Web Blocks Exercise

Table of Contents

Introduction

In this Lab, we will introduce the concept of Web Blocks. These Blocks promote UI reusability, since they can be defined once, and then used in multiple Screens (or other Blocks), just by simply dragging and dropping them.

For this purpose, we will add the functionality of rating a movie, by clicking on stars in the MovieDetail Screen. Also, on the same Screen we will display the average rating of the movie, considering all the ratings given at that point to the movie.

To be used on both scenarios, the Web Block will display a certain number of stars, with the specified "rating" amount filled in.

To help us design a rating based on stars, we will create a loop in the Preparation of the Web Block, to determine which stars should be "painted" and which ones are not.

In summary, in this specific exercise lab, we will:

- Create reusable Web Blocks
- Learn how to trigger Events from a Web Block to its parent
- Define the logic to handle Events from Web Blocks

Create a Web Block for Star Ratings

In this part of the exercise, we will create a reusable Web Block, with some UI and functionality that can be included in any Screen or Block (except itself). The Web Block will display a set number of stars, with the specified "rating" amount filled in. In the **MovieDetail** Screen, it will be used to display not only the current user's rating, but also the average rating from all **OSMDb** users.

The Web Block will display a list of consecutive stars that will be displayed horizontally. Some of these stars will be filled (representing the rating given) and some hollowed. For example, a rating of 3 out of 5 would have 3 filled stars and 2 hollow stars.

We will support this particular behavior by converting a *numeric rating* passed as Input Parameter into a list of Booleans. This list will have a *maximum rating* number of elements. The list will have the value TRUE until the numeric rating, and false from the numeric rating to the maximum rating. Here are some examples::
- for a rating of 3 (out of 5), the list would be [TRUE, TRUE, TRUE, FALSE, FALSE]
- for a rating of 1 (out of 3), the list would be [TRUE, FALSE, FALSE]

In this scenario, we will start with a fixed maximum rating of 5, while the numeric rating will be an Input Parameter of the Web Block.

1) Add a Web Block called *StarDisplay* to the **MainFlow**. The Web Block should have an Integer *Rating* as Input Parameter, that represents the rating given. Also, we need a Preparation that implements the logic of transforming a rating into a Boolean List, described above, with a maximum rating of *5*. For that we need a Local Variable of type *Boolean List* and to implement a loop. This loop will assign to the List a TRUE value on the first elements until the *Rating* value, and FALSE from *Rating* to the maximum rating. **Hint:** Use a Local Variable to help control the loop.

    a) In the **Interface** tab, right-click the **MainFlow** and select *Add Web Block*. Set its **Name** to *StarDisplay*.
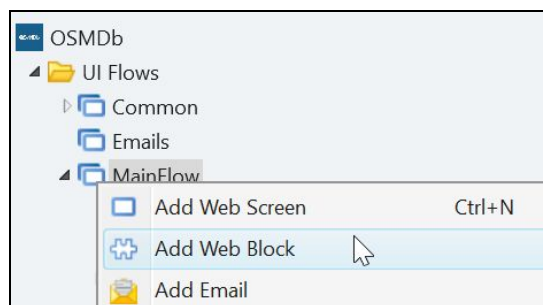
b) Right-click the **StarDisplay** Block and select *Add Input Parameter*. Set its **Name** to *Rating* and **Data Type** to *Integer*.

c) Right-click the **StarDisplay** Block and select *Add Preparation*.

d) In the Preparation we want to convert the Rating to a List of Booleans. So, we need to create a **Local Variable** *RatingList,* with **Data Type** *Boolean List*, in the Web Block.
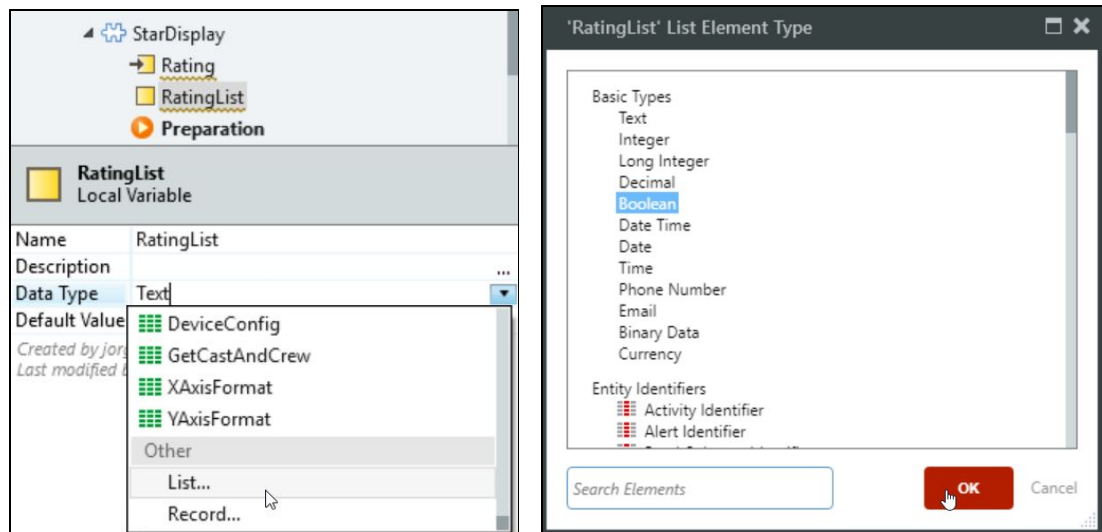
NOTE: A Web Block, as any Screen, can have Input Parameters, Local Variables, Preparation and Screen Actions.

e) Now, we need to implement a loop to help us fill the RatingList Variable with the boolean values. To help control the loop and populate the list, we need a **Local Variable** *Counter*, of **DataType** *Integer*. This will be the "gatekeeper" to make sure the loop stops, and to make sure new values are added to the list.
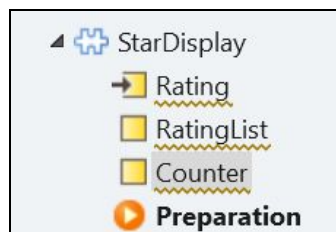
f) Drag and drop an **If** to the Preparation's Action flow. Set its condition to

> *Counter < 5*

Here, the value 5 indicates the Max Rating allowed, meaning that a rating of a movie can go as high as 5. This will be improved in later labs.

g) Drag and drop a **Run Server Action** statement to the right of the If.
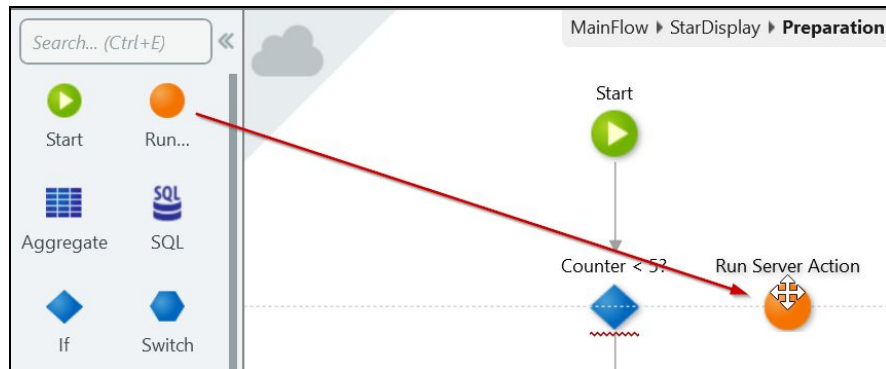
h) In the **Select Action** dialog, type in *ListAppend* in the **Search Actions** filter and double-click **ListAppend** to select it.

i) On the **ListAppend** properties, set **List** to *RatingList* and set the **Element** to the expression *Counter < Rating*.


