

Screen Interaction Exercise

Star Wars: The Force Awakens

Title *

Star Wars: The Force Awakens

Year *

2015

Plot Summary

Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.

Genre

-

Gross Takings

936627416

Is Available On DVD

☒

Save

[Back to list](#)

Table of Contents

Introduction	3
Add the Edit Movie feature	4
Add the New Movie feature	13
Testing the app: Add and Update movies	16
Add the Edit Person feature	18
Add the New Person feature	21
Testing the app: Add and Update People	22
Search and filter the list of movies	23
Testing the app: Searching and filtering	27
End of Lab	28

Introduction

In a previous Lab, we created the List and Detail Screens for the movies and people managed by our application. However, it was not possible to create new movies / people, or to edit existing ones.

In this Lab, we will add the functionality to create and update movies and people in the database. For that, we will add the functionality to the MovieDetail and PersonDetail Screens, where users can edit the respective Forms, click on a Button, and have the data saved in the database. Also, in the Movies and People Screen, we will add the option to create new movies and people, with Links that navigate to the respective Detail Screens, which will have empty Forms, ready to have new data being typed in.

Finally, we will implement a search filter in the Movies Screen, where we can filter the Movies in our database by title and plot summary.

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

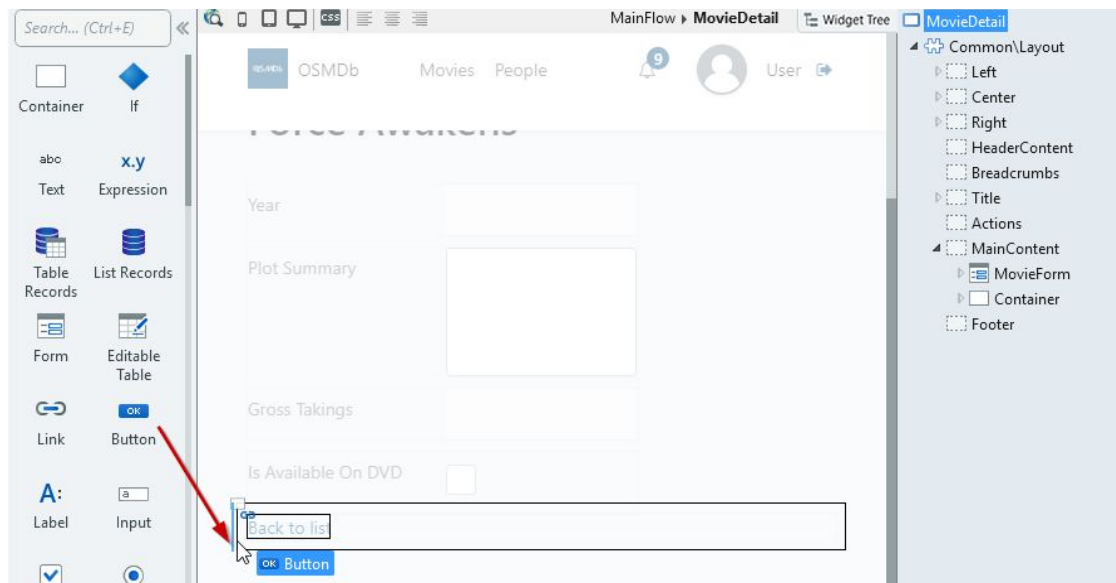
- Add buttons to Screens that execute Actions when clicked
- Add record creation (and update) logic to Screen Actions
- Add genre categorization to the **MovieDetail** Screen
- Create and update a few movies and people
- Create a search filter for movie title and plot summary

Add the Edit Movie feature

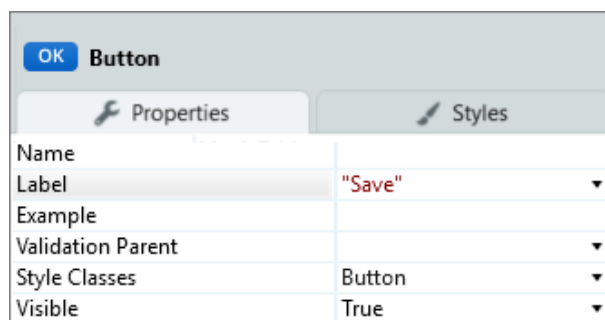
We will start this Lab by extending the **MovieDetail** Screen functionality. Here, we want to enable users to edit information of existing movies and save it in the database. By clicking on a Button, the end-user will trigger a Screen Action that will add the changes (made in the Movie Form) to the Movie Entity.

Also, we will add the possibility for an end-user to set the movie's genre in the Form, using the new attribute created in the previous Lab.

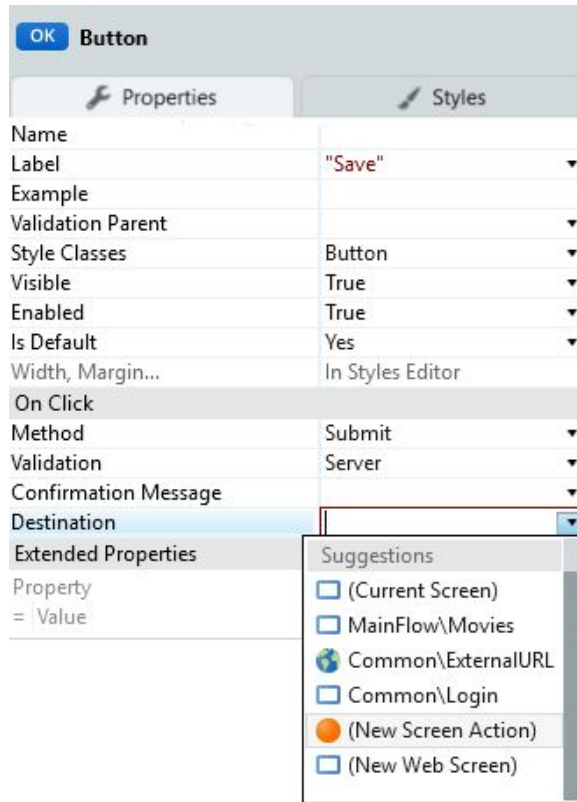
- 1) Add a Save Button to the **MovieDetail** Screen and configure it to execute a Screen Action on Click.
 - a) In the **MovieDetail** Screen, drag and drop a **Button** Widget to the left of the **Back to list** Link. Notice that when you are dragging the Widget, the Widget Tree appears on the right. We can use it to see the structure of the Screen, including all the Screen areas (Title, MainContent, ...) and the Widgets inside them.



