Pagination and Sorting Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, we will extend an existing Screen with some common features that are usually present in List Screens. This exercise will require a bit of work in Service Studio, but also some testing in the browser.

The application we are going to use has a set of employees. Besides other features, the application allows viewing the existing employees in a Table. In this exercise, we want to:

- Add Pagination to the Table
- Add Sorting to the Table
- Control the number of records displayed per page

Resources

This exercise has a Quickstart application already created with everything you need to start the exercise. This Quickstart application can be found in the Resources folder of this exercise, with the name **Pagination and Sorting Exercise.oap**.

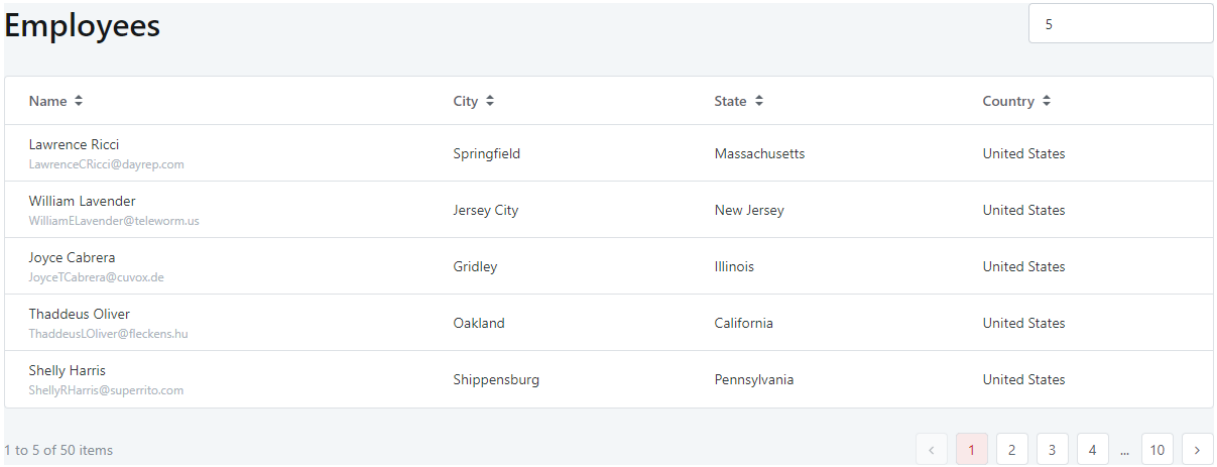
Scenario

In this exercise, we'll start from an existing application with one module. Inside that module, we already have an Employees Screen created, with a Table listing employees. This Table is where most of the User Interface work will actually be done.

The module also has built-in logic to import data from Excel to populate the Employees Entity. So, when the application (and its module) is installed, the data will be automatically bootstrapped and saved in the database.

The Table defined in the Employees Screen is already configured to display 10 records from the Employee Entity. This means that the source of the Table is set to an Aggregate (GetEmployees) that is also already defined.

In this exercise, we will extend the existing Screen to support pagination and sorting. At the end we will also add a functionality to control the number of records displayed per page.

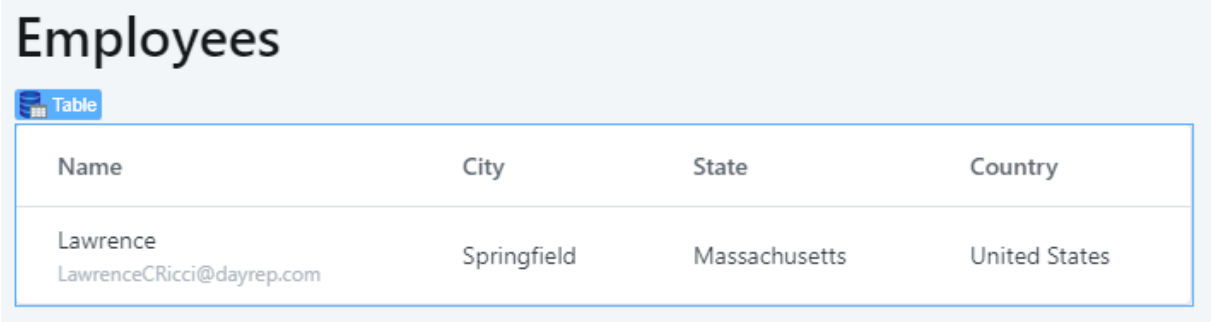


How-To

In this section, we'll show you how to do this exercise, with thorough, step-by-step instructions. **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.

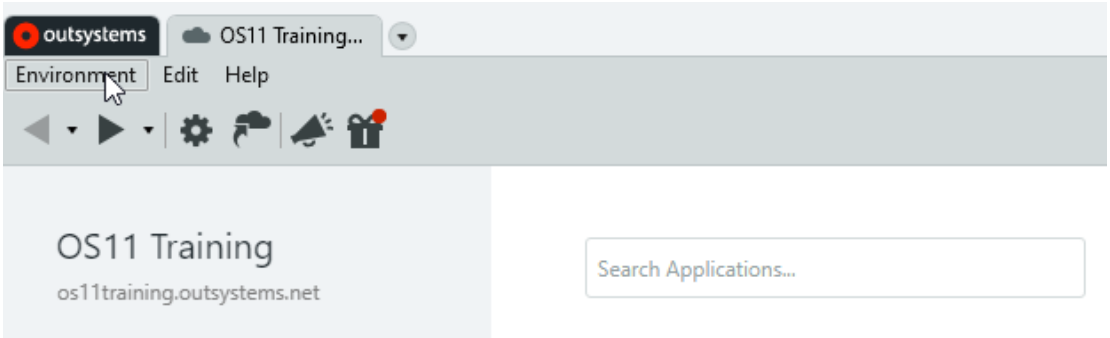
Getting Started

To start this exercise, we need to install the Quickstart file, **Pagination and Sorting.oap**. This file has an application containing the Employees Screen. This Screen already has a Table created and its source is set to the GetEmployees Aggregate that is part of the same Screen.

