

User Guide – Loop and Condition

[Create a variable at design time](#)

[Create performers](#)

[Create actions](#)

[SetLoopNumber](#)

[ClearAllItems](#)

[AddOneItem](#)

[DecreaseLoopIndex](#)

[LoopAction](#)

[CreateItems](#)

[Assign actions to the button-click event](#)

This sample application demonstrates a way of creating loop in Limnor programming, and use action condition to exit the loop. You may also use LoopWhile and LoopFor performers (see <http://www.limnor.com/downloads/UseLoopPerformers.doc> or <http://www.limnor.com/downloads/UseLoopPerformers.pdf>).

This sample uses a variable as the loop index to loop from a positive number to 0. The variable is saved in the Variables property of the page under name “i”. A Math Expression performer is used to decrease variable i. When variable i is reduced to 0, the loop stops. An action condition is used to check that variable i is larger than 0.

Action “SetLoopNumber” sets the number user typed in the text box to variable i. This gives variable i the initial value.

Action “DecreaseLoopIndex” is created to reduce variable i by 1. The Math Expression’s formula is $x - 1$, and x is linked to variable i. The action “DecreaseLoopIndex” evaluates $x - 1$ and set the result back to variable i.

The loop has to do something useful. For this sample, it adds variable i to a list box so that we can see the loop is working. Action “AddOneItem” is created to do that.

The major things demonstrated in this sample are the action list “LoopAction” which creates a loop and an action condition is used to control when to exit the loop. The action list “LoopAction” includes the following actions:

1. [AddOneItem](#)
2. [DecreaseLoopIndex](#)
3. [LoopAction](#)

Note that the last action is the action list itself. This is one of ways of creating a loop in Limnor. To get out of this loop, we use a condition, `Page1.Variables(i) > 0`, on the action “LoopAction”. That is, if variable i is 0 or below 0, “LoopAction” will be disabled and thus stops the loop.

In Limnor (after version 3.3.1430.36555) if you create a loop by including the action list itself at the end of the action list, then you create a “tail-recursion”. Tail-recursion does not have stack-overflow problem. The loop can go unlimited times.

This sample application works in this way: the user clicks the button to start the following actions: 1) remove all existing items in the list box; 2) set the value of the variable i to the value on the TextBox performer (the user types in the value); 3) append the value of variable i to the list box; 4) reduce the value of variable i by 1; 5) if the value of variable i is larger than 0, go back to step 3, otherwise finish the actions.

Below we describe how this sample application is made.

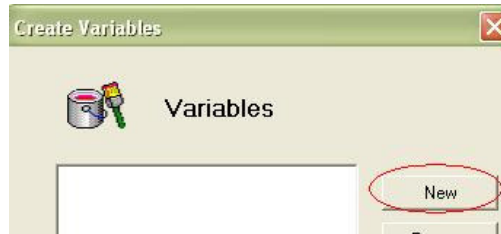
Create a variable at design time

To create variables at the design time, set Variables property:

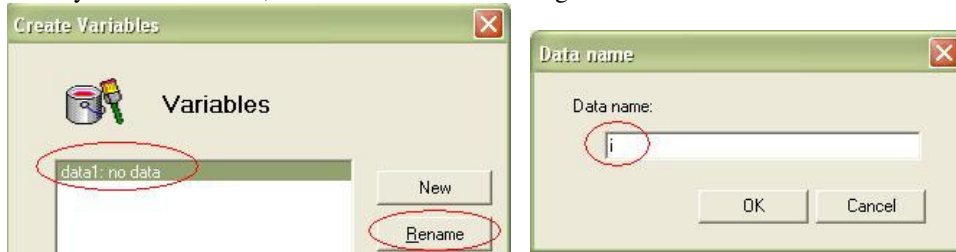
Loop and Logic



Click “New” button to create a new variable:

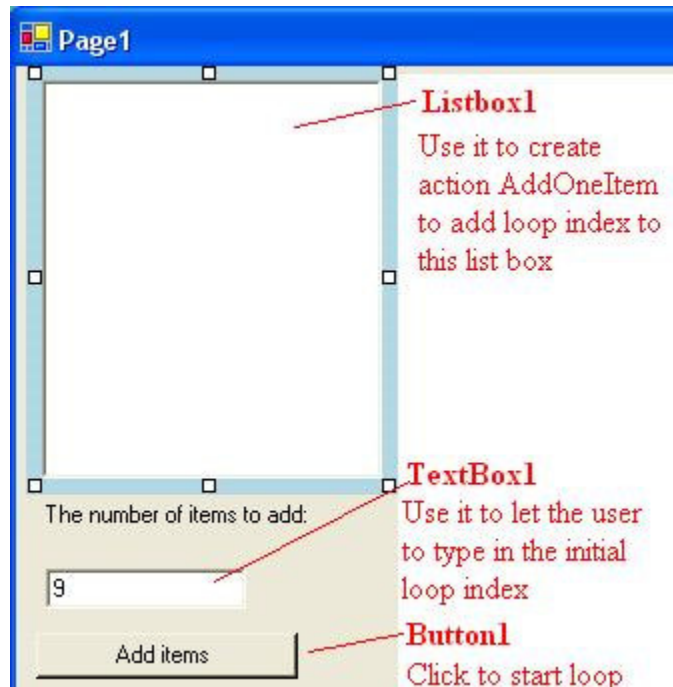


Select the newly created variable, click “Rename” button to give it the name we want:



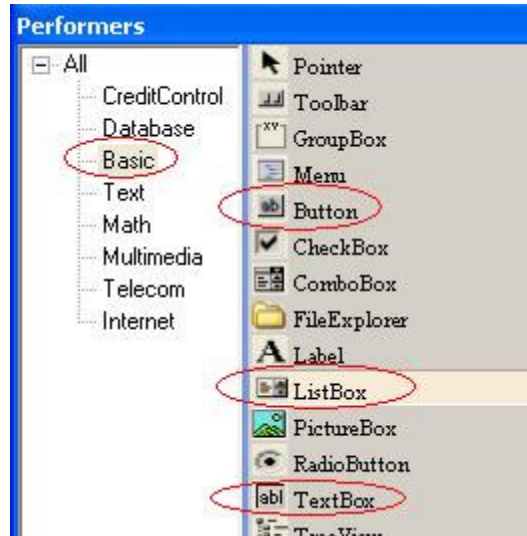
Create performers

For the user interface of this application, create a ListBox performer, a TextBox performer and a Button performer:

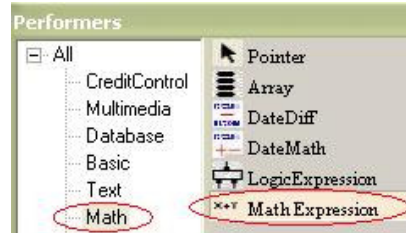


You may find the above 3 types of performers under “Basic”:

Loop and Logic



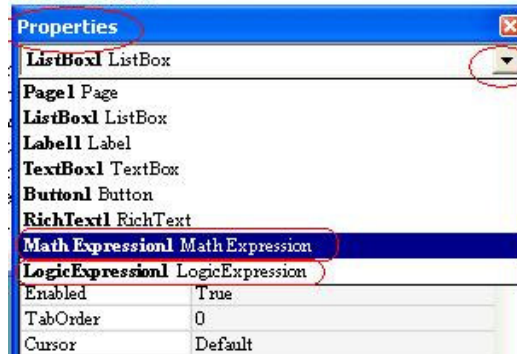
We need to use math calculations. We add a Math Expression performer. You can find the Math Expression performer type under “Math”:



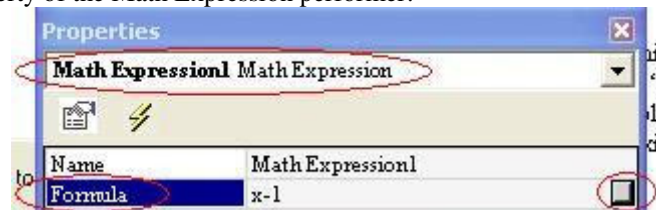
Because Math Expression performer does not have user interface, it is displayed in the “Extra Performers” window of the page:



You may also access such performers through the Properties window of the page:

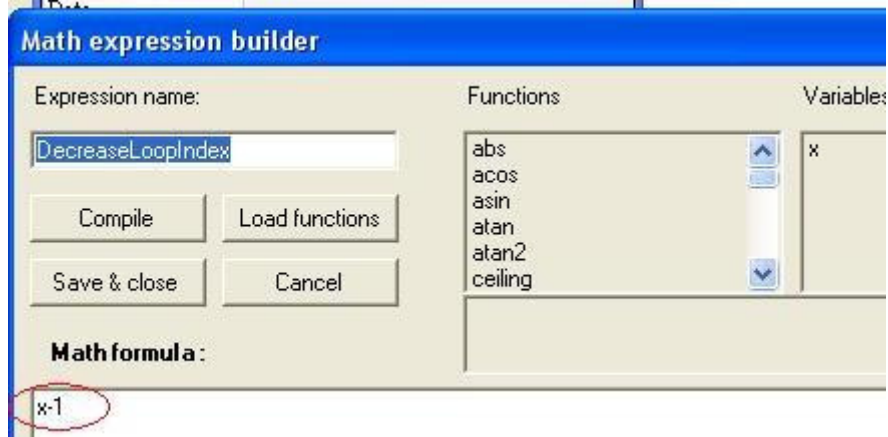


Choose Formula property of the Math Expression performer:



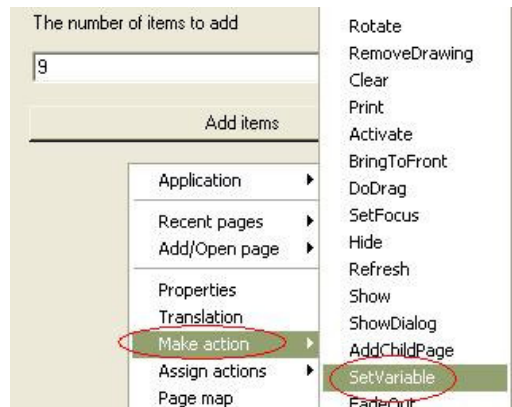
And set it to $x - 1$:

