outsystems

DEVELOPING OUTSYSTEMS MOBILE APPS

Blocks

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 a new Block to represent a To Do
- Instantiate Blocks
- Prevent users from modifying completed To Dos

Table of Contents

Part 1: Create a new Block for To Dos

In this part of the exercise, you will create a new Block to display the information of each To Do item.

1. Create a new **ToDoItem** Block to display the To Do item information.

a) Switch to the **Interface** tab, right click the **MainFlow** and select 'Add Block'.

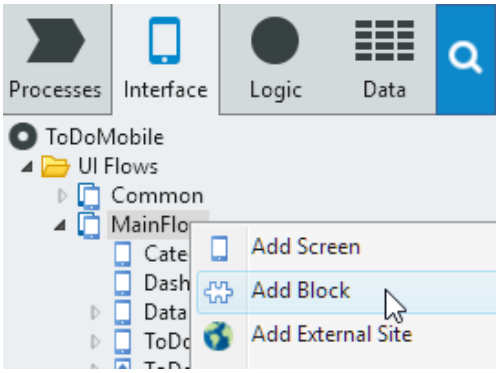

Figure 1. Add Block to MainFlow

b) Change the name of the Block to 'ToDoItem'.

c) Right-click the **ToDoItem** Block and add a 'LocalToDo' Input Parameter. Verify that the **LocalToDo** Input Parameter **Data Type** is 'LocalToDo'.

d) Add a second Input Parameter named 'LocalPriority', and verify that **Data Type** is set to 'LocalPriority'.

> **NOTE:** Since this Block will be reused inside the list Screen, we need to send as Input Parameters the values that will be needed to render each instance of the Block.
>
> It would be possible to only pass the **LocalToDo Identifier**, but in that case each instance would have another Aggregate to fetch the data. This would lead to an increased number of queries and could impact the application's performance.

2. Have the **ToDoItem** Block to display To Do information in a **List Item** and indicate title, due date, importance, completeness and priority.

a) Drag a **List Item** Widget from the toolbar and drop it inside the Block.

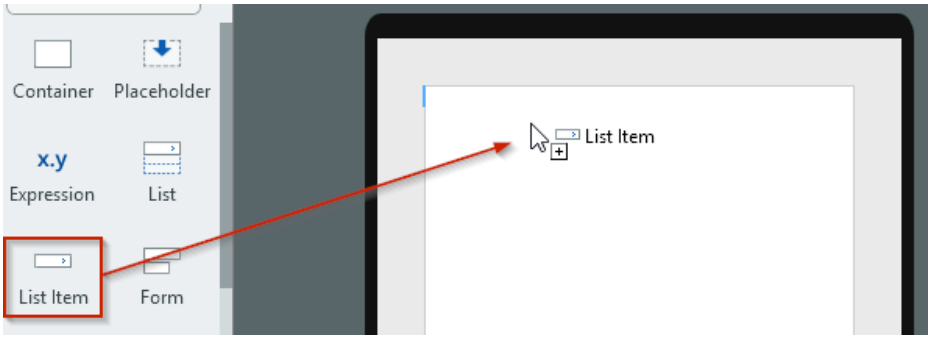

Figure 2. Drag and drop List Item Widget

b) Drag a **List Item Content** Widget and drop it inside the **Content** placeholder of the **List Item** created in the previous step.

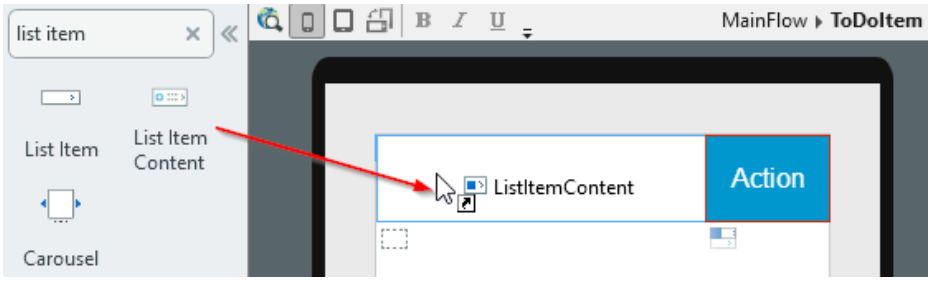

Figure 3. Drag and drop List Item Content Widget

c) Expand the **LocalToDo** Input Parameter of the Block, then drag the **Title** attribute and drop it inside the **Title** placeholder.

