# Wizard - Approve and Reject Order

## Table of Contents

**Outline**........................................................ **2**

Scenario 2

**How-to**........................................................ **4**

Getting Started 4

Outdated Dependencies 4

Approving an Order 5

Approve Button 6

Logic for Approving an Order 7

Reject Orders 14

Reject Button 15

Logic for Rejecting an Order 17

Testing a Submitted Order 21

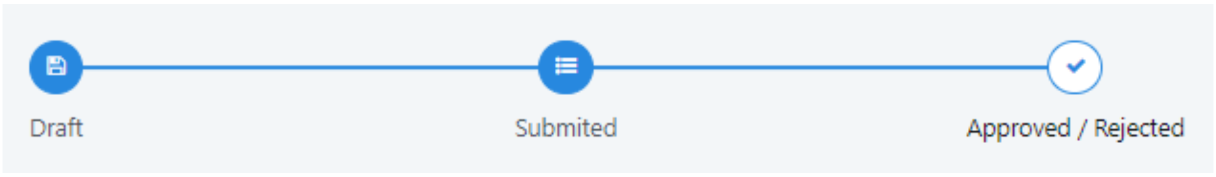
Controlling the Visibility of the Approve and Reject Buttons 22

**Wrapping up**........................................................ **25**

References 25

## 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 the approved or rejected statuses.



To achieve that, you will:

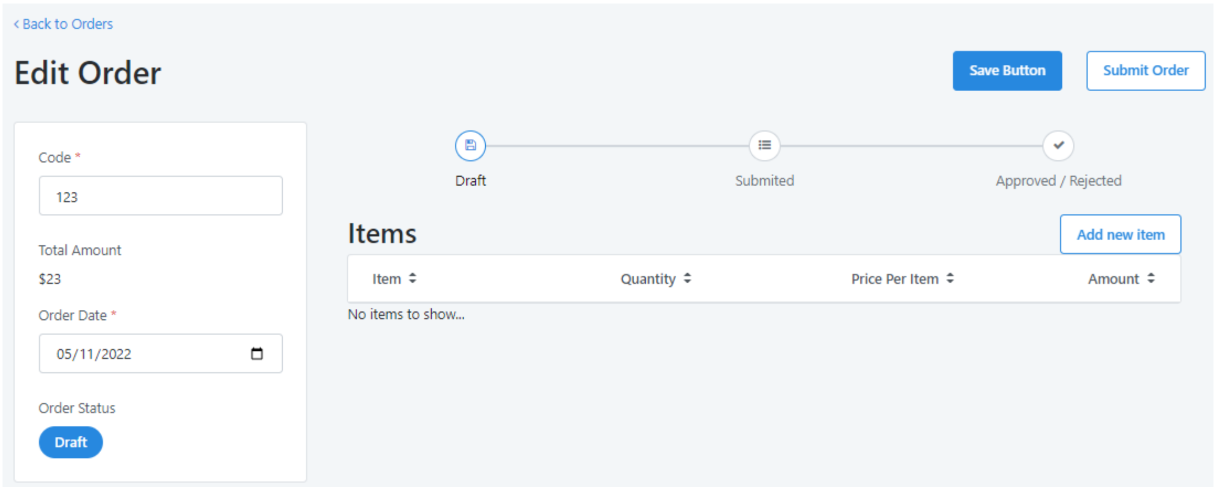
- Create a new Approve Order Button and associated logic to approve an order.
- Create a new Reject Order Button and associated logic to reject an order.
- Make sure the Buttons are **only** visible when the order is in the Submitted status.

At the end of this tutorial, you will be able to open your app in the browser and test the complete order workflow, from draft to approved or rejected.

## Scenario

At this point, your Order Management application already has most of the order status workflow ready, using the wizard. The orders can already be created, making them Draft, and then you can move them to the status of Submitted.

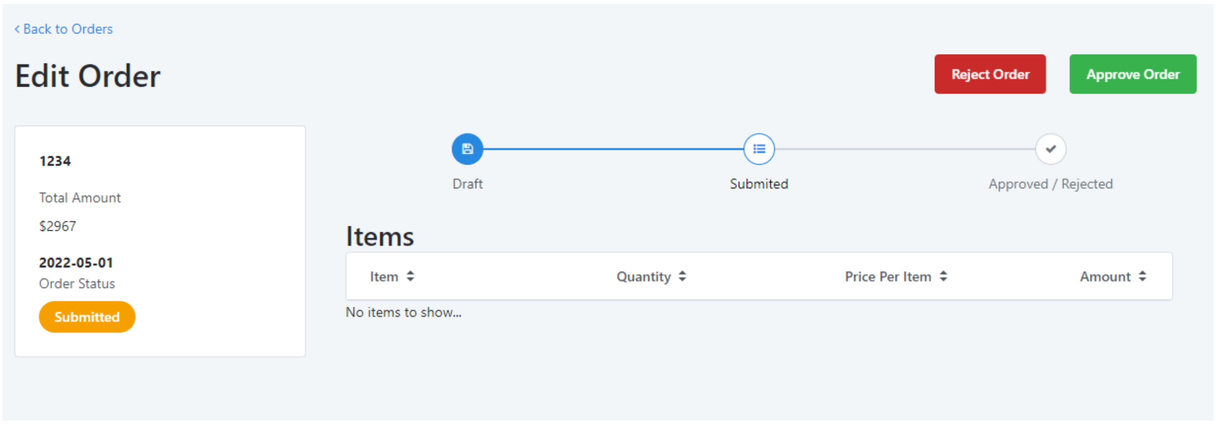
So, in the previous tutorial, you implemented the UI and logic for the Submitted status...



Code *
123
Total Amount
$23
Order Date *
05/11/2022
Order Status
Draft

No items to show...

... and now you will do something very similar for the last ones: approved or rejected. The OrderDetail Screen will have two new Buttons, one to Approve Orders and one to Reject Orders.



You will also create the logic to update the order accordingly in the database and make sure the Buttons will only appear to the user if the order is in the Submitted status.

At the end, you will will be able to change the status of an order all the way from Draft to Approved or Rejected.

## How-to

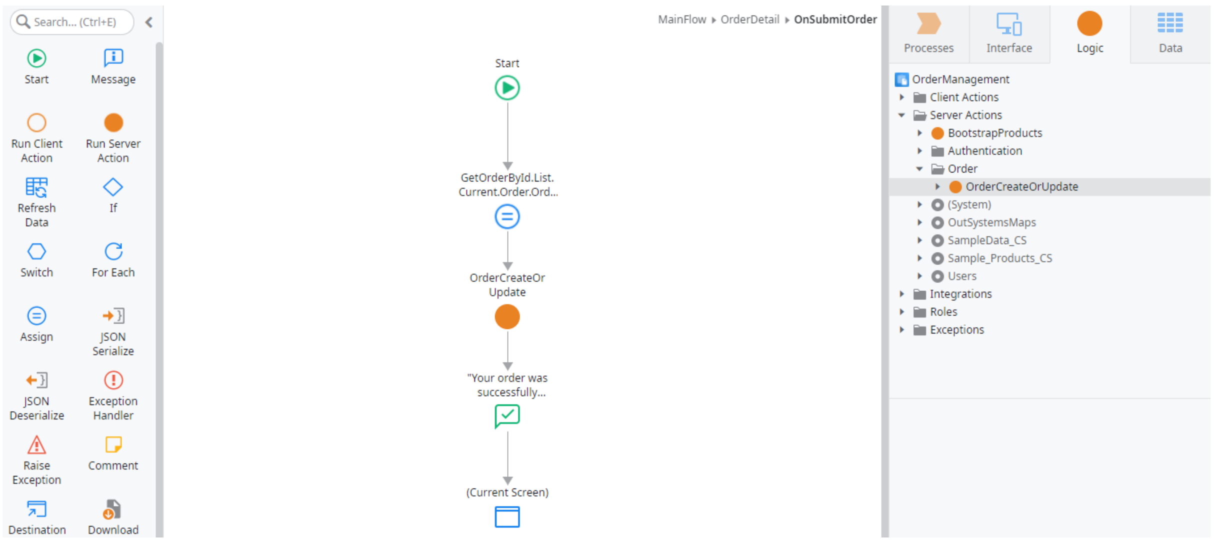
In this section, we'll show a thorough step-by-step description of how to implement the scenario described in the previous section.

### Getting Started

In this tutorial we are assuming that you have already followed the previous tutorials, and have the Orders, OrderDetail and Items Screens ready. In particular, at this point, the OrderDetail is expected to have the wizard with the draft and submitted status logic already defined.

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

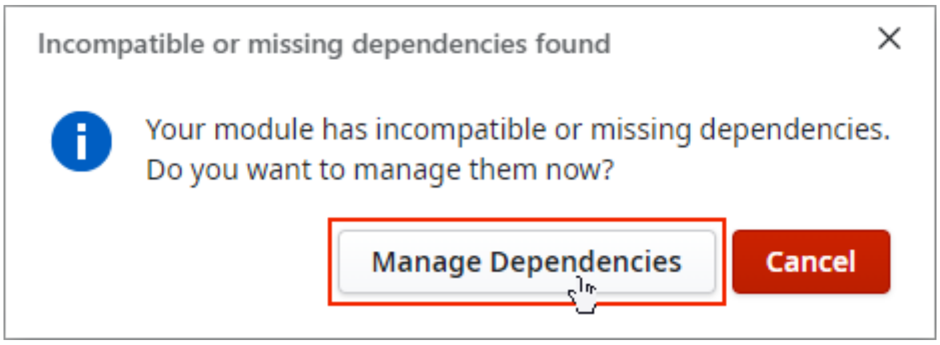
To start this exercise, you need the Service Studio with the module OrderManagement opened. You should see the Screen below with the source of our application.



