Form Validations Exercise

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Outline

In this exercise, we will define two custom validations in an application that manages Employees and Projects. This exercise will be split in two, with one validation being done and tested at a time.

In the application we have a set of projects and employees. The application allows creating new employees and projects, but no custom validation is defined. However, there are some business rules we want to define in this application:

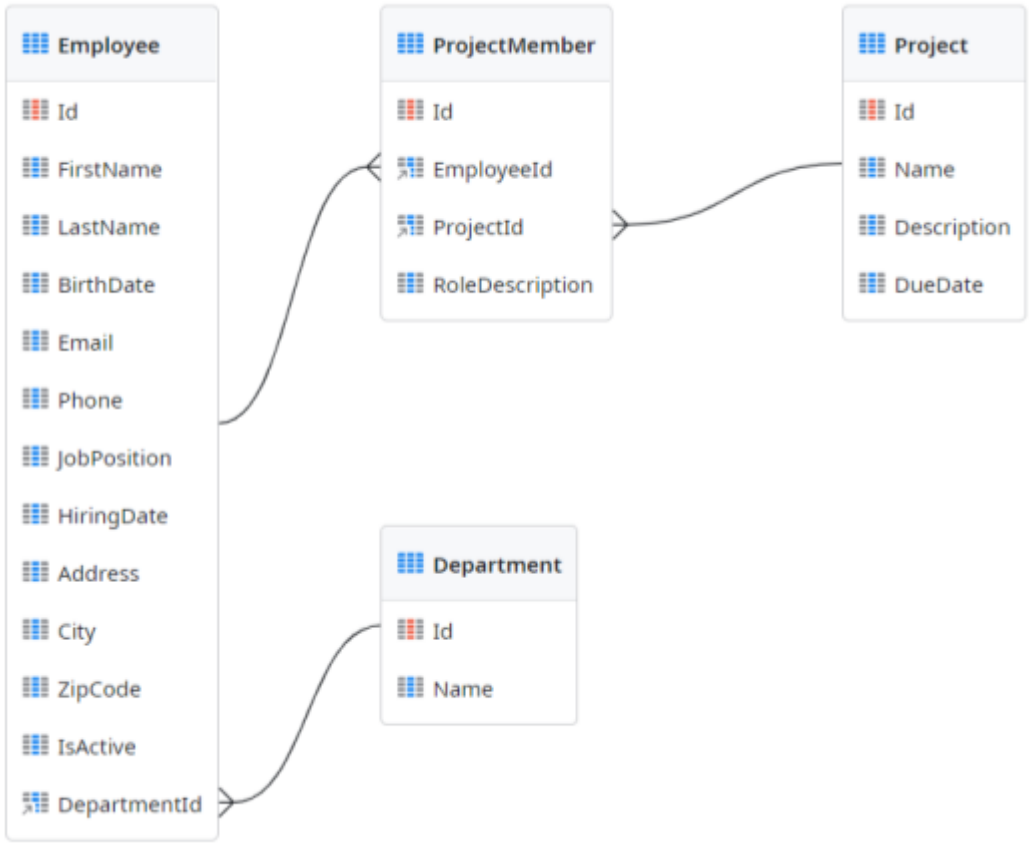
- Every Project must have a Due Date in the future.
- Every Employee must be at least 18 years old when hired.

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 **Form Validations Exercise.oap**.

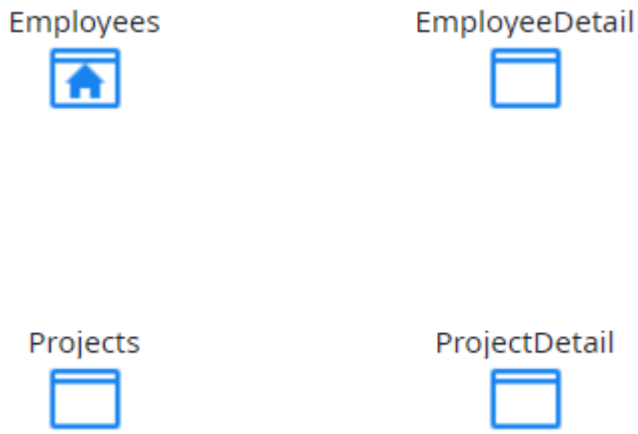
Scenario

In this exercise, we'll start from an existing application with one module. Inside that module, we have a data model already defined with four Entities.



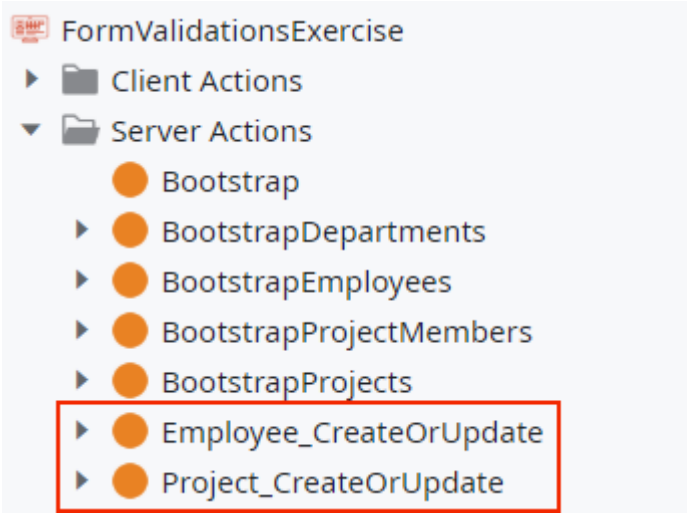
The module has the logic to import data from Excel to populate these Entities. So, when the application (and its module) is installed, the data will automatically be bootstrapped.

The module also has four screens defined.



Regarding the screens, the Employees shows a list of all the Employees, while the EmployeeDetail shows the detailed information about a particular Employee. A similar scenario occurs for Projects, with the Projects and ProjectDetail screens.

There are also two Server Actions with the logic to create or update Employees and Projects: **Employee_CreateOrUpdate** and **Project_CreateOrUpdate**.

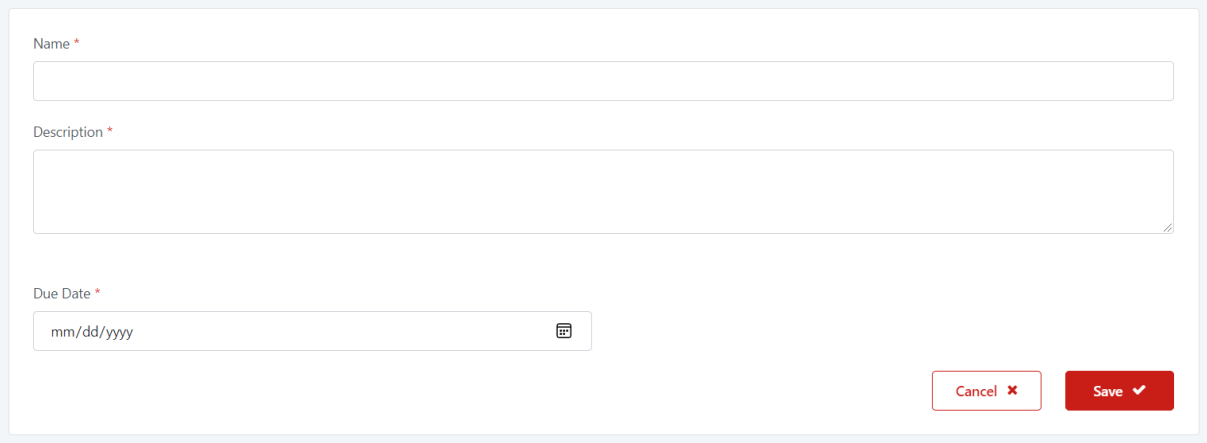


In this exercise, we want to make sure that some rules are followed when adding / editing projects and employees in the application. As mentioned before, these are the custom validations we want to define, and in both cases, we need to make sure that we validate the data before adding it to the database:

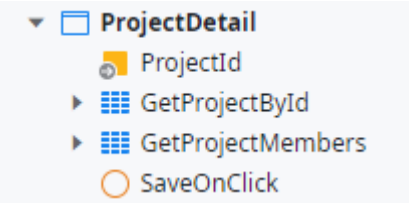
- Every Project must have a Due Date in the future.
- Every Employee must be at least 18 years old when hired.