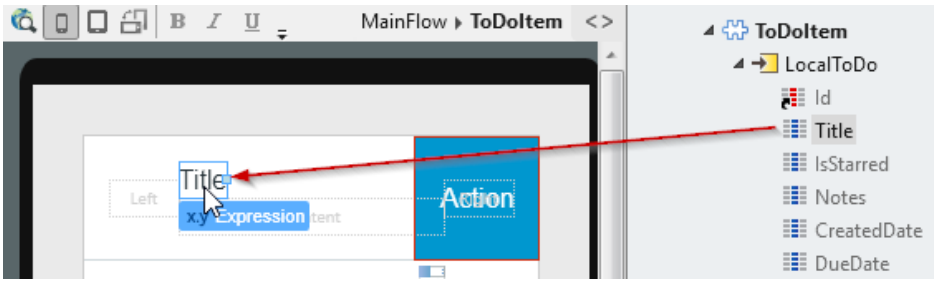

Figure 4. Drag Title attribute to create Title Expression

d) Drag an **Icon** Widget and drop it just before the **Title** Expression.

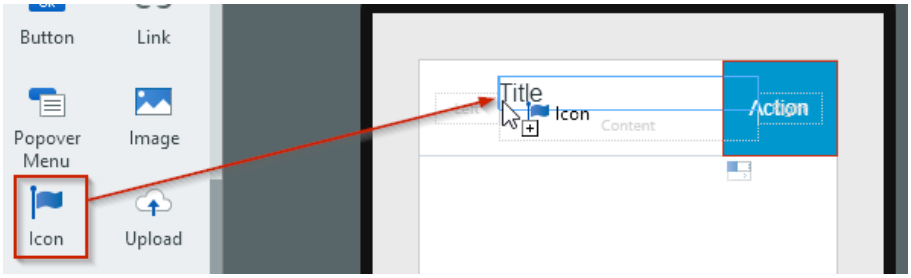

Figure 5. Drag and drop an Icon Widget

e) In the **Pick an Icon** dialog, select the 'check circle' icon.

 Do not duplicate

f) Change the **Size** property to 'Font size' and add the 'text-green' style to the **Style Classes** property.

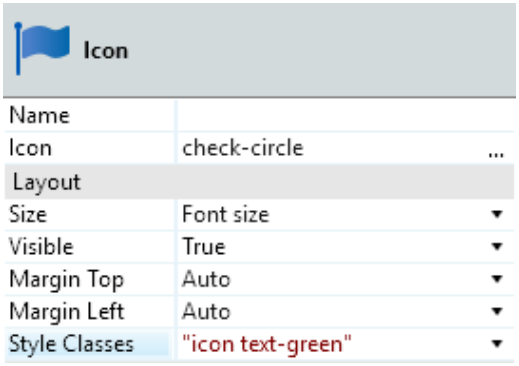

Figure 6. Icon properties

g) Right-click the **Icon** Widget and choose 'Enclose in If'.

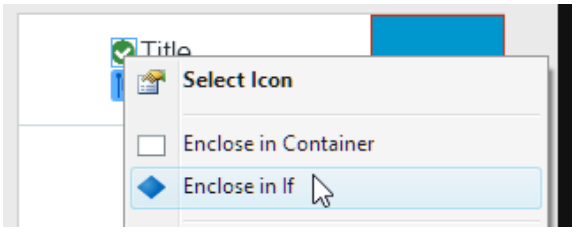

Figure 7. Enclose Icon in If

h) Set the **Condition** property of the If to

```
LocalToDo.CompletedDate <> NullDate()
```

i) Drag the **DueDate** attribute and drop it in the **Content** placeholder to create the 'DueDate' Expression.

j) Right-click the **DueDate** Expression and choose 'Enclose in If'.

k) Set the **Condition** of the If to

```
LocalToDo.DueDate <> NullDate()
```

l) Add the text 'Due ' just before the **Due Date** Expression, but still inside the **If** Widget.

m) Drag an **Icon** Widget and drop it in the **Left** placeholder, then choose the filled 'star' icon.

n) Add the 'gold-color' style to the **Style Classes** property.

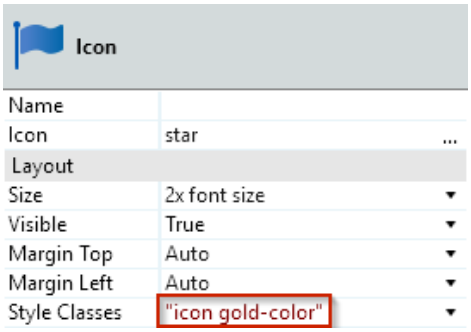

Figure 8. Style Classes with 'gold-color'

o) Right-click the **star** Icon and choose 'Enclose in If'.

p) Set the **Condition** property of the If to

```
LocalToDo.IsStarred
```

q) Drag another **Icon** Widget and drop it in the **False** branch of the previous as, then choose the hollow 'star' icon. Add the 'gold-color' style to the **Style Classes** property of the **Icon**.

r) Select the **List Item** Widget then hide its placeholders by clicking the Collapse circle.

