Data Queries and Widgets II

Indiana Jones and the Last Crusade

Title *

Indiana Jones and the Last Crusade

Year *

1989

Plot Summary

When Dr. Henry Jones Sr. suddenly goes missing while pursuing the Holy Grail, eminent archaeologist Indiana Jones must follow in his father's footsteps and stop the Nazis.

Genre

-

Gross Takings

197171806

Is Available On DVD

☑

Save Back to list

Production Talent

Steven Spielberg (Director)

George Lucas (Producer)

Cast and Crew (4)

Harrison Ford, Janet Tebrooke, Richard Brierley, Sean Connery

Table of Contents

Introduction

In this exercise lab, we will extend our OSMDb application to enable adding participants to a movie. Those participants will consist of People that will have a certain Role in a Movie. This will require adding a new Screen and some Combo Boxes to allow the user to select the data that he or she desires.

Also, we will add more information to the MovieDetail Screen. First, we will display a Table with the Production Talent of the movie: Directors and Producers. Then, we will add a list of the cast and crew of the movie, where we will use for the first time a SQL Query and a List Records.

Finally, we will also extend the movie search filter, by adding an option to choose the movie genre and another one to search for movies available (or not) in DVD.

In this specific exercise lab, you will:

- Create a Screen with logic to add participants to a movie
- Use Aggregates and SQL Queries to fetch movie participants
- Display participants by role, as detail sections of MovieDetail
- Add extra search and filter capabilities to the movies list

Add Participants to a Movie

In this section, we are going to allow users to add people to the roster of participants in a movie. This functionality will be developed in a new Screen named *AddMovieParticipant*, where users will be able to choose the person and which role he/she plays in a movie.

1) Create a *Add Cast/Crew to Movie* Link to the Actions area (top right) of the **MovieDetail** Screen. Configure it to navigate to a new Screen **AddMovieParticipant**, passing the **MovieId** from the MovieDetail Screen as Input Parameter.

   a) Open the **MovieDetail** Screen, which can be found under the Interface tab.

   b) In the **Actions** placeholder, type in *Add Cast/Crew to Movie*.

   c) Right-click the previously added text and select *Link to > (New Web Screen)*. This will create a Link which navigates to a new Screen when clicked.
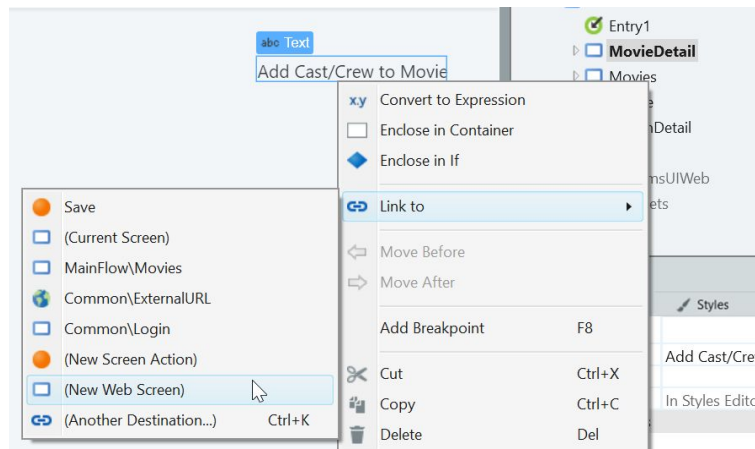

   d) In the New Screen window, select the **Empty** template, name the screen *AddMovieParticipant*, and click the **Create Screen** button.

   e) Check the **Anonymous** role check box in the Web Screen properties.

   f) Open the MovieDetail Screen again and select the Link created.
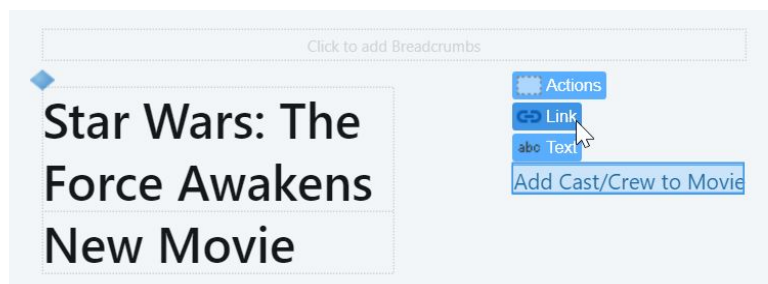
g) In the Properties editor, set the **(New Argument)** property of the **Destination** to *MovieId*. This will pass the movie to which we will add a new cast or crew member to the new Screen.
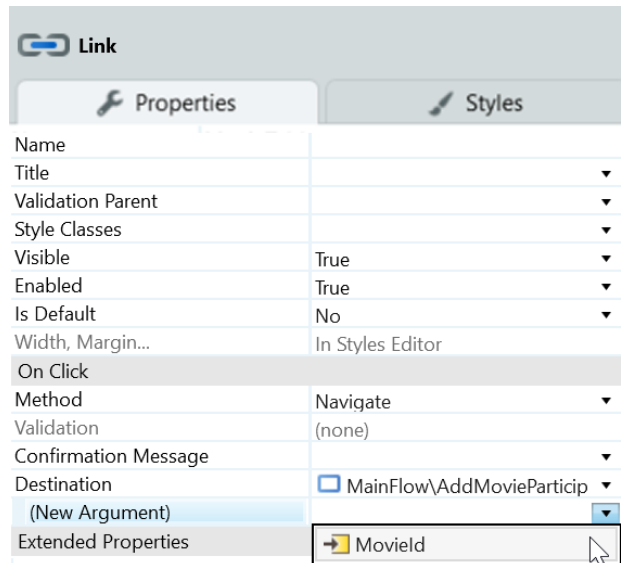
h) Notice that a new Input Parameter named **MovieId** was added to the **AddMovieParticipant** Screen**.**
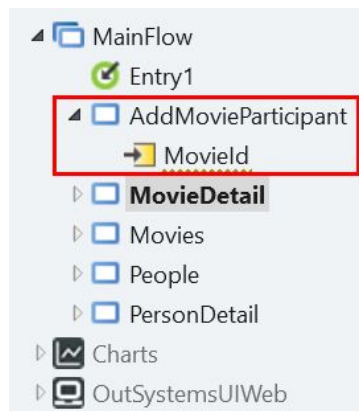
NOTE: These are two examples of OutSystems allowing the creation of "target" elements without leaving the "source" context. We created a new Screen and a new Input Parameter, without leaving the MovieDetail Screen.

The (default) names of these elements are based on the source from where they were generated. Since we pass the MovieId value to the AddMovieParticipant Screen, OutSystems creates the Input Parameter with that name. However, these can be changed if needed.

2) Since we want to allow users to add participants to a movie, we need to fetch the specific movie, using the MovieId Input Parameter. For that, we will create a **Preparation** with an Aggregate that gets the desired movie.

   a) Add a **Preparation** to the **AddMovieParticipant** Screen.

   b) Drag and drop an **Aggregate** to the flow. Open it and then drag and drop the **Movie** Entity to the Aggregate.

   c) Add a filter to fetch the movie specified by the **MovieId** parameter:

   Movie.Id = MovieId
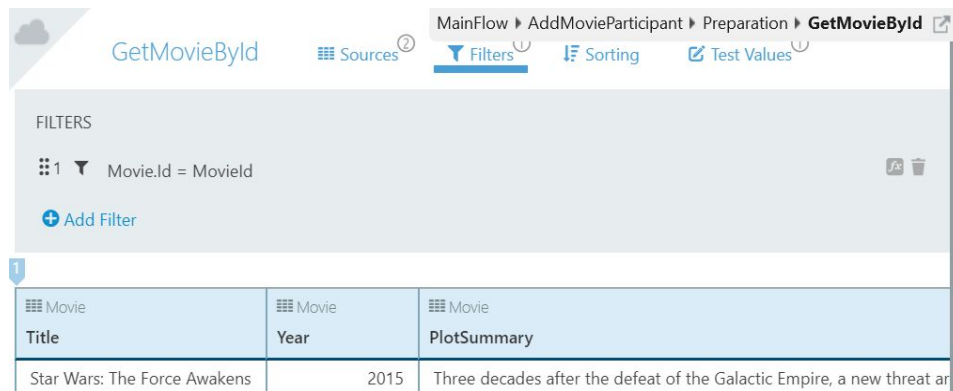
3) At this point we already have the movie. Now we need the end-user to be able to select the Person and its Role in the movie. Since the PersonRole is a Static Entity, it is easy to access the roles even at design time. However, we need to use an Aggregate to get all people in the database. On top of that, we want the Aggregate to return the concatenated name of the people in the database.

