



Style Sheets Exercise

 Book Hotel 

Title*

Book Hotel


Notes

Family Vacations

Due Date

12/11/2018

Category

Family 

Priority

Low

Medium

High

Save

Table of Contents

Table of Contents	2
Introduction	3
Customizing the Title of Screens	4
Customizing the Button Group	8
Is Starred	11
End of Lab	16

Introduction

In this exercise lab, we will slightly improve the look and feel of the ToDo application, by using CSS and editing the module's theme style sheet.

First, we will make sure that long titles on Todos will not cause overflows on the Title of the ToDoDetail Screen. In the previous lab, we set that the Title of the Screen would be the title of the ToDo, when editing existing Todos. So, we will use some CSS to add some ellipsis at the end of a ToDo with a long title, avoiding it overflowing to the next row.

Second, we will add some color to the ButtonGroup in the ToDoDetail, by giving a background color to each button. The colors will be green for the low priority, orange for the medium priority and red for the high priority.

Finally, we will add the functionality of starring Todos, by clicking on a star on the top right of the ToDoDetail Screen. For that we will use some icons, a container, an If and a Screen Action. Then, we will create a new Style for a gold color and will make sure that the stars use that color.

As a summary, In this specific exercise lab, we will:

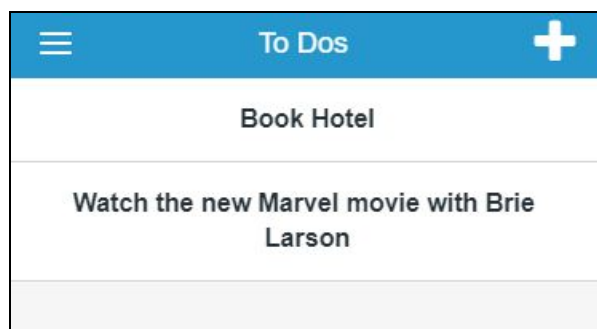
- Customize the title of Screens to prevent overflow
- Customize the background color of each Button in a Button Group Widget
- Create the functionality to star To Dos.
- Apply CSS to change the color of icons

Customizing the Title of Screens

We will start this exercise lab by customizing the **Title** of the **ToDoDetail** Screen. In the last lab, we defined this Screen to have the Title of the ToDo, when editing an existing ToDo. Depending on the ToDo, the title can become quite large, which could result in a strange user experience, when accessing the ToDoDetail Screen to edit it.

For this reason, we will use some CSS to prevent the Screen titles from overflowing.

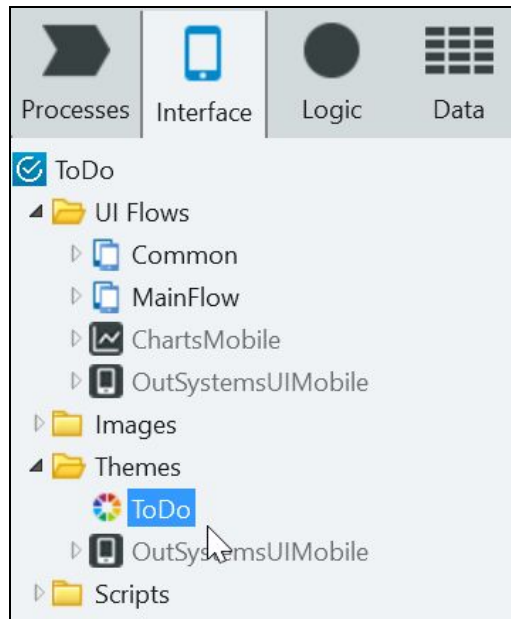
- 1) Open the application in the browser and create a ToDo with a long title, to understand how it appears in the application.
 - a) Click the **Open in Browser** button to open the application.
 - b) Click on the plus icon to create a new ToDo.
 - c) Create a ToDo with a rather long title and Save it to the database.