- b) In the Properties editor, set the Button's **Label** property to "Save"

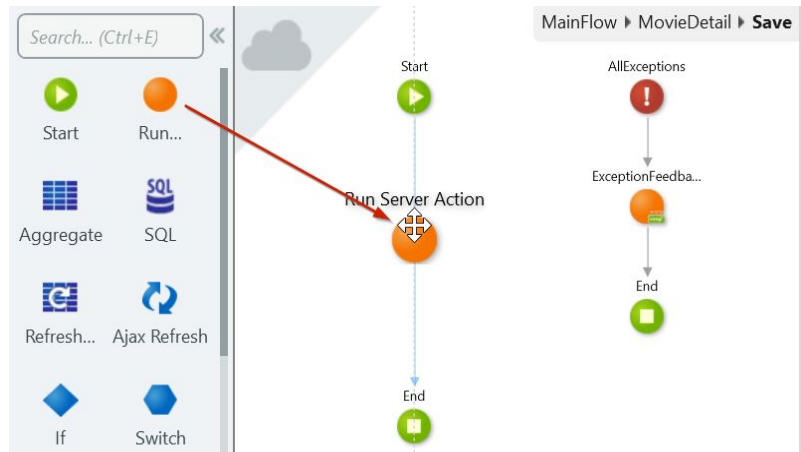


- c) Set the Button's **Destination** to *(New Screen Action)*. This creates a new Screen Action under the MovieDetail Screen. This way, when a user clicks the Save Button, its Destination is this Action, meaning that the Action runs.

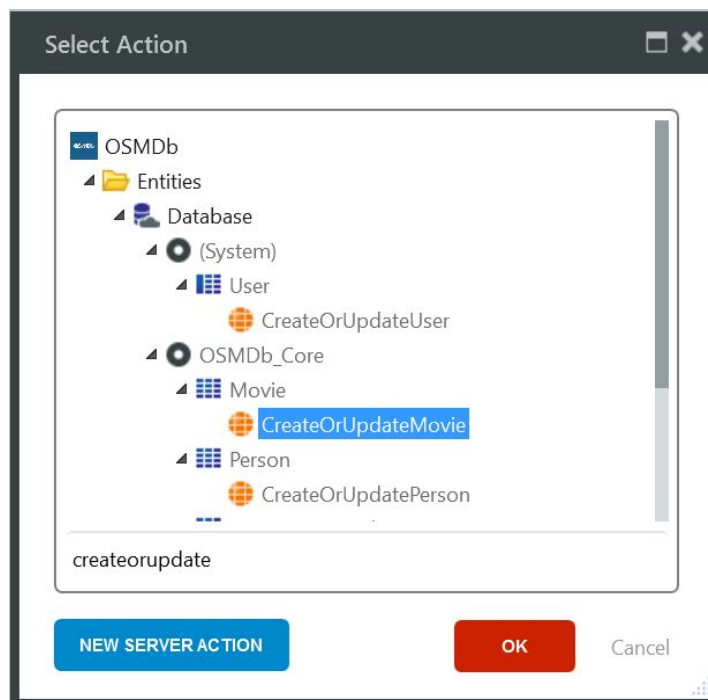


NOTE: When a Screen Action is created in this manner, OutSystems sets its name to match the **Label** property of the Button. When using this approach for a Link, OutSystems will use the Link's text for the Screen Action name.

- 2) In the **Save** Screen Action, update the movie information in the database and then redirect the end-user to the **Movies** Screen.
 - a) Ensure the **Save** Screen Action is opened in the main editor.
 - b) Drag and drop a **Run Server Action** between the Start and the End Node.

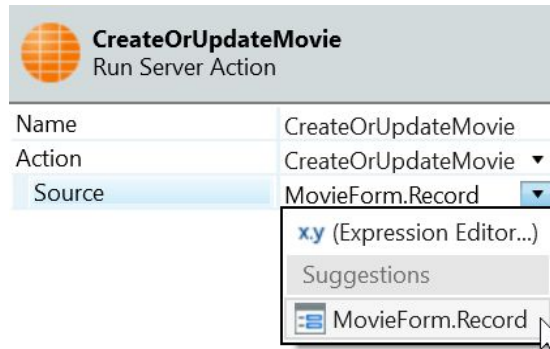


- c) In the **Select Action** dialog, type in *CreateOrUpdate*. Notice that the tree listing all the Actions available in this module start filtering as you type. Once **CreateOrUpdateMovie** comes into view, double-click to select it.



NOTE: CreateOrUpdateMovie is one of the six Entity Actions automatically created by OutSystems with every Entity. The CreateOrUpdate Entity Action will create a new record or update an existing one, depending whether the record passed in (as an argument) has an Identifier already set or not.

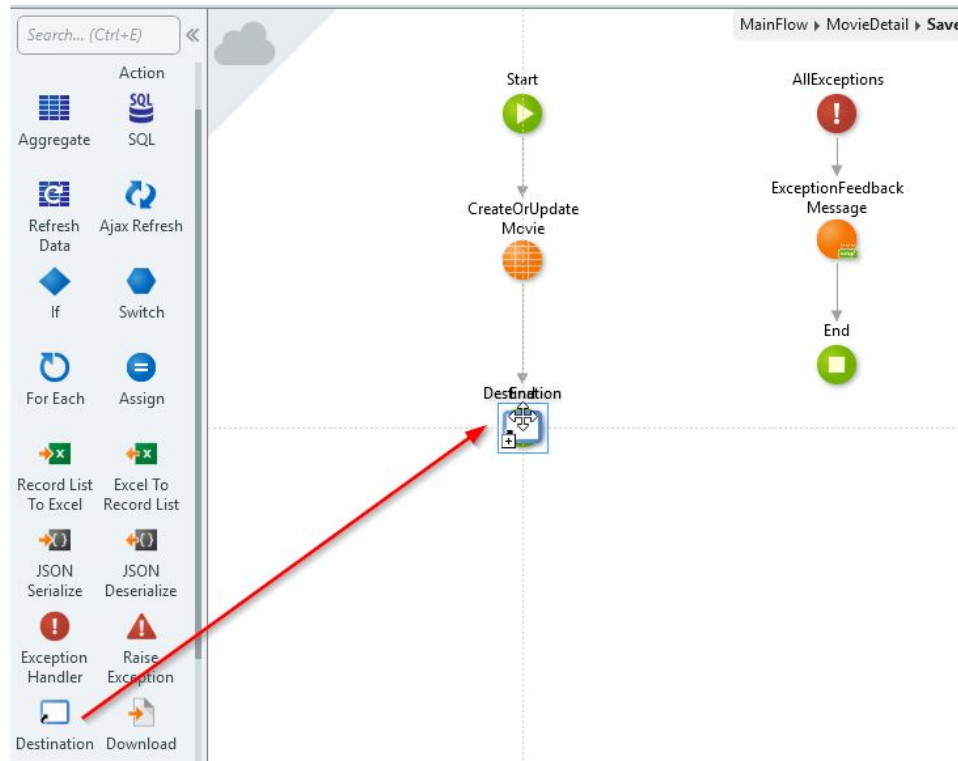
- d) In the Properties editor, set the **Source** argument for the **CreateOrUpdateMovie** Action to *MovieForm.Record*. This guarantees that the movie being added to the database, is the one defined in the Form of the MovieDetail Screen (MovieForm).



