outsystems

DEVELOPING OUTSYSTEMS MOBILE APPS

Sidebar

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 search sidebar
- Add a Priority filter to the search sidebar
- Add a Category filter to the search sidebar
- Add a ‘Clear Filters’ button
- Implement the logic to refresh the **ToDos** Screen automatically
- Build the Categories Screen
- Link the Categories Screen to the search of To Dos

Table of Contents

Part 1: Create a Search Sidebar

In this part of the exercise, you will create a search sidebar to filter the list of To Dos.

1. Add a Sidebar Pattern to the **ToDos** Screen.

a) Switch to the **Interface** tab and open the **ToDos** Screen.

b) Right-click the **ToDos** Screen and choose 'Add Local Variable'.

c) Rename the Local Variable to 'ShowSearchSidebar', set its **Data Type** to 'Boolean' and the **Default Value** to 'False'.

d) From the Widget toolbar, drag the **Sidebar** Widget and drop it on the end of the Screen.

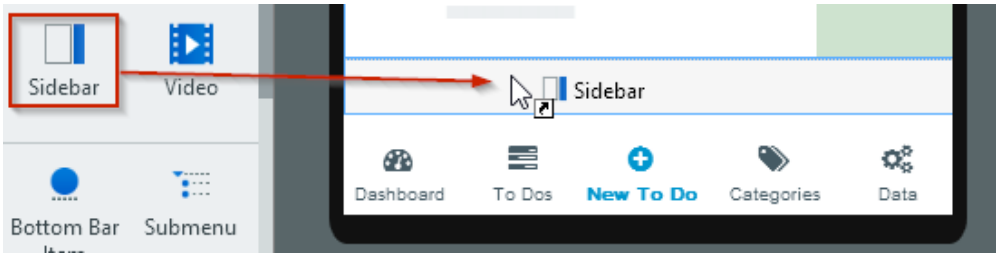

Figure 1. Drag and drop a Sidebar Widget

e) Set the **IsOpen** property of the Sidebar Widget to the **ShowSearchSidebar** Local Variable.

> **NOTE:** The **ShowSearchSidebar** will control when the **Sidebar** opens. By changing its value in any Client Action flow to True / False, the **Sidebar** will be displayed or not. The **Sidebar** can also be opened by swiping from the right edge of the Screen. Similar to opening the Menu on the left side.

2. Create the heading and controls for the **Sidebar**.

a) Drag a **Container** Widget and drop it inside the **Header** placeholder of the **Sidebar**.

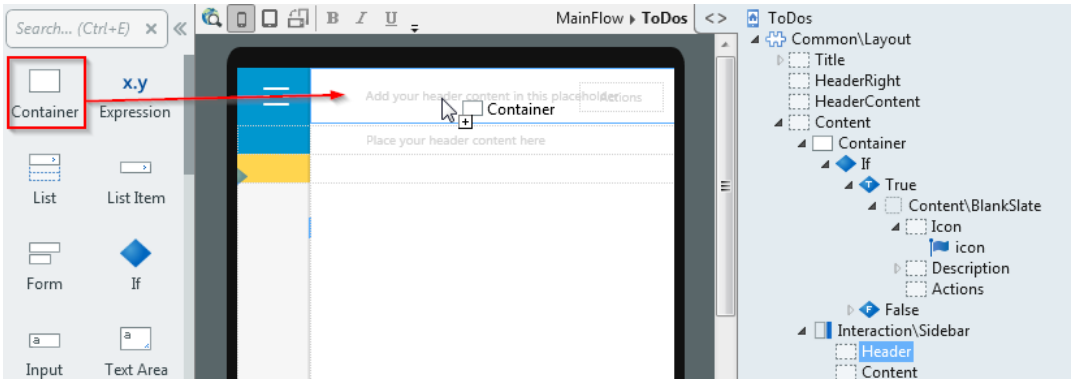

Figure 2. Drag and drop a Container into the Sidebar Header

b) Set the **Style Classes** property to “heading4”.

c) Drag and drop an **Icon** Widget and drop it inside the **Container** created above, then choose the ‘times-circle’ icon.

d) Set the **Size** property to ‘Font size’.

e) Place the cursor just after the ‘times-circle’ icon and type ‘Search Filters’.

f) Select the surrounding **Container**, then right-click it and choose ‘Link to > (New Client Action)’.

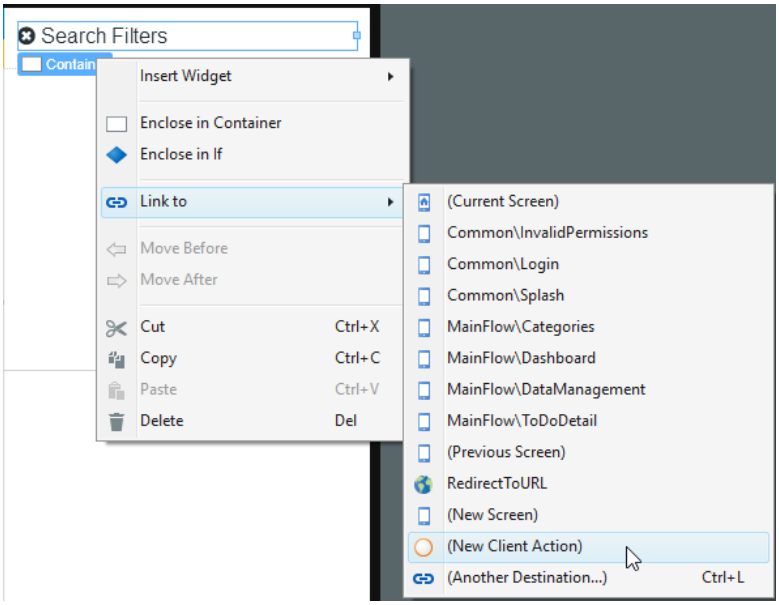

Figure 3. Link to new Client Action

g) Rename the Client Action to ‘ToggleSidebar’.

h) Add an **Input Parameter** to the Client Action.

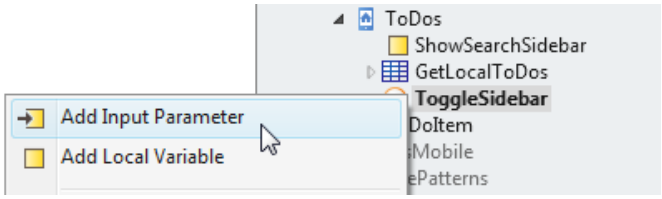

Figure 4. Add Input Parameter to Client Action

i) Set the **Name** of the input parameter to 'Show' and the **Data Type** to 'Boolean'.

j) Drag an **Assign** statement and drop it between the Start and End, then define the following assignment

```
ShowSearchSidebar = Show
```

k) Go back to the **ToDos** Screen, and select the **Link** created before.

l) Set the **Width** property to '(fill parent)', and the **Show** input of the **On Click** event to 'False'.

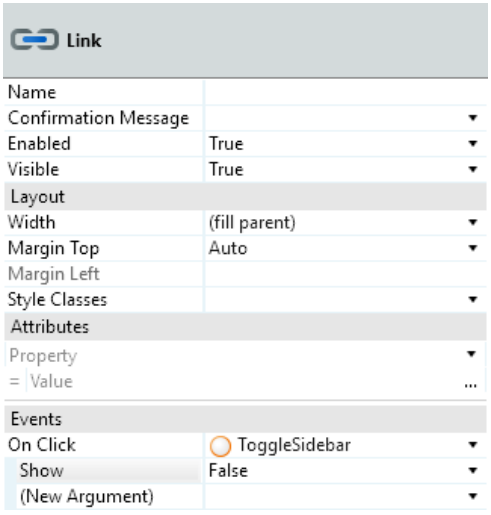

Figure 5. Toggle Sidebar Link properties

3. Create a Search Input Field for the **Sidebar**.

a) Drag a new **Container** and drop it inside the **Content** placeholder.

