Using Server Action Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, our main goal is to enhance logical flows by implementing Server Actions, incorporating them into the existing logic flows, and managing variables effectively.

Upon successful completion of this exercise, the application will input new employee or project data into the respective tables in the database on a click of a button. Subsequently, it will reset the Forms to a blank state, ensuring a seamless transition after the insertion of data.

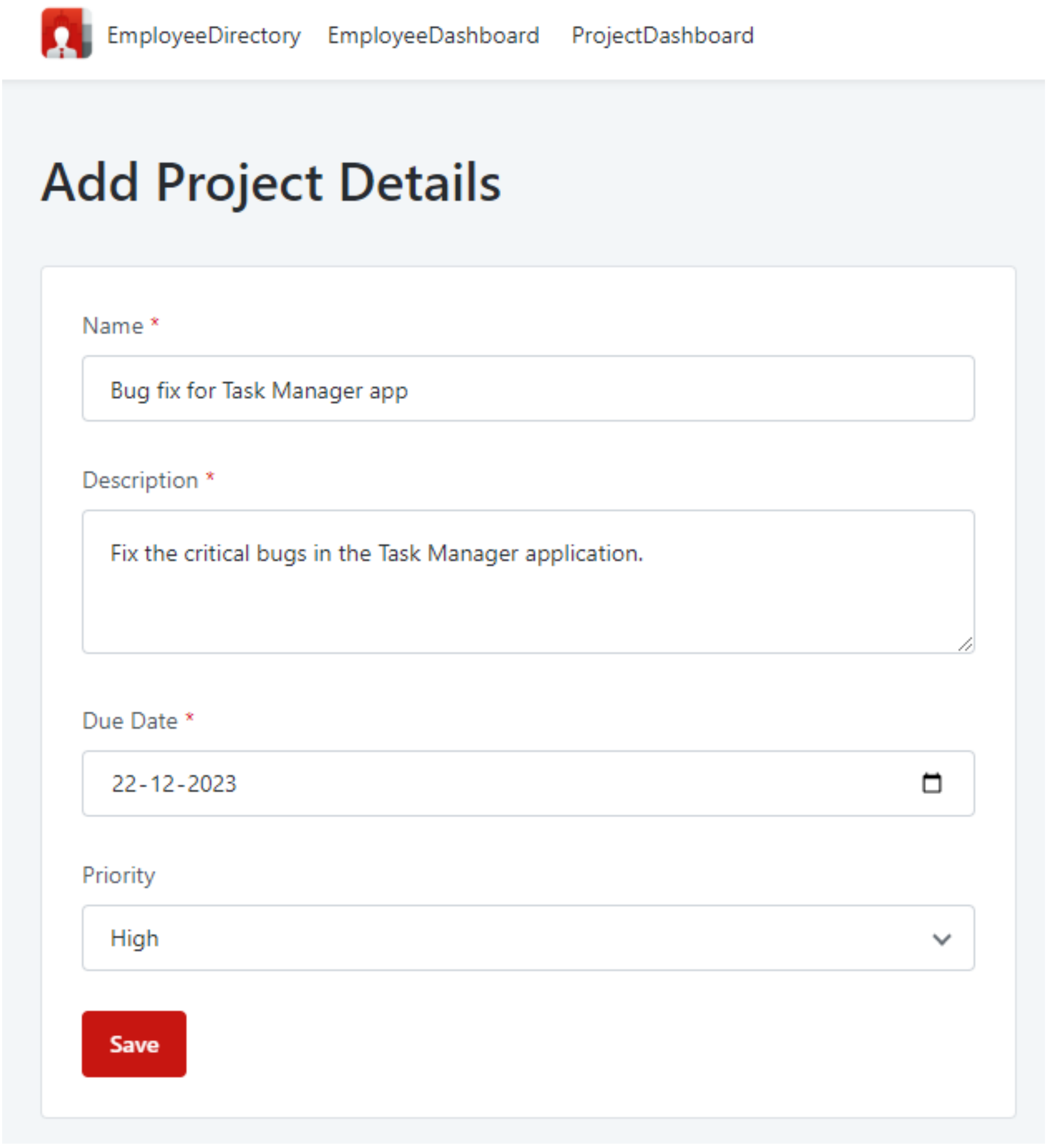
Scenario

In this exercise, you will enhance the existing **Employee Directory** app, which consists of a single module containing essential entities, screens, screen and server actions established in previous activities.

The module comprises six distinct screens, which have some Aggregates and widgets created, providing a functional user interface.

The *EmployeeDetail* and *ProjectDetail* screens are created using forms and an array of input widgets. These screens incorporate specialized Aggregates, along with both screen and server actions. In particular, the *GenerateEmailId* Client Action, defined within the *EmployeeDetails* screen, automates the process of generating email IDs in specific format for employees upon joining the organization. Additionally, the *SaveOnClick* action facilitates the addition or update of employee data by correlating with the corresponding identifier in the database.

Similarly, within the *ProjectDetail* screen, the *SaveOnClick* action plays a important role in the hassle-free addition or update of project details by aligning with the corresponding identifier in the database.



Add Project Details

Name *	Bug fix for Task Manager app
Description *	Fix the critical bugs in the Task Manager application.
Due Date *	22-12-2023
Priority	High

In this exercise, we aim to enhance the application by implementing variables in both the *EmployeeDetails* and *ProjectDetails* screens. This will facilitate the transfer of information between screens during navigation, and also ensure that the employee and project forms revert to a blank state upon launch.

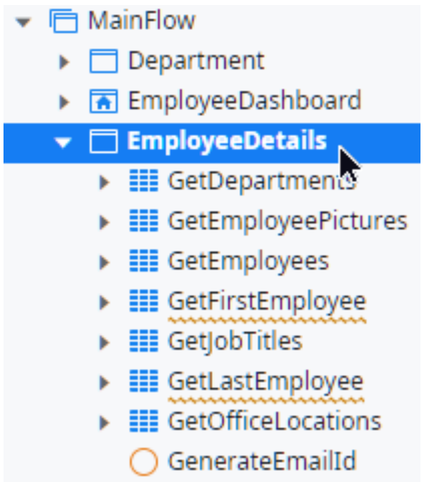
How-To

In this section, we will 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 go through it again**. If you didn't finish the exercise, that's fine! We're here to help you out.

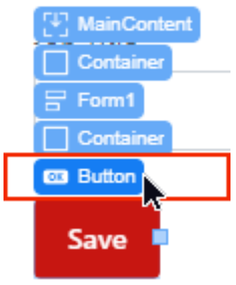
Using Server Actions in Screens

With the server actions now established, let's incorporate them into the *SaveOnClick* logic flow in the EmployeeDetails screen to efficiently initiate the creation or update of user inputs within the database.

1) Navigate to the Interface tab and open the **EmployeeDetails** page.

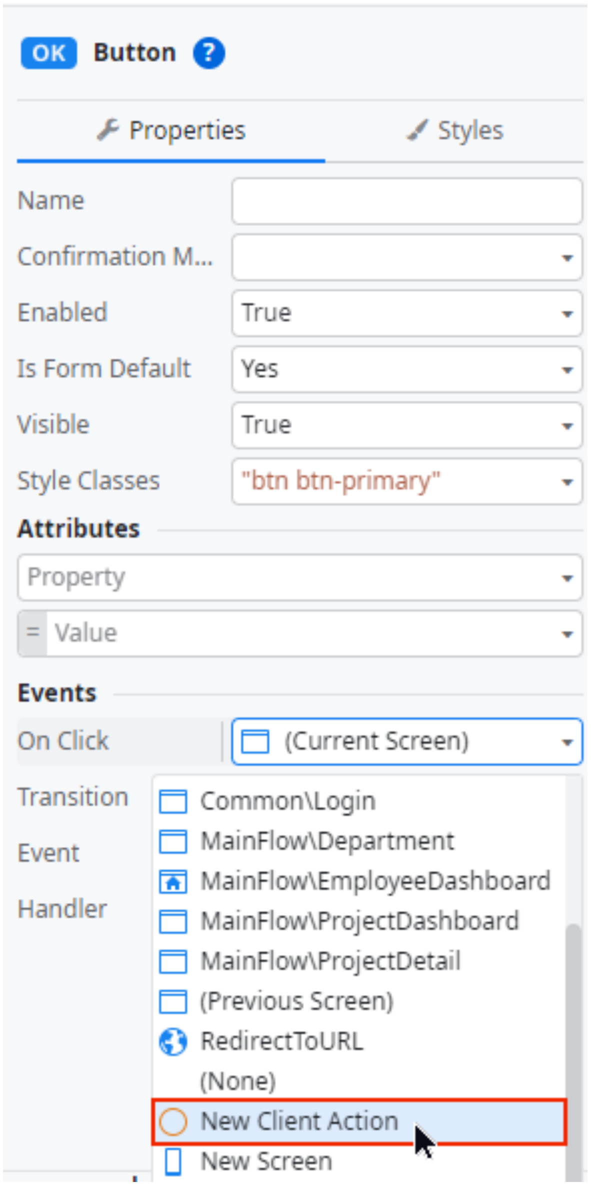


2) Click the *Save* button, and opt for **Button** from the list.



3) Choose **New Client Action** from the **On Click** property drop-down menu.

In a previous exercise, we temporarily selected *Current Screen* as a workaround to address an error. Now, in the current exercise, we will assign the specific function that the button needs to execute.



