# Master/Detail Pages and Microflows

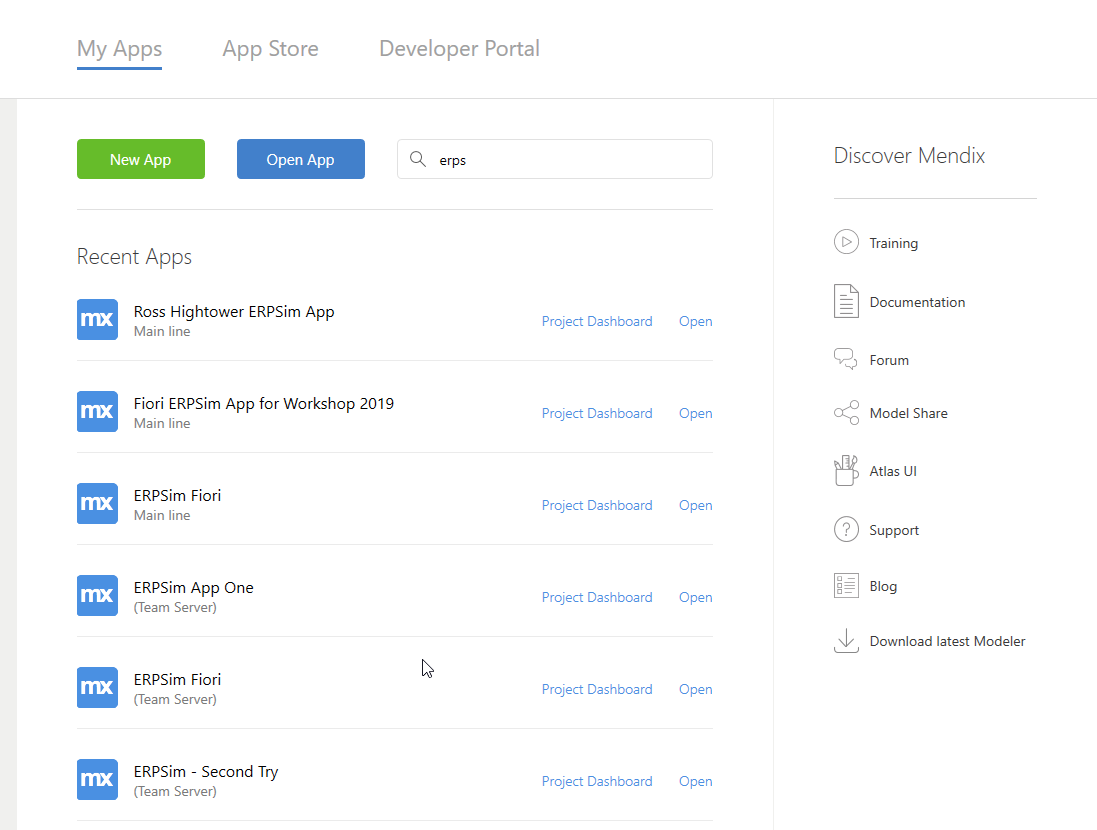
This case is the second in the series of introduction cases. In this case, you will a sales order master/detail page and create a simple microflow.

### Prerequisites

This case was prepared using version 7.21.0 of the desktop Mendix Modeler. You must have completed Introduction to Mendix case 1- Creating Apps, adding entities and creating pages before starting this case.

## Open the Introductory Project

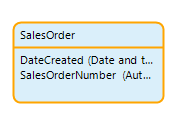
Open the project you created in case 1 of the introductory series in the desktop modeler. You can click it on the list of Recent Apps or use the search box to find it.



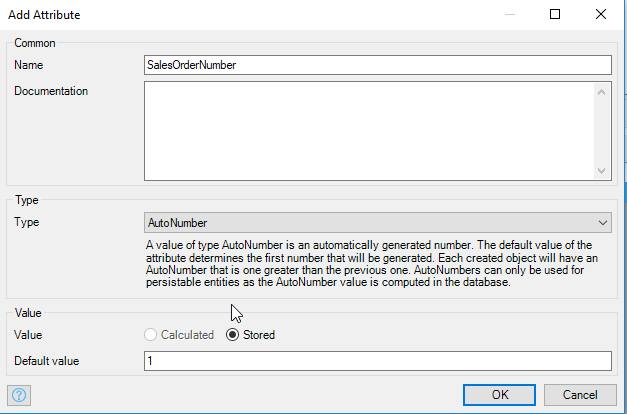
## Adding a Sales Order Page

### Create the Sales Order Entity

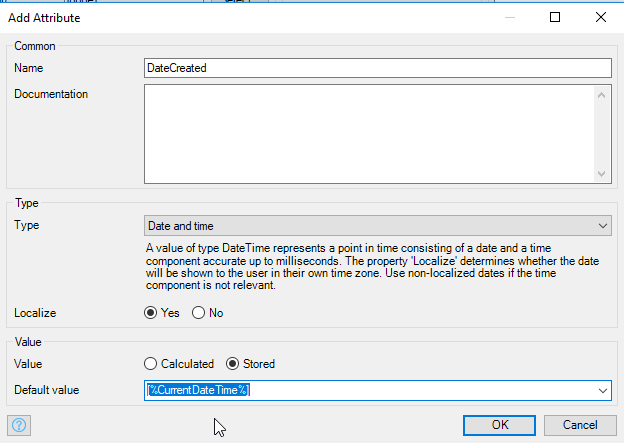
The SalesOrder entity has only two attributes as shown below.



The SalesOrderNumber is an AutoNumber type.

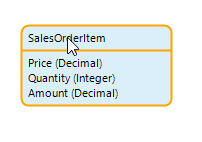


DateCreated is a Date and time type. You also want to select the Default value as [%CurrentDateTime%]. This is one of many system provided parameters that can be used in a project.



### Create the Sales Order Item Entity

The SalesOrderItem entity has three attributes.

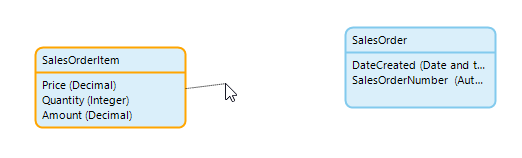


### Create Associations

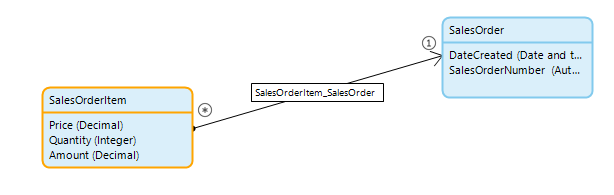
Notice that we did not add a Product attribute to SalesOrderItem or a Customer attribute to SalesOrder. That’s because Mendix handles the keys (primary and foreign) automatically, behind the scenes. We just need to configure the associations in the Domain Model. We’ll need three associations:

* One-to-Many association between SalesOrderItem and SalesOrder
* One-to-One association between Product and SalesOrderItem
* One-to-Many association between Customer and SalesOrder

To create an association, you start with the entity on the many side of the association (if it is a one-to-one association, you can start with either side and edit it after you have created it). To create the association between SalesOrderItem and SalesOrder, hover over the border of the SalesOrderItem entity then click and drag….

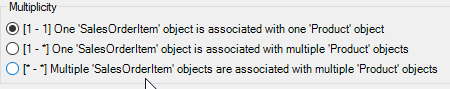


…and drop on the border of SalesOrder to create the association.

