outsystems

Master-Detail - How-to

Table of Contents

Outline

In this exercise, we will extend the UI functionality of our application. First, we will add the chance 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 the new Screen, to allow adding a cast and crew to that movie.
- This link should be the only way to get to that Screen when navigating in the application.

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

- A List with the Production Talent involved in the movie (Directors and Producers).
- A List with the cast and crew of the movie, with the total number of the cast and crew members in the section title.

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

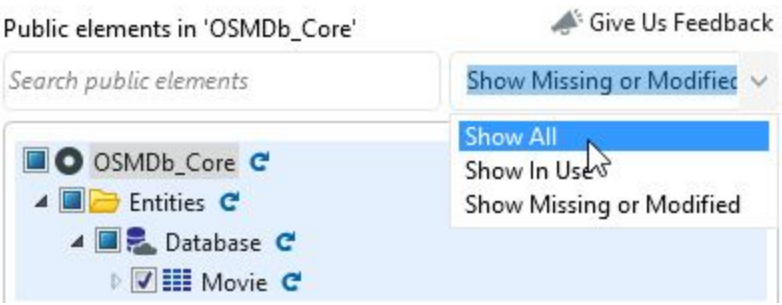
How to

In this section we'll describe, step by step, the exercise *4.5 - Master-Detail.*

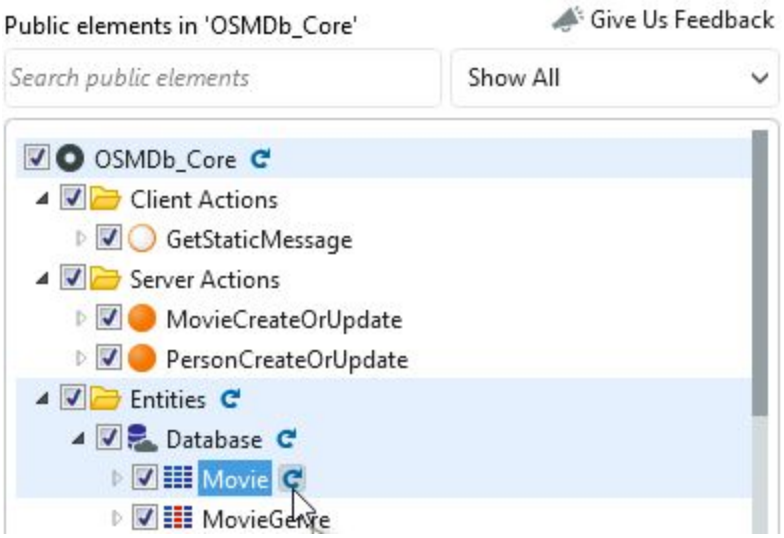
Edit the Movie Genre

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.

1. Switch back to the OSMDb module and reference the Entities created in the previous exercise, using the manage dependencies window. Don't forget to select **Show All** after selecting the OSMDb_Core.

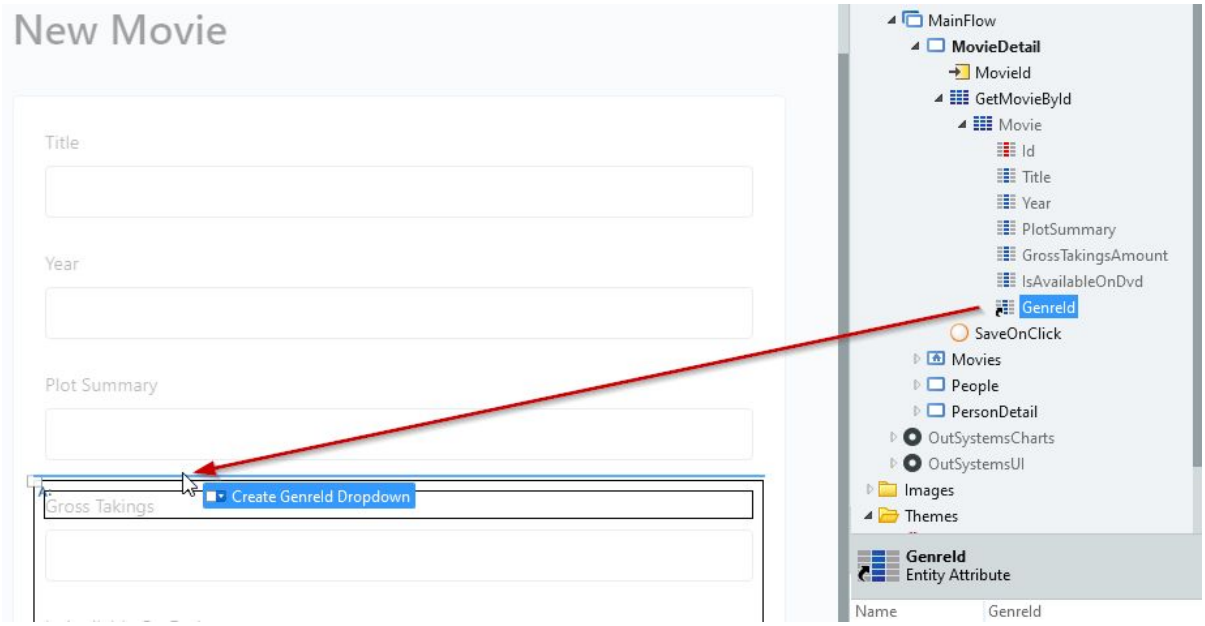


2. Now, we have a special case, since besides the new Entities, we also changed the Movie Entity to include a new attribute. In that case, we need to refresh the dependency. Click **Apply** to close to close the dialog.



