

Value Merge with String Tool and Application Configuration

Contents

Introduction	2
Create Run Time Application	3
Application Functionality	4
Tag markers.....	4
Output UI	4
Write to file	5
Copy to Clipboard	13
Print output.....	15
Create configurations.....	20
Load configurations	21
Show software title	22
Load document template.....	24
Collect User Inputs.....	26
Tag Index.....	26
Show question.....	27
Wizard Navigation.....	40
Save user input.....	40
Previous Button.....	44
Next Button.....	49
Modify “Previous” button handler	55
About Form	57
Create Design Time Application.....	58
Remember EXE file path	59
Program preview.....	60
Save program	61
Test.....	61

More Information	73
Feedback	74

Introduction

The sample projects created in this document can be downloaded from

<http://www.limnor.com/studio/RunTimeApp.zip>

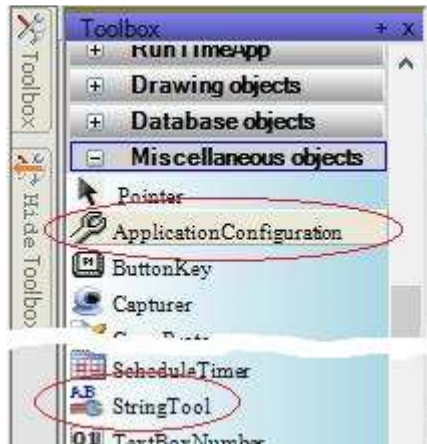
<http://www.limnor.com/studio/DesignTimeApp.zip>

Rapid application development usually is done by using proper existing components in software libraries. For standalone application development, all Microsoft .Net Framework libraries developed by companies and individuals can be used. As an example, in a discussion in Limnor Studio user forum Mr. Ramzi wants to make a Software Creator, as described in his video http://www.youtube.com/embed/5FsaZC_e_XI?rel=0&vq=hd720. Let's review his requirements and see what components should be used for which business requirements.

- Application functionality
 - Document creation by substituting "tags" with user inputs – It can be accomplished by a component named StringTool provided by Limnor Studio.
 - Create text files – It can be accomplished by a component named StreamWriter provided by Microsoft
 - Read text file – It can be accomplished by a component named StreamReader provided by Microsoft
 - Set data to the Clipboard – It can be accomplished by a component named Clipboard provided by Microsoft
 - Print text – It can be accomplished by a component named PrintDocument provided by Microsoft. Microsoft also provides other printing related components.
- Application customization – It can be accomplished by a component named ApplicationConfiguration provided by Limnor Studio.

In this sample, two applications should be developed. One application accepts user inputs and generates document outputs using a document template (referred to as recipe); it can be customized; let's call it "Run Time Application". Another application makes customizations to the first application; let's call it "Design Time Application". Each customization of the "Run Time Application" is considered to be different software; thus the "Design Time Application" is called "Software Creator" by Ramzi.

StringTool and ApplicationConfiguration can be found in the Toolbox:

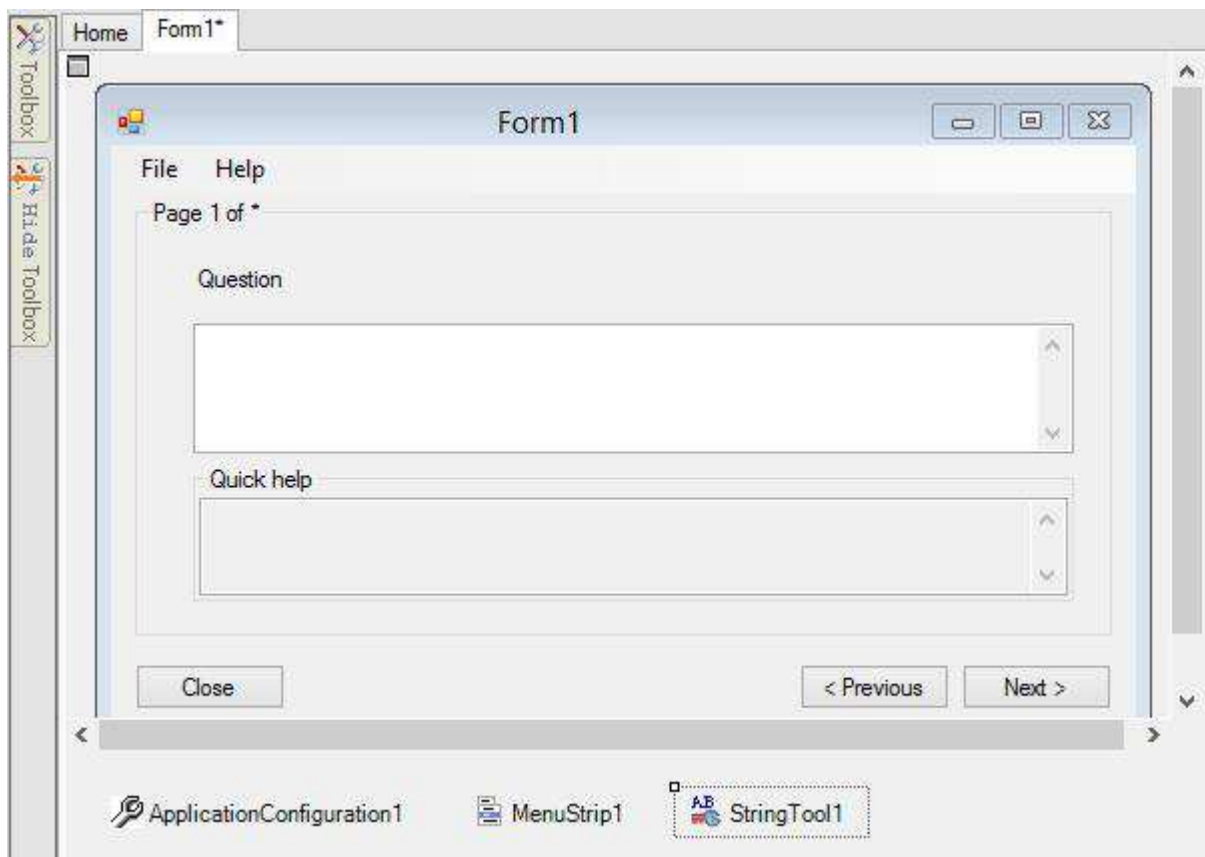


Create Run Time Application

You may download the project from <http://www.limnor.com/studio/RunTimeApp.zip>.

There are two basic design aspects: accomplish application functionality; weave customization into the application.

User interface is created according to the requirements:



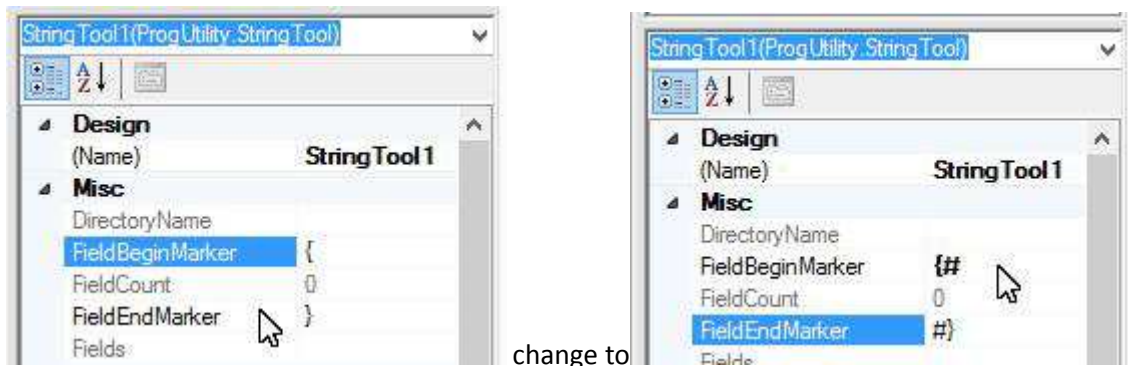
You can see that we added components StringTool1 and ApplicationConfiguration1 to the form.

Application Functionality

The major functionality of this application is to form document and create output. Forming document can be easily done by StringTool1. Creating output can be done by StreamWriter, Clipboard, or other components depending on the types of the outputs.

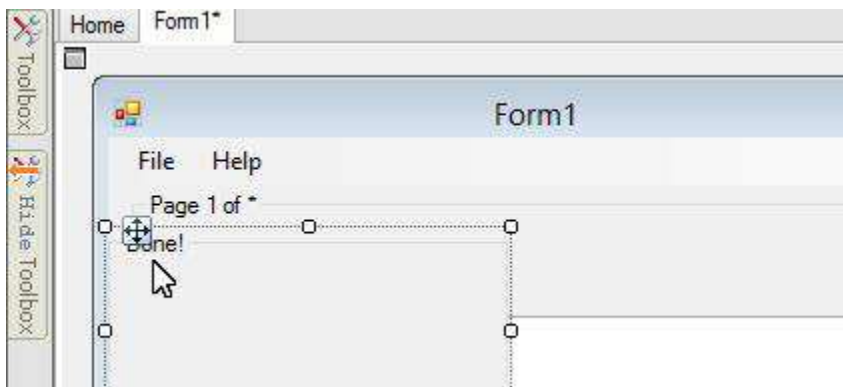
Tag markers

By default, StringTool uses "{" as the beginning tag marker and uses "}" as the ending tag marker. You may change them by setting the properties. For example, change them to "{#" and "#}":

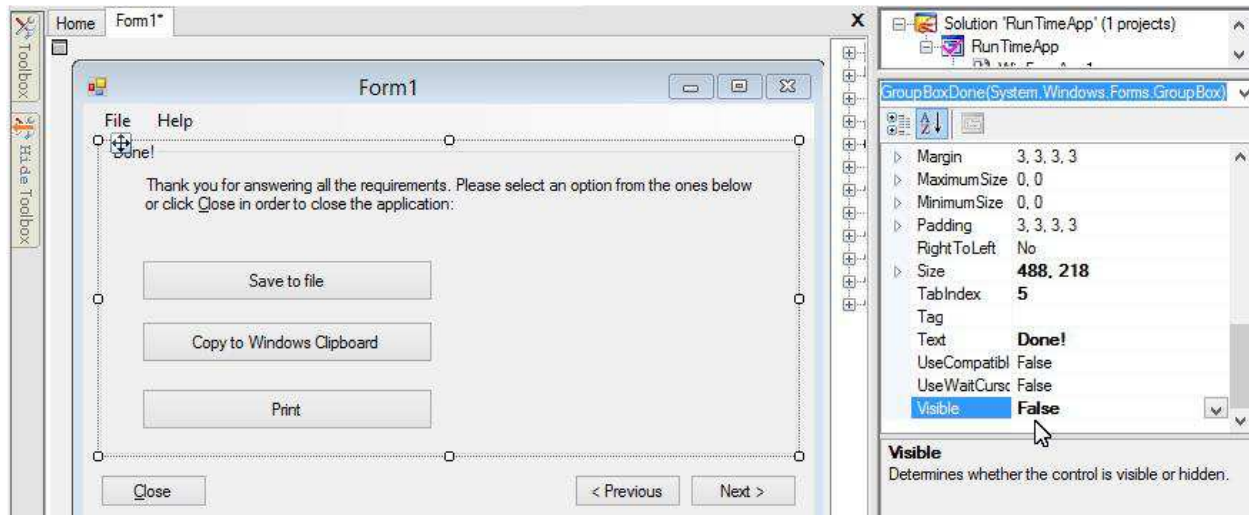


Output UI

From the UI requirement, we use another Group Box to show output UI:

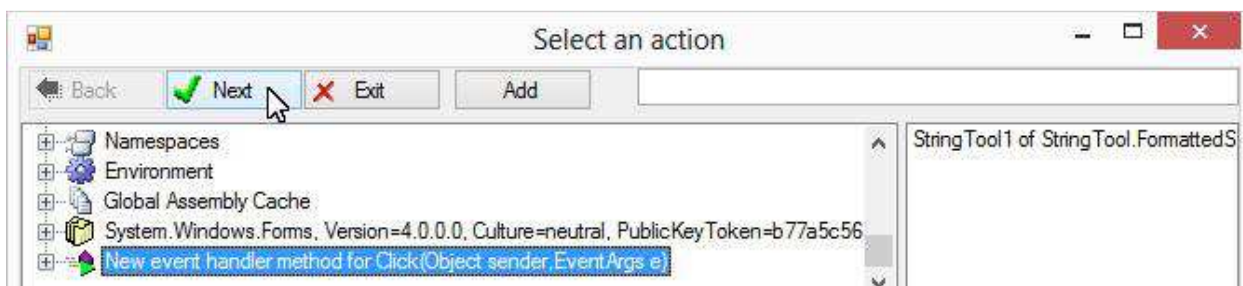
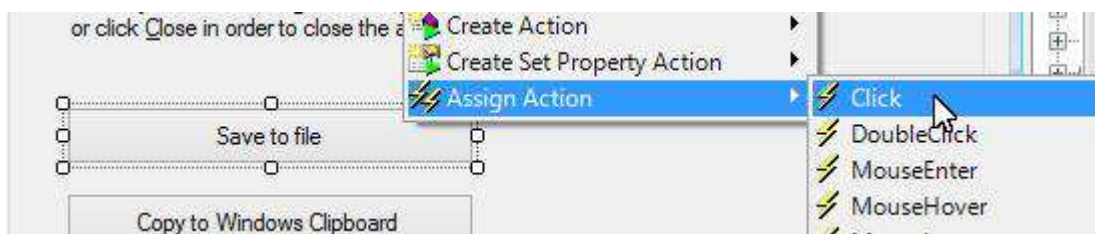


Set Visible property of the group box to False so that initially it is not visible.

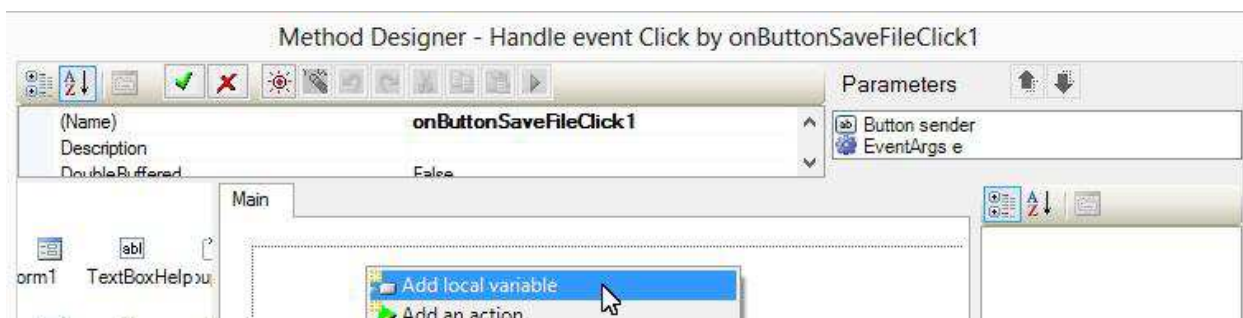


Write to file

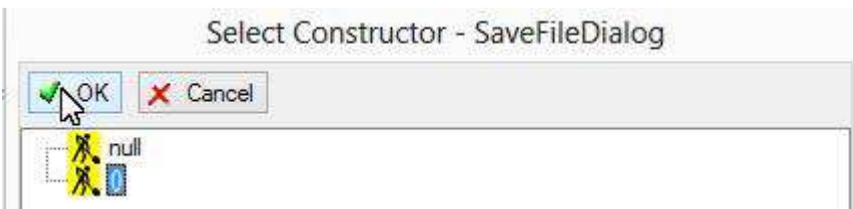
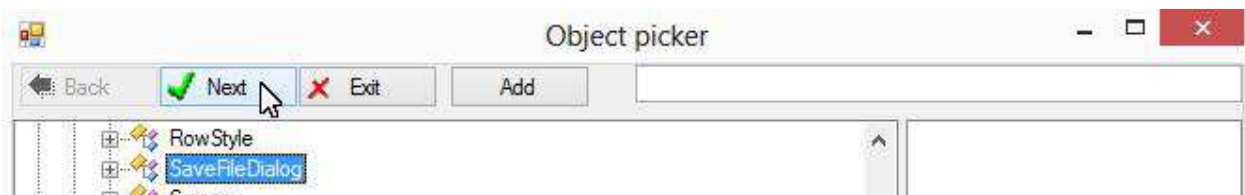
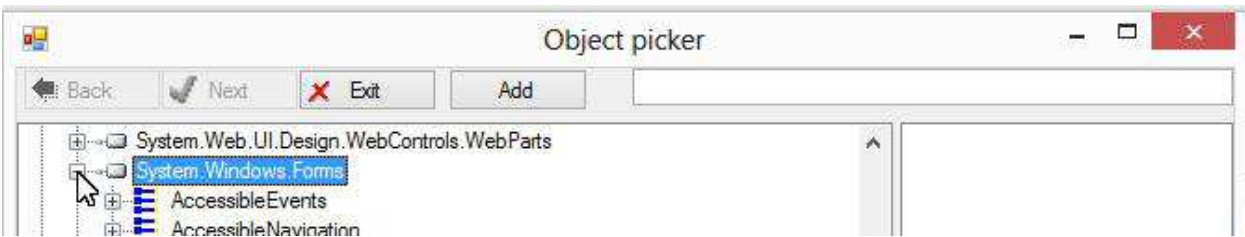
Right-click the button to assign “write to file” operations to the button:



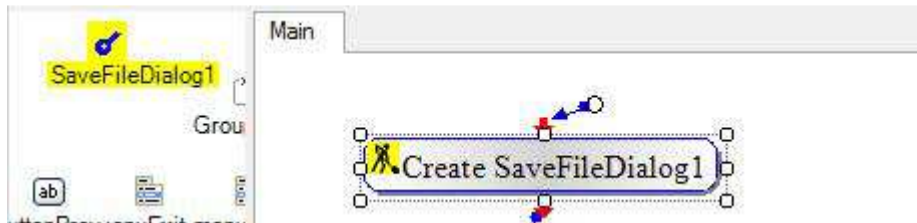
A Method Designer appears for creating operations. Create an Save File Dialogue to allow the user to select a file:



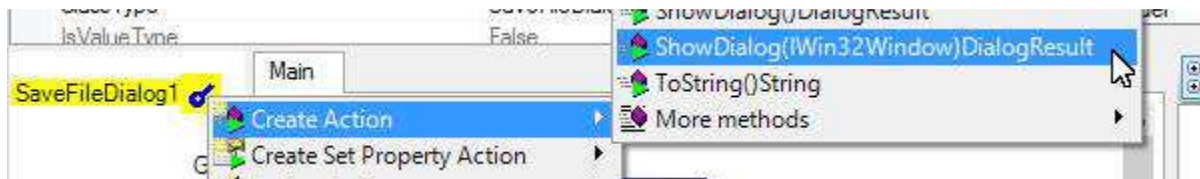
Select SaveFileDialog:



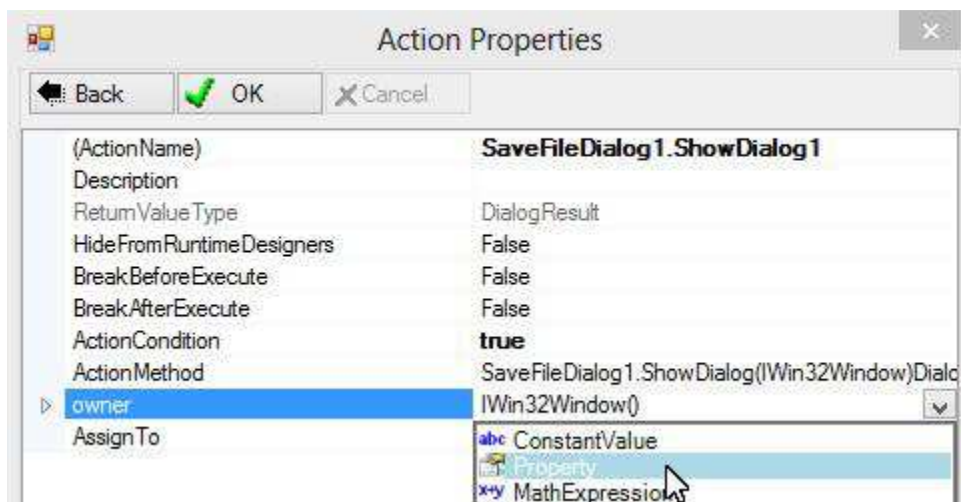
An action is created to initialize a SaveFileDialog object; a variable named SaveFileDialog1 is created:



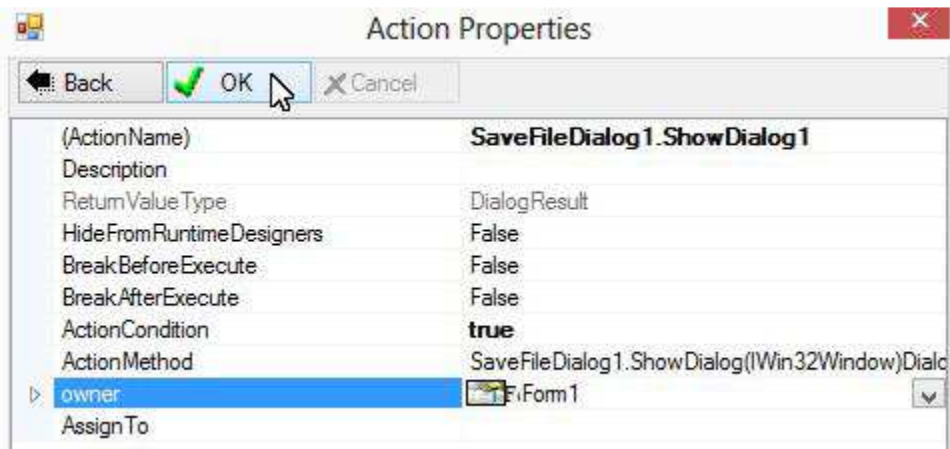
Show the save-file dialogue box for the user to select a file full path:



Use the current form as the dialogue box owner:

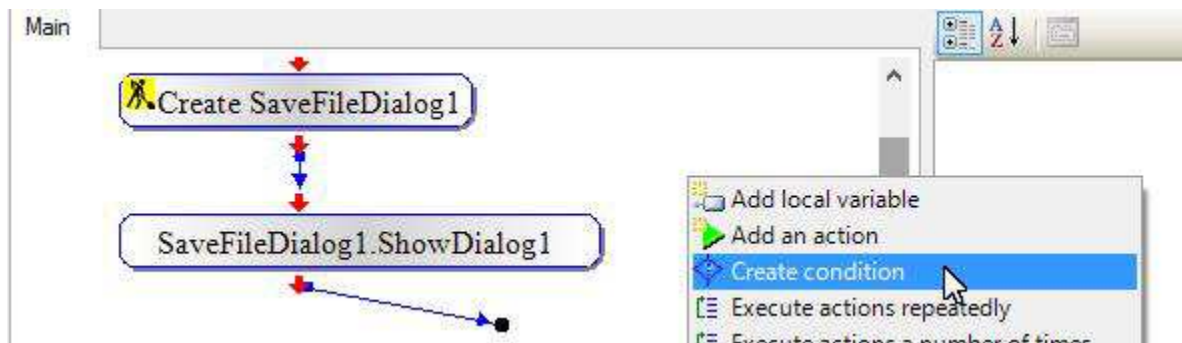


Click OK:

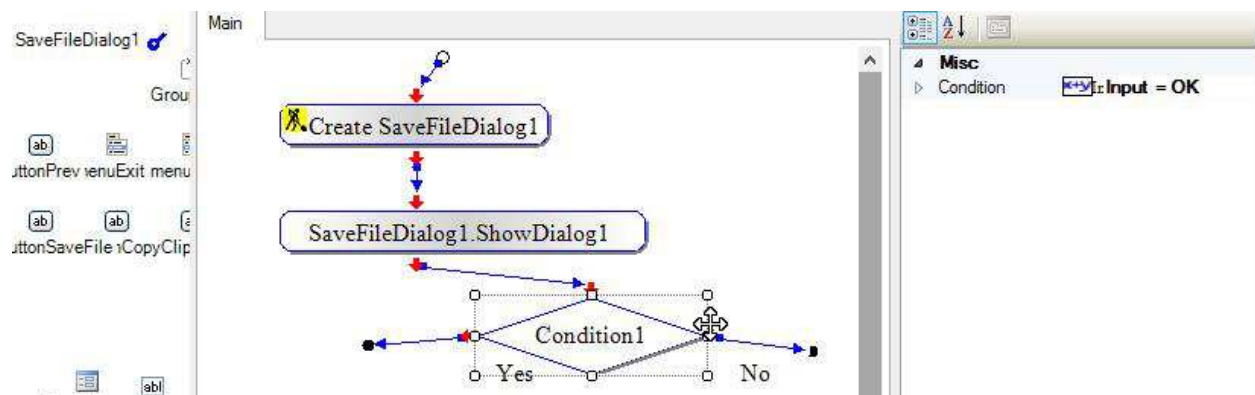


Link the action to the last action.

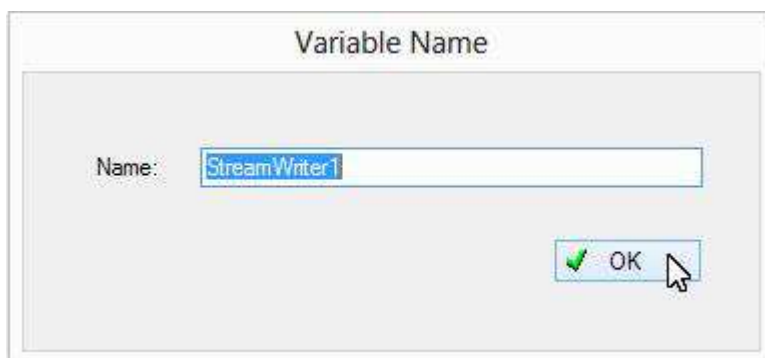
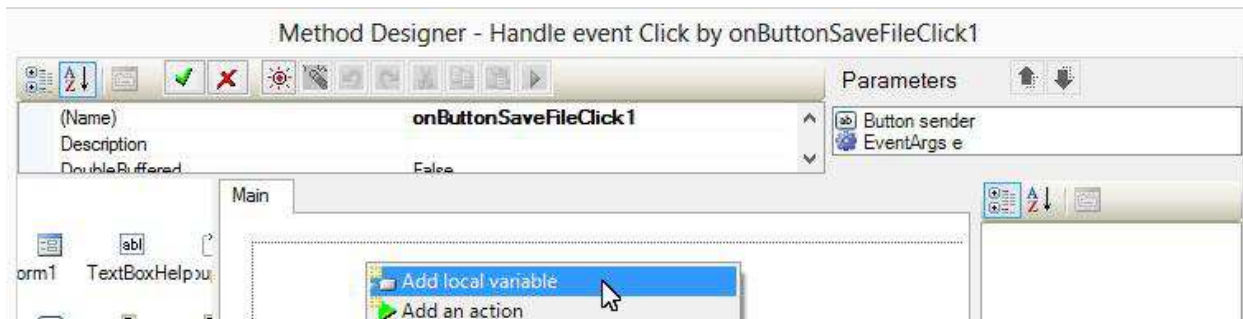
Add a Condition action to check whether the user canceled the save-file dialogue box:

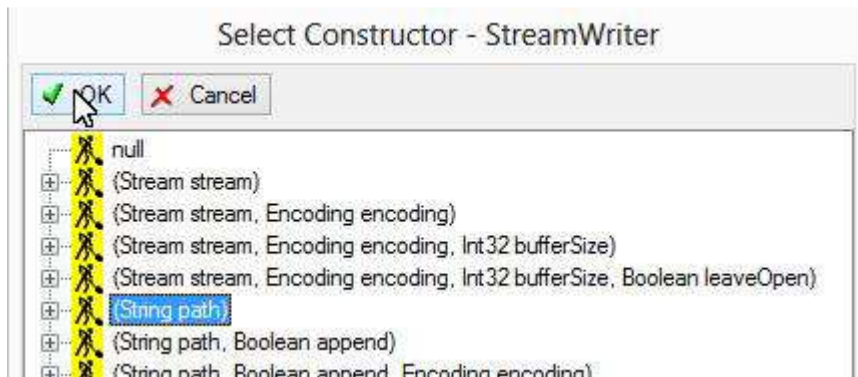


Link the new Condition action to the last action. The Yes port is the control flow when a file is selected:

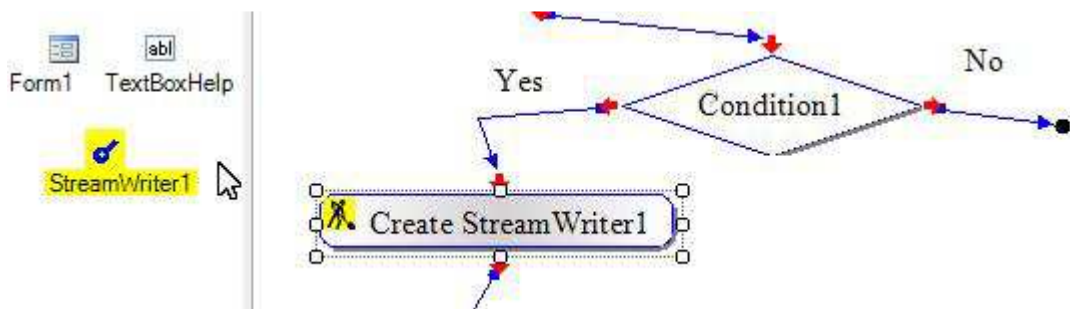


Create a StreamWriter to write to file:

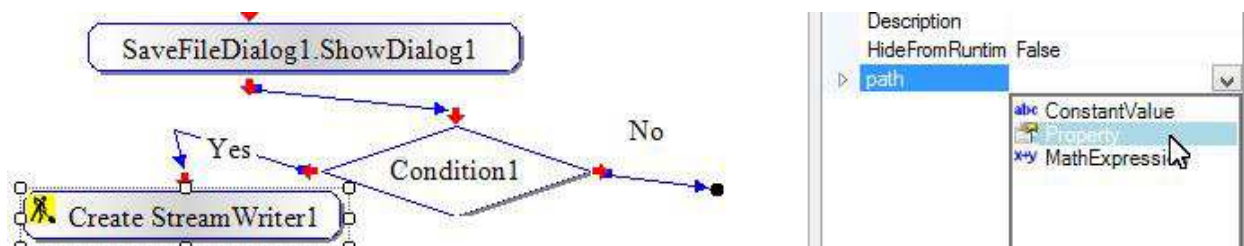


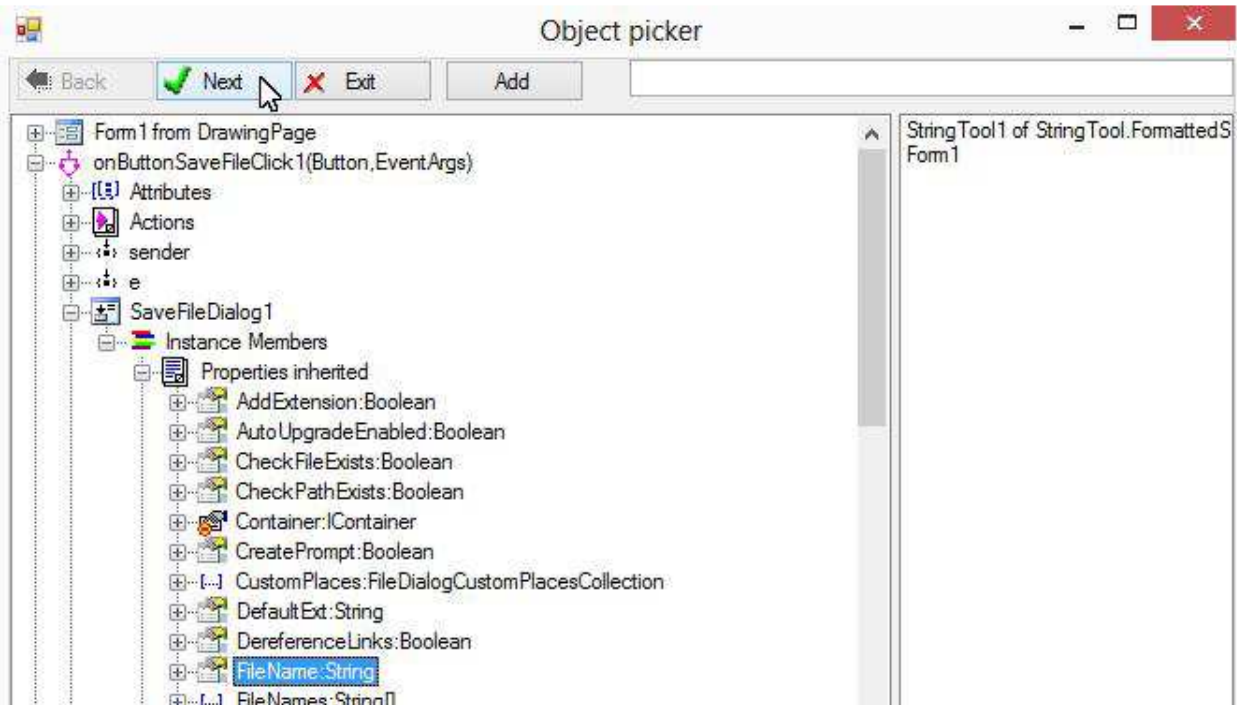


An action is created to initializing a StreamWriter object represented by variable StreamWriter1. Link the action to the Yes port of the Condition action:

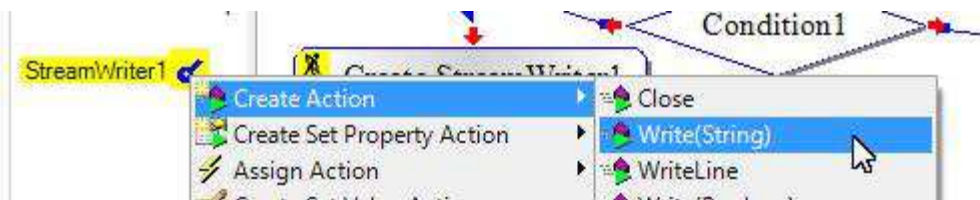


Set the action's "path" to FileName property of the save-file dialogue box:

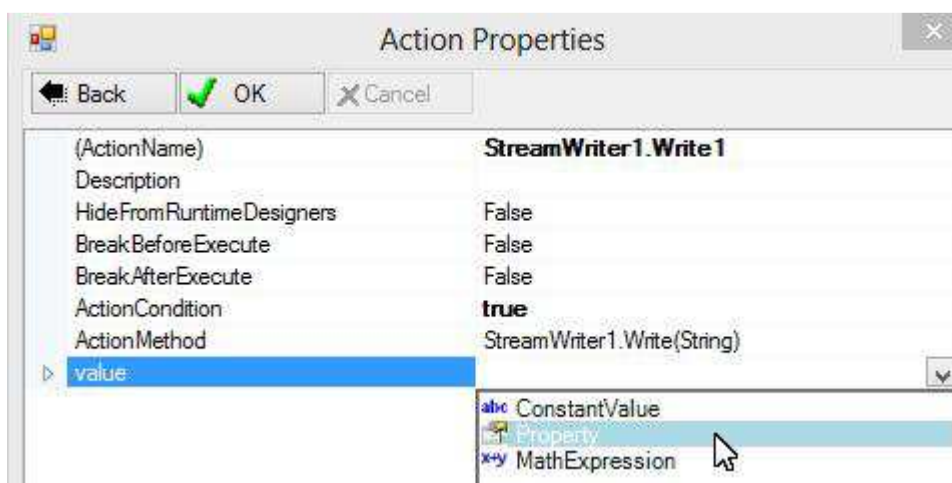


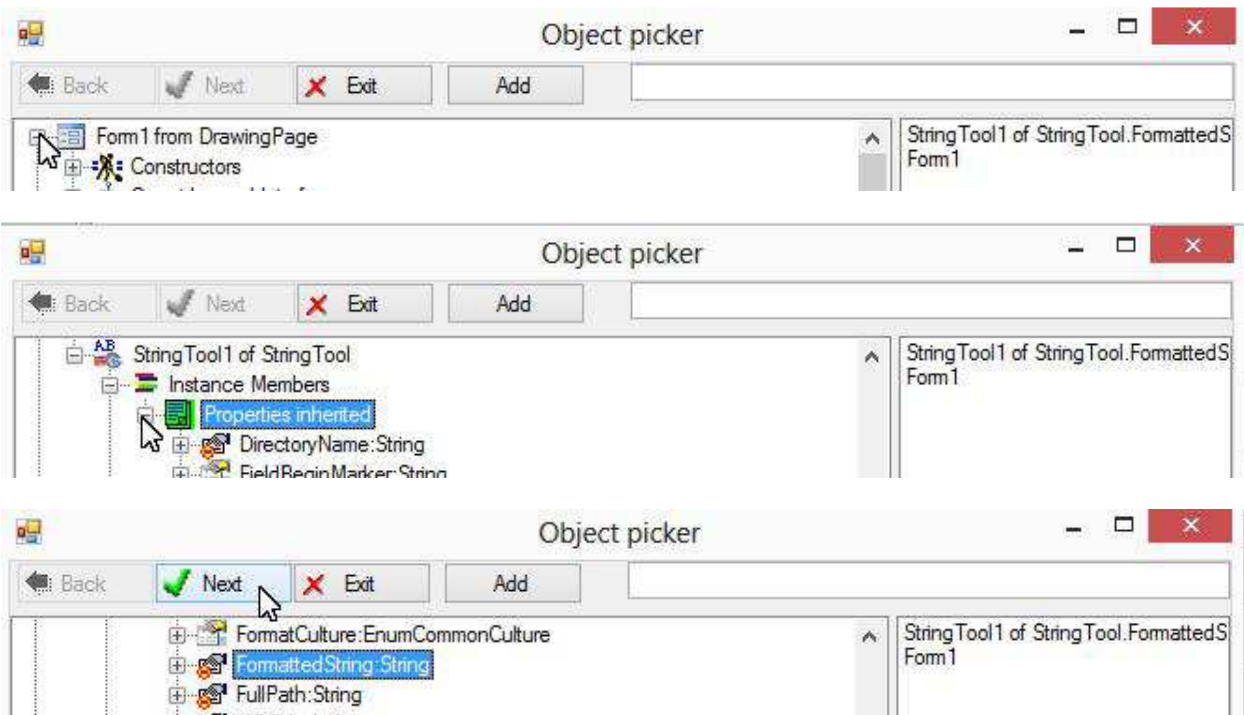


Create Write action to write the text:

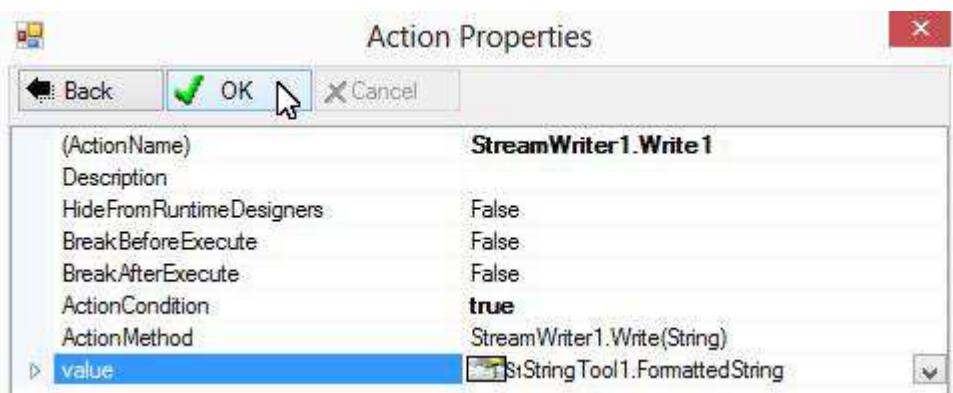


Select formattedString property of Stringool1 for the value parameter:



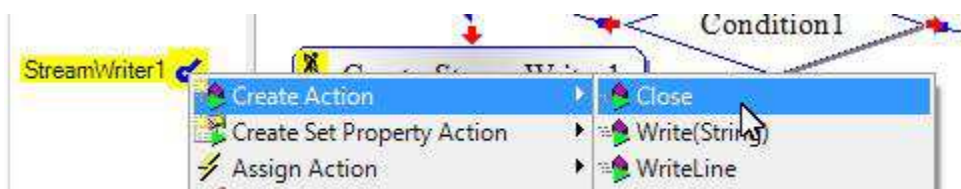


Click OK:



Link the action to the last action.

