DEVELOPING OUTSYSTEMS MOBILE APPS

Input Validation

Introduction

Over the course of this set of exercise labs, you will create a mobile application. The application will focus on creating and managing To Dos. The To Dos will be persisted in a database so they can be accessed from and shared across multiple devices. To Dos will have attributes such as category, priority (low, medium or high), due date and they can be marked as important (starred) by the user.

Users of the To Do application will be able to access all of this information regardless of whether the device is online or offline. When offline, users will still be able to keep interacting with the application and changes will be saved locally in the device local storage. When the device returns to online mode, changes made while offline will automatically be synced to the server.

You constantly will be expanding your application, publishing it to the server and testing it in your mobile device. Throughout the process you will be learning and applying new OutSystems concepts.

At the end of this set of exercise labs, you will have a small, but well-formed application, spanning multiple screens and concepts that you can easily access from your mobile device.

In this specific exercise lab, you will:

- Create custom Form validations
- Verify the ownership of data

Table of Contents

Introduction 2
Table of Contents 3
Part 1: Form Validations 4
Part 2: Validate To Do Ownership on Save 7
End of Lab 11
List of Figures 12

Part 1: Form Validations

In this part of the exercise, you will ensure that all To Dos can only be saved with a Due Date that is in the future.

1. Ensure that when a To Do is saved the Due Date is in the future.

a) Switch to the **Interface** tab, then open the **SaveOnClick** Client Action of the **ToDoDetail** Screen.

b) Drag an **If** statement and drop it between the Start and 'Form1.Valid?' statement.

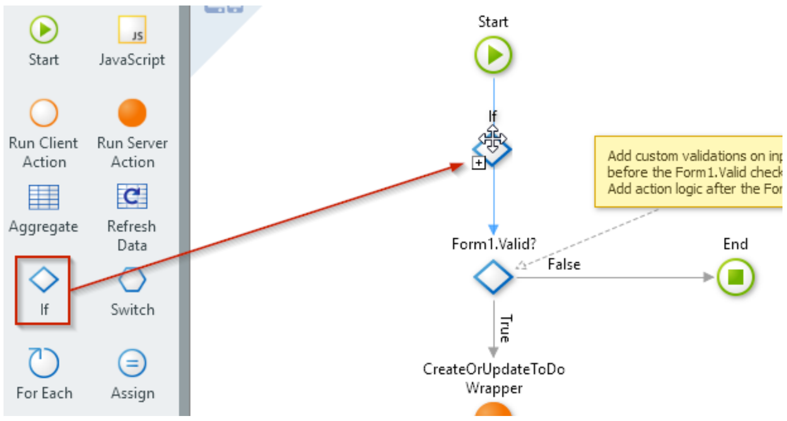


Figure 1. Drag and drop an If statement

c) Set the **Condition** property of the created **If** statement to

```
GetToDoById.List.Current.ToDo.DueDate <> NullDate() and
GetToDoById.List.Current.ToDo.DueDate < CurrDate()
```

d) Set the **Label** property of the **If** to 'Due Date in the past?', to better describe its condition.

e) Drag an **Assign** statement and drop it to the right of the If statement.

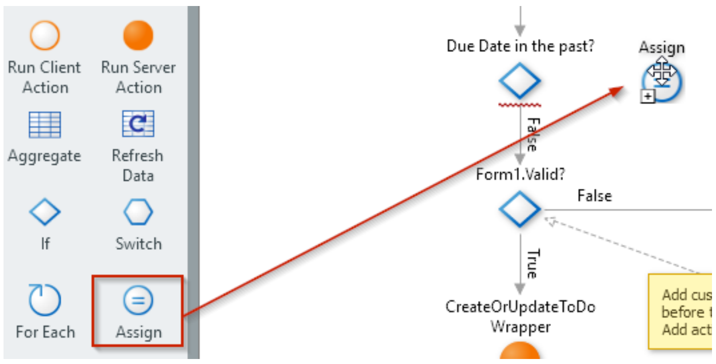


Figure 2. Drag an Assign statement

f) Click and drag a link from the **If** to create the **True** branch connector to the Assign statement.

g) In the Assign statement, define the following assignments

```
Input_DueDate.Valid = False

Input_DueDate.ValidationMessage = "Due Date cannot be in the
past."
```

h) Click and drag a link from the **Assign** to create the connector to the 'Form1.Valid?' **If** statement.

i) The **SaveOnClick** Client Action should look like this.