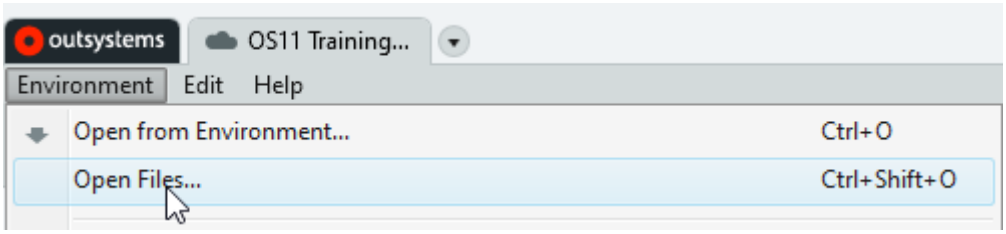


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

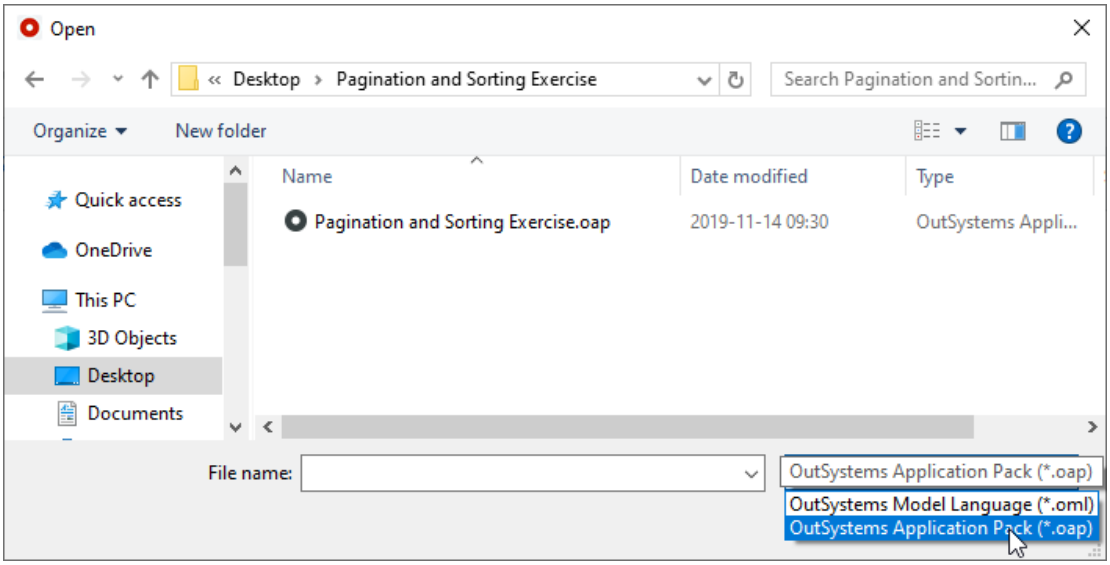
1) In Service Studio's main window, select the Environment menu on the top left.



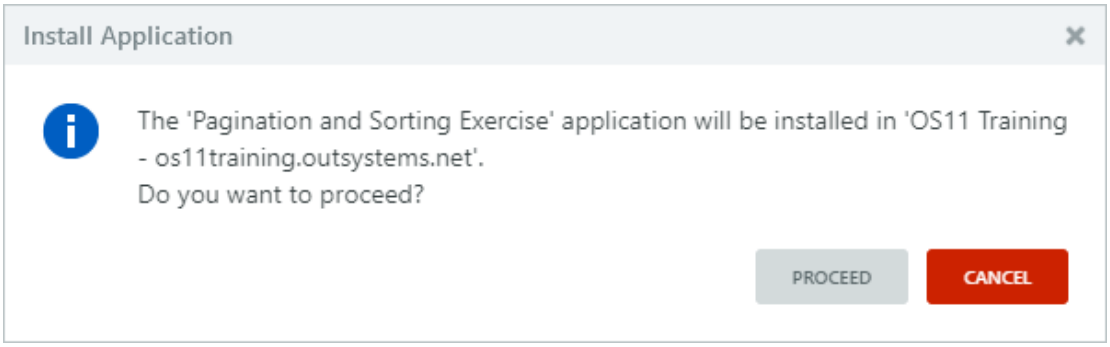
2) Select Open Files...