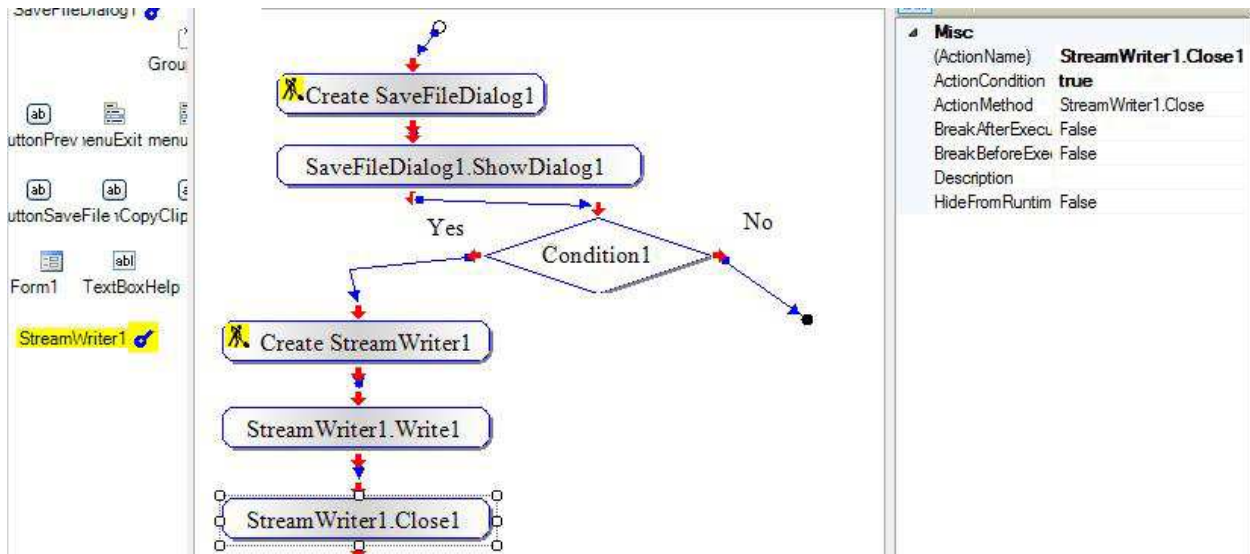
Create a Close action for the StreamWriter:



Click OK:



Link it to the last action. That is all we need to write the output to file:

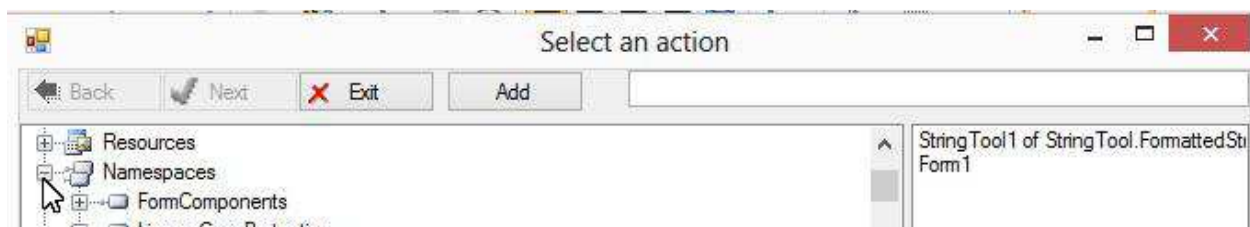


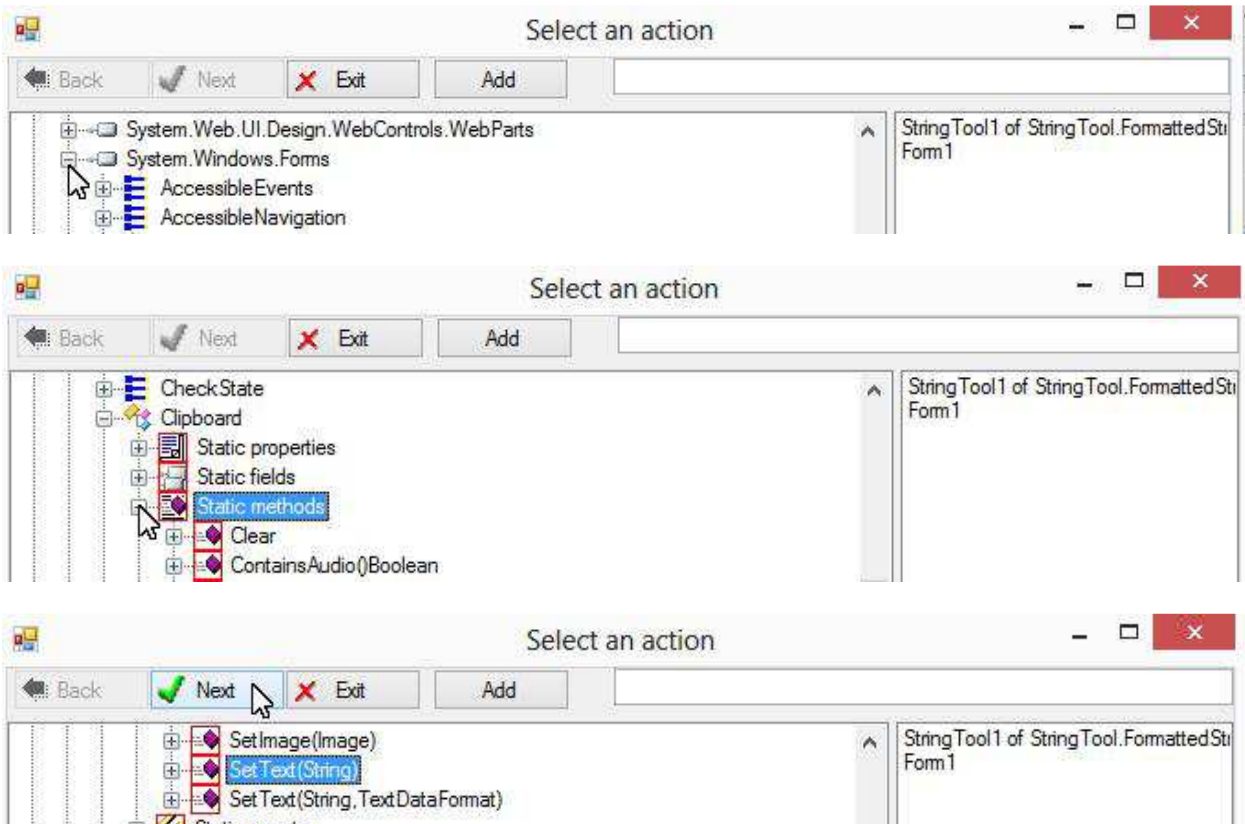
Copy to Clipboard

Right-click the button to copy the output to the Clipboard:

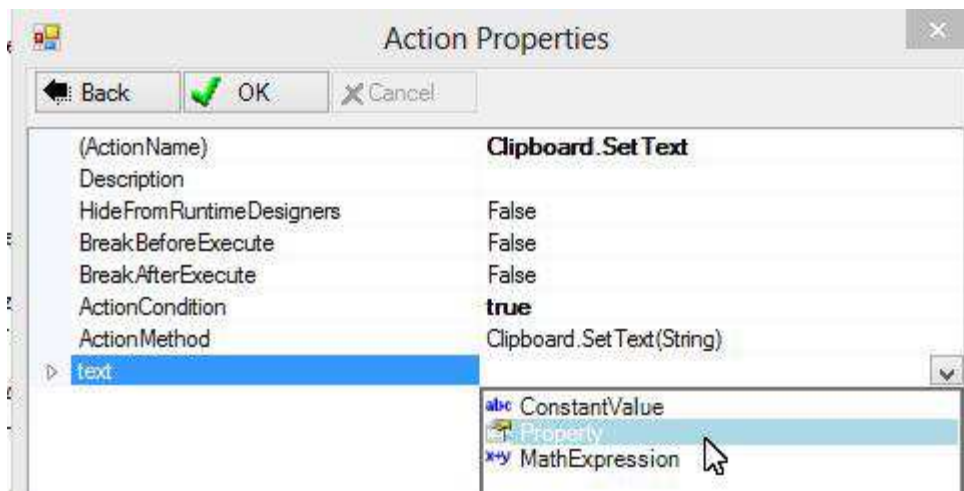


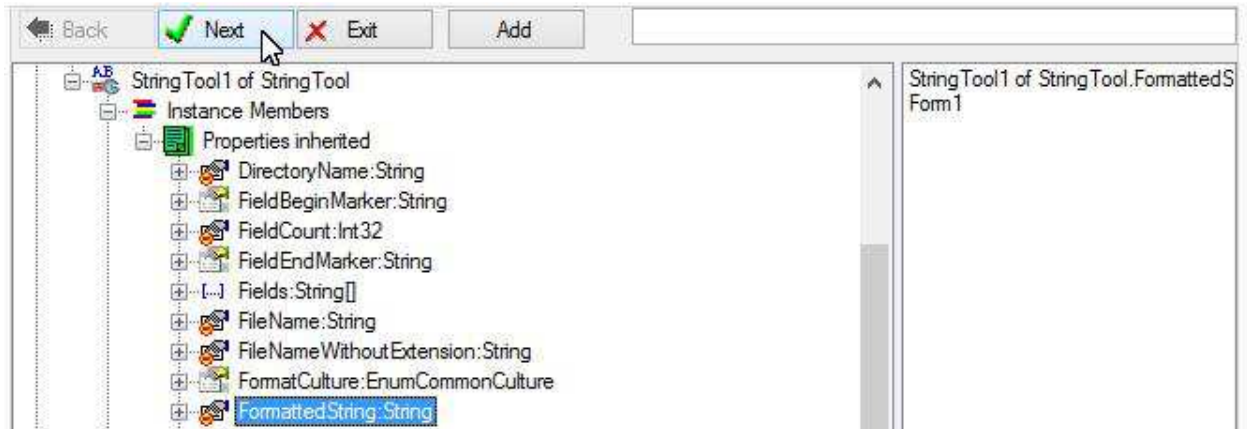
Select the SetText method of the Clipboard:



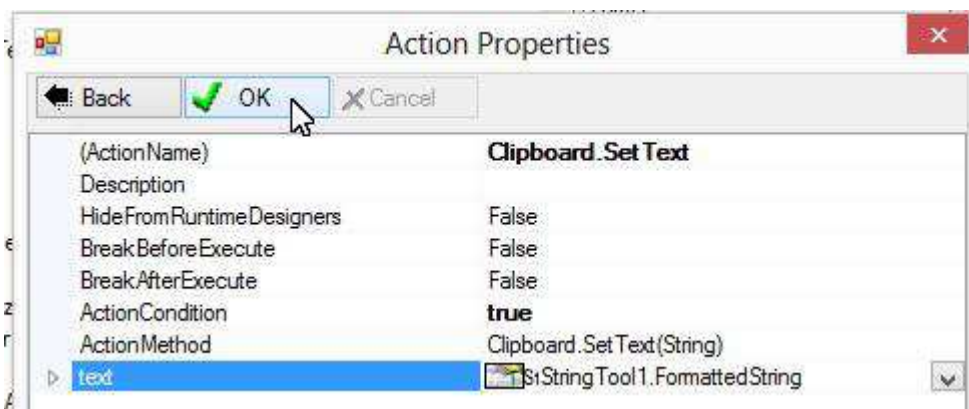


Select FormattedString of StringTool1 as the Clipboard contents:

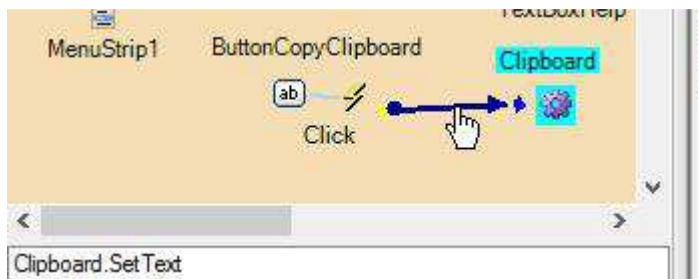




Click OK:

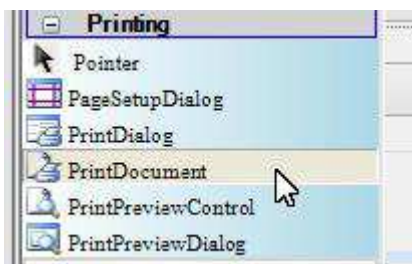


The action is created and assigned to the button:

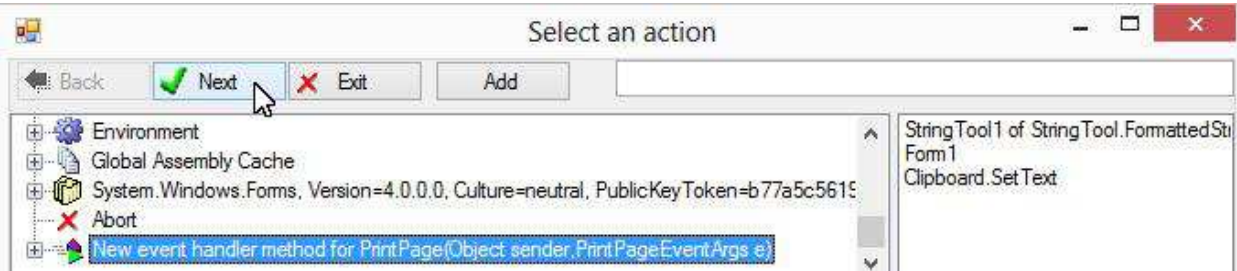
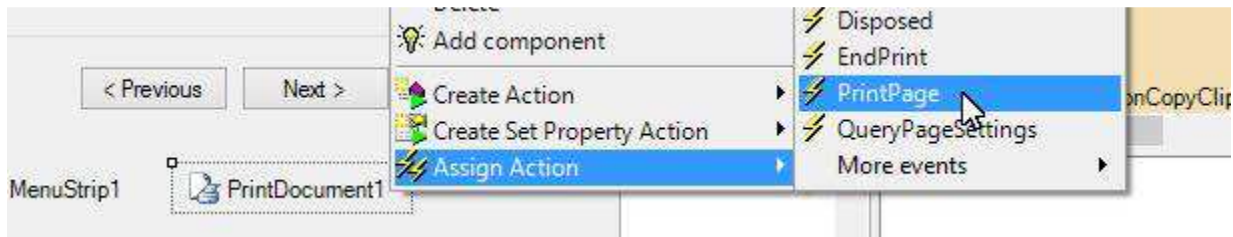


Print output

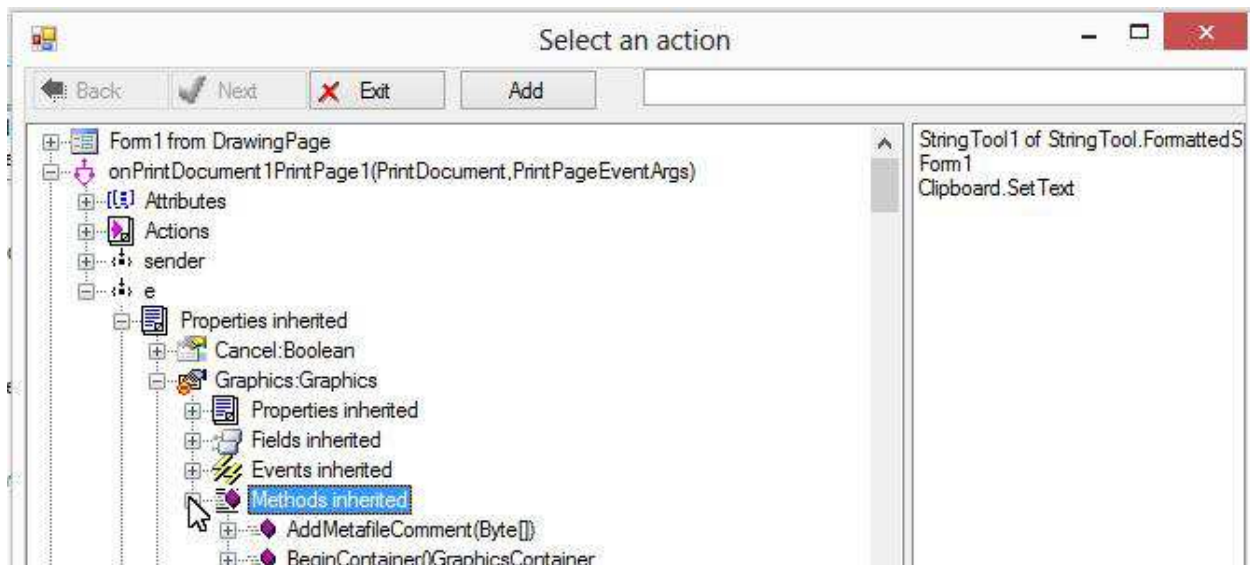
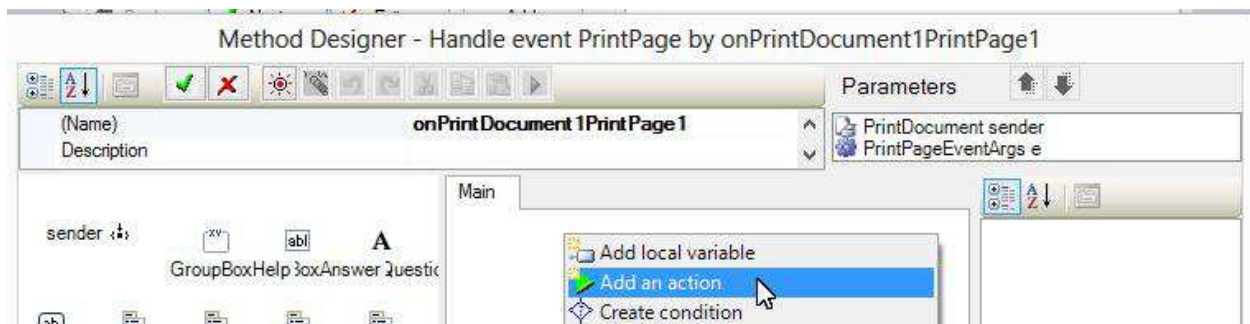
Add a PrintDocument to the form to do the printing:



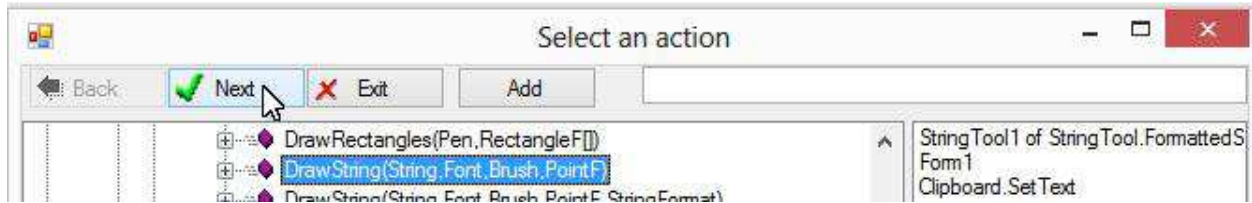
Handle PrintPage event to do the printing:



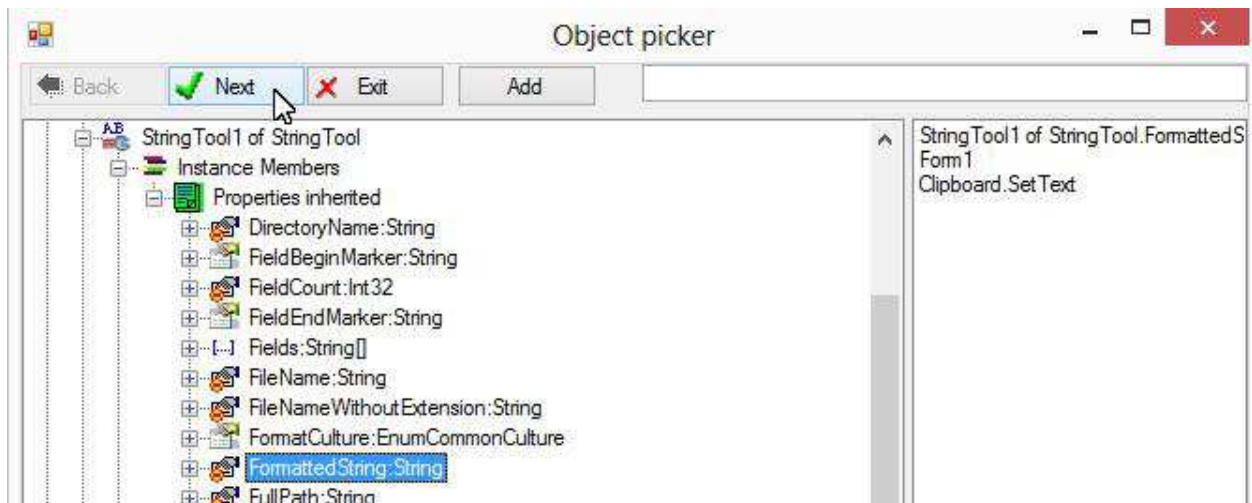
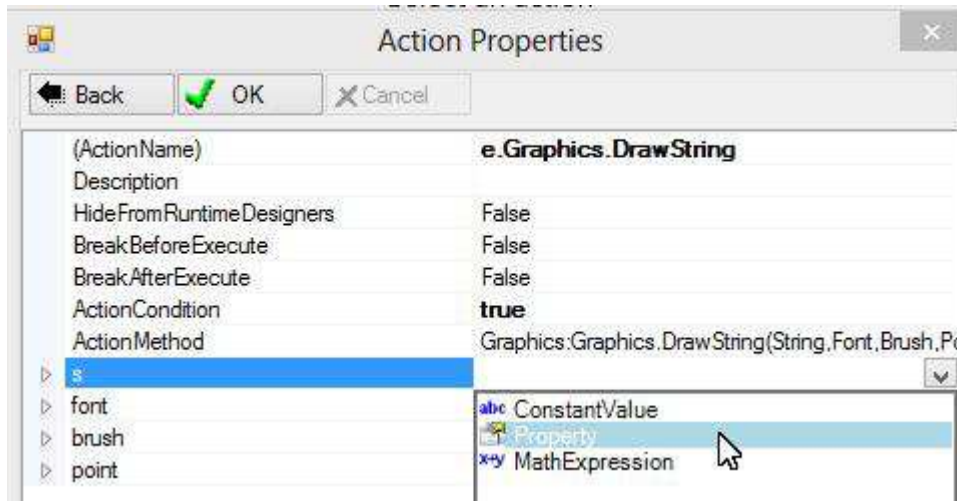
Create an action to draw output:



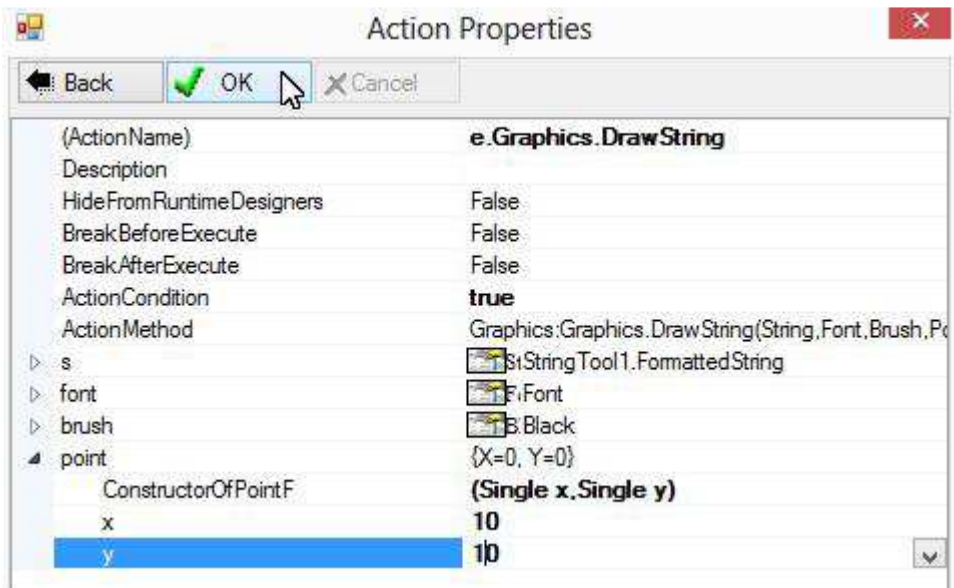
Select DrawString method:



Select FormattedString of StringTool1 as the string to draw:

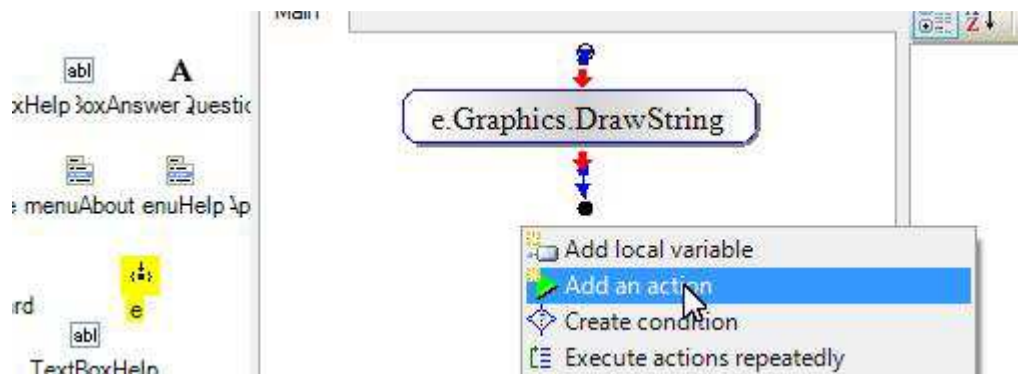


Provide font and brush for the drawing, and specify drawing starting location:

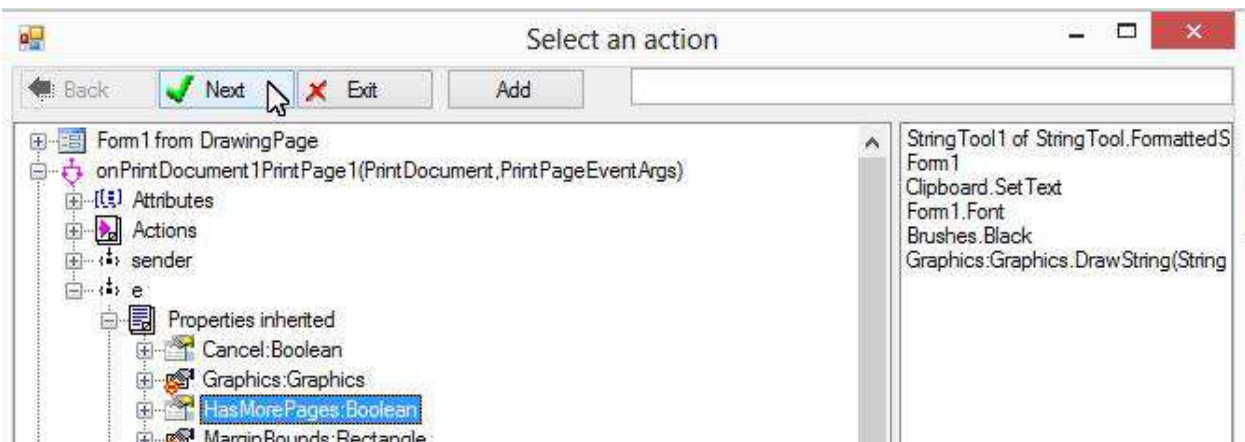


The action appears in the method.

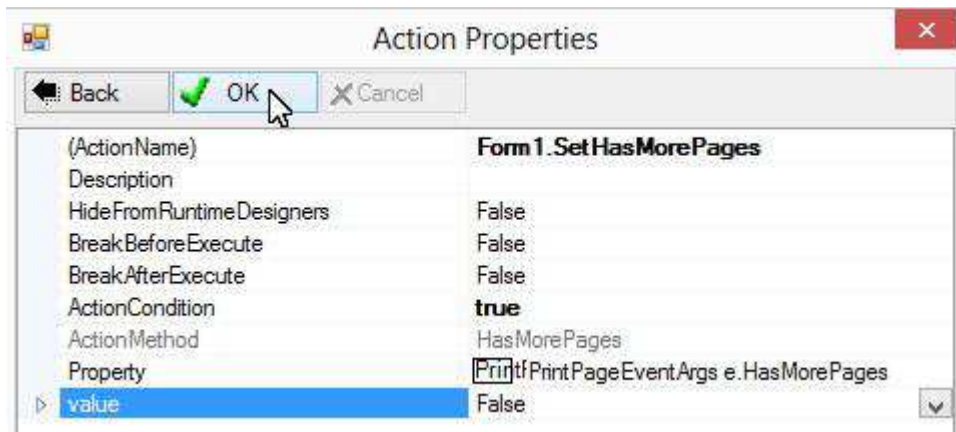
Add an action to finish printing:



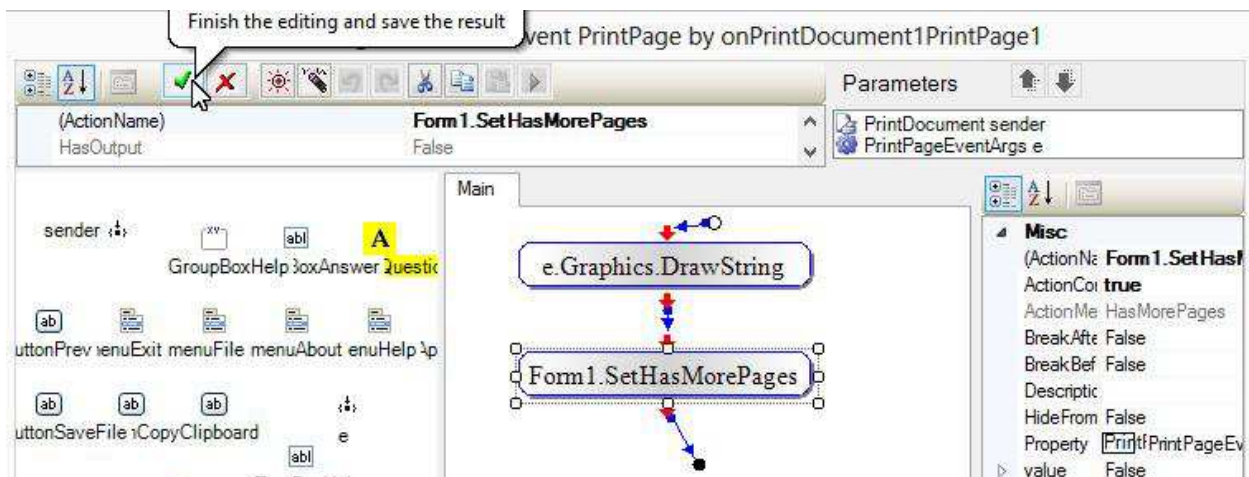
Select HasMorePage:



False for the “value” indicates that the printing should finish:

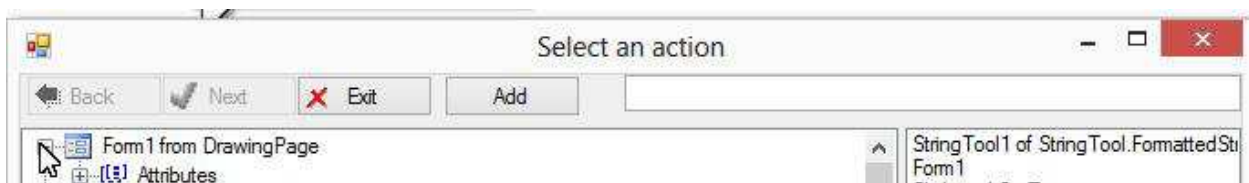
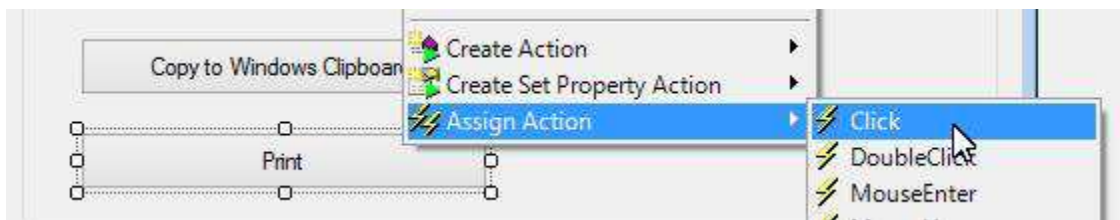


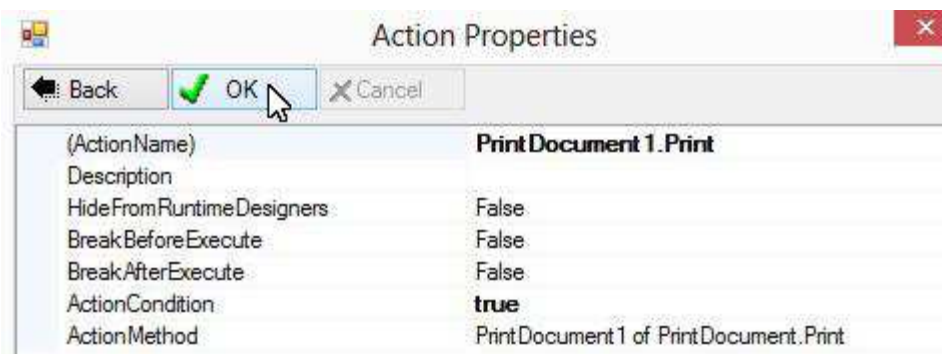
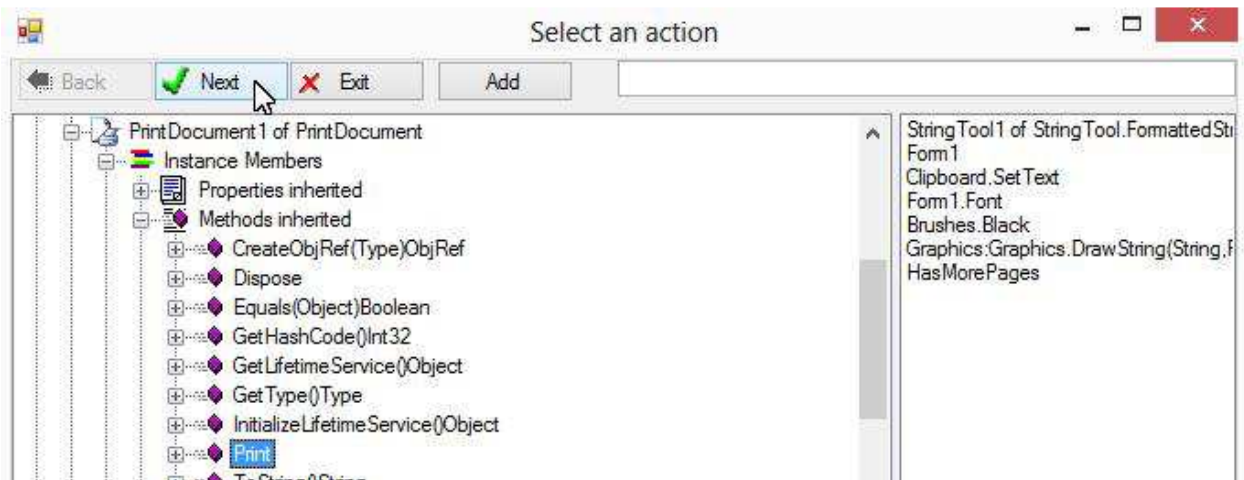
These two actions form our printing operations.