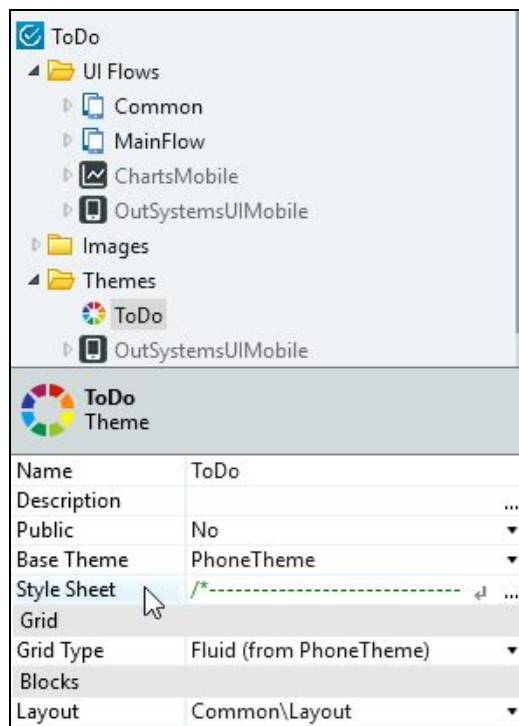
- d) Open the ToDo just created and check how the Title appears in the Screen.



- 2) Open the **module's Theme** Style Sheet and add some CSS to prevent long Screen titles to overflow, just as seen in the previous step.
 - a) In the **Interface** tab of Service Studio, select the module's default Theme, ToDo.



- b) In the properties area, double-click the Style Sheet to open the module's theme Style Sheet.

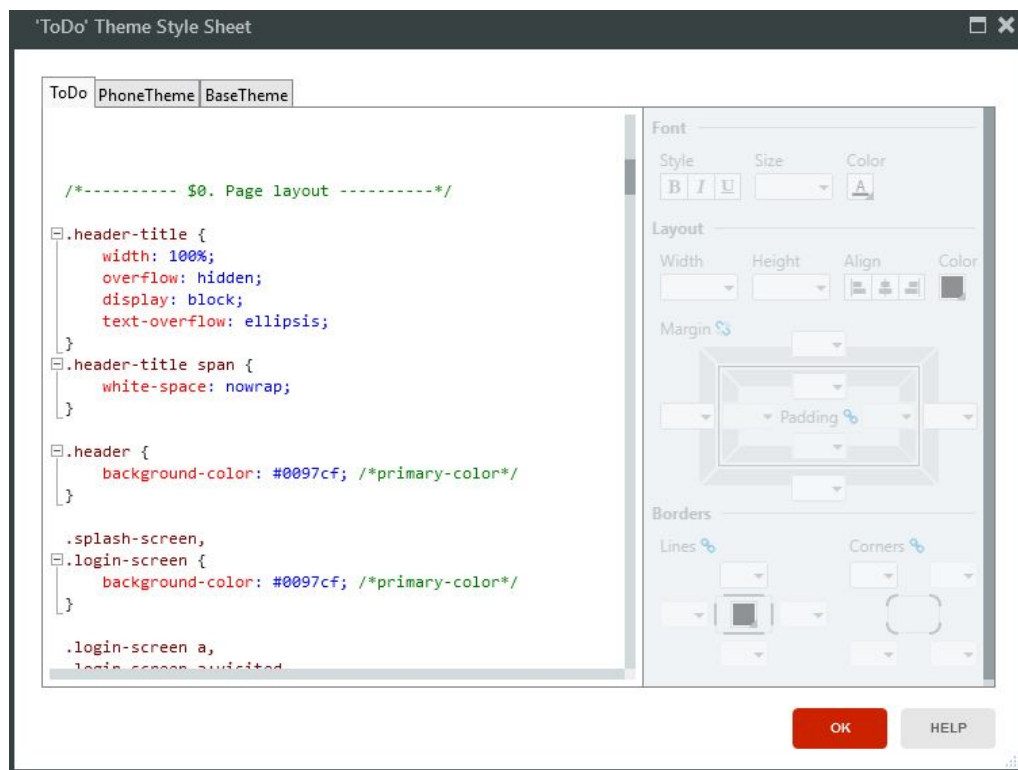


- c) In the **ToDo** tab, add the following CSS code, next to the *\$0. Page Layout* comment, which will display ellipsis when the text overflows the normal width. This will be available in the scope of the module, thus it will be applied in all Screens of the module.

```

.header-title {
    width: 100%;
    overflow: hidden;
    display: block;
    text-overflow: ellipsis;
}
.header-title span {
    white-space: nowrap;
}

```



NOTE: In the above screenshot of the Style Sheet editor, you can find three tabs. The first (**ToDo**) defines the styles for the module's theme. The second (**PhoneTheme**) and third (**BaseTheme**) define the styles for the Base Themes and cannot be edited. The module's theme is actually a copy of the PhoneTheme's Style Sheet, where existing Style Classes can be overridden and new ones can be created. As we add or change styles, Screens and Blocks automatically reflect those changes.

