Blocks Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, we will focus on creating a block that will use input parameters, and trigger an event. These tools will be used to help us on the following scenarios:

- By creating a block, it will be easier to focus on how information is displayed in a more controlled context.
- The On Parameters Changed event will allow to refresh the block information whenever the parent changes an input parameter of the block.
- By creating and triggering an event, we will be able to notify the parent that something has occurred, and have the parent handle that occurrence accordingly.

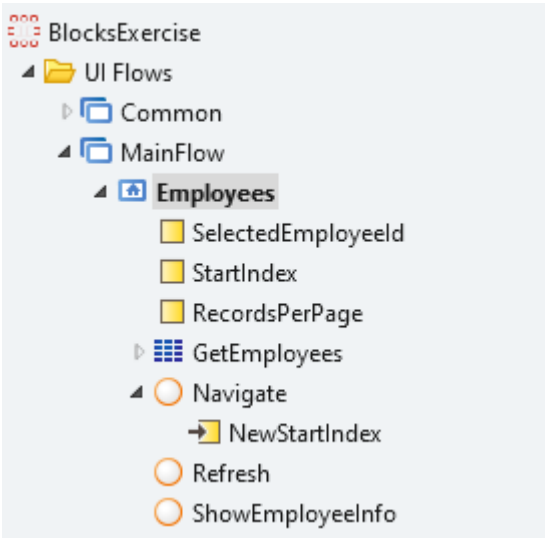
When this is completed, the simple application will have a block that uses an input parameter, the On Parameters Changed event, and triggers an event.

Resources

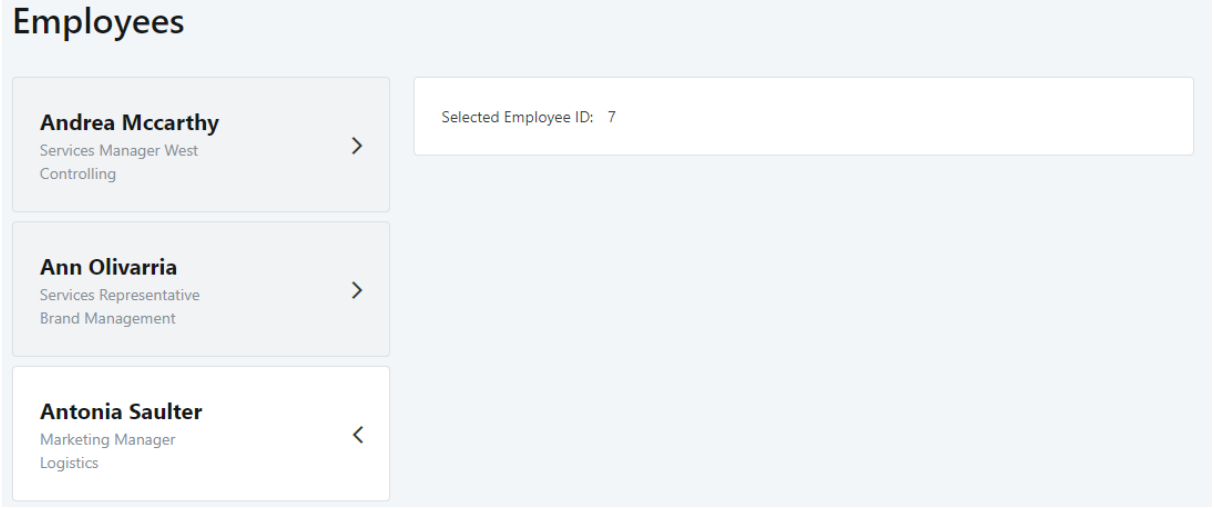
This exercise has a Quickstart application already created. This application has everything needed to start the exercise. This quickstart application can be found in the Resources folder of this exercise, with the name **Blocks Exercise.oap**.

Scenario

In this exercise, we'll start from an existing application with one module. Inside that module we have one Screen, called **Employees**. The Screen has an Aggregate that fetches all the Employees, alongside some extra logic to support the behavior on the UI.



The Screen has a left section that displays the list of employees. By clicking on an employee from the list, the detailed information of that particular Employee should appear on the right. At the start, we will only see the ID of the Employee, but in this exercise we will enrich that area.

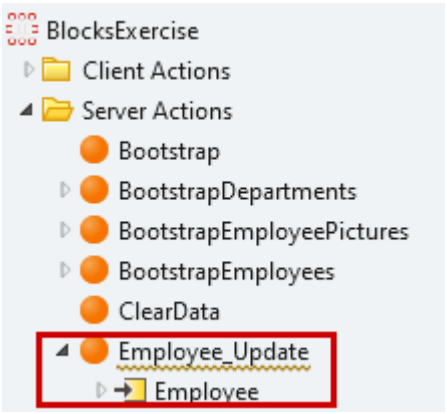


The area on the right is where we'll later place the Block we're going to create. This block will fetch and display the detailed information of an Employee from the database.

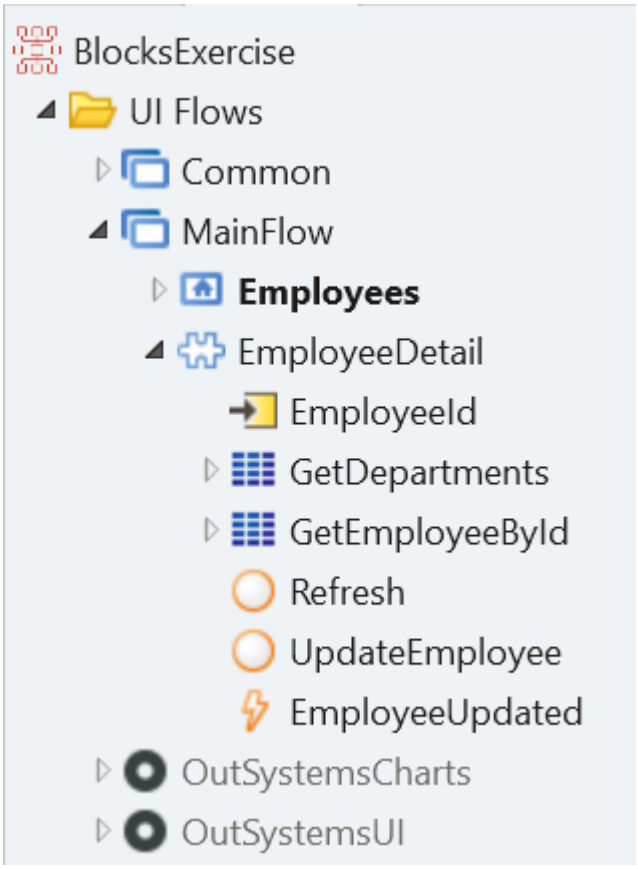
Later on, we'll implement the On Parameters Changed event handler. This will allow refreshing the information displayed inside the block, whenever the Employee selected changes in the parent screen.

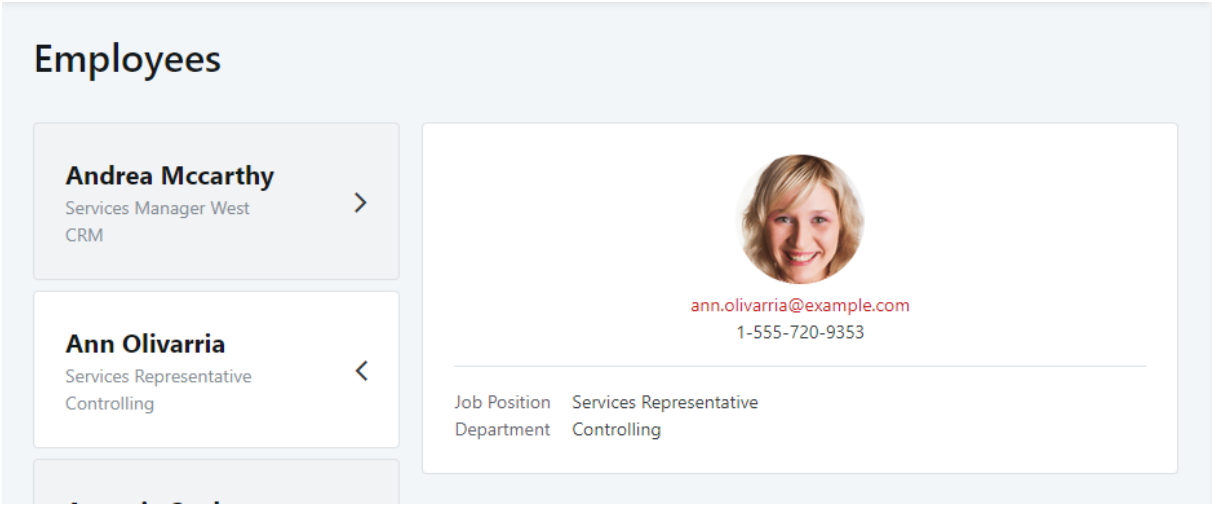
After that, the block will be modified to allow updating the Department of an Employee using a Dropdown. When the Department is changed, the block will trigger an event to notify the parent. The parent will handle that event, and refresh the information shown in the list (that includes the department name).

The module already has a Server Action defined with the logic to update the Employee in the database: **Employee_Update**.



By the end of this exercise your application should look like this.