You can make very complex printing by creating various drawing actions, and use your contents to determine when to set HasMorePage to True or False for multi-page printing.

We use the button to trigger printing:





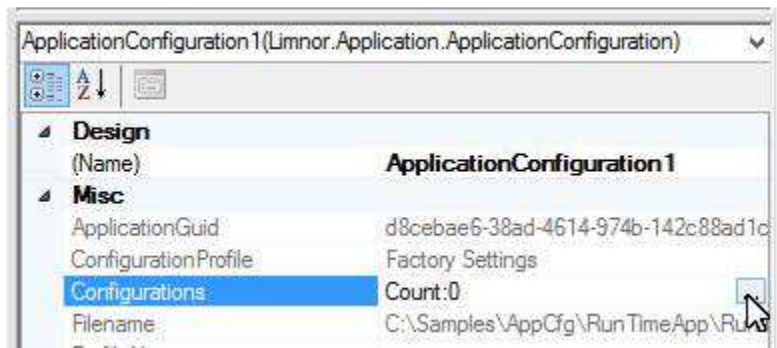
The action is created and assigned to the button:



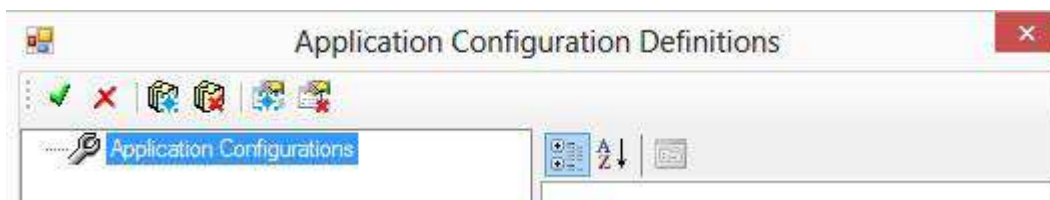
You can see that the button Click event triggers Print action; the Print action triggers the event handler method onPrintDocument1PrintPage1.

Create configurations

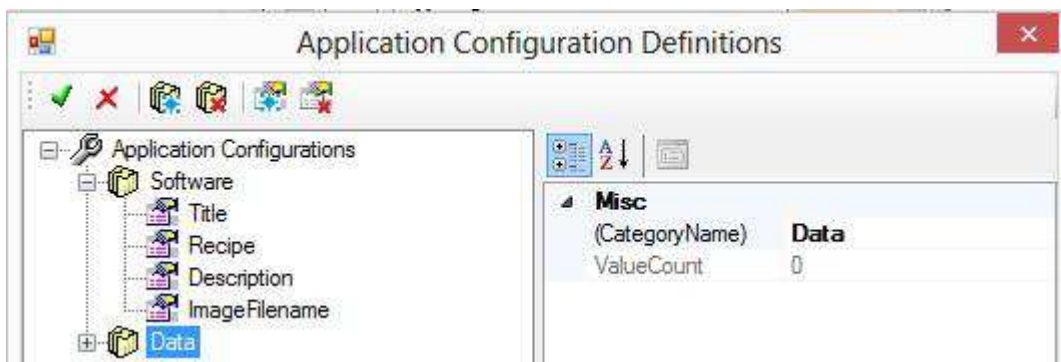
Click "..." for Configurations property of ApplicationConfiguration1 to manage configurations:



A dialogue box appears for us to manage all configurations:



We are not going into details of creating configurations. You may read <http://www.limnor.com/support/UseAppConfig.pdf> for how to create configurations.



Values under “Software” category are for customizing this application. Value for “Recipe” is the document template.

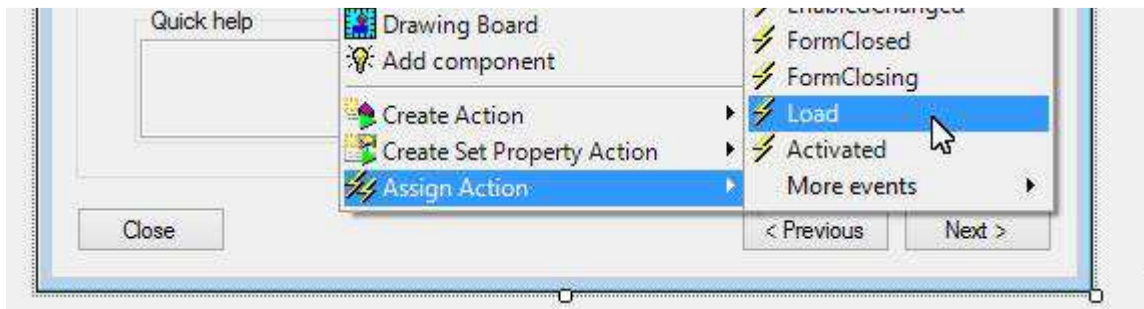
Category “Data” will hold questions and help text for each “tag” in the “Recipe”. We use the following naming conventions for values under “Data”. The value name for a question for a “tag” is formed by {tag}_Q; the value name for help text for a “tag” is formed by {tag}_H. For example, if a tag is named “CompanyName” then the question for the tag is saved under “Data” category by name “CompanyName_Q”; the help text for the tag is saved under “Data” category by name “CompanyName_H”.

Load configurations

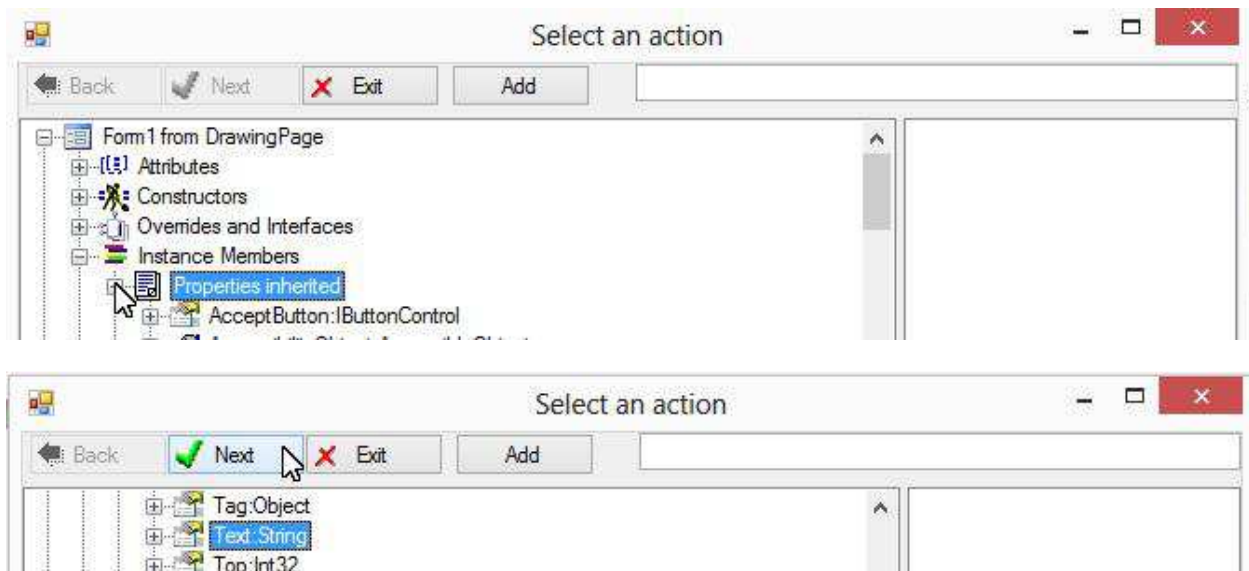
We load configurations at the Load event of the form.

Show software title

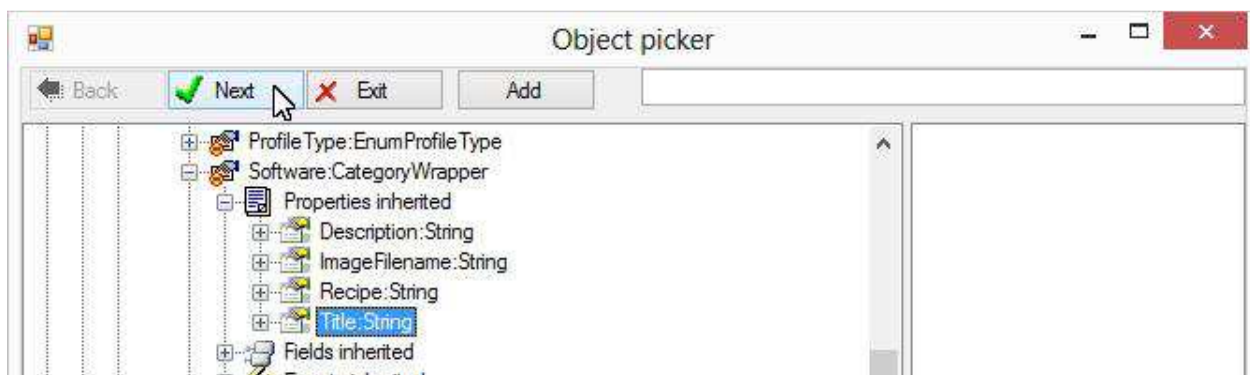
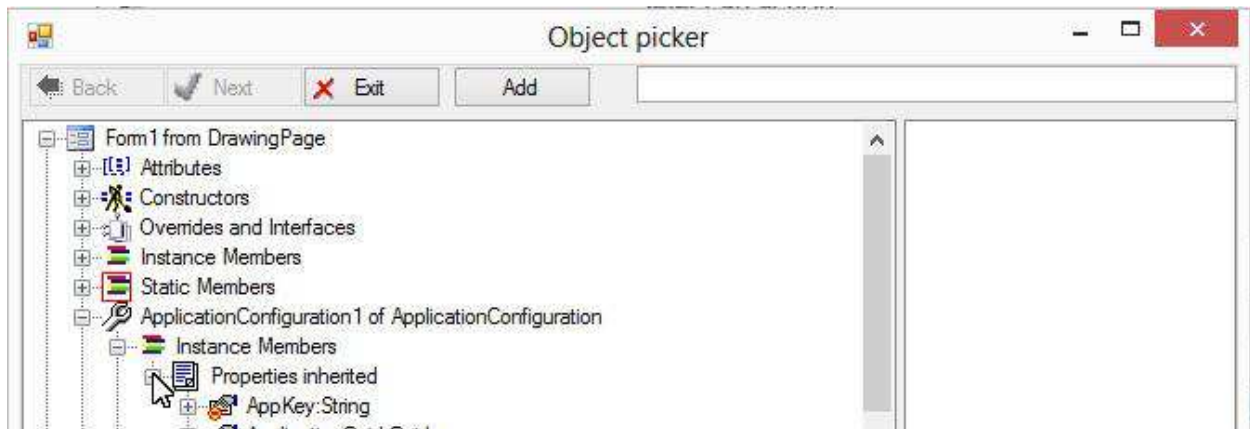
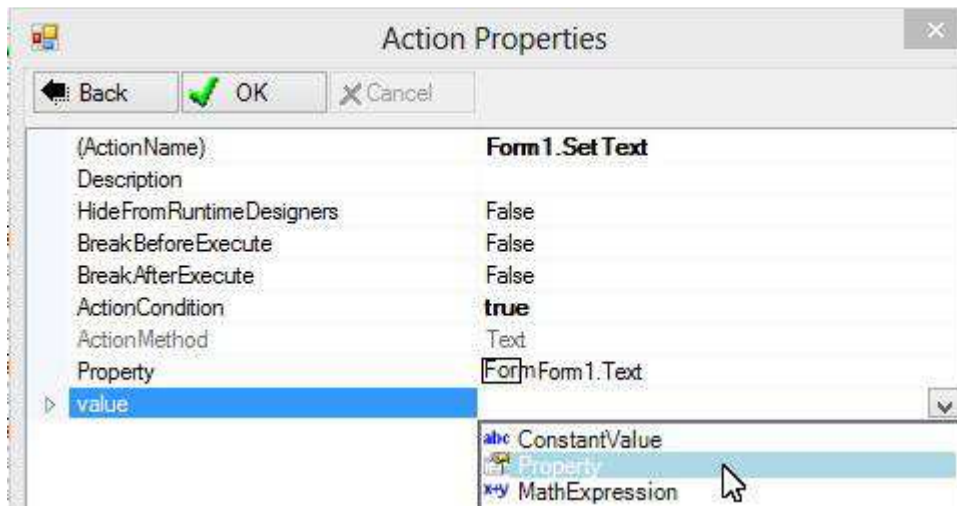
Set the form caption to the value named "Title" in the configuration. Right-click the form; choose "Assign actions"; choose "Load" event:



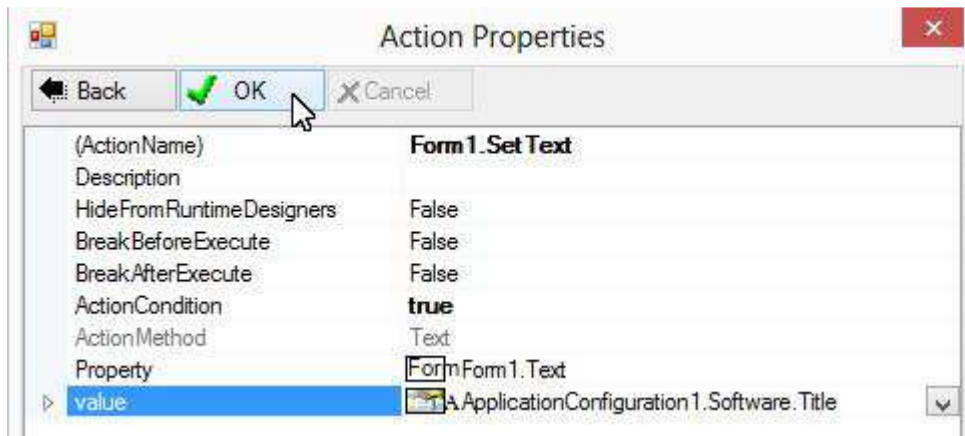
Select the Text property of the form:



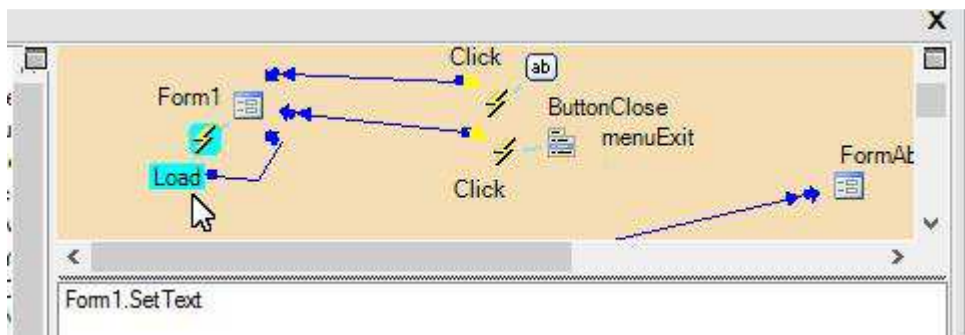
Select the configuration value "Title" for the value:



Click OK:

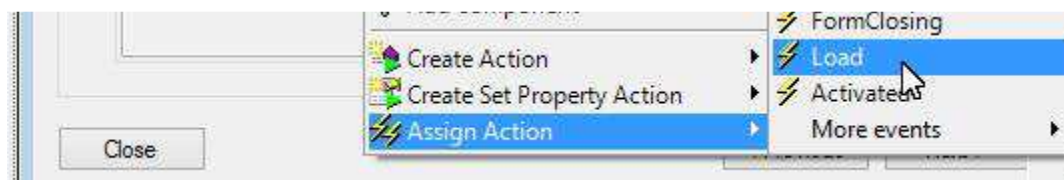


The action is created and assigned to the event:



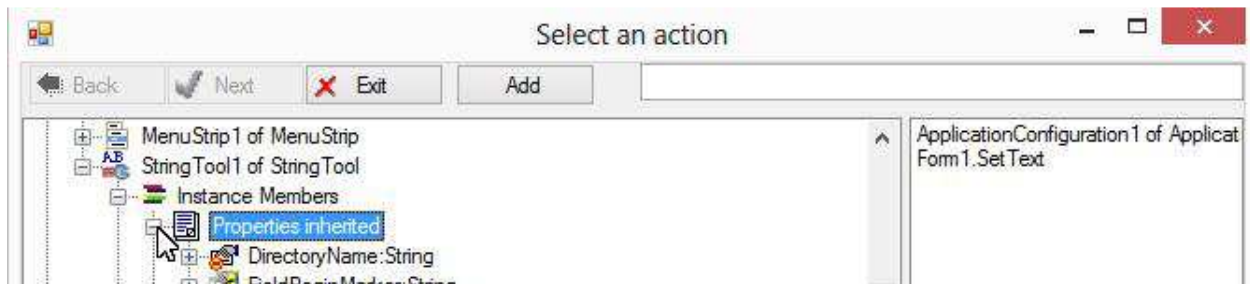
Load document template

We also do it at the Load event of the form.

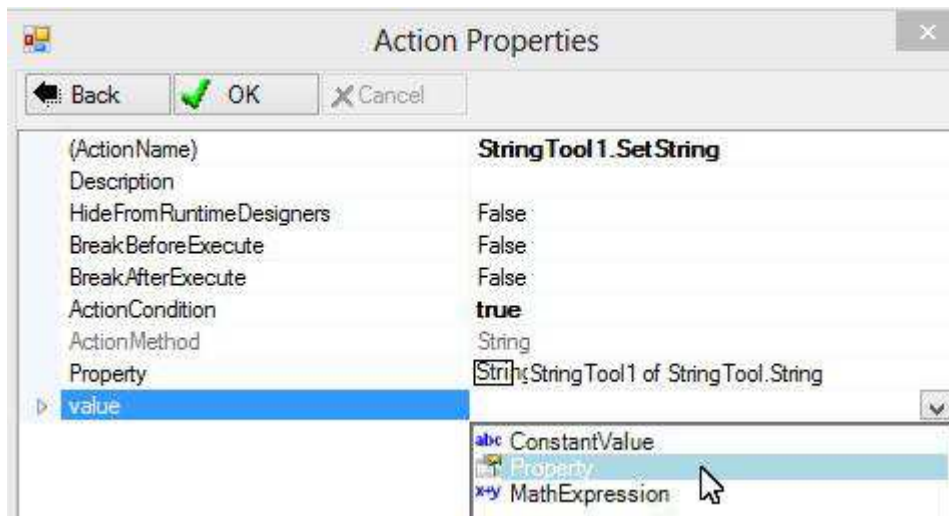


Select the String property of StringTool1:

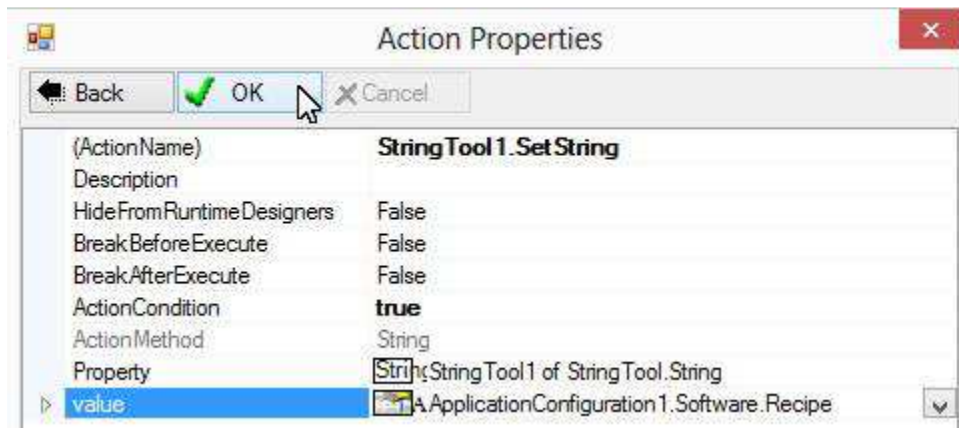




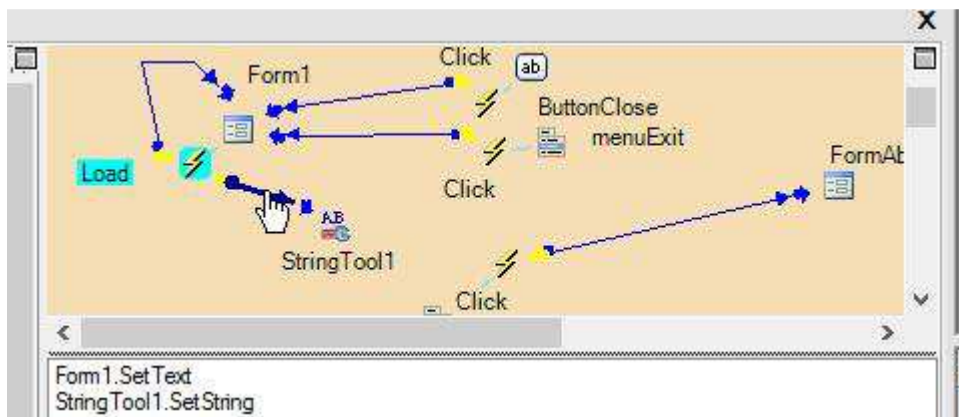
Select "Recipe" for the value:



Click OK:



The action is created and assigned to the event:

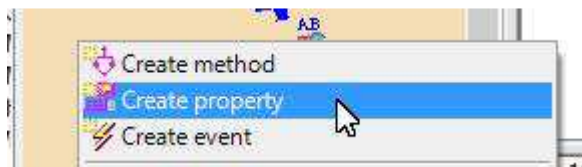


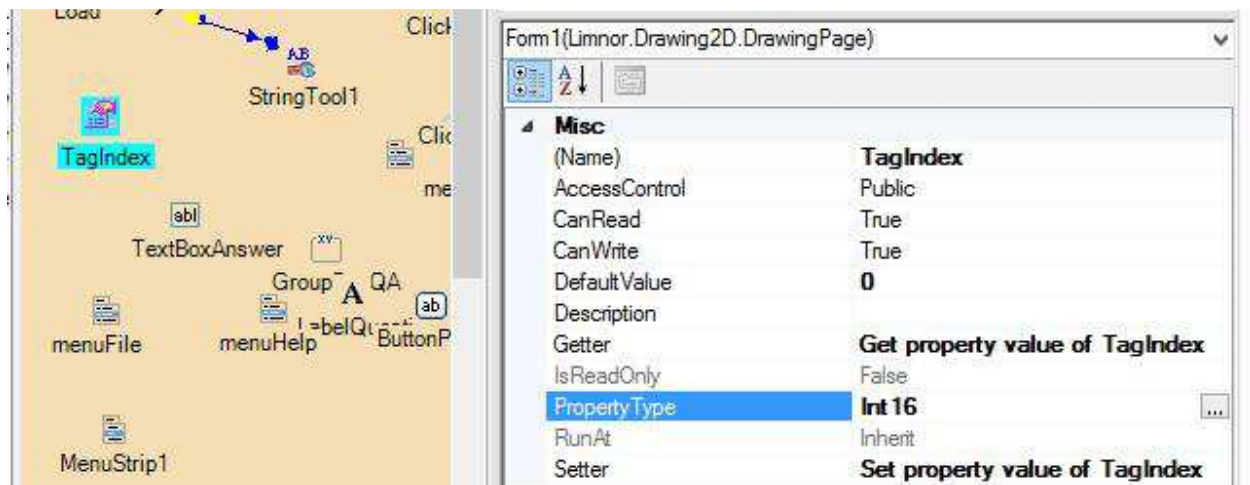
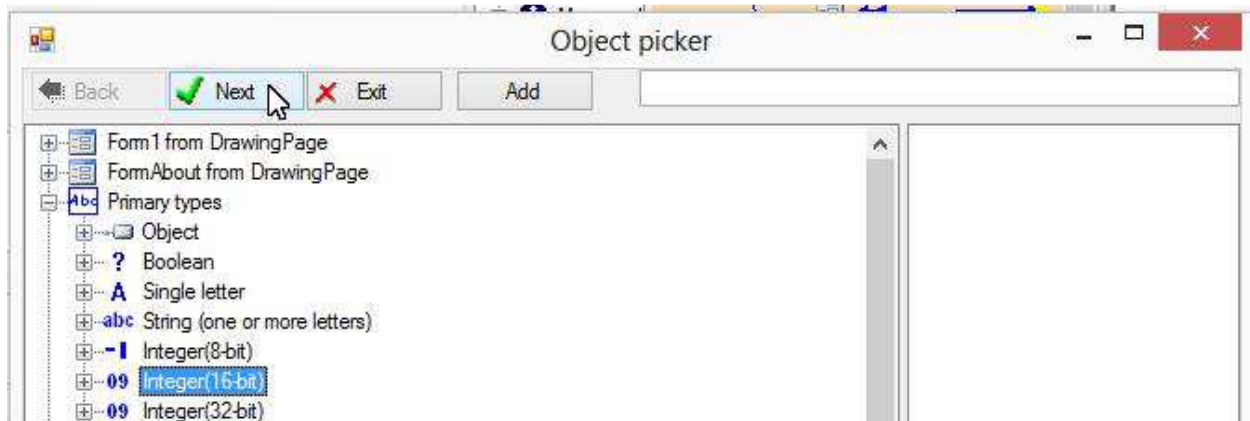
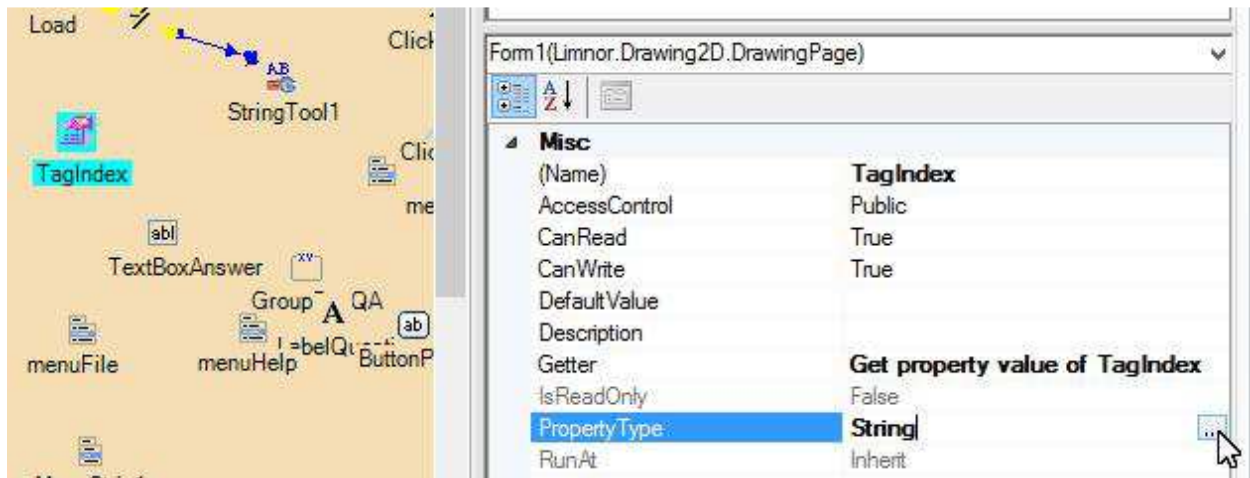
Collect User Inputs

We'll display each question and help on the form to get user input.

Tag Index

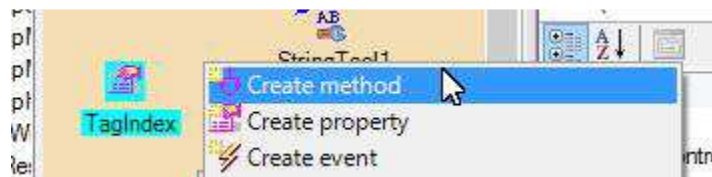
We create an integer property to indicate the question to be displayed. The value of the property should be from 0 to the number of the tags minus 1. 0 indicates the first question, 1 the second, and so on.





Show question

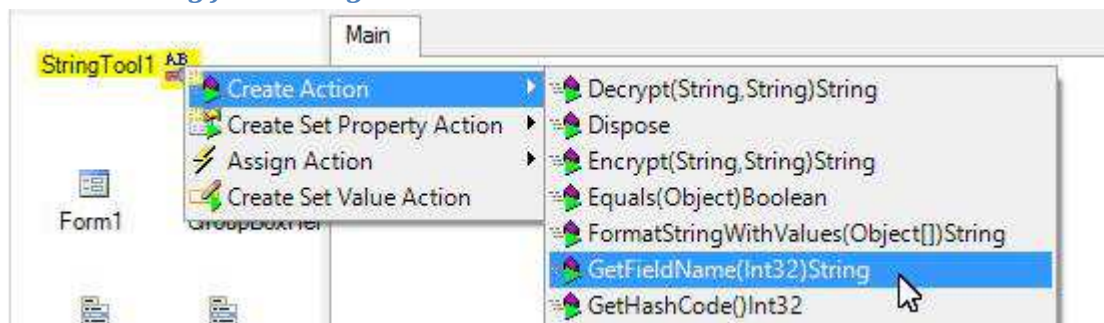
We create a method to show a question.



