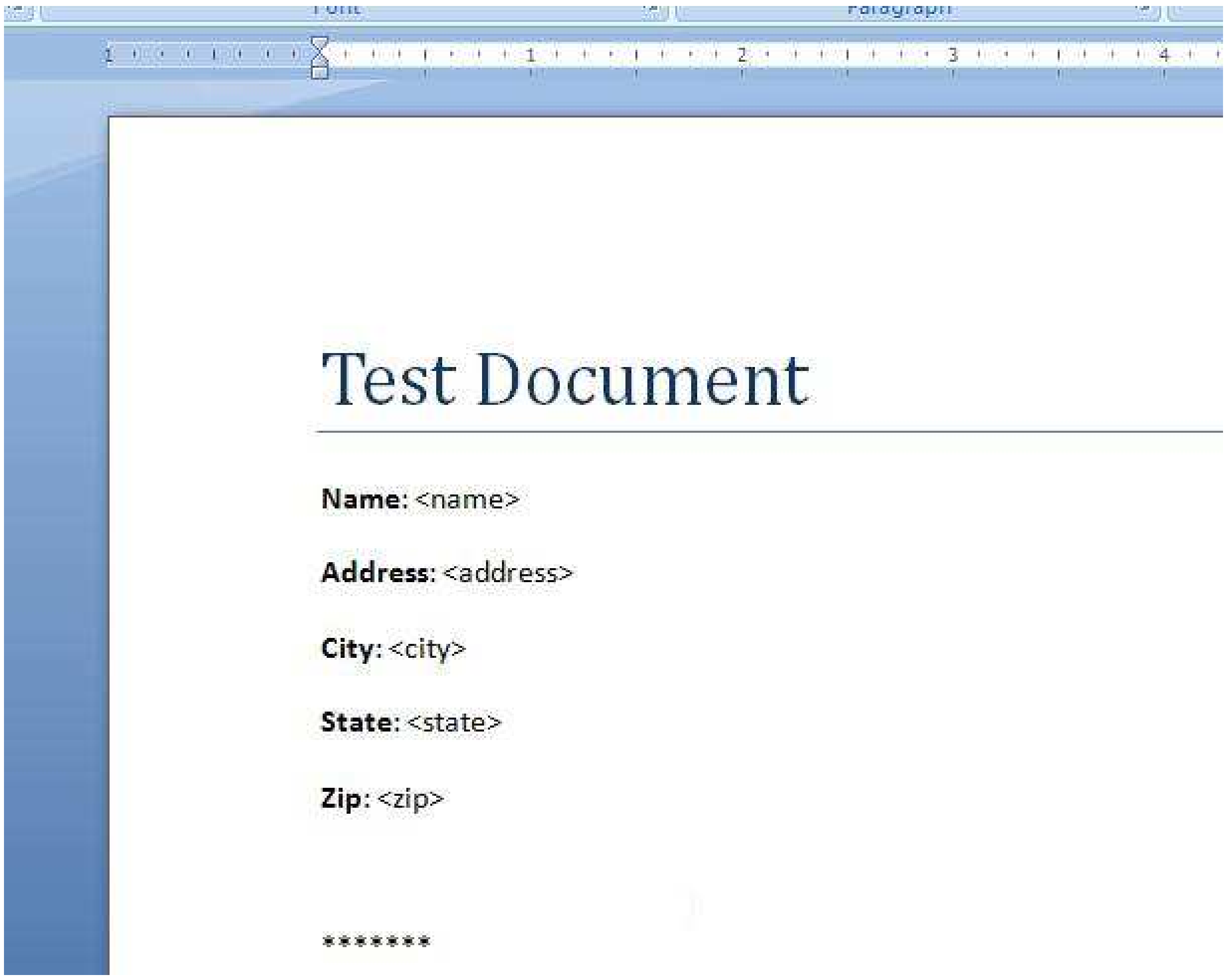
Address: 100 Main Street

City: Louisville

State: KY

Zip: 40201

Suppose we use such a test WORD document:



Test Document

Name: <name>

Address: <address>

City: <city>

State: <state>

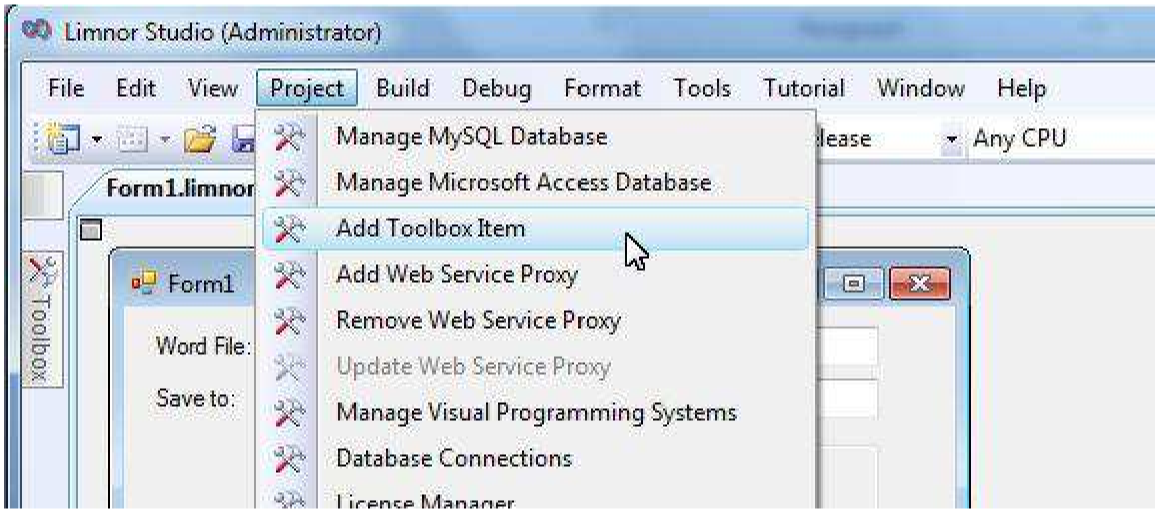
Zip: <zip>

****

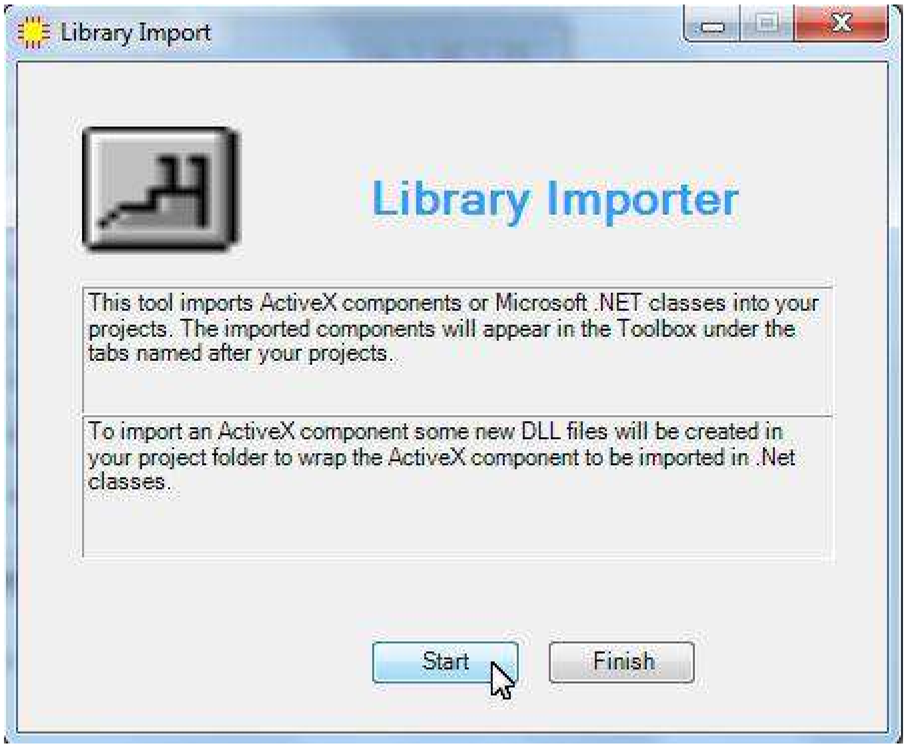
When the button is clicked we want to use the data in the text boxes to replace the place holders marked with "<" and ">".

Add a Word Application to the form

We want to add a Word Application object to the form so that all methods we will be creating can access it. We may add a Word Application object to the Toolbox first. Select "Project" menu and choose "Add Toolbox Item"



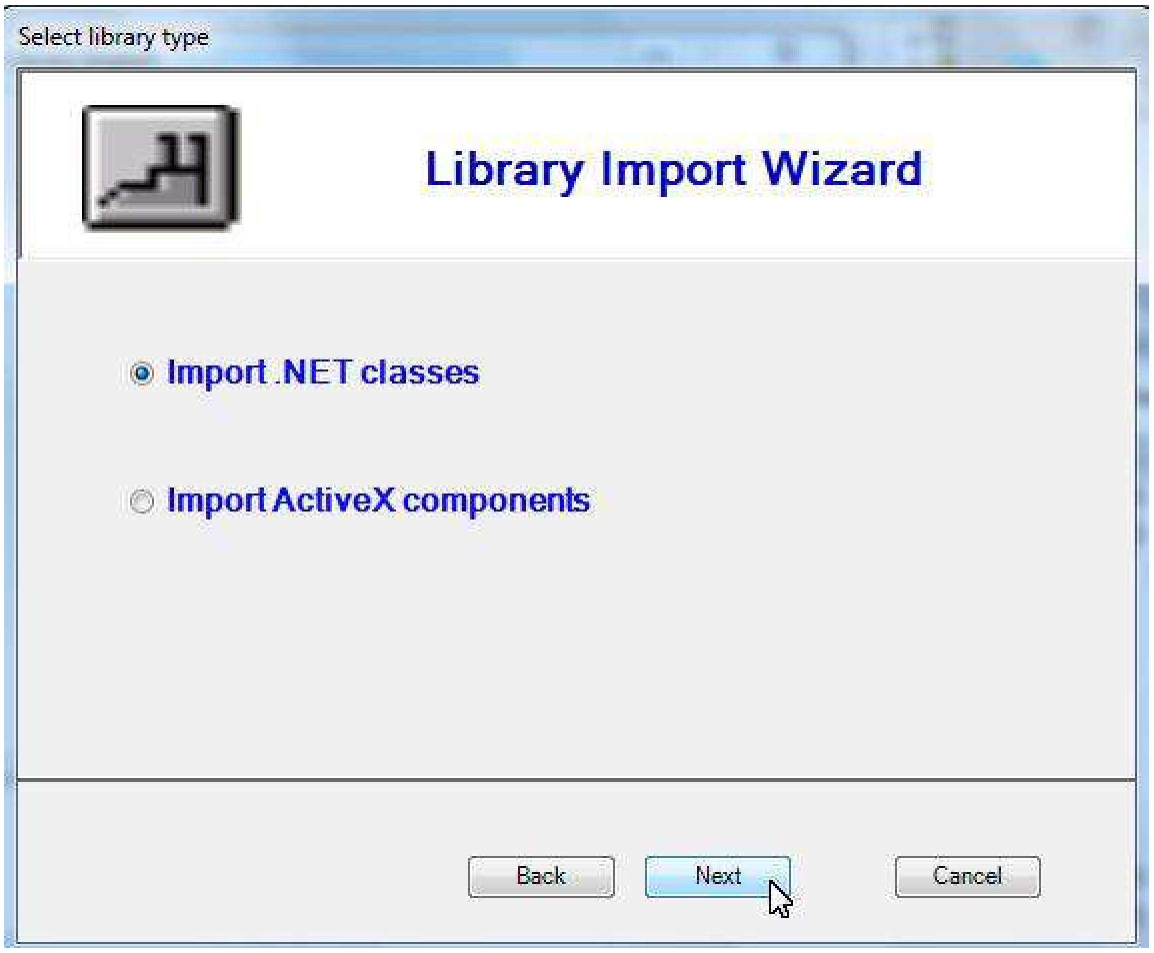
Click Start:



This tool imports ActiveX components or Microsoft .NET classes into your projects. The imported components will appear in the Toolbox under the tabs named after your projects.

To import an ActiveX component some new DLL files will be created in your project folder to wrap the ActiveX component to be imported in .Net classes.

Select “Import .Net classes”; click “Next”:

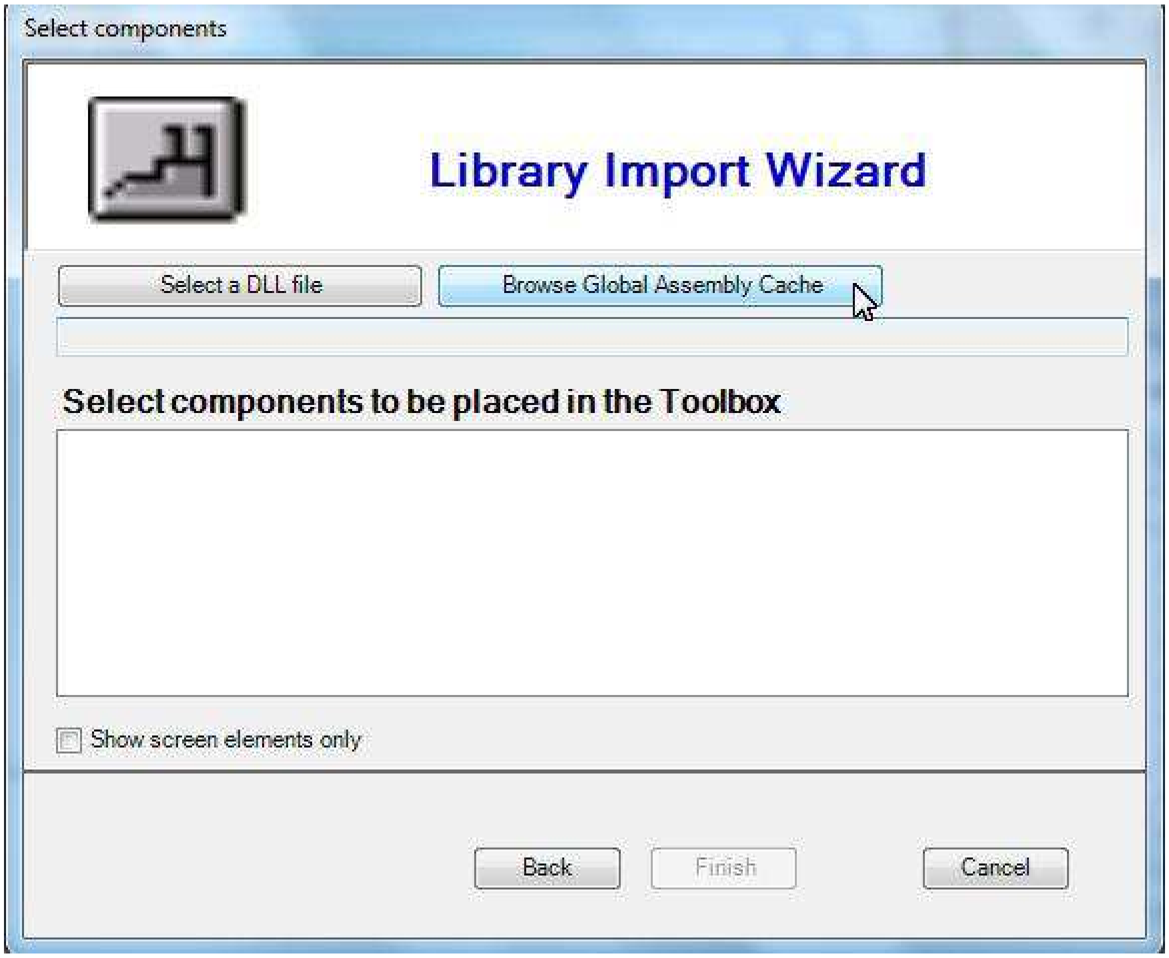

Select library type

Library Import Wizard

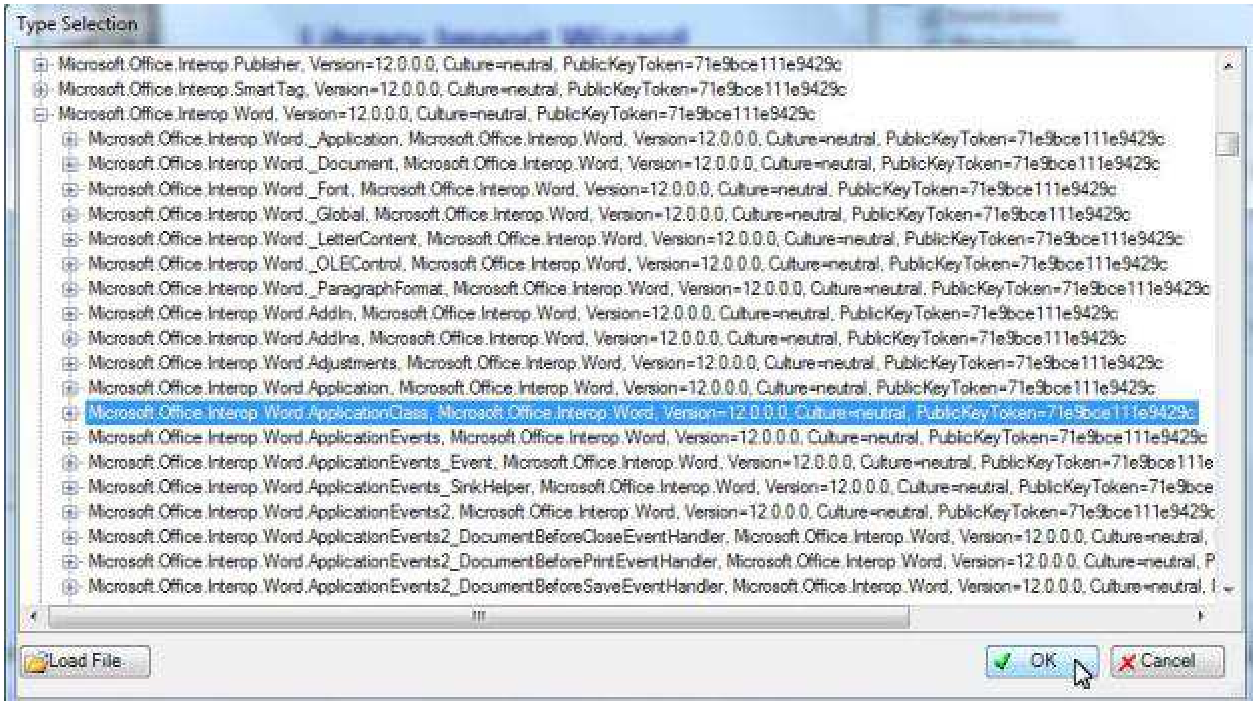
Import .NET classes [✓]

Import ActiveX components

Click “Browse Global Assembly Cache”:

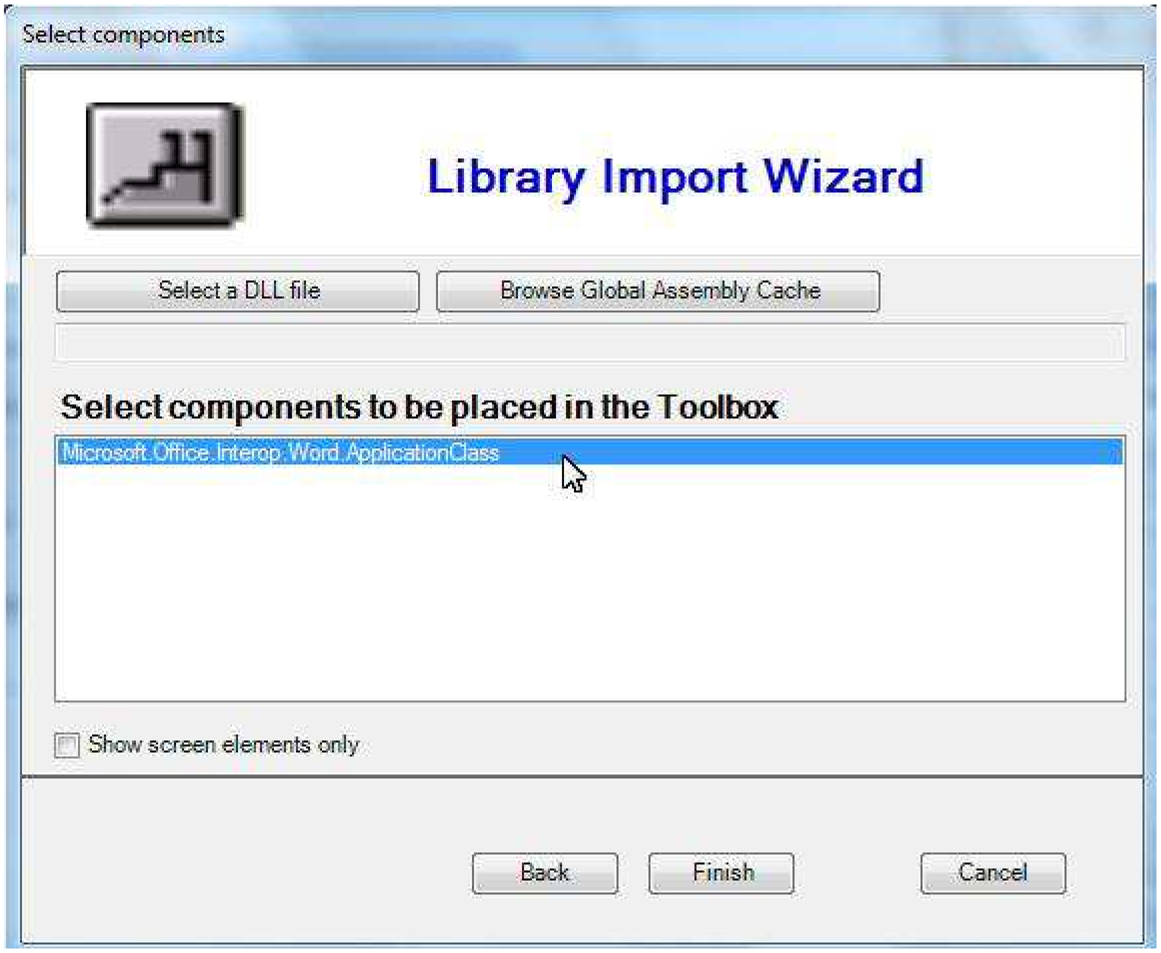

Select components
Library Import Wizard
Select a DLL file
Browse Global Assembly Cache
Select components to be placed in the Toolbox
Show screen elements only
Back
Finish
Cancel

Expand “Microsoft.Office.Interop.Word”; select “Microsoft.Office.Interop.Word.ApplicationClass”; click OK:

