

Master-Detail

Exercise - How to

Outline	2
How To	2
Movie Genre Field	2
Add Cast/Crew to a Movie	4
AddCastAndCrew Screen	4
Linking the MovieDetail Screen with the AddCastAndCrew Screen	9
Logic to Save the PersonMovieRole Record	12
Testing the App and Extra Adjustments	16
Listing the Production Talent and the Cast/Crew	19
List of Production Talent	19
List of Cast and Crew	30

Outline

In this exercise, we will extend our UI functionality. First, we will add a feature to add cast/crew to a movie. For that, we will require:

- A new Screen where we can add a person with a certain role to a movie.
- A link on the MovieDetail Screen to navigate to the new Screen that allows adding a cast and crew to that movie.
- This link should be the only way to get to that Screen when navigating the app.

Then, in the second part of the exercise, we want to display the following information on the MovieDetail Screen:

- List of the Production Talent involved in the movie (Directors and Producers).
- List of the cast and crew of the movie (Actors and Crew).

Also, we want to add a new field to the MovieDetail Form to allow a user to select a genre for the movie.

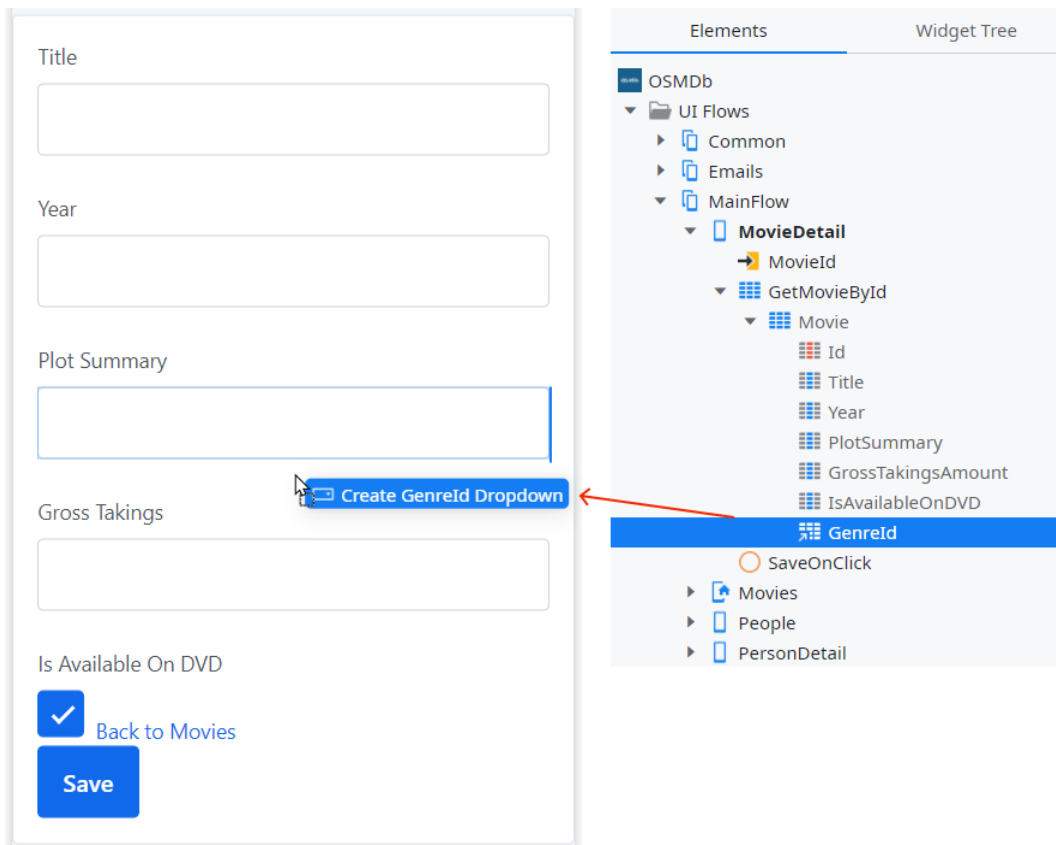
How To

In this section, we'll describe the exercise 5.2 – *Master-Detail*, step by step.

Movie Genre Field

We will start this exercise by adding the option for an end-user to assign or edit the genre of a movie, in the MovieDetail Form. The GenreId field was added to the Movie Entity in the previous exercise, so it's time to update the Form to include this new field.

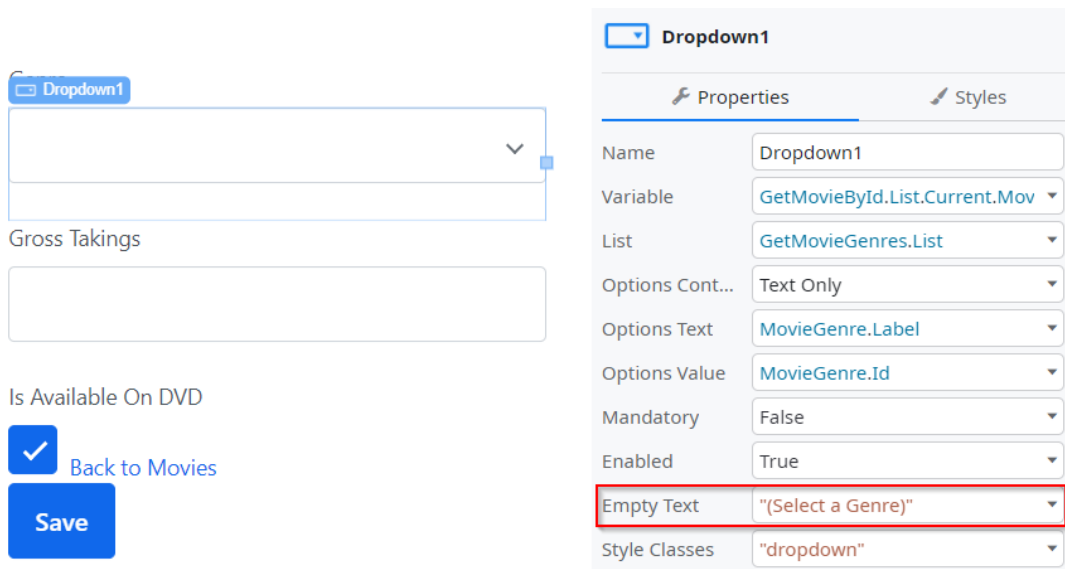
1. In the Interface tab, double-click on the **MovieDetail** Screen to open it.
2. Expand the **GetMovieById** Aggregate, drag the **Movie.GenreId** attribute, and drop it above the Gross Takings input in the Form.



This creates the field for the movie genre automatically.

Note: Since the GenreId was added after the bootstrap from excel, at this point, no movie has a Genre associated with it. You can later associate genres with movies while using the app in the browser.

3. Currently, the dropdown shows no information to the end-user. We want it to show the text (*Select a Genre*) before the user actually selects the genre that they want. So click on the recently created dropdown and set the **Empty Text property** to "(Select a Genre)".



Note: The Empty Text property represents an empty selection in the dropdown. Its content does not count as an option that can be selected. However, it can be very useful to provide hints to the user about what can be selected within the dropdown.

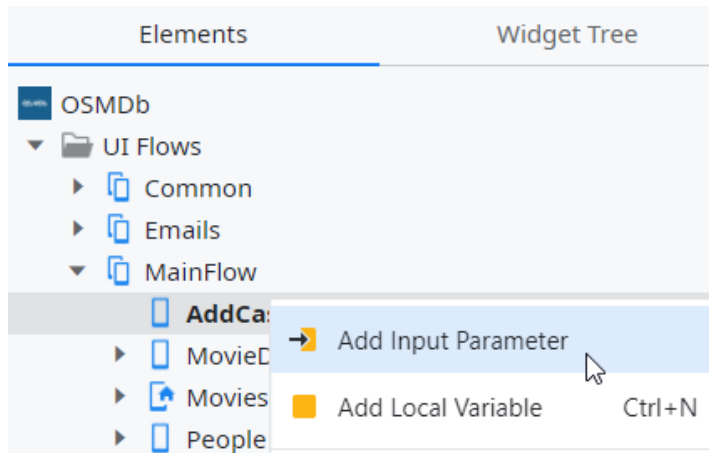
Add Cast/Crew to a Movie

In this section, we will create a new Screen that allows the app users to add a person to a movie, with a specific role. The logic to add a `PersonMovieRole` record to the database should also be created.

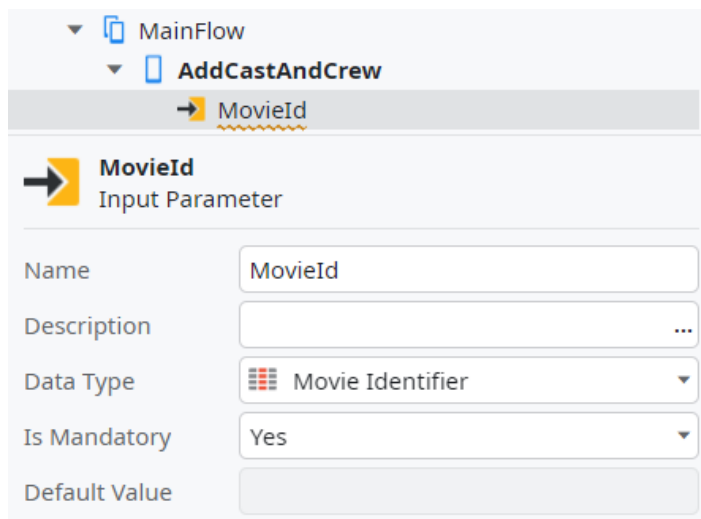
AddCastAndCrew Screen

We will start to implement the functionality with a new Screen, *AddCastAndCrew*. This new Screen will have a Form with two dropdowns: one to choose the person and one to choose the Role. The Screen should be accessible via a link from the *MovieDetail* Screen and should have the movie identifier as input parameter.

1. Create a new Empty Screen called **AddCastAndCrew** with an Input Parameter named **MovieId**.
 - a. Create an **Empty** Screen, name it *AddCastAndCrew* and make it accessible by **Everyone**.
 - b. Right-click on the Screen and choose the option **Add Input Parameter**.

