Submitting Orders

Table of Contents

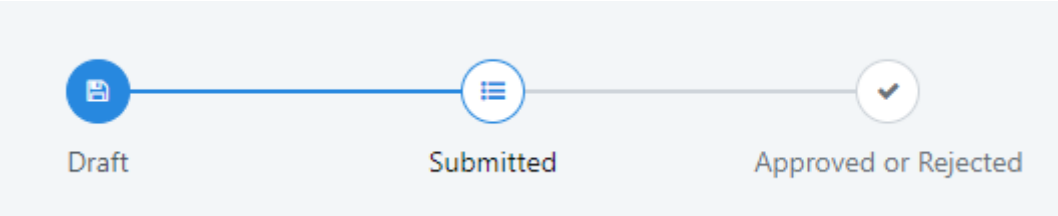
knowledge@outsystems.com

www.outsystems.com

Outline

In this tutorial you will continue to extend the Order Management application and the wizard functionality to control the statuses of an order. Specifically, you will:

- Create a new Submit Order Button to change the status of an order from Draft to Submit.



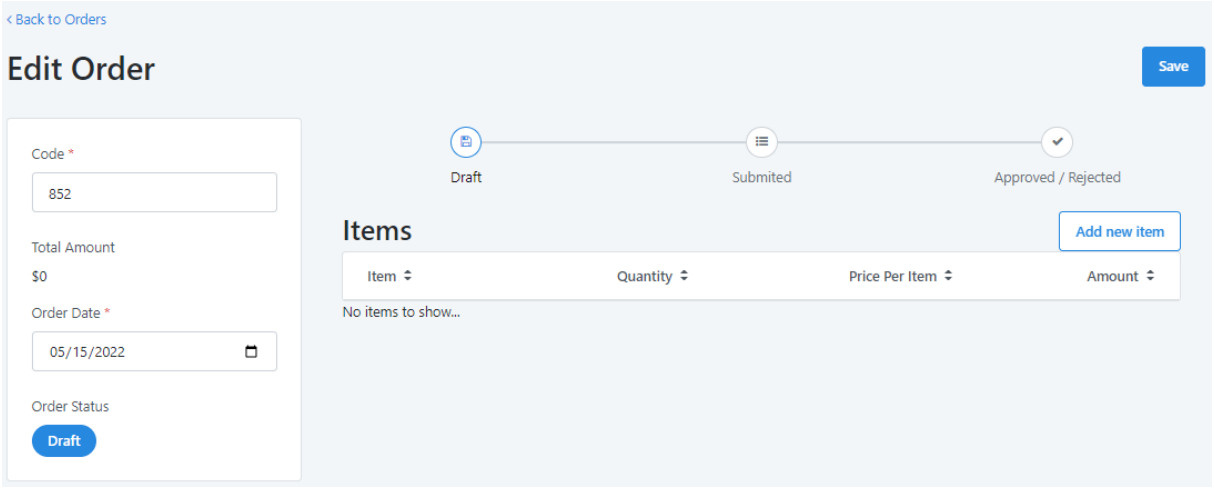
- Make sure that the code and order date Form fields are only editable when the order is being created, or with status set as Draft.
- Control the visibility of the wizard and its content, as well as the buttons, to make sure they only appear when they are useful. For instance, the Submit Order Button should only appear if the Order has the status Draft.

At the end of this tutorial, you will be able to open your app in the browser and test the new UI and Logic created for the OrderDetail Screen, even without having items in orders yet.

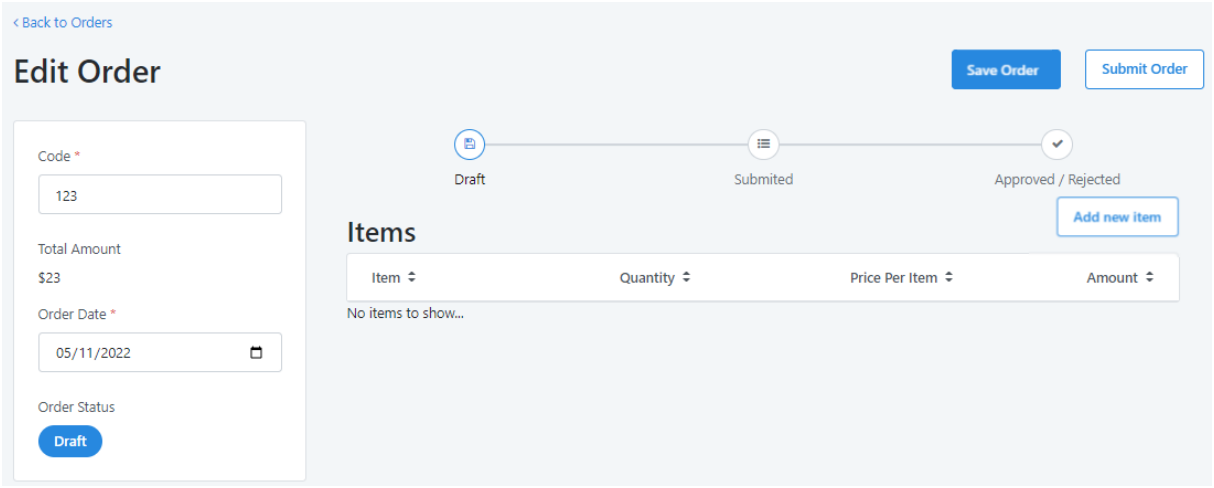
Scenario

At this point, your Order Management application is almost half-way there to get the wizard fully functional. The previous tutorials helped you create the wizard, with a step representing a status of an order.

Also, you were able to create some UI for what the user will see when the order is in Draft, which is the table of Items associated with the order (OrderItem), as well as a button to add new items to an order, which is not yet fully implemented.



In this tutorial, you will work on the Submitted status. When an order is in Draft, it will be possible to submit the order, which will change its status to Submitted. This includes adding an option to Submit the order and the logic to update the order in the database.



Also, you will need to create some logic to make sure the user does not have all the controls available at any time. So, you will define some rules for when the order is being created, when the order is in draft, and finally when the order is in the Submitted status. The new requirements are:

- A user should **only** be able to edit the order code and date when the order is being created or is in draft.
- A user should **only** see the Save Button when the order is being created or is in draft.
- A user should **only** see the Submit Order Button if the order is in draft.

- A user should **only** see the wizard and the table of items associated with an order when the order is already created.
- A user should **only** be able to add items to an order when the order is in draft.

In this tutorial, you will edit the OrderDetail Screen to implement this behavior. At the end, you will have the wizard transitioning from the Draft to the Submitted status.

How-to

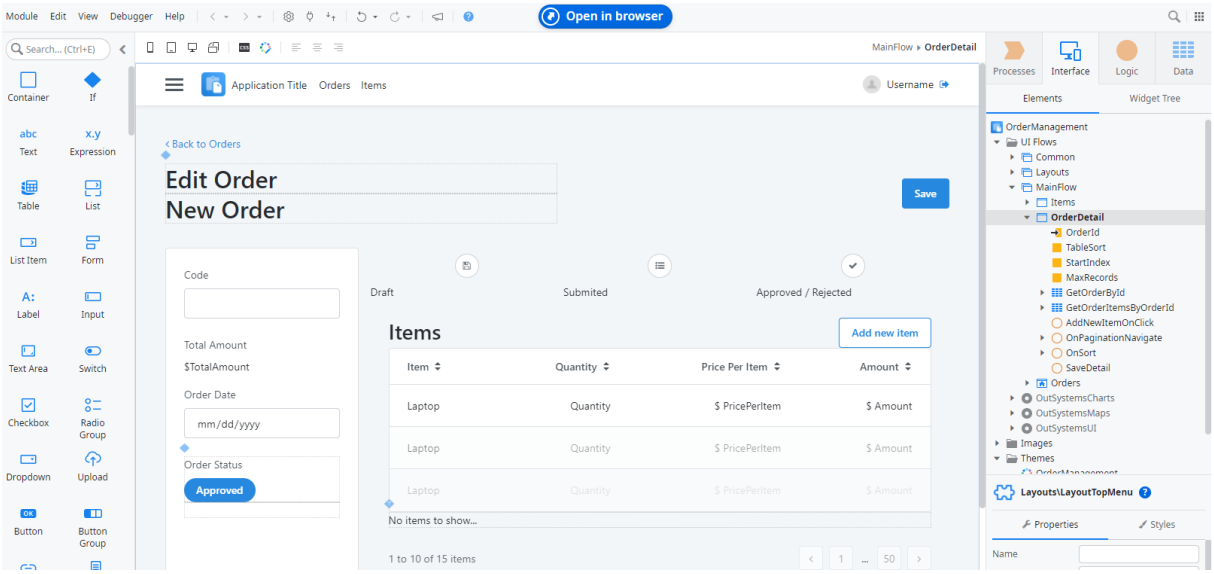
In this section, we'll show a thorough step-by-step description of how to implement the scenario described in the previous section.

Getting Started

In this tutorial we are assuming that you have already followed the previous tutorials, and have the Order Management application with the Orders, OrderDetail and Items Screens ready.

If you haven't created it yet, it is important to go back to the previous tutorials and create the application.

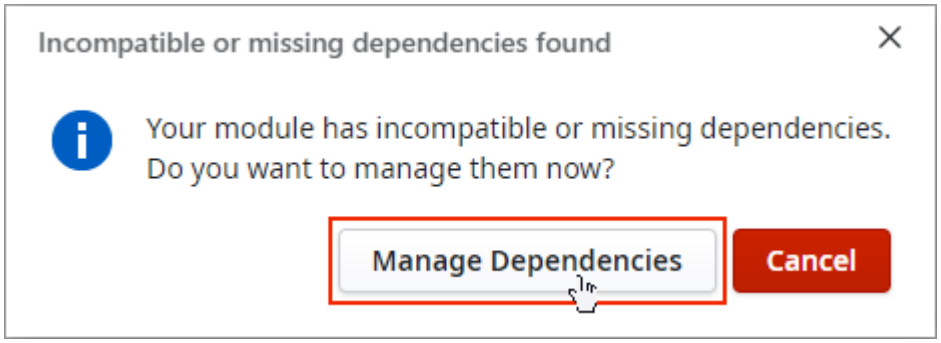
To start this exercise, you need the Service Studio with the module OrderManagement opened. You should see the Screen below with the source of your application.



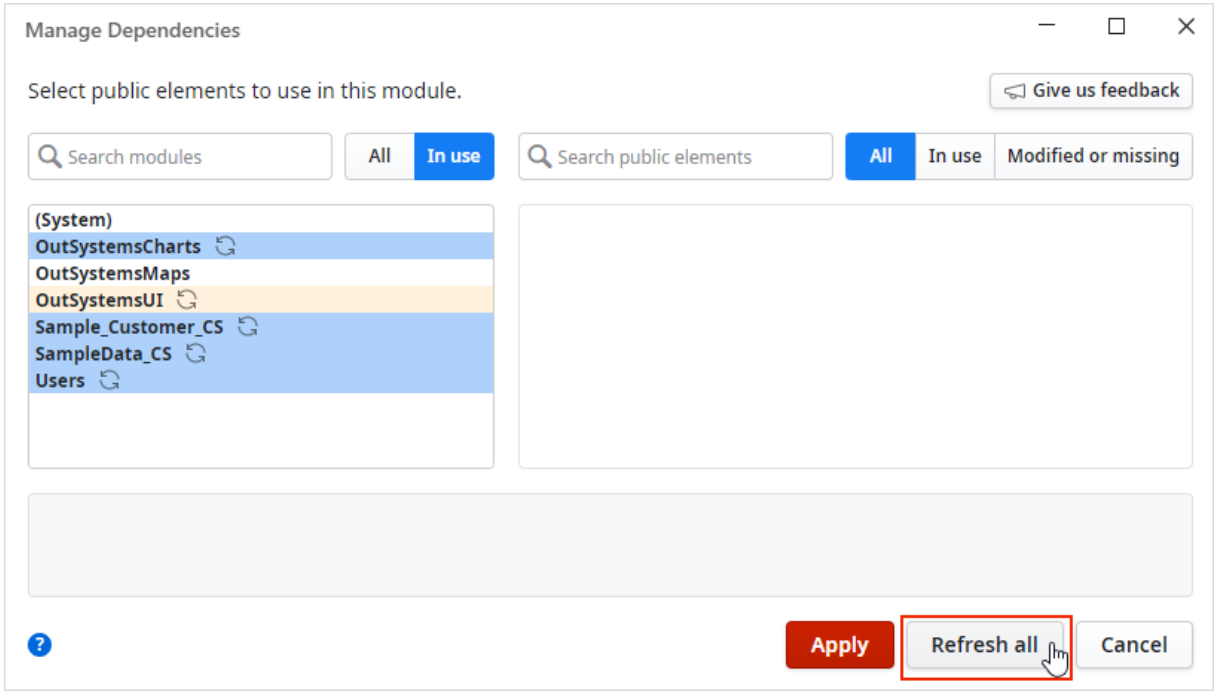
Outdated Dependencies

You might get a popup message informing that you have outdated dependencies. This is completely normal, since we are always trying to bring a new and updated version of our components!