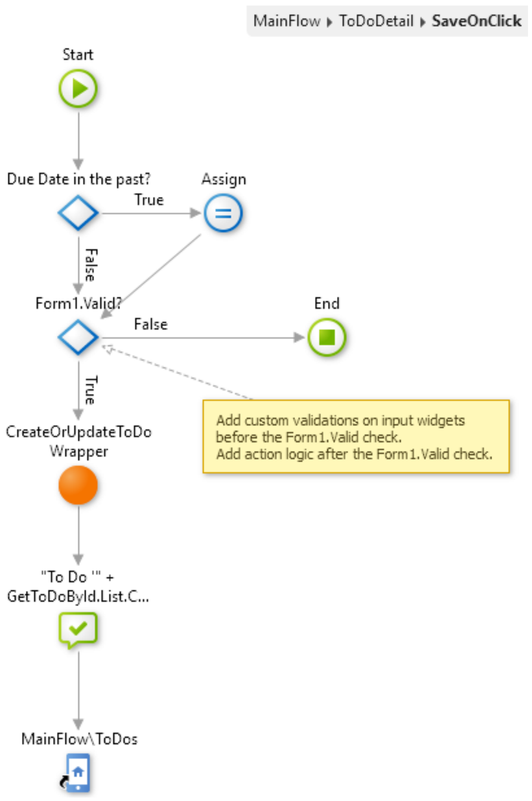

MainFlow ▸ ToDoDetail ▸ SaveOnClick
Start
Due Date in the past?
True
Assign
False
Form1.Valid?
False
End
True
Add custom validations on input widgets before the Form1.Valid check.
Add action logic after the Form1.Valid check.
CreateOrUpdateToDo Wrapper
"To Do '" + GetToDoById.List.C...
MainFlow\ToDos

Figure 3. SaveOnClick Client Action

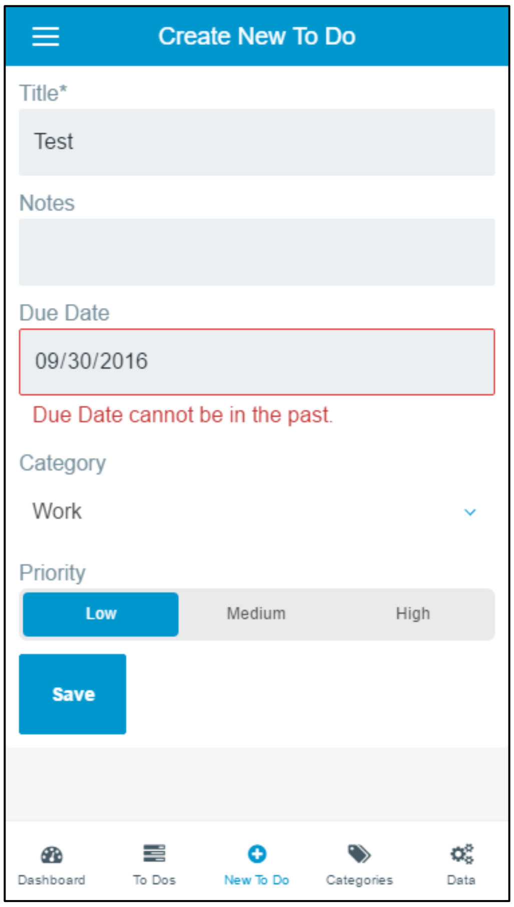
2. Publish the module and test that it is not possible to create To Dos with the Due Date in the past.

a) Click the **1-Click Publish** button to publish the module to the server.

b) Verify in the 1-Click Publish tab that the publishing process was successful.

c) Click the **Open in Browser** button to open the application and navigate to the 'Create New To Do' Screen.

d) Fill in the form setting the Due Date to the past and then click **Save**.


Create New To Do

Title*
Test

Notes

Due Date
09/30/2016
Due Date cannot be in the past.

Category
Work

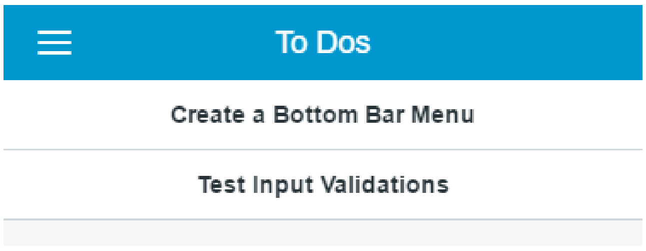
Priority
Low | Medium | High

Save

Figure 4. Due Date Validation

e) Change the Due Date to today, and click **Save** again.

f) The new To Do should now appear in the **ToDos** Screen.