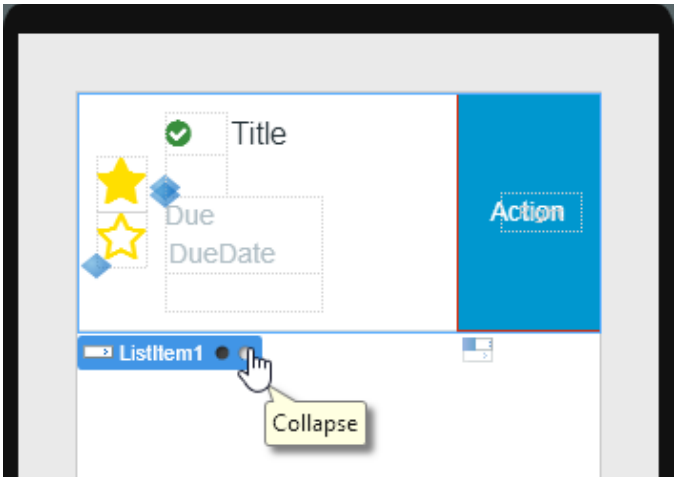

Figure 9. Collapse placeholders

s) From the Widget toolbar, drag and drop the **Tag** Widget and drop it in the **Right** placeholder. If there is a Widget already in the placeholder, don't forget to delete it.

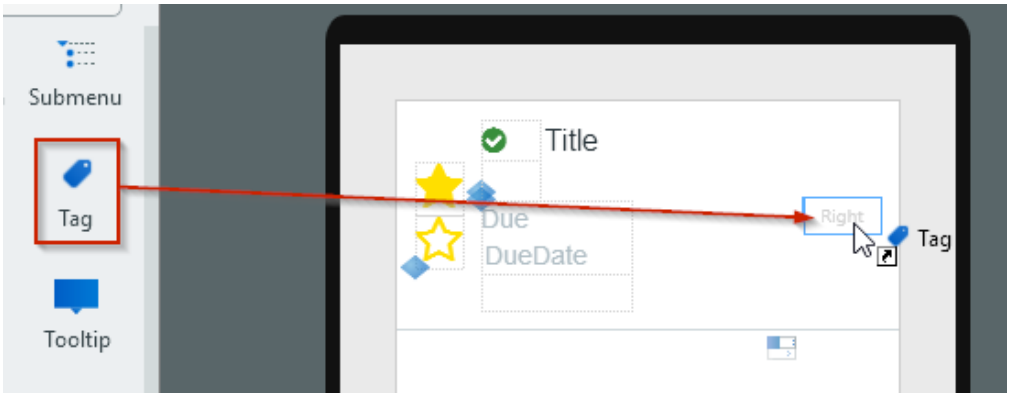

Figure 10. Drag and drop a Tag Widget

t) Double-click the **BackgroundColor** property to open the Expression Editor, then inside build the following expression

```
If(LocalToDo.PriorityId = Entities.Priority.High,
Entities.Color.Red, If(LocalToDo.PriorityId =
Entities.Priority.Medium, Entities.Color.Orange,
Entities.Color.Green))
```

u) Expand the **LocalPriority** Input Parameter, then drag the **Label** attribute and drop it inside the **Text** placeholder of the **Tag** Widget.

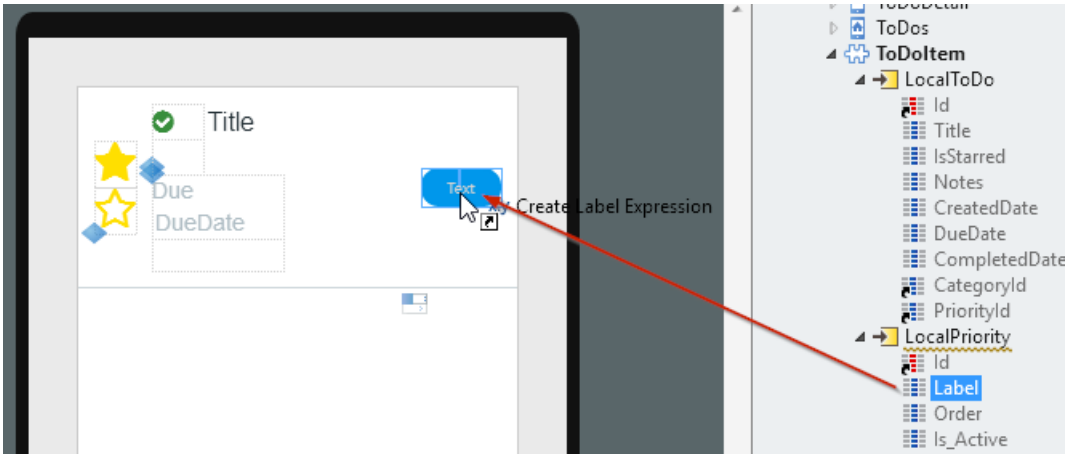

Figure 11. Drag and drop Priority Label attribute to create an Expression

v) Select the **List Item** Widget, then in the properties area set the **On Click** Event to navigate into the **ToDoDetail** Screen.

w) Set its Input Parameter to 'LocalToDo.Id'.