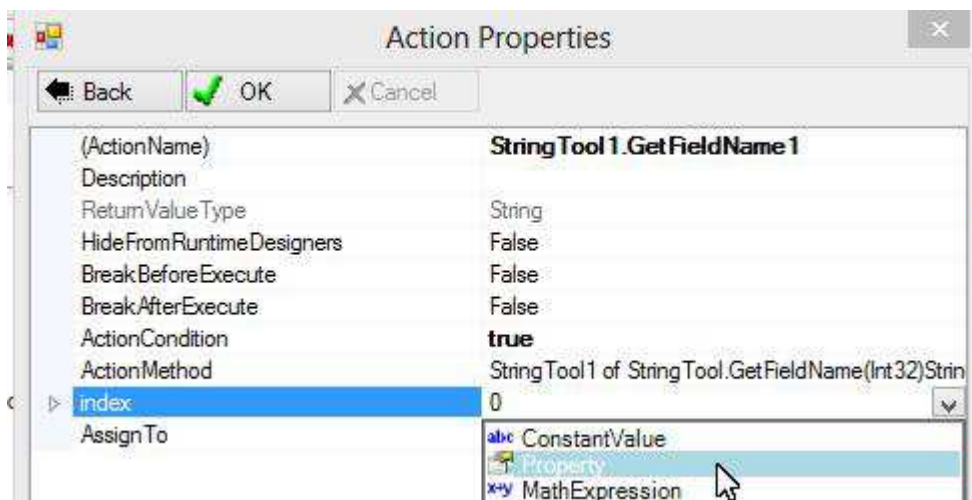
Name it "showQuestion":

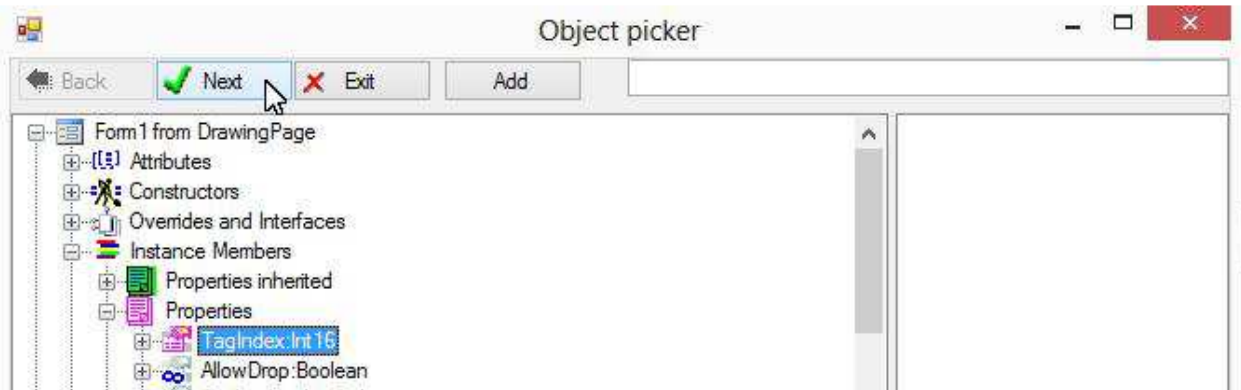


Get current tag from StringTool1

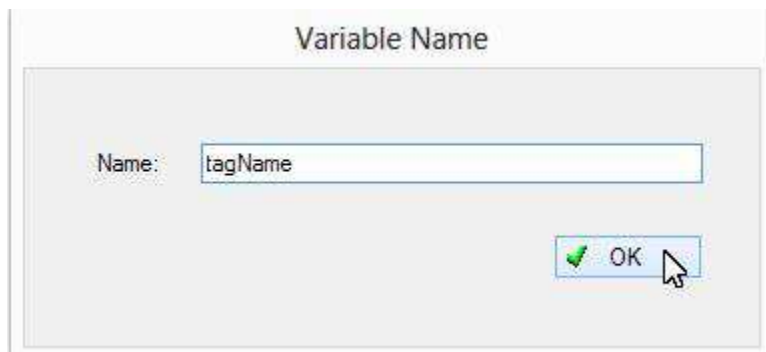
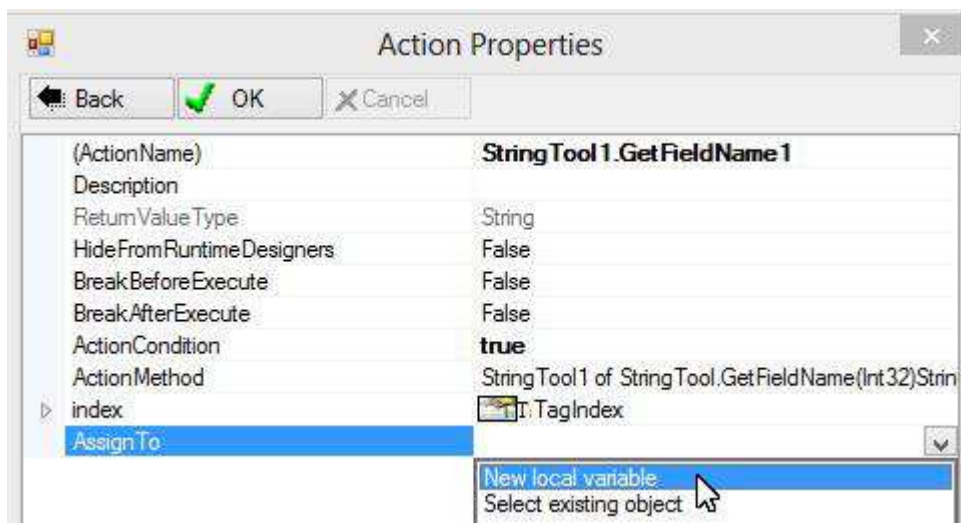


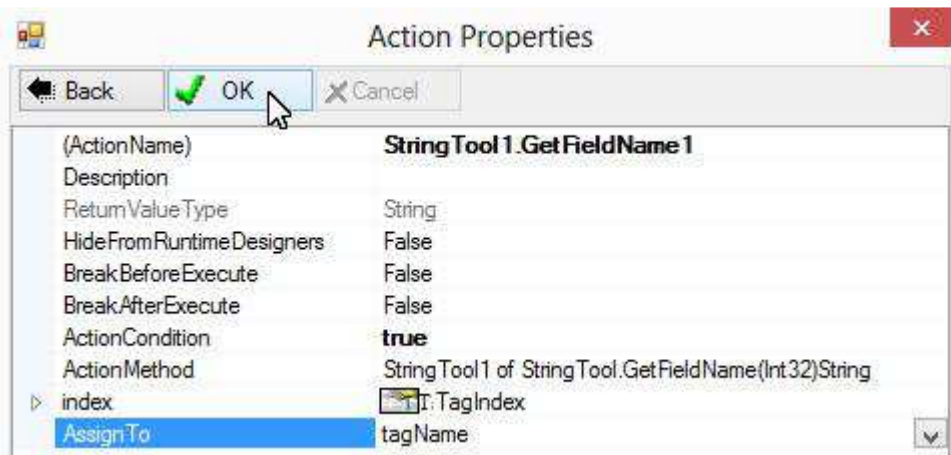
Provide the tag index:



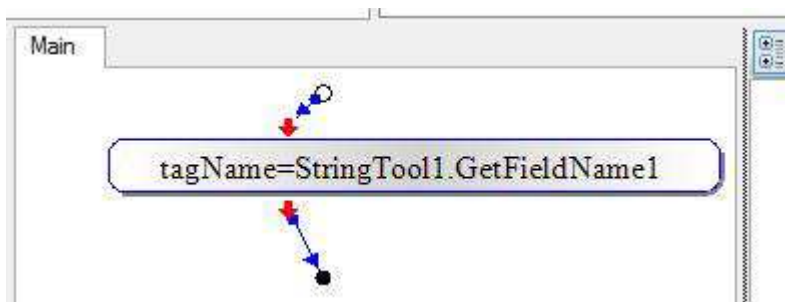


Set the result to a variable:

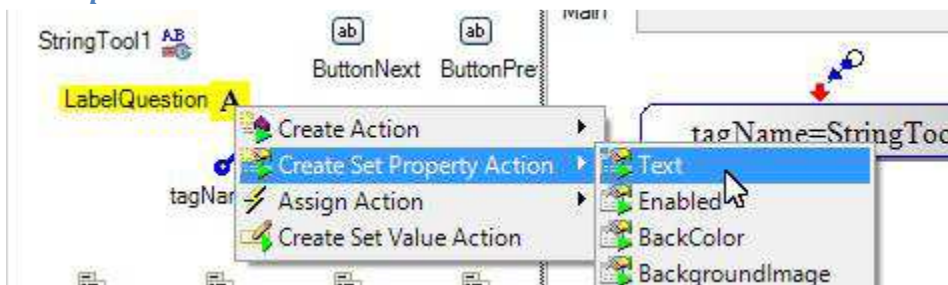




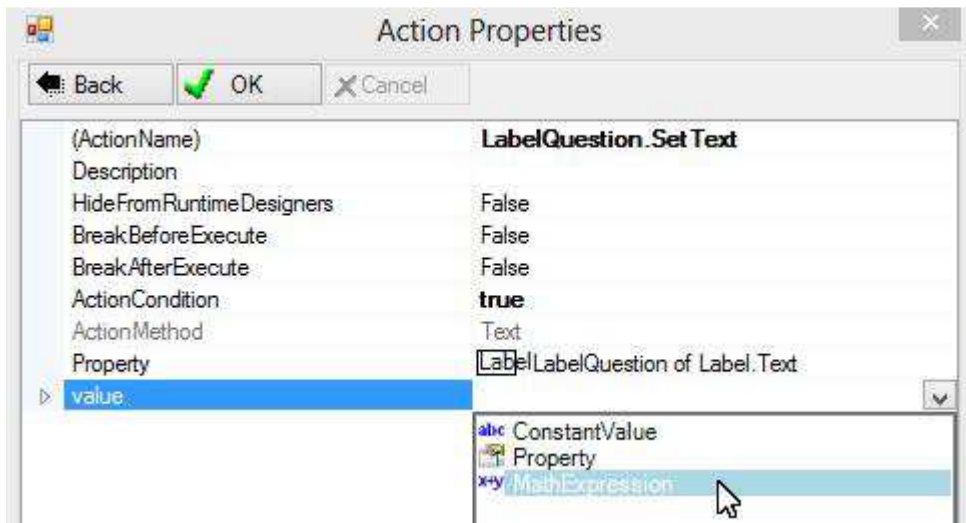
An action appears:



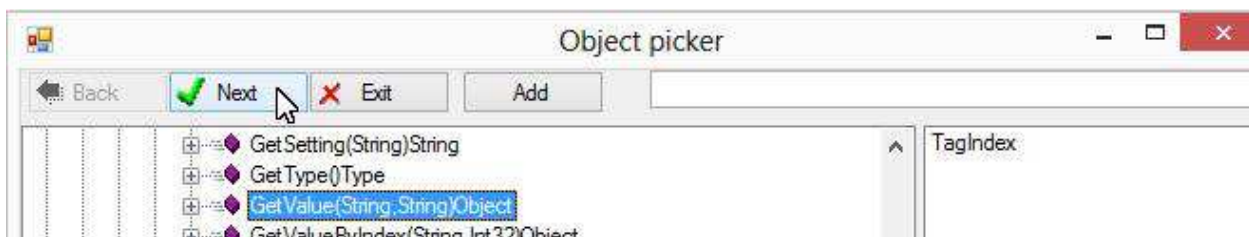
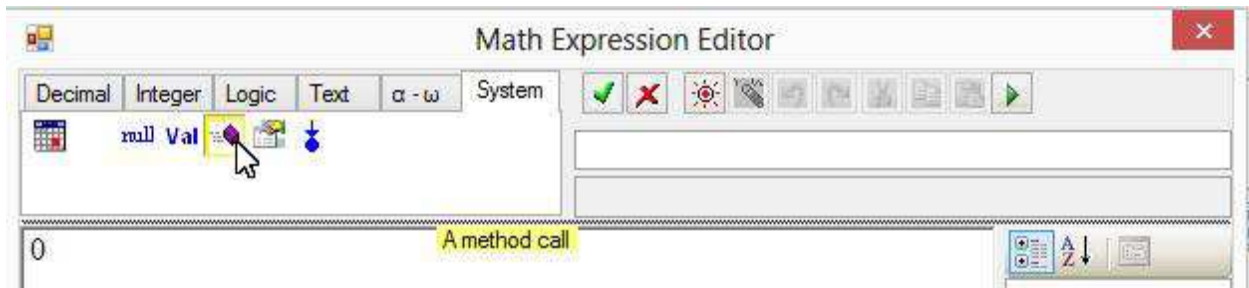
Show question



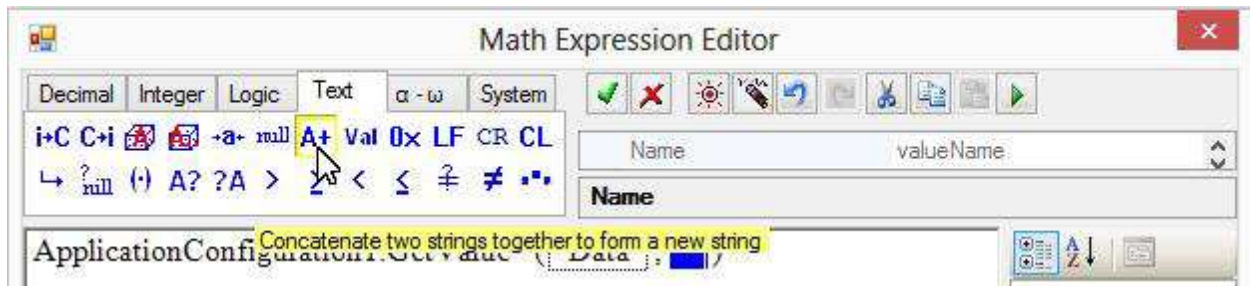
Use an expression to form the question:



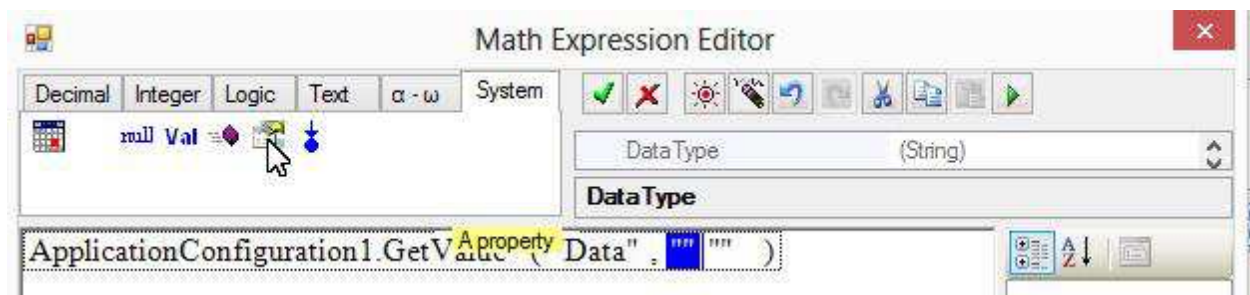
Call a method to get question from the configuration:



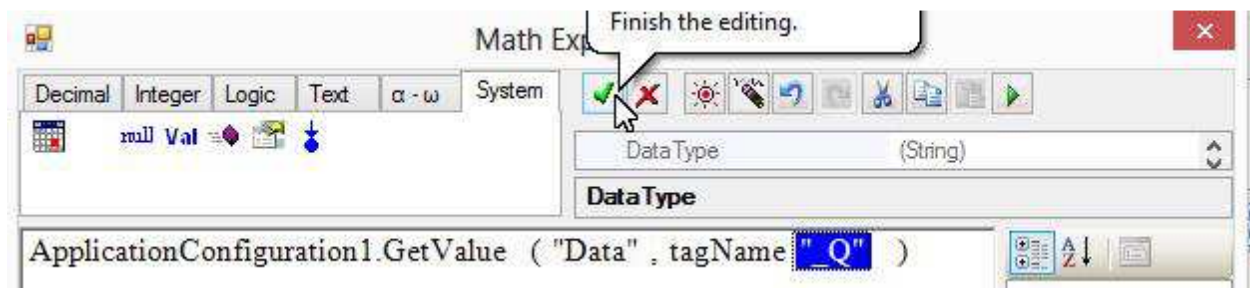
The first parameter is the configuration category, we use "Data". The second parameter is the value name; it is formed by tag name and "_Q". Click A+ to use string concatenation:

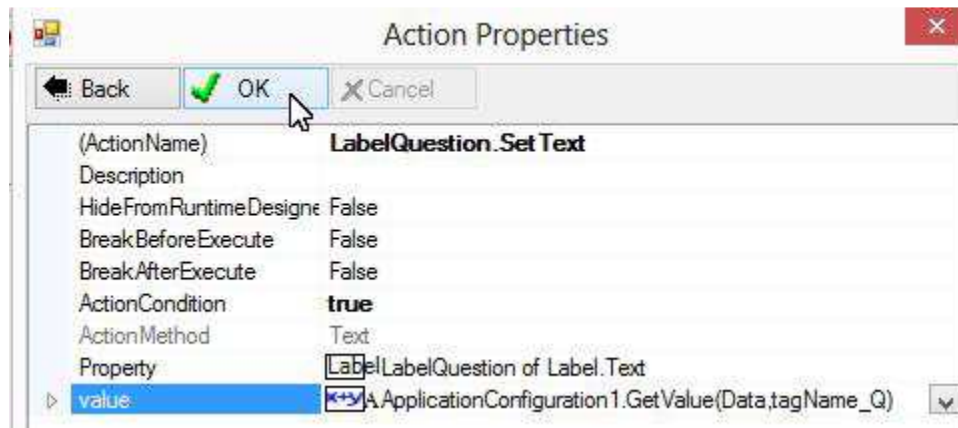


Select tag name for the first part:

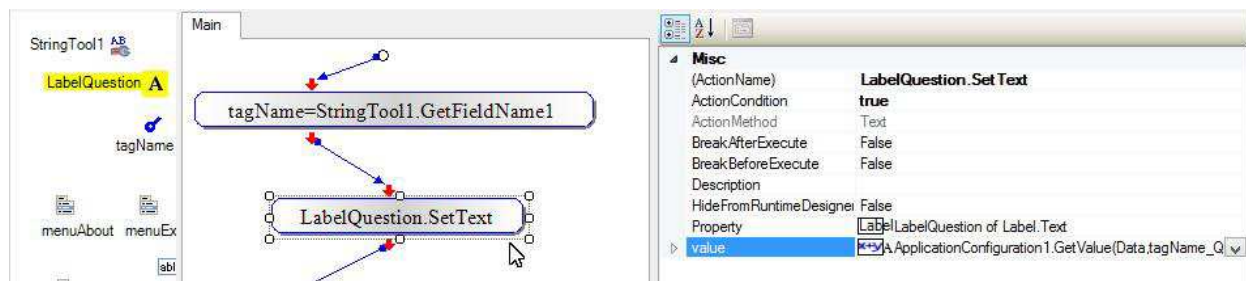


The second part is "_Q":



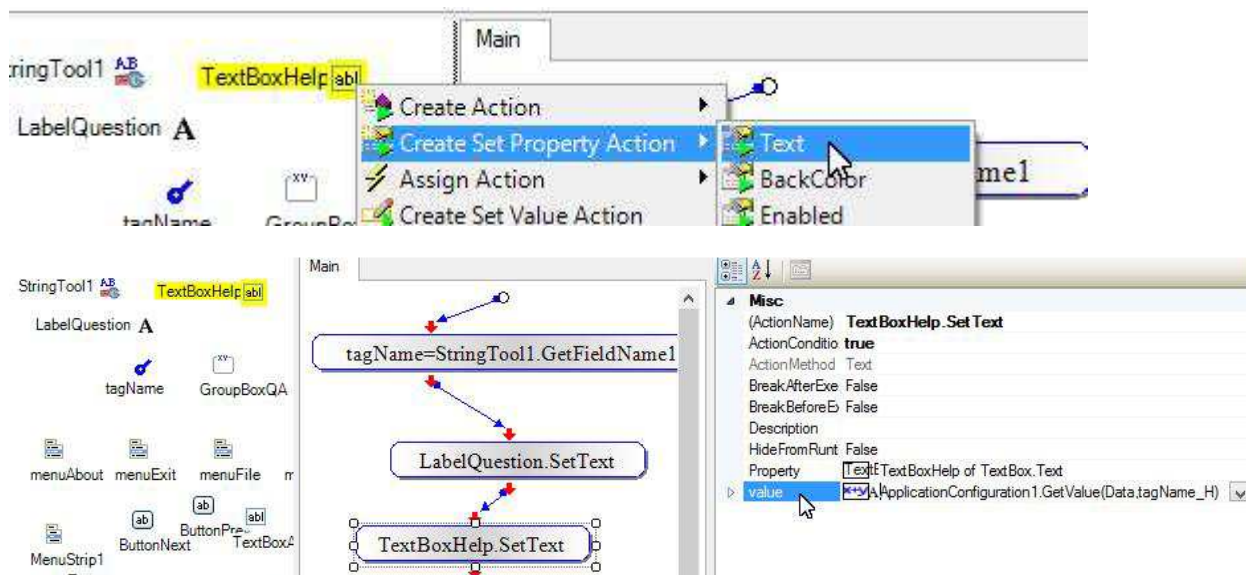


Link the actions:



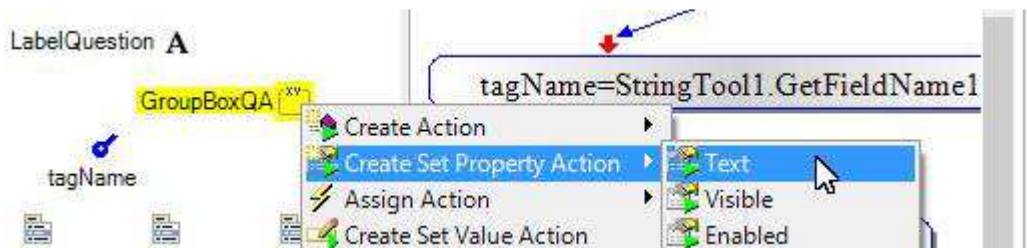
Show help

It is the same process as above.

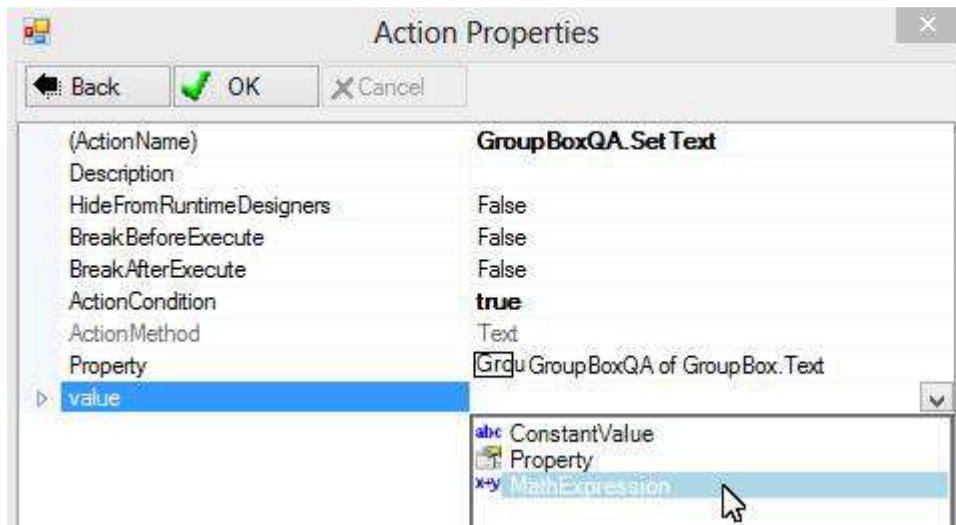


Show question number

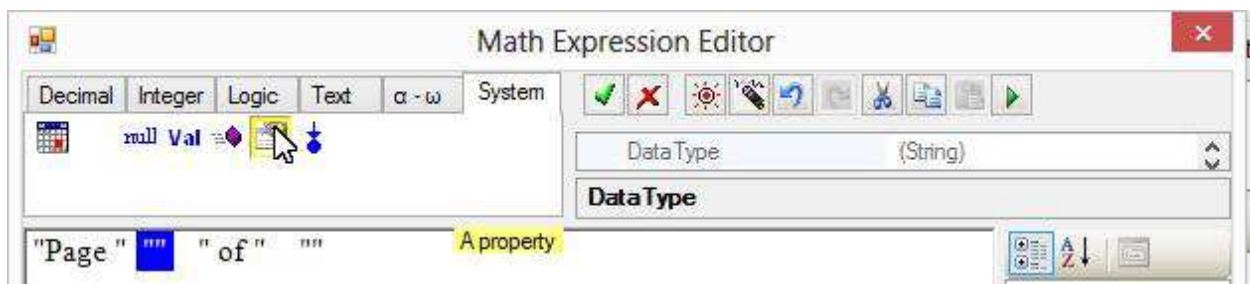
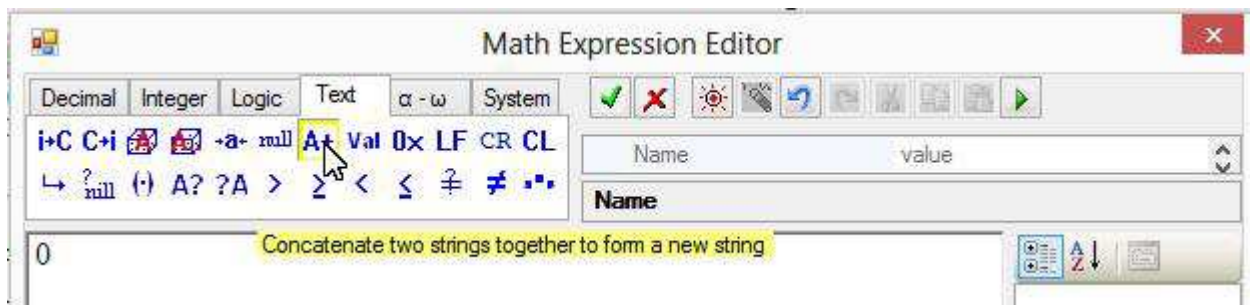
We want to show "Page n of m", where n is the tag index and m is the number of tags.

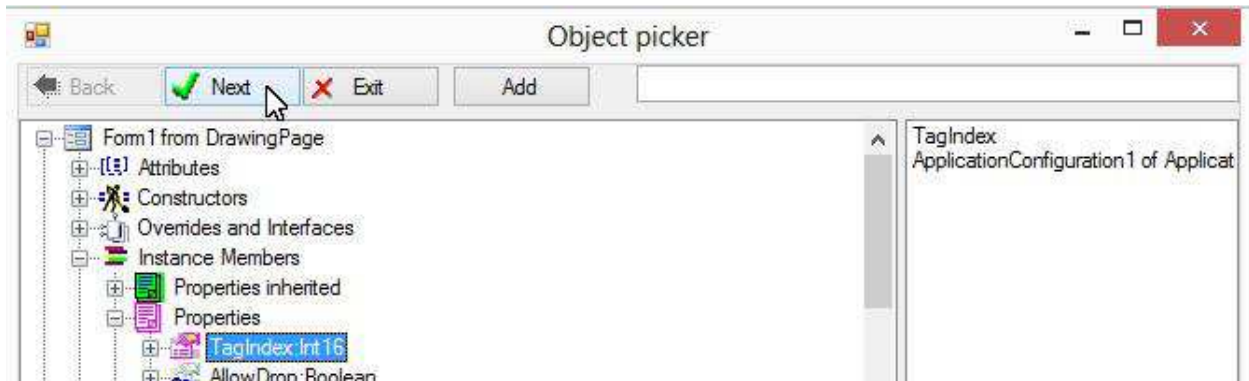


Use an expression to form the text:

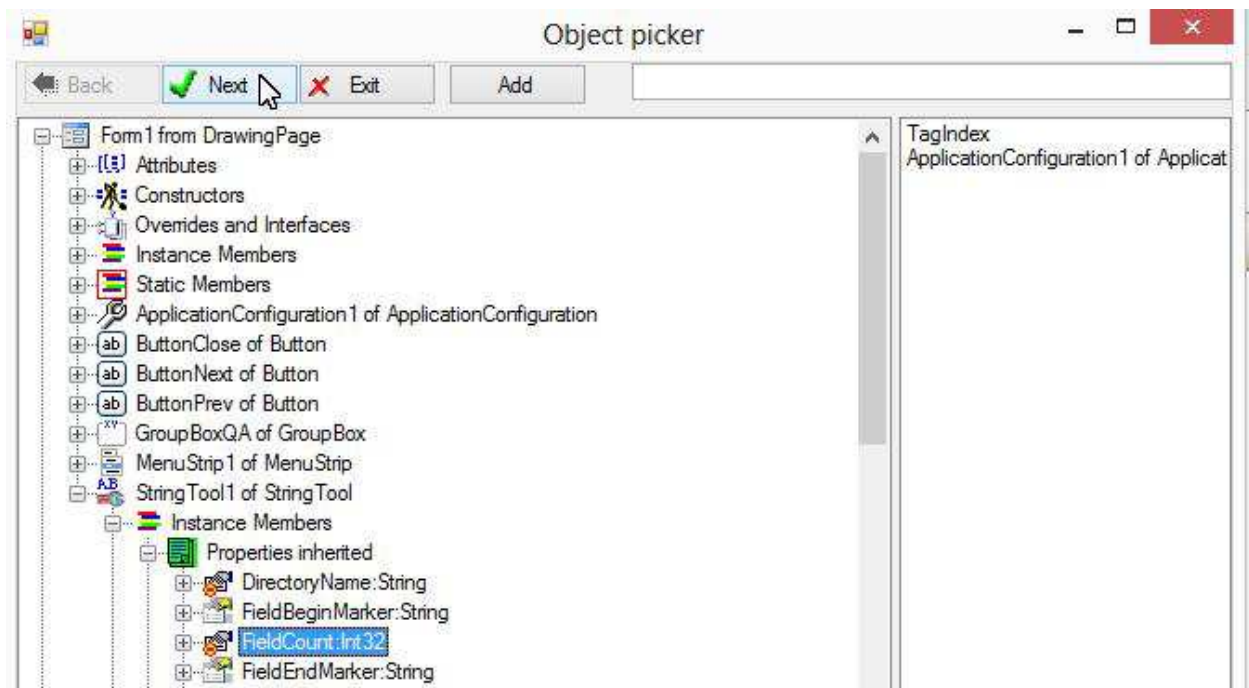
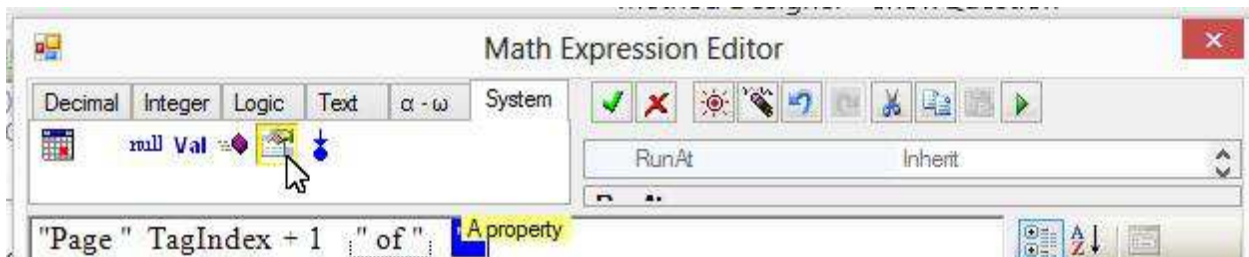


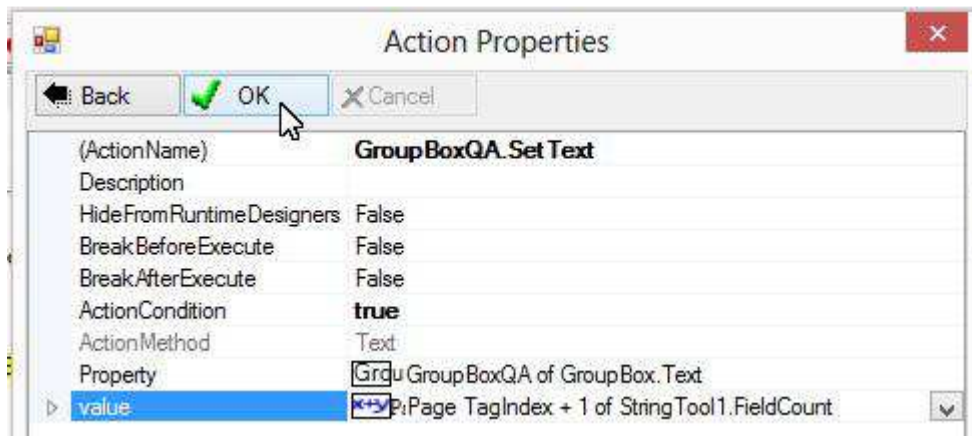
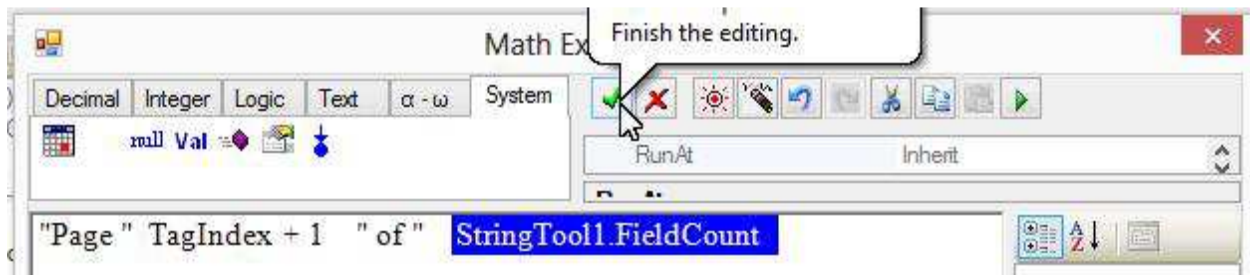
Create 4 parts:





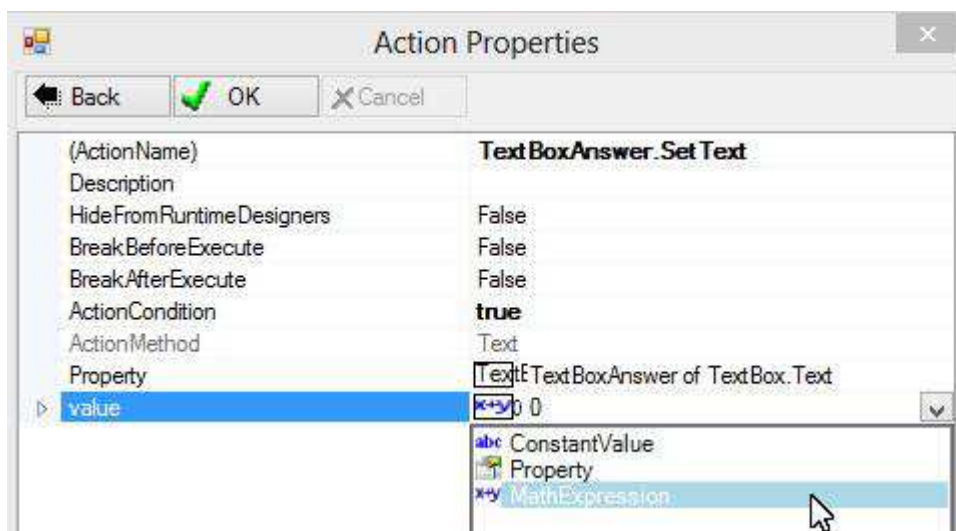
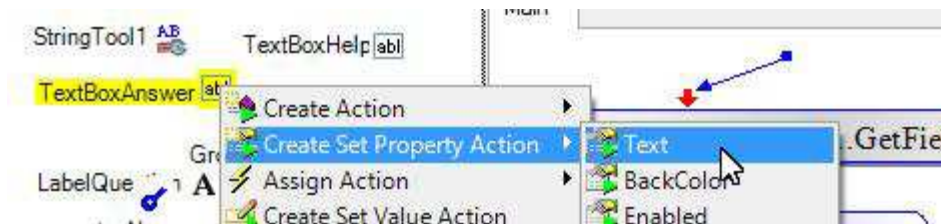
Because TagIndex starts from 0, we add 1 to it to display it from 1. The last part if the number of the tags:



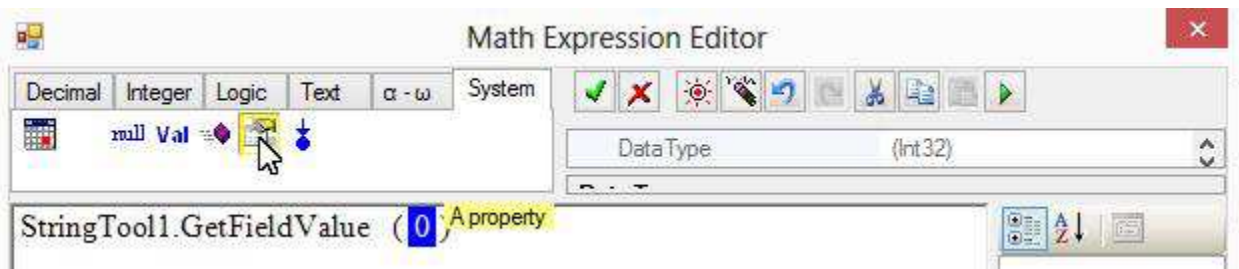
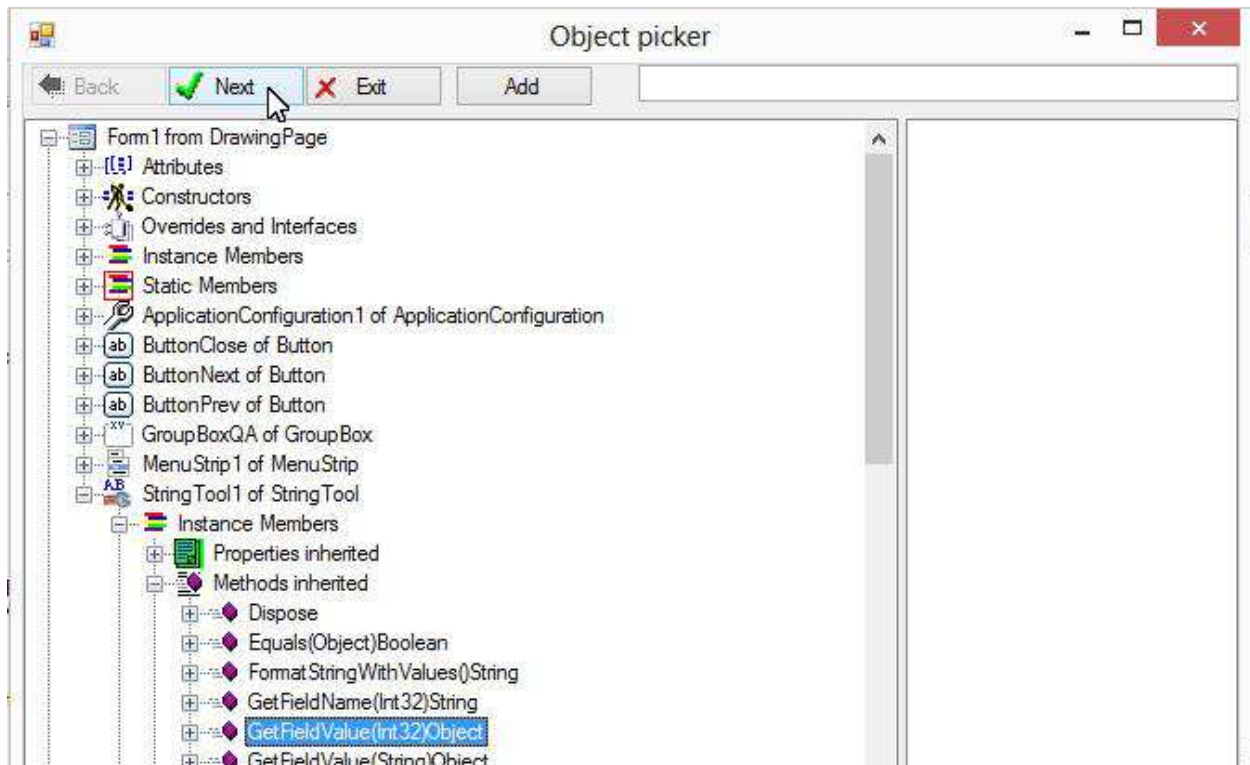


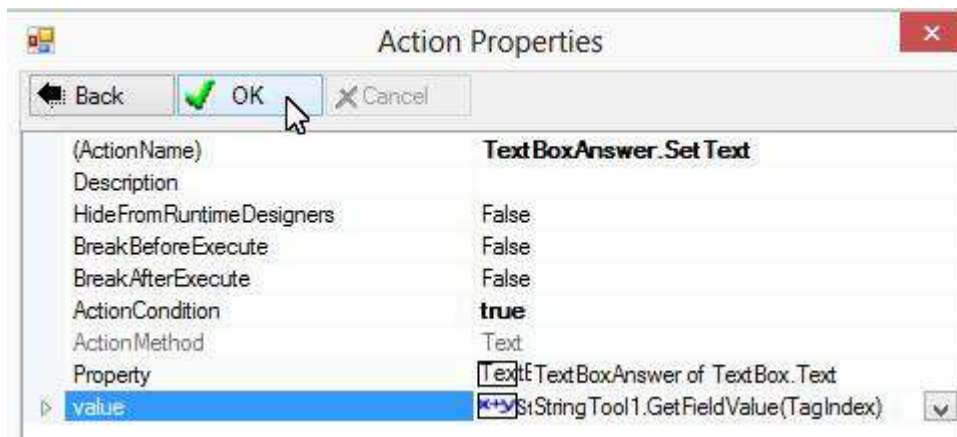
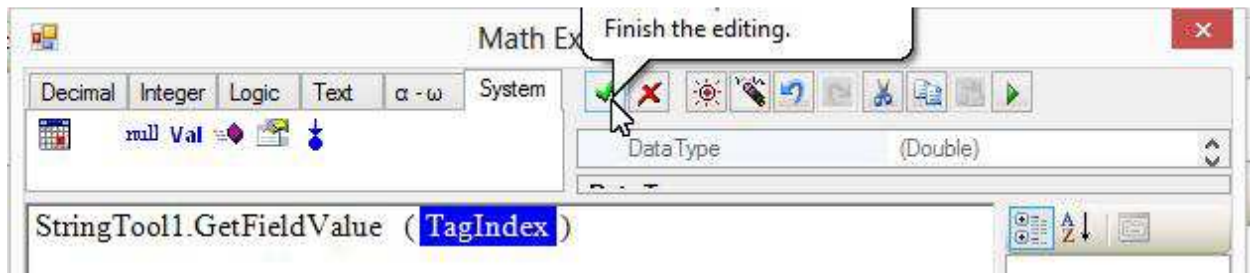
Link the actions.