   a) Go back to the Preparation of the AddMovieParticipant Screen.

   b) Switch to Data tab, and drag and drop the **Person** Entity in the Preparation flow, below to the previous Aggregate.
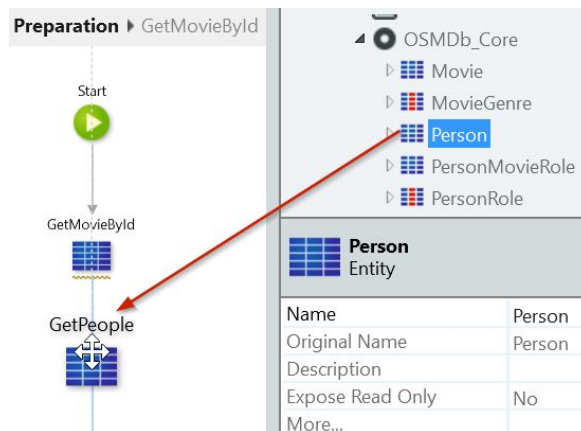
c) Double-click the **GetPeople** Aggregate to open it in the main editor.

d) Click the **New attribute** in order to create a computed attribute. In OutSystems, one can add new attributes to the records returned by the Aggregate, based on the value of the other attributes. In our case, the computed attribute will have the name and surname together. This attribute will be later used in the Screen.

e) Name it *FullName* and then click on the **Add formula**.

f) Enter the following expression *Person.Name + " " + Person.Surname* for the computed attribute.

4) Set the **Title** of the AddMovieParticipant Screen to be *Add Cast/Crew,* plus the name of the movie fetched by the GetMovieById Aggregate (with a space in between). This way, it will be clear to which movie the user is adding participants.
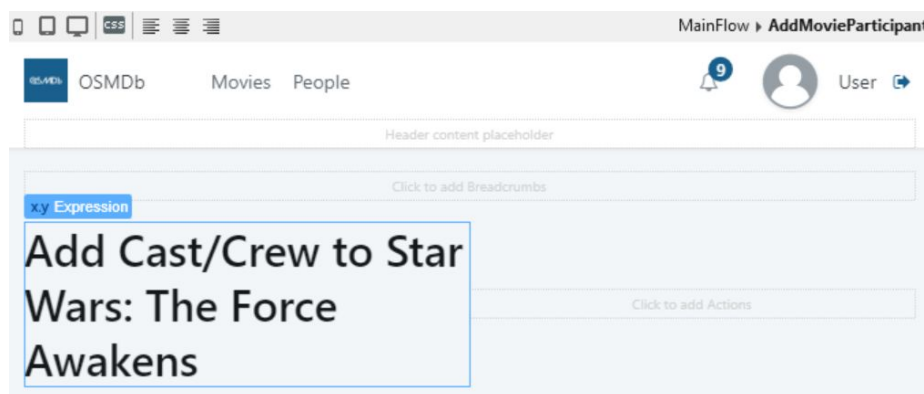
a) Open the **AddMovieParticipant** Screen.

b) Drag and drop an Expression to the **Title** area of the Screen.
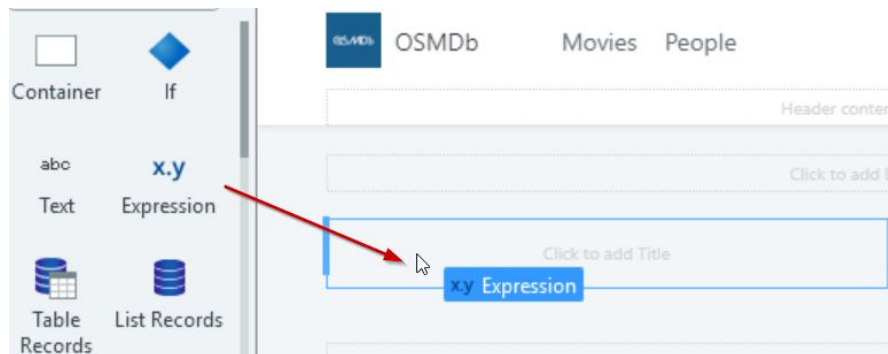

c) Set the Expression to:

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

This Expression guarantees that the Title of the Movie is highlighted at the top of the Screen, to make sure everyone knows the movie to which we will add a member of a cast or crew. The Title of the Movie is fetched from the Aggregate in the Screen Preparation.

5) Now, let's add two new **Combo Boxes** to the AddMovieParticipant Screen. One will select the Person and the other one the Role. Both Combo Boxes will be defined inside a Form, which will have as **Source Record**, a Local Variable of type **PersonMovieRole**. This guarantees that the Form holds a record of PersonMovieRole data type, with all the information to be added to the database in the next step.

a) Add a **Local Variable** to the AddMovieParticipant Screen and name it *Participation*. Set its **Data Type** to *PersonMovieRole*.
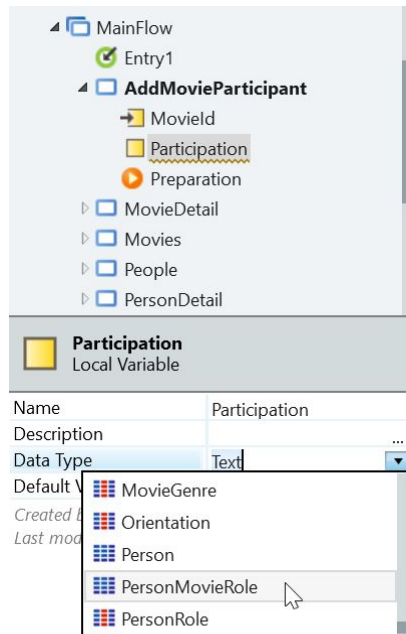
b) Drag and drop a **Form** into the MainContent area of the Screen. Set the Form **Source Record** to the *Participation* Variable.

c) From the Data tab, drag the **PersonRoleId** attribute from the **PersonMovieRole** Entity into the Form. This will create a **Label** and **Combo Box** in the Form. This Combo Box will have all the Roles in the database, and the user will be able to select which role he wants to give to a Person.
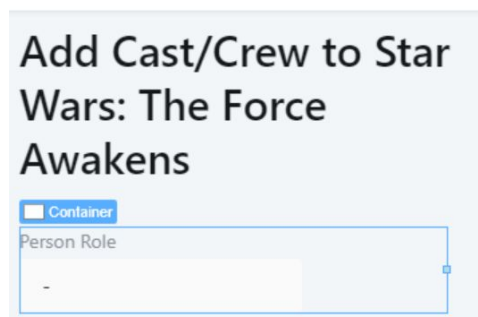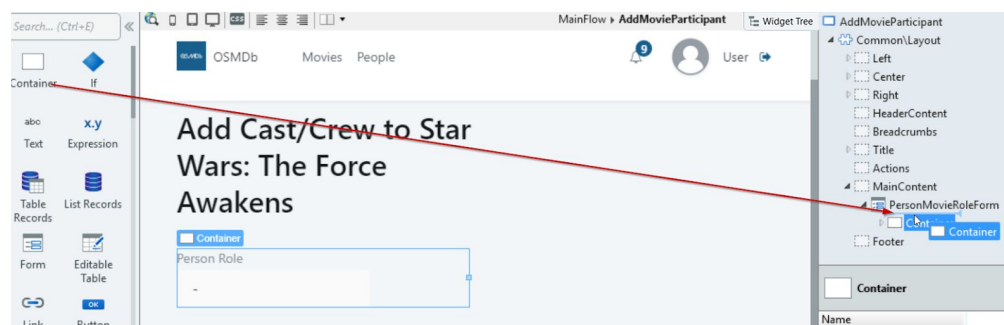

d) Drag and drop a **Container** *inside* the Form, directly above the Container generated on the previous step, to accommodate the Input for the Person. To accomplish this step, we can drag and drop the Container into the Canvas or into the Widget Tree. Sometimes placing the widgets using the Widget Tree is easier. Use **one** of the approaches.