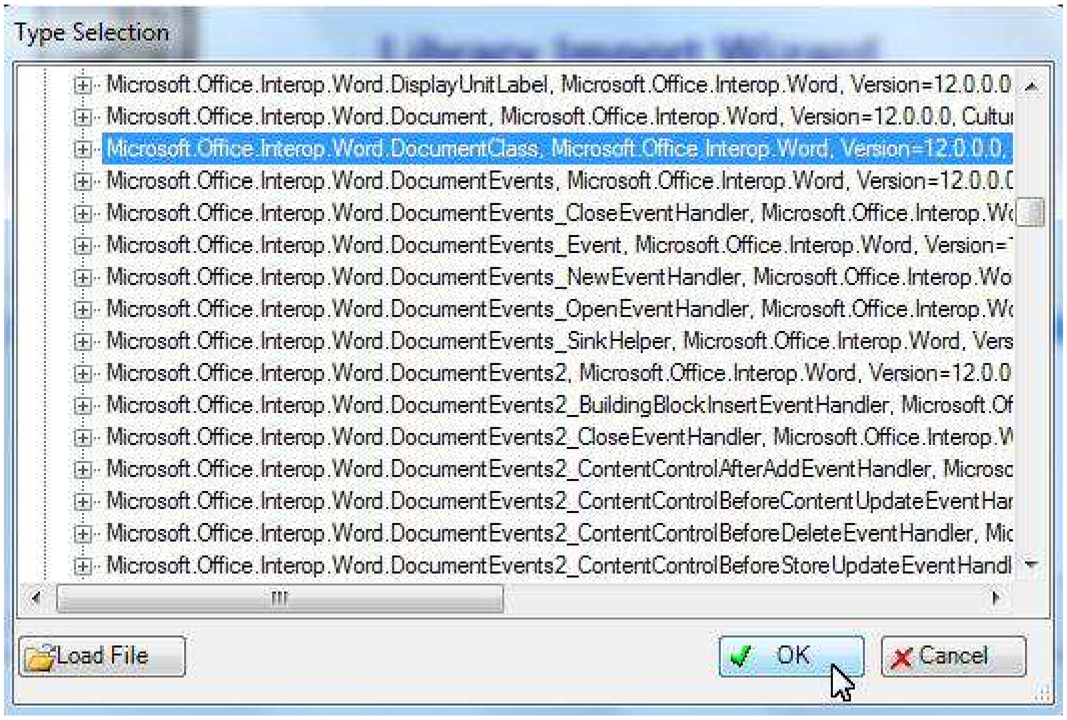


```
⊞ Microsoft.Office.Interop.Publisher, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
⊞ Microsoft.Office.Interop.SmartTag, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
⊟ Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._Application, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._Document, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._Font, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._Global, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._LetterContent, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._OLEControl, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word._ParagraphFormat, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.AddIn, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.AddIns, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.Adjustments, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.Application, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.ApplicationClass, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents_Event, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents_SinkHelper, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents2, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents2_DocumentBeforeCloseEventHandler, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral,
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents2_DocumentBeforePrintEventHandler, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, P
    ⊞ Microsoft.Office.Interop.Word.ApplicationEvents2_DocumentBeforeSaveEventHandler, Microsoft.Office.Interop.Word, Version=12.0.0.0, Culture=neutral, I
```

The selected component appears in the list:

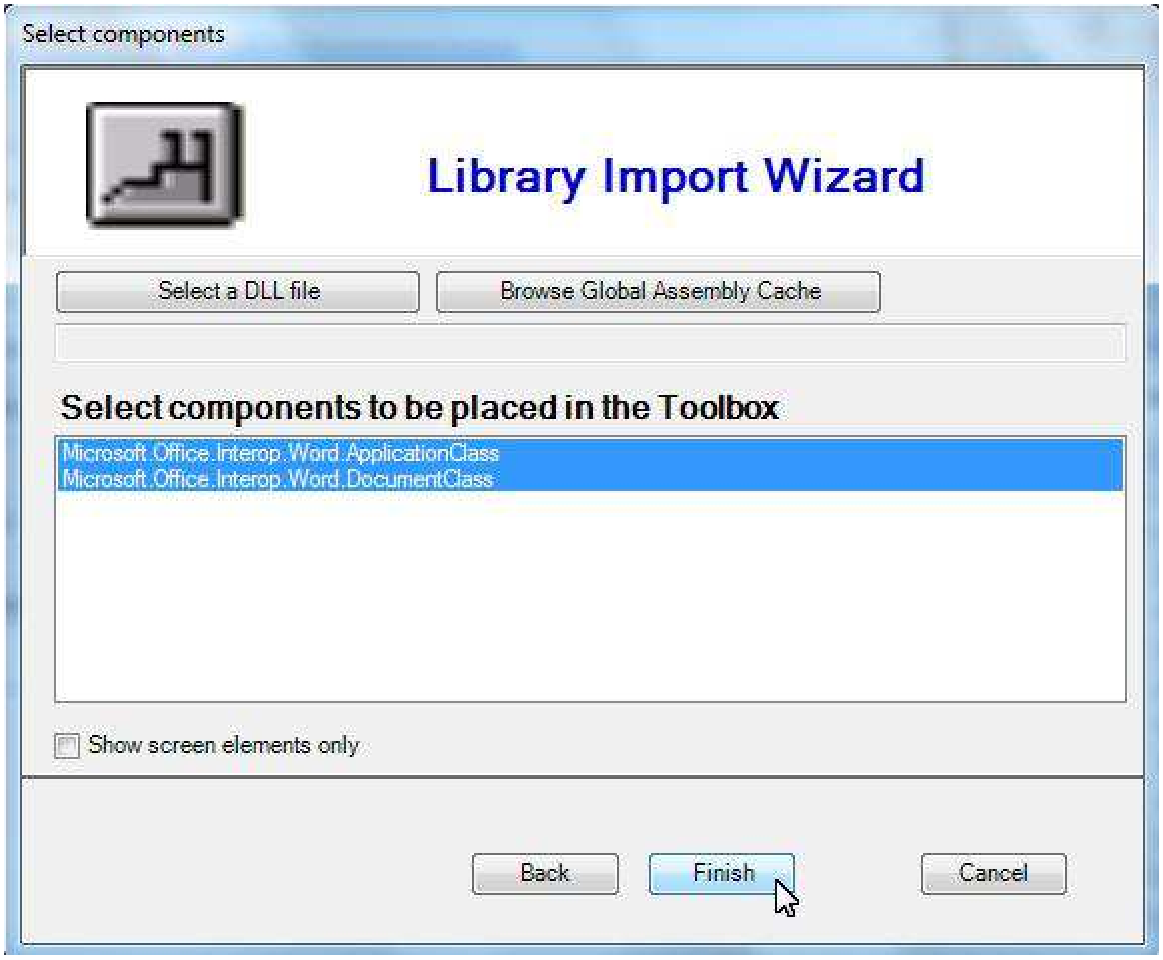

Select components
Library Import Wizard
Select a DLL file
Browse Global Assembly Cache
Select components to be placed in the Toolbox
Microsoft.Office.Interop.Word.ApplicationClass
Show screen elements only
Back
Finish
Cancel

We may add all components we are going to use. For example, add DocumentClass:

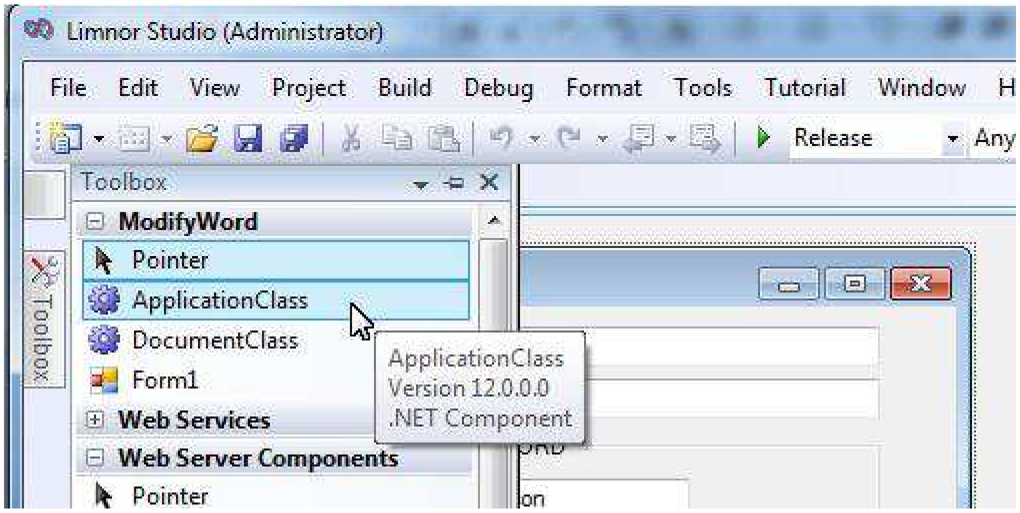


```
Microsoft.Office.Interop.Word.DisplayUnitLabel, Microsoft.Office.Interop.Word, Version=12.0.0.0
Microsoft.Office.Interop.Word.Document, Microsoft.Office.Interop.Word, Version=12.0.0.0, Cultu
Microsoft.Office.Interop.Word.DocumentClass, Microsoft.Office.Interop.Word, Version=12.0.0.0,
Microsoft.Office.Interop.Word.DocumentEvents, Microsoft.Office.Interop.Word, Version=12.0.0.0
Microsoft.Office.Interop.Word.DocumentEvents_CloseEventHandler, Microsoft.Office.Interop.Wo
Microsoft.Office.Interop.Word.DocumentEvents_Event, Microsoft.Office.Interop.Word, Version=
Microsoft.Office.Interop.Word.DocumentEvents_NewEventHandler, Microsoft.Office.Interop.Wo
Microsoft.Office.Interop.Word.DocumentEvents_OpenEventHandler, Microsoft.Office.Interop.Wo
Microsoft.Office.Interop.Word.DocumentEvents_SinkHelper, Microsoft.Office.Interop.Word, Vers
Microsoft.Office.Interop.Word.DocumentEvents2, Microsoft.Office.Interop.Word, Version=12.0.0
Microsoft.Office.Interop.Word.DocumentEvents2_BuildingBlockInsertEventHandler, Microsoft.Of
Microsoft.Office.Interop.Word.DocumentEvents2_CloseEventHandler, Microsoft.Office.Interop.W
Microsoft.Office.Interop.Word.DocumentEvents2_ContentControlAfterAddEventHandler, Microso
Microsoft.Office.Interop.Word.DocumentEvents2_ContentControlBeforeContentUpdateEventHar
Microsoft.Office.Interop.Word.DocumentEvents2_ContentControlBeforeDeleteEventHandler, Mic
Microsoft.Office.Interop.Word.DocumentEvents2_ContentControlBeforeStoreUpdateEventHandl
```

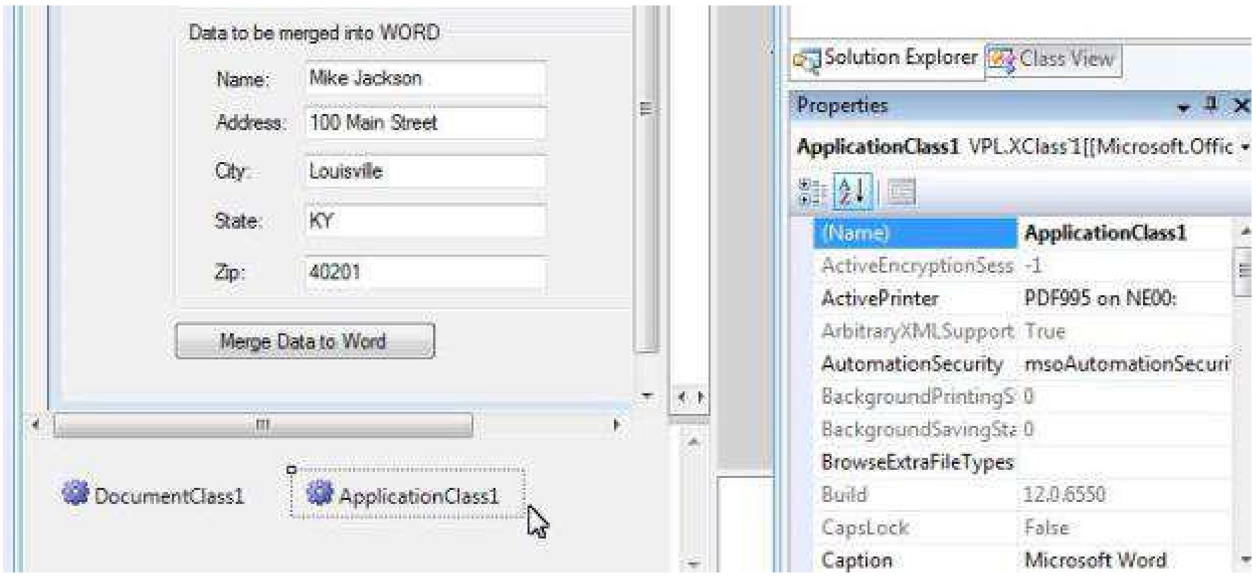
Highlight the components in the list; click Finish button:


Select components
Library Import Wizard
Select a DLL file
Browse Global Assembly Cache
Select components to be placed in the Toolbox
Microsoft.Office.Interop.Word.ApplicationClass
Microsoft.Office.Interop.Word.DocumentClass
Show screen elements only
Back
Finish
Cancel

The components appear in the Toolbox. We may use them just like other components in the Toolbox:



The ApplicationClass component appears in the designer:

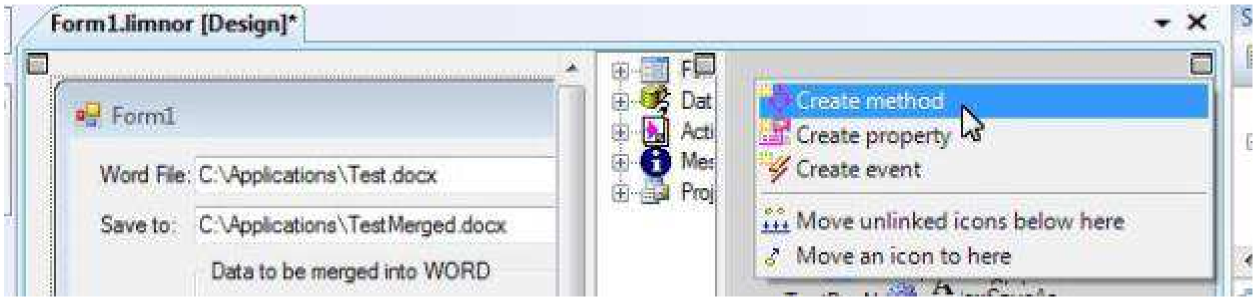

Data to be merged into WORD

Name:	Mike Jackson
Address:	100 Main Street
City:	Louisville
State:	KY
Zip:	40201

(Name)	ApplicationClass1
ActiveEncryptionSess	-1
ActivePrinter	PDF995 on NE00:
ArbitraryXMLSupport	True
AutomationSecurity	msoAutomationSecuri
BackgroundPrintingS	0
BackgroundSavingSta	0
BrowseExtraFileTypes	
Build	12.0.6550
CapsLock	False
Caption	Microsoft Word

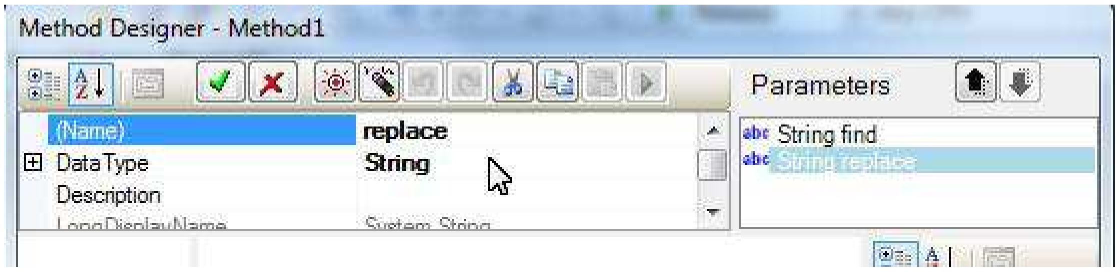
Manipulate Word Document – Search and Replace

Let’s make a method to do search and replace in a Word document.



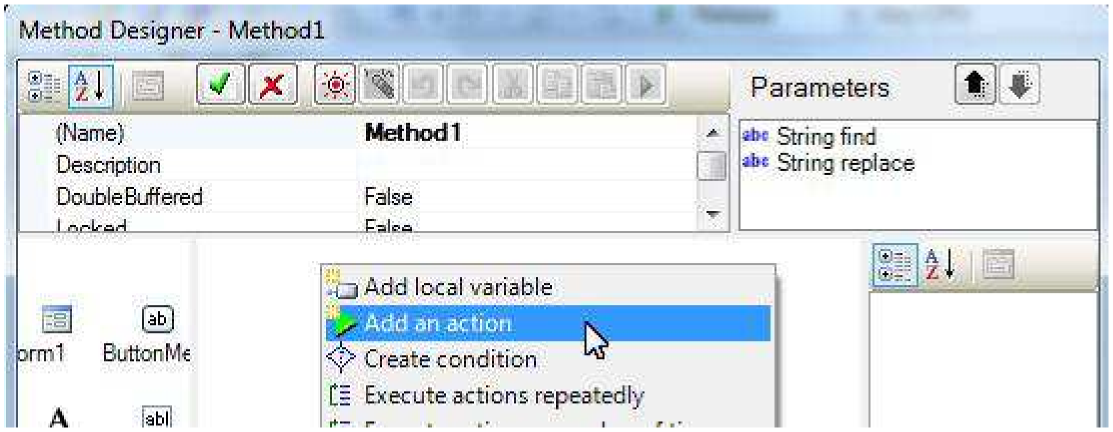
Word File: C:\Applications\Test.docx
Save to: C:\Applications\TestMerged.docx
Data to be merged into WORD

Add two parameters: find and replace

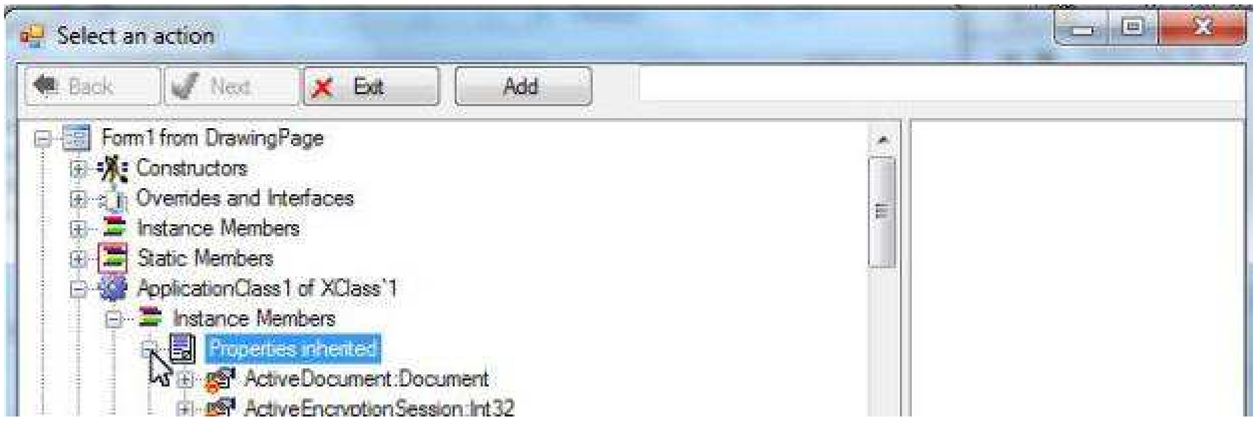


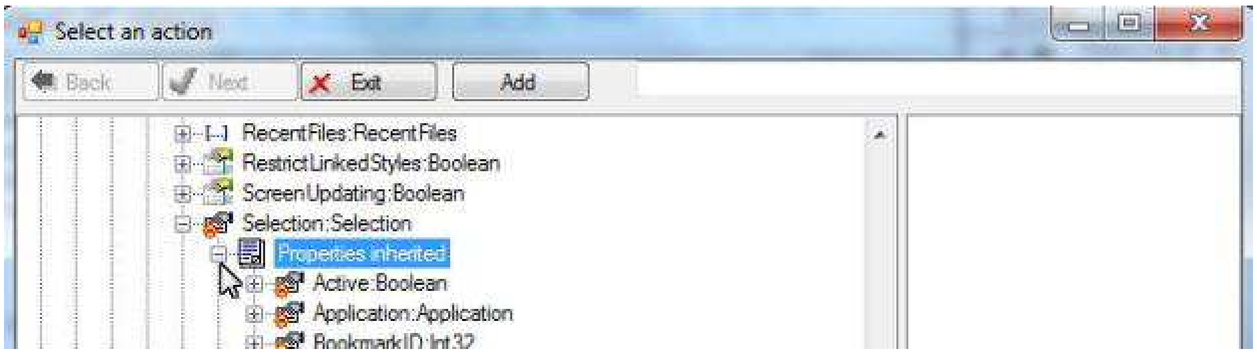
(Name)	replace
DataType	String
Description	

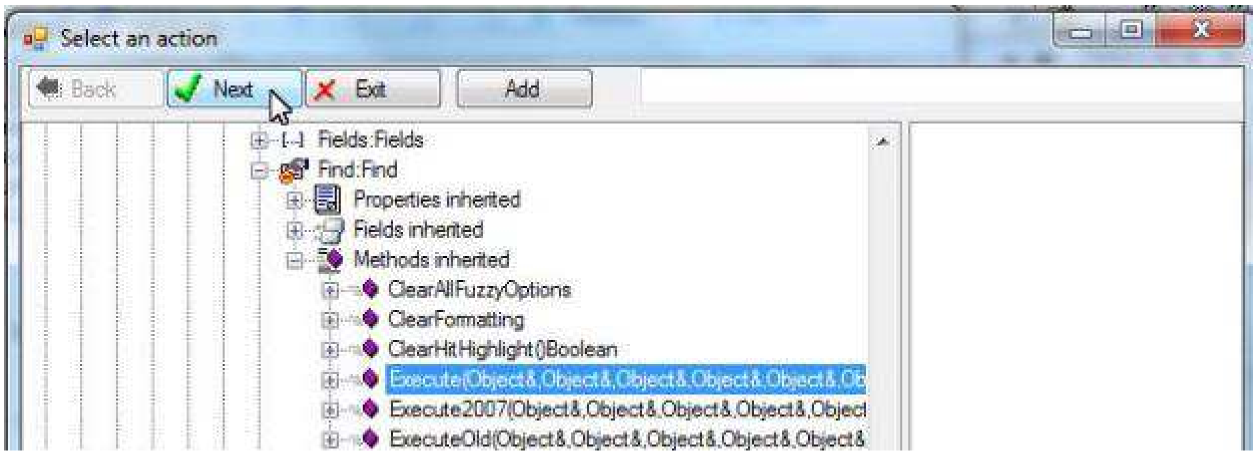
Right-click the Action Pane; choose “Add an action”:



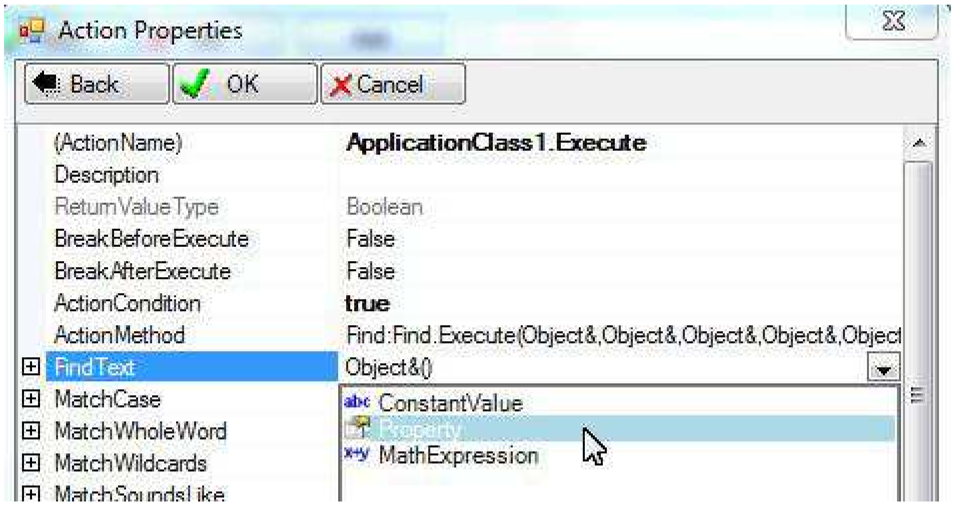
Select Execute method of the Find property of the Selection property of the Word application:





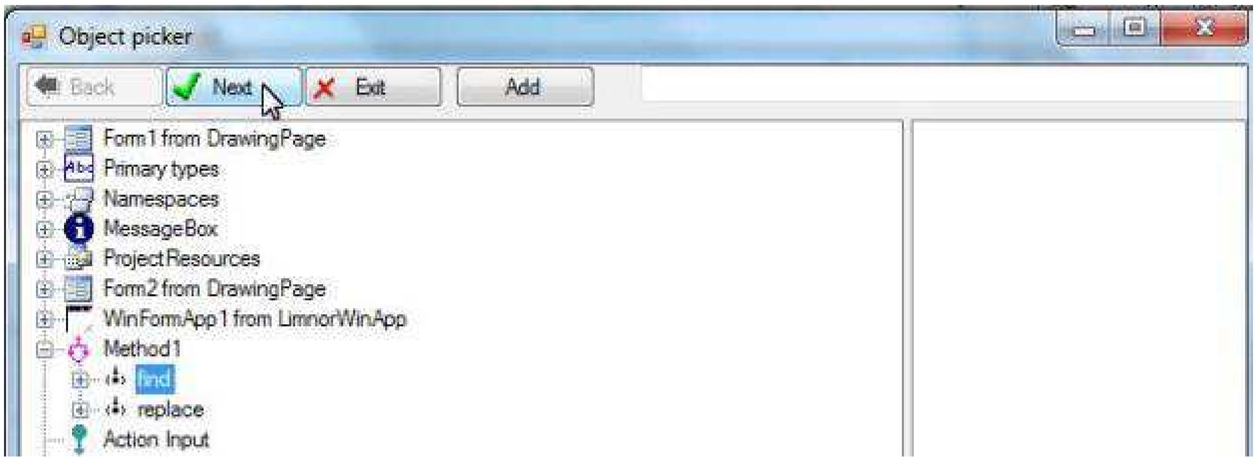

Select an action
Back
Next
Exit
Add
Fields:Fields
Find:Find
Properties inherited
Fields inherited
Methods inherited
ClearAllFuzzyOptions
ClearFormatting
ClearHitHighlight()Boolean
Execute(Object&,Object&,Object&,Object&,Object&,Ob
Execute2007(Object&,Object&,Object&,Object&,Object
ExecuteOld(Object&,Object&,Object&,Object&,Object&

For "FindText" of the action, select "Property":

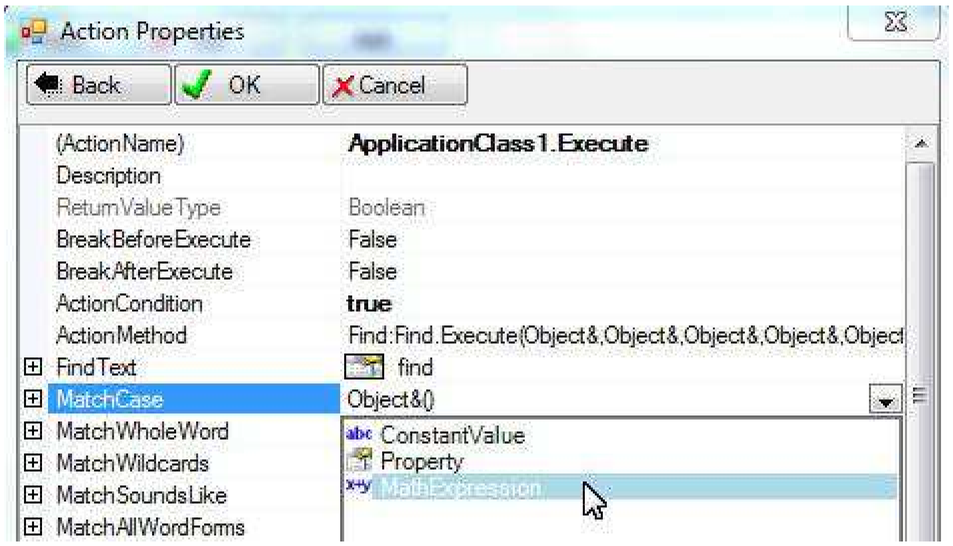


(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object
FindText	Object&()
MatchCase	
MatchWholeWord	
MatchWildcards	
MatchSoundsLike	

Select "find" parameter of the method; click "Next":

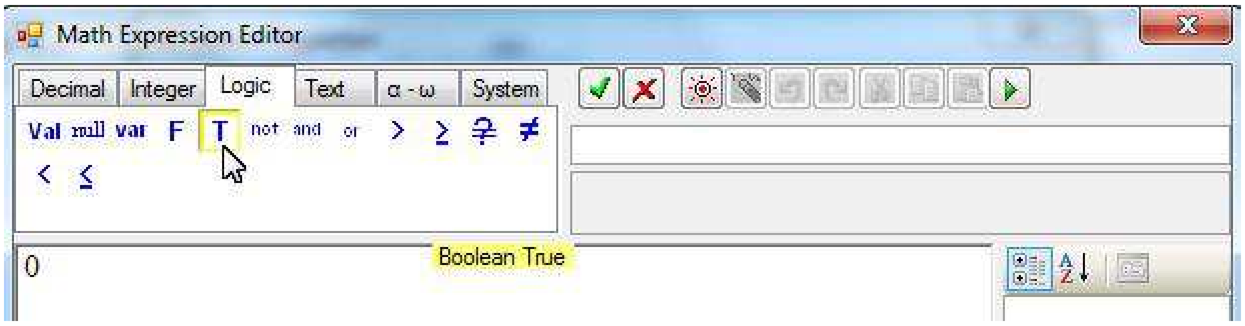


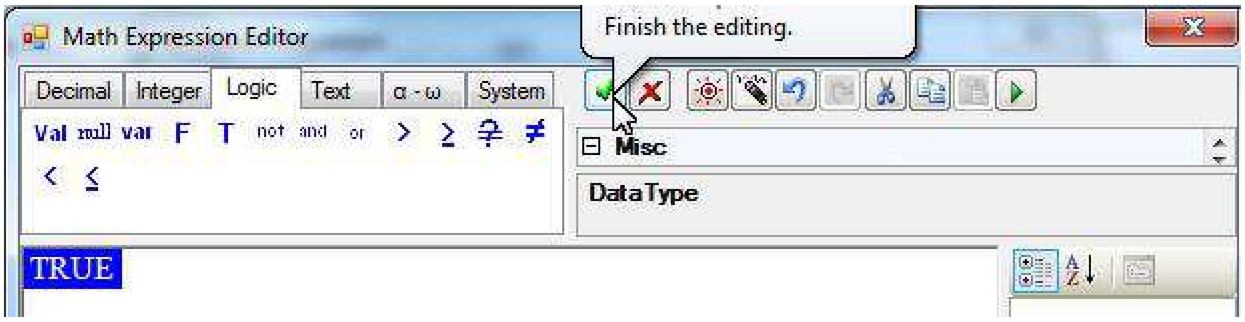
For "MatchCase", select "MathExpression":



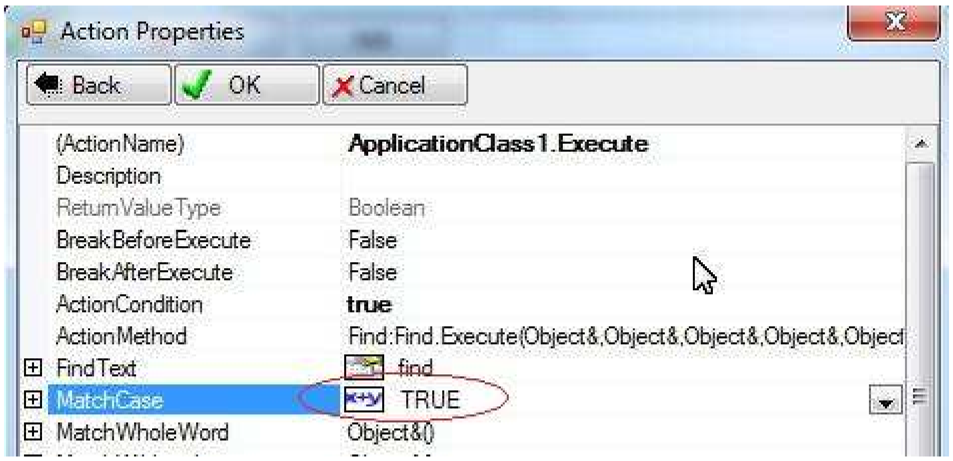
(ActionName)	**ApplicationClass1.Execute**
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object
FindText	find
MatchCase	Object&()
MatchWholeWord	
MatchWildcards	
MatchSoundsLike	
MatchAllWordForms	

Select logic True:



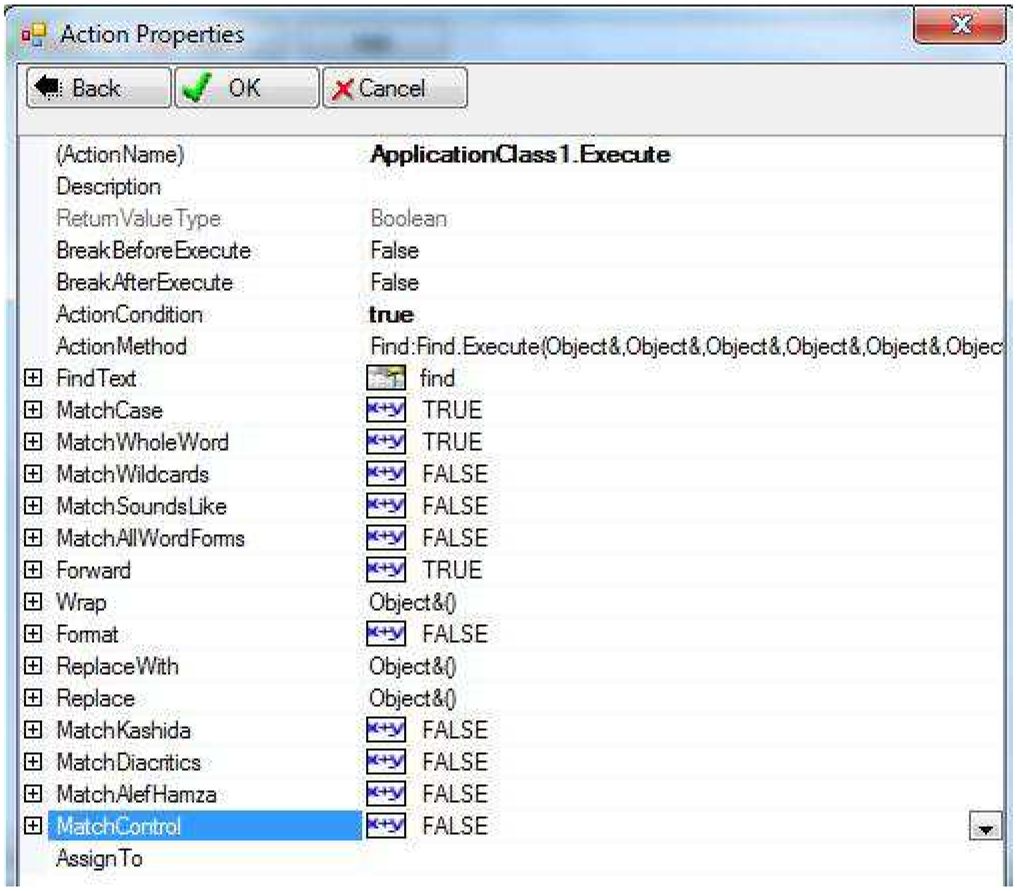


"True" appears as the value for "MatchCase":



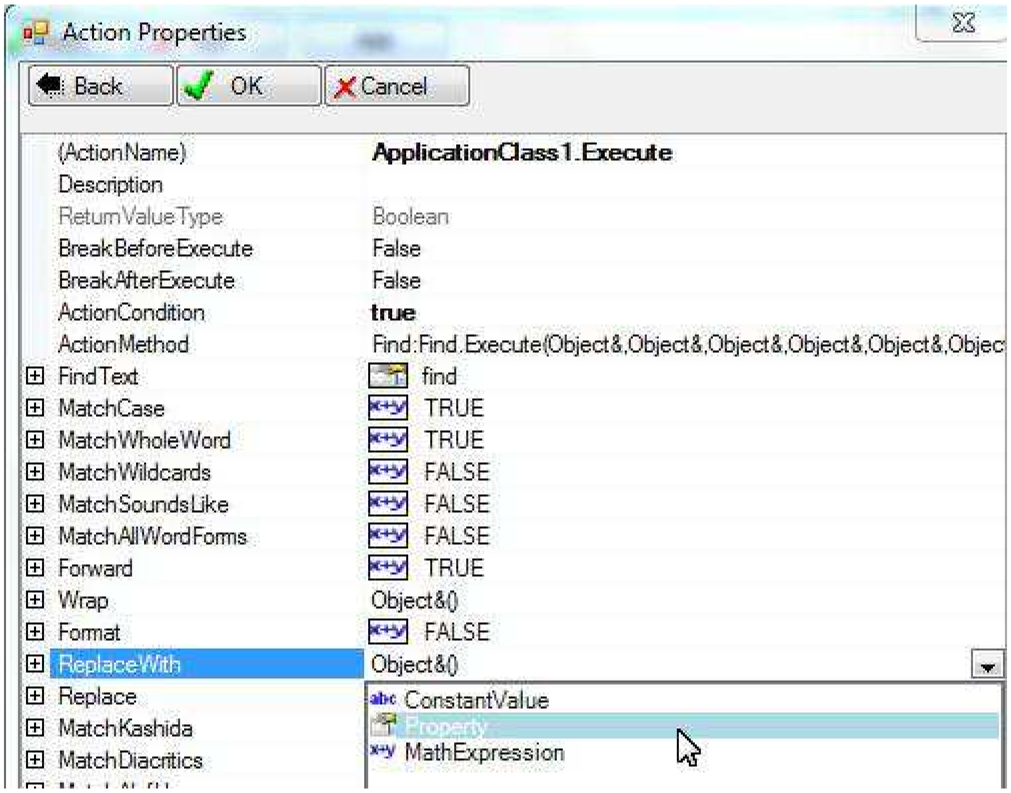
(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object
FindText	find
MatchCase	TRUE
MatchWholeWord	Object&()

Set other parameters use True or False:



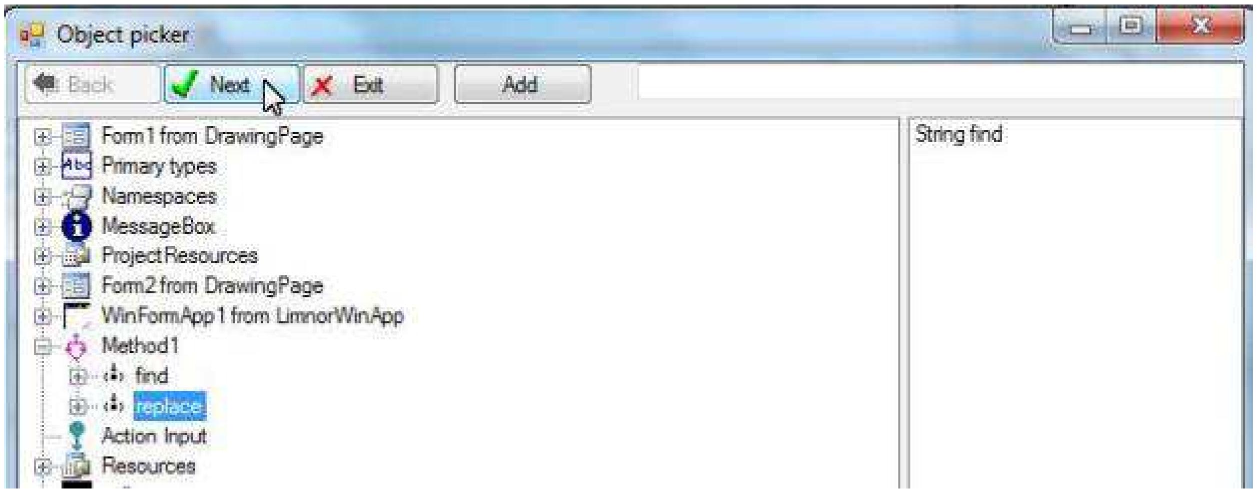
(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object&,Objec
FindText	find
MatchCase	TRUE
MatchWholeWord	TRUE
MatchWildcards	FALSE
MatchSoundsLike	FALSE
MatchAllWordForms	FALSE
Forward	TRUE
Wrap	Object&()
Format	FALSE
ReplaceWith	Object&()
Replace	Object&()
MatchKashida	FALSE
MatchDiacritics	FALSE
MatchAlefHamza	FALSE
MatchControl	FALSE
AssignTo	

For “ReplaceWith” of the action, select “Property”:

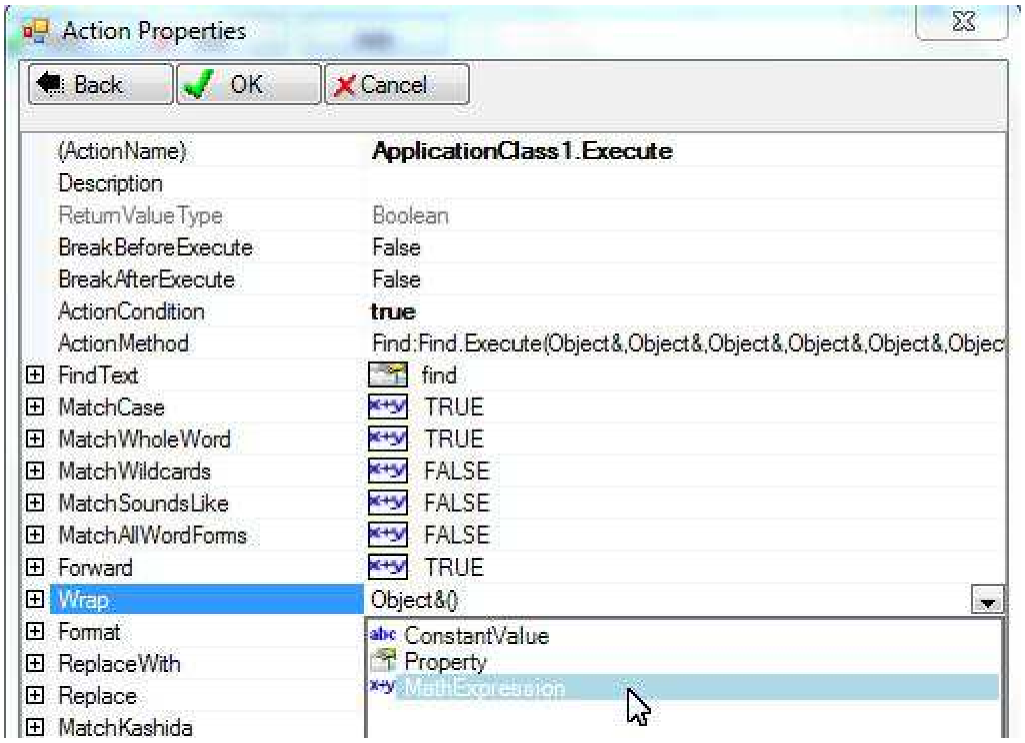


(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object&,Objec
FindText	find
MatchCase	TRUE
MatchWholeWord	TRUE
MatchWildcards	FALSE
MatchSoundsLike	FALSE
MatchAllWordForms	FALSE
Forward	TRUE
Wrap	Object&()
Format	FALSE
ReplaceWith	Object&()
Replace	
MatchKashida	
MatchDiacritics	

Select “replace” parameter of the method:

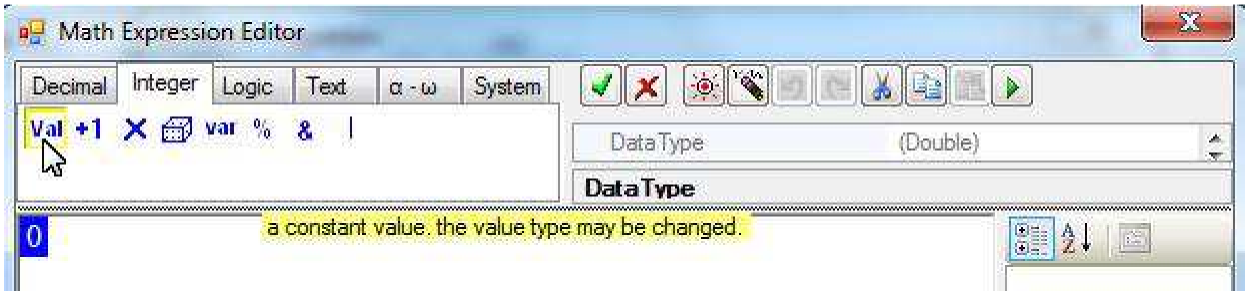


For “wrap”, select “MathExpression”:



(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object&,Objec
FindText	find
MatchCase	TRUE
MatchWholeWord	TRUE
MatchWildcards	FALSE
MatchSoundsLike	FALSE
MatchAllWordForms	FALSE
Forward	TRUE
Wrap	Object&()
Format	
ReplaceWith	
Replace	
MatchKashida	

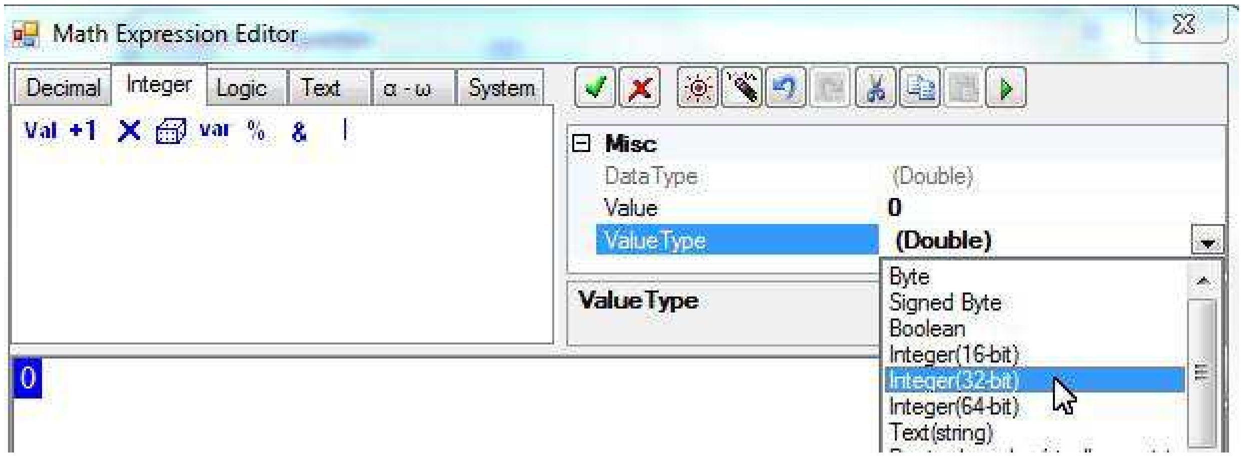
Click integer constant icon:



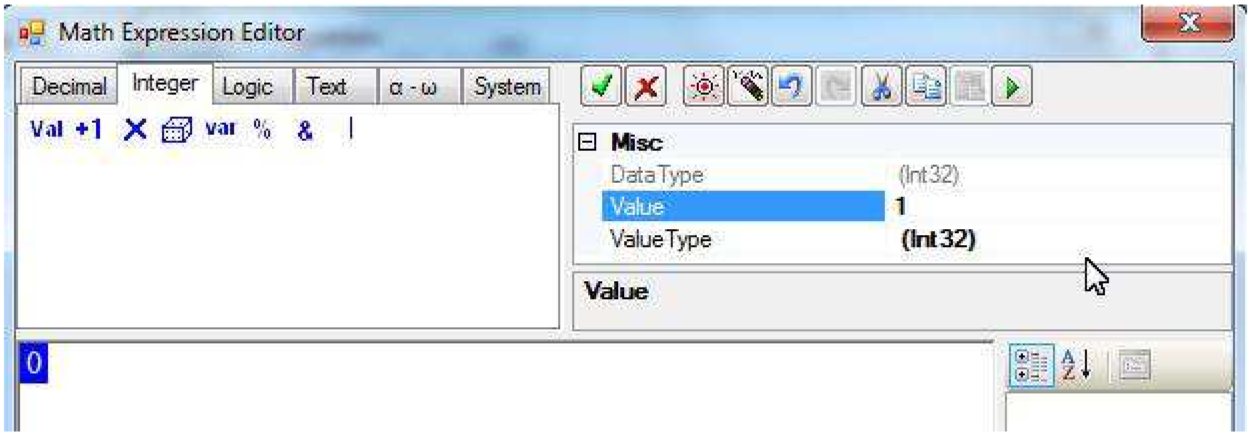
DataType (Double)

a constant value. the value type may be changed.

