

Search Filters

Exercise

Outline	1
How to	1
Adding Search Input Fields	1
Applying the Search	8

Outline

In this exercise, we will expand the functionality of the app with a new filter functionality on the Movies Screen. We want our users to be able to search by movie title, plot summary, and movie genre and we want to make sure that when we change to a different Screen and come back, the search filter is not lost. The search filter should look like this:

Movies

Select a Movie Genre

▼

Title ↕	Year ↕	Plot Summary ↕	Gross Takings ↕
Star Wars: The Force Awakens	2015	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.	\$815,843,529.00
Raiders of the Lost Ark	1981	Archaeologist and adventurer Indiana Jones is hired by the US government to find the Ark of the Covenant before the Nazis.	\$242,374,454.00
Schindler's List	1993	In Poland during World War II, Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.	\$16,439,233.00

Note that the search should be done automatically when the users finished typing the title/plot summary keyword, or when they just chose the movie genre. So, we don't want a button/link to trigger the search. Also, the end-user does not need to type the whole movie title and the whole plot summary, for the search to return the movies that match the keyword that was typed.

This needs to be done in two steps:

- 1) The UI part, where we need to add input fields to allow searching for the keywords
- 2) The logic part, where we need to adjust the Aggregate that fetches the movies, save the search filters in variables that persist even when we change Screens, and trigger the search logic automatically. For this, we will use Client Variables.

How to

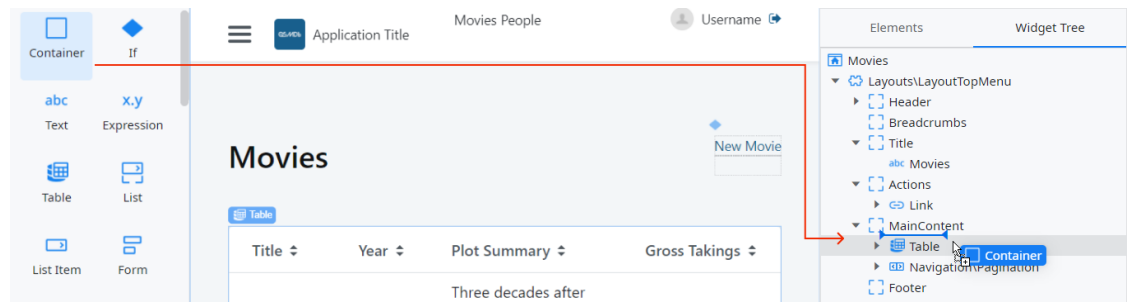
In this section, we'll describe exercise 9 - *Search Filters*, step by step.

Adding Search Input Fields

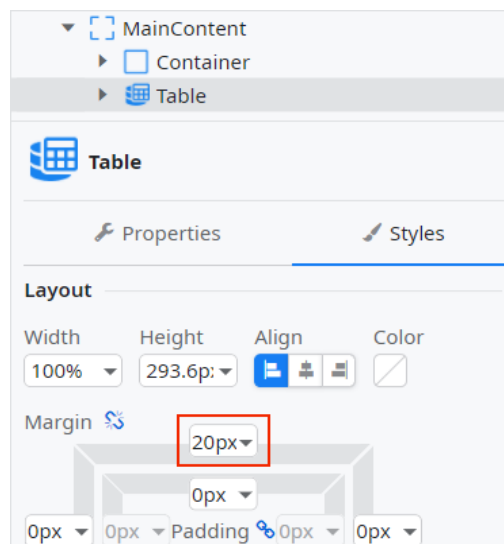
To allow searching for movies by title/plot summary and movie genre, we need to add input fields to the Screen where the end-user can define the desired search filters.

1. Add two input widgets (one **Input** and one **Dropdown**) above the Movies table.

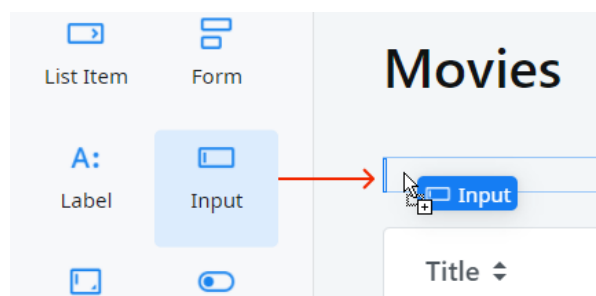
- a. Start by finding space on the Screen for the search widgets. In the Movies Screen, open the **Widget Tree**, expand the **MainContent** and drag a **Container** right above the Table.