- d) Click **OK** to close the Style Sheet Editor.

- 3) Publish the module and create a ToDo with a large title, to see the added Style Sheet in action.
 - a) Click the **1-Click Publish** button to publish the module to the server.
 - b) Open again the ToDo you just created and verify that the title appears with the ellipsis at the end.

The screenshot shows a mobile application interface for creating a 'ToDo' item. At the top, there is a blue header bar with a hamburger menu icon and the text 'Watch the new Marvel movie with Bri...'. Below the header, the form has several sections: 'Title*' with a text input field containing 'Watch the new Marvel movie with Brie Larson'; 'Notes' with a text input field containing 'Midnight Session'; 'Due Date' with a text input field containing '12/12/2018'; 'Category' with a dropdown menu showing 'Movies to Watch'; and 'Priority' with three radio buttons labeled 'Low', 'Medium', and 'High', where 'High' is selected. At the bottom left, there is a blue 'Save' button.

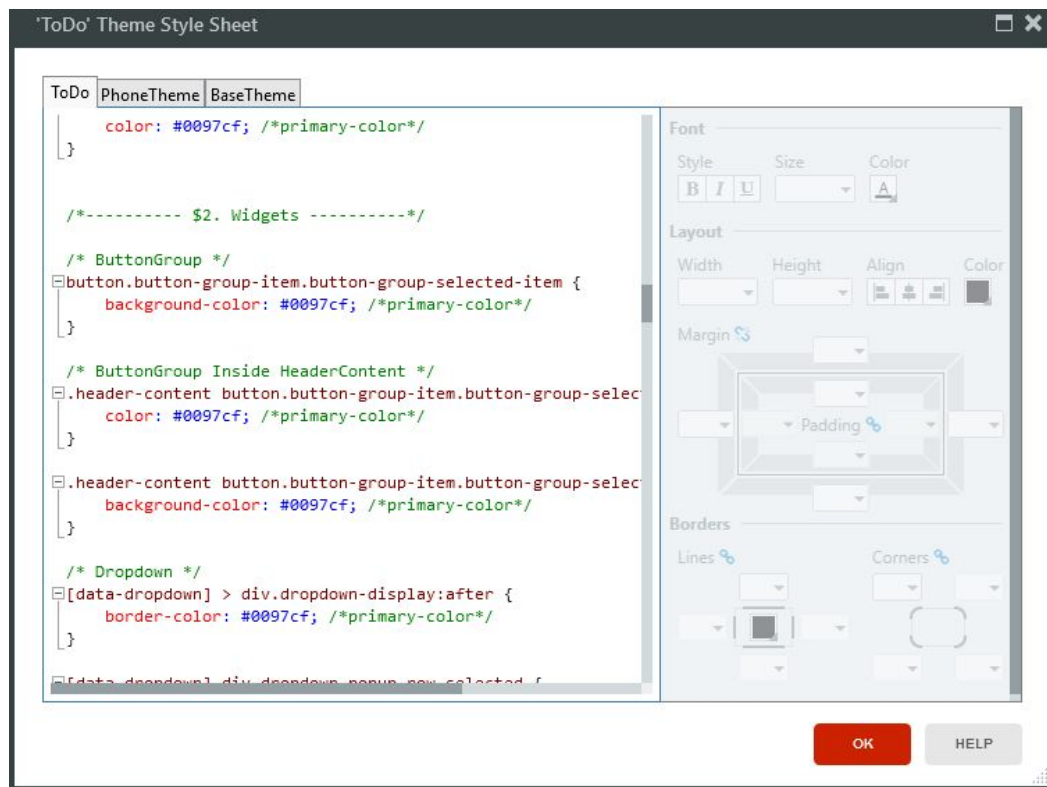
NOTE: The truncated Title depends on the size of the Screen being used. This example may not appear to you truncated, if you are using the browser, but may appear truncated on your phone, for instance.