Loop and Logic

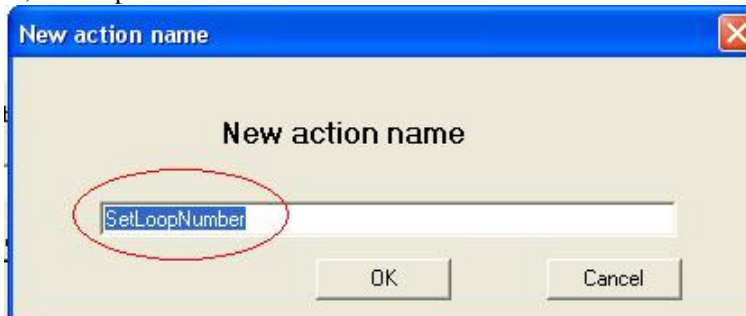


Create actions

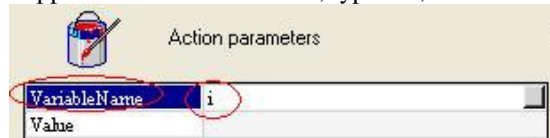
1. Create an action to assign the value on TextBox1 to variable i. We will name it as **SetLoopNumber**. Executing of this action initialize variable i to the value the user typed in. Right-click on the page, choose “Make action”, choose “SetData”:



Give an action name, SetLoopNumber:

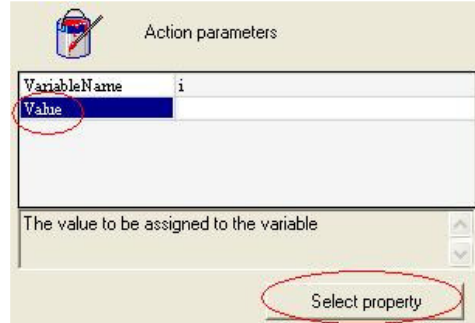


The action data dialogue box appears. For VariableName, type in i, this is the variable name:

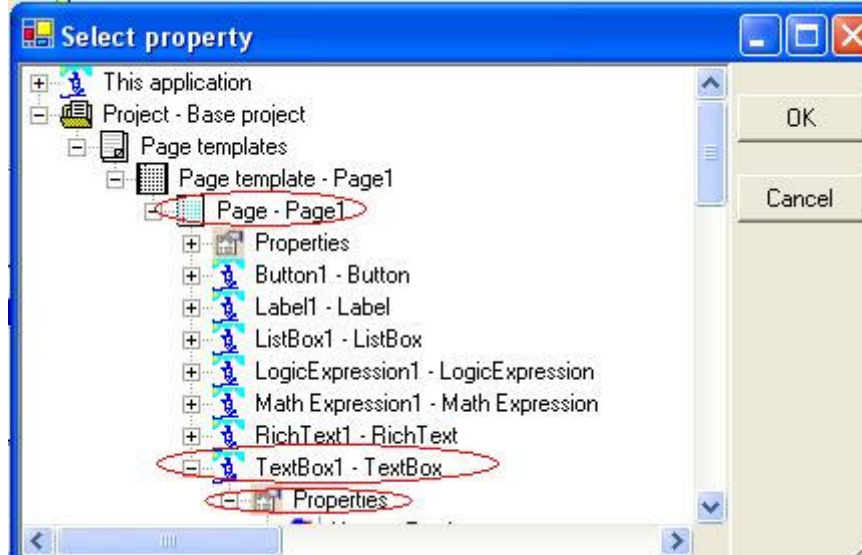


For Value, click “Select property” button:

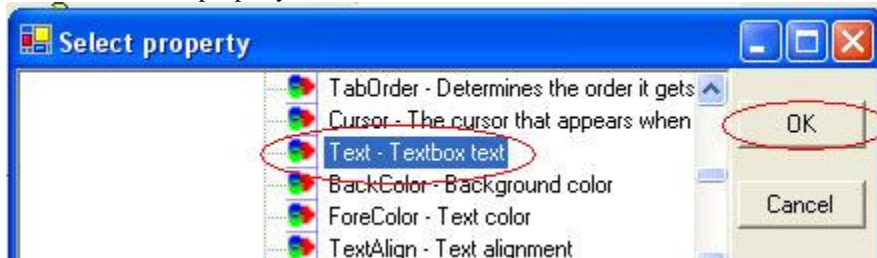
Loop and Logic



Find the TextBox performer, expand its properties node:

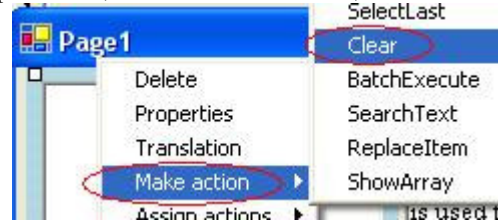


Scroll down and select "Text" property:



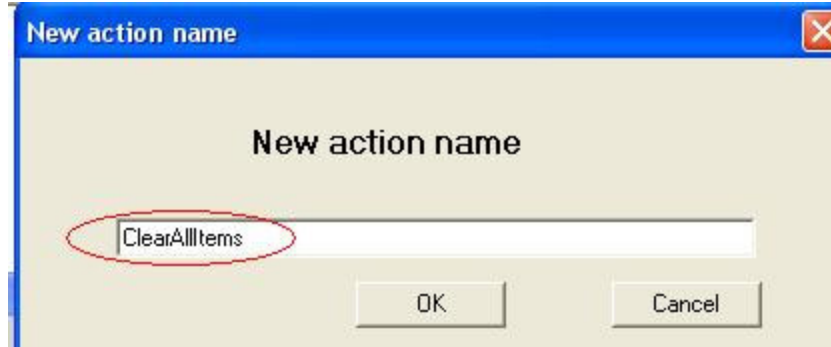
2. Create an action to clear the ListBox. We will name it as **ClearAllItems**. Executing of this action removes all items in the list box.