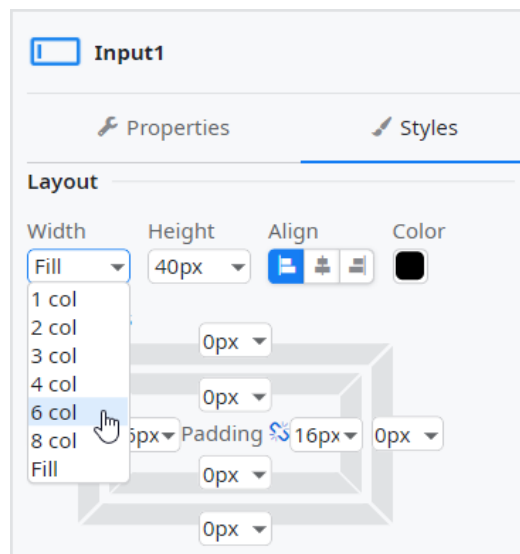
- b. Select the **Table**, switch from the Properties to the **Styles** view, and add a margin-top of 20px.



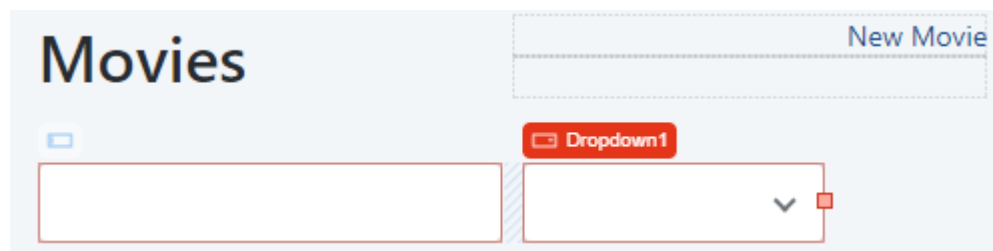
- c. Drag an **Input** widget and drop it inside the Container.



- d. Change the **Width** of the Input to 6 *columns* to just use half of the screen's width.



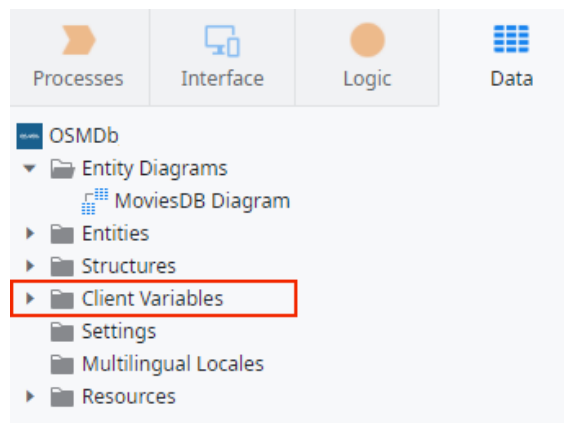
- e. Drag a **Dropdown** widget and drop it right next to the Input widget.
- f. Set its **Width** to 4 *columns*. *The widgets should appear side by side now.*



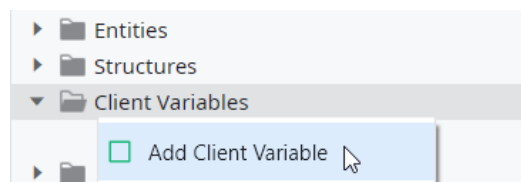
Note: At this point, the two widgets have errors. Don't worry, they will be fixed in the next steps.

2. Now that we have the two input fields on the Screen, we need to define the variables that will store the search filters, meaning the values that the end-users insert in the input fields. We will use Client Variables to make sure that the search filter is not lost when we change to a different Screen and come back to the Movies Screen.

