



DEVELOPING OUTSYSTEMS MOBILE APPS

List and Detail Screens



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:

- Build the To Do list and detail Screens
- Create navigation links between the list and detail Screens

Table of Contents

| | |
|---------------------------------------|----|
| Introduction..... | 2 |
| Table of Contents..... | 3 |
| Part 1: Create a List Screen | 4 |
| Part 2: Edit the Detail Screen | 15 |
| Part 3: Create Navigation Links | 25 |
| Part 4: Publish and Test | 27 |
| End of Lab | 28 |
| List of Figures..... | 29 |

Part 1: Create a List Screen

In this part of the exercise, you will modify the existing **ToDo** Screen to display the list of To Dos.

1. Add references to the **ToDo_Core** Entities that will be used in the list.

a) In the **ToDo** module, click the 'Manage Dependencies...' icon.

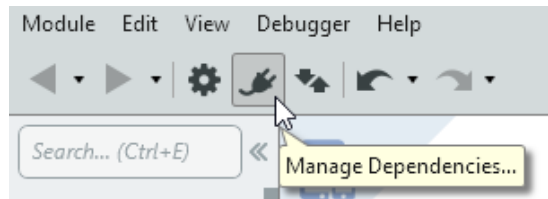


Figure 1. Manage Dependencies

b) In the **Manage Dependencies** dialog, select the **ToDo_Core** module on the left, and then tick all Database Entities.

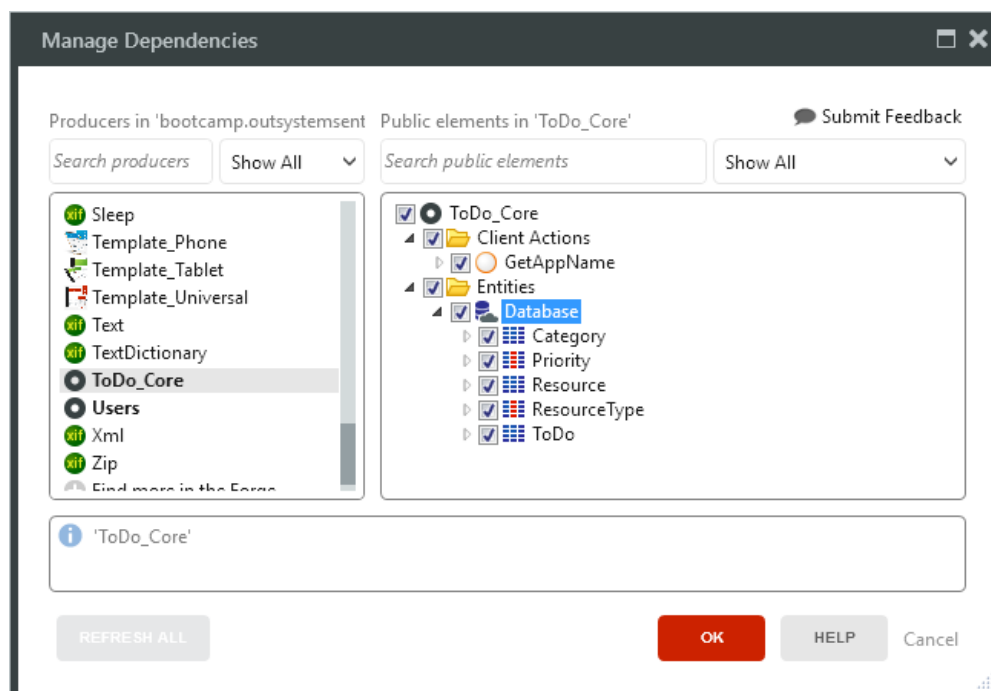


Figure 2. Add references to all Database Entities

c) Click **Ok** to add the new dependencies.

2. Fetch To Dos from the database **ToDo** Entity.

a) In the **Interface** tab, right-click the **ToDo** Screen item and select 'Fetch Data from data Database'.

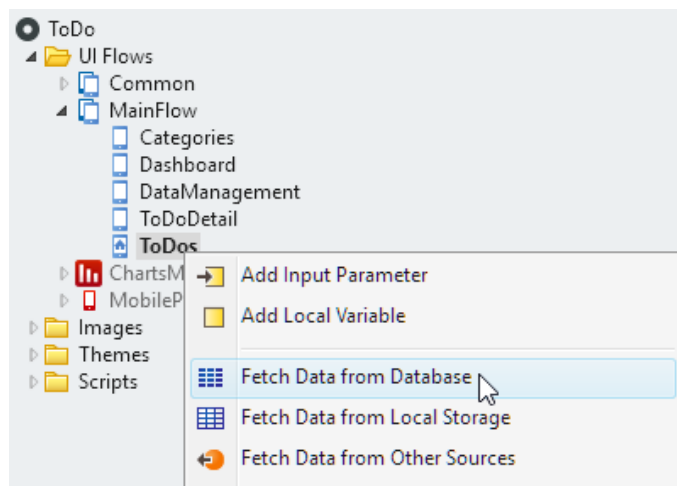


Figure 3. Fetch Data from Database

- b) A new Aggregate named **Aggregate1** was created and opened. From the **Data** tab locate the **ToDo** Entity and drag it into the Aggregate.

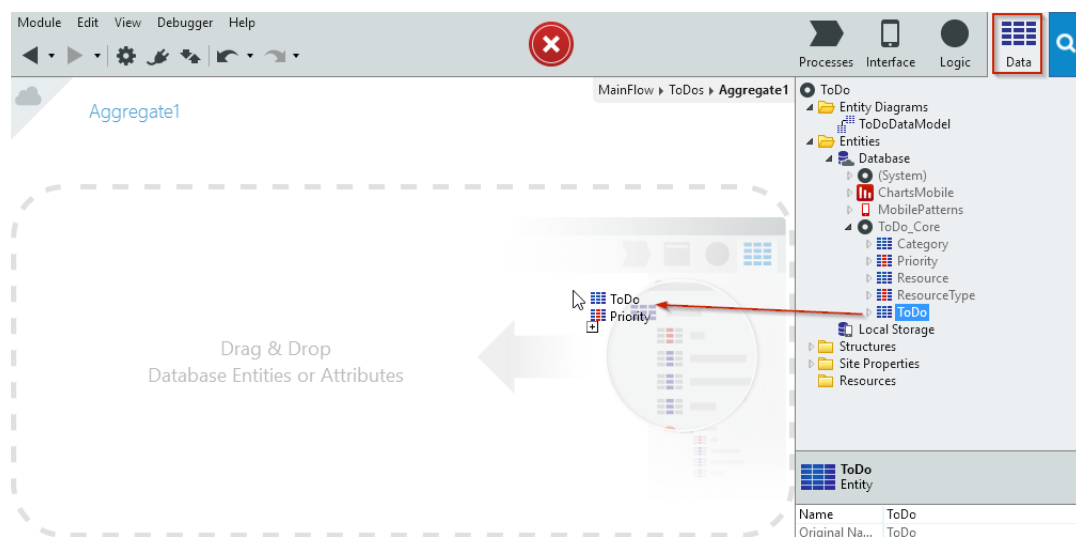


Figure 4. Add ToDo Entity to an Aggregate

- c) Notice that the Aggregate name has changed to 'GetTodos'.
- d) In the **Filters** tab, click the **+Add Filter** to create a new filter.