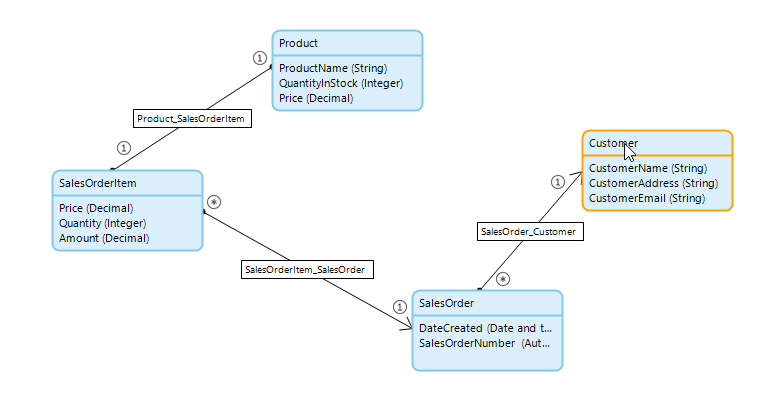


Note that the association has a name.

Now create the other two associations. To create the one-to-one association, create it as a many-to-one association then double-click the association to configure it as a one-to-one association.



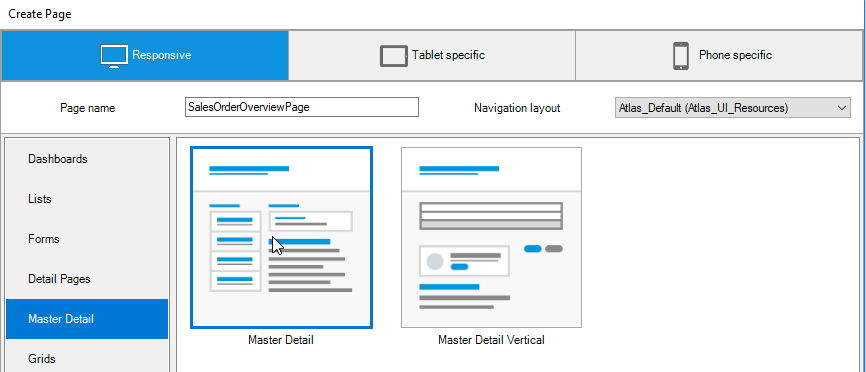
The final Domain Model looks like this:



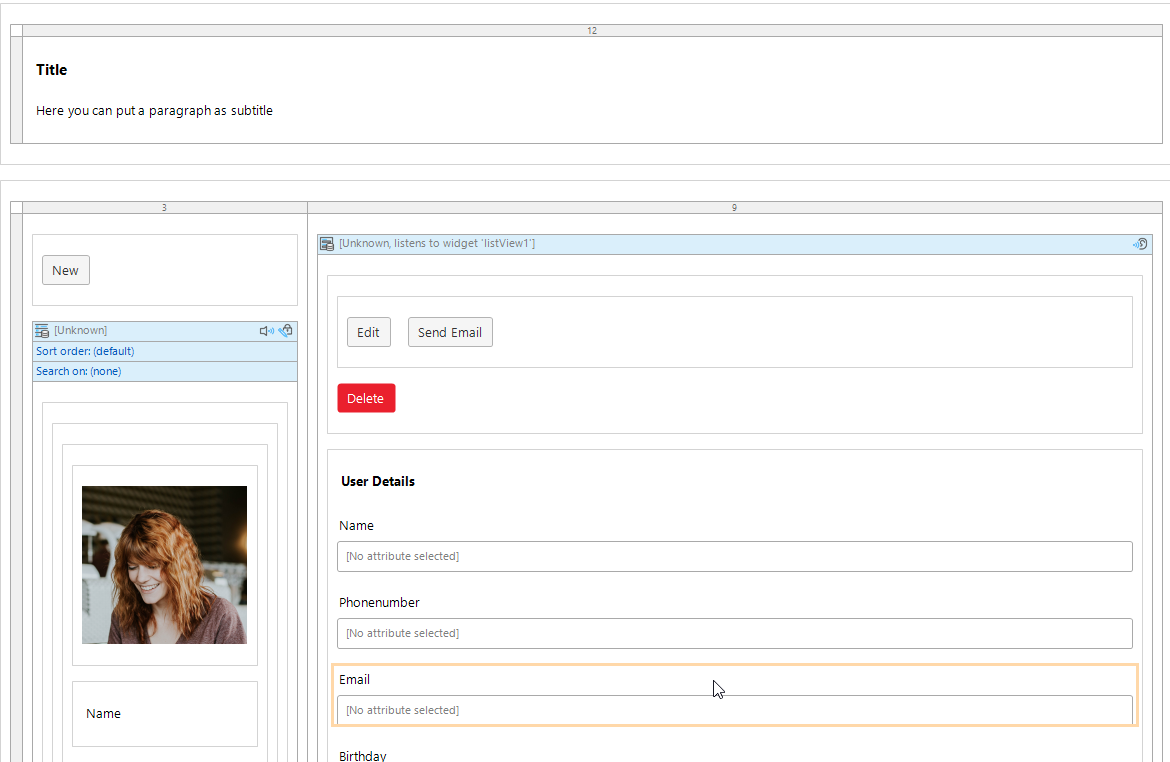
### Create the Sales Order Overview Page

The process for creating the SalesOrderOverview page is similar to the process used in the previous case. However, this page is a Master/Detail page that shows a list of sales orders and the items for the selected sales order.

Add a new page to the Pages folder. Configure it as shown in the image below.

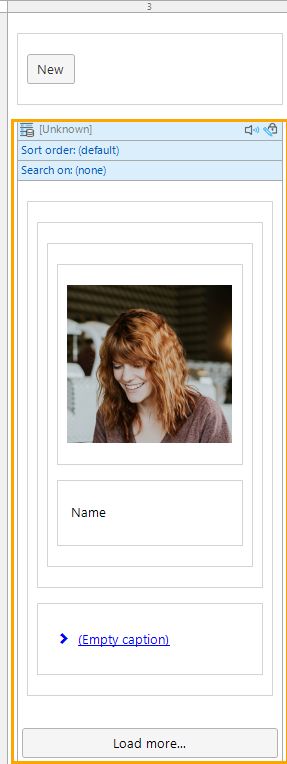


The page is created with some default content that we need to change.

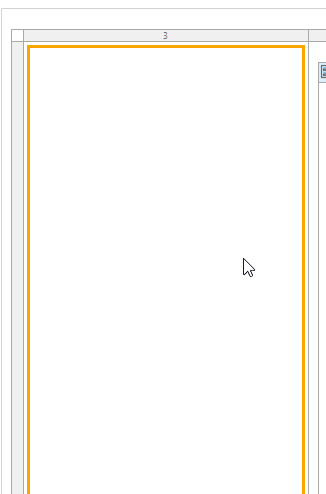


#### Replace the Listview

The master side of the page uses a Listview widget.



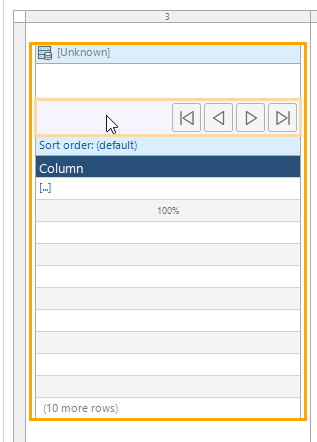
A Listview is similar to a Data grid but allows a wider range of formatting options. Since our requirements are fairly simple, we’ll replace the Listview with a Data grid. Delete the Listview widget and the Container widget that includes the New button.



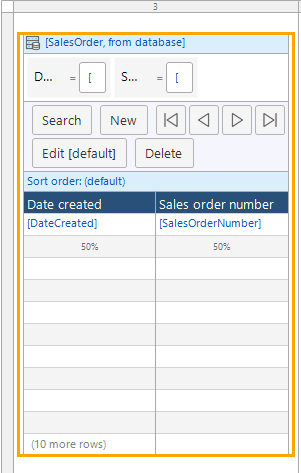
Next, click the Data grid item above the editor…



Then click where the Listview was previously.