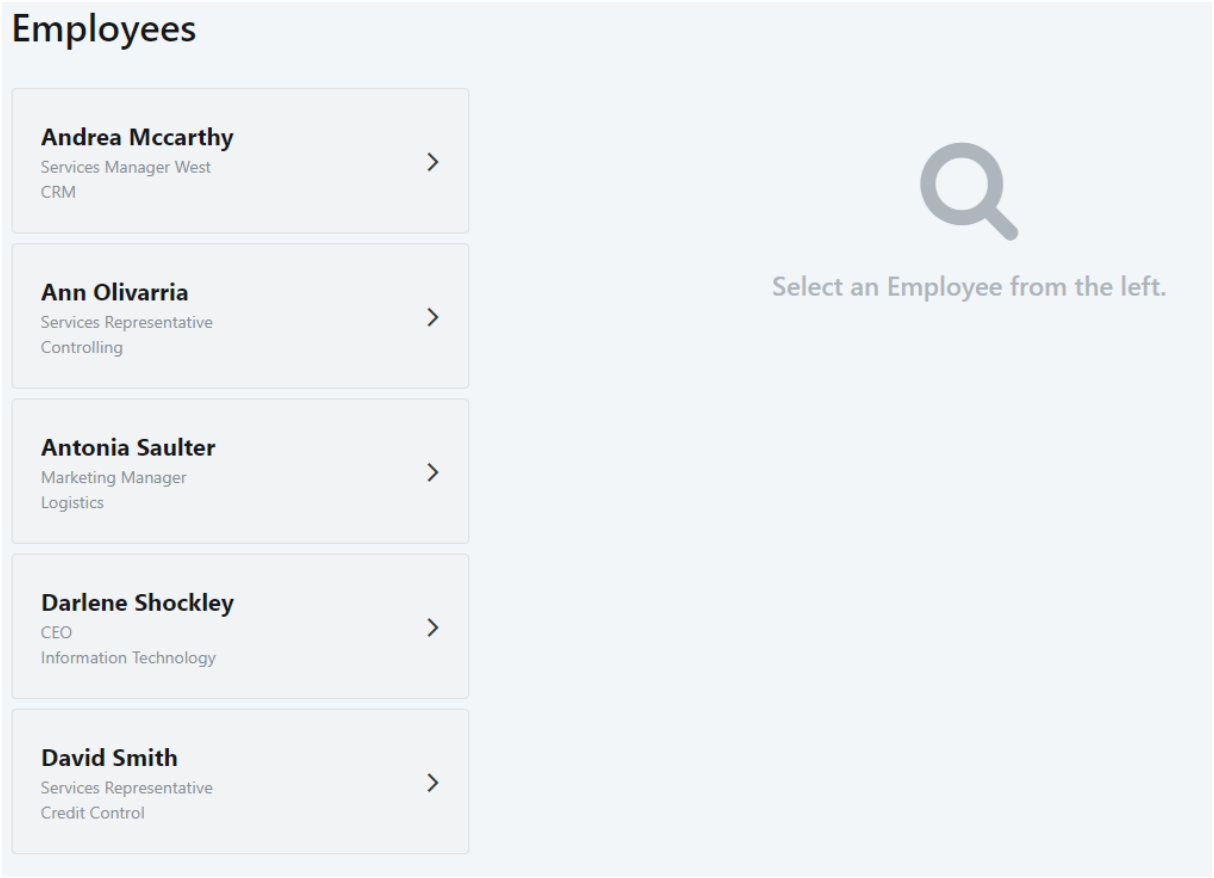


How-To

In this section, we'll 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.

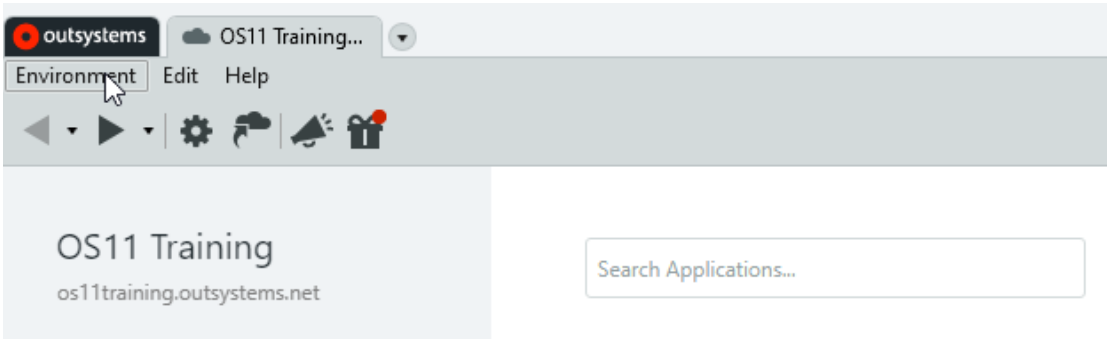
Getting Started

To start this exercise, we need to install the Quickstart file, which has the application already created. This application has one screen, Employees, which has an area to the right to display the details of an Employee. Most of the logic is already created, but the interface is not yet displaying the Employee information.

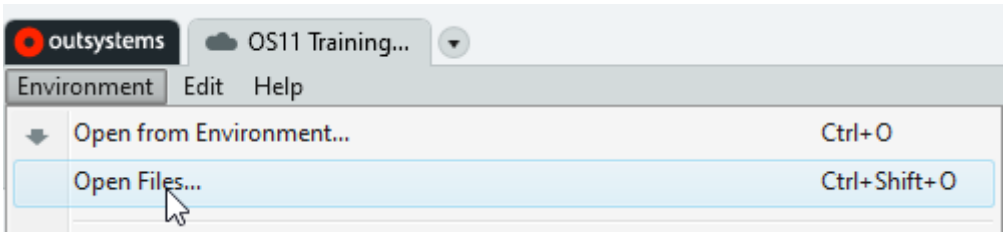


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

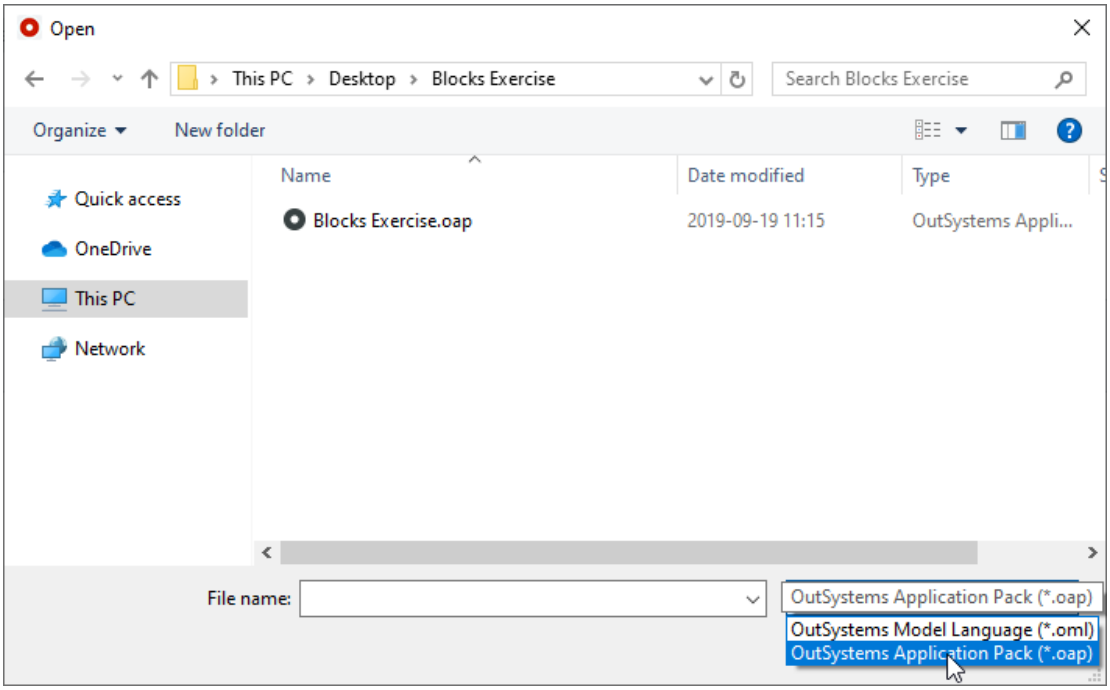
1) In Service Studio's main window, select the Environment menu on the top left.



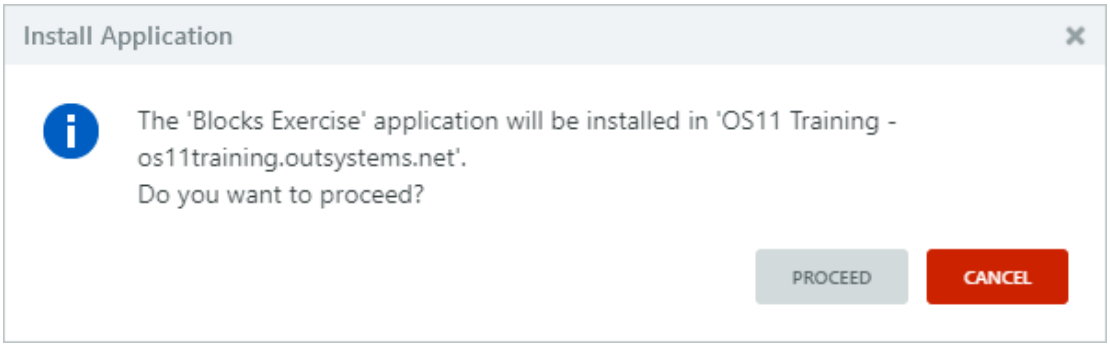
2) Select Open Files...



