

Blocks and Events - How-to

Table of Contents

| | |
|-------------------------------------|----------|
| Outline | 2 |
| How to | 2 |
| StarDisplay Block | 2 |
| Display the User and Average Rating | 10 |
| User Rating | 14 |
| Average Rating | 25 |

Outline

In this exercise we will introduce Blocks in our application, to allow a user to rate a movie. That rating will be based on a star rating system, where the user can rate a movie from 1 to 5 stars. This rating functionality should be added to the MovieDetail Screen.

Also in the same Screen, we should have the information of the Average Rating of the movie, which will combine all the ratings given by the end-users to that particular movie.

To accomplish this we will:

1. Create a Block to display a number of stars.
2. Make sure that the rating is visible to any end-user by the number of filled stars.
3. Display the average rating of a movie.
4. Without implementing the stars a second time, display a set of stars for the user to rate a movie.
5. Make the stars “clickable” and trigger the logic to create/update the rating of that user for that particular movie.
6. Guarantee that the average rating stars are not clickable.
7. Change the application to allow only Registered users to rate a movie.

A user can rate the same movie more than once. In those cases, the application should consider this as an update of the previous rating, and not as a new rating from that user.

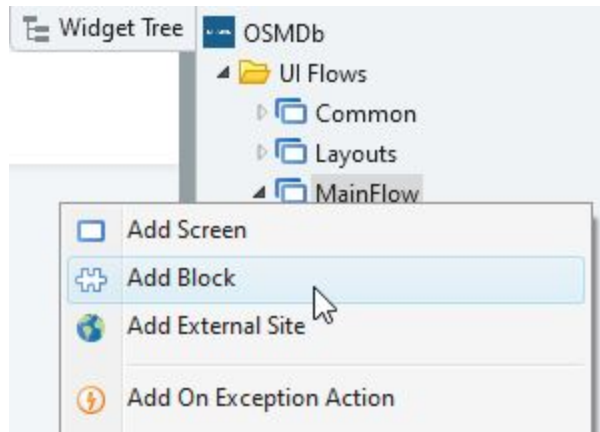
How to

In this section, we'll describe, step by step, the exercise *7.3 - Blocks and Events*.

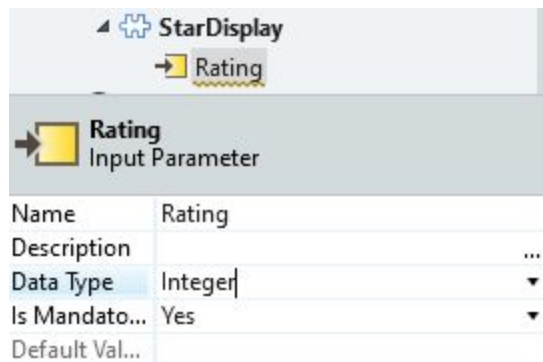
StarDisplay Block

1. Create the *StarDisplay* Block to display a list of stars. This will depend on a Rating input parameter.

- a. Under the Interface tab, right-click the MainFlow and select **Add Block**

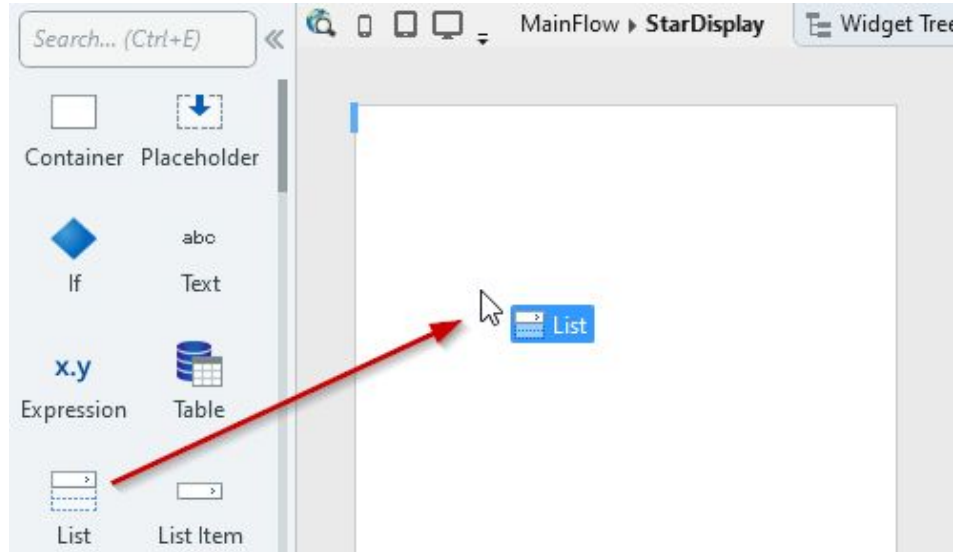


- b. Set the **Name** of the Block to *StarDisplay*
- c. Just like with Screens, right-click on a Block and select **Add Input Parameter** to add a new input. Set its **Name** to *Rating* and confirm the **Data Type** is *Integer*.

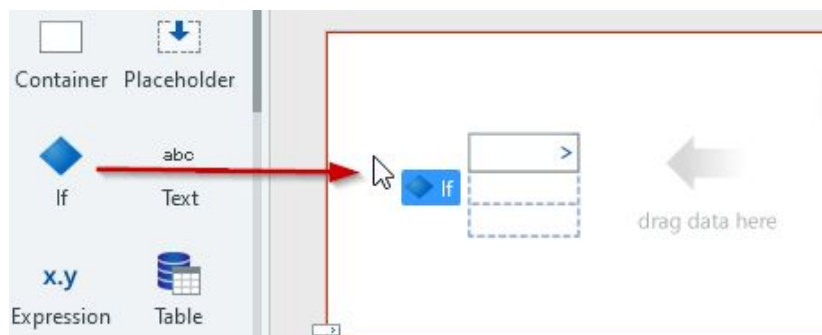


2. In this StarDisplay Block, create a list of stars that can be filled with color or hollowed, depending on the rating of a movie. For instance, if the movie has a rating of 3, we will have three colored stars and two "empty" stars.

- a. Drag a **List** to the main content area of the Block.



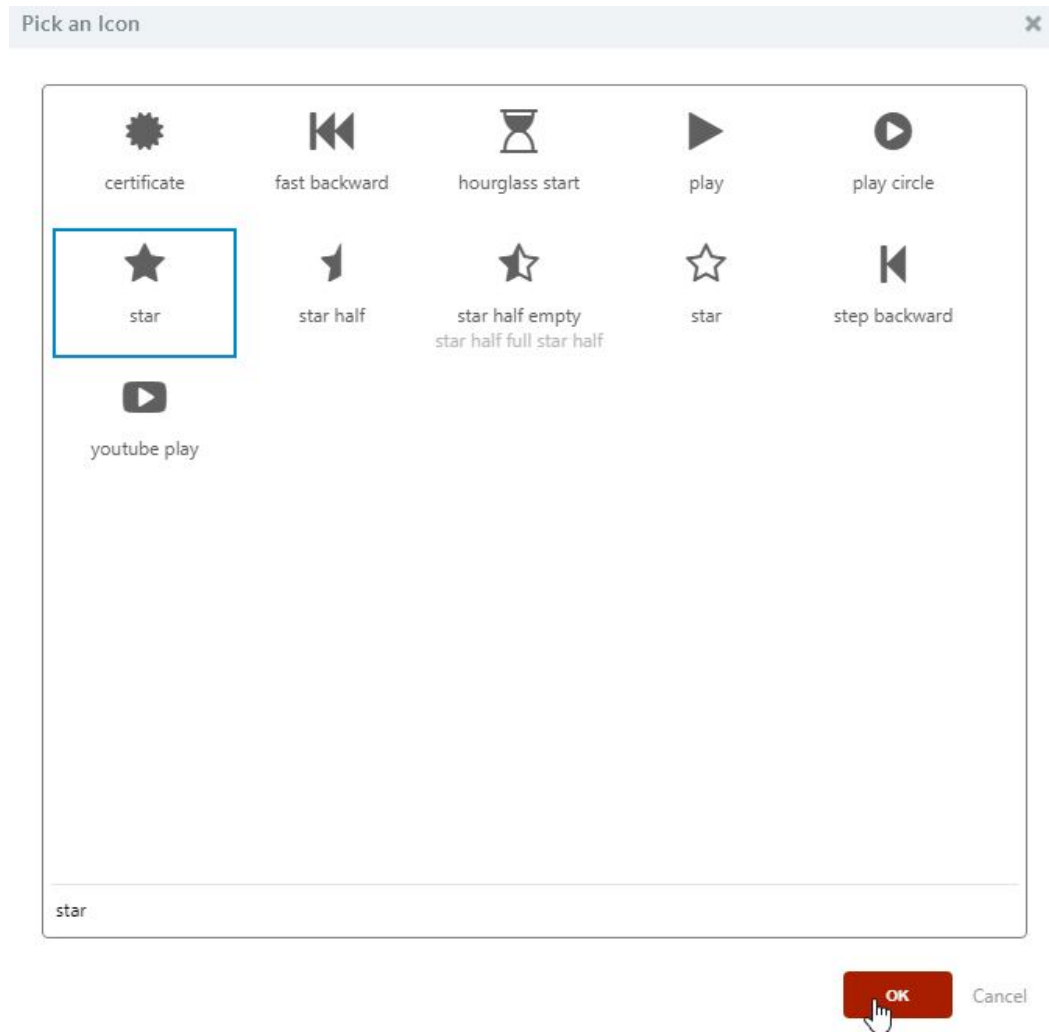
- b. Drag an **If** and drop it inside the List.