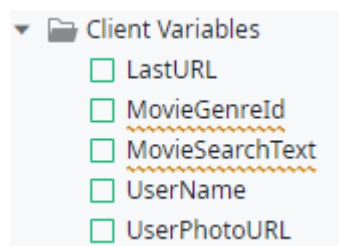
- a. Switch to the Data tab in **ODC Studio** and locate the **Client Variables** folder.



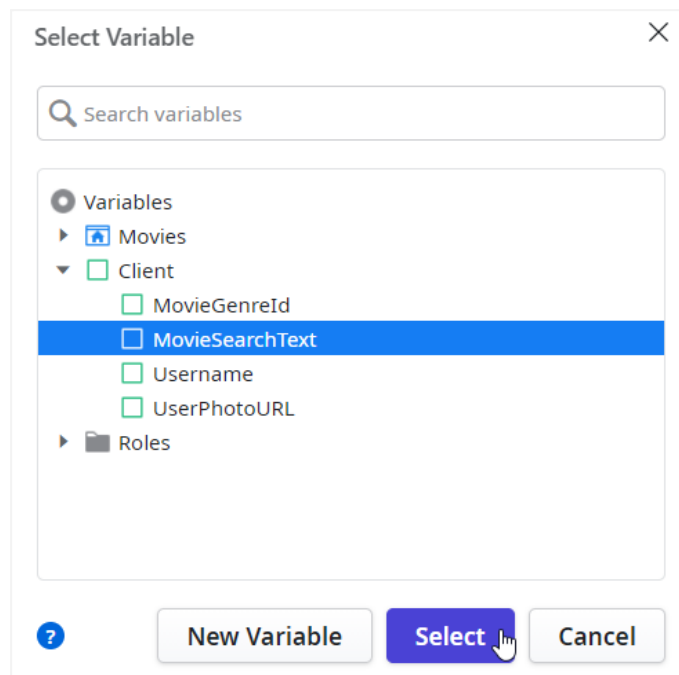
- b. Right-click on the **Client Variables** folder and select **Add Client Variable**



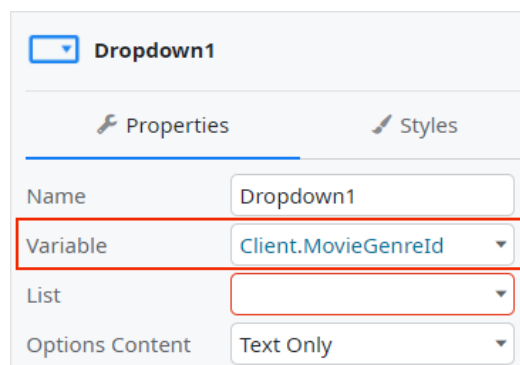
- c. Set the **Name** of the Variable to *MovieSearchText* and make sure the **Data Type** is set to *Text*.
- d. Repeat the previous two steps, but this time for a Client Variable named *MovieGenreId* of **Data Type** *MovieGenre Identifier*.



- e. Go back to the Movies Screen, select the Input field, make sure you see its Properties (and not the Styles), click on the **Variable** property, then on **Select Variable...**, and chose the Client Variable *MovieSearchText*.



- f. Select the **Dropdown** and set the **Variable** property to *Client.MovieGenreId*



As you see, they can be used on widgets just like a Screen's Local Variable.

3. Now we need to finish the implementation of the input widgets to remove the errors and also help the end-user understand which search filters are allowed.

- a. Select the Input widget and set the **Prompt** property to "(Search for Text or Plot Summary)".

The screenshot shows the 'Input1' widget configuration panel. It has two tabs: 'Properties' (selected) and 'Styles'. The 'Properties' tab contains the following settings:

Property	Value
Name	Input1
Variable	Client.MovieSearchText
Prompt	"(Search for Text or Plot Summary)"
Input Type	Search
Max. Length	50
Mandatory	False
Enabled	True
Style Classes	"form-control"

Don't forget that these visual cues are very important to the end-user of the app!
Remember to always put yourself in the user's shoes.