If that happens, simply click on the button that says *"Manage Dependencies"* to see the outdated components.



Then, click on *"Refresh all"* to update everything at once and *"Apply"* when you are done.



Now publish the module to update the project

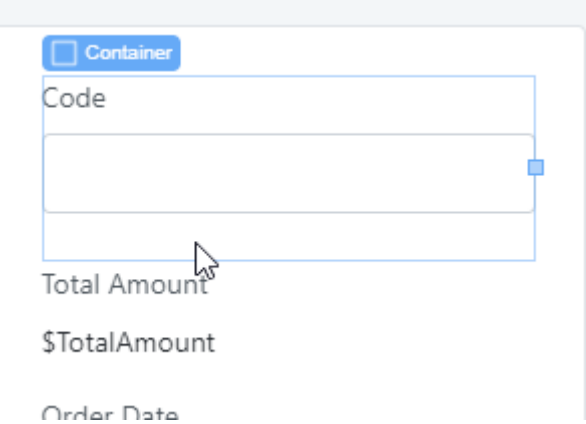
Making Fields not Editable

Let's start by working on the Form fields that should not be editable, when the order was already submitted. The first one will be the order code.

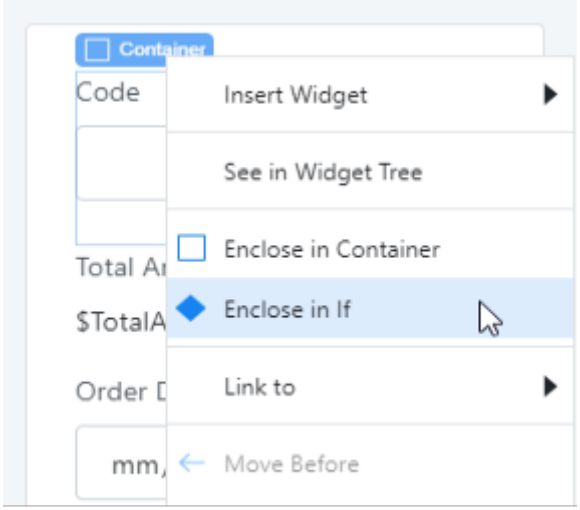
Disabling the Order Code

Make sure you have the OrderDetail Screen open. Ready? Let's do it!

1) Click on the **Container** that's enclosing the Code Label and Input inside the Form.

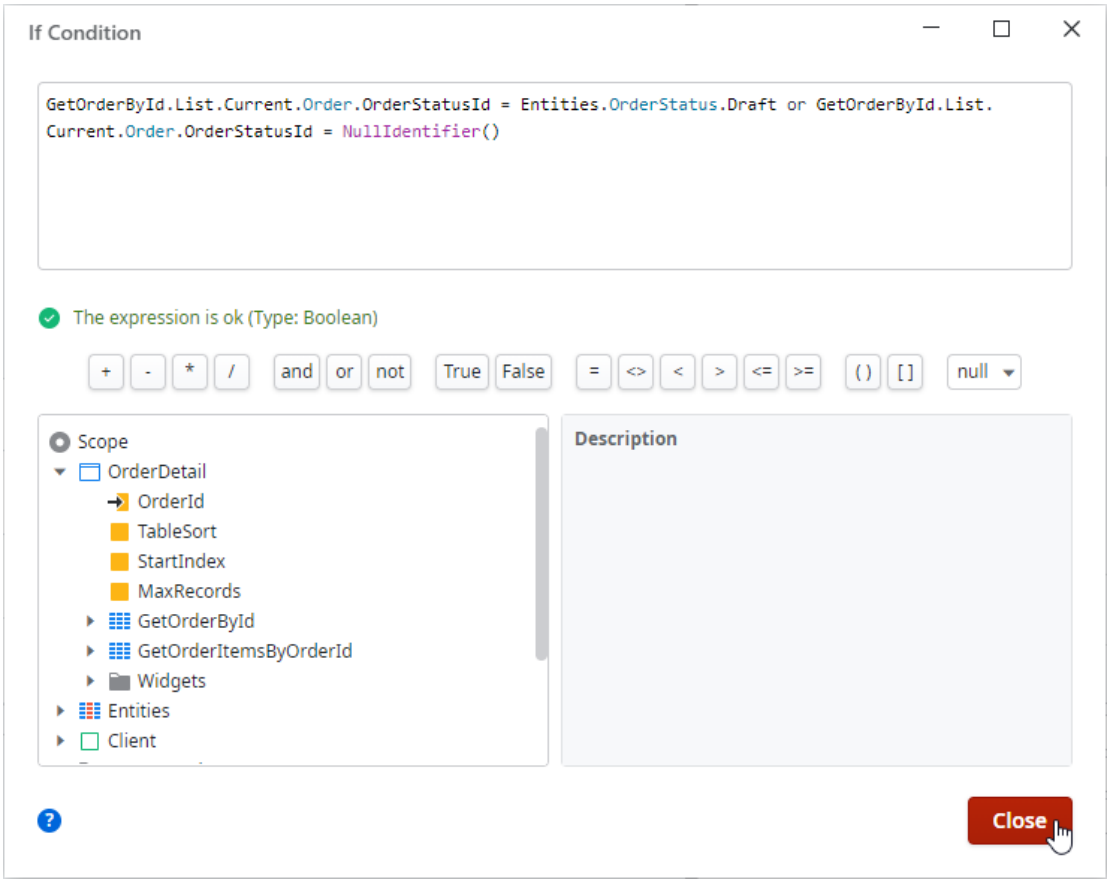


2) Right-click the **Container** and select **Enclose in If**.



3) On the right sidebar, set the If **Condition** to:

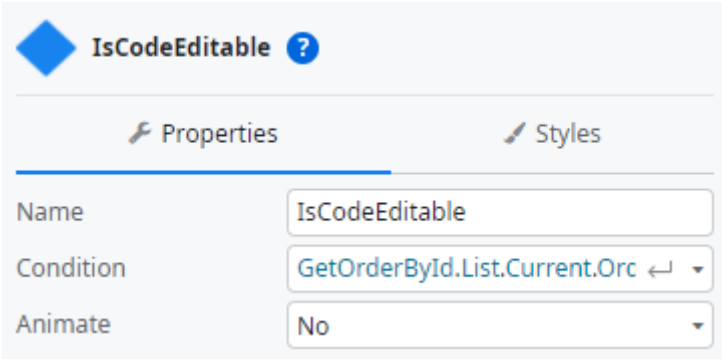
```
GetOrderById.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Draft or
GetOrderById.List.Current.Order.OrderStatusId = NullIdentifier()
```



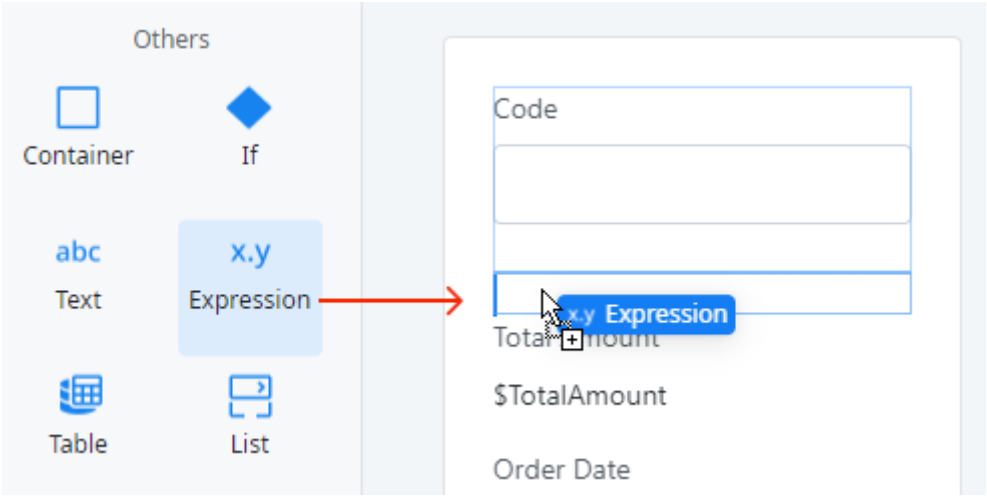
An If element has two "areas" that we call branches. The True branch is where you will add the UI and behavior that you want to have when the condition of the If is met. The False branch is the same, but for when the condition is not met.

This condition evaluates if the order is a draft or a new one. If the condition is True, the Code can be modified. If not, the Code should be visible, but not editable. You don't need to change anything in the True branch. But you will need to add the Code in a non editable format in the False branch.

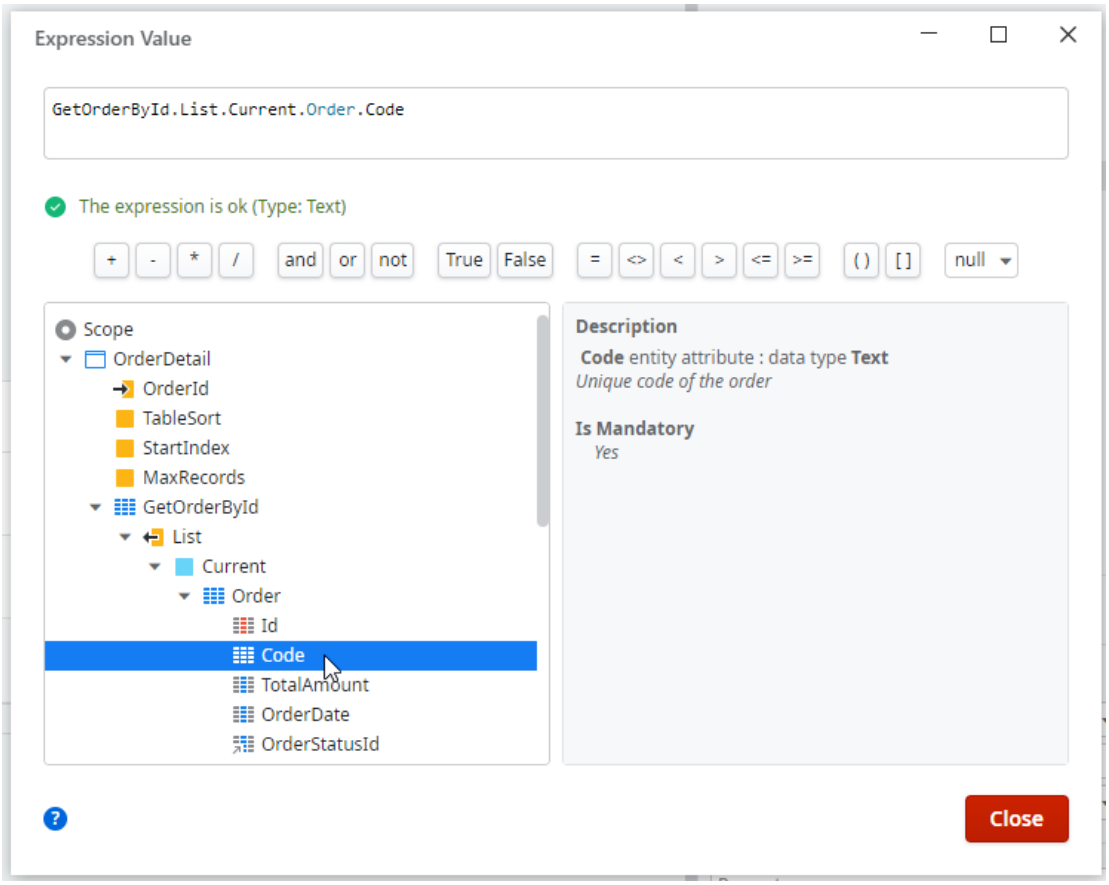
4) Type *IsCodeEditable* in the **Name** property of the If.