Projects

The first custom validation will be defined for Projects, thus in the scope of the ProjectDetail Screen. The Screen has a Form already defined with a Save Button that triggers a **SaveOnClick** Client Action. This Action is where the logic for updating / adding a project to the database is defined.



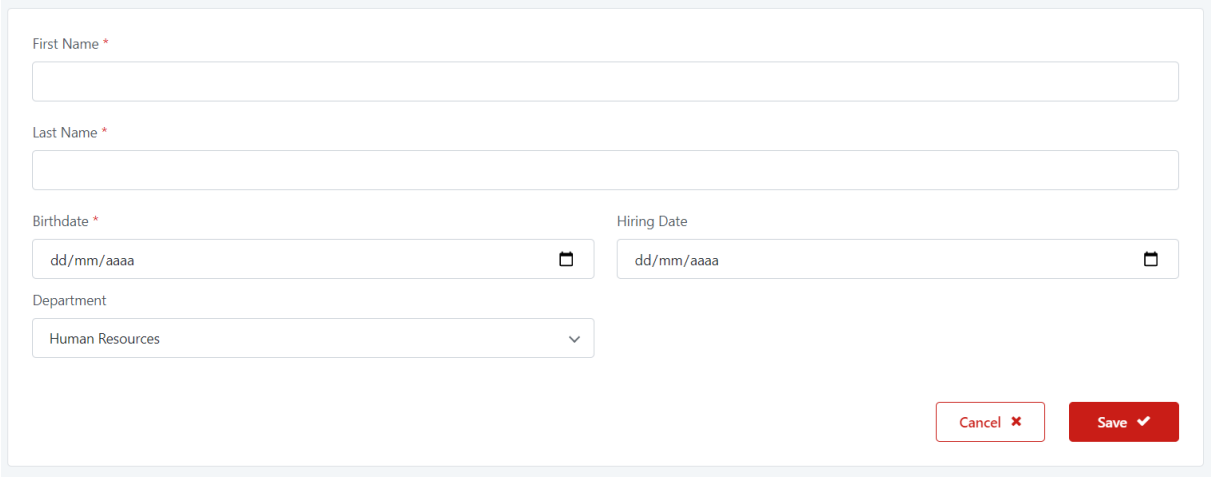
The Project information is fetched from the database through the Aggregate GetProjectById, which will be helpful to implement the validation. Also, the `CurrDate()` Built-in Action can be helpful as well.



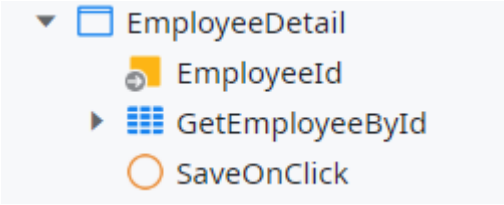
Employees

The second custom validation will be defined in the scope of the EmployeeDetail Screen. The Screen also has a Form already defined with a Save Button that also triggers a Client Action called SaveOnClick (which is different from the previous one).

This Action is where the logic for updating / adding an employee to the database is defined.



The Employee information is fetched from the database through the GetEmployeeById Aggregate. Similarly to what the previous custom validation, we need to manipulate the employee data through this data and implement the validation we want. The `Year()` built-in Action can be helpful in this particular validation.

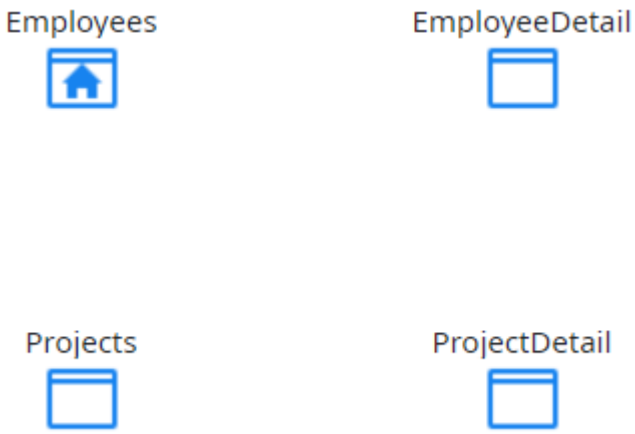


How-To

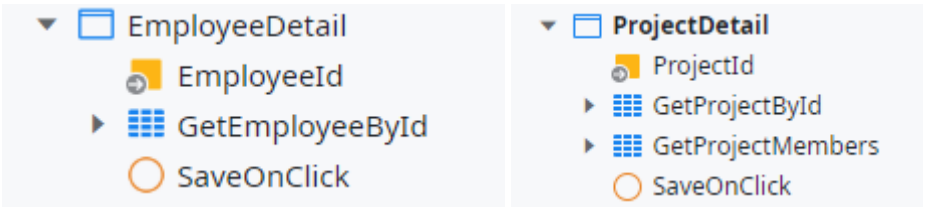
In this section, we show you how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again**. If you didn't finish the exercise, that's fine! We are here to help you.

Getting Started

To start this exercise, we need to install the Quickstart file: **Form Validations Exercise.oap**. This file has an application with four screens, two for the Employees (List and Detail) and two for the Projects (List and Detail).

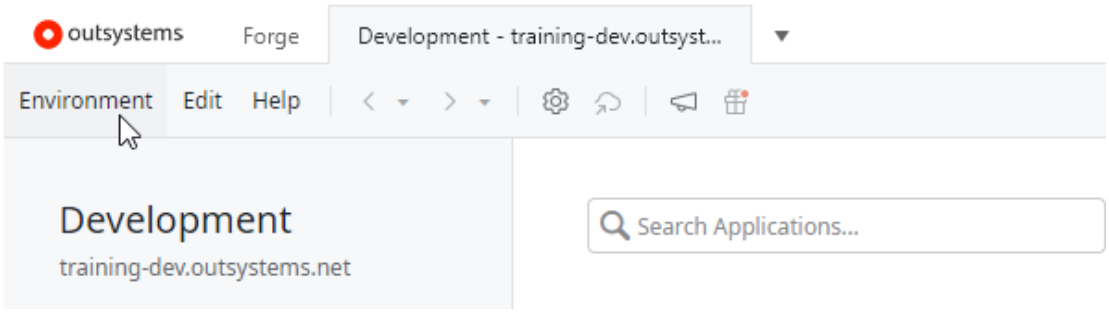


Each of the screens have Aggregates that fetch the necessary information from the database, namely from the Employee and Project Entities, for the application to work properly. For this particular exercise, the Aggregates **GetEmployeeById** and **GetProjectById**, respectively in the **EmployeeDetail** and **ProjectDetail** Screens, are the ones that fetch the information we need to implement the validations.

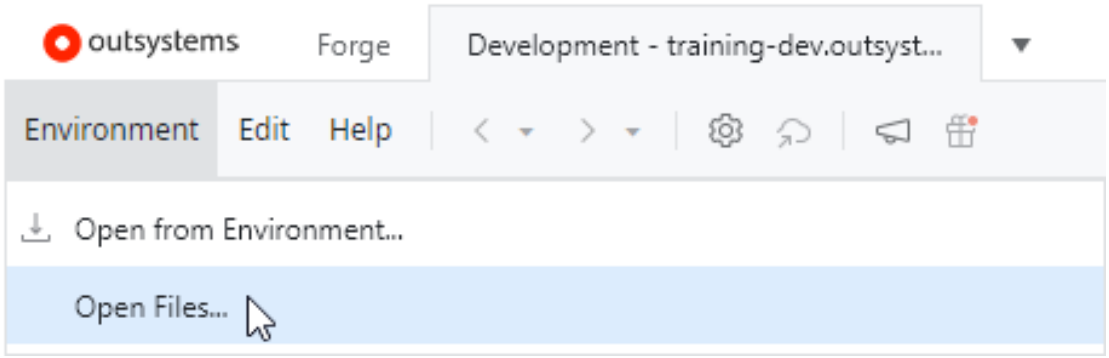


The first step that we need to take is to install the Quickstart application in our development environment. Before proceeding, you must have Service Studio opened and connected to an OutSystems Environment (e.g. Personal Environment).

