outsystems

DEVELOPING OUTSYSTEMS MOBILE APPS

# Style Sheets

# 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:

- Customize the title of Screens to prevent overflow
- Customize the background color of each Button in a Button Group Widget
- Apply CSS to change the color of icons

# Table of Contents

## Part 1: Customizing the Title of Screens

In this part of the exercise, you will use CSS to prevent the Screen titles from overflowing.

**1.** Edit the **Layout** Block Style Sheet and add CSS to prevent long Screen titles to overflow.

**a)** Switch to the **Interface** tab and open the **Layout** Block from the **Common** flow.

**b)** Click the **CSS** button in the toolbar to open the Style Sheet Editor.

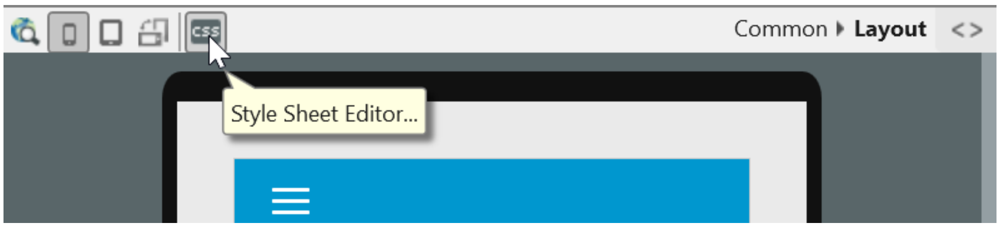
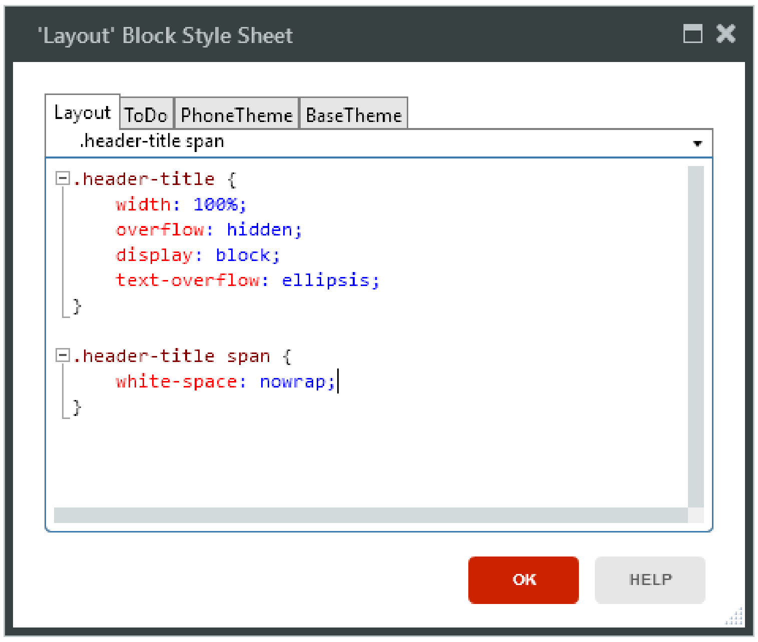


Figure 1. Open Style Sheet Editor

**c)** In the **Layout** tab add the following CSS code

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



```
.header-title {
    width: 100%;
    overflow: hidden;
    display: block;
    text-overflow: ellipsis;
}

.header-title span {
    white-space: nowrap;
}
```

Figure 2. Layout Block Style Sheet Editor

**NOTE:** In the above screenshot of the Style Sheet editor, you can find four tabs. The first (**Layout**) defines the styles for this Block. The second (**ToDo**) defines the styles for the module. The third and fourth tabs contain the styles from the base theme.

As you add or change styles, Screens and Blocks automatically reflect those changes.

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

**2.** Publish and see the added Style Sheet in action.

**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)** Click a To Do from the list of To Dos to open the detail Screen.

**e)** Notice that as you change the Title input field, the Screen title also changes, even without clicking the **Save** Button.