Right-click on the ListBox performer, choose "Make action", choose "Clear":

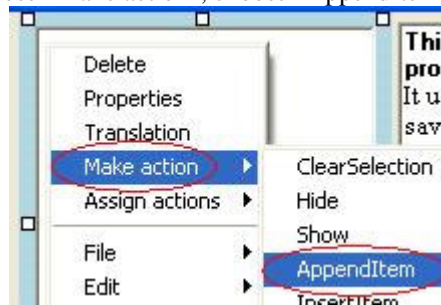


Type in the action name, ClearAllItems:

Loop and Logic



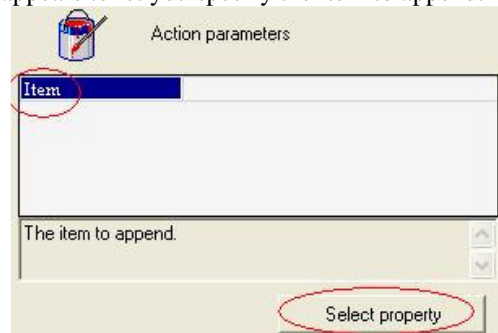
3. Create an action to append variable *i* to the ListBox as a list box item. We will name it as **AddOneItem**. Executing of this action shows the value of variable *i* in the list box. Right-click on the ListBox1, choose "Make action", choose "AppendItem":



Type in the action name, AddOneItem:

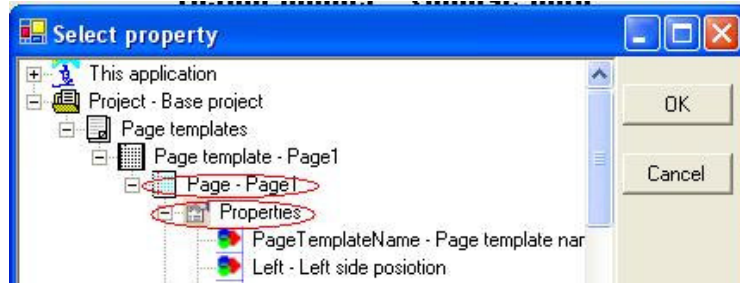


The action data dialogue box appears to let you specify the item to append. Click "Select property" button:

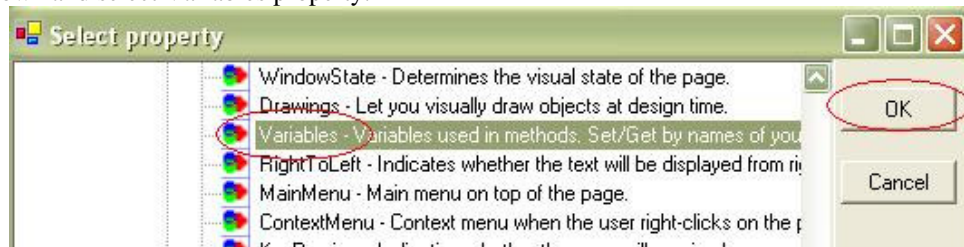


Find the page and expand its properties node:

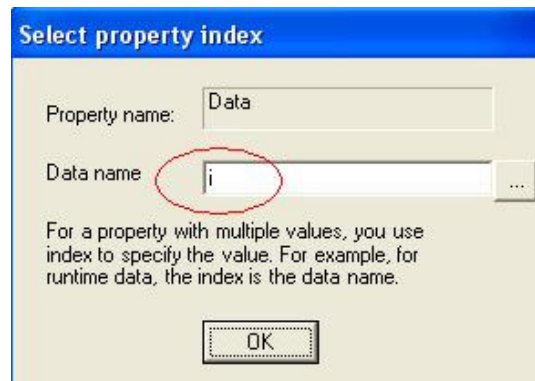
Loop and Logic



Scroll down and select Variables property:

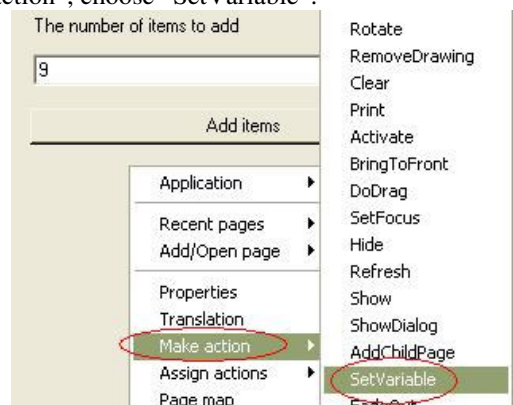


Type in the variable name i:



4. Create an action to reduce the value of variable *i* by 1. We will name this action as **DecreaseLoopIndex**.

Because the Math Expression performer is using an expression of $x - 1$, and x is linked to variable *i*, so the Result property of the Math Expression performer is the value of variable *i* reduced by 1. We just need to set Result property to variable *i*. Setting variable value is done by Page's SetVariable method. Right-click on the page, choose "Make action", choose "SetVariable":