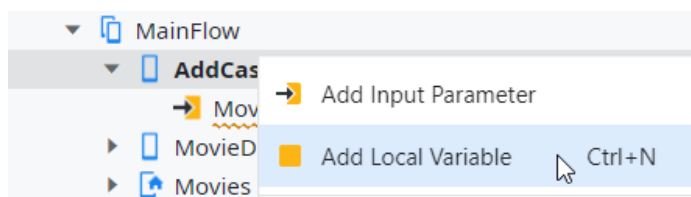


- c. Set the **Name** of the Input Parameter to *MovieId*, and confirm the **Data Type** is set to **Movie Identifier**.

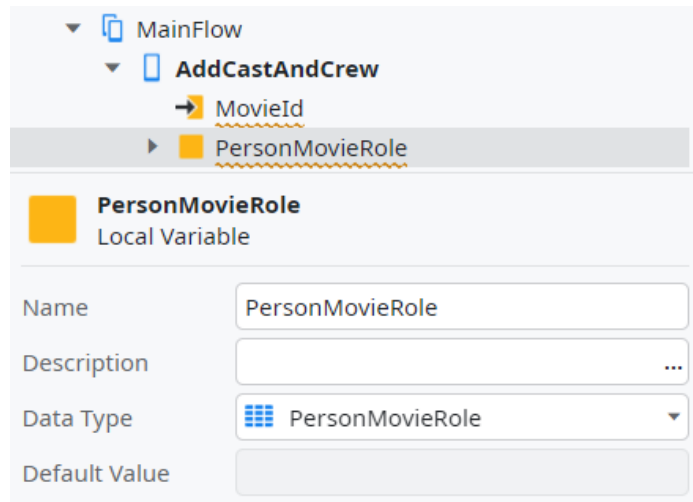


2. The Screen will have a Form with two dropdowns: one to choose the person and one to choose the role of the person in the movie. The data selected in both Dropdowns will be saved in a Local Variable on the Screen, *PersonMovieRole*, with the same structure of the PersonMovieRole Entity.

- a. Right-click on the Screen and choose the option **Add Local Variable**.

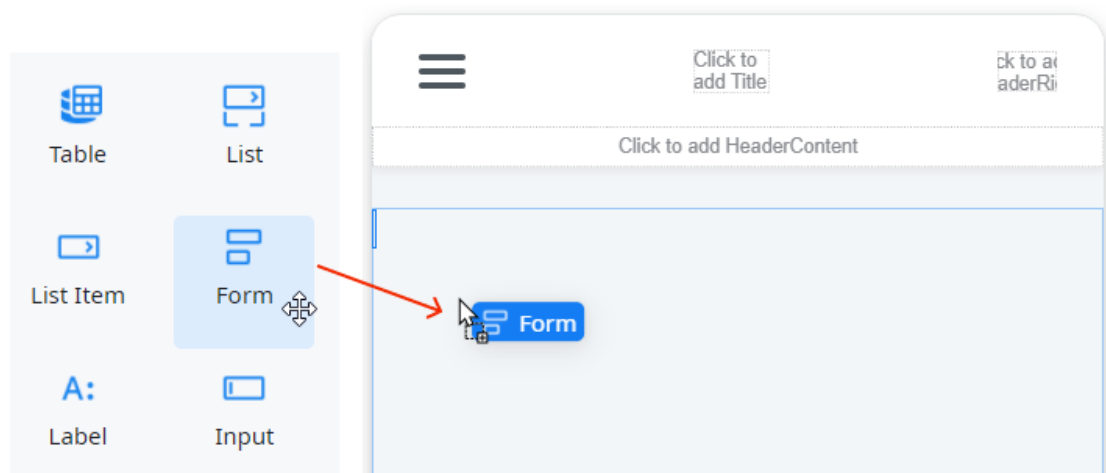


- b. Set its **Name** to *PersonMovieRole* and verify the **Data Type** is set to *PersonMovieRole*.

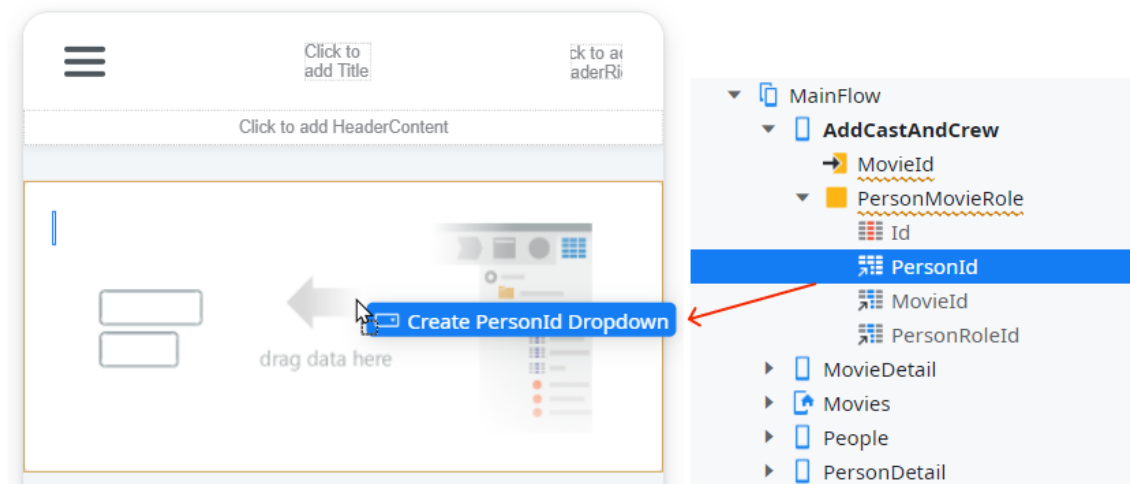


This variable will hold the information about the person's role in the movie, and will be used to update the database later.

- c. Drag a **Form** widget and drop it to the main content area of the Screen.

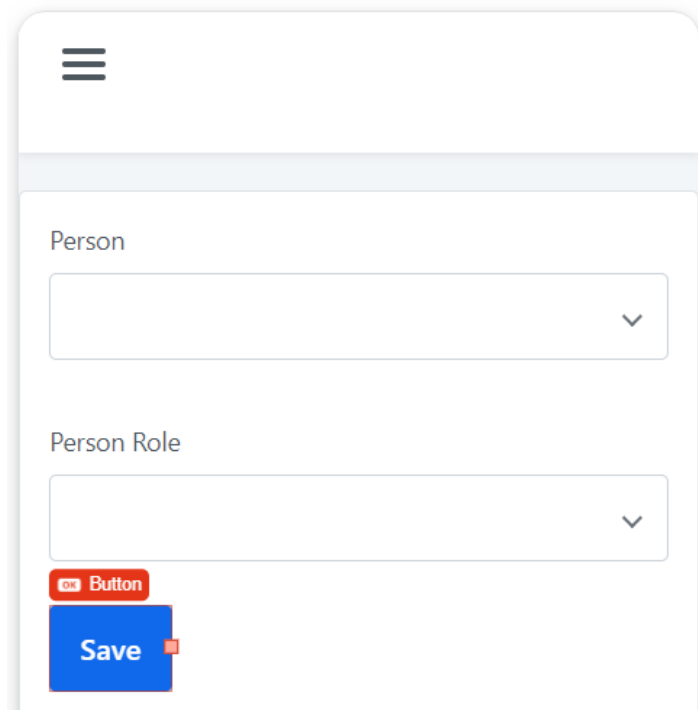


- d. Let's start building the Form by creating the first Dropdown to choose the person. Expand the **PersonMovieRole** Local Variable and drag the **PersonId** attribute to the Form.

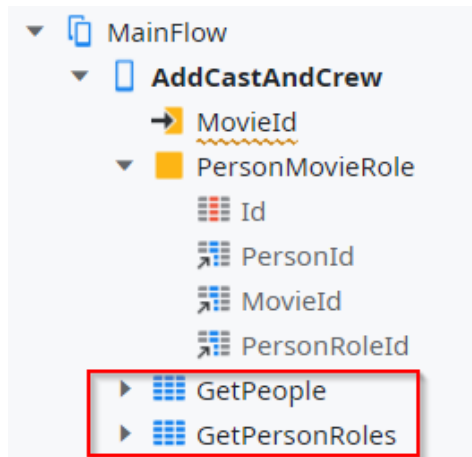


This step automatically creates the first dropdown.

- e. Do the same for the **PersonRoleId** attribute to create the second dropdown.



Note: The Save Button has an error, but that will be addressed later. Also, two Aggregates were automatically created after the drag and drop, GetPeople and GetPersonRoles. These Aggregates populate the dropdowns with the list of people and list of people roles in the database.



3. Make sure that when a user accesses the AddCastAndCrew Screen, the dropdowns have the following text appearing: *(Select a Person)* and *(Select a Role)*.
 - a. Select the **Person** dropdown.
 - b. In the properties area, set the **Empty Text** property to *"(Select a Person)"*.

Dropdown1	
Properties	Styles
Name	Dropdown1
Variable	PersonMovieRole.PersonId
List	GetPeople.List
Options Cont...	Text Only
Options Text	Person.Name
Options Value	Person.Id
Mandatory	False
Enabled	True
Empty Text	"(Select a Person)"
Style Classes	"dropdown"

- c. Do the same thing for the Person Role dropdown, but this time with the text *"(Select a Role)"*.
4. To complete the UI of the Screen, we need a Title for the Screen. We want to have something like:

Add Cast and Crew to (name of the movie)

To do that, we need to fetch information about the movie.

- a. Add an **Aggregate** that fetches the Movie that matches the **MovieId**, just like we did in the MovieDetail Screen.
- b. In the **Title** section of the Screen, drag an **Expression** and set it to be:

"Add Cast and Crew to " + GetMovieById.List.Current.Movie.Title

Expression Value

"Add Cast and Crew to " + GetMovieById.List.Current.Movie.Title

✓ The expression is ok (Type: Text)

+ - * / and or not True False = <> < > <= >= () [] null ▼

▼ List
 ▼ Current
 ▼ Movie
 Id
 Title
 Year
 PlotSummary
 GrossTakingsAmount
 IsAvailableOnDVD

Description
Title entity attribute : data type Text
No Description

Is Mandatory
Yes

?