OK **Button**

Properties | Styles

Name	
Confirmation M...	
Enabled	True
Is Form Default	Yes
Visible	True
Style Classes	"btn btn-primary"

Attributes

Property

= Value

Events

On Click: (Current Screen)

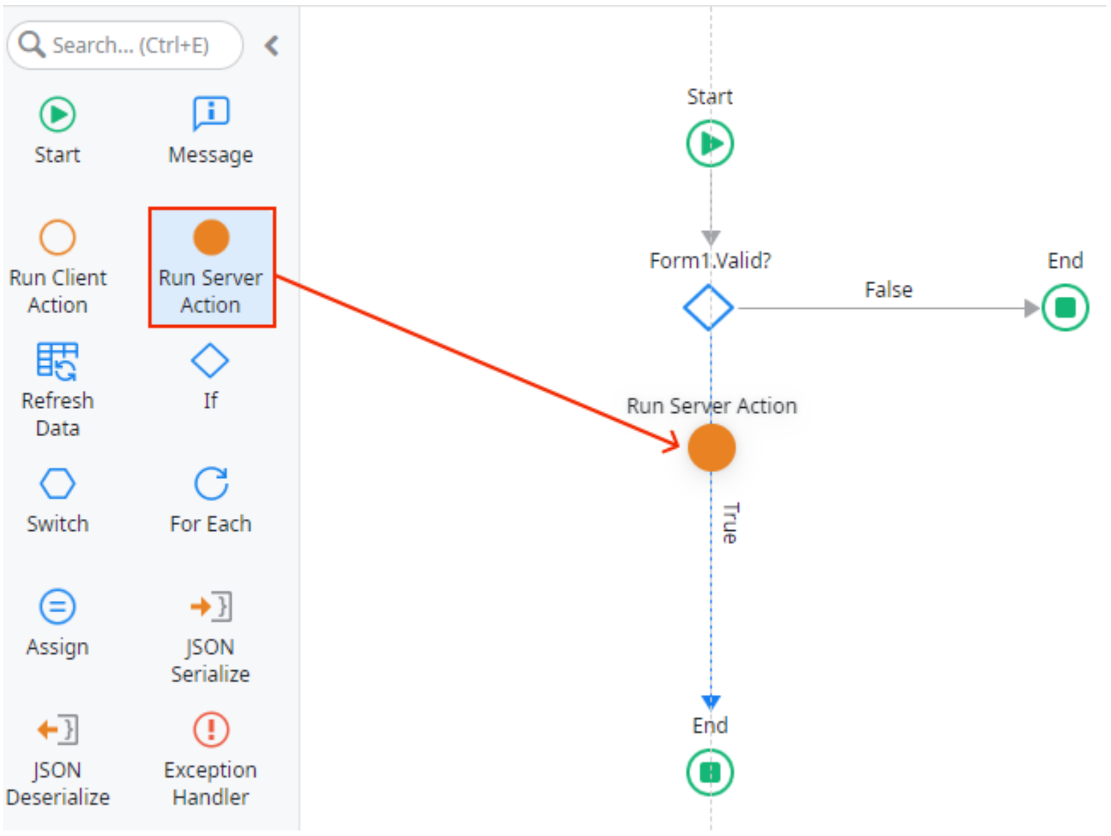
Transition

Event

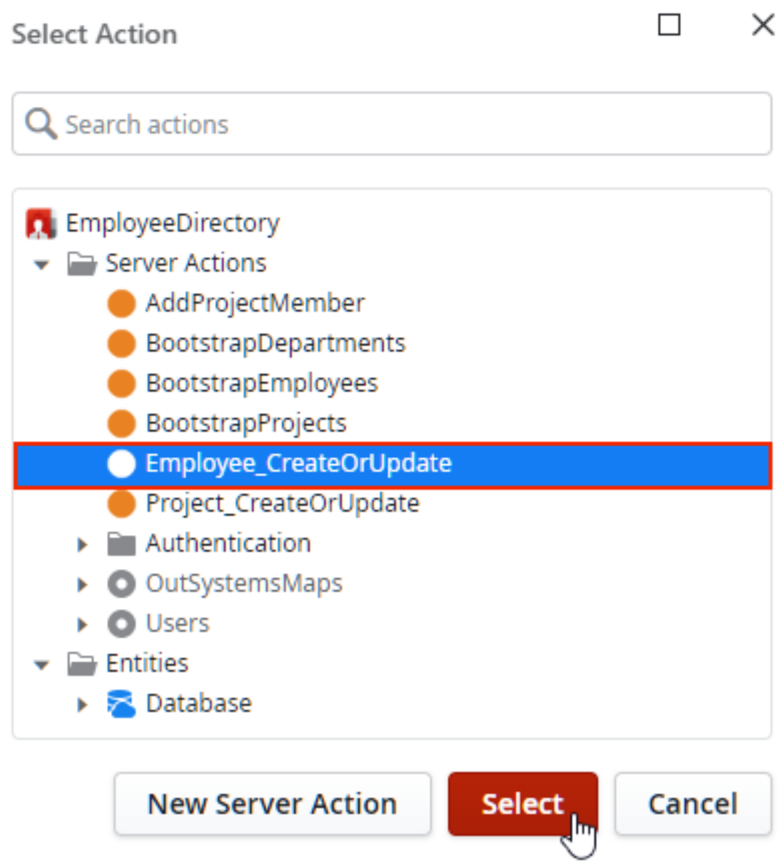
Handler

- Common\Login
- MainFlow\Department
- MainFlow\EmployeeDashboard
- MainFlow\ProjectDashboard
- MainFlow\ProjectDetail
- (Previous Screen)
- RedirectToURL
- (None)
- New Client Action
- New Screen

4) Drag and drop the **Run Server Action** below the Form validation in the logic flow.

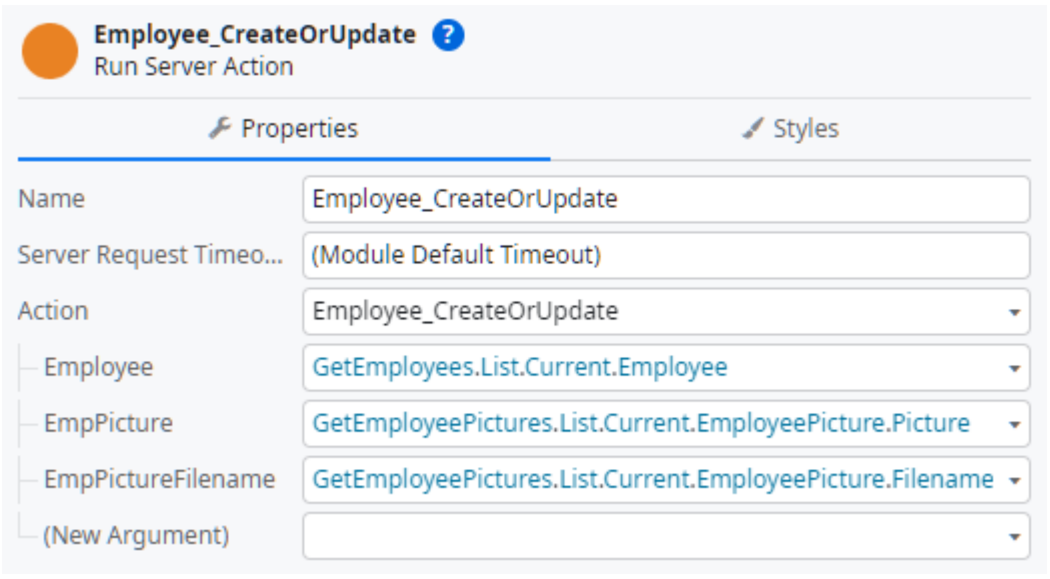


5) Select the **Employee_CreateOrUpdate** server action and confirm your choice. The server action integrates into the flow.



6) Set the **Employee** Input Parameters as follows:

```
Employee: Employee_CreateOrUpdate.EmpPicOut
EmpPicture:
GetEmployeePictures.List.Current.EmployeePicture.Picture
EmpPictureFileName:
GetEmployeePictures.List.Current.EmployeePicture.Filename
```