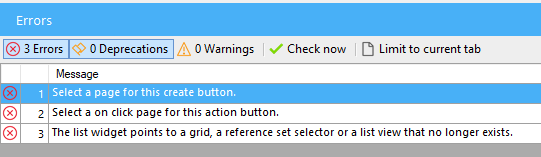


Set the data source to the SalesOrder entity.



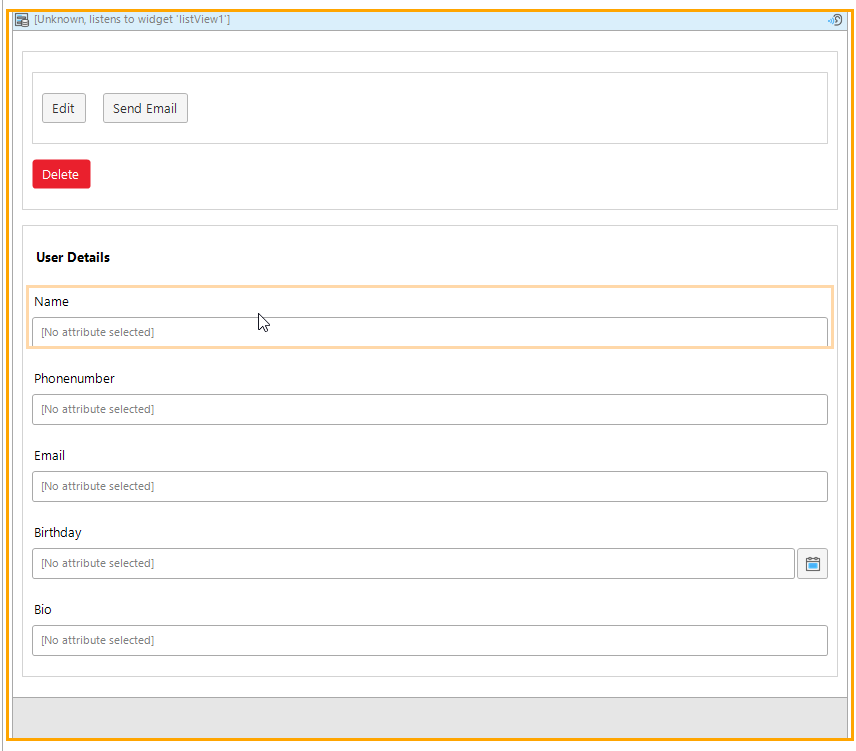
You can drag the Date created column to the right of the Sales order number column.

This introduced a number of errors, but we’ll fix them soon.

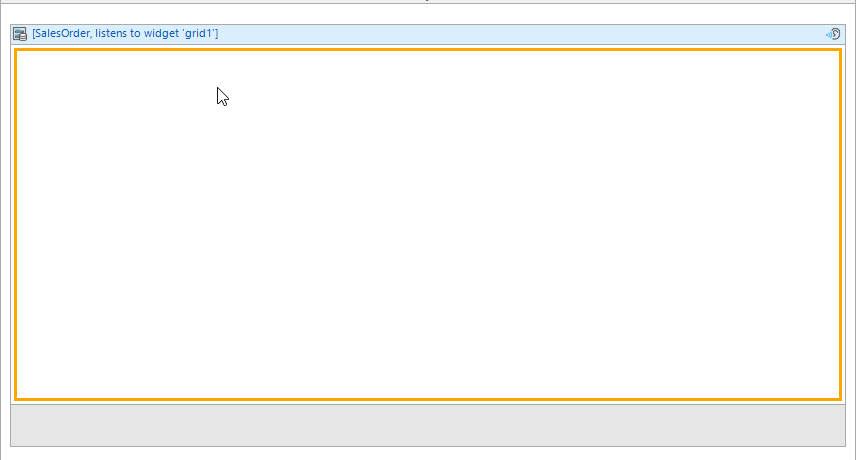


#### Update the Detail Side of the Page

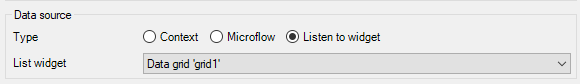
The detail side of the page consists of a Data view widget. A Data view is similar to a Data grid except it is designed to show a single object.



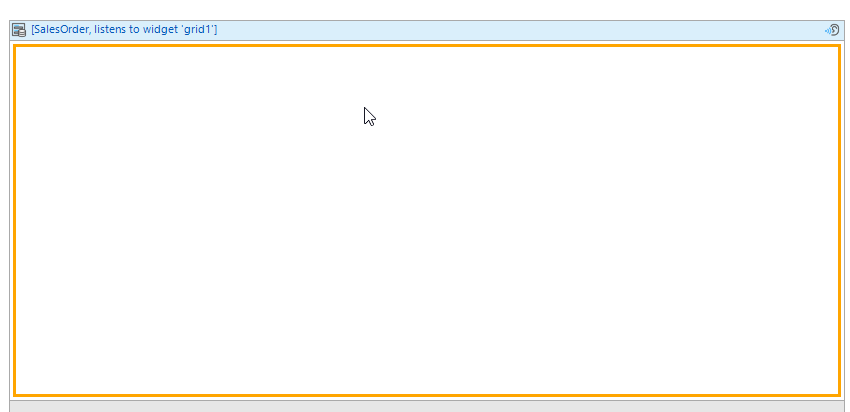
Delete everything inside the Data view.



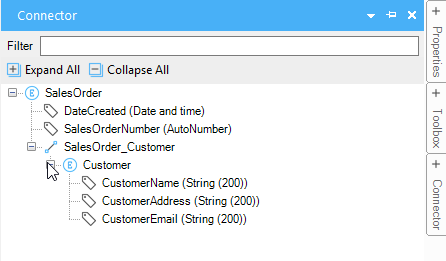
Double-click the header of the Data view to edit the Data source. The Listen to widget option means that the Data view will listen to the Data grid on the master side and bind to whichever sales order object is selected by the user. When prompted to automatically fill the contents of the Data view, choose No.



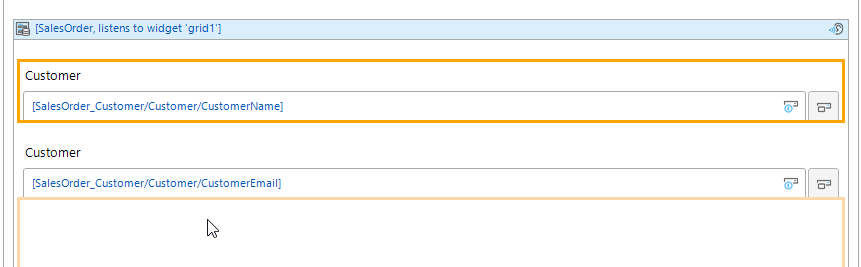
Make sure the Data view is selected…



Open the Connector tab on the right side of the modeler. This shows all the attributes available in the context of the Data view. There are the two attributes of the SalesOrder entity available of course. However, also listed are the attributes of the Customer entity which are available through the association between SalesOrder and Customer.