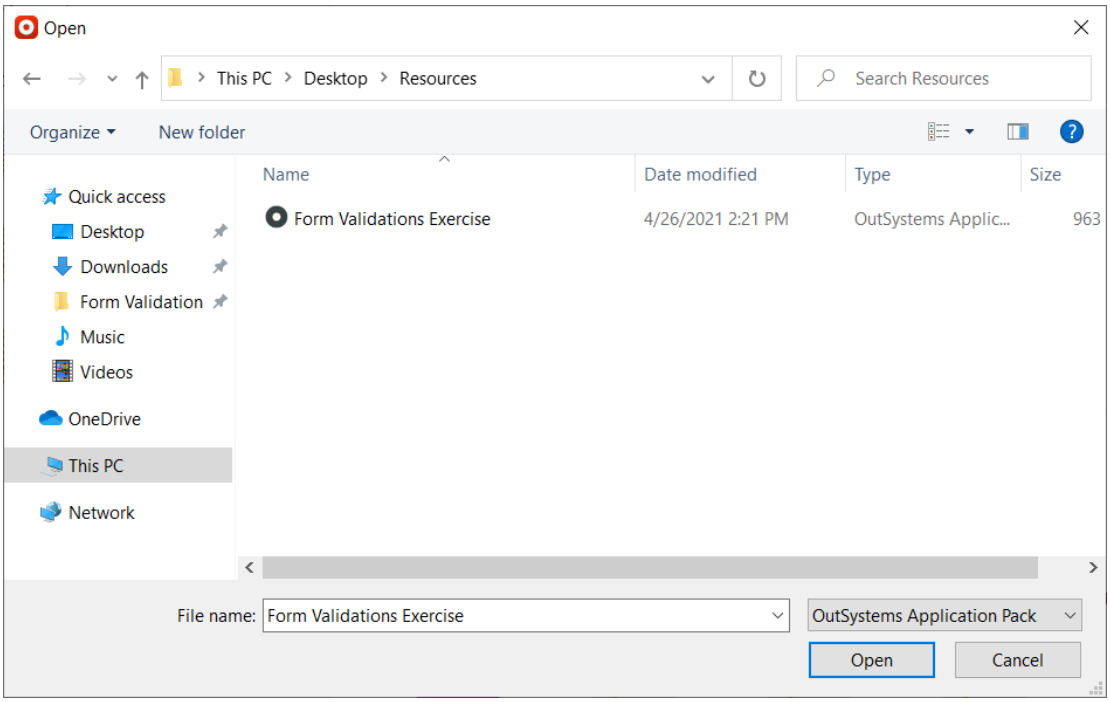
1) In Service Studio's main window, select the **Environment** menu on the top left.



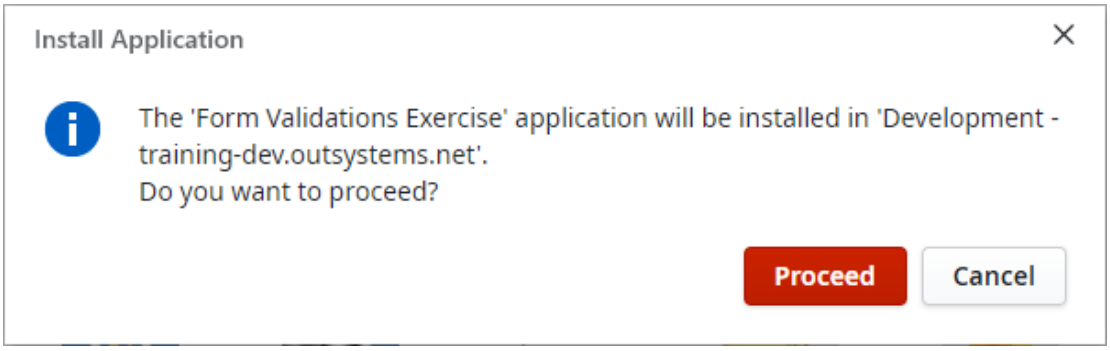
2) Select **Open Files...**



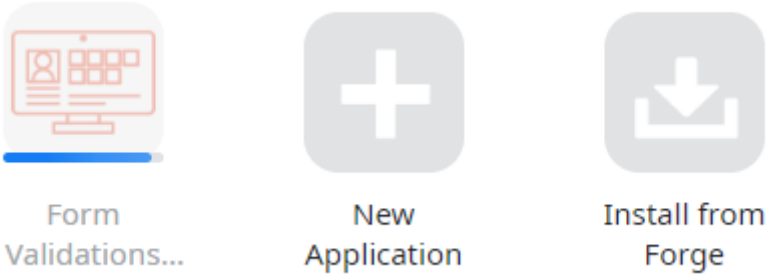
3) In the following dialog, change the file type to OutSystems Application Pack (.oap), find the location of the Quickstart and open the file named **Form Validations Exercise.oap**.



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

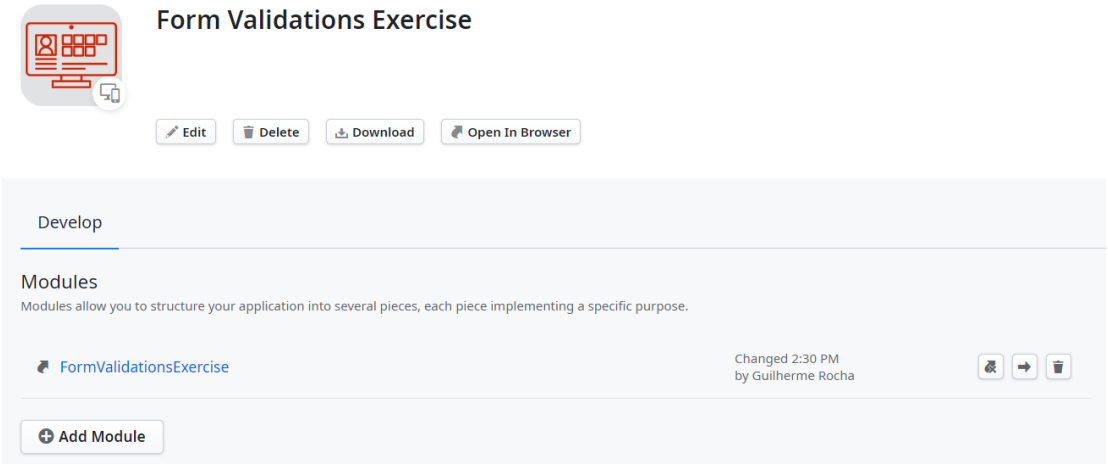


5) The application will begin installing automatically. When it's finished, we're ready to start!



6) Open the application in Service Studio.

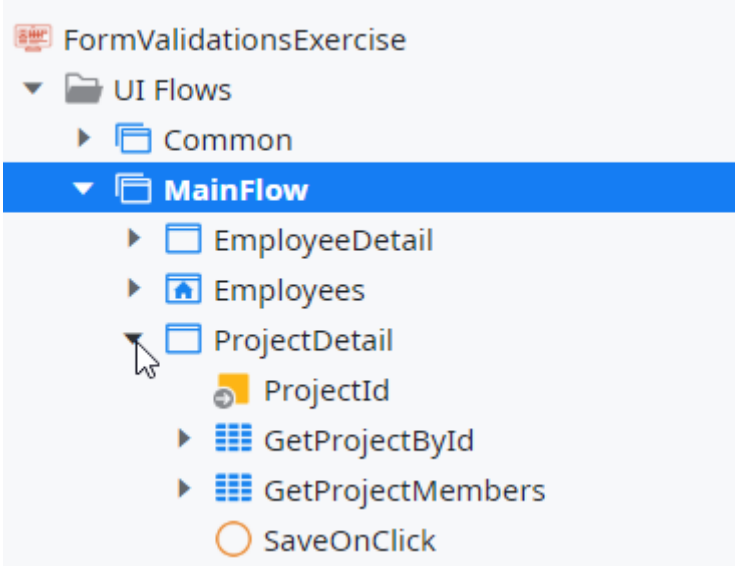
7) The application has only one module. Let's open it!