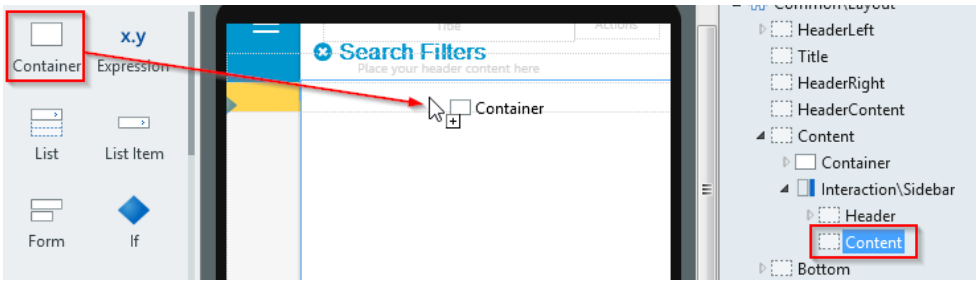

Figure 6. Drag and drop a Container into the Sidebar Content placeholder

b) Drag a **Label** Widget and drop it inside the **Container** created in the previous step.

c) Change the Text inside the **Label** to 'Text and Notes'.

d) Drag a **Search** Widget and drop it inside the same **Container**. Make sure that it was not added inside the **Label** Widget.

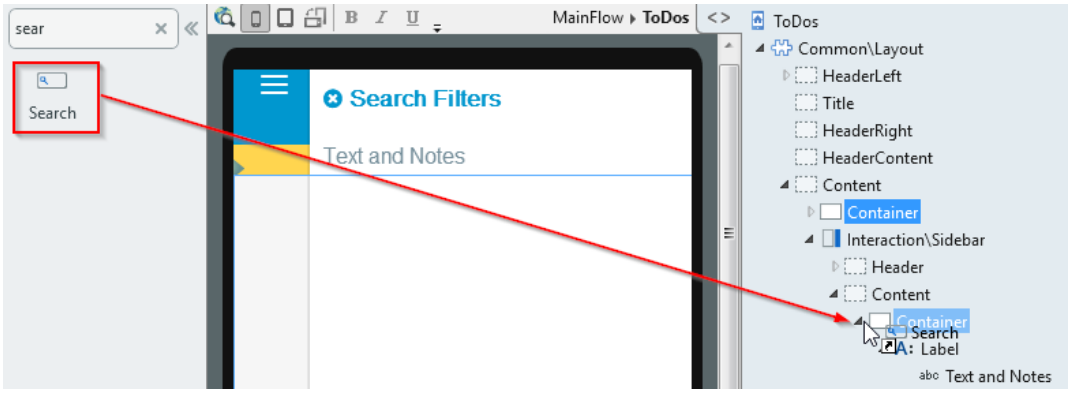

Figure 7. Drag and drop a Search Widget

e) Drag and drop an **Input** Widget inside the **Search** Widget, if there is not any already.

f) Select the **Input** Widget inside the Search Widget, expand the drop down of suggestions for the **Variable** property and choose '(New Local Variable)'.

g) Rename the new Local Variable from **TextVar** to 'SearchKeyword'.

h) Drag a new **Container** and drop it below the **Search** Widget created above.

i) Set the **Align** property of the new **Container** to 'Center'.

j) Drag a **Button** Widget and drop it inside the **Container** created before.

k) Change the text inside the button to 'Search'.

l) Double-click the **Button** to create a new Client Action and associate it to the **On Click** Event.

m) Rename the **SearchOnClick** Client Action to 'Search'.

n) Inside the **Search** Client Action, drag a **Refresh Data** statement and drop it between the Start and End.

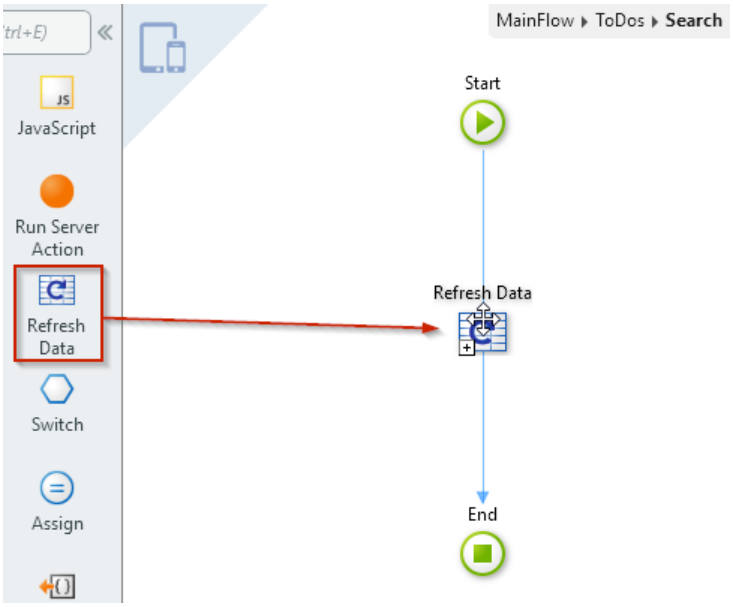

Figure 8. Drag and drop a Refresh Data statement

o) In the **Select Data Source** dialog choose the **GetToDos** Aggregate.

> **NOTE:** The **Refresh Data** statement re-executes the Aggregate. At this point, our Aggregate does not filter the results based on the search keyword. This will be done in the following steps.

p) Drag a **Run Client Action** statement and drop it between the **Refresh Data** and End.

q) In the **Select Action** dialog choose the **ToggleSidebar** Client Action of the **ToDos** Screen.

> **NOTE:** Notice that from the **Search** Client Action you are executing another Client Action (**ToggleSidebar**), from a different Screen.

r) Set the **Show** parameter to 'False'.

4. Add a **Link** to show the **Sidebar**.

a) Return to the **ToDos** Screen and open the Widget Tree, then right-click the **Sidebar** and choose 'Display mode > Hide placeholders'.

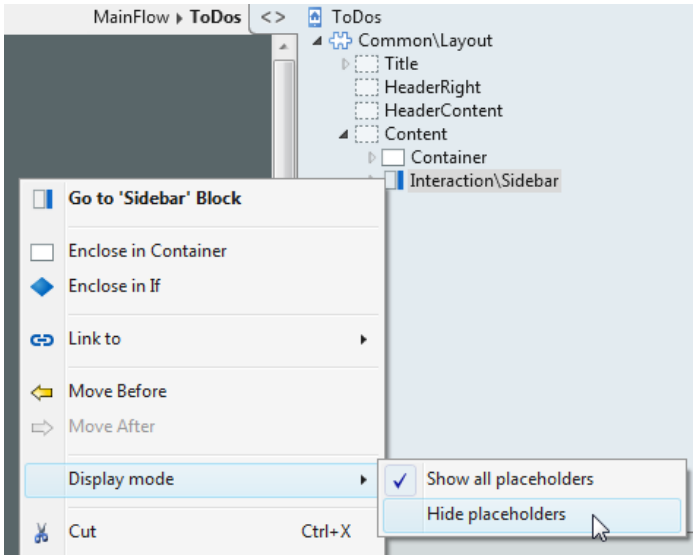

Figure 9. Hide Sidebar placeholders

b) Drag an **Icon** Widget and drop it in the **HeaderRight** placeholder of the Screen then choose the 'search' icon.