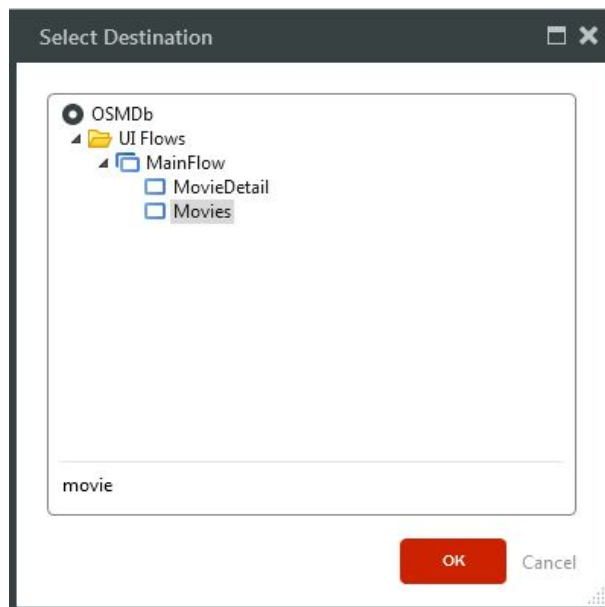
NOTE: To recap the flow of data in the MovieDetail Screen:

- 1) In the **Preparation**, the movie to be displayed is fetched from the database, using an Aggregate (and the Input Parameter **MovieId**).
 - 2) In the Screen, the Form Widget takes the output of the Aggregate as the Source Record and presents it to the user.
 - 3) The user changes one or more of the values in the Form inputs and clicks on the **Save** Button, submitting changes to the server.
 - 4) The **Save** Screen Action updates (or creates) the record in the database, using the value from the Form, which contains the movie.
-

- e) Drag and drop a **Destination** over the existing End statement in the main editor. This replaces the End for the **Destination** statement.

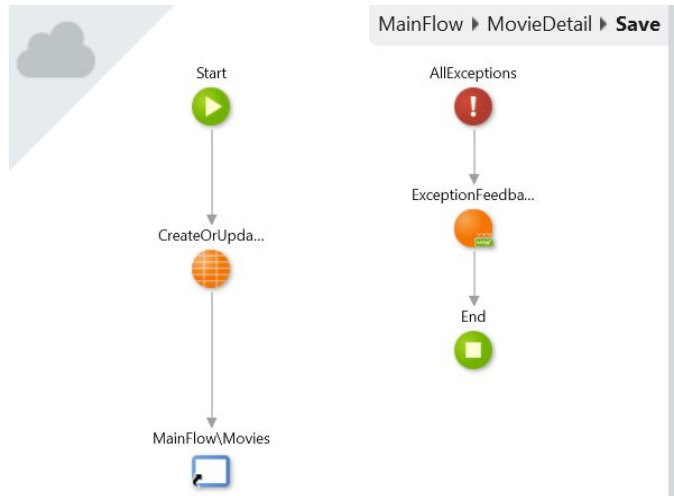


- f) In the **Select Destination** dialog, type in *movie*. Notice that the tree listing all the Screens available in this module start filtering as you type. Double-click the **Movies** Screen to select it.



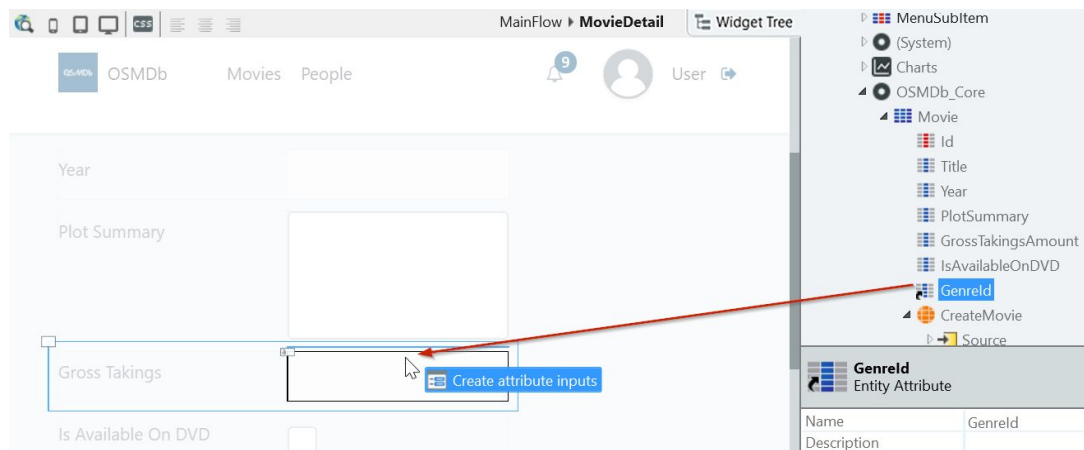
NOTE: Screen Actions by default terminate in an End statement. This causes execution to go back to the current Screen's **Preparation** and then the Screen is re-built. By replacing the End with a **Destination**, the server will redirect the end-user to the specified Screen.

g) The Save Screen Action should look like this



3) Add the **Genrelid** and **Title** attributes to the **MovieDetail** Form. Place the Genre right after the Plot Summary and the Title at the beginning, before Year.

- Open the **MovieDetail** Screen.
- In the **Data** tab, expand the **Movie** Entity and drag the **Genrelid** attribute over the **MovieForm** in the main editor. Drop the attribute between the rows containing the **Plot Summary** and the **Gross Takings**.



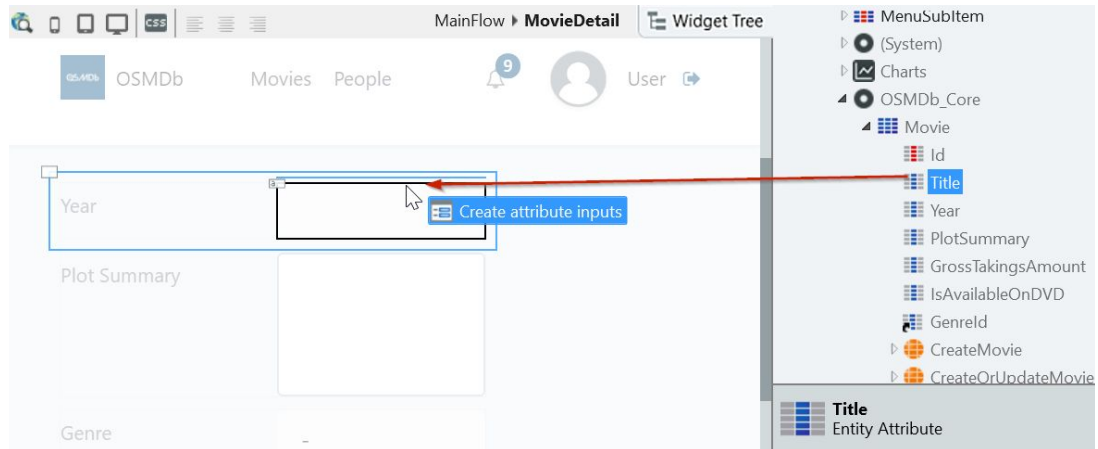
- c) Select the **Input** that was automatically added to the Form for this attribute.