This step adds an element to the end of the RatingList. This operation only runs if the Counter Variable is smaller than 5 (maximum rating), and adds the result of the Boolean Condition: **True** if the Counter is smaller than the Rating; **False** if the Counter is equal or greater than the Rating. This gives us the desired behavior.

j) Drag a second outbound connector from the If to the **ListAppend**. This connector will have the **True** label.
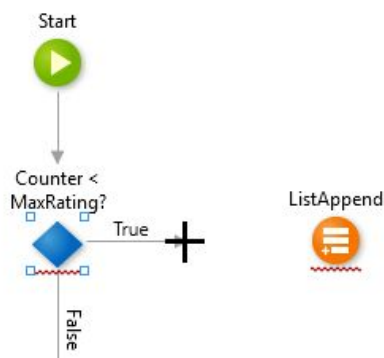
k) Drag and drop an **Assign** under the **ListAppend**. Connect the **ListAppend** to the Assign statement, and the Assign statement back to the If statement, to close the loop.

l) Add the following assignment to the Assign statement

Counter = Counter + 1
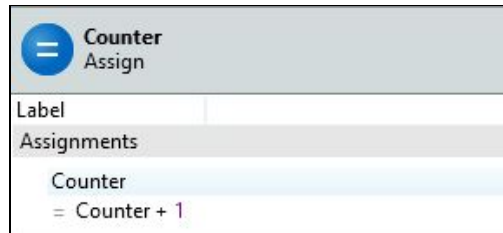
This will guarantee that the Counter variable increases in each iteration of the loop. Otherwise, the If Condition would never fail and it would cause an infinite loop.

m) The Web Block's Preparation should look like this
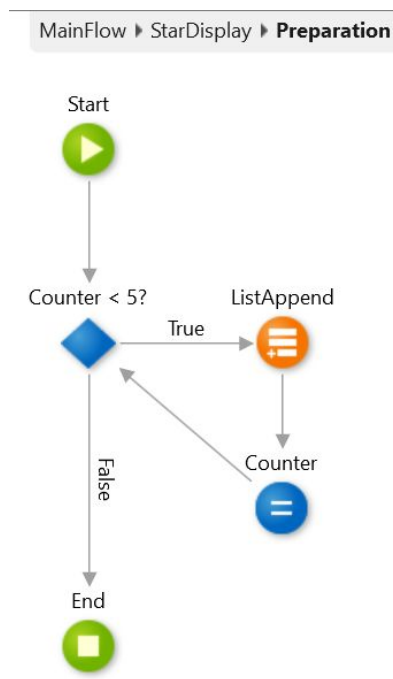
2) The **StarDisplay** Web Block must use the **RatingList** Variable as a source to a **List Records.** The List Records will display a filled, or hollowed, star icon, depending on the value of each of the list's items: if it is True, then a filled star appears, otherwise a hollowed star appears.

a) Open the **StarDisplay** Web Block in the canvas.
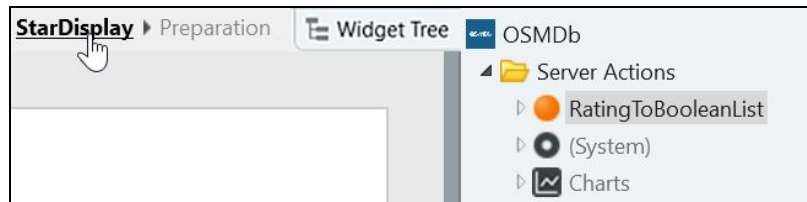
b) Drag and drop a **List Records** into the **StarDisplay** Block's main display area. Set its **Name** to *StarIterator*. Set its **Source Record List** to *RatingList* and its **Line Separator** to *None*.

c) Click on the magnifying glass, to the right of the module tabs, to expand the Search Box.

d) Type *Icon* in the Search Box. All the elements inside the current module with that substring in their name will appear.
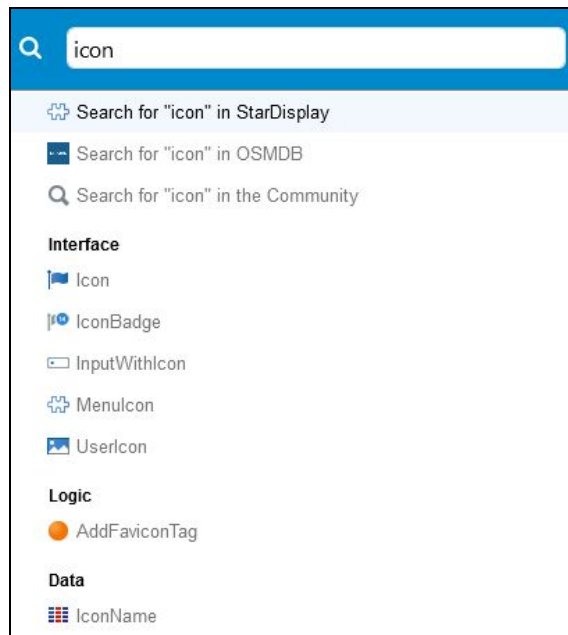
e) Hover the mouse over the dots on the left of the **Icon** element. Notice that the cursor turns into a hand, indicating it can be grabbed.

f) Drag and drop the **Icon** inside the List Records Widget.
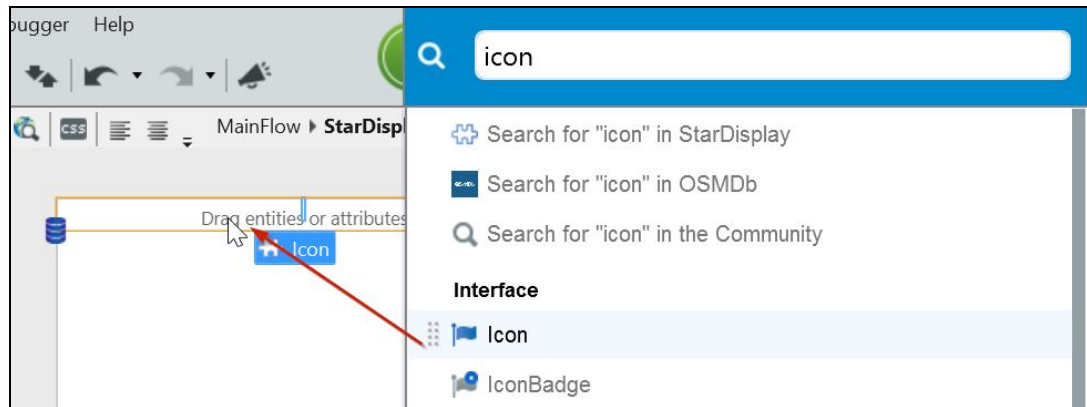
NOTE: The results of a search can also be double-clicked instead of drag and dropped. This will move the focus of Service Studio to where that element is defined. This is an invaluable tool to locate elements in a complex module.