- c. Drag an **Icon** inside the **True** branch of the If.



- d. In the new dialog, select the **star** icon.



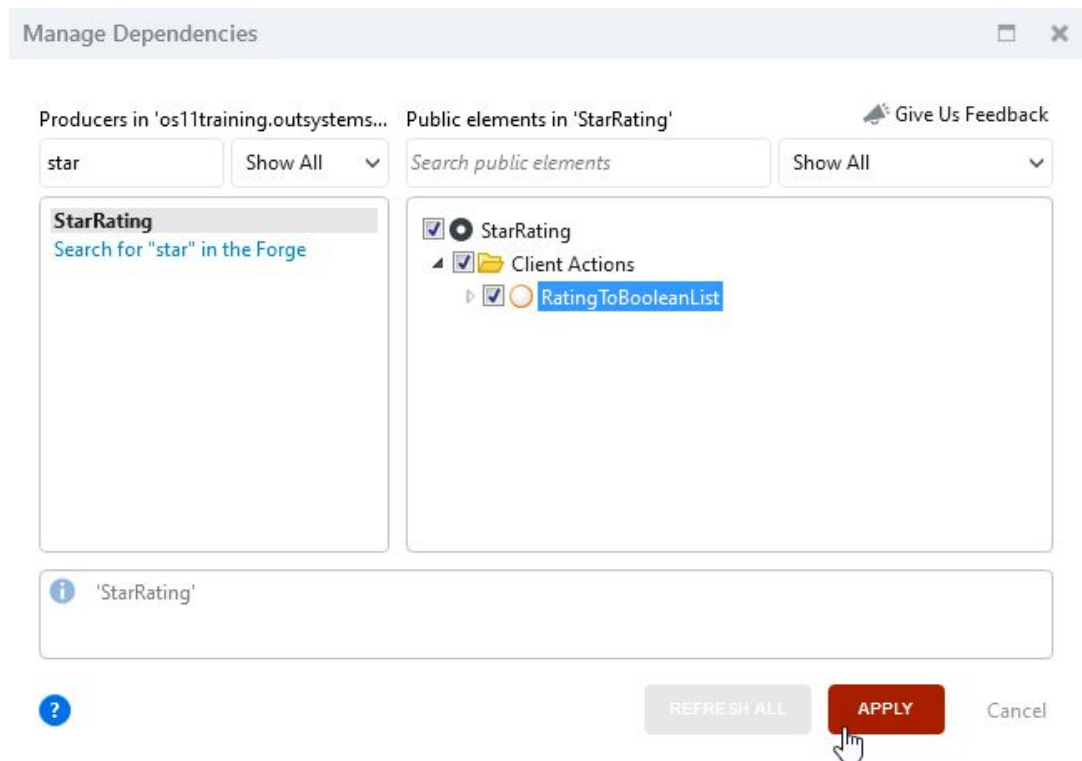
- e. In the properties of the Icon, set the **Size** to *Font size*
- f. Repeat the same process for the **False** branch, but this time choose the **white star** on the icons.



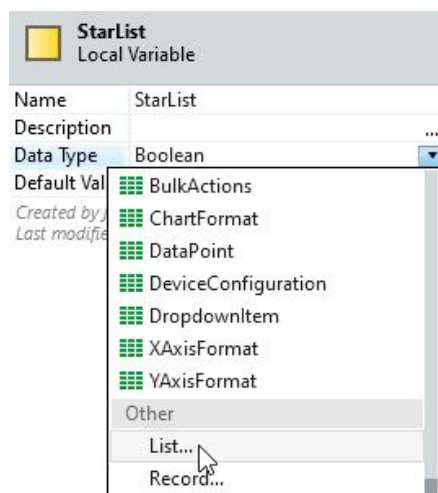
3. Now that we have the UI, we need the source of data. For that, we will leverage an existing Action, in the already existing **StarRating** module, called **RatingToBooleanList**.

This Action gets a Rating and transforms it into a list of booleans. Then, when the value in the List is true, the colored star should appear. Otherwise, it should be the white star.

- a. Open the Manage Dependencies window, select the **StarRating** module and reference the **RatingToBooleanList** Action.

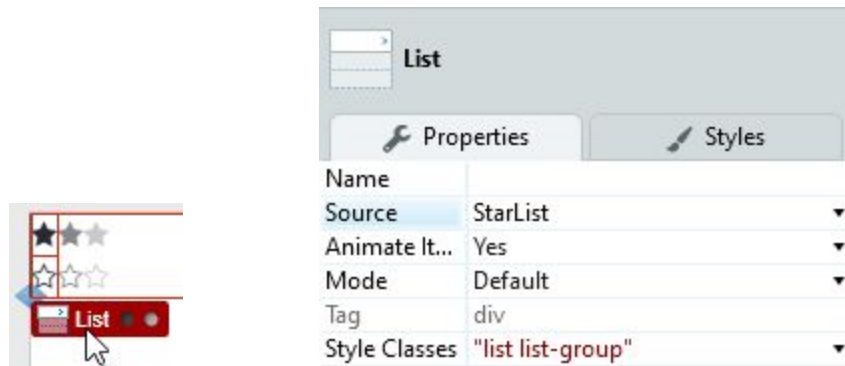


- b. Add one Local Variable to the StarDisplay block with the name *StarList* to the block. Set the **Data Type** to *Boolean List*.



NOTE: This variable will represent the boolean list that will come from the RatingToBooleanList Action and will influence how many colored stars will appear.

- c. Select the List on the Block and set its **Source** to be the *StarList*.

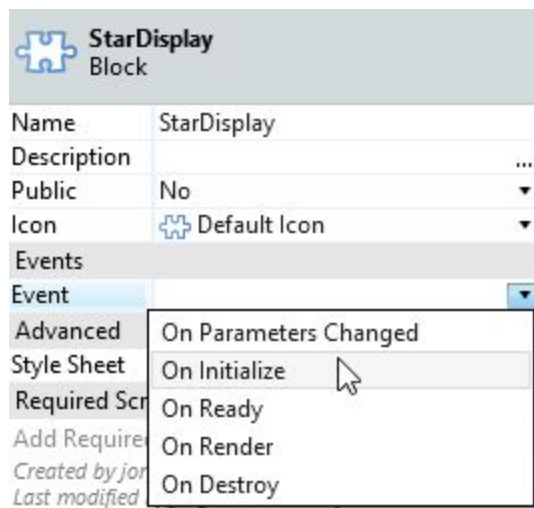


- d. Select the If and set its **Condition** to be *StarList.Current*

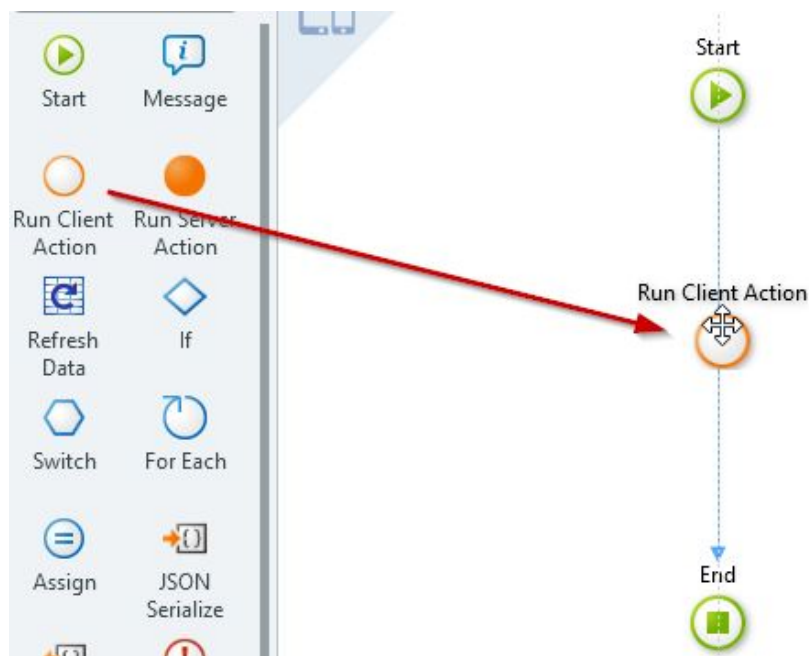
NOTE: Since the StarList variable is a List of Booleans, this means that every element of the list will be true or false. If it is true, the Block will display the colored star. In OutSystems, the List has a Current property, which contains by default the first element of the list. However, when used in a list or a loop, the Current can be used as an iterator. For instance, in the List in this Block, the first star will use the first element of the list, the second star will use the second element, etc. And this is done automatically under the covers. The developer just needs to use the Current property of the List.

- e. To initialize the StarList variable with data, since it is empty, we need to use the RatingToBooleanList Action. The question is: where to use it? Screens and Blocks have some Events that can be leveraged. One of them is the **On Initialize** Event, that is triggered automatically when the Screen/Block is initializing (more [here](#)).