i) Using the canvas.
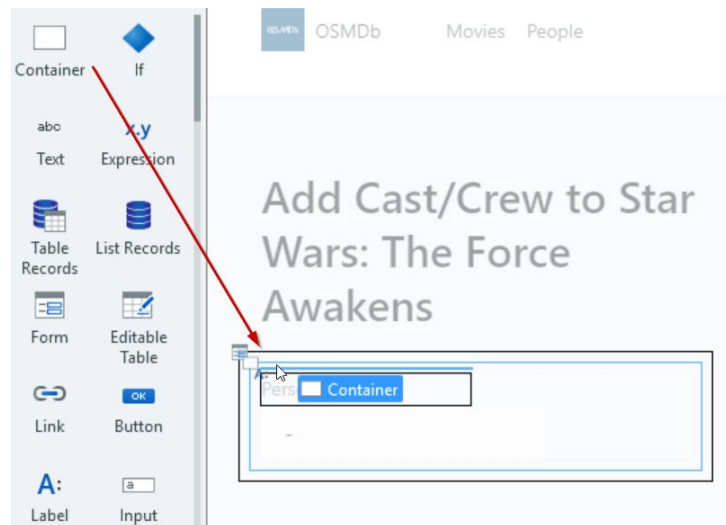
ii) Using the Widget Tree.

e) Drag and drop a **Label** and a **Combo Box** inside the new Container, to emulate the layout generated when dragging the **PersonRoleId** attribute.

f) Set the following Combo Box properties
    Source Record List to *GetPeople.List*
    Source Attribute to *FullName*
    Source Identifier Attribute to *Person.Id*
    Variable to *PersonMovieRoleForm.Record.PersonId*

This will set the source of data of the Combo Box to the list of People fetched from the Database. The options displayed in the Combo Box will appear with the *FullName* computed attribute, with the first and last name of the Person. Because this is a Computed Attribute, we need to specify what is the Id of the Entity (*Person.Id*), so that OutSystems can map it with the corresponding record in the database. Finally, the Variable where the value chosen will be saved correspond to the *PersonId* of the Form.

g) Set the following Combo Box additional properties
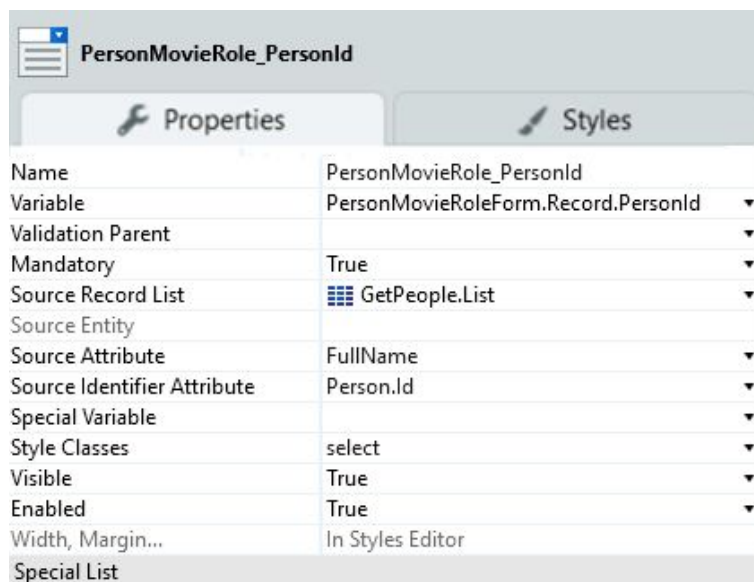>   **Special List Value 1** to *-1*
>   **Special List Option 1** to *(Select Person)*

This allows the Combo Box to start without any person selected. The first option in the Combo Box would be *(Select Person)*. It's worth noting that, in this example, this selection value you could set **Value 1** to anything, since *(Select Person)* is not a Person, just the first thing the end-user will see before expanding the Combo Box. If this value was important, then the Value 1 would need to have a meaning.

h) Change the name of the Combo Box to *PersonMovieRole_PersonId*

i) Change the Label's **Value** property to "Person" and **Input Widget** to *PersonMovieRole_PersonId,* to associate the Label to the Input.

NOTE: A Label's **Input Widget** property indicates the browser where the input focus should be moved to if the label is clicked. It's optional, but a suggested accessibility improvement.

---

j) The **Combo Box** properties should look like this:

k) The layout of the Screen should look like this:

6) Add a Save Button to the **AddMovieParticipant** Screen and configure it to execute a Screen Action On Click. This Screen Action should add a new record to the **PersonMovieRole** Entity, based on the selected options, and then redirects the user to the **MovieDetail** Screen afterwards. Don't forget to set up all the attributes of the Record before adding it to the database.

a) Drag and drop a **Button** below the **PersonMovieRoleForm**. Set its **Label** to *"Save"*

b) Set the Button's **Destination** to *(New Screen Action)*.

c) In the new **Save** Screen Action, drag and drop an **Assign** statement between the Start and End statements.

d) Add the following assignment:

PersonMovieRoleForm.Record.MovieId = MovieId

If we think about the AddMovieParticipant page, the user selects the person and the role for that person in the movie. However, the user does not explicitly select

the Movie. That information comes via Input Parameter. So, we need to ensure that we assign that value to the Form Record, before it is added to the database.

NOTE: Assignments need to be filled in using **Variable** and **Value** pairs: the first is where the value is stored (left side of the assign), the second is the value being stored (right side of the assign). In case of an **Assign** with multiple assignment statements, they are carried out top to bottom.

---

e) Switch to the **Data** tab and expand the **PersonMovieRole** Entity. Drag the Entity Action **CreatePersonMovieRole** immediately after the Assign statement.

f) Set the **Source** input parameter for the **CreatePersonMovieRole** Action to *PersonMovieRoleForm.Record*

g) Drag and drop a **Destination** statement over the existing End statement, to replace one with the other.

h) In the **Select Destination** dialog, double-click the **MovieDetail** Screen to select it as the destination after the Save Action completes.

i) Set the **MovieId** argument of the **Destination** statement to this Screen's *MovieId*.

j) Your Screen Action should look like this
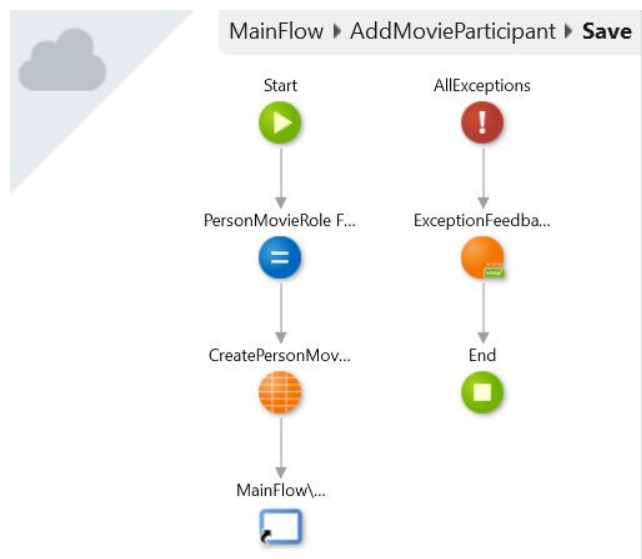
k) Publish the module using the **1-Click Publish** button.

Testing the app: Add Participants to a Movie

Now it's time to test the application, by adding cast and crew members to the movies.