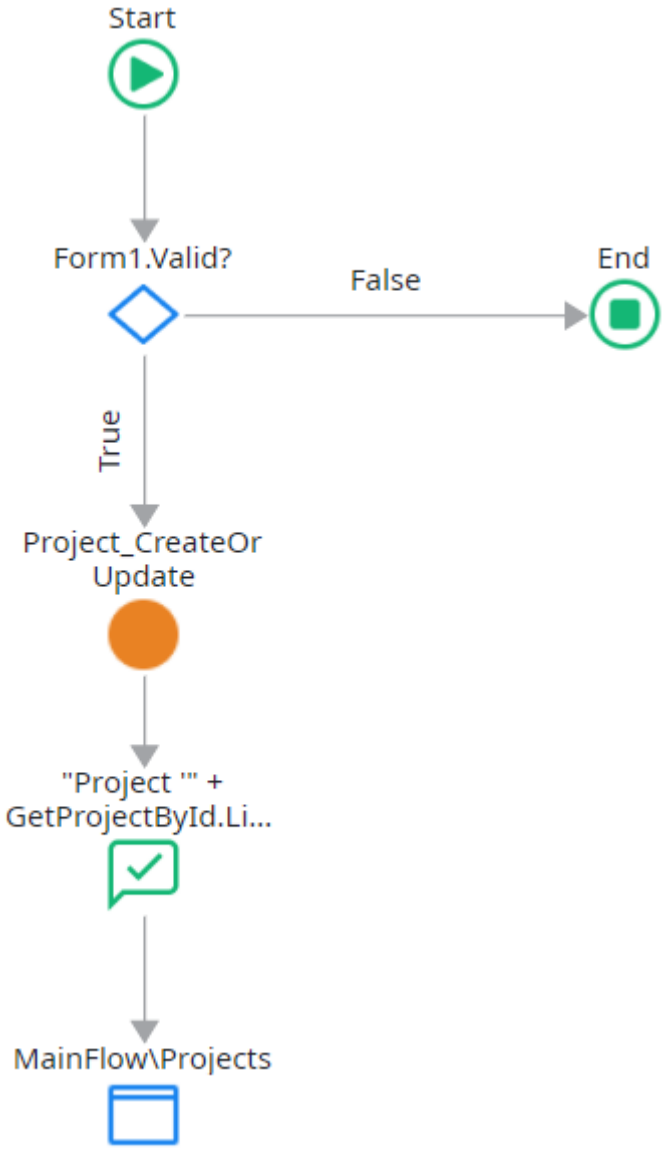
Project Due Date must be in the future

In this section, we'll implement our first validation: the project due date must be in the future. Or, in other words, the project due date must be bigger than the current date.

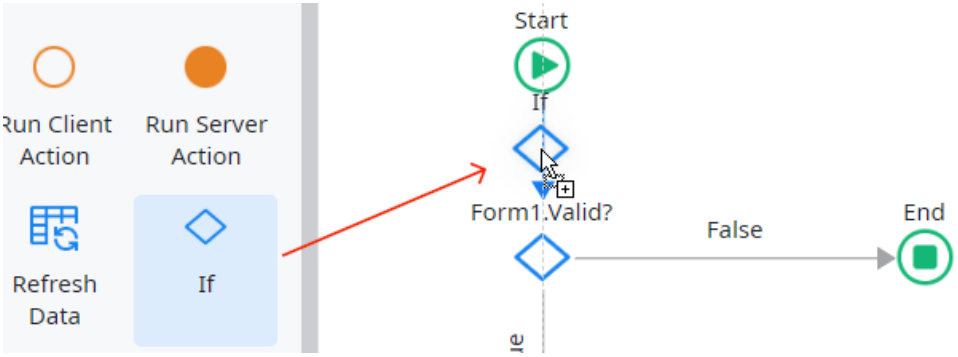
1) Under the **Interface** tab in Service Studio, expand the **ProjectDetail** Screen.



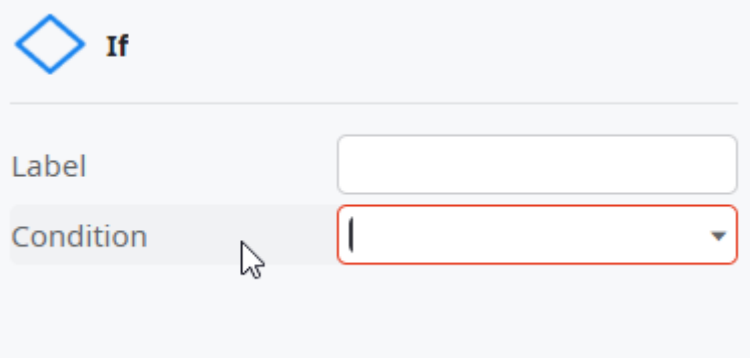
2) Double-click the **SaveOnClick** Action to open it.



3) Drag an **If** node to the Action flow, before the existing If.



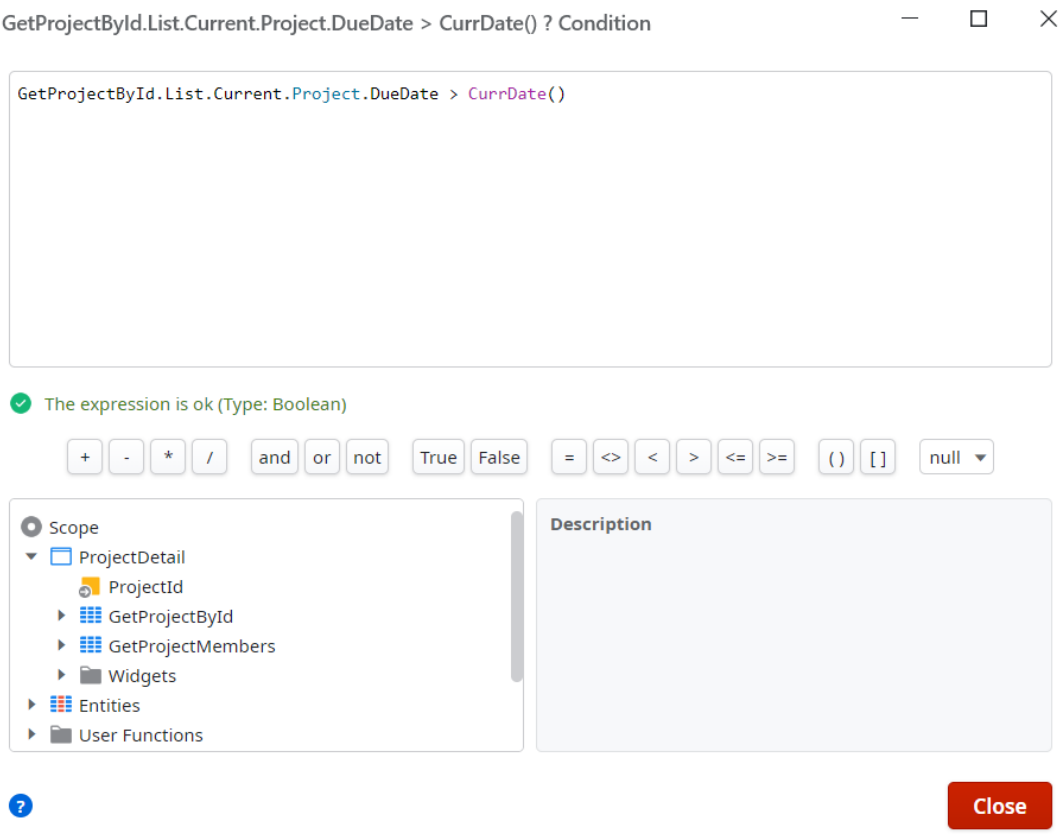
4) Select the recently added If and in the properties area right-click the **Condition** property to open the Expression Editor.