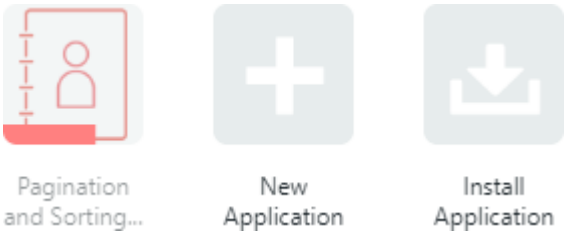
3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Pagination and Sorting.oap**



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

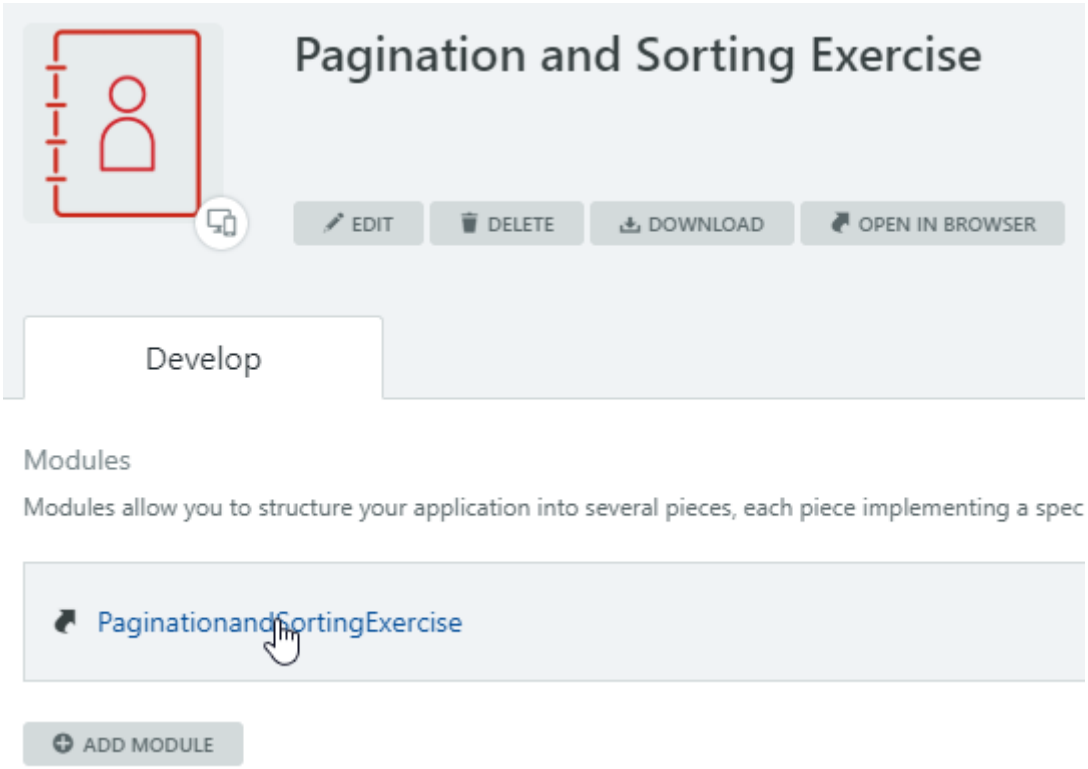


5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.

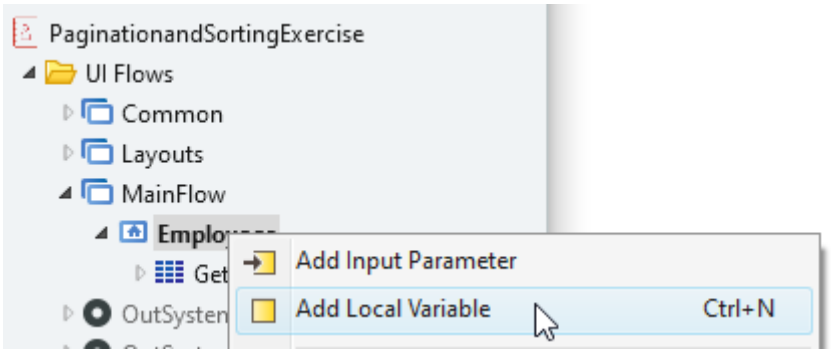
7) The application has only one module. Let's open it!



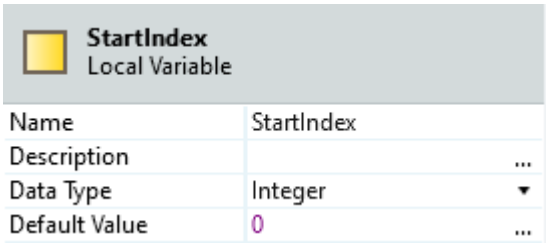
Add Pagination to a Table

In this section, we'll add pagination to the Table that displays the employee's information.

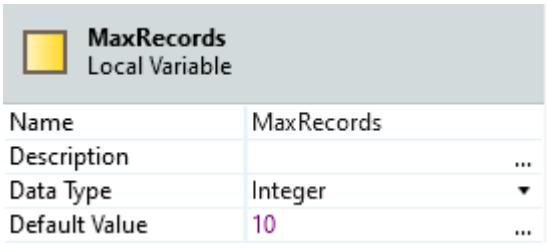
1) Right-click the Employees Screen and then select **Add Local Variable**.