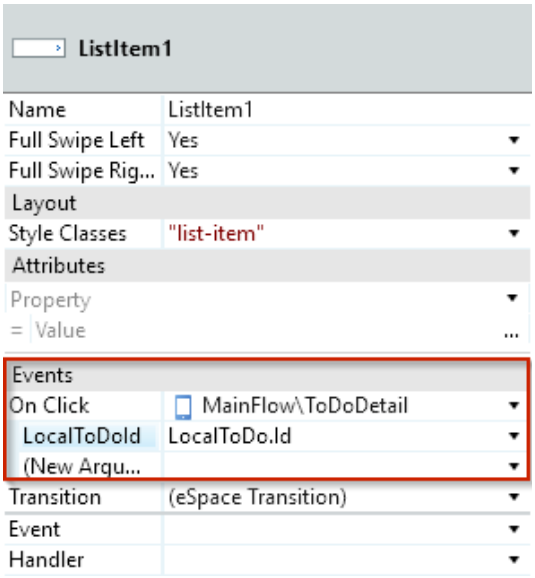

Figure 12. List Item OnClick Event

3. Toggle the **Is Starred** property of To Dos when the star icons are clicked.

a) Select the **If** Widget that contains both star icons, then right click it and choose 'Link to > (New Client Action)'

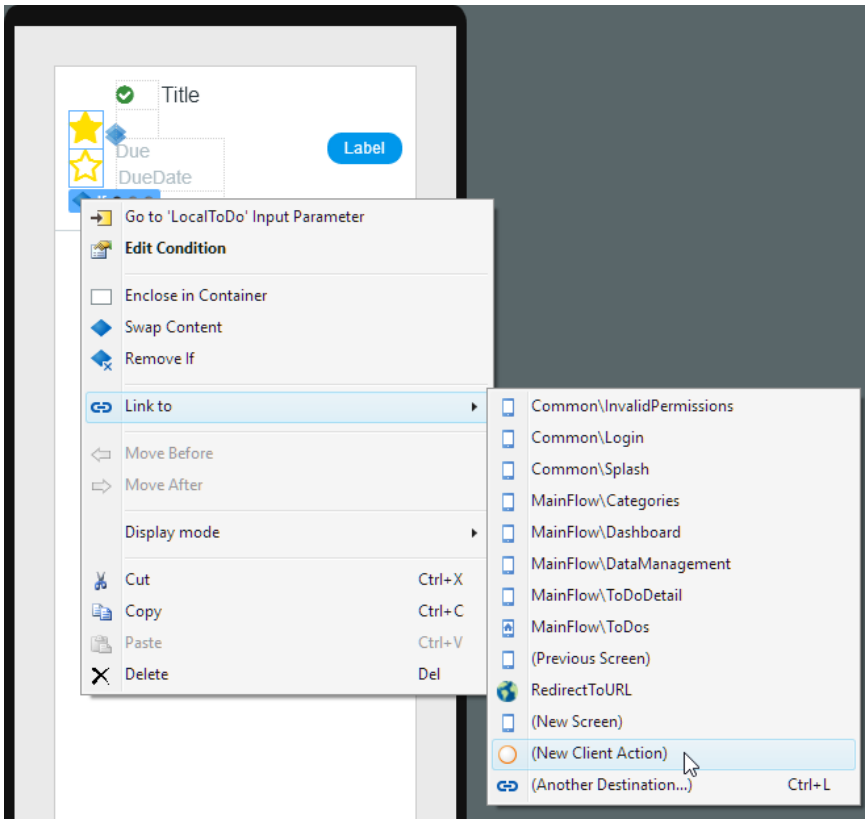

Figure 13. Link to new Client Action

b) Rename the Client Action to 'StarOnClick'.

 Do not duplicate

c) Drag an **If** statement and drop it after the Start, then set its **Condition** to

```
not GetNetworkStatus()
```

d) Drag a **Message** statement and drop it on the right side of the **If** statement.

e) In the properties of the **Message** statement, set the **Type** to 'Error' and the **Message** to "Unable to update To Dos while offline.".

f) Create the **True** branch connector from the **If** to the **Message** statement.

g) Add a new **End** statement on the right of the **Message** statement and create the connection between the **Message** and End statements.

h) Drag an **Assign** statement and drop it on the **False** branch connector, then add the following assignment

```
LocalToDo.IsStarred = not LocalToDo.IsStarred
```

i) Drag and drop a **Run Server Action** statement and drop it between the Assign and End.

j) In the **Select Action** dialog choose 'CreateOrUpdateToDoWrapper'.

