Defining Logic Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, we will focus on developing some logic flows using an If, a Switch, and an implementation of an ad-hoc loop. These tools will be used to help us on the following scenarios:

- By clicking on a button with a number, the application will display a message indicating if the number is positive or negative.
- By clicking on a button with a number, the application will display a message indicating the number's order of magnitude (e.g: 10K, 100K, ...)
- By clicking on a button with a number, the application will display a message with the value of the number's factorial.

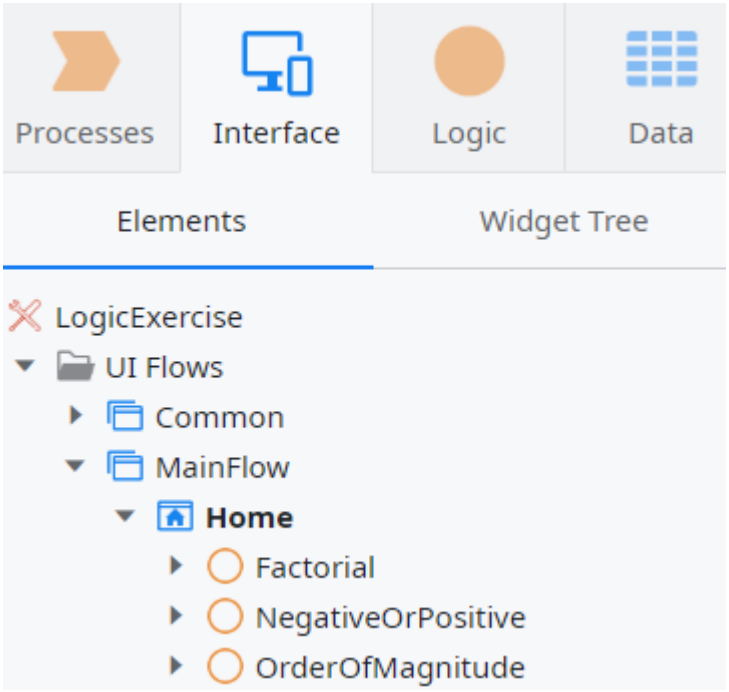
When this is completed, the application will have three Actions that will support all of these scenarios.

Resources

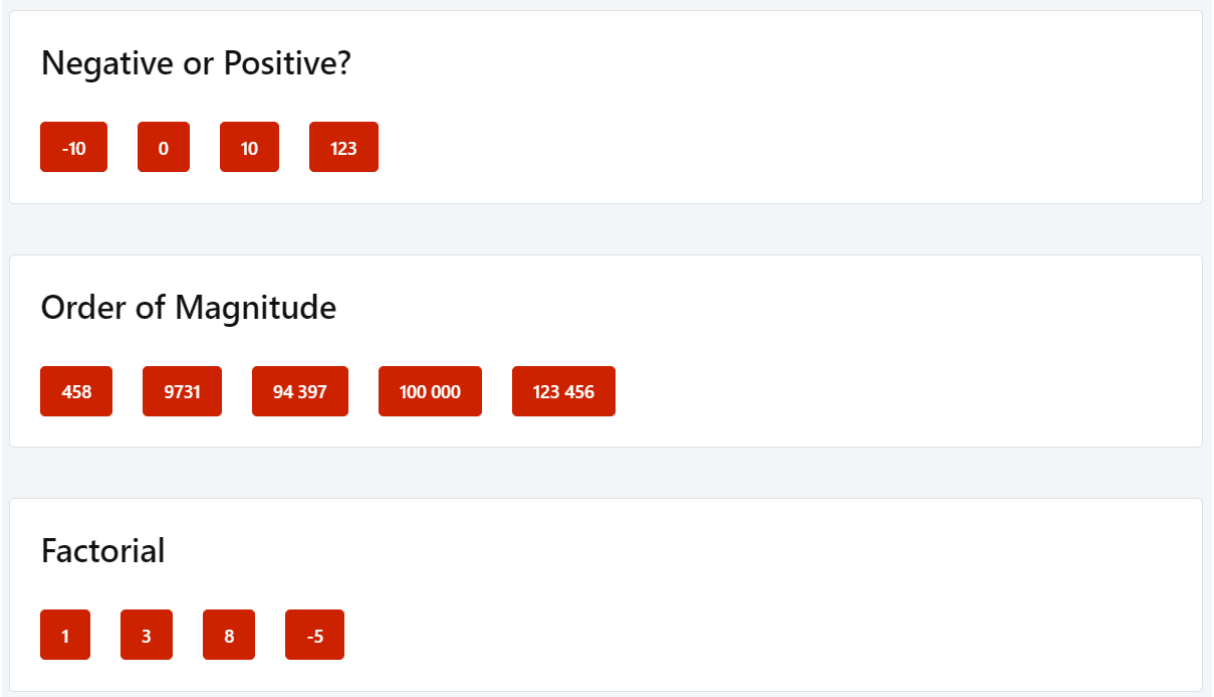
This exercise has a Quickstart application already created. This application has everything needed to start the exercise. This quickstart application can be found in the Resources folder of this exercise, with the name **Logic Exercise.oap**.

Scenario

In this exercise, we will start from an existing application with one module. Inside that module we have one screen, called **Home**.



The screen has three sections with a set of Buttons: **Negative or Positive?**, **Order of Magnitude**, and **Factorial**.



Each of these sections is related with the Screen Actions of the Home screen, with the same name. Specifically, each one of the Screen Actions is called when a button of the section of the screen with the same name is clicked.

This mechanism is already implemented. Starting with this, we want to implement three different scenarios, in each one of these Screen Actions. The general idea of the exercise consists of the user clicking one of the buttons, and a feedback message being displayed with a particular information related to the number in the button.

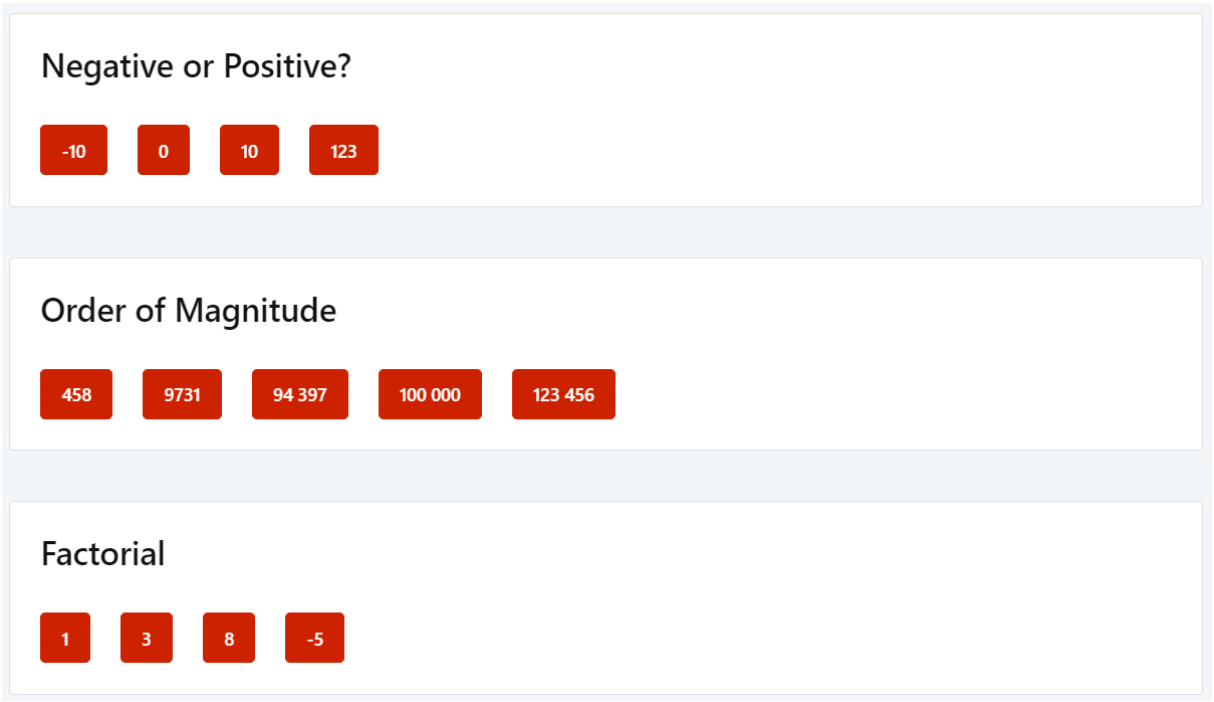
In the **Negative or Positive?** section, by clicking on a button, a message should appear indicating whether the number is positive, negative or zero. In the **Order of Magnitude** section, the message should indicate the order of magnitude of a number: 1K, 10K, 100K or >=100K. In the **Factorial** section, the message should display the factorial of the number in the button that is clicked.

How-To

In this section, we will show you how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again**. If you didn't finish the exercise, that's fine! We are here to help you.

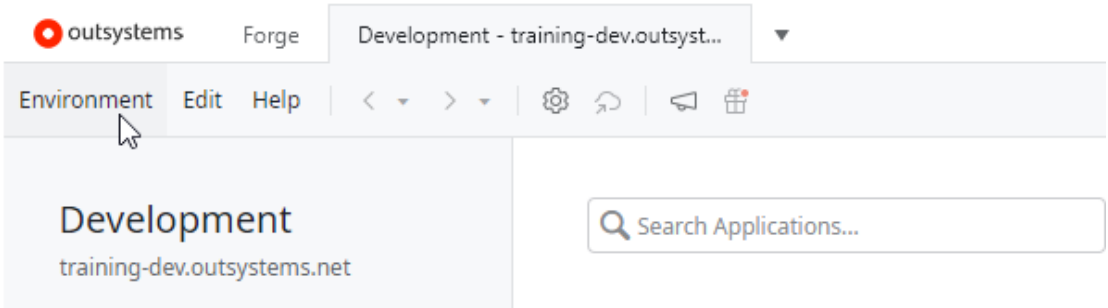
Getting Started

First, we need to install the Quickstart file, which has an application ready for the exercise. This application has one screen containing a set of buttons. These buttons are bound to three screen actions that you will implement in the course of the exercise.