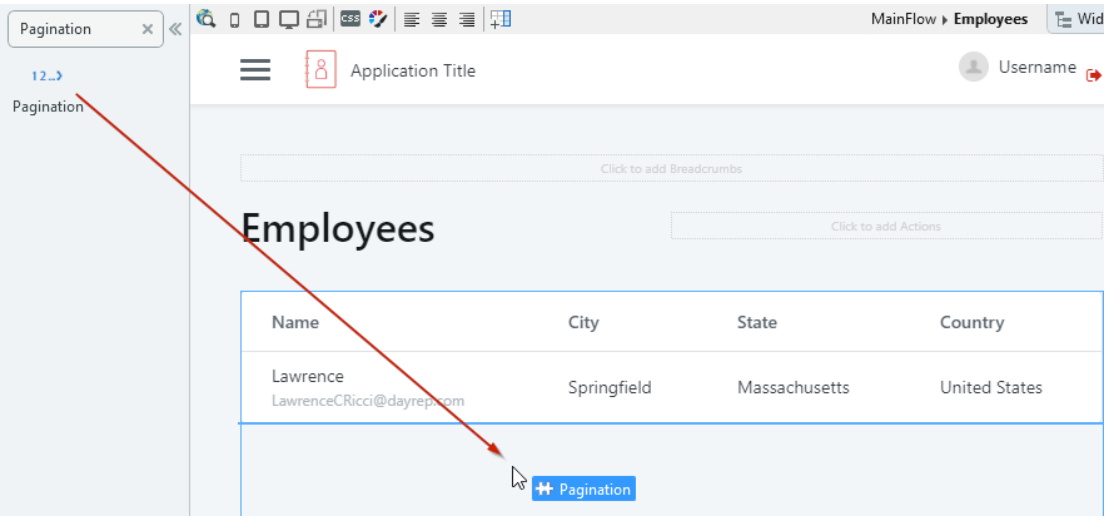
2) Set the name of the Local Variable to *StartIndex*. Make sure the **Data Type** is set to *Integer* and the **Default Value** is set to *0*.



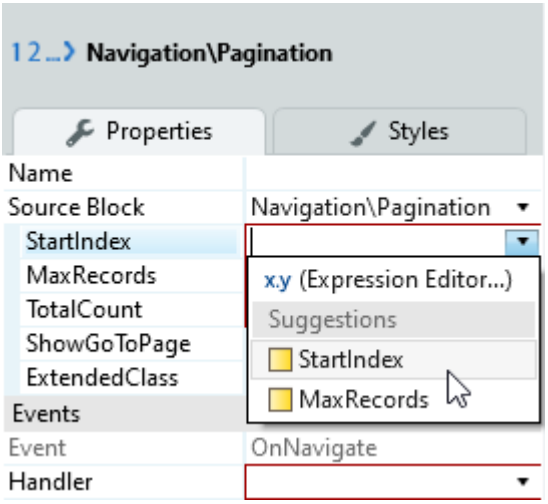
3) Add another Local Variable, and set its name to *MaxRecords*, and the **Data Type** to *Integer*. Set the **Default Value** to *10*.



4) From the toolbox to the left, drag the **Pagination** widget and drop it below the table.

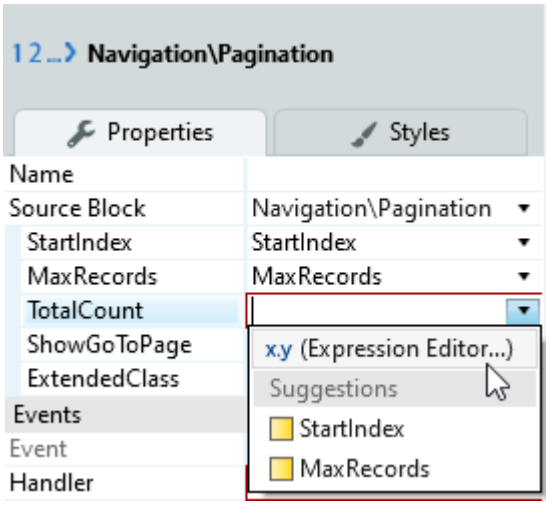


5) Set the **StartIndex** parameter to the *StartIndex* Local Variable.



6) Set the **MaxRecords** parameter to the *MaxRecords* Local Variable.

7) Open the suggestions dropdown of the **TotalCount** and select *(Expression Editor...)*.