5) Drag an **Expression** from the Toolbox (left sidebar) and drop it on the False branch.



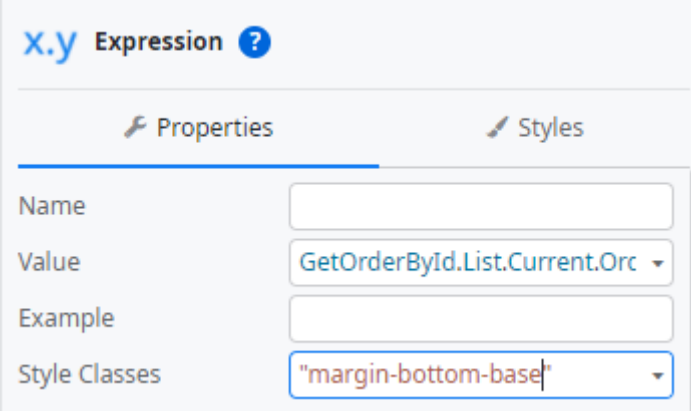
6) In the new dialog, set the expression to be:
`GetOrderById.List.Current.Order.Code`.



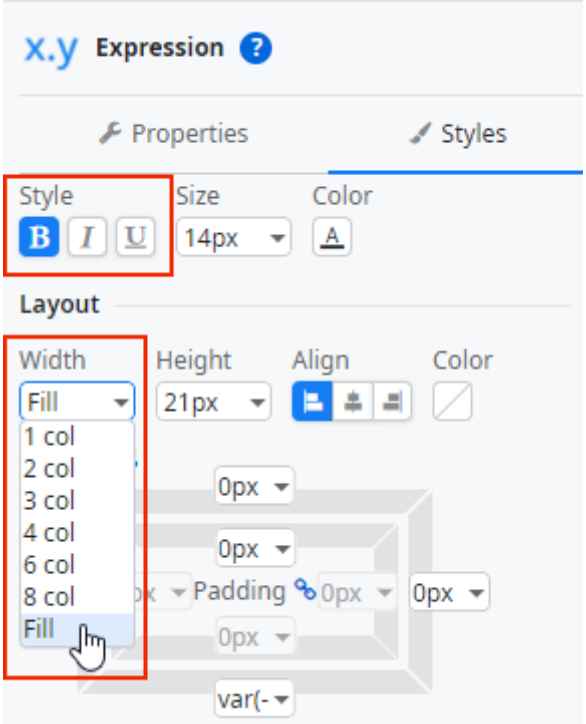
If the condition is false, then the expression with the code (and not the input) will appear.

Let's now make the code appear in bold and with a margin to the next fields in the Form.

7) With the Expression still selected, look at its properties on the right sidebar. Set the **Style Classes** property to *"margin-bottom-base"* to add a margin to the bottom of the expression.



8) Switch to the Styles tab. Set **Font style** to *Bold* and the **Layout Width** to *Fill*.

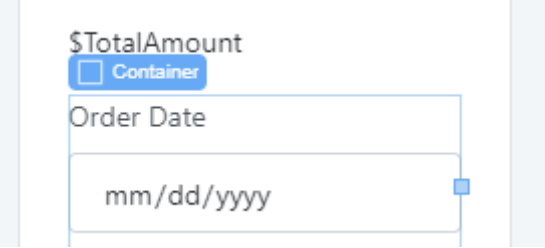


When you are editing the Order Detail Screen in Service Studio you can see both branches. However, a user will only see one of the options in the browser.

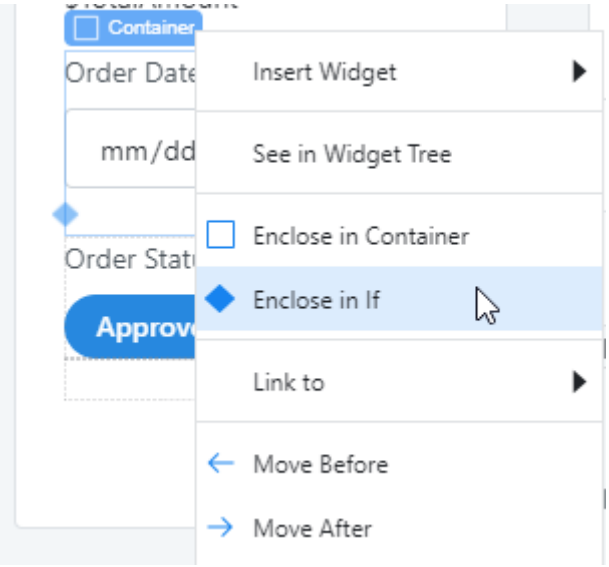
Order Date Fields

Let's focus now on the Order Date field, which will only be editable if the Order is new or a draft. The steps will be very similar to the ones made for the Code.

1) Select the **Container** enclosing the Order Date Label and Input field.

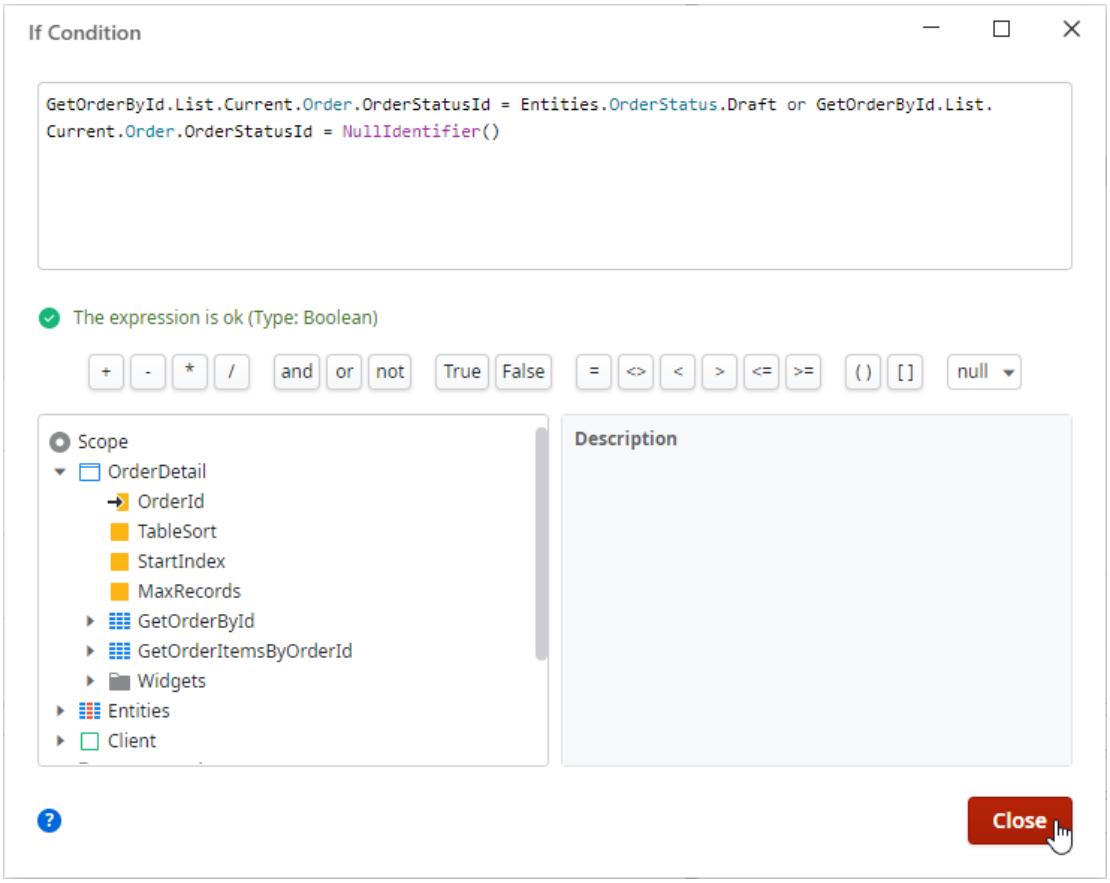


2) Right-click the **Container** and select **Enclose in If**.



3) On the right sidebar, set the If **Condition** to:

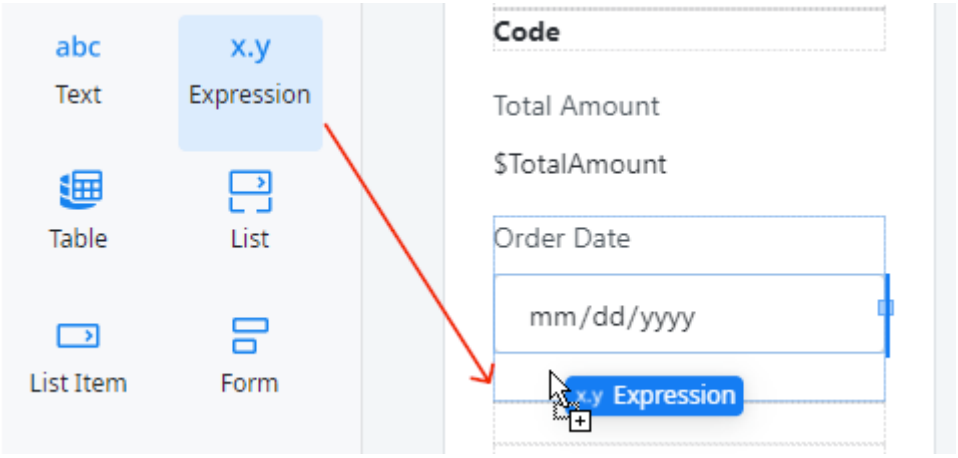
```
GetOrderById.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Draft or
GetOrderById.List.Current.Order.OrderStatusId = NullIdentifier()
```



This is just like we did for the code.

4) Type *IsDateEditable* in the **Name** property of the If.

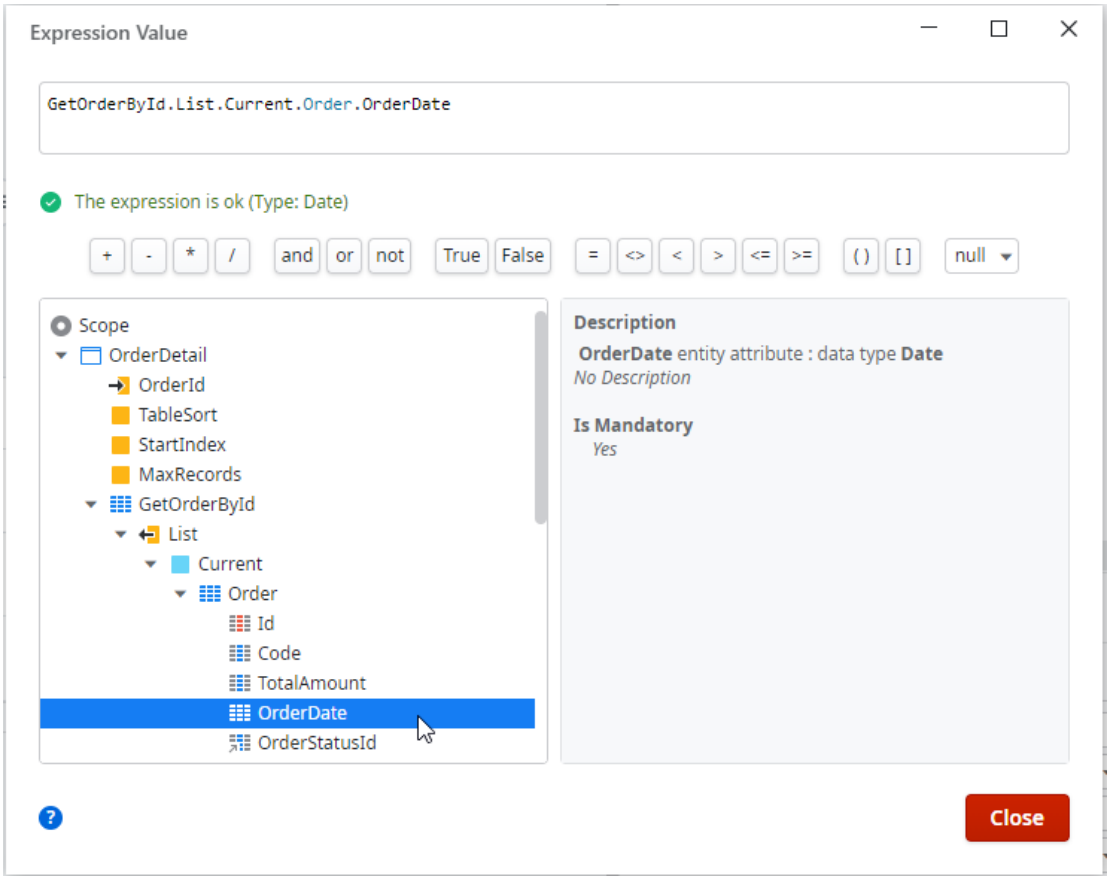
5) Drag an **Expression** from the Toolbox and drop it on the False branch.