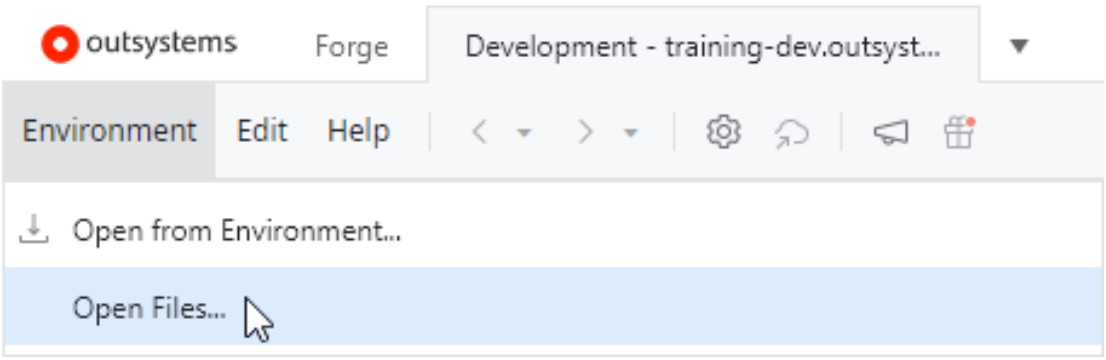


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

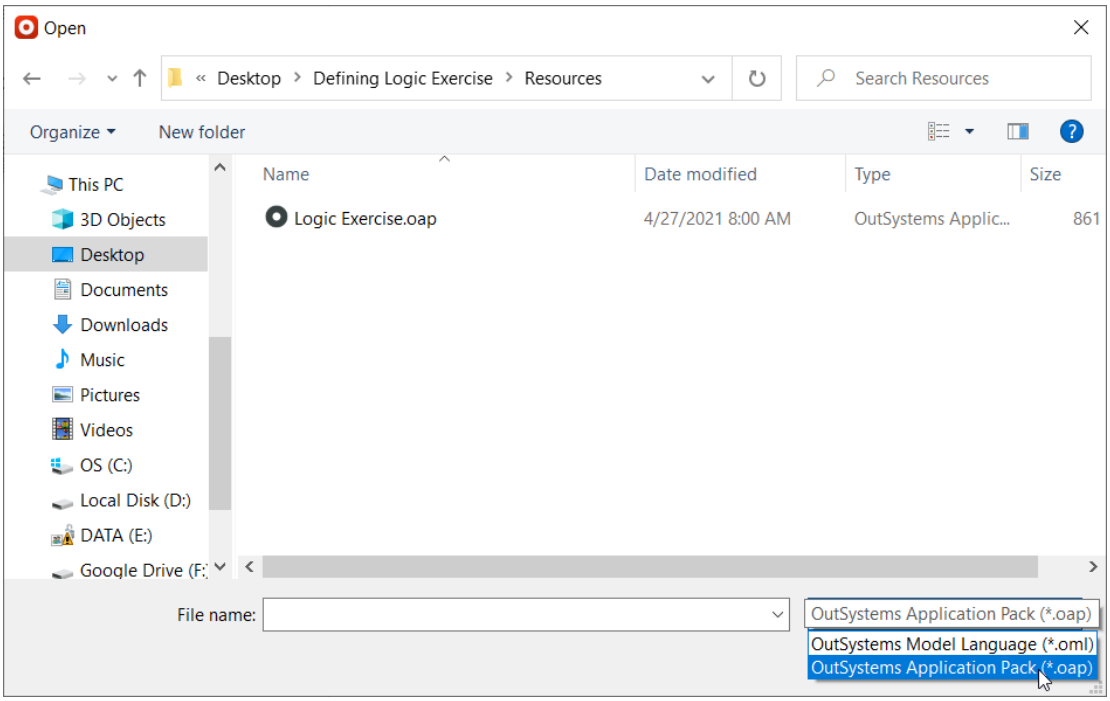
1) In the main window of Service Studio, select the Environment menu on the top left.



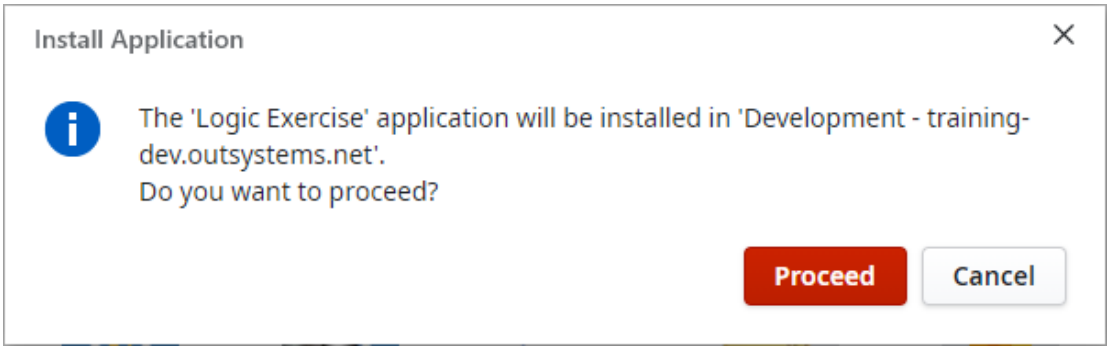
2) Select the Open Files...



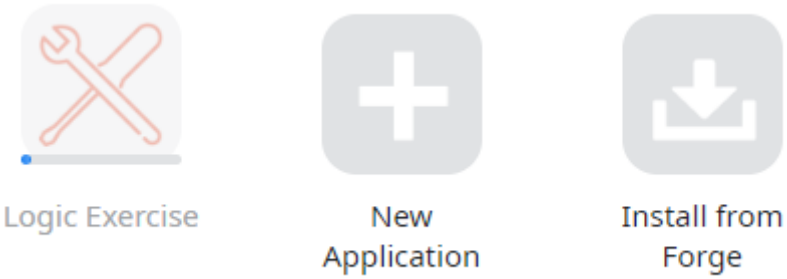
3) In the new dialog that appears, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Logic Exercise.oap**



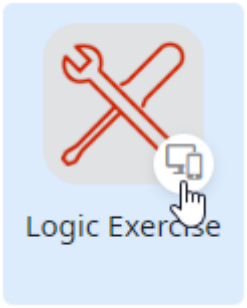
4) In the new confirmation dialog, select Proceed



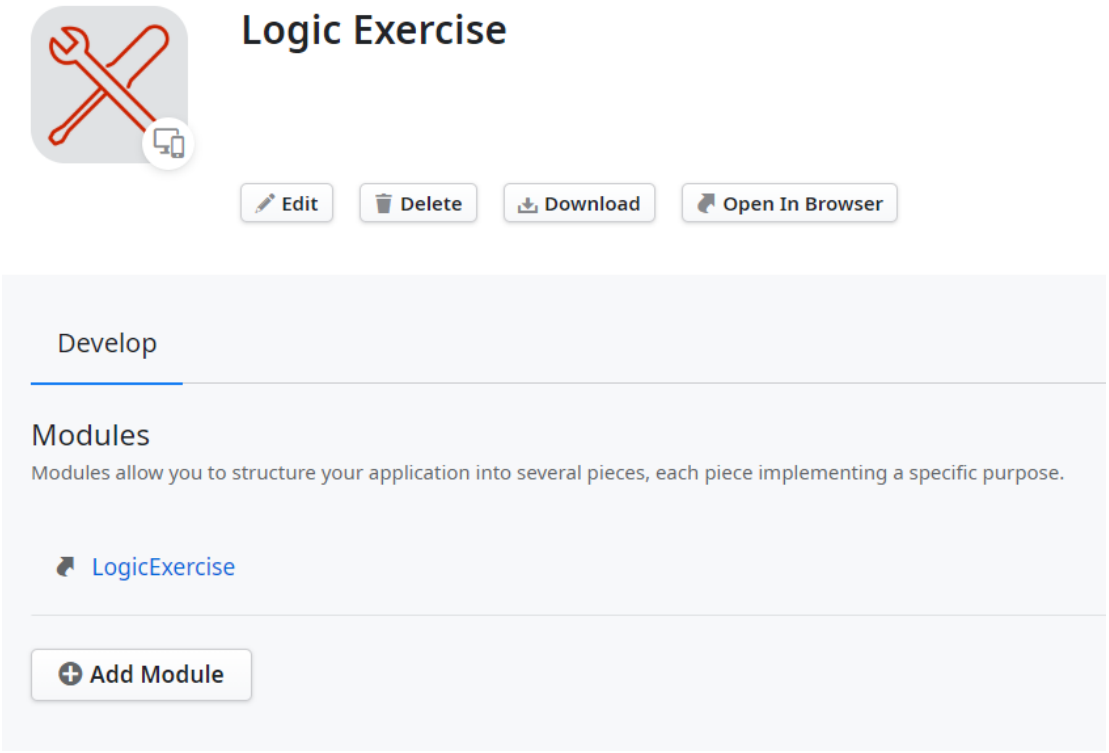
5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.



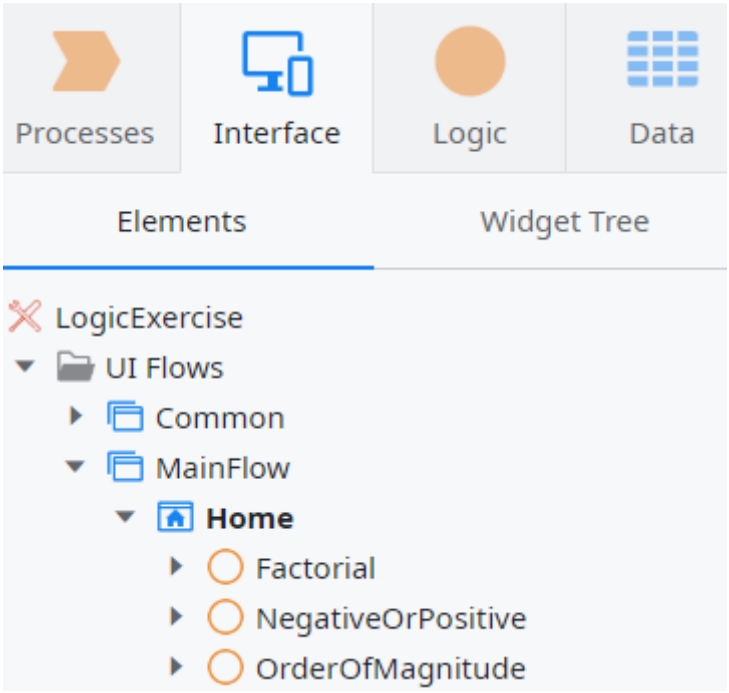
7) The application has only one module. Let's open it!



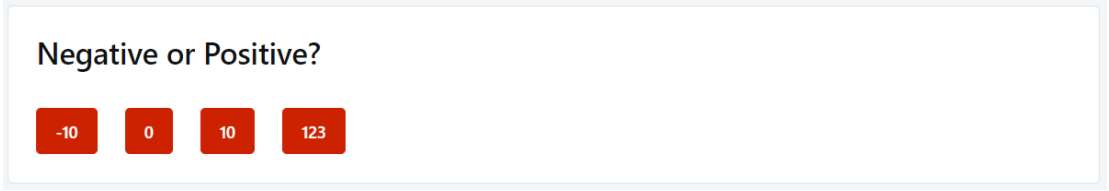
Negative Or Positive?

In this section, we will implement the first of three screen actions. This one will validate if a number is positive, negative or zero.

1) The module already has the Home Screen created, with three Screen Actions.