Switch ValueType to 32-bit integer:



Misc	
DataType	(Double)
Value	0
ValueType	(Double)

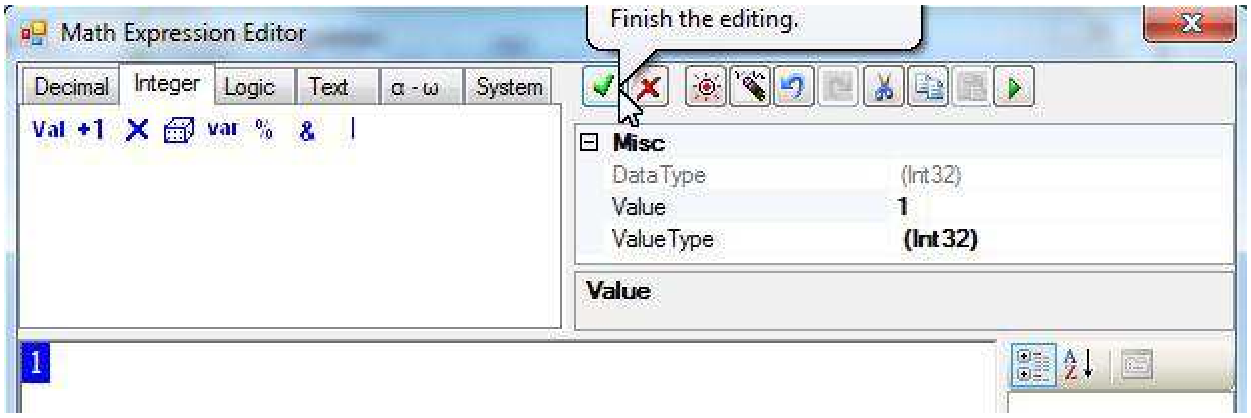


Misc

DataType	(Int32)
Value	1
ValueType	(Int32)

Value

0

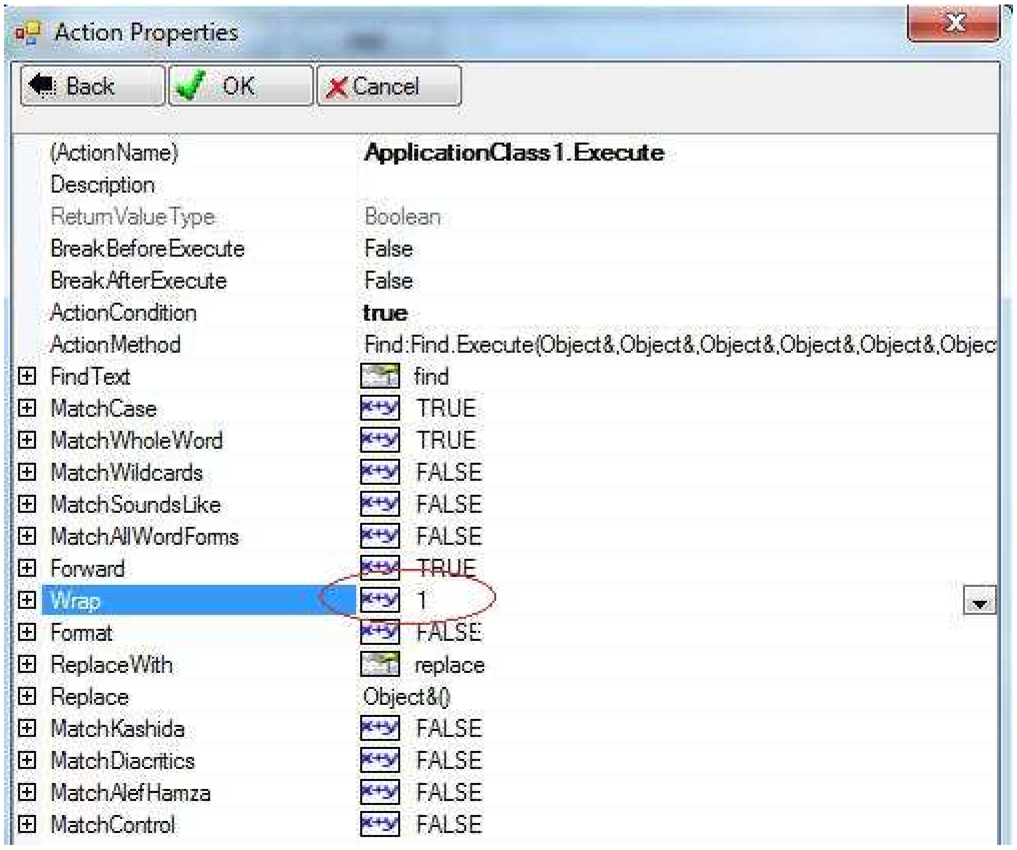


Misc

DataType	(Int32)
Value	1
ValueType	(Int32)

Value

1

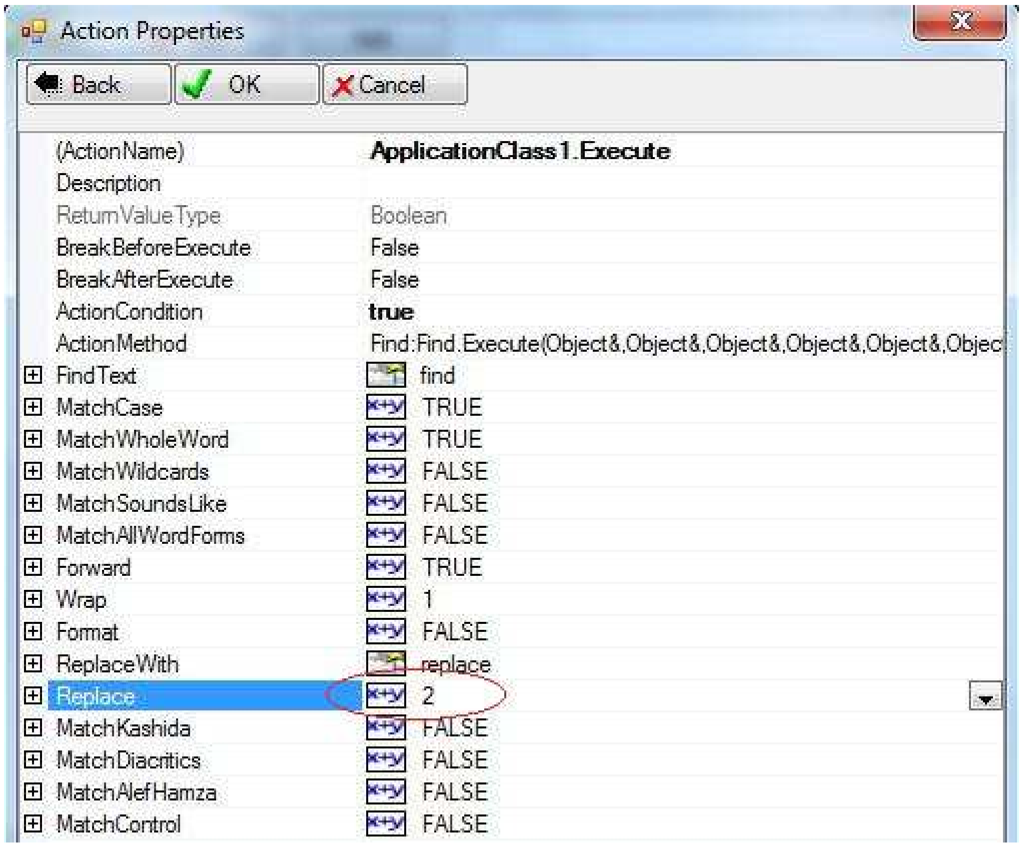


Action Properties

Back | OK | Cancel

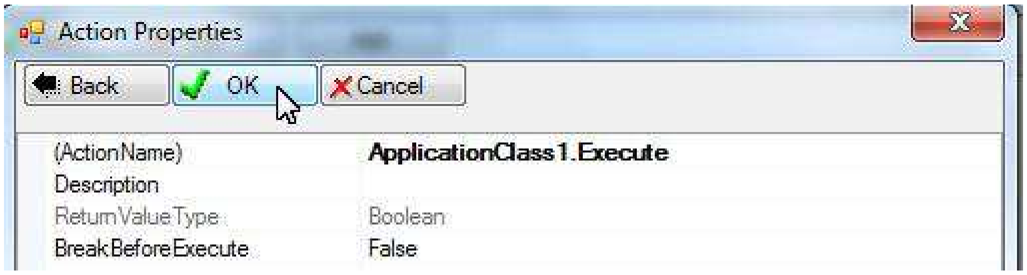
(ActionName)	**ApplicationClass1.Execute**
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object&,Objec
FindText	find
MatchCase	TRUE
MatchWholeWord	TRUE
MatchWildcards	FALSE
MatchSoundsLike	FALSE
MatchAllWordForms	FALSE
Forward	TRUE
Wrap	1
Format	FALSE
ReplaceWith	replace
Replace	Object&()
MatchKashida	FALSE
MatchDiacritics	FALSE
MatchAlefHamza	FALSE
MatchControl	FALSE

Set “Replace” to 2:



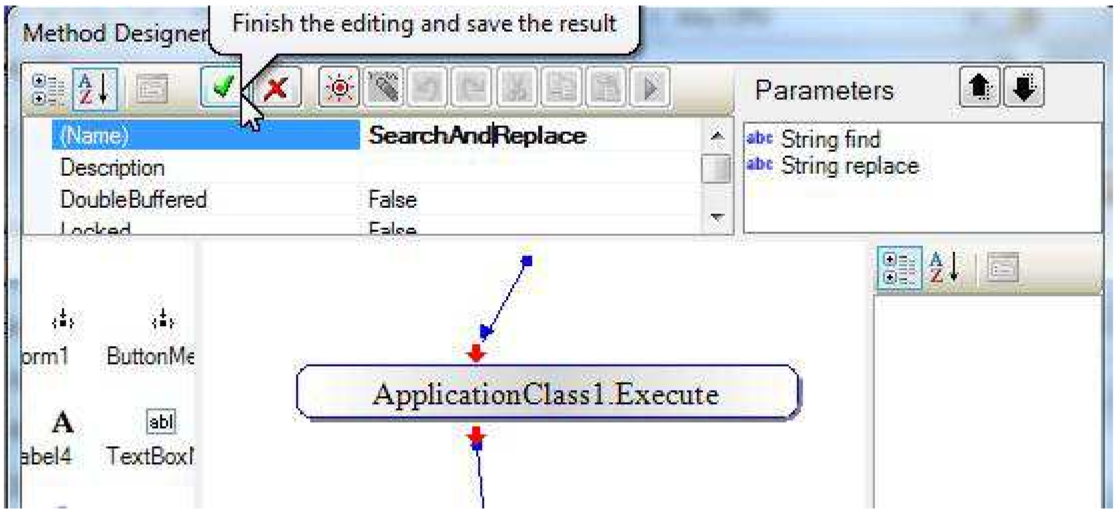
(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Find:Find.Execute(Object&,Object&,Object&,Object&,Object&,Objec
FindText	find
MatchCase	TRUE
MatchWholeWord	TRUE
MatchWildcards	FALSE
MatchSoundsLike	FALSE
MatchAllWordForms	FALSE
Forward	TRUE
Wrap	1
Format	FALSE
ReplaceWith	replace
Replace	2
MatchKashida	FALSE
MatchDiacritics	FALSE
MatchAlefHamza	FALSE
MatchControl	FALSE

Click OK to finish creating this action:



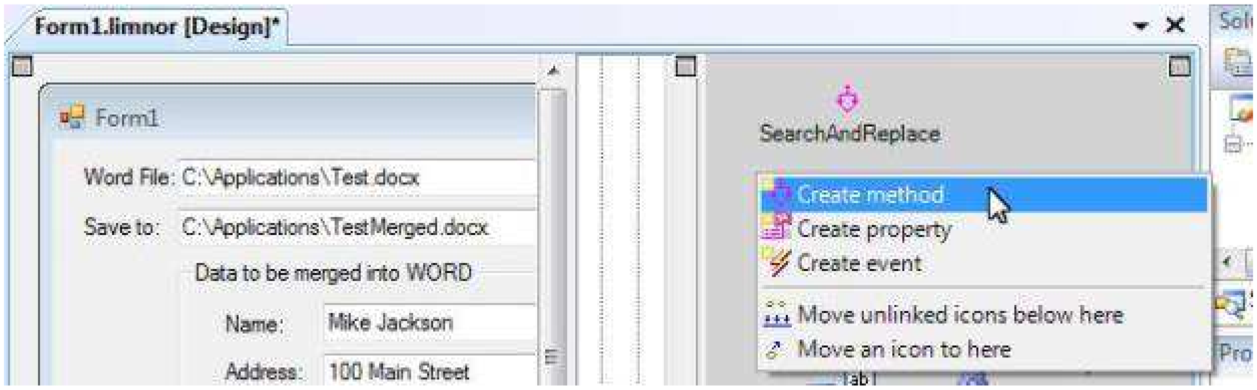
(ActionName)	ApplicationClass1.Execute
Description	
ReturnValueType	Boolean
BreakBeforeExecute	False

The action appears in the Action Pane. For this method we just need this one action. Rename the method to SearchAndReplace

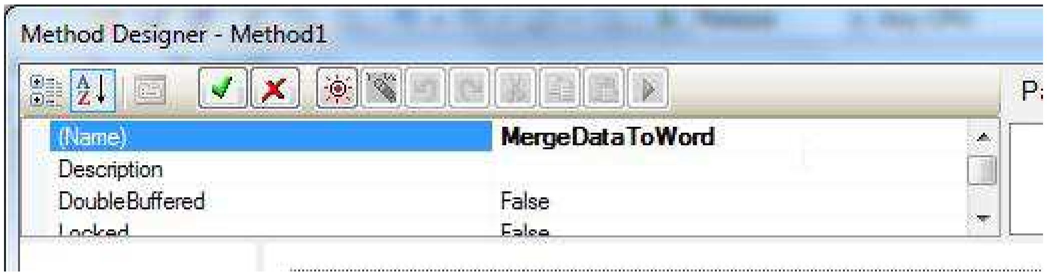


Manipulate Word Document – Modify and Save

Let’s create a method to do following actions: open a Word document; search and replace using text boxes values; save to a new file; close Word document.

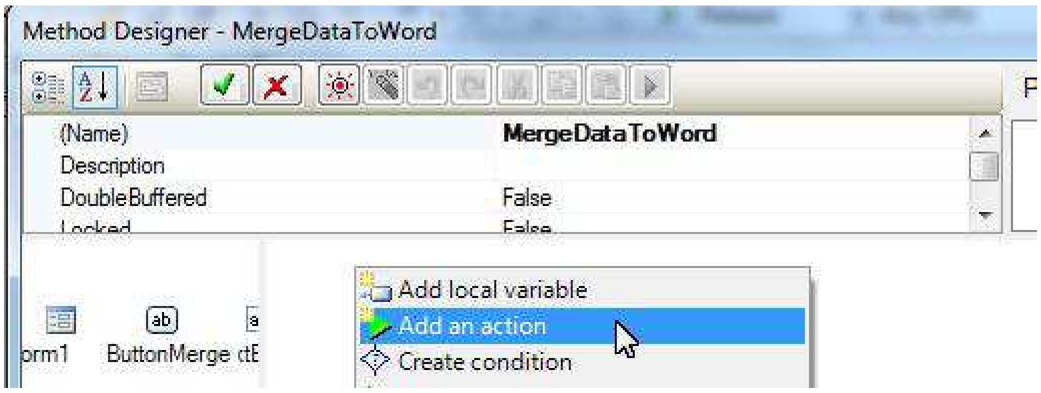


Rename the method to MergeDataToWord:

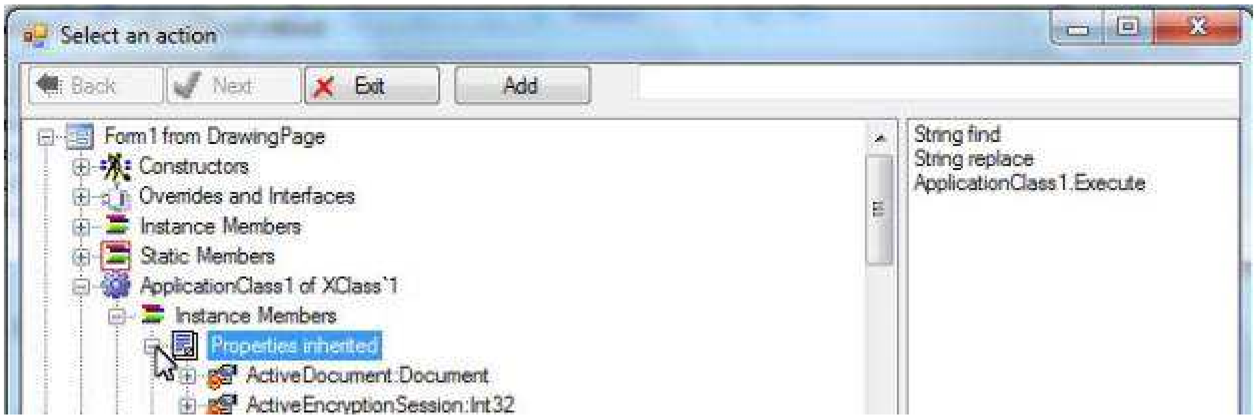


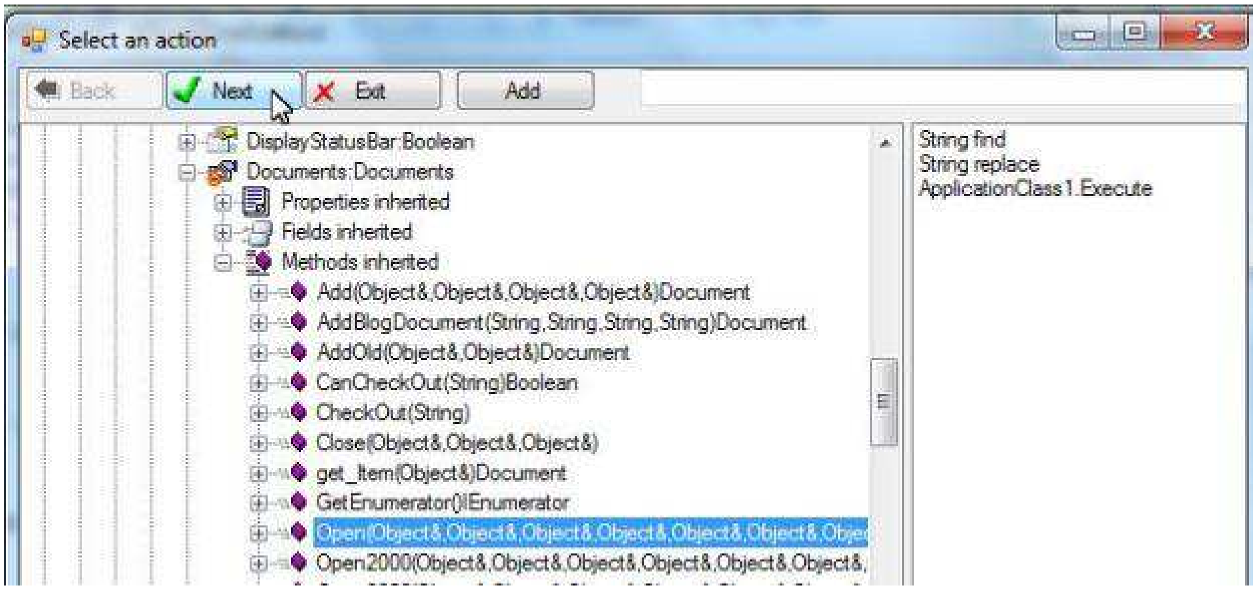
The Word Application has a Documents property. The Documents property has an Open method. We may use it to open a Word document.