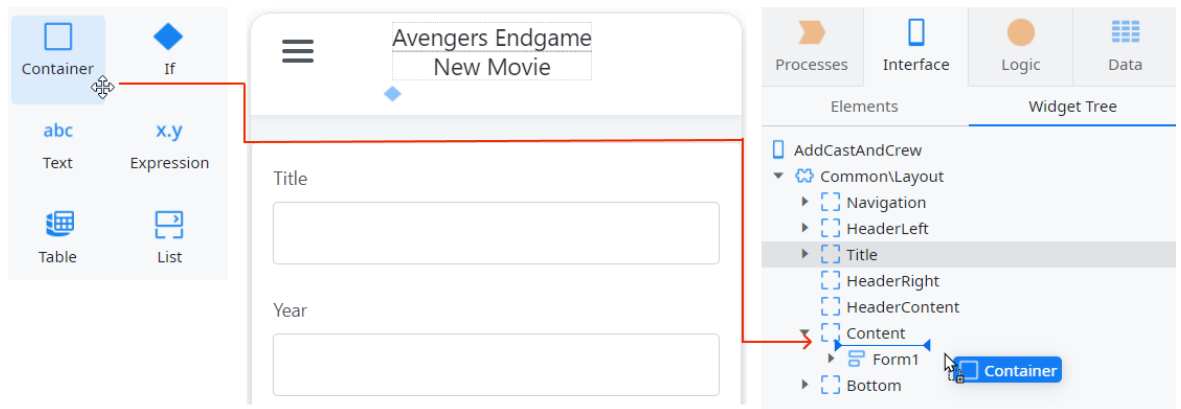
Close

Linking the MovieDetail Screen with the AddCastAndCrew Screen

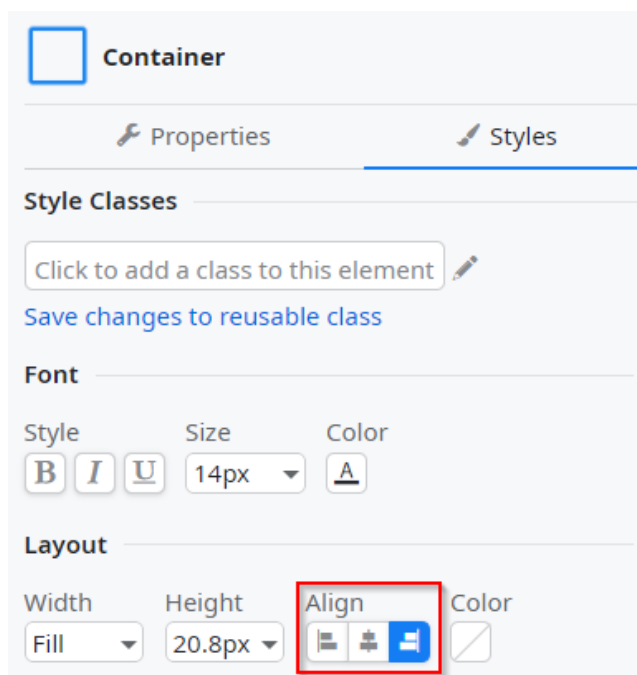
Now that we have the AddCastAndCrew Screen created, we need to find a way to navigate to it. So, the app will have a link in the MovieDetail Screen to navigate to this new Screen. Let's do it!

1. In the **MovieDetail** Screen, add a link to the AddCastAndCrew Screen so that end-users can navigate to it.
 - a. Open the MovieDetail Screen in the preview area by double-clicking on it.

- b. Drag a **Container** right above the Form. Use the widget tree to help you position the Container.

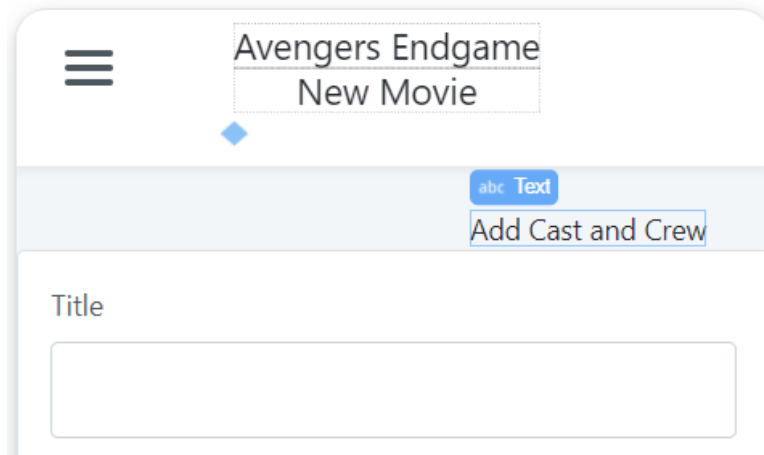


- c. Select the Container and in its properties area, switch to the Styles tab. Change the Container alignment to the right, to align the content inside the Container to the right.



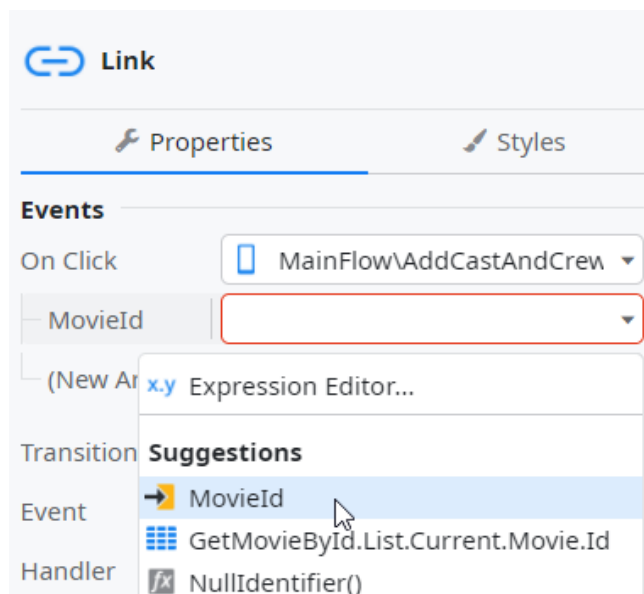
Note: When a widget is selected, the developer has two tabs to choose from: Properties and Styles. The Styles tab allows the developer to define some styles to the specific widget, such as bolds, italics, sizes, colors, alignments, width, margins, etc.

- d. Add a text to the Container: *"Add Cast and Crew"*.



e. Right-click on the text and **Link** it to the **AddCastAndCrew Screen**.

2. The AddCastAndCrew Screen expects a value for the movie identifier that we are working on. So, we need to pass that information in the Link. In the new **Link**, set the **MovieId** value to the MovieId Input Parameter of the Screen.

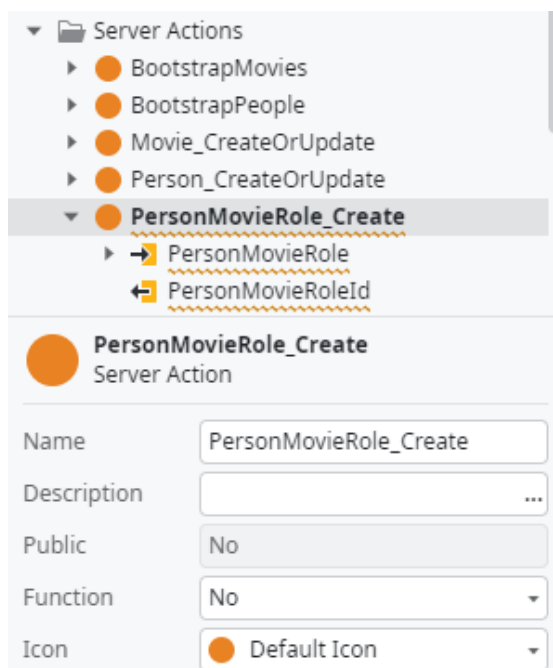


The app still has an error in the Save Button of the AddCastAndCrew Screen. So, let's implement the logic to save the PersonMovieRole record.

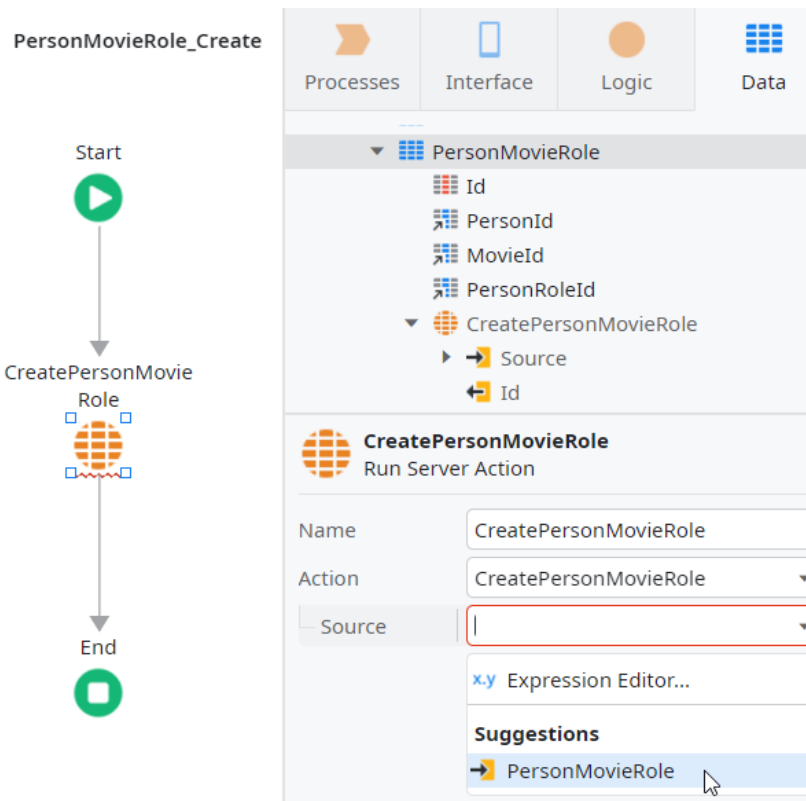
Logic to Save the PersonMovieRole Record

As we mentioned above, the Save button has an error, and that's because the OnClick property of the Button is not defined yet. The Save Button should trigger a Client Action with the logic to save the role of the person in a certain movie in the PersonMovieRole Entity in the database.