3. Open the **MovieDetail** Screen and expand the **GetMovieById** Aggregate.

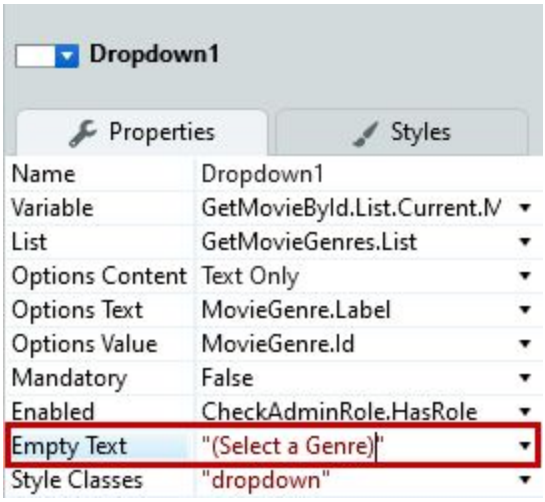
4. Drag the **Movie.GenreId** and drop it above the Gross Takings input.



NOTE: Since the GenreId was added after the bootstrap, no movie at this point has a Genre defined.

---

5. Set the **Empty Text** of the recently created dropdown to "(Select a Genre)". This will be the text visible to the end-user, until a movie genre is actually selected.



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 of what can be selected within the dropdown.

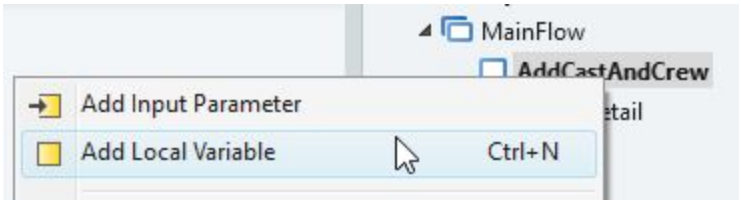
Add Cast/Crew to a Movie

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

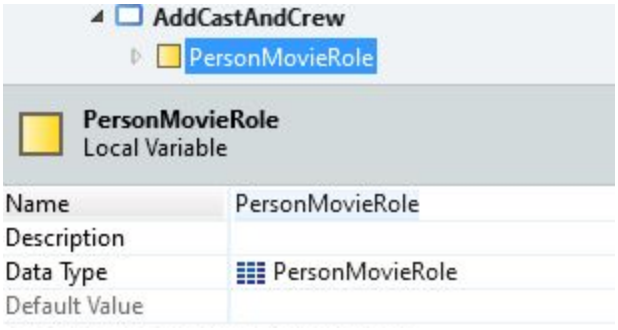
1. Create a new Empty Screen, called AddCastAndCrew, with an Input Parameter, MovieId. This Screen has two input fields: one to choose the person and one to choose the role of the person in the movie. The Screen should be Anonymous.

    a. Create an Empty Screen, name it *AddCastAndCrew* and set it to **Anonymous**

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

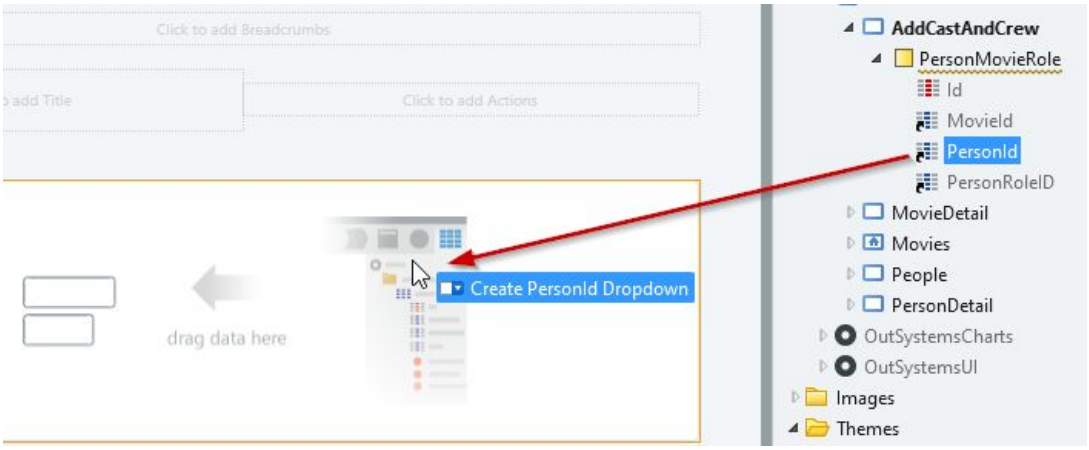


c. Set its **Name** to *PersonMovieRole* and verify the **Data Type** is set to *PersonMovieRole*.