k) Set the **ToDoInput** parameter to 'LocalToDo', then in the mapping set the missing **UserId** to 'GetUserId()'.

l) Drag a **Run Client Action** statement and drop it just before the End.

m) In the **Select Action** dialog choose 'CreateOrUpdateLocalToDo' then set the **Source** parameter to 'LocalToDo'.

n) The **StarOnClick** Client Action should look like this

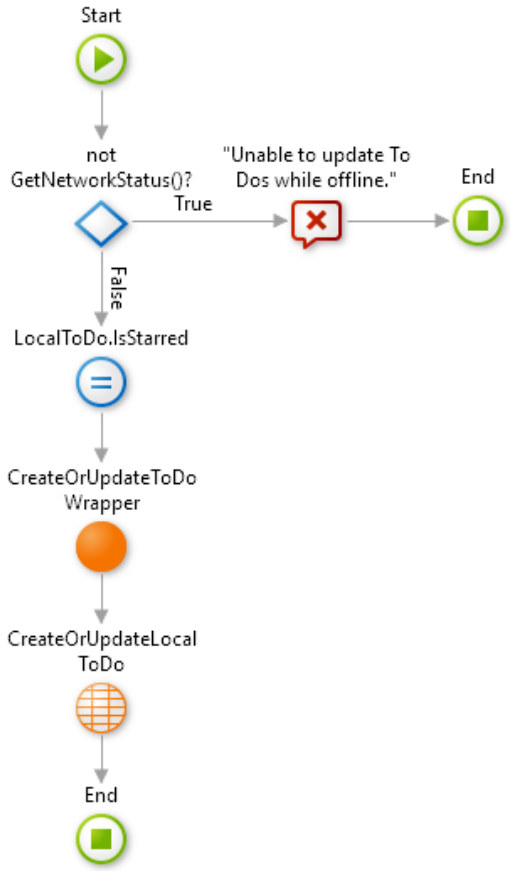

Figure 14. StarOnClick Client Action

o) Return the **ToDoItem** Block, and select the **Link** created just above.

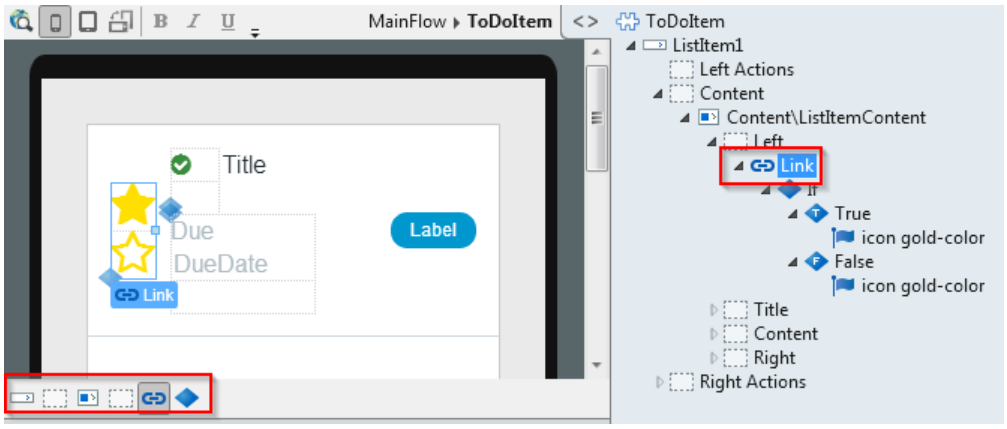

Figure 15. Link to StarOnClick Client Action

p) Set the **Enabled** property of the Link to

```
LocalToDo.CompletedDate = NullDate()
```

4. Define a swipe action to mark To Dos as complete.

a) Select the **List Item** Widget, then click the Expand circle to display the placeholders.

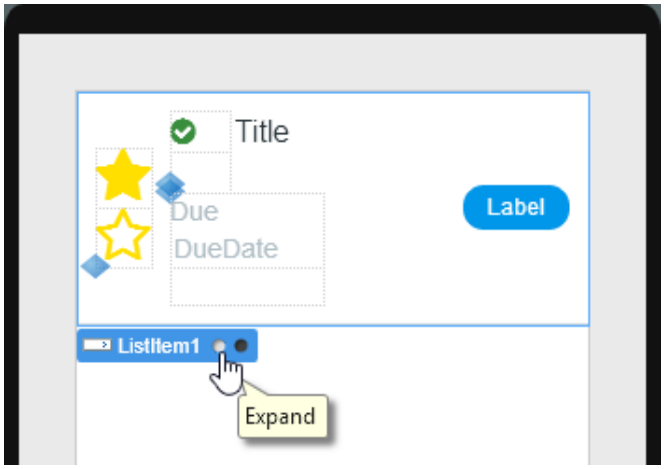

Figure 16. Expand placeholders

b) On the **List Action** on the right, change the 'Action' text to 'Complete'.