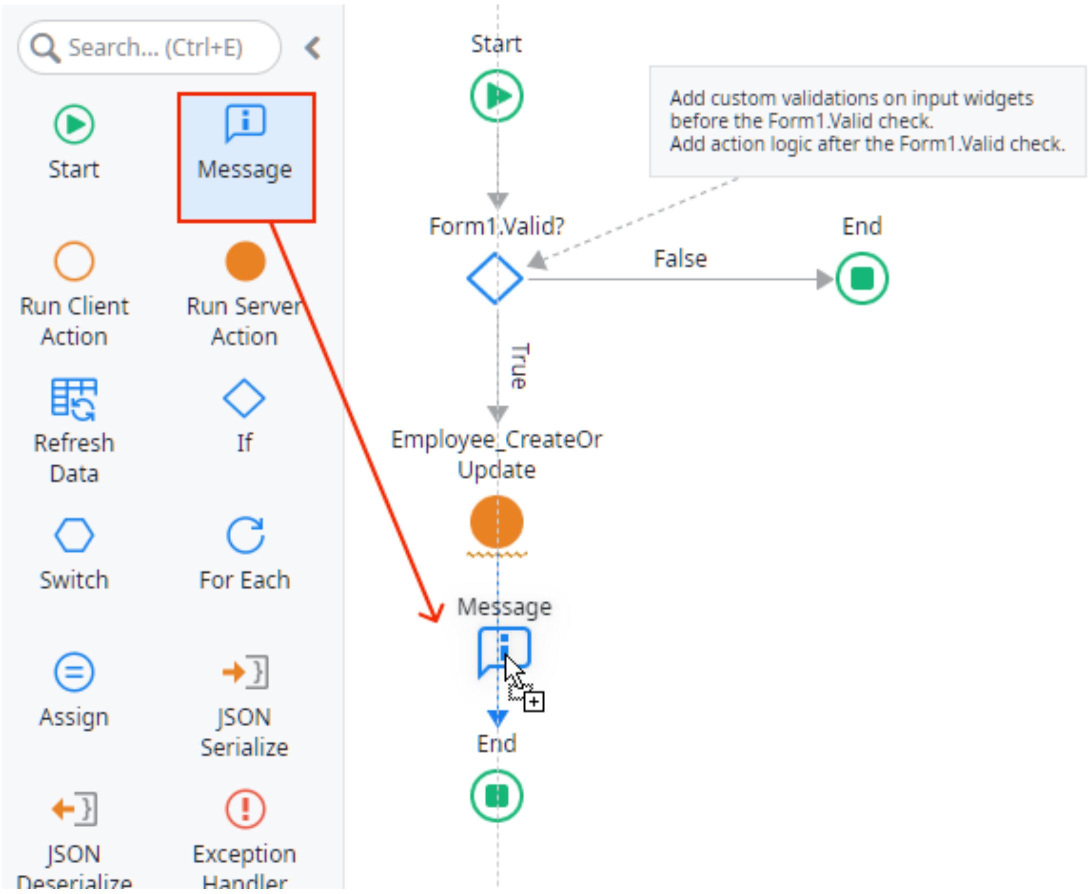


Employee_CreateOrUpdate
Run Server Action

Properties | Styles

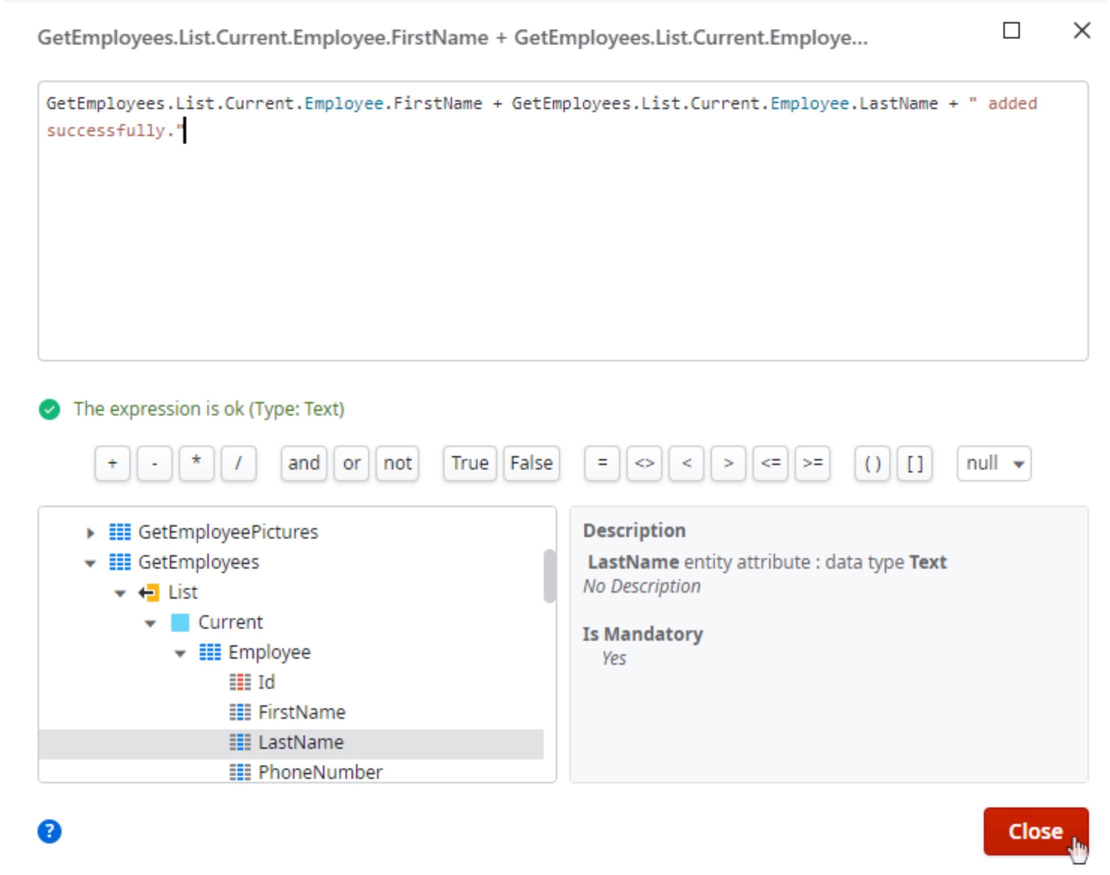
Name	Employee_CreateOrUpdate
Server Request Timeo...	(Module Default Timeout)
Action	Employee_CreateOrUpdate
Employee	GetEmployees.List.Current.Employee
EmpPicture	GetEmployeePictures.List.Current.EmployeePicture.Picture
EmpPictureFilename	GetEmployeePictures.List.Current.EmployeePicture.Filename
(New Argument)	

7) Add a **Message** widget below the last server action to confirm the record has been added, if successful.



8) Double-click the Message property and enter the message provided below.

```
GetEmployees.List.Current.Employee.FirstName +
GetEmployees.List.Current.Employee.LastName + " added
successfully."
```



GetEmployees.List.Current.Employee.FirstName + GetEmployees.List.Current.Employe...

```
GetEmployees.List.Current.Employee.FirstName + GetEmployees.List.Current.Employee.LastName + " added
successfully."
```

The expression is ok (Type: Text)

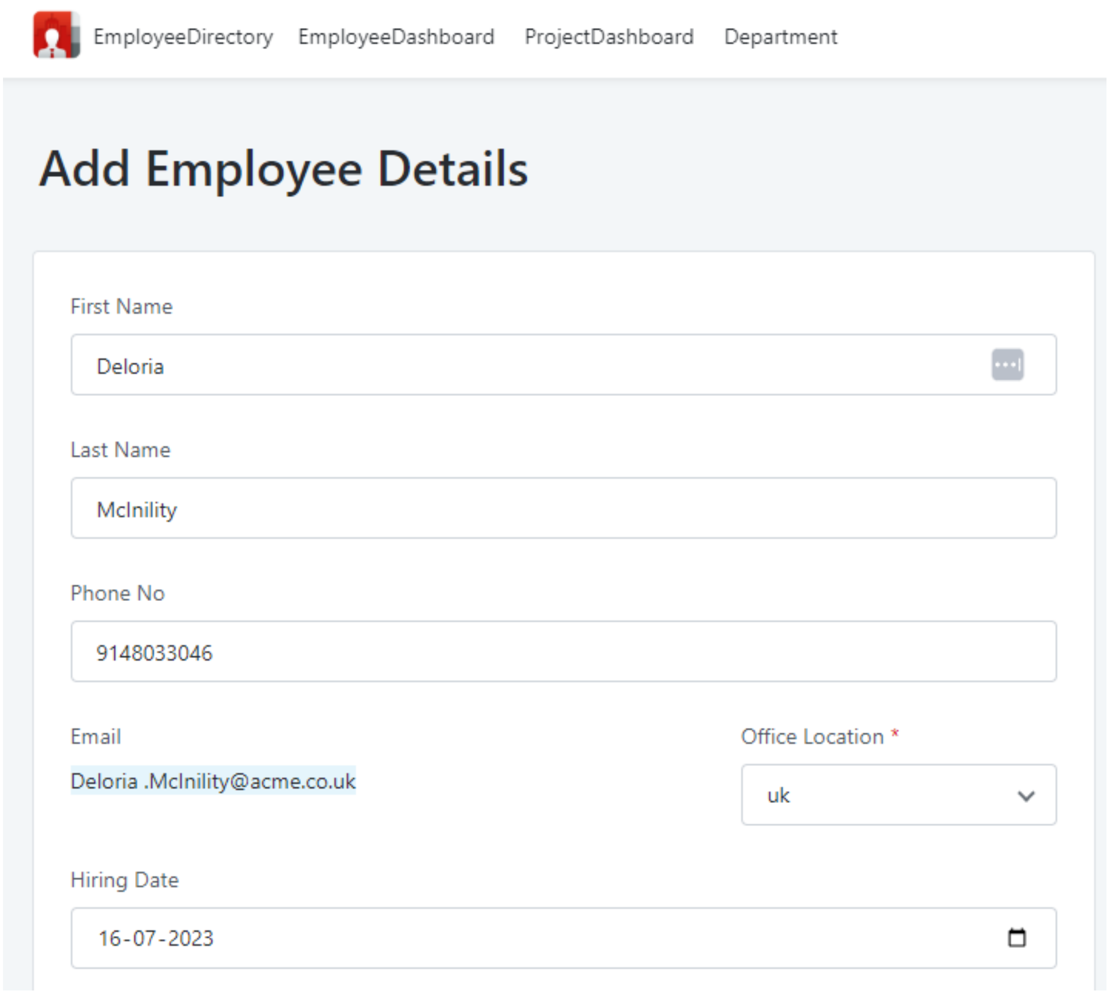
Description
LastName entity attribute : data type **Text**
No Description

Is Mandatory
Yes

9) Publish the module to save the changes.

10) Let's open the application in the browser to insert new employee data in the form.

Notice that the form displays the first record of the Employee Entity. To enter data, we need a blank state. To achieve this, let's return to Service Studio.



Add Employee Details

First Name	Deloria
Last Name	McInility
Phone No	9148033046
Email	Deloria .McInility@acme.co.uk
Office Location *	uk
Hiring Date	16-07-2023

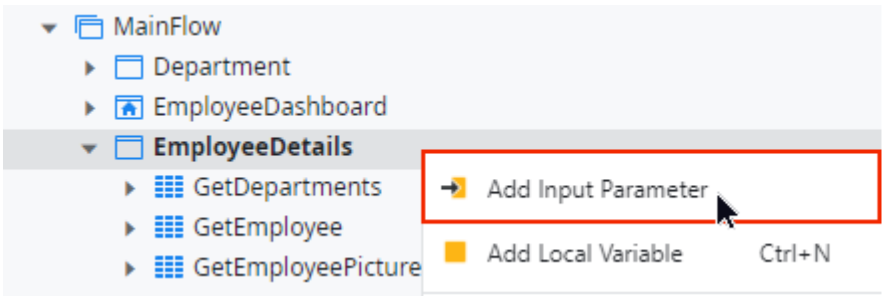
Using Variables

With the SaveOnClick logic flow now firmly in place, this segment will guide you through:

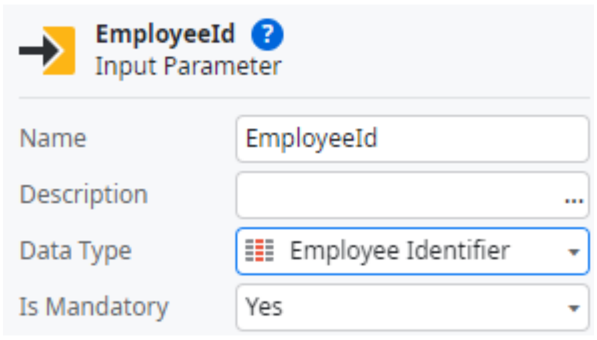
- Ensuring that the Screen displays a blank form upon launch

- Resetting the form to a blank state after successful data insertion

1) In the Interface tab, right-click on **EmployeeDetails** Screen and select **Add Input Parameter**.

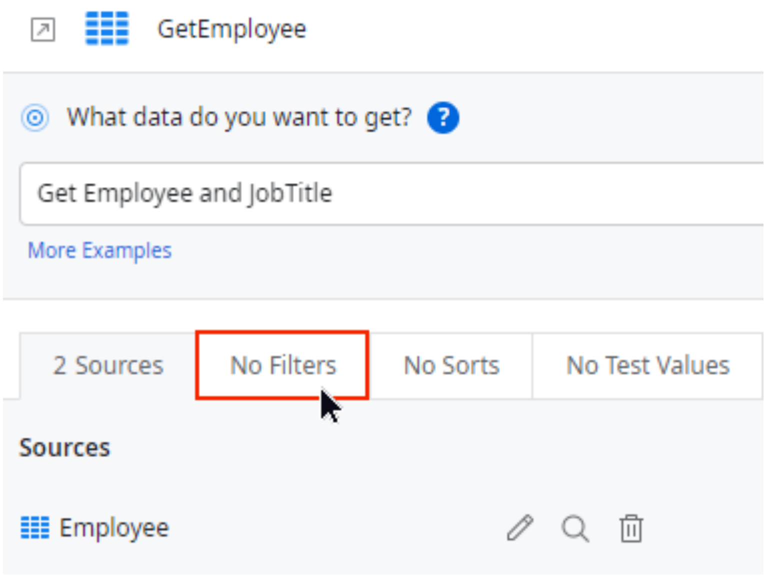


2) Set the name of the input parameter to *EmployeeId* and the Data Type to **Employee Identifier**.