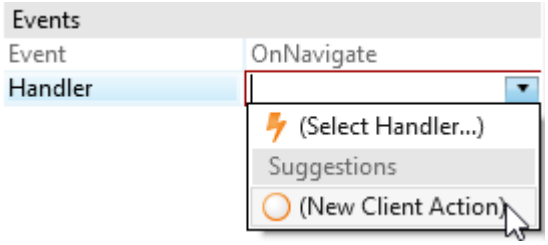


8) Set the Expression to:

```
GetEmployees.Count
```

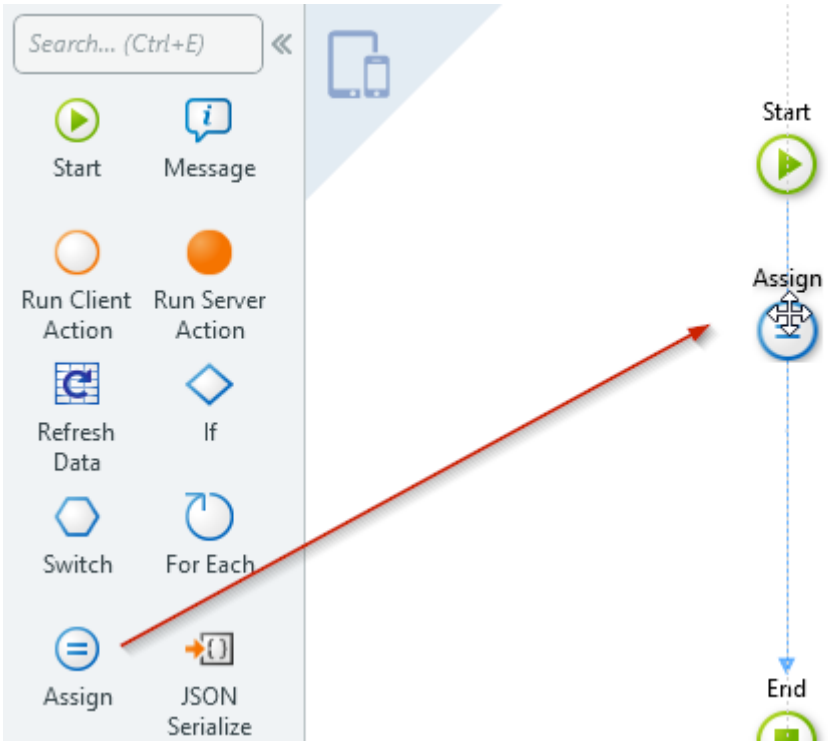
9) Click **Done** to close the Expression Editor.

10) Open the suggestions dropdown of the **On Navigate Handler**, then select *(New Client Action)*.



11) Rename the newly created Client Action to *Refresh*.

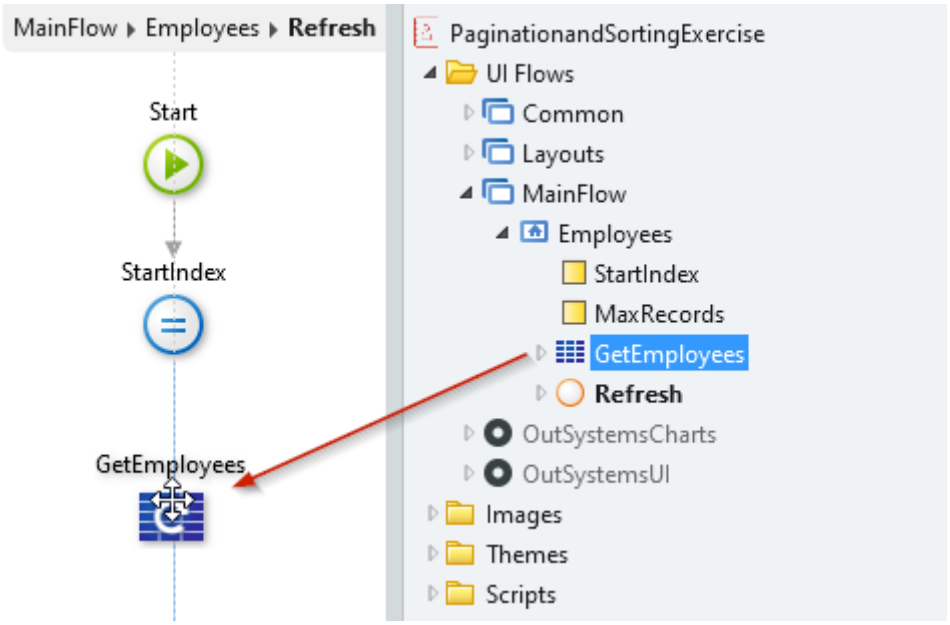
12) Drag an **Assign** and drop it after the Start.



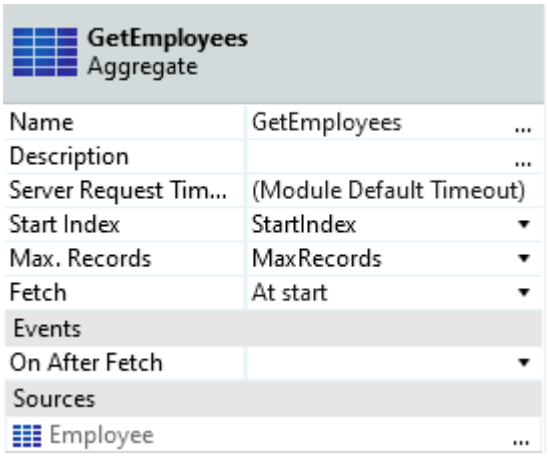
13) In the Assign, define the following assignment:

```
StartIndex
= NewStartIndex
```

14) Drag the **GetEmployees** Aggregate and drop it between the Assign and the End, to create a *Refresh Data*.



15) Select the **GetEmployees** Aggregate, then change the **Start Index** property and **Max. Records** property as follows:



16) Publish the module, then open it in the browser.