c) Select the List Action Widget, and set the **Visible** property to

```
LocalToDo.CompletedDate = NullDate()
```

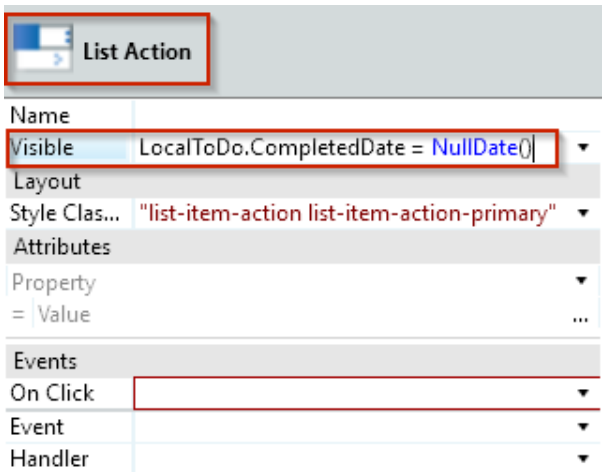

Figure 17. List Action properties

d) Set the **Style Classes** property to "list-item-action background-green".

e) Open the drop down of suggestions for the **On Click** event, then choose '(New Client Action)'

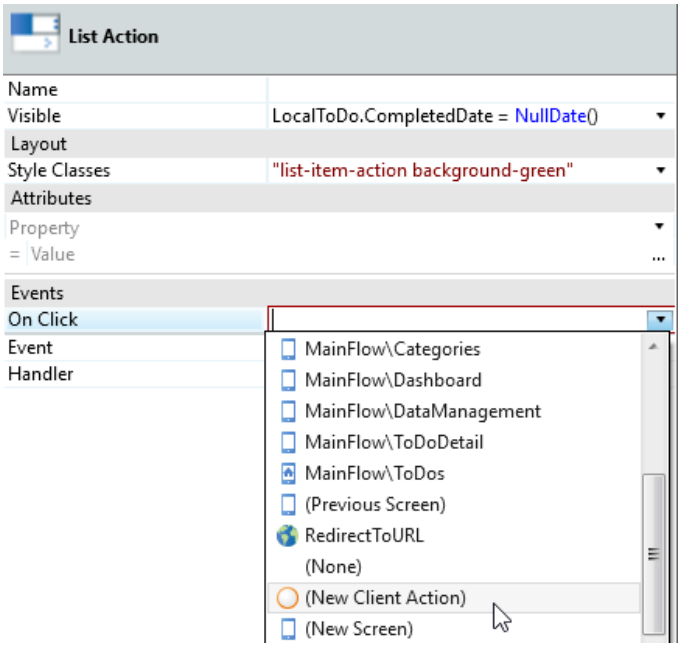

Figure 18. New Client Action for On Click Event

f) In the newly created Action, drag an **If** statement and drop it after the Start, then set its **Condition** to

```
not GetNetworkStatus()
```

g) Drag a **Message** statement and drop it on the right side of the **If** statement.

h) In the properties of the **Message** statement, set the **Type** to 'Error' and the **Message** to "Unable to update To Dos while offline.".

i) Create the **True** branch connector from the **If** to the **Message** statement.

j) Add a new **End** statement to the right of the **Message** statement and create the connection from the Message to End statement.

k) Drag an **Assign** statement and drop it on the **False** branch connector, then add the following assignment:

```
LocalToDo.CompletedDate = CurrDate()
```

l) Drag and drop a **Run Server Action** statement and drop it between the Assign and End.

m) In the **Select Action** dialog choose 'CreateOrUpdateToDoWrapper'.

n) Set the **ToDoInput** parameter to 'LocalToDo', then in the mapping set the missing **UserId** to '`GetUserId()`'.

o) Drag a **Run Client Action** statement and drop it just before the End.

p) In the **Select Action** dialog choose 'CreateOrUpdateLocalToDo' then set the **Source** parameter to 'LocalToDo'.