### 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!

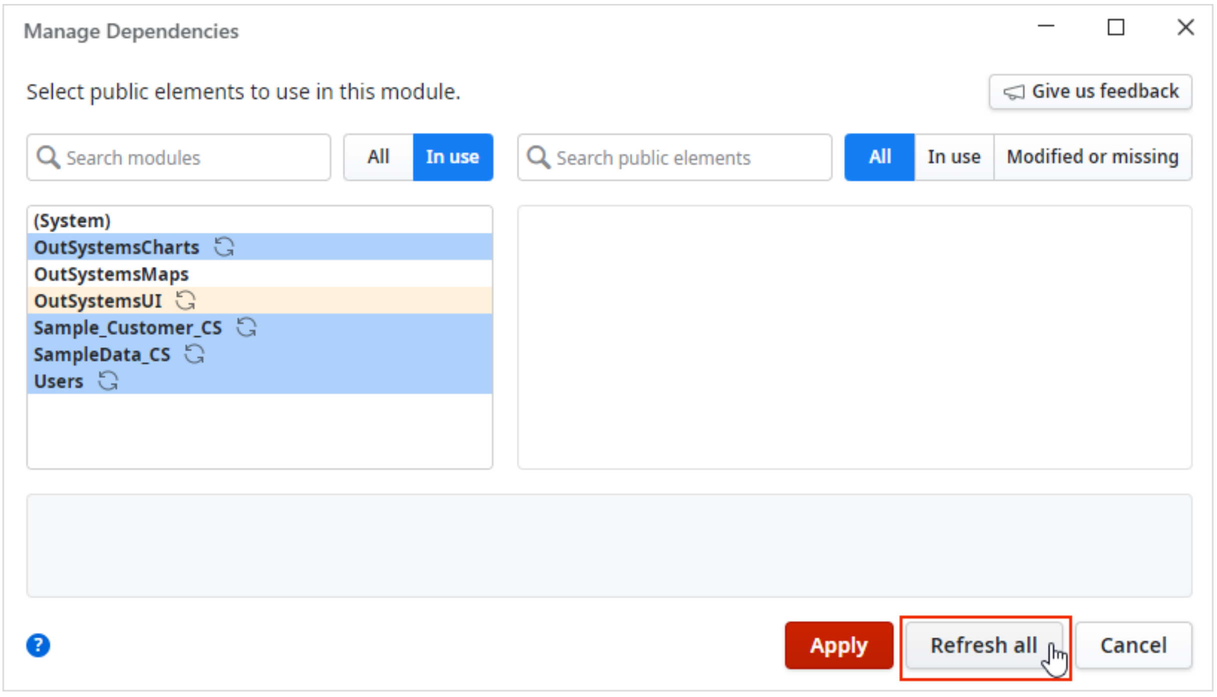
If that happens, simply click on the button that says *"Manage Dependencies"* to see the outdated components.



Incompatible or missing dependencies found

Your module has incompatible or missing dependencies. Do you want to manage them now?

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



Manage Dependencies

Select public elements to use in this module.

(System)
OutSystemsCharts
OutSystemsMaps
OutSystemsUI
Sample_Customer_CS
SampleData_CS
Users

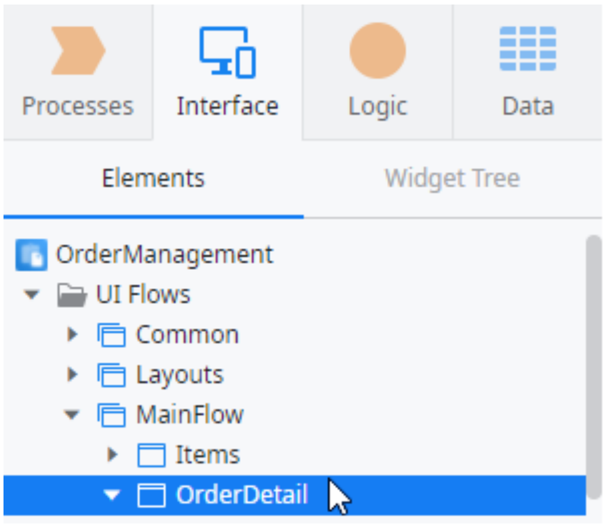
Now publish the module to update the project

## Approving an Order

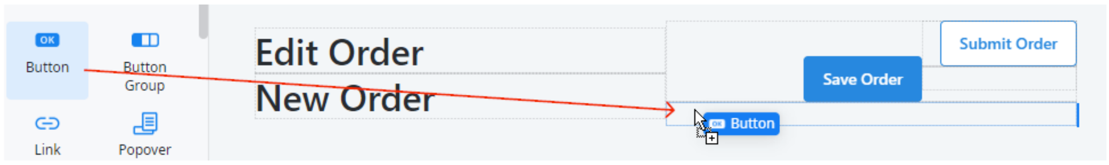
Let's start by defining the UI and logic for approving an order.

## Approve Button

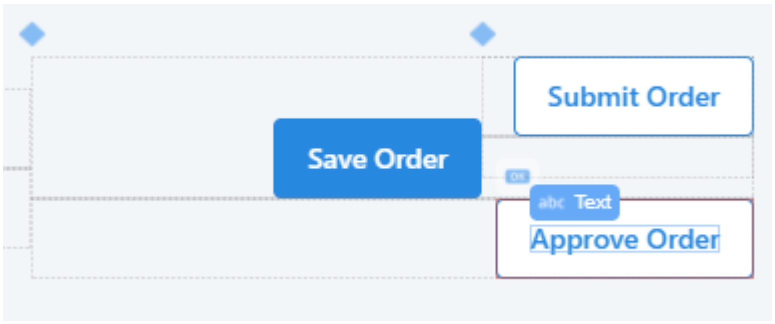
1) Switch to the Interface tab and double-click the **OrderDetail** Screen to open it.



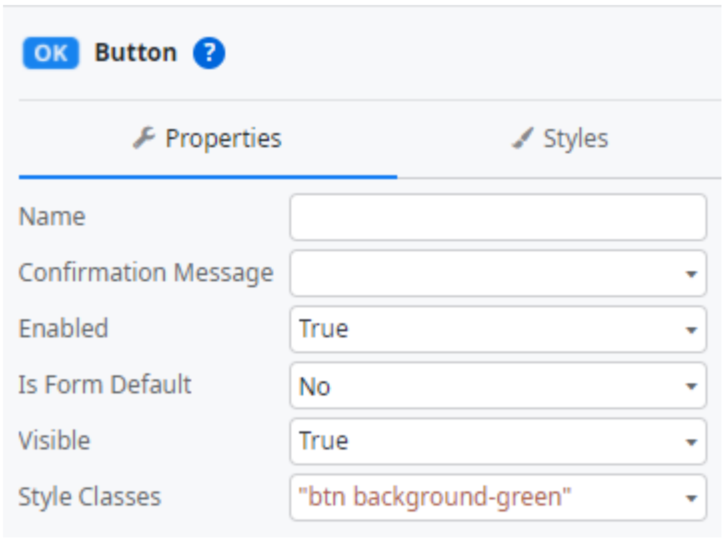
2) Drag a Button from the Toolbox (left sidebar) and drop it on the **False** branch of the **AreButtonsVisible** If (the one with the Save and Submit buttons).



3) Type the text *"Approve Order"* in the Button.



4) Click on the Button (and not the text inside it) to see its properties on the right sidebar. Let's make the Button green! Set the **Style Classes** property to *"btn background-green"*



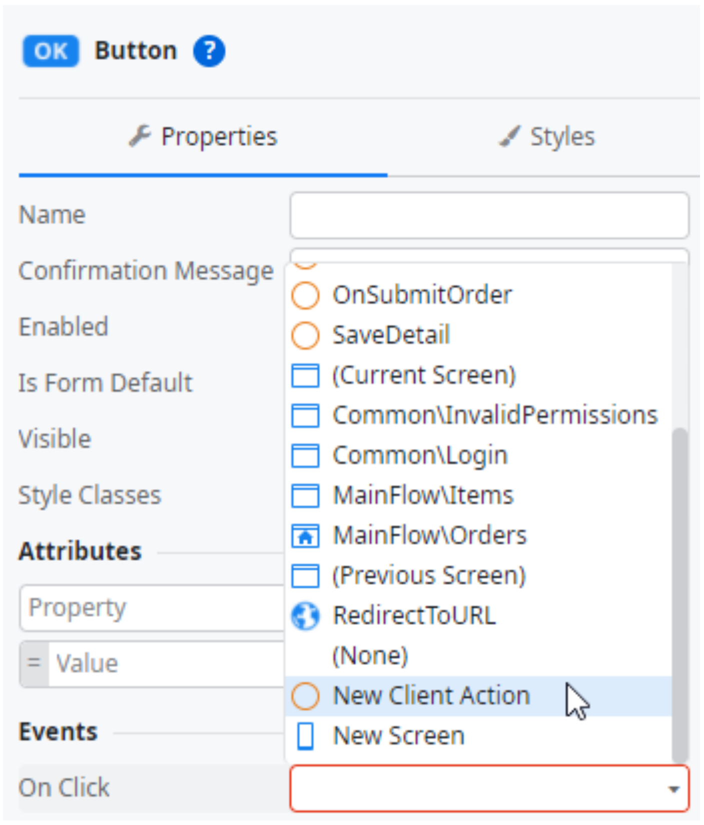
|  |  |
|---|---|
| Name |  |
| Confirmation Message |  |
| Enabled | True |
| Is Form Default | No |
| Visible | True |
| Style Classes | "btn background-green" |