Customizing the Button Group

In this part of the exercise, we will use CSS to customize the background color of each button in the Priority **Button Group** Widget. We will give some color to the buttons, where the Low would become green, the Medium orange and the High red, when selected.

As in the previous step, this CSS will be added to the module's Style Sheet. Despite that is going to be exclusively used in the ToDoDetail Screen, it is a performance best practice to have the CSS in the module's Style Sheet, to reduce the number of CSS files loaded.

- 1) Add CSS to the **ToDo**'s Style Sheet to give the **green** color to the *Low* ButtonGroupItem, **orange** to the *Medium* ButtonGroupItem and **red** to the *High* ButtonGroupItem.
 - a) Open the ToDo Style Sheet by double-clicking the module's Theme.
 - b) Scroll down in the Style Sheet until you find the **\$2. Widgets** comment.

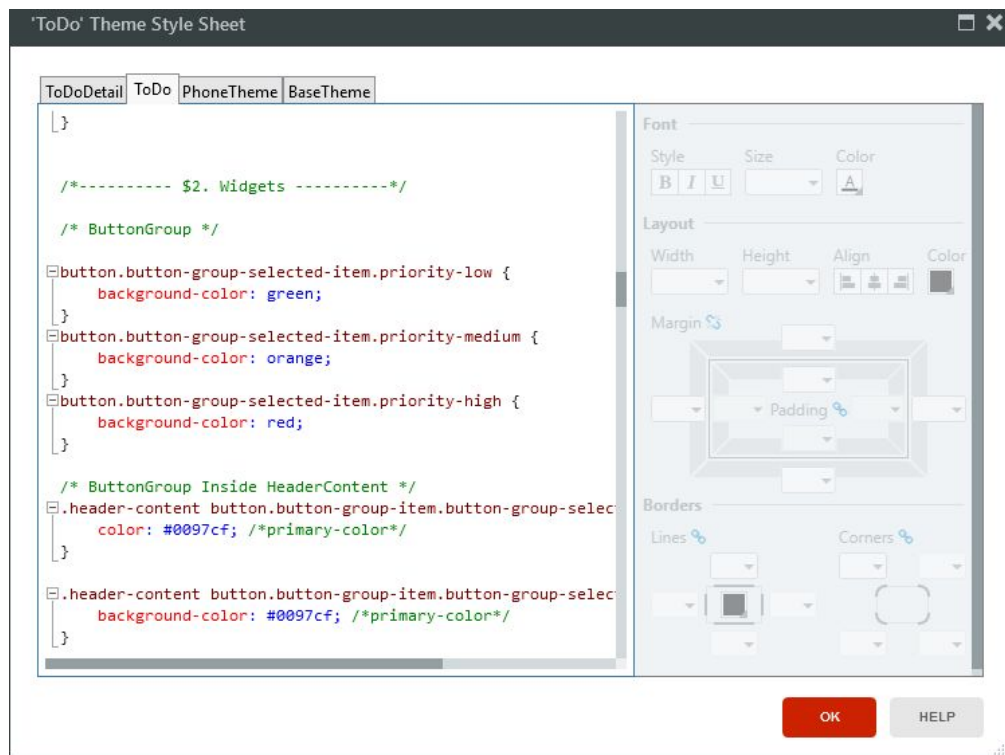


- c) Right after the comment */* ButtonGroup */*, replace the current Style Class under it, with the following CSS. This CSS defines that when the ButtonGroupItems are selected, their background color turn to green, orange or red, depending on the priority selected.


```

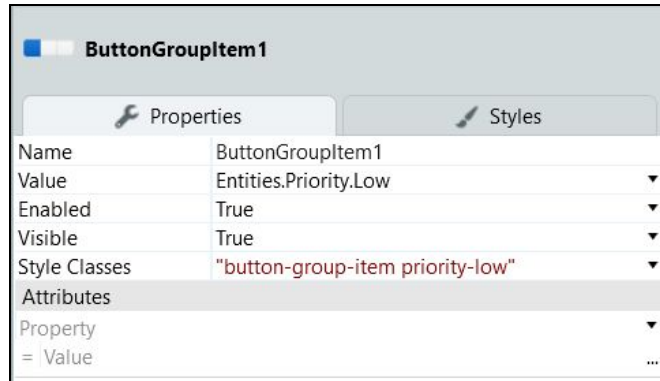
button.button-group-selected-item.priority-low {
    background-color: green;
}
button.button-group-selected-item.priority-medium {
    background-color: orange;
}
button.button-group-selected-item.priority-high {
    background-color: red;
}