q) The **CompleteOnClick** Client Action should look like this

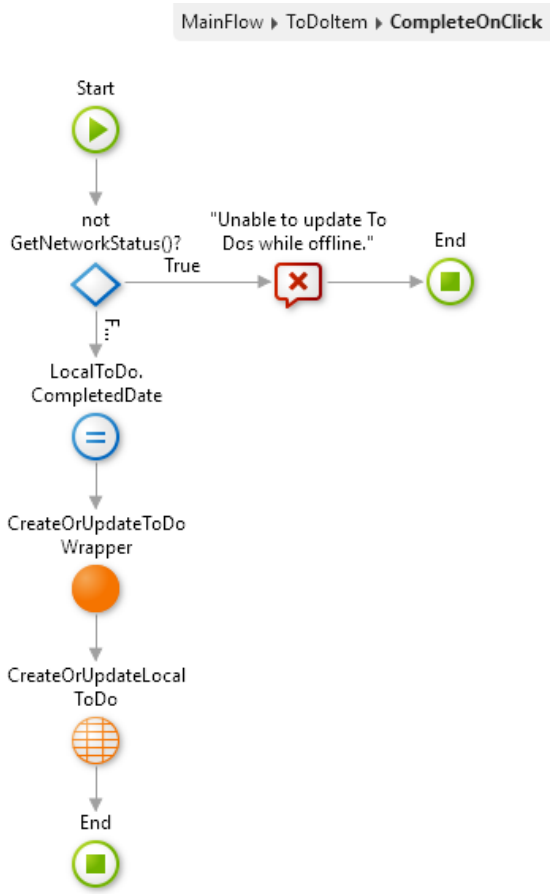

Figure 19. CompleteOnClick Client Action

Part 2: Instantiate Blocks

In this part of the exercise, you will modify the **ToDos** Screen to display an instance of **ToDoItem** Block for each To Do. You will also add the logic to prevent users from modifying To Dos already marked as complete.

1. Modify the **ToDos** list Screen and display an instance of the **ToDoItem** Block for each item in the list of To Dos.

a) Open the **ToDos** Screen and sing the Widget Tree, select the **ListItem1** Widget and delete it.

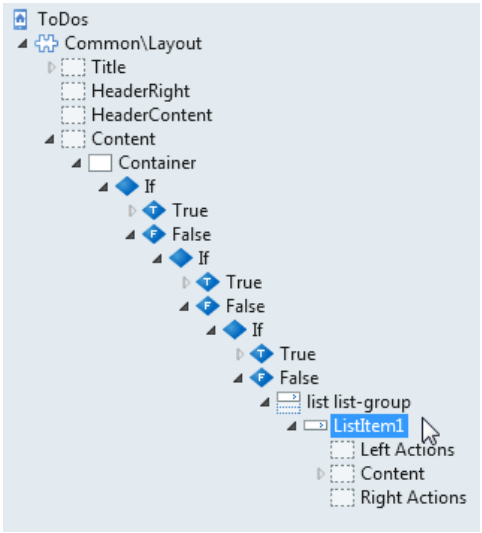

Figure 20. ListItem1 Widget

b) From the **Interface** tab, drag the **ToDoItem** Block and drop it inside the **List** Widget.

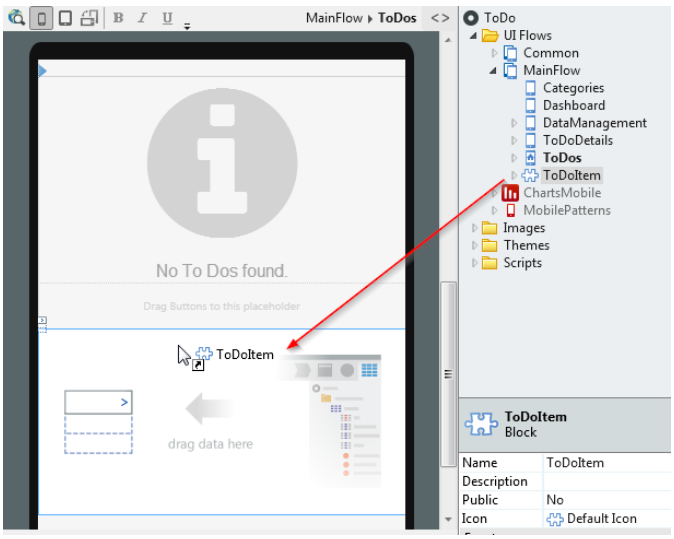

Figure 21. Drag and drop ToDoItem Block

c) Set the **LocalToDo** Input Parameter of the Block instance to 'GetToDos.List.Current.LocalToDo'.

> **NOTE:** Since the **GetToDos** Aggregate does not return the **LocalPriority** as Output, we cannot define the **LocalPriority** Block instance parameter, yet.

d) Double-click the **GetToDos** local Aggregate of the **ToDos** Screen to open the Aggregate Editor.

e) From the **Data** tab, drag the **LocalPriority** Entity and drop it into the Aggregate Editor.

f) Verify that the Join between the **LocalPriority** and **LocalToDos** Entities was created.

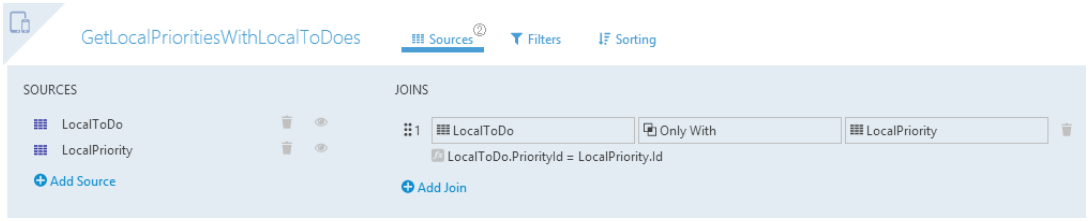

Figure 22. LocalToDo and LocalPriority Join

g) Return to the **ToDos** Screen.

h) Select the **ToDoItem** Block instance, and set the **LocalPriority** parameter to 'GetToDos.List.Current.LocalPriority'.

i) Notice that the Title and Due Date are centered. Right-click the Block instance and choose 'Enclose in Container'.