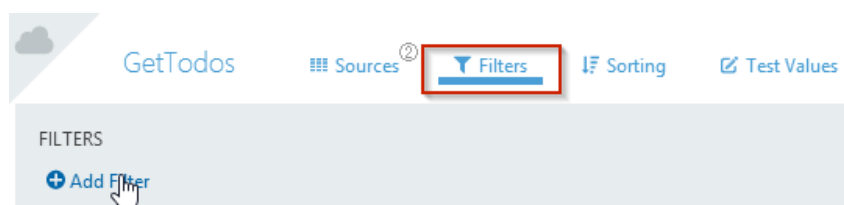


Figure 5. Add Filter to Aggregate

- e) Set the Filter Condition to

```
ToDo.UserId = GetUserId()
```

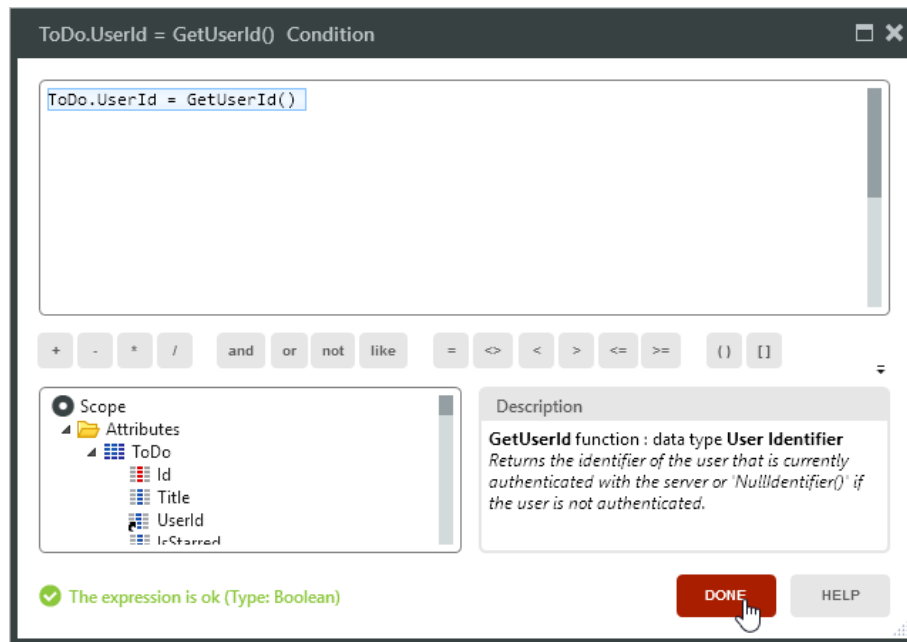


Figure 6. Filter Condition Expression Editor

NOTE: The function **GetUserId()** returns the identifier of the user that is currently authenticated with the server, or **NullIdentifier()** if the user is not authenticated.

- f) Notice that the Aggregate name has changed to 'GetTodosByUserId'.
 - g) Click **Done** to close the Expression Editor.
- 3.** Define the Screen logic to display the List of To Dos returned by the Aggregate.
- a) Switch to the **Interface** tab and open the **ToDo**s Screen.
 - b) Select the "Hello from " Expression created in an earlier exercise and delete it.
 - c) Drag a new **If** Widget and drop it to the main area of the Canvas, in the **Content** placeholder of the **ToDo**s Screen, as shown in the following screenshot

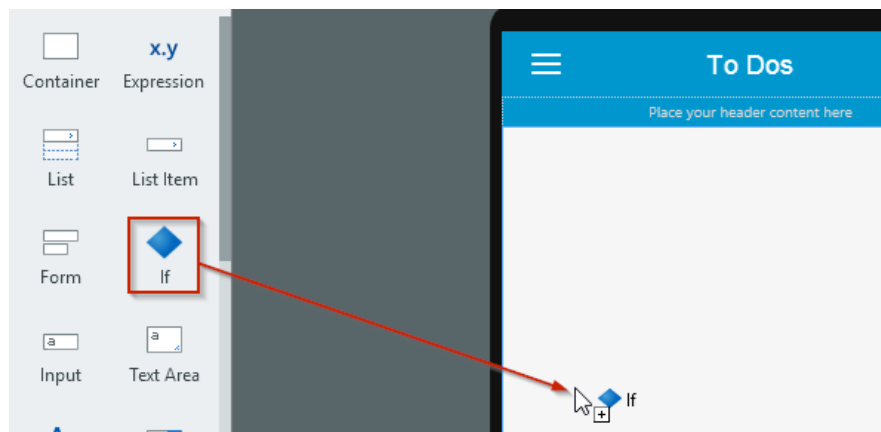


Figure 7. Drag If Widget

d) Right click the **If** Widget and enclose it in a **Container**.

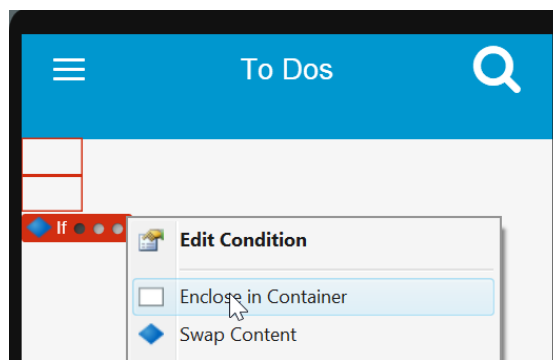


Figure 8. Enclose the If Widget in a Container

e) Select the newly created Container and check its Properties area.

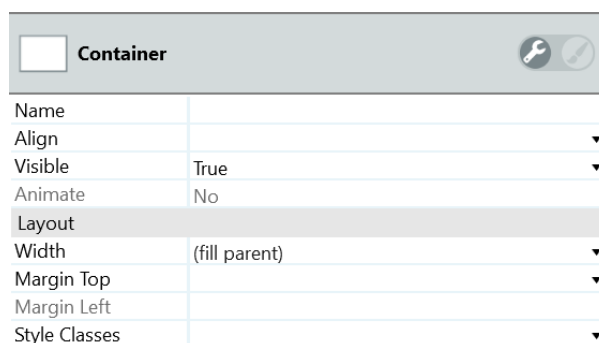


Figure 9. Properties area of a Container

f) On the top right corner of the Properties area, toggle the Styles Editor view, by clicking on the respective icon.



Figure 10. Change from the Properties to the Styles Editor area

NOTE: The Styles Editor is a panel for editing basic visual properties of widgets. It can be used to change the font color, margins, border thickness, or the alignment, as in this example. You can find more about it here.

g) Now you should see the Styles Editor are, as seen in the following screenshot.

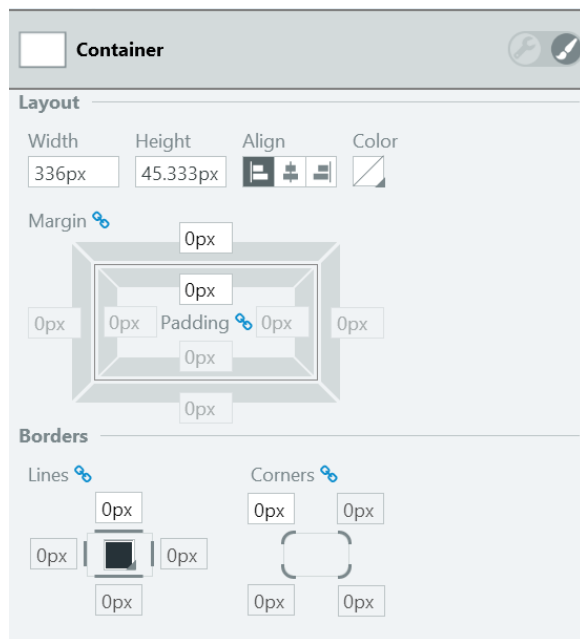


Figure 11. Styles Editor area

h) Align the **Container** (and the If within it) to the center of the page.

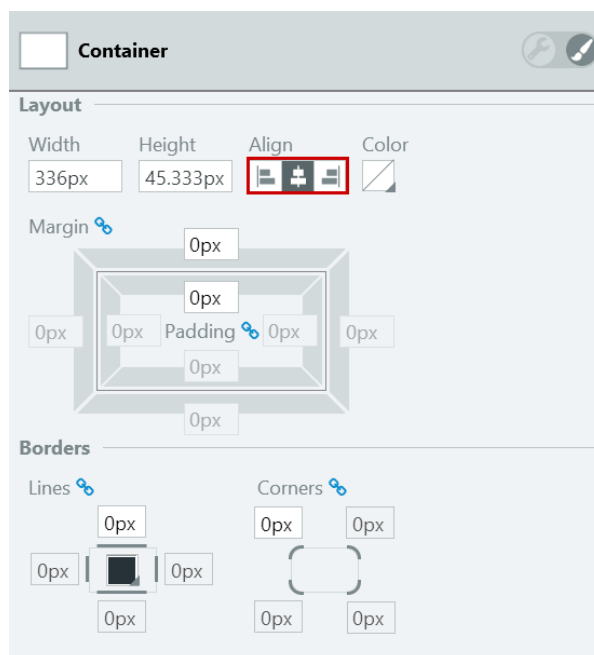


Figure 12. Align a Container to center

- i) Set the **Condition** of the **If** inside the newly created **Container** to

```
GetTodosByUserId.HasFetchError
```

NOTE: The **HasFetchError** is an Output Parameter of the Database Aggregate. This value is True when there is an error during the data fetching, due to a server error or communication timeout.