The screenshot shows a form with four fields: Year, Plot Summary, Genre, and Gross Takings. The Genre field is highlighted with a red box. A dropdown menu is open for the Genre field, showing a list of options. The first option is 'Movie_GenreId'.

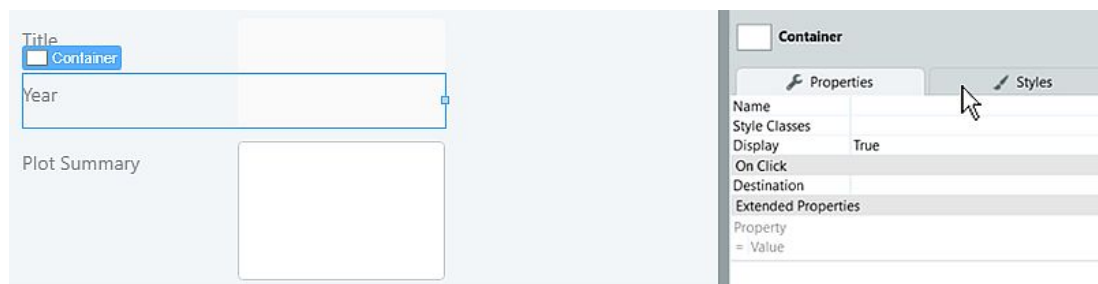
Notice that the Input for **Genre** is a **Combo Box**. The **Source Attribute** chosen from a given **Source Entity** provides the list of possible options the user can select from. In this case, the **Label** of the **MovieGenre** Entity will be displayed. The **Variable** that will hold the selected value in the Combo Box is *MovieForm.Record.Movie.GenreId*, just like in the other Inputs in the Form.

Movie_GenreId	
Properties	
Name	Movie_GenreId
Variable	MovieForm.Record.Movie.GenreId
Validation Parent	
Mandatory	False
Source Record List	
Source Entity	MovieGenre
Source Attribute	Label
Source Identifier Attribute	
Special Variable	
Style Classes	select

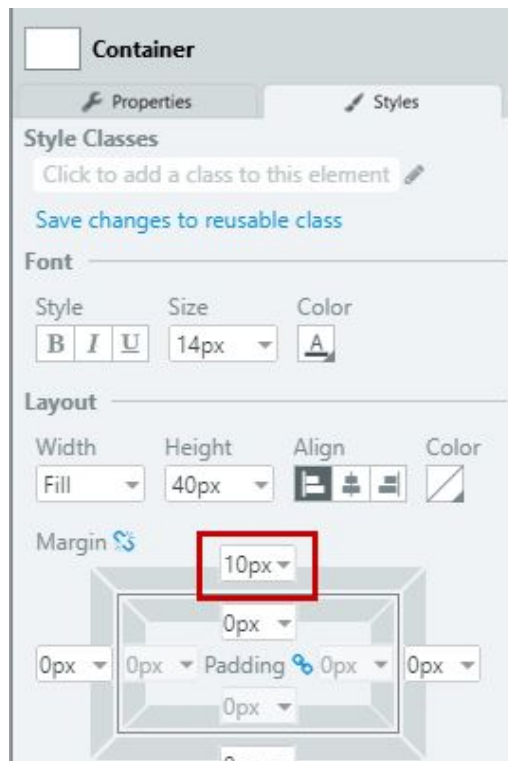
- d) Drag the **Title** attribute (from the **Movie** Entity) and drop it at the very top of the **MovieForm**, before the row containing the **Year**.



- e) You may notice that the Title Input does not have a spacing to the Year. Let's fix that, by adding a margin top. Select the Container surrounding the Year Label and Input and switch to the Styles Editor.



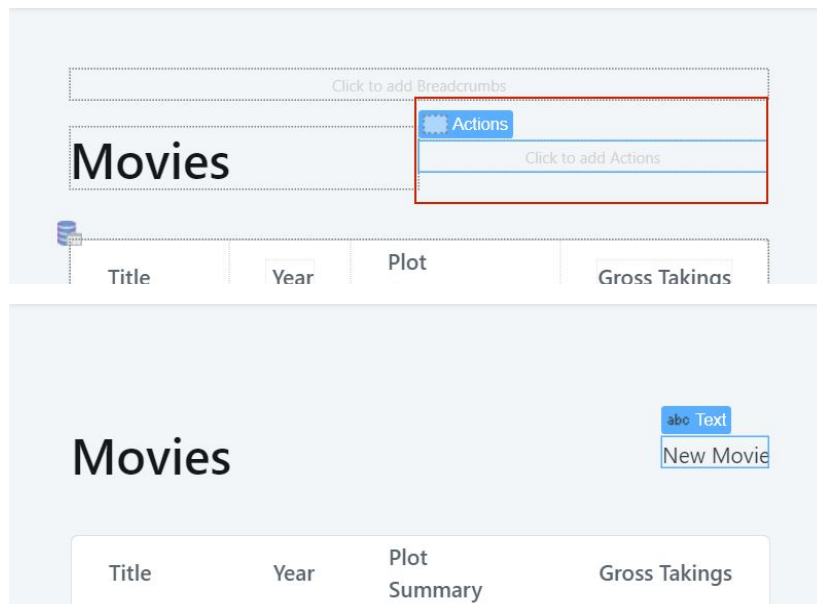
- f) Set the **Margin Top** to *10 px*.



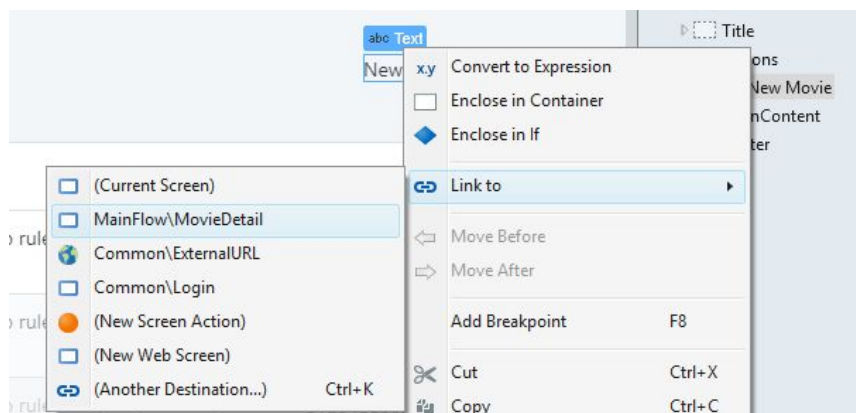
Add the New Movie feature

In this part of the exercise, we will keep developing the **MovieDetail** Screen. Here, we want to enable end-users to add a new movie to the OSMDb application. For that, we need a Link in the Movies Screen that redirect users to the MovieDetail Screen. In there, users will be able to add a new movie, using the Form.