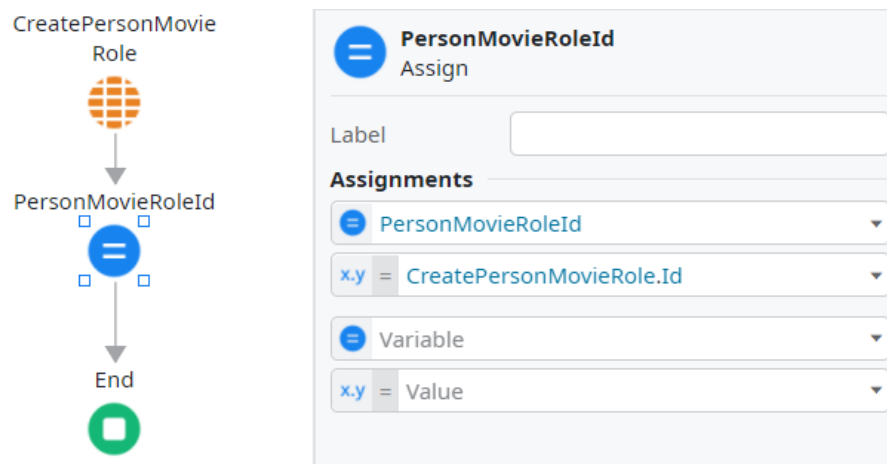
1. Create a new Server Action called *PersonMovieRole_Create* with the logic to save a PersonMovieRole record in the database. This PersonMovieRole record information is passed to the Action via input parameter. The Action should return the id of the PersonMovieRole record that was created in the database.
 - a. Switch to the **Logic** tab and create a new **Server Action** called *PersonMovieRole_Create*.
 - b. Add an Input Parameter called *PersonMovieRole* and an Output Parameter called *PersonMovieRoleId*.



- c. Under the Data tab, in the PersonMovieRole Entity, you can find the **CreatePersonMovieRole** Entity Action. Drag it and drop it on the recently created Action flow. Set its **Source** to be the Input Parameter *PersonMovieRole*.



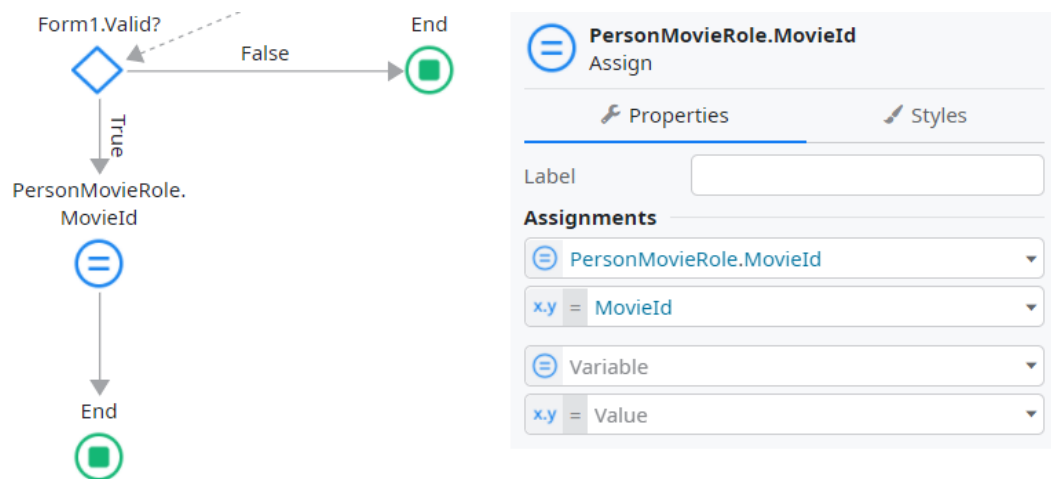
- d. Drag an **Assign** and drop it after the **CreatePersonMovieRole** Action. Set the assignment to be: *PersonMovieRoleId = CreatePersonMovieRole.Id*



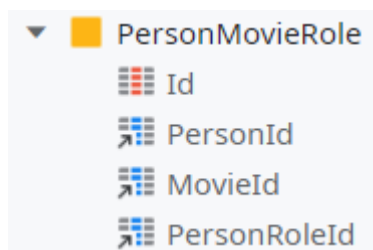
This completes the Server Action logic by assigning the Output Parameter to the identifier of the record created in the database.

2. Create a new Client Action, **SaveOnClick**, that is triggered when the Save button of the AddCastAndCrew Screen is clicked by a user. This Action should call the Server Action created above and navigate back to the MovieDetail Screen at the end.

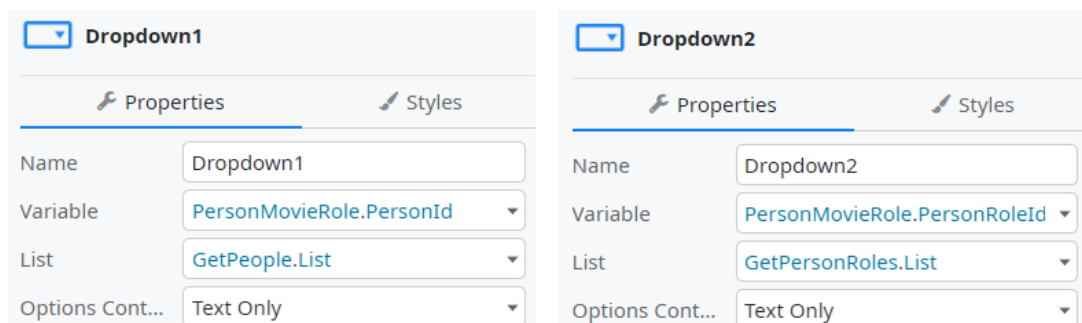
- Under the Interface tab, double-click the **AddCastAndCrew** Screen to open it in the preview area of ODC Studio.
- Double-click on the **Save** button to create the *SaveOnClick* Screen Action.
- Drag and **Assign** and drop it after the If in the Action flow. Define the assignment to be: *PersonMovieRole.MovieId = MovieId*



This assignment has a special role. Remember that the Form has two dropdowns: one for the person and one for the role. The PersonMovieRole record has a PersonId, MovieId and PersonRoleId.

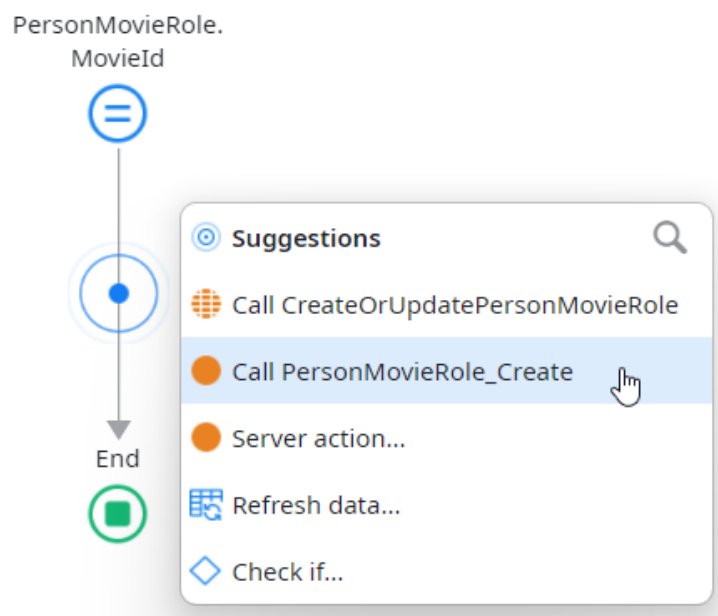


The PersonId information is already associated with the Person dropdown, while the PersonRoleId is associated with the Role dropdown.



However, the Movie information was never added to the Local Variable. We know which movie we are working on because of the input parameter of the Screen (MovieId), but the Local Variable does not know that, and it needs to know before we send it to be saved in the database (through the PersonMovieRole_Create Server Action). And that's why we create this Assign, to complete the Local Variable information with the Movie Identifier, using the MovieId Input Parameter to help us out in the assignment. Could we request the movie information from the user like we do with the person and role? Sure! But that would be a strange behavior, since the user will navigate to this Screen from the MovieDetail Screen, where the movie was already selected.

- d. Mouse over the last arrow, click the blue icon and click on the **Call PersonMovieRole_Create** option.

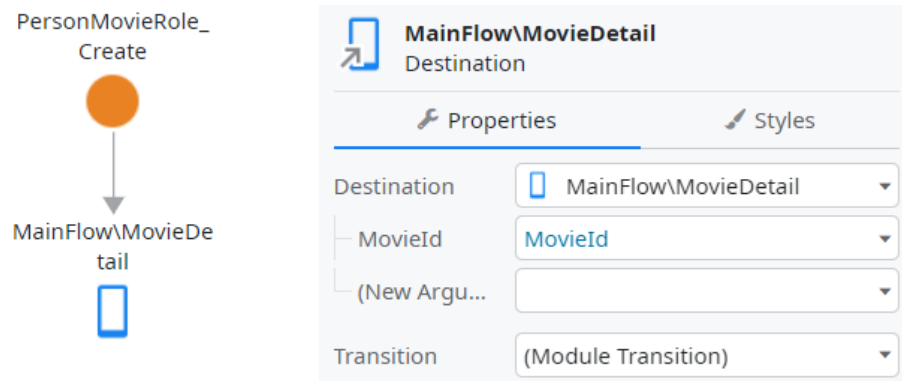


This automatically calls the Server Action we created earlier and passes the PersonMovieRole Local Variable as the value of the input parameter.

Note: If this option does not appear through the AI suggestions, just switch to the Logic tab, drag the Action and drop it in the flow and set the input parameter value to the PersonMovieRole Local Variable.

- e. Drag a Destination element from the left sidebar and drop it on top of the End node to replace it. Set the MovieDetail Screen as the destination and pass the MovieId Input Parameter Replace the End node with a **Destination** to the

MovieDetail Screen. Pass the **MovieId** Input Parameter of the Screen as the value of the MovieDetail Input Parameter.



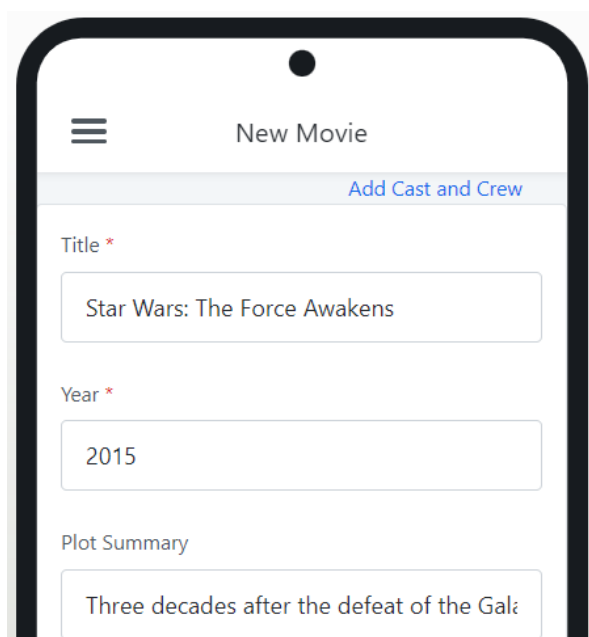
Testing the App and Extra Adjustments

At this point, we have the UI and the logic created, so it's time to publish and test it in the browser.