- j) Set the **Animate** property of the **If** Widget to 'Yes'

NOTE: The **Animate** property performs an animation on the content when the If condition changes its value.

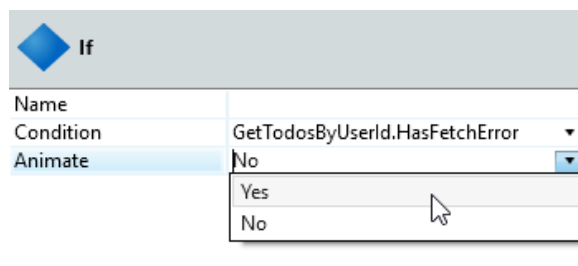


Figure 13. Animate property of If Widget

- k) Drag a **Blank Slate** Widget and drop it inside the **True** branch of the **If** Widget.

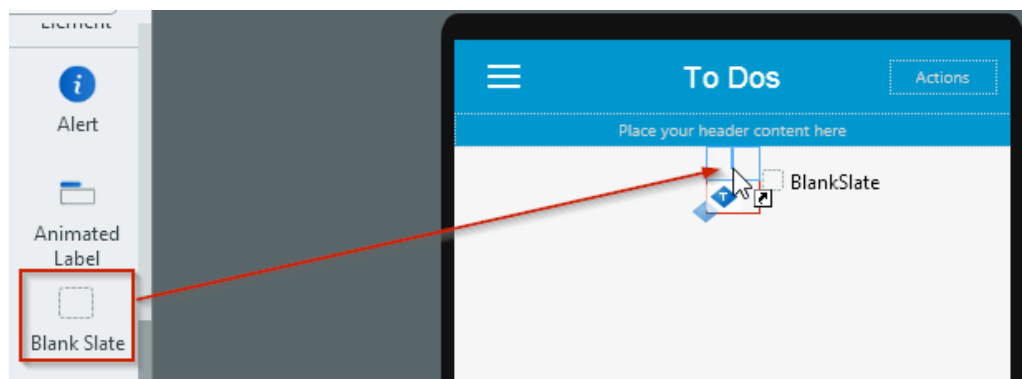


Figure 14. Drag a Blank Slate Widget

- l) Notice that the **BlankSlate** has three placeholders: **Icon**, **Description** and **Actions**.

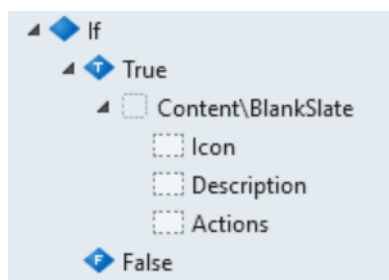


Figure 15. BlankSlate Placeholders

- m) If the **BlankSlate** pattern comes with any content inside it, delete it. This content is an example of usage of the pattern. In the following patterns that you are going to use, delete any content that may be within the pattern, just like you did with the BlankSlate.
- n) Drag an **Icon** Widget to the **Icon** placeholder in the Blank Slate, select the unhappy smiley face (frown) and click **Ok**.
- o) Inside the **Description** placeholder of the Blank Slate type 'An error occurred while fetching To Dos.'.
- p) The Screen should look like this.

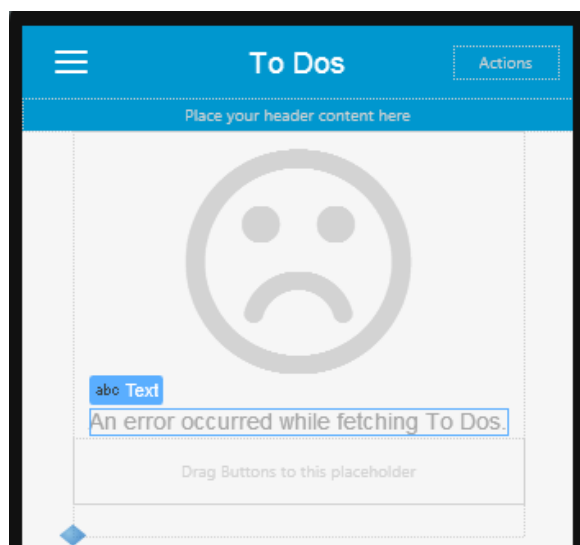


Figure 16. Blank Slate with Icon and error message

- q) Drag other **If** Widget and drop it inside the **False** branch of the existing If.

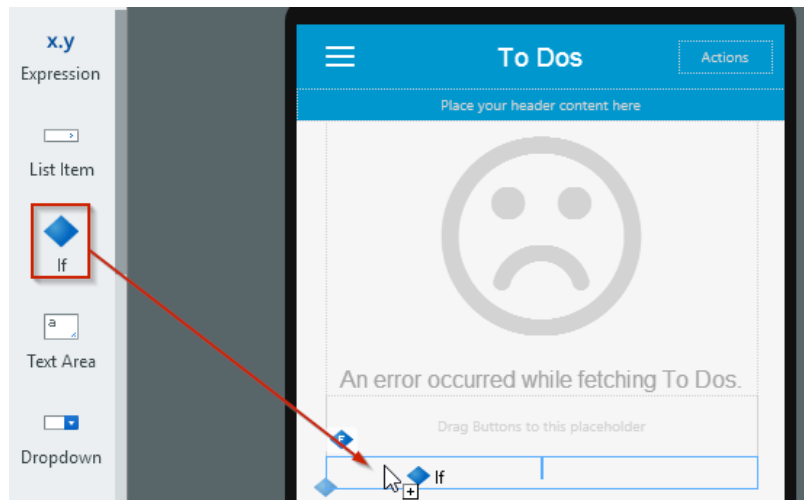


Figure 17. Drag and drop If Widget inside False branch

- r) In the properties of the new If, set the **Animate** property to 'Yes' and the **Condition** to

```
not GetTodosByUserId.IsDataFetched
```

NOTE: **IsDataFetched** is a runtime property of the Aggregate, which has the value True when data has been fetched from the database and it is ready to be used.

- s) Drag a new **Blank Slate** Widget to the **True** branch of the newly create If.
- t) Drag an **Icon** Widget to the **Icon** placeholder in the Blank Slate and choose the 'hourglass-half'. Don't forget to delete the Icon that was generated after dragging the pattern, if any exists.
- u) Type 'Loading...' in the **Description** placeholder. The resulting Screen and Widget Tree should look like the following screenshot

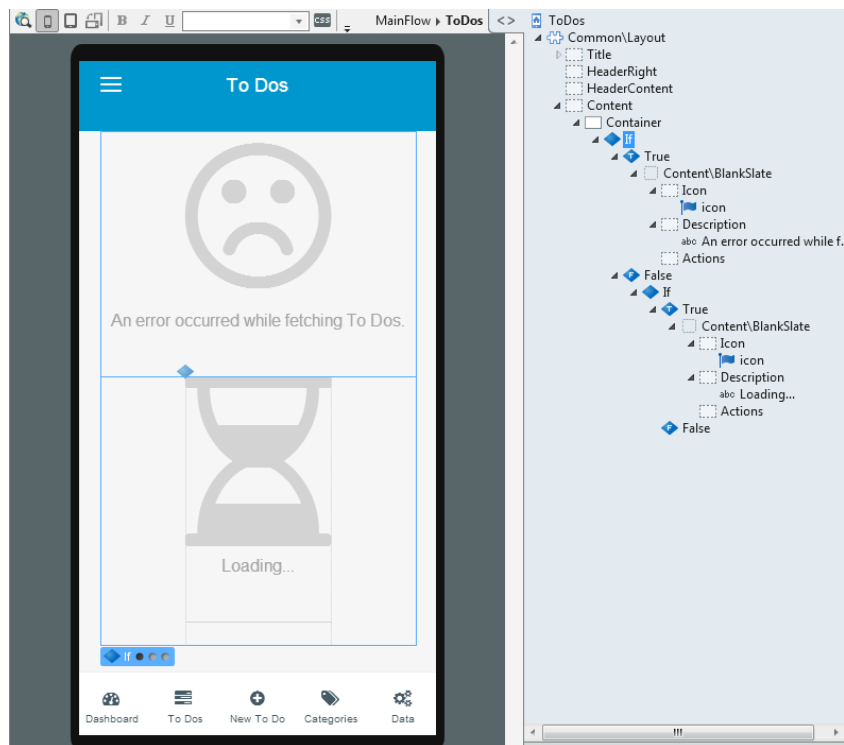


Figure 18. ToDos Screen and Widget Tree

- v) Drag a new **If** Widget and drop it inside the **False** branch of the last **If**.