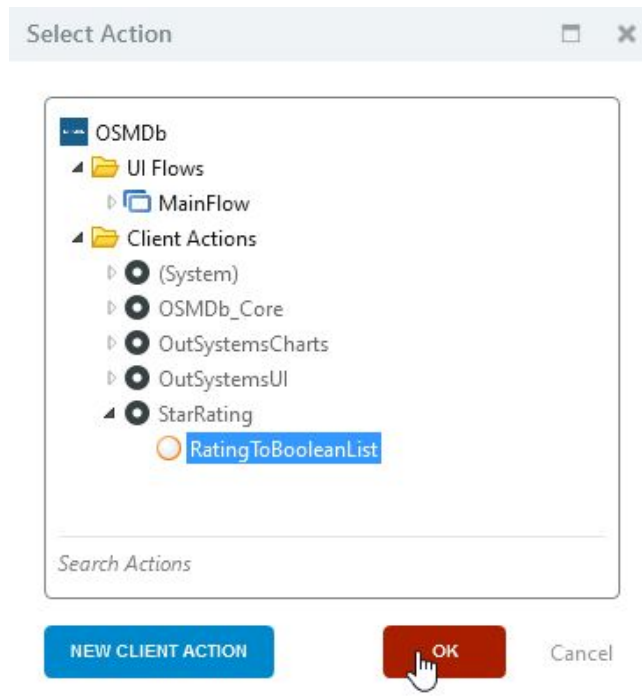
- f. In the properties of the StarDisplay Block, under Events, select the **On Initialize** option.



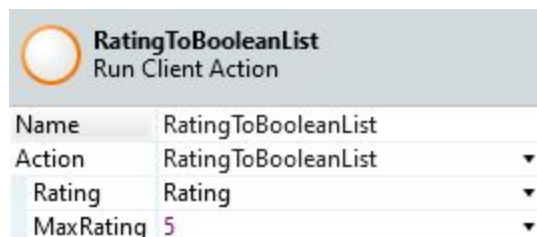
- g. This creates a new Action on the Block, called *OnInitialize*. Drag a **Run Client Action** and drop it on the new Action flow.



- h. In the new window, select the **RatingToBooleanList** Action.



- i. Set the **Rating** input of the Action to the Block's *Rating* Input Parameter and the **MaxRating** to 5.



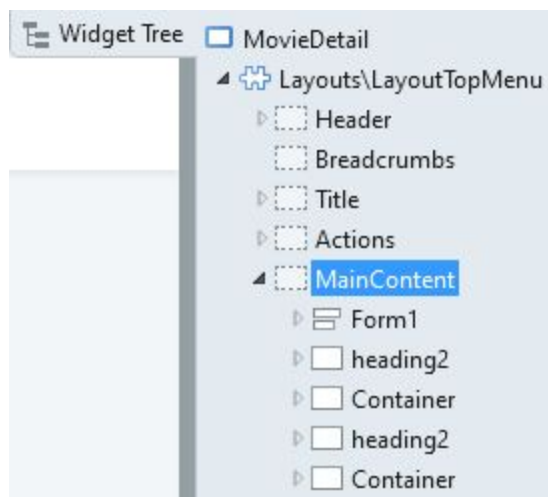
- j. Drag an Assign and drop it after the RatingToBooleanList Action. Define the following assignment to populate our local variable with the output of the Action.



Display the User and Average Rating

In the previous section, we created the UI for the stars in the StarDisplay Block. Now, we will leverage this UI to display the user and average ratings in the MovieDetail Screen.

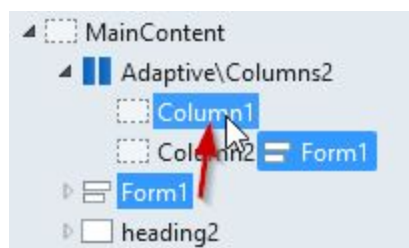
1. Let's get some space for the User Rating and Average Rating on the MovieDetail Screen. Split the Screen in 2 columns and use the right column to display the two ratings.
 - a. In the MovieDetail Screen, open the Widget Tree and expand the **MainContent** placeholder



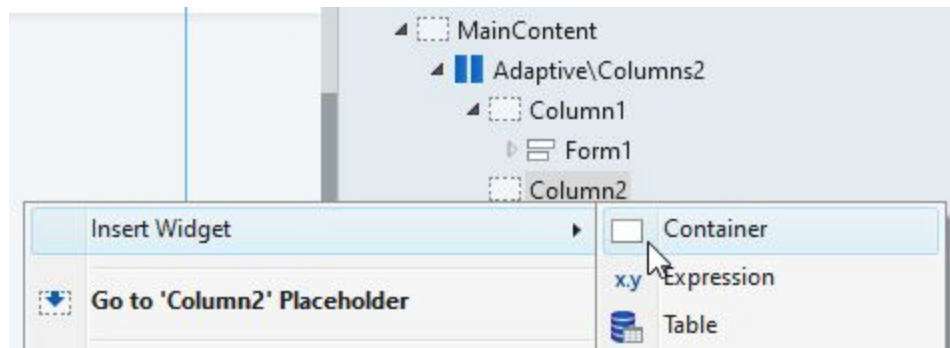
- b. Drag the **Columns 2** widget to the MainContent area, above the **Form1**.



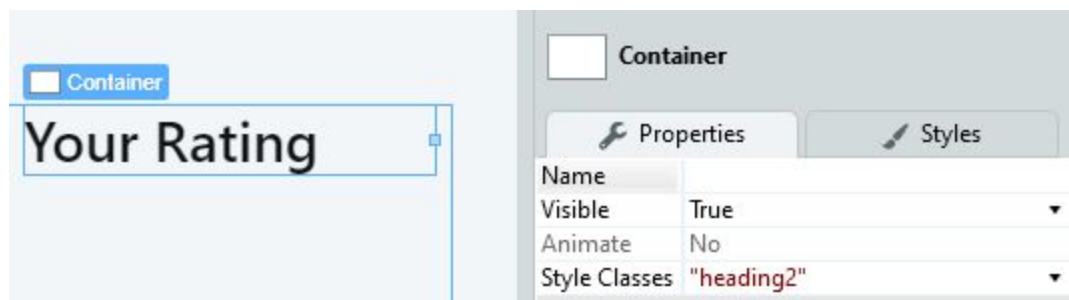
- c. Expand the Columns2 in the Widget Tree, drag the Form and drop it on **Column1**.



- d. Add two new Containers to the **Column2**



- e. On the first Container type *Your Rating* and set its **Style Classes** property to *"heading2"*.

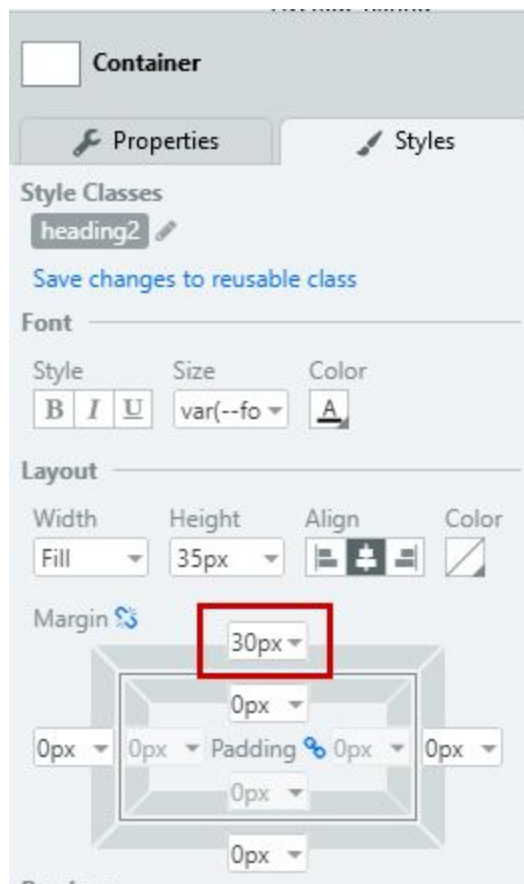


- f. Center the text inside the Container



- g. On the second Container type *Average Rating* and set its **Style Classes** property to *"heading2"*. Also center the Text inside the Container.

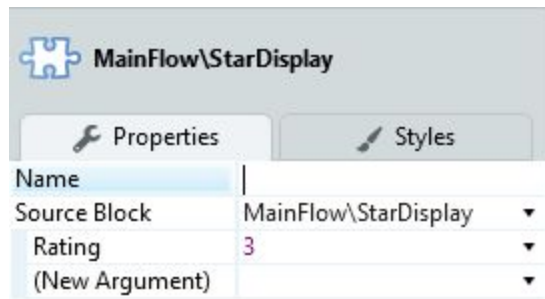
- h. Select the second Container, switch to the **Styles** tab and add a **Margin top** of 30 px.



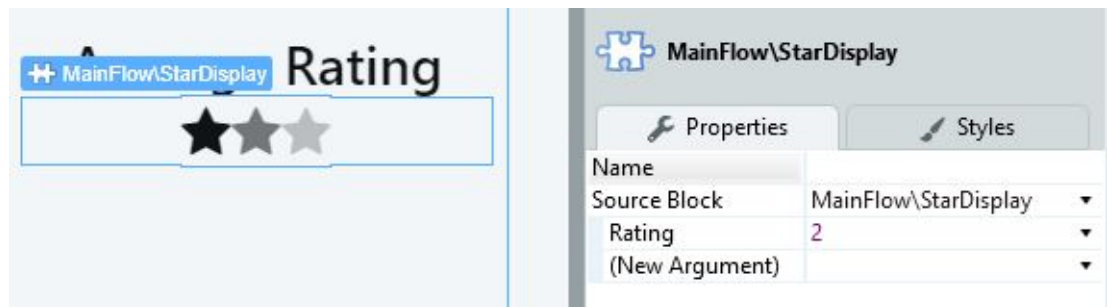
2. Use the StarDisplay Block to display the user rating and the average rating. Use some static rating values just to see the stars working.
- a. Drag the **StarDisplay** Block and drop it on the first Container, right below the Your Rating text.