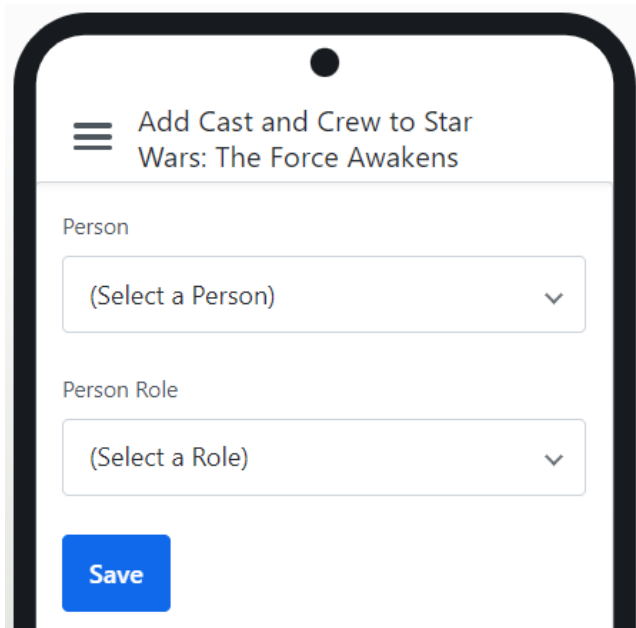
1. Publish the app and open it in the browser.



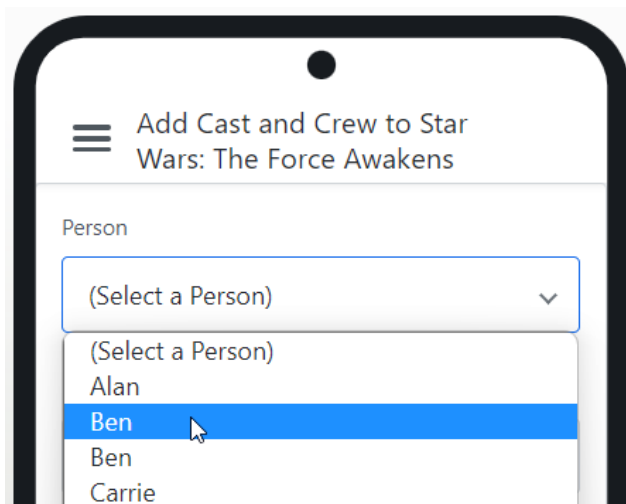
2. Select a movie in the Movies Screen to navigate to the MovieDetail Screen. Make sure the link to the Add Cast/Crew Screen is visible and that you can click on it.



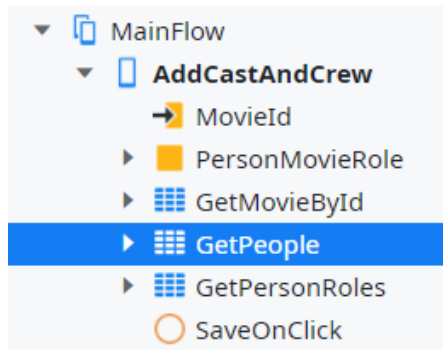
3. Make sure the UI of the Screens looks something like the next image.



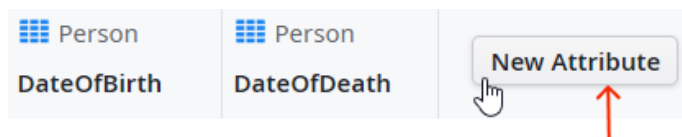
4. Try to add a member of the cast and crew to a movie. Did you notice that the AddCastAndCrew Screen has the Person dropdown with only the first name appearing on the options? Let's change it to include the surname as well.



5. Adjust the GetPeople Aggregate to fetch the full name of the person. To do that, we will add a new calculated attribute to the Aggregate.
 - a. Expand the **AddCastAndCrew** Screen and then open the **GetPeople** Aggregate inside it.

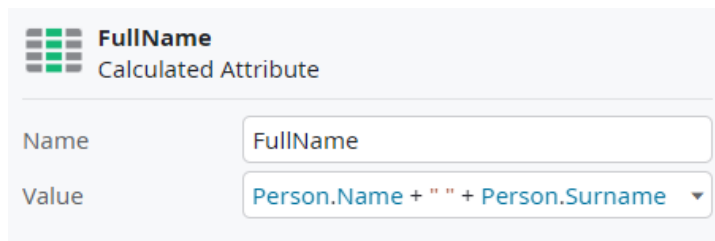


- b. In the Aggregate, click on **NewAttribute**.



- c. Name the new attribute *FullName* and add the **Value** below:

Person.Name + " " + Person.Surname



This simply picks the values in the Name and Surname attributes and concatenates them together with an empty space in between. The Aggregate will not return all the attributes of the Person Entity + this new calculated attribute.

- d. Go back to the AddCastAndCrew Screen, select the Person dropdown, and on the **Options Text** property, select **FullName**.

Dropdown1

Properties | **Styles**

Name: Dropdown1

Variable: PersonMovieRole.PersonId

List: GetPeople.List

Options Content: Text Only

Options Text: Person.Name

Options Value: **Suggestions**

Mandatory: ☒

Suggestions:

- FullName
- Person.Id

- e. Publish the app and test it in the browser.



- f. Make sure the dropdown appears with the full name and that the app works as expected.

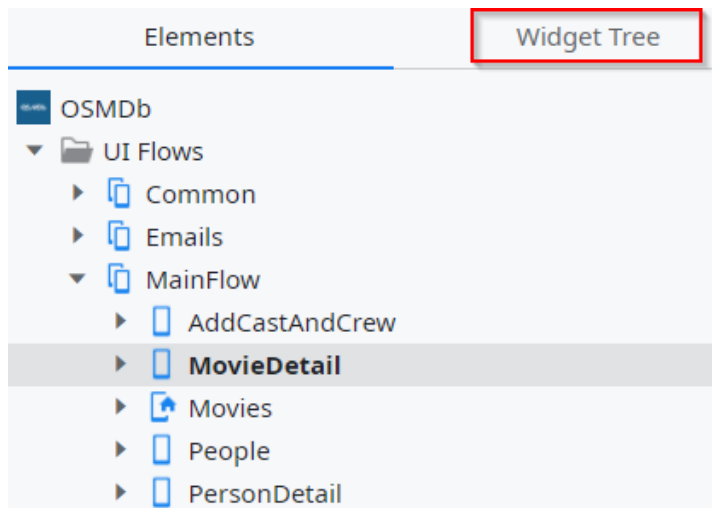
Listing the Production Talent and the Cast/Crew

The logic to add a person to a movie with a certain role is done. Now, we want to list some information about the cast and crew of the movie. In this section, we will create the UI components on the **MovieDetail** Screen to display the directors and producers, and another one for the cast and crew, associated with the movie.

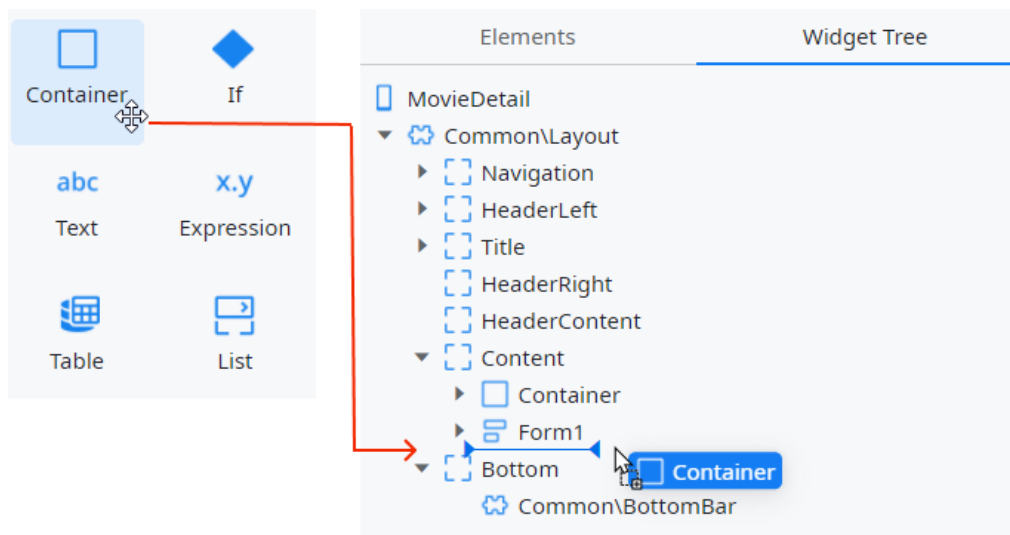
List of Production Talent

Let's start with the directors and producers.

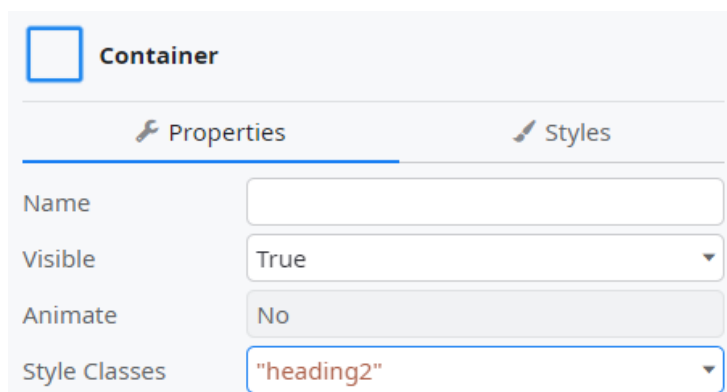
1. We will create a List to display the full name of the producers and directors involved in the movie in the MovieDetail Screen. However before that, we will create a subtitle right above the List, *Production Talent*, with the **heading2** style.
 - a. Open the MovieDetail Screen and open the Widget Tree.