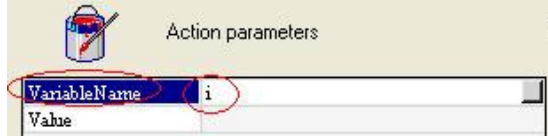


Type in the action name, DecreaseLoopIndex:

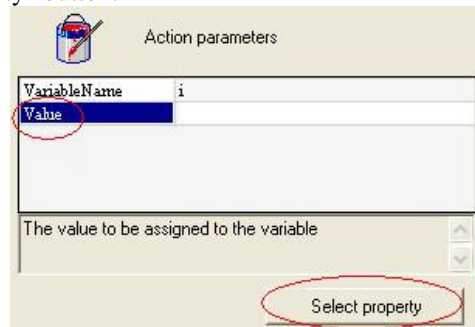
Loop and Logic



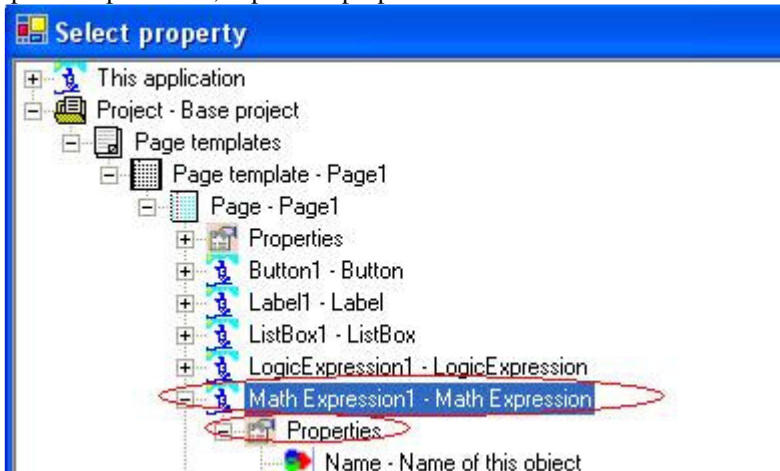
The action data dialogue box appears. For VariableName, type in i, this is the variable name:



For Value, click “Select property” button:

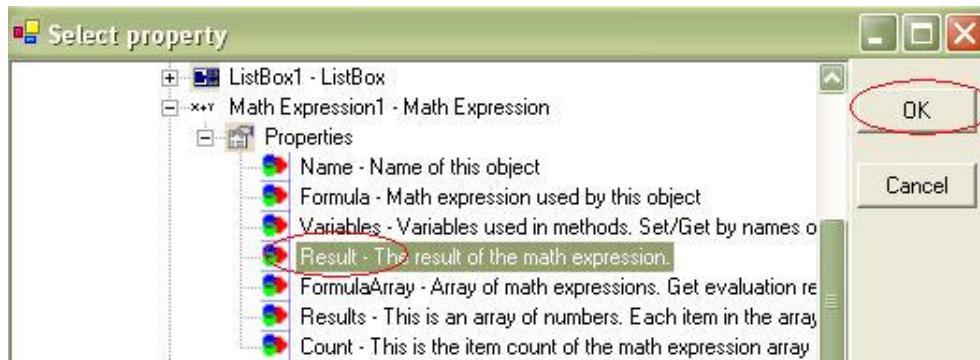


Find the Math Expression performer, expand its properties node:



Scroll down and select Result property:

Loop and Logic

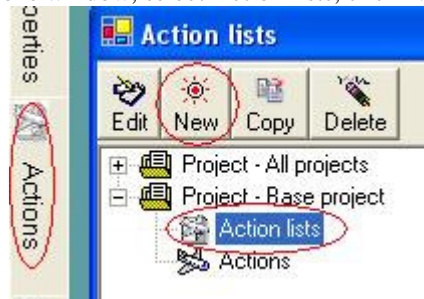


5. Create an action list to do looping. We will name it as **LoopAction**. The action list LoopAction includes the following actions:

1. **AddOneItem** – this action shows the value of variable *i* in the list box.
2. **DecreaseLoopIndex** – this action reduce the value of variable *i* by 1.
3. **LoopAction** – this is the action list itself, so it goes back to execute **AddOneItem** and **DecreaseLoopIndex** again.

Note that the last action is the action list itself. This is one of ways of creating a loop in Limnor. To get out of this loop, we use a logic expression, $i > 0$, to enable “LoopAction”. That is, if variable *i* is 0 or below 0, “LoopAction” will be disabled and thus stops the loop. The Result property of the Logic Expression performer contains the result of the logic expression, $i > 0$. So we will use the Result property to enable this action list.

Click Actions to bring up the actions window, select Action lists, click New button:



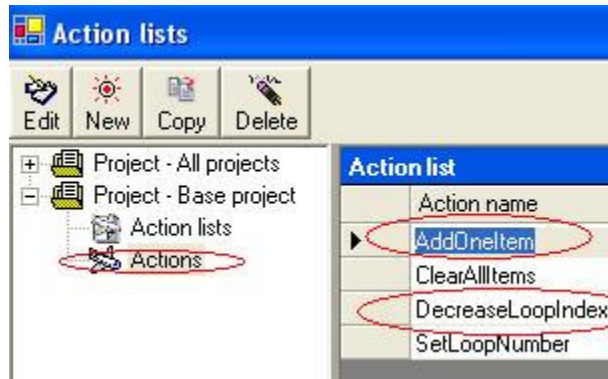
Type in the action list name, LoopAction:



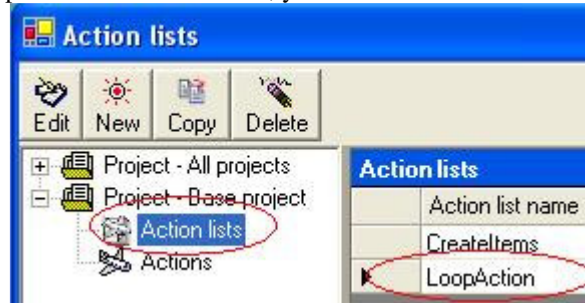
Use the Add button to add AddOneItem, DecreaseLoopIndex, and LoopAction to the list.