## Logic for Approving an Order

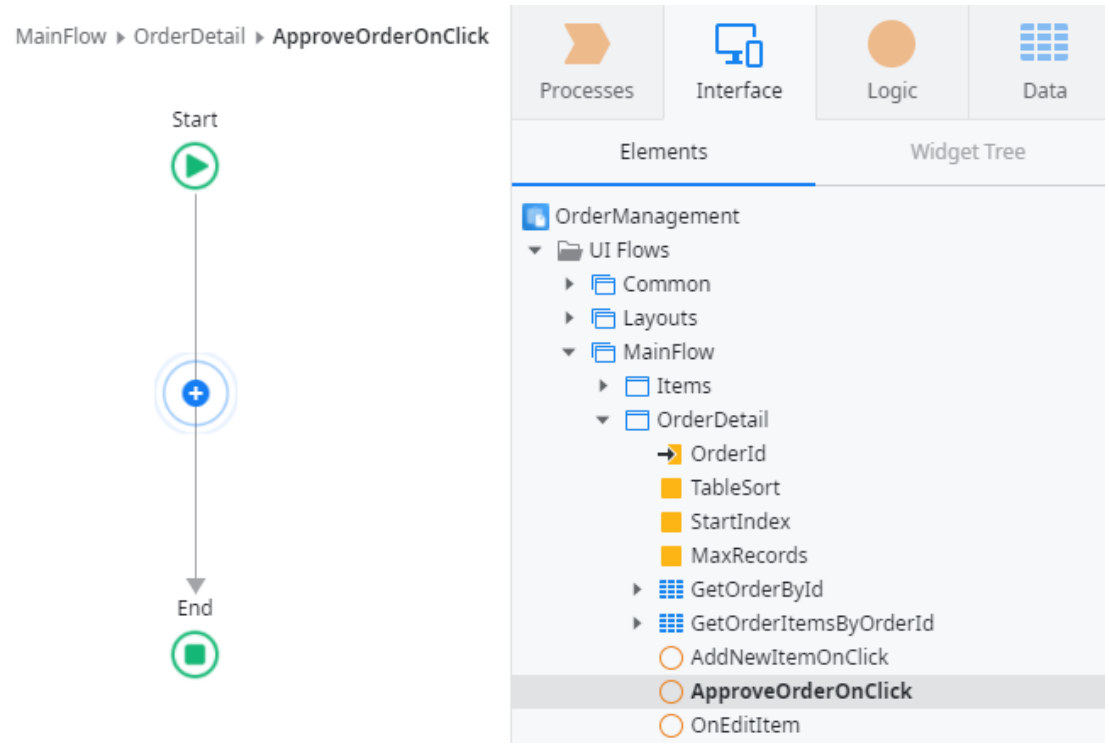
At this point, the Button has an error because its behavior has not been defined yet. So let's create a new Client Action that runs when the button is clicked.

This Action will have the logic to change the status of the Order to Approved and save it in the database, just like we did in the previous tutorial for the Submitted status.

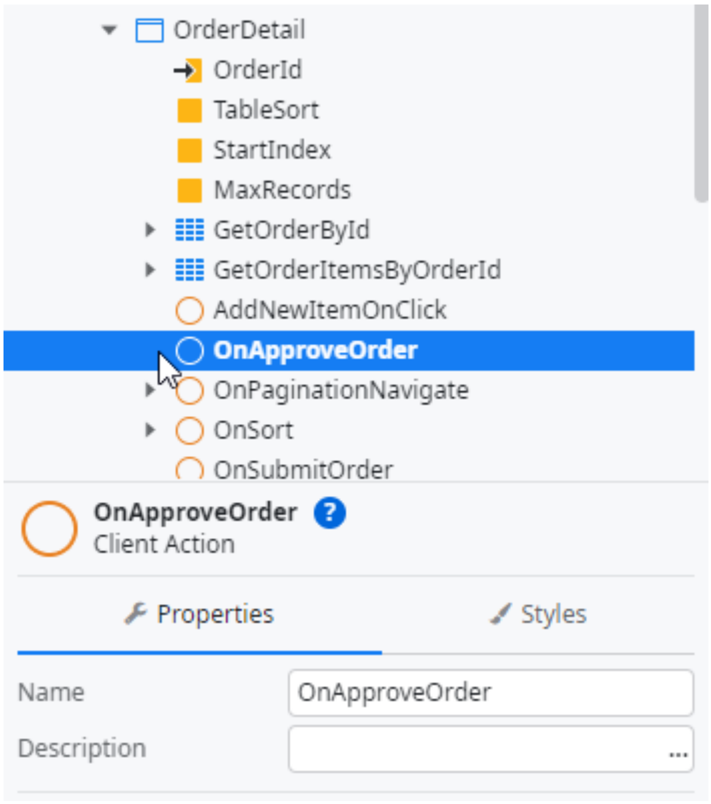
1) Still in the Button's properties, expand the dropdown of the **On Click** property and select **New Client Action**.



A new Client Action is created and Service Studio will automatically open it and name it *ApproveOrderOnClick*.

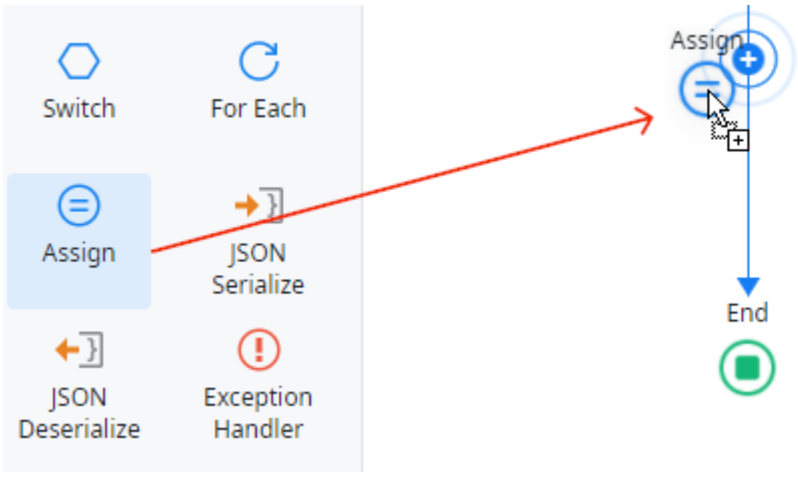


2) Click on the Action on the right sidebar, and in its properties change the **Name** to *OnApproveOrder*.



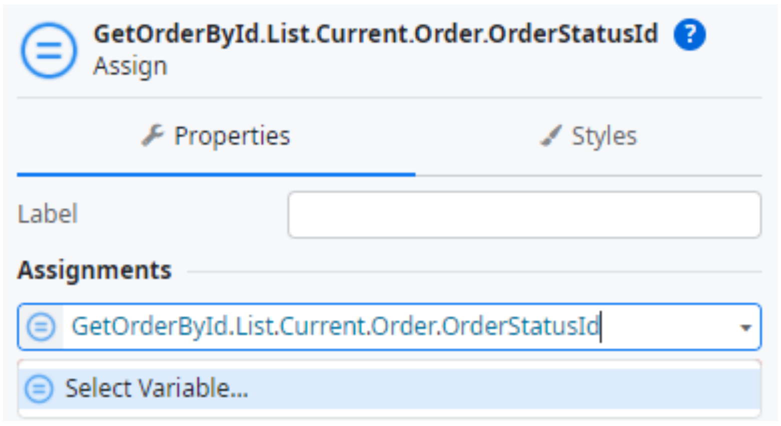
| | |
|---|---|
| Name | OnApproveOrder |
| Description | |

3) Inside the Action, drag an **Assign** from the left sidebar and drop it in the flow.



You will use it to change the Order status to Approved.

4) In the Assignments section that is now visible on the right sidebar, set the **Variable** field to: `GetOrderById.List.Current.Order.OrderStatusId`



GetOrderById.List.Current.Order.OrderStatusId
Assign

Properties | Styles

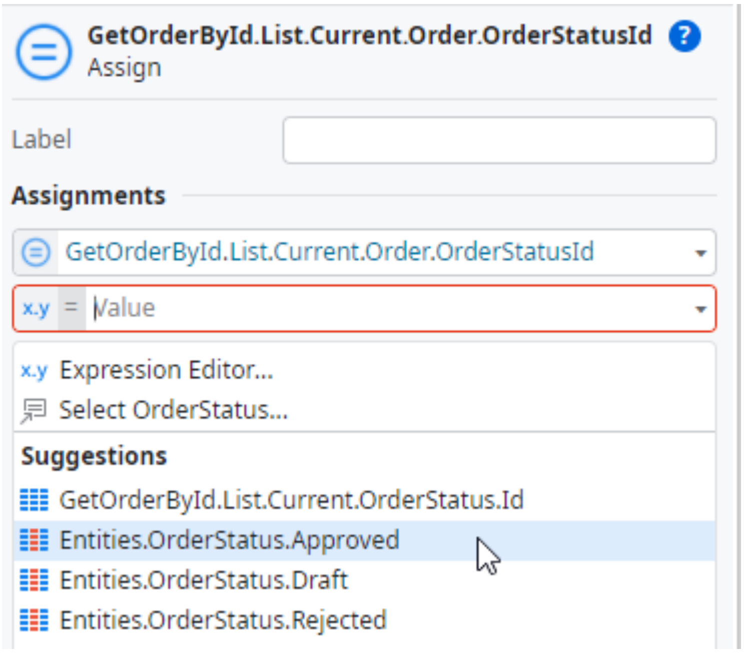
Label

**Assignments**

GetOrderById.List.Current.Order.OrderStatusId

Select Variable...

5) Now set the **Value** field to the suggested value *Entities.OrderStatus.Approved*.



**GetOrderById.List.Current.Order.OrderStatusId**
Assign

Label:

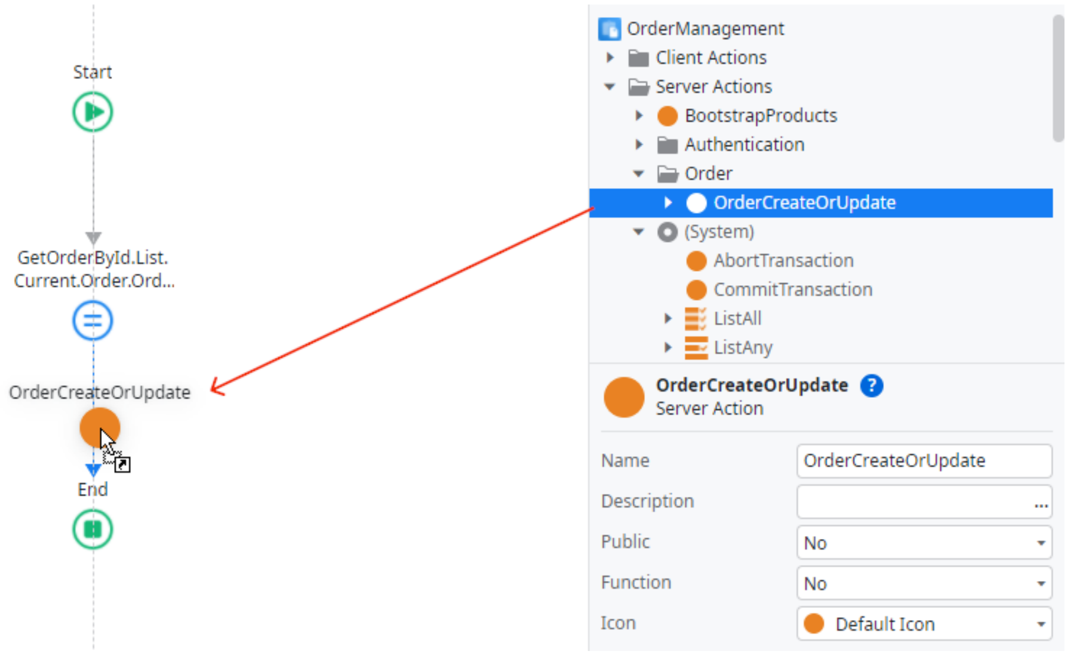
**Assignments**

GetOrderById.List.Current.Order.OrderStatusId
x.y = Value

**Suggestions**
GetOrderById.List.Current.OrderStatus.Id
Entities.OrderStatus.Approved
Entities.OrderStatus.Draft
Entities.OrderStatus.Rejected

We just changed the status to Approved. Now we need to save the order in the database. Remember the last tutorial? There is an Action ready for you to use it that will do just that!