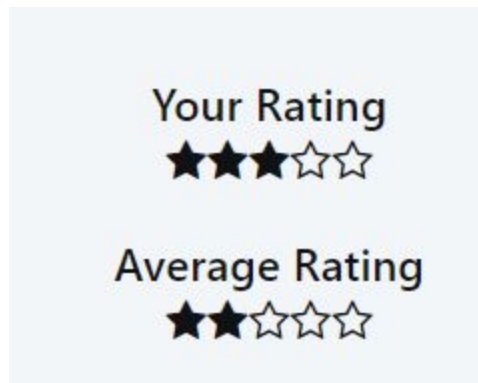
- b. Set the **Rating** input of the Block to 3.



- c. Drag again the **StarDisplay** Block, but this time drop it on the second Container, below the Average Rating text. Set the **Rating** to 2.



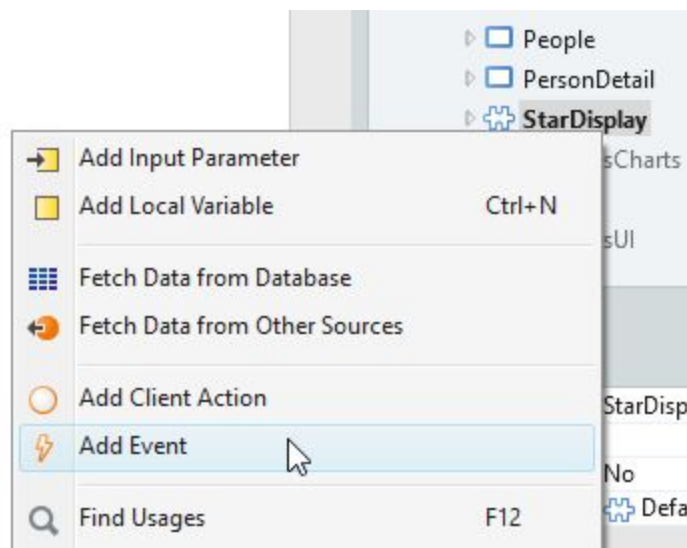
- d. Publish the module and make sure the stars appear properly on the browser.



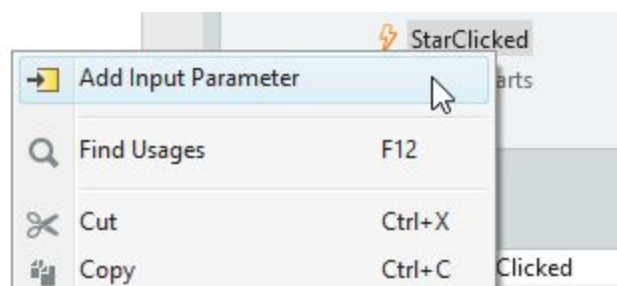
User Rating

The MovieDetail Screen already displays the stars using the StarDisplay Block. Now, we want to allow end-users to actually rate the movie, by clicking on the star. Then, we need to make sure that only the User Rating stars are clickable.

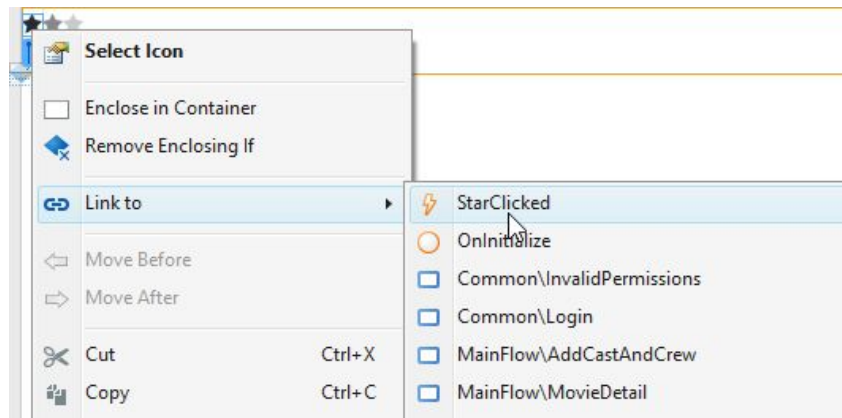
1. In the StarDisplay Block, use Links on the stars to support the clicking behavior. The Link should trigger an event that would send to the parent the star clicked, which will also mean the Rating.
 - a. Open the StarDisplay Block.
 - b. Right-click on the StarDisplay Block and select **Add Event**.



- c. Set the **Name** of the Event to *StarClicked*. Set its **Is Mandatory** property to **No**.
 - d. Add an Input Parameter to the Event and name it *SelectedStar*, with **Data Type** set to *Integer*.

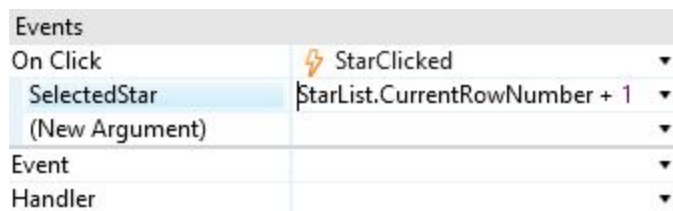


- e. Open the StarDisplay block and right-click on the colored star icon. Select **Link to** -> **StarClicked**



NOTE: We can define a Link to trigger an Event. In that case, the Event will be triggered when the end-user clicks on the Link.

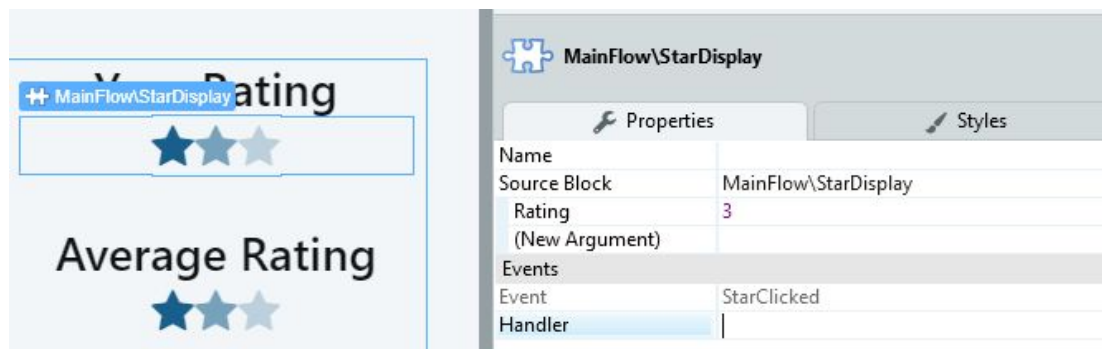
- f. On the new Link properties, set the **SelectedStar** input value to *StarList.CurrentRowNumber + 1*



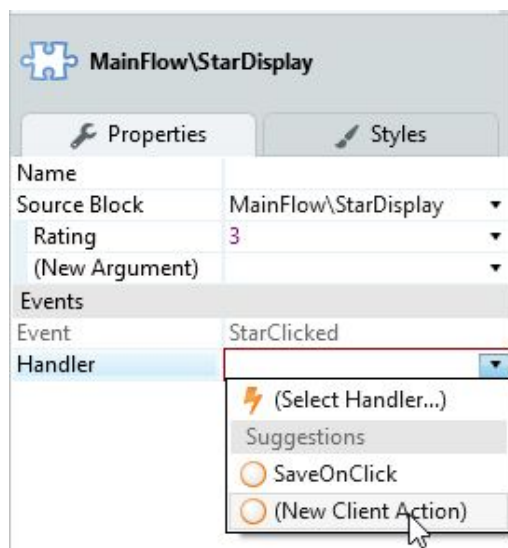
NOTE: The StarList is the Local Variable that is used as source of the list of stars in the Block. To know which star the end-user clicked on, we can use the CurrentRowNumber property that all lists have. The CurrentRowNumber has the position on the List of the Current property (which has the value). This property starts at zero, instead of one, hence the need for the +1 in our use case.

Let's see an example. In a List with {True, True, False, False}, the first element has the value True (Current = True) and the CurrentRowNumber will be 0, since we are in the first position of the list. The third element would be Current = False and CurrentRowNumber = 2. Back in our example, if the end-user clicks on the first star, the CurrentRowNumber is 0, but the rating the user is giving is actually 1. So we set the SelectedStar Input of the event to CurrentRowNumber + 1 (0+1), which corresponds to the rating of one star.

- g. Repeat the previous 2 steps for the “empty” star.
2. Since we have the Block triggering a non-mandatory event, now the parent may, or may not, respond to that event. On the User Rating, it makes sense that the parent uses the information about the star that was selected, to actually save the rating in the database. This operation should only be possible to Registered Users. For the Average Rating, we don't need to do anything for now.
 - a. Open the MovieDetail Screen and select the instance of the Block below the Your Rating Text.

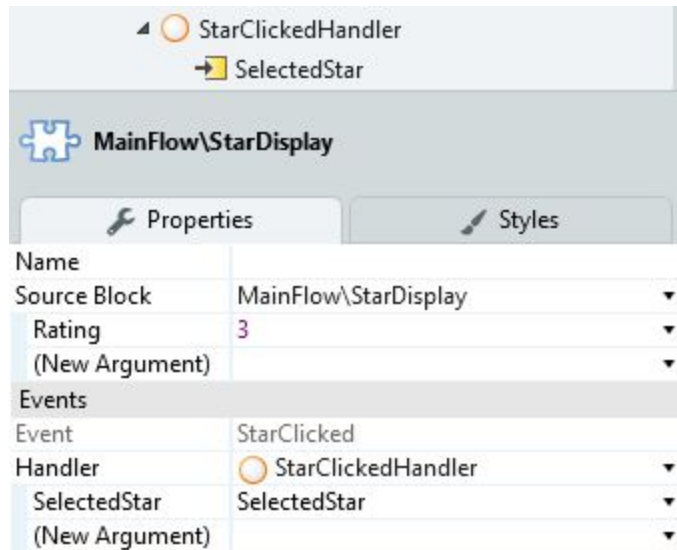


- b. On the **Handler**, select **(New Client Action)**. Give it the name *StarClickedHandler*.