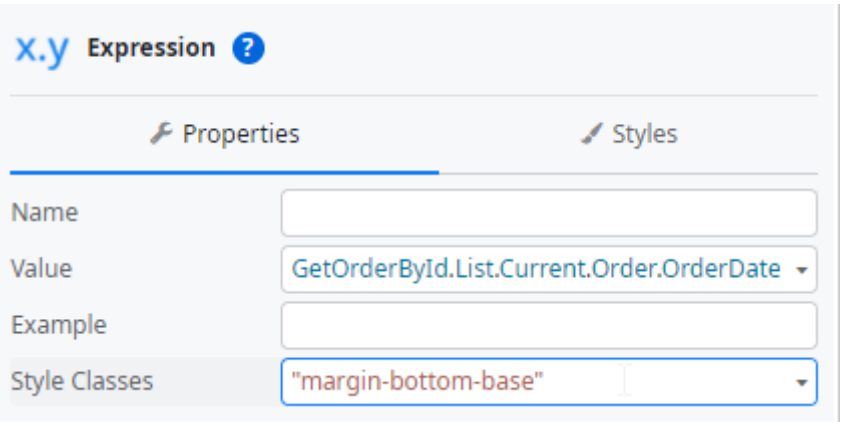
Note: If you drag an element to the wrong part of the Screen, you can simply move it to the correct place in the Widget Tree.

6) In the new dialog, set the expression to be:

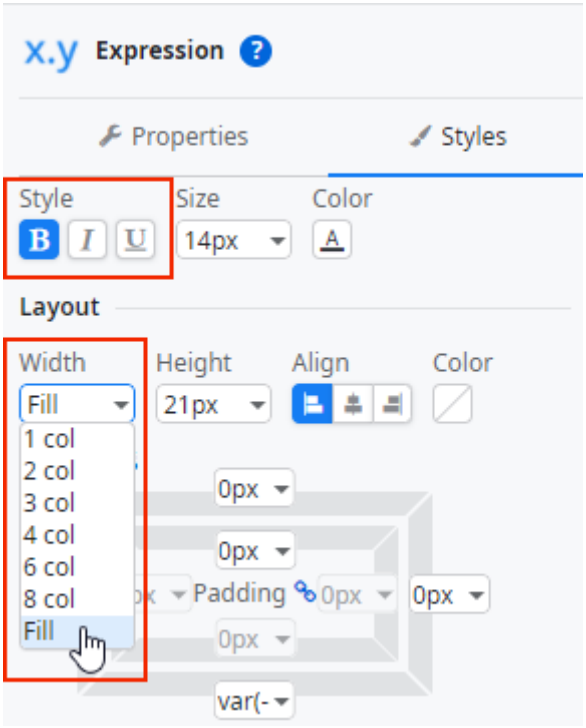
```
GetOrderById.List.Current.Order.OrderDate
```



7) With the Expression still selected, look at its properties on the right sidebar. Set the **Style Classes** property to *"margin-bottom-base"* to add a margin to the bottom of the expression.



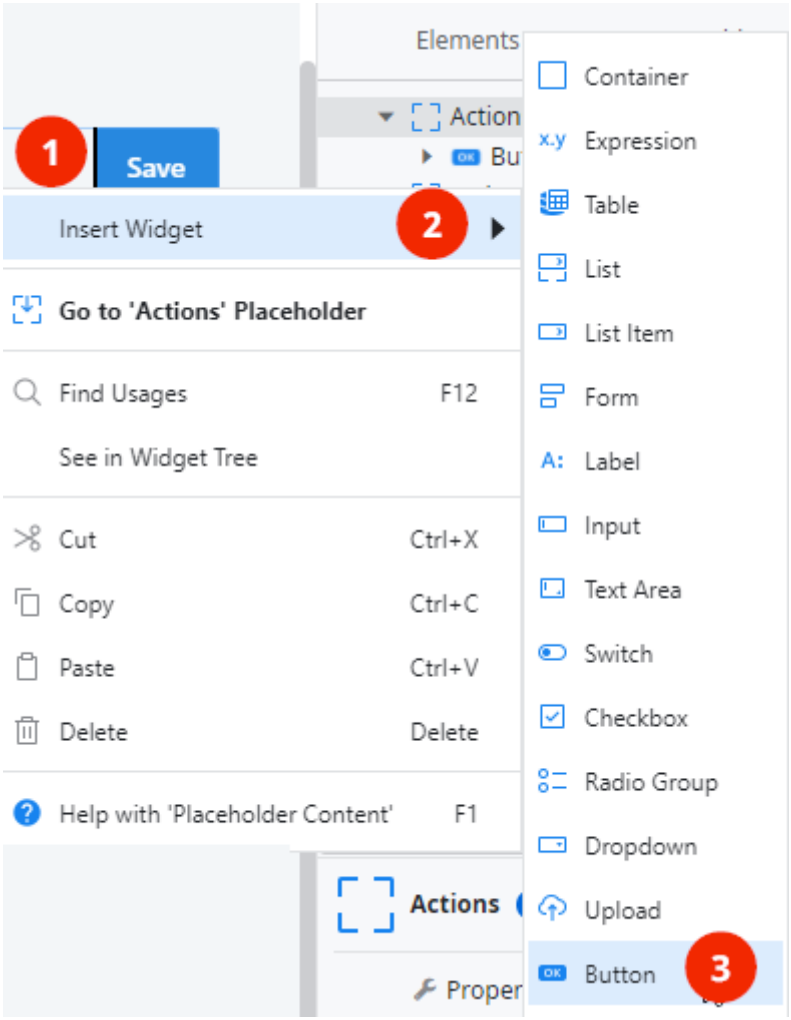
8) Switch to the Styles tab. Set **Font style** to *Bold* and the **Layout Width** to *Fill*.



Save and Submit Button

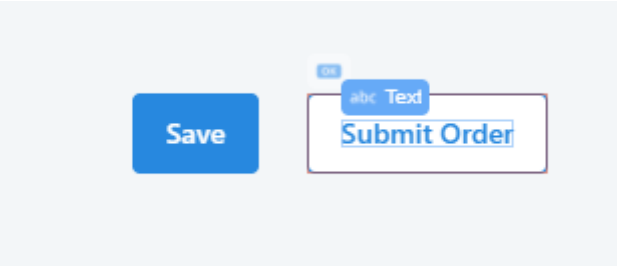
Now that the Order Form is ready, you will need a Submit Button that changes the status of the orders from **Draft** to **Submitted** when clicked. The Button should only be visible if the Order Status is **Draft**.

1) Right-click on the area to the left of the Save Button, select **Insert Widget** and choose another Button.

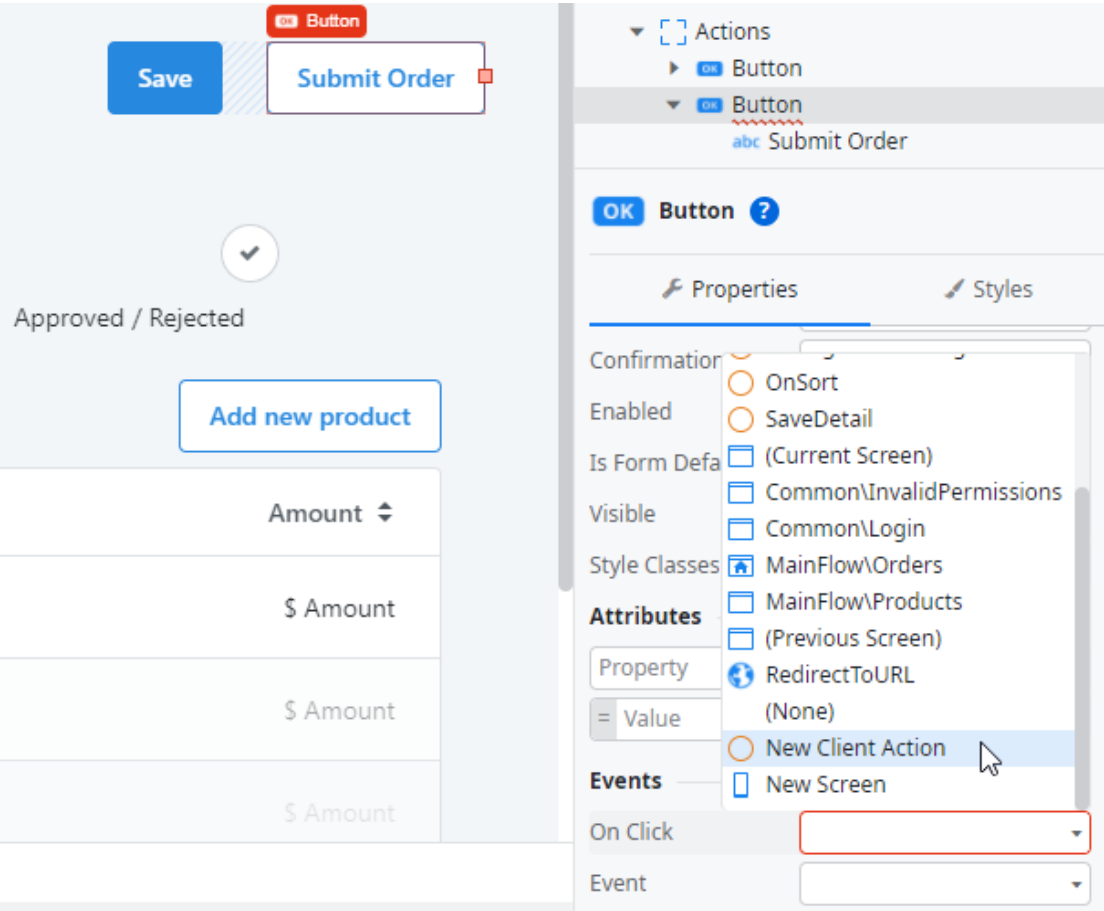


This is just another way of adding elements to the Screen.