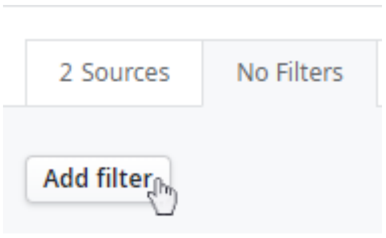


Name	EmployeeId
Description	
Data Type	Employee Identifier
Is Mandatory	Yes

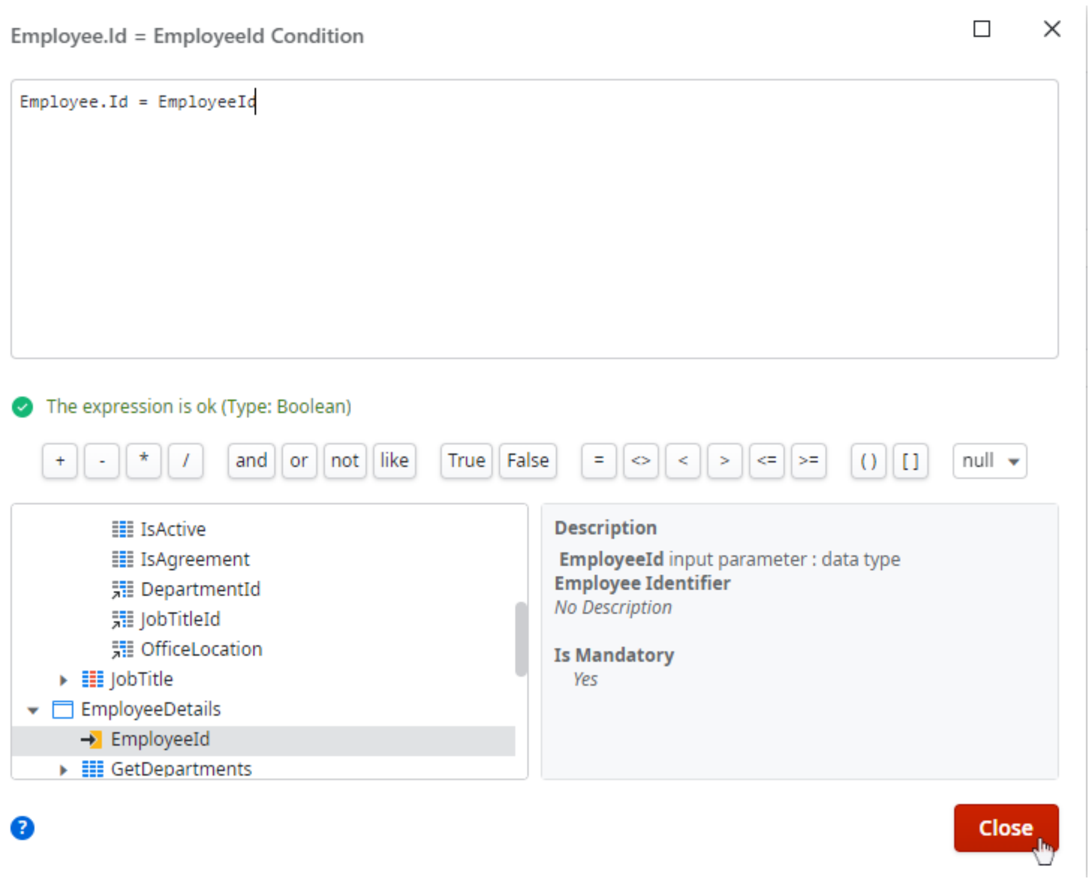
3) Open the **GetEmployees** Aggregate and navigate to the **No Filters** tab.



4) Click **Add filter**.



5) Set the filter condition to `Employee.Id = EmployeeId` and close the window.



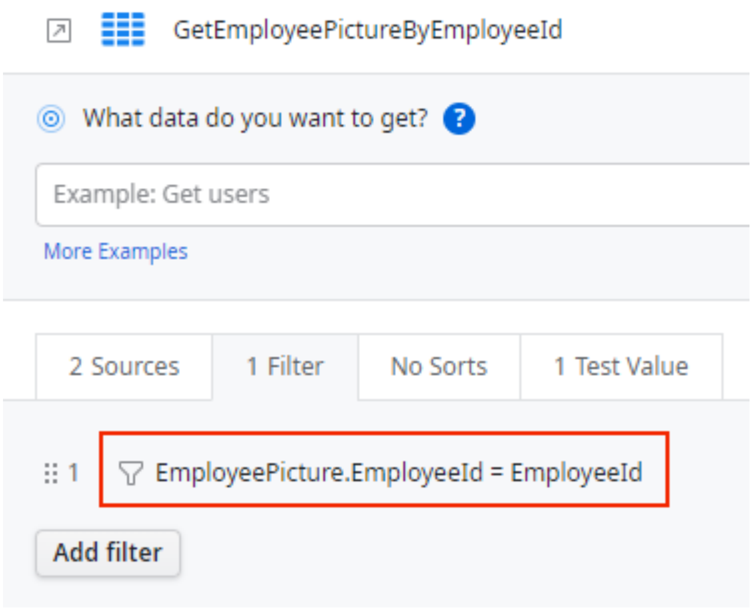
```
Employee.Id = EmployeeId
```

The expression is ok (Type: Boolean)

Description
EmployeeId input parameter : data type **Employee Identifier**
No Description

Is Mandatory
Yes

6) Similarly, open the **GetEmployeePicture** Aggregate, and in the **Filter** tab, add the condition `EmployeePicture.EmployeeId = EmployeeId`


GetEmployeePictureByEmployeeId

What data do you want to get?

Example: Get users

More Examples

2 Sources | 1 Filter | No Sorts | 1 Test Value

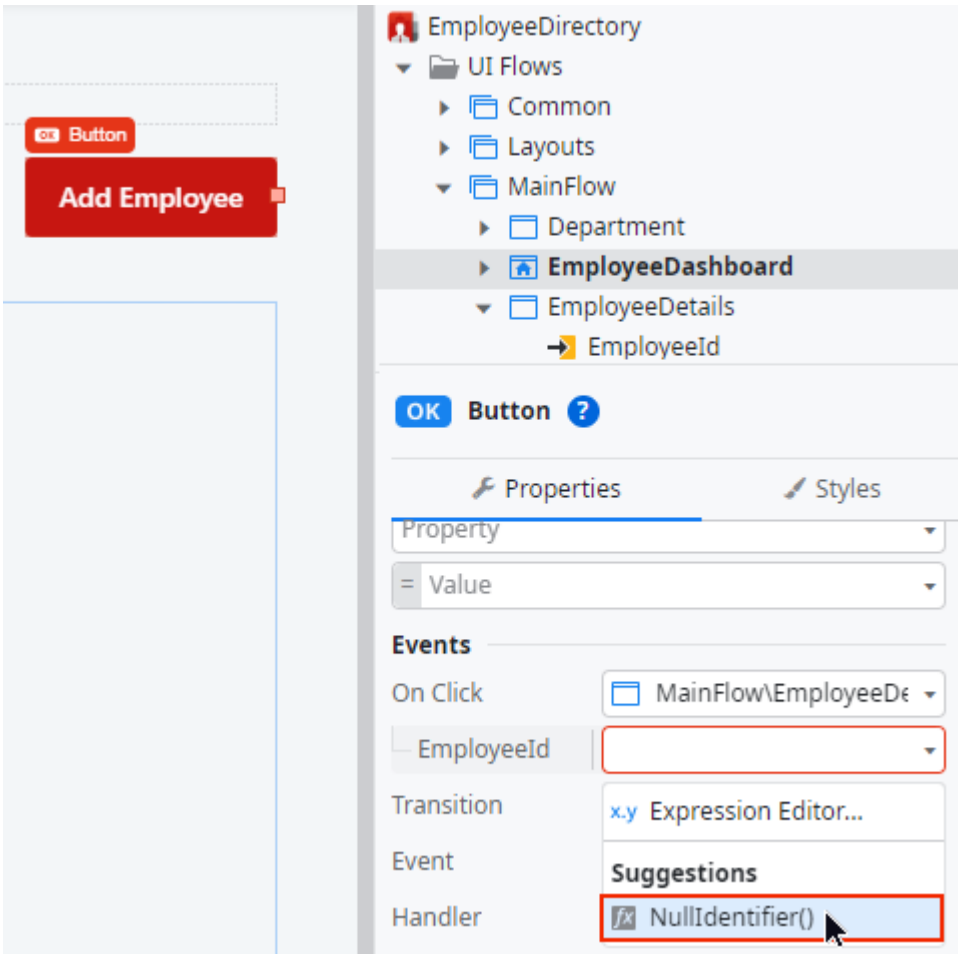
1 EmployeePicture.EmployeeId = EmployeeId

Add filter

Please note that an error has occurred due to the *EmployeeDashboard* being linked to the *EmployeeDetails* Screen. To resolve this issue, it is essential to specify the value of the *EmployeeId* input parameter for the Add Employee button on the dashboard Screen.

7) Open the **EmployeeDashboard** Screen.