Note that when you are adding AddOneItem and DecreaseLoopIndex to the action list, you need to find them under “Actions”:

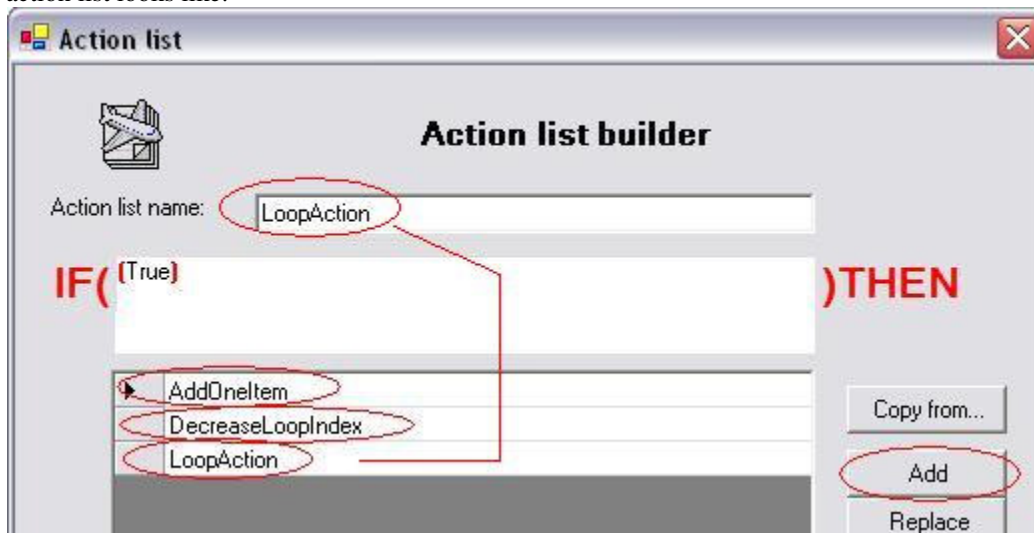
Loop and Logic



When you are adding LoopAction to the action list, you need to find them under “Action lists”:



This action list looks like:



Note that this action list, LoopAction, is also the last action of the action list. This is a way of creating looping.



Execution actions in a loop

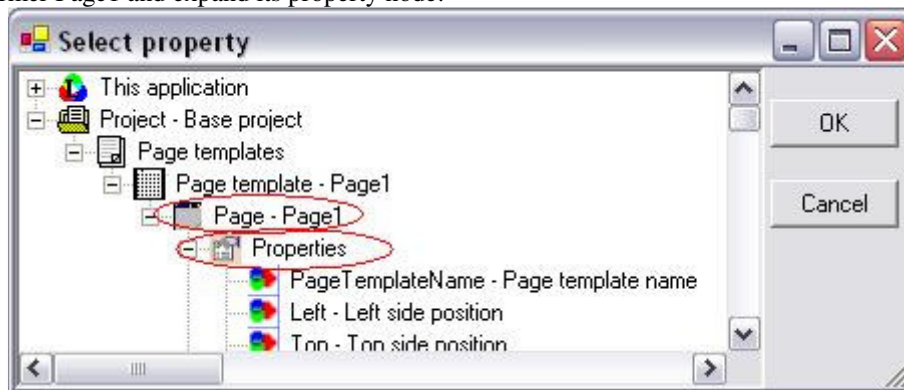
Loop and Logic

When this action list is executed, you can see it will run into an endless loop. Your computer will appear dead. So, it is very important to set its condition to exit the loop. For this action list, every time the action DecreaseLoopIndex is executed, the variable *i* is decreased by 1. We use condition $i > 0$ for this action list. So, when *i* is equal or smaller than 0 the action list is disabled and thus the loop breaks.

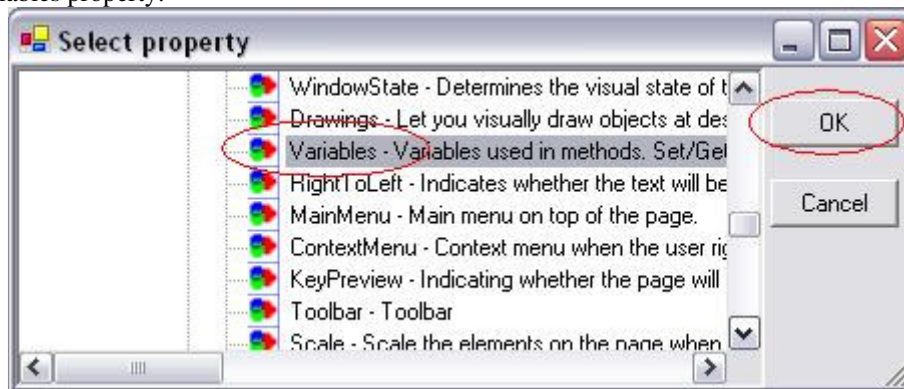
To set such a condition, right-click on the action condition (True) and choose "Add":