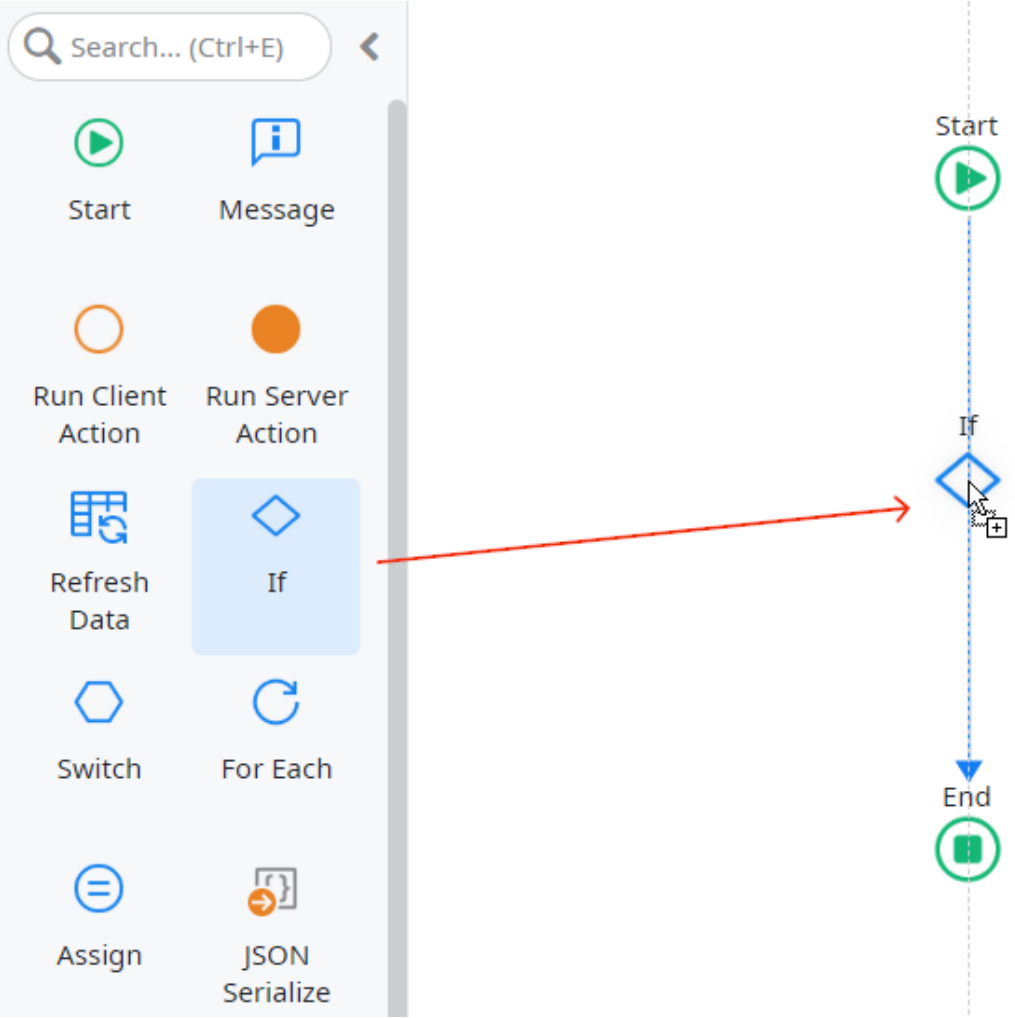


2) The Screen has a set of four buttons that call the **NegativeOrPositive** Screen Action. Each one of these buttons is associated to that screen action with a different value for the Number Input Parameter.



3) Double-click one of the buttons or the **NegativeOrPositive** Screen Action on the elements area to open it.

4) Drag an If and drop it between the Start and End.

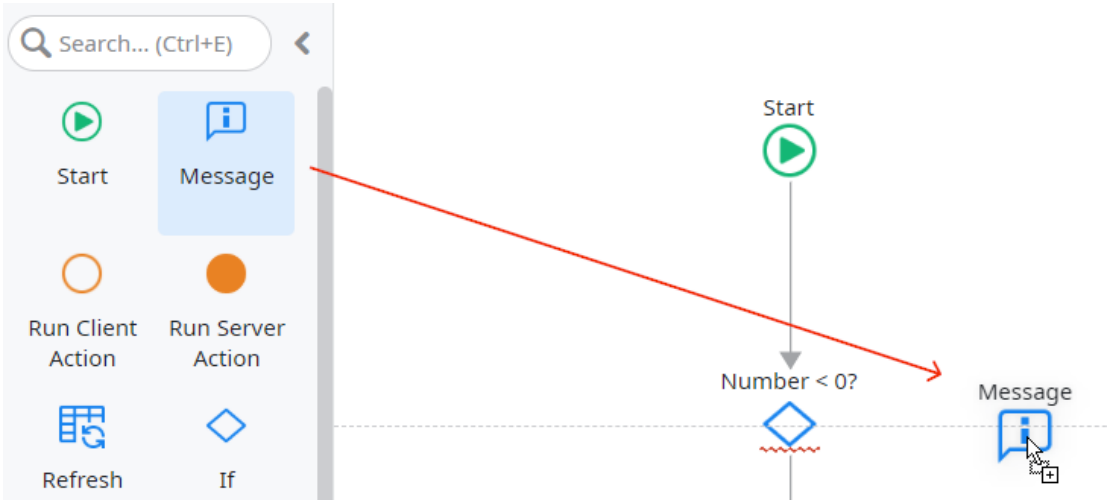


5) Double-click the If, and set its Condition to

```
Number < 0
```

6) Click **Done** to close the Expression Editor.

7) Drag a Message to the right of the If

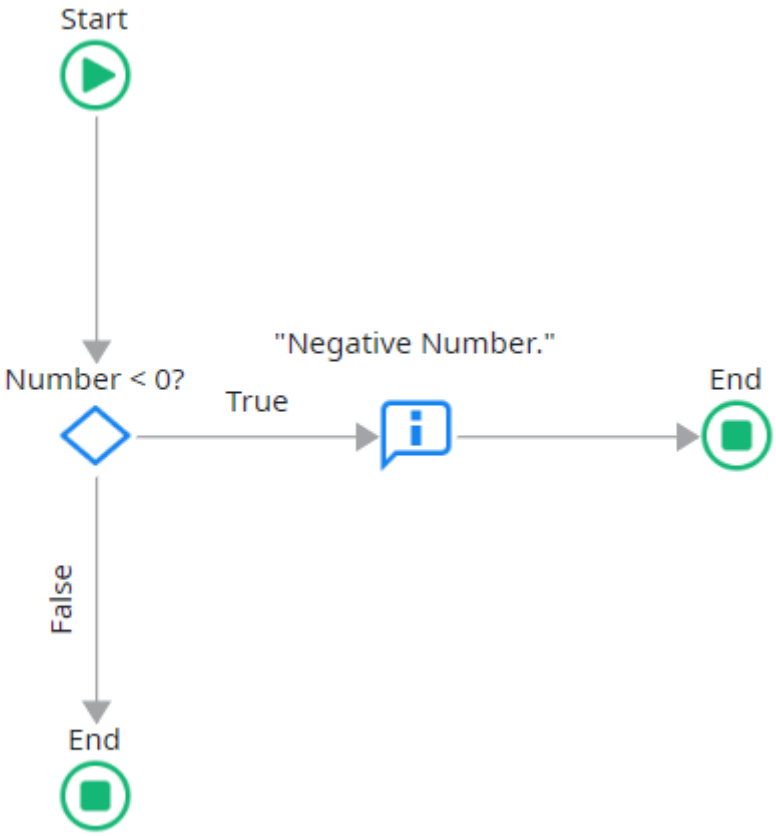


8) Set the **Message** property to

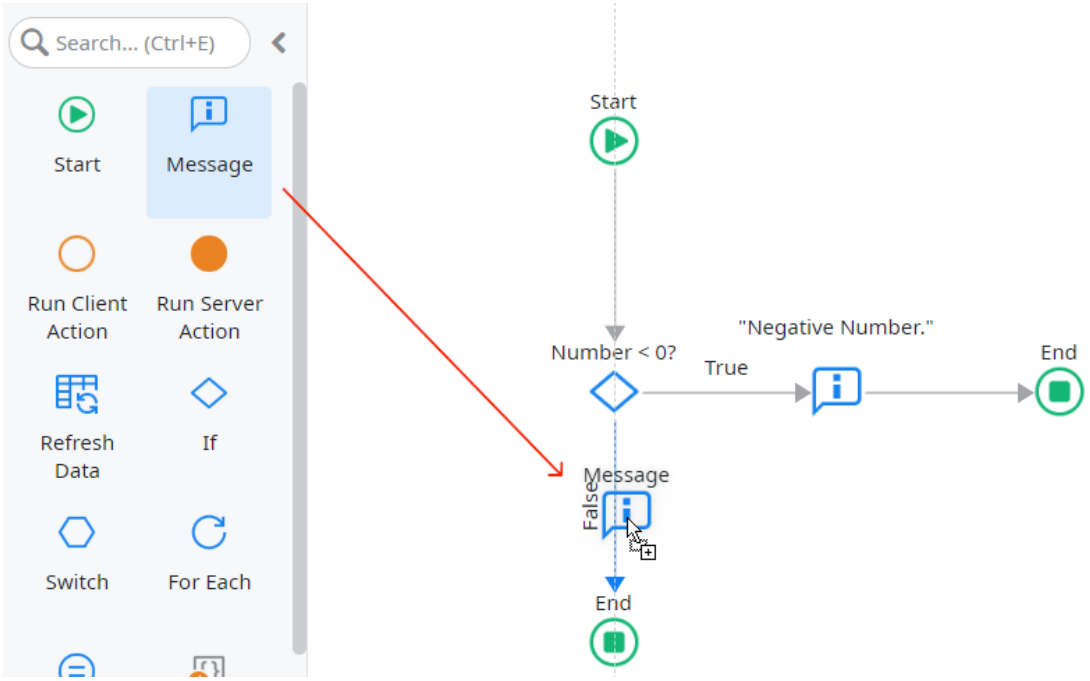
```
"Negative Number."
```

9) Create the **True** branch connector from the If to the Message.

10) Drag an **End** and drop it to the right of the Message, then create the connector between both.



11) Drag another **Message**, and drop it on the False branch of the If.



12) Set the **Message** property of the added element to

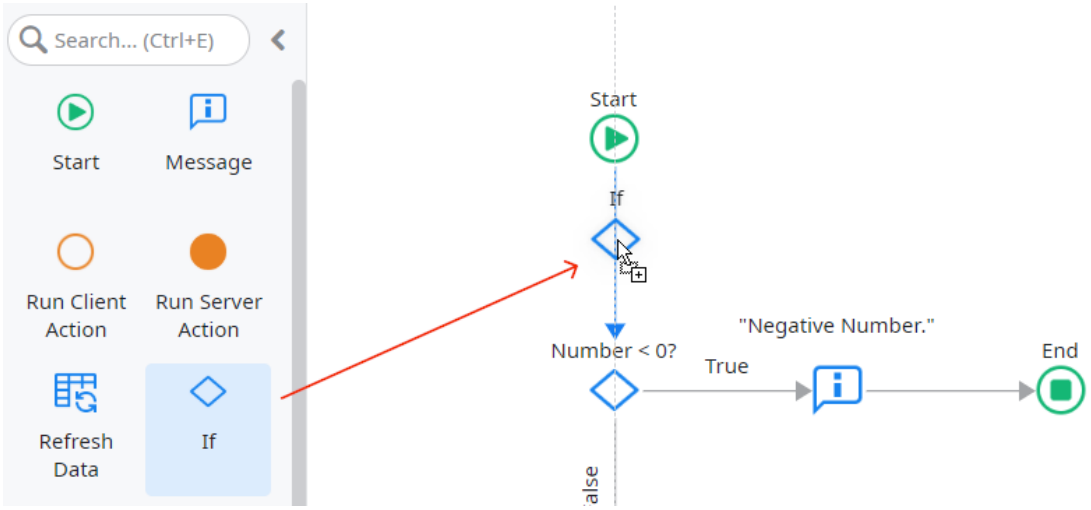
```
"Positive Number."
```

13) Publish the module, and test the four buttons inside the **Negative or Positive?** area.