g) As you want your Icon to be a star, in its properties, set its **Name** (second) property to *star* using the drop down.
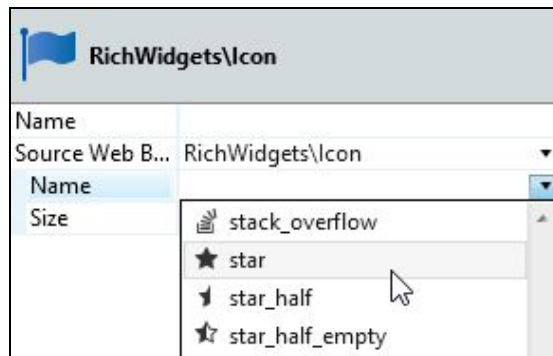
h) Right-click the Icon and select *Enclose in If*. This will move the (filled) star to the **True** branch of the **If** Widget.

i) Drag and drop another **Icon** into the **False** branch of the If. On this one, set its (second) **Name** to *star_o* (a hollow star).

j) To determine if the star is filled (i.e. less than or equal to the rating) or hollow, set the **Condition** property for the If to:

> *StarIterator.List.Current*

NOTE: To recap what was achieved in the steps above: the Preparation has a loop that produces a list of Booleans, that are *True* all the way up to the specified **Rating** and *False* from there on.

The **List Records** is using that list as source and iterates over it: on each element in the list, a star icon is displayed – filled if the element is true, hollow if it is not.

The **CurrentRowNumber** of the List Records indicates the row number, so while the row number is smaller than the **Rating**, the star will be filled. The Rating will depend on the star we clicked.

3) Add a **Placeholder** called *Suffix*, after the List Records, so that Screens that use this Web Block can optionally add other content into the Block.

a) Drag a **Placeholder** immediately after the List Records and set its **Name** to *Suffix*.
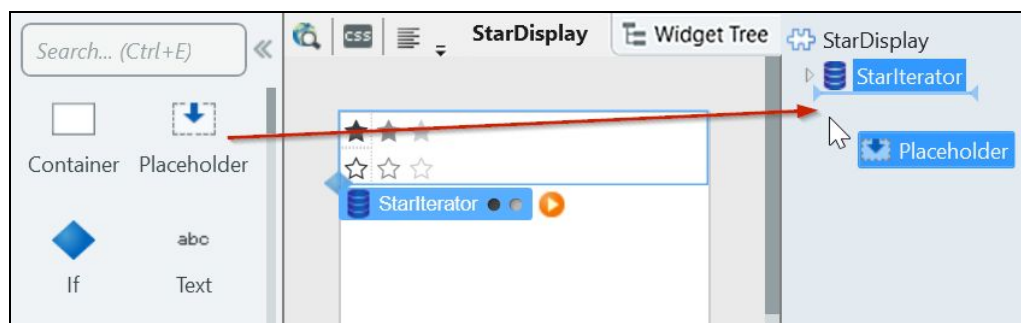
b) In the **Styles Editor** of the Placeholder and set the **Width** to *Fill*.
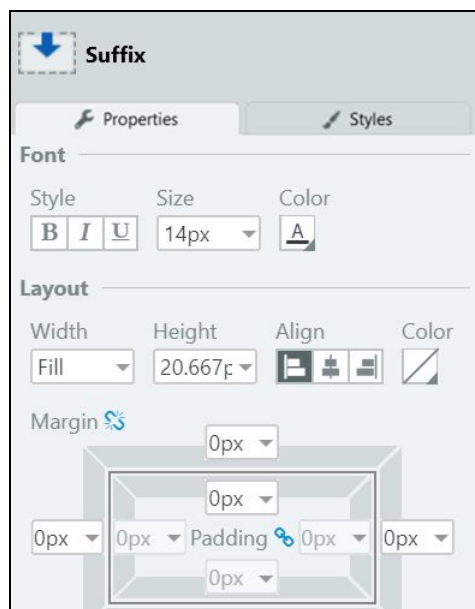
NOTE: A Placeholder allows the developer to define an area (a "hole") in a Web Block, that can (optionally) receive Screen content from where it is being used.

When a Block is used, its Placeholders will be highlighted with the specified **Name** and occupy the specified **Width**, even if the parent Screen doesn't specify any content to be injected on them. By setting the Placeholder's Width to *Fill*, the **Suffix** Placeholder will use the available space on the parent.

c) Ensure that the **Suffix** Placeholder is really after the List Records, and not inside. The full **StarDisplay** Block should have the following structure
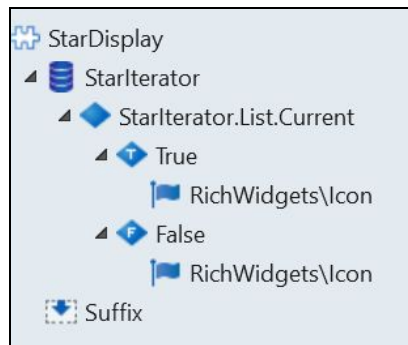
Add the Web Block to the Screen

A new instance of a Web Block is created when it's placed in the parent Screen or Block. Here, we will add the new Web Block to the MovieDetail Screen. This block instance will be for the purpose of showing the average movie star rating.

1) Add a new row to the **MovieDetail** Form, to contain an instance of the **StarDisplay** Web Block. Make it match the layout of the rows created via scaffolding.

   a) Open the **MovieDetail** Screen.

   b) Drag and drop a **Container** into the **MovieForm**, between the rows containing the **Title** and the **Year** Inputs. Set the **Margin-top** to *10px*.
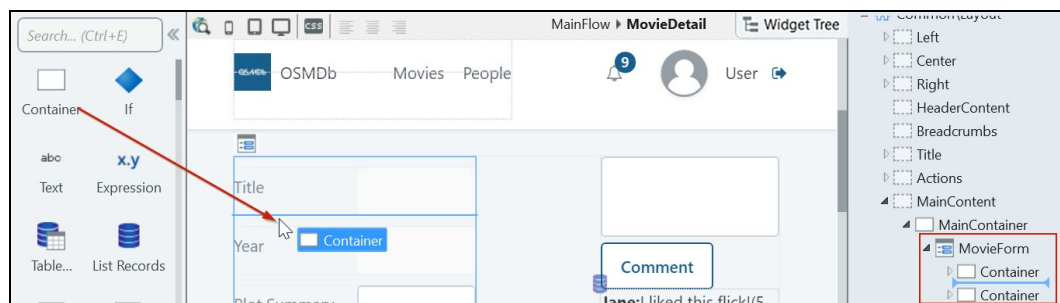

   c) Drag and drop a **Label** inside the new Container. Set its **Value** to *Average Rating*.

   d) Adjust the **Width** of the Label so that it is *4 col*.