1) Add some cast and crew elements to their respective movies.

a) Open the application in the browser.

b) Add the following **People** to your application:
> *Janet Tebrooke (1950-04-04)*
> *Richard Brierley (1962-11-08)*

c) Select the movie *Indiana Jones and the Last Crusade* and add the following **People** to the movie, with the respective **Person Role**:
> *Steven Spielberg (Director)*
> *George Lucas (Producer)*
> *Harrison Ford (Actor)*
> *Sean Connery (Actor)*
> *Janet Tebrooke (Crew)*
> *Richard Brierley (Crew)*

d) Add *Harrison Ford* (as an Actor) to all his other movies, that you may have in your application, e.g.:
> *Star Wars: The Force Awakens*
> *Raiders of the Lost Ark*

e) Add *Philip Seymour Hoffman* (as an Actor) to all his movies, that you may have in your application, e.g.:
> *Capote*
> *Along Came Polly*

2) Try to add a Person to a movie without its **Name** or **Person Role**.

a) Select a movie and select the Link **Add Cast/Crew to Movie**

b) Do not fill one of the mandatory fields, Name or Person Role, or both, and click on the **Save** Button.

c) Despite being mandatory fields, the application tries to insert the record in the database, and you will get an error message just like in the following screenshot.
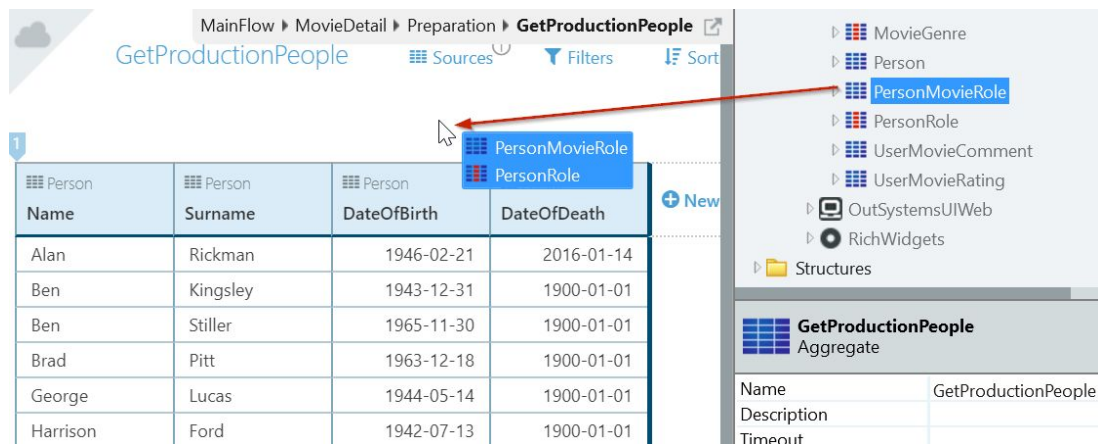

NOTE: Clicking on the **Save** Button without specifying either a Person or a Role, will trigger a database error. This is because attempting to **CreatePersonMovieRole** without filling in all the foreign keys violates a database constraint. We will address later on how we can handle user mistakes while filling Forms, when we cover Form Input Validation.

Display the Participants of a Movie

Users are now able to add participants to a movie. However, there is no way for us to see what we added. When someone opens the MovieDetail Screen to display the details of a particular movie, we would also like to display the participants of that movie. To accomplish that, in the MovieDetail Screen, we will display the movie directors and producers, cast (actors) and crew.

1) Add an Aggregate in the Preparation of the **MovieDetail** Screen Preparation, which returns the directors and producers, in that order, of the movie that matches the Id from the Input Parameter. Add a new computed attribute that returns the full name, plus the Role under parenthesis: e.g. Harrison Ford (Actor). Make sure the Directors appear before the Producers in the output list.

   a) Open the Preparation of the **MovieDetail** Screen.

   b) Drag and drop the **Person** Entity after the **GetMovieById** Aggregate. This will create a new Aggregate. Set its **Name** to *GetProductionPeople*.

   c) Open the **GetProductionPeople** Aggregate in the main editor.

   d) Drag and drop the **PersonMovieRole** Entity to the Aggregate, to add it to the sources. We will need this to find the Roles we want (Directors and Producers).

   NOTE: The **PersonRole** Entity is dragged in together as well, because there is a one to many relationship between the PersonRole Static Entity and the PersonMovieRole Entity.

   e) Click **Filters** and then **+Add Filter** with the following condition, to filter the results by the Movie that we are currently displaying on the page

   *PersonMovieRole.MovieId = MovieId*

f) **+Add Filter** again with the following condition
> *PersonMovieRole.PersonRoleId = Entities.PersonRole.Director or*
> *PersonMovieRole.PersonRoleId = Entities.PersonRole.Producer*

Here we are filtering for the Roles we want: Director and Producer.

g) Click **+New Attribute** at the end of the preview area columns in order to add a new computed attribute.

h) Set the new calculated attribute **Name** to *NameAndRole*.

i) In the properties editor, double-click the **Value** property to open the **NameAndRole Value** dialog. Set the Expression to:
> *Person.Name + " " + Person.Surname + " (" + PersonRole.Label + ")"*

j) Click on the **Sorting** tab and delete the (default) sort by clicking the trash icon.
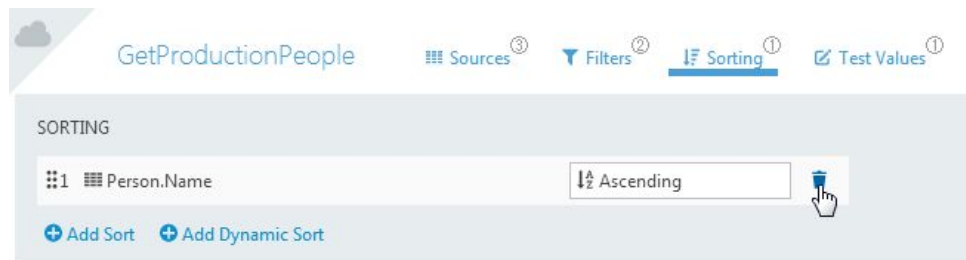
k) Click **+Add Sort**. In the **Select Attribute** dialog, double-click the *PersonRole.Order* attribute to select it. Leave the order as **Ascending**, so that directors appear before producers.
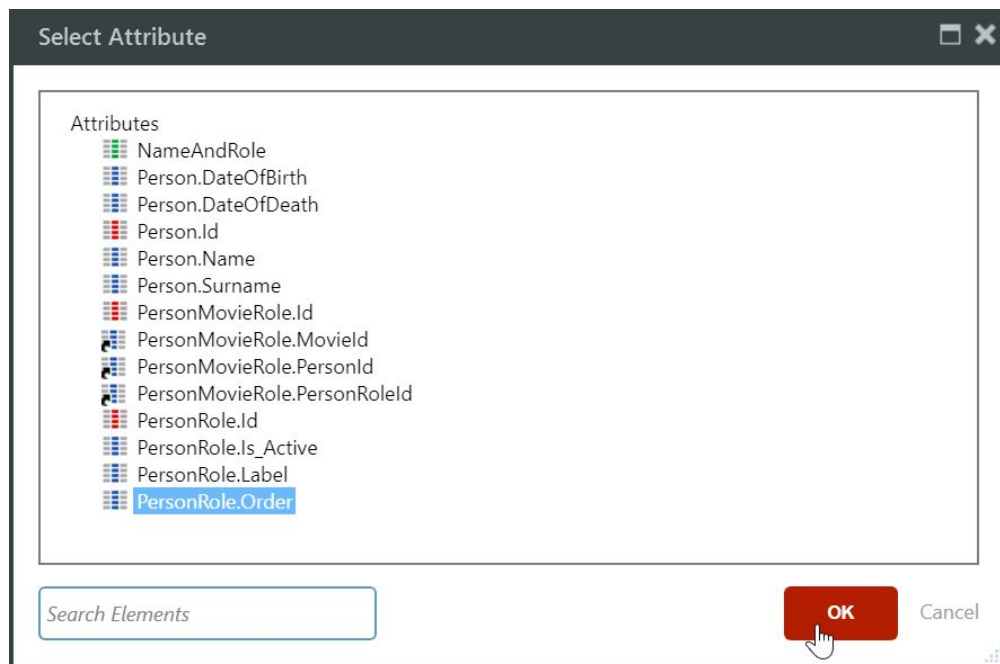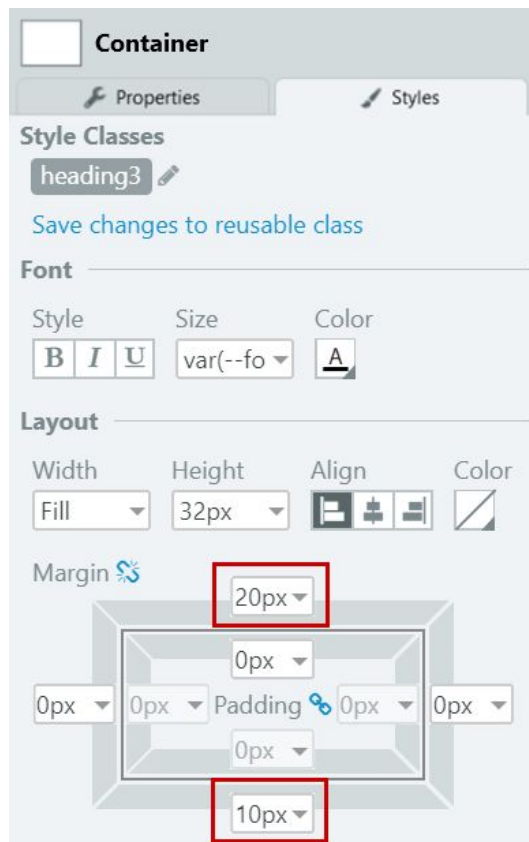
2) Add a *Production Talent* header below the buttons in the **MovieDetail** Screen followed by a **Table Records** to display the output of the **GetProductionPeople** Aggregate. Each row of the Table Records should link to the respective **PersonDetail** Screen, with the appropriate Id.