14) Open the application in the browser to validate that each button shows the correct info message.

15) Notice that when the button with 0 is pressed, the Positive Number feedback message is shown.

16) Drag another **If** and drop it between the Start and the existing If.

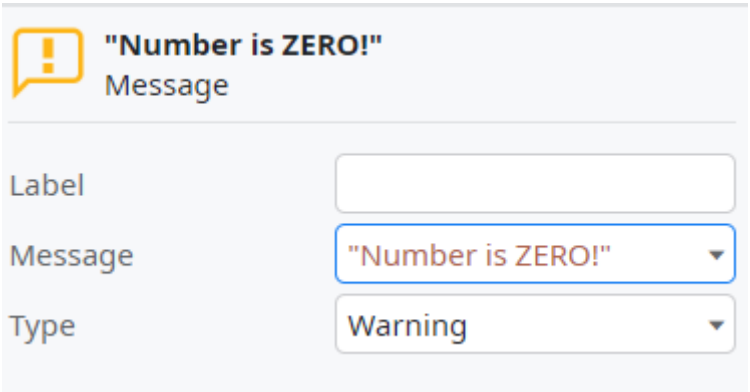


17) Set the **Condition** of the new If to

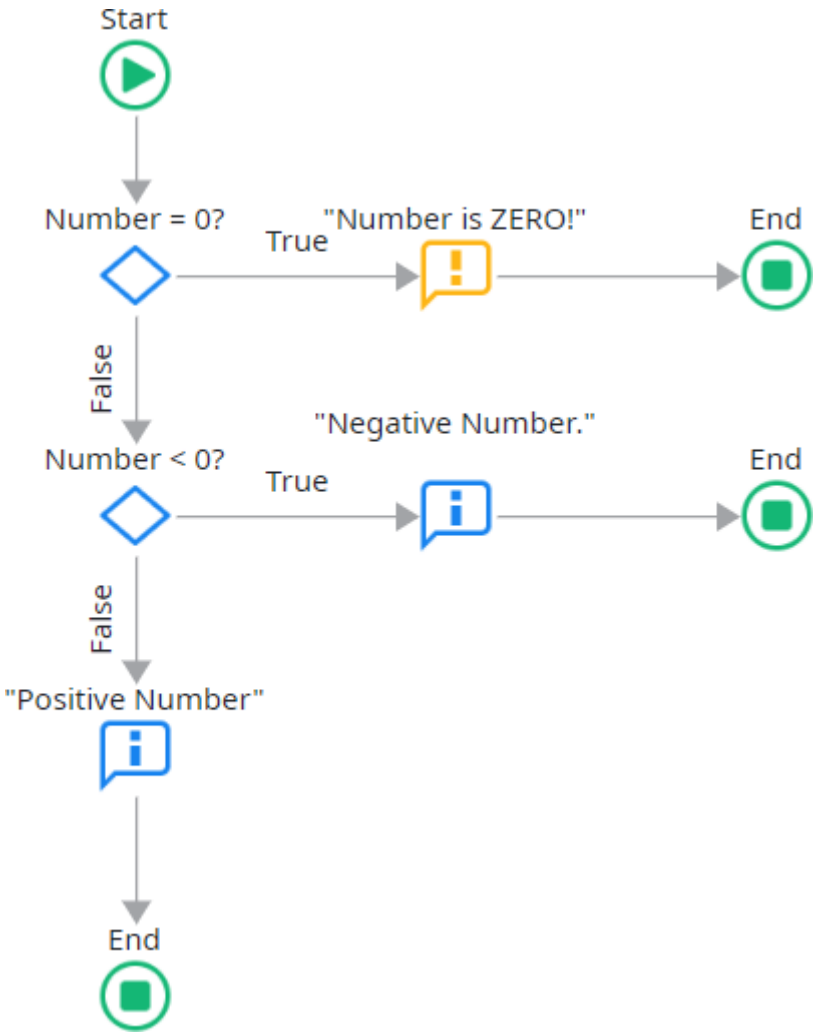
```
Number = 0
```

18) Drag another Message and drop it to the right of the newly created If, then create the **True** branch connector from the If to the Message.

19) Set the properties of the Message as follows



20) Drag an **End** node and drop it to the right of the newly created Message, then connect both. The flow should look like this

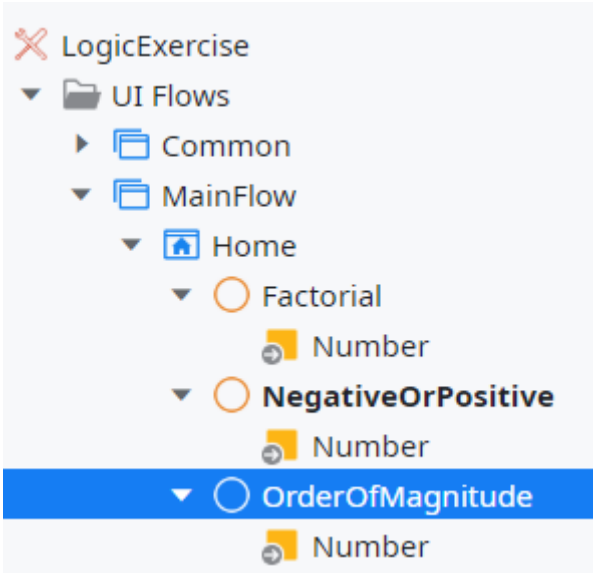


21) Publish the module and test again the existing buttons inside the **Negative or Positive?** area.

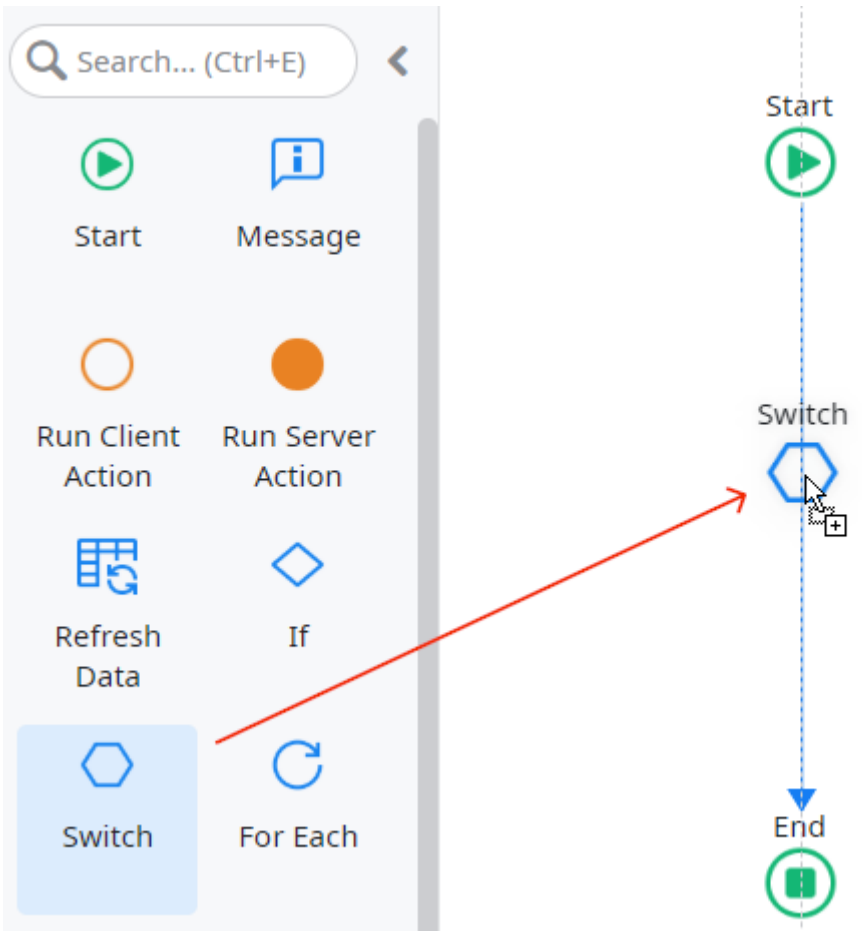
Order of Magnitude

In this section, we will implement another Screen Action. This one will analyze a given number and return either one of the following options: "<1K", "<10K", "<100K" or ">=100K".

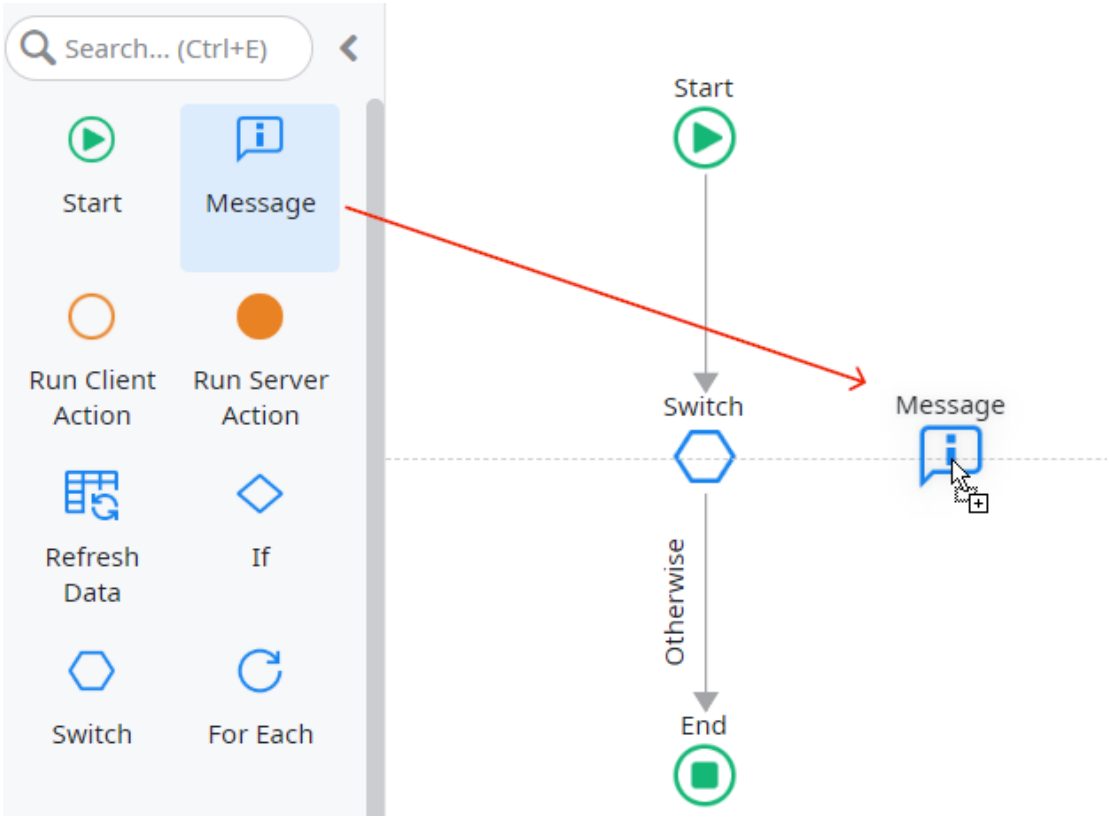
1) Double-click the **OrderOfMagnitude** Screen Action to open its flow.