   e) Drag and drop the **StarDisplay** Block to the right of the Label, inside the Container.
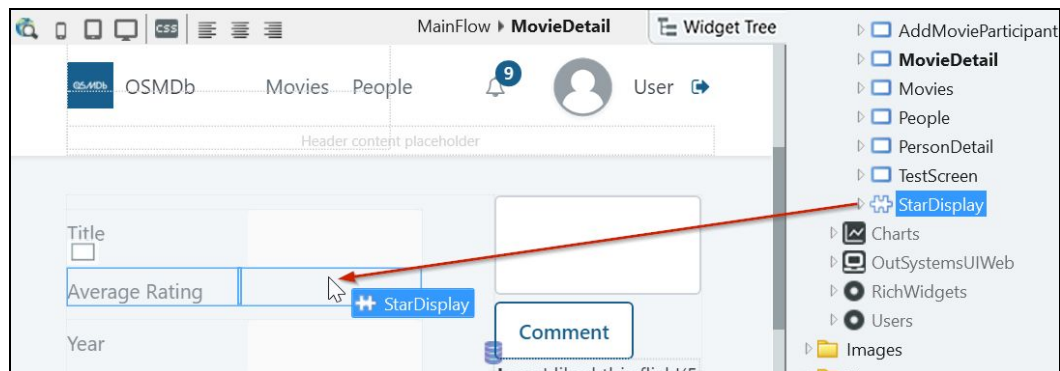
f) Right-click the **StarDisplay** Block instance in the Screen and select *Enclose in Container*. Set the new Container's **Width** to *6 col* and make sure the **Margin-left** is set to *(Auto)*. This will make the StarDisplay Block horizontally aligned with the other inputs.
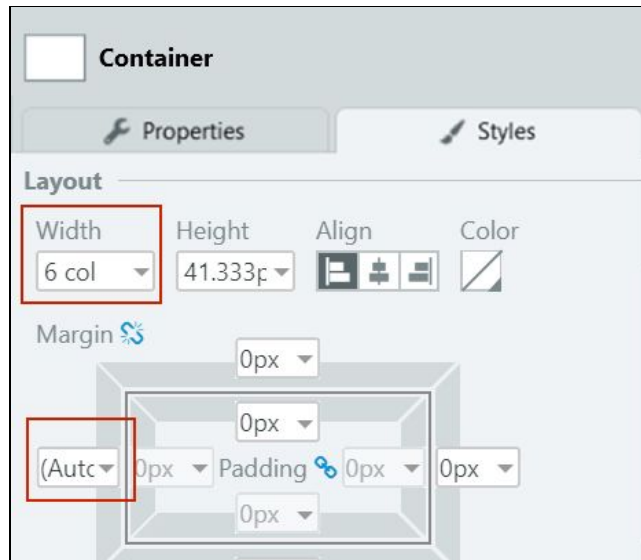
g) Set the new Container's **Name** to *AverageRatingContainer*.

h) The top part of the Screen's **MainContent** should have the following structure
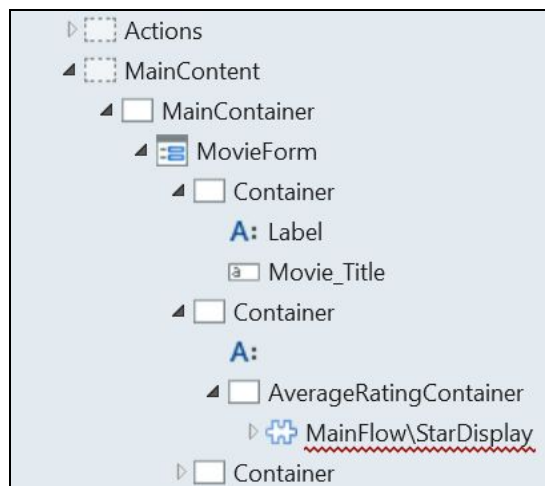
2) Add a new **Aggregate** to the MovieDetail Screen Preparation, that determines the average rating, as well as the total number of reviews, for the current movie. Pass the average rating to the **StarDisplay** Block, as an Input Parameter, and display the total number of reviews in the **Suffix** Placeholder of the Block.
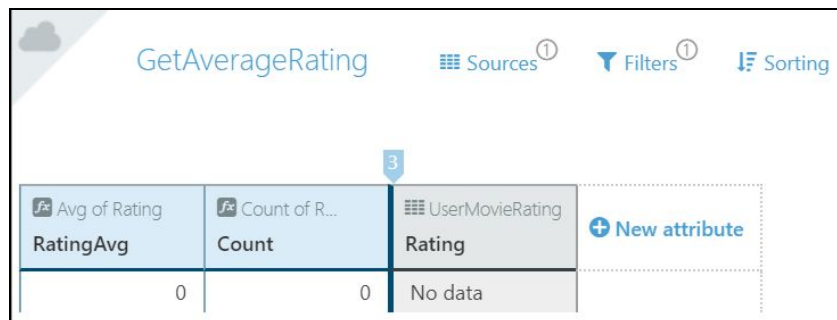
a) Open the **MovieDetail** Screen Preparation.

b) Drag and drop the **UserMovieRating** Entity, just before the End statement, to create an **Aggregate** to get all the ratings. Set its **Name** to *GetAverageRating*.

c) Open the **GetAverageRating** Aggregate and add the following filter:

> *UserMovieRating.MovieId = MovieId*

d) From the **Rating** attribute context menu, select *Average* to create a new aggregated column named *RatingAvg*.
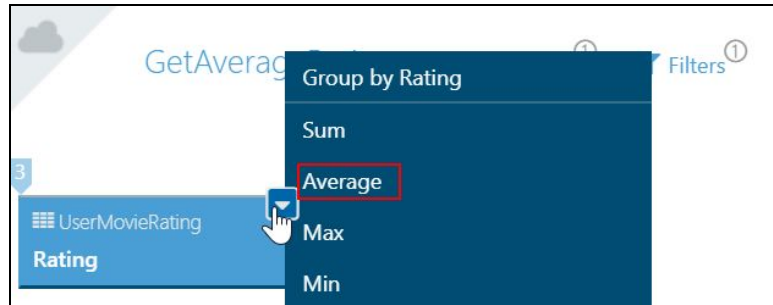
e) From the **Rating** attribute context menu, select *Count* to create a new aggregated column named *Count*.

f) The Aggregate should look something like this

g) Open the **MovieDetail** Screen and select the **StarDisplay** Block added in the previous steps.

h) The **StarDisplay** Block expects an Input Parameter **Rating**. Select the average calculated in the **GetAverageRating** Aggregate, by setting the **Rating** argument to *GetAverageRating.List.Current.RatingAvg*.

i) In the **StarDisplay** Block's **Suffix** Placeholder, add an **Expression** with the following **Value**
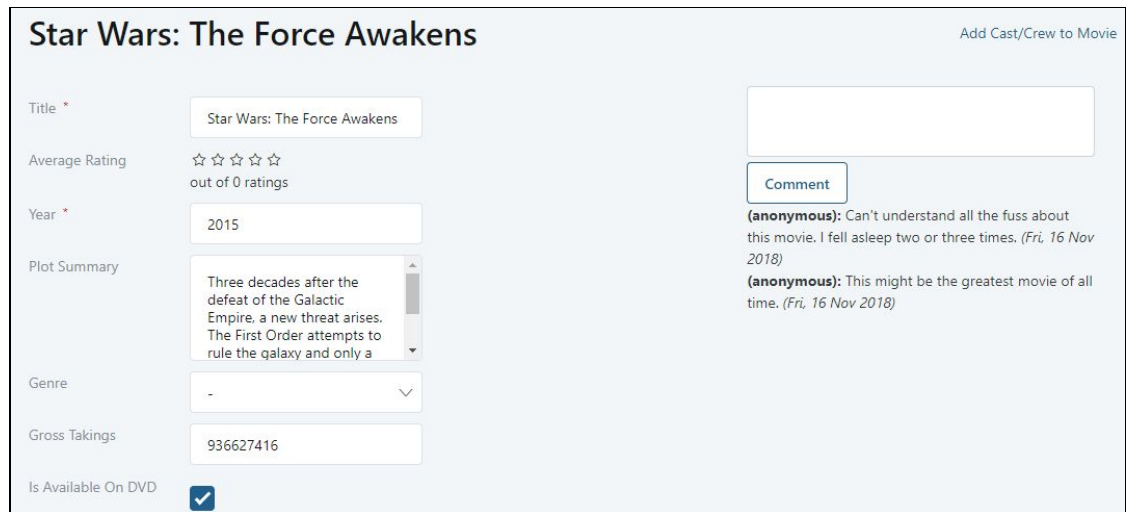
> *"out of " + GetAverageRating.List.Current.Count + " ratings"*