NOTE: This Action will automatically run when the Event is triggered.

- c. Notice that the Action was created with an Input Parameter automatically and set with the **SelectedStar** value.

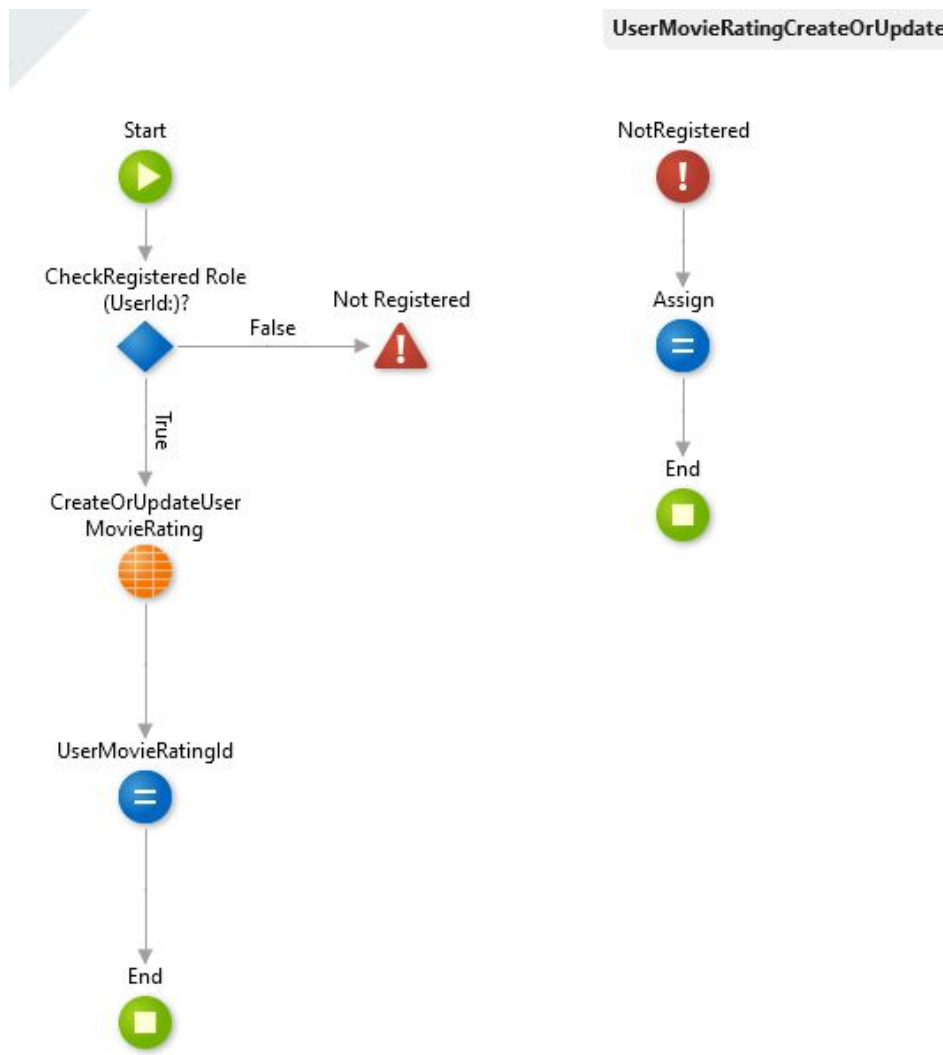


NOTE: The Event Handlers have access to the values of the Input Parameters of the respective Events. In this case, the Handler has an input parameter, which will hold the value of the Event Input Parameter with the same name.

- d. When the user rates a movie, we need to save that rating in the database. The UserMovieRating Entity will hold the ratings of all users. However, it is exposed as read-only. So, switch to the **OSMDB_Core** module to create the logic for adding the rating to the database.
- e. Under the Logic tab, create a new **Server Action** called *UserMovieRatingCreateOrUpdate*. Add one Input Parameter called *UserMovieRating* and three Output Parameters, *UserMovieRatingId* (UserMovieRating Identifier), *Success* (Boolean) and *Message* (Text). Set the Action as **Public**.

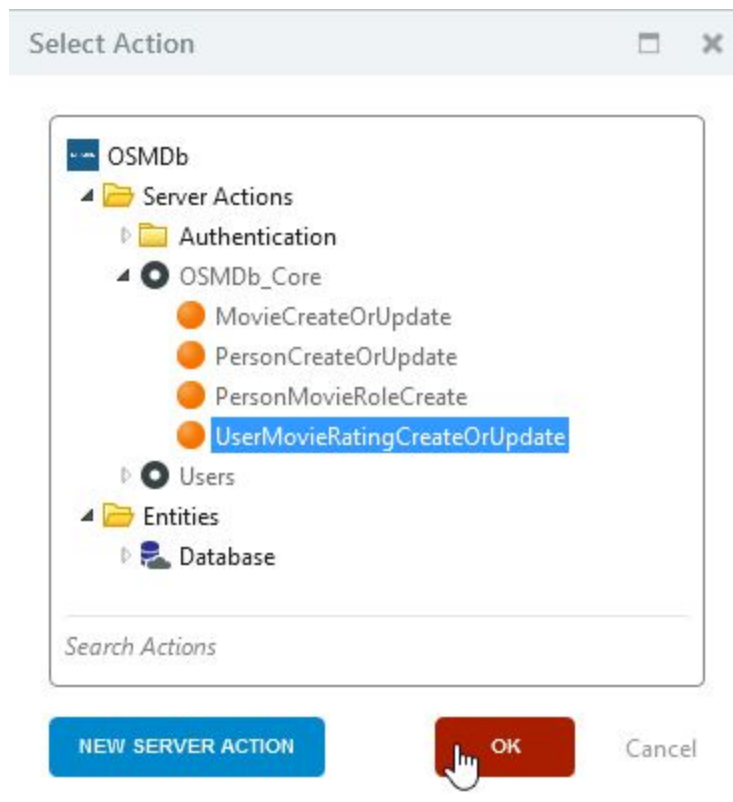


- f. The Action logic should be similar to the ones already in the OSMDb_Core module. But this time, the operation should be allowed for all **Registered** users.

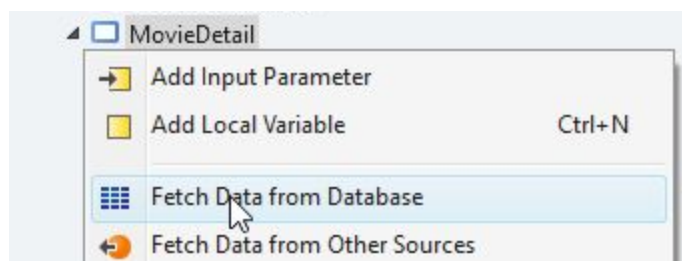


- g. Publish the module. Switch back to the OSMDb module and reference this Action using the Manage Dependencies.
- h. Under the MovieDetail Screen, open the **StarClickedHandler**.

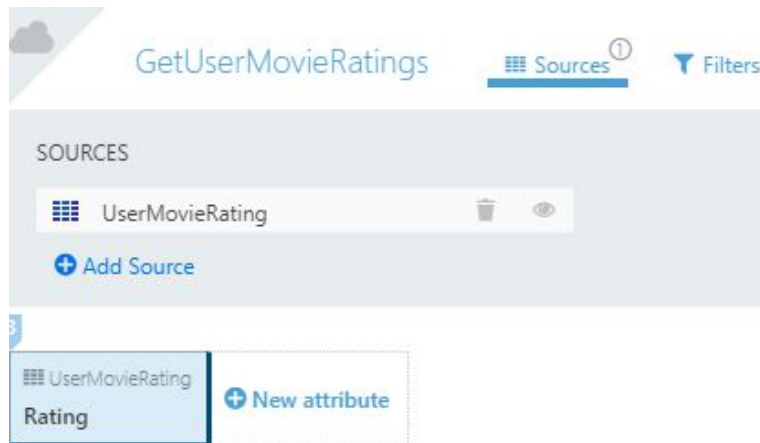
- i. Drag a **Run Server Action** and drop it on the Action flow. Select the **UserMovieRatingCreateOrUpdate** Action.



- j. The Action expects a UserMovieRating as input. Since we are on a scenario that we may be adding a new rating, or updating an existing one, we need to fetch the rating from the database, if it exists. Right-click on the MovieDetail Screen and select **Fetch Data from Database**.



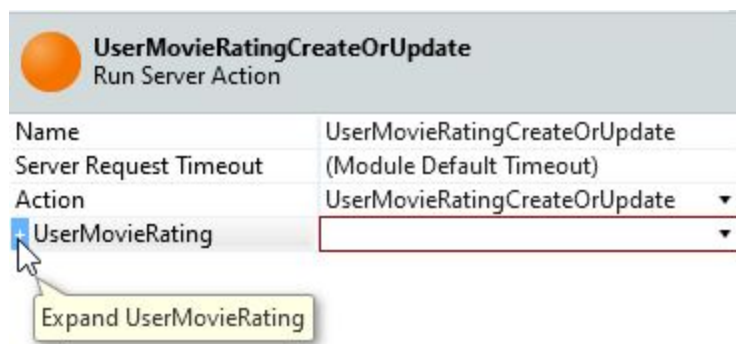
- k. In the Aggregate, set the **UserMovieRating** as Source of the Aggregate.