2) Drag a **Switch** node and drop it between the Start and End.



3) Drag a **Message** and drop it to the right of the Switch

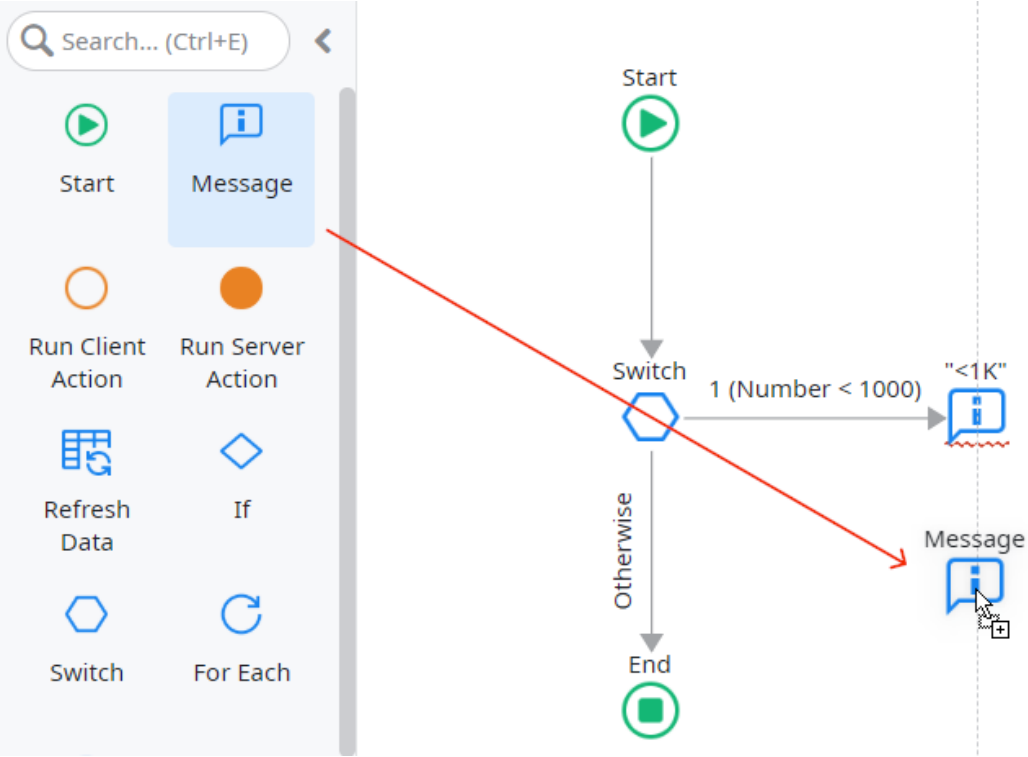


4) Set the **Message** property of the recently added element to `"<1K"`, then create a connector from the Switch to the Message.

5) Set the **Condition** of the Connector 1 created in the previous step to

```
Number < 1000
```

6) Drag another **Message** and drop it below the existing one

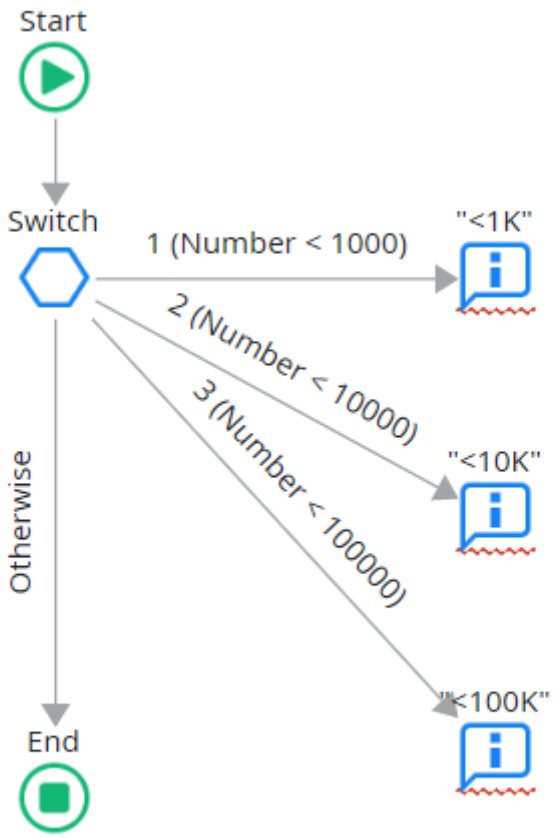


7) Set the **Message** property to `"<10K"`.

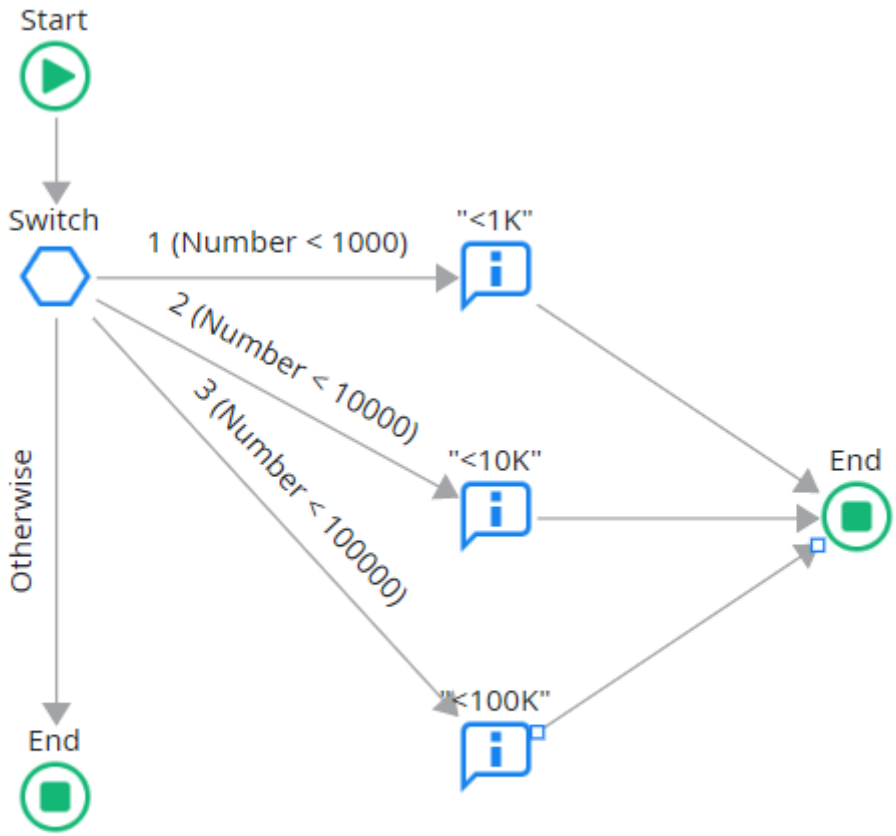
8) Create another connector from the Switch to this new Message, then set the Condition of the connector to

```
Number < 10000
```

9) Repeat the previous three steps to make your logic look like the following

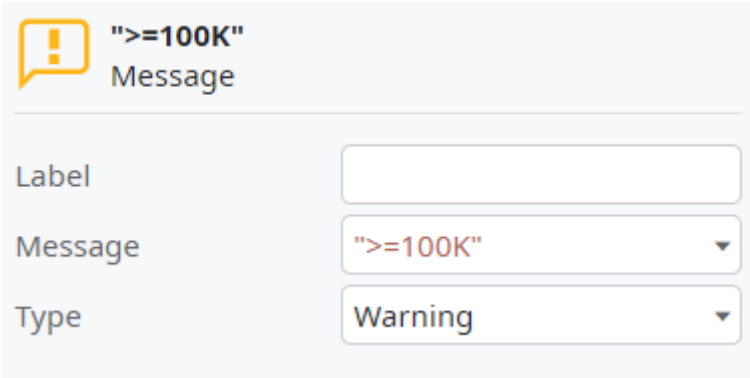


10) Drag one **End** node and drop it to the right of the flow, then connect the three existing Messages to that End node.

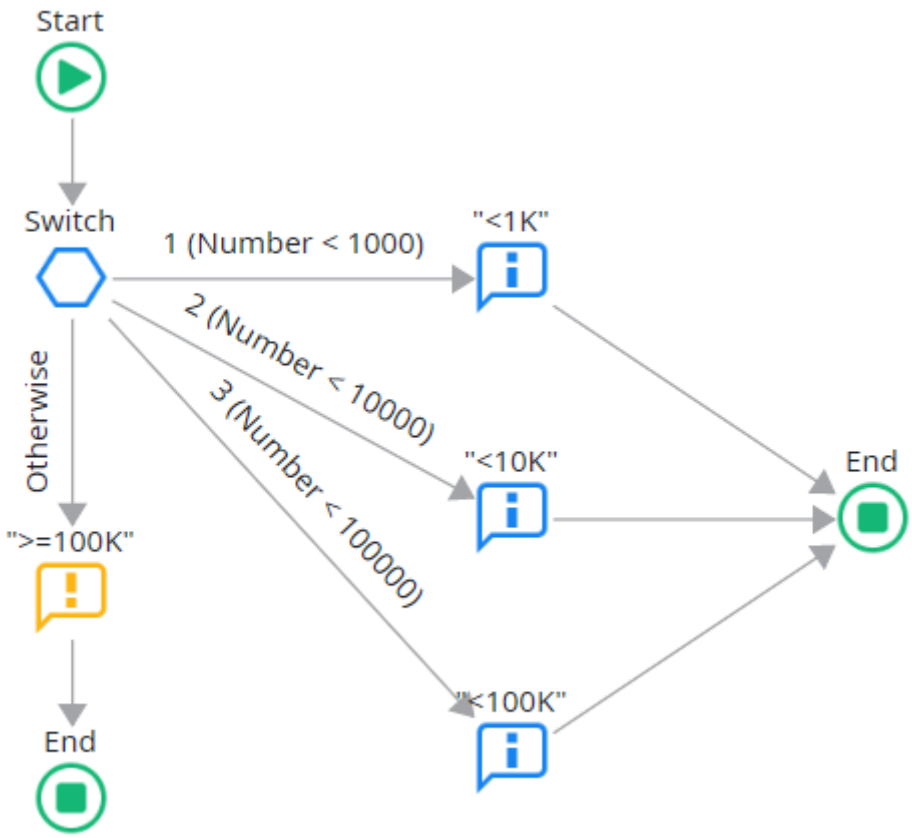


11) Drag another **Message** and drop it on the **Otherwise** branch.