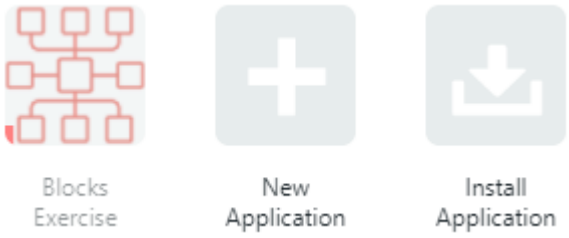
3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Blocks Exercise.oap**.



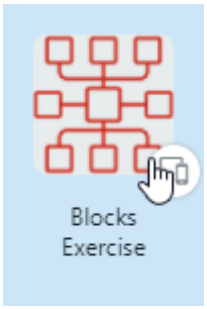
4) In the new confirmation dialog, select **Proceed**.



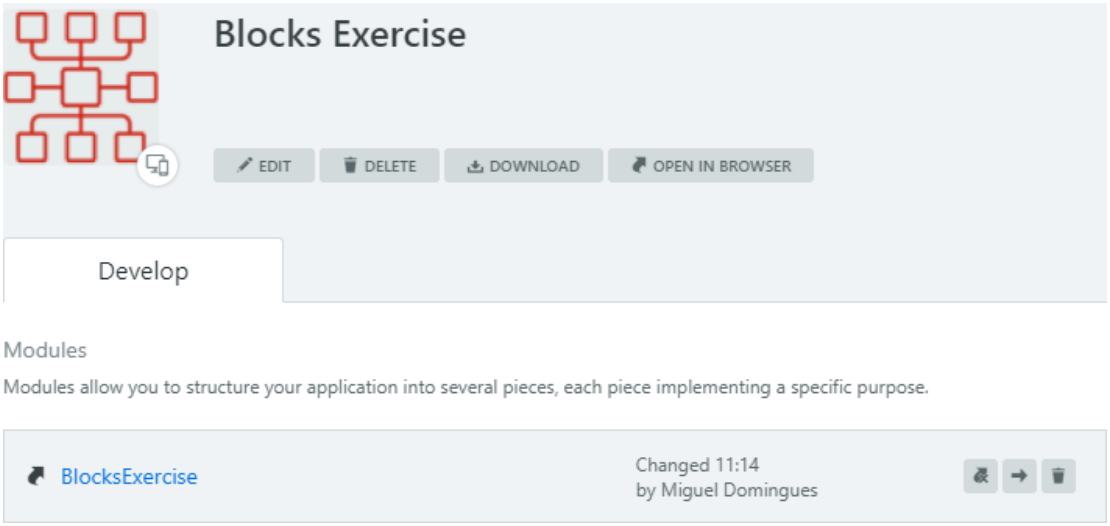
5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.



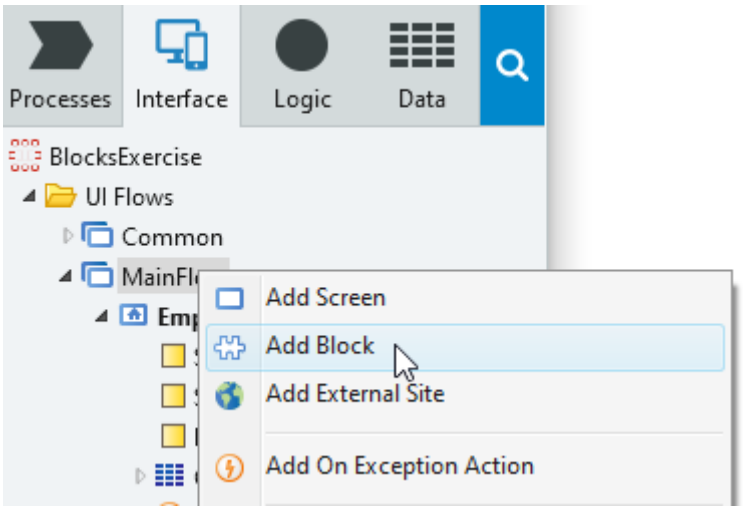
7) The application has only one module. Let's open it!



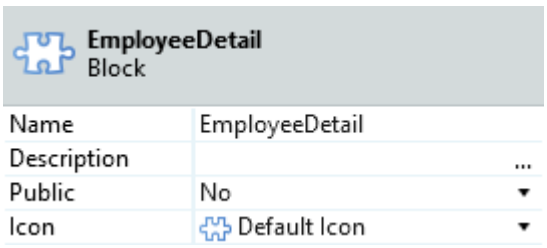
Creating a Block

In this section, we'll create and implement a Block to display the information of one Employee. This block will then be used inside the Employees list screen to display the information of the selected Employee.

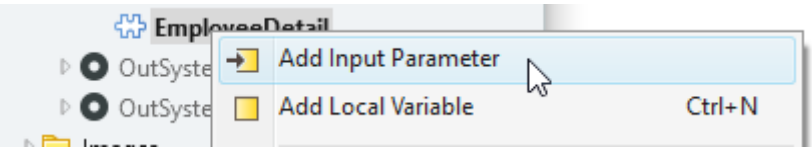
1) Under the Interface tab, right-click the MainFlow and select **Add Block**.



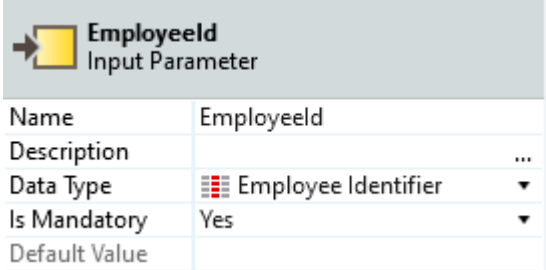
2) Set the Name of the Block to *EmployeeDetail*.



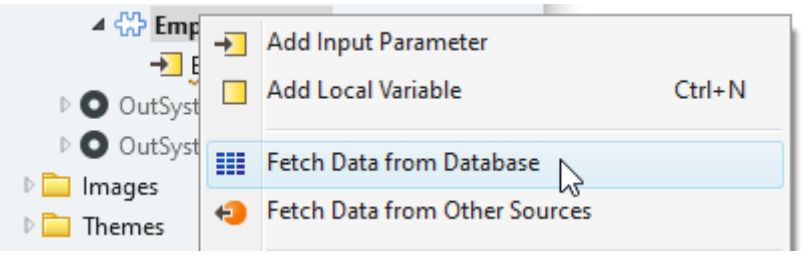
3) Right-click the EmployeeDetail Block and select **Add Input Parameter**.



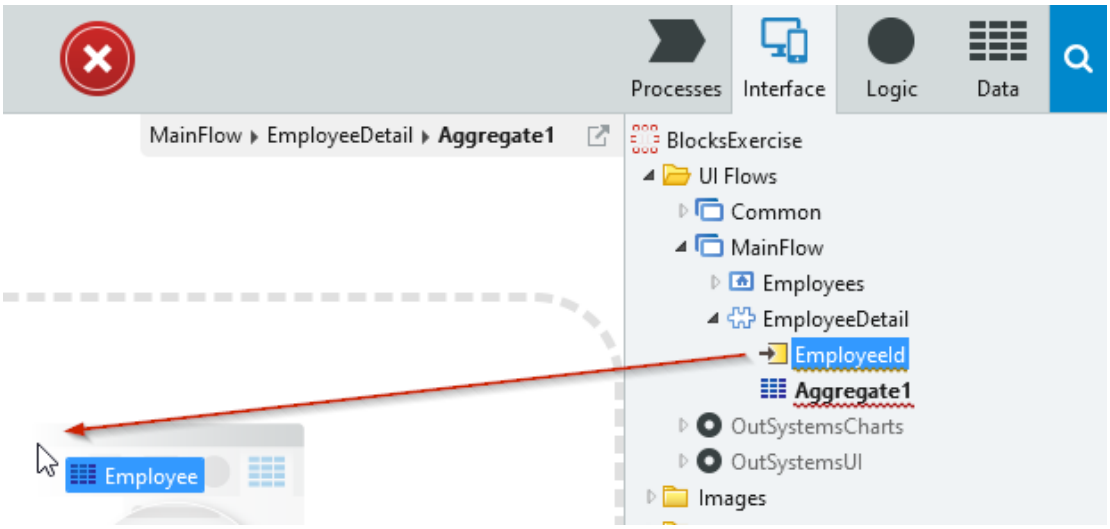
4) Set the **Name** of the Input Parameter to EmployeeId and make sure the **Data Type** is set to *Employee Identifier*.