- 1) Create a *Add Movie* Link to **Movies**. Configure it to navigate to the **MovieDetail** Screen, with the goal of creating a new movie.
 - a) Open the **Movies** Screen.
 - b) In the canvas, select the **Actions** area and type in *New Movie*.

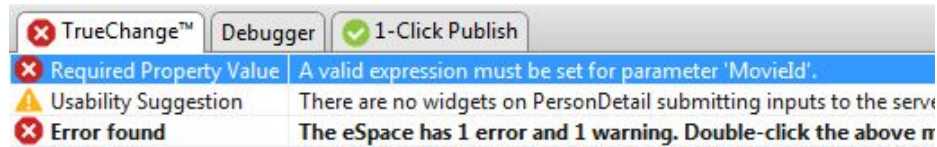


- c) Right-click the **New Movie** Text and link it to *MainFlow\MovieDetail*.

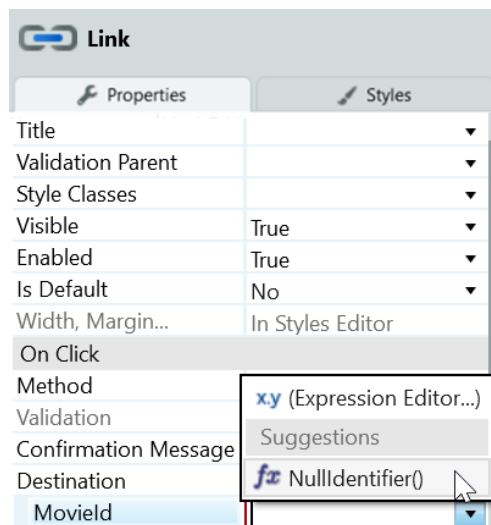


NOTE: Since the current **Movies** Screen is already linked to **MovieDetail** (for the edit movie functionality), this Screen now appears in the shortlist of destinations.

- d) Notice that the **TrueChange** icon turns red to indicate there is an error. The **MovieDetail** Screen has a **Mandatory** input parameter that must be filled in when navigating to it. Double-click the **Required Property Value** error.



- e) Since we want to create a new movie, which does not have any Id yet, In the **MovieId** Input Parameter dropdown, select the option *NullIdentifier()*



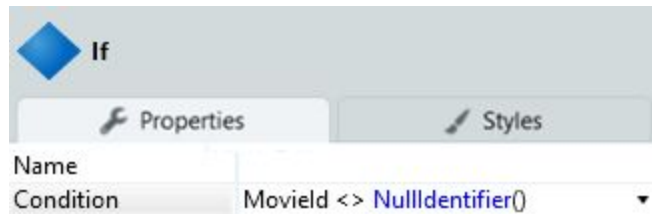
NOTE: When the application navigates to **MovieDetail** Screen using the **Movie Title Link**, the value passed as input parameter is the Identifier of the movie clicked. As a result, in the MovieDetail Screen, the movie record is fetched in its Preparation and displayed (for showing and editing).

When it navigates from the **New Movie Link**, the value passed as the input parameter is the *NullIdentifier()*. As a result, in the MovieDetail Screen, no record will be fetched in its Preparation (there are no movies with null identifier in the database), and therefore, there is no data displayed in the Form (a user can add a new movie).

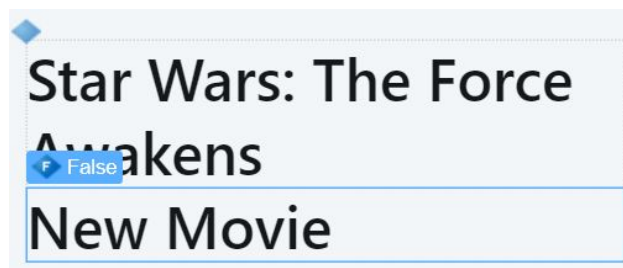
- 2) Make the **MovieDetail** Screen title display *New Movie*, when adding a new movie.
 - a) Open the **MovieDetail** Screen.
 - b) Right-click the **Expression** in the **Title** placeholder of the Screen and select *Enclose in If*.
 - c) Set the **Condition** property for this new **If** to:

MovieId <> NullIdentifier()

With this If, the name of the movie will only appear as the Screen Title, when the MovieId Input Parameter is not *NullIdentifier()*.



- d) In the canvas, select the **False** branch of the previously created If. Then, type in *New Movie*. This will be the title displayed when the user is creating a new movie (*MovieId = NullIdentifier()*).



- e) Publish the OSMDb module using the **1-Click Publish** button.

Testing the app: Add and Update movies

It's time to test our app! We will create a new movie and modify the information of an already existing one.

- 1) Add a new movie to the database. Feel free to find real data about your favorite movies online. Remember that testing with real data is always better! We will do it with the movie Fight Club from 1999.
 - a) Open your application in the browser.
 - b) Click the **New Movie** Link.
 - c) Fill the Form with data from a movie and then press **Save**.

New Movie

Title *

Year *

Plot Summary

Genre

Gross Takings

Is Available On DVD ☒

[Save](#) [Back to list](#)

- d) The new movie should appear in the list of movies.
- 2) Update the information about an existing movie. As a suggestion, change the Gross Takings of Star Wars: The Force Awakens. After editing, confirm that the operation finished successfully in the Movies table.
 - a) Click the *Star Wars: The Force Awakens* movie title Link.

- b) In the details page set the **Gross Takings** to 936627416, and then **Save**.

Star Wars: The Force Awakens

Title *	Star Wars: The Force Awakens
Year *	2015
Plot Summary	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.
Genre	-
Gross Takings	936627416
Is Available On DVD	<input checked="" type="checkbox"/>

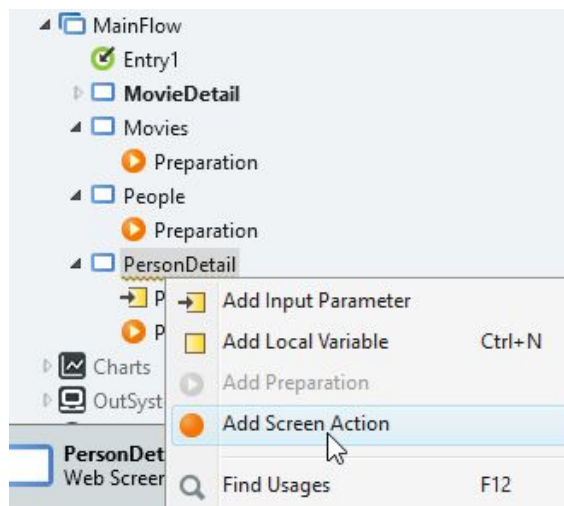
[Save](#) [Back to list](#)

- c) Confirm that the **Gross Takings** of the *Star Wars: The Force Awakens*, in the movie table, changed to the new value that was introduced.