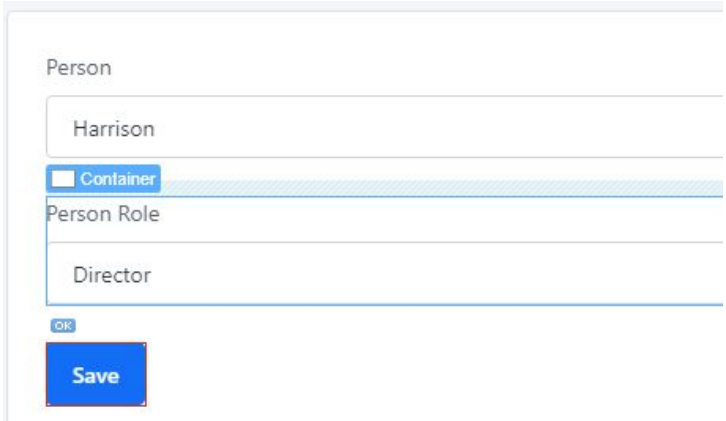


d. Drag a **Form** widget to the main content area of the Screen.

e. Expand the Local Variable and drag the *PersonId* attribute to the Form



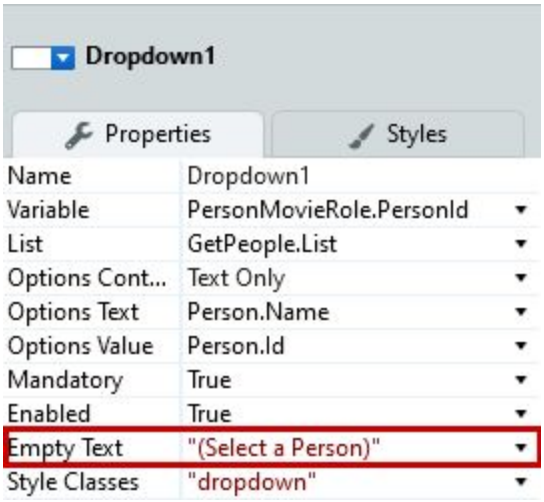
f. Do the same for the *PersonRoleId* attribute. The Save Button has an error, but that will be addressed later.



2. 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)"*



    c. Do the same thing for the Person Role dropdown, but this time with the text *"(Select a Role)"*

3. For the Title of 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 a **MovieId** input parameter to the Screen

    b. Add an Aggregate that fetches the Movie that matches the MovieId, just like we did in the MovieDetail Screen.

    c. In the Title section of the Screen, drag an Expression and set it to be:

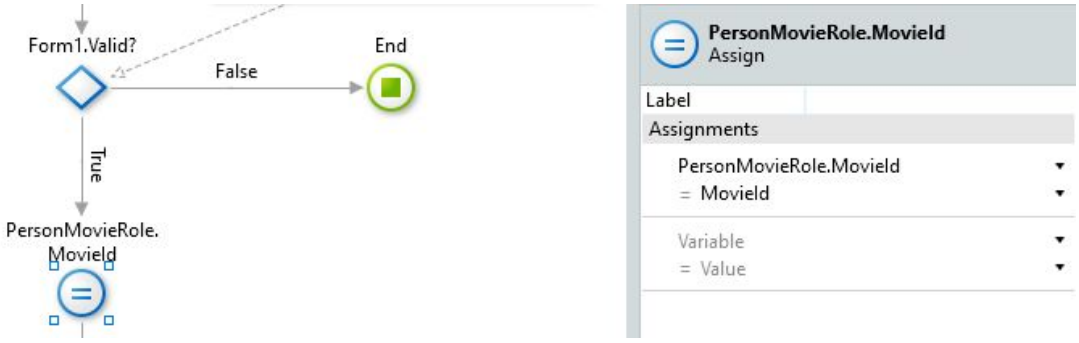
    "Add Cast and Crew to " + GetMovieById.List.Current.Movie.Title

4. Implement the logic to create a PersonMovieRole record in the database, using the information submitted by the end-user in the AddCastAndCrew Form. Since the Entity is exposed as read-only, we need to create a Server Action in the Core module with the server-side logic.

    a. Double click the **Save** button to create the *SaveOnClick* Screen Action

    b. Drag and Assign and drop it after the If in the Action flow. Define the assignment to be:

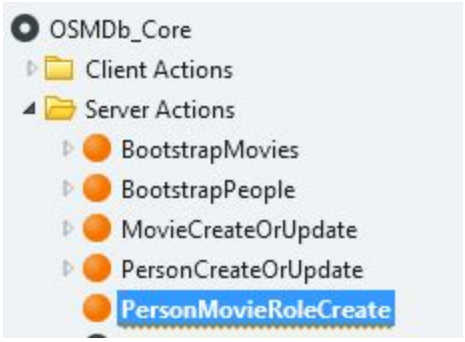
    PersonMovieRole.MovieId = MovieId



NOTE: This assignment is necessary, since the Form only has two input fields: person and role. However, the record also has the MovieId attribute, which is mandatory in the database. Since we know the MovieId from the input parameter of the Screen, we can use that to complete the information of the record.