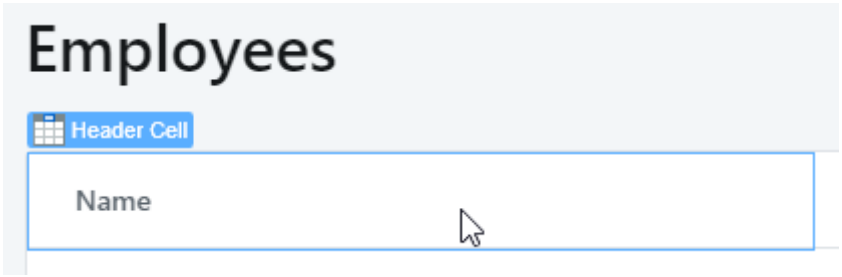
17) Navigate through the different pages and make sure the data in the Table is updated.

Add Sorting to a Table

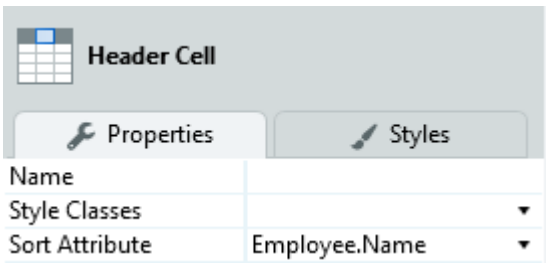
Now that we have taken care of pagination, we will now add sorting to the same Table.

1) Return to the Employees Screen.

2) Click on the **Name** Header Cell.



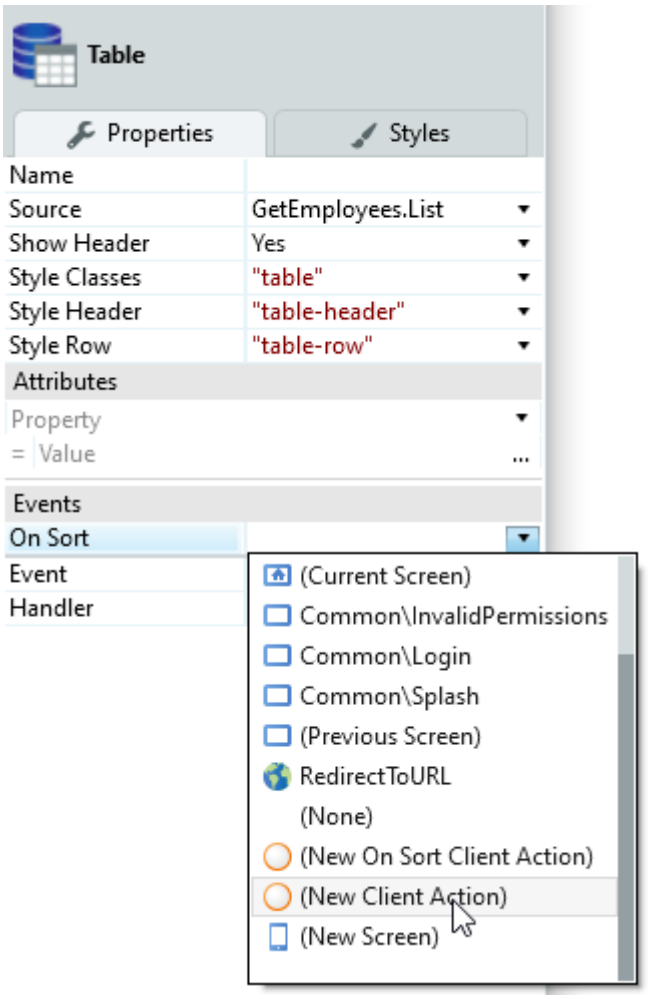
3) Set the **Sort Attribute** property to Employee.Name.



4) Repeat the previous step for the **City**, **State**, and **Country** header cells.

5) Using the breadcrumb at the bottom, select the Table.

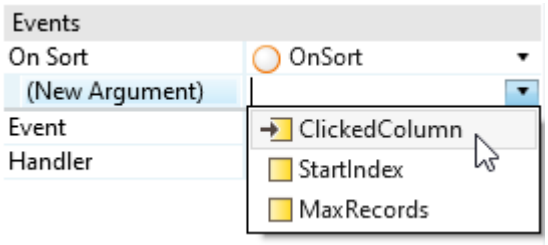
6) In the Table properties, open the Suggestions Dropdown of the **On Sort** event, then select *(New Client Action)*.



In this exercise we will create the On Sort Client Action manually by selecting the *(New Client Action)*. This will allow to better understand how the sorting logic is achieved. The *(New On Sort Client Action)* option is a Service Studio accelerator that creates the logic for you.

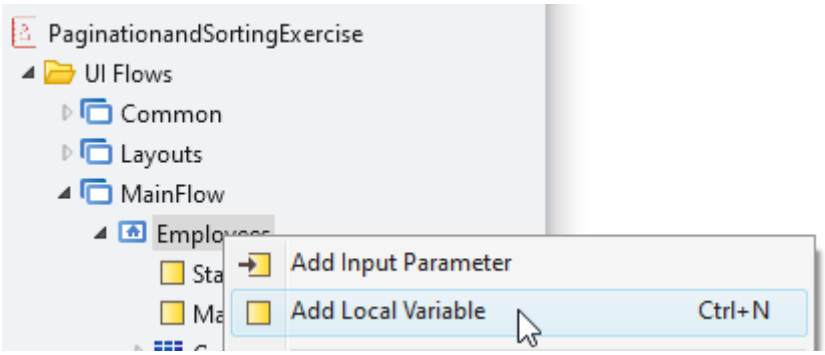
7) Click the Back Button in the toolbar to return to the Screen.