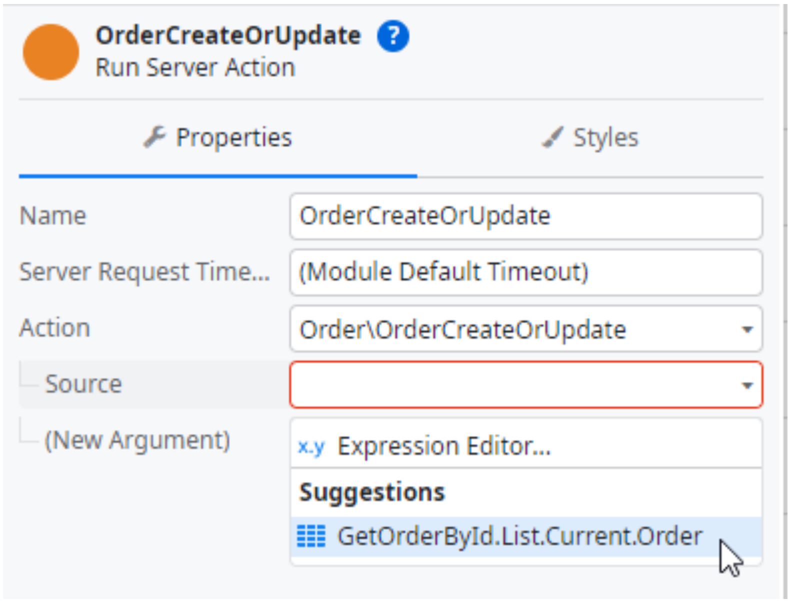
6) Go to Logic tab, then open the Server Actions folder and then the Order folder. You will see the **OrderCreateOrUpdate** Action. Drag the Action and drop it after the assign in the flow.



**OrderCreateOrUpdate**
Server Action

| | |
|---|---|
| Name | OrderCreateOrUpdate |
| Description | |
| Public | No |
| Function | No |
| Icon | Default Icon |

7) You will see an error in the OrderCreateOrUpdate properties, more specifically in the **Source** property. Set the property to `GetOrderById.List.Current.Order`

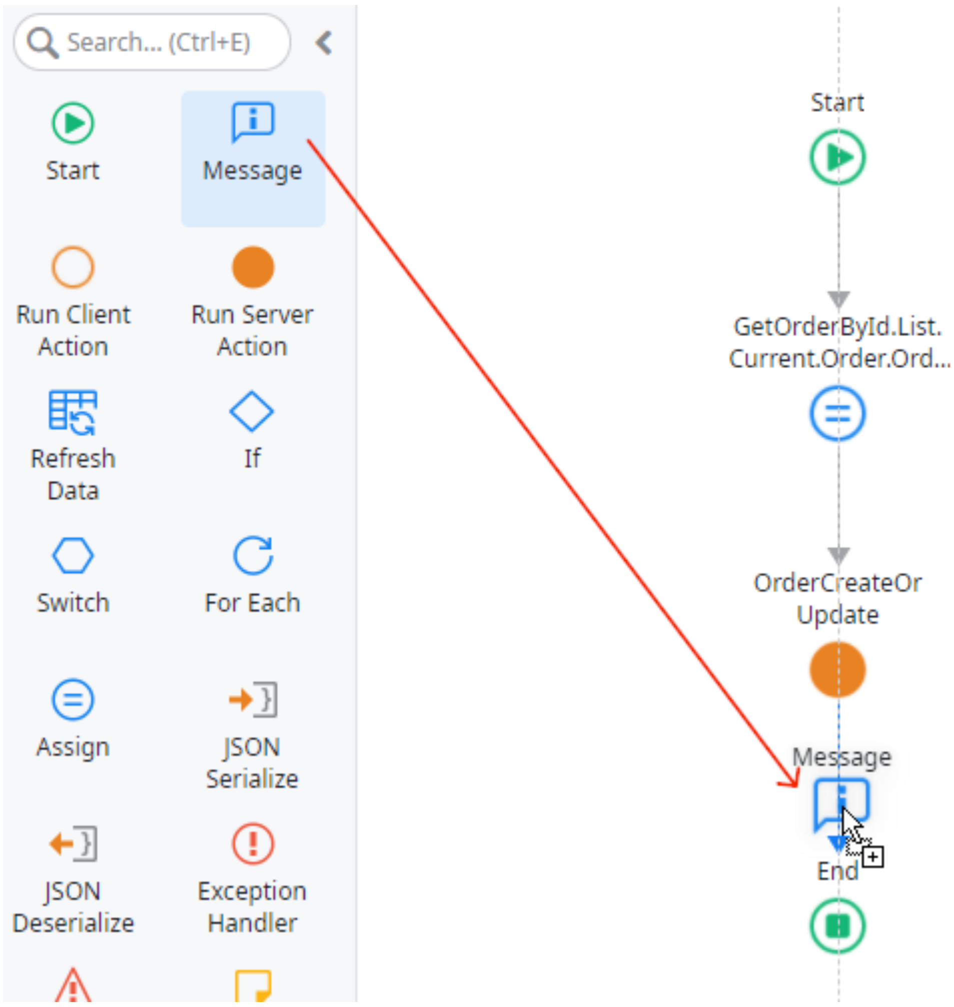


OrderCreateOrUpdate
Run Server Action

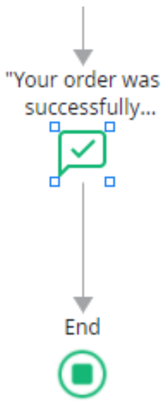
| | |
|---|---|
| Name | OrderCreateOrUpdate |
| Server Request Time... | (Module Default Timeout) |
| Action | Order\OrderCreateOrUpdate |
| Source | |
| (New Argument) | |

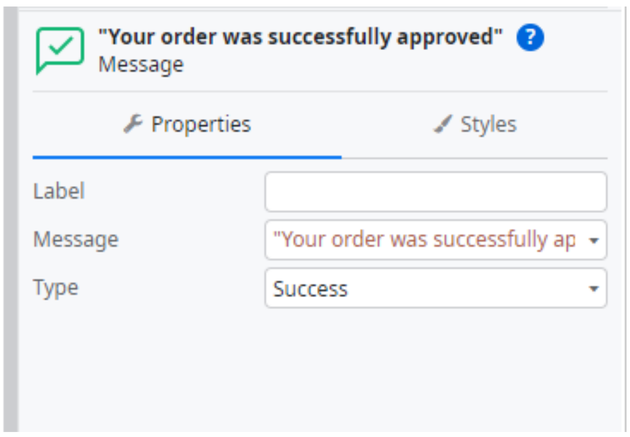
The Source property is used to determine which Order will be modified. So what you just did is to pass the Order that is being edited on the OrderDetail Screen to the Action.

8) Now is time to send a message to the user saying that the order was successfully approved. Drag a **Message** from the left sidebar and drop it on the flow after the OrderCreateOrUpdate Action.



9) Change the **Type** property to **Success** and set the **Message** to *"Your order was successfully approved"*.