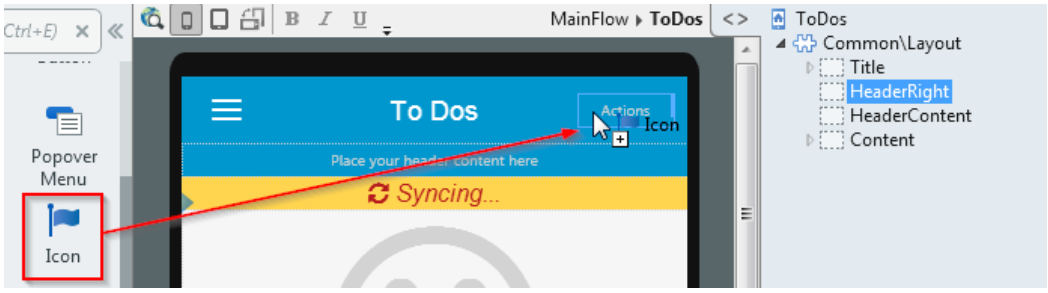

Figure 10. Drag an Icon Widget into the Actions placeholder

c) Right-click the **Icon** and choose 'Link to > ToggleSidebar'.

d) Set the **Show** parameter to 'True'.

5. Modify the Aggregate to use the **SearchKeyword** Local Variable.

a) Open the **GetToDos** Local Aggregate of the **ToDos** Screen.

b) In the **Filters** tab, click the **+Add Filter** and write the following expression, to filter the ToDos by Tile and Notes, using the **SearchKeyword**.

```
LocalToDo.Title like "%" + SearchKeyword + "%" or
LocalToDo.Notes like "%" + SearchKeyword + "%"
```

6. Publish and test the search **Sidebar**.

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

b) In the **ToDos** Screen, either click the **Search** Icon or drag the right edge of the Screen to open the **Sidebar**.

c) In the input field type ‘validation’, then click the **Search** Button.

d) Notice that the list of To Dos was refreshed and shows only the ‘To Do’ with ‘validation’ in the title.

Part 2: Add a Priority filter to the Sidebar

In this part of the exercise, you will add a Priority filter to the search **Sidebar**.

1. Add a **ButtonGroup** to filter To Dos by priority.

a) In the **ToDos** Screen, open the Widget Tree, then right click the **Sidebar** and choose 'Display mode > Show all placeholders'.

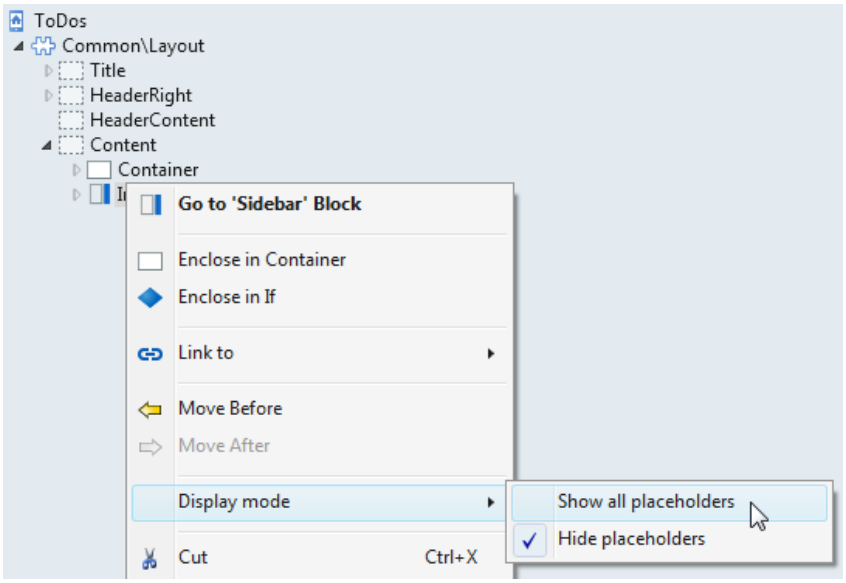

Figure 11. Show Sidebar placeholders

b) Drag a **Container** Widget and drop it between the **Search** Widget and the Container with the **Search** Button.

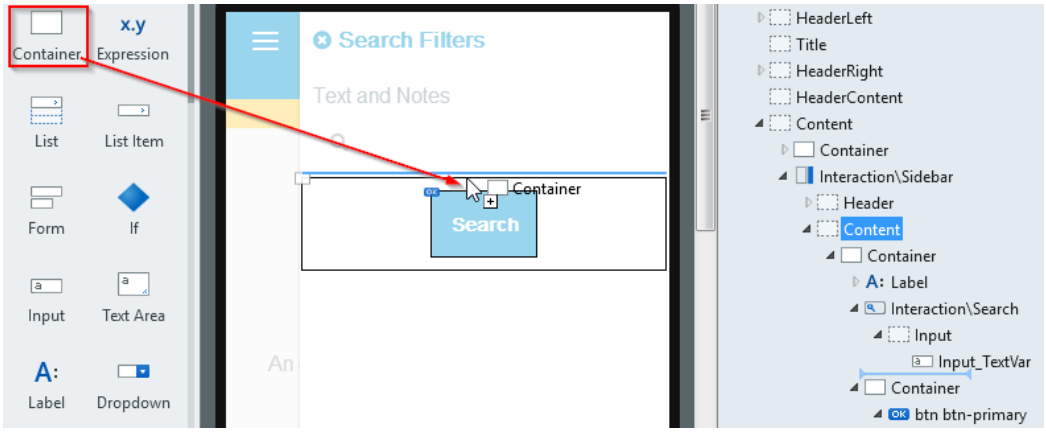

Figure 12. Drag and drop a Container

c) Drag a **Label** Widget and drop it inside the **Container**.

d) Set the **Text** of the Label to 'Priority'.

e) Drag a **Button Group** and drop it inside the Container and below the Label.

f) Right click the **Button Group** and choose 'Add New Item'.

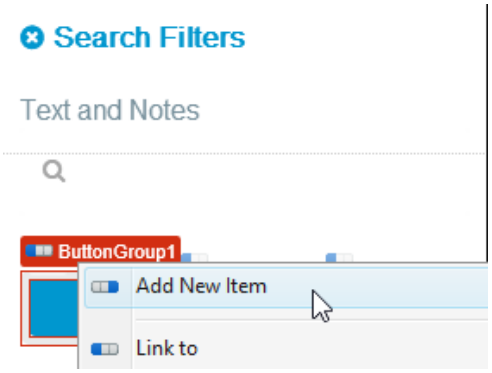

Figure 13. Add New Item to the Button Group

g) With the **ButtonGroup1** Widget select, open the drop down for the **Variable** property and choose '(New Local Variable)'.

h) Change the Local Variable name to 'LocalPriorityId' and make sure the **Data Type** is set to 'LocalPriority Identifier'.

i) Change the **Text** of the first Button of the Group to 'All'.

j) Set the **Value** property to '`NullIdentifier()`'.

k) Change the **Text** of the second Button to 'Low' and set the **Value** property to 'Entities.Priority.Low'.

l) Change the **Text** of the third Button to 'Medium' and set the **Value** property to 'Entities.Priority.Medium'.

m) Change the **Text** of the fourth Button to 'High' and set the **Value** property to 'Entities.Priority.High'.

2. Modify the **GetToDos** to filter To Dos based on the selected priority.

a) Open the **GetToDos** Aggregate of the **ToDos** Screen.

b) Switch to the **Filters** tab and click the **+Add Filter**.

c) In the Expression editor of the filter write the following

```
LocalPriorityId = NullIdentifier() or LocalToDo.PriorityId =
LocalPriorityId
```


> **NOTE:** The previous condition filters the To Dos by their priority. The first part of the condition checks if the variable that holds the choice made by the user, in the Sidebar filter, is NullIdentifier(). This covers the situation where no priority, or the 'All' priority, is chosen. Otherwise, if the condition only considered the second part, no To Do would be returned, when the **LocalPriorityId** variable had NullIdentifier().

d) Click **Done** to close the expression editor.

3. Publish and test the priority filter of the search Sidebar.

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

b) In the **ToDos** list Screen, either click the **Search** Icon or drag the right edge of the Screen to open the **Sidebar**.

c) Switch the priority filter to 'Low', then click the **Search** Button.

d) Notice that the list of To Dos was refreshed and shows only To Dos with 'Low' priority.