To Dos

Create a Bottom Bar Menu

Test Input Validations

Figure 5. Example of a ToDos Screen

Part 2: Validate To Do Ownership on Save

In this part of the exercise, you will ensure that the users of the application can only edit the To Dos that were created by them.

1. Ensure that users can only edit their own To Dos.

a) Switch to the **Logic** tab and open the **CreateOrUpdateToDoWrapper** Server Action.

b) Drag an **If** statement and drop it between the Start and the first Assign.

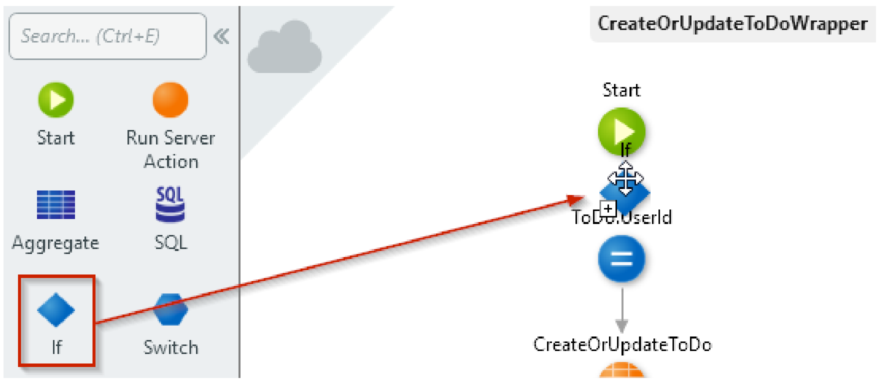


Figure 6. Drag and drop an If statement

c) Set the **Condition** of the If statement to

```
ToDo.Id <> NullIdentifier()
```

d) Drag an **Aggregate** statement and drop it on the right of the If statement.

e) Create the **True** branch of the If statement to connect it to the 'Aggregate1' statement.

f) Double-click on **Aggregate1** to open it.

g) From the **Data** tab, drag the **ToDo** Entity into the Aggregate editor.

h) Rename the **ToDo** Input Parameter of the **CreateOrUpdateToDoWrapper** to 'ToDoInput'. Notice that the existing logic was automatically refactor to reflect that change.

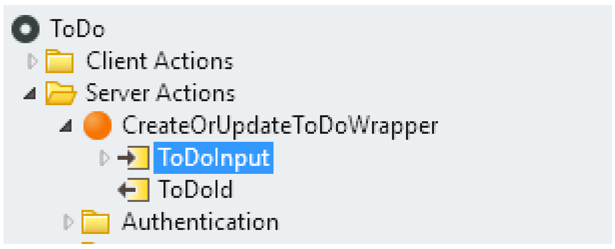


Figure 7. Renamed ToDo Input Parameter to ToDoInput

NOTE: This step is important in the context of this particular Aggregate. The Aggregate contains a Source Entity named **ToDo**, therefore it will hide any variable, Input or Output Parameter in the current scope that also has the name **ToDo**. If the Input Parameter is not renamed then, while creating the Aggregate filter, it would not be available in the current scope.

i) In the **Filters** tab of the Aggregate editor, create a new Filter to retrieve only the To Do that matches the one being updated, by setting the Aggregate Filter expression to:

```
ToDo.Id = ToDoInput.Id
```

j) The Aggregate should look like this

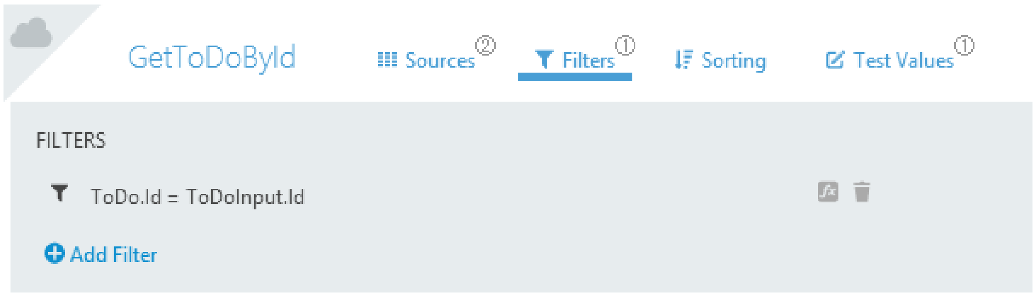


Figure 8. GetToDoById Aggregate.

k) Return to the **CreaterOrUpdateToDoWrapper** flow.

l) Drag another **If** statement and drop it on the right side of the **GetToDoById** Aggregate.

m) Create the connector between the **GetToDoById** Aggregate and the new If statement.

n) Set the **Condition** property of the new **If** to

```
GetToDoById.List.Current.ToDo.UserId <> GetUserId()
```

o) Drag a **Raise Exception** statement and drop it on the right of the **If** statement.

p) In the **Select Exception** dialog select the 'New User Exception' button.

q) In the **Logic** tab locate the created exception named **UserException1** and rename it to 'InvalidToDoException'.