```

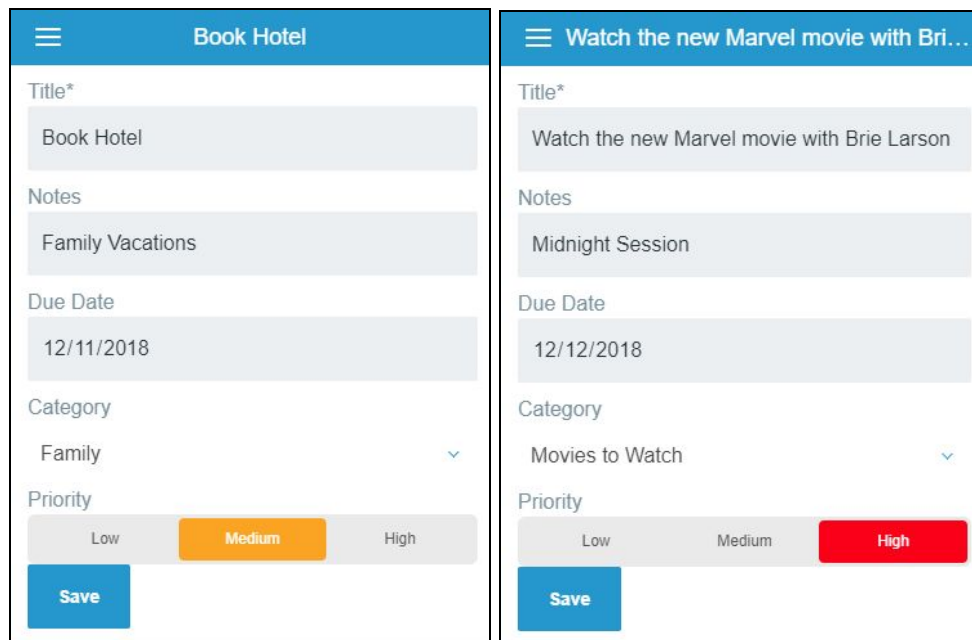


NOTE: The CSS code above acts on any button from a **Button Group** that is selected and has one of the priority style classes: *priority-low*, *priority-medium* or *priority-high*.

- d) Select the *Low* priority **ButtonGroupItem**. It should have the **Name** *ButtonGroupItem1*.
- e) Change the **Style Classes** property and add the *priority-low* style class.



- f) Select the *Medium* priority **ButtonGroupItem**, and add the *priority-medium* style class to the **Style Classes** property.
 - g) Select the *High* priority **ButtonGroupItem**, and add the *priority-high* style class to the **Style Classes** property.
- 2) Publish and see the added Priority colors in action.
- 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.
 - c) Click a To Do from the list of To Dos to open the detail Screen.
 - d) Test the three priority buttons and see that, when selected, each ButtonGroupItem has the color defined in the Style Classes.



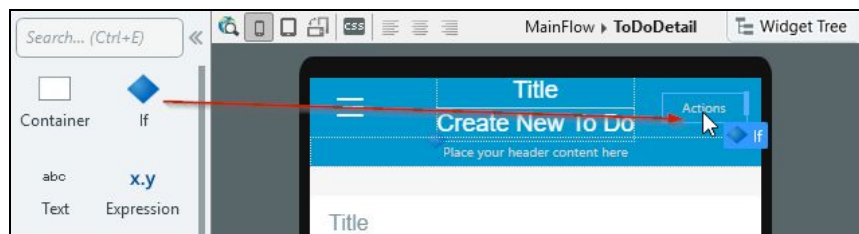
Is Starred

Until now, it is not possible to star Todos, despite the `ToDo` Entity has the `IsStarred` attribute. So, in the `ToDoDetail` Screen, we will add a clickable star, for the user to click on in it to star the `ToDo`.