- l. Create two new filters in the Aggregate to get the rating for the user logged in, using *GetUserId()*, and for the movie currently on display in the MovieDetail. Update the name of the Aggregate to *GetUserMovieRatingsByUserId* if it does not automatically update.



- m. Back in the **StarClickedHandler** Action, select the **UserMovieRatingCreateOrUpdate**, expand the **UserMovieRating** input parameter of the Action.




- n. Set the attributes of the UserMovieRating record with the following values

Id = GetUserMovieRatingsByUserId.List.Current.UserMovieRating.Id

UserId = GetUserId()

Movied = Movied

Rating = SelectedStar

|  UserMovieRatingCreateOrUpdate Run Server Action | |
|---|---|
| Name | UserMovieRatingCreateOrUpdate |
| Server Request Time... | (Module Default Timeout) |
| Action | UserMovieRatingCreateOrUpdate ▼ |
| UserMovieRating | |
| Id | GetUserMovieRatingsByUserId.List.Current.UserMovieRating.Id ▼ |
| UserId | GetUserId() ▼ |
| Movied | Movied ▼ |
| Rating | SelectedStar ▼ |

NOTE: It is possible to set individually the values for each attribute in a record, which is what we are doing in this case with the UserMovieRating input of the Server Action.

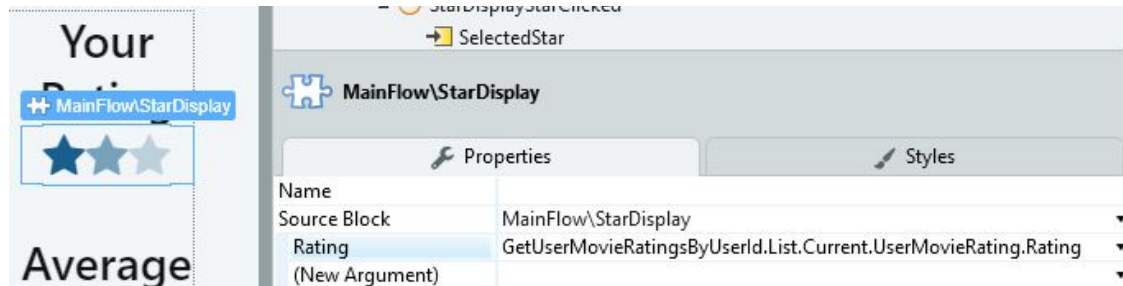
For the ID, we get the record that was fetched from the database. If there is already a rating, we use the same Id, if there's not, then the Id will go as 0 and a new record will be created in the database.

For the UserId, we set it to be the user currently logged in.

For the Movied, we use the input parameter of the MovieDetail Screen of the same name.

And for the Rating, we use the SelectedStar value, which is the input parameter of the Event of the StarDisplay Block.

- o. Going back to the MovieDetail Screen, select again the instance of the Block below the Your Rating text and change the Rating value to *GetUserMovieRatingsByUserId.List.Current.UserMovieRating.Rating*



NOTE: Now, we can pass to the Block the actual rating that the movie has in the database, which is being fetched in the Aggregate created above, instead of a static value.

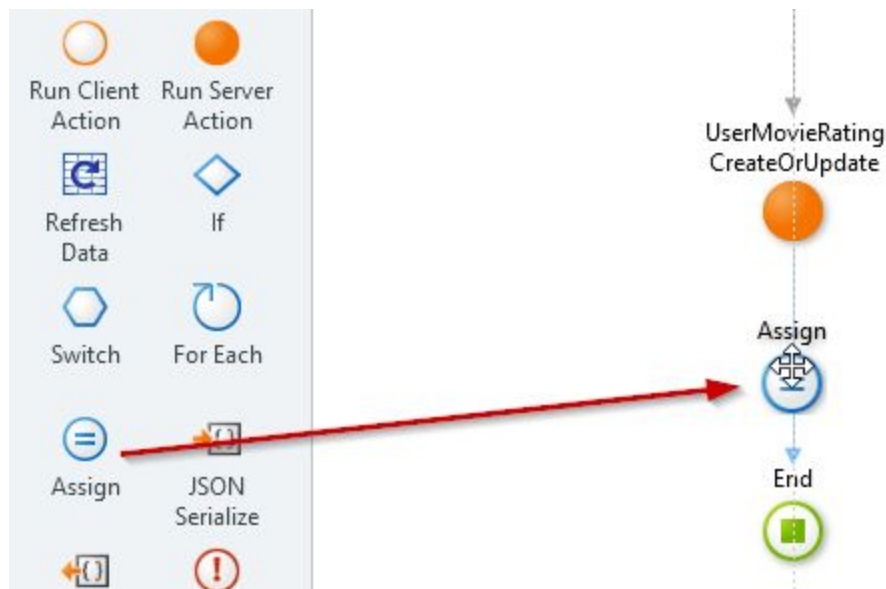
3. Let's test this in the browser by rating a movie. We will notice that the rating does not change, meaning that the right amount of colored stars do not appear. Let's fix that by guaranteeing that when a rating is given, the stars refresh automatically on the Screen.
 - a. Publish the module and open it in the browser.
 - b. Login with any of your users.
 - c. Select any movie that does not have a rating and click on a certain number of stars. Do they appear colored? The answer should be no...



NOTE: This happens because we save the rating in the database, but two things are missing: the rating fetched from the database (with the Aggregate) still does not reflect these changes, and the Block "does not know" that the Rating has changed. In the case of the Block, do not forget that the stars depend on a List, which has not changed yet, so the stars do not change as well.

4. Let's solve the problem above, by updating the information about the Rating and by updating the list of stars inside the Block.

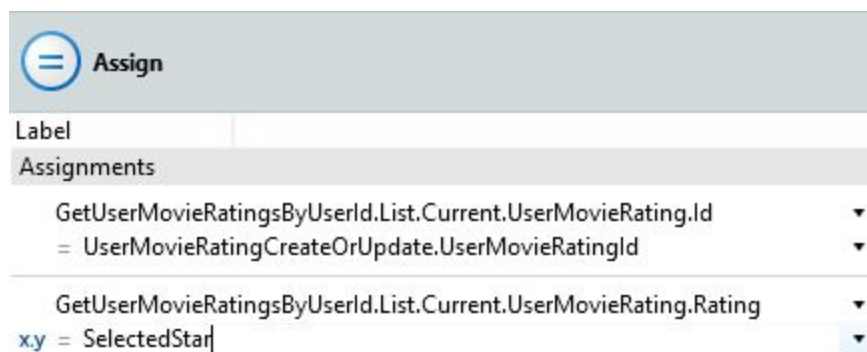
- a. Open the StarClickedHandler Action, drag an **Assign** and drop it after the **UserMovieRatingCreateOrUpdate** Action



- b. Define the following assignments

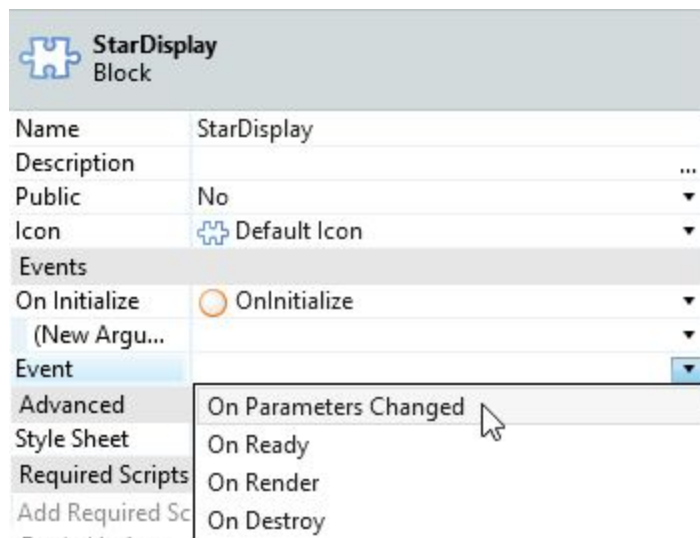
GetUserMovieRatingsByUserId.List.Current.UserMovieRating.Id = UserMovieRatingCreateOrUpdate.UserMovieRatingId