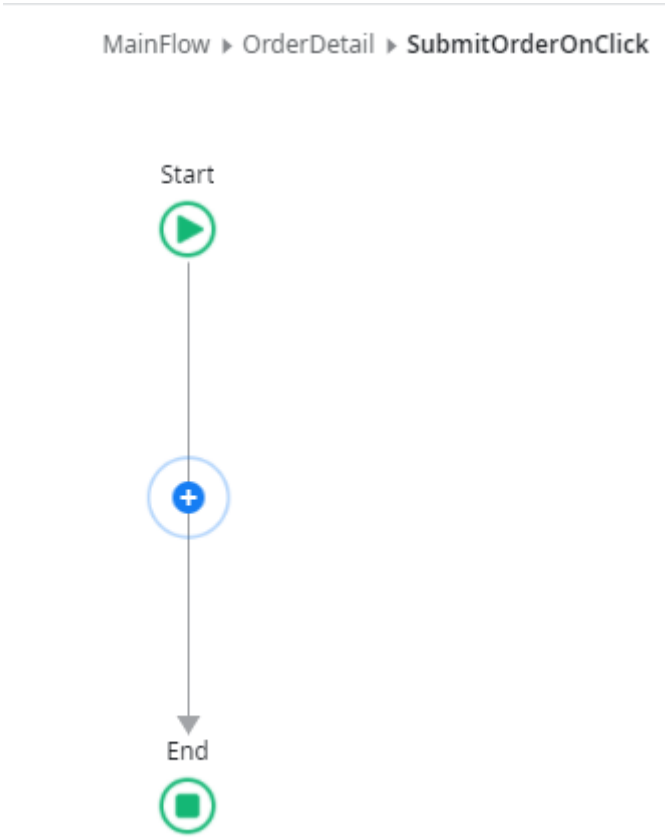
2) Set the text of the Button to *Submit Order*.



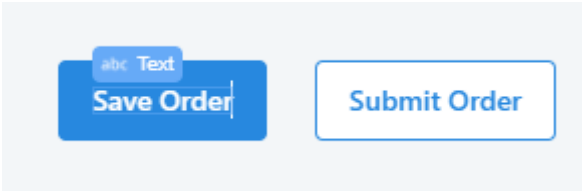
3) Click on the border of the new Button to select it and look at its properties on the right sidebar. Set the **On Click** property to **New Client Action**.



An Action called *SubmitOrderOnClick will* be created automatically. This Action will determine the behavior of the Button when clicked. You will come back to this Action later, we just wanted you to continue without errors.



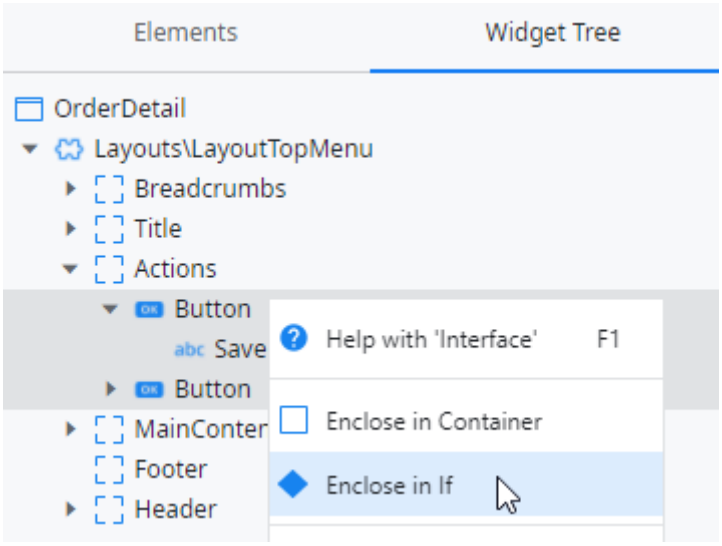
4) Go back to the OrderDetail Screen and set the text of the **Save** Button to *Save Order*.



Controlling the Buttons Visibility

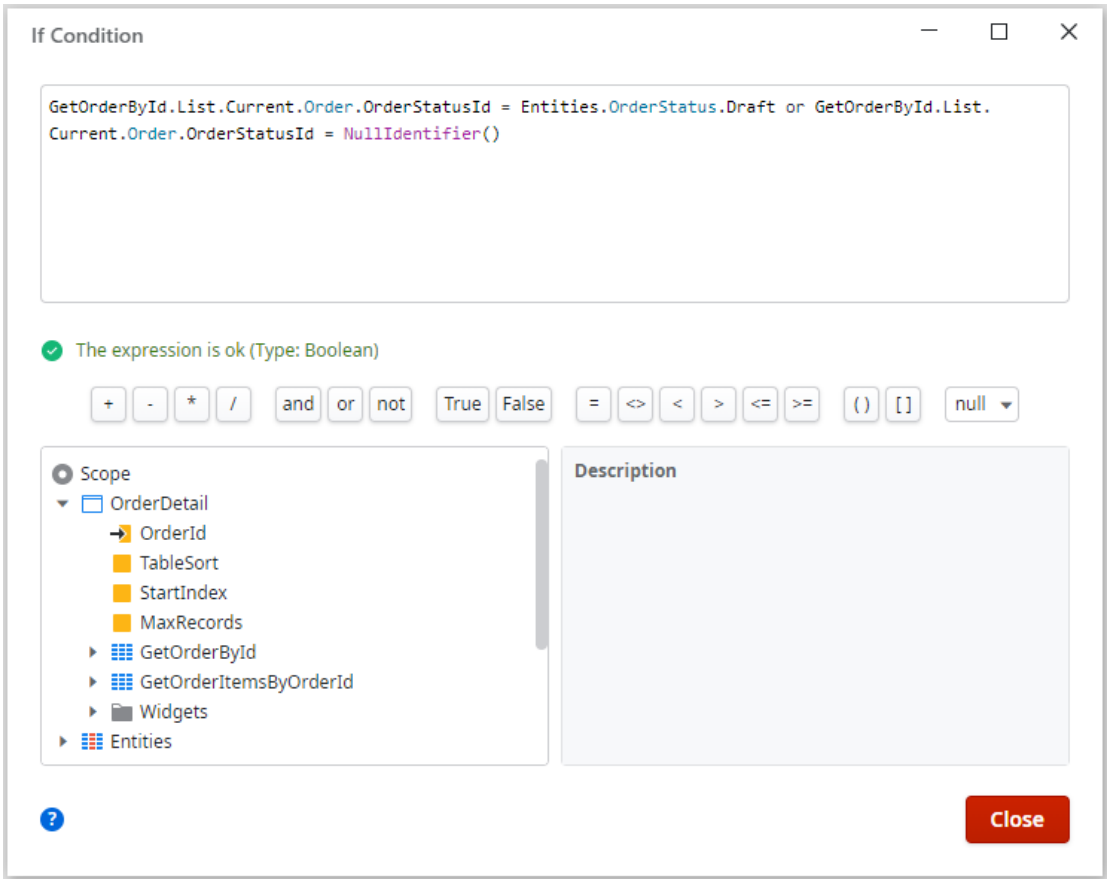
The Buttons are ready for now, but now you need to control their visibility. First, both buttons should not appear when the order is submitted or approved/rejected.

1) Select both Buttons using the CTRL key, then right-click the selection and choose the option **Enclose in If**.



2) Set the **Condition** of the If to:

```
GetOrderById.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Draft
or GetOrderById.List.Current.Order.OrderStatusId =
NullIdentifier()
```

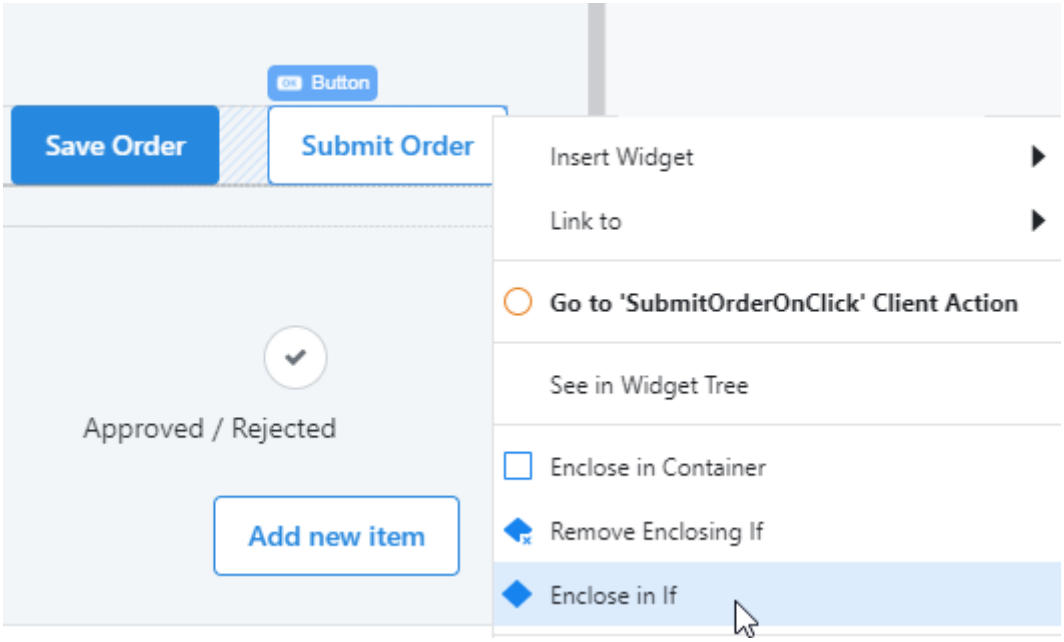


Both Buttons should only be visible if the order is being created or has the status draft.

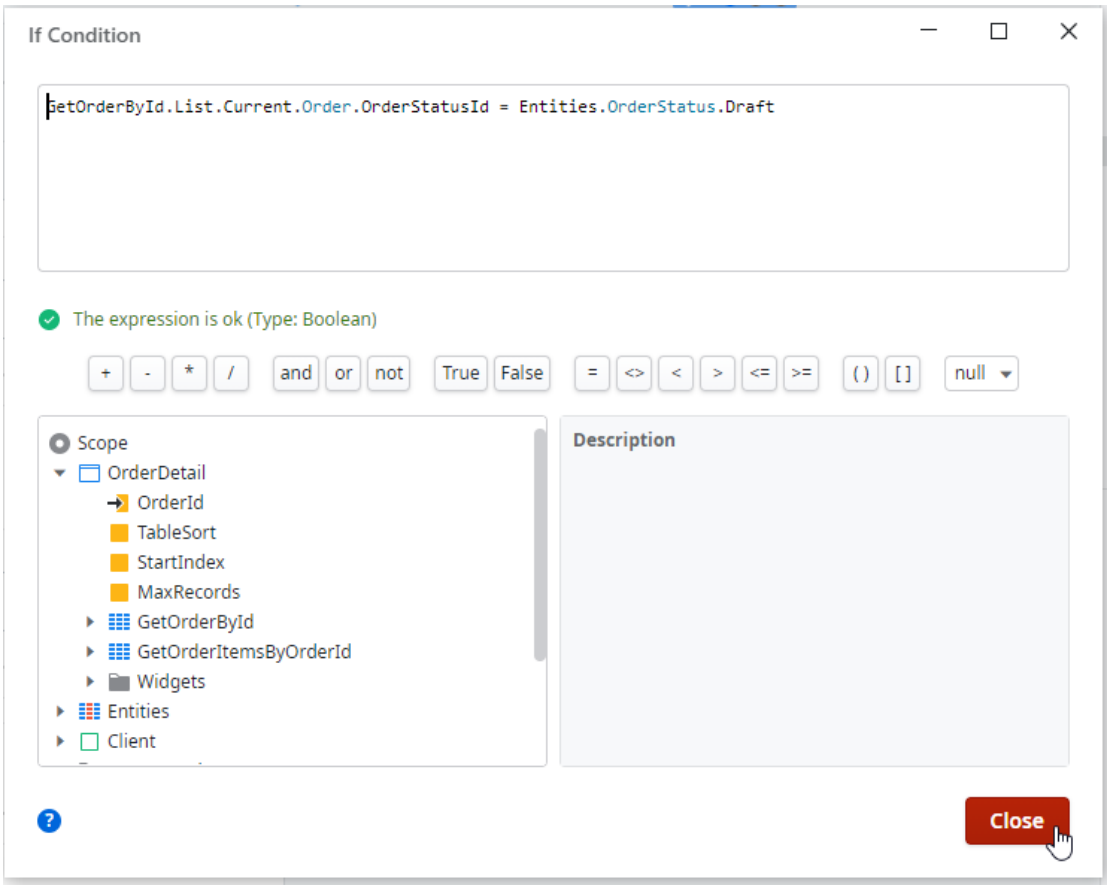
3) Type *AreButtonsVisible* in the **Name** property of the If.

But now, the Submit Order Button should only appear if the order has the status draft.