Then, we will add a `Style` to make the star gold, to highlight better the functionality.

- 1) Add two star icons inside the `Actions` placeholder of the `ToDoDetail` Screen, one filled and one hollowed, to show if the `ToDo` is starred or not. These icons should be placed inside an `If`, one in the `True` branch and one in the `False` branch, so that only one appears at a time. The Condition of the `If` should verify if the `ToDo` is starred (filled star) or not (hollowed star).

- a) Switch to the **Interface** tab and open the **ToDoDetail** Screen.
- b) Drag an **If** Widget to the **Actions** placeholder of the Screen.

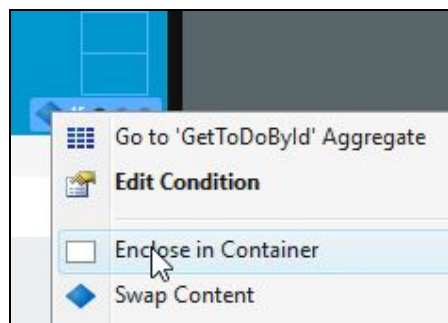


- c) Set the **Condition** of the **If** Widget to

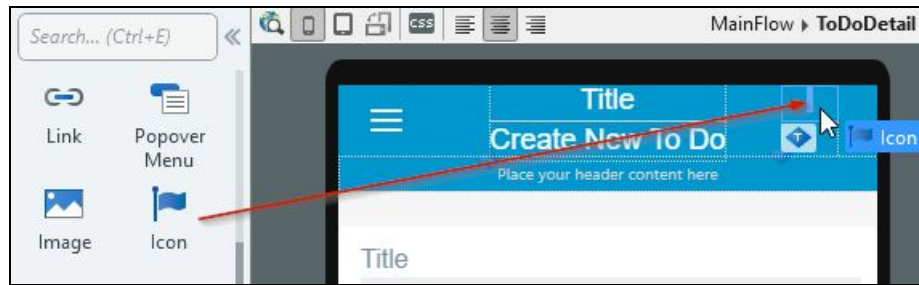
GetToDoById.List.Current.ToDo.IsStarred

This condition verifies if the `ToDo` being displayed on the Screen has the **IsStarred** attribute set to *True* or not.

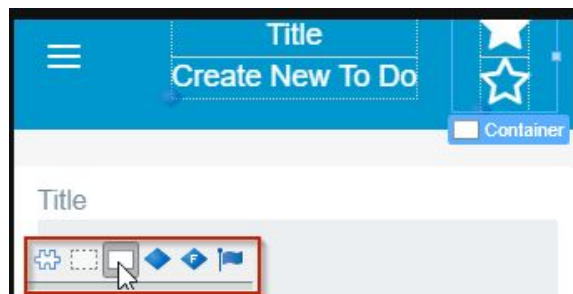
- d) Enclose the **If** Widget in a `Container` and align it to the center of the placeholder.



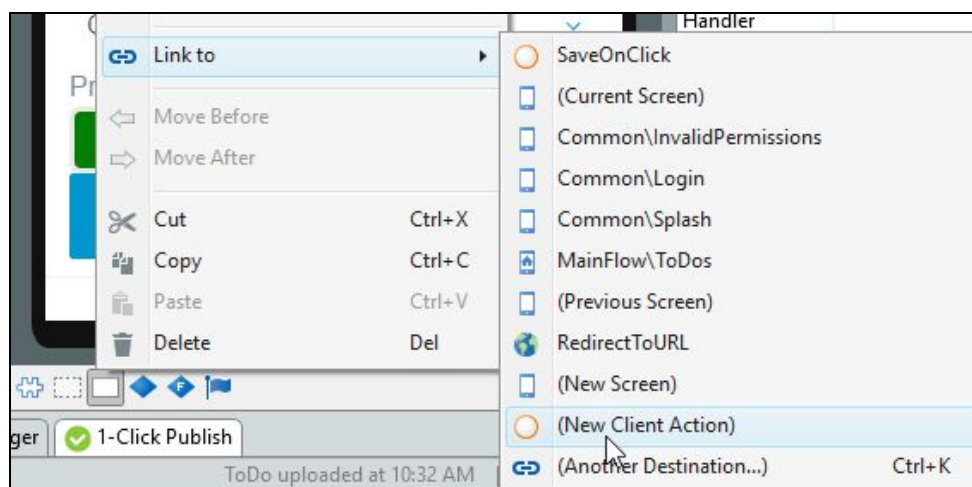
- e) Drag an **Icon** Widget to the **True** branch of the `If`.