Show value

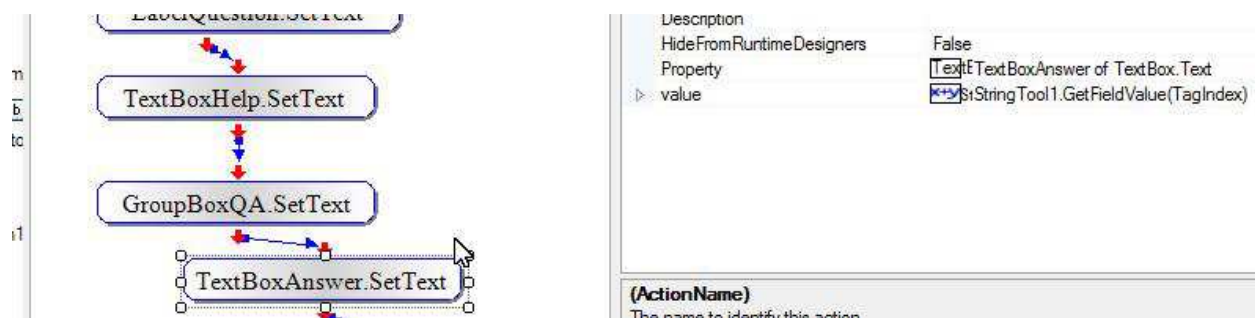


Get the value from the StringTool:

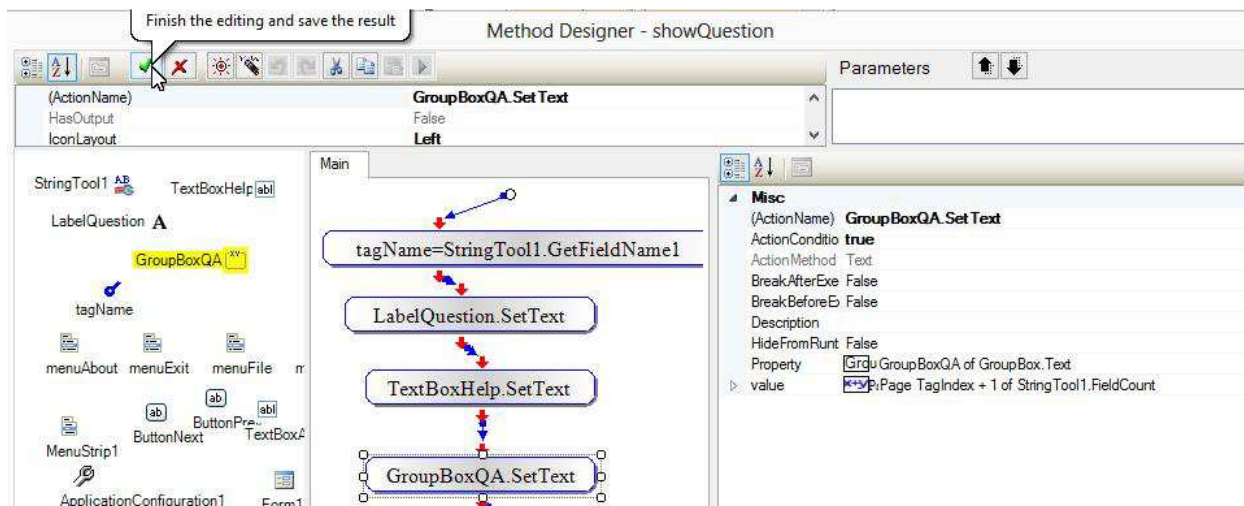




Link the action:

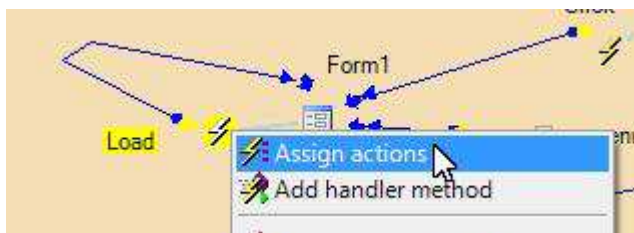


That is all we need for the method:

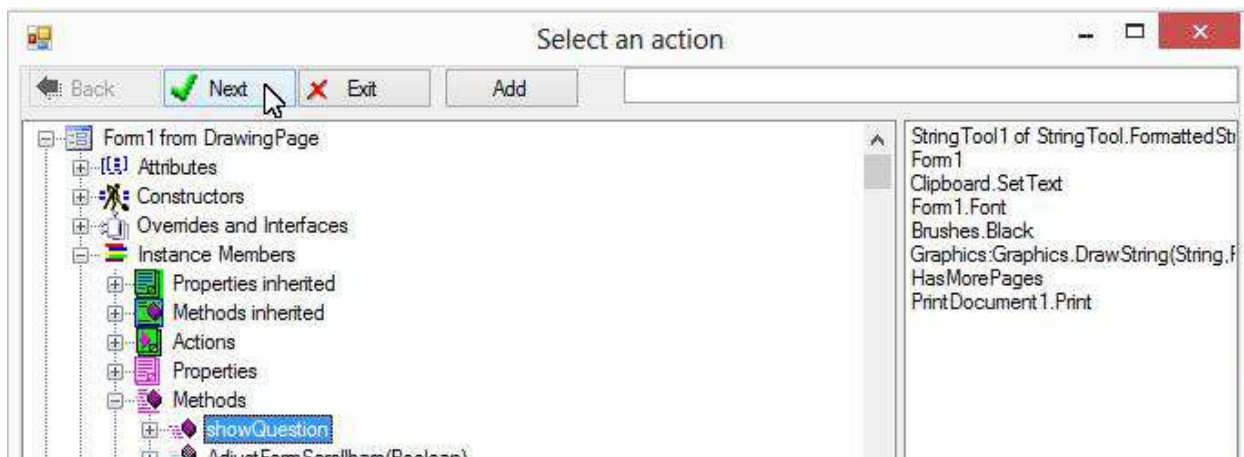


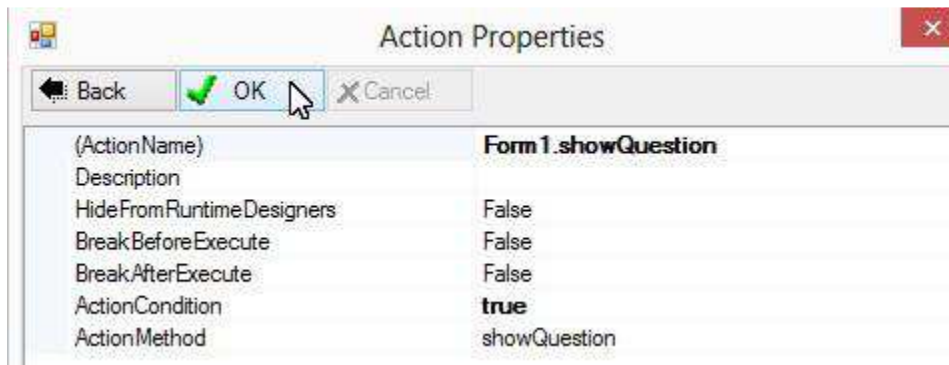
Execute the method

The method will be executed at the Load event of the form after loading the configurations, and on clicking the “Previous” and “Next” button. We’ll program for the buttons later. Here we program for form Load:

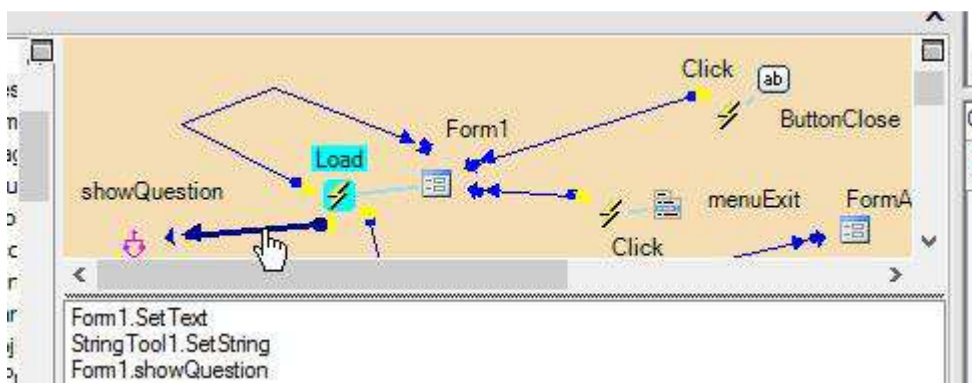


Select the method:





The action is assigned to Load event:

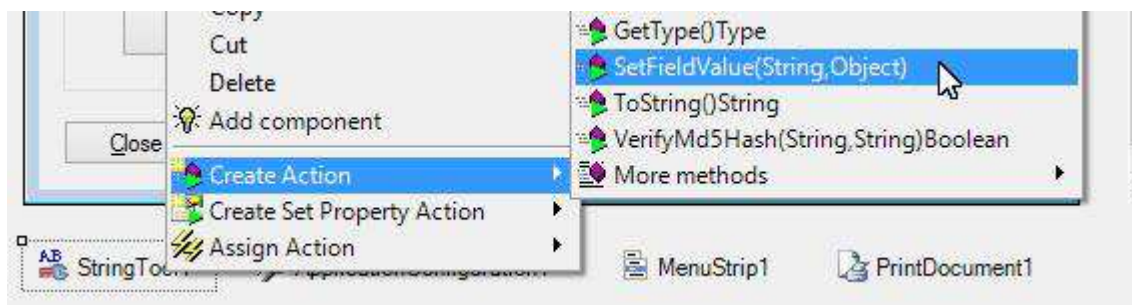


Wizard Navigation

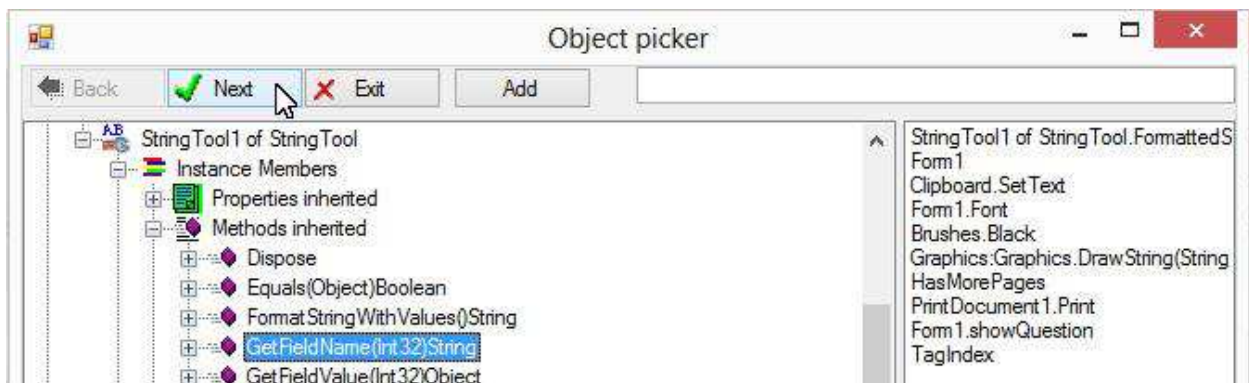
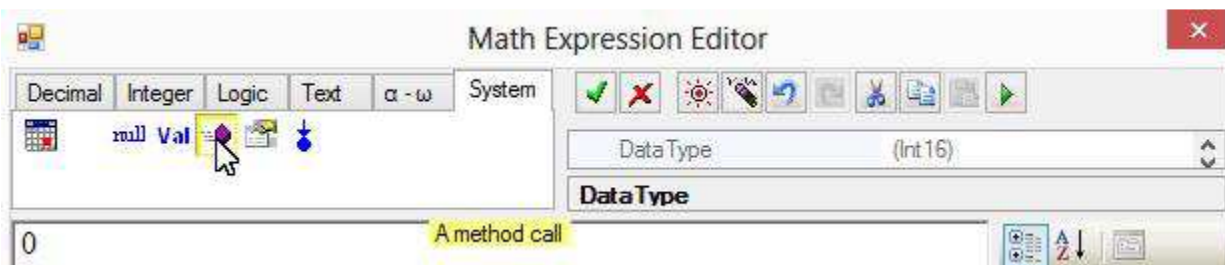
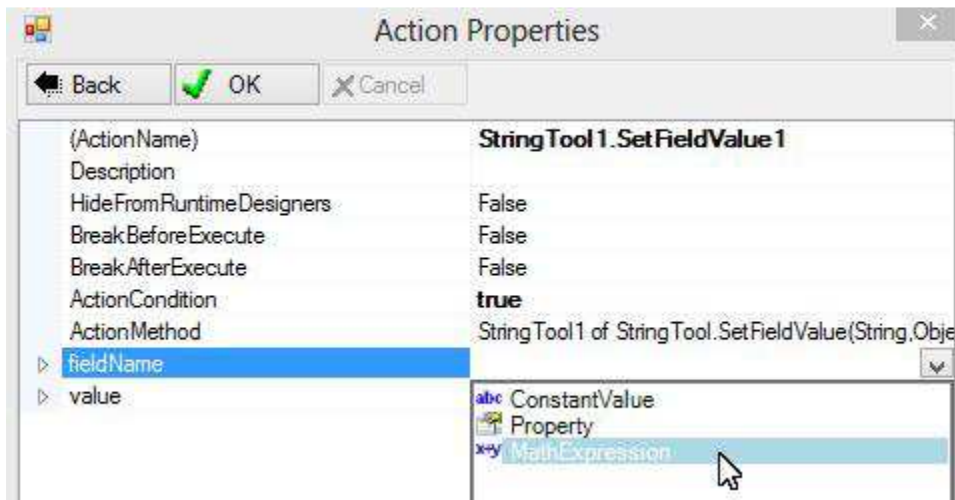
Let's do programming for buttons "Previous" and "Next".

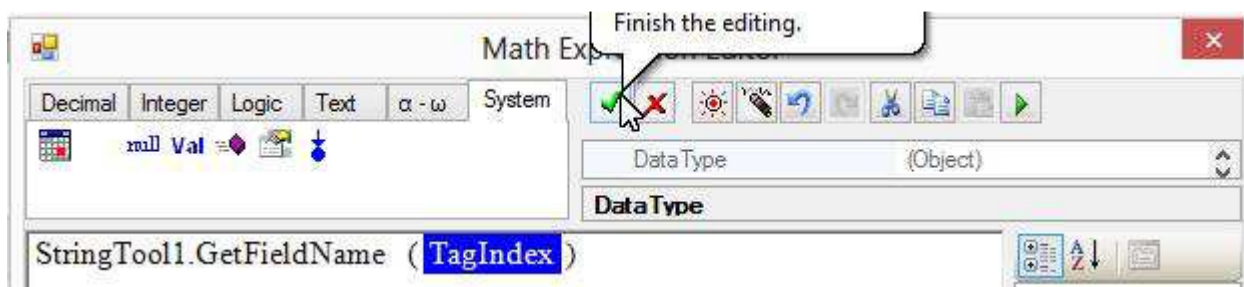
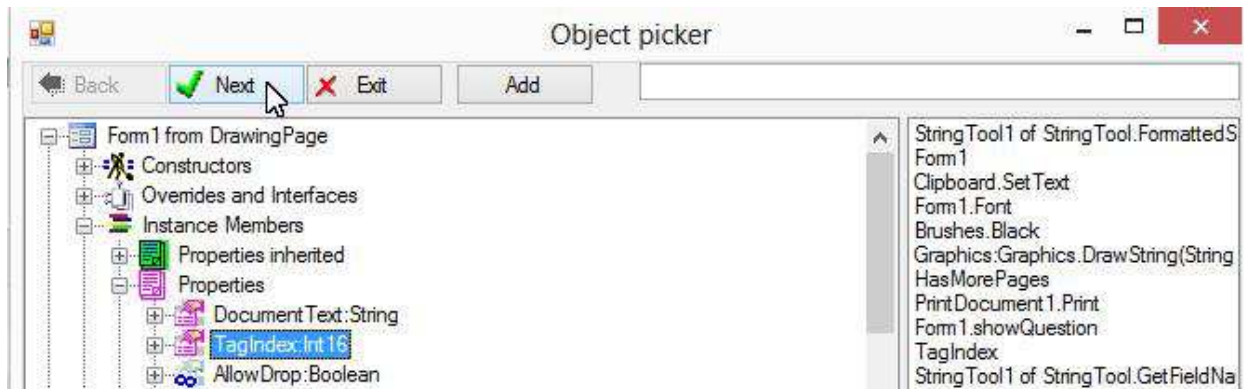
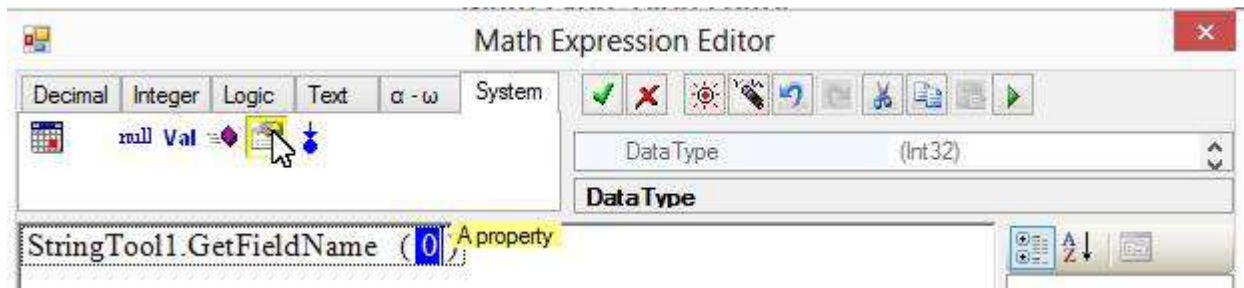
Save user input

Let's create an action to save user input. The action will be used by both buttons.

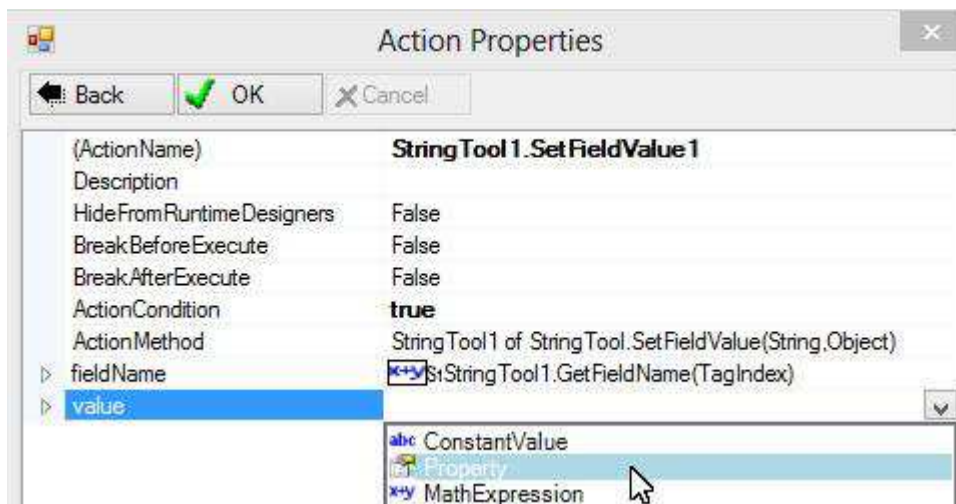


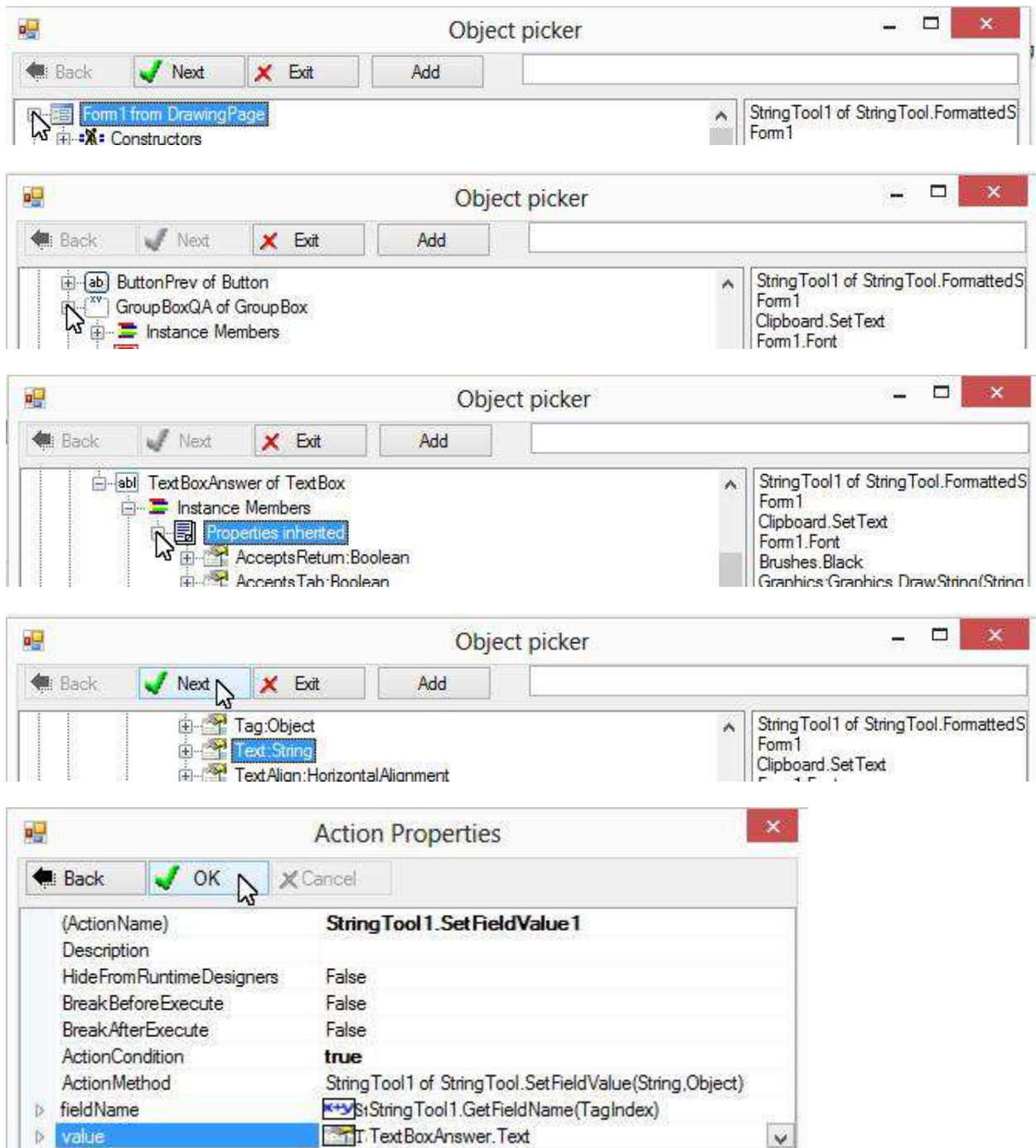
Use an expression to get the tag name via TagIndex:



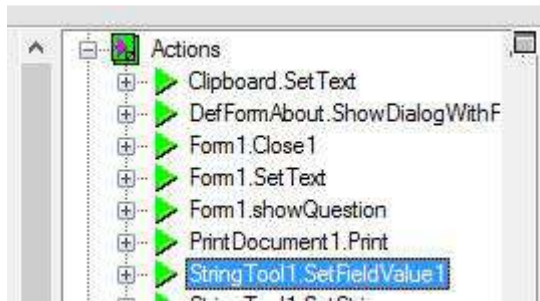


Select the Text property of the text box for the value:

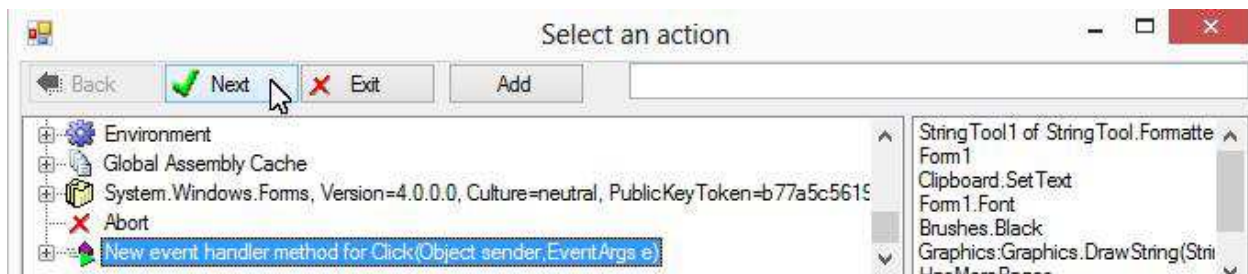
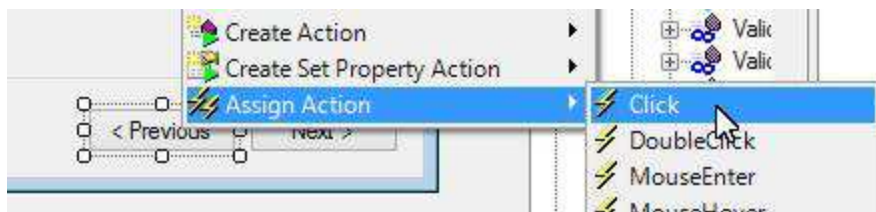




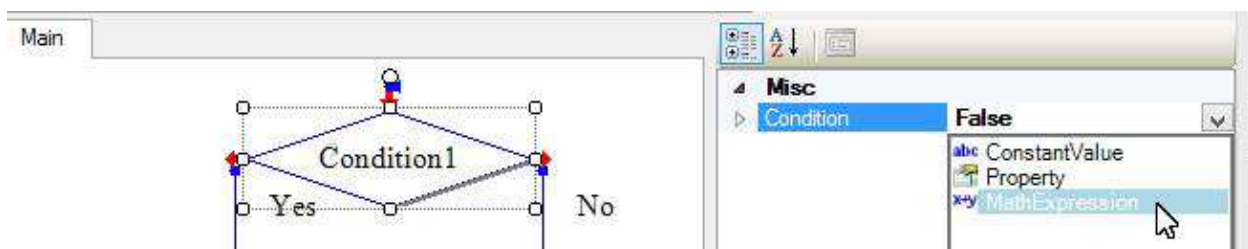
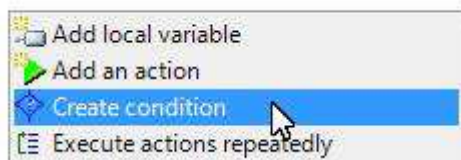
The action is created:

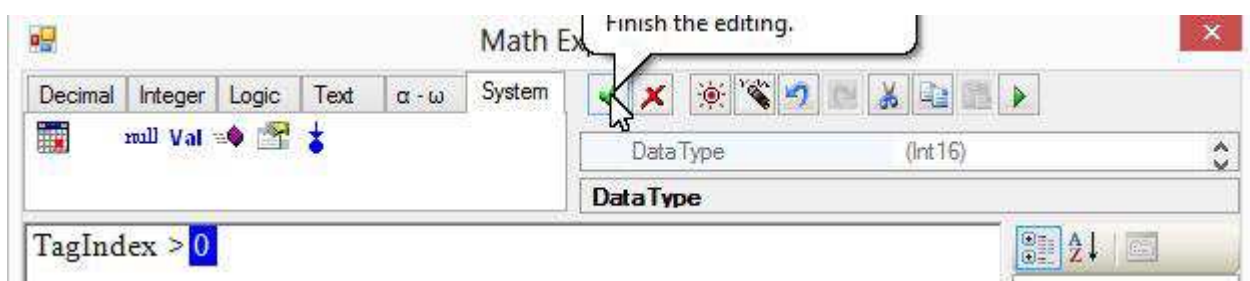
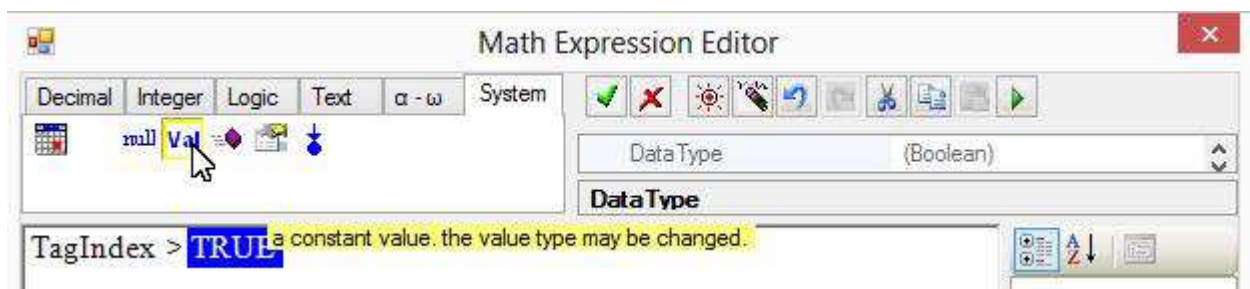
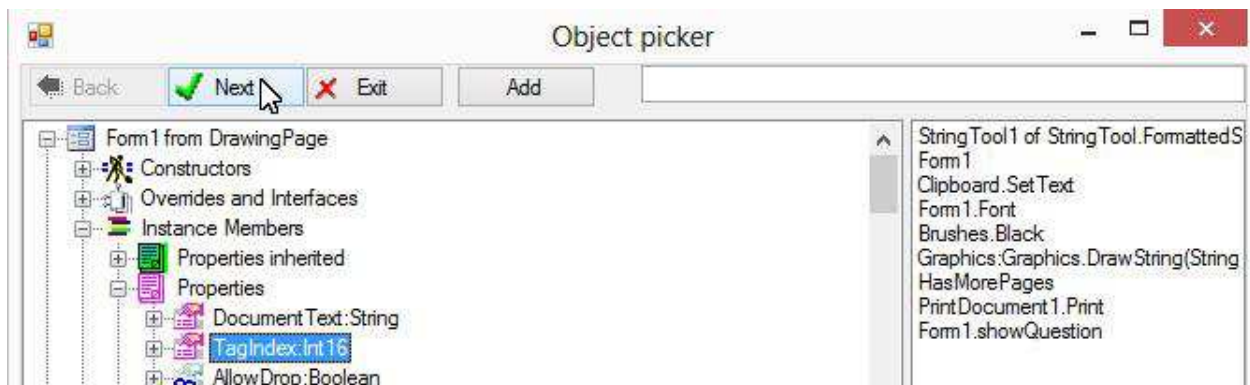
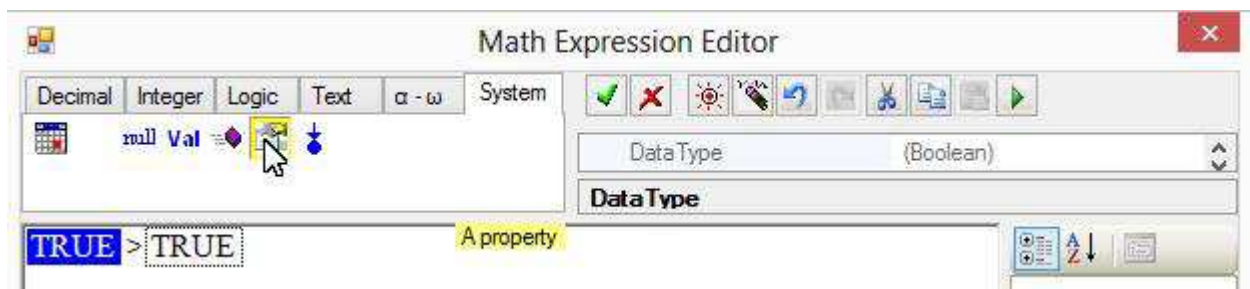
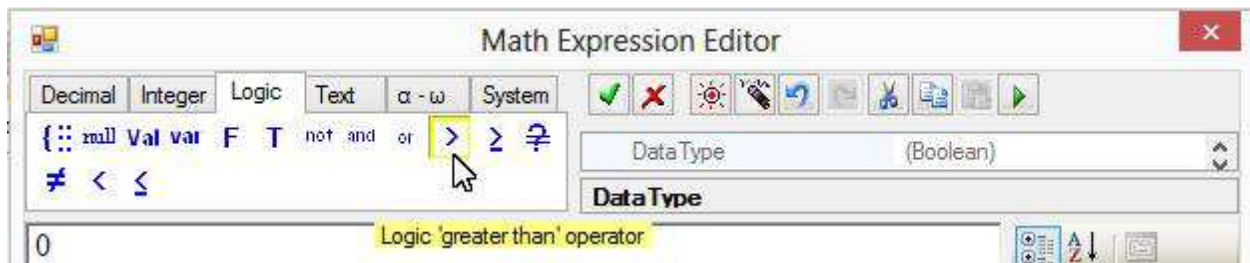


Previous Button

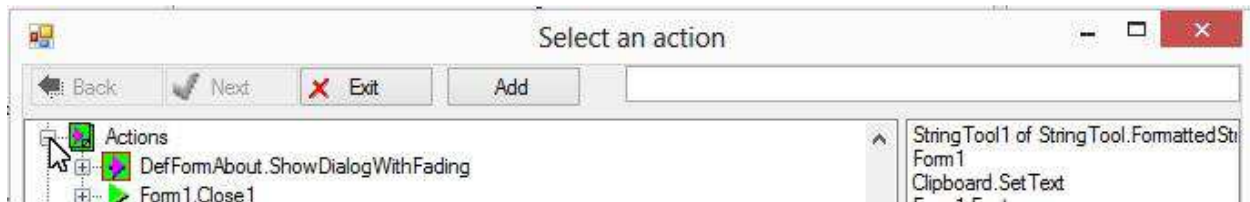
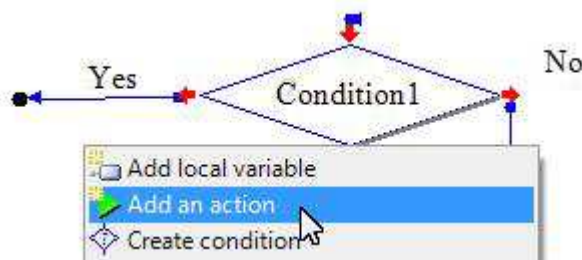


Add a Condition action to check TagIndex is greater than 0:

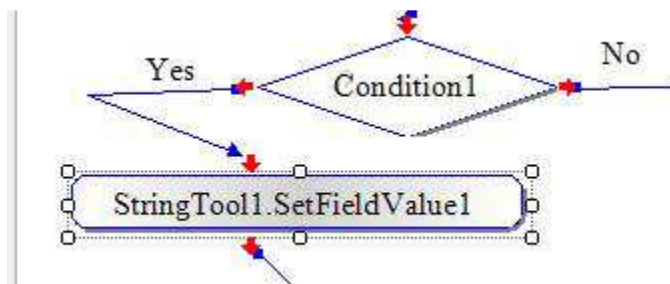




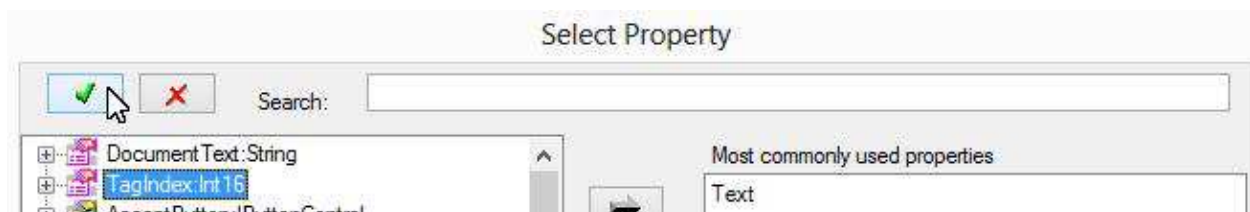
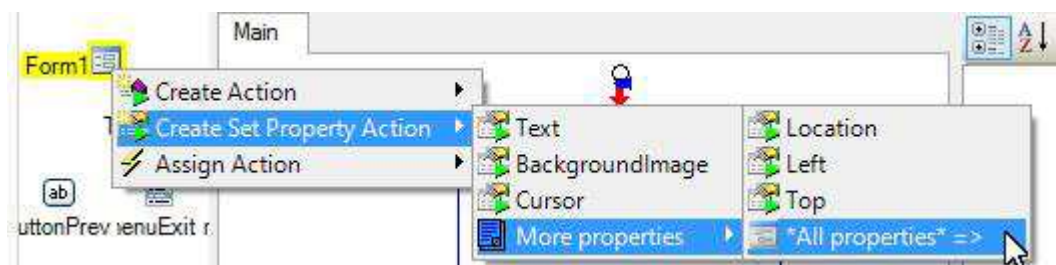
Save user input:

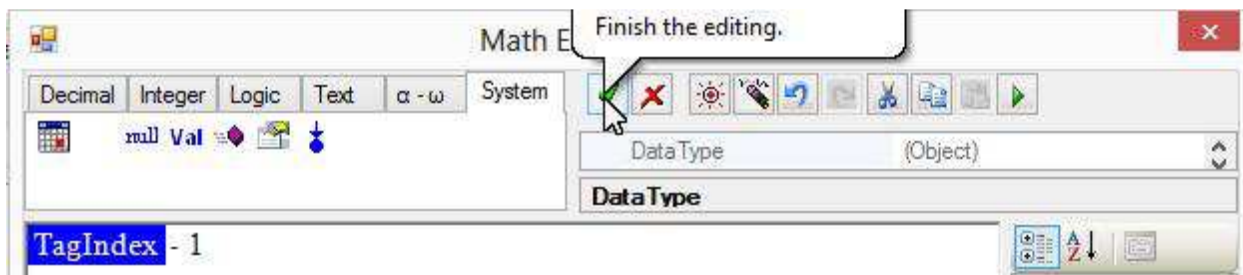
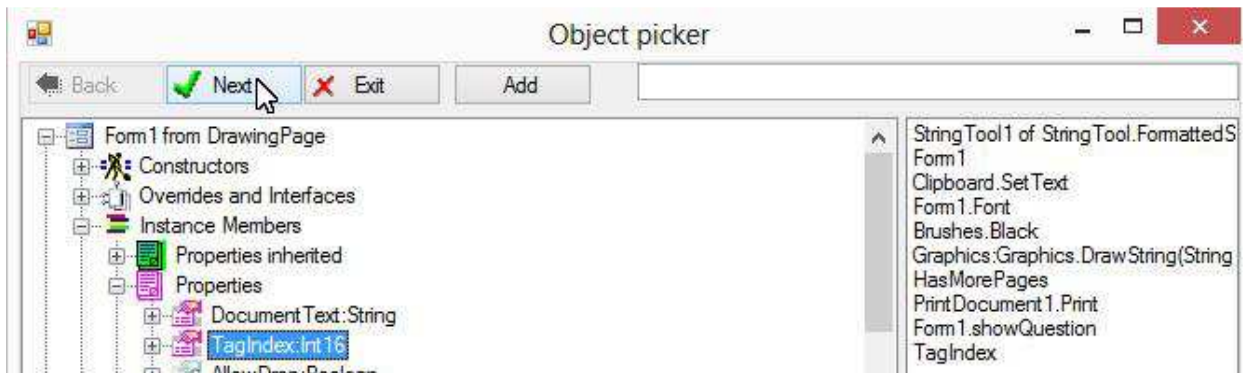
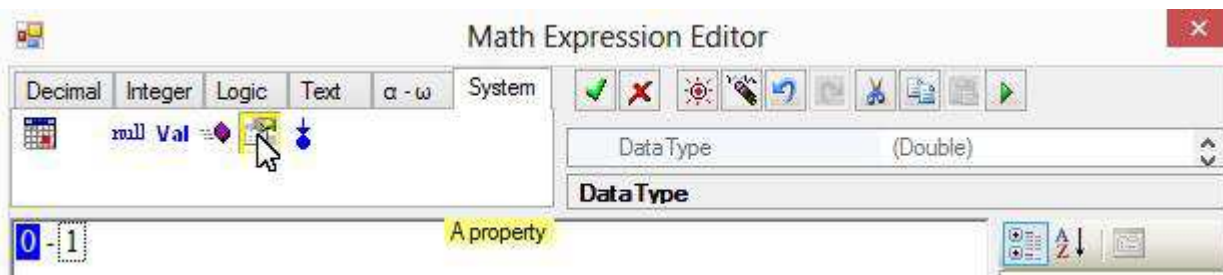
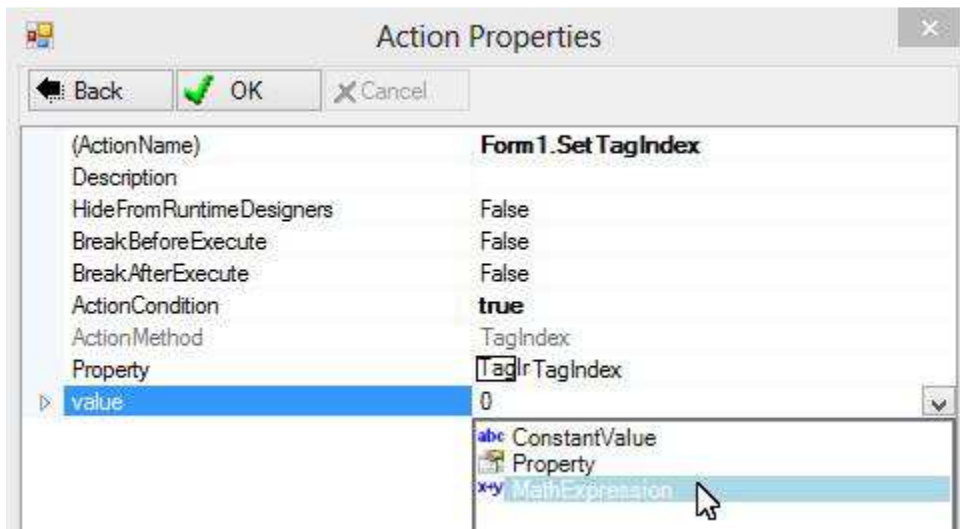


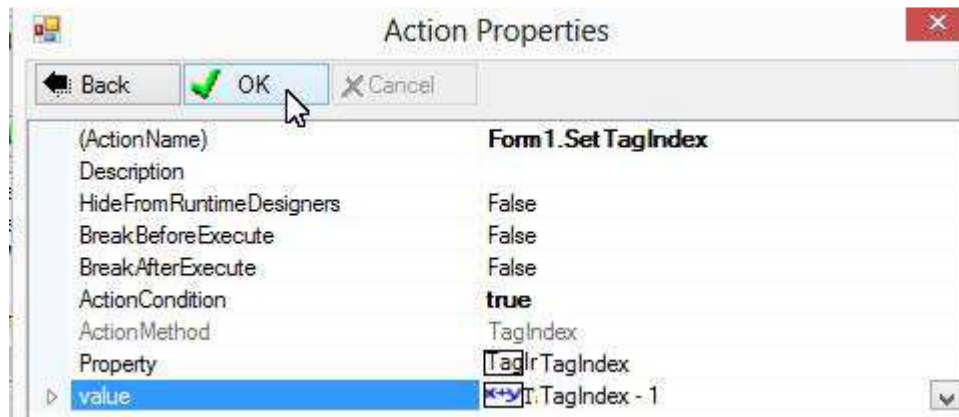
Link the action to the Yes port:



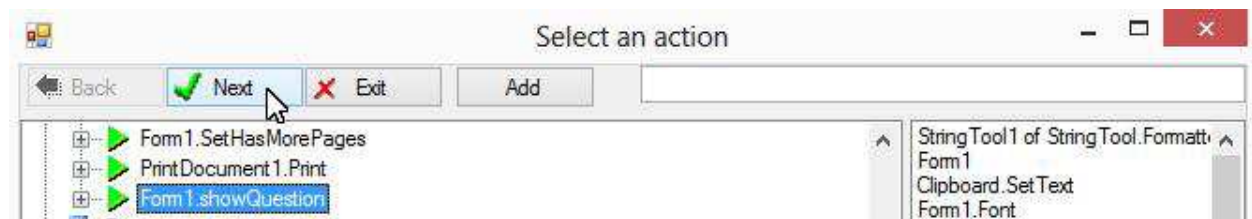
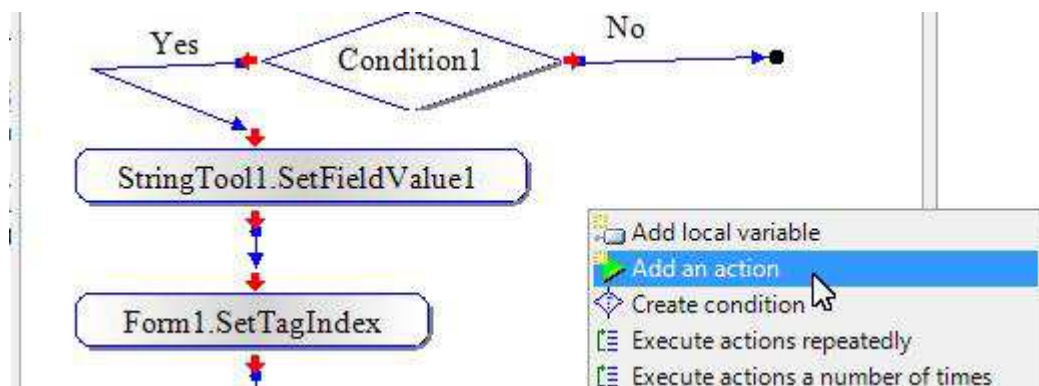
Decrease TagIndex:



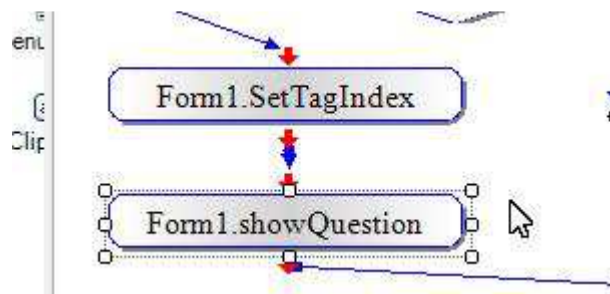




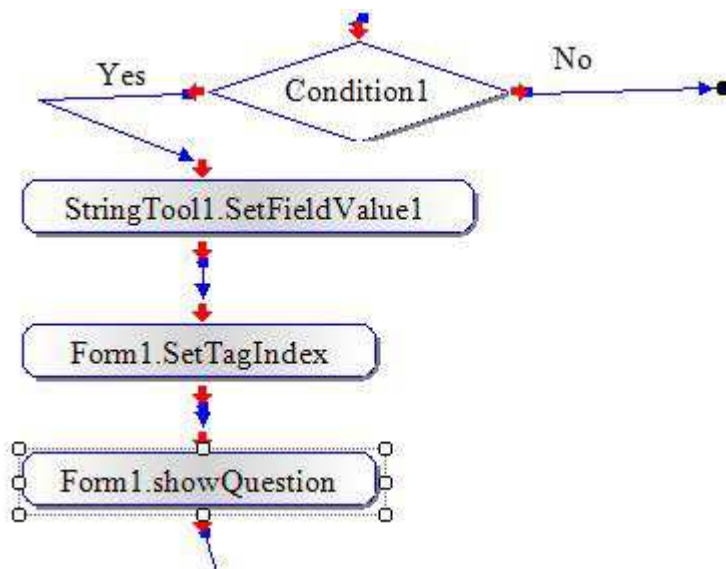
Link the action to the last action. Add the showQuestion action we created before:



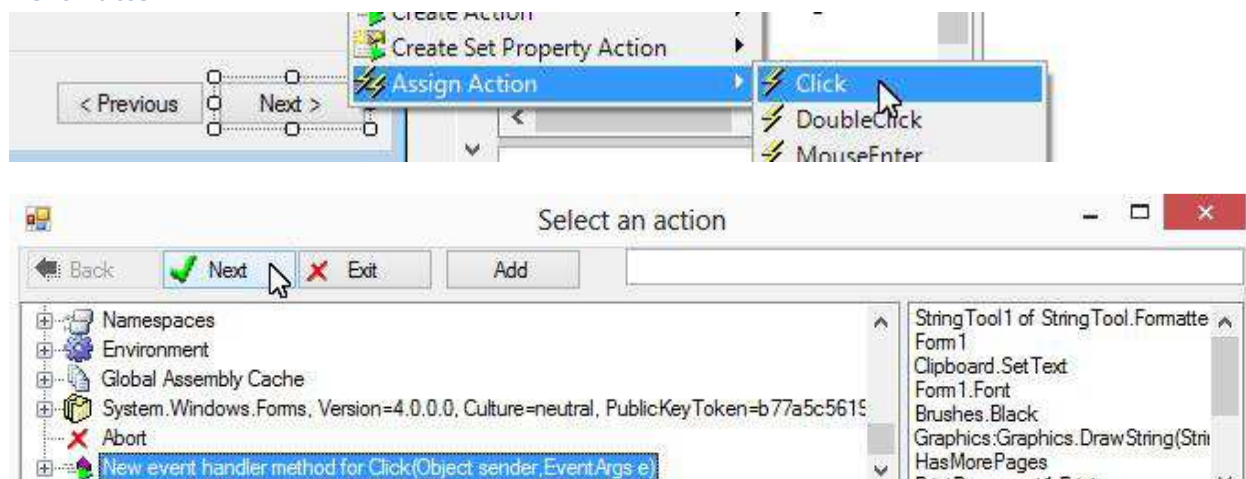
Link it to the last action:



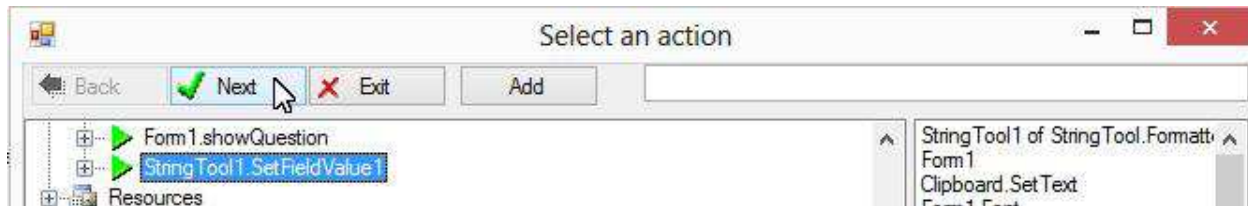
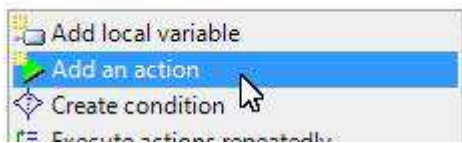
That is all for the “Previous” button:



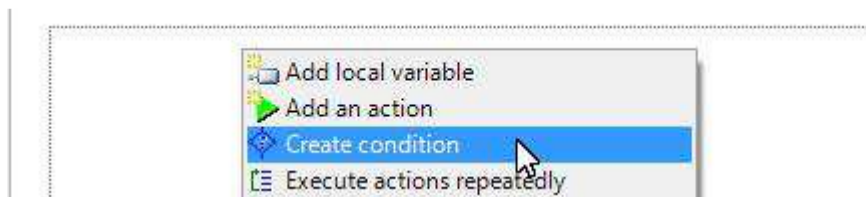
Next Button



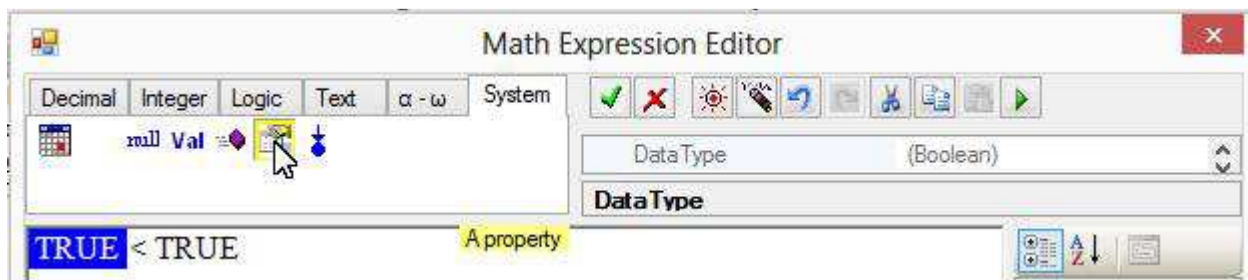
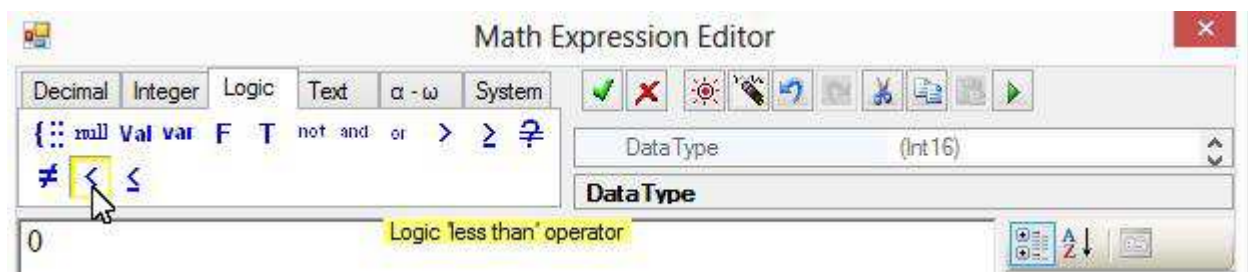
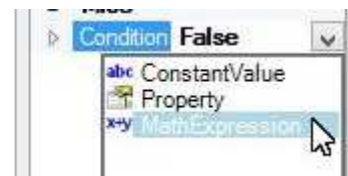
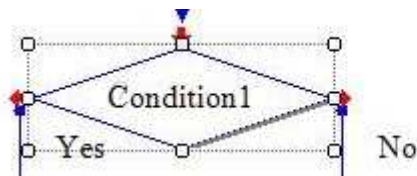
Save user input:

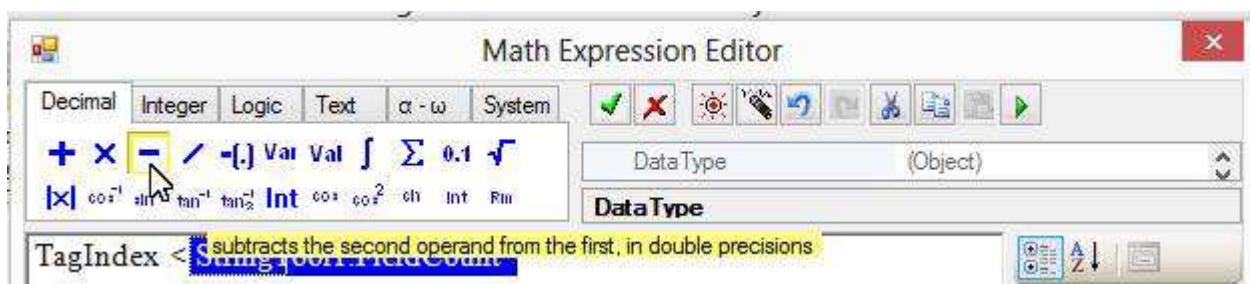
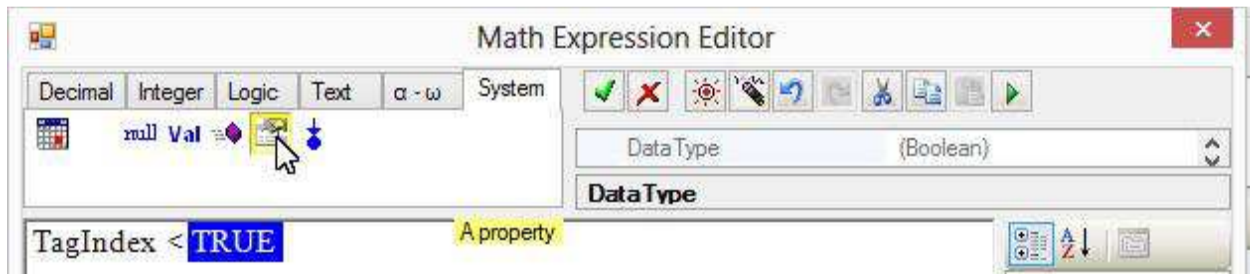
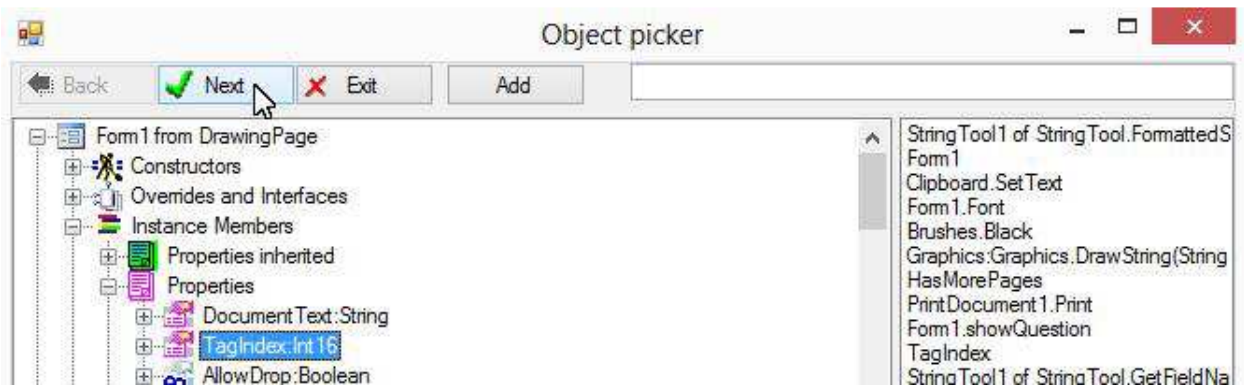


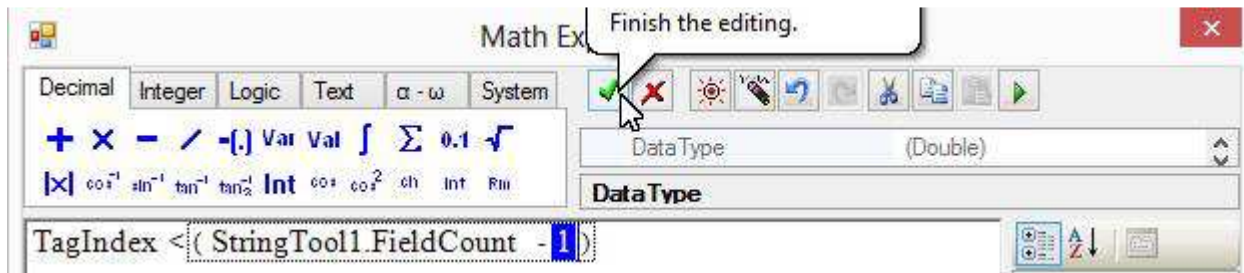
Add a Condition action to check that TagIndex is not at the end:



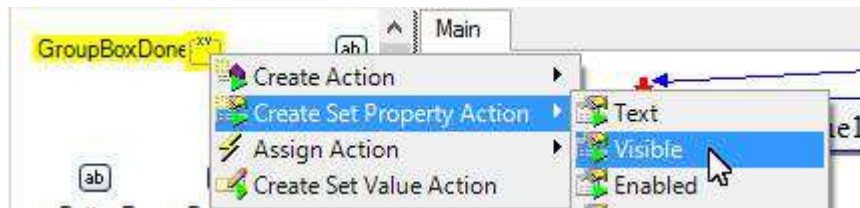
Link it to the last action and set condition:



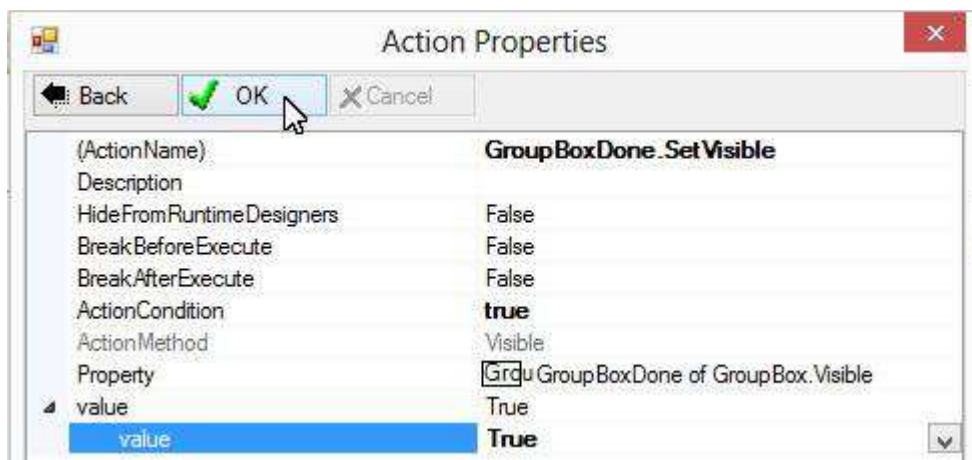




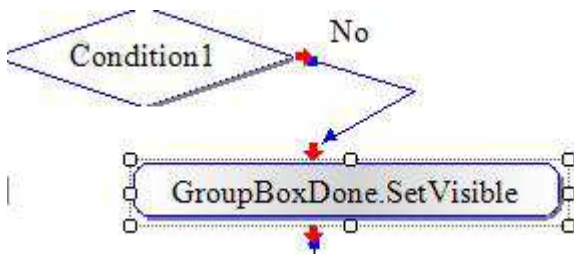
Show the Group Box for generating output:



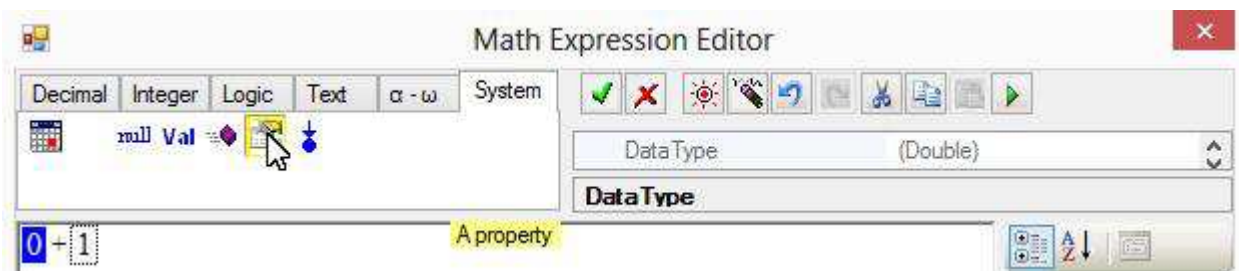
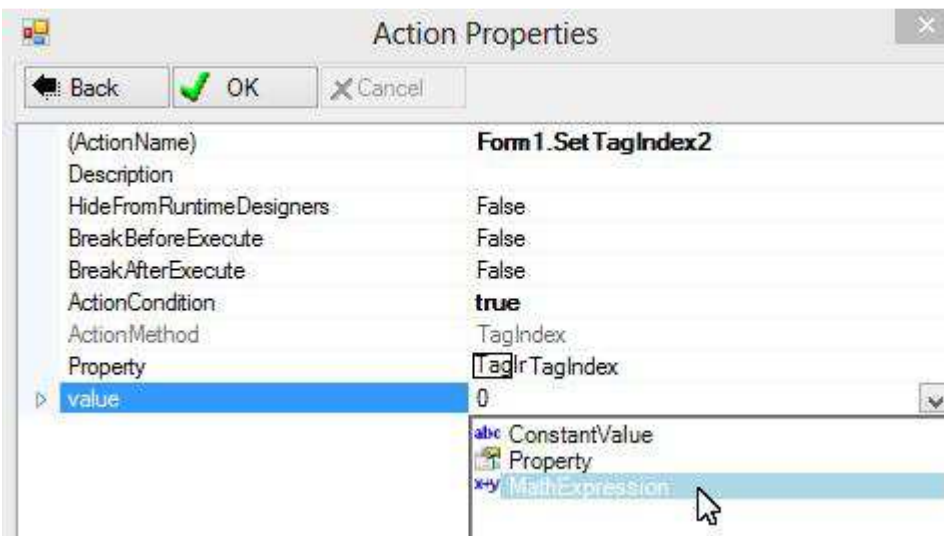
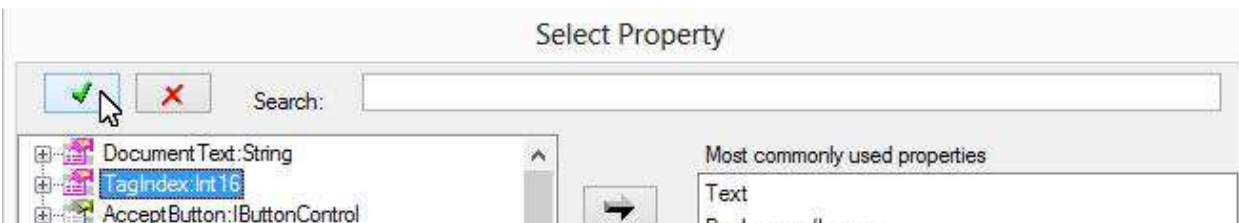
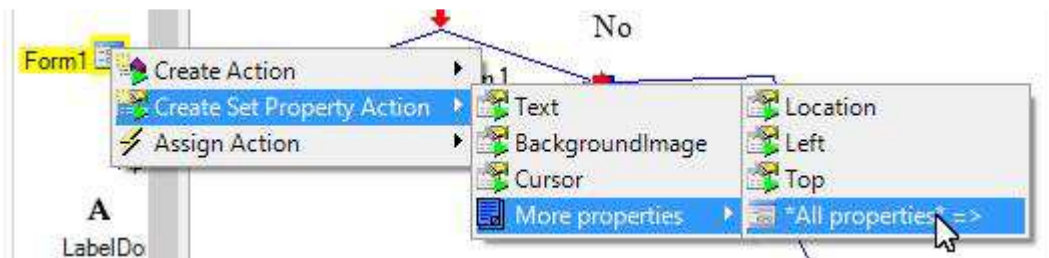
Set value to True to show the output group box:

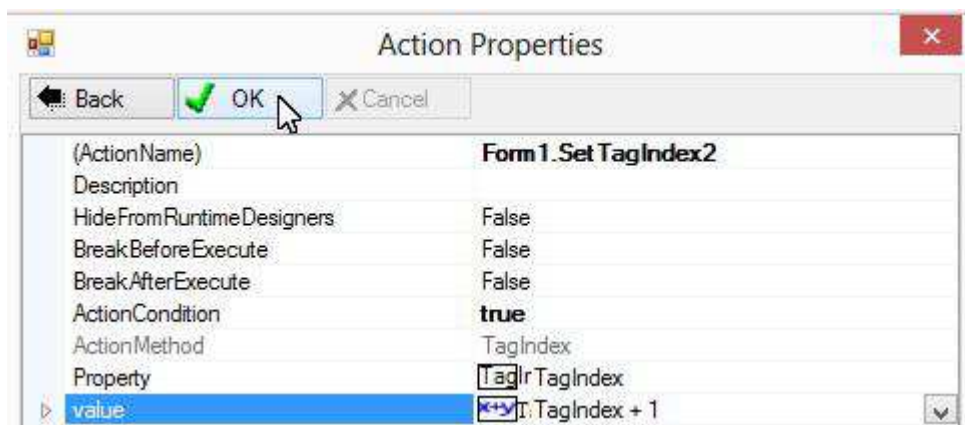
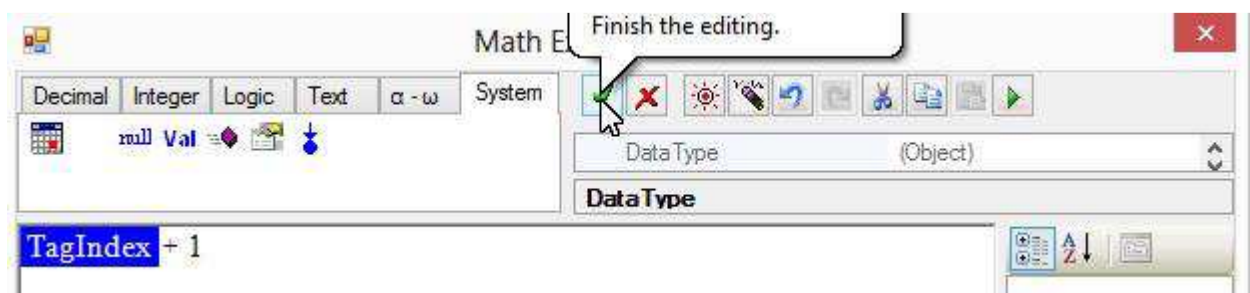
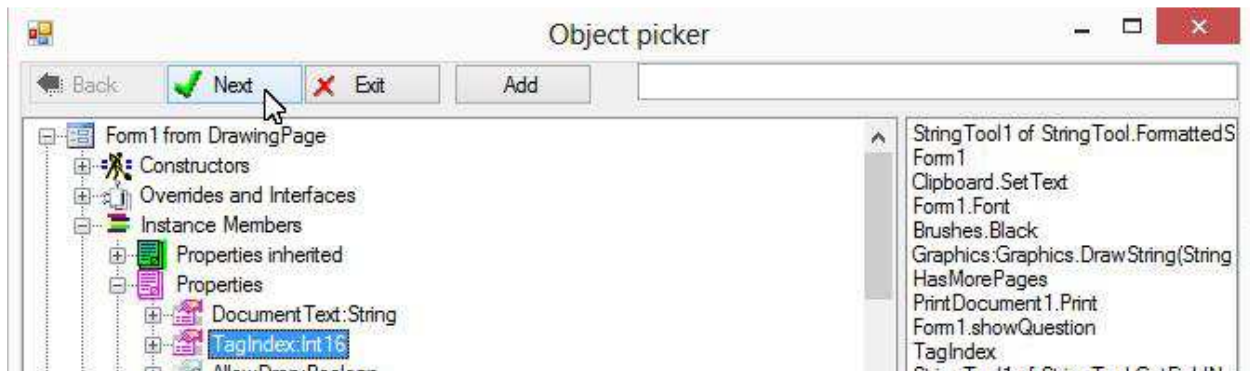


Link it to the No port of the Condition:



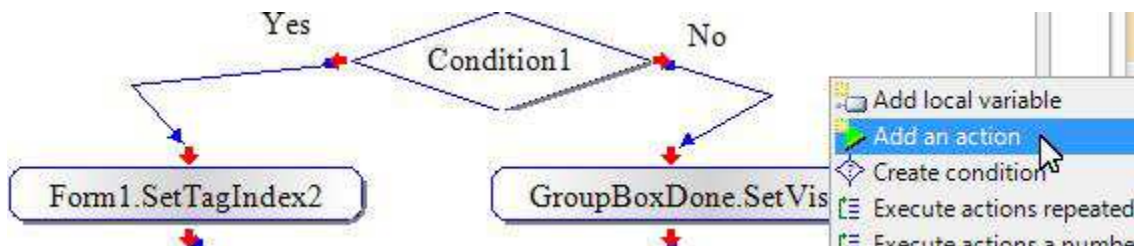
Increase TagIndex:





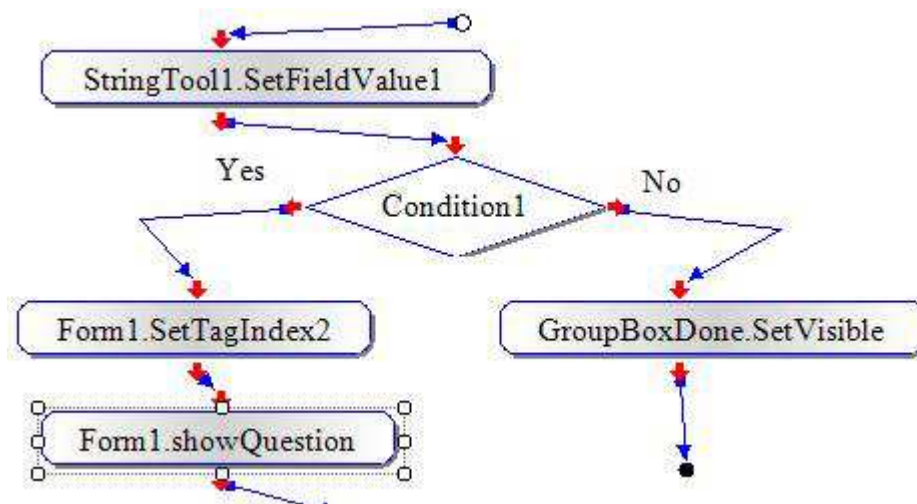
Link the action to Yes port of the Condition.