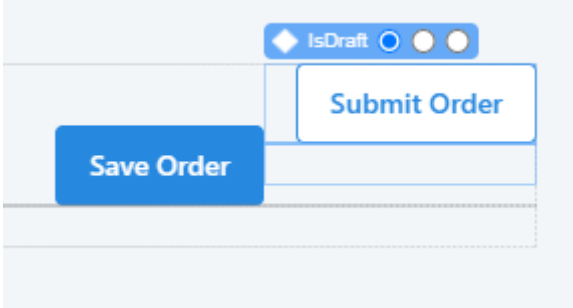
1) Right-click the **Submit Order** Button and select **Enclose in If**.



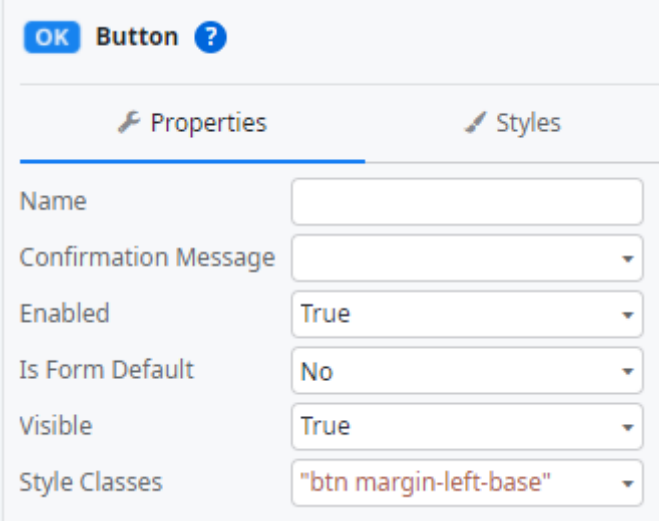
2) Set the If **Condition** to: `GetOrderById.List.Current.Order.OrderStatusId = Entities.OrderStatus.Draft`



3) Type *IsDraft* in the **Name** property of the If.



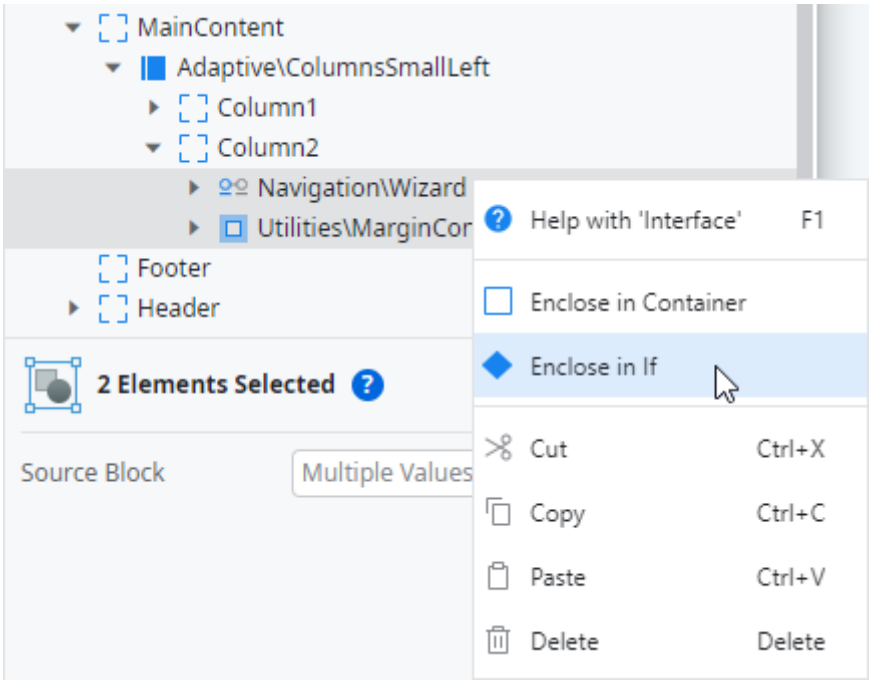
4) In the properties of the **Submit Order Button**, set the **Style Classes** property to *"btn margin-left-base"*. This is just to give some breathing room between this button and the other one on its left.



Visibility of the Wizard and its Content

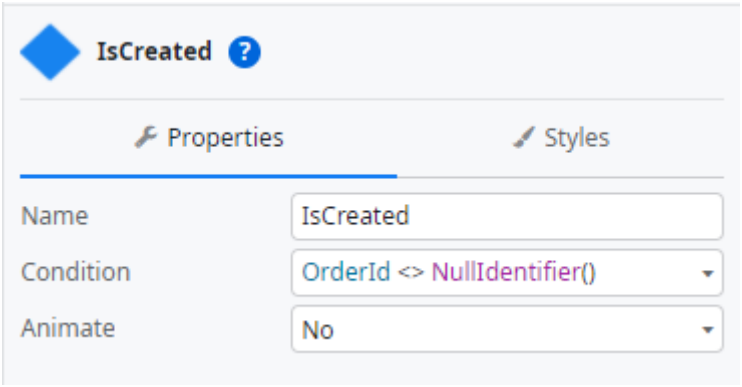
Now we just have to also adjust the visibility of the wizard and the content inside it. The content of the wizard should only be seen when an order is already created. The Add New Item Button should only appear if the order has the status draft.

1) Go to the Widget Tree, expand the MainContent area and the second column of the ColumnsSmallLeft. Select the **Wizard** and the **MarginContainer**. Right-click the selected elements and choose the option **Enclose in If**.



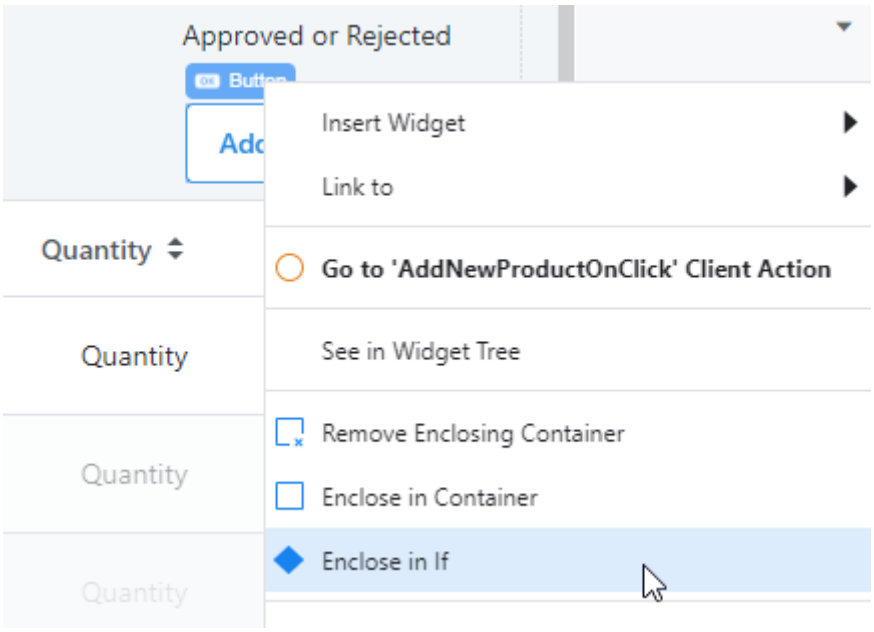
Tip: Dont forget to hold the CTRL key to select multiple elements in the Widget Tree.

2) Type *IsCreated* in the **Name** property of the If and add the **Condition** : `OrderId <> NullIdentifier()`



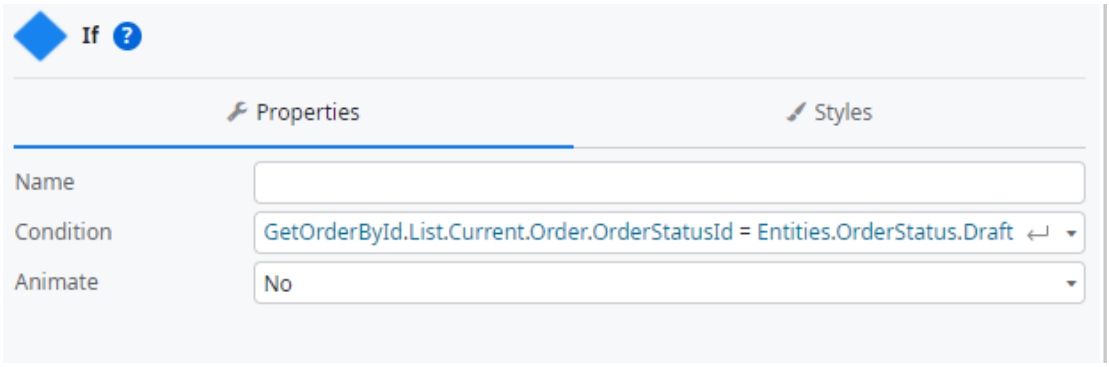
The identifier of an Order is *NullIdentifier* when it is being created, meaning it does not exist yet. So, this condition ensures that the wizard and its content is only visible for orders that are already created.

3) Right-click the **Add new item** button and select the option **Enclose in If**.

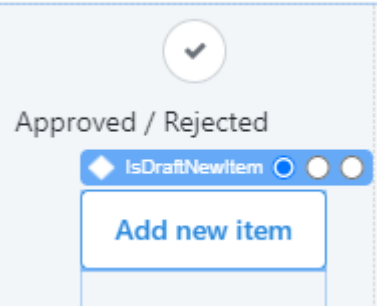


4) Set the If **Condition** to:

```
GetOrderById.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Draft
```



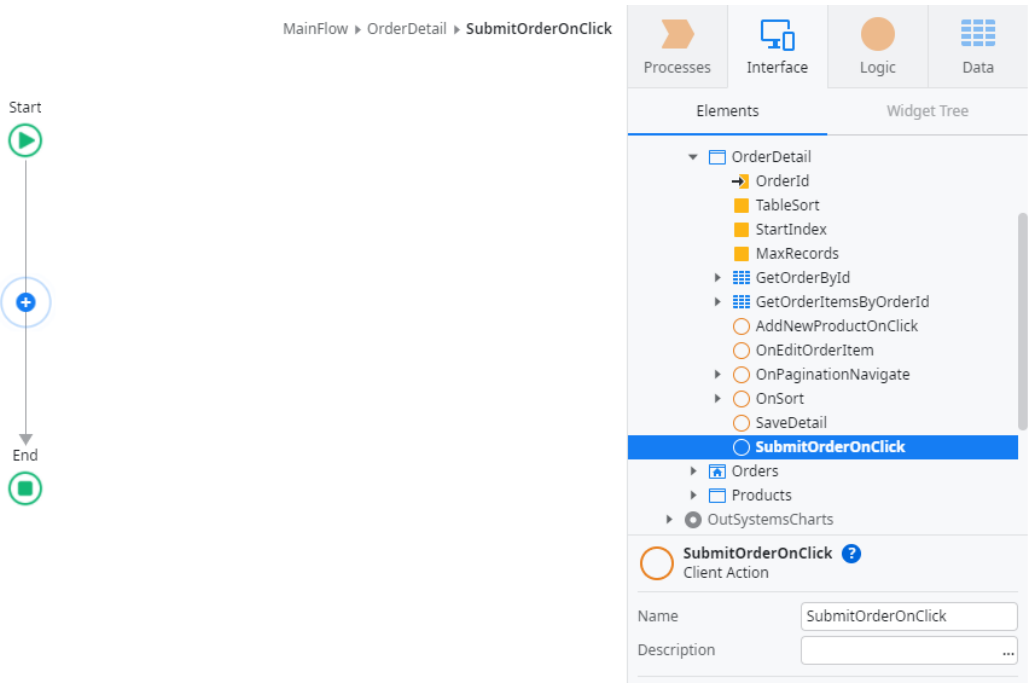
5) Type *IsDraftNewItem* in the **Name** property of the If.



Defining the Buttons' Behavior

Ok! No more Ifs! They were definitely very important to control which options should appear to the end-user, but it's time to close this tutorial by doing one final thing: to actually change the status of the order when the user clicks on the Submit Button. Here, the order status should change from draft to submit.