- f) In the Pick an Icon dialog choose the filled *star* icon.
 - g) Drag another Icon Widget to the **False** branch of the If, and choose the hollow *star* icon.
- 2) Make the stars clickable, by Linking the surrounding Container to a new Client Action.
- This new Client Action should star, or unstar, a ToDo, by changing its IsStarred attribute accordingly. If the ToDo already exists in the database, then we need to update it with the new information.
- a) Select the **Container** that surrounds the If Widget, using the Widget breadcrumb.



- b) Right-click the **Container** and choose *Link to > (New Client Action)*



- c) Rename the new Client Action from LinkOnClick to *StarOnClick*.

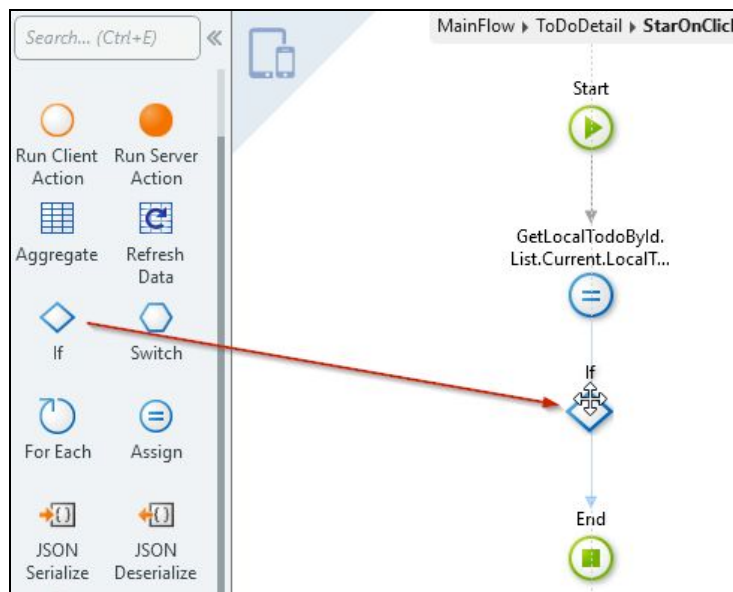
- d) Drag an Assign statement to the Action flow and drop it between the Start and End. Then, define the following assignment:

```
GetToDoById.List.Current.ToDo.IsStarred =
not GetToDoById.List.Current.ToDo.IsStarred
```

This assignment will guarantee that when a user clicks on the filled star, the ToDo will become unstarred, and vice-versa. The assignment changes the value of the **IsStarred** attribute to its opposite (True to False and False to True).

NOTE: By using this method, we can reuse the same Client Action to star and 'unstar' the To Do, with the click of the user.

- e) Drag an If statement and drop it between the Assign statement and the End.



- f) Set the **Condition** property of the If statement to

```
GetToDoById.List.Current.ToDo.Id <> NullIdentifier()
```

This checks if the ToDo exists in the database or not. If it exists, we need to update it.

- g) Drag a **Run Server Action** and drop it on the right of the If statement.
- h) In the **Select Action** dialog choose the **CreateOrUpdateToDo** Client Action.
- i) Create the **True** branch connector between the If and the Action statements.
- j) Select the Run Server Action statement and set the Source parameter to
- ```
GetToDoById.List.Current
```

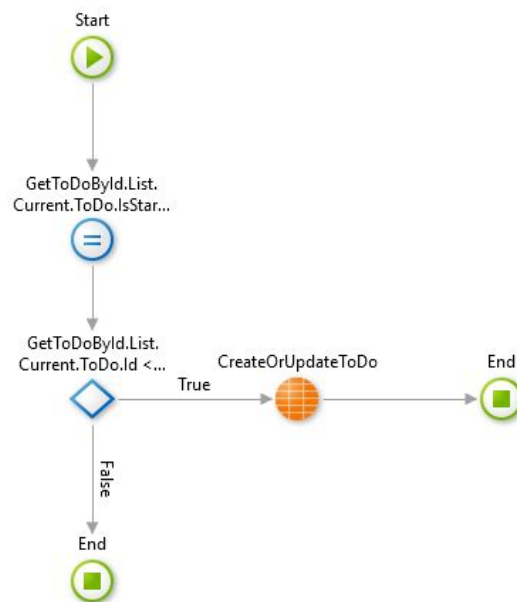
- k) Drag an **End** statement and drop it on the right side of the Action statement.
- l) Create the connector between the Action and the new End statement.