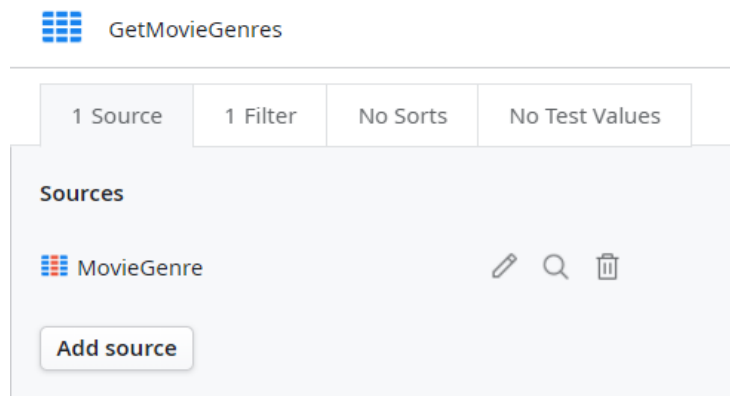
- b. Select the **Dropdown** widget. Notice that there are still several errors we need to solve. The first one is to set what would be the List of values that the dropdown will display.

The screenshot shows the 'Dropdown1' widget configuration panel. It has two tabs: 'Properties' (selected) and 'Styles'. The 'Properties' tab contains the following settings:

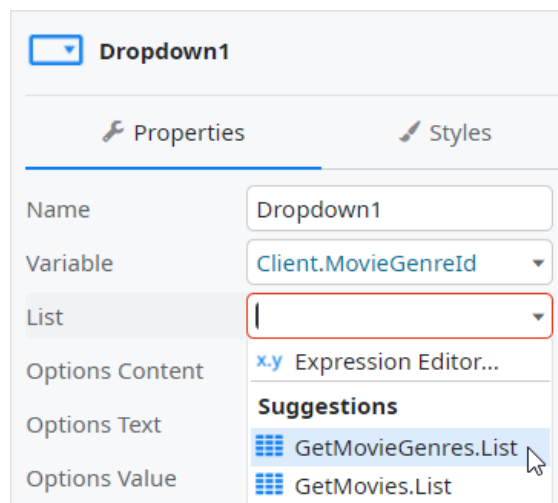
Property	Value
Name	Dropdown1
Variable	Client.MovieGenreId
List	
Options Content	Text Only
Options Text	
Options Value	
Mandatory	False
Enabled	True
Empty Text	
Style Classes	"dropdown"

A red rectangular box highlights the 'List', 'Options Content', 'Options Text', and 'Options Value' fields, indicating they need to be configured.

- c. Add an **Aggregate** to the Screen and select the **MovieGenre** Entity as the single Source.



- d. Back on the Screen preview, set the **List** property of the Dropdown widget to the output of the Aggregate: *GetMovieGenres.List*. All the errors will be automatically fixed.



So, by recognizing the Source of the Aggregate, OutSystems infers that we want to save the Identifier of the movie genre selected and we want to display the genre's label to identify the movie genre. Which is exactly what we want!

- e. To finish, set the **Empty Text** to *"Select a Movie Genre"*.

The screenshot shows the 'Properties' tab of a widget named 'Dropdown1'. The properties are as follows:

Property	Value
Name	Dropdown1
Variable	Client.MovieGenreId
List	GetMovieGenres.List
Options Content	Text Only
Options Text	MovieGenre.Label
Options Value	MovieGenre.Id
Mandatory	False
Enabled	True
Empty Text	"Select a Movie Genre"
Style Classes	"dropdown"

The Empty Text property defines what the user sees before selecting a genre.

Applying the Search

The input fields for the search were already created on the Movies Screen. Now, we need to define the logic to apply the search on the Movies table. At this point, we're only saving the selection in Client Variables. This has two steps: filter the GetMovies Aggregate using the variables and trigger the search whenever the user uses the search filter.

1. We need to define two additional filters in the GetMovies Aggregate so that movies can be filtered by the search criteria chosen by the end-users (title or plot summary).
 - a. Open the **GetMovies** Aggregate, select the **Filters** tab and click on **Add Filter**.

The screenshot shows the 'GetMovies' aggregate configuration. At the top, there is a grid icon and the text 'GetMovies'. Below this, there are four status boxes: '1 Source', 'No Filters', 'No Sorts', and 'No Test Values'. At the bottom, there is a button labeled 'Add filter' with a hand cursor pointing to it.