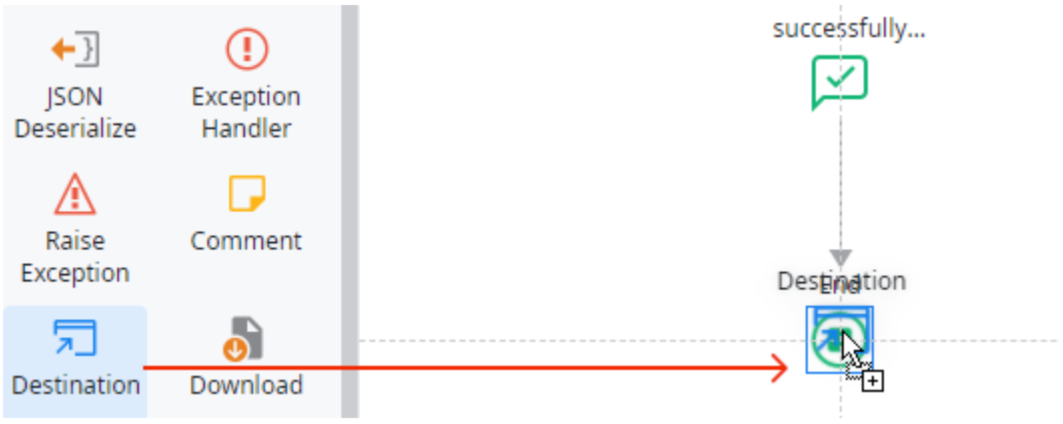




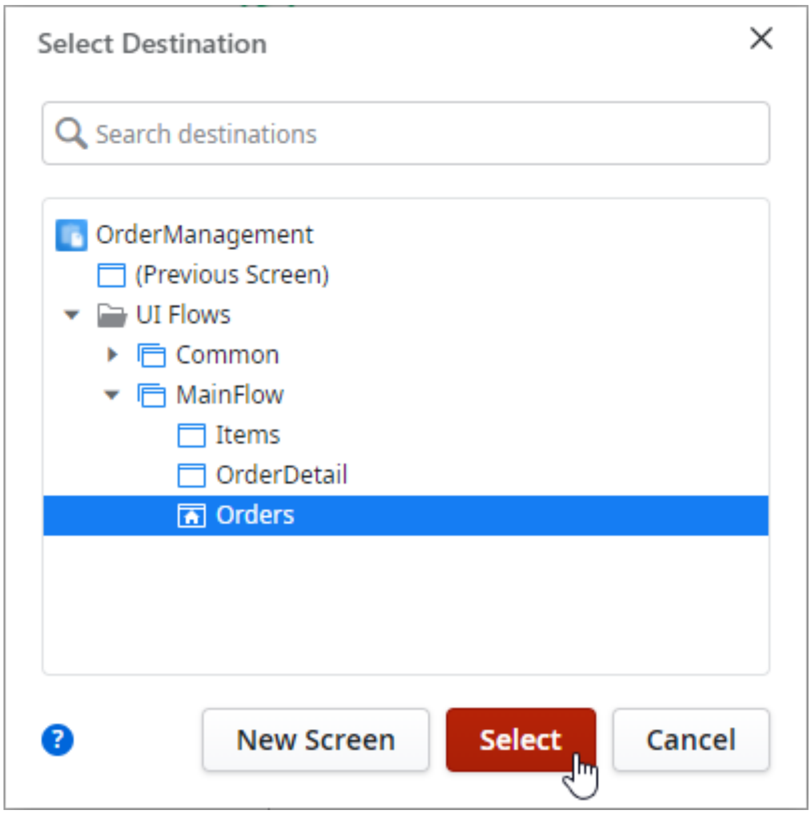
"Your order was successfully approved"
Message

| | |
|---|---|
| Label | |
| Message | "Your order was successfully ap |
| Type | Success |

10) Last but not least, you will redirect the user to the Orders Screen. Drag and drop the *Destination* element **on** the *End* element to replace it.



11) Select the **Orders** Screen in the new dialog. This will make sure that after the order is approved, the user will be back to the Orders Screen.

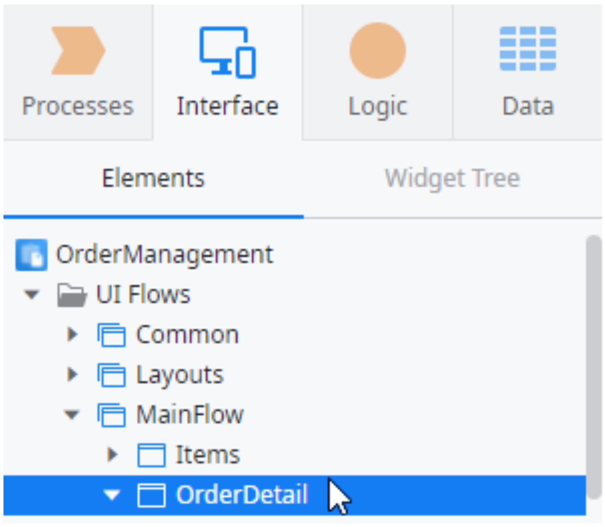


## Reject Orders

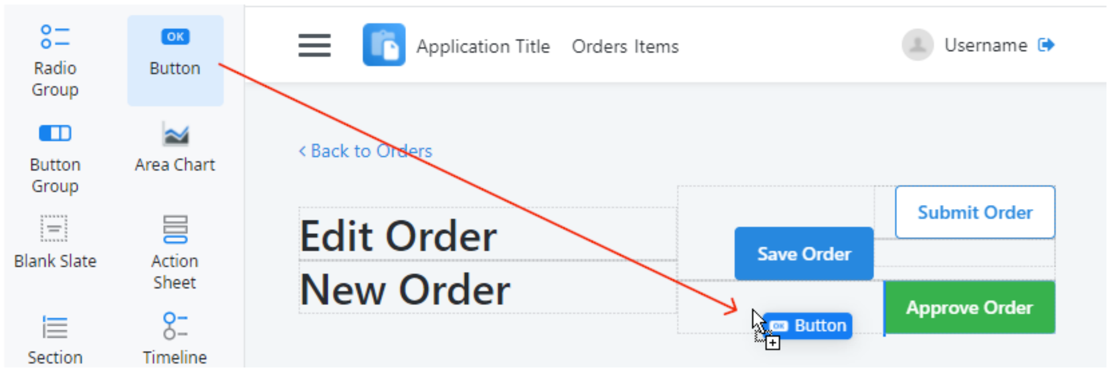
The logic for rejecting orders is very similar. You will need a Button that changes the status of the Order to Rejected when clicked. This behavior will also be defined in a Client Action.

## Reject Button

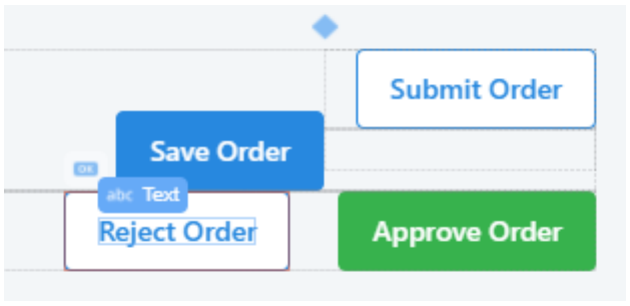
1) Double-click the **OrderDetail** Screen to open it.



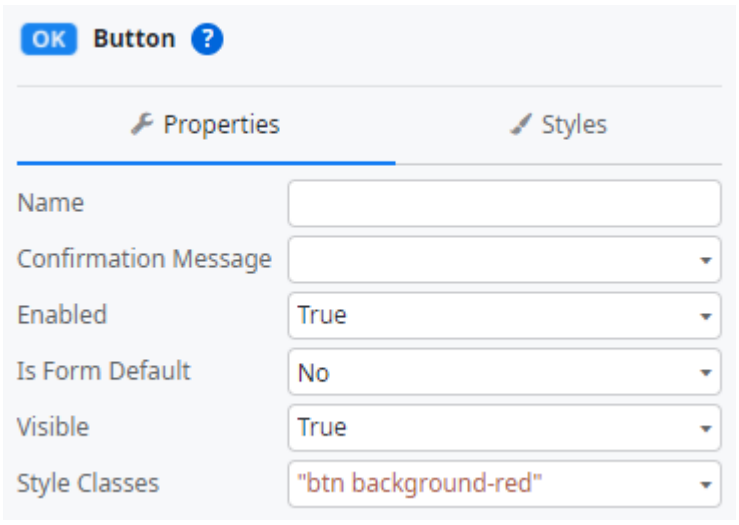
2) Drag a **Button** from the Toolbox and drop it before the Approve Order button.



3) Type the text *Reject Order* in the Button.



4) Click on the Button to see its properties on the right sidebar. Set the **Style Classes** property to *"btn background-red"* to make the Button have a red background.

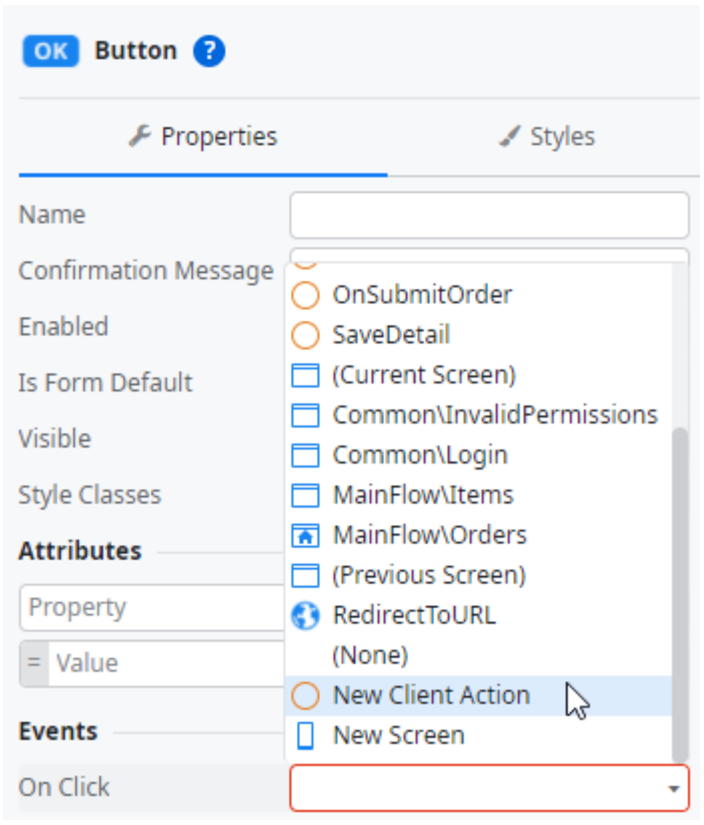


OK Button ?

Properties | Styles

| | |
|---|---|
| Name | |
| Confirmation Message | |
| Enabled | True |
| Is Form Default | No |
| Visible | True |
| Style Classes | "btn background-red" |

5) Still in the Button's properties, set the **On Click** property to **New Client Action**.



OK Button ?

Properties | Styles

Name

Confirmation Message

Enabled

Is Form Default

Visible

Style Classes

**Attributes**

Property

= Value

**Events**

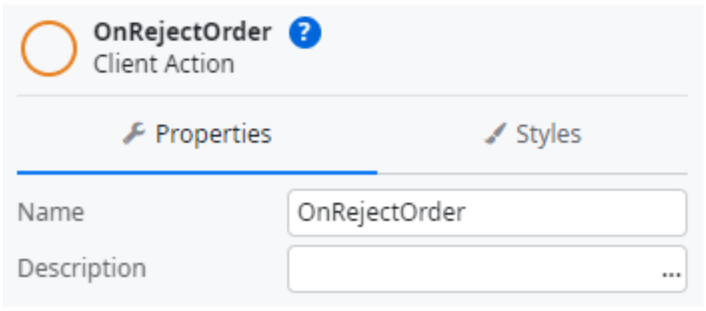
On Click

OnSubmitOrder
SaveDetail
(Current Screen)
Common\InvalidPermissions
Common\Login
MainFlow\Items
MainFlow\Orders
(Previous Screen)
RedirectToURL
(None)
New Client Action
New Screen

A new Client Action is created and Service Studio will automatically open it and name it *RejectOrderOnClick*.