---

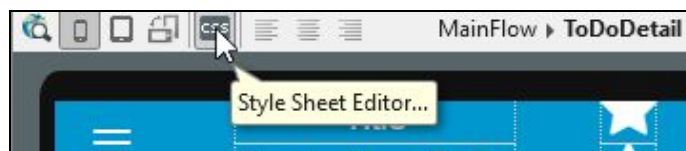
**NOTE:** The **If** statement verifies if the To Do is already created. If it is created (**True** branch), the Action updates the record in the database. In the **False** branch, the To Do does not yet exist in the Database. However, the value is stored in the Aggregate result variable (in the Assign statement), which will be used in the **GetToDoById.List.Current** Action that will create or update the Database record.

---

- m) The **StarOnClick** Client Action should look like this.



- 3) Define a new **Style Class** to the ToDo's Style Sheet, defining a Gold color. Then, apply it to the star icons.
  - a) From the **ToDoDetail** Screen, open the Style Sheet Editor by clicking on the CSS icon and switch to the **ToDo** tab.




---

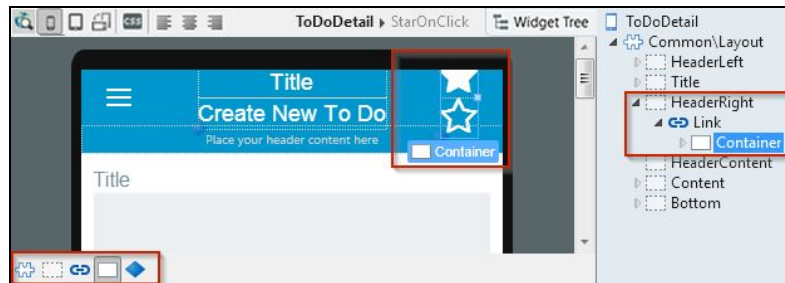
**NOTE:** To edit the Style Sheet of the module's theme, we can open the Style Sheet Editor in the context of any Screen or Block. Another option is to select the Theme in the Interface tab and then double-click the Style Sheet property.

---

- b) Add the following CSS code to the beginning of the existing Style Sheet

```
.gold-color {
 color: #FFDF00;
}
```

- c) Click **Ok** to close the Style Sheet Editor.
- d) Back in the **ToDoDetail** Screen, select the **Container** that surrounds the **If** Widget and the star icons.



- e) Set the **Style Classes** property of the Container to *"gold-color"*.
- 4) Publish the module and confirm that the **IsStarred** feature works properly.
- Click the **1-Click Publish** button to publish the module to the server. Then, open the application in the browser.
  - Select a To Do from the list of To Dos to open the detail Screen. You should see the hollow star icon on the top right of the Screen.
  - Click the hollow star. The star should turn into a filled star.
  - You may have noticed by now that it is not possible to go back to the ToDos Screen, without opening the application again on the browser. This will be fixed on the next lab.
  - Open again the ToDos Screen and open the same ToDo to verify that it is indeed starred yet. If desired, open another ToDo and verify it is not starred.



## End of Lab

In this exercise, we used CSS for several scenarios in our application. First, we applied CSS to avoid the titles of Screens to overflow, in particular for the case of the `ToDoDetail`, which depended on end-user inputs. Then, we customized the `Priority Button Group` Widget, so that each button has a different color, according to the Priority it represents. Finally, we created a Style for the gold color and applied to some icons in the `ToDoDetail` Screen.

Those icons were also created to support the functionality of starring `ToDo`s. The end-user is now able to open a `ToDo`, and click on an icon to star or unstar the `ToDo`.

All of the Styles were added to the `ToDo` (module's theme) Style Sheet, as a best practice.