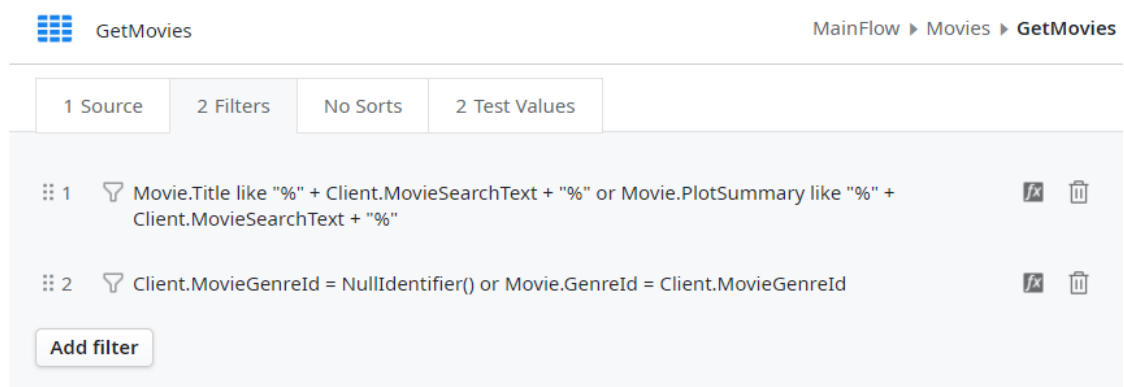
- b. Add the following filter condition for the title and plot summary

Movie.Title like "%" + Client.MovieSearchText + "%" or Movie.PlotSummary like "%" + Client.MovieSearchText + "%"

The % are used as wildcards, so this expression can be read as: we want a movie title with some text (possibly empty), followed by the search keyword, followed by more text (also possibly empty). So, if we search for "Force", the movie *Star Wars: The Force Awakens* would appear in the Table since it has the keyword Force, with text before and after. The same logic is valid for the plot summary. If we didn't add the %, then the user would have to type exactly the Title or Plot Summary of the movie to find it.

- c. Add a second filter for the **Movie Genre**

Client.MovieGenreId = NullIdentifier() or Movie.GenreId = Client.MovieGenreId



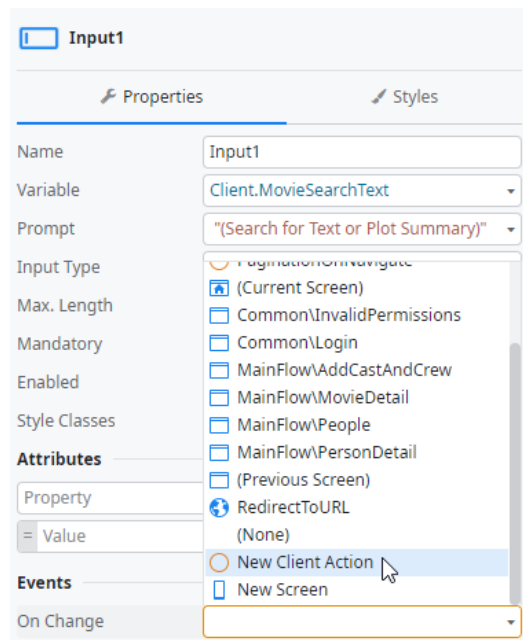
This condition is kind of special! So let's recap the desired behavior:

- If the user selects a genre, only movies of that genre should appear.
- If the user doesn't select any genre, then all movies should appear.

Now, consider all movies have a genre. If the condition was just *Movie.GenreId = Client.MovieGenreId*, whenever the user did not select a genre to filter the movies from, no results would be returned. Why? Because there would be no movies in the database with *GenreId = NullIdentifier()*. So, we need to consider that scenario on the Aggregate filter as well.