---

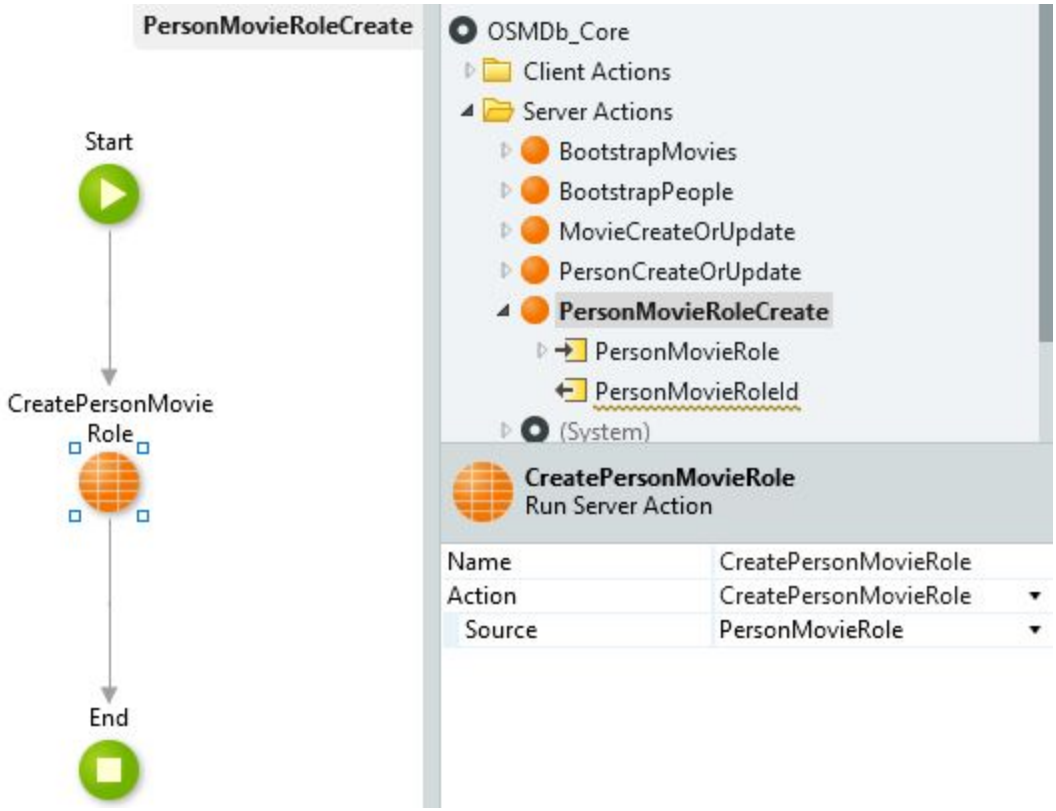
    c. Switch to the **OSMDb_Core** module, open the Logic tab and create a new Server Action called *PersonMovieRoleCreate*



    d. Set the **Public** property of the Action to *Yes*.

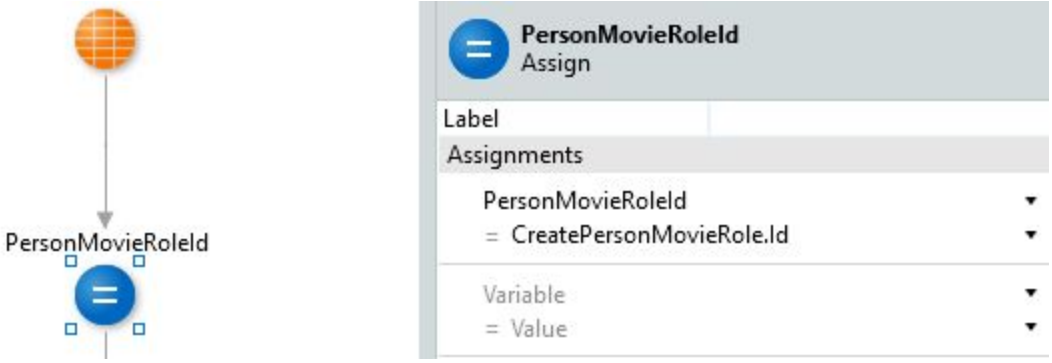
e. Add an Input Parameter called *PersonMovieRole* and an Output Parameter called *PersonMovieRoleId*.

f. Drag the CreatePersonMovieRole Entity Action and drop it on the recently created Action flow. Set its **Source** to be the Input Parameter *PersonMovieRole*.



g. Drag an Assign and drop it after the CreatePersonMovieRole Action. Set the assignment to be:

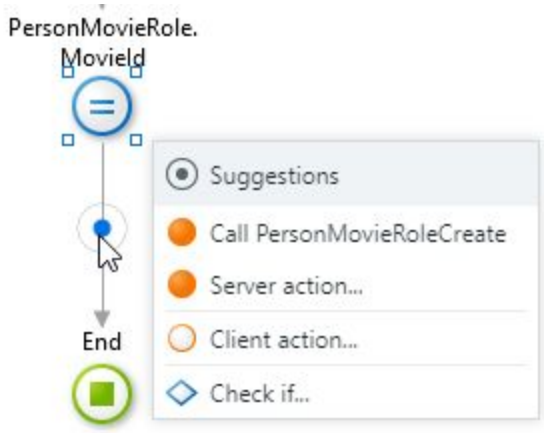
PersonMovieRoleId = CreatePersonMovieRole.Id



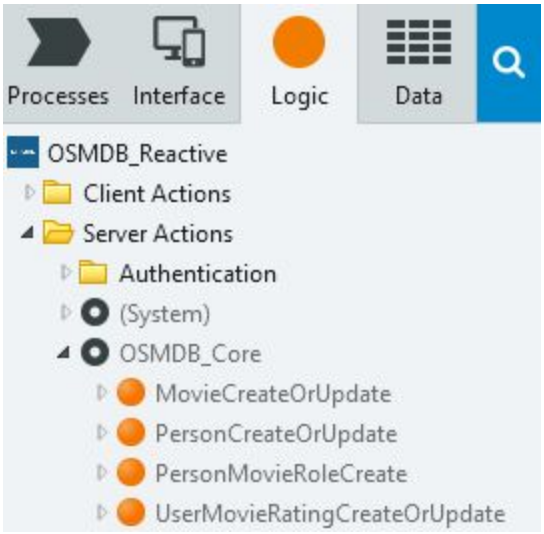
h. Publish the module to save the recent changes.

i. Go back to the **OSMDb** module and reference this recently created Action.