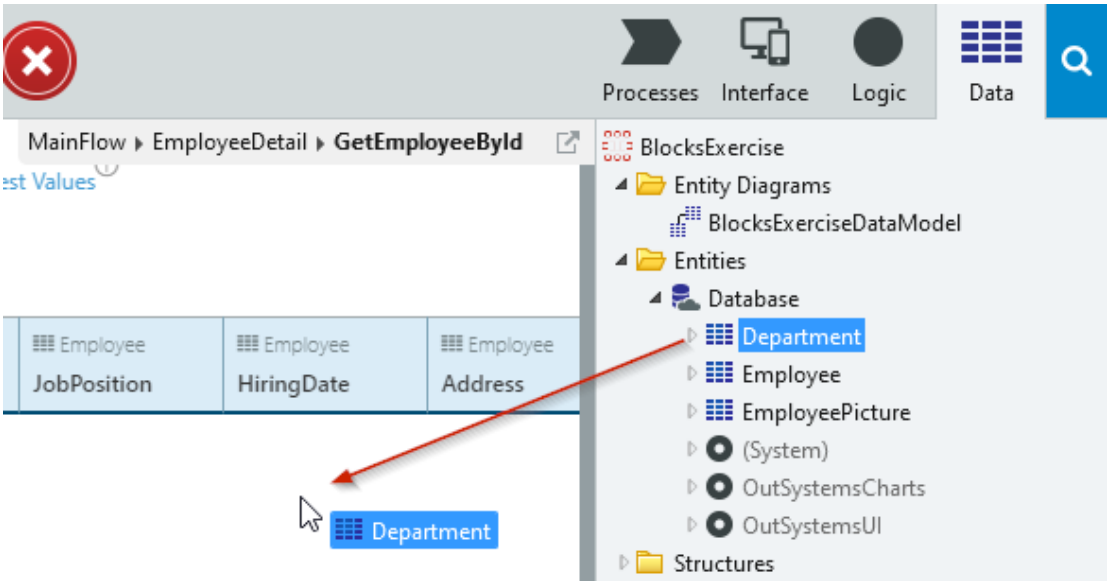
5) Right-click the Block, and select **Fetch Data from Database**.



6) Drag the **EmployeeId** input parameter of the Block and drop it inside the Aggregate.



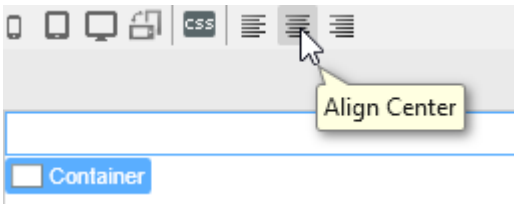
7) From the Data tab, drag the **Department** entity and drop it inside the Aggregate.



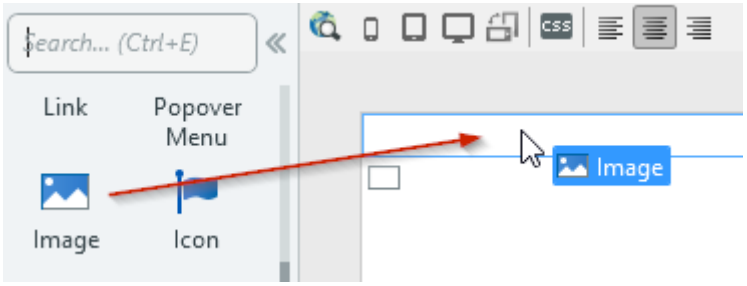
8) Return to the Interface tab, and open the **EmployeeDetail** Block.

9) Drag a **Container** and drop it inside the Block canvas.

10) Click the **Align Center** icon on the toolbar.



11) Drag an **Image** and drop it inside the Container created before.

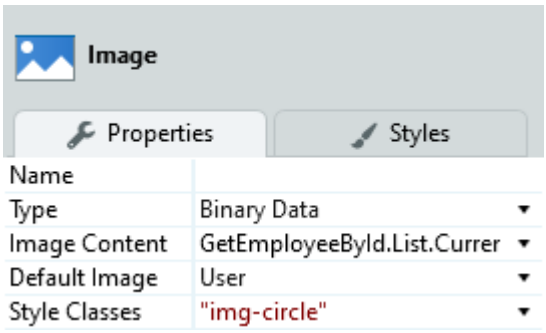


12) Change the **Type** property to *Binary Data* and then set the **Image Content** property to:

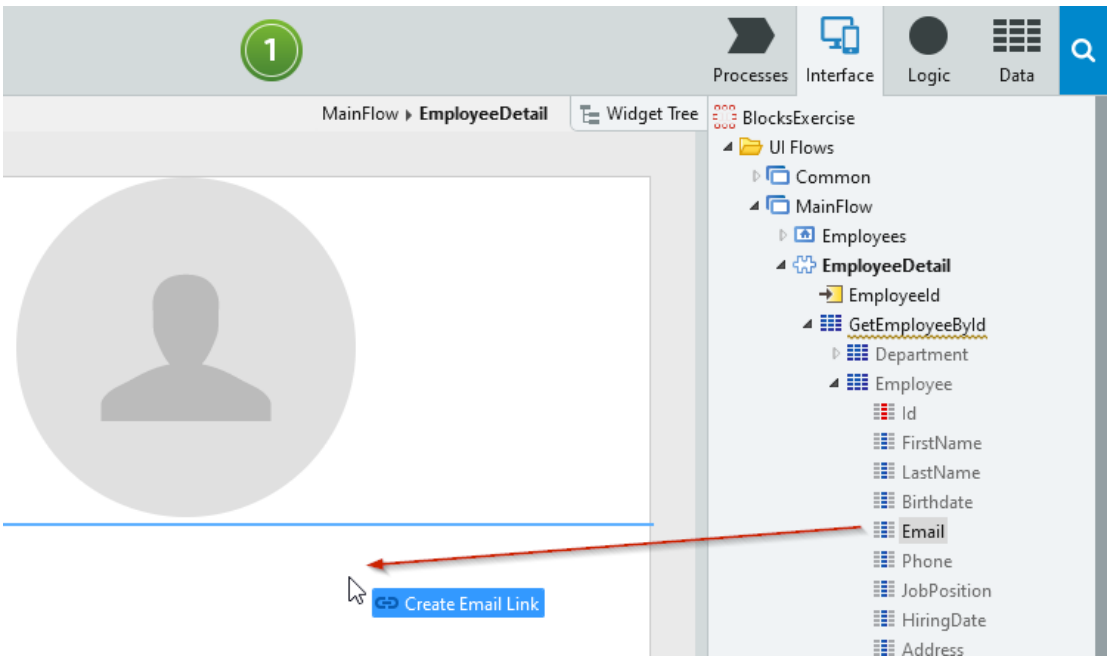
```
GetEmployeeById.List.Current.EmployeePicture.Picture
```

13) Set the **Default Image** property to *User*, and the **Style Classes** property to:

```
"img-circle"
```



14) Expand the **GetEmployeeById** Aggregate, and locate the **Email** attribute from the Employee entity. Then, drag it and drop it below the Image created before.

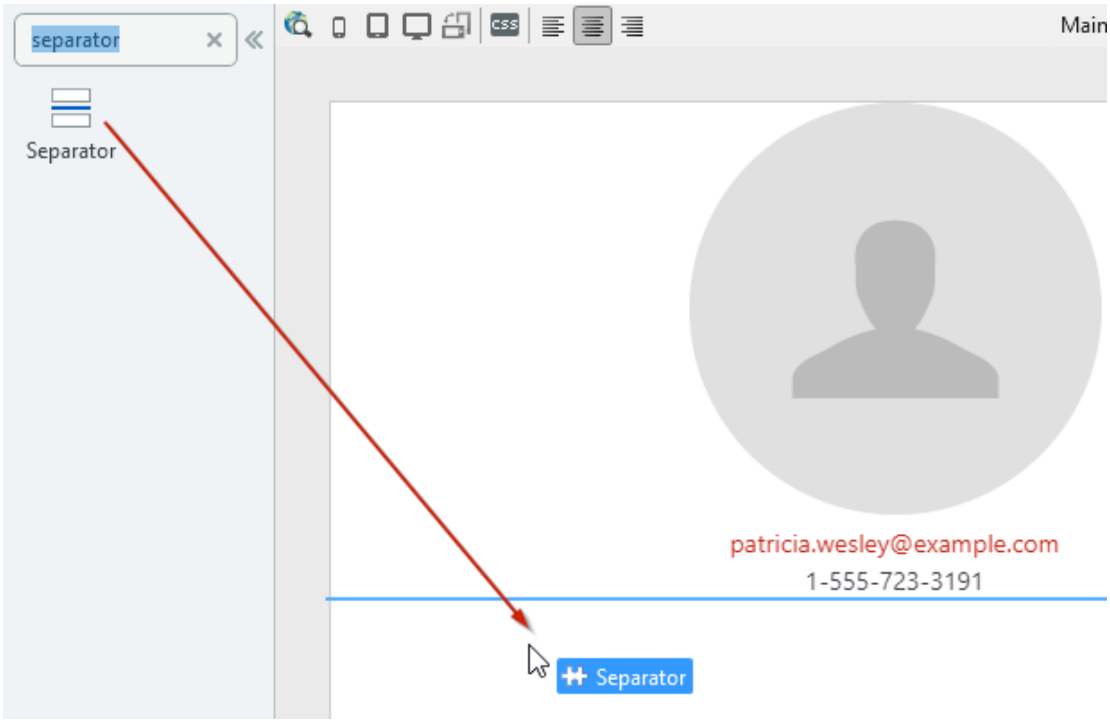


Make sure that the email address appears below the picture.

15) Click the **Align Center** icon from the toolbar to center the Email.

16) Repeat the two previous steps for the Phone attribute.

17) In the Widget Search box type separator, then drag the **Separator** pattern and drop it below the Phone.

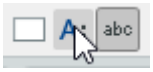


18) Drag a **Container** and drop it below the Separator. Don't forget to clear the search box.

19) Drag a **Label** and drop it inside the Container created in the previous step.