    a) Drag and drop a **Container** below the Button and Link (and their surrounding Container if exists) in the **MovieDetail** Screen. Type in *Production Talent*.
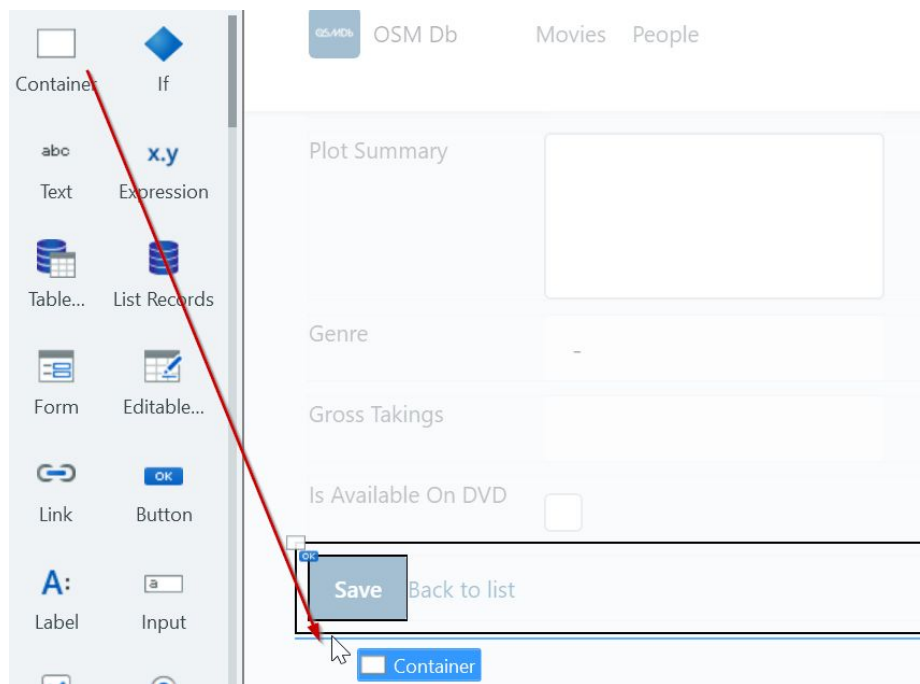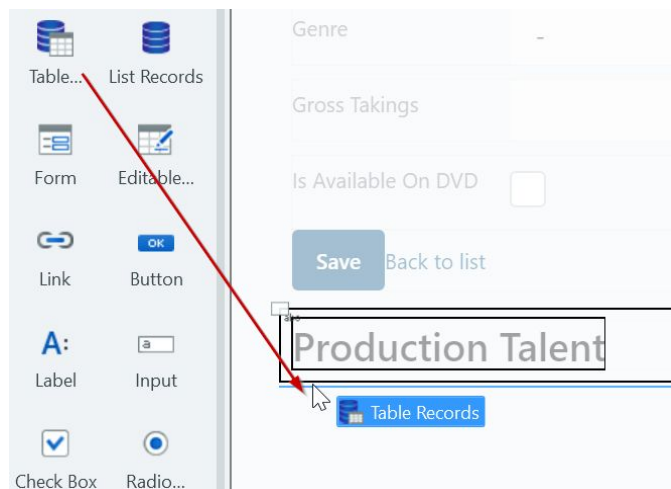

    b) Set the **Style Classes** of the container dragged to *heading3*.
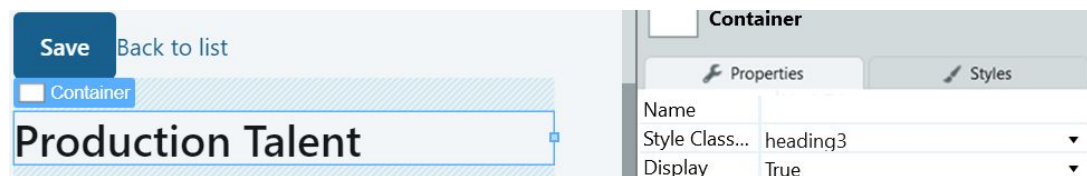
c) Set the Margin Top of the Container to 20 px and the Bottom to 10 px.

d) Drag and drop a **Table Records** below the Container surrounding the **Production Talent** heading.

e) Set the Table Records **Name** to *ProducersTable*, the **Source Record List** to *GetProductionPeople.List* and the **Show Header** to *No*.

NOTE: Since we created a custom header for this section, the property above hides the Table Records header.

---

f) The properties of the **ProducersTable** should look like this
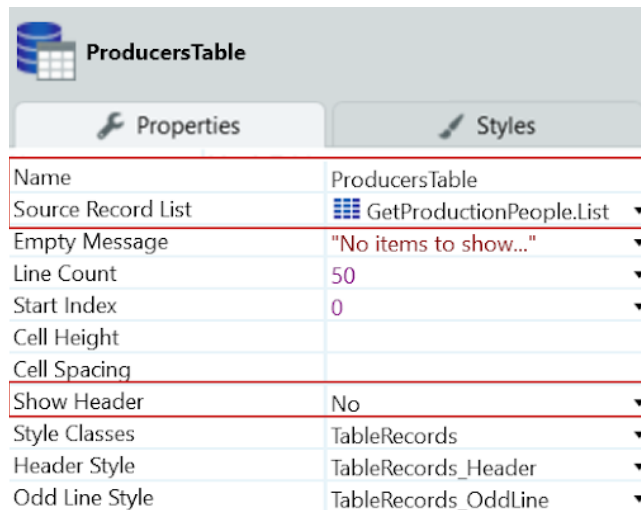
g) With the Table Records still selected, press the **Insert Column to the Left** button in the Main Editor toolbar.
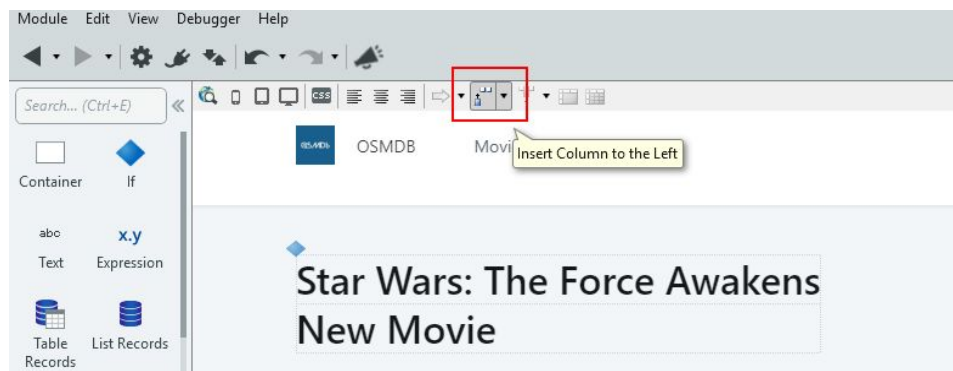
NOTE: Depending on your Screen resolution, some of the toolbar options may be hidden and you may need to expand them.