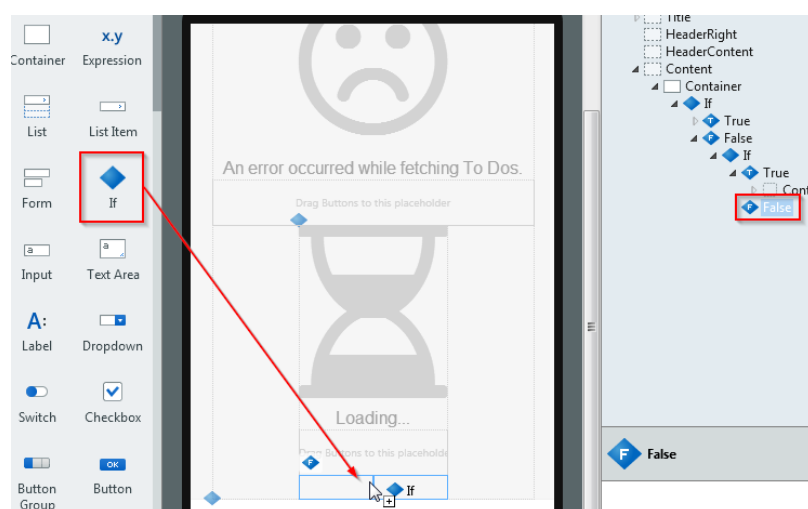


Figure 19. Drag and drop If Widget

- w) Set the **Animate** property to 'Yes' and the **Condition** to

```
GetTodosByUserId.List.Empty
```

- x) Drag another **Blank Slate** and drop it inside the **True** branch of the **If** added just before.
- y) Drag an **Icon** Widget to the **Icon** placeholder and choose the 'info circle'.

- z) Type 'No To Dos found.' in the **Description** placeholder of the Blank Slate.
4. Define the Screen logic to display the list of To Dos returned by the Aggregate.
- a) Drag a **List** Widget and drop it inside the empty **False** branch.

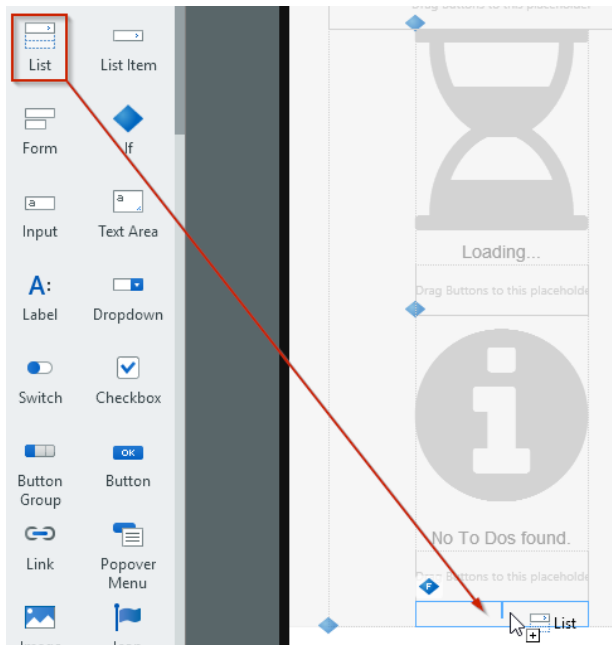


Figure 20. Drag List Widget

- b) The Widget Tree of the **ToDo**s Screen should look like this

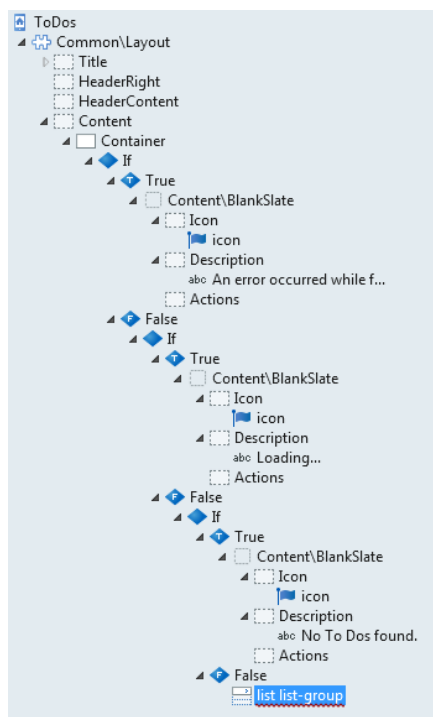


Figure 21. Widget Tree

- c) Expand the **GetTodosByUserId** Aggregate and locate the **Title** attribute of the **ToDo** Entity. Drag the **Title** attribute and drop it inside the List.

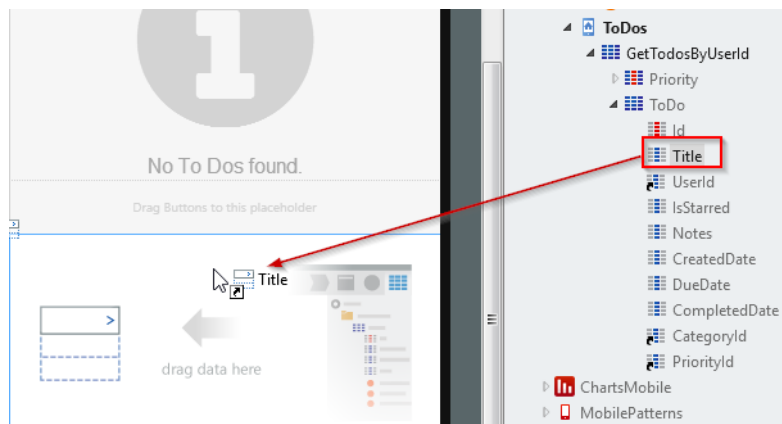


Figure 22. Drag Aggregate result attribute into the List Widget

NOTE: By dragging an attribute from an Entity of the **GetTodosByUserId** Aggregate, the Source property of the List was automatically set to 'GetTodosByUserId.List'.

Part 2: Edit the Detail Screen

In this part of the exercise, you will edit the **ToDoDetail** Screen so that the users are able to create and edit To Dos.

1. Set the title of the **ToDoDetail** Screen to reflect whether we are editing an existing To Do or creating a new To Do.

a) Right click the **ToDoDetail** Screen and select 'Add Input Parameter'.

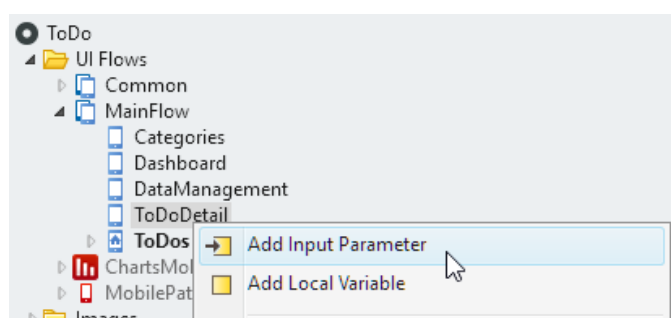


Figure 23. Add Input Parameter

- b) Set the new Input Parameter name to 'ToDold' and verify that the **Data Type** property is set to **ToDo Identifier**.

| ToDold Input Parameter | |
|---------------------------|-------------------|
| Name | ToDold |
| Description | ... |
| Data Type | ToDo Identifier ▼ |
| Is Mandatory | Yes ▼ |
| Default Value | |

Figure 24. ToDold Input Parameter properties

NOTE: The **TrueChange** tab should display two errors at this point. These are related to the existing links in the **Menu** and **Bottom Bar**, which were added before creating the Input Parameter. This will be fixed later.