j) This section of **MovieDetail** should look like this
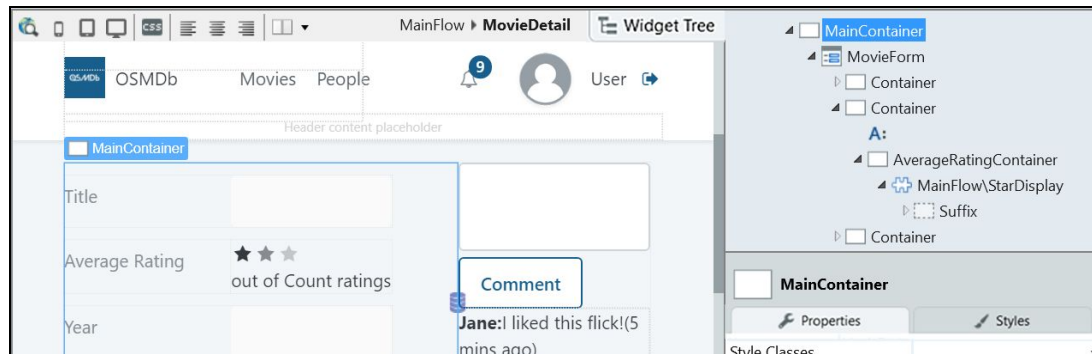
k) Click the **1-Click Publish** button to publish the application, and access it using your browser.

l) Navigate to the **MovieDetail** Screen**,** by selecting an existing movie. The Screen should look like this

Make the Stars in the Block Clickable

Now that we have used the Web Block to display the Average Rating, it's time to actually allow the user to rate a movie. For that, we will need to add a new instance of the Web Block to the Screen. However, the StarDisplay Web Block needs to be modified to allow the stars to be clickable, so that users can rate the movies. Let's do this, before we add a new instance of the Web Block to the Screen.

1) Make each of the stars in the **StarDisplay** Web Block clickable, to allow setting the rating selected. Use an Input Parameter to determine if the clickable behavior is applicable.

   a) In the Interface tab, right-click the **StarDisplay** Block and *Add Input Parameter*. Set its **Name** to *AllowClicking* and make sure its **Data Type** is Boolean.
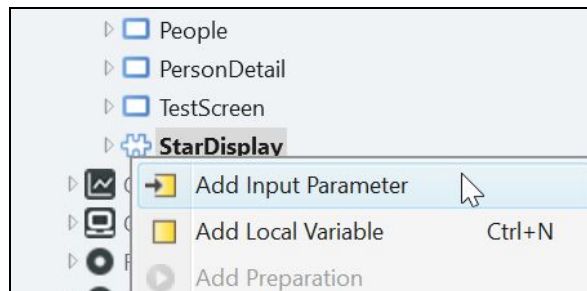

   b) With the **StarDisplay** Block opened, right-click the **filled star icon**, select *Link to* and choose *(New Screen Action)*.
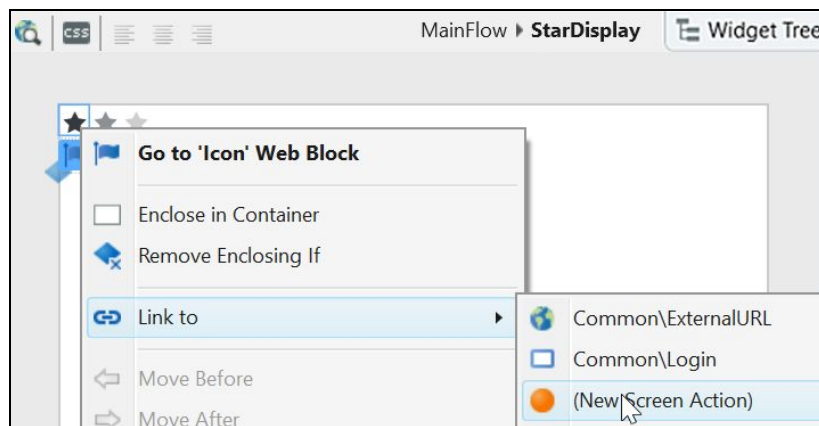

   c) Rename the new Screen Action to *StarClicked*.

   d) Right-click the hollow star and link it to the same Screen Action.

e) In both Links, set the **Enabled** property to *AllowClicking*, and the **Method** to *Ajax Submit*. In OutSystems, we can select multiple elements and configure common properties. So, in this step we can select both links in the Widget Tree and then set up the needed properties.
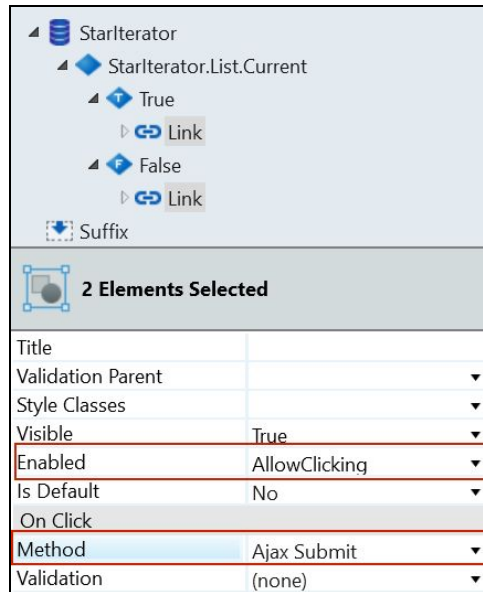

NOTE: Setting the **Enabled** property to the value of the Input Parameter AllowClicking allows disabling or enabling the Links based on the value of the parameter received. We can now control the logic of the application, when we allow the stars to be clickable.

f) Your **TrueChange** tab will show the following error
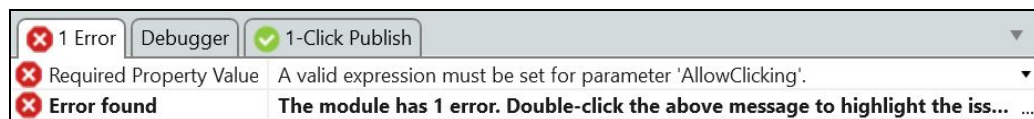
g) By double-clicking the error, Service Studio opens the **MovieDetail** Screen and selects the instance of the **StarDisplay** Web Block with the error.

h) Set the value of the Input Parameter **AllowClicking.** In this instance, set it to *False*, as the clicking is not applicable.
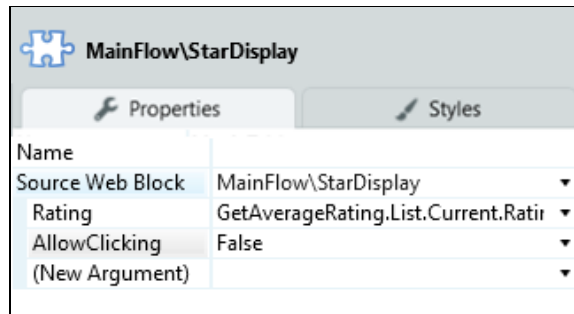
NOTE: In this case, we are showing the Average Rating for a movie, and therefore we do not want the stars to be clickable. Later, we will reuse this Web Block to allow the user to rate movies, with clickable stars.