---

h) Drag an **Expression** Widget to the new column of the Table Records. Set its **Value** to *ProducersTable.List.Current.NameAndRole*. This way, each row of the Table Records will display our calculated attribute.

i) Right-click the Expression, select *Link to (Another Destination...)*. Double-click **PersonDetail** in the **Select On Click Destination** dialog to select it.

j) If needed, open again the MovieDetail Screen and select the link created.
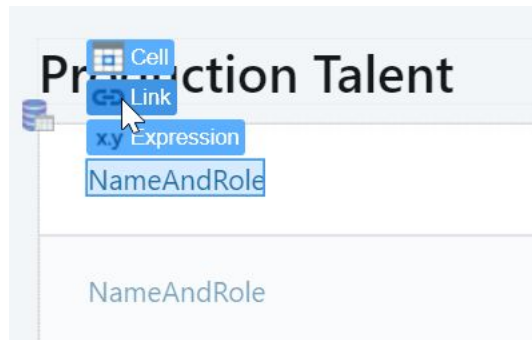
k) In the properties editor for the new Link, set the **PersonId** argument of the **Destination** Screen to

ProducersTable.List.Current.PersonMovieRole.PersonId.

This will pass to the PersonDetail Screen the respective Id of the Person selected.
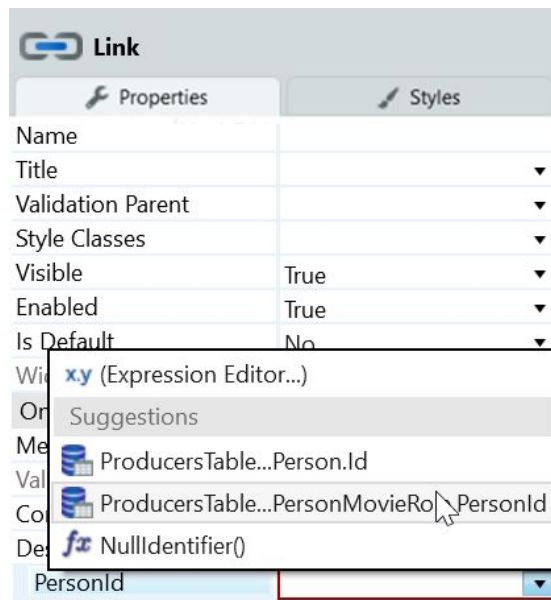
l) Publish the module using the **1-Click Publish** button. Select the movie *Indiana Jones and the Last Crusade*. The lower part of the **MovieDetail** Screen should look like this
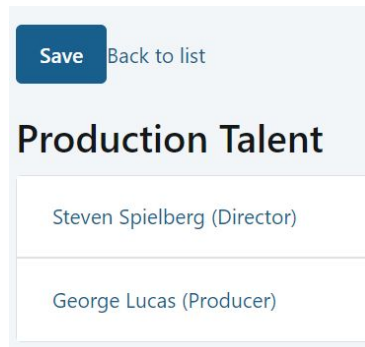

3) Add a SQL Query in the Preparation of the MovieDetail Screen, which returns the people that have a Cast or Crew Role in the movie. The query should have as Input the Movie Identifier and the Roles that we want to look for, and return the Person Id and full name, sorted by the full name.

a) Open the Preparation of the **MovieDetail** Screen. Drag and drop an **SQL Widget** from the toolbox below the **GetProductionPeople** Aggregate.
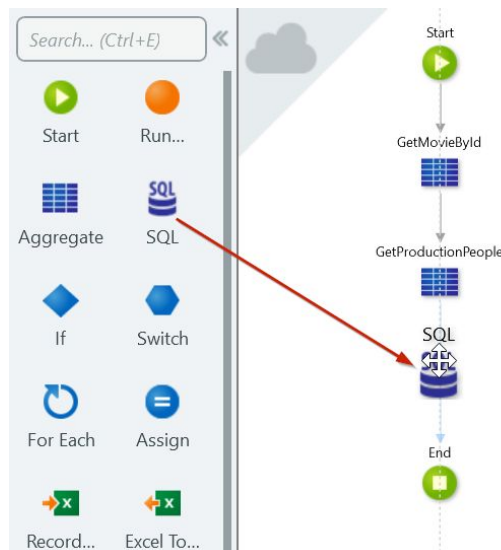
b) Name the new SQL Query as *GetCastAndCrew* and double-click to open the **GetCastAndCrew** SQL in the main editor.

c) Right-click the **Parameters** and add a new Query Parameter and name it *MovieId*. Make sure the type is *Movie Identifier*. This passes to the query the movie we want to filter by.
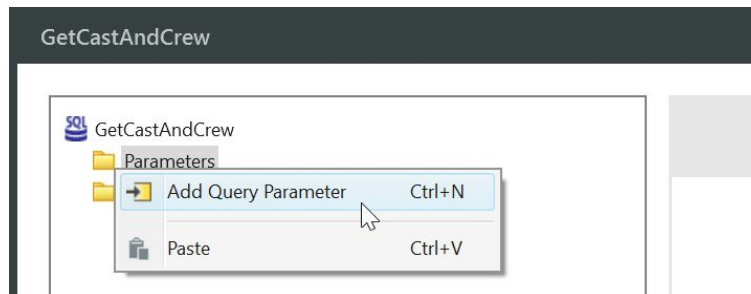
d) Add two more Query Parameters: *ActorRole* and *CrewRole.* These two parameters represent the Actor and Crew Role Id (PersonRole Identifier Data Type).

e) In the editor area insert the following SQL query:

SELECT {Person}.[Id], {Person}.[Name] + ' ' + {Person}.[Surname] Fullname
FROM {Person}
JOIN {PersonMovieRole} ON {PersonMovieRole}.[PersonId] = {Person}.[Id]
JOIN {PersonRole} ON {PersonRole}.[Id] = {PersonMovieRole}.[PersonRoleId]
WHERE {PersonMovieRole}.[MovieId] = @MovieId
AND {PersonMovieRole}.[PersonRoleId] IN (@ActorRole, @CrewRole)
ORDER BY Fullname

This query checks for People that participated in the movie (MovieId) with the Role of Actor or Crew.
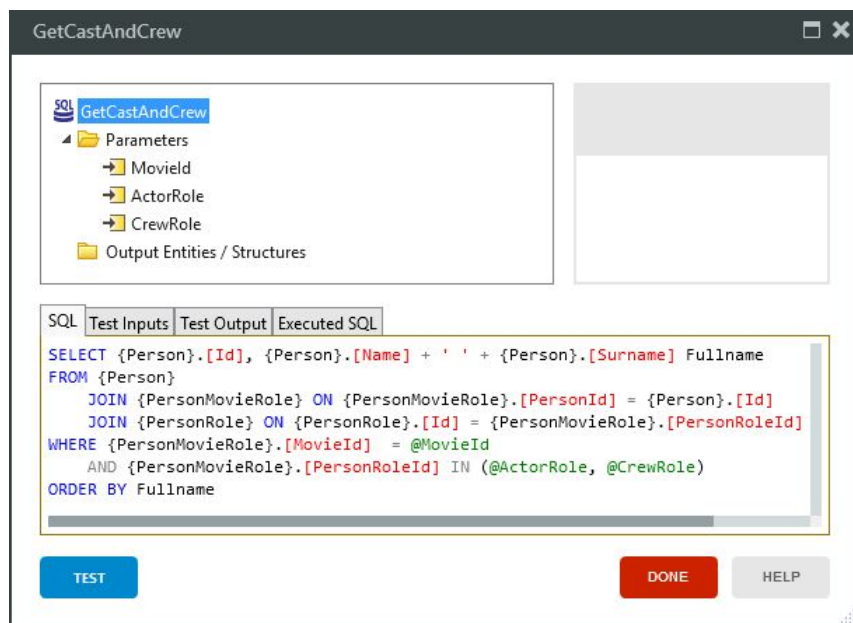
f) Now we just need to define the Output Structure for the query. For that, we will define an Output Structure that matches the SELECT clause of the SQL query.