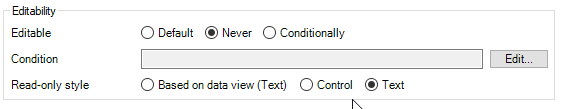


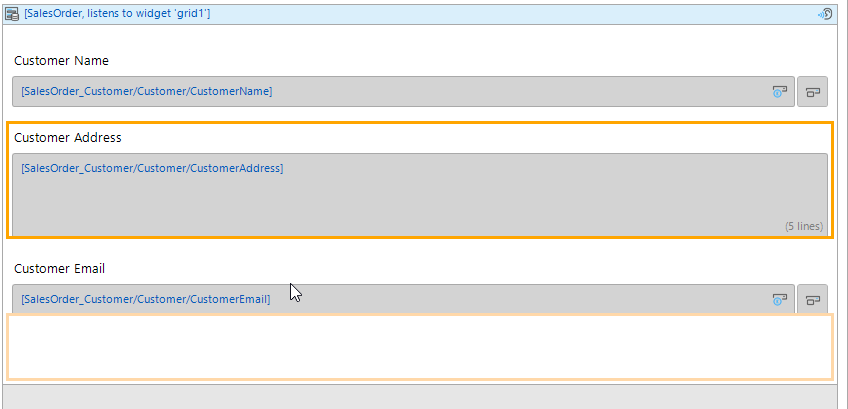
Drag the CustomerName and CustomerEmail attributes onto the Data view.



Also, use the Toolbox to drag a Text area widget and configure its data source as the CustomerAddress attribute.

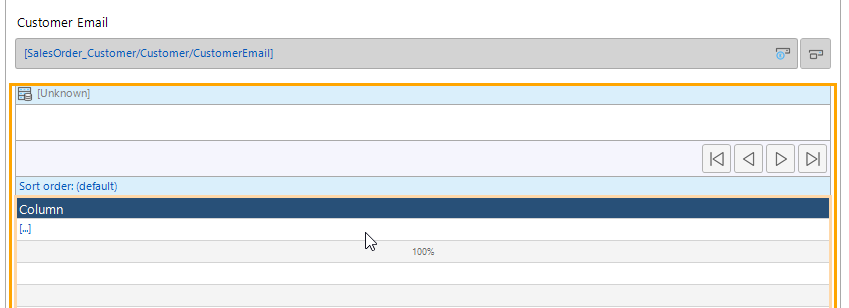
We shouldn’t be able to edit these values so edit their properties so they are never editable and the display style is Text. Also update the captions.



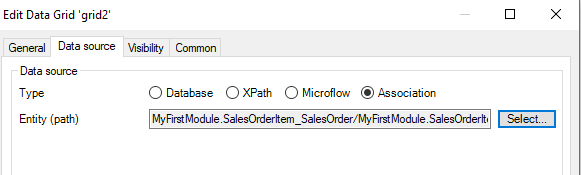


#### Add the SalesOrderItem Data grid

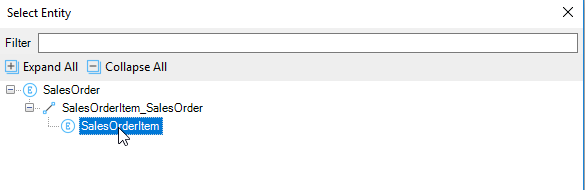
Add a Data grid below the customer widgets. Make sure it’s inside the Data view.



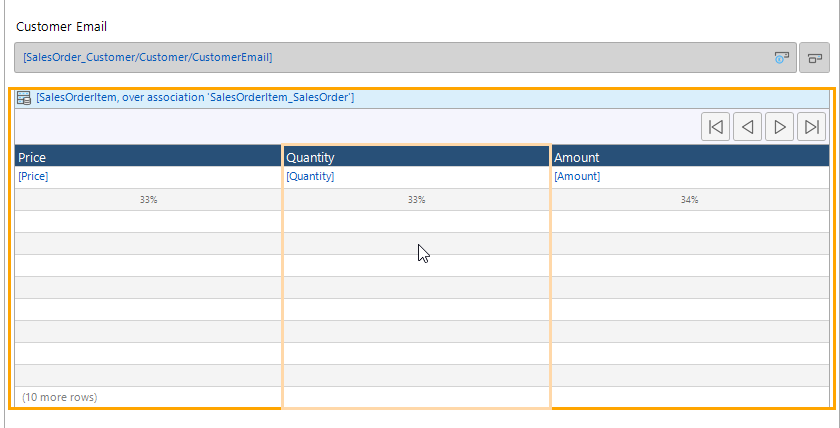
Change the data source to SalesOrderItem using the association between SalesOrder and SalesOrderItem.



When you click the Select… button you can navigate to the SalesOrderItem entity as shown below.

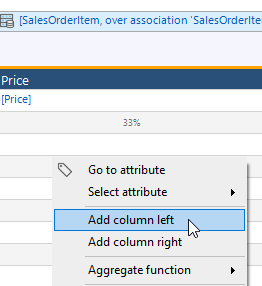


Choose yes to allow the modeler to fill the contents of the Data grid.

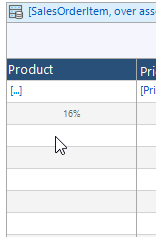


#### Add the Product Column

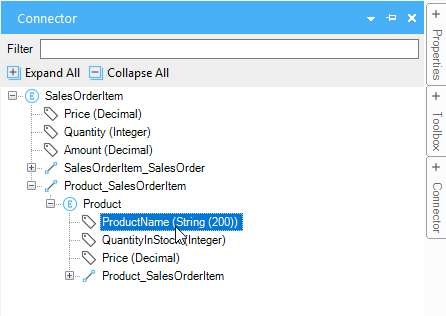
The one missing element of the sales order item Data grid is the product. Select the left-most column then right-click it and select Add column left…



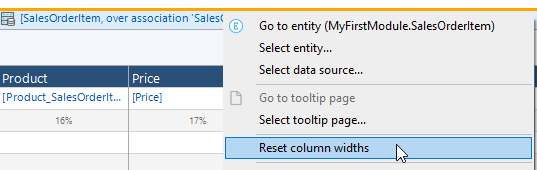
Select the new column and type Product to change the header.



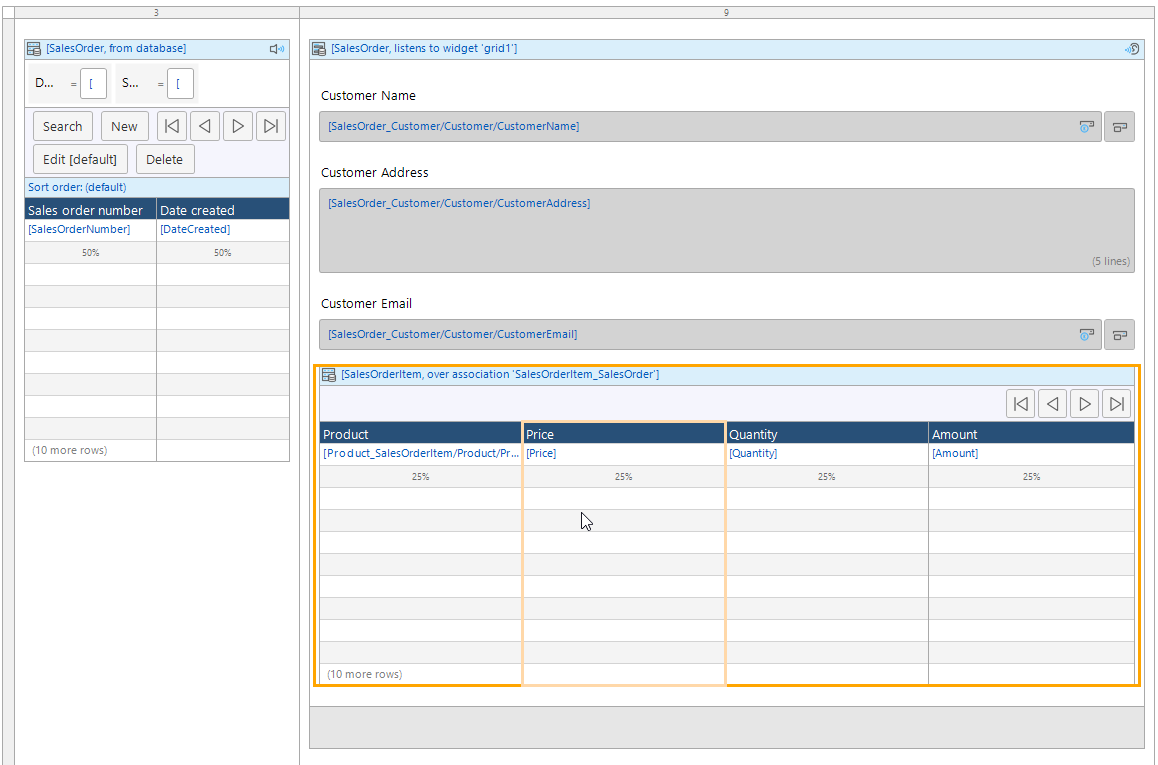
Open the Connector tab and drag the ProductName attribute onto the column.