- c) Right-click the **ToDoDetail** Screen and select 'Fetch Data from Database'.

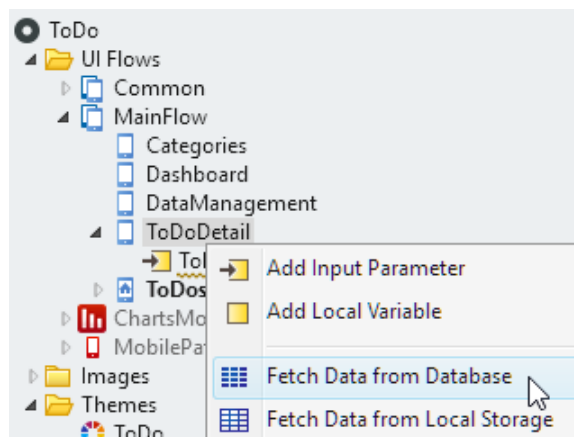


Figure 25. Create a new Screen Aggregate

- d) Drag the **ToDold** Input Parameter and drop it inside the **Aggregate1** editor.

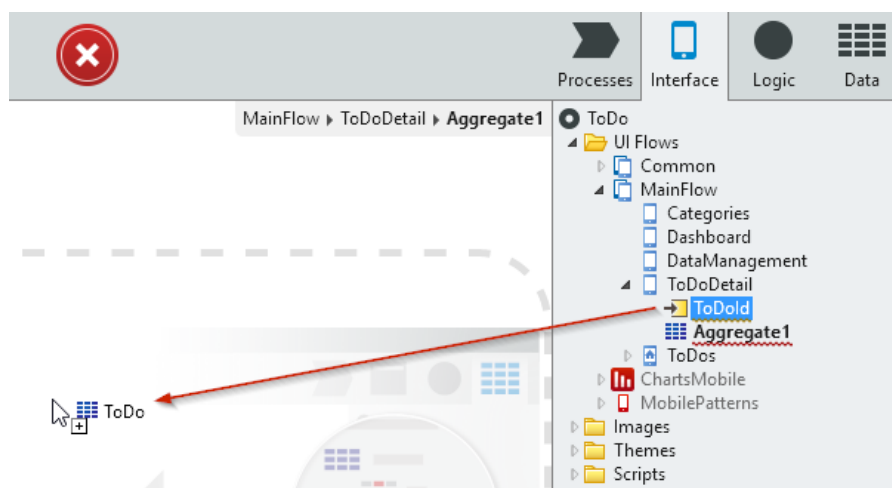


Figure 26. Drag ToDold into Aggregate

NOTE: By dragging the **ToDold** Input Parameter, the **ToDo** Entity was added as a Source Entity to the Aggregate. Also, a new Filter Condition was created to filter the **ToDos** Entity, based on the To Do Identifier. Notice also that the Aggregate has been renamed to 'GetToDoById'

- e) Double-click the **ToDoDetail** Screen to open it and delete the text that is inside its **Title** placeholder.
- f) Drag an **If** Widget and drop it inside the **Title** placeholder of the **ToDoDetail** Screen.

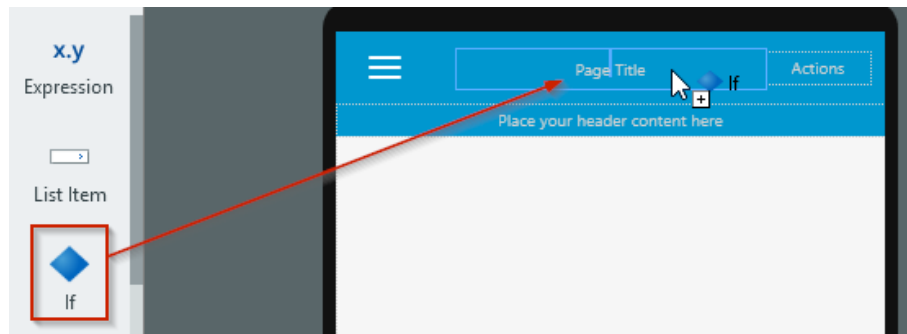


Figure 27. Add an If Widget to the Screen Title placeholder

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

```
ToDoId <> NullIdentifier()
```

h) Expand the **GetToDoById** Aggregate and drag the **Title** attribute of the **ToDo** Entity and drop it on the **True** branch of the If Widget.

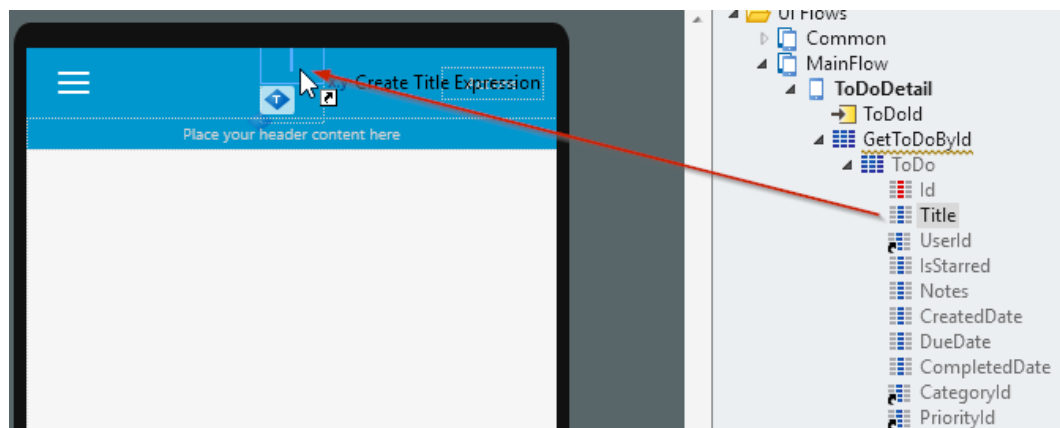


Figure 28. Create Title Expression

i) Inside the **False** branch, type 'Create New To Do'.

2. Create a **Form** to allow the **ToDoDetail** Screen to accept input from the users, to fill information for the **ToDo** attributes.

a) Drag a **Form** Widget and drop it inside the Screen's **Content** placeholder.

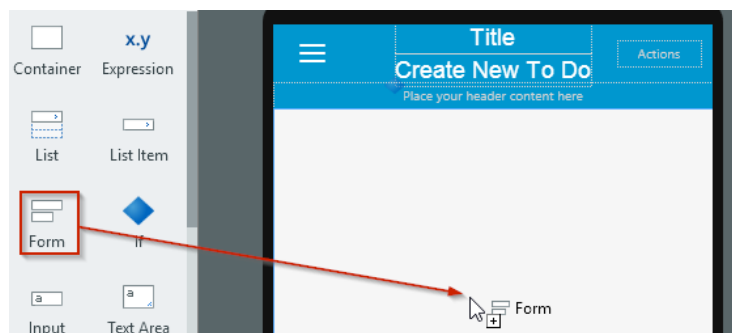


Figure 29. Drag a Form Widget

- b) Locate the **Title** attribute of the **ToDo** Entity, from the **GetToDoById** Aggregate, then drag it and drop it inside the **Form** Widget.
- c) Drag the **Notes** attribute and drop it between the **Title** input and the **Save** Button.