Right-click the Action-Pane, choose “Add an action”:

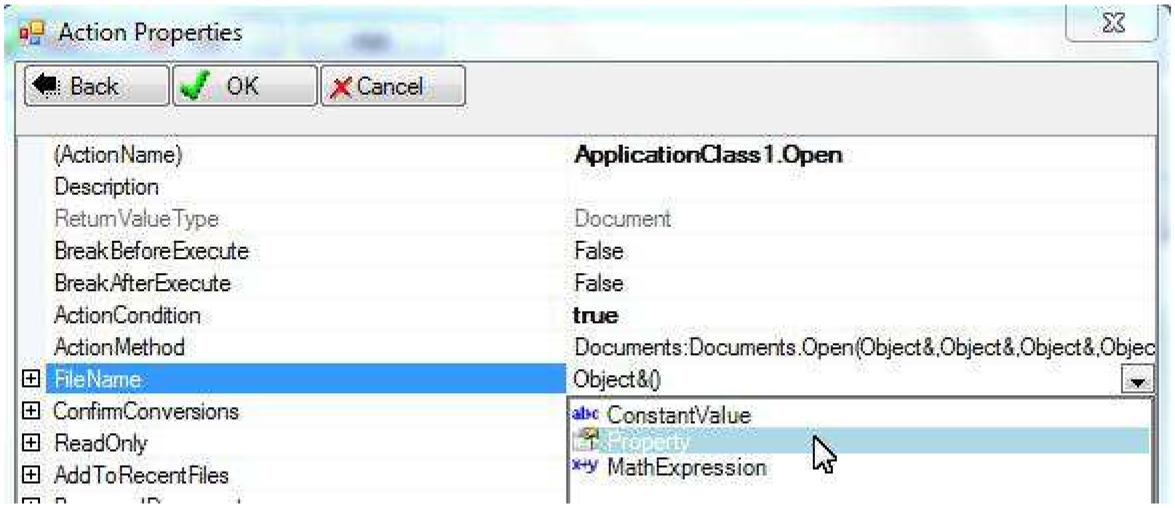


Select the Open method of the Documents:



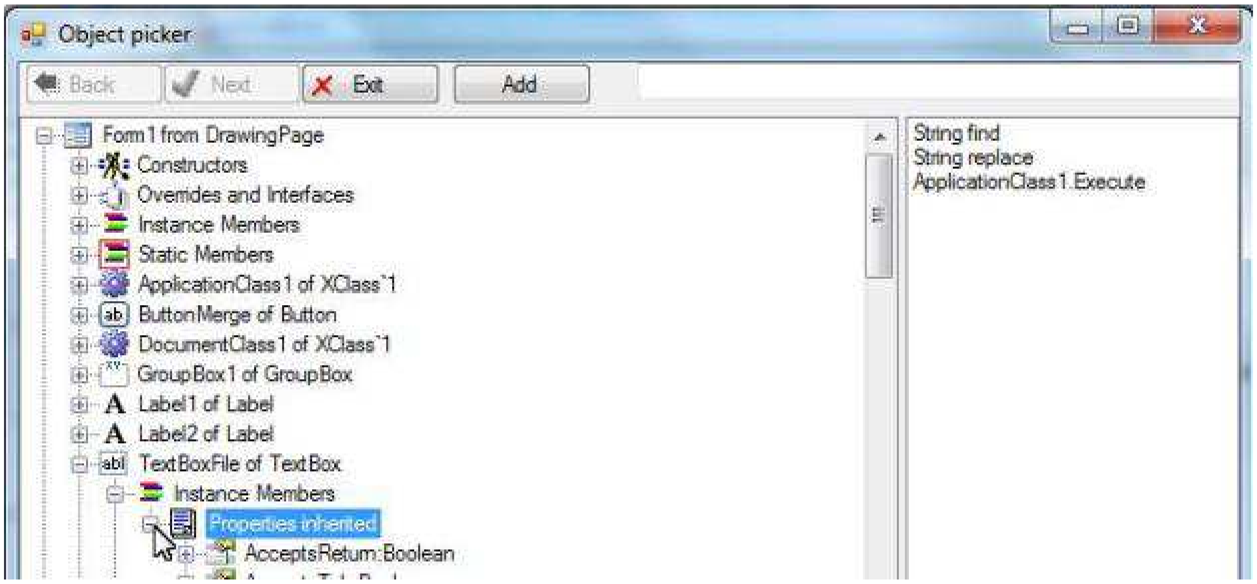

Select an action
Back
Next
Exit
Add
DisplayStatusBar:Boolean
Documents:Documents
Properties inherited
Fields inherited
Methods inherited
Add(Object&,Object&,Object&,Object&)Document
AddBlogDocument(String,String,String,String)Document
AddOld(Object&,Object&)Document
CanCheckOut(String)Boolean
CheckOut(String)
Close(Object&,Object&,Object&)
get_Item(Object&)Document
GetEnumerator()IEnumerator
Open(Object&,Object&,Object&,Object&,Object&,Object&,Objec
Open2000(Object&,Object&,Object&,Object&,Object&,Object&,
String find
String replace
ApplicationClass1.Execute

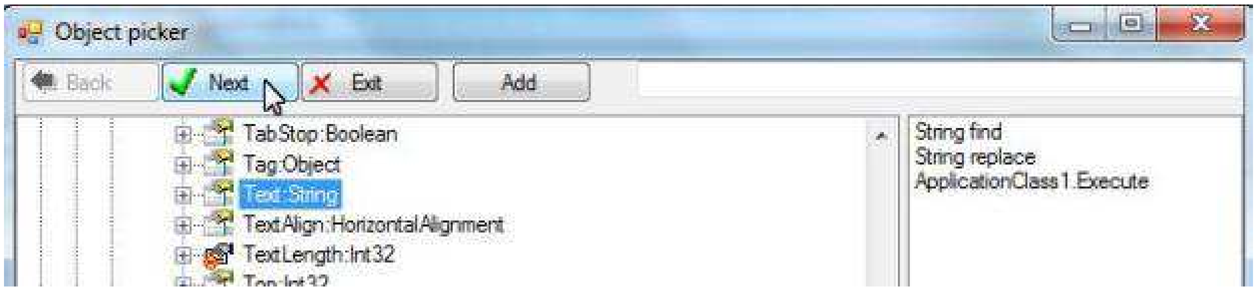
Choose “Property” for “FileName”:



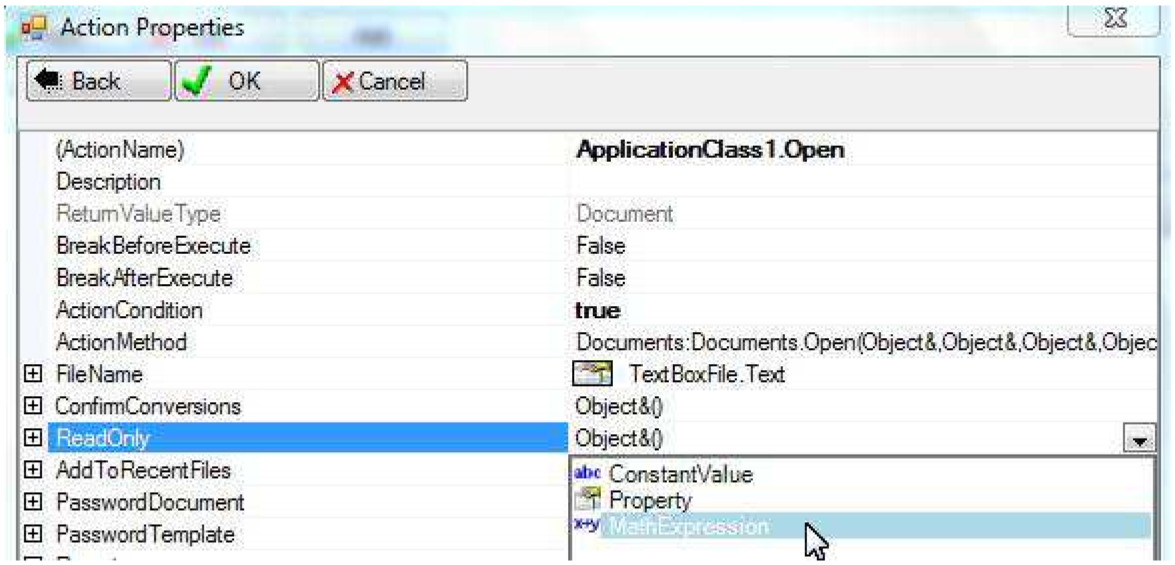
(ActionName)	ApplicationClass1.Open
Description	
ReturnValueType	Document
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Documents:Documents.Open(Object&,Object&,Object&,Objec
FileName	Object&()
ConfirmConversions	
ReadOnly	
AddToRecentFiles	

Select the Text property of the text box for the Word document:

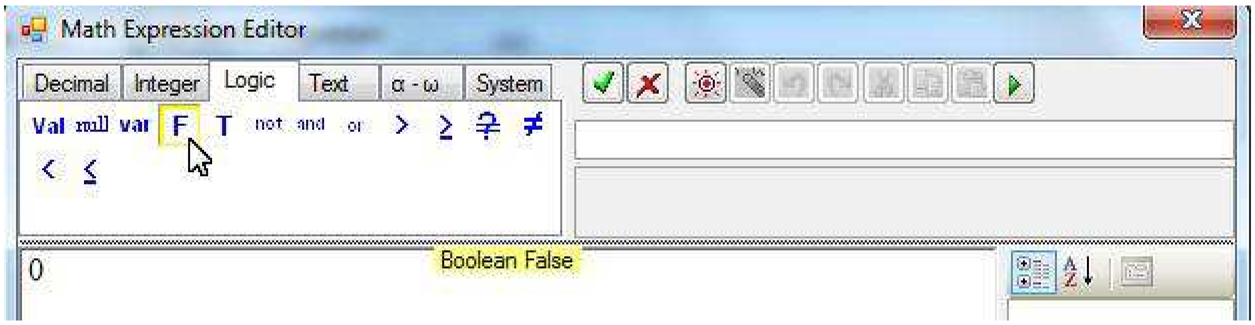




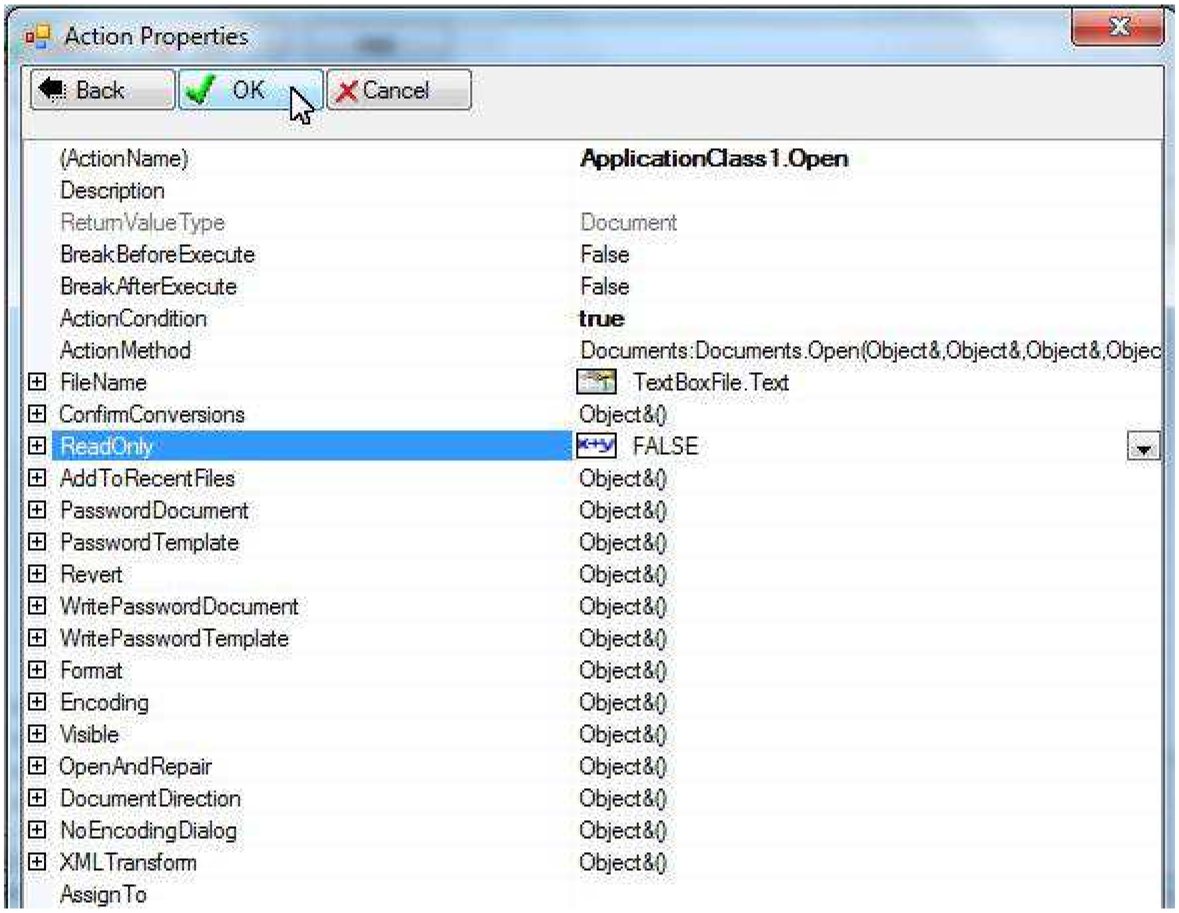
For “ReadOnly”, use “MathExpression” to set it to False:



(ActionName)	**ApplicationClass1.Open**
Description	
ReturnValueType	Document
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Documents:Documents.Open(Object&,Object&,Object&,Objec
FileName	TextBoxFile.Text
ConfirmConversions	Object&()
ReadOnly	Object&()
AddToRecentFiles	
PasswordDocument	
PasswordTemplate	

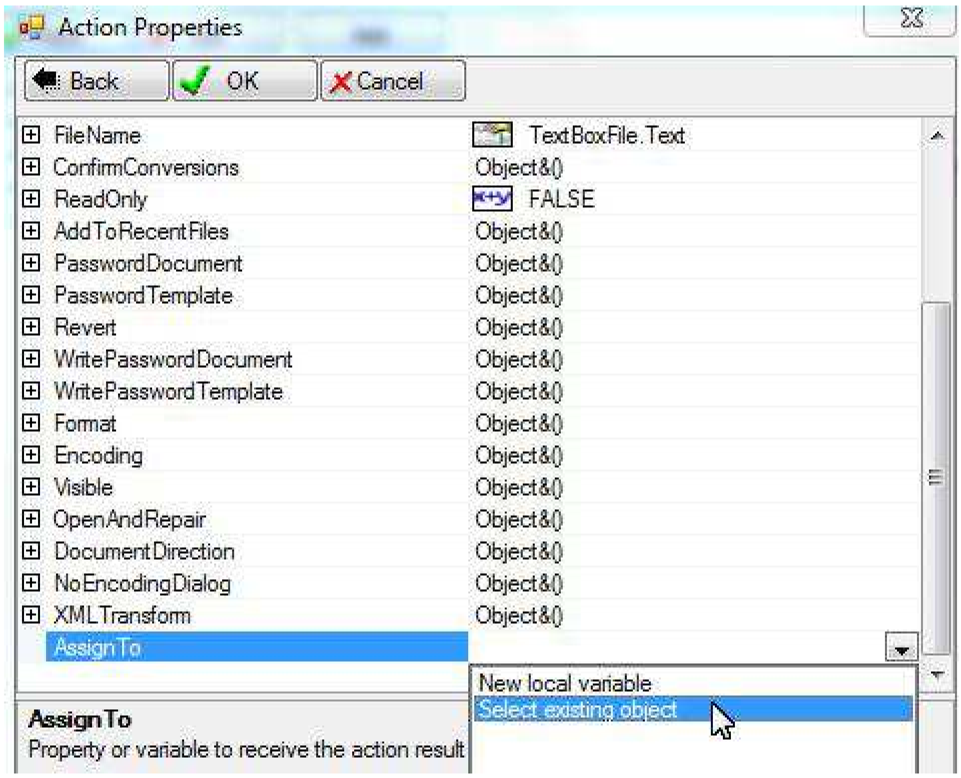


```
0
```



(ActionName)	ApplicationClass1.Open
Description	
ReturnValueType	Document
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Documents:Documents.Open(Object&,Object&,Object&,Objec
FileName	TextBoxFile.Text
ConfirmConversions	Object&()
ReadOnly	FALSE
AddToRecentFiles	Object&()
PasswordDocument	Object&()
PasswordTemplate	Object&()
Revert	Object&()
WritePasswordDocument	Object&()
WritePasswordTemplate	Object&()
Format	Object&()
Encoding	Object&()
Visible	Object&()
OpenAndRepair	Object&()
DocumentDirection	Object&()
NoEncodingDialog	Object&()
XMLTransform	Object&()
AssignTo	

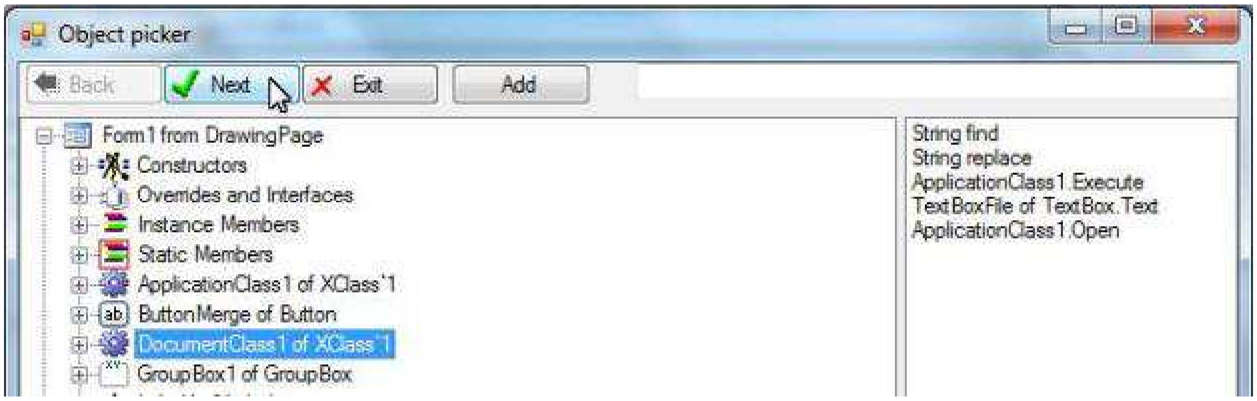
This action returns a Word document object. Set its “AssignTo” to point to a variable to receive the object. We may choose “New local variable” to create a new local variable to receive the object. In our sample, since we already created a Word document, we select “Select existing object” to use it.

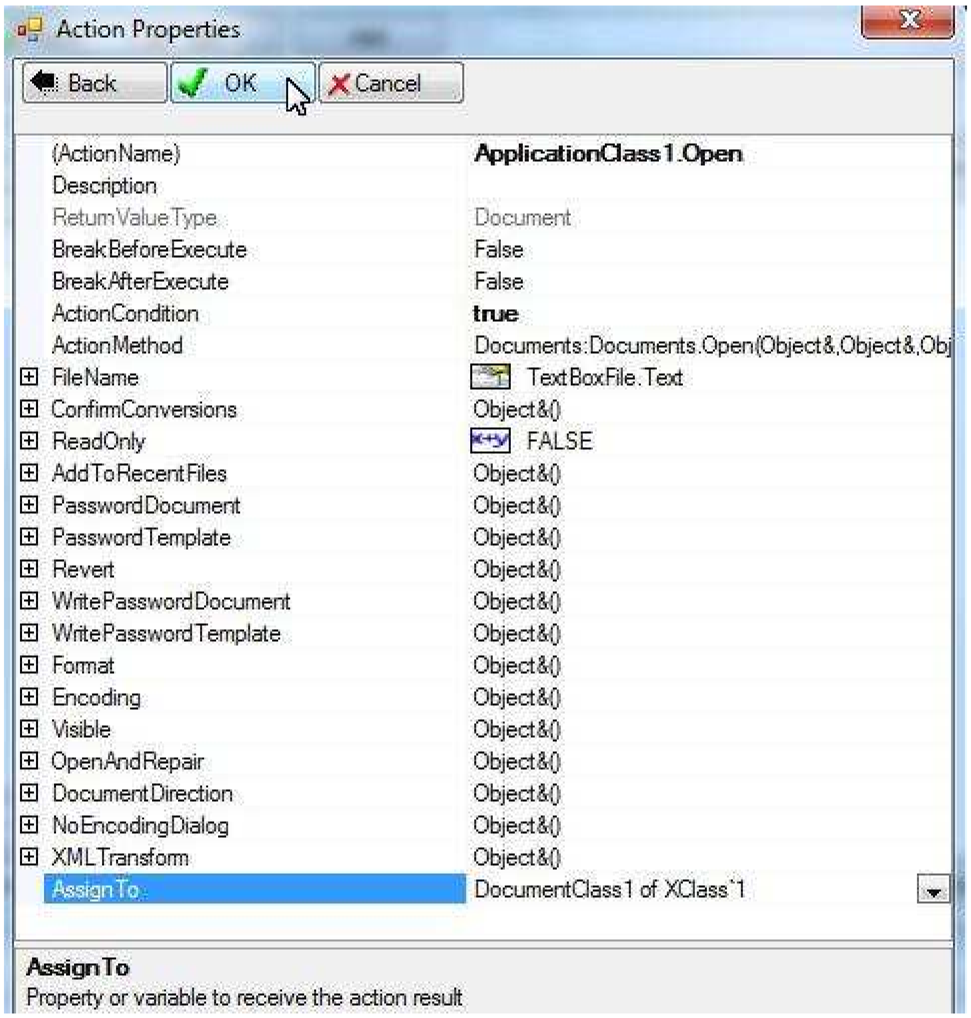


FileName	TextBoxFile.Text
ConfirmConversions	Object&()
ReadOnly	FALSE
AddToRecentFiles	Object&()
PasswordDocument	Object&()
PasswordTemplate	Object&()
Revert	Object&()
WritePasswordDocument	Object&()
WritePasswordTemplate	Object&()
Format	Object&()
Encoding	Object&()
Visible	Object&()
OpenAndRepair	Object&()
DocumentDirection	Object&()
NoEncodingDialog	Object&()
XMLTransform	Object&()
AssignTo	

AssignTo
Property or variable to receive the action result

Select the document object:

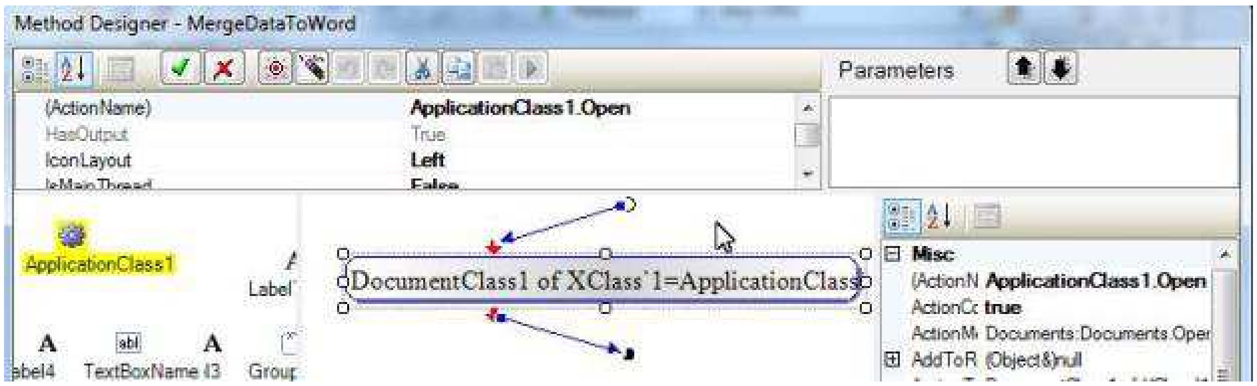




(ActionName)	**ApplicationClass1.Open**
Description	
ReturnValueType	Document
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	Documents:Documents.Open(Object&,Object&,Obj
⊞ FileName	TextBoxFile.Text
⊞ ConfirmConversions	Object&()
⊞ ReadOnly	FALSE
⊞ AddToRecentFiles	Object&()
⊞ PasswordDocument	Object&()
⊞ PasswordTemplate	Object&()
⊞ Revert	Object&()
⊞ WritePasswordDocument	Object&()
⊞ WritePasswordTemplate	Object&()
⊞ Format	Object&()
⊞ Encoding	Object&()
⊞ Visible	Object&()
⊞ OpenAndRepair	Object&()
⊞ DocumentDirection	Object&()
⊞ NoEncodingDialog	Object&()
⊞ XMLTransform	Object&()
AssignTo	DocumentClass1 of XClass`1

AssignTo
Property or variable to receive the action result

The action appears in the Action Pane:

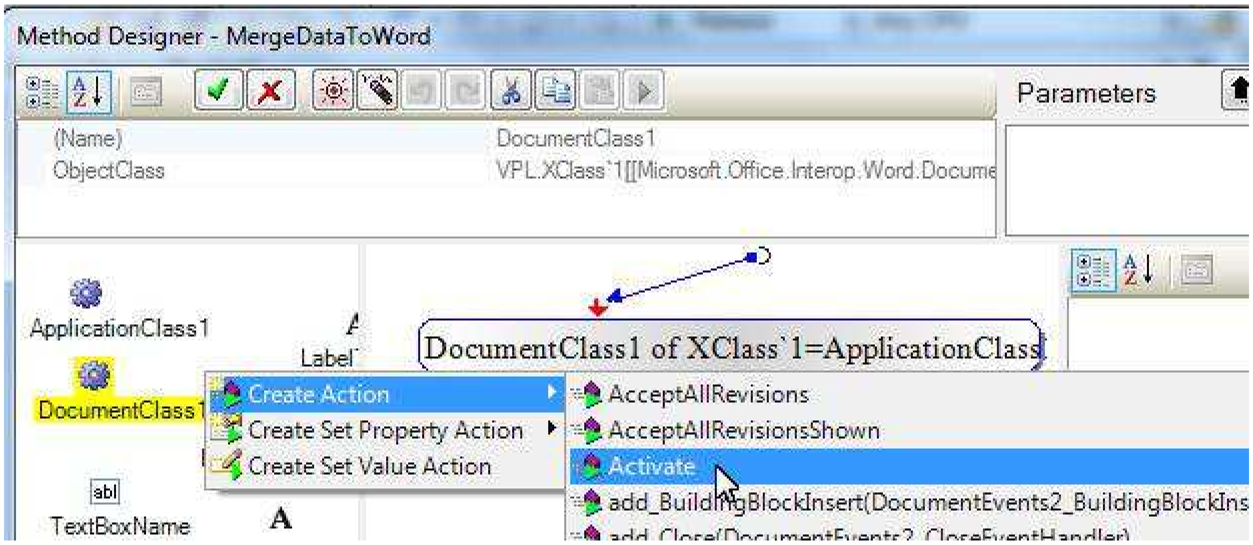


(ActionName)	**ApplicationClass1.Open**
HasOutput	True
IconLayout	**Left**
IsMainThread	**False**

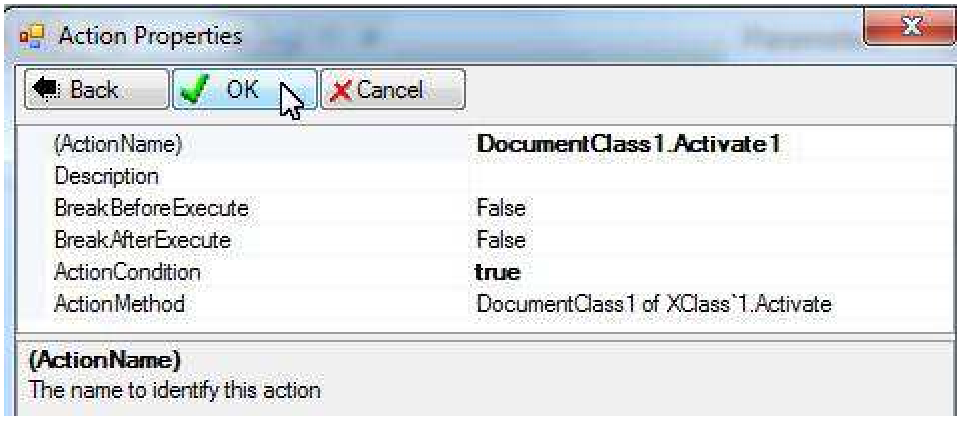
DocumentClass1 of XClass`1=ApplicationClass

(ActionN	**ApplicationClass1.Open**
ActionCc	**true**
ActionM	Documents:Documents.Oper
⊞ AddToR	(Object&)null

Create an action to activate the newly opened document. Right-click the document class icon; choose "Create Action"; choose "Activate" method:


Method Designer - MergeDataToWord
Parameters
(Name)
DocumentClass1
ObjectClass
VPL.XClass`1[[Microsoft.Office.Interop.Word.Docume
ApplicationClass1
Label
DocumentClass1
TextBoxName
DocumentClass1 of XClass`1=ApplicationClass
Create Action
Create Set Property Action
Create Set Value Action
AcceptAllRevisions
AcceptAllRevisionsShown
Activate
add_BuildingBlockInsert(DocumentEvents2_BuildingBlockIns
add_Close(DocumentEvents2_CloseEventHandler)

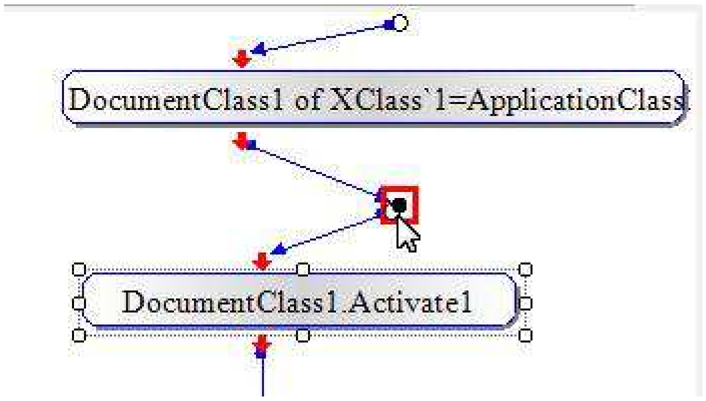
Click OK:



(ActionName)	**DocumentClass1.Activate1**
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	DocumentClass1 of XClass`1.Activate

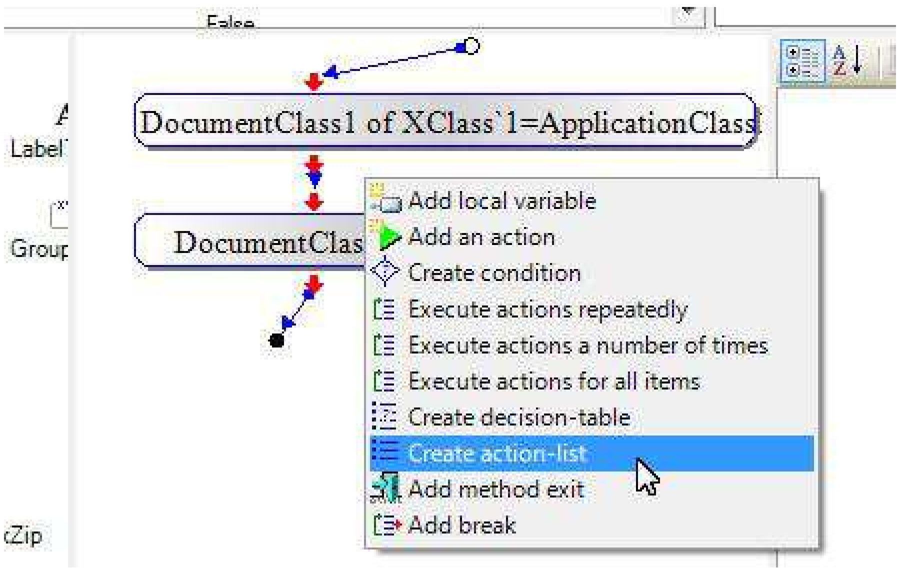
(ActionName)
The name to identify this action

Link the two actions:

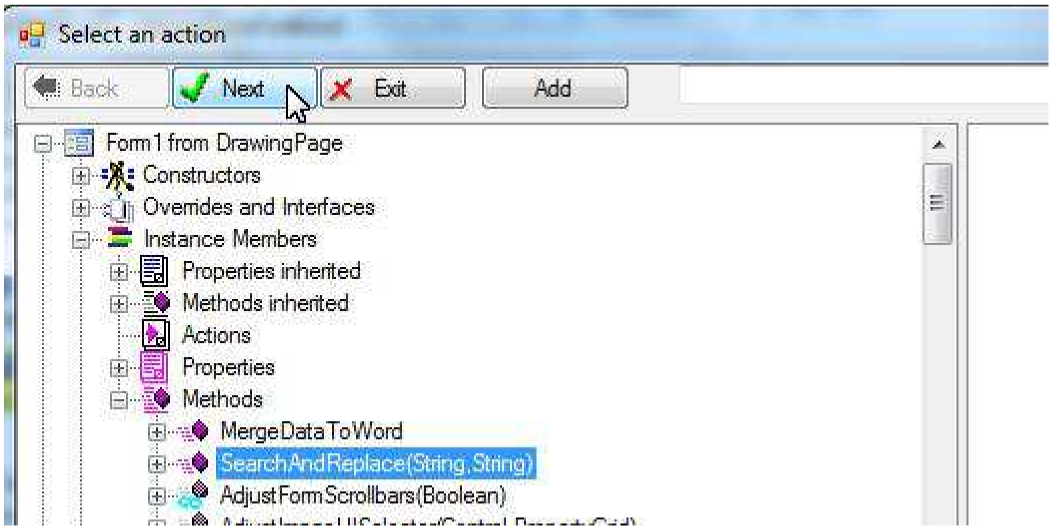


We now may call the SearchAndReplace method to modify the contents of the document. Because we want to call this method many times by creating many actions, instead of showing all actions individually in the Action-Pane, we may add an action-list to include all such actions.

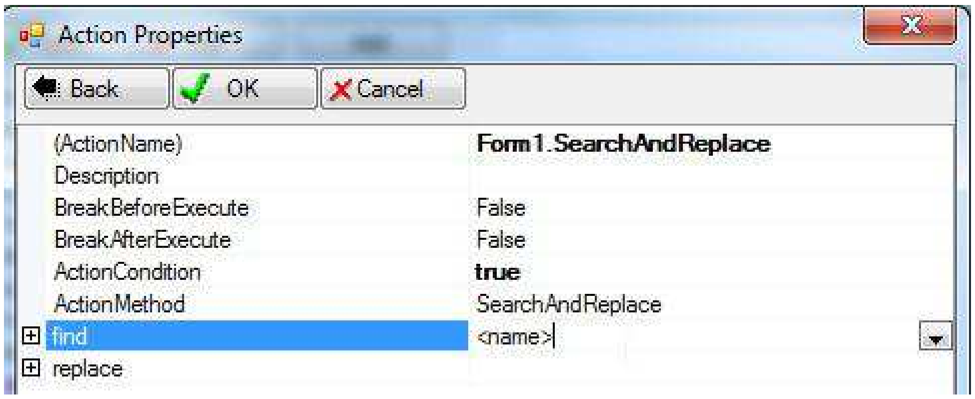
Right-click the Action-Pane; choose "Add action-list":



It asks us to select the actions to be included in the action-list. We haven't created the actions to be included in the action-list. We may create the first action now. Choose "SearchAndReplace" method; click "Next":

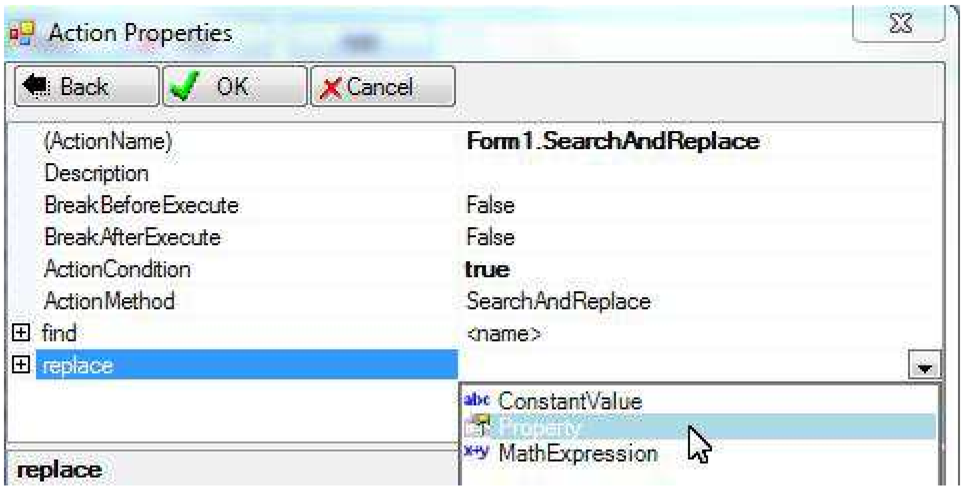


For the "find" parameter of the action, type the text to be replaced. For this action, type "<name>":



(ActionName)	**Form1.SearchAndReplace**
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	SearchAndReplace
⊞ find	<name>
⊞ replace	

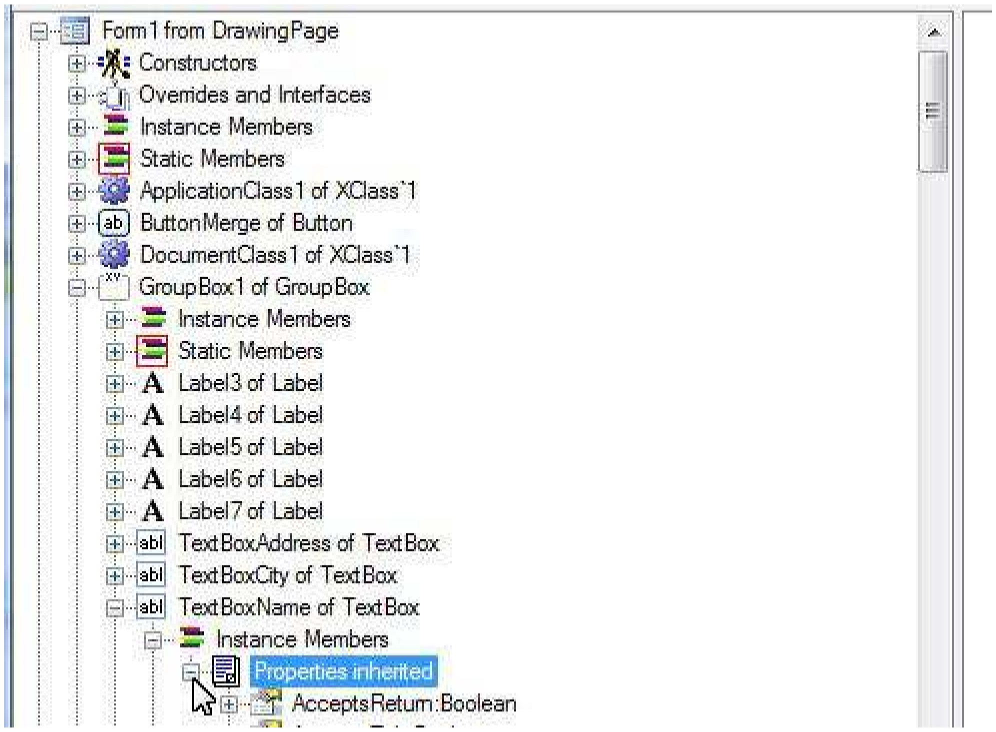
For parameter “replace”, select “Property” because we want to use value from a text box:

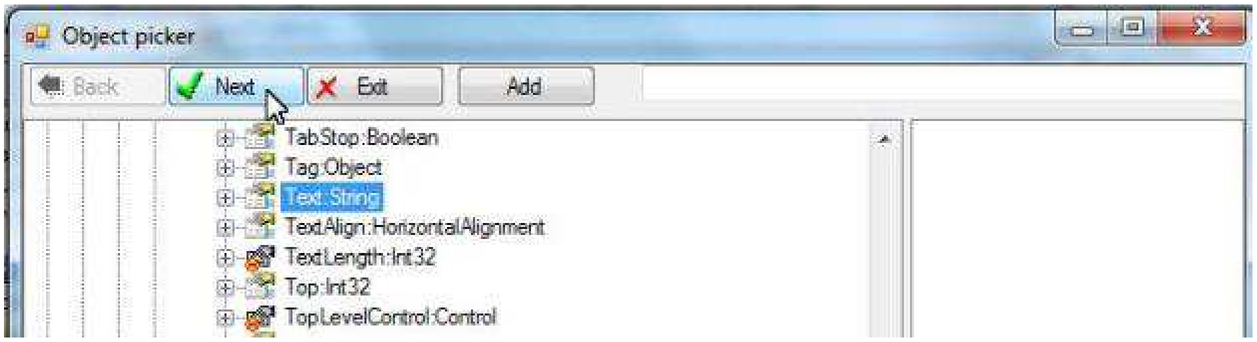


(ActionName)	**Form1.SearchAndReplace**
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	SearchAndReplace
⊞ find	<name>
⊞ replace	

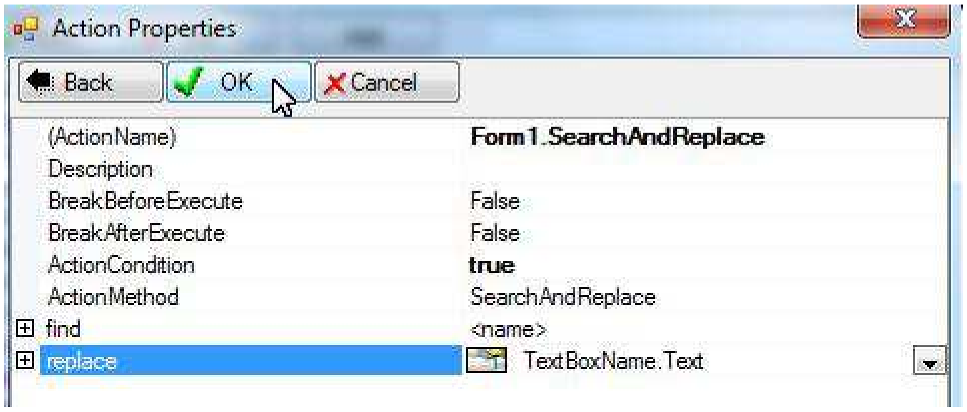
replace

Select the Text property of the text box for name:



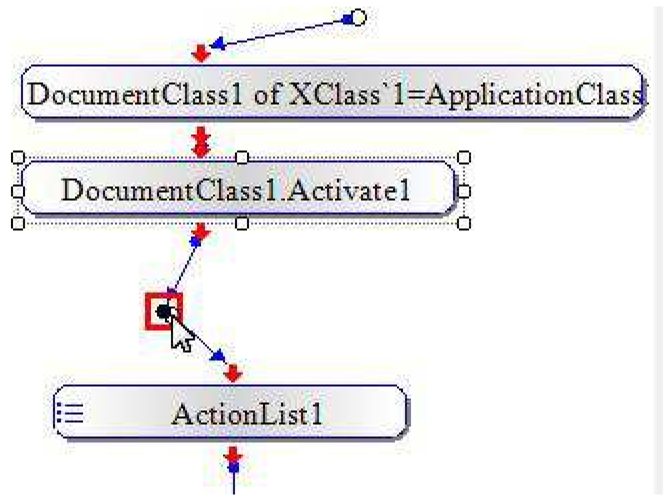


Click OK:

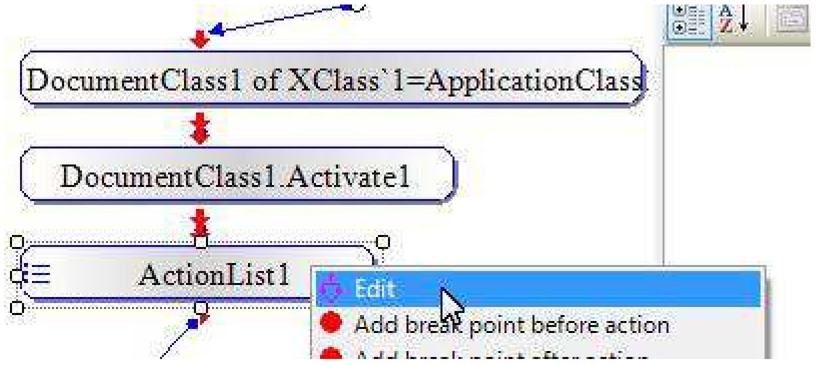


(ActionName)	Form1.SearchAndReplace
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	SearchAndReplace
find	<name>
replace	TextBoxName.Text

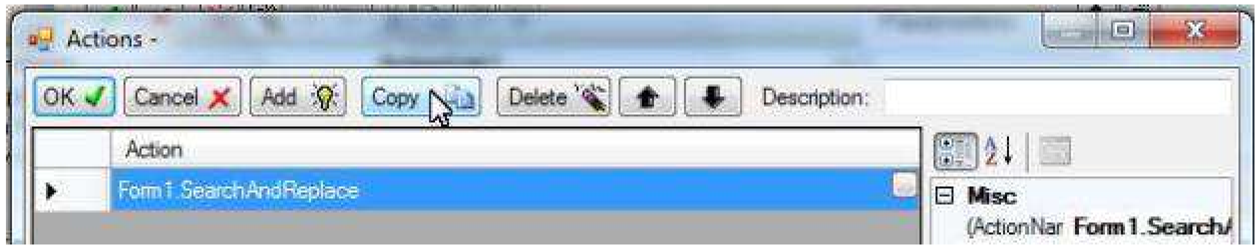
The action-list appears in the Action-Pane. Link it to the previous action:



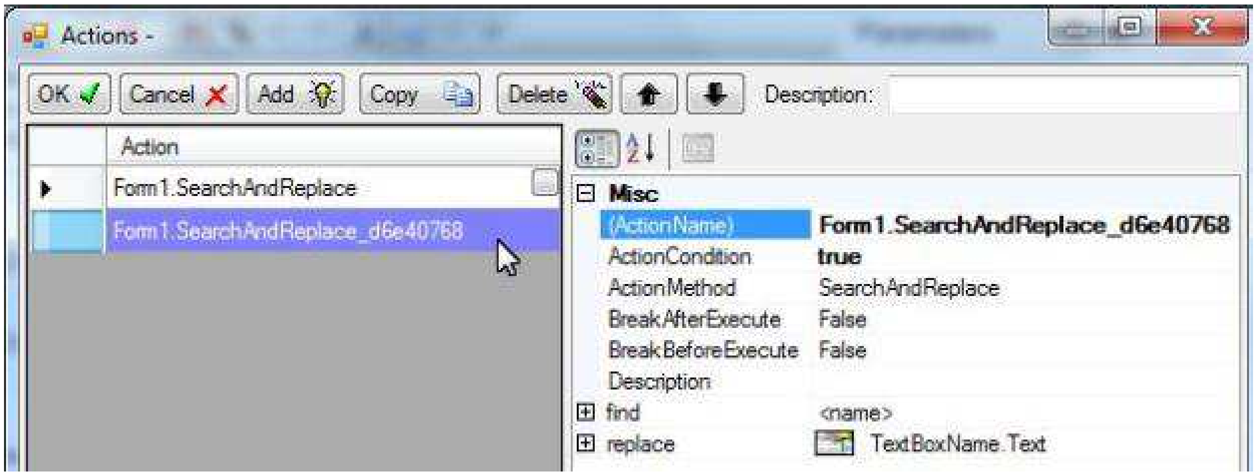
To add more actions into the action-list, right-click the action-list; choose “Edit”:


DocumentClass1 of XClass`1=ApplicationClass
DocumentClass1.Activate1
ActionList1
Edit
Add break point before action

We can see that the search and replace action we created is in the action list. Click “Copy” button to create a new Search and Replace action:



A new action appears in the list:

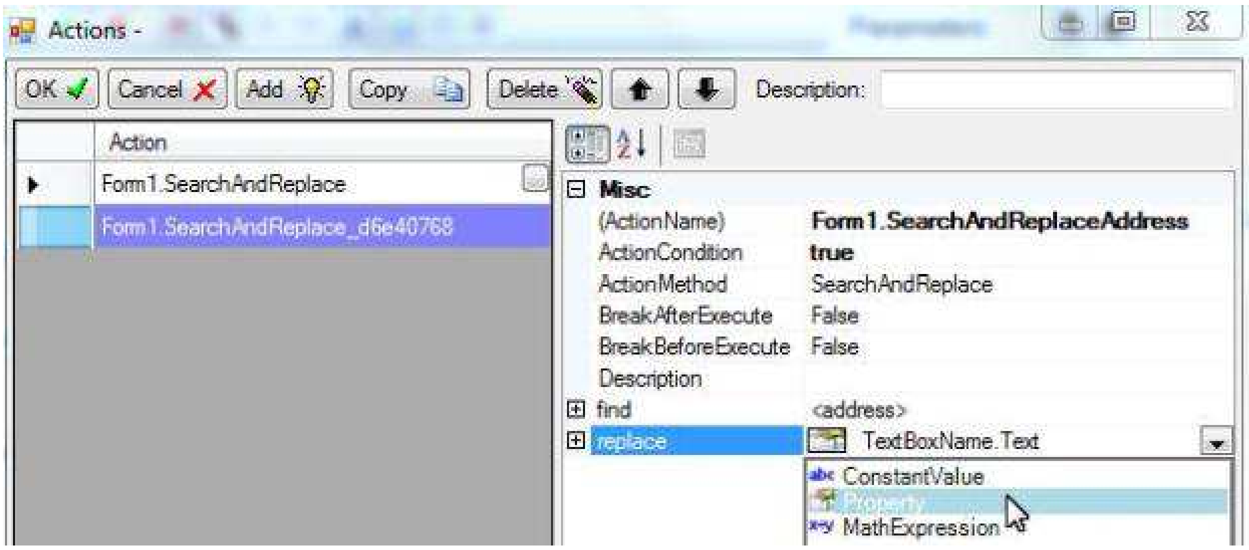


Form1.SearchAndReplace
Form1.SearchAndReplace_d6e40768

Misc

(ActionName)	Form1.SearchAndReplace_d6e40768
ActionCondition	true
ActionMethod	SearchAndReplace
BreakAfterExecute	False
BreakBeforeExecute	False
Description	
find	<name>
replace	TextBoxName.Text

We may rename the action to SearchAndReplaceAddress. Set “find” to <address>. Select “Property” for “replace”:

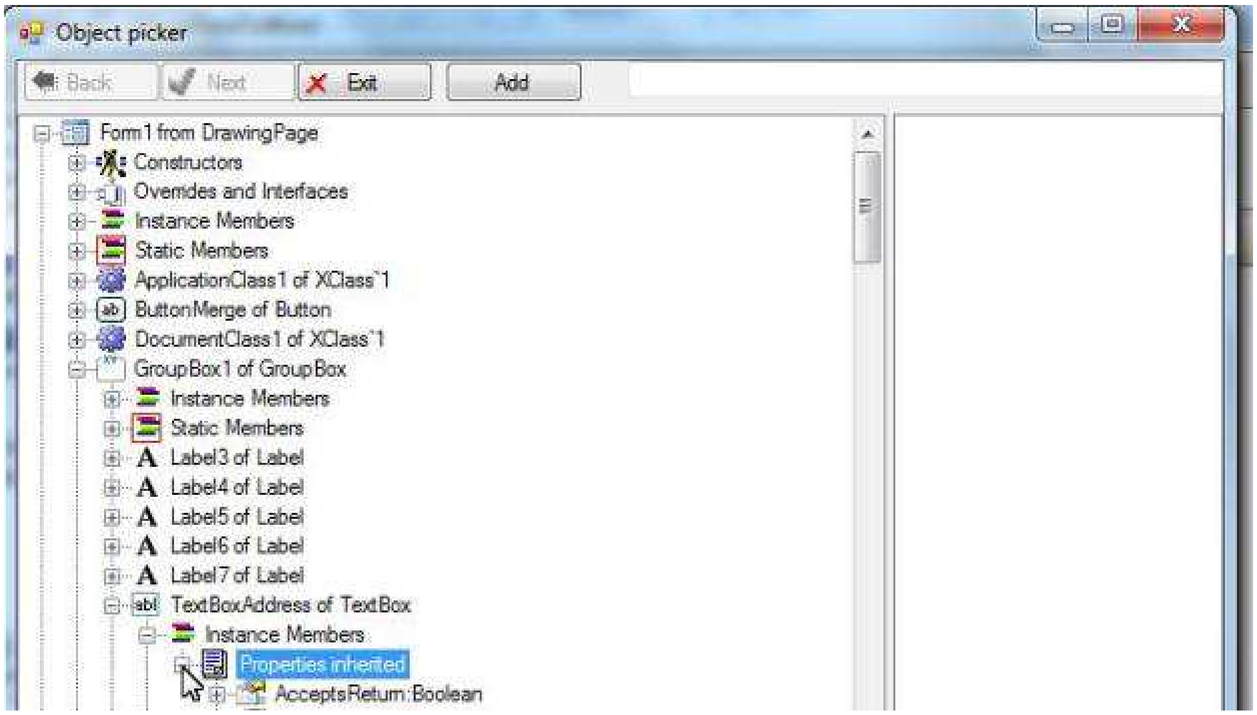


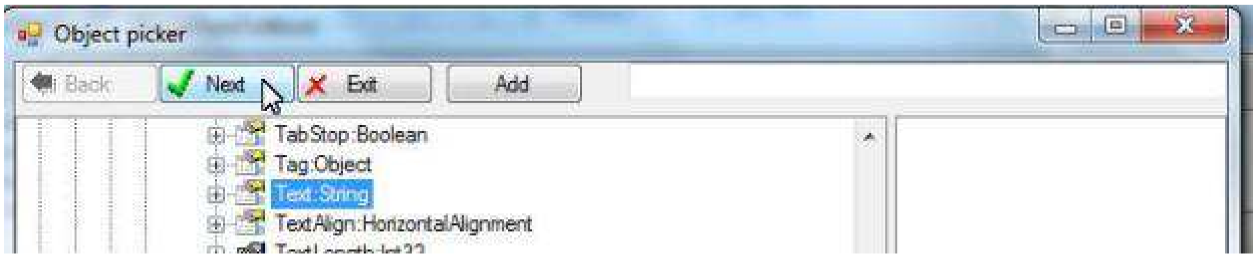
Form1.SearchAndReplace
Form1.SearchAndReplace_d6e40768

Misc

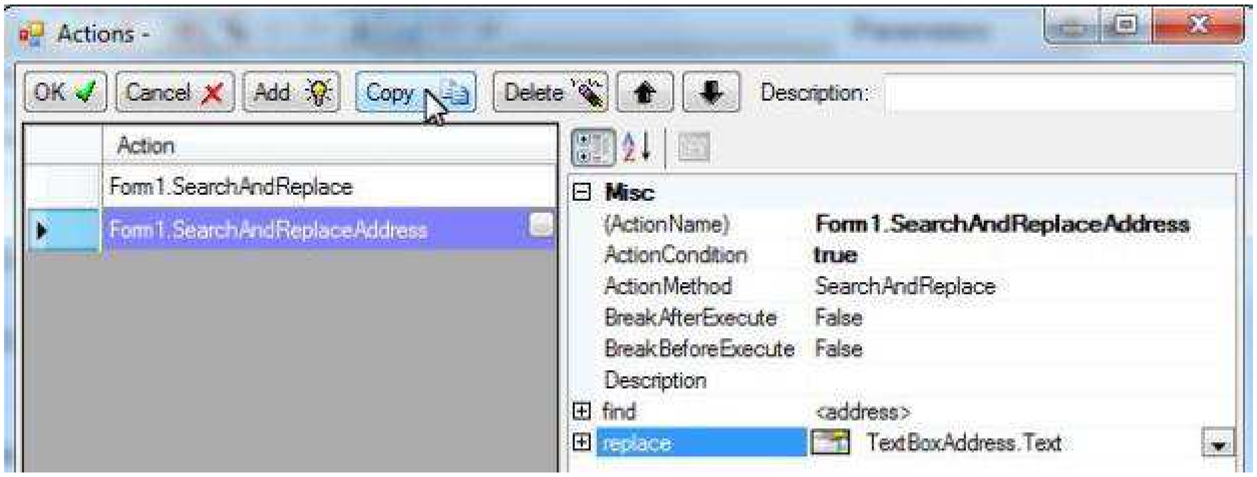
(ActionName)	Form1.SearchAndReplaceAddress
ActionCondition	true
ActionMethod	SearchAndReplace
BreakAfterExecute	False
BreakBeforeExecute	False
Description	
find	<address>
replace	TextBoxName.Text

Select Text property of the text box for address:



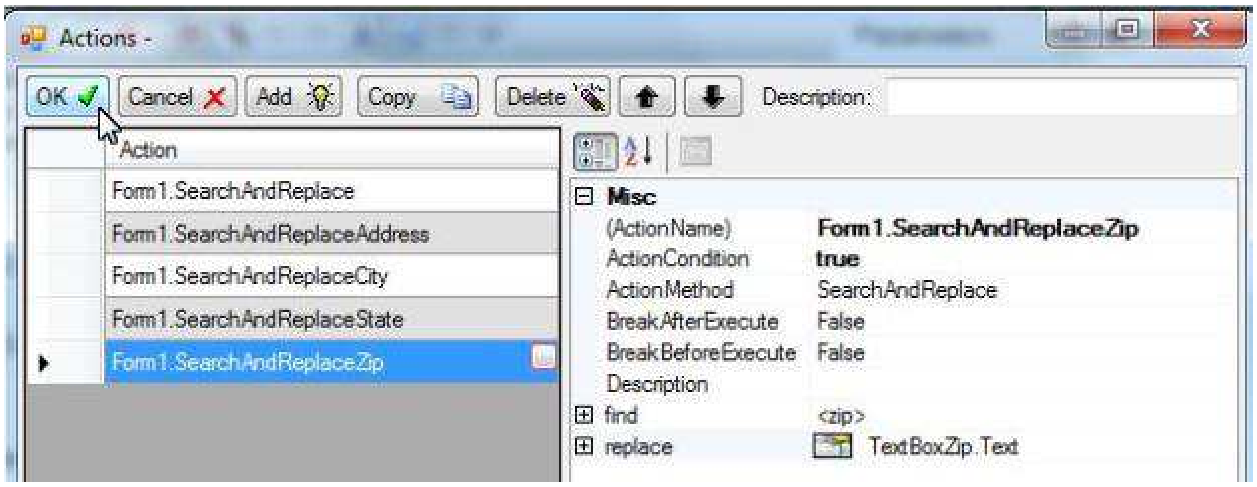


The new action will replace <address> with value from the text box:



(ActionName)	Form1.SearchAndReplaceAddress
ActionCondition	true
ActionMethod	SearchAndReplace
BreakAfterExecute	False
BreakBeforeExecute	False
Description	
find	<address>
replace	TextBoxAddress.Text

We may copy more actions for other data to be replaced:


Actions -

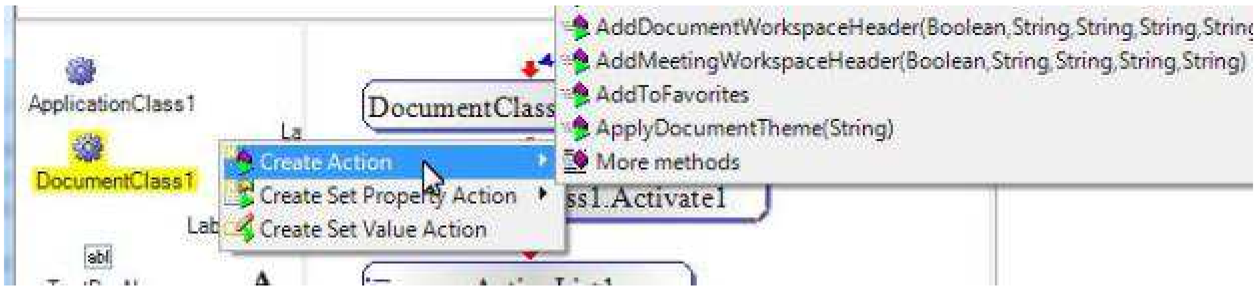
OK | Cancel | Add | Copy | Delete | Description:

Action
Form1.SearchAndReplace
Form1.SearchAndReplaceAddress
Form1.SearchAndReplaceCity
Form1.SearchAndReplaceState
Form1.SearchAndReplaceZip

Misc	
(ActionName)	**Form1.SearchAndReplaceZip**
ActionCondition	**true**
ActionMethod	SearchAndReplace
BreakAfterExecute	False
BreakBeforeExecute	False
Description	
find	<zip>
replace	TextBoxZip.Text

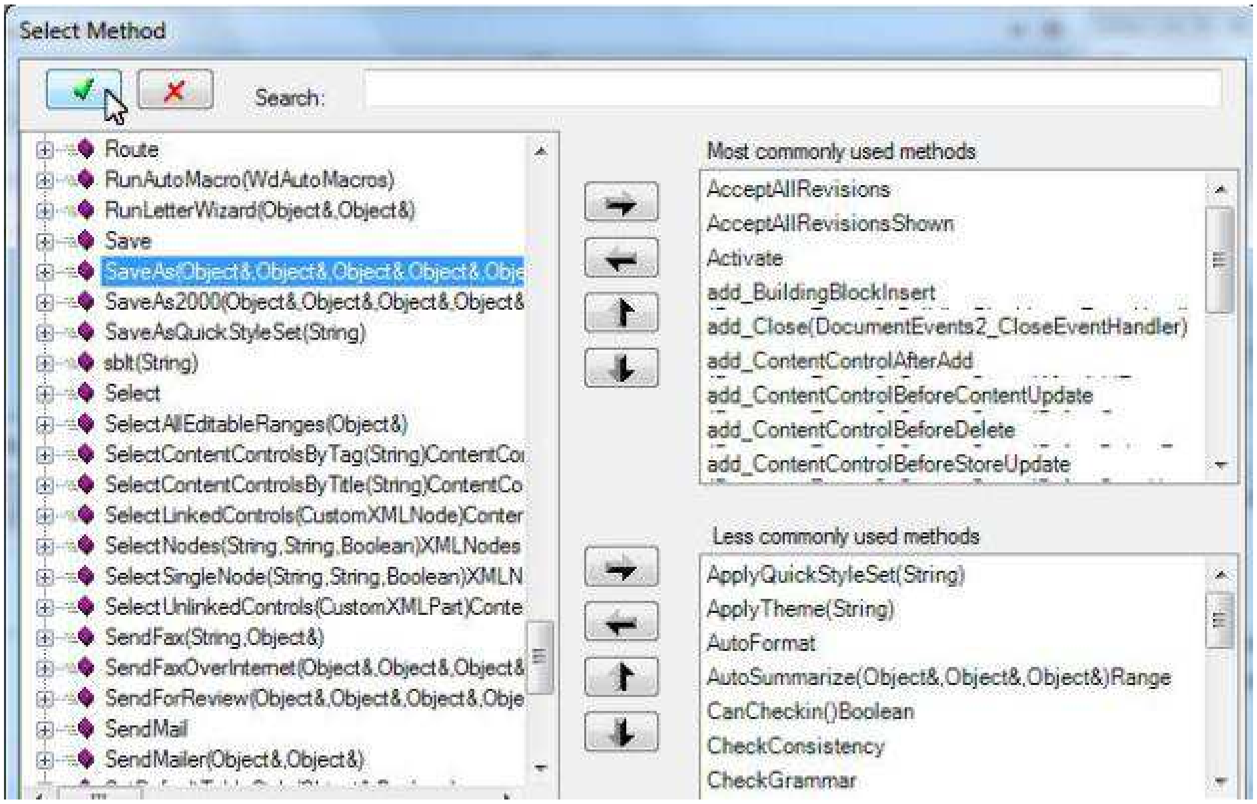
We may create actions to save and close the document.

Right-click the document class icon; choose “Create action”; choose “More methods”; choose “*All methods* =>”:



ApplicationClass1
DocumentClass1
DocumentClass
Create Action
Create Set Property Action
Create Set Value Action
AddDocumentWorkspaceHeader(Boolean,String,String,String,String
AddMeetingWorkspaceHeader(Boolean,String,String,String,String)
AddToFavorites
ApplyDocumentTheme(String)
More methods
ss1.Activate1

Select “SaveAs” method:

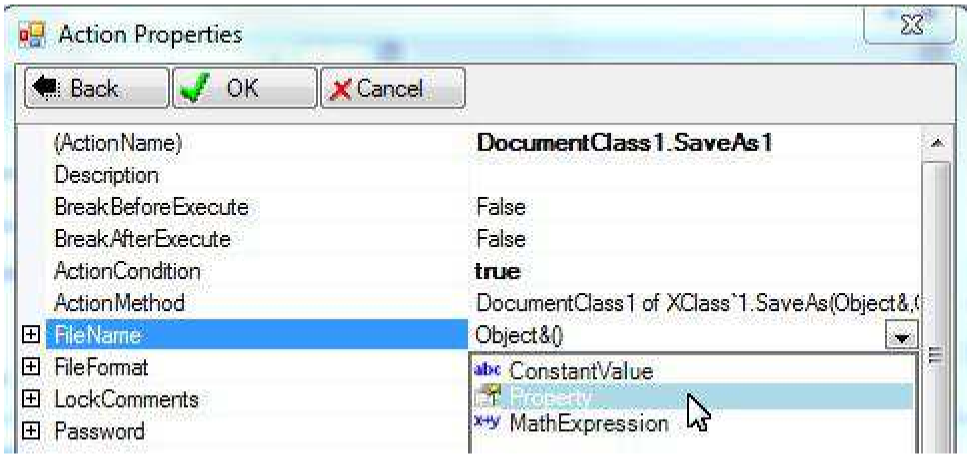


Route
RunAutoMacro(WdAutoMacros)
RunLetterWizard(Object&,Object&)
Save
SaveAs(Object&,Object&,Object&,Object&,Obje
SaveAs2000(Object&,Object&,Object&,Object&
SaveAsQuickStyleSet(String)
sblt(String)
Select
SelectAllEditableRanges(Object&)
SelectContentControlsByTag(String)ContentCo
SelectContentControlsByTitle(String)ContentCo
SelectLinkedControls(CustomXMLNode)Conter
SelectNodes(String,String,Boolean)XMLNodes
SelectSingleNode(String,String,Boolean)XMLN
SelectUnlinkedControls(CustomXMLPart)Conte
SendFax(String,Object&)
SendFaxOverInternet(Object&,Object&,Object&
SendForReview(Object&,Object&,Object&,Obje
SendMail
SendMailer(Object&,Object&)

AcceptAllRevisions
AcceptAllRevisionsShown
Activate
add_BuildingBlockInsert
add_Close(DocumentEvents2_CloseEventHandler)
add_ContentControlAfterAdd
add_ContentControlBeforeContentUpdate
add_ContentControlBeforeDelete
add_ContentControlBeforeStoreUpdate

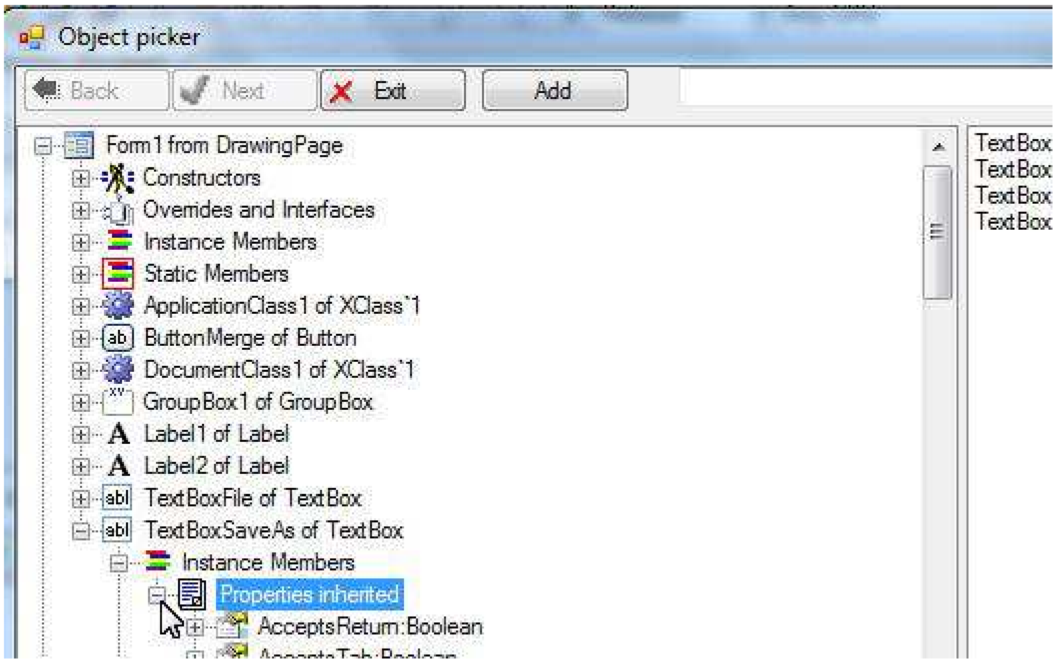
ApplyQuickStyleSet(String)
ApplyTheme(String)
AutoFormat
AutoSummarize(Object&,Object&,Object&)Range
CanCheckin()Boolean
CheckConsistency
CheckGrammar

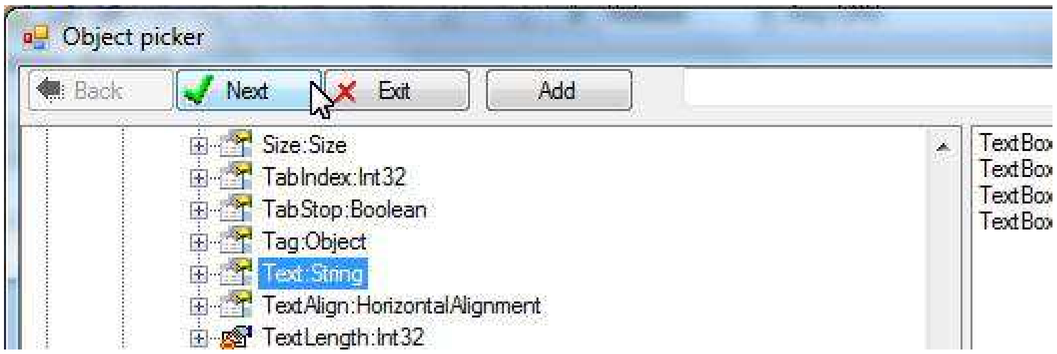
Select “Property” for the “FileName” parameter:



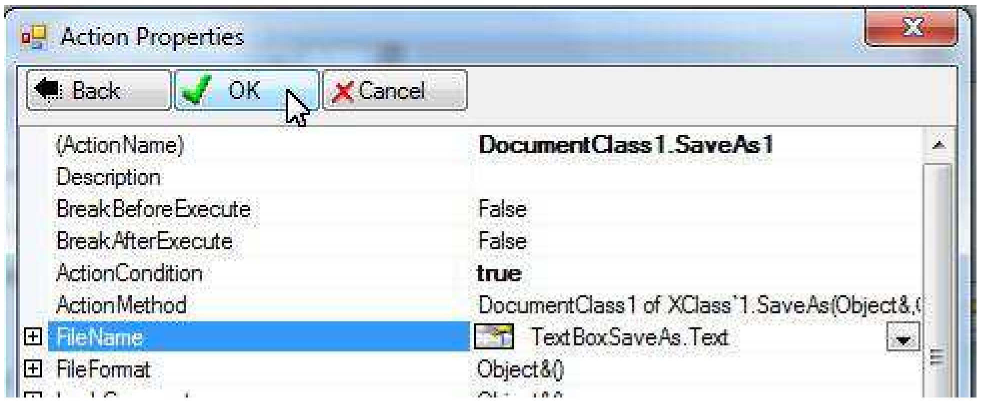
(ActionName)	DocumentClass1.SaveAs1
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	DocumentClass1 of XClass`1.SaveAs(Object&,(
FileName	Object&()
FileFormat	
LockComments	
Password	

Select Text property of the text box for target file path:



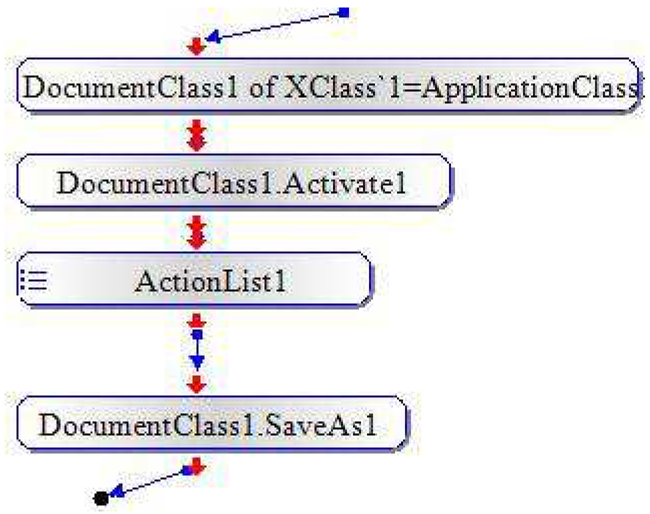


Click OK:

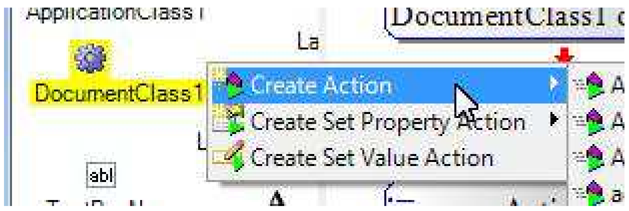


(ActionName)	DocumentClass1.SaveAs1
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	DocumentClass1 of XClass`1.SaveAs(Object&,(
FileName	TextBoxSaveAs.Text
FileFormat	Object&()

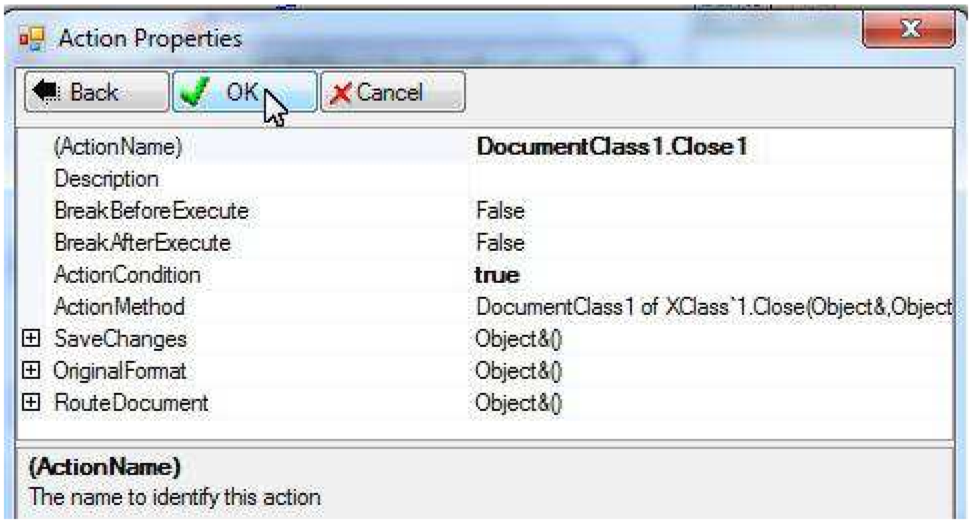
Link the action to the last action:



To create a Close document action, right-click the document class icon; choose “Create action”; choose “More actions”; choose “Close” method.



Click OK:



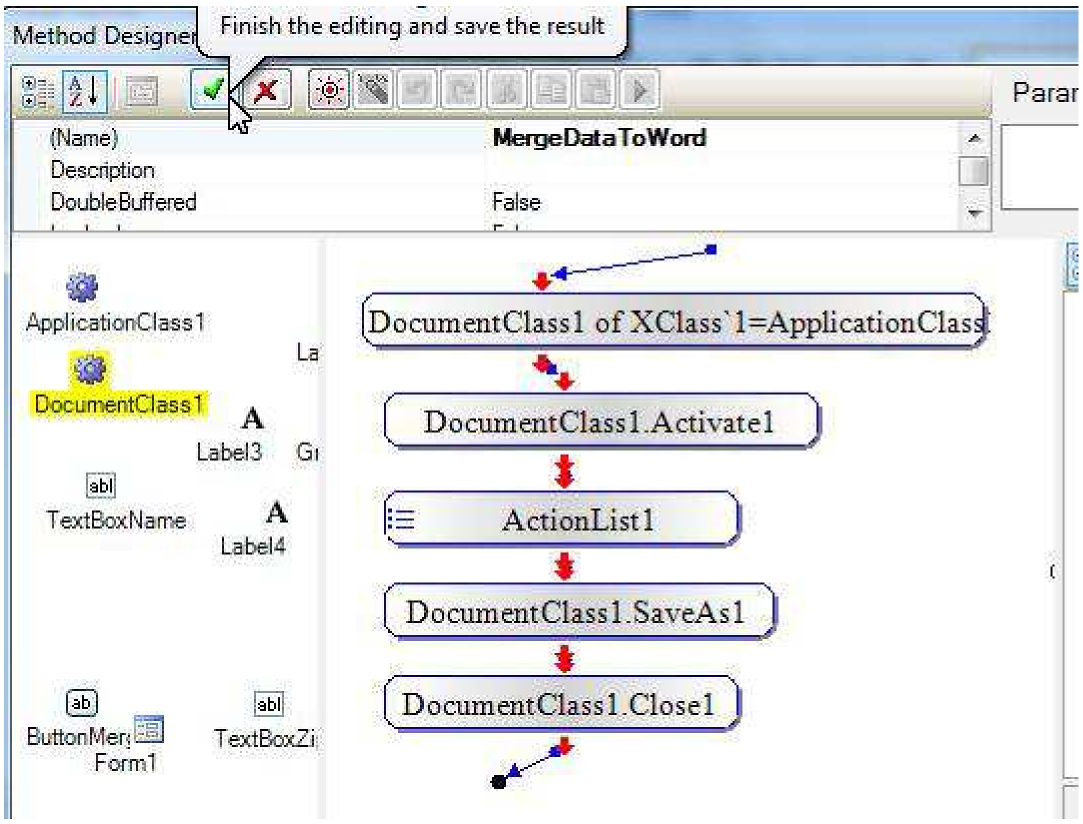
Action Properties

Back | OK | Cancel

(ActionName)	**DocumentClass1.Close1**
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	DocumentClass1 of XClass`1.Close(Object&,Object
⊞ SaveChanges	Object&()
⊞ OriginalFormat	Object&()
⊞ RouteDocument	Object&()

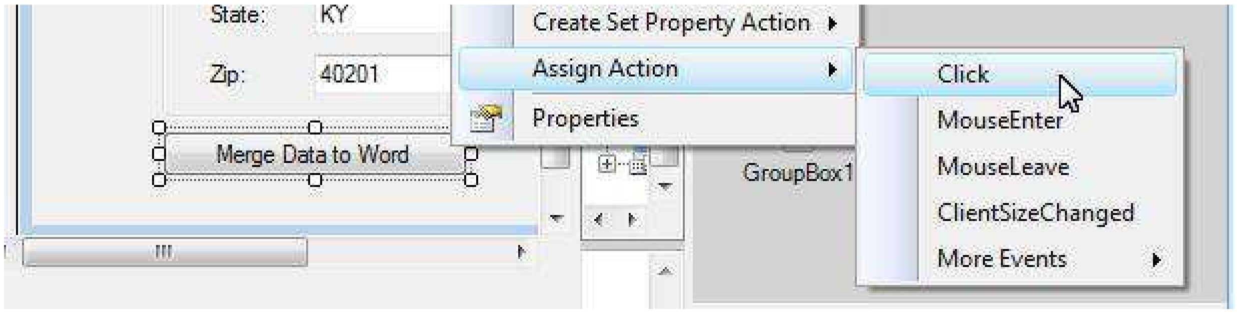
(ActionName)
The name to identify this action

Link it to the last action. We are done creating this method:

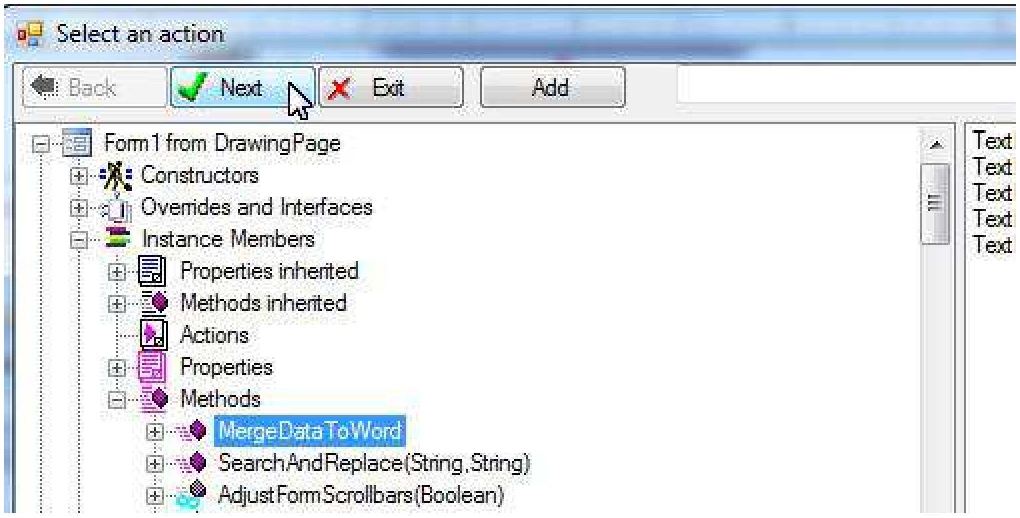


Execute Method

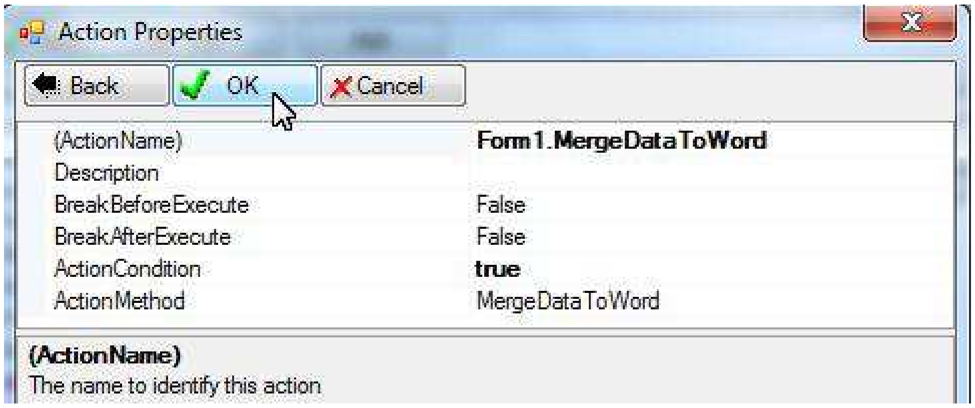
Right-click the button; choose “Assign action”; choose “Click” event:



Select MergeDataToWord method:



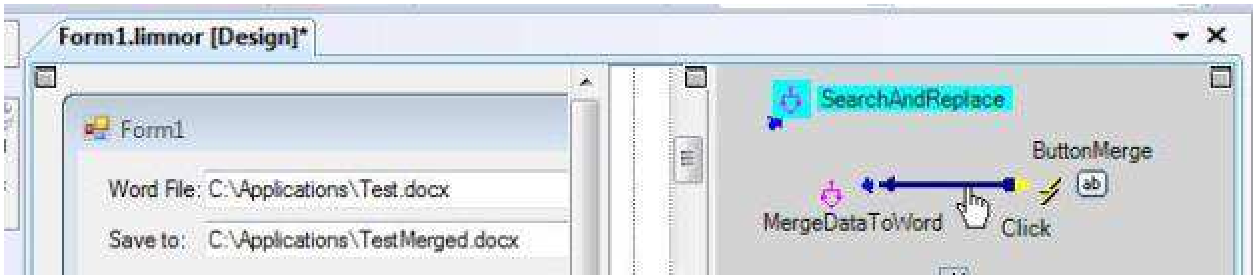
Click OK:



(ActionName)	**Form1.MergeDataToWord**
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	**true**
ActionMethod	MergeDataToWord

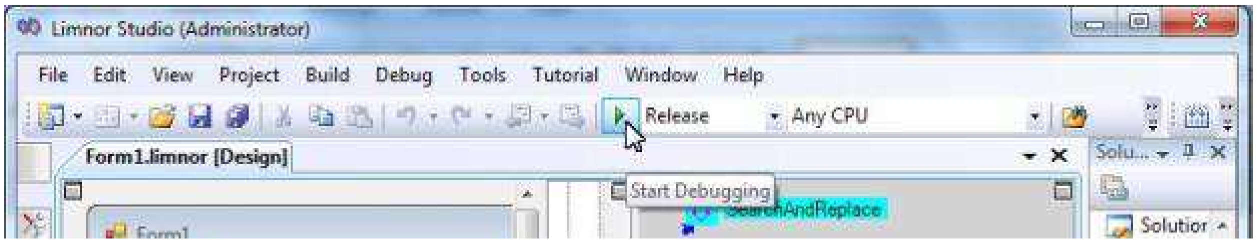
(ActionName)
The name to identify this action

An action is created and assigned to the button:

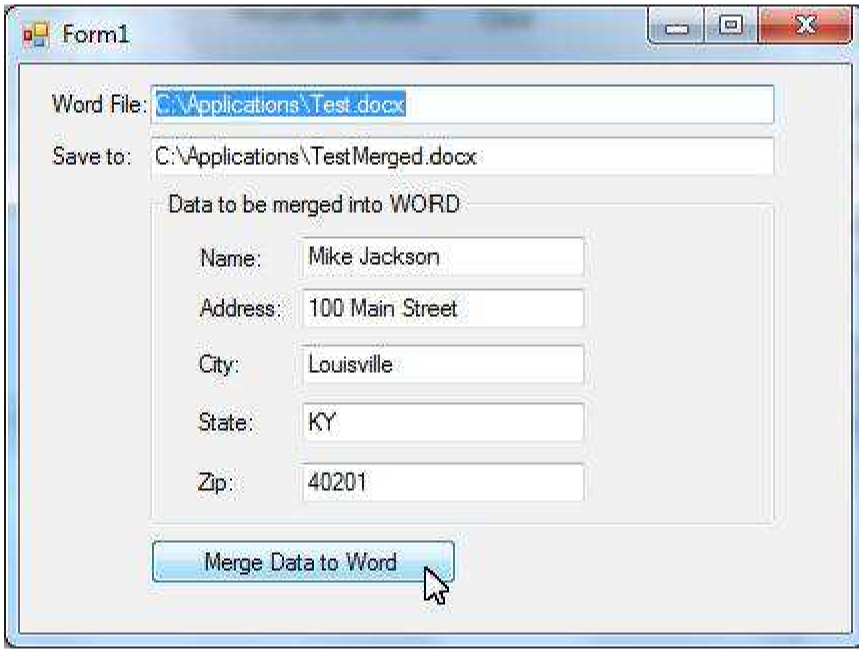

Form1.limnor [Design]*
Form1
Word File: C:\Applications\Test.docx
Save to: C:\Applications\TestMerged.docx
SearchAndReplace
ButtonMerge
MergeDataToWord
Click

Test

Click the Run button to compile and run the sample.


Limnor Studio (Administrator)
File Edit View Project Build Debug Tools Tutorial Window Help
Release
Any CPU
Form1.limnor [Design]
Start Debugging
SearchAndReplace
Solution

The form appears. Click the button.



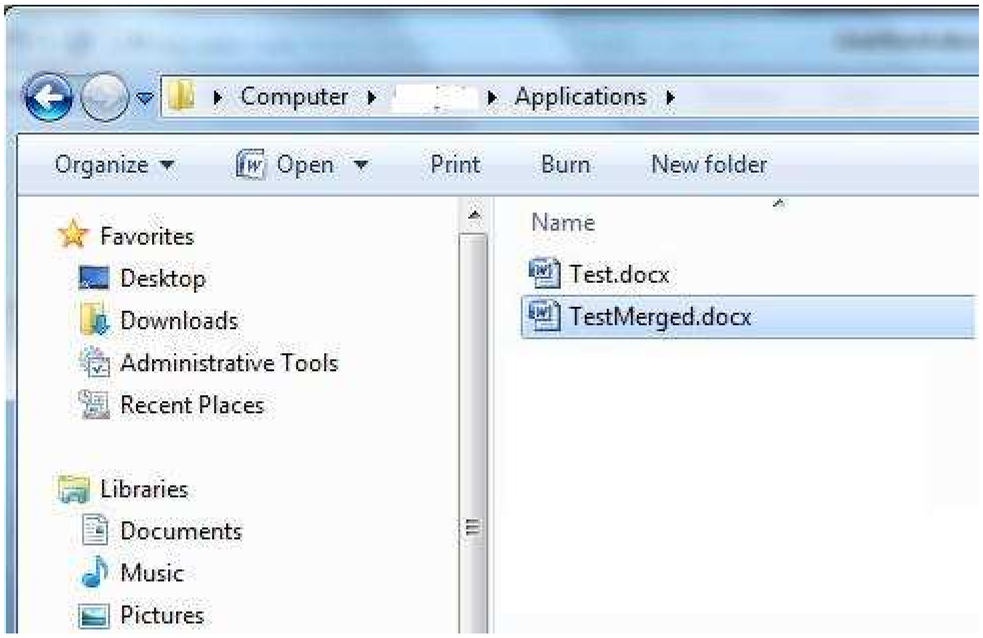
Word File: C:\Applications\Test.docx

Save to: C:\Applications\TestMerged.docx

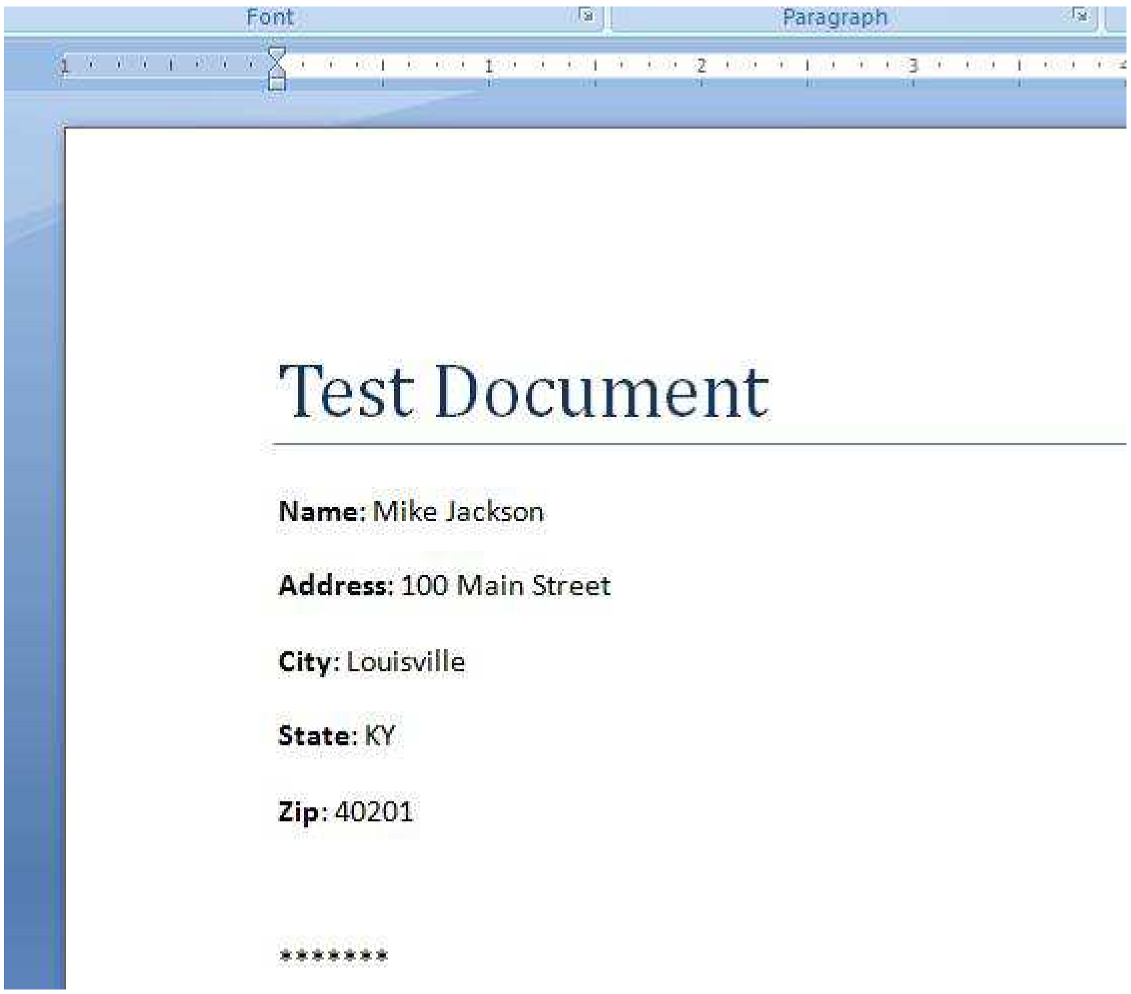
Data to be merged into WORD

Name:	Mike Jackson
Address:	100 Main Street
City:	Louisville
State:	KY
Zip:	40201

We can see that the “Save to” file, TestMerged.docx, is created:



Open this document file; we can see that the data from the form are merged into the document:



Test Document

Name: Mike Jackson

Address: 100 Main Street

City: Louisville

State: KY

Zip: 40201

****

So, the sample works as we expected.

Problems and fixes

One problem we found during the creation of this sample is that an empty Word document kept showing up. It is because we create a Word document instance on the form, DocumentClass1. Actually for the sample, we do not need to create a DocumentClass instance on the form because we do not need to access it outside of the MergeDataToWord method. We may use a local variable of DocumentClass inside the MergeDataToWord method. To do it, when creating the Open action, instead of choosing “Select existing object”, choose “New local variable”. The SaveAs and Close action should be created using the new local variable. We’ll leave it as an exercise for you.