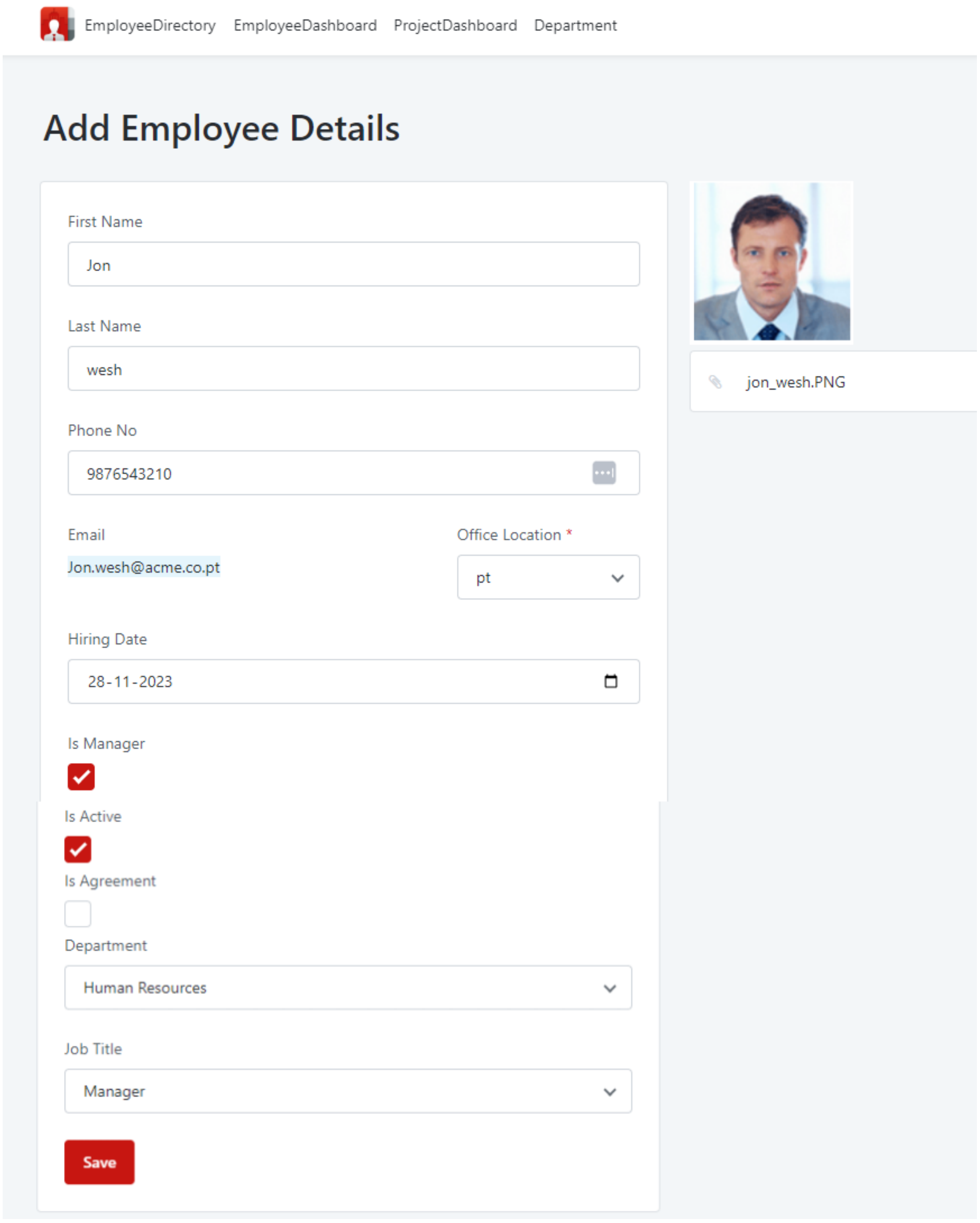
8) Select the Add Employee button, then set the *EmployeeId* input parameter to `NullIdentifier()`



9) Publish the module to save the changes.

10) Open the application in the browser and navigate to the *EmployeeDetails* Screen. You will see a clear form with no filled data.

11) Fill in the required details in each field and then click **Save**.



Add Employee Details

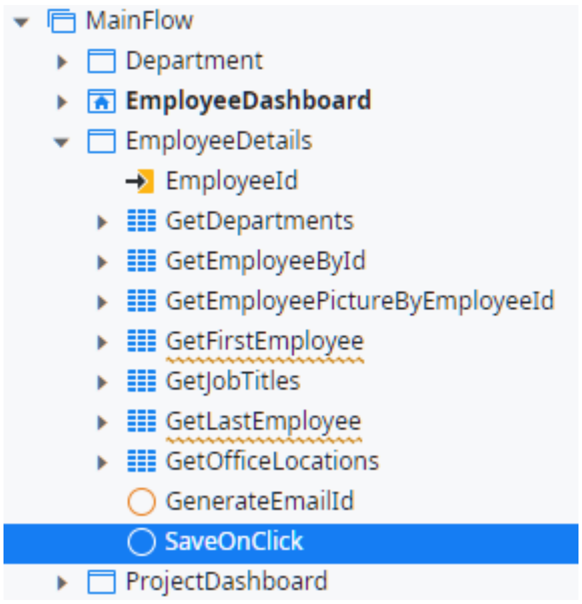
First Name	Jon
Last Name	wesh
Phone No	9876543210
Email	Jon.wesh@acme.co.pt
Office Location *	pt
Hiring Date	28-11-2023
Is Manager	✓
Is Active	✓
Is Agreement	
Department	Human Resources
Job Title	Manager

jon_wesh.PNG

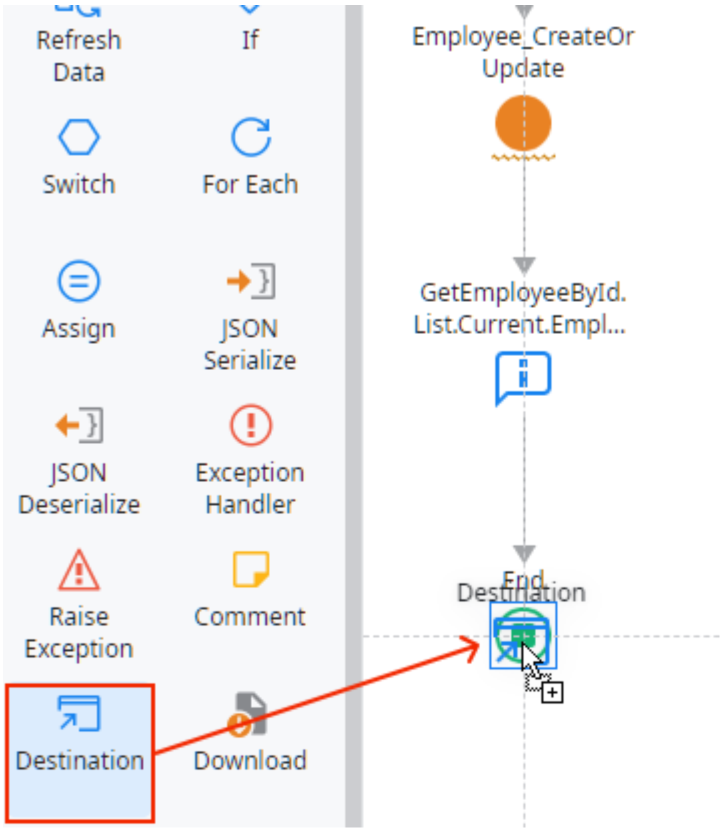
Upon successful submission, a confirmation message set up within the application will be displayed. Please observe that the form remains unrefreshed, retaining the inputted data within it.

After the data is successfully inserted into the database, we want the form to automatically refresh to blank state to facilitate the addition of the next employee's details.

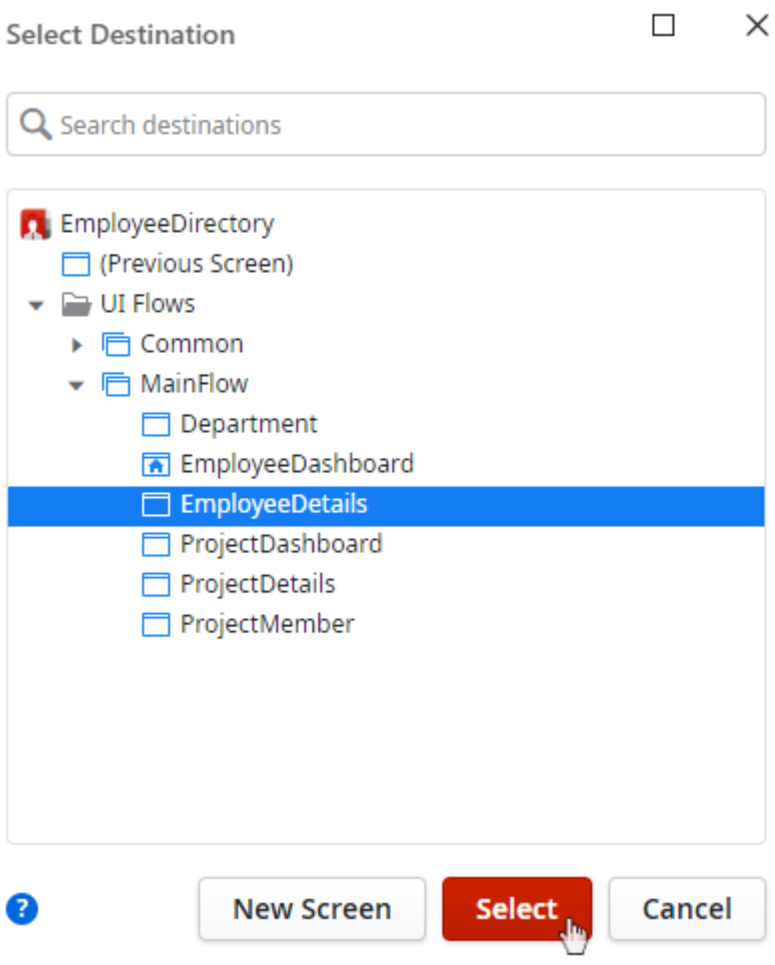
12) Return to **Service Studio**, and double-click the **SaveOnClick** action of the EmployeeDetails Screen to open the logic flow.



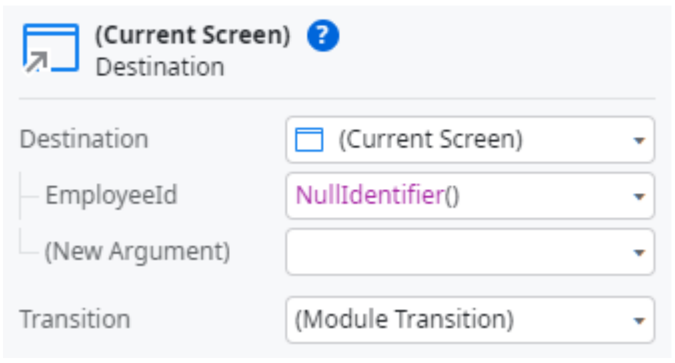
13) Drag the **Destination** element and drop it on top of **End**.



14) Select the **EmployeeDetails** Screen as the destination.



15) Set the EmployeeId value to `NullIdentifier()` in the **Current Screen** properties window.



Destination	(Current Screen)
EmployeeId	NullIdentifier()
(New Argument)	
Transition	(Module Transition)

16) Publish the module and launch the EmployeeDetails Screen in your web browser.

Notice that after saving the inputted data, the form automatically resets to its initial blank state.

17) To confirm the successful insertion of the data to the database, return to the EmployeeDashboard page and notice that the new employee is added to the table.