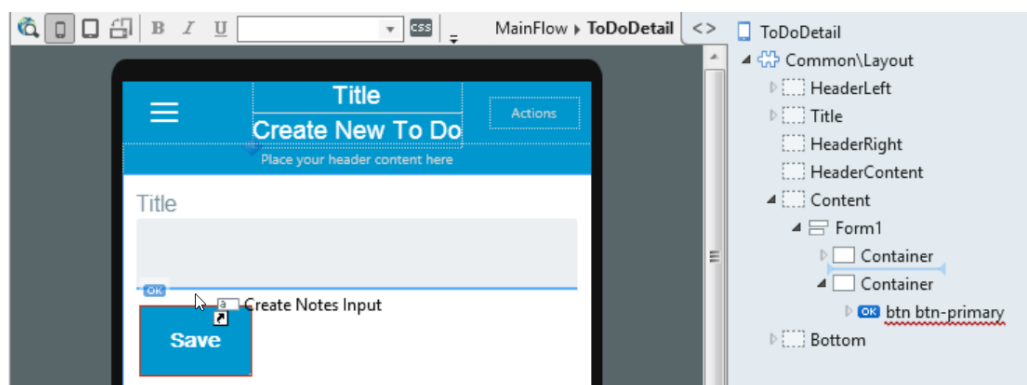


Figure 30. Drag and drop the Notes attribute to the Form Widget

- d) Repeat the same step for the **DueDate** and **CategoryId** attributes.

NOTE: Dragging the **CategoryId** attribute created a **Dropdown**, with all the Categories options. This happened because you dragged the foreign key attribute of the **ToDo** Entity. Also, a new Aggregate, **GetCategories**, was added to the Screen to populate the Dropdown.

- e) Drag a **Container** Widget and drop it between the Categories Dropdown and the **Save** Button.
- f) Drag a **Label** Widget and drop it inside the **Container** created in the previous step and set the **Text** of the Label to 'Priority'.

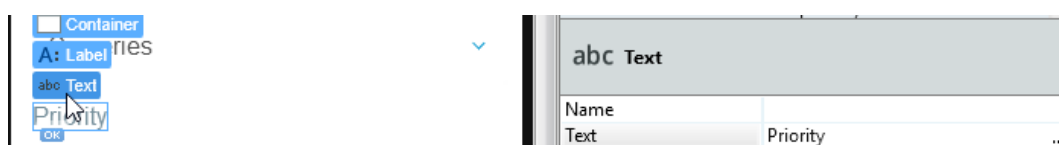


Figure 31. Change the Label Text

- g) Drag a **Button Group** Widget and drop it below the **Label**, but still inside the surrounding **Container**.

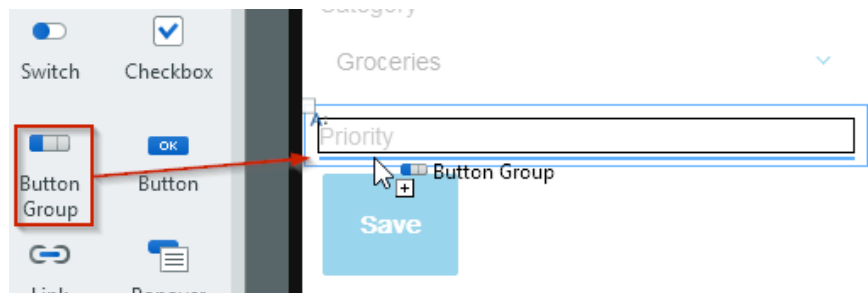


Figure 32. Drag and Drop a Button Group

- h) Set the **Name** property of the **Button Group** to 'PriorityGroup'.
- i) Set the **Variable** property to 'GetToDoById.List.Current.ToDo.PriorityId'.

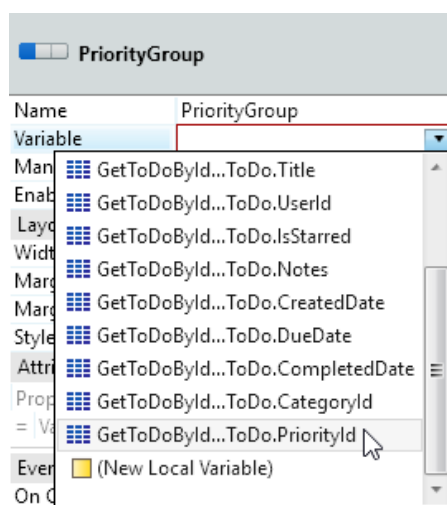


Figure 33. Set Variable property of the PriorityGroup Widget

- j) Change the text of the first ButtonGroupItem in the group to 'Low', the text of the second ButtonGroupItem to 'Medium', and the last one to 'High'.
- k) Select the first ButtonGroupItem of the **PriorityGroup** and set the **Value** property to 'Low'.

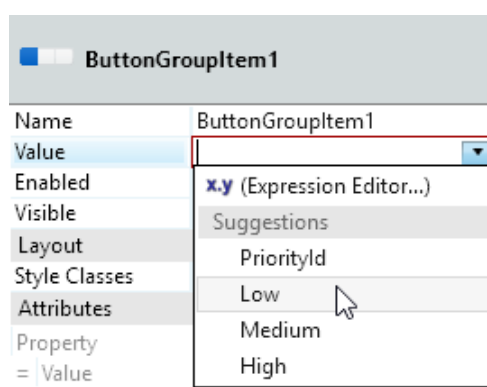


Figure 34. Set 'Low' in the Value property

- l) Set the **Values** for the two remaining ButtonGroupItems in the group to 'Medium' and 'High' respectively.
3. Create the Logic to save and create new To Dos in the database.
 - a) Double-click the **Save** Button to create the **SaveOnClick** Client Action.
 - b) Drag a **Run Server Action** into the Action flow and drop it in the **True** branch of the **If** statement.

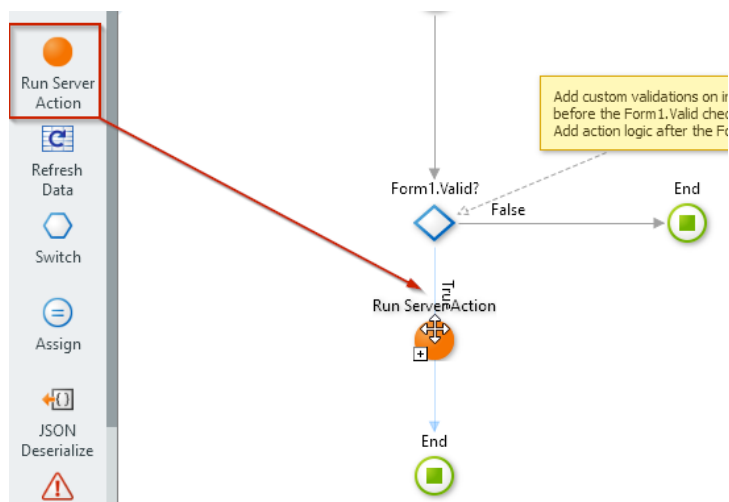


Figure 35. Drag Run Server Action statement

- c) In the **Select Action** dialog choose the 'New Server Action'. A new Server Action named 'Action1' should have been created.
- d) Double-click the **Action1** to open its flow, and rename it to 'CreateOrUpdateToDoWrapper'.
- e) In the **Logic** tab, right click the **CreateOrUpdateToDoWrapper** Server Action and choose 'Add Input Parameter'.

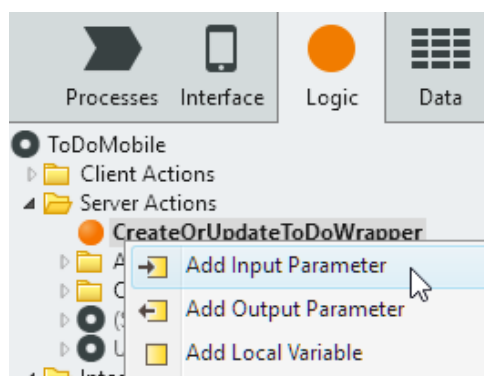


Figure 36. Add Input Parameter to CreateOrUpdateToDoWrapper Server Action

- f) Set the name of the Input Parameter to 'ToDo' and set its **Data Type** to **ToDo** (located under the Entities section of the pulldown menu).

| ToDo Input Parameter | |
|-------------------------|--------|
| Name | ToDo |
| Description | ... |
| Data Type | ToDo ▼ |
| Is Mandatory | Yes ▼ |
| Default Value | |

Figure 37. ToDo Input Parameter

- g) Add an Output Parameter and name it 'ToDoId'. Verify that its **Data Type** has automatically changed to **ToDo Identifier**.

| ToDoId Output Parameter | |
|----------------------------|-------------------|
| Name | ToDoId |
| Description | ... |
| Data Type | ToDo Identifier ▼ |
| Default Value | |

Figure 38. ToDoId Output Parameter

- h) In the Server Action flow, drag an **Assign** statement and drop it between the Start and End. Then, define the following assignment

```
ToDo.UserId = GetUserId()
```

- i) Drag a **Run Server Action** statement into the flow, and drop it between the previous Assign and the End.
- j) In the **Select Action** dialog choose the **CreateOrUpdateToDo** Entity Action.