## Logic for Rejecting an Order

1) Select the Action in the right sidebar to see its properties and rename it as *OnRejectOrder*.

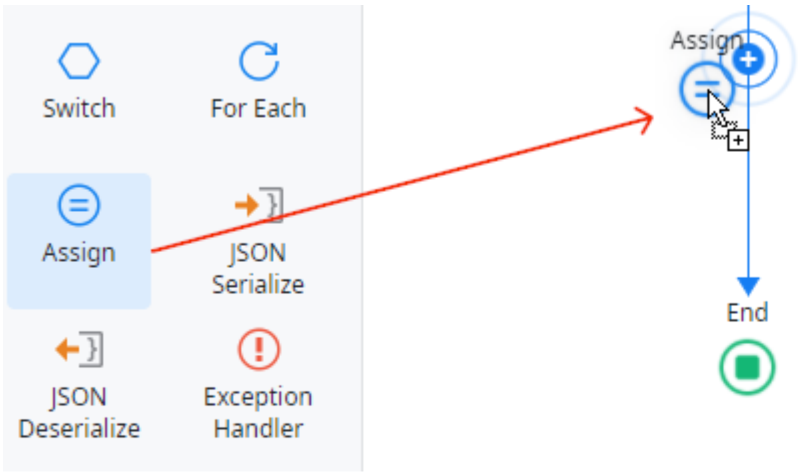


OnRejectOrder
Client Action

Properties | Styles

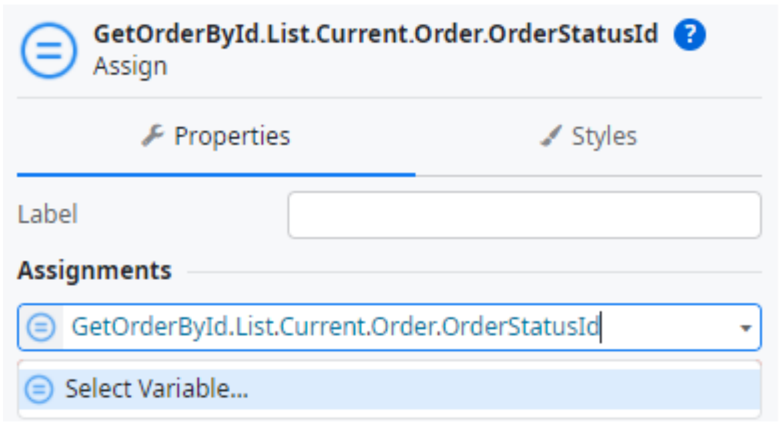
| | |
|---|---|
| Name | OnRejectOrder |
| Description | |

2) Inside the Action, drag an **Assign** from the left sidebar and drop it on the Action flow.



In this case, the goal is to change the Order Status to Rejected.

3) Set the **Variable** field of the Assign to:
`GetOrderById.List.Current.Order.OrderStatusId`



GetOrderById.List.Current.Order.OrderStatusId
Assign

Properties | Styles

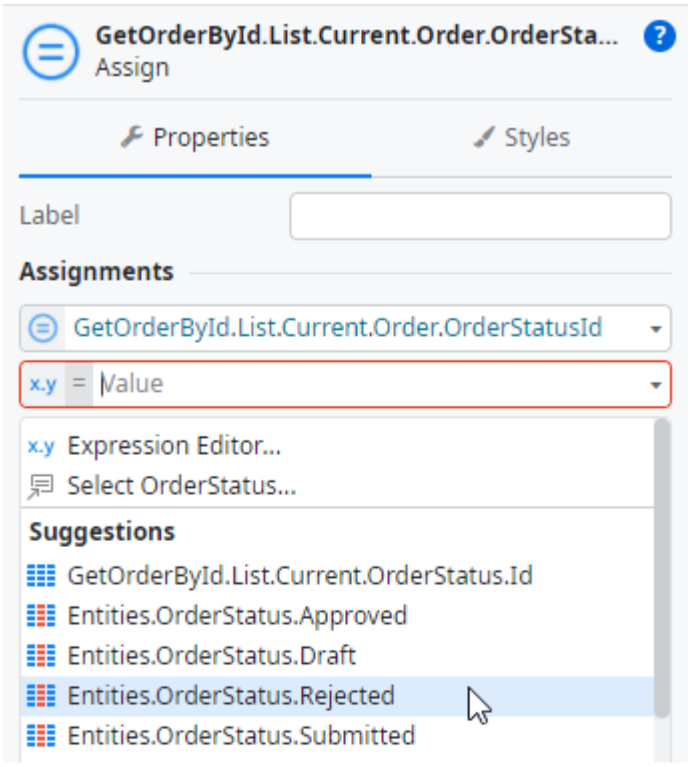
Label

**Assignments**

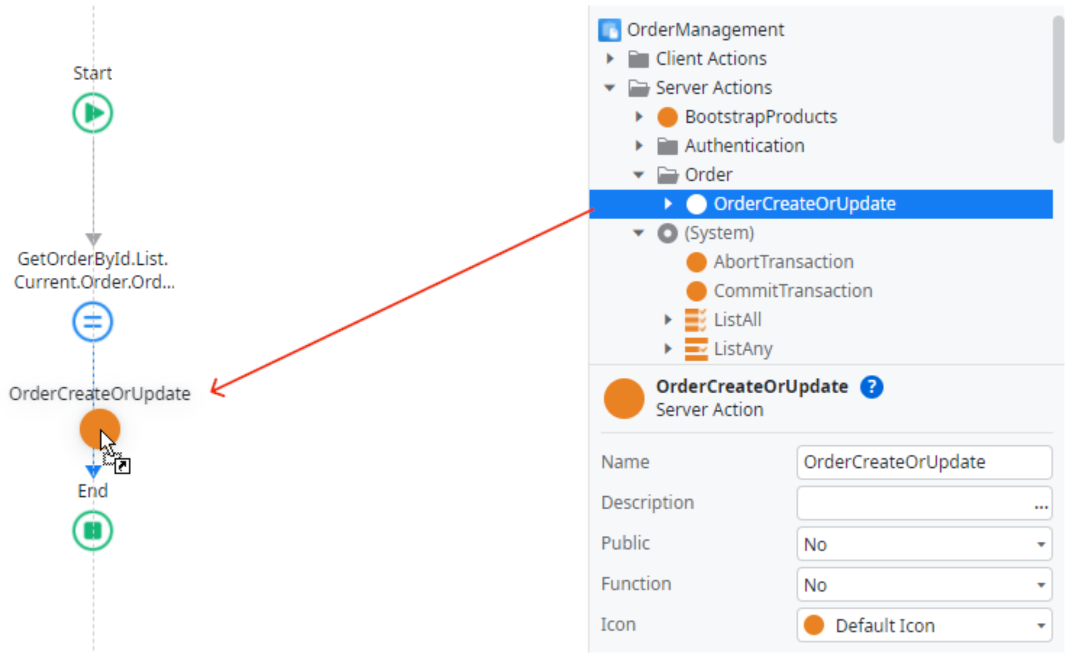
GetOrderById.List.Current.Order.OrderStatusId

Select Variable...

4) Set the **Value** field to *Entities.OrderStatus.Rejected*.


GetOrderById.List.Current.Order.OrderSta...
Assign
Properties
Styles
Label
Assignments
GetOrderById.List.Current.Order.OrderStatusId
x.y = Value
Expression Editor...
Select OrderStatus...
Suggestions
GetOrderById.List.Current.OrderStatus.Id
Entities.OrderStatus.Approved
Entities.OrderStatus.Draft
Entities.OrderStatus.Rejected
Entities.OrderStatus.Submitted

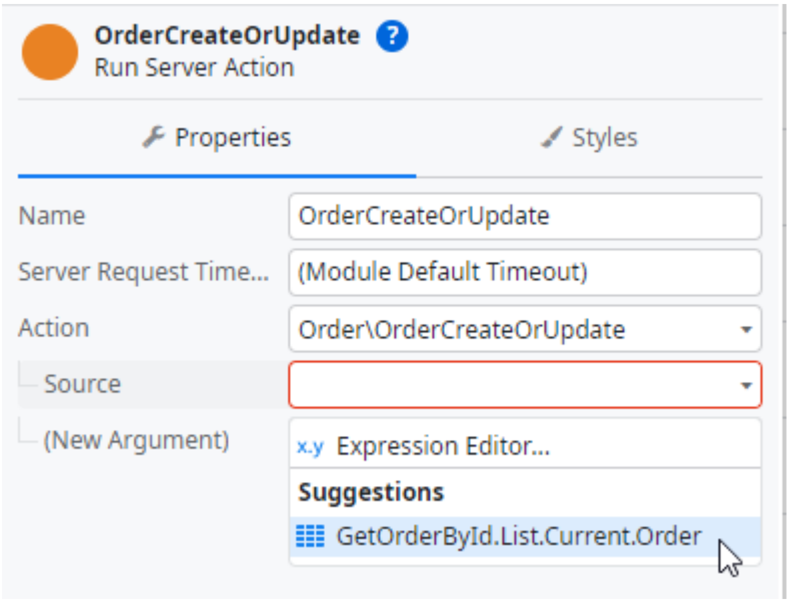
5) Switch to the Logic tab, then open the Server Actions folder and then the Order folder. You will see the **OrderCreateOrUpdate** Action. Drag and drop it after the Assign in the flow.



| | |
|---|---|
| Name | OrderCreateOrUpdate |
| Description | ... |
| Public | No |
| Function | No |
| Icon | Default Icon |

You will use this Action again to update the order in the database.