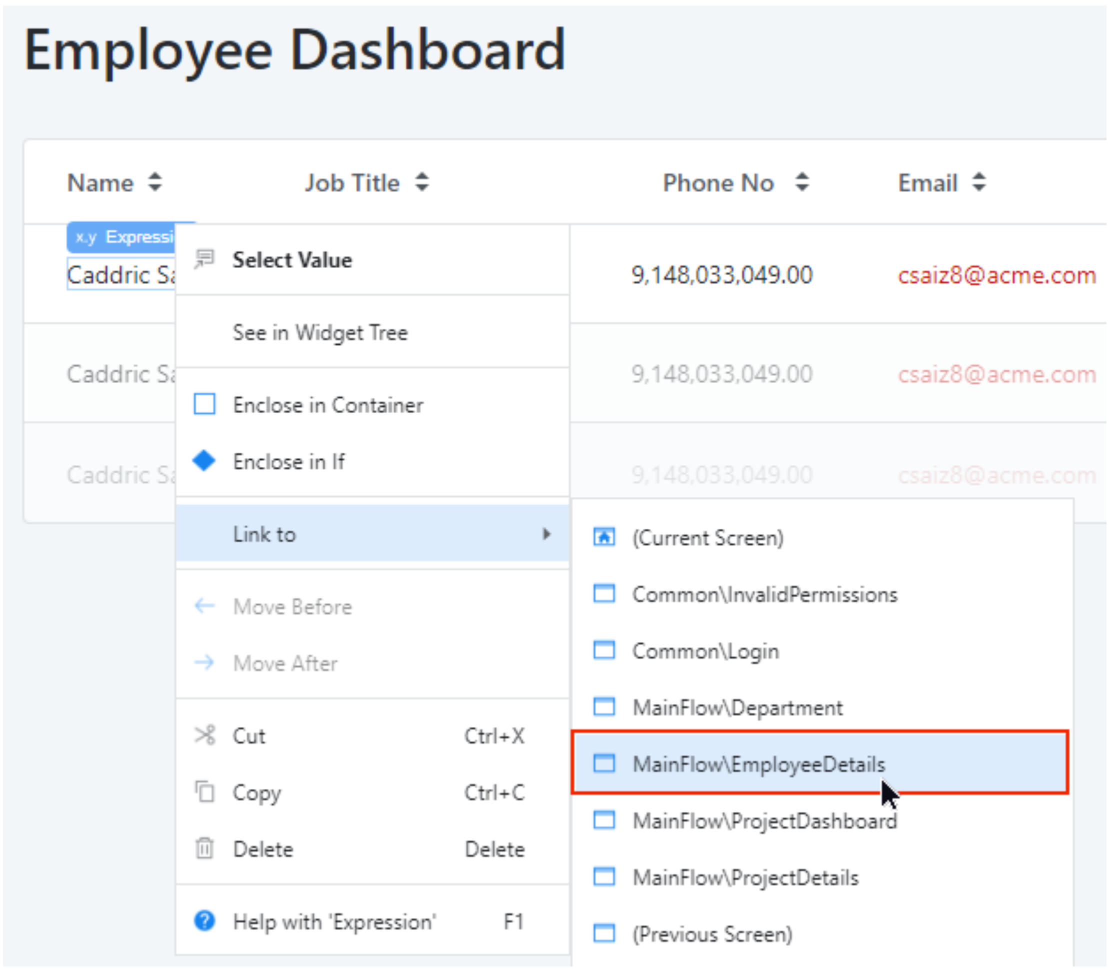
View and Update Employee Details

In this section, we will establish a connection between the employees listed on the EmployeeDashboard Screen and the EmployeeDetails Screen, enabling users to view employee details in the EmployeeDetails Screen, with the option for further updates.

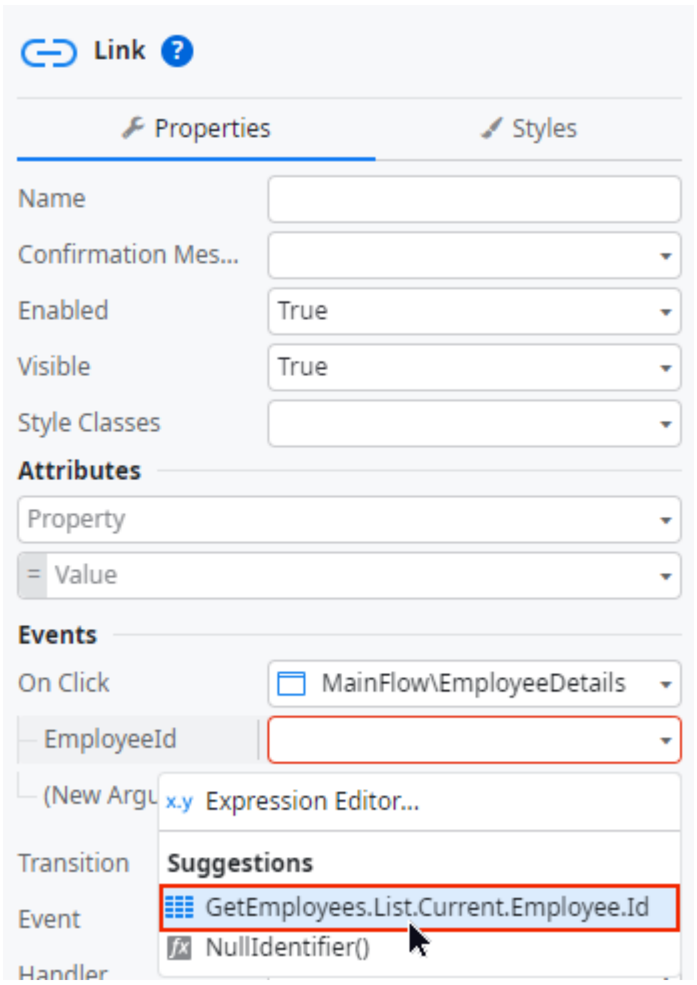
1) Open the *EmployeeDashboard* Screen in the Interface tab.

2) Select the **Expression** with the Name, by right-clicking the *Caddric Saiz* on the Table.

3) Select **Link to > MainFlow\EmployeeDetails** in the menu that is displayed.



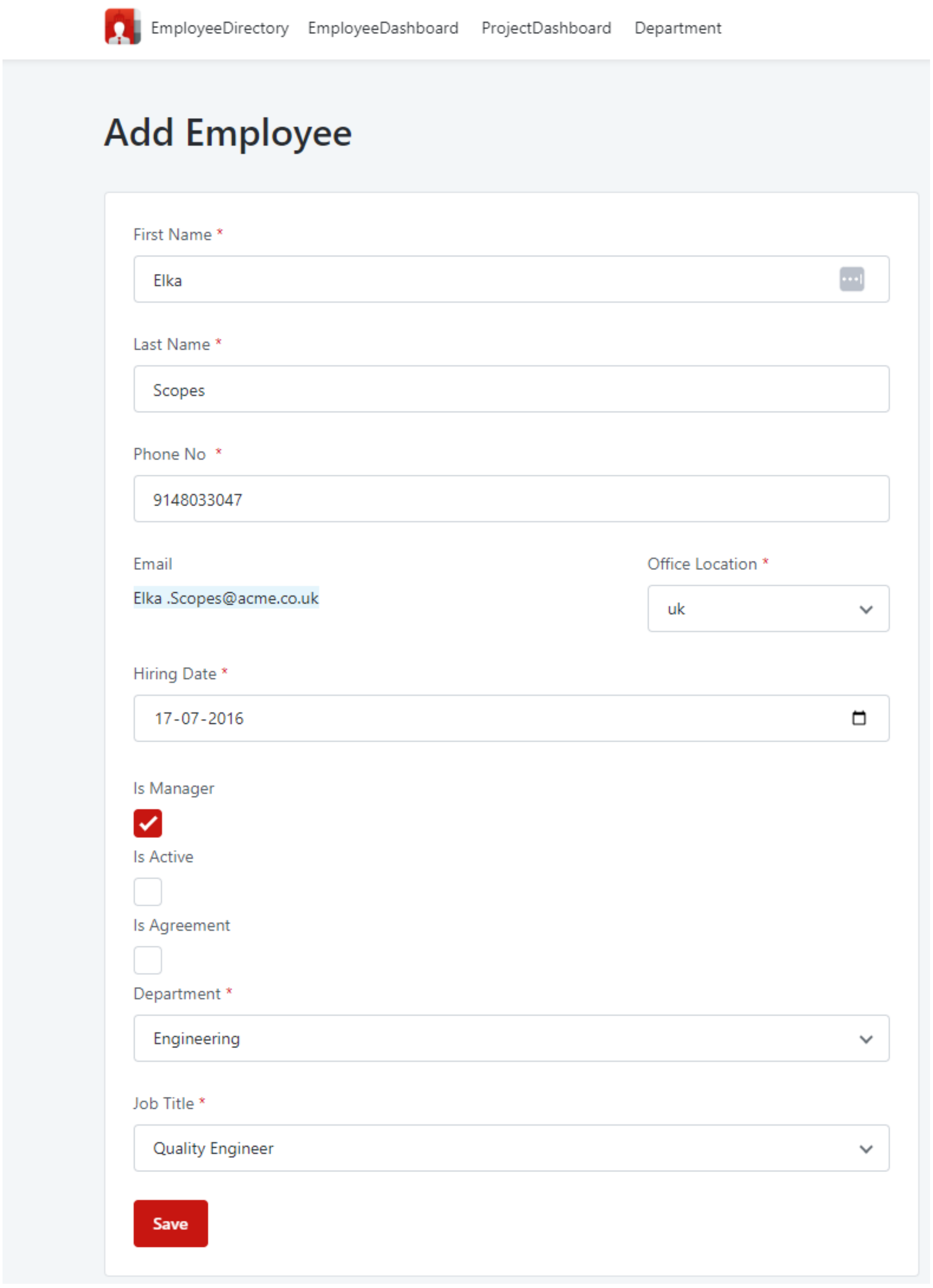
4) On the properties of the recently created Link, set the value of the **EmployeeId** input parameter of the **OnClick** to *GetEmployees.List.Current.Employee.Id*.



This makes sure that the Id of the Employee clicked on is passed to the EmployeeDetails Screen. This Id is useful to fetch the details of the employee in the EmployeeDetails.

5) Publish the module to the server to save the latest changes, and open the application in the browser.

6) Click on a employee and make sure that it opens the EmployeeDetails Screen with the details of the selected employee.