12) Set the properties of the newly created Message as follows



13) The flow should look like this



14) Publish the module to save these latest changes.

15) Click the Open in Browser button and test the five existing buttons on the **Order of Magnitude** area.

Factorial

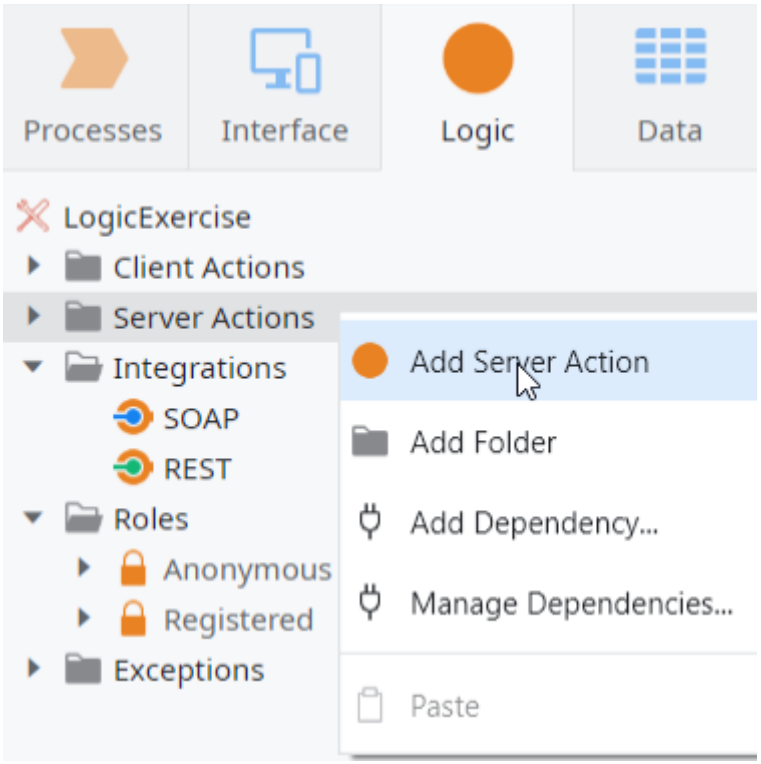
In this section, we will implement the final Screen Action logic that will calculate the factorial of a number. This action will be implemented as a Server Action, as opposed to a Client Action as before to demonstrate that it is possible to call Server Actions from inside a Client Action.

The factorial is a mathematical function that is represented as n! and essentially is the multiplication of all numbers between n and 1. Below you have a few examples of the factorial function

```
4! = 4 x 3 x 2 x 1 = 24
7! = 7 x 6 x 5 x 4 x 3 x 2 x 1 = 5040
```

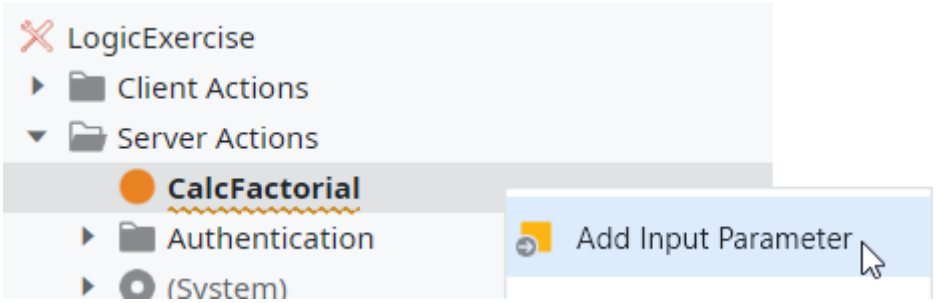
More information about the factorial function can be found in https://en.wikipedia.org/wiki/Factorial

1) Switch to the Logic tab, right-click the Server Actions folder, then select Add Server Action.

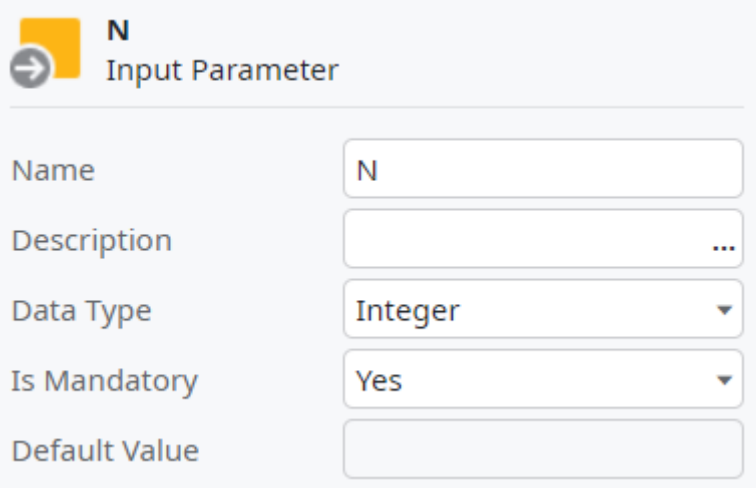


2) Set the **Name** of the Server Action to *CalcFactorial*.

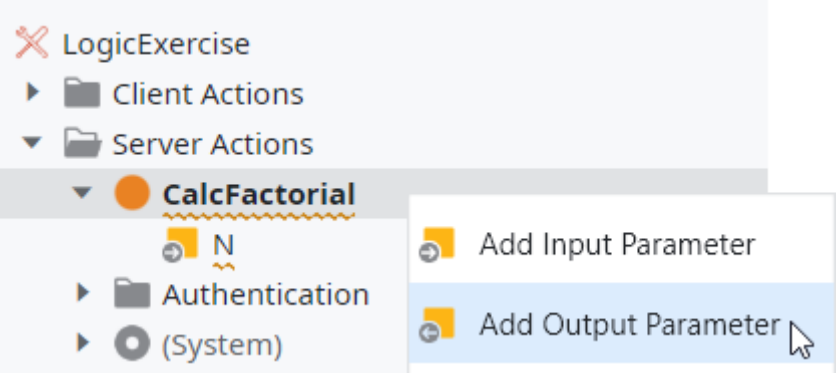
3) Right-click the CalcFactorial Server Action and select **Add Input Parameter**



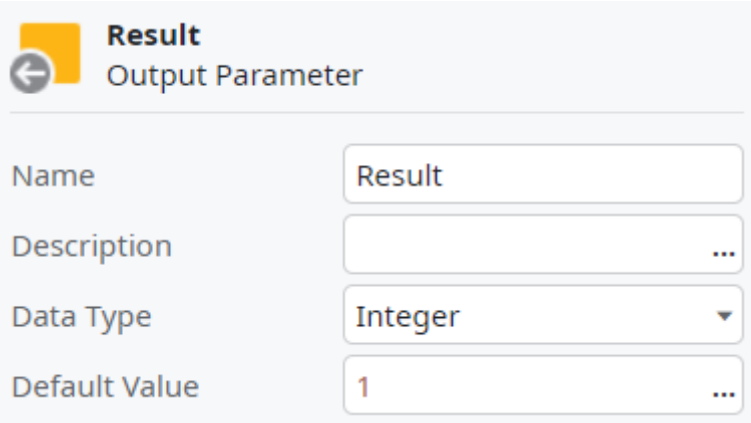
4) Set the **Name** of the Input Parameter to *N*, and make sure the **Data Type** is set to *Integer*.



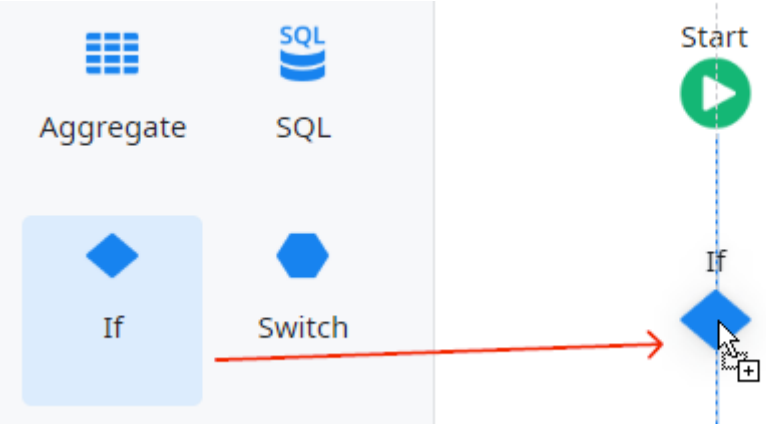
5) Right-click the CalcFactorial and select **Add Output Parameter**



6) Set the **Name** of the Output Parameter to *Result*, make sure the **Data Type** is set to *Integer*, and the **Default Value** is *1*.



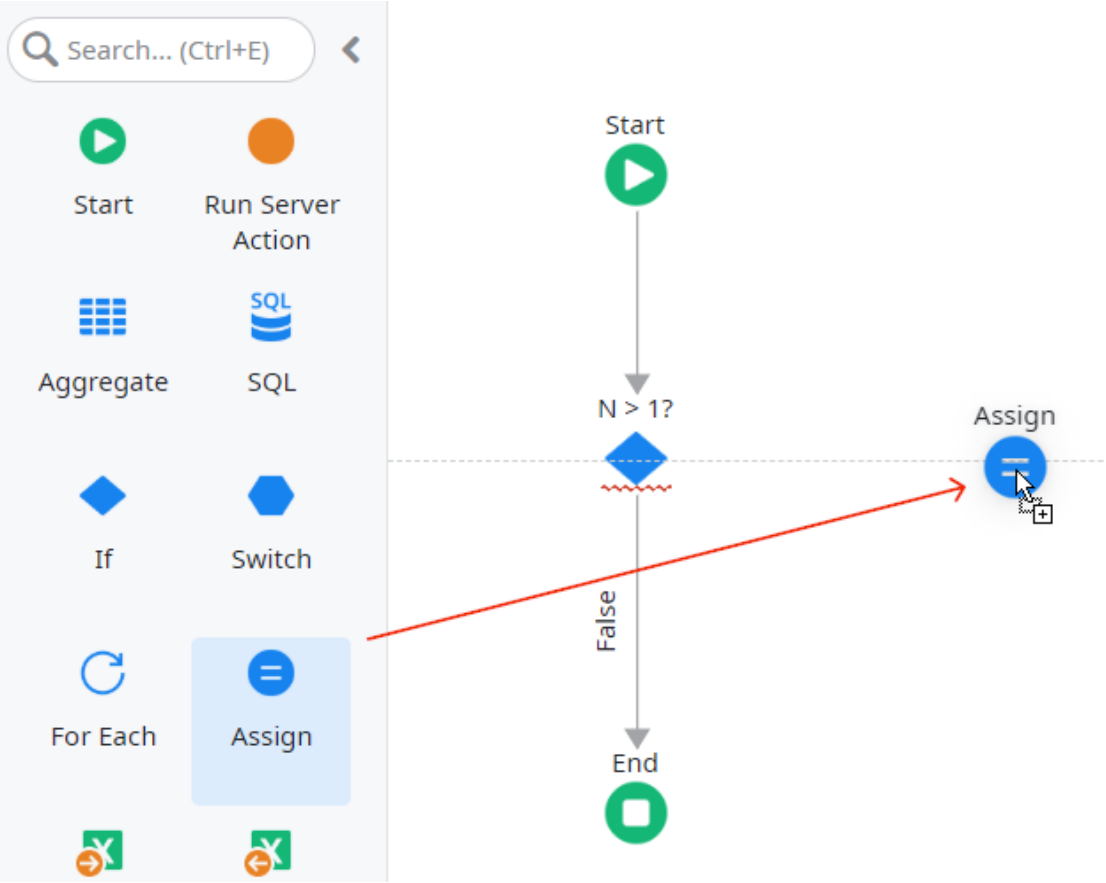
7) Drag an **If** and drop it between the Start and End