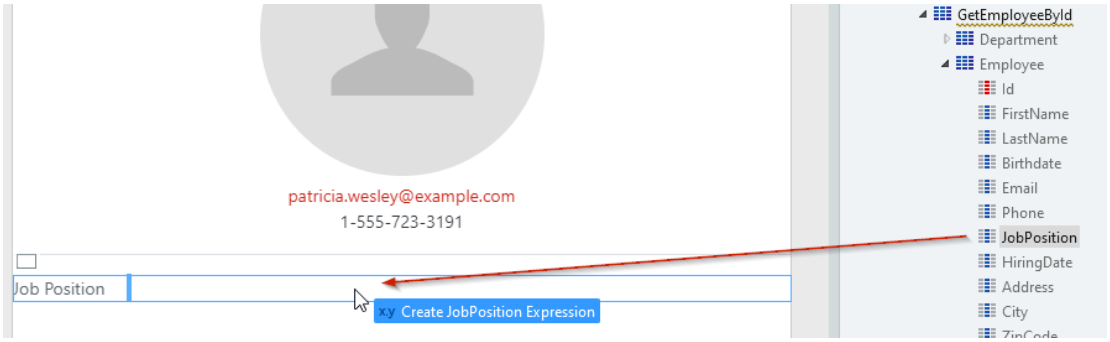
20) Change the text inside the Label to *Job Position*.

21) Using the Widget breadcrumb at the bottom, select the **Label** that is surrounding the Text, and inside the Container.



22) Resize the Label to two columns by dragging the square market to the right.

23) Drag the **JobPosition** attribute that is under the GetEmployeeById Aggregate, and drop it to the right of the Label.



24) Drag another **Container** and drop it at the bottom of the Block (below the Job Position container).

25) Drag a **Label** widget and drop it inside the Container created in the previous step.

26) Change the text to *Department*.

27) Select the Label that surrounds the text and change it to **2 columns**.

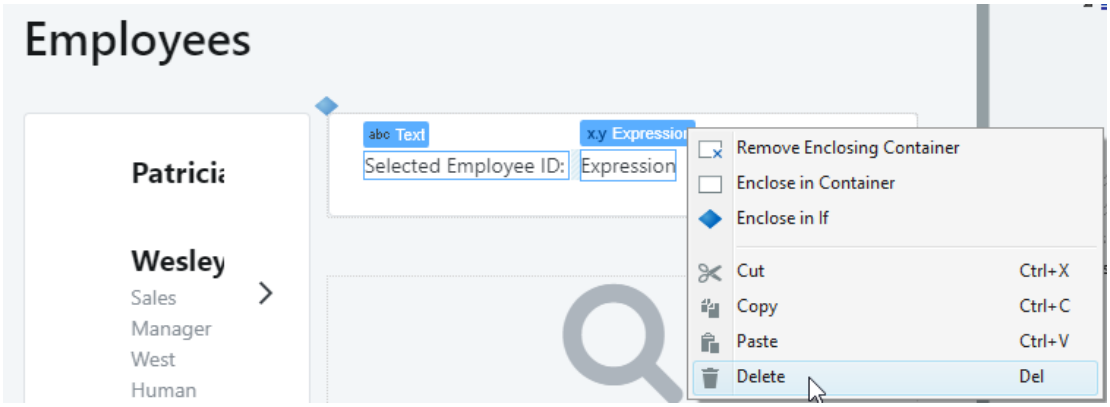
28) Drag the **Name** attribute of the Department entity from the GetEmployeeById Aggregate and drop it to the right of the Department Label.

Using a Block

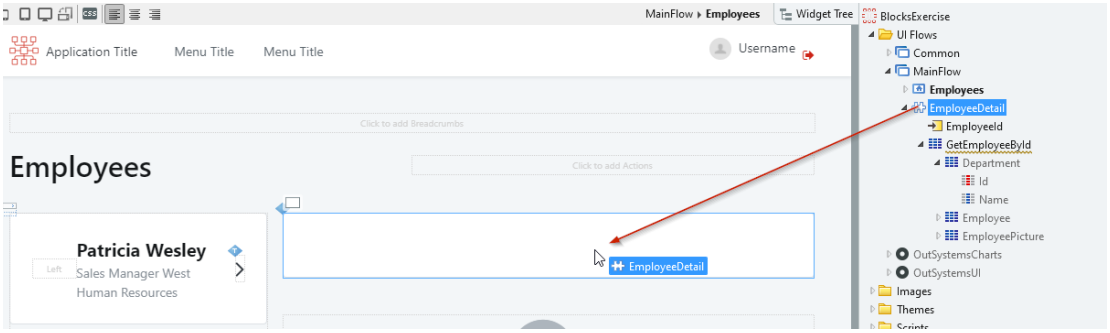
In this section, we'll use the Block that was created in the previous section. This Block will be used inside the Employees Screen to display the information of an Employee.

1) Open the **Employees** Screen.

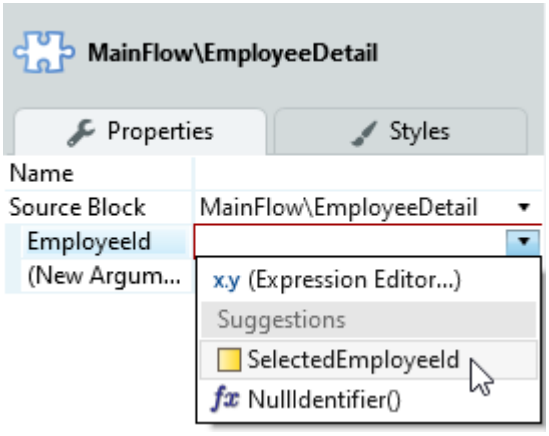
2) Select the text and expression that is inside the white Container at the right, then delete those widgets.



3) Drag the **EmployeeDetail** Block, and drop it inside the white Container.



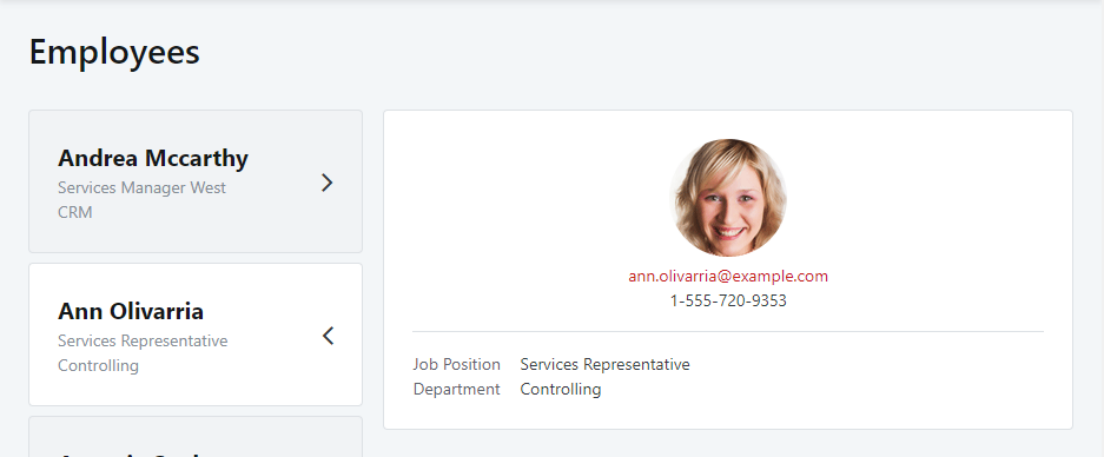
4) In the properties of the instance of the Block, set the **EmployeeId** input parameter to *SelectedEmployeeId*.



5) Publish the module and then open it in the browser.

A Licensing Warning about Server Side Caching might appear on the publish log. You can ignore it.

6) In the browser, select Ann Olivarria from the list to the left. Notice that on the right the Block is displayed.



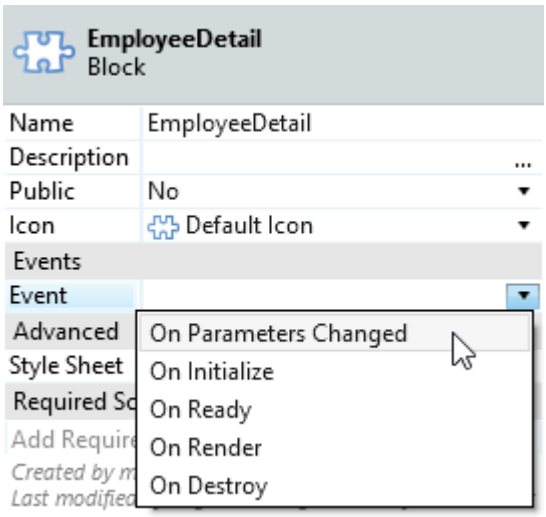
7) Clicking on another Employee on the list to the left does not refresh the displayed information. This is happening because the handler for On Parameters Changed event of the EmployeeDetail block was not defined yet.

On Parameters Changed

In this section, we'll implement the **On Parameters Changed** event handler. This will improve the usability of our application, by allowing clicking the Employees on the list to the left and updating the information on the right inside the Block.