Find performer Page1 and expand its property node:

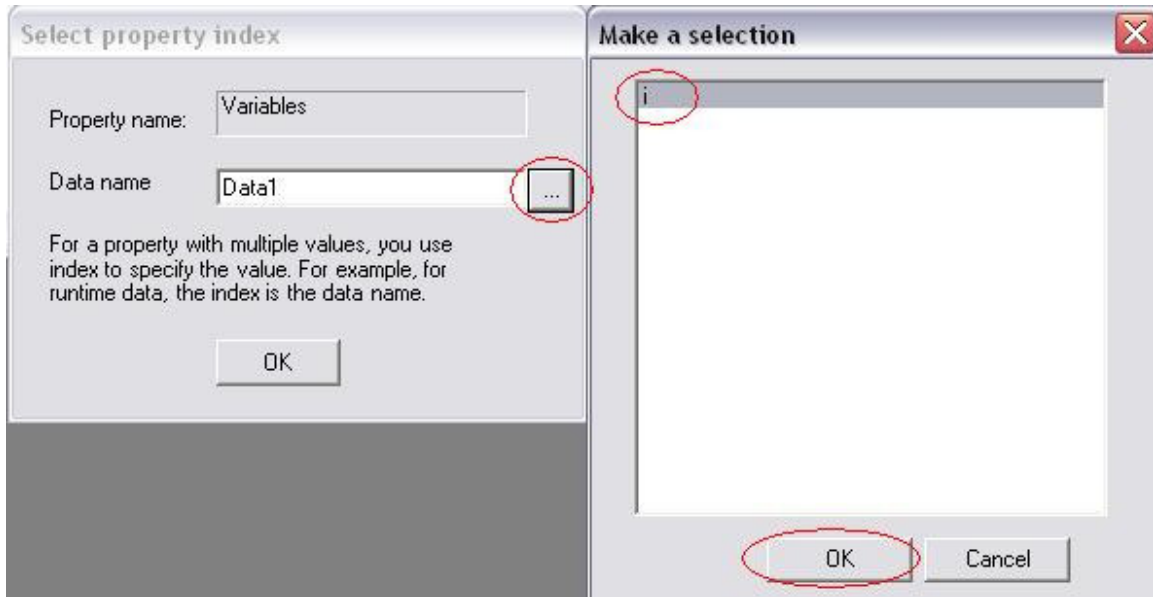


Select Variables property:



Select variable i:

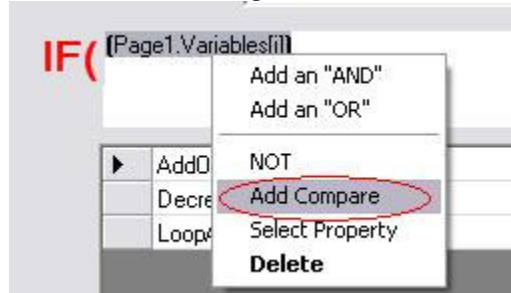
Loop and Logic



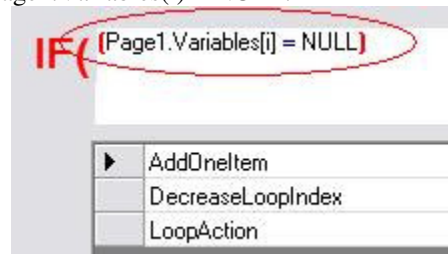
Now the variable i becomes the action condition:



Right-click on Page1.Variables(i), choose "Add compare":

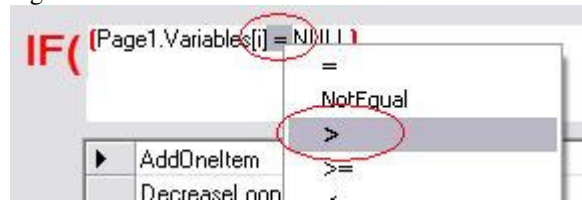


The action condition becomes Page1.Variables(i) = NULL:

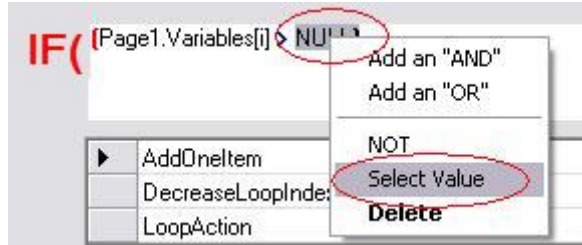


Loop and Logic

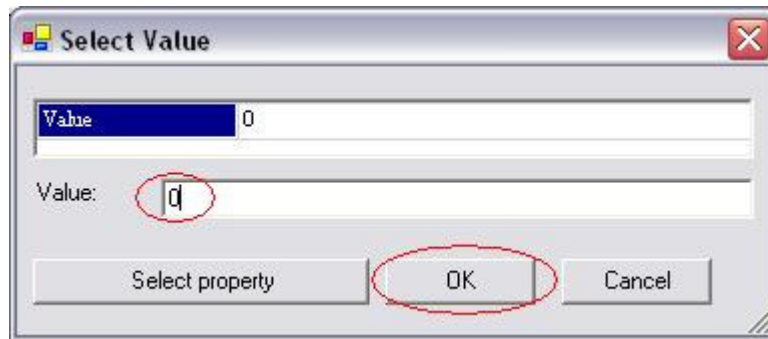
Select the symbol “=” and right-click on it. Choose “>”:



Right-click “NULL” and choose “Select Value”:



Type “0” for the Value:



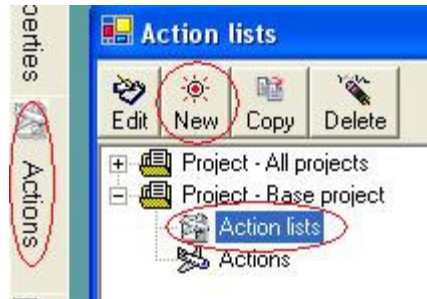
We are done making this action condition:



6. Create an action list to start the actions when the user clicks the button. We will name it as **CreateItems**. This action list will be assigned to the Click event of the button. This action list includes the following actions and action list:
 - ClearAllItems – this action removes all existing items in the list box
 - SetLoopNumber – this action set the value of the variable i to the Text property of the TextBox1. The Text property holds whatever the user types in the text box.
 - LoopAction – this is the action loop. Let N be the value the user types in the text box, this action list shows N, N-1, N-2, ..., 1 in the list box

Click Actions to bring up the actions window, select Action lists, click New button:

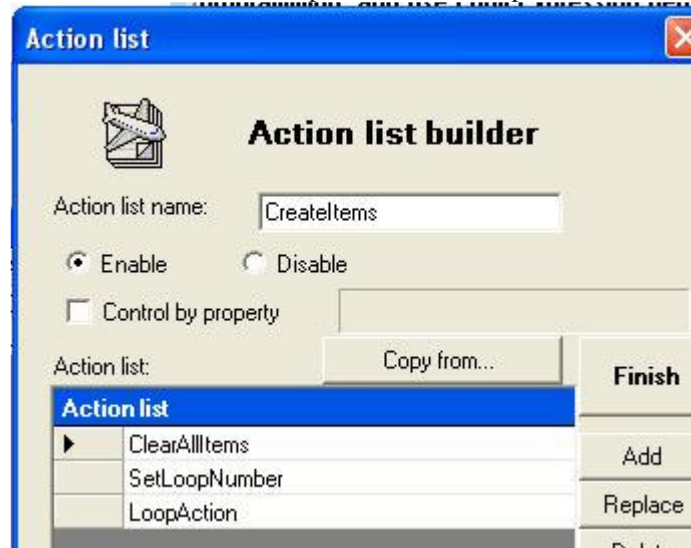
Loop and Logic



Type in the action list name, CreateItems:

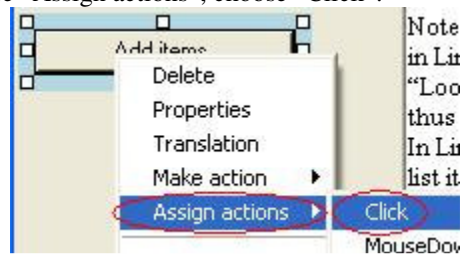


Use the Add button to add ClearAllItems, SetLoopNumber, and LoopAction to the list:



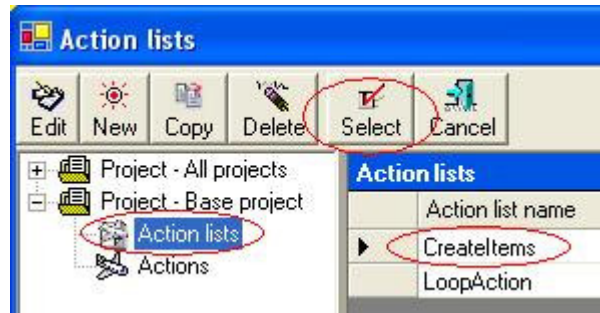
Assign actions to the button-click event

We want to execute CreateItems when the user clicks the button.
Right-click on the button, choose "Assign actions", choose "Click":



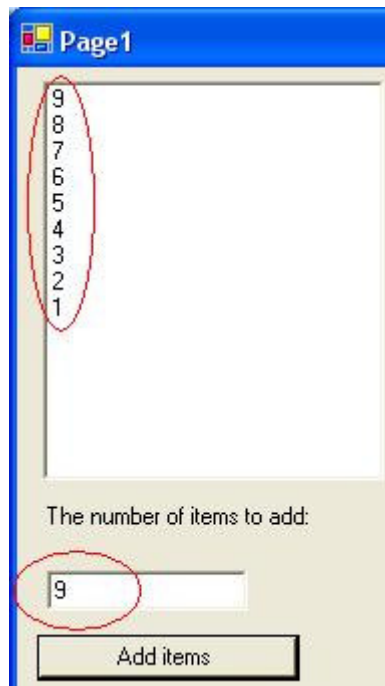
Select "Action lists", choose "CreateItems", click "Select" button:

Loop and Logic



We are done now.

Press F2 to run it and try it. Type in a value in the text box, click the button. You will see numbers appear in the list box:



=== EOF ===