**f)** Type a long title and see that the Screen Title gets truncated.

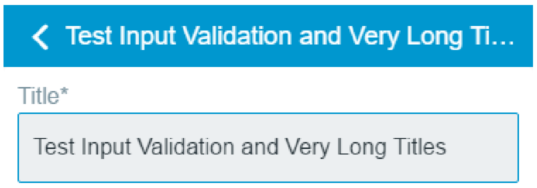

Test Input Validation and Very Long Ti...
Title*
Test Input Validation and Very Long Titles

Figure 3. Truncated Screen title

**g)** Without the CSS code defined above, the Screen would look like this

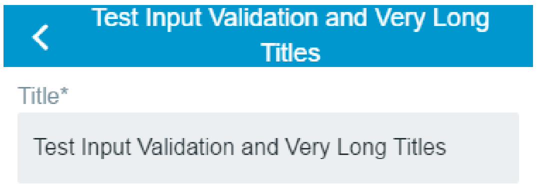

Test Input Validation and Very Long Titles
Title*
Test Input Validation and Very Long Titles

Figure 4. Screen title overflow

**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.

## Part 2: Customizing the Button Group

In this part of the exercise, you will use CSS to customize the background color of each button in the Priority **Button Group** Widget.

**1.** Change the background color of each Button in the Priority **Button Group** input Widget in the **ToDoDetail** Screen.

**a)** From the **Interface** tab open the **ToDoDetail** Screen.

**b)** Click the **CSS** button in the toolbar to open the Style Sheet Editor.

**c)** Ensure that the selected tab is **ToDoDetail**.

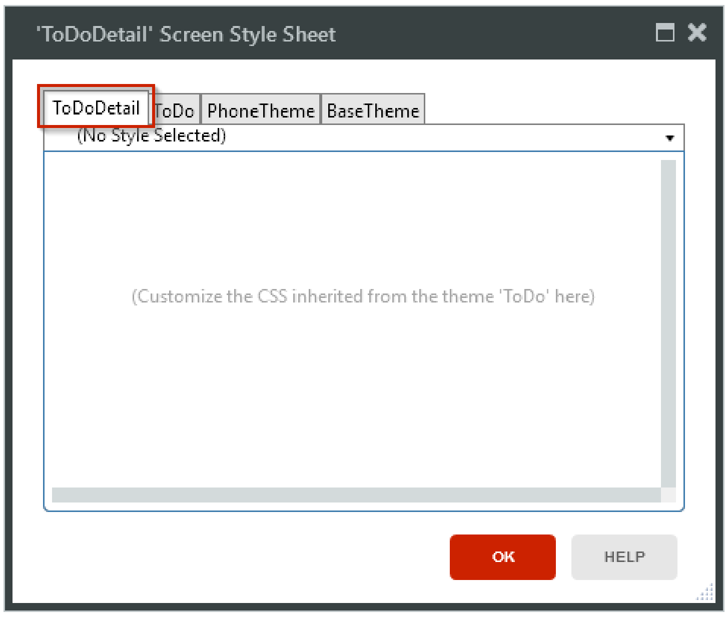
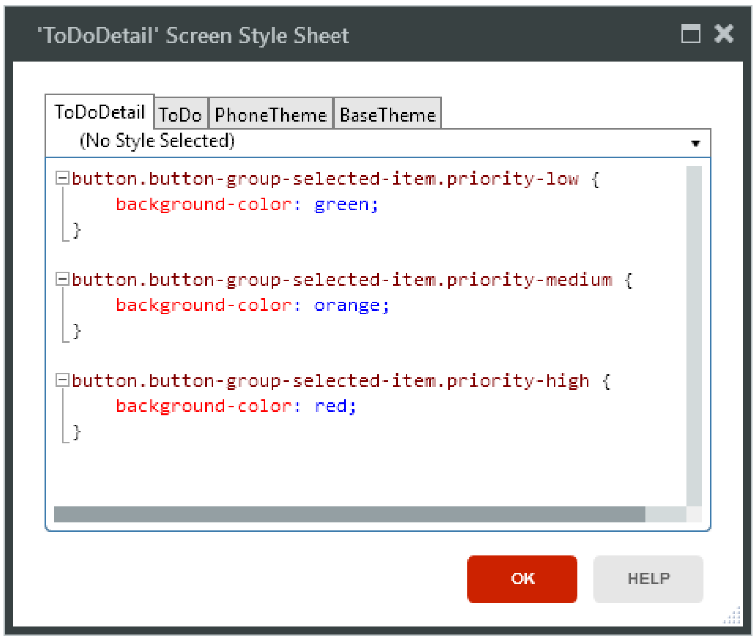