Part 3: Add a Category filter to the Sidebar.

In this part of the exercise, you will add a Category filter to the search **Sidebar**.

1. Add a device setting to enable filtering by categories.

a) Switch to the **Data** tab and locate the **LocalCategory** Entity from the Local Storage.

b) Right-click the **LocalCategory** Entity and select 'Add Entity Attribute'.

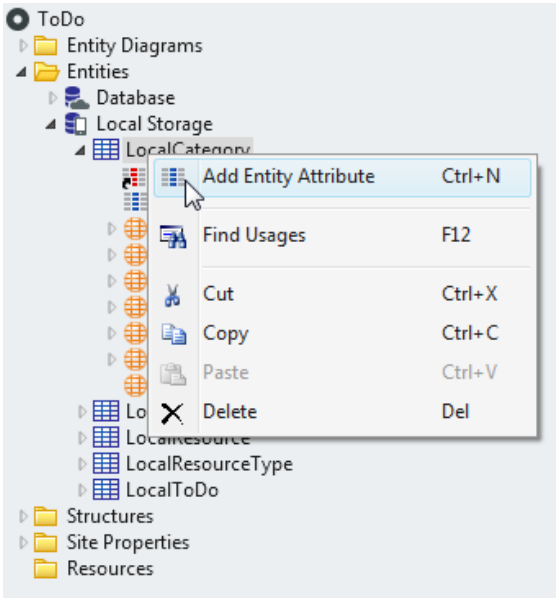

Figure 14. Add an attribute to a Local Entity

c) Set the **Name** of the new attribute to 'IsSelected' and make sure that the **Data Type** is 'Boolean'.

d) Set the **Default Value** of the **IsSelected** attribute to 'True'.

> **NOTE:** The **IsSelected** will act as a device setting and be persistent, hence being stored in a Local Entity.
>
> In order to not overwrite the user's settings every time a synchronization between the device and server occurs, the **OfflineDataSync** Client Action must be adapted to keep the existing values.

e) From the **Logic** tab, open the **OfflineDataSync** Client Action.

f) Delete the **CreateOrUpdateAllLocalCategories** statement, and in its place, drop a **For Each** statement.

g) Set the **Record List** property of the **For Each** to 'ServerDataSync.LocalCategories'.

h) On the right side of the **For Each**, drop an **Aggregate** statement, then create the 'Cycle' connector between the **For Each** and the Aggregate.

i) Double-click the Aggregate to open its editor.

j) From the **Data** tab, drag the **LocalCategory** Local Entity and drop it in the editor.

k) In the **Filters** tab add the following filter

```
LocalCategory.Id = ServerDataSync.LocalCategories.Current.Id
```

l) Return to the **OfflineDataSync** Client Action.

m) Next to the **GetLocalCategoryById** Aggregate add an **Assign** statement and define the following assignment

```
ServerDataSync.LocalCategories.Current.IsSelected =
GetLocalCategoryById.List.Current.LocalCategory.IsSelected
```

n) Create the connector from the **GetLocalCategoryById** Aggregate to the **Assign** statement.

o) From the **Data** tab drag the **CreateOrUpdateLocalCategory** Entity Action of the **LocalCategory** Local Entity, then drop it below the Assign statement.

p) Create the connection between the Assign and the Entity Action dropped in the previous step.

q) Set the **Source** parameter of the entity action to 'ServerDataSync.LocalCategories.Current'.

r) Create the connector between the Entity Action and the **For Each**, completing the cycle.

s) The **OfflineDataSync** Client Action should like this

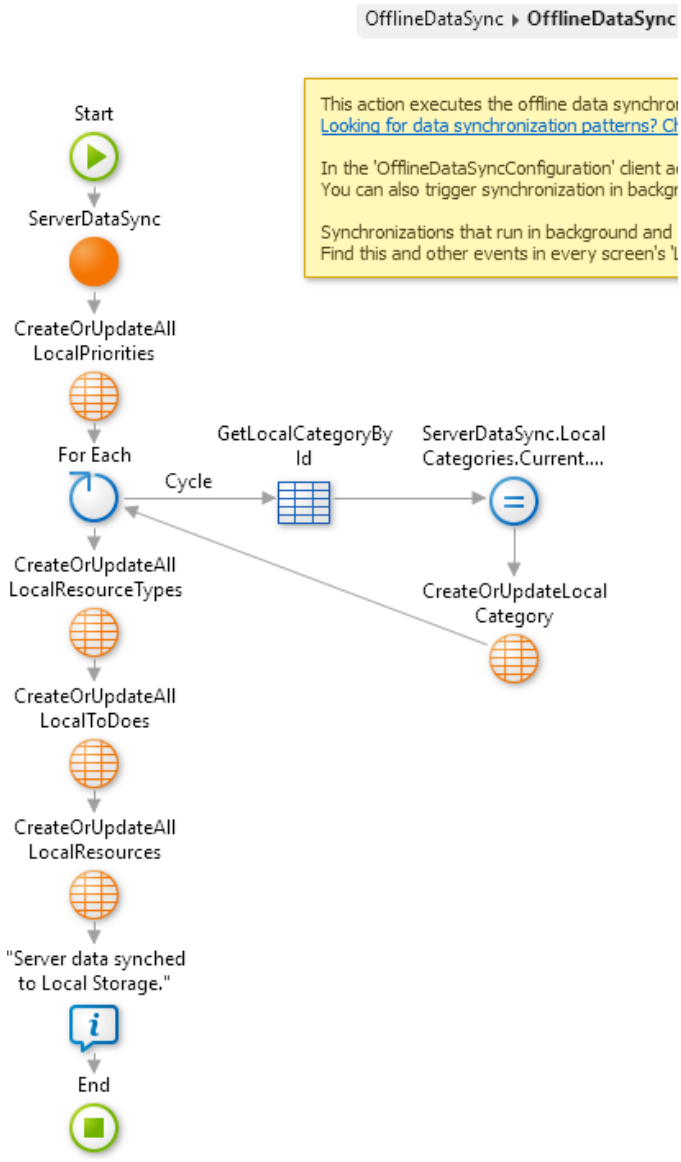

Figure 15. OfflineDataSync Client Action

2. Add the Category filter to the **Sidebar**.

a) In the **Interface** tab, right-click the **ToDos** Screen and choose 'Fetch data from Local Storage'.

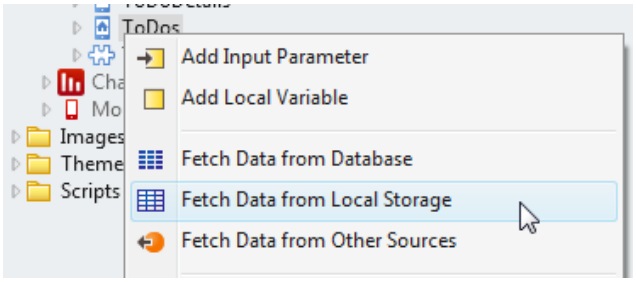

Figure 16. Fetch data from Local Storage

b) From the **Data** tab, drag the **LocalCategory** Local Entity and drop it inside the Aggregate editor.

c) In the **Sorting** tab, add a new sort clause to sort by the 'LocalCategory.Name'.

d) Open the **ToDos** Screen.

e) Drag a **Container** Widget and drop it between the Priority filter and the **Search** Button.