You can equalize the widths of the columns by right-clicking the header of the Data grid and choosing Reset column widths.

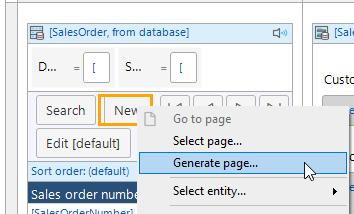


The final page looks like this.

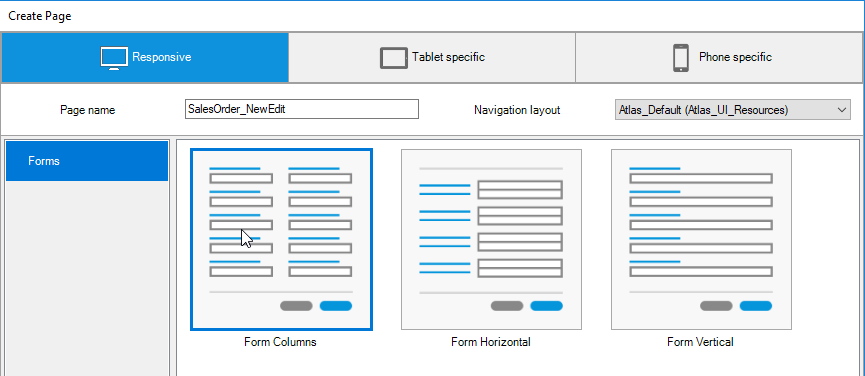


### Create the Sales Order New/Edit Page

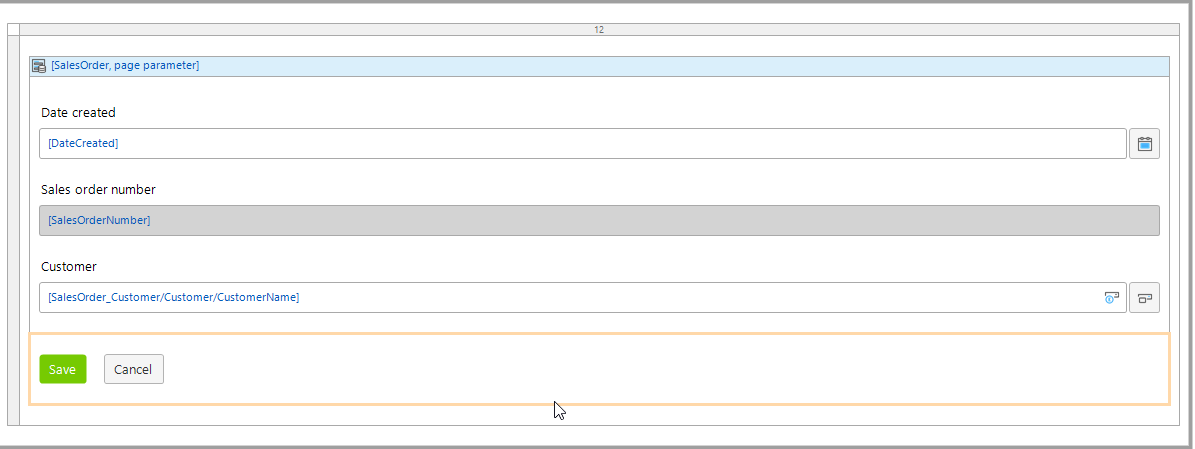
Right-click the New button on the SalesOrder Data grid and select Generate page.



Configure the new page as shown. Note that previous NewEdit pages that we created were popup pages. Make sure this page is not a popup page.



The new page looks like the image below initially. It includes a Data view that is bound to the SalesOrder that was created automatically when the page was opened.

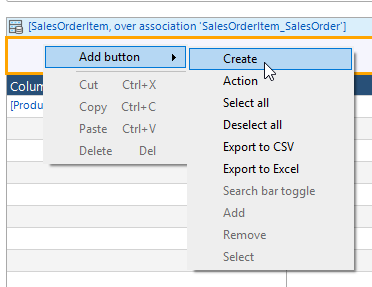


Change the Date created widget so that it is read only. Change both the Date created and Sales order number widgets so they display as Text.

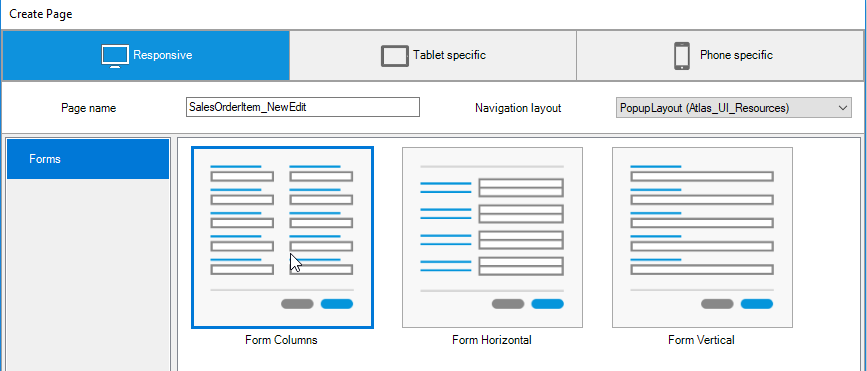
Just as you did with the SalesOrderOverview page, add a Data grid below the Customer field and configure it to display SalesOrderItems using the association with SalesOrder.

### Create the Sales Order Item New/Edit Page

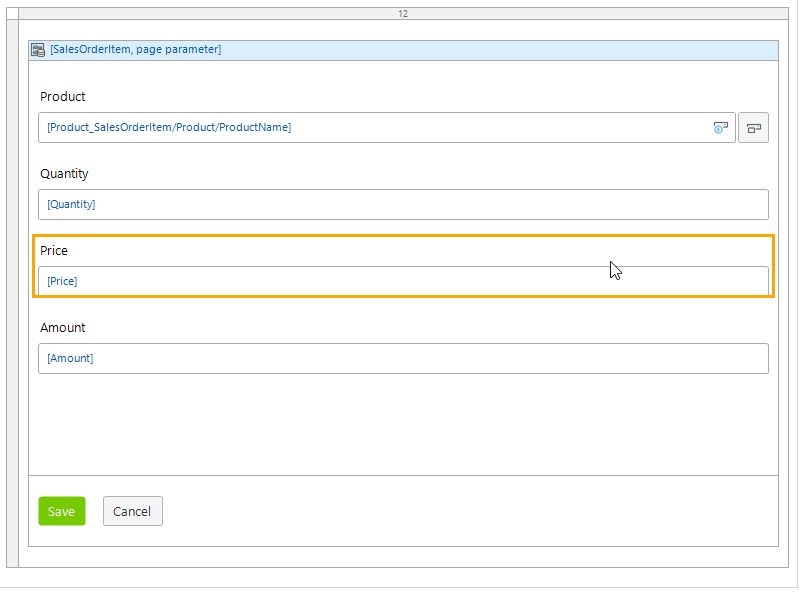
Right-click in the space below the header in the sales order item Data grid and select to add a Create button.