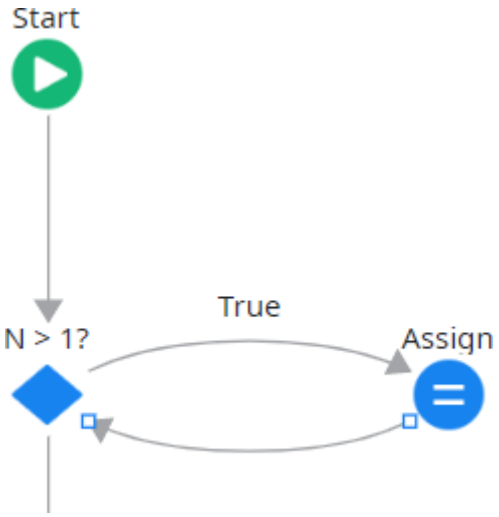
8) Set the **Condition** of the If to

```
N > 1
```

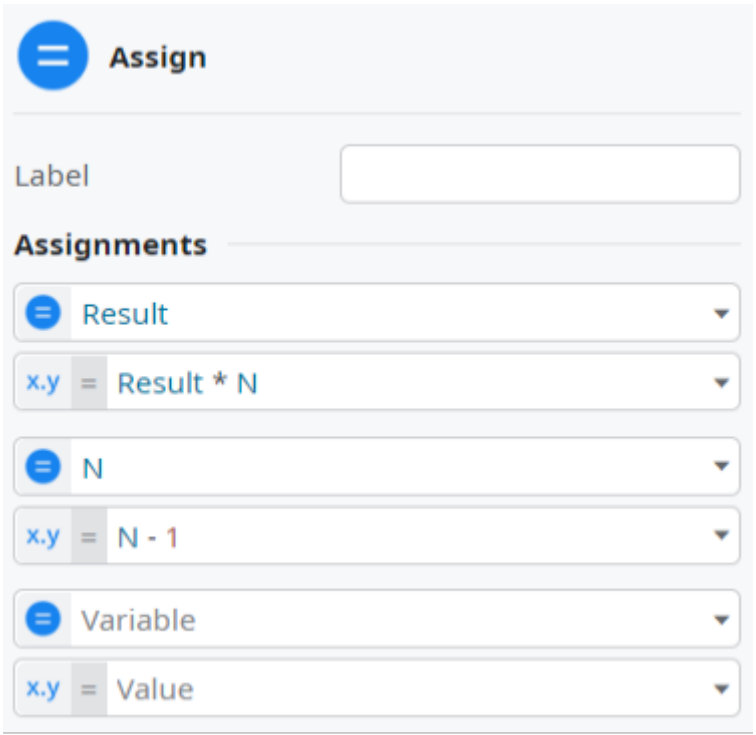
9) Drag an **Assign** and drop it to the right of the If



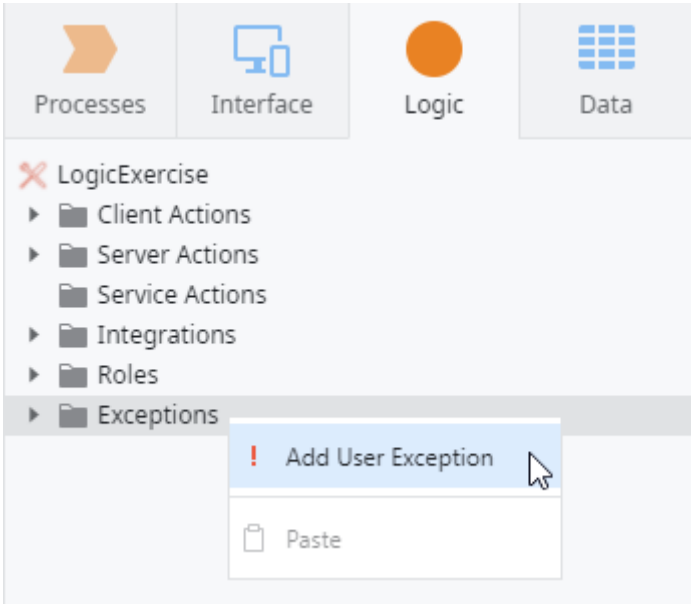
10) Create two connectors as follows



11) Define the following assignments in the Assign

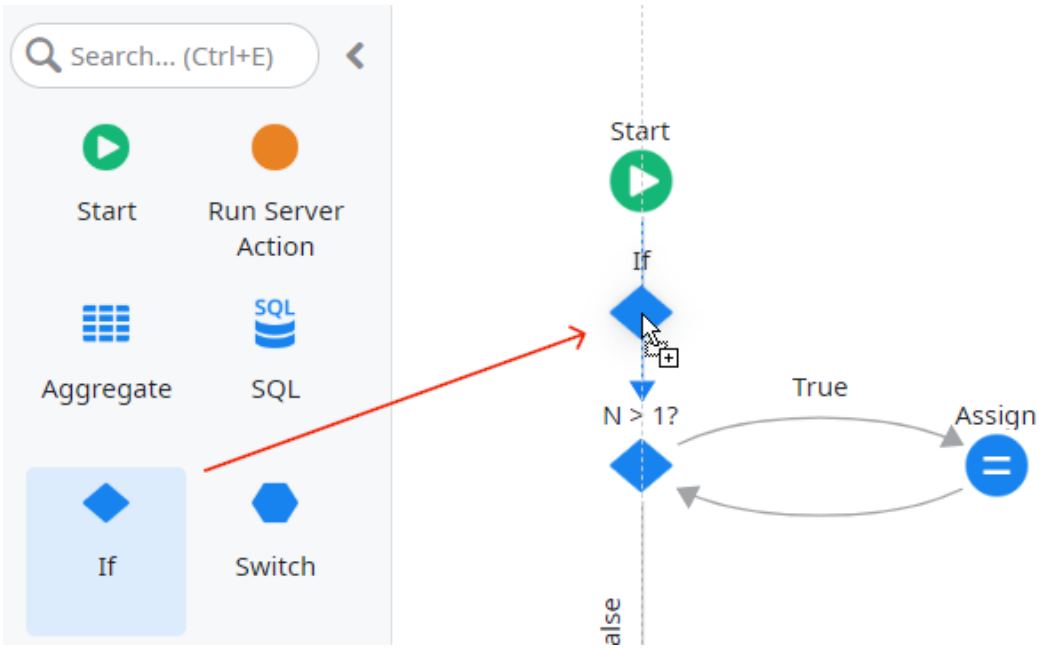


12) Right-click the Exceptions Folder and select **Add User Exception**



13) Set the Exception **Name** to *NegativeNumberException*

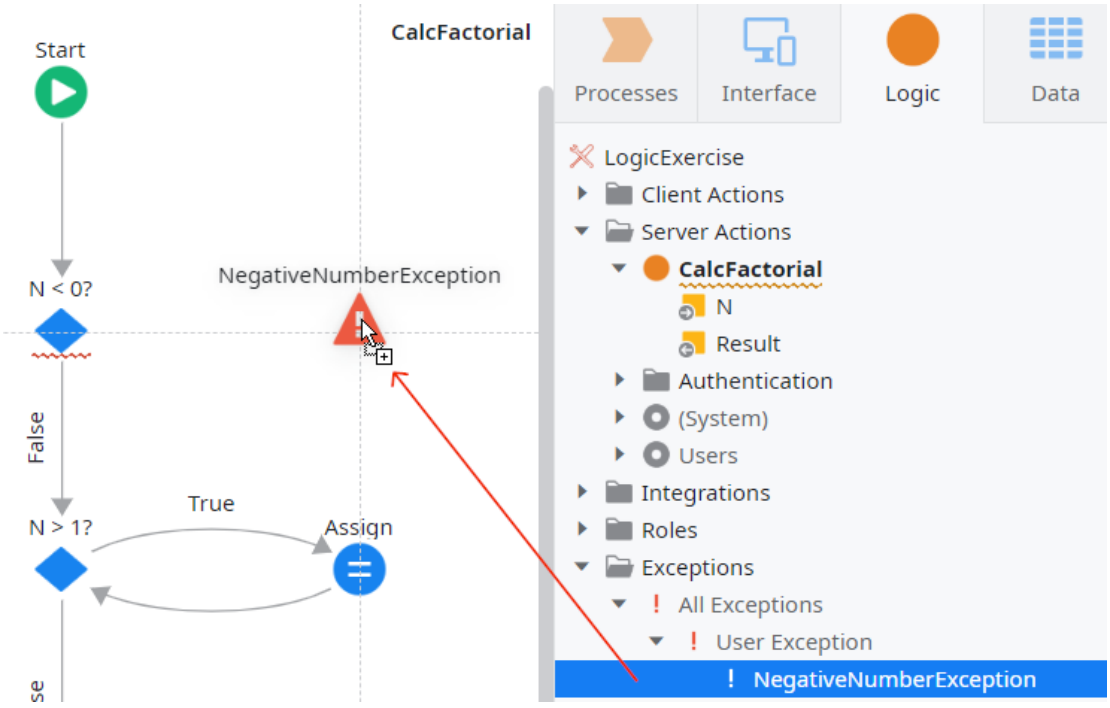
14) Drag another **If** and drop it between the Start and the existing one



15) Set the Condition of the If to

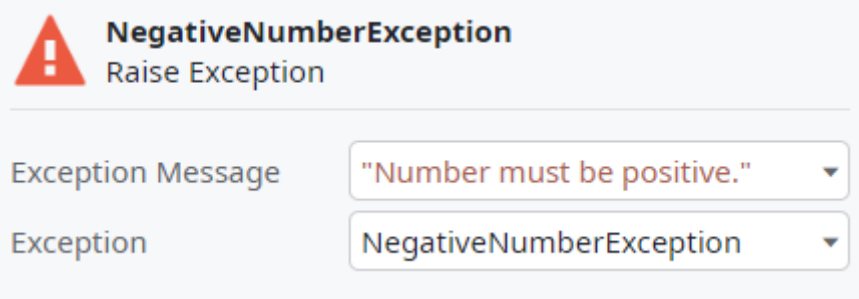
```
N < 0
```

16) Drag the previously created Exception and drop it to the right of the If created above