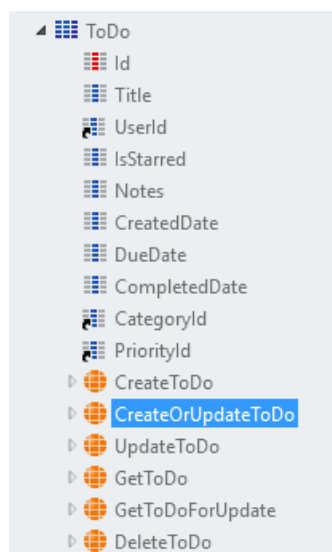


Figure 39. Select the CreateOrUpdateToDo Action

NOTE: This Server Action wrapper was not strictly necessary, as the **CreateOrUpdateToDo** Entity Action could be called from the Client Action. However, as the **UserId** attribute is assigned, it is best practice in terms of security, to do it server-side. And for that, the wrapper Action is necessary.

- k) Set the **Source** parameter to the **ToDo** Input Parameter of the Server Action.

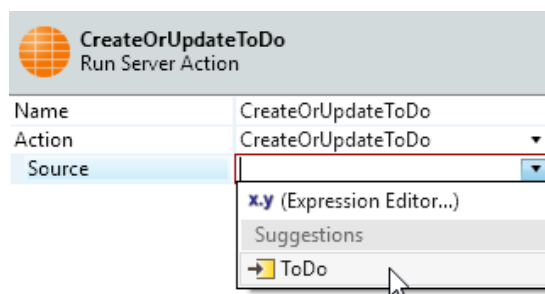


Figure 40. Set the Source parameter to the Input Parameter

- l) Drag another **Assign** statement and drop it between the **CreateOrUpdateToDo** statement and End.

- m) Define the following assignment

```
ToDoId = CreateOrUpdateToDo.Id
```

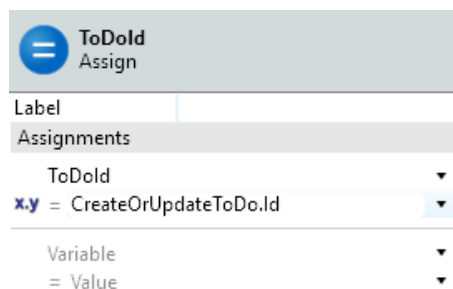


Figure 41. ToDoId Output Parameter assignment

- n) Your **CreateOrUpdateToDoWrapper** should look like this

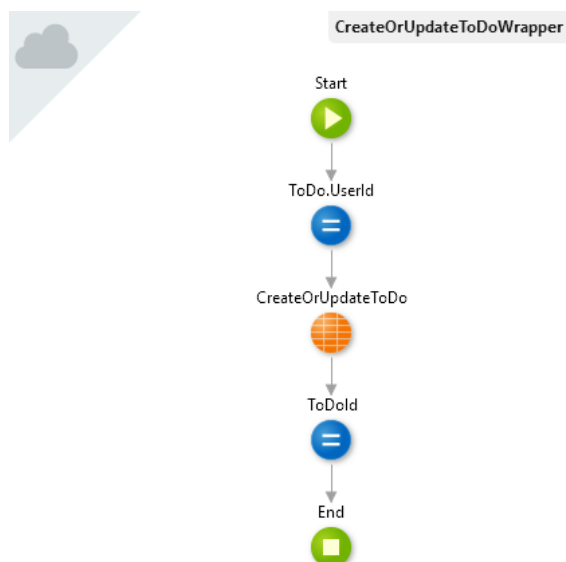


Figure 42. CreateOrUpdateToDoWrapper Action flow

- o) Return to the **SaveOnClick** Client Action of the **ToDoDetails** Screen.
- p) Select the **CreateOrUpdateToDoWrapper** statement and set the **ToDo** parameter to 'GetToDoById.List.Current.ToDo'.

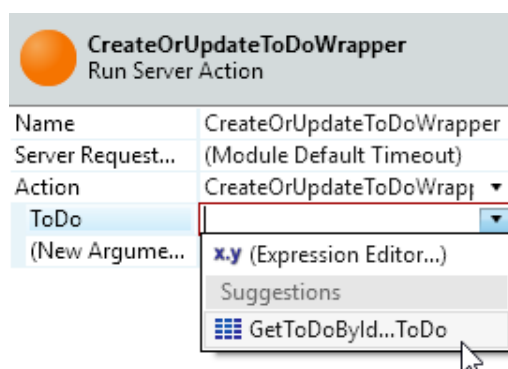


Figure 43. Set the ToDo Parameter for the CreateOrUpdateToDoWrapper Action

- q) Drag a **Message** statement and drop it between the **CreateOrUpdateToDoWrapper** and **End**.
- r) Set the **Type** to 'Success' and the **Message** property expression to:

"To Do '" + GetToDoById.List.Current.ToDo.Title + "' saved."

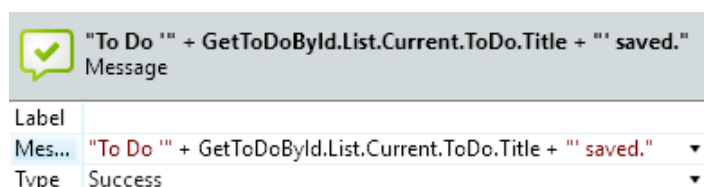


Figure 44. SaveOnClick Success Message

s) Drag a **Destination** statement and drop it on top of the End to replace it.



Figure 45. Replace End by a Destination

t) In the **Select Destination** dialog choose the **ToDos** Screen.

Part 3: Create Navigation Links

In this part of the exercise, you will edit the **ToDos** Screen so that the details of each To Do are accessible from there.

1. Add a Link from each To Do in the **ToDos** Screen to the **ToDoDetail** Screen.

a) Open the **ToDos** Screen and select the **ListItem1** Widget that contains the **Title** Expression.

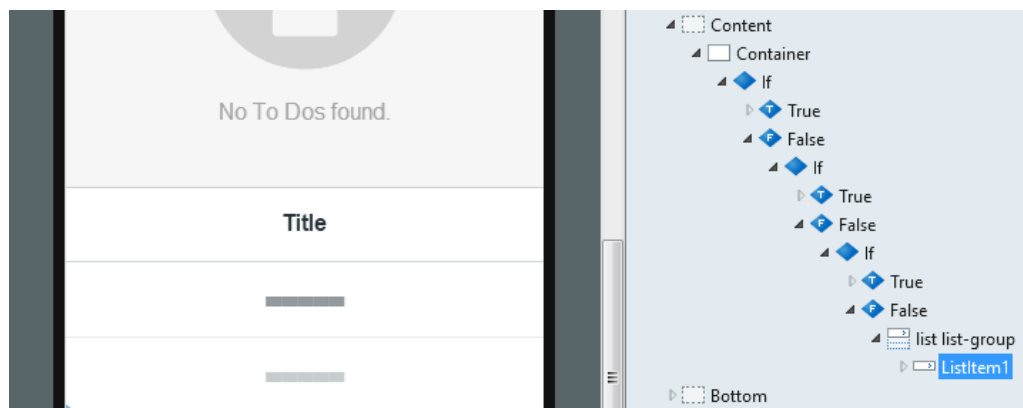


Figure 46. ListItem1 Widget

b) In the properties area, set the **On Click** Event to 'MainFlow\ToDoDetail'.