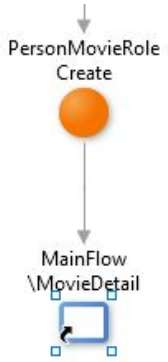
j. In the **SaveOnClick** Action of the AddCastAndCrew Screen, mouse over the last arrow, click the blue icon and choose **Call PersonMovieRoleCreate**

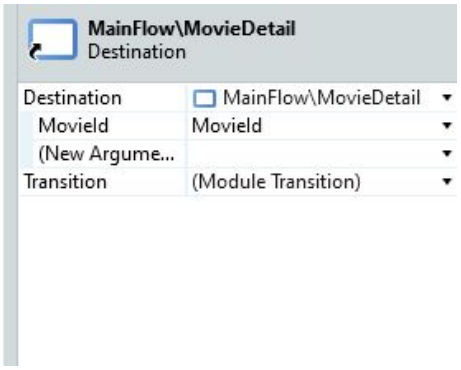


If this option does not appear, select the **Server action...** option. In the new dialog that appears, select the **PersonMovieRoleCreate** Action. As a last resort, if nothing appears, just find the PersonMovieRoleCreate Action in the logic tab, under the Server Actions and the Core module, drag and drop it in the flow.



k. 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.





5. In the MovieDetail Screen, add a link to the AddCastAndCrew Screen so that end-users can navigate to it.

   a. Open the MovieDetail Screen.

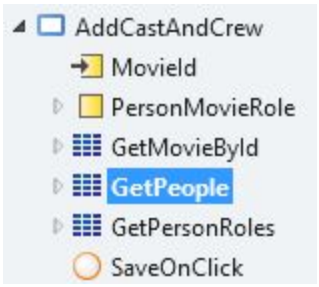
   b. In the Actions area on the top right of the Screen, type the text *Add Cast and Crew to Movie*

   c. Right-click the text and **Link** it to the AddCastAndCrew Screen, passing the MovieId as value of the input parameter.

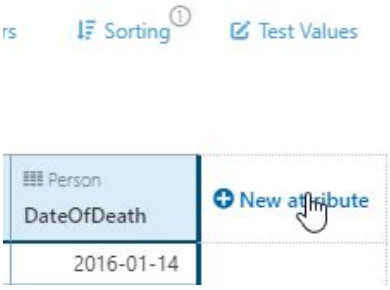
   d. Publish the module and test the application in the browser.

6. Notice that the AddCastAndCrew Screen has the Person dropdown, with just the first name appearing on the options. Let's change it to include the Surname as well.

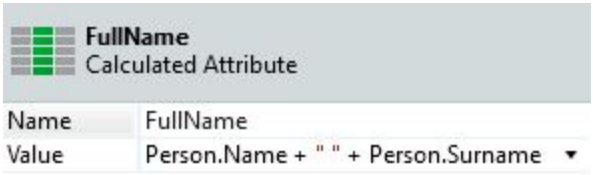
a. Open the AddCastAndCrew Screen and then open the GetPeople Aggregate inside it.



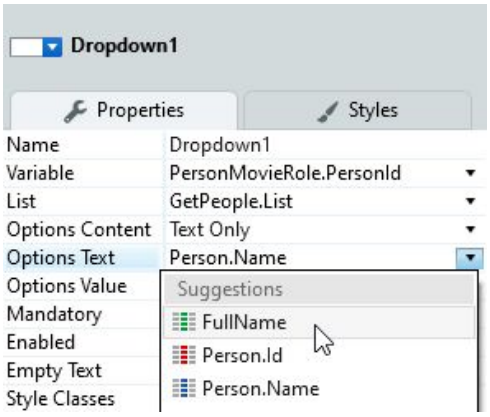
b. In the Aggregate, click on **+NewAttribute**.



c. Set the **Name** of the new attribute to *FullName* and the **Value** to *Person.Name* + " " + *Person.Surname*



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



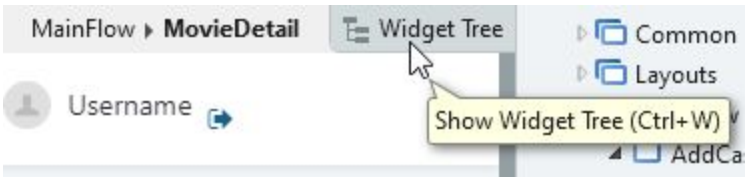
e. Publish the module and test it in the browser.

Listing the Production Talent and the Cast/Crew

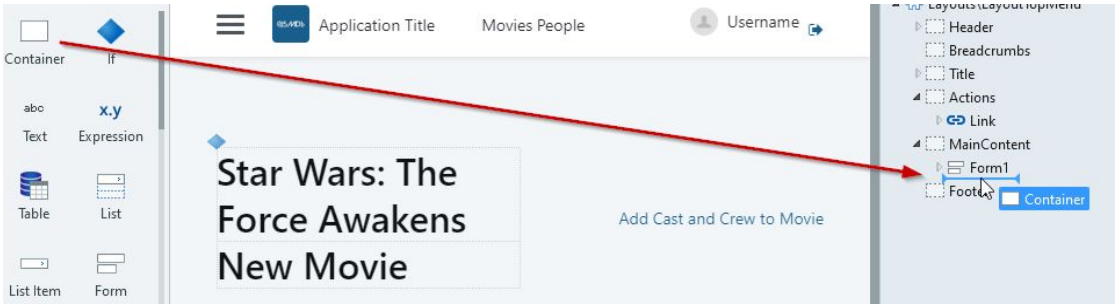
The logic to add a person to a movie with a certain role is done. In this section, we will create the UI components on the MovieDetail Screen to display the directors and producers associated with the movie and the cast/crew of the same movie.