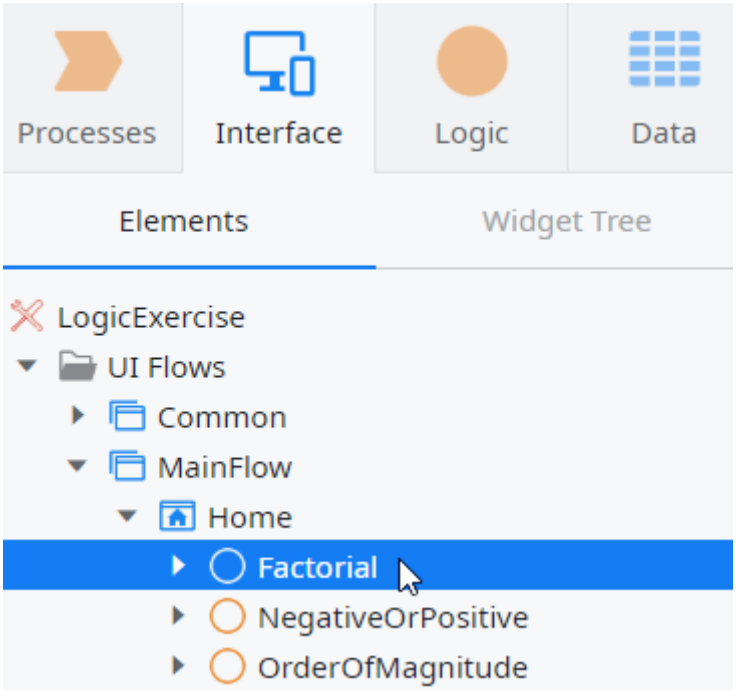


17) Create the **True** branch connector from the If to the Raise Exception element.

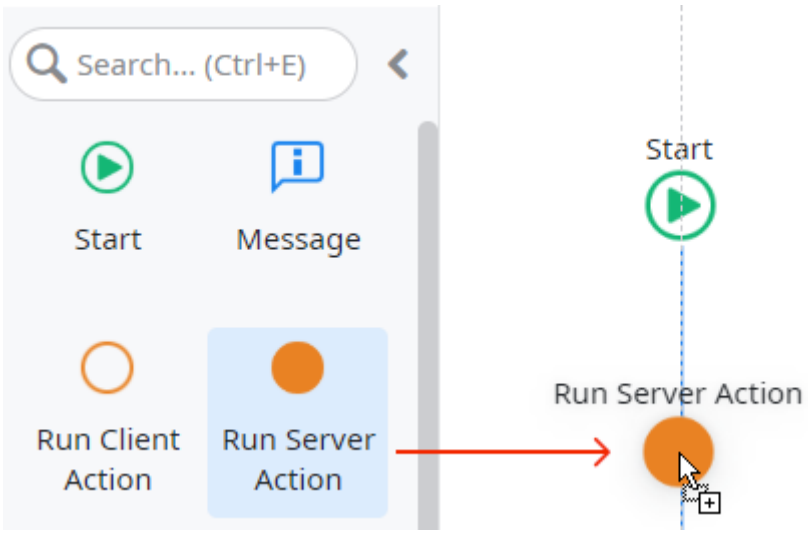
18) Select the Raise Exception element, and in its properties set the **Exception Message** to *"Number must be positive."*.



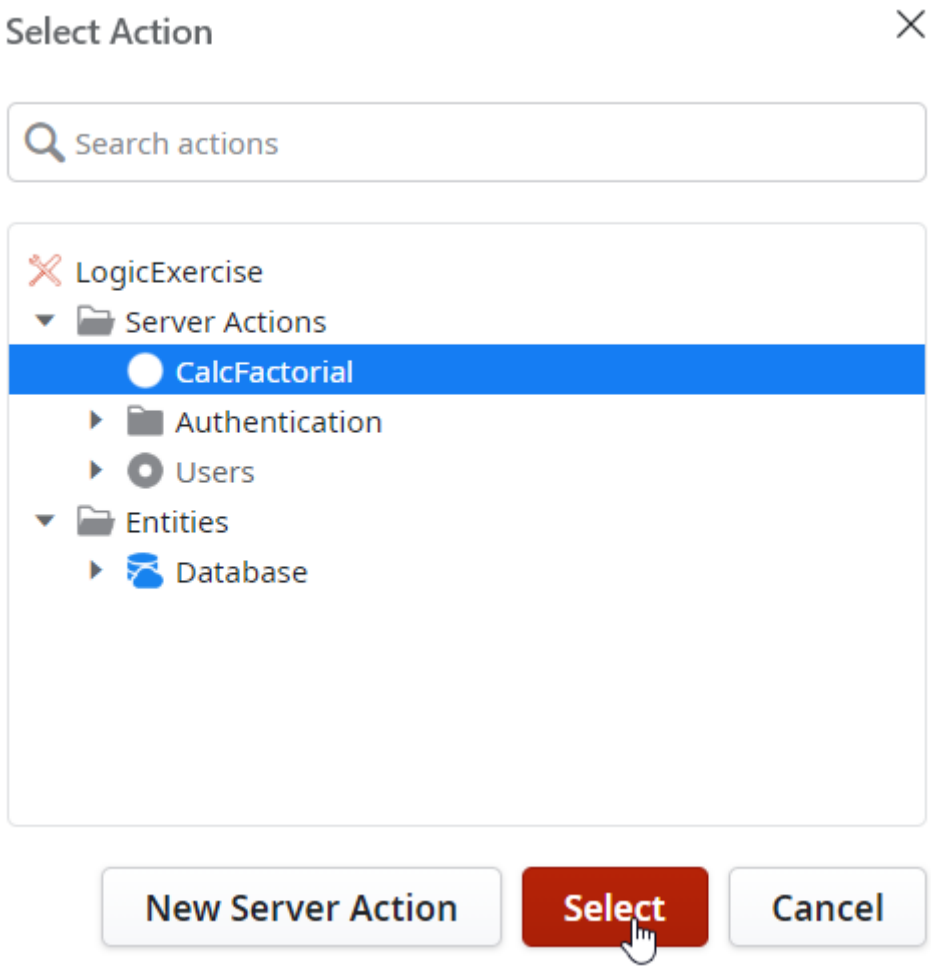
19) Return to the Interface tab, and open the **Factorial** Screen Action



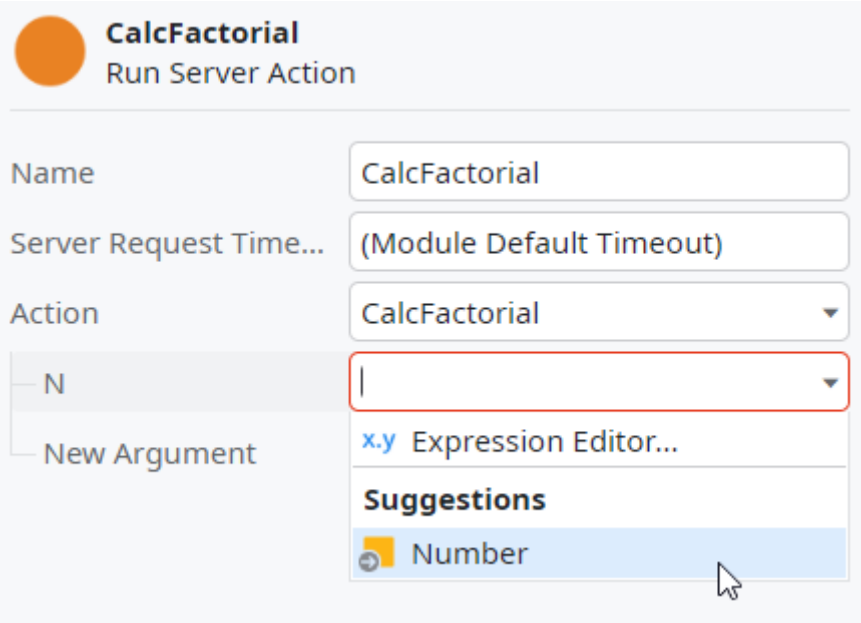
20) Drag a **Run Server Action** from the toolbox and drop it between the Start and End.



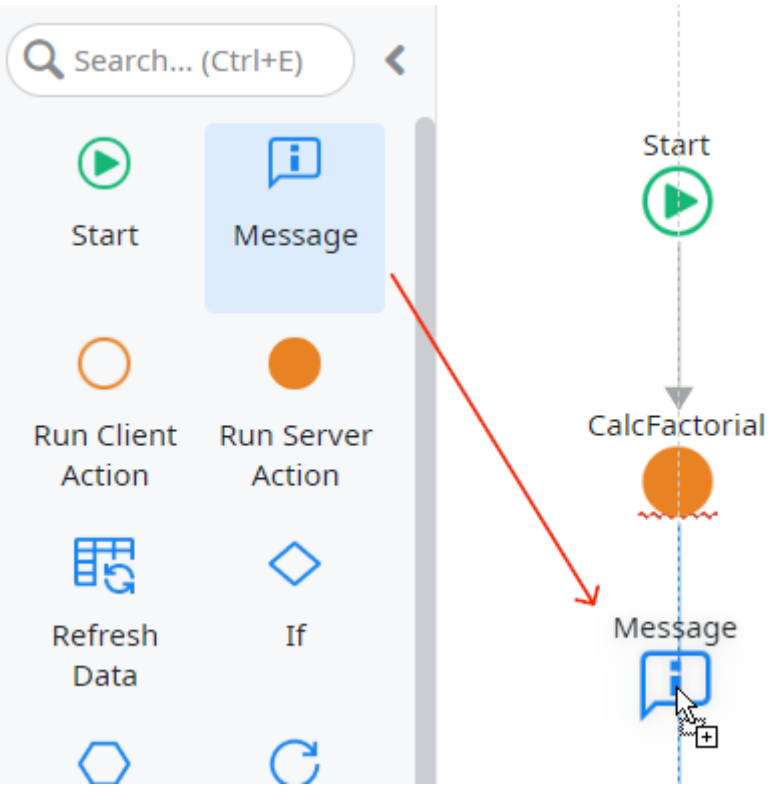
21) In the dialog, select the **CalcFactorial** Server Action that we created and implemented in previous steps, then click OK.



22) In the properties, set the **N** argument to the *Number* input parameter.



23) Drag a **Message** and drop it between the Run Server Action and the End



24) Double-click the Message, and set the expression to

```
"Factorial of " + Number + " is " + CalcFactorial.Result
```

25) Close the Expression Editor.

26) **Publish** the module, then click the **Open in Browser** button to test the application.

27) Below you can find the expected results for each number

```
1! = 1
3! = 6
8! = 40320
-5! = (Invalid Number Exception)
```