Add the Edit Person feature

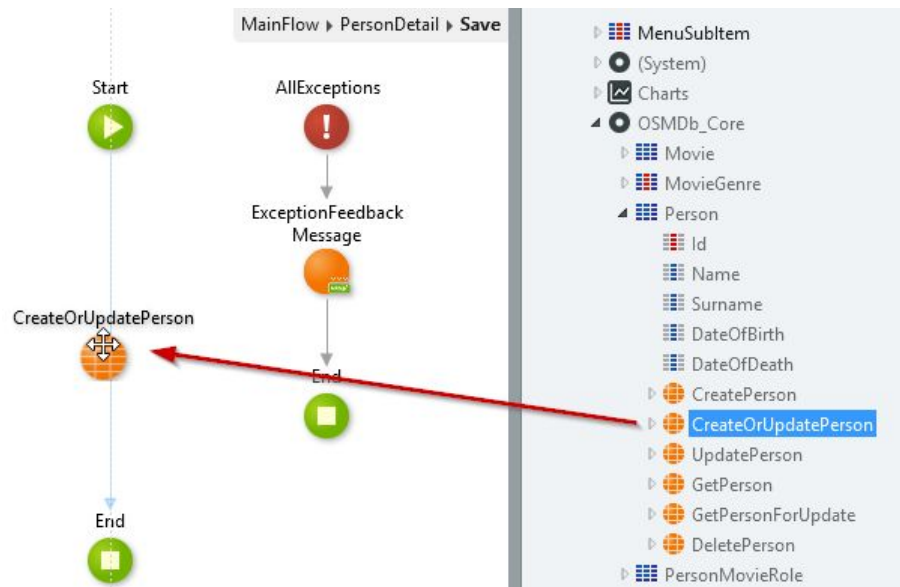
Now that we are able to add and edit movies, we want to do the same for the People, in the **PersonDetail** Screen. We will do exactly the same things we did for movies, but at some stages we will do it in a different way, so that we can see all the options available to build what we need in OutSystems.

- 1) Create a new Screen Action under **PersonDetail** Screen with the name *Save*.
 - a) Switch to the **Interface** tab, right-click the **PersonDetail** Screen and select *Add Screen Action*.

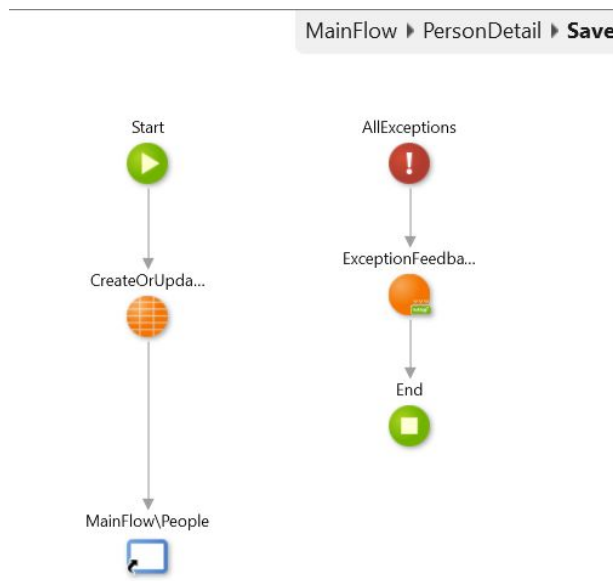


- b) Set the Action **Name** to *Save*.
 - 2) In the **Save** Screen Action, update the person information in the database. At the end, redirect the user to the **People** Screen. This step is very similar to what we did for saving (or updating) a new movie in the database, so it follows the same logic.
 - a) Ensure that you have the **Save** Action (from the **PersonDetail** Screen) open.
 - b) Switch to the **Data** tab and expand the **Person** Entity.

- c) Drag the Entity Action **CreateOrUpdatePerson** on to the connection between the Start and the End.

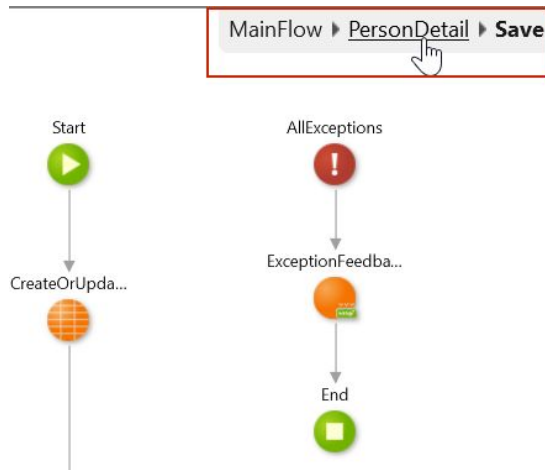


- d) In the Properties editor, set the **Source** argument for the **CreateOrUpdatePerson** Action to *PersonForm.Record*
- e) Drag and drop a **Destination** over the End node (the one below the **CreateOrUpdatePerson** Action). This action should replace the End statement with the dragged Destination.
- f) In the **Select Destination** dialog, double-click the **People** Screen to select it. The Save Screen Action should look like this



3) Add a **Save** Button to the **PersonDetail** Screen. Configure it to execute the **Save** Screen Action on Click.

a) Open the **PersonDetail** Screen using the breadcrumbs.



- b) Drag and drop a **Button** to the left of the **Back to list** Link, inside the Container.
- c) In the properties editor, set the Button's **Label** property to **Save**.
- d) Set the Button **Destination** to the **Save** Screen Action. The **PersonDetail** Screen should look like this

The screenshot shows the **PersonDetail** screen. At the top, the name **Harrison Ford** is displayed in a large font. Below this, there are four input fields: **Name**, **Surname**, **Date Of Birth**, and **Date Of Death**. The **Date Of Birth** and **Date Of Death** fields have calendar icons next to them. At the bottom left, there is a blue button labeled **Save**. To the right of the **Save** button is a link labeled **Back to list**.