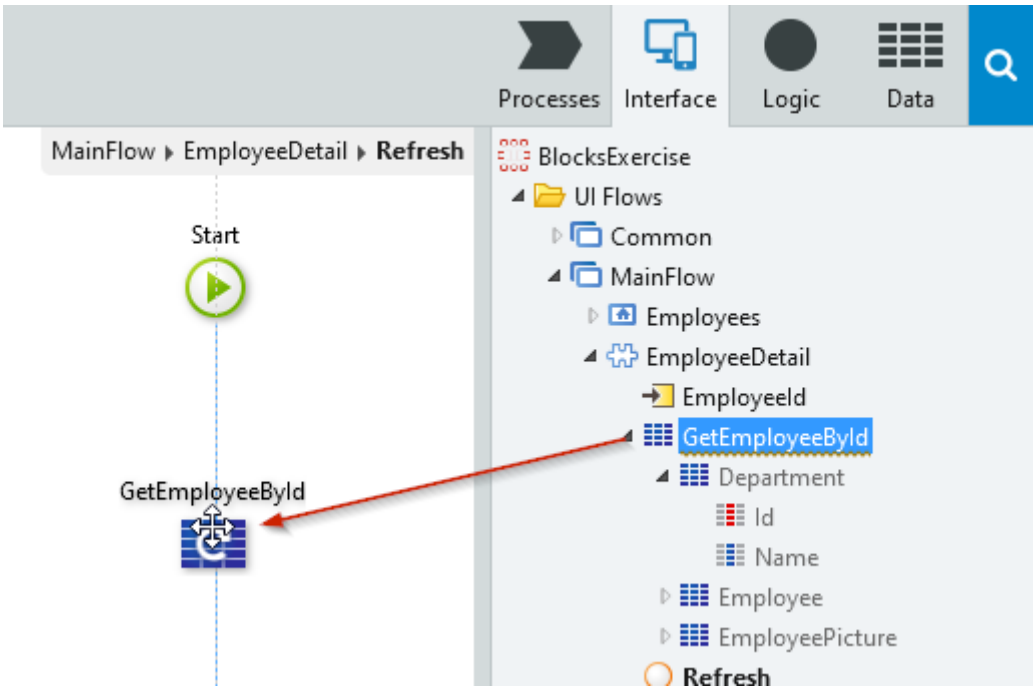
1) Return to Service Studio and select the **EmployeeDetail** Block.

2) In the properties of the Block, open the dropdown of Event property and select the **On Parameter Changed** event.



3) Rename the newly created client action from OnParametersChanged to Refresh.

4) Drag the **GetEmployeeById** aggregate and drop it between the Start and End.



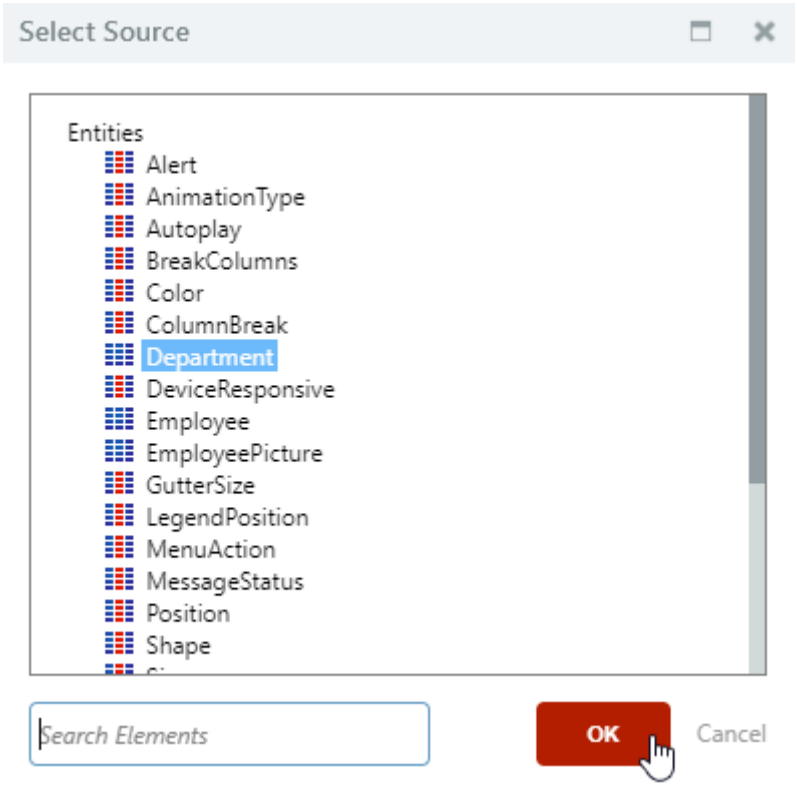
5) Publish the module and test the application again, by selecting one Employee, and then clicking on another one. The information of the Employee should be updated accordingly.

Triggering and Handling Events

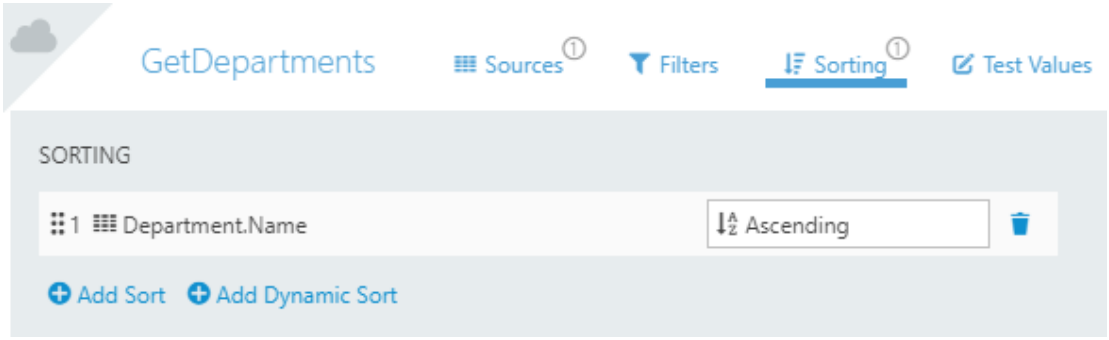
In this section, we'll change the **EmployeeDetail** Block, to allow updating the Department to which an Employee is associated. When this change occurs, an event will be triggered. This event will then be handled by the parent Screen (Employees) to update the information displayed on the list. We will start by fetching the list of departments from the Database, and the add a Dropdown to the Block to allow updating the department.

1) Right-click the EmployeeDetail Block and select Fetch Data from Database.

2) Click the Aggregate canvas and then select the Department Entity.

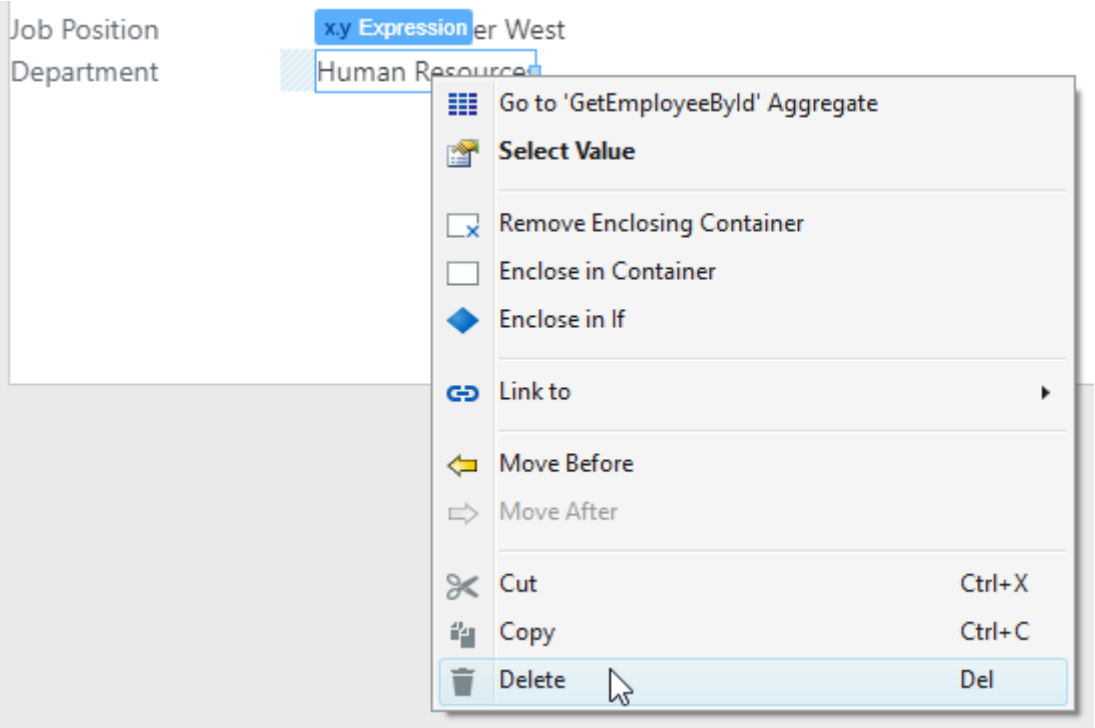


3) In the **Sorting** tab, add a new sort clause by the Department **Name**.