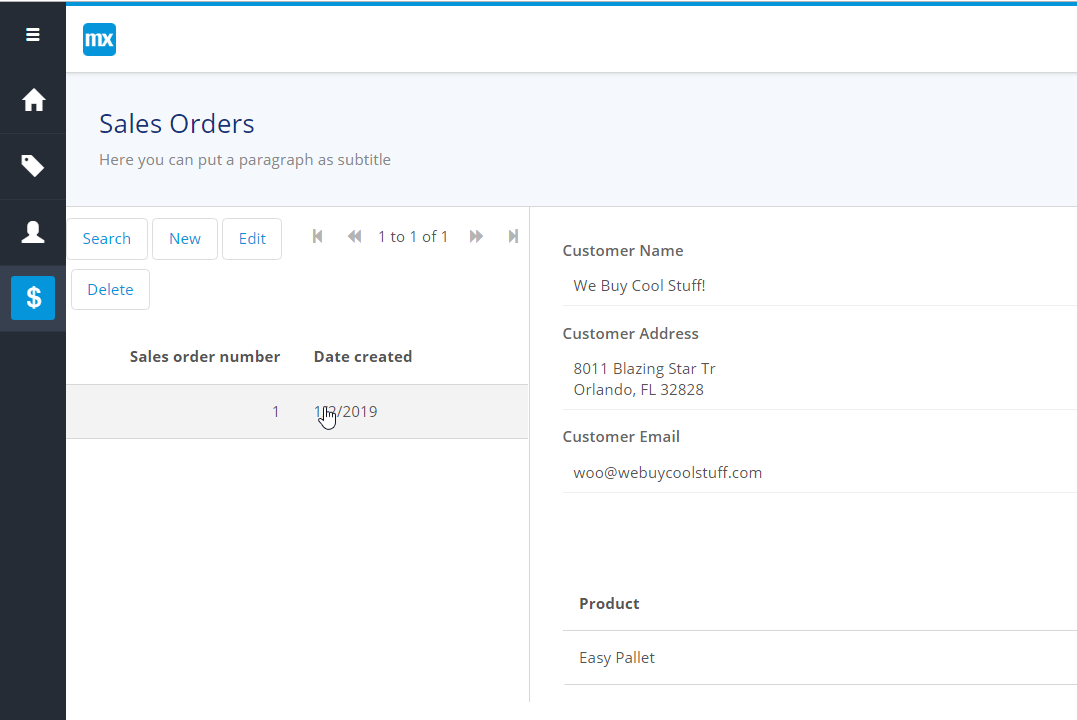
Generate a page for the new button and configure the page as shown below.



Rearrange the widgets on the new page as shown below.

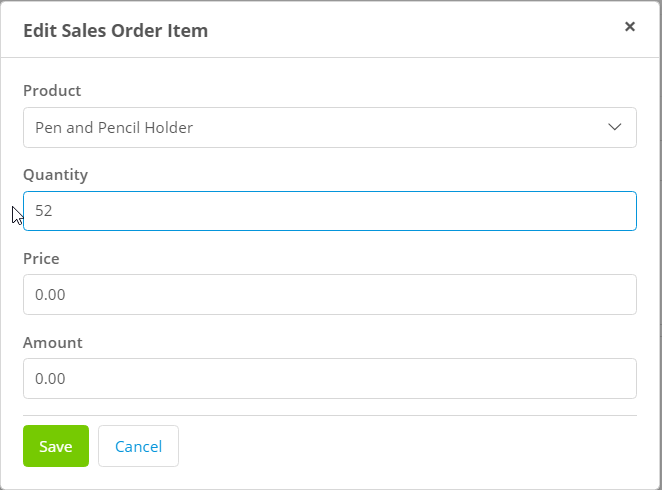


Now add the SalesOrderOverviewPage to the navigation menu and run the app. You can create sales orders.



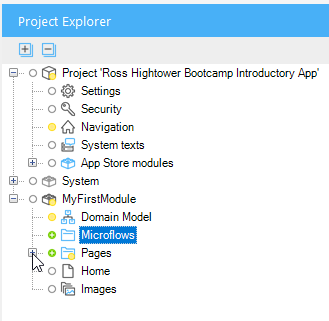
### Create the Price Lookup Microflow

One problem with the app so far is that the user is forced to enter the price and amount when entering a sales order item.

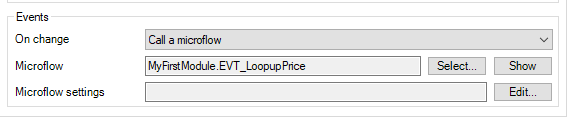


The price is stored with the product, so it would be nice if the app would fill in the price and amount automatically when the user selected a product or changed the quantity. We can add logic like this to the app using Microflows.

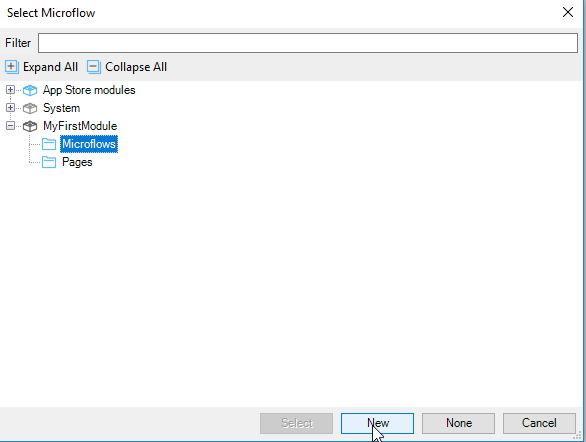
Create a folder called Microflows in the MyFirstModule module in the Project Explorer.



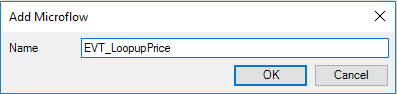
Edit the properties of the Product widget on the SalesOrderItem\_NewEdit page. Change the On change field in the Events section to Call a microflow. Click Select…



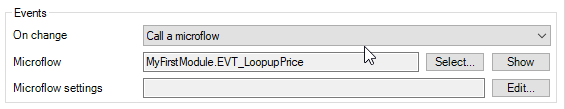
Select the Microflows folder and click New.



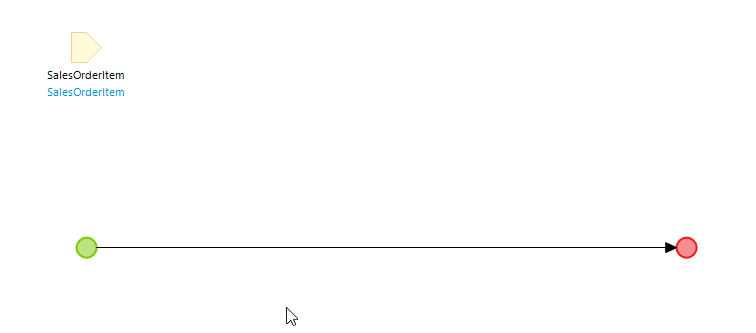
Name the microflow EVT\_LookupPrice.



Before clicking OK, click Show to open the microflow.