NOTE: If the SQL statement output considers all the attributes of an Entity, simply define the **Output Entity / Structure** as the Entity itself. If the output is a join between Entities *A* and *B*, just add both.

For this particular case, the SQL statement fetches a **Person Identifier** and a **Text**, thus it is better to define a Structure than using the Entity itself as output. First, the Person Entity has more attributes that would not be filled, and also does not have an attribute that matches the concatenated name.

g) Switch to the **Data** tab, right-click the **Structures** folder and select *Add Structure*.
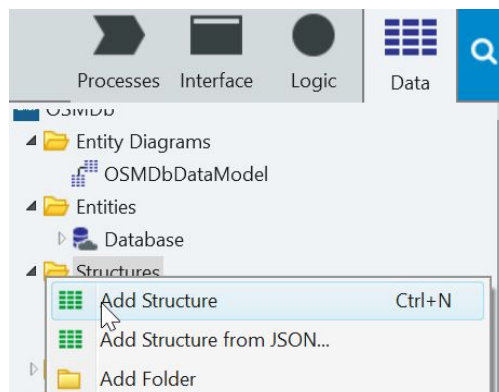
h) Name the new structure *GetCastAndCrew*. Now we have to add two attributes to the **GetCastAndCrew** structure.

i) Right-click the *GetCastAndCrew* structure and select the *Add Structure Attribute* and name it *PersonId*. Make sure its **Data Type** is *Person Identifier*
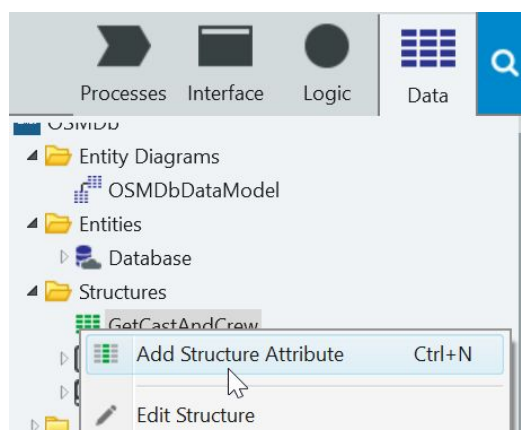
j) Add another attribute and **name** it *PersonFullName*. Make sure the **data type** is set to *Text*.

k) Drag and drop the new structure into the **Output Entities / Structures** folder.
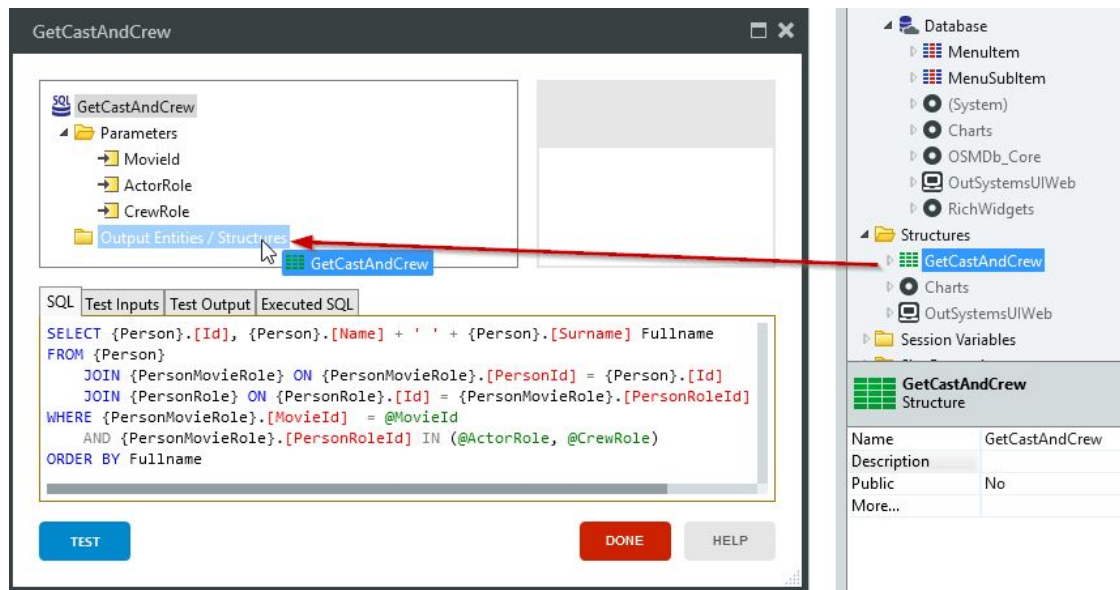
l) In the flow of the MovieDetail's Preparation, select the **GetCastAndCrew** SQL Query. Then, in the properties editor, set the values for the Input Parameters of the GetCastAndCrew SQL Query as follows:

> **MovieId:** MovieId
> **ActorRole:** Entities.PersonRole.Actor
> **CrewRole:** Entities.PersonRole.Crew

4) Create a List Records below the Producers Table to list the cast and crew of the movie, in the MovieDetail Screen. In this list we want to display the full name of the person. Each Person is going to be separated by a comma. This List and the previously created Table should only appear when we are editing an existing movie.

a) Drag and drop a **Container** below the **ProducersTable**. Type in *Cast and Crew* inside of it and set the Container's **Style Classes** to *heading3*.

b) Add a Margin to the top of *20 px* and to the bottom of *10 px*.

c) Drag and drop a **List Records** Widget below the previous Container. **Name** it *CastAndCrewList* and set its **Source Record** List to *GetCastAndCrew.List* and the **Line Separator** to *None*.

d) Drag an **Expression** Widget to the List Records. Set its **Value** to:

> *CastAndCrewList.List.Current.GetCastAndCrew.PersonFullName*

This will display the person's full name on the List.

e) Right-click the Expression, select Link to *MainFlow\PersonDetail*

f) In the properties editor for the new Link, set the **PersonId** property of the **Destination** to *CastAndCrewList.List.Current.GetCastAndCrew.PersonId*.

g) In the Canvas, click next to the recently created Expression, and type a comma and a space. Ensure these two characters are after the Link, but still inside the List Records. The widget tree should look something like this
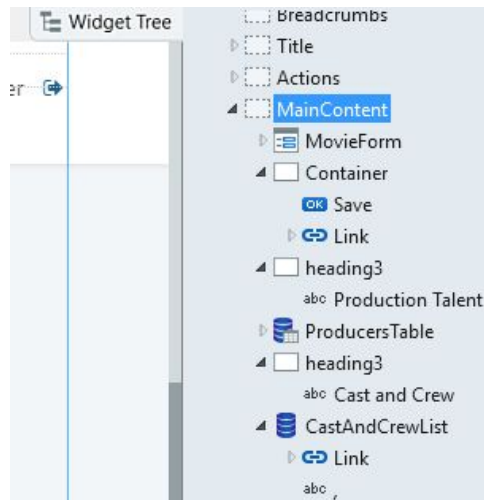
h) Using the Widget Tree, select the Table Records and the List Records alongside the respective headings (to select multiple elements keep the CTRL key pressed). Then, right-click the selection and choose *Enclose in If*.
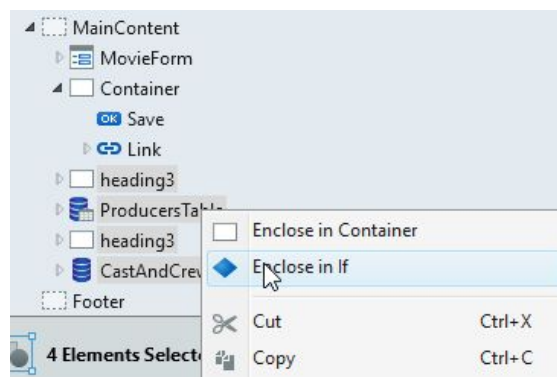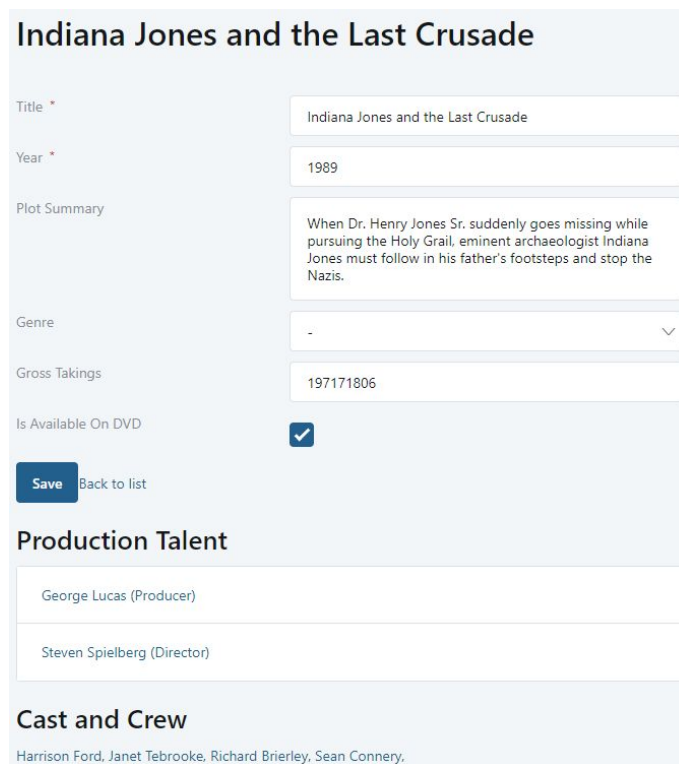