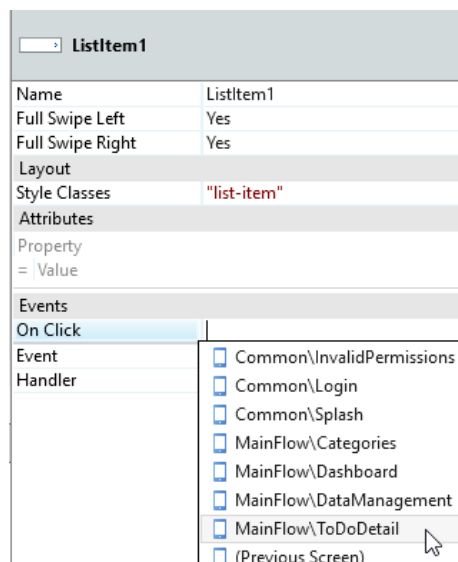
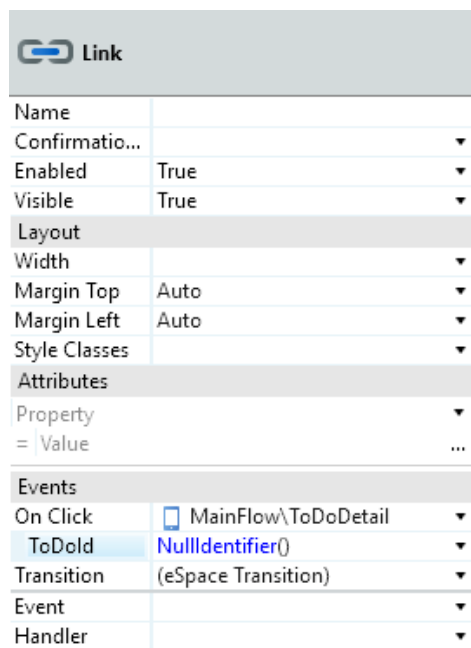


Figure 47. OnClick Event of ListItem1

c) Open the dropdown of suggestions for the **ToDoid** parameter and select 'GetTodosByUserId...ToDo.Id'.

2. Fix the Links to the **ToDoDetail** Screen in the Menu and Bottom Bar.

- a) Open the **Menu** Block located under the **Common** flow and select the **Link** that contains the 'Create New To Do' text.
- b) In the properties of the Link, set the **ToDold** parameter to `'NullIdentifier()'`



| Link | |
|----------------|--|
| Name | |
| Confirmatio... | ▼ |
| Enabled | True ▼ |
| Visible | True ▼ |
| Layout | |
| Width | ▼ |
| Margin Top | Auto ▼ |
| Margin Left | Auto ▼ |
| Style Classes | ▼ |
| Attributes | |
| Property | ▼ |
| = Value | ... |
| Events | |
| On Click | <input type="checkbox"/> MainFlow\ToDoDetail ▼ |
| ToDold | NullIdentifier() ▼ |
| Transition | (eSpace Transition) ▼ |
| Event | ▼ |
| Handler | ▼ |

Figure 48. ToDold parameter

- c) Open the **BottomBar** Block, also located under the **Common** flow.
- d) Select the 'New To Do' Link and then set the **ToDold** parameter to `'NullIdentifier()'`.

Part 4: Publish and Test

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

1. Publish the 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.
- c) Click the **Open in Browser** button to open the application.
- d) Open the **Menu** by clicking the 3-dashes icon on the top left of the emulator, then click the 'Create new To Do' link.
- e) Fill in the form to create a new To Do, then press **Save**.

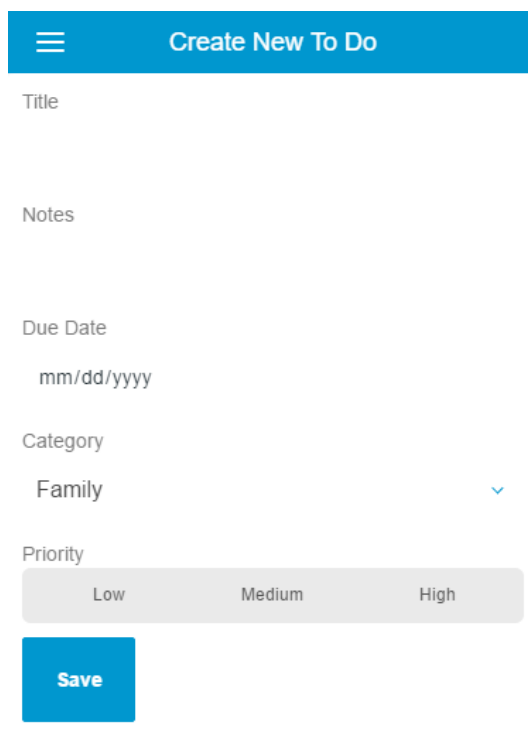


Figure 49. Create the first To Do

- f) In the **ToDos** Screen you should now see the newly created To Do.
- g) Clicking the To Do title in the **ToDos** Screen, should open the details of the To Do, where you can edit it (**ToDoDetail** Screen).

End of Lab

In this exercise, you edited the existing **ToDo** Screen to add the list of To Dos. Then you modified the **ToDoDetail** Screen to make it possible for users to create new and edit existing To Dos.

You have learned how to display a list of records (To Dos) fetched from a Database Entity using an Aggregate, and how to create new records or update existing ones.

Finally, the application module was published to the server and you were able to test the changes you made.

List of Figures

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

| | |
|--|----|
| Figure 1. Manage Dependencies | 4 |
| Figure 2. Add references to all Database Entities | 4 |
| Figure 3. Fetch Data from Database | 5 |
| Figure 4. Add ToDo Entity to an Aggregate | 5 |
| Figure 5. Add Filter to Aggregate | 5 |
| Figure 6. Filter Condition Expression Editor | 6 |
| Figure 7. Drag If Widget | 7 |
| Figure 8. Enclose the If Widget in a Container | 7 |
| Figure 9. Properties area of a Container | 7 |
| Figure 10. Change from the Properties to the Styles Editor area | 7 |
| Figure 11. Styles Editor area | 8 |
| Figure 12. Align a Container to center | 8 |
| Figure 13. Animate property of If Widget | 9 |
| Figure 14. Drag a Blank Slate Widget | 9 |
| Figure 15. BlankSlate Placeholders | 10 |
| Figure 16. Blank Slate with Icon and error message | 10 |
| Figure 17. Drag and drop If Widget inside False branch | 11 |
| Figure 18. ToDos Screen and Widget Tree | 12 |
| Figure 19. Drag and drop If Widget | 12 |
| Figure 20. Drag List Widget | 13 |
| Figure 21. Widget Tree | 13 |
| Figure 22. Drag Aggregate result attribute into the List Widget | 14 |
| Figure 23. Add Input Parameter | 15 |
| Figure 24. ToDold Input Parameter properties | 15 |
| Figure 25. Create a new Screen Aggregate | 16 |
| Figure 26. Drag ToDold into Aggregate | 16 |
| Figure 27. Add an If Widget to the Screen Title placeholder | 17 |
| Figure 28. Create Title Expression | 17 |
| Figure 29. Drag a Form Widget | 17 |
| Figure 30. Drag and drop the Notes attribute to the Form Widget | 18 |
| Figure 31. Change the Label Text | 18 |
| Figure 32. Drag and Drop a Button Group | 19 |
| Figure 33. Set Variable property of the PriorityGroup Widget | 19 |
| Figure 34. Set 'Low' in the Value property | 19 |
| Figure 35. Drag Run Server Action statement | 20 |
| Figure 36. Add Input Parameter to CreateOrUpdateToDoWrapper Server Action | 20 |
| Figure 37. ToDo Input Parameter | 21 |

| | |
|---|----|
| Figure 38. ToDold Output Parameter | 21 |
| Figure 39. Select the CreateOrUpdateToDo Action | 21 |
| Figure 40. Set the Source parameter to the Input Parameter | 22 |
| Figure 41. ToDold Output Parameter assignment | 22 |
| Figure 42. CreateOrUpdateToDoWrapper Action flow | 23 |
| Figure 43. Set the ToDo Parameter for the CreateOrUpdateToDoWrapper Action | 23 |
| Figure 44. SaveOnClick Success Message | 23 |
| Figure 45. Replace End by a Destination | 24 |
| Figure 46. ListItem1 Widget | 25 |
| Figure 47. OnClick Event of ListItem1 | 25 |
| Figure 48. ToDold parameter | 26 |
| Figure 49. Create the first To Do | 27 |