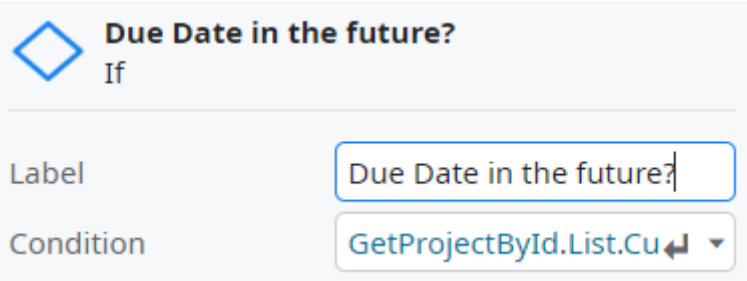
5) Set the Condition to:

```
GetProjectById.List.Current.Project.DueDate > CurrDate()
```

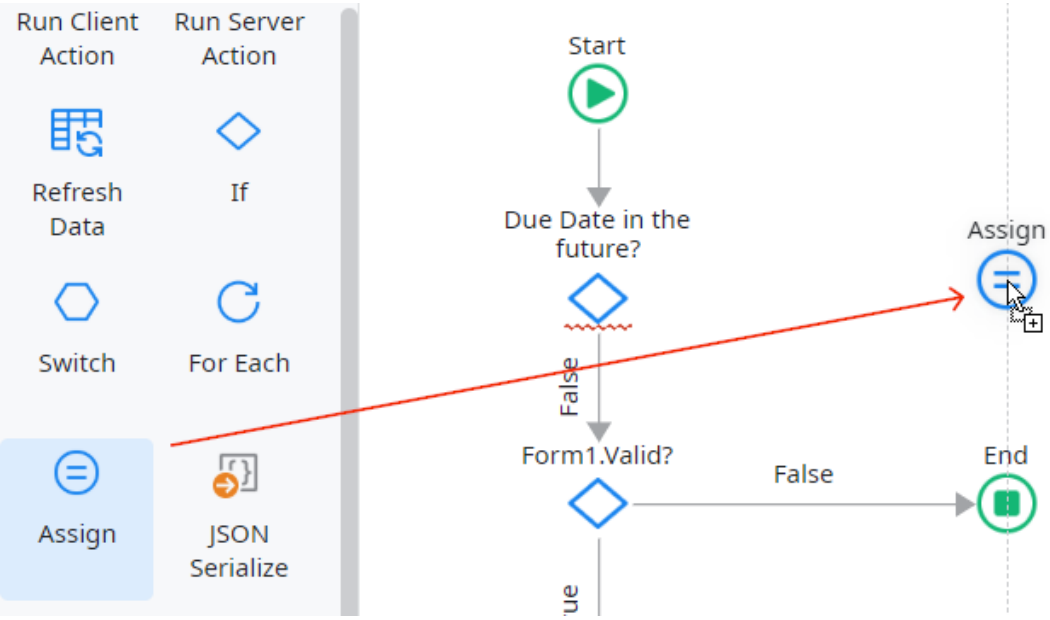
6) This checks whether the Due Date of the Project fetched in the GetProjectById Aggregate, and being edited on the Screen, is bigger than the current date. Click on **Done** to close the editor.



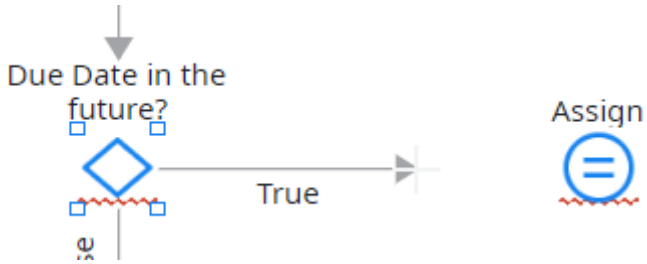
7) Change the **Label** of the If to be *Due Date in the future?*



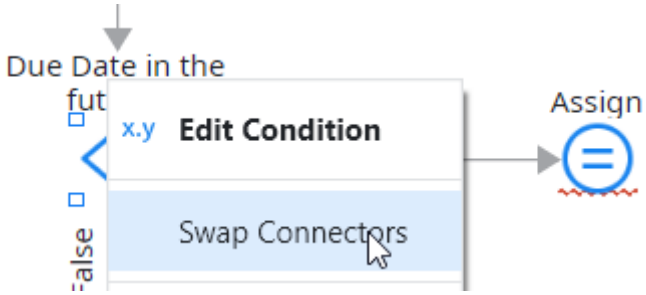
8) Drag an **Assign** node and drop it to the right of the Due Date If



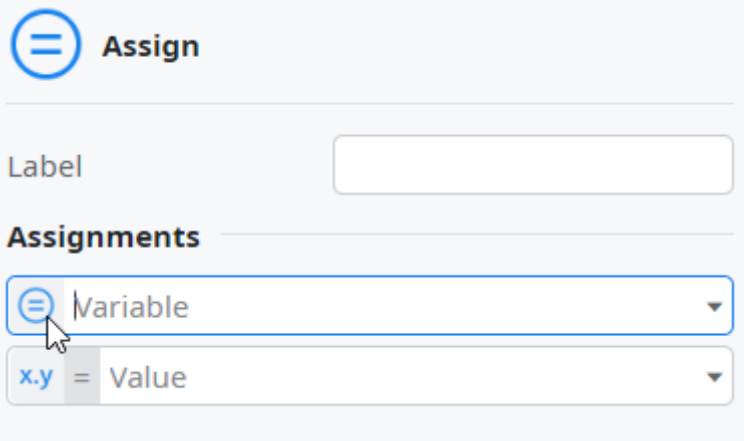
9) Connect the If to the Assign to create the **True** branch of the If.



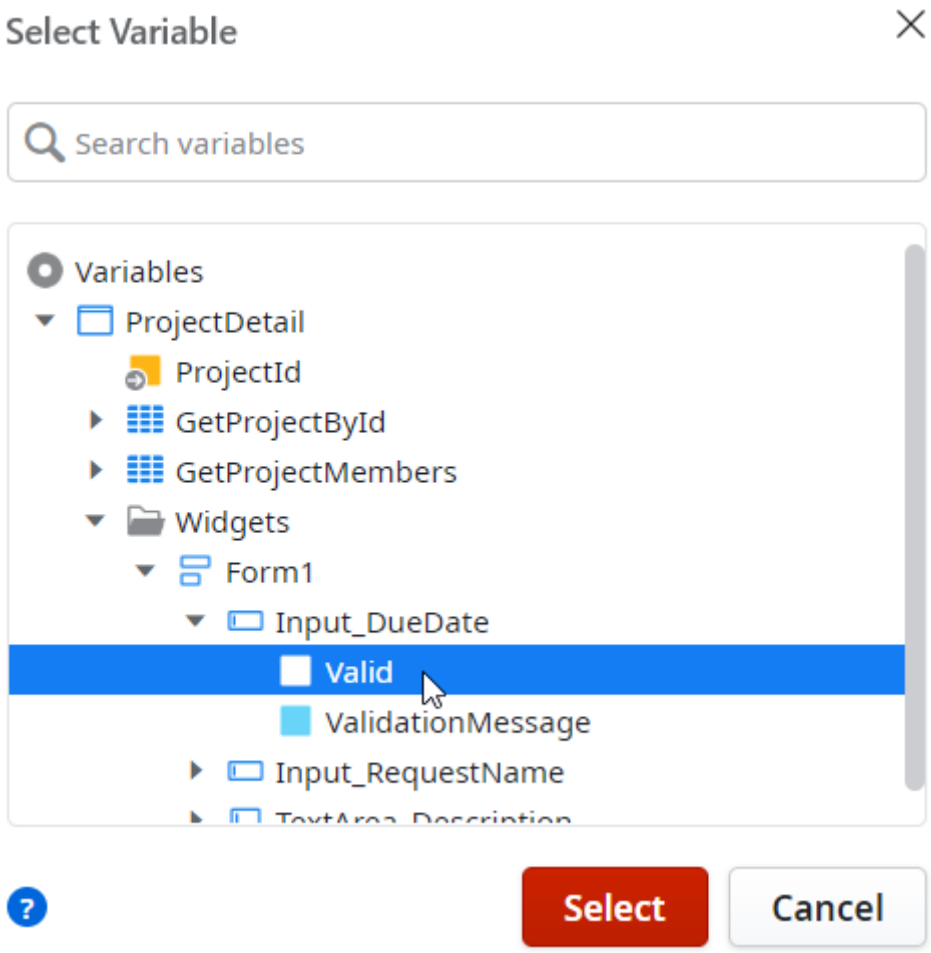
10) Since this should be the **False** branch, instead of the True branch, right-click the If and choose **Swap Connectors**.