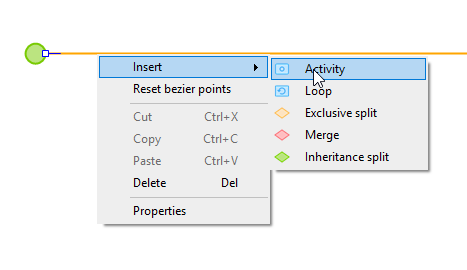
When the microflow is first created, it consists of a SalesOrderItem parameter, a Start event and an End event. The SalesOrderItem parameter is equal to the SalesOrderItem object bound to the Data view on the SalesOrderItem\_NewEdit page.



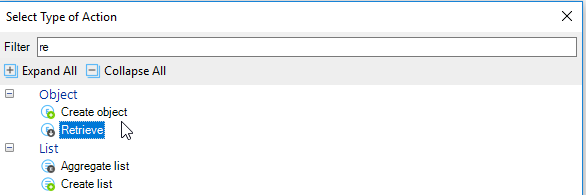
The steps in the microflow are:

1. Retrieve the Product associated with the SalesOrderItem
2. If no product is found, end the microflow
3. If the product is found, update the price and amount attributes of SalesOrderItem and refresh the client interface

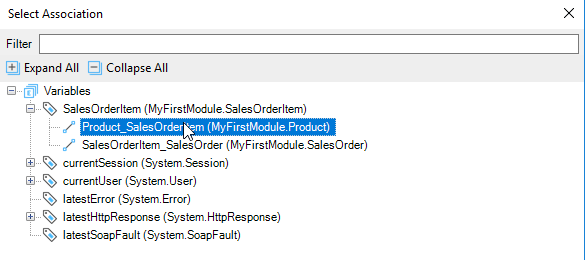
Add an Activity by right-clicking the line between the Start and End events and selecting Insert→Activity (alternatively you can use the Activity item above the editor).



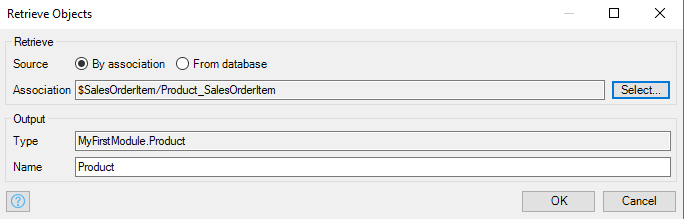
Configure the activity as a Retrieve type.



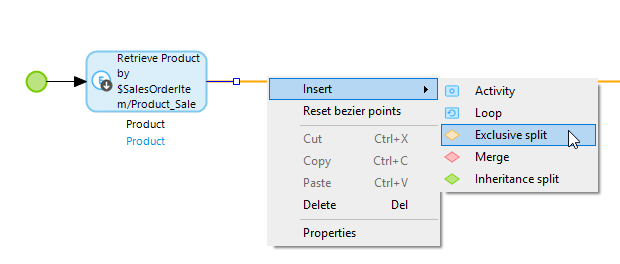
Set the Source to By association and use the Select… button to select the Product\_SalesOrderItem association.



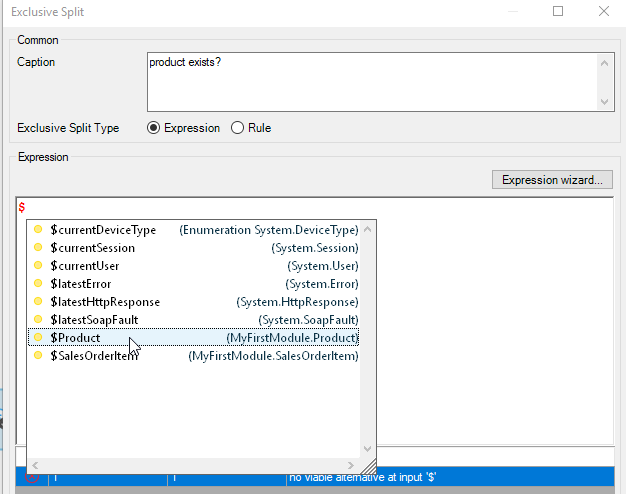
The final configuration of the activity should look like the image below.



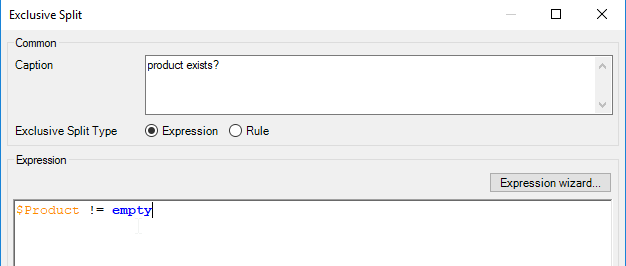
The Product variable should hold the selected Product. To check if it was retrieved successfully, add an Exclusive split to the right of the Retrieve activity.



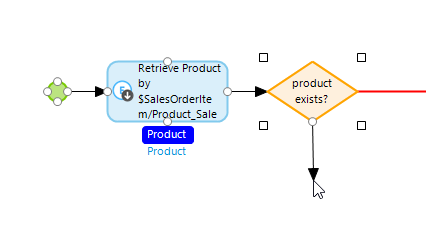
Enter a caption then add the expression. If you type a $, the dropdown shows all the variables available to use in the expression. Select $Product then complete the expression as shown in the next image.



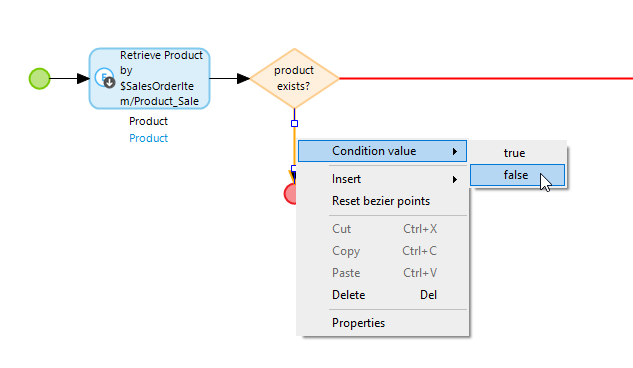
This will be false if the no product was retrieved.



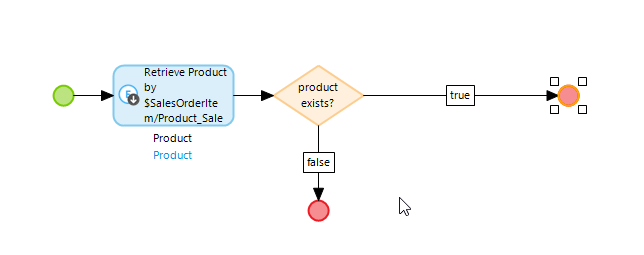
Click on the bottom point of the exclusive split and drag down and add an End event.



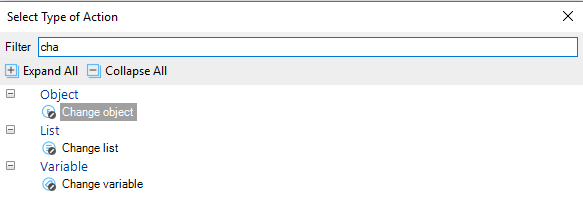
Right-click the line you just created and set its condition value to false. This will end the microflow if the product doesn’t exist.



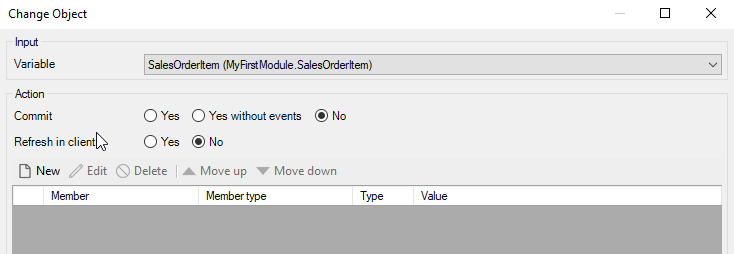
The microflow so far looks like this.