GetUserMovieRatingsByUserId.List.Current.UserMovieRating.Rating = SelectedStar

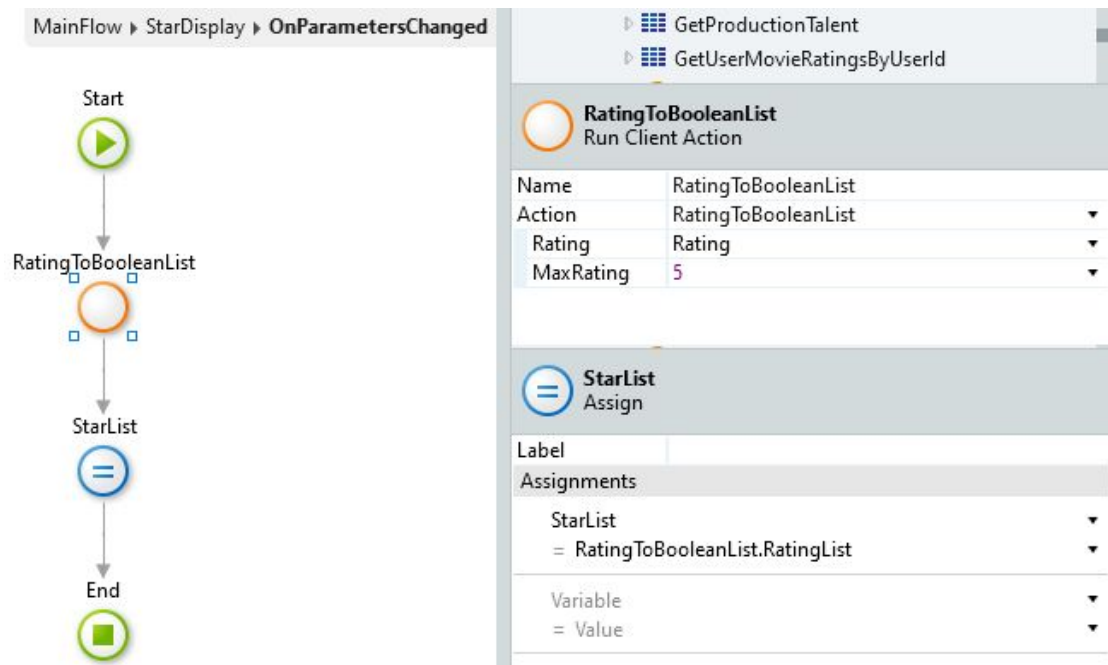


NOTE: This updates the UserMovieRating record fetched from the Aggregate with the new Id (if the rating didn't exist) and with the new rating.

- c. Open the StarDisplay Block and on the Event properties, select the **On Parameters Changed**



- d. This creates a new Client Action inside the Block, *OnParametersChanged*. This Action should recalculate the **StarList** variable, using the **RatingToBooleanList** Action. The logic is just like the one in the **OnInitialize** Action.



NOTE: The On Parameters Changed Event is automatically triggered when one input parameter of the Block is modified in the parent. Since we changed the rating in the previous step, the Rating Input Parameter of the Block will have its value changed and the Event will be automatically triggered. In the logic, we execute the RatingToBooleanList with the new Rating, resulting in a new StarList, which will then display the right amount of colored stars in the Block.

- e. Publish the module and test the app in the browser. Does the rating appear correctly now?

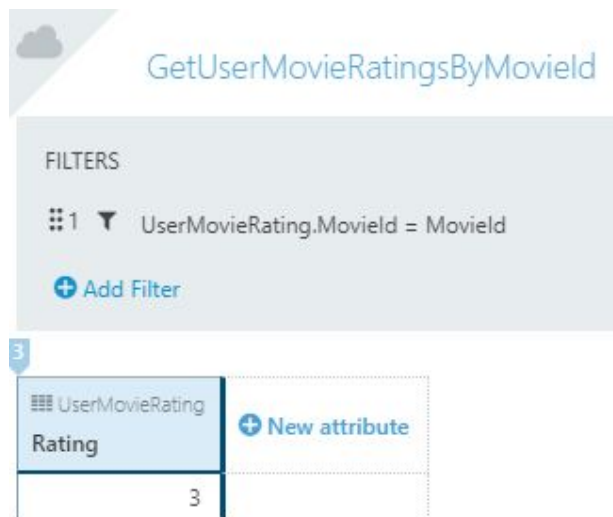


Average Rating

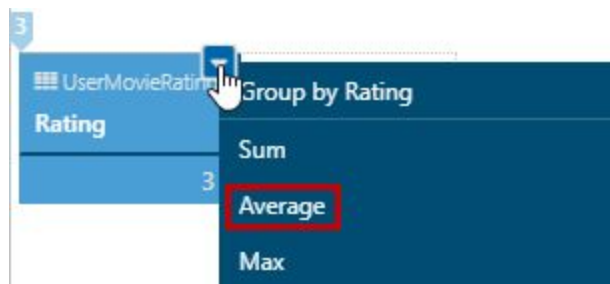
Now that we have the user movie rating implemented, we need to deal with the Average Rating. First, we need to calculate the average rating for all the movies and pass that information as the input parameter of the Block. That will indicate how many colored stars should appear on that particular instance of the Block in the MovieDetail Screen. Then, we need to make sure that the average rating stars are non clickable. Only the user rating can be clickable.

1. Fetch from the database the average rating of votes for a particular movie and pass that information to the StarDisplay Block.
 - a. Right-click on the MovieDetail Screen and select **Fetch Data from Database**.
 - b. In the new Aggregate, select the **UserMovieRating** Entity as Source.

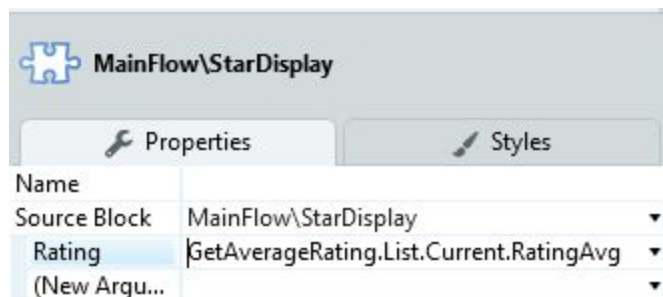
- c. Add a new Filter to guarantee that we are only fetching the ratings from this movie



- d. Select the Rating column in the Aggregate and choose the **Average** option.



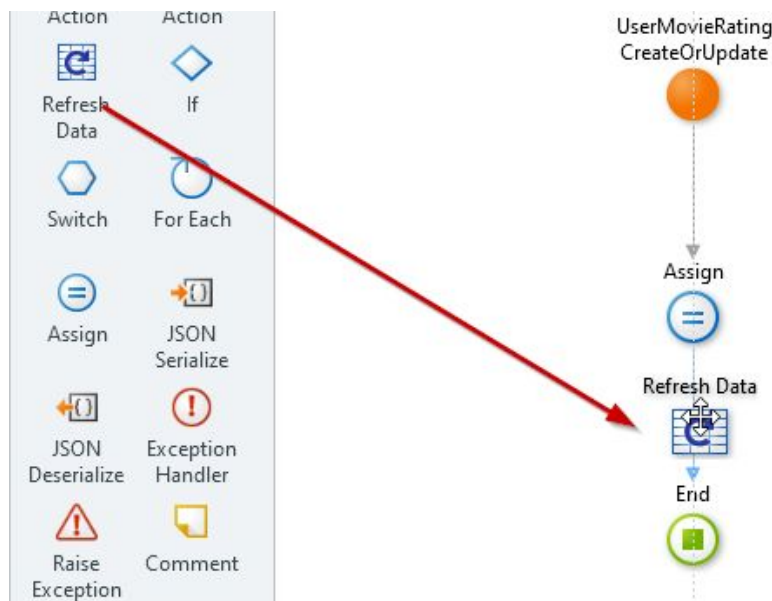
- e. Change the name of the Aggregate to *GetAverageRating*
- f. Open the MovieDetail Screen again, select the Block below the **Average Rating** text and change the **Rating** Input Parameter value to *GetAverageRating.List.Current.RatingAvg*



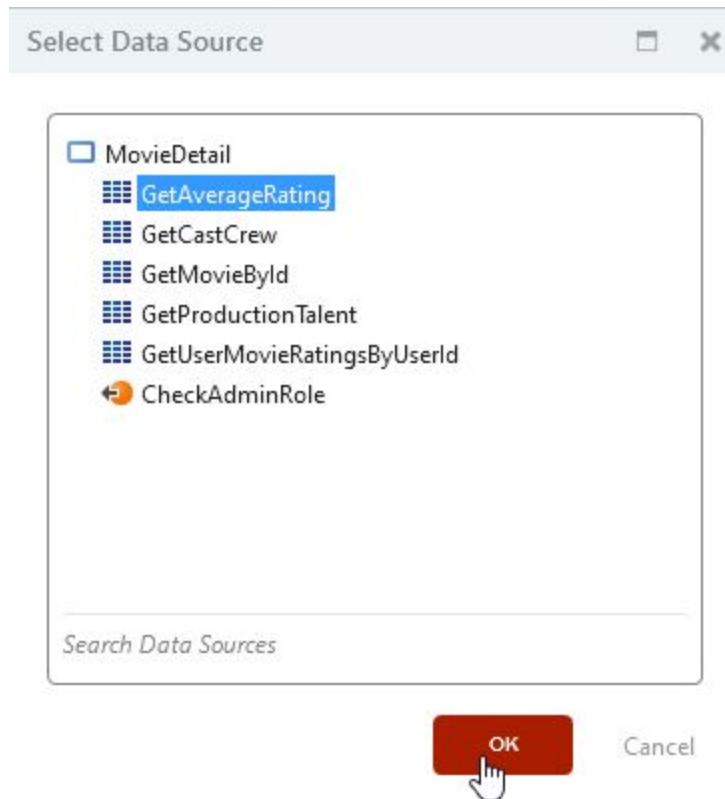
- g. Publish the module and open the application in the browser.
- h. Login with a different user from before and rate the same movie. Has the Average Rating changed? It shouldn't.

NOTE: For the same reason of the user rating, the Aggregate fetched the average rating at the time. With the new rating, we need to make sure the average is recalculated, so that the Block displays the correct number of stars.

- i. Open the StarClickedHandler Action. Drag a **Refresh Data** node and drop it below the Assign.