Add Employee

First Name *	Elka
Last Name *	Scopes
Phone No *	9148033047
Email	Elka .Scopes@acme.co.uk
Office Location *	uk
Hiring Date *	17-07-2016
Is Manager	✓
Is Active	
Is Agreement	
Department *	Engineering
Job Title *	Quality Engineer

7) Change the Department of the selected employee and click Save.

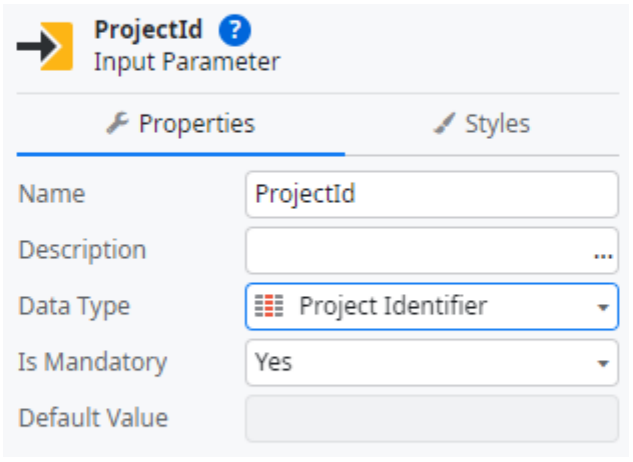
8) Return to the EmployeeDashboard page and ensure that the Department is updated for the employee in the table.

Use Server Actions in the Project Detail Screen

In this segment, we'll incorporate the **Project_CreateOrUpdate** server action to the 'SaveOnClick' action in the ProjectDetails Screen and use variables to aid in pass information between screens during navigation.

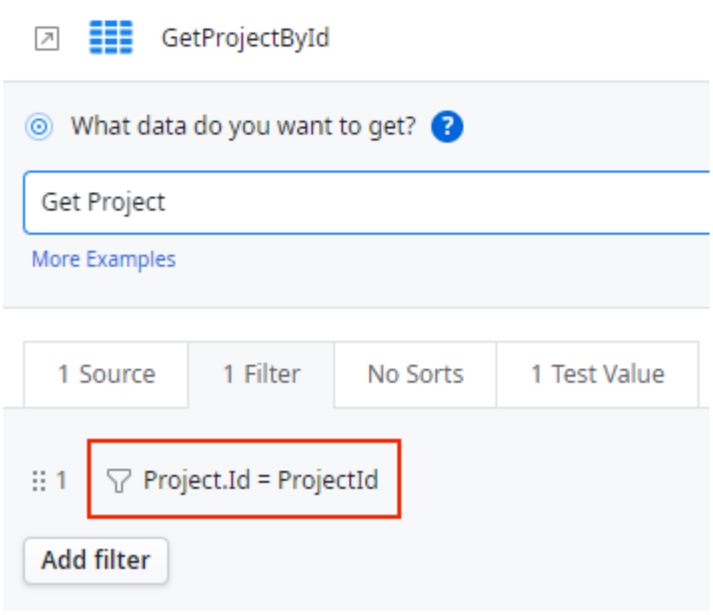
1) Open the ProjectDetails Screen in the Interface tab.

2) Create an input parameter named *ProjectId* for the ProjectDetails Screen, and set its Data Type as **Project Identifier**.



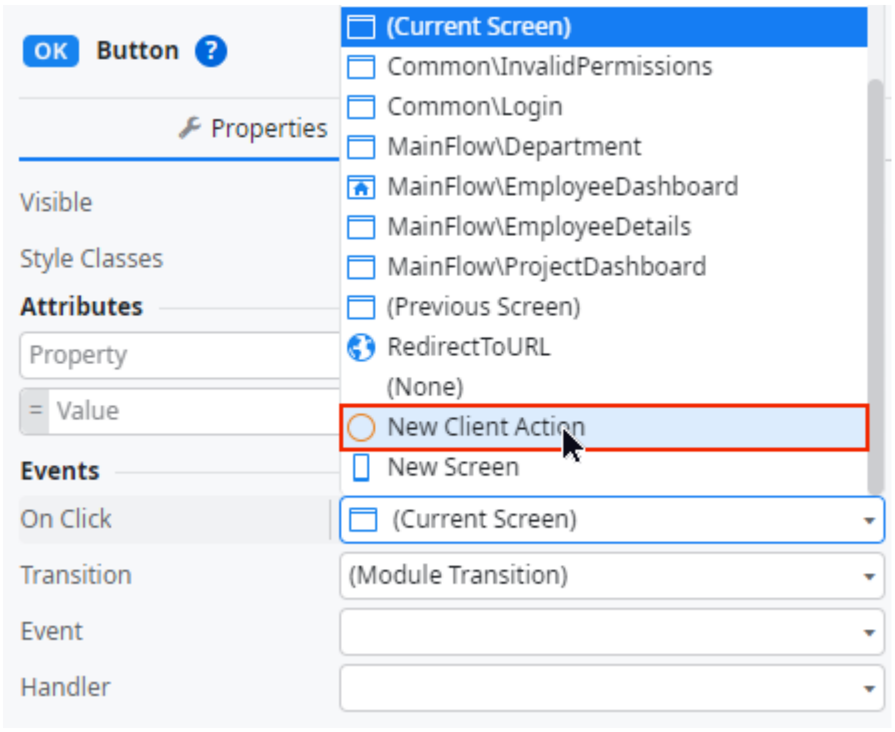
Name	ProjectId
Description	
Data Type	Project Identifier
Is Mandatory	Yes
Default Value	

3) Modify the **GetProjects** Aggregate by setting the Filter condition to `Project.Id = ProjectId`

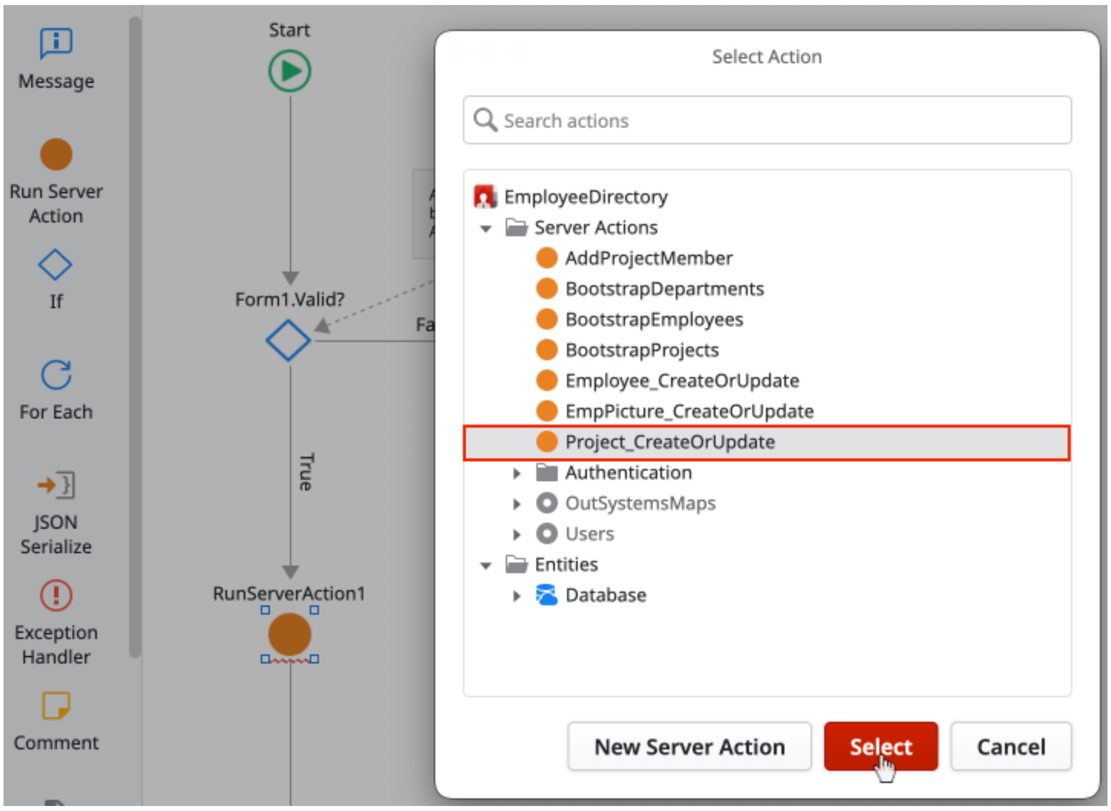


1 Project.Id = ProjectId

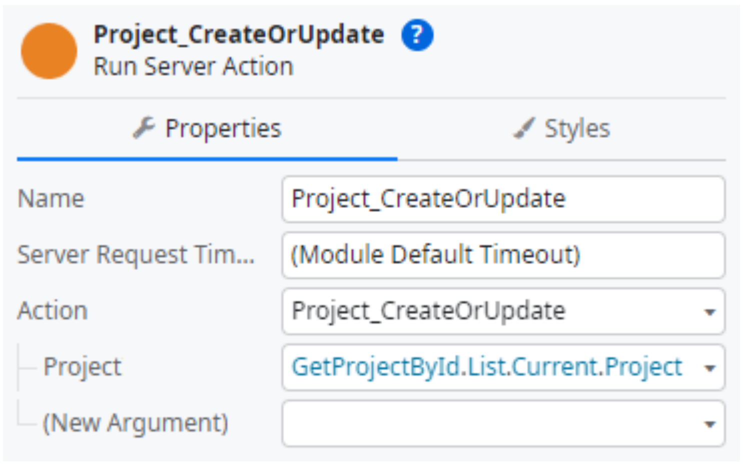
4) Back in the *ProjectDetails* screen, click the *Save* button and in the **On Click** property of the button, select **New Client Action**.



5) Drag the **Run Server Action** to the logic flow and select the *Project_CreateOrUpdate* action.