Add an activity to the right of the Exclusive split and make it a Change object type.



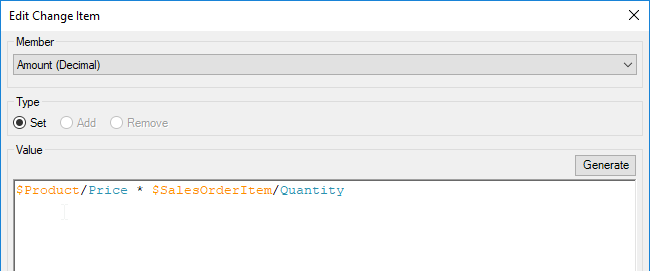
Select the SalesOrderItem variable.



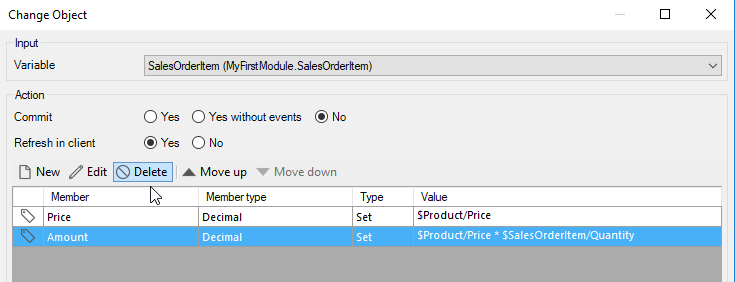
Click New and configure the new item as shown below. This sets the Price attribute of the SalesOrderItem equal to the Price of the Product variable.



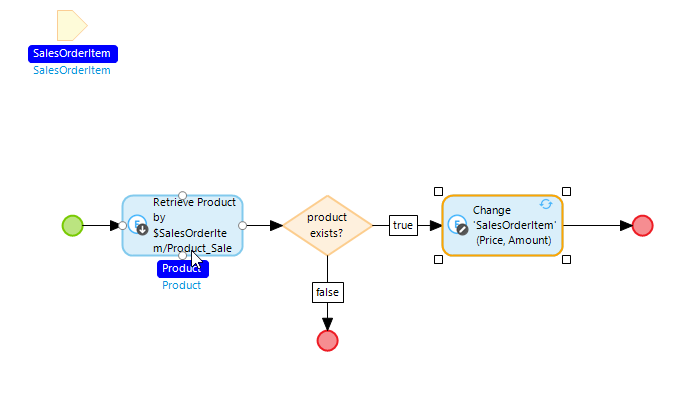
Add another item to update the Amount attribute. The configuration is shown below.



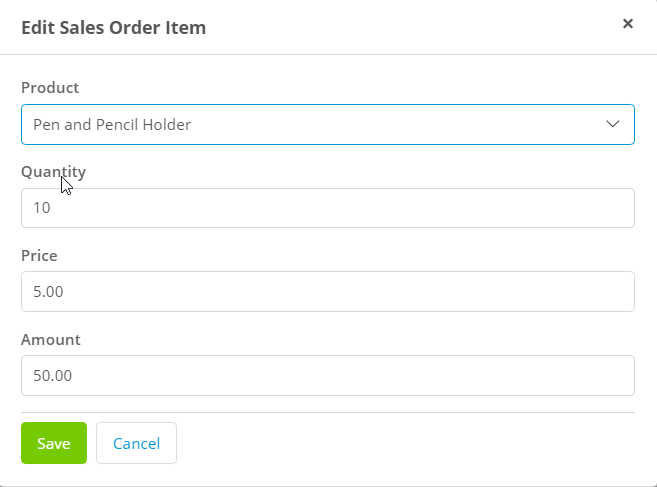
Make sure you select Yes on Refresh in client. This will cause Mendix to update the values in the interface.



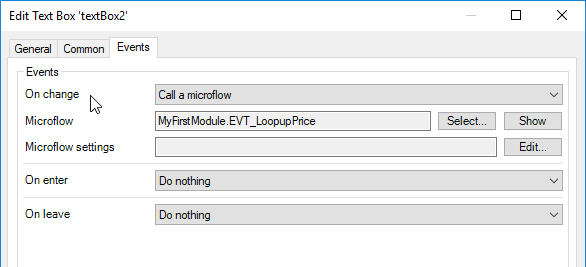
The final microflow is shown below.



Now when you select a Product, the price appears automatically. However, the amount remains 0 unless you enter the quantity before selecting the Product.



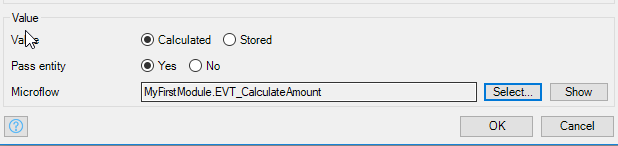
Edit the properties of the Quantity widget and configure the On change event as shown below. Now when a user selects a Product or changes the quantity, the microflow will run.



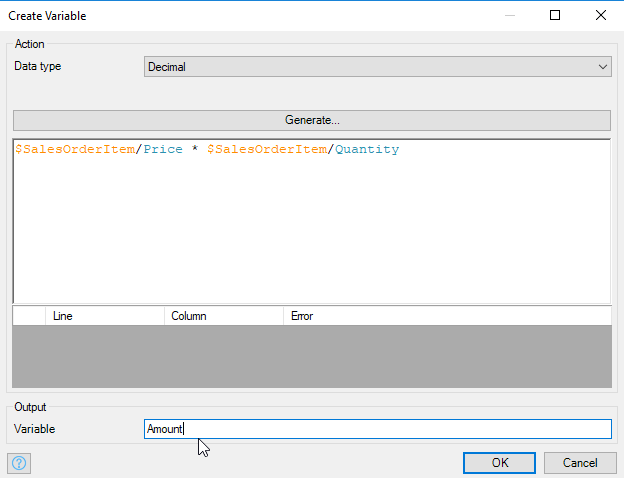
### An Alternative Approach for Amount

Another way to handle a calculated attribute like amount is to attach the microflow to the attribute in the entity. The advantage of this is that it will always calculate the correct value no matter how price or quantity is changed.

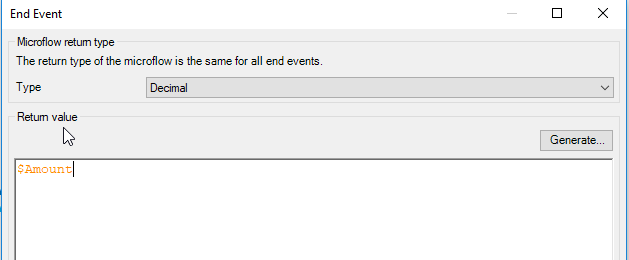
Edit the Amount attribute of the SalesOrderItem entity to change the value to calculated. Create a new microflow called EVT\_CalculatedAmount.



Add an activity to the new microflow and make it a Create variable type. Configure it as shown below.



Open the properties of the End event and configure it to pass the new variable back to the entity.



Finally…

* delete the Amount attribute from the Change object activity in the EVT\_LookupPrice microflow
* set the onChange event handler on the Quantity widget to Do Nothing
* delete the Amount widget on the SalesOrderItem\_NewEdit page

## Final Touches

One final thing we can do is to add New, Edit and Delete buttons to the sales order item Data grid on the SalesOrderOverview page.