Add the New Person feature

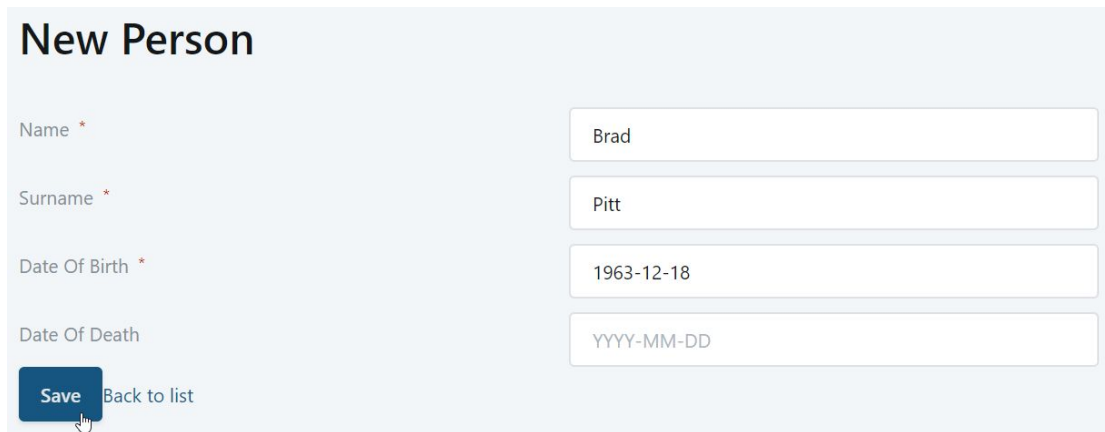
Similarly to what was done with movies, we will now update the People Screen to enable the users to add new people to the database.

- 1) Add a *New Person* Link to **People**. Configure it to navigate to the **PersonDetail** Screen with the behavior for creating a new person.
 - a) Open the **People** Screen.
 - b) Type in *New Person* inside the **Actions** area of the PeopleScreen.
 - c) Right-click the **New Person** text, and **Link** it to *MainFlow\PersonDetail*.
 - d) Set to *NullIdentifier()* the value to be passed to the **PersonId** parameter.
- 2) Make the **PersonDetail** Screen Title display *New Person*, when adding a new person to the database. When editing a Person, we will keep using its full name.
 - a) Open the **PersonDetail** Screen.
 - b) Enclose the Expression in the **Title** area in an **If**, setting its **Condition** to:
PersonId <> NullIdentifier()
 - c) In the **False** part of the If, type in *New Person*.
 - d) Ensure the **Title** placeholder of this Screen was implemented similarly to the Title in **MovieDetail**.
 - e) Publish the module using the **1-Click Publish** button.

Testing the app: Add and Update People

Let's test our app, this time for the new People functionality we just added. We will create a couple of new people and modify the information of a few of the existing ones.

- 1) Add a new person to the database. As an example, we will add *Brad Pitt*.
 - a) Open your application in the browser.
 - b) Navigate to the **People** Screen, and click the **New Person** Link.
 - c) Fill the Form with data from the Person, and then press **Save**.

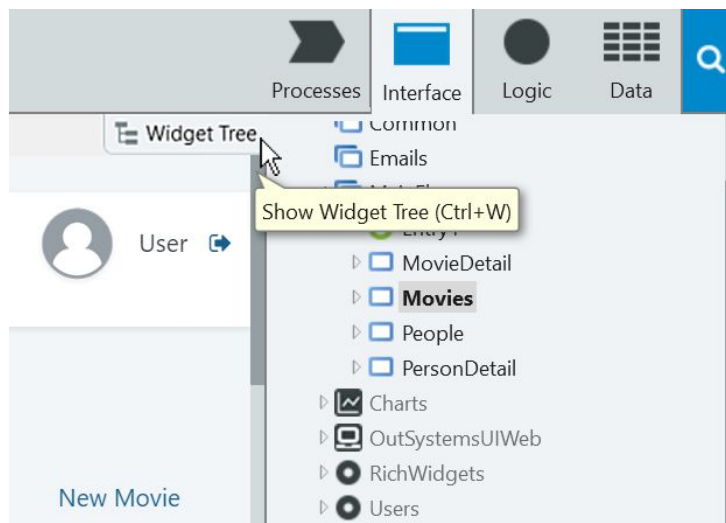


- d) The new person should now appear in the **People** Screen.
- 2) Update the details of a person in the database. As an example, we will add the Date of Death for Carrie Fisher.
 - a) Navigate to the details Screen of *Carrie Fisher*.
 - b) Set the **Date of Death** to *2016-12-27* and click **Save**.
 - c) Confirm that the new **Date of Death** appears in the 'Carrie Fisher' details Screen.

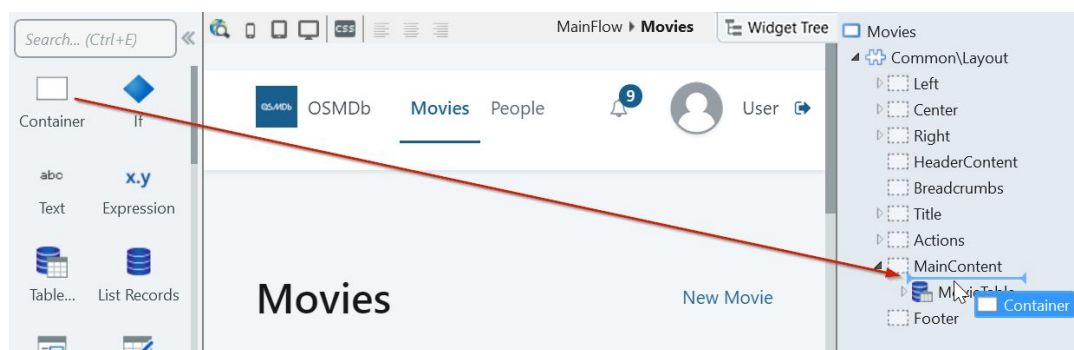
Search and filter the list of movies

Let's now add a Search filter for the Movies. We will add a filter area in the **Movies** Screen, which will allow searching the movie list for the movie title and plot summary.

- 1) Add a **Container** at the top of the **Movies** Screen and inside of it add an **Input**. This Input will be used to search for the movie title and plot summary. The Input Widget should be associated with a **Local Variable** of the appropriate type, that will save the value typed by the user of the app.
 - a) Open the **Movies** Screen.
 - b) Open the **Widget Tree** by clicking the **Widget Tree** icon in the upper right corner of the canvas area. This shows the structure and placement of all the Screen elements.

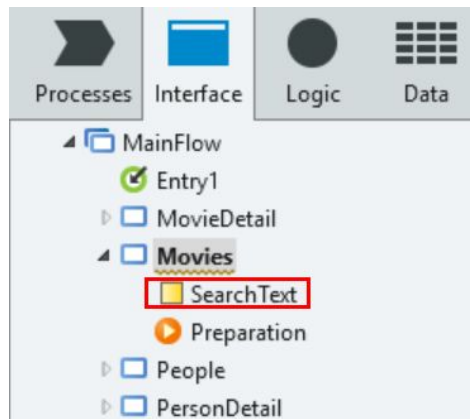


- c) Expand the **MainContent** area using the triangle symbol.
 - d) Drag a **Container** just above the **MovieTable**.



- e) Drag and drop an **Input Widget** in the previous container.