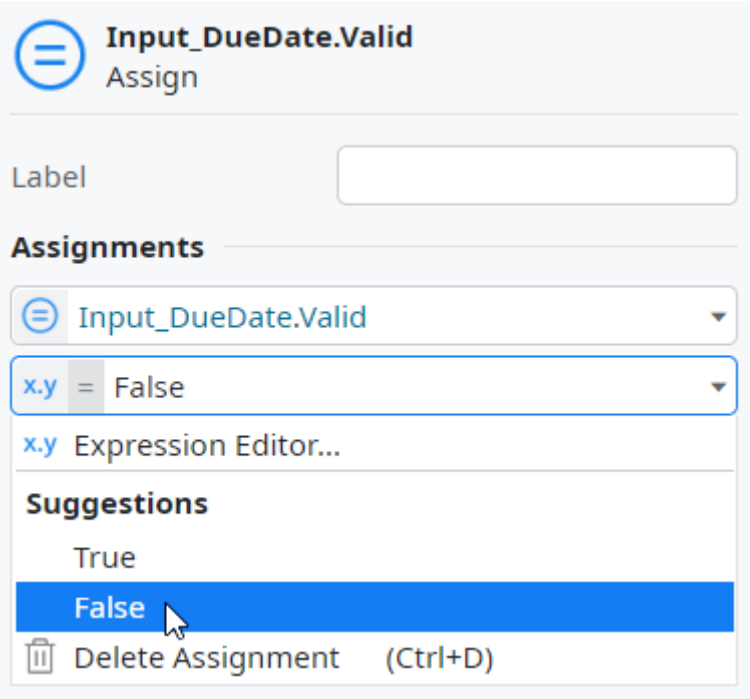
11) Select the Assign and in its properties hover the mouse on the **Variable** field and click the small assign icon that appears.



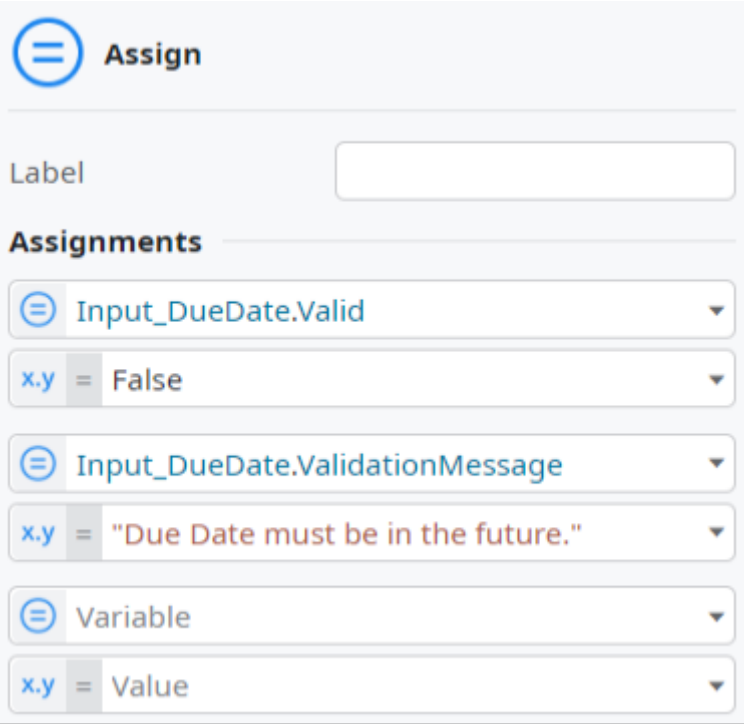
12) In the following dialog, select the **DueDate** input field **Valid** property and click **OK**.



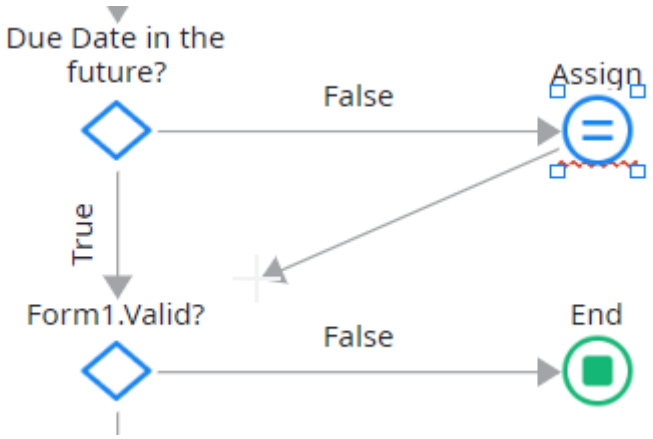
13) In the Assign, set the **Value** to *False*.



14) Define a second assignment setting the **ValidationMessage** to *"Due Date must be in the future"*



15) Connect the **Assign** with the **Form1.Valid?** If to finish the flow.



16) Publish the module to the server to save the latest changes.

Testing the application: Project Due Date in the Future

In this section, we'll see the previous validation at work, by using the application in the browser.

1) Open the application in the browser.

2) Navigate to the Projects Screen and click on the Add Project Button.

3) Fill up the Form with the project information. Make sure that all the mandatory fields are filled and that the Due Date of the Project is in the past.

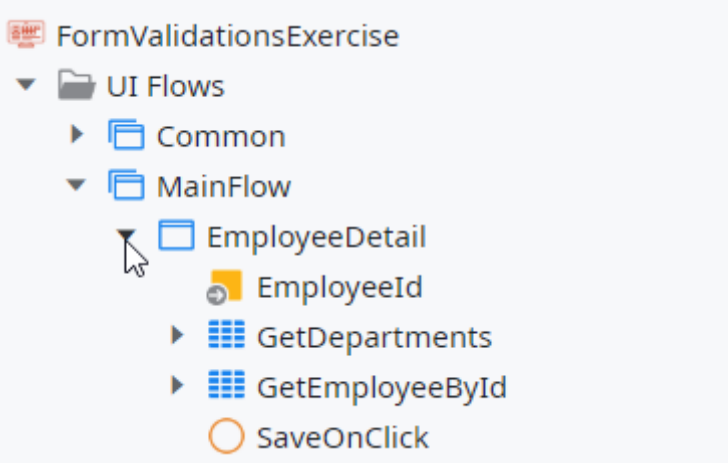
4) Click on **Save** and make sure the validation message appears next to the Due Date input field.

5) Fix the Due Date to be in the future and click Save. Did it work now? Great!