2. In the MovieDetail Screen, create a List to display the full name of the producers and directors involved in the movie.

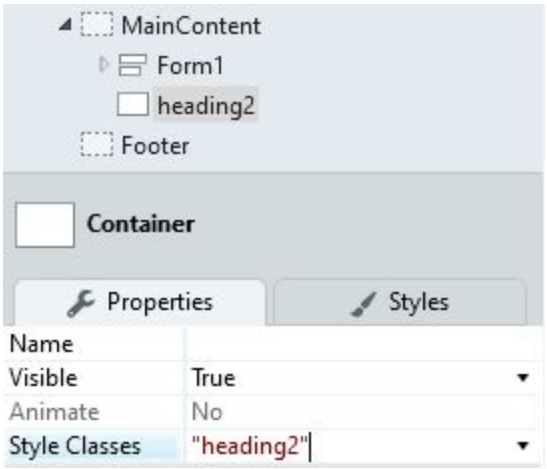
   a. Open the **MovieDetail** Screen and open the Widget Tree.



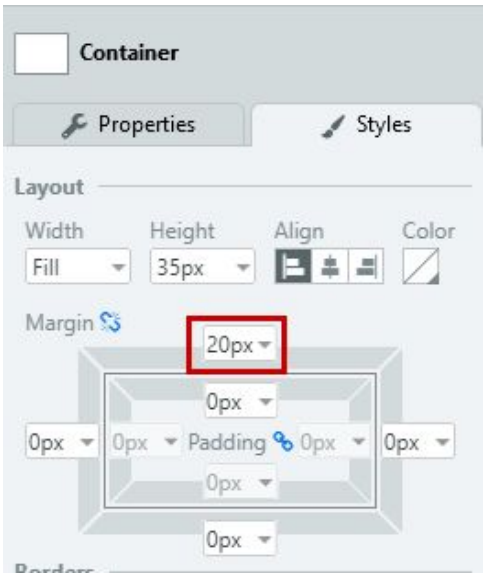
   b. Using the Widget Tree, drag a **Container** below the Form.



   c. Set the **Style Classes** to *"heading2"*

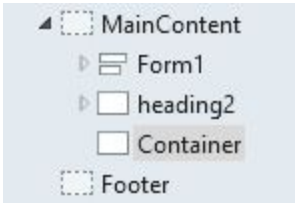


d. Click the **Styles** separator of the Container and add a **Margin top** of *20px*

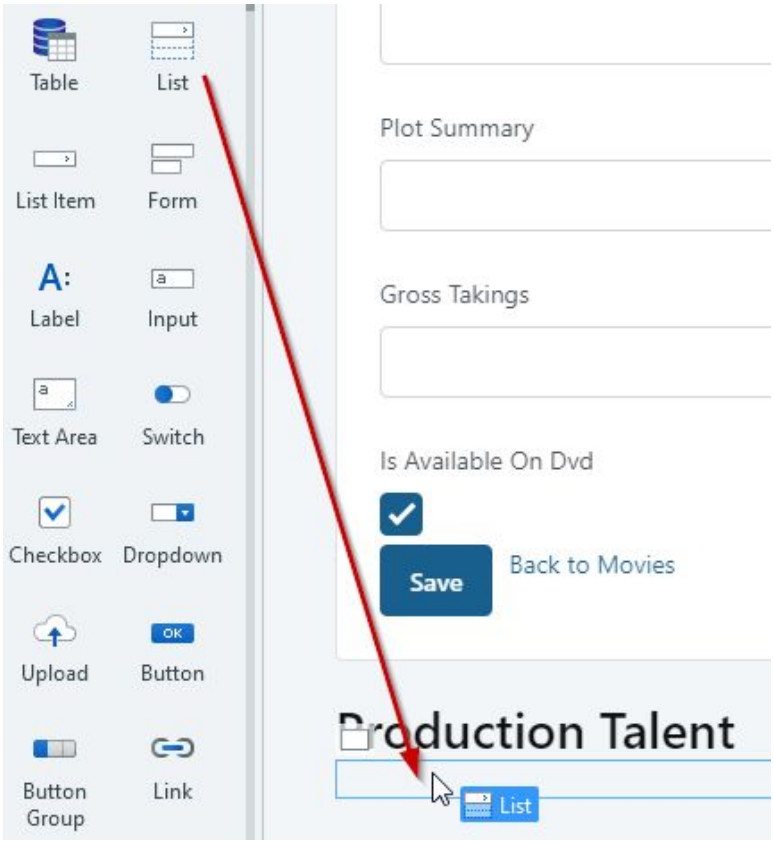


e. Type *Production Talent* inside the Container

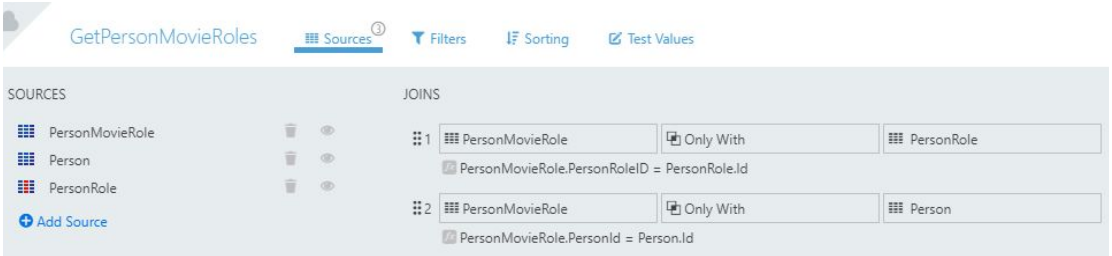
f. Drag a new **Container** below the previous one



g. Drag a **List** inside the new Container. We now need to define a Source for the List.