8) Select the Table again, open the Suggestions Dropdown for the **(New Argument)** input of the event, then select *ClickedColumn*.



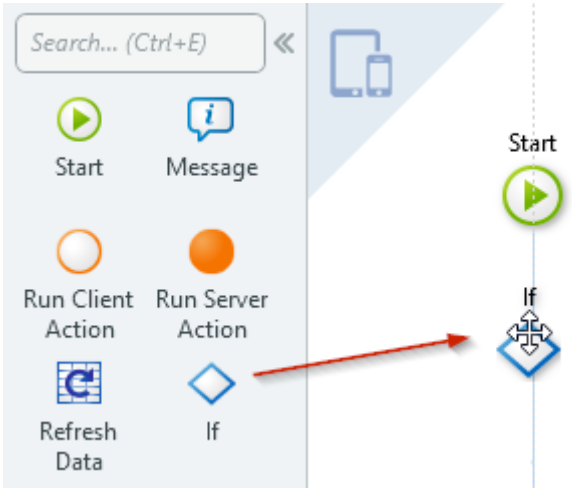
9) From the Elements tree, open the **OnSort** client action to see its flow.

10) Right-click the Employees Screen, then select **Add Local Variable**.



11) Set the variable name to *SortColumn* and make sure the **Data Type** is set to *Text*.

12) Drag an **If** and drop it after the Start in the flow.



13) Set the Condition of the If to:

```
ClickedColumn = SortColumn and ClickedColumn <> ""
```

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

15) Add an **Assign** to the right of the If, then create the **True** branch connector from the If to the Assign.

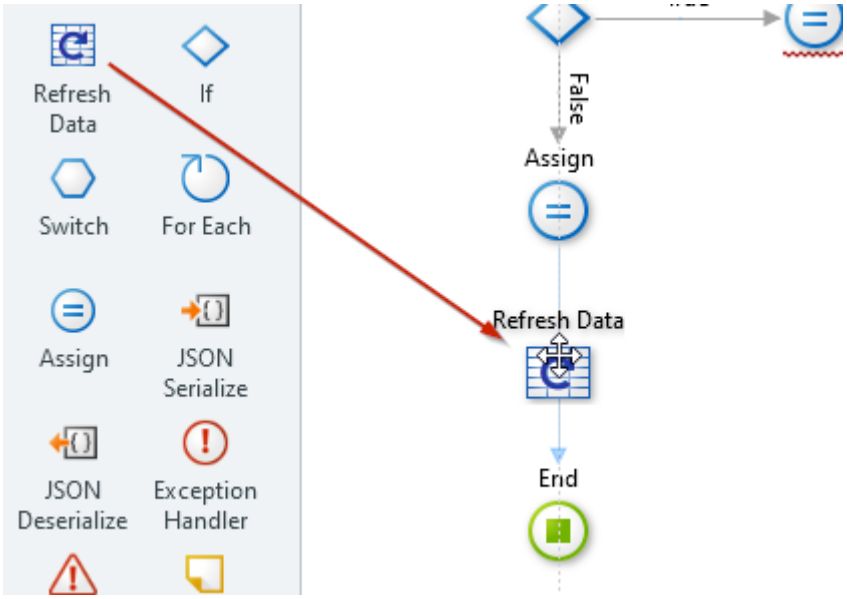
16) In the newly created assign, set the following assignments:

```
SortColumn
= ClickedColumn + " DESC"

StartIndex
= 0
```

17) Drag another Assign, and drop it in the False branch of the If, then set the following assignment:

```
SortColumn
= ClickedColumn

StartIndex
= 0
```

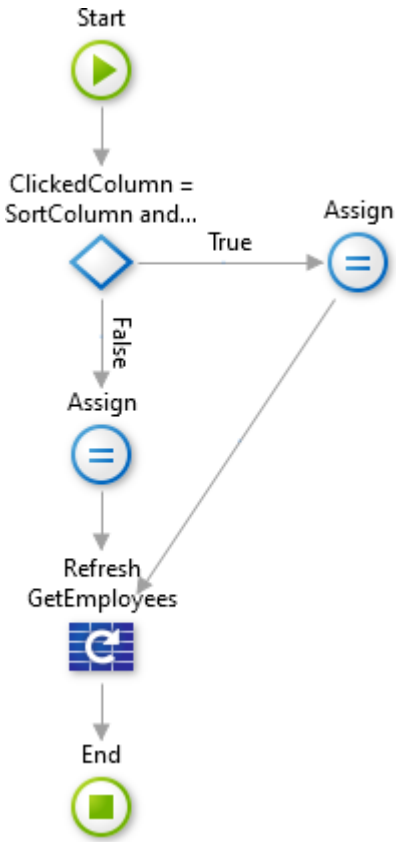
18) Drag a **Refresh Data**, and drop it in the connector between the Assign and the End.



19) In the Select Data Source dialog, select the **GetEmployees** Aggregate.