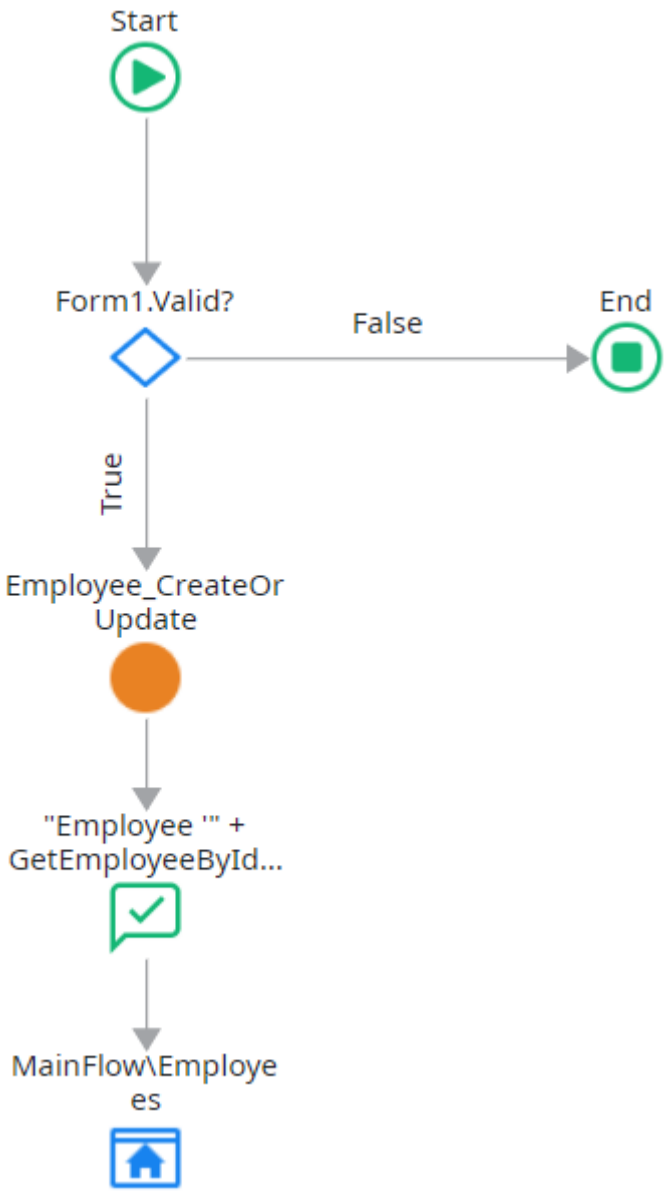
Every Employee Must be Older than 18

In this section, we'll define a second validation, this time for the employees, that guarantees that every employee must be older than 18. This means that the difference between the Hiring Date and the Birth Date, in years, must be bigger than 18.

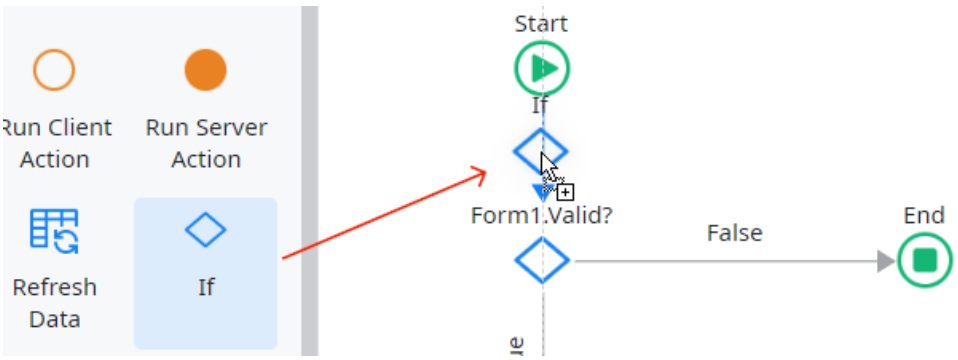
1) Go back to Service Studio and, under the Interface tab, expand the **EmployeeDetail** Screen.



2) Open the **SaveOnClick** Action.



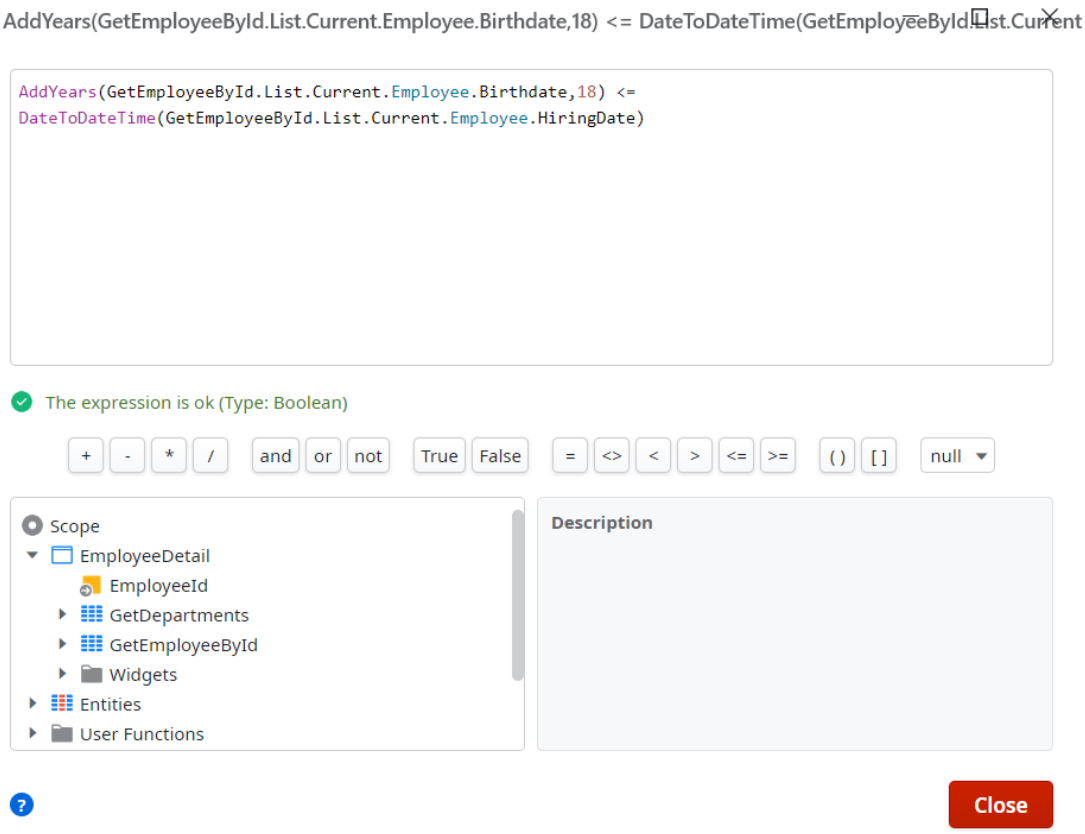
3) Drag an **If** node to the Action flow, before the existing If.