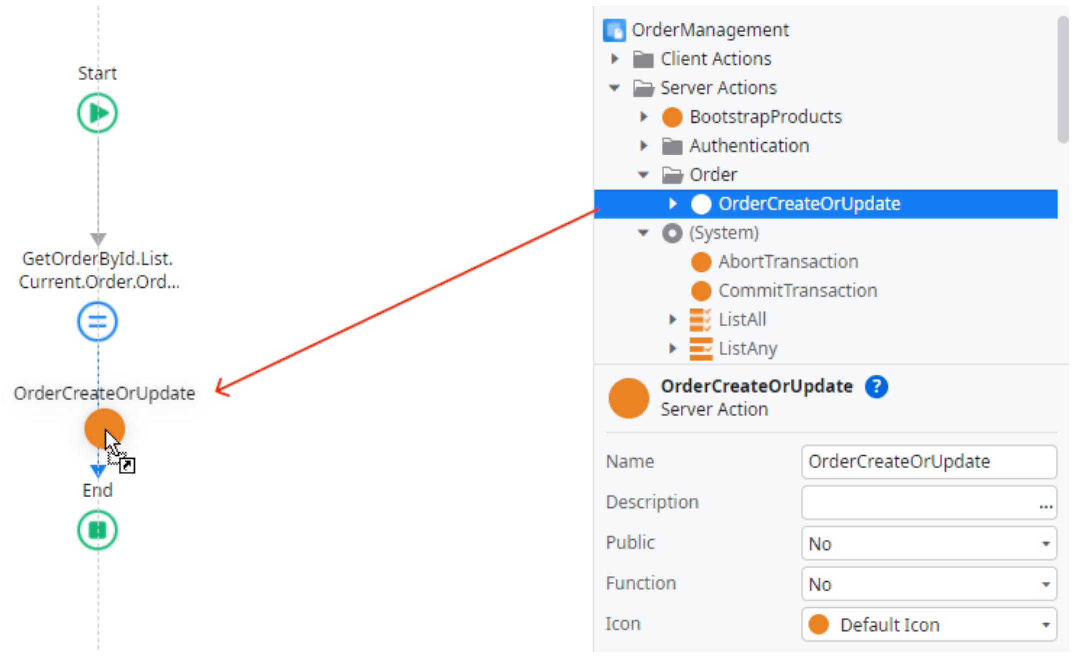
6) Set its **Source** property to `GetOrderById.List.Current.Order`.



OrderCreateOrUpdate
Run Server Action

| | |
|---|---|
| Name | OrderCreateOrUpdate |
| Server Request Time... | (Module Default Timeout) |
| Action | Order\OrderCreateOrUpdate |
| Source | |
| (New Argument) | |

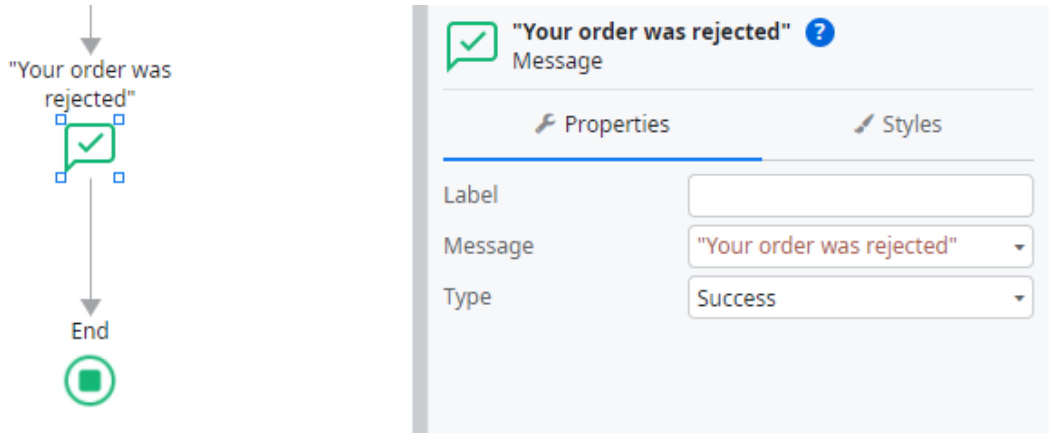
7) Drag and drop a Message element under the OrderCreateOrUpdate Action in the flow.



OrderCreateOrUpdate
Server Action

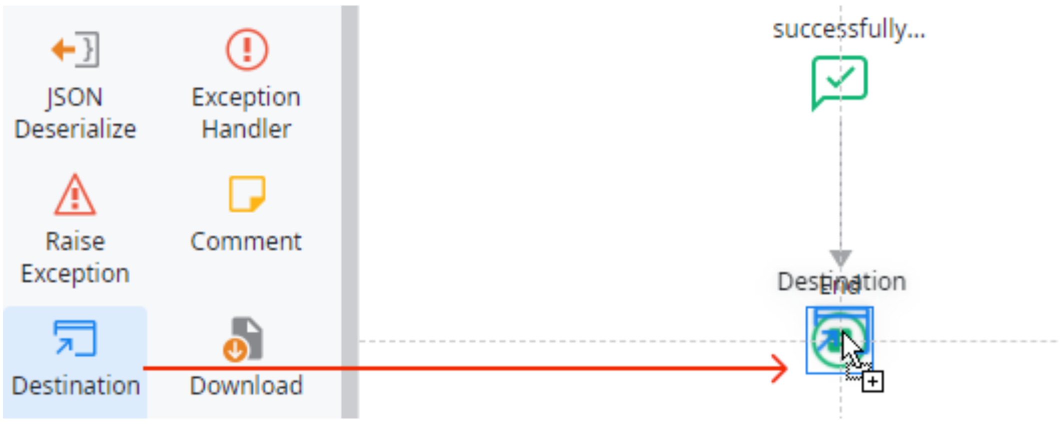
| | |
|---|---|
| Name | OrderCreateOrUpdate |
| Description | ... |
| Public | No |
| Function | No |
| Icon | Default Icon |

8) Change the Type property to **Success** and add the message *"Your order was rejected"*.

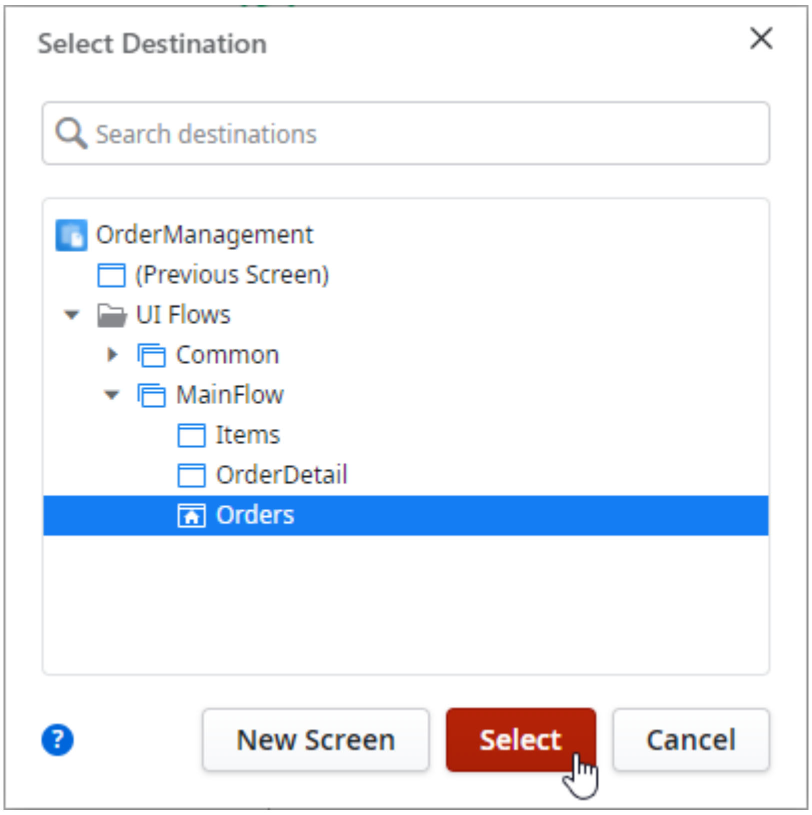


Label:
Message: "Your order was rejected"
Type: Success

9) Drag the **Destination** element from the left sidebar and drop it **on** the End element to replace it.



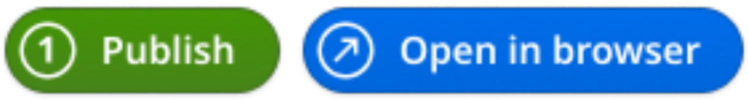
10) Select the **Orders** Screen in the new dialog. Just like with the Approve Order Button, here the user will navigate to the Orders Screen when the order is rejected.



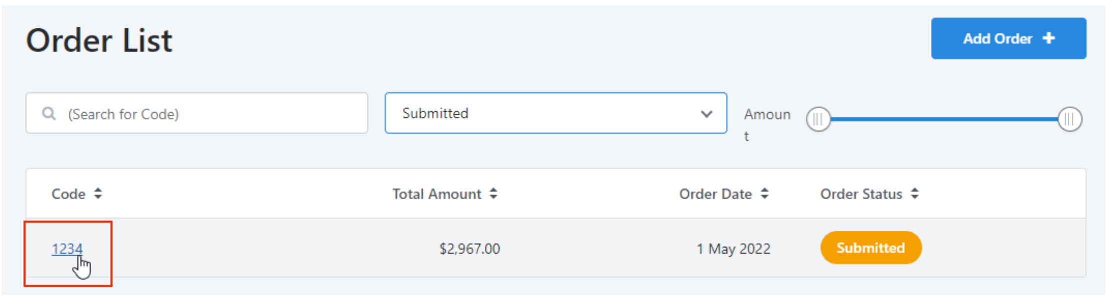
## Testing a Submitted Order

Ok, let's test the app to see if it works as expected.

1) Publish the module and open it in the browser.