'ToDoDetail' Screen Style Sheet
ToDoDetail
ToDo
PhoneTheme
BaseTheme
(No Style Selected)
(Customize the CSS inherited from the theme 'ToDo' here)
OK
HELP

Figure 5. Style Sheet Editor Screen tab

**d)** Add the following CSS code to the **ToDoDetail** Screen tab

```
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;
}
```



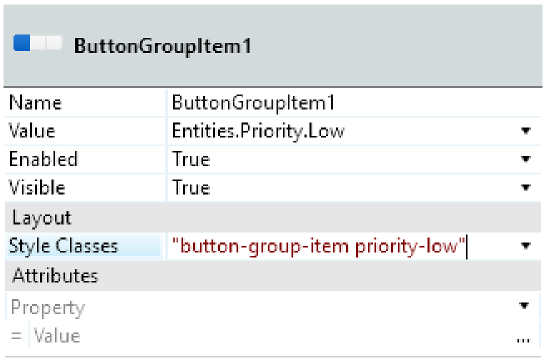
```
button.button-group-selected-item.priority-low {
    background-color: green;
}

button.button-group-selected-item.priority-medium {
    background-color: orange;
}

button.button-group-selected-item.priority-high {
    background-color: red;
}
```

Figure 6. ToDoDetail Style Sheet editor with priority classes

**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.

**e)** Select the 'Low' priority **ButtonGroupItem**. It should have the name 'ButtonGroupItem1'.

**f)** Change the **Style Classes** property and add the 'priority-low' style class.



| | |
|---|---|
| Name | ButtonGroupItem1 |
| Value | Entities.Priority.Low |
| Enabled | True |
| Visible | True |
| Layout | |
| Style Classes | "button-group-item priority-low" |
| Attributes | |
| Property | |
| = Value | ... |

Figure 7. Low Priority Style Class

**g)** Select the 'Medium' priority **ButtonGroupItem**, and add the 'priority-medium' style class to the **Style Classes** property.

**h)** 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.

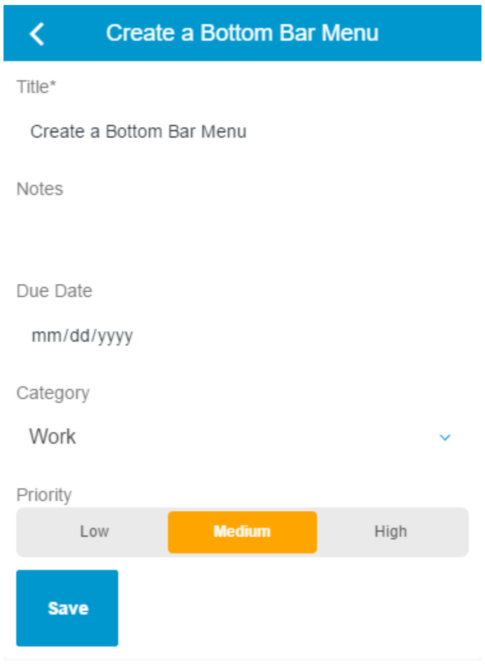
**a)** Click the **1-Click Publish** button to publish the module to the server.

**b)** 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)** Click a To Do from the list of To Dos to open the detail Screen.

**e)** Test the three priority buttons and see that, when selected, each **ButtonGroupItem** has the color defined in the Style Classes.



Create a Bottom Bar Menu

Title*

Create a Bottom Bar Menu

Notes

Due Date

mm/dd/yyyy

Category

Work

Priority

**Figure 8. Priority Button Group Widget with background colors**

# Part 3: Is Starred

In this part of the exercise, you will add a star to the **ToDoDetail** Screen and then you will use CSS to change its color.