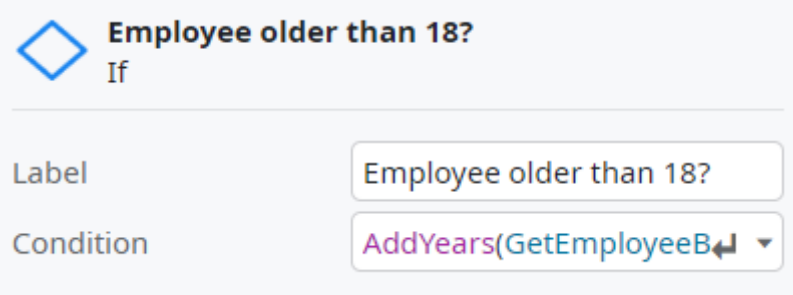
4) Set the If Condition to:

```
AddYears(GetEmployeeById.List.Current.Employee.Birthdate,18) <=
DateToDateTime(GetEmployeeById.List.Current.Employee.HiringDate)
```

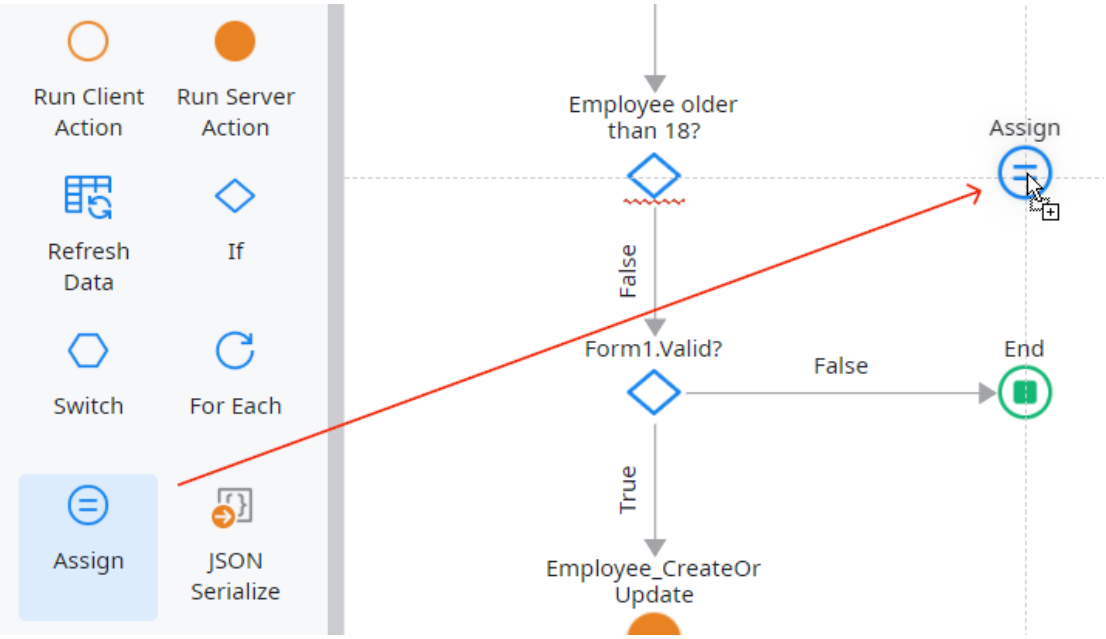
5) This checks if the Employee is at least 18 years old. For that, we use the Hiring Date and the Birth Date, and the Year built-in function which returns the year of a date. Click **Done** to close the Expression Editor.



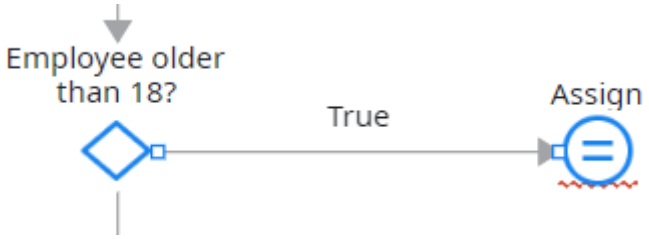
6) Change the **Label** of the If to be *Employee older than 18?*



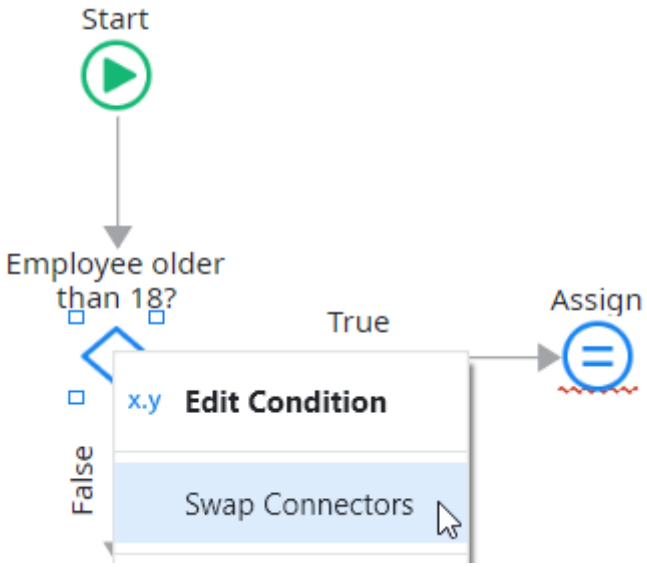
7) Drag an **Assign** node and drop it to the right of the most recent If



8) Connect the If to the Assign to create the True branch of the If.



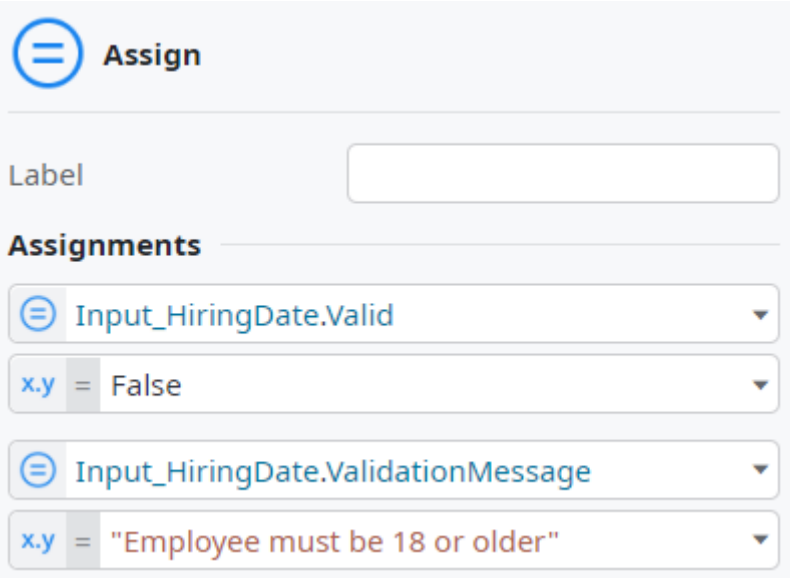
9) Since this should be the **False** branch, instead of the True branch, right-click the If and choose **Swap Connectors**.



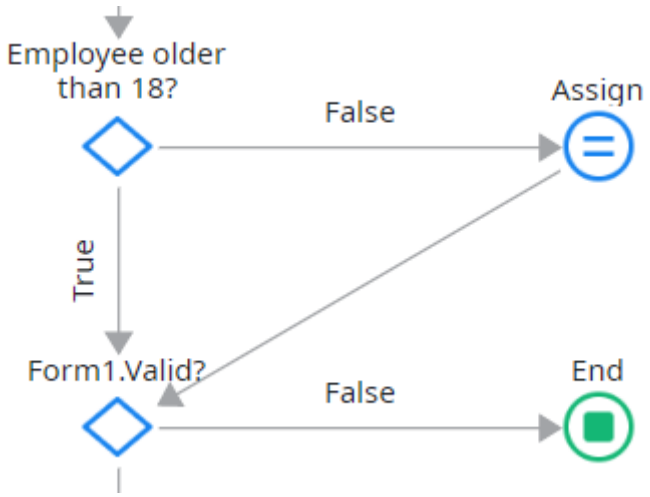
10) Select the Assign node and define the following assignments:

```
Input_HiringDate.Valid
= False

Input_HiringDate.ValidationMessage
= "Employee must be 18 or older"
```



11) Connect the Assign with the *Form1.Valid?* If to finish the flow.



12) Publish the module to the server to save the latest changes.

Testing the application: Employee must be older than 18

In this section, we'll see the previous validation at work, by using the application in the browser.

1) Open the application in the browser.

2) Navigate to the Employees Screen and click the **Add Employee** Button to open the EmployeeDetail screen with an empty Form.

3) Fill up the Form with the employee information. Make sure that all the mandatory fields are filled and that the hiring date is 17 years after the birth date.

4) Click on **Save** and make sure the validation message appears next to the Hiring Date input field.

5) Fix the Hiring Date accordingly and click on Save to make sure it works.