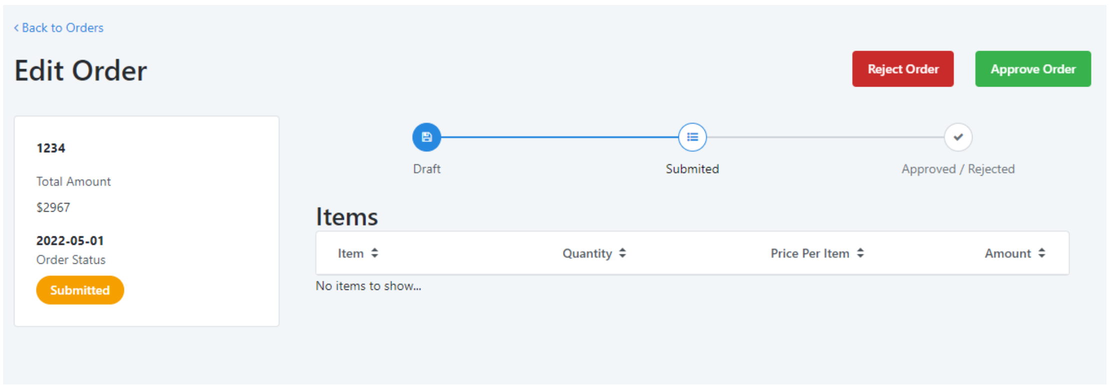


2) Click on a Submitted Order to open its Detail Screen.



| Code | Total Amount | Order Date | Order Status |
|---|---|---|---|
| 1234 | $2,967.00 | 1 May 2022 | Submitted |

The Approve and Reject Buttons should appear on the top right corner.



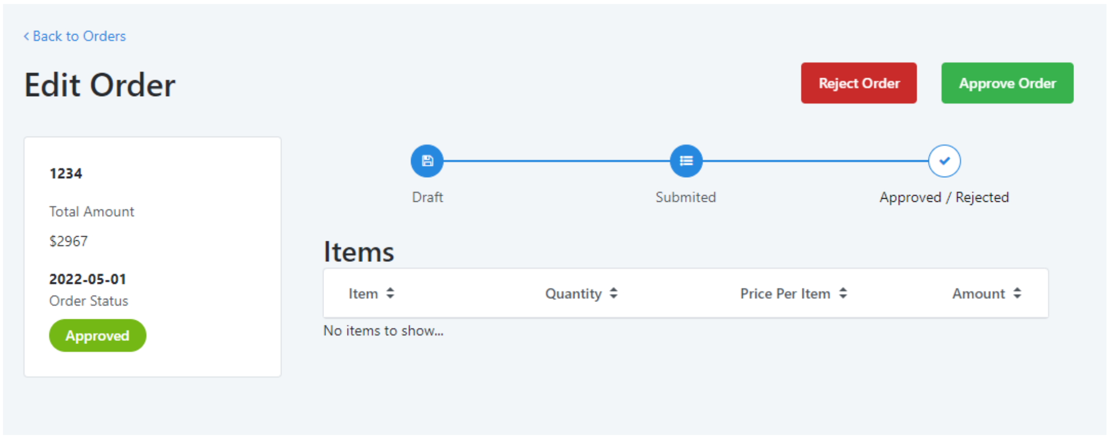
3) Approve or Reject the Order to see if the status has changed and if you are redirected to the Orders Screen.

## Controlling the Visibility of the Approve and Reject Buttons

Nice! You did it! The workflow of the wizard already considers all the statuses.

But wait a minute... what about orders with other statuses? Should the new Buttons also be visible for them? The answer is **no**! So, let's test this too!

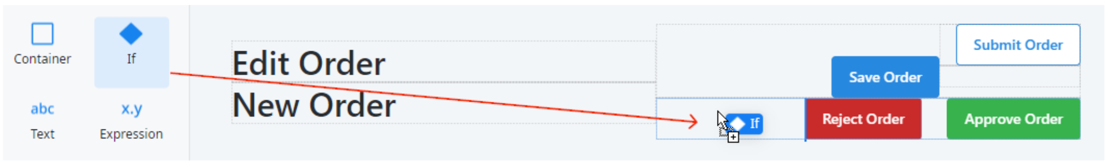
1) Still in the browser, open an order with an Approved or Rejected Status.



The Buttons are visible, but they should **not** be! So, we need to fix that. How? With our good old friend If.

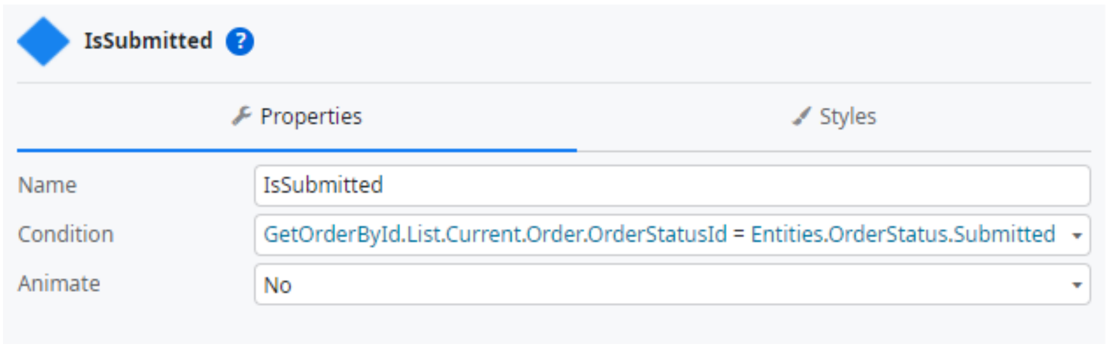
2) Go back to Service Studio and open the **OrderDetail** Screen again.

3) Drag an **If** from the Toolbox and drop it next to the Reject Order Button.



4) Change the **Name** of the If to *IsSubmitted* and the **Condition** to:

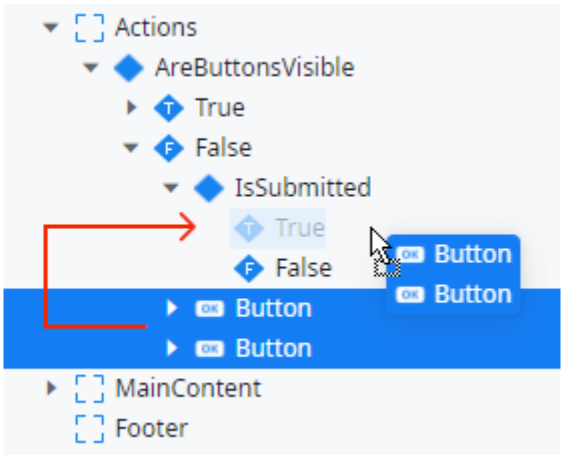
```
GetOrderById.List.Current.Order.OrderStatusId =
Entities.OrderStatus.Submitted
```



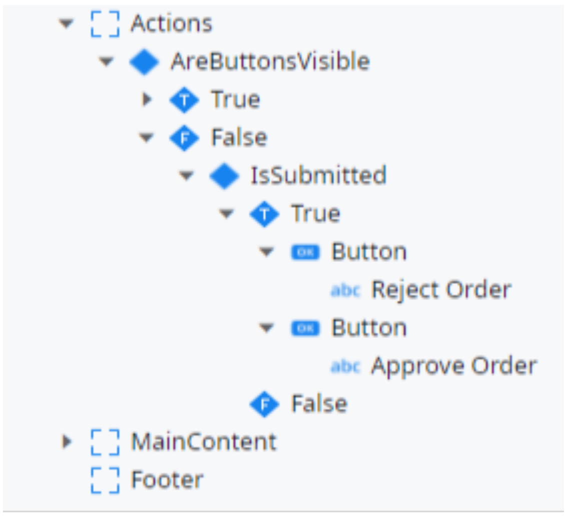
| | |
|---|---|
| Name | IsSubmitted |
| Condition | GetOrderById.List.Current.Order.OrderStatusId = Entities.OrderStatus.Submitted |
| Animate | No |

This means these Buttons will only appear if the order is in the status Submitted.

5) Open the Widget Tree, then select the Reject and Approve Buttons. Drag them to the **True** Branch of the **IsSubmitted** If.



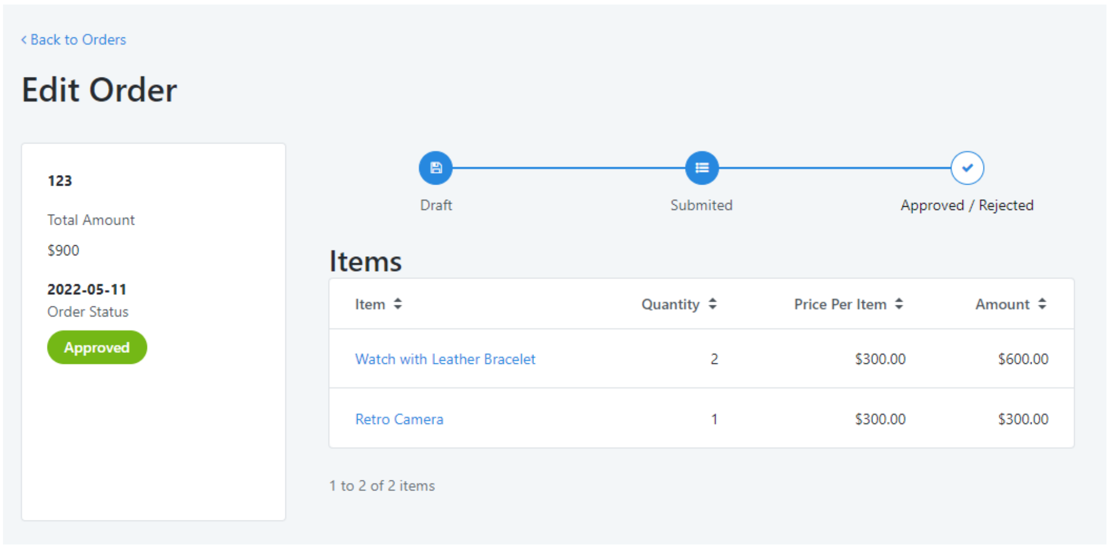
Your Widget Tree should look like this:



This is just another way of creating your Ifs.

6) Publish the module and open it in the browser.

7) Click on an Approved or Rejected Order. The Buttons should not appear any more!



‹ Back to Orders

Edit Order

123

Total Amount

$900

2022-05-11

Order Status

Approved

Draft — Submited — Approved / Rejected

Items

| Item | Quantity | Price Per Item | Amount |
|---|---|---|---|
| Watch with Leather Bracelet | 2 | $300.00 | $600.00 |
| Retro Camera | 1 | $300.00 | $300.00 |

1 to 2 of 2 items

## 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) Assign

2) Client Actions

3) Logic Course

**See you in the next tutorial!**