**1.** Add a star to the detail Screen to show if the To Do is starred or not.

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

**b)** Drag an **If** Widget to the **HeaderRight** actions placeholder.

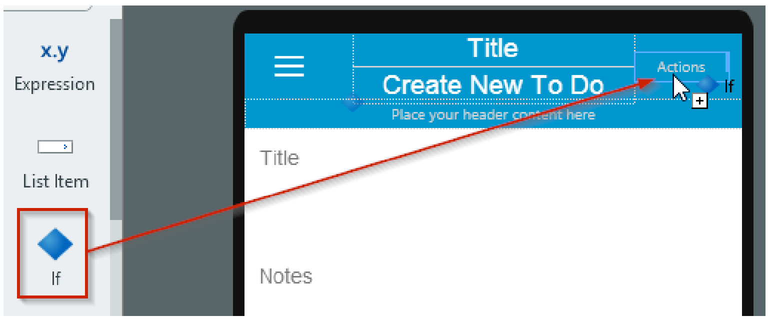


Figure 9. Drag and drop an If Widget into the HeaderRight placeholder

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

```
GetToDoById.List.Current.ToDo.IsStarred
```

**d)** Enclose the **If** Widget in a Container and align it to the center of the Screen, using the Styles Editor area.

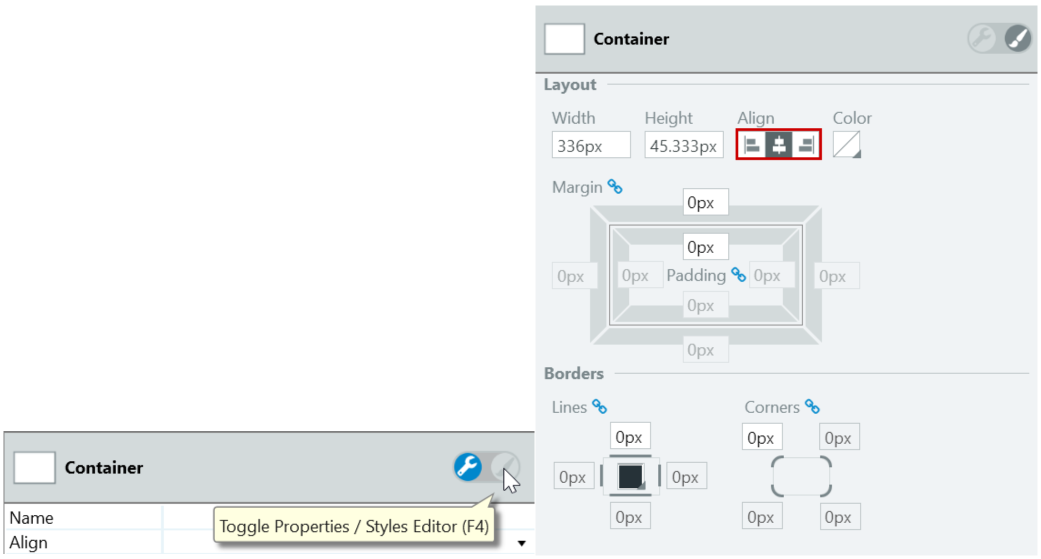


Figure 10. Align Center a Container to center

**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.

**h)** Select the **Container** that surrounds the If Widget, using the Widget breadcrumb.

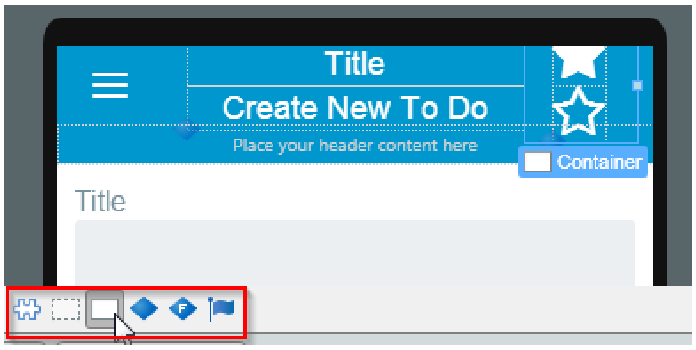


Figure 11. Widget breadcrumb

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

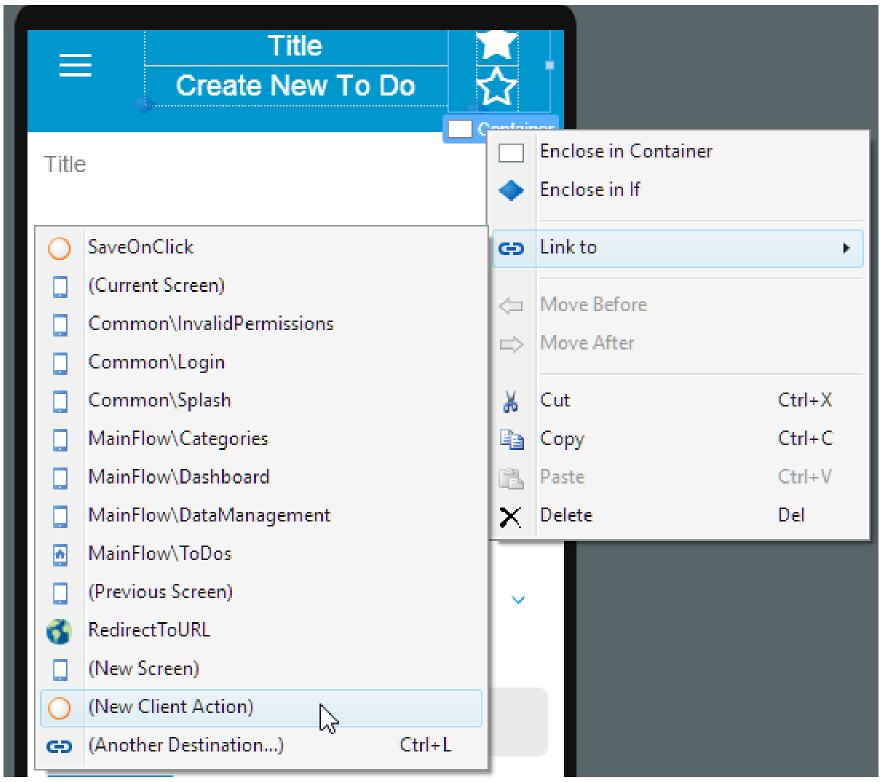


Figure 12. Link to a new Client Action

**j)** Rename the new Client Action from **LinkOnClick** to ‘StarOnClick’.

**k)** 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
```

**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.

**l)** Drag an **If** statement and drop it between the Assign statement and the End.

**m)** Set the **Condition** property of the If statement to

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