---

2) When a star icon is clicked, we need to send a notification to the parent page, containing the selected rating. Since a Web Block does not have the context of the parent and vice-versa, the Screen does not know in which star the user clicked. We need to inform the Screen, by triggering an Event in the Block, that needs to be handled in the Screen.

a) Right-click the StarDisplay, select **Add Event** and name it *StarClickedEvent*. Set the **Is Mandatory** property to *No*.
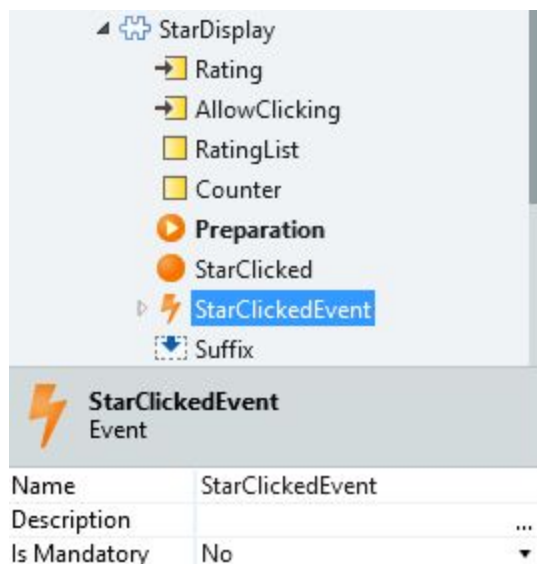

b) Right-click on the **StarClickedEvent** and add an *Input Parameter* with the name *StarClicked* and make sure its **Data Type** is an *Integer*.

c) Double-click the **StarClicked** Screen Action and drag the **StarClickedEvent** into the flow and set the StarClicked parameter to

StarIterator.List.CurrentRowNumber + 1

NOTE: When you add an **Event** to the logic inside a Block (in this case, the **StarClicked** Action), you need to catch and handle the **StarClickedEvent** wherever the Block is instantiated. This allows Web Blocks to communicate upward in the hierarchy to their parents.
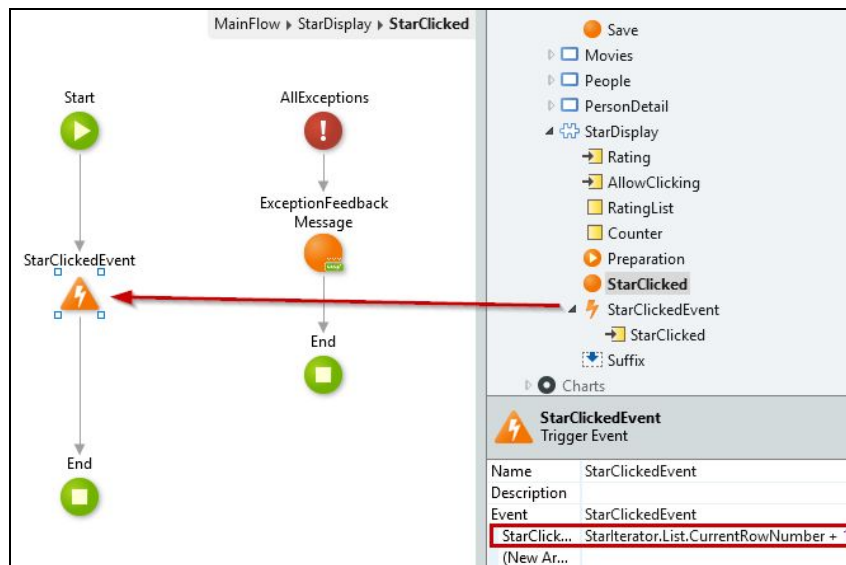

Make the Screen Handle Block Events

Here, we will add a new instance of the **StarDisplay** Web Block. This will allow the user to rate movies, as well as see his own rating.

1) Add a new instance of the existing **StarDisplay** Web Block to the **MovieDetail** Screen.

   a) Open the Preparation of the **MovieDetail** Screen.

   b) Drag and drop the **UserMovieRating** Entity, just before the **End** statement of the Preparation flow to produce an **Aggregate**. Set its **Name** to *GetUserMovieRating*.

   c) Double-click the **Aggregate** and add two filters

   > *UserMovieRating.MovieId = MovieId*

   > *UserMovieRating.UserId = GetUserId()*

   d) In the MovieDetail Screen, drag a new **Container** and drop it above the comments pane on the right, but still inside the **SideBar** Container.
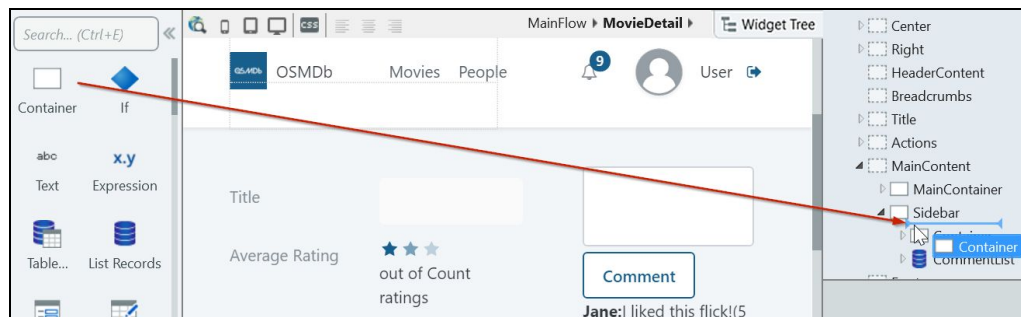
   e) Set the **Name** of the Container to *UserRatingContainer*.

   f) Drag the **StarDisplay** Web Block inside the Container created.

   g) Set the **Rating** parameter to *GetUserMovieRating.List.Current.UserMovieRating.Rating* and the **AllowClicking** to *True*.

h) Under Events, set the **Handler** property to *(New Screen Action)*. *OnRatingEvent* and the **StarClicked** parameter to *StarClicked*.
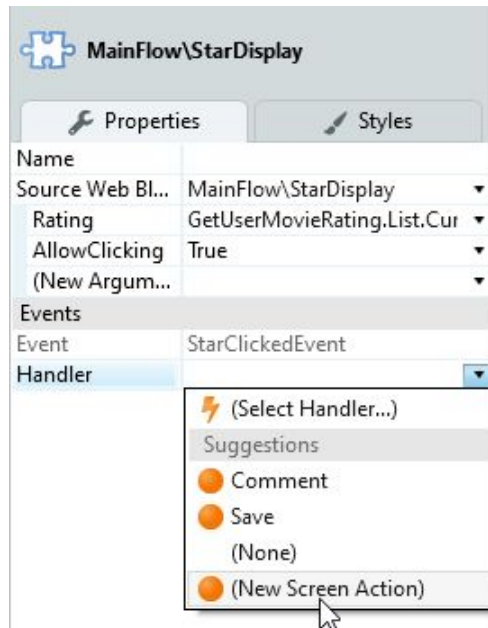

This creates a new Screen Action called **StarDisplayStarClickedEvent**. Rename it to *OnRatingEvent*.