h. Now lets create an **Aggregate** to get the directors and producers from the database. Add the **PersonMovieRole** and the **Person Entity** to that Aggregate. The **PersonRole** Entity will be added automatically if you use drag and drop to add the other two Entities, otherwise we need to add it as well. Also, note that the Join conditions between these Entities are automatically created.



i. Add a filter to the aggregate to get only the Directors and Producers:

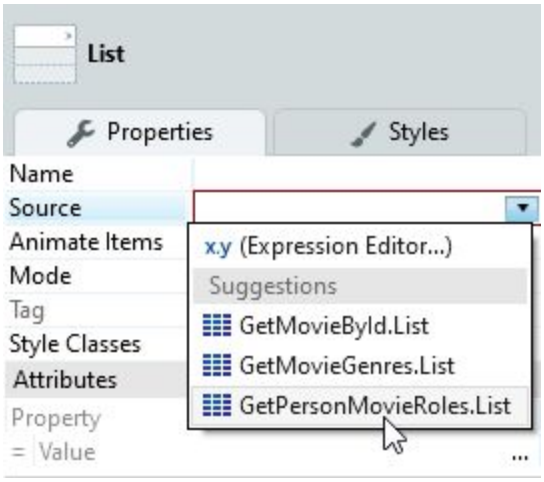
PersonMovieRole.PersonRoleId = Entities.PersonRole.Director or
PersonMovieRole.PersonRoleId = Entities.PersonRole.Producer

j. Add an additional filter to get only this information for the Movie being displayed in the MovieDetail Screen

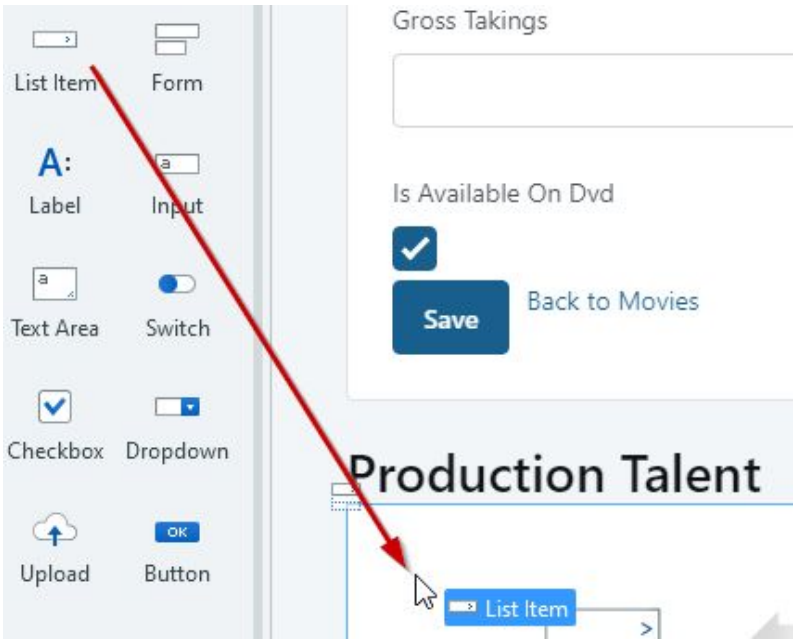
PersonMovieRole.MovieId = MovieId

If the Aggregate changes its name after adding the filters, rename it back to *GetPersonMovieRoles*.

k. Go back to the MovieDetail Screen and set the **Source** of the List to the *GetPersonMovieRoles.List*



l. Drag a ListItem and drop it in the List. Each List Item will have the information for a particular person in the List.

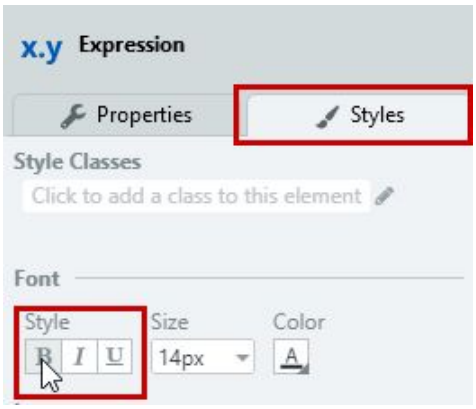


m. Drag an Expression and drop it inside the List Item.

n. The value of the Expression should be:

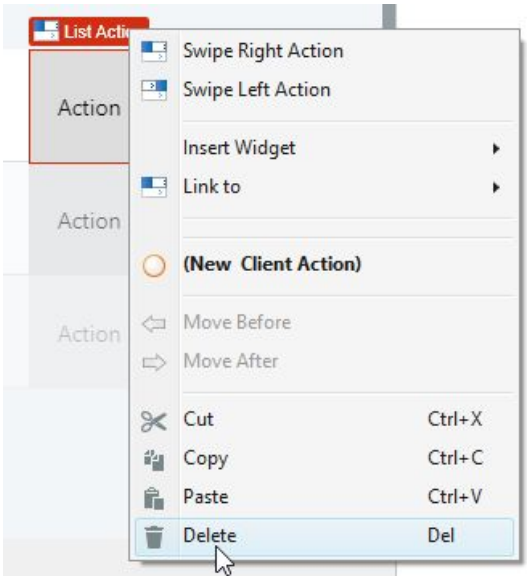
GetPersonMovieRoles.List.Current.Person.Name + " " +
GetPersonMovieRoles.List.Current.Person.Surname + " (" +
GetPersonMovieRoles.List.Current.PersonRole.Label + ")"

o. Select the Expression, switch from the Properties to the Styles and make it **Bold**.