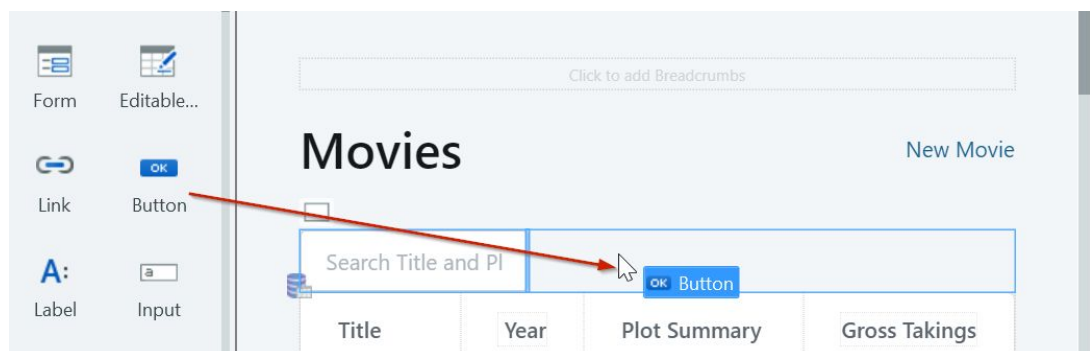
- f) Click again on the widget tree icon, right-click the **Movie** Screen and select *Add Local Variable* and name the Local Variable as *SearchText*. Ensure its **Data Type** is *Text*.



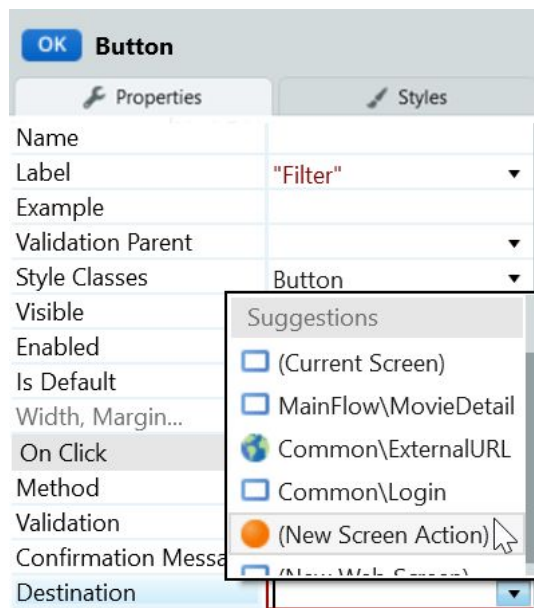
- g) Select the **Input** Widget created above. Set its **Variable** property to *SearchText* and **Prompt** to "Search Title and Plot". By setting the Variable property, it will automatically bind the input to the **SearchText** Local Variable, meaning that the Variable will hold what the end-user types in the Input.

NOTE: The **Prompt** property specifies what text should appear (greyed out) in an Input, while it has no content. It's normally used to explain to the user what should be typed into that input.

- 2) Add a Filter **Button** inside the Container with the Input filter. Pressing this Button should influence the movies being returned by the **GetMovies** Aggregate, in the face of the text typed in the Input. As an example, searching for "star wars" would display two movies in the Table Records: The Force Awakens and The Last Jedi.
- a) Drag and drop a **Button** immediately to the right of the input box.



- b) Change the Button's **Label** to *Filter*. For the **Destination**, select (*New Screen Action*). This will create a Screen Action called **Filter**.



NOTE: While you may expect to be working on the **Filter** Screen Action, the next steps actually involve changes to the Preparation.

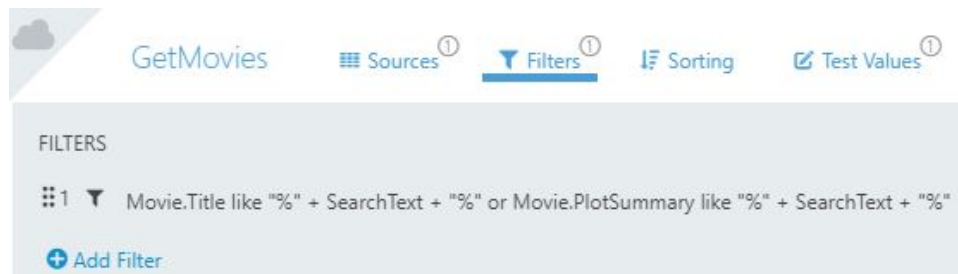
When doing the filter, it is in fact the data that is fetched that is going to be affected by it. As seen earlier regarding the lifecycle of Screens, the data sources for Widgets need to exist before the Widgets. In this case, the source of the Table Records is the **GetMovies** Aggregate in the Preparation.

Although the **Filter** Screen Action will have no logic, the Filter's flow ends with the End statement, which will make the Movies' Preparation to re-run and the Screen to re-render. On this second run, the **Search** Local Variable will have the value typed by the user and will be used to filter the **GetMovies** Aggregate, which will then affect what is visible on the Screen.

- c) Open the **Movies** Preparation. Double-click the **GetMovies** Aggregate to open it in the main editor.
- d) In the **Filters** tab, press **+Add Filter** and add the following condition:
- Movie.Title like "%" + SearchText + "%"*
or Movie.PlotSummary like "%" + SearchText + "%"

NOTE: The LIKE operator, combined with "%", allows the detection of substrings. The condition above will filter the Aggregate's results to have only the movies where the **Title**, or the **Plot Summary**, includes anywhere the **SearchText** Local Variable value in it. As an example, "tar" would need to include in the output, at least, the Star Wars movies in the database.

- e) The **Filters** area of the **GetMovies** Aggregate should look like this



- f) Publish the OSMDb module using the **1-Click Publish** button.

Testing the app: Searching and filtering

Now that the filter is implemented, let's test it to confirm it works. Search for a part of a movie title or plot summary and make sure that the expected movies appear. As a suggestion, we will use the keywords "Star" and "life".

- 1) When no filters are applied, make sure that all of the movies appear in the respective Table Records.
- 2) Search for **Star** in the text input box and guarantee that at least the following movies appear in the Table Records (this may vary depending on the movies you added):

Star Wars: The Force Awakens

Star Wars: The Last Jedi

- 3) Search for **life** in the text input box and guarantee that at least the following movies appear in the Table Records:

Along Came Polly

End of Lab

In this lab, we added the functionality to add / update movies and people in the application. On both Screen details, we added a Button that triggers an Action to create or update a Movie or a Person respectively.

On the Movies and People Screen, we defined new Links to create new movies and new people respectively. The Link navigates to the respective Detail Screens, passing a Null Identifier as parameter. Since the database does not have any record with Null Identifier, the Forms appear empty and ready to be filled with the information about the new movie and new person.

Finally, we added a search filter to the Movies Screen, to search for movie titles and plot summary. For that, we changed the Aggregate in the Movies Screen Preparation, to filter the query to the database.