**n)** Drag a **Run Server Action** and drop it on the right of the If statement.

**o)** In the **Select Action** dialog choose the **CreateOrUpdateToDoWrapper** Server Action.

**p)** Create the **True** branch connector between the If and the **Run Server Action** statements.

**q)** Select the **Run Server Action** statement and set the **ToDoInput** parameter to 'GetToDoById.List.Current.ToDo'

**r)** Drag an **End** statement and drop it on the right side of the **Run Server Action** statement.

**s)** Create the connector between the **Run Server 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 **IsStarred** attribute of the To Do. In the **False** branch, the To Do does not yet exist in the Database, and therefore it is not possible to update the Database record. However, the value is stored in the Aggregate result variable (in the Assign statement), which will be used in the **CreateOrUpdateToDoWrapper** Action that will create or update the Database record.

**t)** The **StarOnClick** Client Action should look like this.

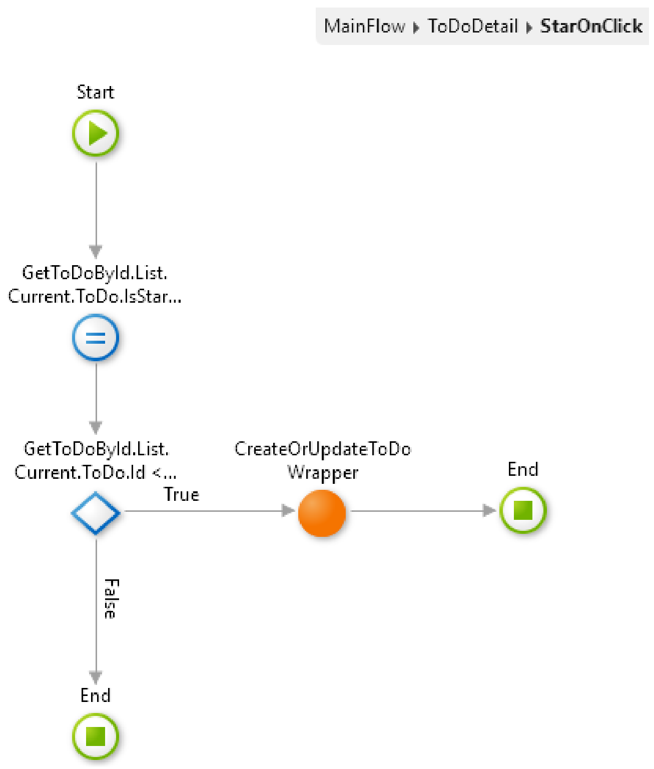


Figure 13. StarOnClick Client Action

**2.** Define a new **Style Class** with a Gold color and apply it to the star icons.

**a)** From the **ToDoDetail** Screen, open the Style Sheet Editor and switch to the **ToDo** tab.

**NOTE:** To edit the Style Sheet of the module's theme you 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.

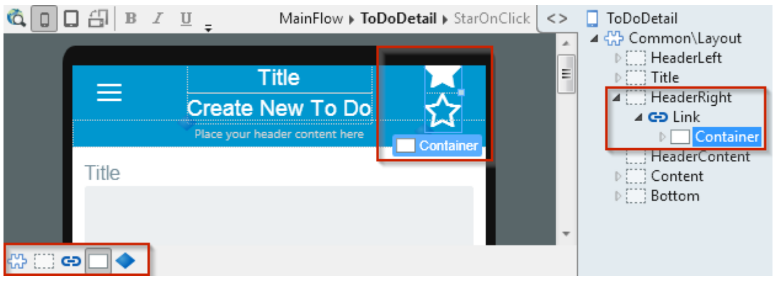