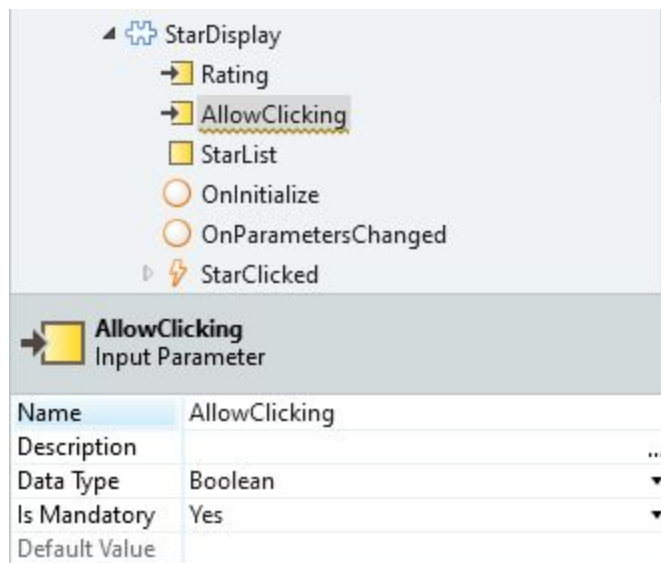
- j. In the new window, select the **GetAverageRating** Aggregate



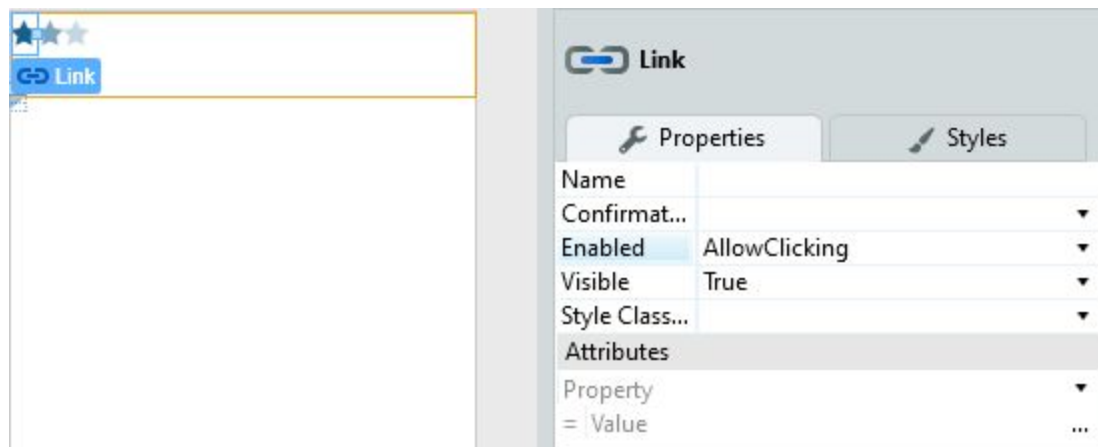
NOTE: The Refresh Data node executes an Aggregate on demand. This way, when the Event is triggered, the rating is saved in the database and the average rating is calculated again.

- k. Publish the module and repeat the same test as above. The Average Rating should work fine.
2. To ensure that only the Block for the user rating is clickable, we need an additional Input Parameter on the Block, *AllowClicking*. This will help us define when the Links on the stars are clickable or not. Then, for the user rating only registered users can click on the stars, while on the average rating no one should be able to click on the stars.

- a. Right-click on the StarDisplay Block and select **Add Input Parameter**. Name it *Allow Clicking* and make sure the **Data Type** is set to *Boolean*.



- b. Select the **Link** surrounding the colored star and set the **Enabled** property to *AllowClicking*.

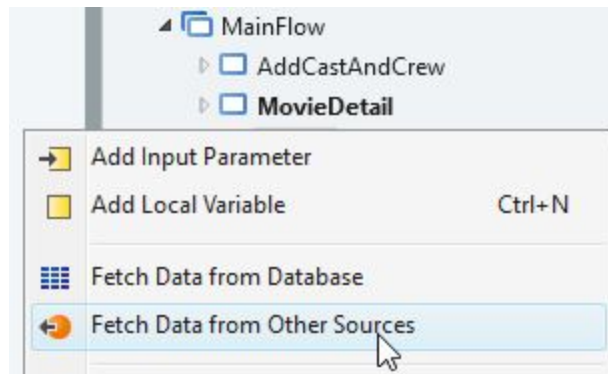


- c. Repeat the same process for the other Link (empty star).

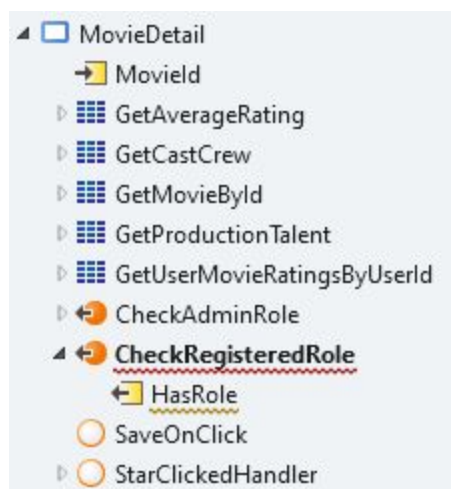
- d. Open the MovieDetail Screen and select the Average Rating Block. Set the **AllowClicking** property to *False*.



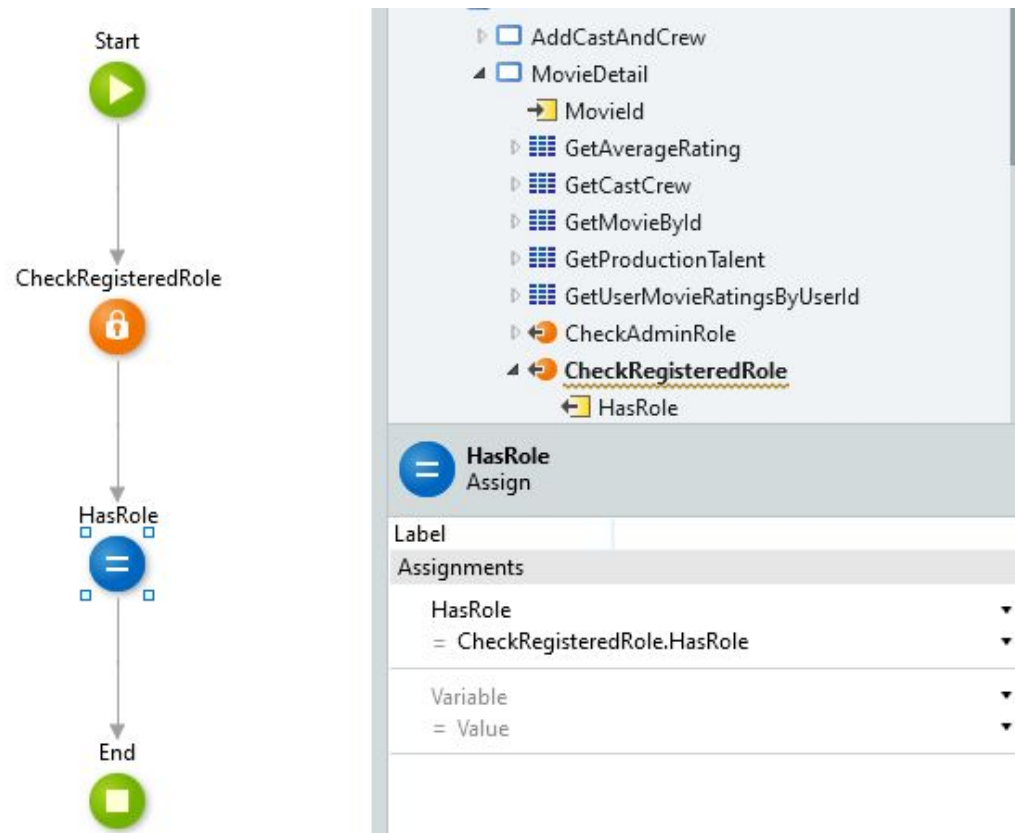
- e. For the user rating, we need a bit more, since we need to check if the currently logged in user is a registered user. Right-click on the MovieDetail Screen and select **Fetch Data from Other Sources**



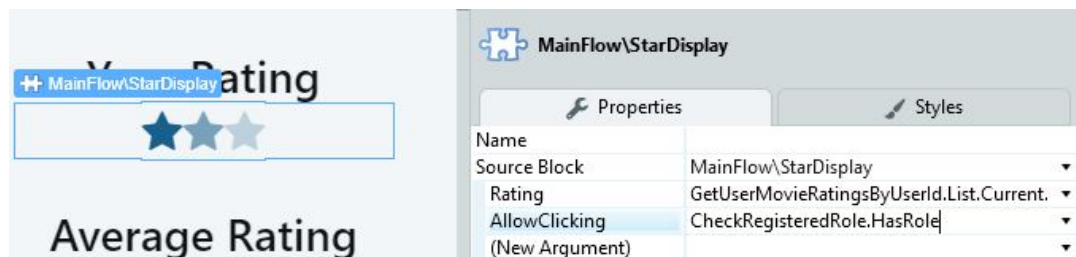
- f. Set the **Name** of the Data Action to *CheckRegisteredRole* and change the Output Parameter to *HasRole*, with **Data Type** set to *Boolean*.



- g. Define the logic of the Data Action to check if the user currently logged in has the Registered role and assign the HasRole output with the result of the CheckRole Action.



- h. Back on the MovieDetail Screen, select the User Rating Block and set the **Allow Clicking** input parameter to *CheckRegisteredRole.HasRole*



- i. Publish the module and test the application in the browser.