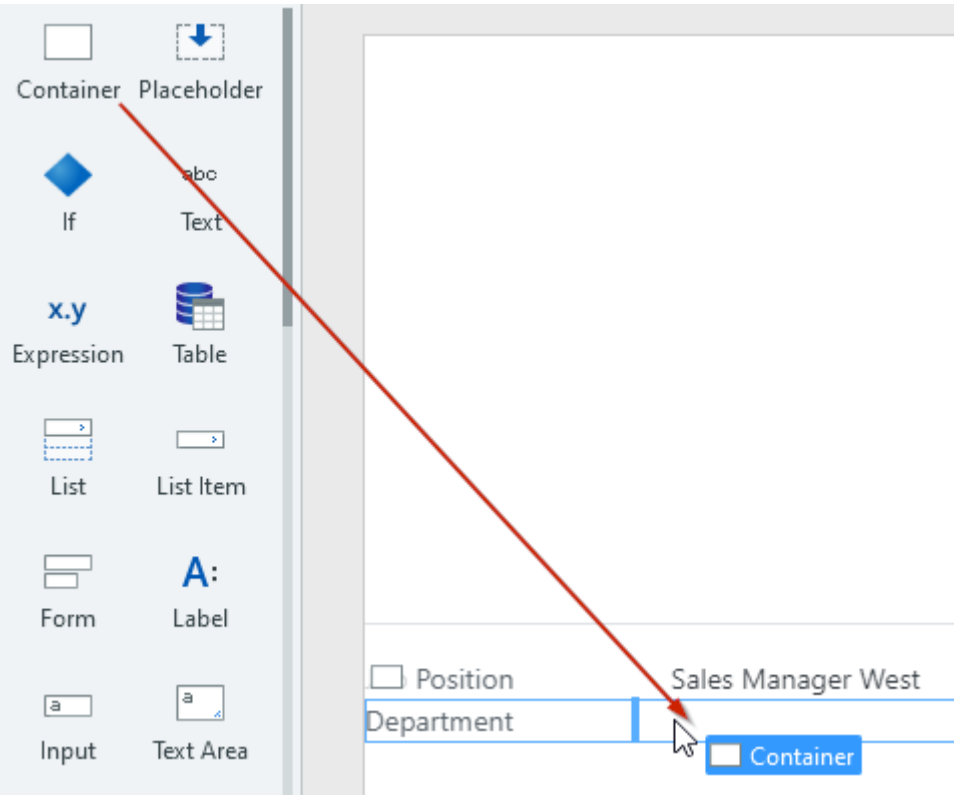


4) Open the **EmployeeDetail** Block.

5) Select the **Expression** that displays the Department, then delete it.



6) Drag a **Container** and drop it where the Expression was located.

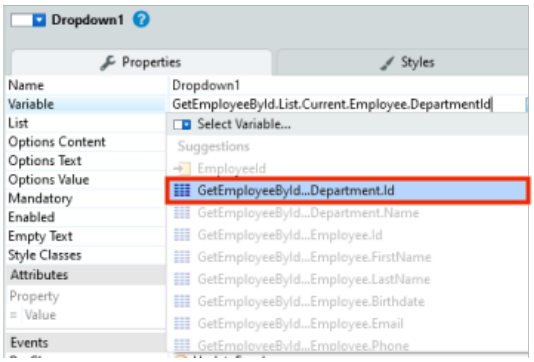


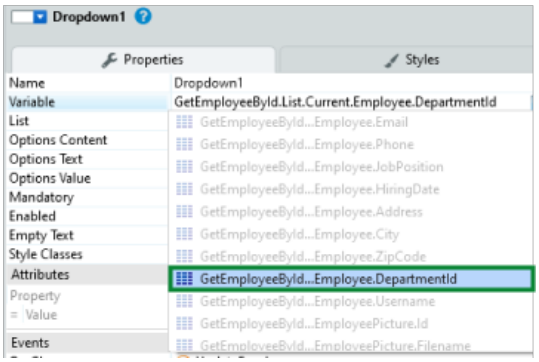
7) Drag a **Dropdown** and drop it inside the Container created above.

8) Change the **Variable** property to:

```
GetEmployeeById.List.Current.Employee.DepartmentId
```

Warning: You need to select the *DepartmentId* attribute of the *Employee* entity. Selecting the *Id* attribute of the *Department* entity will yield different results (e.g. the department won't be updated at the end).

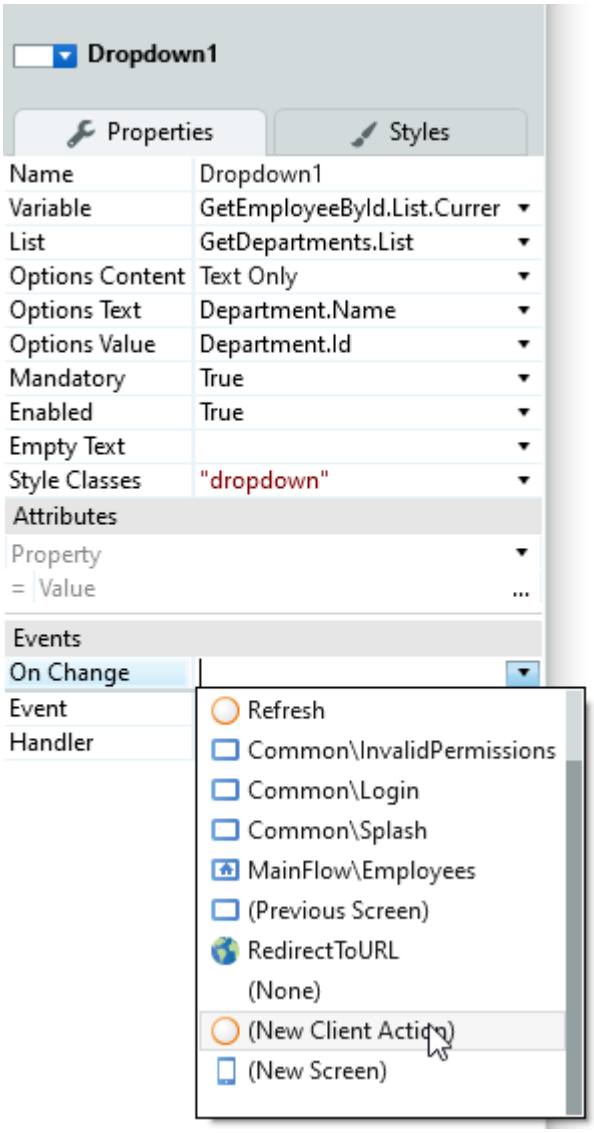




9) Change the **List** property to:

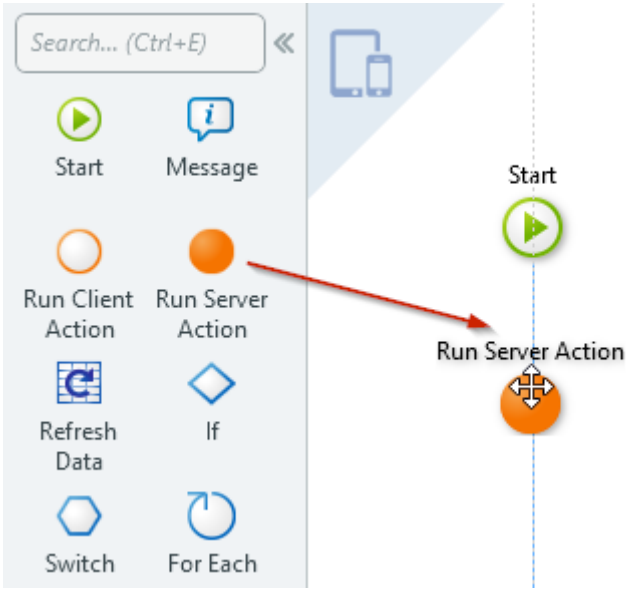
```
GetDepartments.List
```

10) In the **On Change** event property, select (New Client Action) from the dropdown of suggestions.

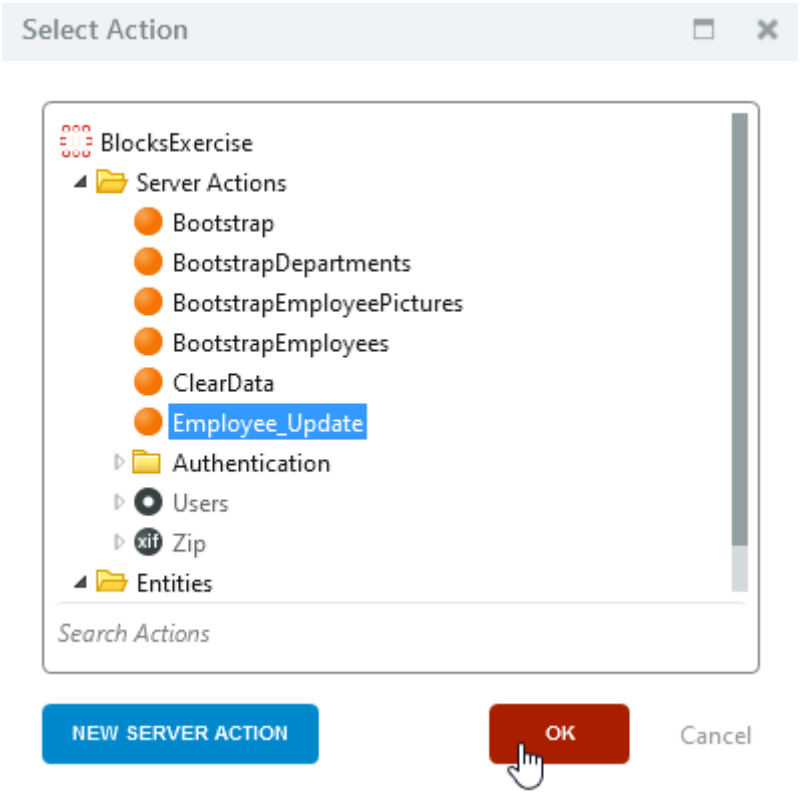


11) Rename the Client Action that was created to *UpdateEmployee*.

12) Drag a **Run Server Action** and drop it after the Start.