NOTE: Since this instance of the Web Block is the one that the end-user will interact with, to choose the movie rating, we need to define an Handler for the Event triggered within the Block, so that on the Screen we know which star was clicked on. Remember that since the stars are defined inside the Block, the Screen itself does not know which star was clicked on, since it does not have access to the scope of the Block.

i) Confirm that the OnRatingEvent Action has an Input Parameter called StarClicked. This will make sure the Handler will have access to the star that was clicked on by the user.
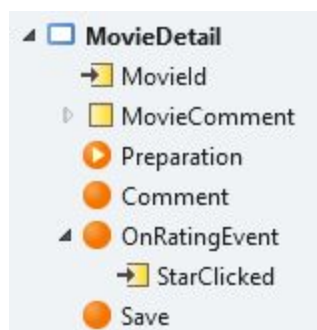
j) We now have an error on the instance of the StarDisplay Block. The error is related to the input parameter introduced in the previous step. To fix it, we need to pass a value to the OnRatingEvent. In the dropdown, select the **StarClicked** value.
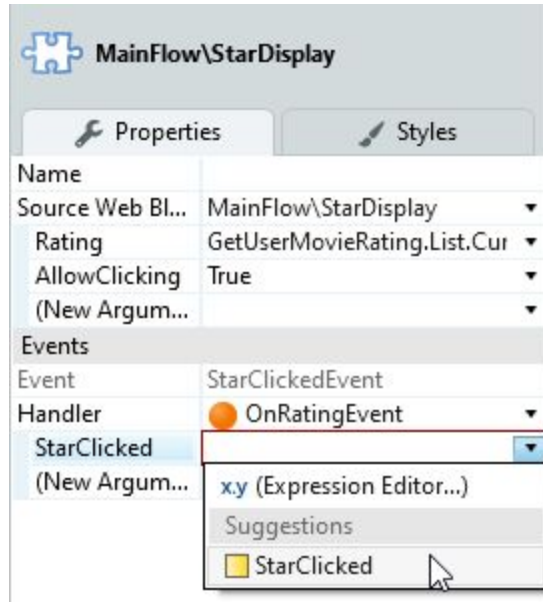

This is the actual value passed by the Event when triggered (its Input Parameter).

k) Add an If Widget on the left of the **StarDisplay** Web Block. This If will show different information, taking into consideration if the user already rated the movie or not.
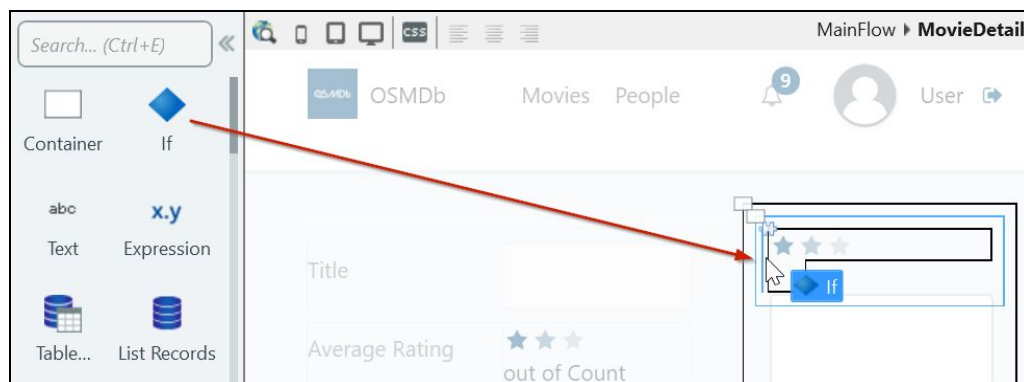
l) Set the **Condition** property of the If Widget to

GetUserMovieRating.List.Empty

m) In the **True** branch of the If Widget, add the following text *Rate this movie:*, as no **Rating** was given yet. In the **False** branch type *Your rating:*.
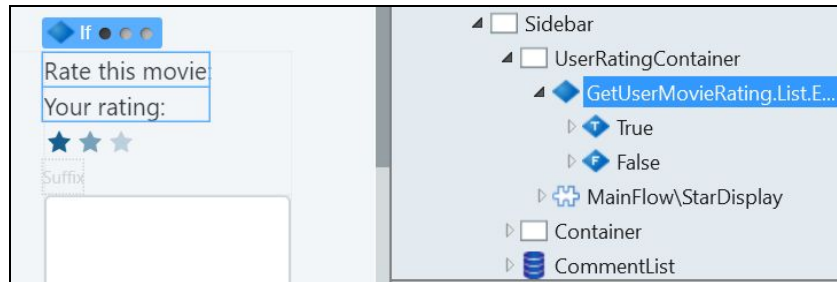
2) Create the logic to create or update the user rating, and refresh only the parts of the Web Screen related to movie ratings.

a) Open the **OnRatingEvent** Screen Action in the **MovieDetail** Screen.

b) Add an **Assign** statement and set the following assignments to define the rating to the movie, given by the user currently logged in.

> *GetUserMovieRating.List.Current.UserMovieRating.UserId = GetUserId()*
>
> *GetUserMovieRating.List.Current.UserMovieRating.MovieId = MovieId*
>
> *GetUserMovieRating.List.Current.UserMovieRating.Rating = StarClicked*

NOTE: The first two assignments are necessary, when creating a new rating, to get the user logged in and to assign the movie that the user is rating. When updating, the user and movie are already known. The third assignment gets the star that was clicked and assigns it to the Rating given to the movie. This way, we have the UserMovieRating record ready to be added to the database.