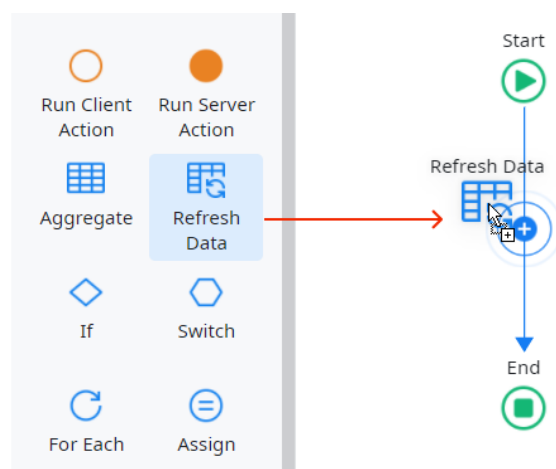
2. Next, we need to trigger the search whenever the user types a title or plot summary or selects a movie genre in the dropdown. For that, we need to use the **OnChange** Event.

- a. Select the **Input** field on the Movies Screen, expand the **OnChange** property under **Events**, and click on *New Client Action*.

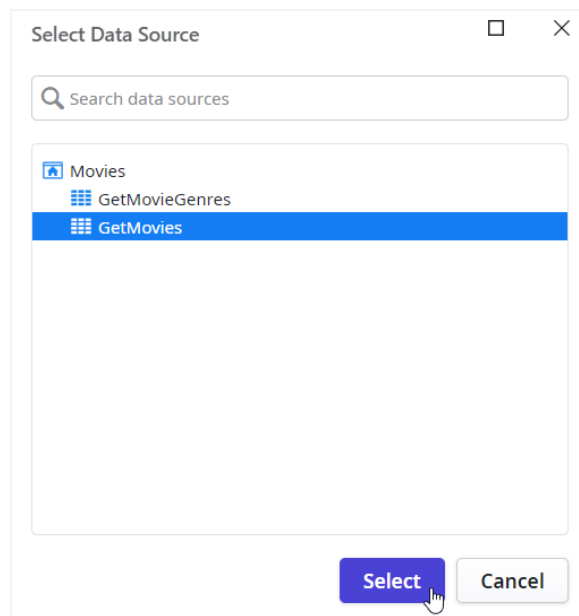


This OnChange Event is triggered automatically when the user types in an input field or selects an option from a selection-type input field, such as a dropdown for instance. This Event is really powerful since it removes the need to have a Search button. The user can just type and the logic in the Action is automatically triggered.

- b. Name the new Client Action *InputOnChange*.
- c. Drag a **RefreshData** and drop it in the Action Flow.

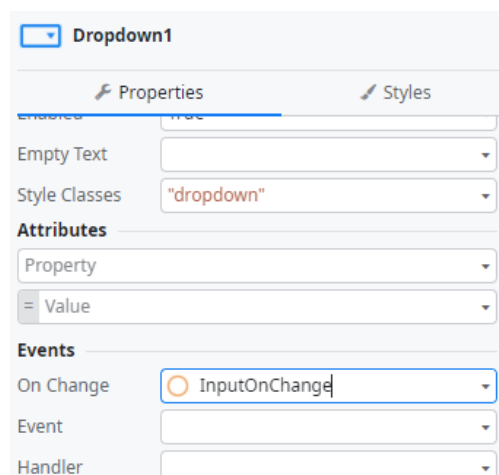


- d. Select the **GetMovies** Aggregate in the new dialog.



What we're doing here is explicitly running the GetMovies Aggregate. Why? Because the filters are already defined and the Client Variable is already filled with the user's keyword. So just by running the Aggregate again, the filter will consider that keyword and filter the results. But, why is the Client Variable already filled with a keyword? Because we're on the OnChange Event. This Action is only triggered whenever the user types or selects a keyword. Now we just need to do the same for the Dropdown.

- e. Go back to the Movies Screen and set the **OnChange** Event of the Dropdown to the same Action.



3. Publish the app and test it in the browser.



4. To properly test the behavior, we recommend these three steps:
 - i. Type a search keyword and wait for the results to appear. Do they make sense? Awesome!
 - ii. Select a movie genre and wait for the results to appear. Don't forget that if you have a keyword defined in the other input field, BOTH will be applied.
 - iii. Change to another Screen, navigate at will in your app and then come back to the Movies. Is the search still being applied? Nice! That's what the Client Variables do for you (well... actually one of the things it does for you!).