- b. Using the Widget Tree, expand the MainContent area. Drag a Container and drop it below the Form.

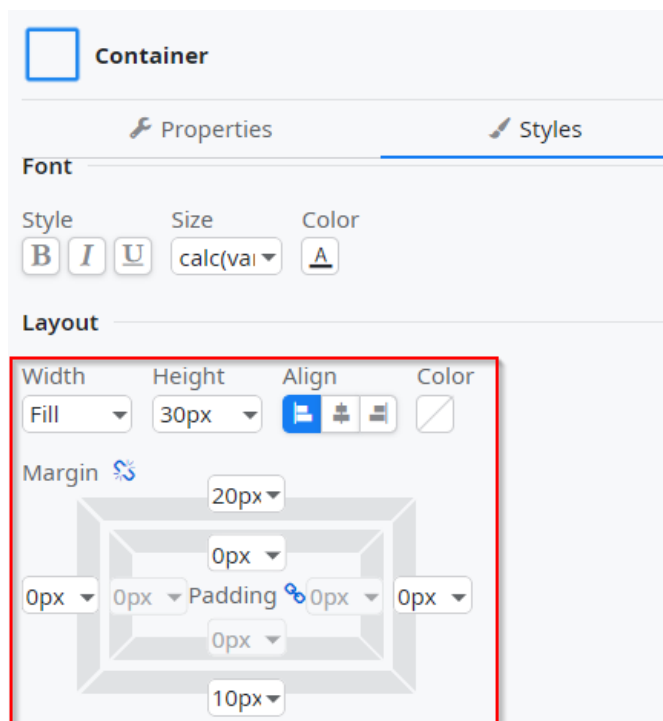


- c. Set the Style Classes of the Container to "heading2"

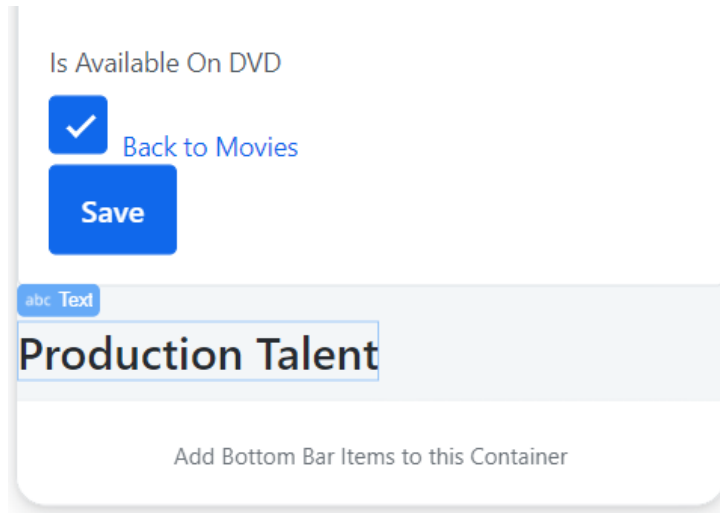


This is using a built-in style class that is available from the moment the app was created. The app has a default theme that already has several style classes defined (as you can see in the long list when you expand the Style Classes property dropdown). You can customize any existing class, or create your own classes, but at this point let's just use the ones that already exist, in this case, the *heading2* to create a subtitle.

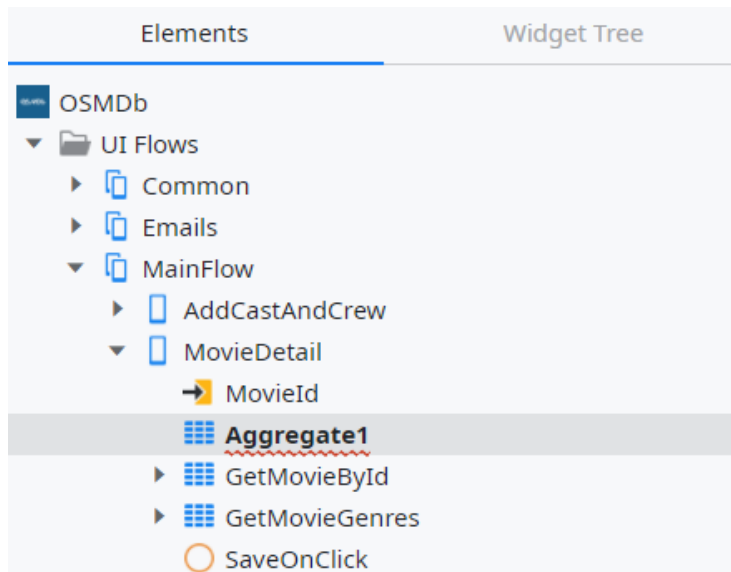
- d. You may have noticed that in the properties area of the widgets there's also a Styles tab. Click the **Styles** separator of the Container to open it. You can see here some styles that we can apply to the widget, related to font and layout, so let's leverage that. Add a **Margin top** of *20px* and a **Margin bottom** of *10 px* to create some margins between the subtitles.



- e. Type *Production Talent* inside the Container.



2. We want to list the producers and directors that are associated with the movie, so let's create an Aggregate that fetches that information.
 - a. Switch back from the Widget Tree to the Elements tab on the right sidebar and create a new **Aggregate** on the MovieDetail Screen.



We have seen many accelerators to create an Aggregate already, but here's a new one. We will type the query we want and the Aggregate will be created automatically.

- b. In the Aggregate's preview, locate the **What data do you want to get?** question on the top right.

Aggregate1

What data do you want to get? ?

Example: How many orders shipped to London

Get Data

[More Examples](#)

- c. Type *list person* in the input field. At this point, you should see a dropdown with the Person Entity appearing. Select the **Person** Entity.

What data do you want to get? ?

list person

Entities

- Person
- PersonMovieRole
- PersonRole

OutSystems is interpreting the text and mapping with the Entities you have in the app. Let's proceed.

- d. Continue to write the query by typing *with role*. At this stage, scroll down a bit in the suggestions' dropdown until you find the **PersonRoleId** attribute of the PersonMovieRole Entity.

What data do you want to get? ?

list person with role

Attributes

- Id (in PersonMovieRole)
- Id (in PersonRole)
- Is_Active (in PersonRole)
- Label (in PersonRole)
- MovieId (in PersonMovieRole)
- Order (in PersonRole)
- PersonId (in PersonMovieRole)
- PersonRoleId (in PersonMovieRole)

- e. Almost there! Now continue to type *equal to Director*. Choose the **Director** record of the PersonRole Entity. Continue the query and type *or producer*, and choose the **Producer** record of the PersonRole Entity.

What data do you want to get? ?

list person with role (in PersonMovieRole) equal to Director

Static Records

Director (in PersonRole)

What data do you want to get? ?

list person with PersonRoleId (in PersonMovieRole) equal to Director (in PersonRole) or Producer

Static Records

Producer (in PersonRole)

- f. We're done! Click on the **GetData** button.

What data do you want to get? ?

list person with PersonRoleId (in PersonMovieRole) equal to Director (in PersonRole) or Producer (in PersonRole)

[More Examples](#)

Get Data

You should now have an Aggregate with 2 sources, a join between the two Entities, and a filter that fetches the people that are directors or producers in the movie.

2 Sources 1 Filter No Sorts No Test Values

Sources

Person

PersonMovieRole

Add source

Joins

1 Person Only With PersonMovieRole

Person.Id = PersonMovieRole.PersonId

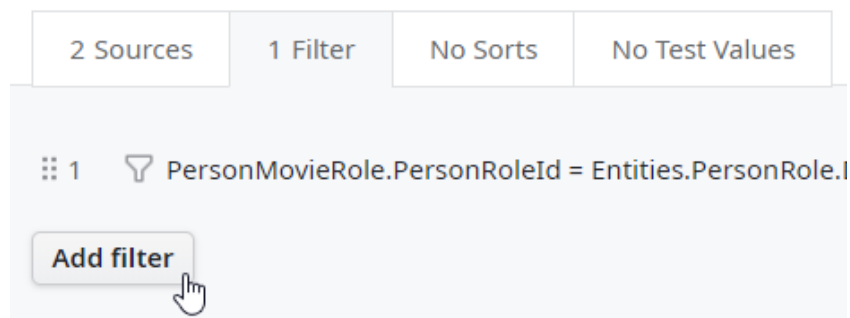
Add join

2 Sources 1 Filter No Sorts No Test Values

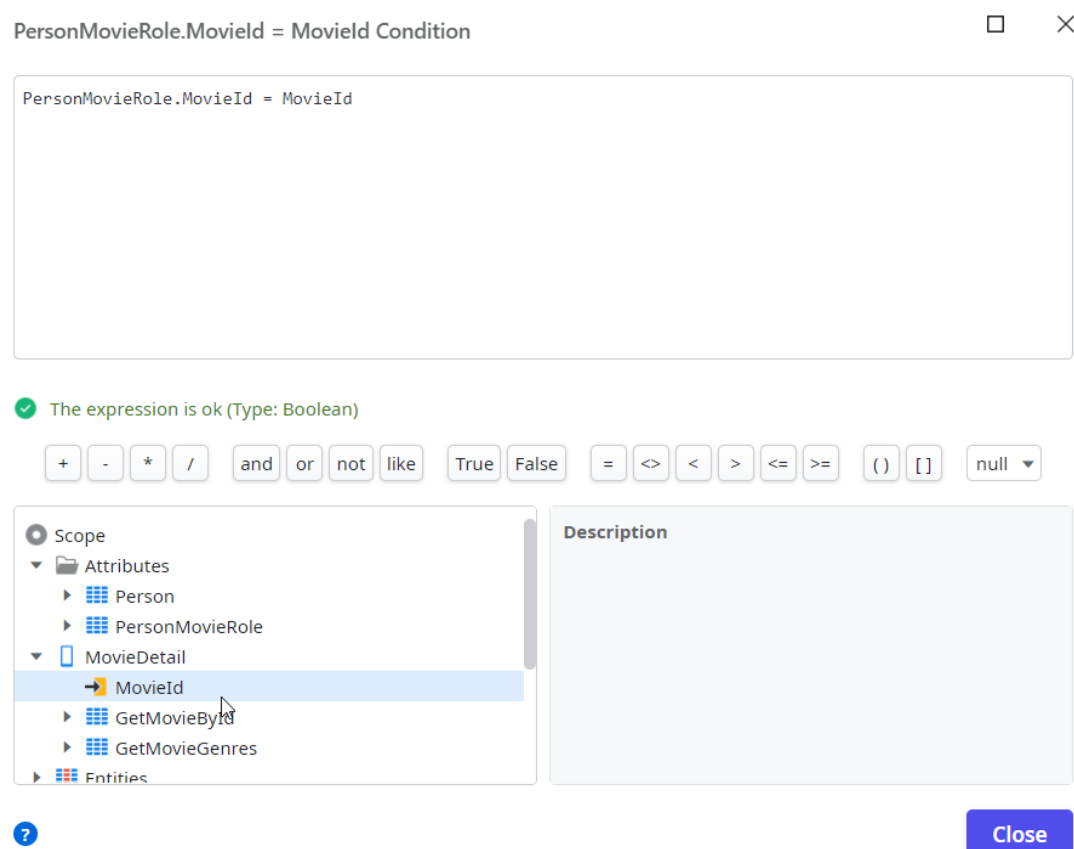
1 PersonMovieRole.PersonRoleId = Entities.PersonRole.Director or PersonMovieRole.PersonRoleId = Entities.PersonRole.Producer

Add filter

- g. There's only a couple of details left. We want to make sure the list of people that appear are associated with the movie being displayed in the MovieDetail Screen. So, in the Aggregate, open the **Filter** tab and click on **Add Filter**.