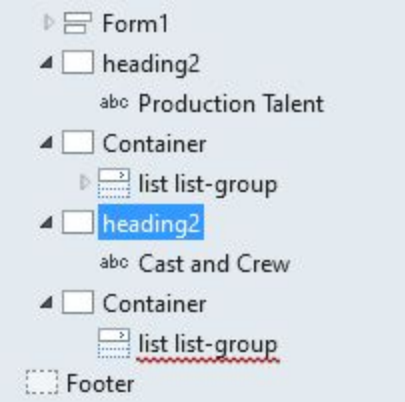
p. Right-click on the Expression and Link it to the PersonDetail Screen. Set the PersonId parameter to *GetPersonMovieRoles.List.Current.Person.Id* to pass to the Screen the Id of the person. This Id will be used to fetch the data about the person and display it.

q. Right click the **List Action** on the right of the List Item and **Delete** it. This is useful for swiping behavior, which we will not use in this particular exercise.

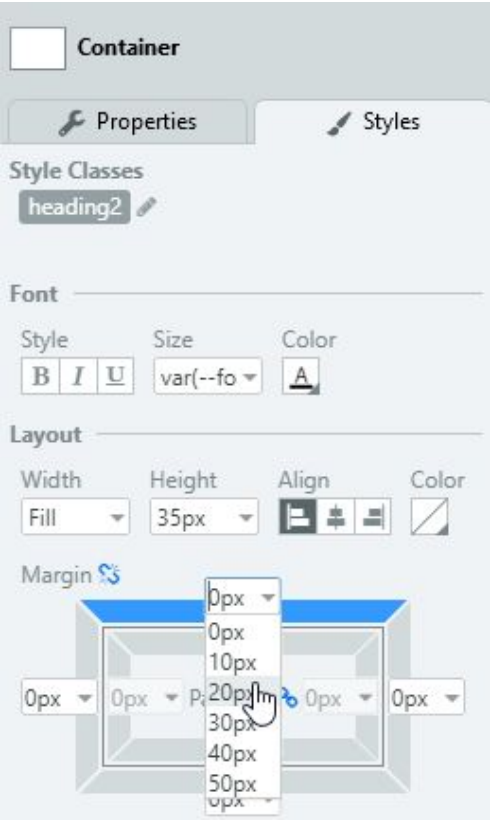


3. In the MovieDetail Screen, create a List to display the full name of the cast and crew involved in the movie.

    a. Similarly to what was done for the Production Talent, using the Widget Tree, create the following structure of widgets in the MovieDetail Screen. The first is a Container with **Style Class** set to *"heading2"* and the text *Cast and Crew* inside it. The second Container has a **List** widget inside it.



    b. On the new **heading2** Container, open the **Styles** tab and set the **Margin top** to *20px*.

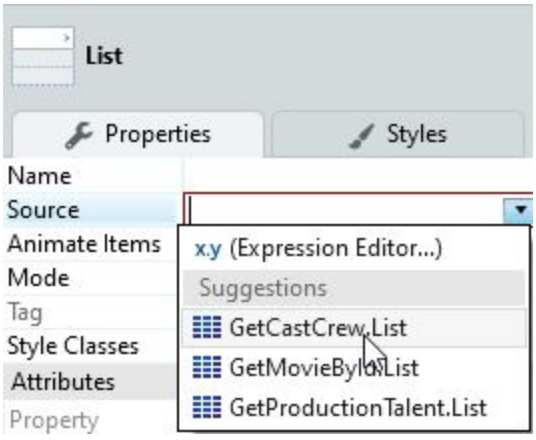


c. Rename the **GetPersonMovieRoles** to *GetProductionTalent* and create a new Aggregate similar to that one, but this time filter the results by **Actor** and **Crew**. Set the name of the new Aggregate to *GetCastCrew*

d. Add a new filter to make sure that only the Movie being displayed on the Screen is fetched

PersonMovieRole.MovieId = MovieId

e. Set the new List **Source** to *GetCastCrew.List*

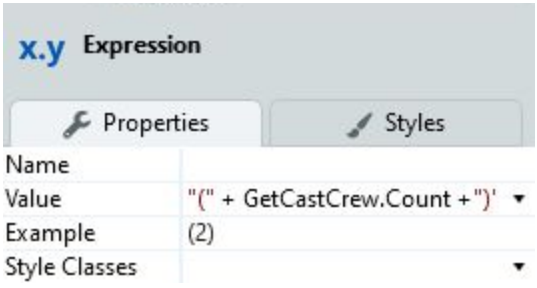


f. Use the same strategy that we used above to display the name of the person and the role. First use a List Item widget, without the List Action element on the right, and define an Expression with the Name, Surname and the Role between parenthesis. Then, Link the Expression to the PersonDetail Screen. At the end, the List should look like this.

g. Drag an Expression next to the Cast and Crew Text and set its value to:

"(" + GetCastCrew.Count +")"

h. Set the Example property of the Expression to *(2)*



i. Publish the module and test it in the browser!