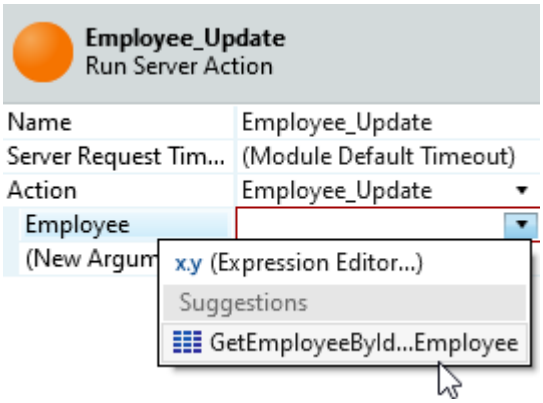


13) In the **Select Action** dialog, select the **Employee_Update** action and click OK.

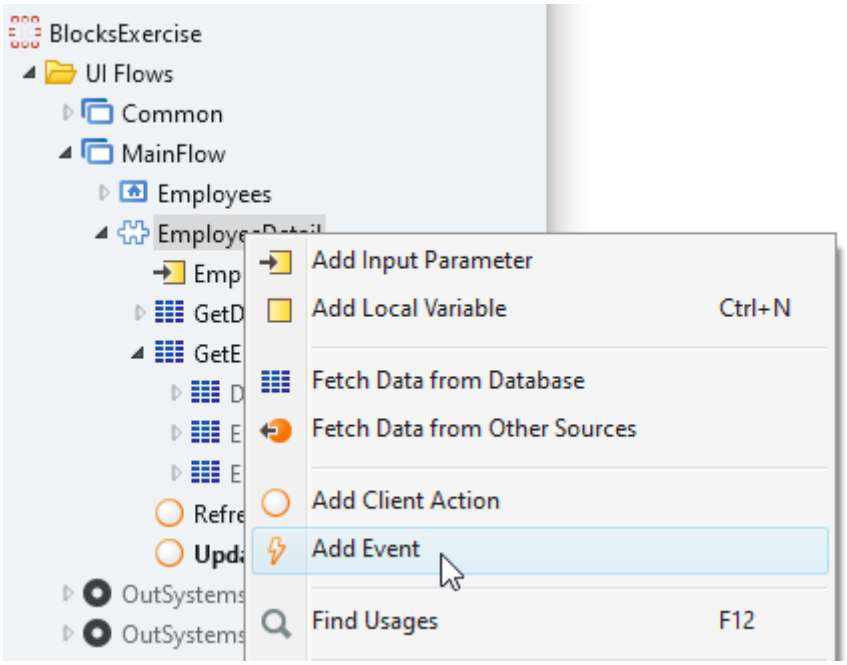


14) In the properties, set the Source property to:

```
GetEmployeeById.List.Current.Employee
```

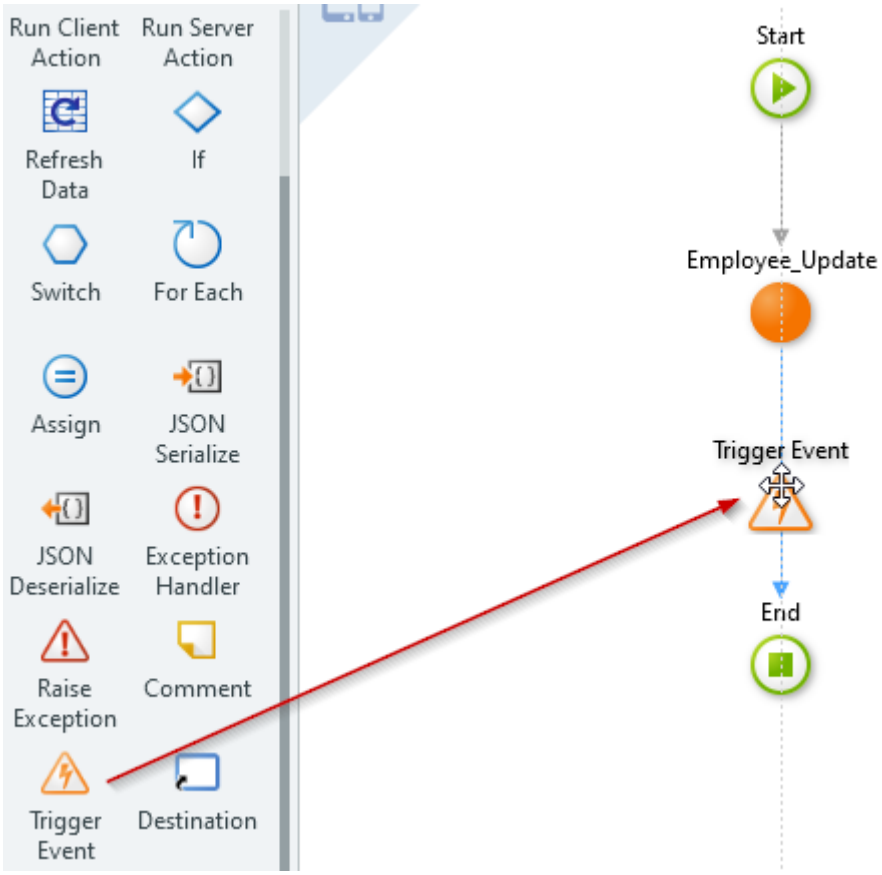


15) Before triggering the event to notify the parent, let's create the event. Right-click the EmployeeDetail Block and select **Add Event**.

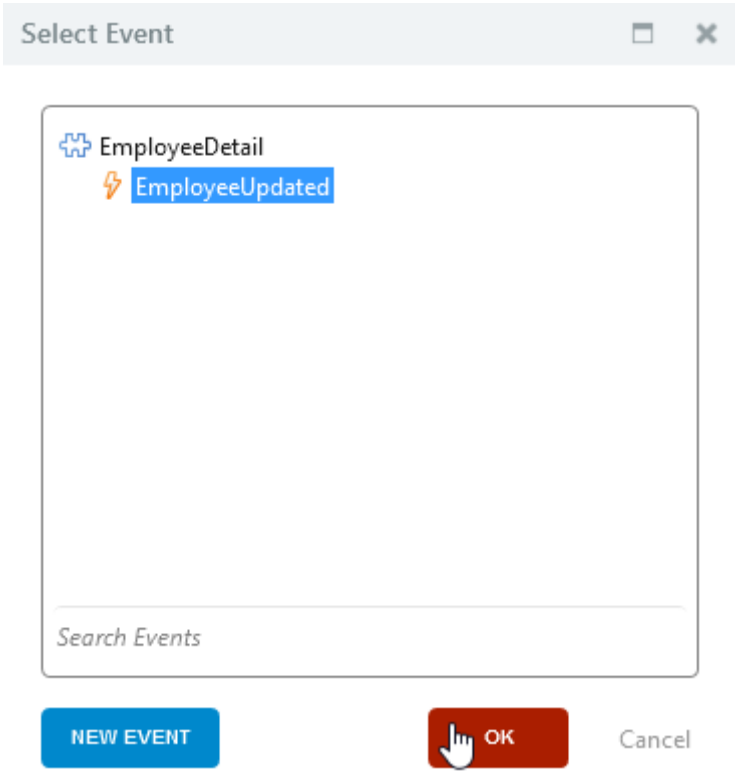


16) Set the Event **Name** to *EmployeeUpdated*.

17) Still inside the UpdateEmployee client action, drag a Trigger Event and drop it just before the End.

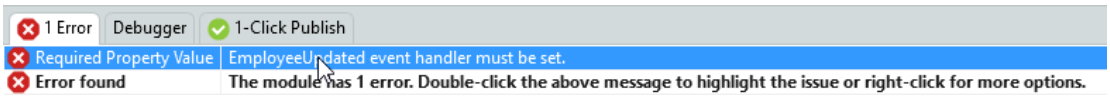


18) In the Select Event dialog, select the **EmployeeUpdated** event.



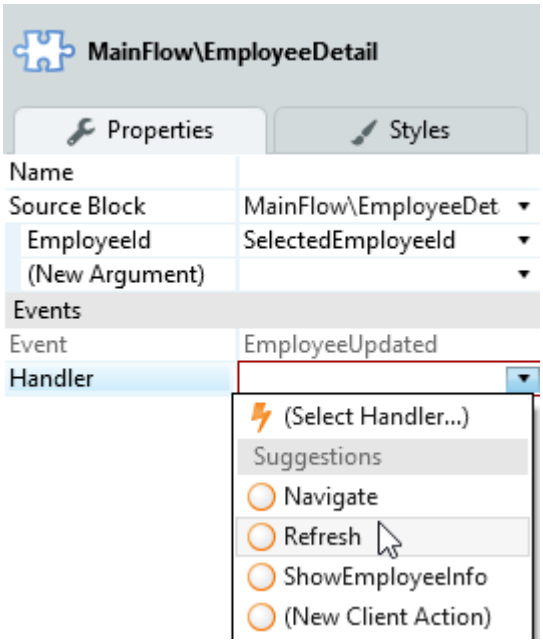
19) Click the Error icon at the top to open the Error panel.

20) Double-click the existing error.



21) The focus moved to the instance of the EmployeeDetail inside the Employees screen. Since the Event that was created is mandatory, a handler must be defined.

22) In the suggestions dropdown for the **Handler** of the EmployeeUpdated event, select **Refresh**.



This will make sure that when the event is triggered, the parent will run the Refresh Action, which will re-execute the Aggregate to fetch the employees information. That in turn will force the widgets on the screen to render again, and the end-user will see the new changes.

23) Publish the module.

24) Open the module in the browser, and then select an employee.

25) Using the dropdown, change the department of the employee. Notice that the list on the left also gets updated.