Show the question:





Link the action to the last action. That is all for the “Next” button:

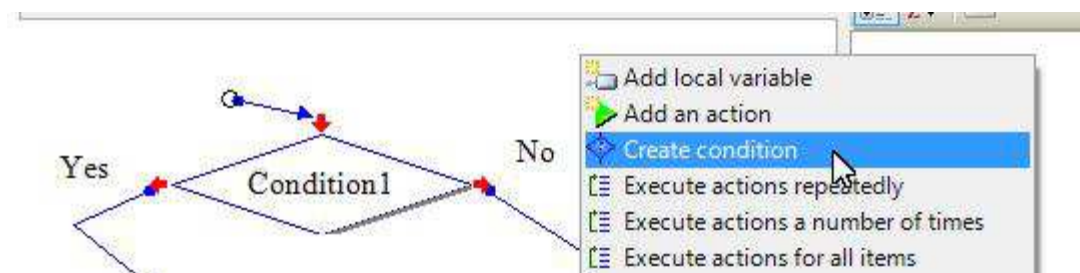


Modify “Previous” button handler

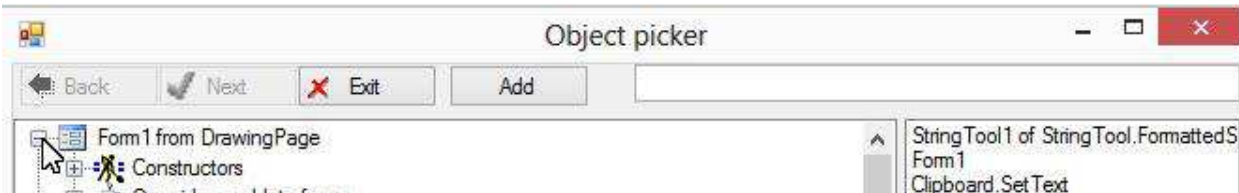
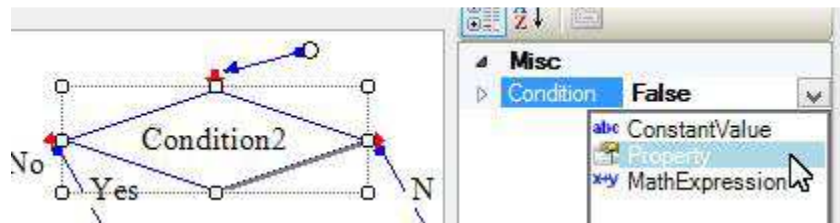
At the time we realize that the “Previous” button needs to hide the output group box if it is visible. Let’s modify the event handler for the “Previous” button:



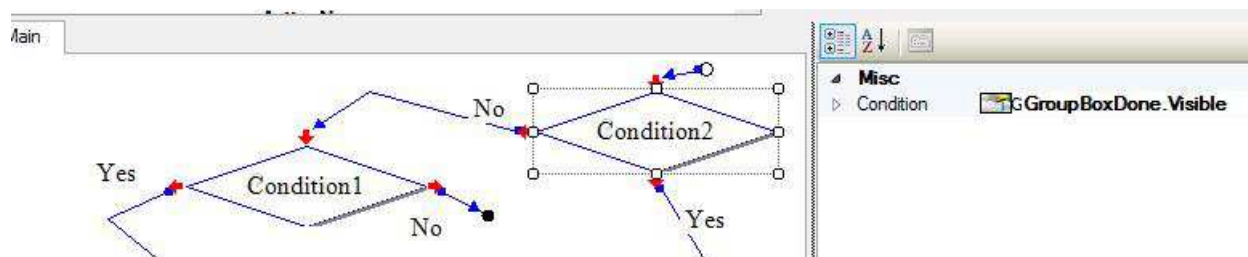
Add a Condition action to check if the output group box is visible:



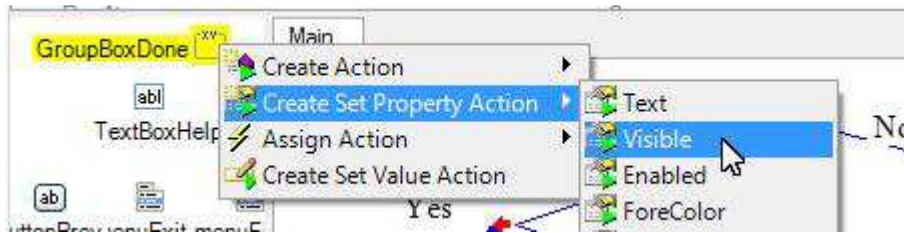
Set its condition to the Visible property of the output group box:



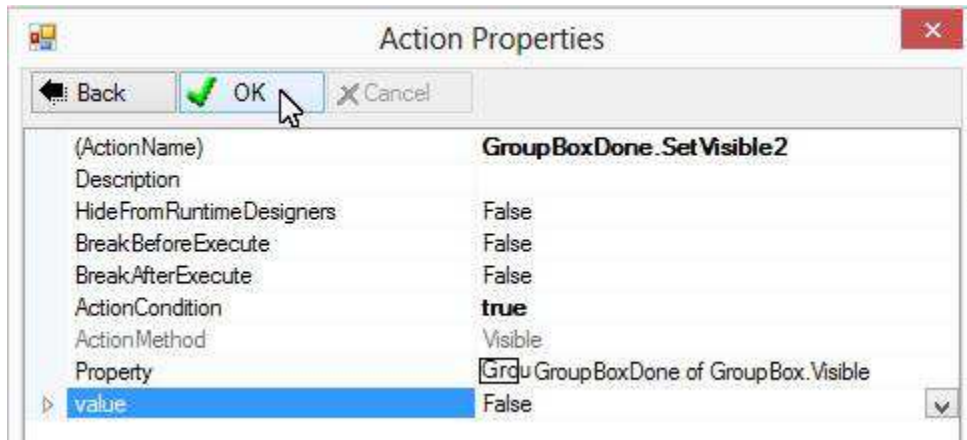
Link the No port to the first action of the original actions:



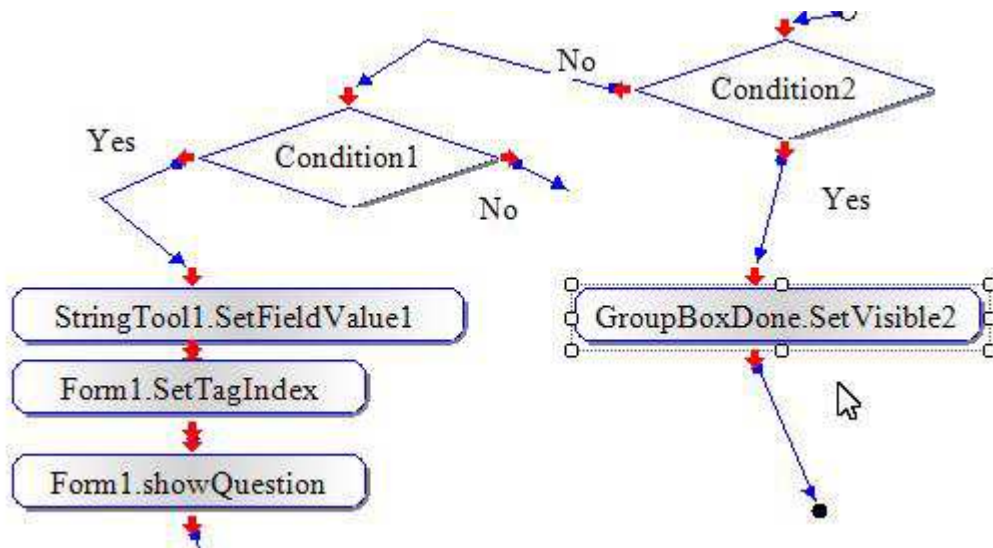
Add an action to hide the output group:



False for “value” hides the group box:



Link it to the Yes port of the condition. This is the new event handler for the “Previous” button:



About Form

The About Form shows software title and description.

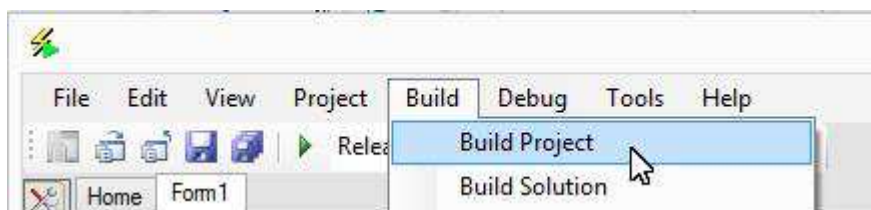


At form Load event we may use values from the application configuration to populate the controls on the form. We left it to you.

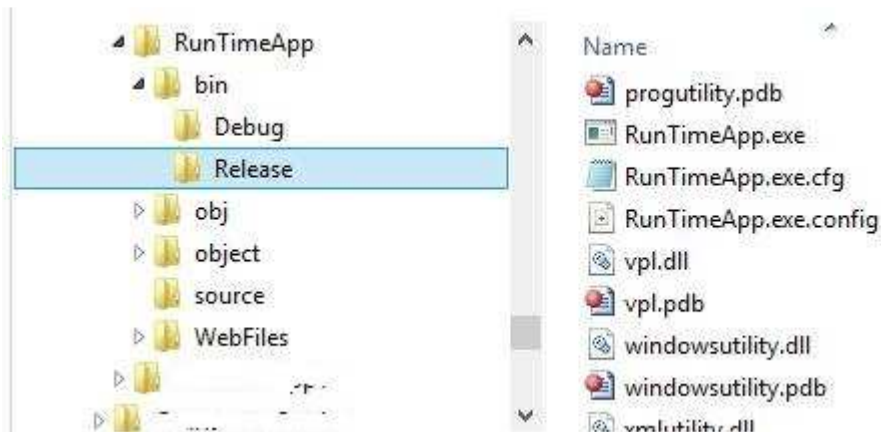
Create Design Time Application

You may download the project from <http://www.limnor.com/studio/DesignTimeApp.zip>.

In the last chapter we have created a program with desired functionality and weaved customization capability into it. We may compile the project:



The result files are in the Release folder:



To distribute this application to your users is to distribute all the files in this folder to your users.

The file RunTimeApp.exe is the EXE file to run. Note that the file RunTimeApp.exe.cfg is the application configuration file. All customizations of this application are recorded in this file. How to generate this file? Actually there is a document and a sample application to do that. See <http://www.limnor.com/support/CreateAppConfigUtility.pdf>. You may download the project from <http://www.limnor.com/studio/configUtil.zip>. This is a generic configuration utility for all applications having customizations weaved.

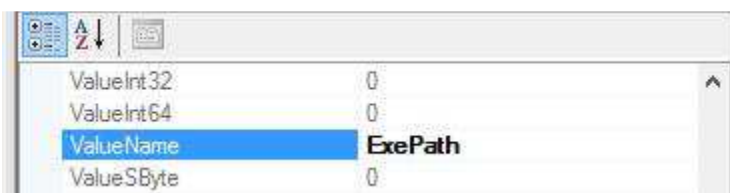
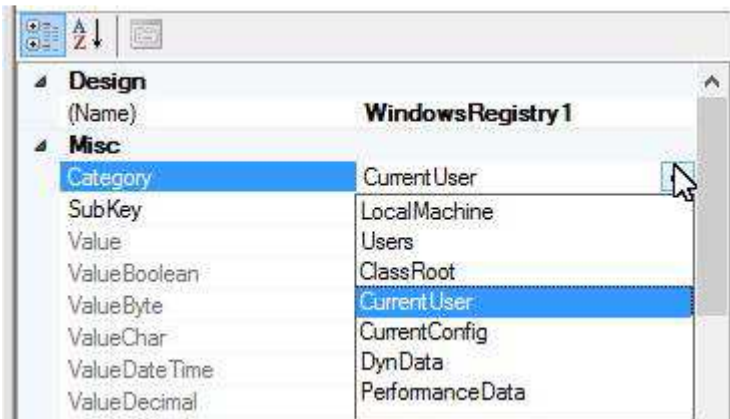
We may create a configuration utility specifically for application RunTimeApp.exe, designing UI specifically for configurations of RunTimeApp.exe.

Remember EXE file path

A configuration utility needs to know which EXE file to be configured. We may use a dialogue box to let the user to tell the program where is RunTimeApp.Exe. When we get the information, we want to save it so that the next time we do not have to ask the user again. This is a typical application configuration issue. Instead of using Application Configuration component as we did in previous chapter, we use WindowsRegistry component to remember the path to RunTimeApp.Exe.



We save the Exe path to CurrentUser, under sub-key Software\DesignTimeApp, data name is ExePath:



Writing to Value property will save the value to the registry; reading ValueString gets the value from the registry as a string:



See the event handler for the Load event of the form to see the use of the WindowsRegistry.

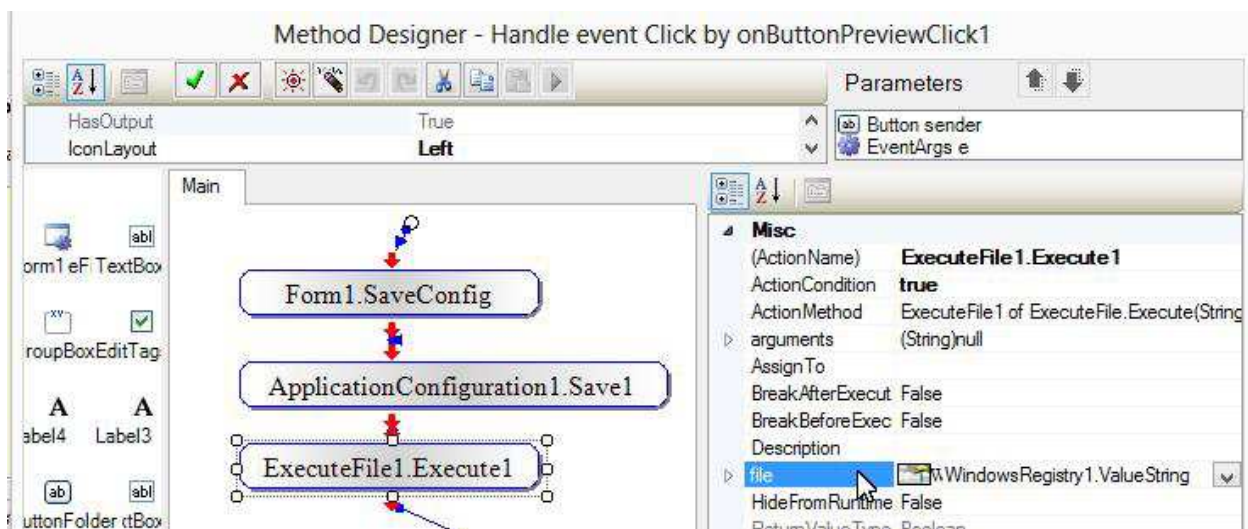
Note the actions using method SetApplicationFullPath. We use the method to set the full path to RunTimeApp.Exe. Once such an action is executed, the ApplicationConfiguration component will read/write configurations to RunTimeApp.Exe.

Program preview

The feature of "Program Preview" is to run RunTimeApp.Exe. We use an ExecuteFile component to do it:

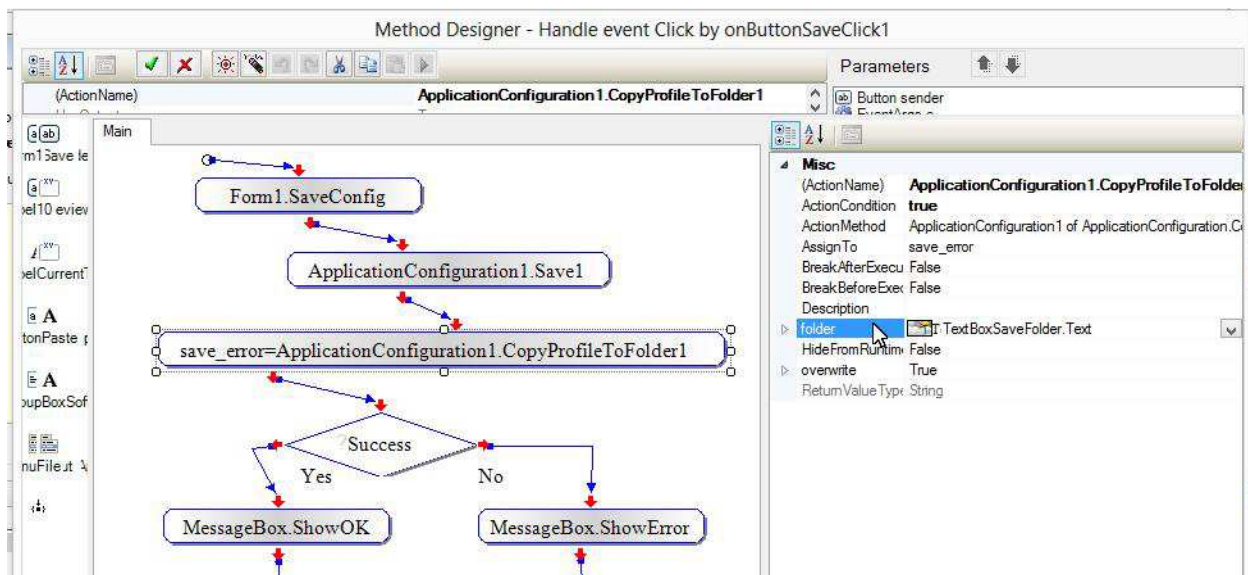


In the “preview” button event handler, we use ExecuteFile component to execute an “Execute” action. The “file” parameter of the “Execute” action is the ValueString property of the WindowsRegistry. The ValueString is the full path to RunTimeApp.Exe.



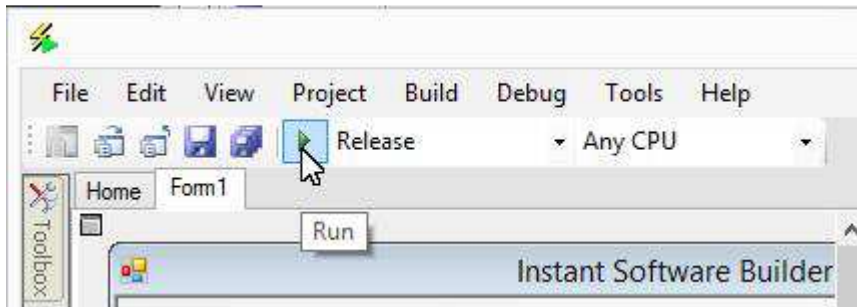
Save program

A CopyProfileToFolder action is used to copy the configuration file to the folder specified by a text box:

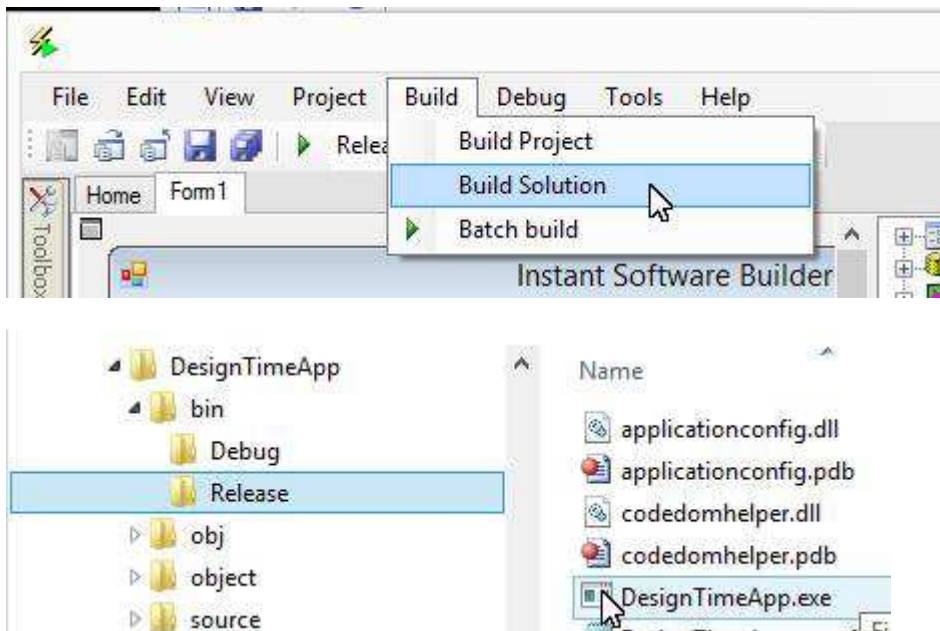


Test

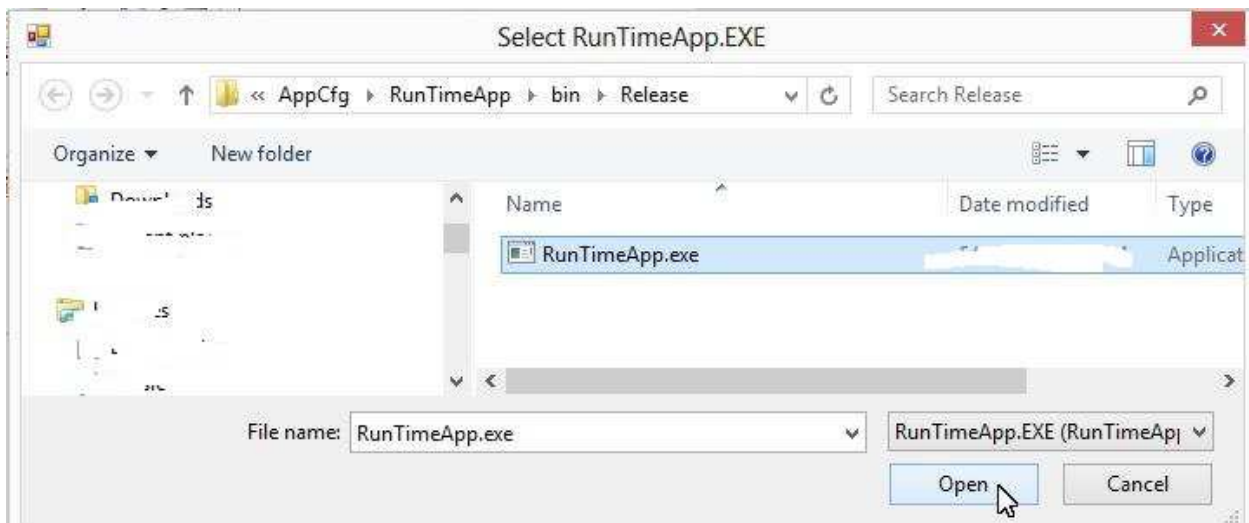
Click the Run button to compile and run DesignTimeApp.exe:



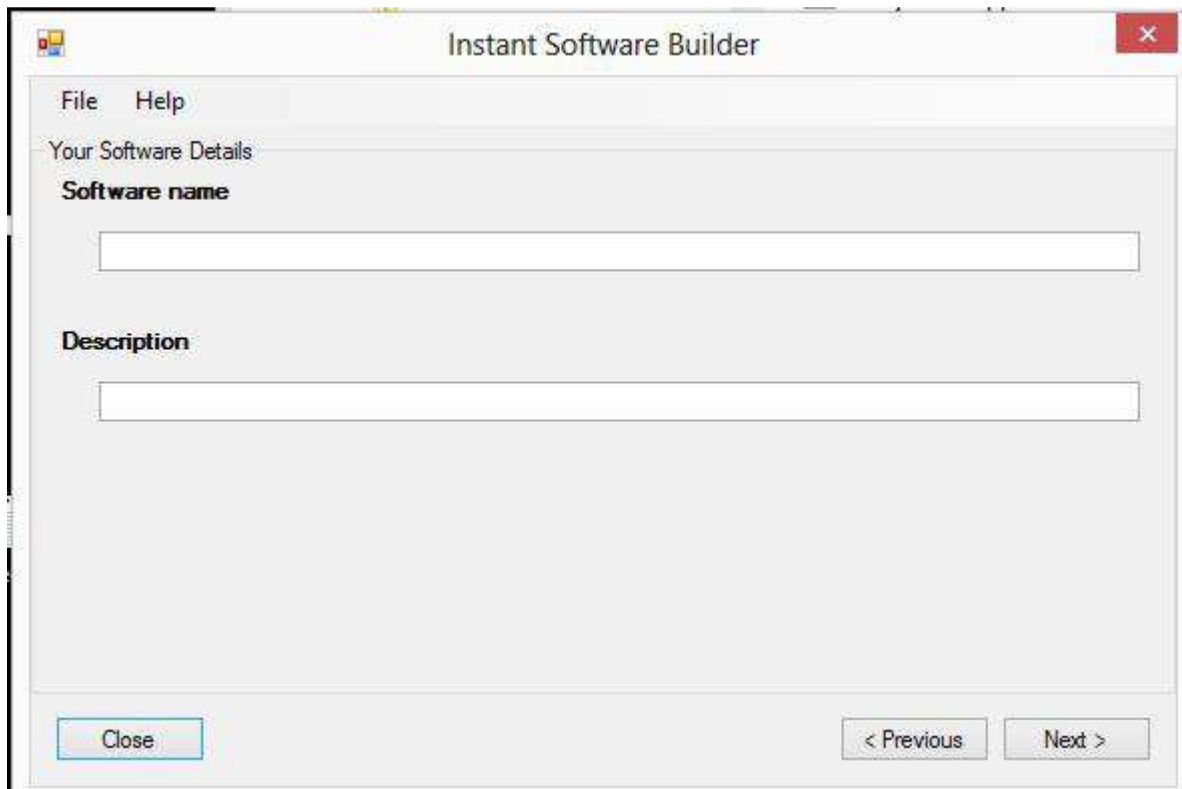
If you do not want to run it from Limnor Studio then you may compile the project and double-click DesignTimeApp.Exe:



For the very first time it runs, a dialogue box appears asking for RunTimeApp.Exe:

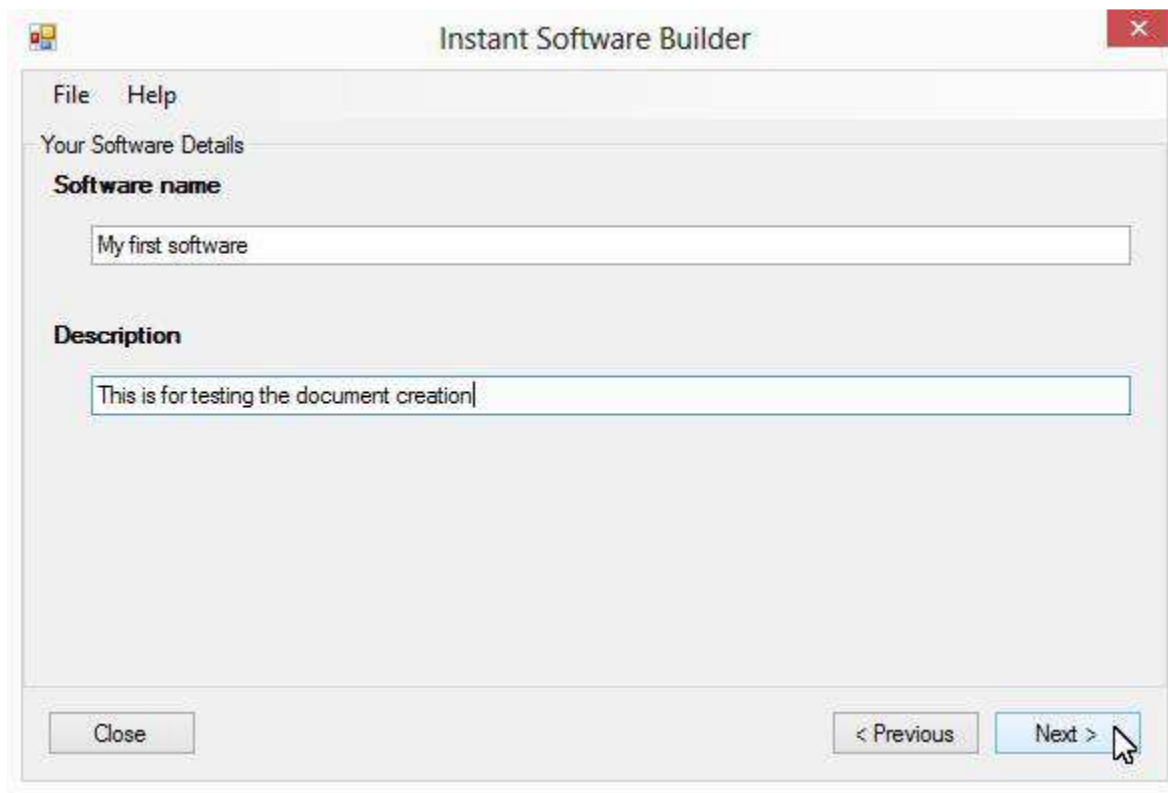


On specifying the location of RunTimeApp.Exe, the first step appears:



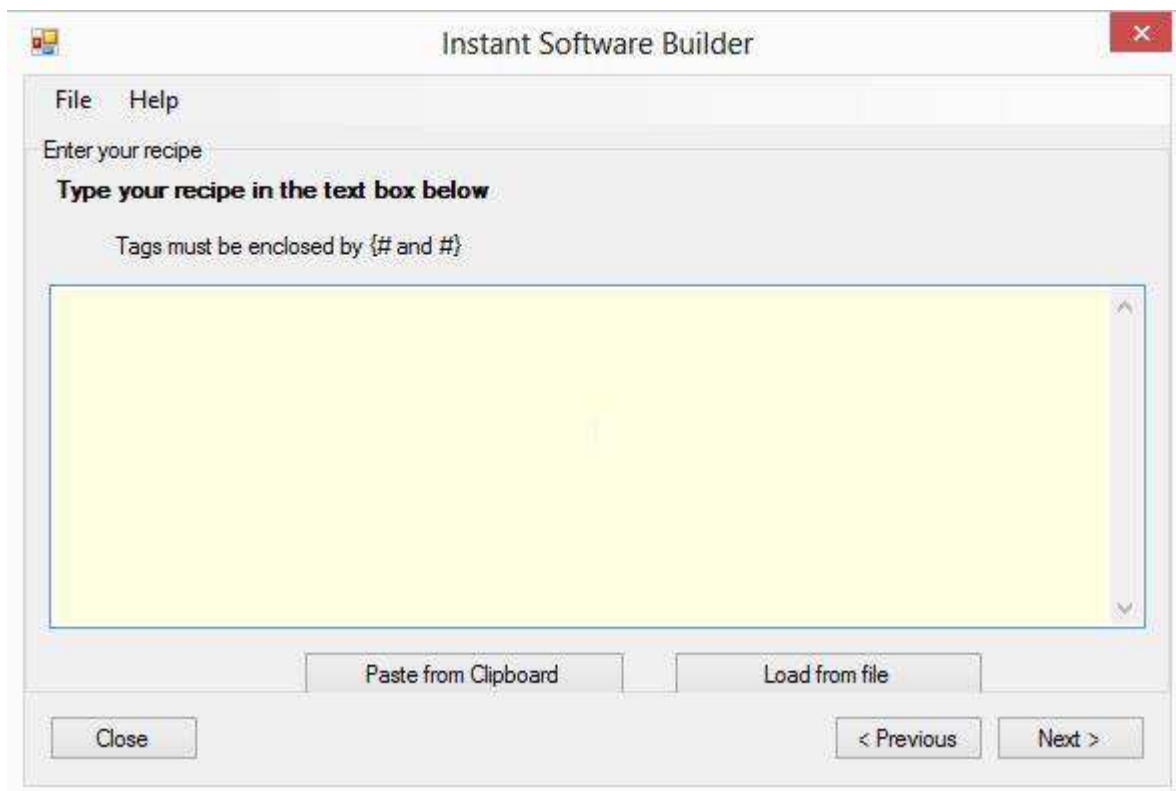
The screenshot shows a window titled "Instant Software Builder". It has a menu bar with "File" and "Help". Below the menu bar is a section titled "Your Software Details". This section contains two labels: "Software name" and "Description", each followed by a text input field. At the bottom of the window, there are three buttons: "Close" on the left, and "< Previous" and "Next >" on the right.

Enter some data and click Next



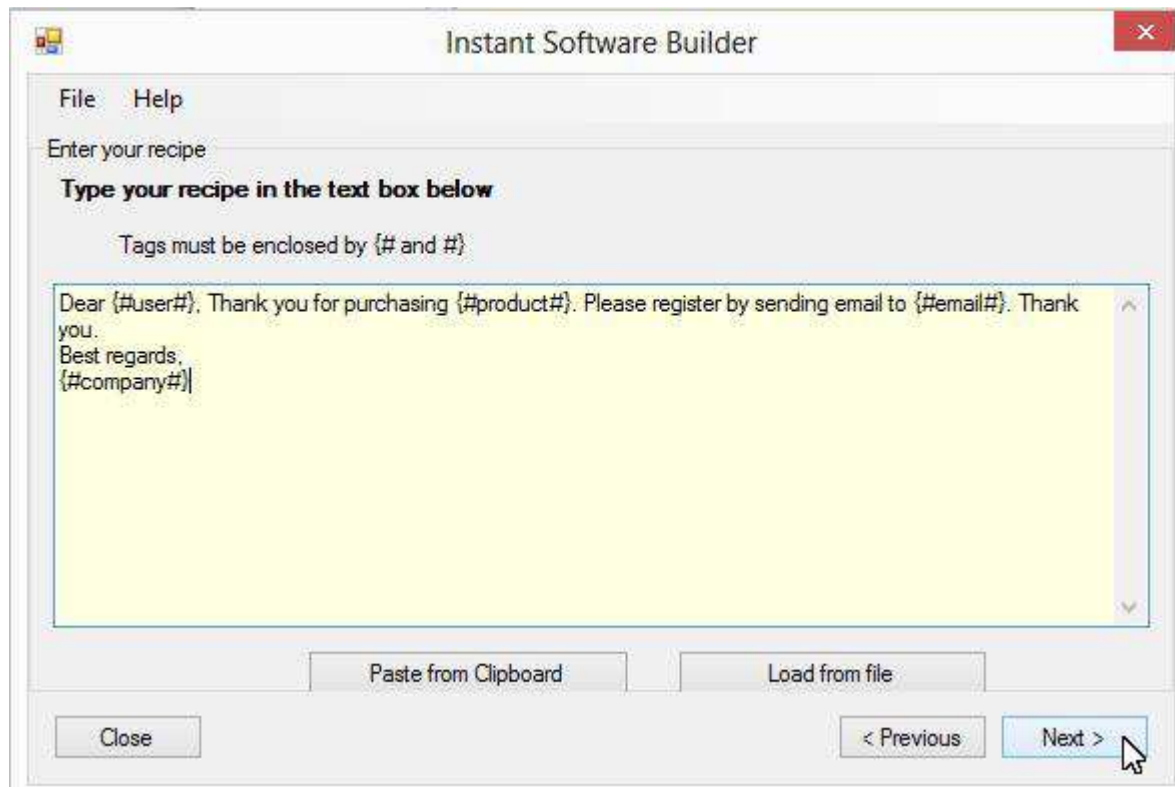
The screenshot shows the 'Instant Software Builder' application window. The title bar includes a standard Windows icon, the text 'Instant Software Builder', and a red close button. The menu bar contains 'File' and 'Help'. The main content area is titled 'Your Software Details' and contains two sections: 'Software name' with a text input field containing 'My first software', and 'Description' with a text input field containing 'This is for testing the document creation'. At the bottom, there are three buttons: 'Close' on the left, and '< Previous' and 'Next >' on the right. A mouse cursor is pointing at the 'Next >' button.