Figure 14 . Select the Container around stars

**e)** Set the **Style Classes** property of the Container to “`gold-color`”.

**3.** Publish and see the Is Starred feature in action.

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

**b)** Select a To Do from the list of To Dos to open the detail Screen. You should see the hollow start icon on the top right of the Screen.

**c)** Click the hollow star. The star should turn into a filled star.

**d)** Return to the **ToDos** Screen and verify that only the To Do changed in the previous step has the filled star.

**e)** Create a new To Do and set it as starred.

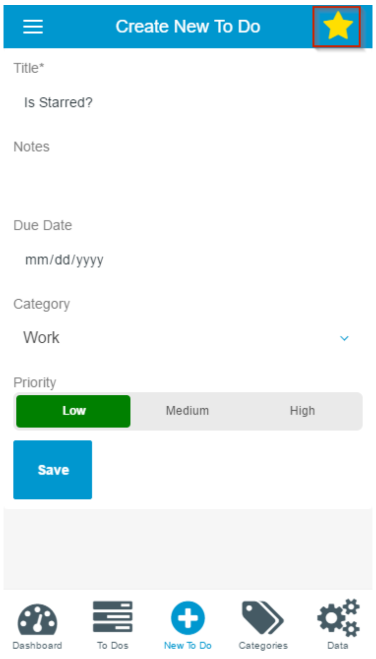


Figure 15. Create 'Is Starred?' To Do

**f)** After saving the new To Do, re-open it from the list of To Dos and verify that is correctly marked as starred.

## End of Lab

In this exercise, you used CSS to customize the titles of the Screens, to prevent them from overflowing. Then you customized the Priority **Button Group** Widget, so that each button has a different color, according to the Priority it represents.

In the end, you added a star to the **ToDoDetail** Screen, to represent if a To Do is starred or not, and to enable to quickly toggle the Is Starred attribute of the To Do. A new Style Class defining the ‘gold’ color was added to change the star icon color.

# List of Figures

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