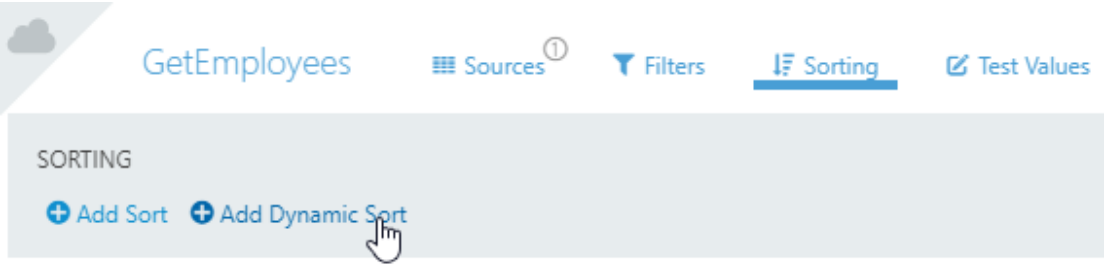
20) Create the missing connector from the right-most Assign to the Refresh Data.

21) The flow should look like this:



22) Double-click the Refresh Data to open the Aggregate.

23) Open the Sorting tab, then click **Add Dynamic Sort**



24) Set the Expression to:

```
SortColumn
```

25) Publish and test the module.

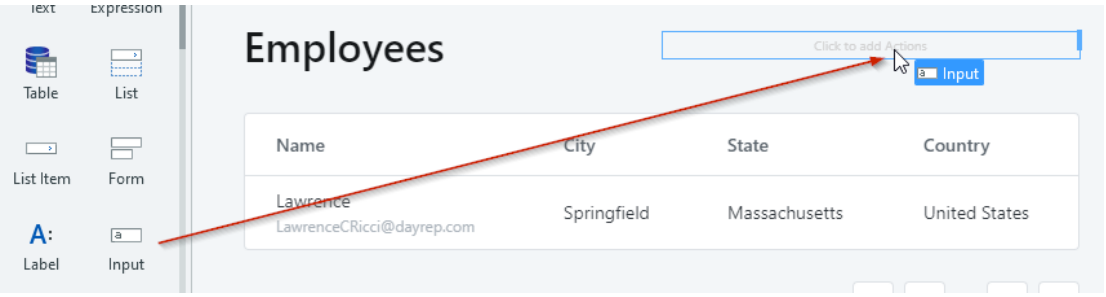
26) Click the header cells, and also test the sorting with pagination. Notice that whenever the current page is not the first, changing the Sort will switch back to the first page (StartIndex = 0).

Control the Number of Records per Page

In this section, we will extend the functionality of our Screen. So, far it already allows us to navigate between pages and sort data. In this section, we will limit the number of records shown on each page, using an input field.

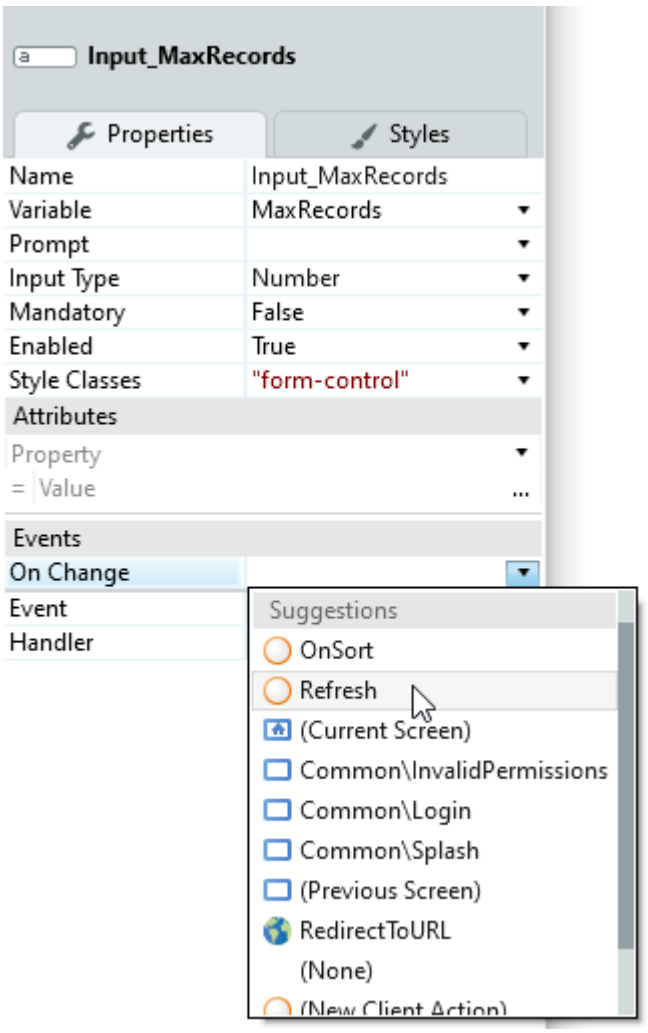
1) Switch back to the Employees Screen.

2) Drag an **Input** and drop it in the Actions placeholder.

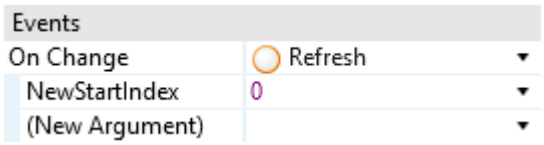


3) Set the **Variable** property to the *MaxRecords* local variable.

4) Set the **On Change** event handler of the Input to the *Refresh* client action.



5) Set the **NewStartIndex** input parameter to *0*.



6) Publish the module and test it on the browser.

7) Change the number in the Input and notice how the number of records displayed in the Table is affected.