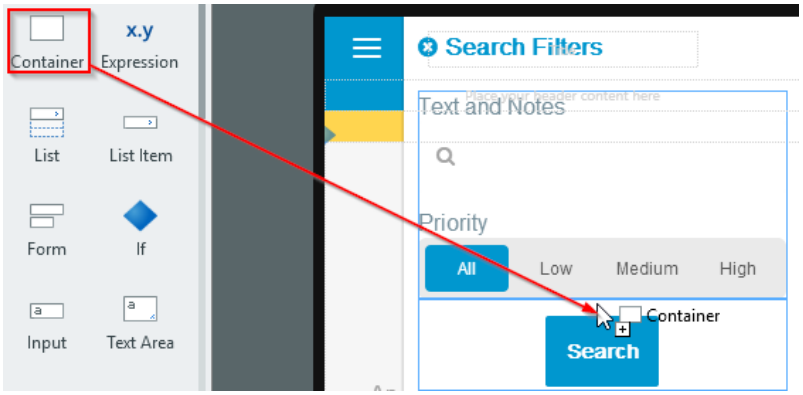

Figure 17. Drag and drop a Container

f) Drag a **Label** Widget and drop it inside the Container created in the previous step.

g) Change the **Text** inside the Label to 'Categories'.

h) Drag a **List** Widget and drop it below the Label but still inside the Container.

i) Set the **Source** property to 'GetLocalCategories.List'.

j) Drag a **Container** Widget and drop it inside the List, then set the **Margin Top** property to '10px'.

k) Drag a **Vertical Align** Widget and drop it inside the **Container** created in the previous step.

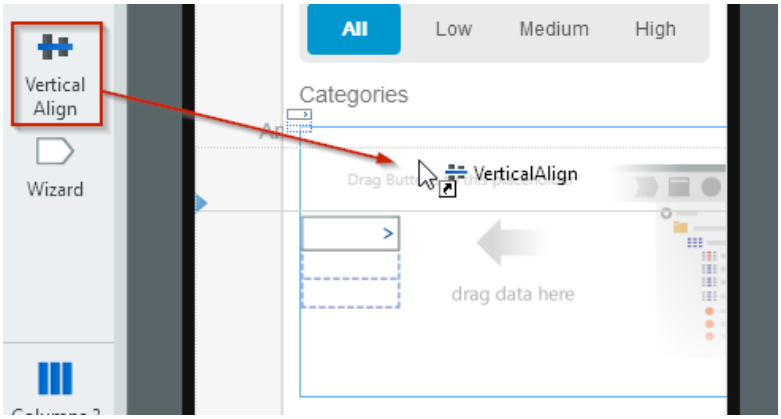

Figure 18. Drag and drop a Vertical Align Widget

l) Drag a **Checkbox** Widget and drop it inside the **VerticalAlign** placeholder of the Vertical Align Widget created in the previous step.

m) Set the **Variable** property of the Checkbox to 'GetLocalCategories.List.Current.LocalCategory.IsSelected'

n) Expand the **GetLocalCategories** Aggregate, then drag the **Name** attribute of the **LocalCategory** Entity, and drop it on the right of the **Checkbox** to create the Name **Expression**.

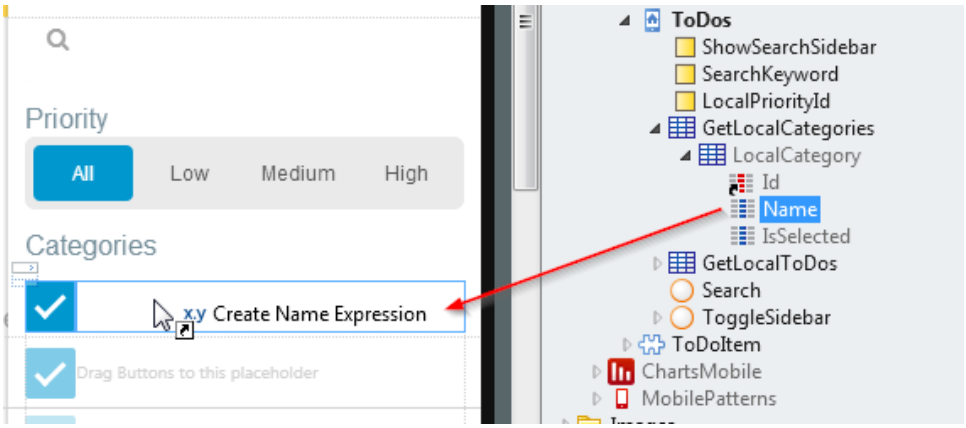

Figure 19. Drag and drop Entity attribute of an Aggregate

o) Select the Checkbox Widget, then on the dropdown of suggestions for the **On Change** Event select '(New Client Action)'.

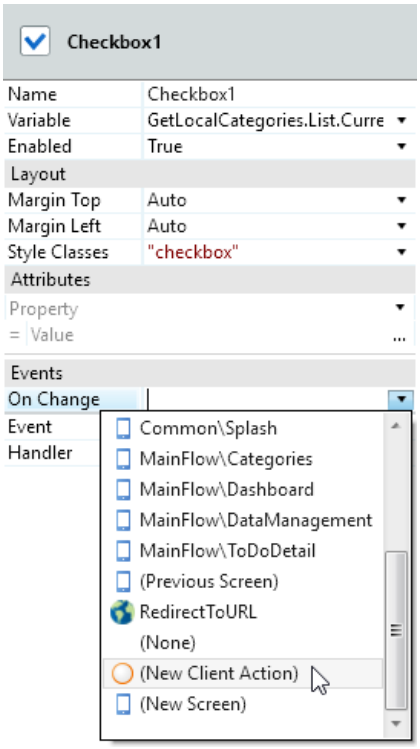

Figure 20. New Client Action for the On Change Event

p) Rename the created Client Action to 'CategorySelectionOnChange'.

q) Drag a **Run Client Action** statement and choose the **UpdateLocalCategory** Local Entity Action.

r) Set the **Source** property expression to

```
GetLocalCategories.List.Current
```

3. Modify the **GetToDos** Aggregate to filter by categories.

a) Open the **GetToDos** Aggregate of the **ToDos** Screen.

b) From the **Data** tab, drag the **LocalCategory** Entity and drop it inside the Aggregate editor. Verify that the **Join** between the **LocalToDo** and **LocalCategory** Entities was defined properly.

Figure 21. GetToDos Aggregate Joins

c) In the **Filters** tab add a filter to only select Categories where the **IsSelected** attribute is True.

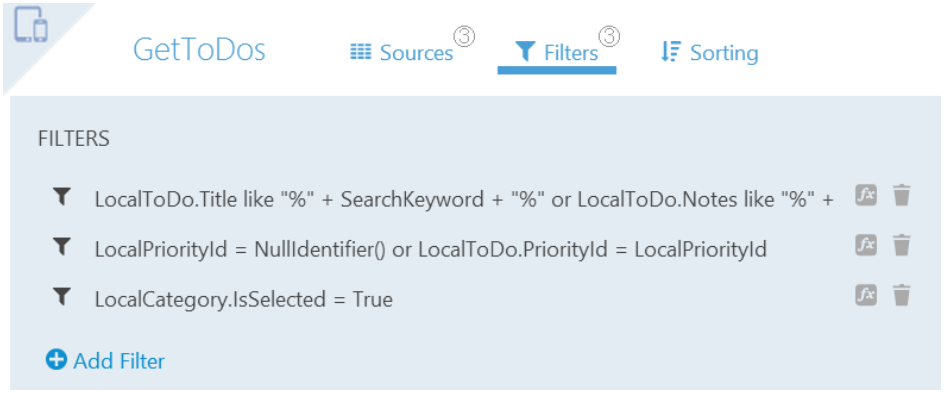

Figure 22. GetToDos Aggregate filters

4. Publish and test the category filter of the search **Sidebar**.

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

b) In the **ToDos** Screen, either click the **Search** icon or drag the right edge of the Screen to open the **Sidebar**.

c) Untick the 'Work' category checkbox, then click the **Search** Button.

d) The list of To Dos should be empty.