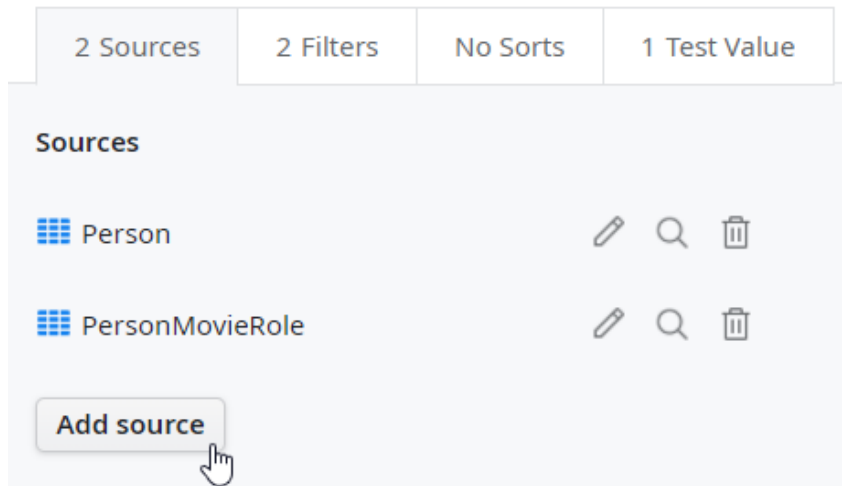


- h. In the new dialog, define the filter to be *PersonMovieRole.MovieId = MovieId*

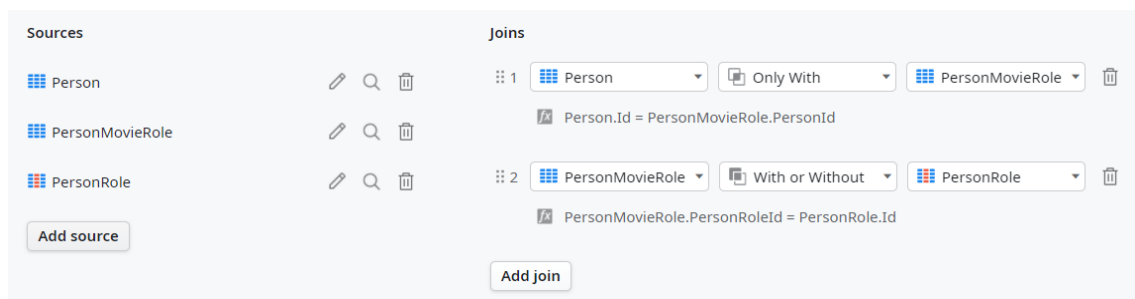


Click the **Close** button when you're done.

- i. Switch back to the **Sources** tab of the Aggregate and click on **Add Source**.



- j. Select the **PersonRole** Entity in the new dialog. You will get a new Source added and a new Join automatically created.

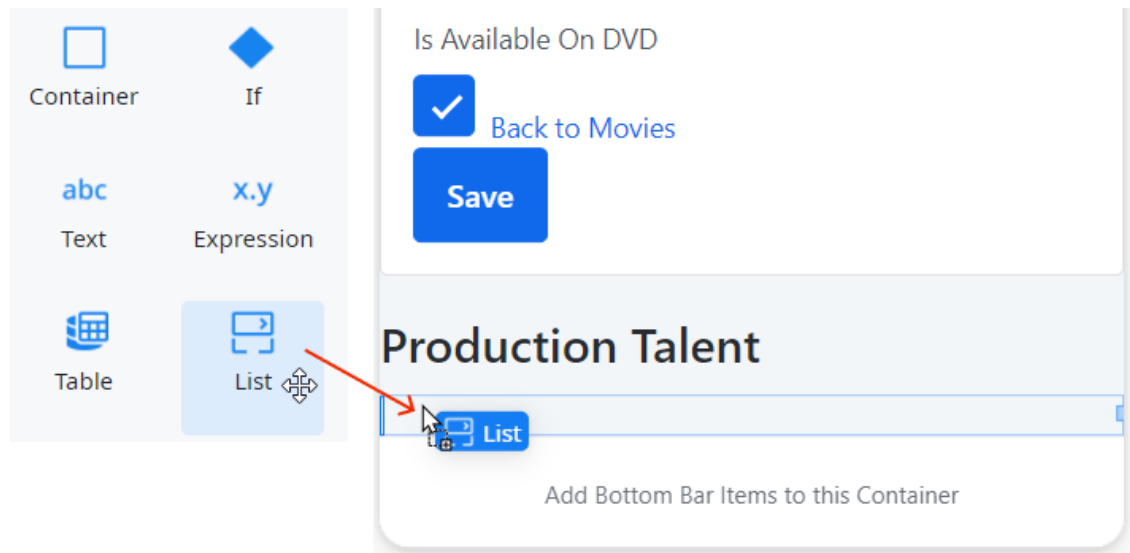


Now you may ask... why do we need that? We will need it because we want to display the role of the person in the production talent list. So, we need the Label attribute of the PersonRole Static Entity, otherwise we would only have the PersonRoleId, which is not very interesting to display in the List since it's a number.

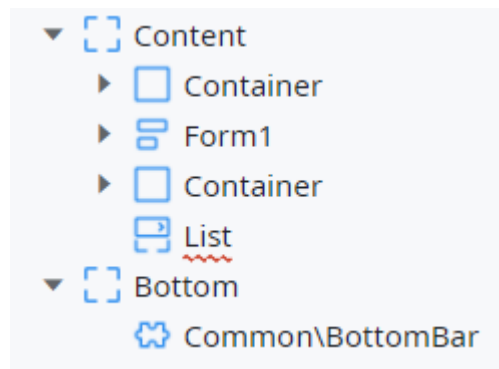
- k. Rename the Aggregate as *GetProductionTalent*.



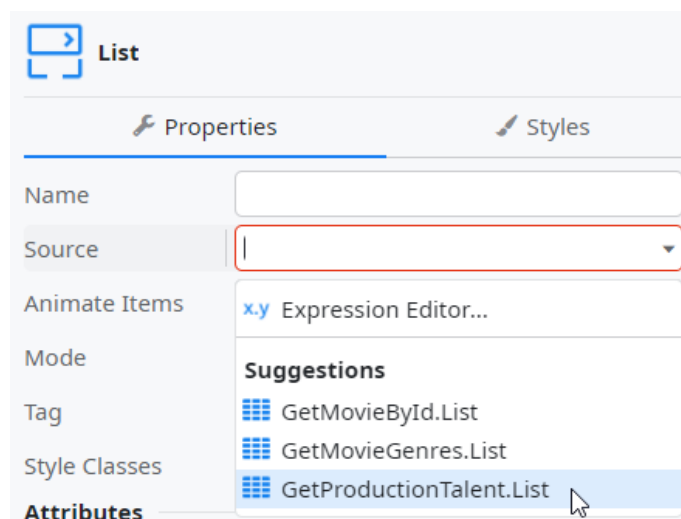
3. Now we can finally add the List of directors and producers. This List should display the full name of the person, plus the role between parenthesis.
 - a. Open the **MovieDetail** Screen by double-clicking on it.
 - b. Drag a List widget from the left sidebar and drop it below the Production Talent Container.



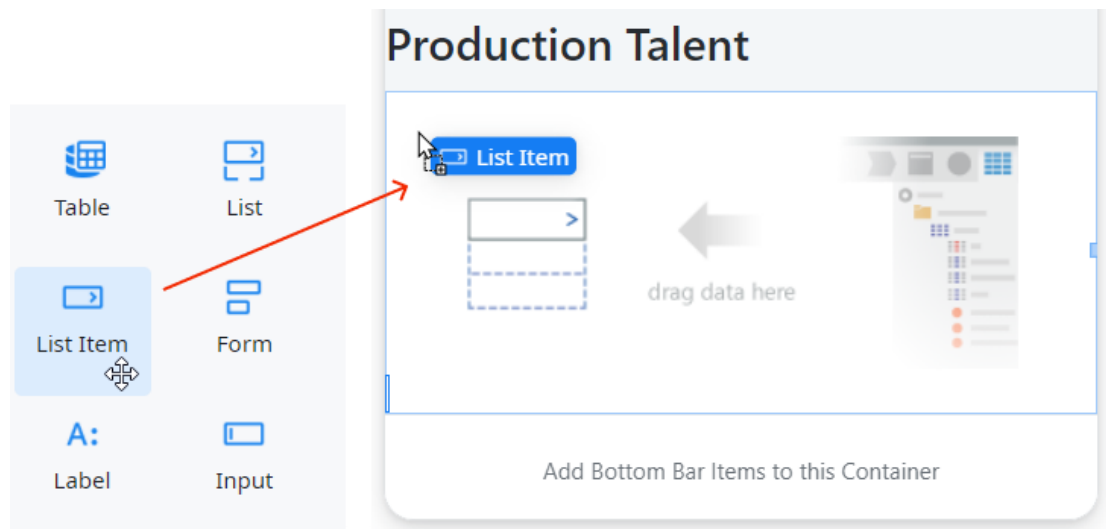
Make sure the List is outside the Container, to avoid being influenced by the *heading2* style. You can use the Widget Tree to help you out.