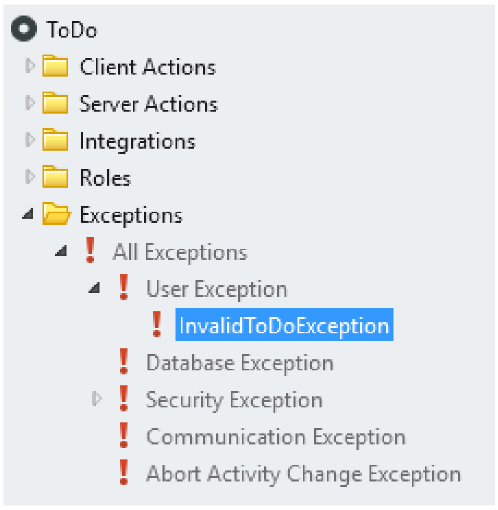


Figure 9. UserException1 in the Logic tab

r) Return the **CreateOrUpdateToDoWrapper** Server Action and select the **Raise Exception** statement created before.

s) Set the **Exception Message** property to "To Do not found.".

t) Create the **True** branch between the last If statement created and the Raise Exception statement.

u) Create the **False** branch of the **If** and connect it to the first Assign statement in the flow.

v) The **CreateOrUpdateToDoWrapper** should look like this

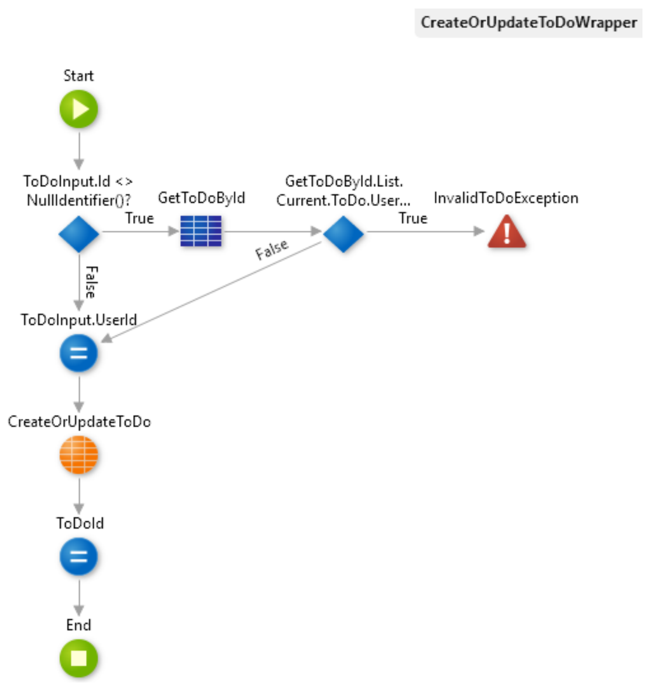


Figure 10. CreateOrUpdateToDoWrapper Server Action

NOTE: These kind of security checks enforce some business rules, and in this case, we can ensure that users will not be able to modify To Dos from other users.

2. Publish the application module.

a) Click the **1-Click Publish** button to publish the module to the server.

b) Verify in the 1-Click Publish tab that the publishing process was successful.

NOTE: Although possible, testing the ownership verification would require, e.g., to tamper with URLs which is out of the scope of this exercise lab.

End of Lab

In this exercise, you implemented custom form validations to ensure that it is impossible to create To Dos with Due Dates in the past.

To improve the security of the application, you implemented the validation of the To Dos ownership on save. The ownership verification ensures that To Dos can only be modified by the owner.

List of Figures

Here is the list of screenshots and pictures used in this exercise.

Figure 1. Drag and drop an If statement 4
Figure 2. Drag an Assign statement...4
Figure 3. SaveOnClick Client Action...5
Figure 4. Due Date Validation 6
Figure 5. Example of a ToDos Screen 6
Figure 6. Drag and drop an If statement 7
Figure 7. Renamed ToDo Input Parameter to ToDoInput.................................8
Figure 8. GetToDoById Aggregate. 8
Figure 9. UserException1 in the Logic tab 9
Figure 10. CreateOrUpdateToDoWrapper Server Action 10