6) Set the **ProjectId** input value to `GetProjectById.List.Current.Project`

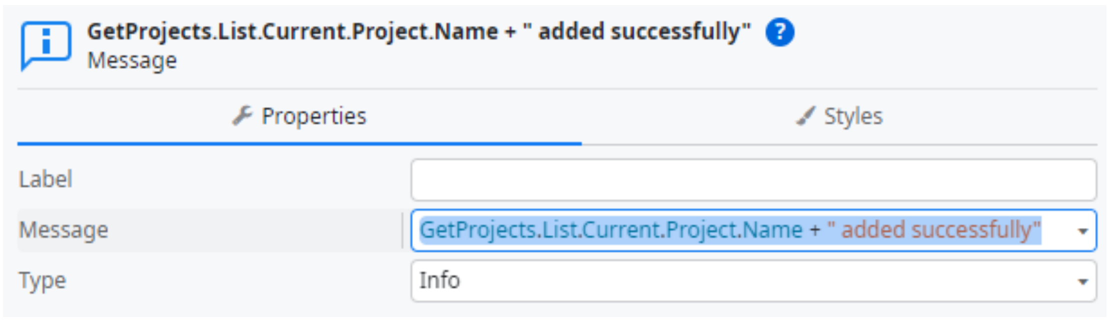


Project_CreateOrUpdate
Run Server Action

Name	Project_CreateOrUpdate
Server Request Tim...	(Module Default Timeout)
Action	Project_CreateOrUpdate
Project	GetProjectById.List.Current.Project
(New Argument)	

7) Drag a **Message** below the server action and set the message to:

```
GetProjectById.List.Current.Project.Name + " added successfully"
```



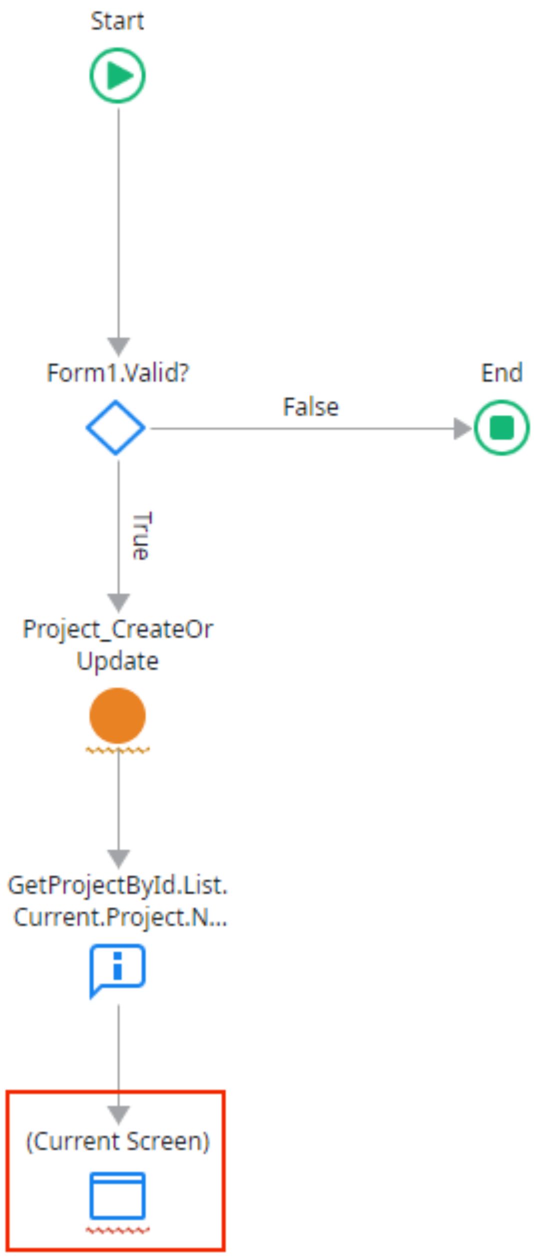
GetProjects.List.Current.Project.Name + " added successfully"
Message

Label	
Message	GetProjects.List.Current.Project.Name + " added successfully"
Type	Info

8) Drag a **Destination** and drop it on top of the **End**.

9) Select the **ProjectDetails** Screen as the destination.

The logic flow of the **SaveOnClick** action to resemble the provided screenshot.

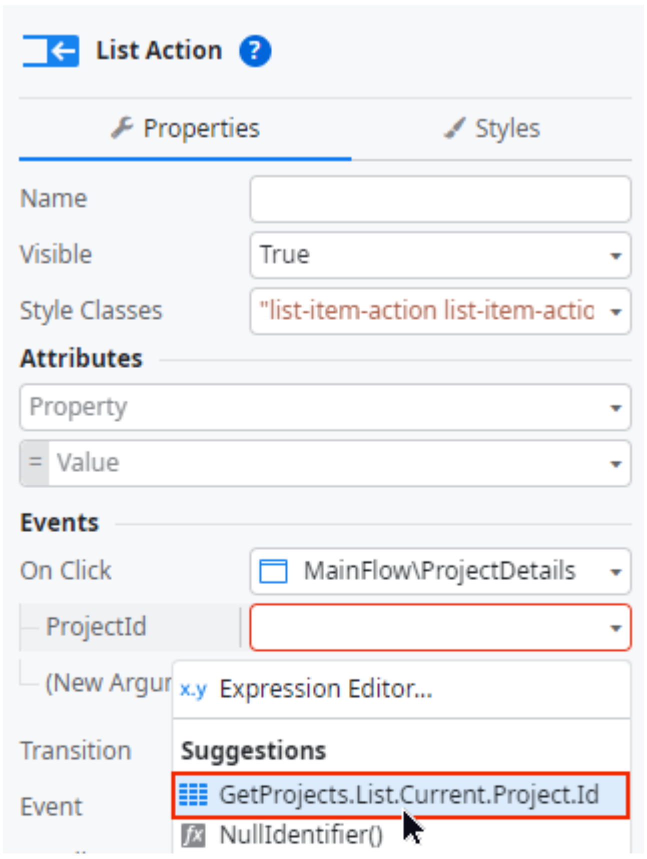


10) Set the value of the input parameter **ProjectId** for the Screen to `NullIdentifier()`

    Notice that there is an error in the *ProjectDashboard* screen. Since the *Add Project* and the *Update* buttons in the ProjectDashboard screen are linked to the ProjectDetails screen, you need to set the value of the ProjectId parameter on these buttons to resolve the error.

11) Navigate to the ProjectDashboard Screen, and for the **Add Project** button, set the value of the **ProjectId** input parameter to `NullIdentifier()`

12) Click the **Update** button in the List, and set the value of the input parameter of the **On Click** to `GetProjects.List.Current.Project.Id`



Name:
Visible: True
Style Classes: "list-item-action list-item-actio
On Click: MainFlow\ProjectDetails
ProjectId:
GetProjects.List.Current.Project.Id

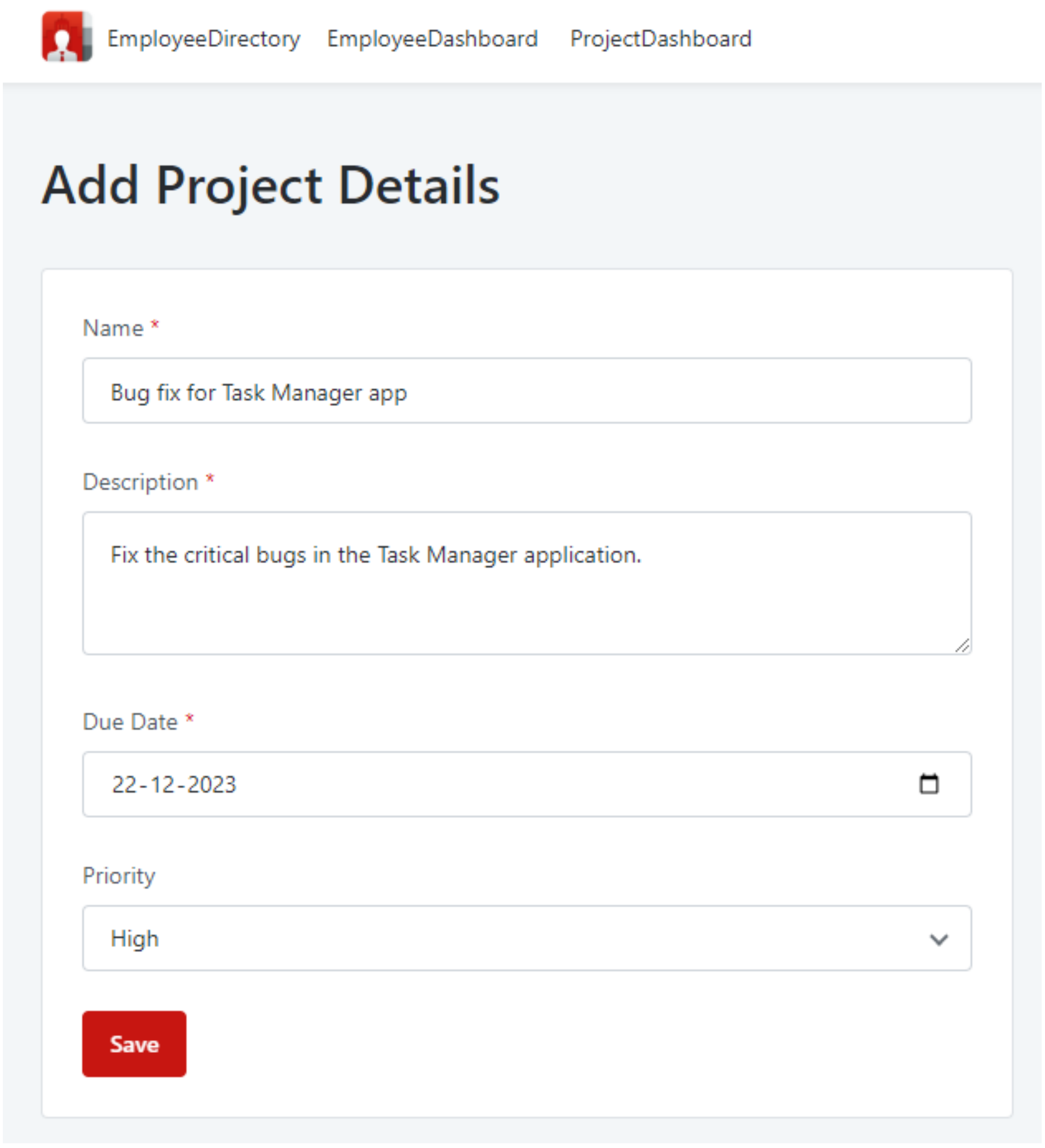
This ensures that the Id of the Project clicked on is passed to the Project Details Screen, helping fetch the project data in the Project Details.

13) Publish the module to save the latest changes.

14) Open the application in the browser.

15) Proceed to the ProjectDetails Screen from the Project Dashboard page.

16) Populate the form with the necessary project details and click **Save**.



Add Project Details

Name *

Bug fix for Task Manager app

Description *

Fix the critical bugs in the Task Manager application.

Due Date *

22-12-2023

Priority

High

Upon successful completion, a confirmation message will indicate that the project has been successfully added to the database, and subsequently, the project form will be reset to its initial blank state.

Congratulations! You've effectively integrated in the server actions in the screen logic to insert or update input data to the database and understood the usage of variables.