e) Add a new To Do with the following details

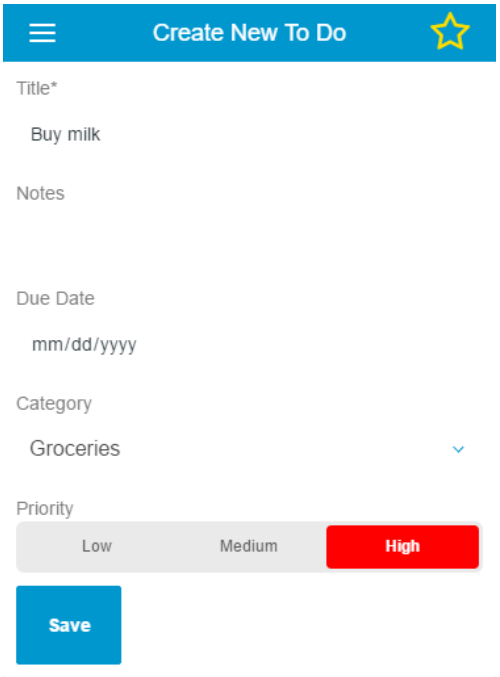

Figure 23. 'Buy milk' To Do

f) Notice that the new To Do appears in the list of To Dos, and the categories selection was kept even after changing between Screens.

g) Re-open the search **Sidebar** and tick the 'Work' category and untick the 'Groceries' category, then click the **Search** Button.

h) You should now see all To Dos except the 'Buy milk' To Do in the **ToDos** list Screen.

Part 4: Clear Filters

In this part of the exercise, you will modify the **ToDos** Screen to show a 'Clear Filters' Button, when there is at least one active filter.

1. Modify the **ToDos** Screen to show a 'Clear Filters' Button when at least one filter is active.

a) In the **Interface** tab, right-click the **ToDos** Screen and choose 'Fetch Data from Local Storage' to create a new Aggregate.

b) Drag the **LocalCategory** Entity from the **Data** tab under the Local Storage, and drop it in the Aggregate editor.

c) Rename the Aggregate to 'GetLocalCategoriesNotSelected'.

d) Add a new Filter in the **Filters** tab to filter only the Local Categories where the 'IsSelected' attribute is False.

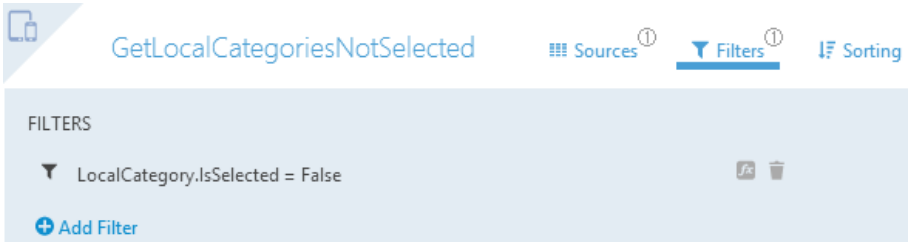

Figure 24. IsSelected attribute filter

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

f) In the Widget Tree, locate the **Sidebar** Widget, then right click it and choose 'Display mode > Hide placeholders'.

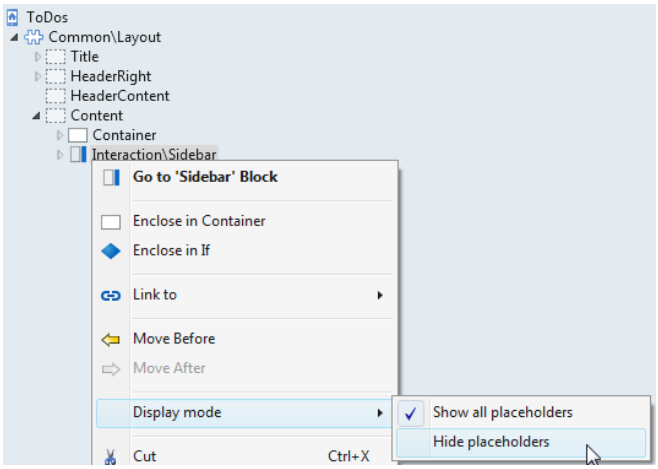

Figure 25. Hide Sidebar placeholders

g) Drag a new **Container** and drop it next to the most inner **If**.

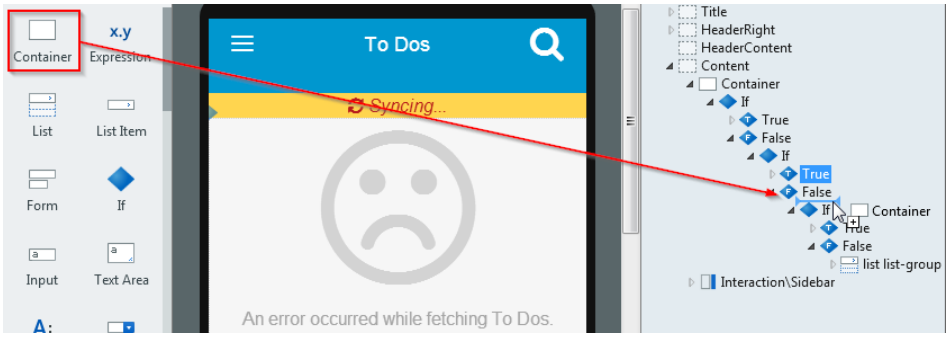

Figure 26. Drag and drop a Container Widget

NOTE: It is possible to drop Widgets directly in the Widget Tree, therefore achieving a more precise placement of Widgets.

h) Set the **Align** property of the Container to 'Center', and the **Visible** property to:

```
SearchKeyword <> "" or LocalPriorityId <> NullIdentifier() or
GetLocalCategoriesNotSelected.Count > 0
```

NOTE: The **Visible** property expression will show the **Container** whenever a search keyword is set, or a priority is selected, or at least one category is not selected (hidden).

i) Set the **Style Classes** property of the **Container** to "padding".

NOTE: The 'padding' Style Class adds a space around the Widgets, inside the **Container** in this case.

j) Drag a **Button** Widget and drop it inside the **padding** Container.

k) Change the **Text** inside the button to 'Clear Filters'.

l) Drag an **Icon** Widget and drop it inside the Button to the left of the text.

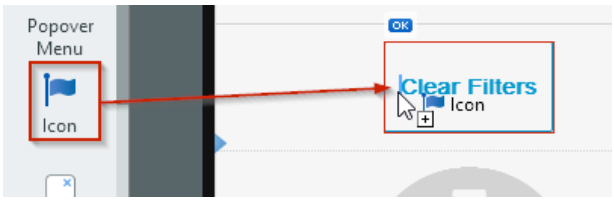

Figure 27. Drag and drop an Icon Widget

m) Chose the 'filter' icon in the **Pick an Icon** dialog.

n) Set the **Size** property of the Icon to 'Font size'.

o) Select the Button and change the **Style Classes** property to "btn btn-small btn-danger".

p) Double-click the **Button** to create a new Client Action and bind it to the **On Click** Event handler.

2. Define the **ClearFiltersOnClick** Client Action to clear all existing filters and show all To Dos available.

a) In the **ClearFiltersOnClick** Client Action flow, add an **Assign** statement and define the following assignments

```
SearchKeyword = ""

LocalPriorityId = NullIdentifier()
```

b) Drag a **For Each** statement and drop it between the Assign and End.

c) Set the **Record List** property to 'GetLocalCategoriesNotSelected.List'.

d) Drag another Assign statement and drop it on the right of the **For Each**, then set the following assignment

```
GetLocalCategoriesNotSelected.List.Current.LocalCategory.IsSe
lected = True
```

e) Create the connector 'Cycle' from the **For Each** to the **Assign**.

f) Drag a **Run Client Action** statement and drop it below the Assign created in the previous step.

g) In the **Select Action** dialog choose the **UpdateLocalCategory** Entity Action.

h) Create the connector from the Assign to the Action, and the connector from the Action back to the For Each.

i) Set the **Source** property of the Entity Action to 'GetLocalCategoriesNotSelected.List.Current'.

j) Drag a **Refresh Data** statement and drop it between the For Each and End, then in the **Select Data Source** choose the **GetLocalCategoriesNotSelected** Aggregate.