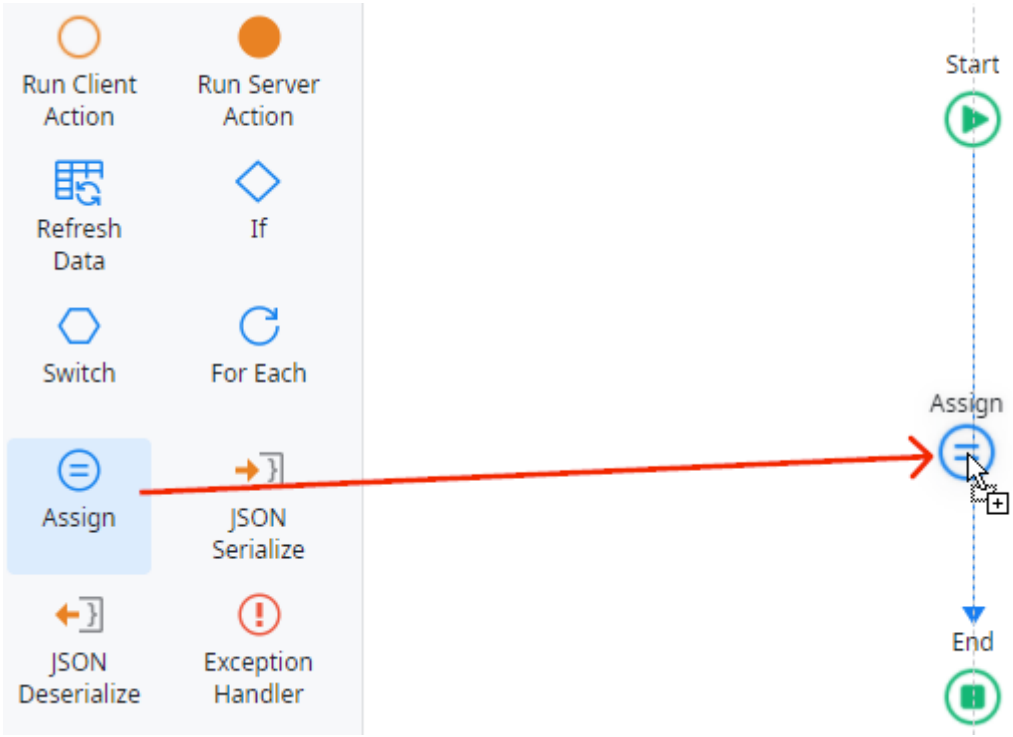
1) Double-click the **SubmitOrderOnClick** inside the OrderDetail Screen to open it.



2) Change the **Name** of the Action to *OnSubmitOrder*.

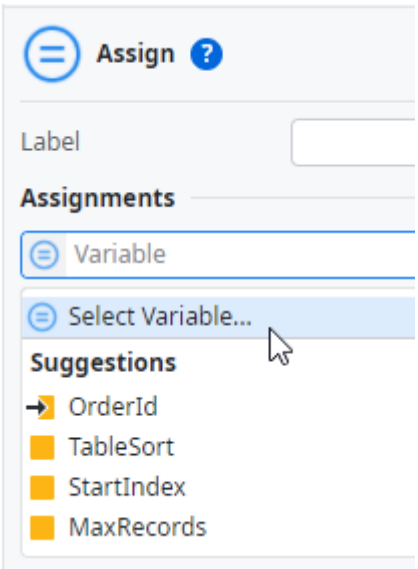
Change the Status to Submitted

1) Drag an **Assign** from the left sidebar and drop it on the Action flow.

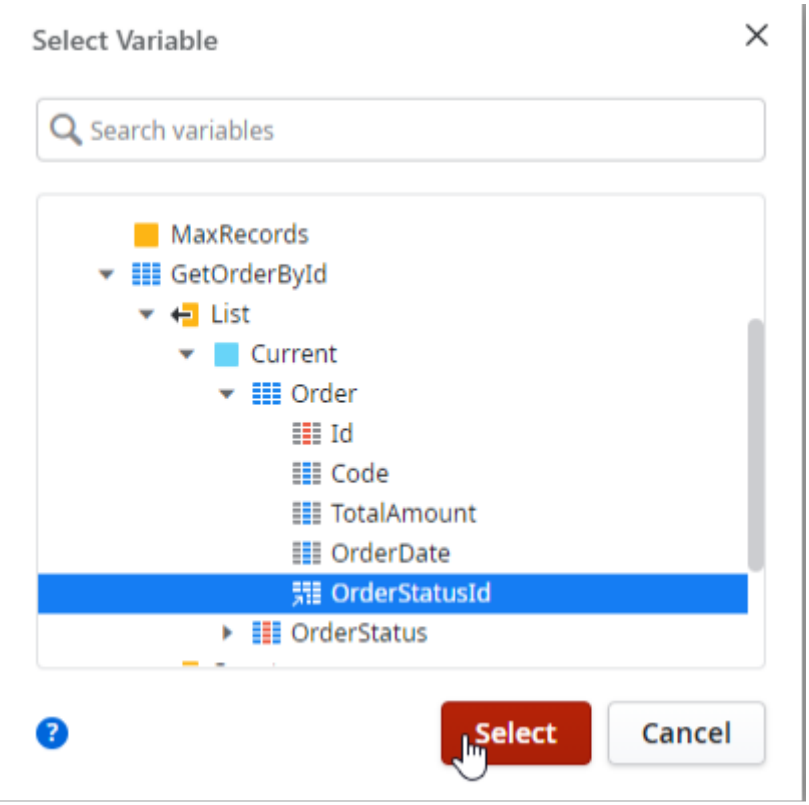


We will use this element to assign the new status to the order.

2) On the right sidebar, expand the Variable dropdown and choose **Select Variable...**.

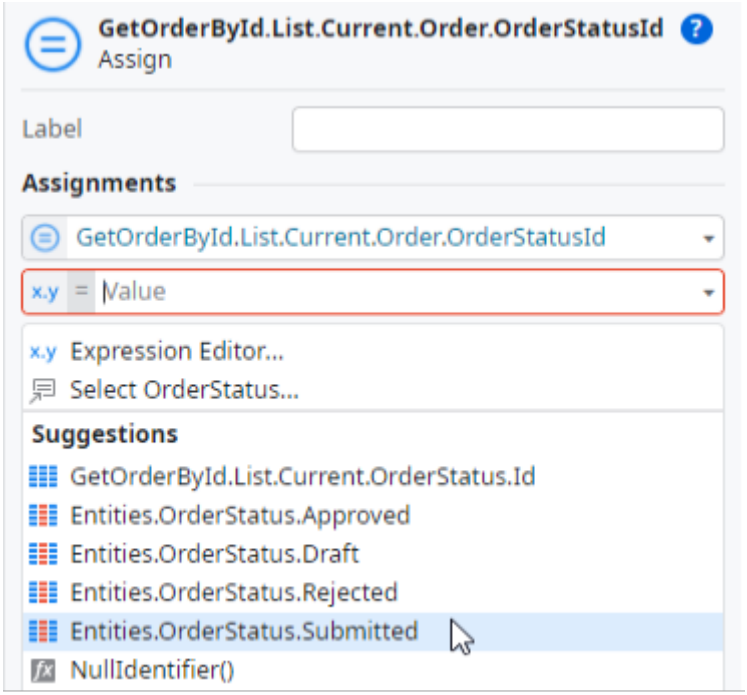


3) In the new dialog, choose the *GetOrderById.List.Current.Order.OrderStatusId* just like shown in the image and click **Select** when you're done.



This is the field that has the status of the order.

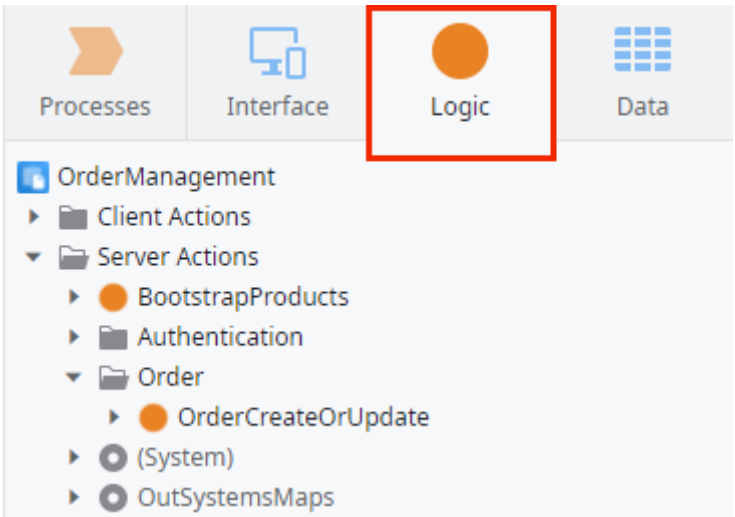
4) In the **Value** field below, add `Entities.OrderStatus.Submitted` to set the status as Submitted.



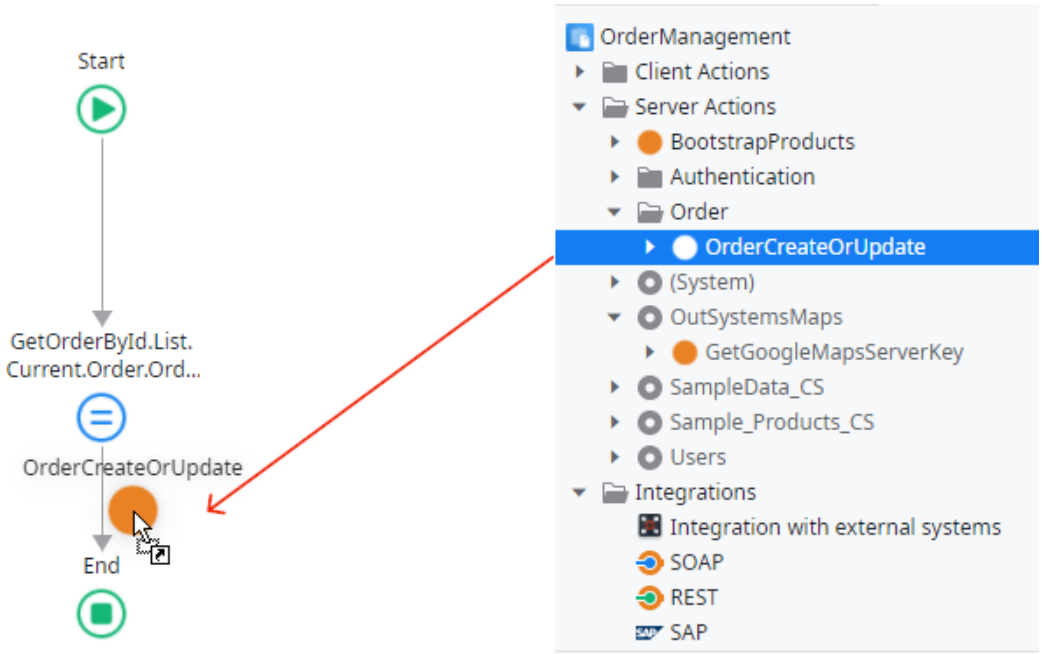
Update the Order in the Database

You changed the status, now you need to save the updated order in the database.

1) Switch to **Logic** tab, open the **Server Actions** folder, and then the Order folder.

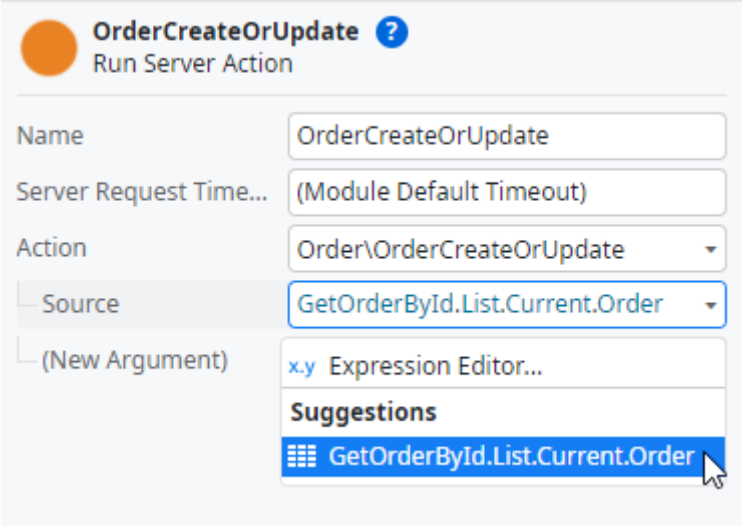


2) Drag the **OrderCreateOrUpdate** Action and drop it on the flow, under the Assign.