The next step appears:



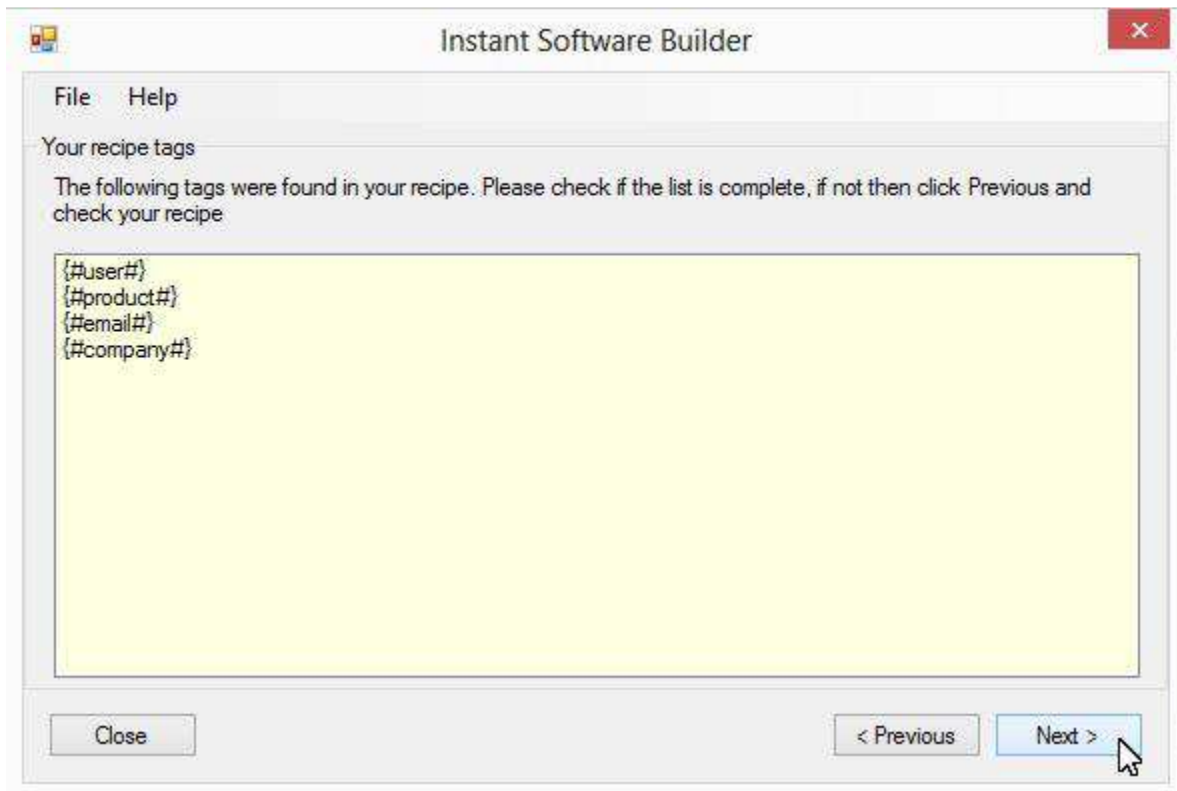
The screenshot shows the 'Instant Software Builder' application window at the next step. The title bar and menu bar are the same. The main content area is titled 'Enter your recipe' and contains the instruction 'Type your recipe in the text box below'. Below this, a note states 'Tags must be enclosed by {# and #}'. A large, empty text area with a yellow background and a vertical scrollbar is provided for input. At the bottom, there are four buttons: 'Close' on the left, 'Paste from Clipboard' and 'Load from file' in the center, and '< Previous' and 'Next >' on the right.

Enter document template:

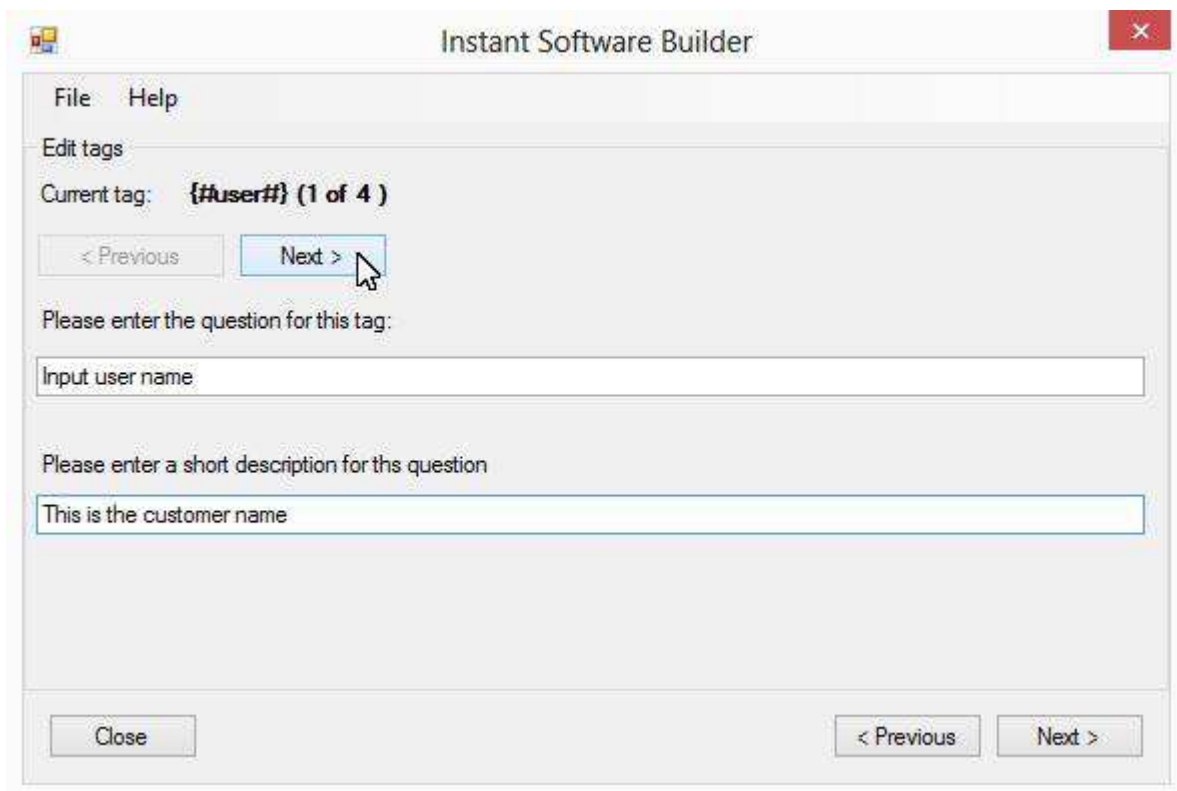


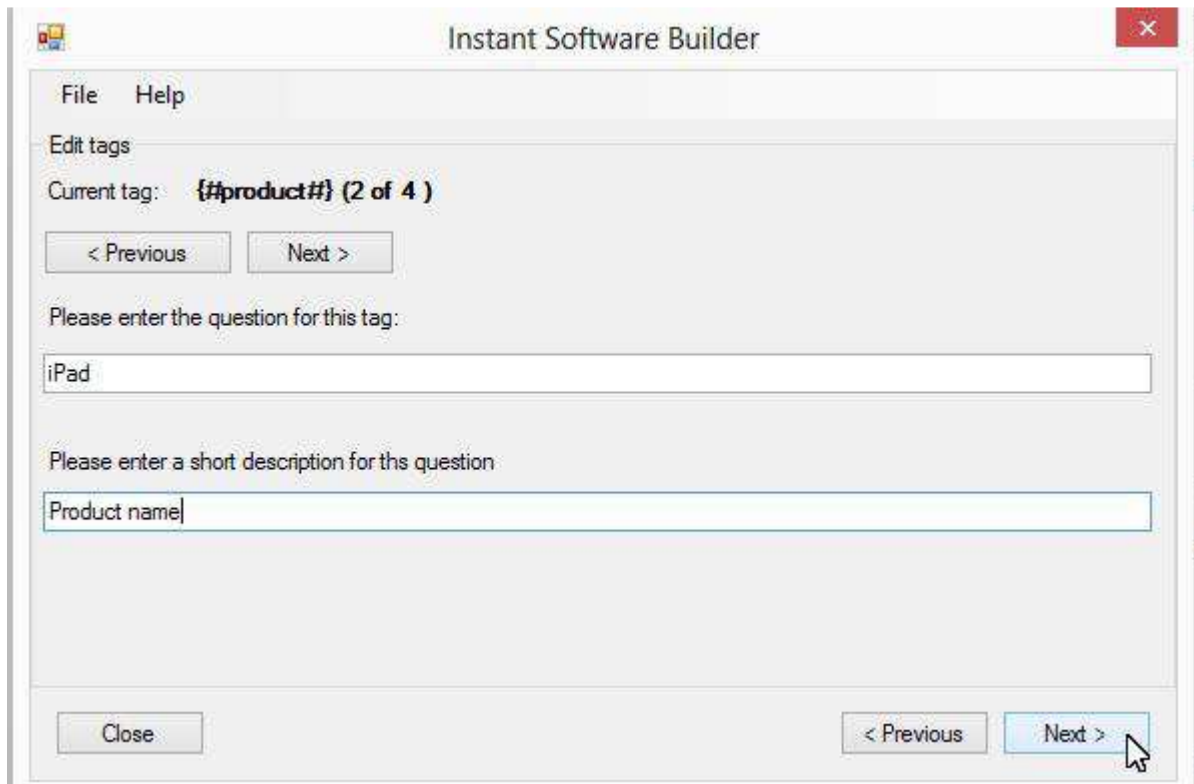
The screenshot shows a window titled "Instant Software Builder" with a standard Windows interface. It has a menu bar with "File" and "Help". Below the menu bar, it says "Enter your recipe" and "Type your recipe in the text box below". A note states "Tags must be enclosed by {# and #}". A large text area contains the following text: "Dear {#user#}, Thank you for purchasing {#product#}. Please register by sending email to {#email#}. Thank you. Best regards, {#company#}|". At the bottom, there are four buttons: "Close", "Paste from Clipboard", "Load from file", and a pair of navigation buttons "< Previous" and "Next >". A mouse cursor is pointing at the "Next >" button.

Tags are listed:



Enter question and help for each tag:





Instant Software Builder

File Help

Edit tags

Current tag: **{#product#} (2 of 4)**

< Previous Next >

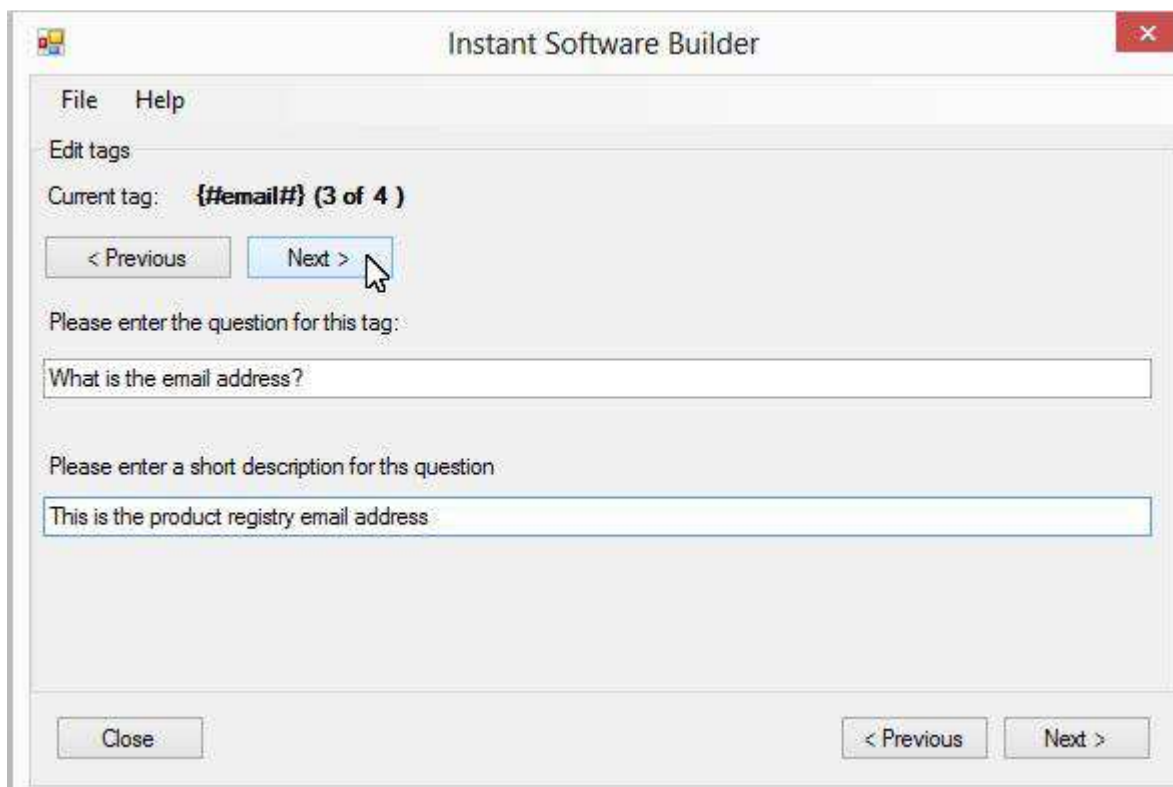
Please enter the question for this tag:

iPad

Please enter a short description for this question

Product name

Close < Previous Next >



Instant Software Builder

File Help

Edit tags

Current tag: **{#email#} (3 of 4)**

< Previous Next >

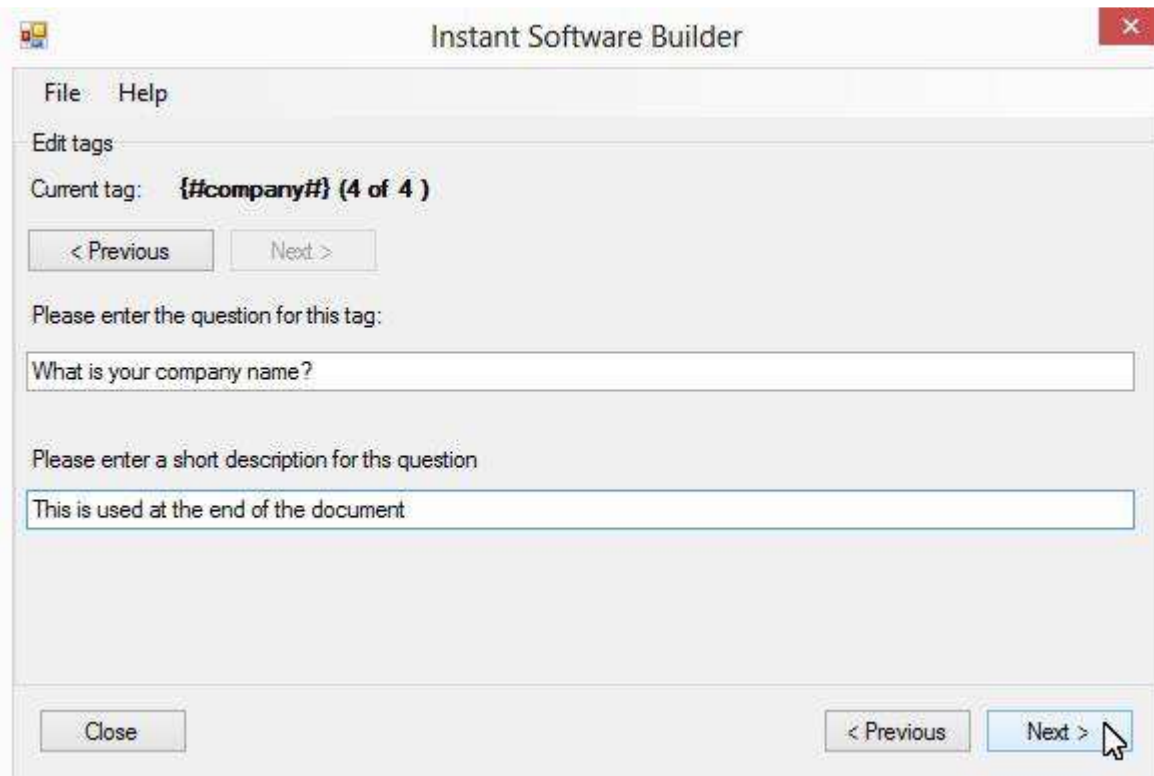
Please enter the question for this tag:

What is the email address?

Please enter a short description for this question

This is the product registry email address

Close < Previous Next >



Instant Software Builder

File Help

Edit tags

Current tag: **{#company#} (4 of 4)**

< Previous Next >

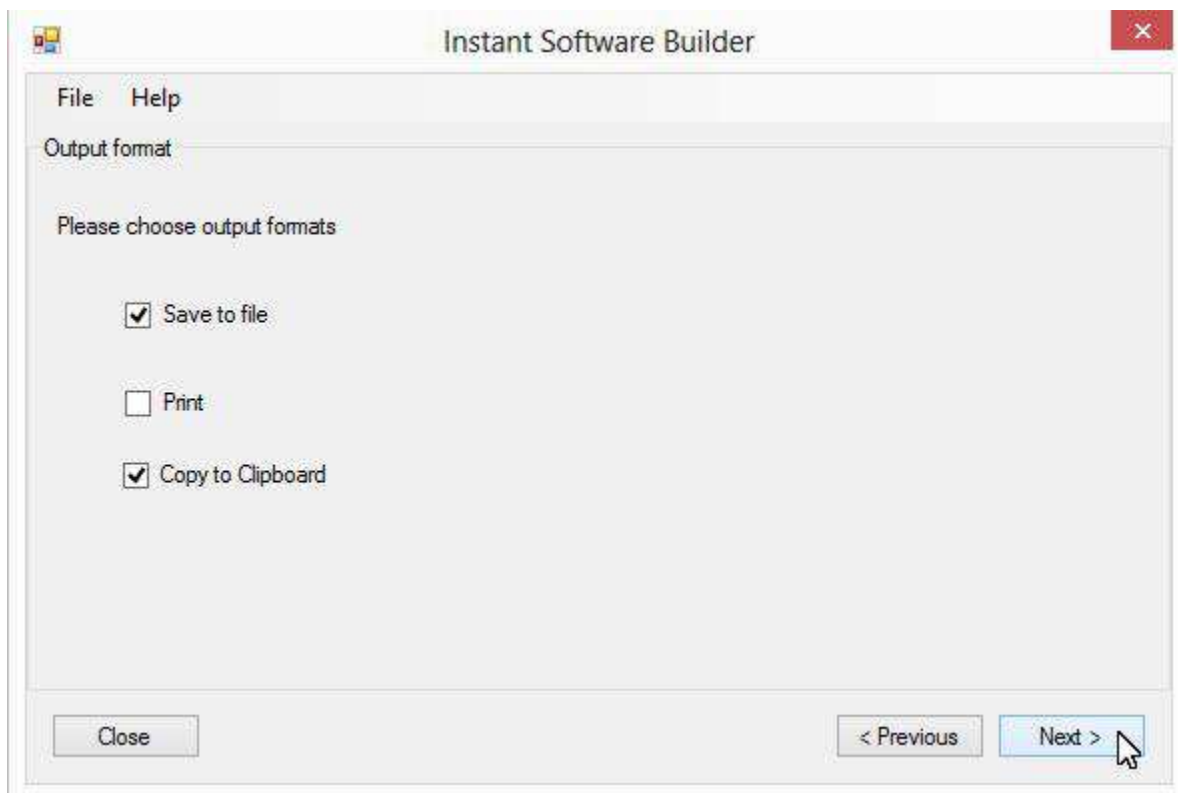
Please enter the question for this tag:

What is your company name?

Please enter a short description for this question

This is used at the end of the document

Close < Previous Next >



Instant Software Builder

File Help

Output format

Please choose output formats

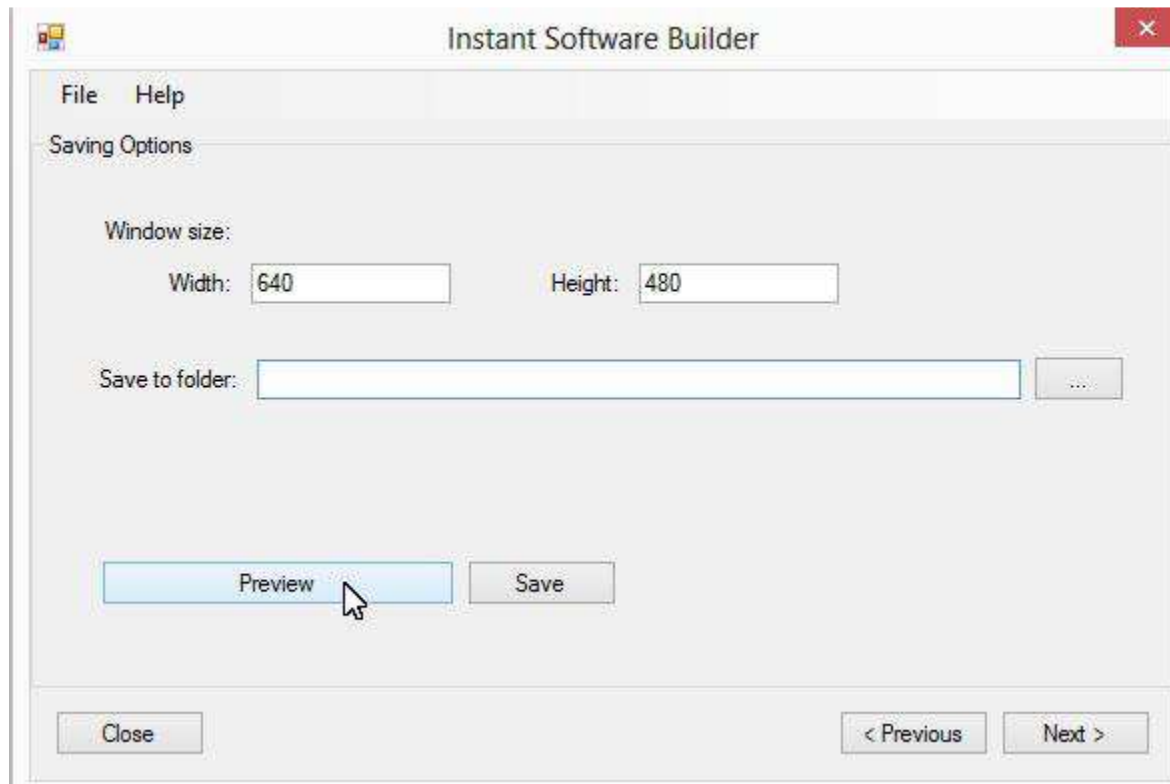
☒ Save to file

☐ Print

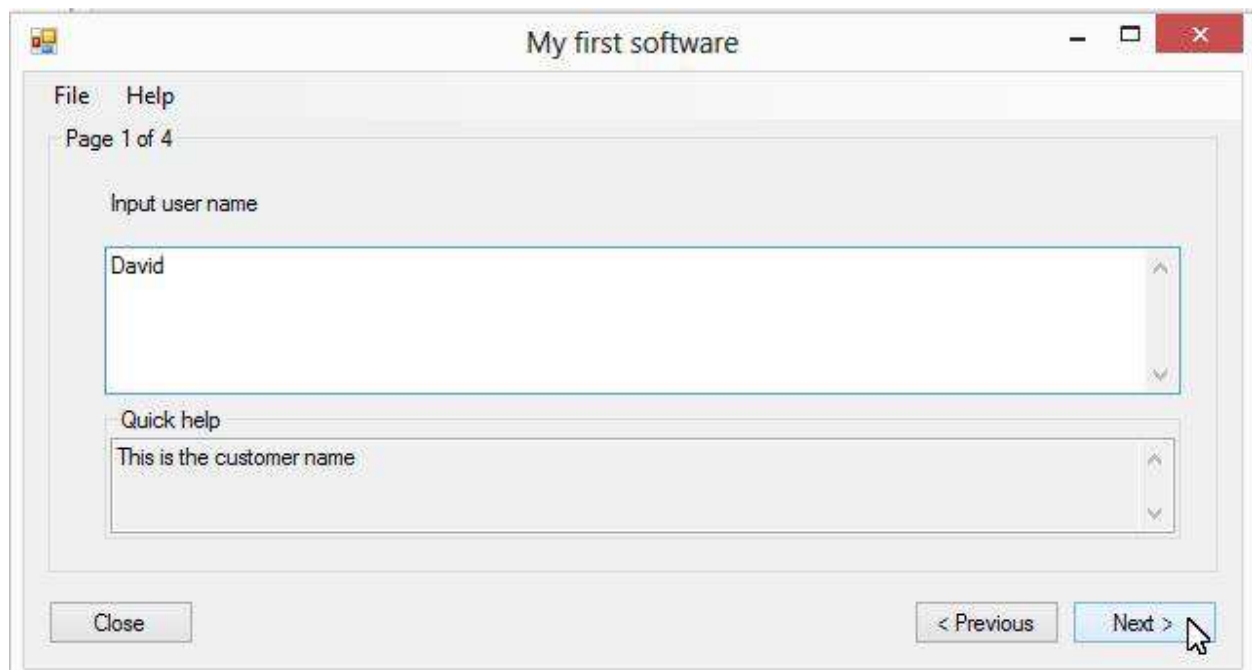
☒ Copy to Clipboard

Close < Previous Next >

Click Preview:



The first screen of RunTimeApp.Exe appears. Enter a name and click Next:



Enter data in the text box and click Next:

My first software

File Help

Page 2 of 4

iPad

iPad Pro

Quick help

Product name

Close < Previous Next >

Enter data in the text box and click Next:

My first software

File Help

Page 3 of 4

What is the email address?

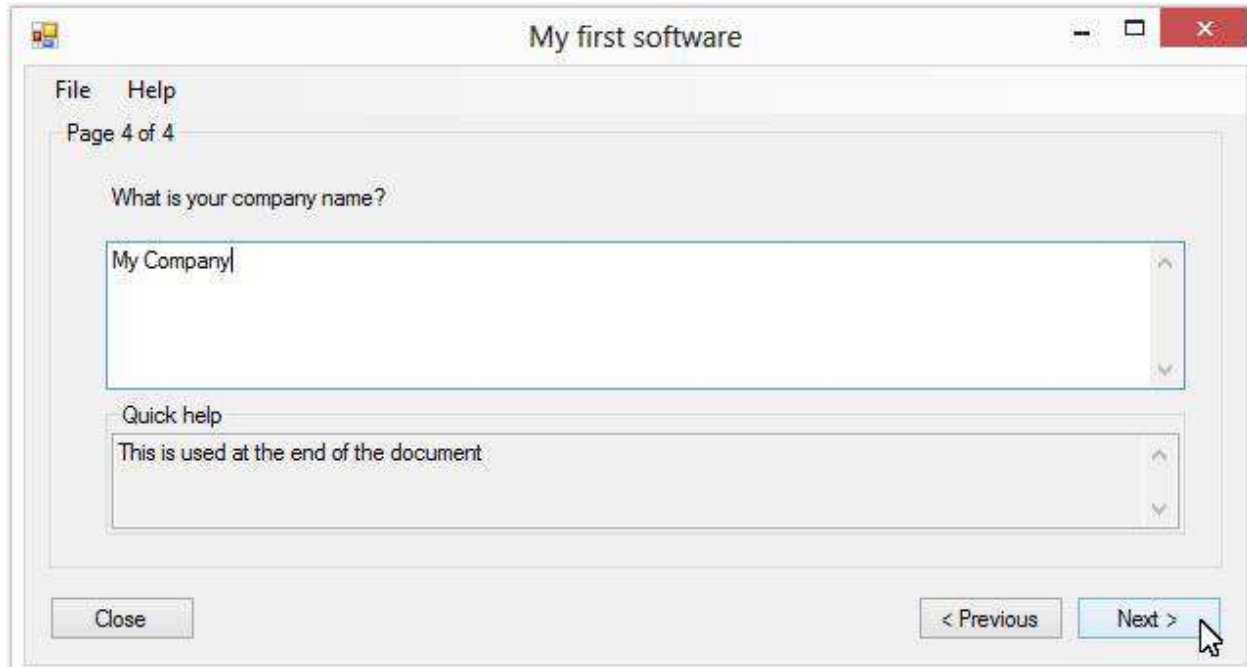
myname@email.com

Quick help

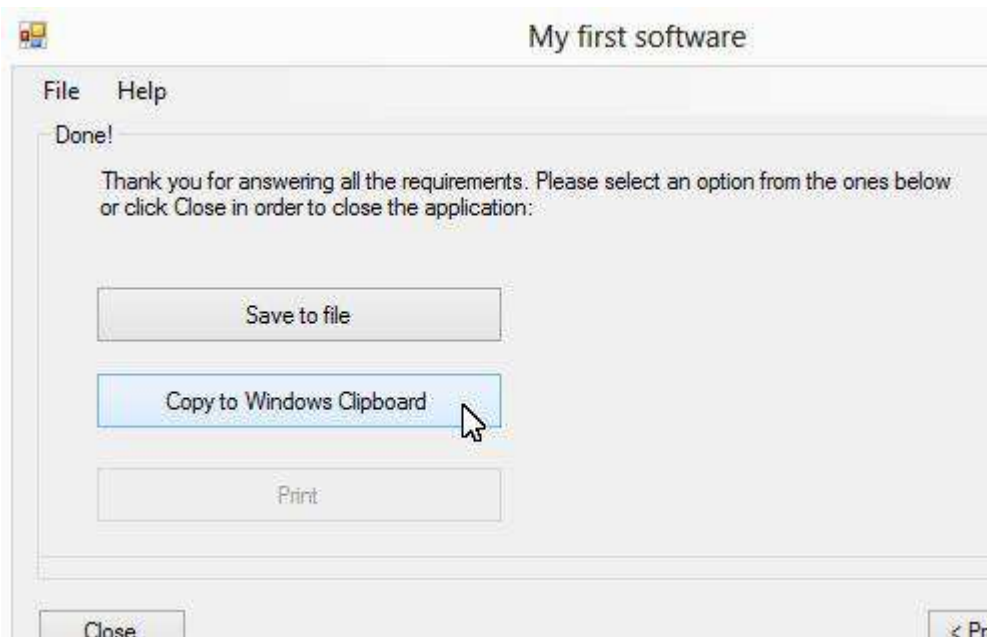
This is the product registry email address

Close < Previous Next >

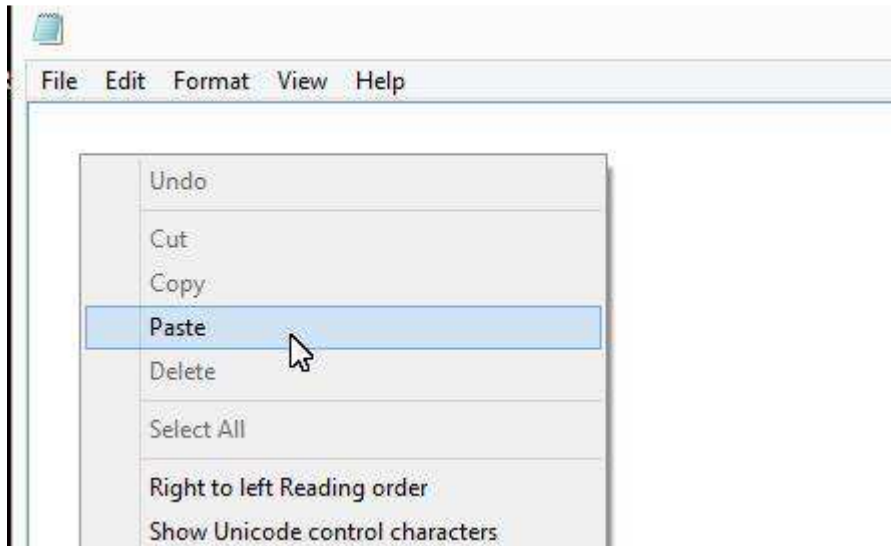
Enter data in the text box and click Next:



Send output to the Clipboard:



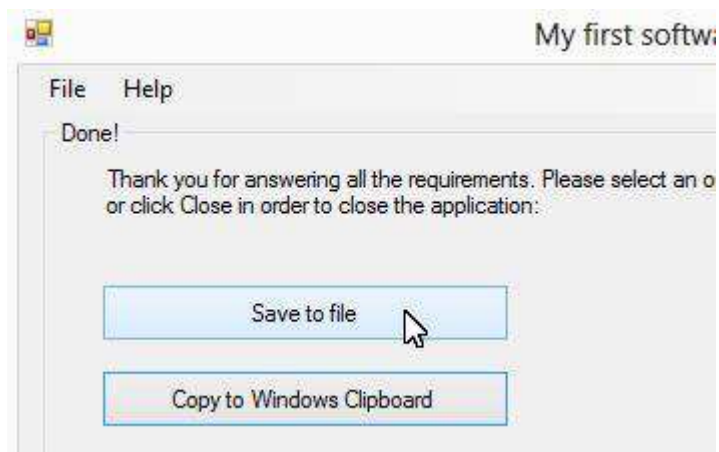
Paste to a Notepad window:



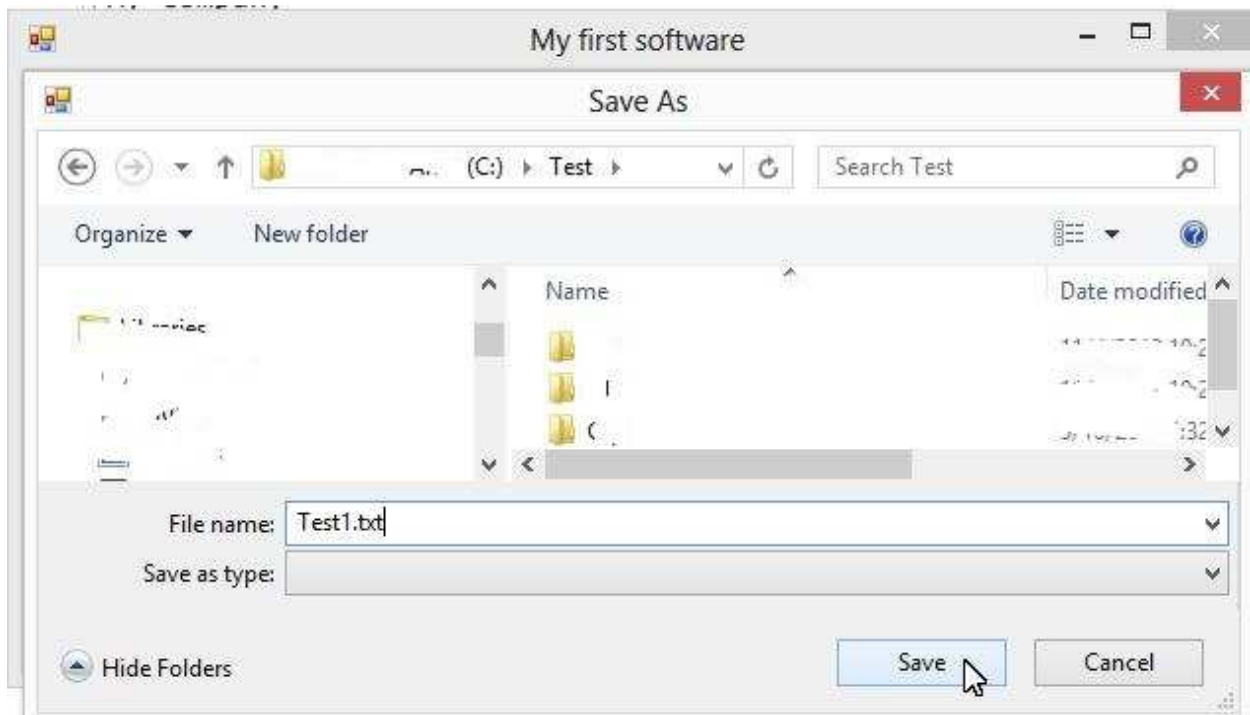
We can see that our data are merged into the document:



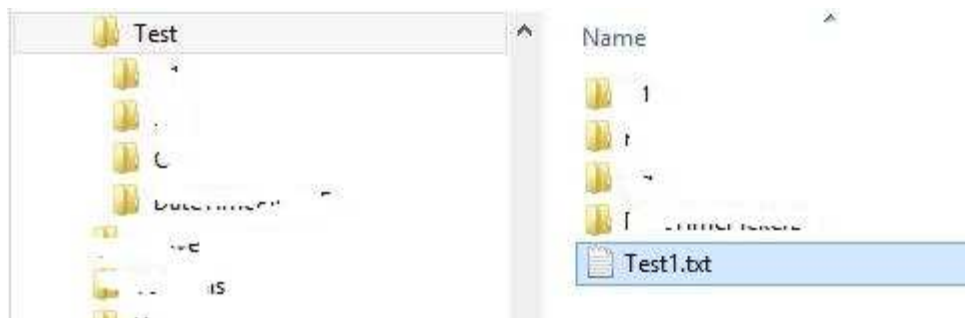
Click "Save to file":



Give a file name:



A file is generated:



File contents are merged data with template:



More Information

Read <http://www.limnor.com/support/UseAppConfig.pdf> for basic understanding of Application Configurations.

Read <http://www.limnor.com/support/UseAppConfigForBranding.pdf> to see how multiple configuration sets are used.

Read <http://www.limnor.com/support/CreateAppConfigUtility.pdf> to see how to separate configuration modification from the program being configured.

Feedback

Please send your feedback to support@limnor.com