> **NOTE:** The Container that surrounds the 'Clear Filters' Button uses the output of the **GetLocalCategoriesNotSelected** Aggregate. Without the **Refresh Data** statement, the Aggregate output list still contains the categories not select, although they have been updated in the Local Storage. Re-executing the Aggregate with the **Refresh Data** statement ensures that the output of the Aggregate is up-to-date with relation to the changes made in the logic.

k) Drag another **Refresh Data** statement to the flow, and drop it between the previous one and the End.

l) In the **Select Data Source** dialog choose the **GetToDos** Aggregate.

m) Repeat the previous two steps for the **GetLocalCategories** Aggregate.

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

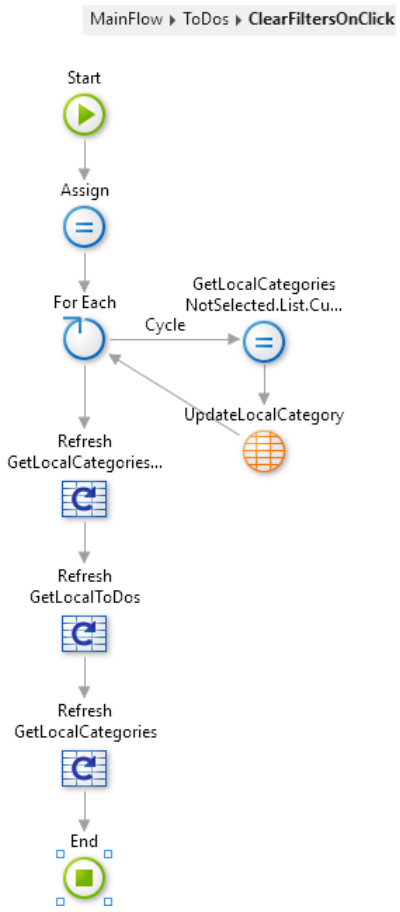

Figure 28. ClearFilterOnClick Client Action

3. Modify the **Search** Client Action to also refresh the **GetLocalCategoriesNotSelected** Aggregate.

a) Open the **Search** Client Action of the **ToDos** Screen.

b) Drag a new **Refresh Data** statement and drop it after the existing **Refresh Data** statement.

c) In the **Select Data Source** dialog, select the **GetLocalCategoriesNotSelected** Aggregate.

4. Publish and test the 'Clear Filters' feature.

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

b) In the **ToDos** Screen, open the search **Sidebar** and define at least one filter then click the 'Search' Button.

c) Notice that the 'Clear Filters' Button is now visible.

d) Click the 'Clear Filters' Button and notice that the list of To Dos is updated, then open the **Sidebar** and verify that all filters were cleared.

Part 5: Refresh To Dos list automatically

In this part of the exercise, you will implement the logic to automatically refresh the **ToDos** Screen when the search filters change.

1. Modify the search filters to automatically refresh the list of To Dos when their values change.

a) Using the Widget Tree, switch the 'Display mode' of the **Sidebar** Widget to 'Show all placeholders'.

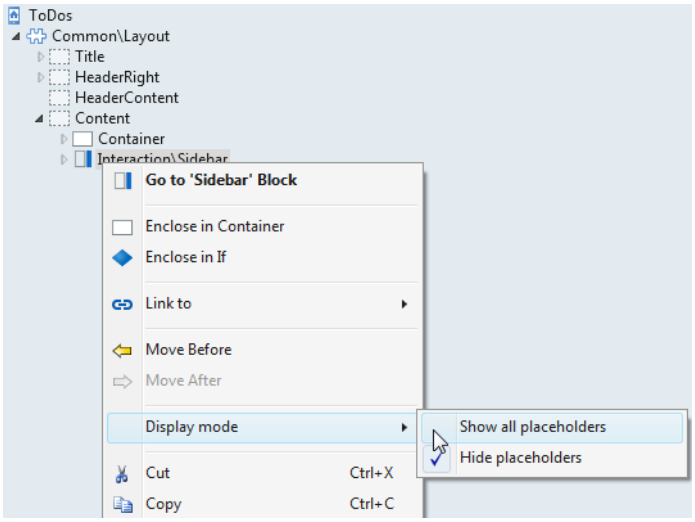

Figure 29. Show all placeholders display mode

b) Select the **Input** Widget that is associated with the **SearchKeyword** Local Variable, then select the **Search** Client Action for the **On Change** Event.

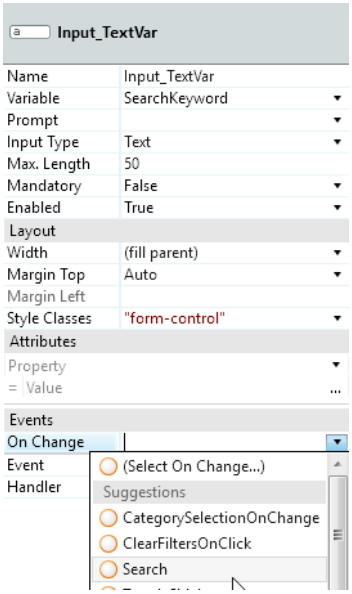

Figure 30. Set the On Change Event

c) Select the **Button Group** Widget, then set the **On Change** Event to the **Search** Client Action.

d) Open the **CategorySelectionOnChange** Client Action, then add a **Run Client Action** to the flow just before the End.

e) In the **Select Action** dialog, choose the **Search** Client Action under the **ToDos** Screen.

f) Open the **Search** Client Action, and delete the **ToggleSidebar** statement, keeping only the two **Refresh Data** statements.

g) Return to the **ToDos** Screen, and select the **Search** Button.

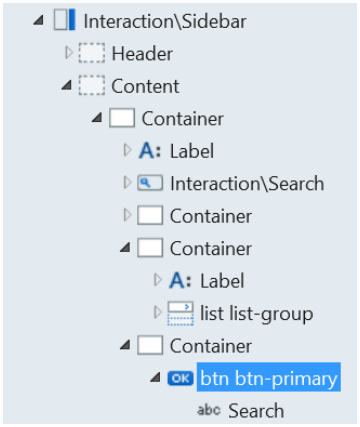

Figure 31. Search Button in the Sidebar

h) Change the **On Click** property from **Search** to the **ToggleSideBar** Client Action, and set the **Show** parameter to 'False'.

> **NOTE:** Notice that now the **Search** Client Action is executed whenever a filter is changed, therefore the **Search** Button act as a Close **Sidebar** Button.

i) Change the **Text** inside the Button to 'Close Sidebar'.

2. Publish and test the automatic refresh of the list.

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

b) In the **ToDos** Screen, open the **Sidebar** and modify one of the filters.

c) Notice that the list of To Dos is automatically updated when any of the filters changes.

Part 6: Build the Categories Screen

In this part of the exercise, you will create a new Screen to show the list of Categories.

1. Show the list of Categories in the **Categories** Screen.

a) Right-click the **Categories** Screen the choose 'Fetch Data from Local Storage'.

b) From the **Data** tab, drag the **LocalCategory** Entity and drop it inside the Aggregate editor.

c) Open the **Categories** Screen and drag a **List** Widget and drop it in the main area Screen, inside the main **Content** placeholder.

d) Set the **Source** property of the **List** to the Aggregate list 'GetLocalCategories.List'.

e) Expand the **GetLocalCategories** Aggregate under the **Categories** Screen, then drag the **Name** attribute of the **LocalCategory** Entity and drop it inside the List.