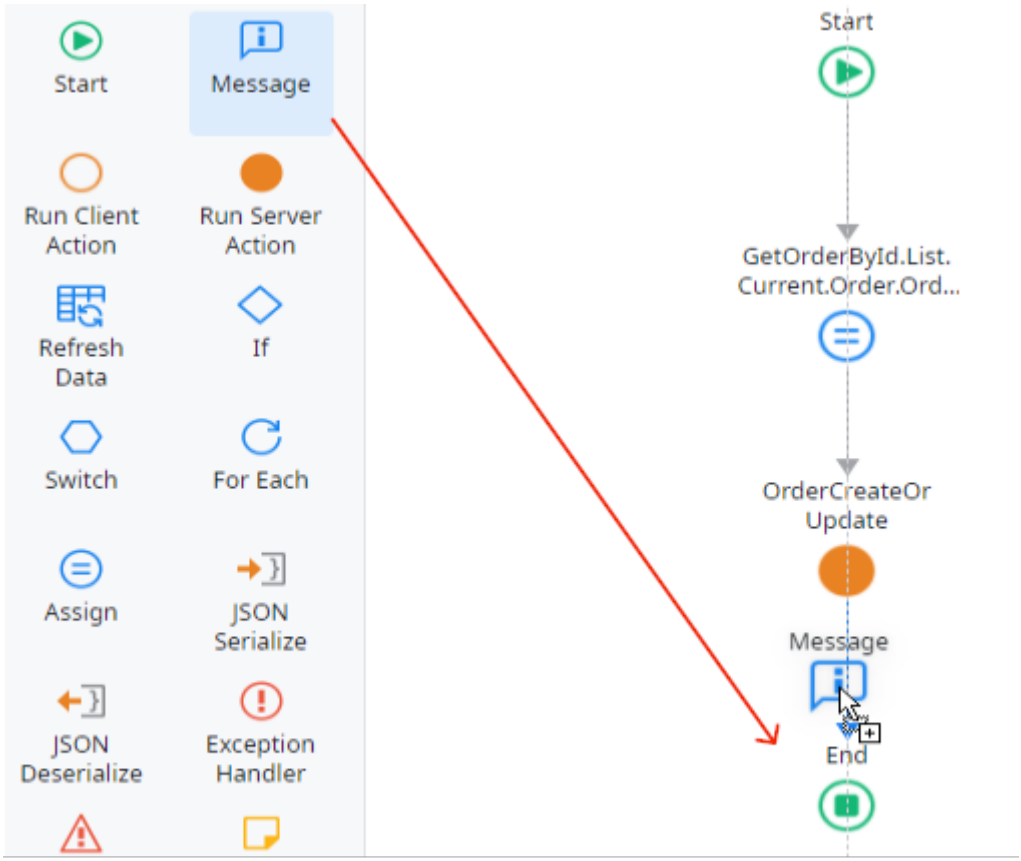
This Action was already created for you (when you created the Orders and OrderDetail Screens) and has the logic to save the Order in the database.

3) There's an error. With the recently dragged element selected, look at the right sidebar to find that the **Source** property is the cause of the error. Basically, the Action expects we pass the Order that is going to be updated in the database. And that's what we need to do. Select *GetOrderById.List.Current.Order* in the **Source** property.

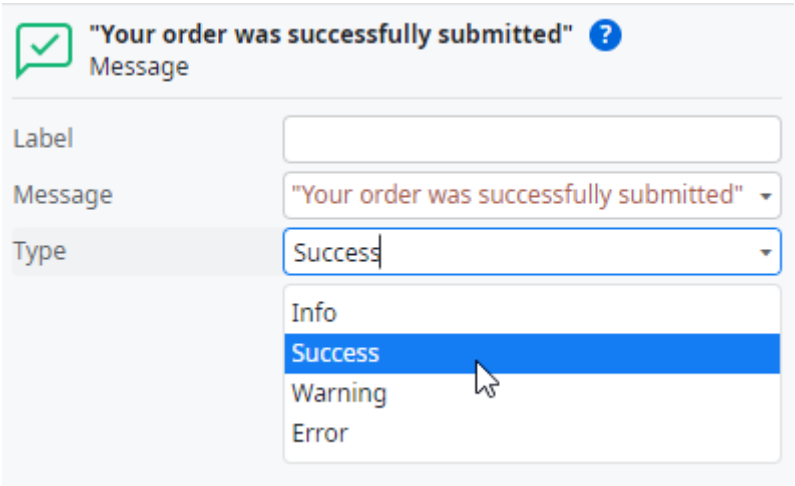


Since updating the order is a big deal, let's add a message to let the user know that everything went well.

4) Drag a **Message** from the left sidebar and drop it on the Action flow, after the OrderCreateOrUpdate.

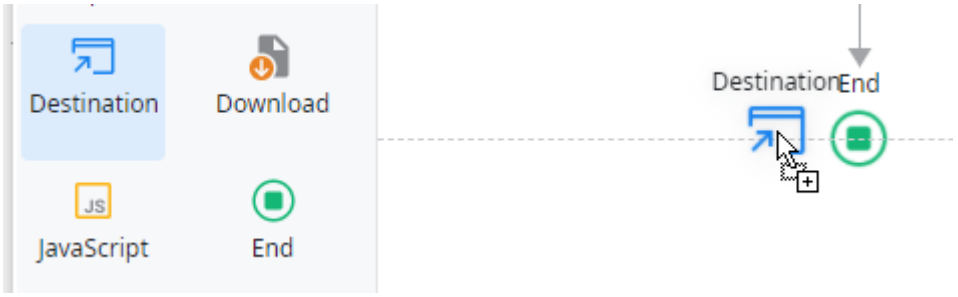


5) In the Message properties on the right sidebar, select *Success:* in the the **Type** and type the **Message** *"Your order was successfully submitted"*.

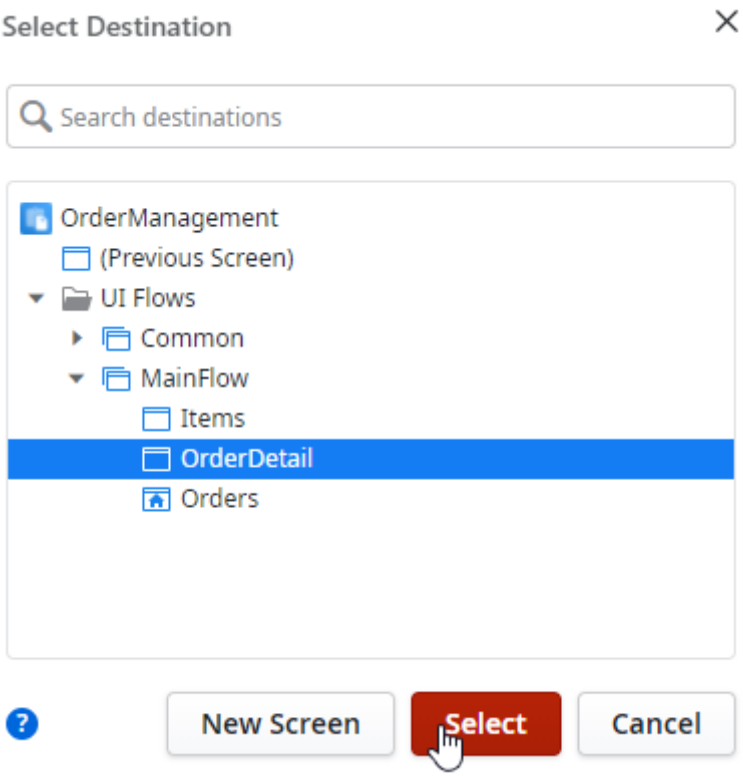


Last but not least, let's finish the Action with a redirect to the Screen we're on, to force the page to be refreshed.

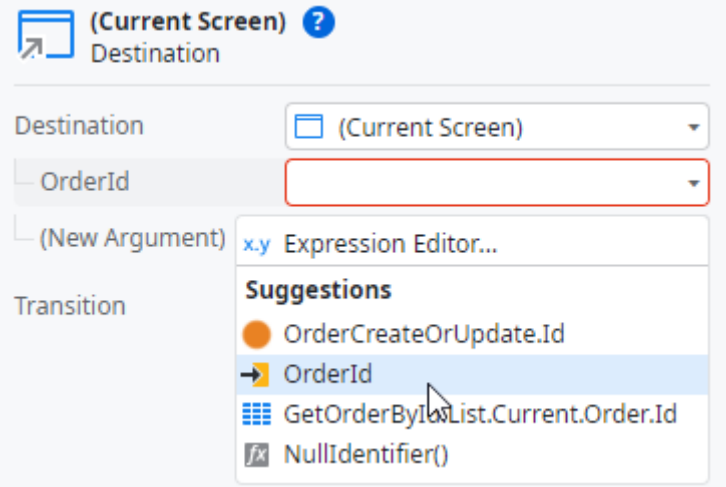
6) Drag the **Destination** element from the left sidebar and drop it on top of the **End** element. Note that you need to replace one with another!



7) On the new dialog that appears, select the **OrderDetail** Screen and click **Select**.



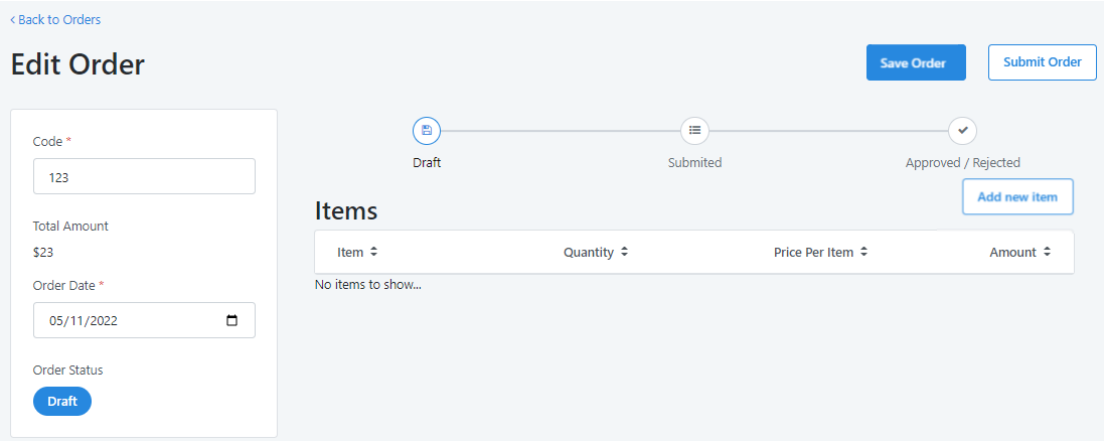
8) In the properties of the Destination element, there's an error. This is because we're navigating to the Screen we're on, which is the OrderDetail, and the OrderDetail Screen expects an input parameter. Set it to be the *OrderId*.



When we navigate to a Screen that has input parameters, it's expected we pass a value for that input. Since we're staying on the same Screen, editing the same Order, we can leverage the *OrderId* input parameter of the Screen itself. Basically, we're just passing the same value again.

9) Publish the module and open the app in your browser.

10) Make sure you test an order in several stages: being created, in draft and submitted, and make sure the options appear as they are supposed to. For instance, if the order is draft, you should see the Submit Button.



Wrapping up

Congratulations on finishing this tutorial. With this exercise, we had the chance to go through some essential aspects of OutSystems and get to know more about the platform.

References

If you want to deep dive into the subjects that we have seen in this exercise, we have prepared some useful links for you:

1) If
2) Assign
3) Messages

See you in the next tutorial!