---

c) After the Assign statement, add the **CreateOrUpdateUserMovieRating** Entity Action. Set the **Source** property to *GetUserMovieRating.List.Current*, to create or update the rating in the database.

d) Add a **Refresh Data** statement to the Action flow just before the End statement, and set the **Data Source** to *GetUserMovieRating*, to re-execute the query and return the rating to be updated, if it exists.

e) Add an **Ajax Refresh** statement, and select *UserRatingContainer* as the Widget to be refreshed, to display the new rating on the Screen.

f) Add another **Refresh Data** statement and select *GetAverageRating* as the **Data Source**, to recalculate the average rating with the new rating.

g) After this last statement, add an **Ajax Refresh** statement, and set the **Widget** property to *AverageRatingContainer*, to display the new average.
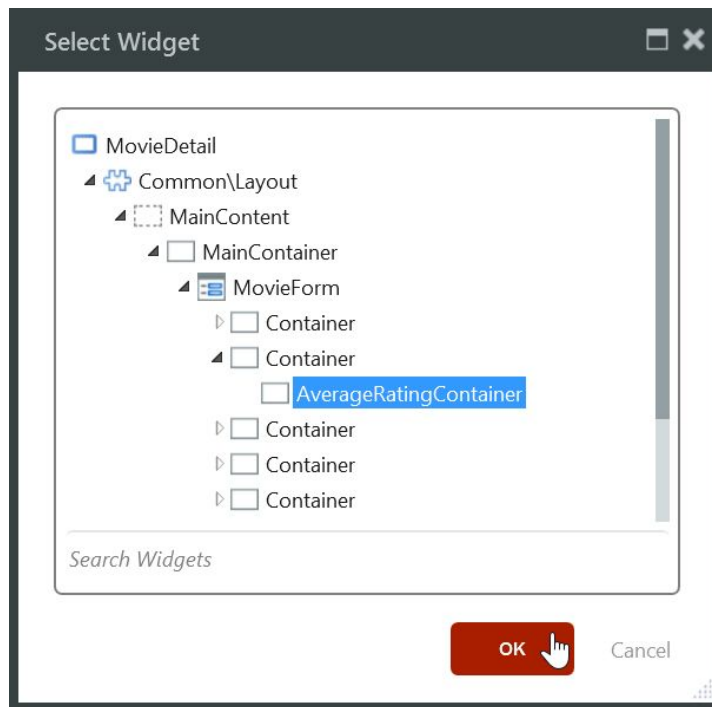
h) Set the **Animation Effect** to *Highlight*, to provide visual feedback.

i) The **OnRatingEvent** Screen Action should look like this
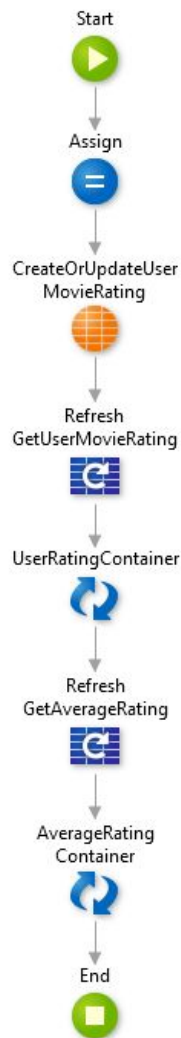
j) Click the **1-Click Publish** button to publish the application, and access it using your browser.

k) Navigate to the **MovieDetail** Screen, by selecting an existing movie. Rate the movie by clicking the stars in the right sidebar.

l) Notice that the rating did **not** change. This was **expected**, since the logic needs some extra steps to make this operation work properly.


To understand why, we need to analyze how the Screen is structured and also the lifecycle of traditional web applications. When we navigate to the Screen for the first time, the **RatingList** and the **Counter** Local Variables are set to their default values (all-False list and *0* respectively). When the Web Block is built in the Screen, its Preparation runs and the loop is executed. When the Preparation runs, the **Counter** Variable is now *5*. Now, when the end-user clicks on a star to give a rating, using the Ajax Submit method, the Preparation of the Block runs again, but the state of the variables are kept. So, as the Counter is still at 5, the loop in the Preparation does not run, since the If condition will be evaluated to False, and the RatingList is not updated. To solve this issue, the Preparation needs to begin with the Counter being reset to zero.

Also, we have another problem to fix. The ListAppend Action always adds an element to the end of the List. On a second rating, we would add five more stars next to the already existing 5, resulting in 10 stars, and so on. To avoid this issue, we need to reset the List as well, by clearing all its values, before a new Rating is given. That needs also to be done in the Web Block's Preparation.

3) Reset the Counter and RatingList, before a new Rating is calculated and the new stars are built.

a) Open the Preparation of the **StarDisplay** Web Block.

b) Drag and drop a **Run Server Action** statement before the If statement.
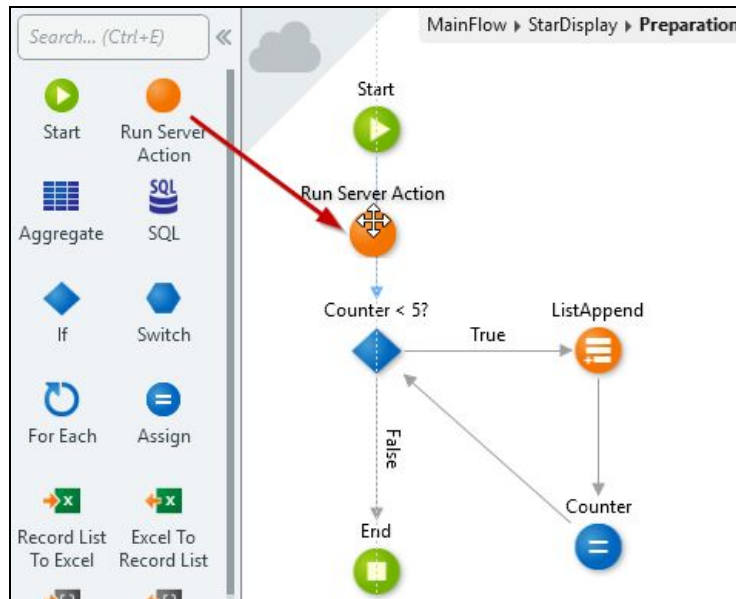
c) In the **Select Action** dialog, choose the **ListClear** Action. This Action removes all elements from a List.

d) In the **List** property of the Action, select the *RatingList* Local Variable.

e) Drag and drop a new **Assign** statement, right after the ListClear and before the If. Set its **Label** to *Reset Counter*.

f) Add the following assignment, to reset the Counter:

> *Counter = 0*

g) Set the **Label** of the other Assign to *Increment Counter*.

h) The Preparation should look like this:
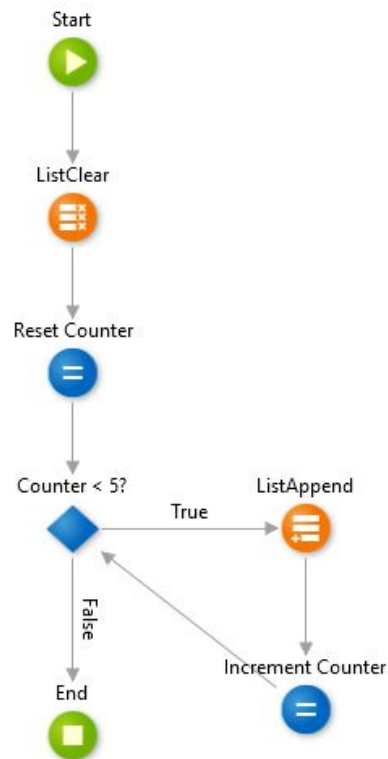
i) Publish the module and open the application in the browser. Make sure that the ratings work properly.

End of Lab

In this Lab, we created a reusable Web Block and learned that Web Blocks can be instantiated multiple times in the same Web Screen. In our case, it was used not only to allow a user to rate a movie, but also to display the average rating from all of the ratings made.

In one of the instances of the Block, the stars were clickable, to allow a user to rate, while in the other part, the stars were only for displaying purpose (average rating).

We also covered how to send notifications from a Web Block to its parent using Events. Finally, we also determined how the parent of a Web Block handles the events triggered, using Screen Actions as Event Handlers, to make sure that the parent had a proper answer to the Event.