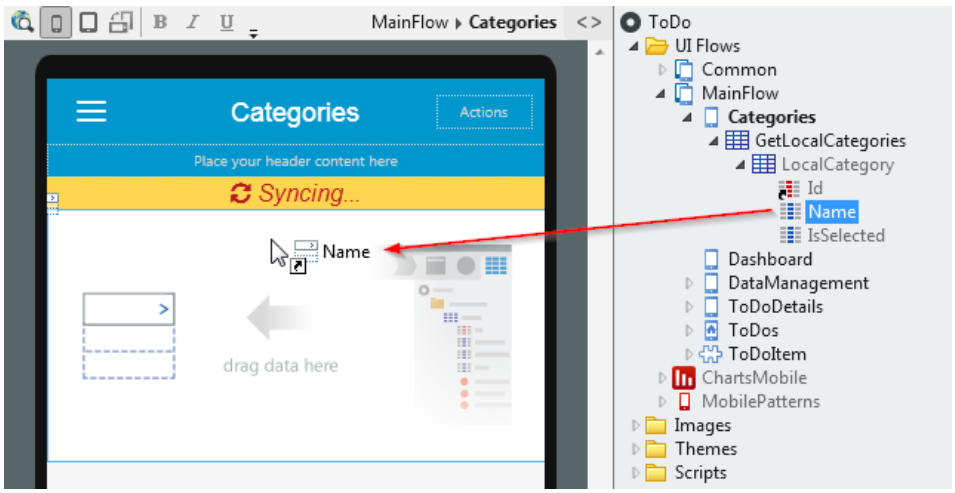

Figure 32. Drag an attribute into a List

f) Select the **List Item** Widget that surrounds the Category Name Expression.

g) Open the drop down of suggestions for the **On Click** Event and choose '(New Client Action)'.

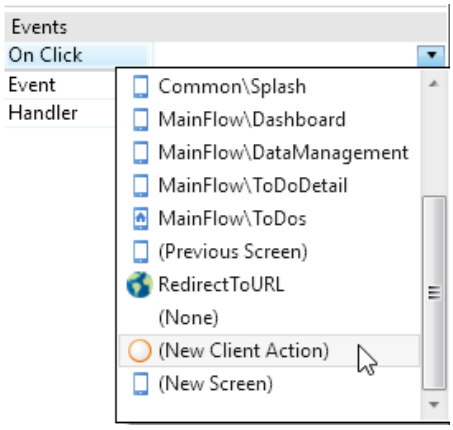

Figure 33. Create new Client Action for the On Click Event

h) Rename the Client Action to 'CategoryOnClick'.

i) Right-click the Client Action and add a new **Input Parameter** named 'LocalCategoryId', then verify that the **Data Type** is set to 'LocalCategory Identifier'.

j) Drag a **For Each** statement and drop it between the Start and End, then define the **Record List** property to 'GetLocalCategories.List'

k) Drag an **If** statement and drop it on the right of the **For Each**, then create the 'Cycle' connector from the **For Each** to the **If** statement.

l) Select the If statement and set the **Condition** to

```
GetLocalCategories.List.Current.LocalCategory.Id =
LocalCategoryId
```

m) Drag an **Assign** statement and drop it on the right side of the If, then create the **True** branch connector from the **If** to the **Assign**.

n) In the **Assign** statement, define the following assignment

```
GetLocalCategories.List.Current.LocalCategory.IsSelected =
True
```

o) Drag a new **Assign** statement and drop it below the If, then create the 'False' branch connector from the **If** to the new **Assign** statement.

p) In the last created **Assign** define the following assignment

```
GetLocalCategories.List.Current.LocalCategory.IsSelected =
False
```


q) Drag a new **Run Client Action** statement and drop it below the last created Assign, then in the **Select Action** dialog choose **UpdateLocalCategory**.

r) Create two new connectors from each Assign to the Action statement.

s) In the properties of the **UpdateLocalCategory** statement, set the **Source** to 'GetLocalCategories.List.Current'.

t) Create a new connector from the **Action** to the **For Each** statement to close the cycle.

u) Drag a **Destination** statement and drop it on top of the End to replace it, then in the **Select Destination** dialog choose the **ToDos** Screen.

v) Your **CategoryOnClick** Client Action should look like this

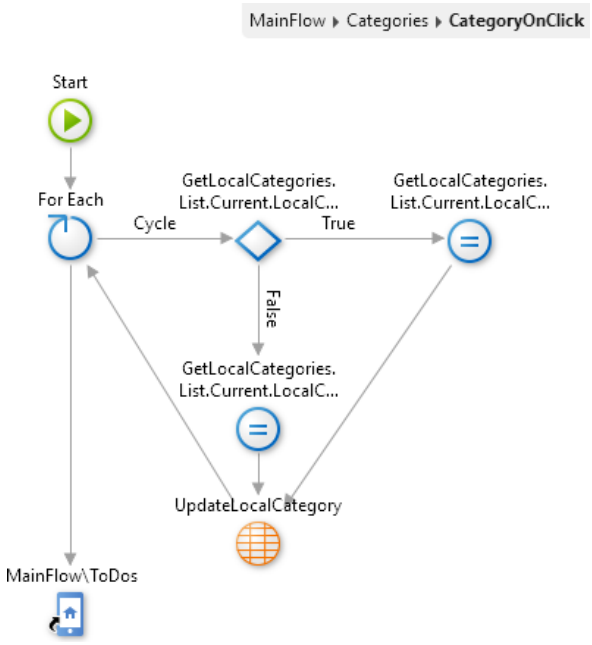

Figure 34. CategoryOnClick Client Action

w) In the **TrueChange** tab you should have the following error

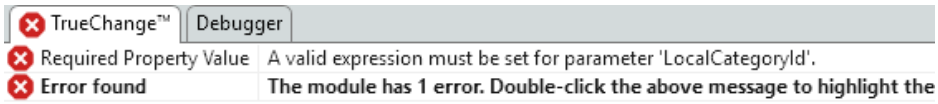

Figure 35. TrueChange tab

x) Double-click the error to move the focus to it, then define the **LocalCategoryId** to 'GetLocalCategories.List.Current.LocalCategory.Id'

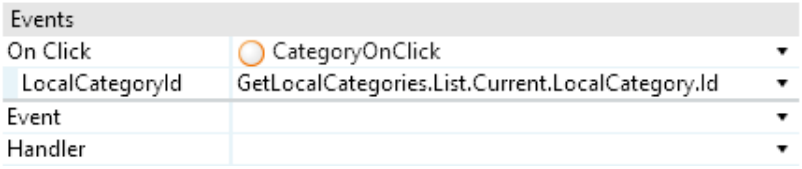

Figure 36. Set LocalCategoryId Input Parameter

2. Publish and test the application.

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

b) Click the **Open in Browser** Button to open the application, or test the application in your device.

c) Navigate to the **Categories** Screen, then click the 'Groceries' category item.

d) Verify that you are redirected to the **ToDos** Screen and that only the To Dos for the 'Groceries' category appear.

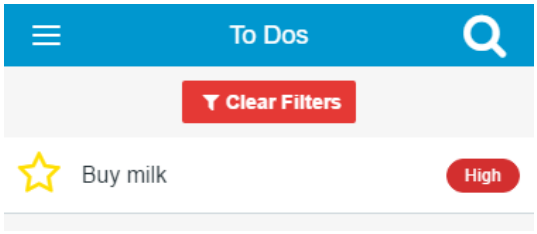

Figure 37. Groceries related To Dos

End of Lab

In this exercise, you created a search **Sidebar** to be able to filter the To Dos in the **ToDos** Screen. You then added Priority and Category filters.

The **ToDos** Screen was modified to show a 'Clear Filters' Button whenever there is at least one filter selected.

In the end, you implemented the logic to automatically refresh the **ToDos** Screen when the values of the search filters change. You also built the **Categories** Screen that allows to set up and filter the existing To Dos by Category.

List of Figures

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