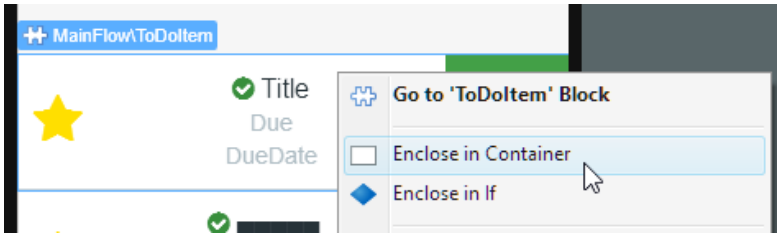

Figure 23. Enclose Block in Container

j) Set the **Align** property of the Container to 'Left'.

> **NOTE:** These last two steps are required because in our application, the whole content of the Screen is enclosed in a Container aligned to Center, and that is being propagated to the inner Widgets.

2. Prevent users from modifying To Dos already marked as completed.

a) Open the **ToDoDetail** Screen.

b) Select the **Input_Title** Widget and change the **Enabled** property to

```
GetLocalToDoById.List.Current.LocalToDo.CompletedDate =
NullDate()
```

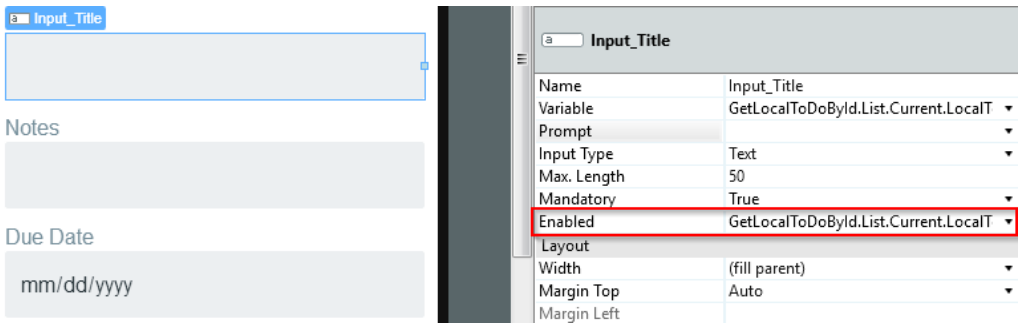

Figure 24. Enabled Property in the Input_Title

c) Repeat the previous step for the remaining inputs (Notes, Due Date, Category and Priority).

d) Change the **Visible** property of the **Save** Button to

```
GetLocalToDoById.List.Current.LocalToDo.CompletedDate =
NullDate()
```

e) Select the **Link** that toggles the 'IsStarred' attribute and change the **Enabled** property to

```
GetLocalToDoById.List.Current.LocalToDo.CompletedDate =
NullDate()
```


Part 3: Publish and Test

In this part of the exercise, you will publish the application to the server and then will test the block created previously.

1. Publish and test the new block.

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

b) In the **ToDos** list Screen, click the 'star' icon next to the 'Test Input Validations' To Do.

c) Notice that the star is now hollow, and the To Do has been updated.

d) Swipe the 'Test Input Validations' To Do from the right to the left.

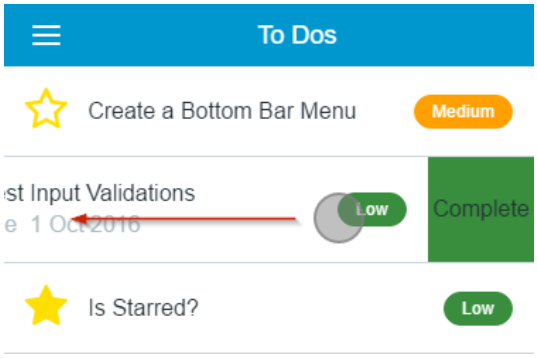

Figure 25. Swipe to Complete

NOTE: In your browser, you can click and drag to emulate a swipe action.

e) The green check mark should now appear next to the title.

f) Click the completed To Do to open the detail Screen, and verify that the **Form** is disabled and the To Do cannot be modified.

End of Lab

In this exercise, you created a new Block that displays the information of each To Do. This Block was used to display the information of each To Do in the **ToDos** list Screen.

The logic was further extended to prevent users from modifying To Dos marked as completed.

List of Figures

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