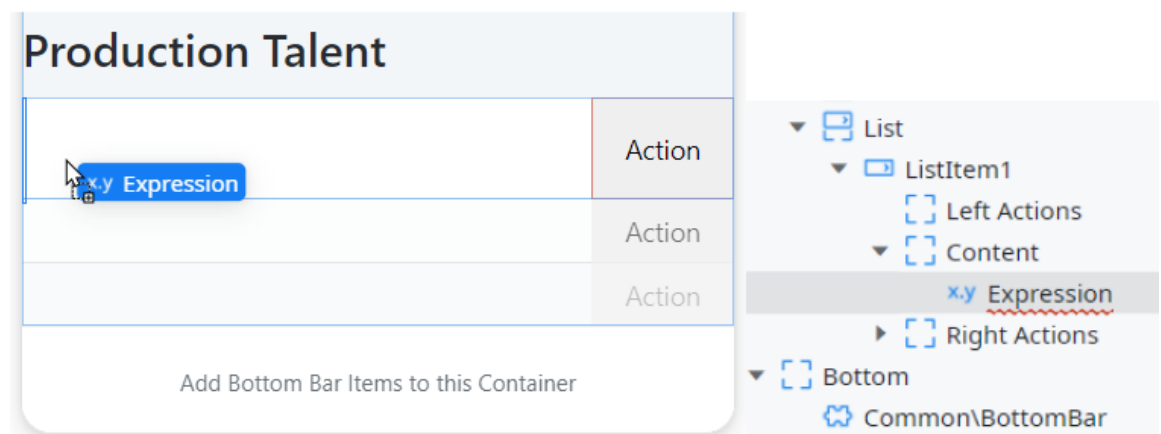
- c. Select the List and make sure you see the Properties area on the right (if you see the Styles, just switch the tab back to the Properties). Set the **Source** of the List to the *GetProductionTalent.List*



- d. Drag a **ListItem** and drop it in the List. Each List Item will have the information for a particular person on the List.



- e. Drag an **Expression** and drop it inside the Content placeholder of the List Item.



- f. Set the value of the Expression to:

```
GetProductionTalent.List.Current.Person.Name + " " +
GetProductionTalent.List.Current.Person.Surname + "(" +
GetProductionTalent.List.Current.PersonRole.Label + ")"
```

- g. Right-click on the Expression and **Link** it to the **PersonDetail** Screen. Set the **PersonId** parameter to `GetProductionTalent.List.Current.Person.Id` so the PersonDetail Screen "knows" which person information it will display.

Link

Properties

Name

Confirmation Mes...

Enabled

Visible

Style Classes

Attributes

Property

Events

On Click

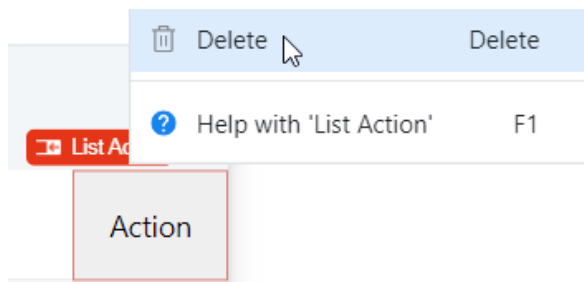
PersonId

x.y Expression Editor...

Suggestions

- GetProductionTalent.List.Current.Person.Id
- GetProductionTalent.List.Current.PersonMovieRole.PersonId
- NullIdentifier()

- h. Back on the MovieDetail Screen, right-click the **List Action** on the right side of the List Item and **Delete** it. The List Action is useful for swiping behavior, which we will not use in this particular exercise.



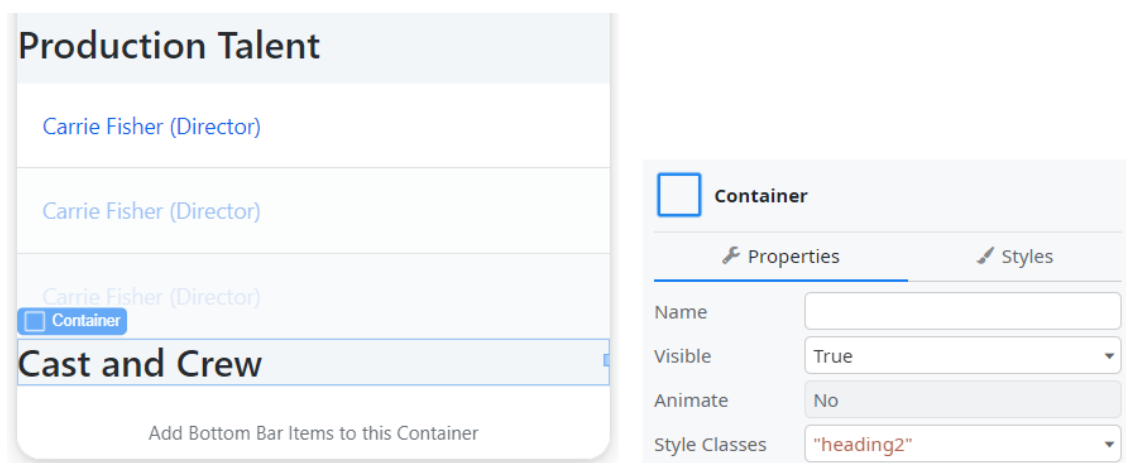
4. It's time to publish and test the application! Make sure you choose a movie that has already some directors and producers added.



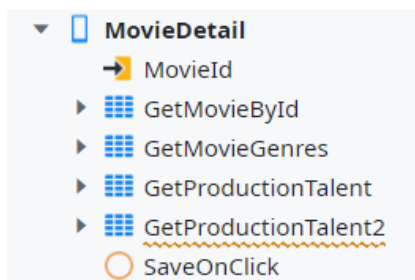
List of Cast and Crew

And let's now finish the exercise with the list of cast and crew. We will have something really similar to what we did for the production talent.

1. Create a new subtitle right below the production talent list, with the text *Cast and Crew*.
 - a. Drag a new **Container** and drop it below the list with the directors and producers.
 - b. Type *Cast and Crew* inside the Container.
 - c. Set the **Style Classes** property to *"heading2"*.

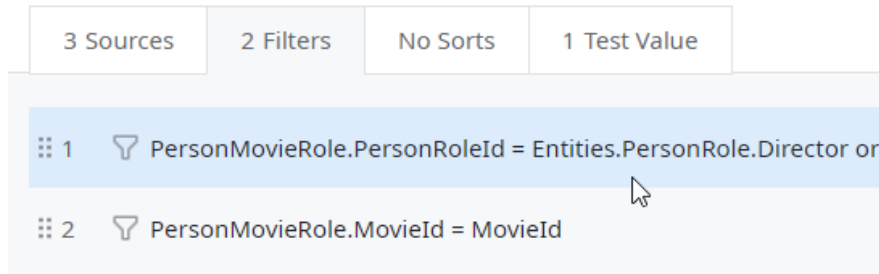


- d. Switch to the **Styles** tab and set the **Margin top** to 20px and the **Margin bottom** to 10px.
2. Now, we need an Aggregate to fetch the actors and members of the crew.
 - a. Under the **MovieDetail** Screen, right-click the **GetProductionTalent** Aggregate, select **Copy** and then right-click on the **MovieDetail** Screen and select **Paste**. You will end up with an Aggregate called **GetProductionTalent2**.



- b. Change the name of the Aggregate to *GetCastAndCrew*.

- c. Double-click on the Aggregate to open it. If you think about it, the Aggregate is very similar to the one we have for the production talent. Only the filters change! Open the **Filters** tab and click on the filter for producers and directors.

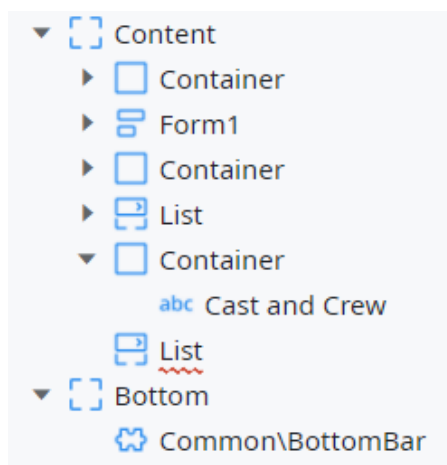


- d. Change the filter to be:

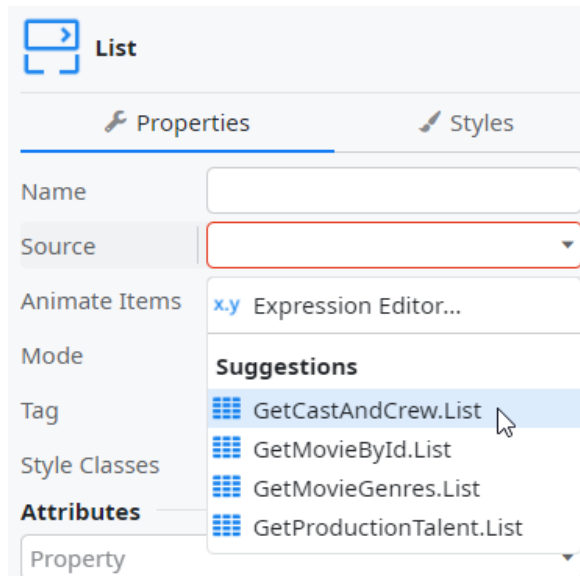
*PersonMovieRole.PersonRoleId = Entities.PersonRole.Actor or
PersonMovieRole.PersonRoleId = Entities.PersonRole.Crew*

Copy and paste is not always that bad right?!

3. Back in the MovieDetail Screen, create a **List** to display the full name of the cast and crew involved in the movie, as well as their role between parenthesis.
- a. Double-click the MovieDetail Screen to open it. Drag a List widget from the left sidebar and drop it right below the Container with the Cast and Crew text. Use the Widget Tree to help you.



- b. Set its **Source** to *GetCastCrew.List*.



- c. Use the same strategy that we used above to display the name of the person and the role. First use a List Item widget, without the List Action element on the right, and define an Expression with the Name, Surname, and the Role between parenthesis.
- d. Link the Expression to the **PersonDetail** Screen.
- e. Publish the app and test it in the browser!
- f. Your app should look like the following image:



We have reached the end! It was a longer exercise, but we created a lot of cool things in our app.