i) Set the **If Condition** property to:

MovieId <> NullIdentifier()

This Condition checks if the MovieId Input Parameter of the Screen is *NullIdentifier()*. We only want to display the Production Table and the Cast and

Crew List if we are editing an existing movie in the database. This way, if a user opens the MovieDetail Screen to add a new movie, nothing appears in that section of the Screen, since the movie has not been created yet.

j) Publish the module using the **1-Click Publish** button.

k) Select the movie *Indiana Jones and the Last Crusade*. The lower part of the **MovieDetail** Screen should look like this

5) In the **CastAndCrewList** List Records, we have a comma after each name, including the one that appears at the end of the Crew list. But the name appearing as last should not display a comma after it. Let's solve this issue.

a) In the Widget Tree, select the **Text** with the comma and select *Enclose in If*.
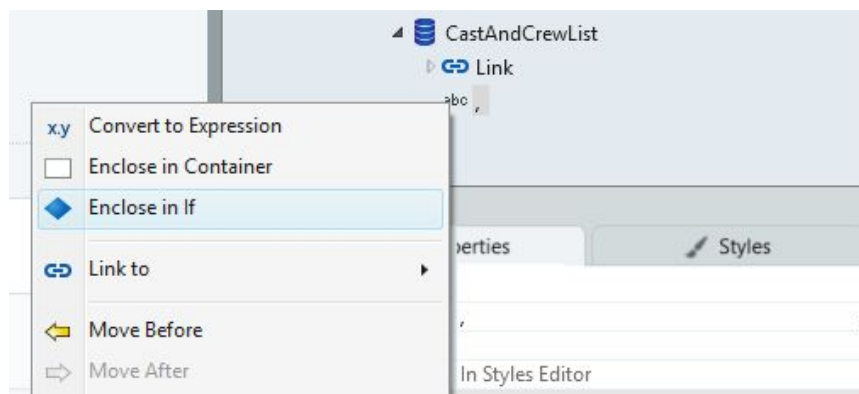
b) Set the **If Condition** property to

CastAndCrewList.List.CurrentRowNumber < CastAndCrewList.List.Length -1

NOTE: The **CurrentRowNumber** field contains the value of the current row number being iterated (in this case by the List Records). The first row has the value 0. In this particular case, we want to display the comma until we get to the last row of the List Records. The last row is indicated by the *Length - 1* expression. The -1 is necessary because if the List has 10 elements (Length = 10), and the CurrentRowNumber starts at 0, then the last row is the ninth.

---

c) Verify that the Text is in the **True** branch, while the **False** branch is empty, like in the following screenshot.

6) Add the total number of Cast and Crew members to the Cast and Crew section Title. This value should appear under parenthesis.

   a) Drag and drop an **Expression** widget next to the **Cast and Crew** title, but still inside the Container.

   b) Set its value to " (" + GetCastAndCrew.Count + ")". Set its example to (3).

   c) Publish the module using the **1-Click Publish** button.

   d) Select the movie 'Indiana Jones and the Last Crusade' and make sure that the Crew list appears without the comma after the last name, and that the total number of cast and crew members is accurate.

# Enhance filter functionality in the Movies Screen

On a previous Lab, we added a filter in the Movies Screen for the movie title and plot summary. Now, we want to add two more. One to filter by Movie Genre, using a Combo Box, and another one to filter by the availability in DVD, using a Check Box.

1) In the Movies Screen, add next to the search input field a **Combo Box** and a **Check Box**. The first will filter by Movie Genre and the second by availability in DVD. Each should be associated to a different **Local Variable** of the appropriate data type.

   a) In the Interface tab, open the Movies Screen.

   b) Add a Local Variable *MovieGenreId* and set its **Data Type** to *MovieGenre Identifier*. Add another Local Variable *IsAvailableOnDVD* and set its **Data Type** to *Boolean*.

   c) Drag and drop a **Combo Box** and a **Check Box** one next to the other.

   d) Select the Combo Box. Set its **Source Entity** property to *MovieGenre* and **Variable** to *MovieGenreId*. All the movie genres will be the source of the Combo Box and the value selected by the user will be stored in the Local Variable. Then, to add the *All* option, to allow a user to search for all genres, set the Combo Box **Value 1** to *-1* and **Option 1** to *- All Genres -* in the Combo Box Special List section. Change its **Width** to *2 col* in the Styles Editor.

e) Select the Check Box. Set its **Variable** property to *IsAvailableOnDVD*. If the user ticks the Check Box, the variable will hold *True*.
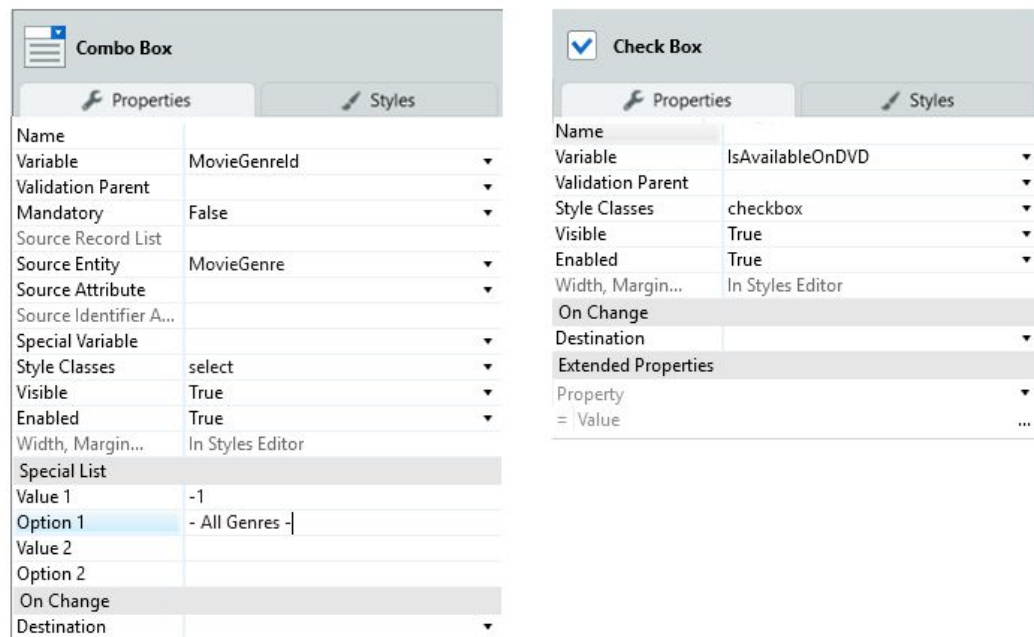
f) Immediately to the right of the Check Box type in *Only on DVD*, to clarify the purpose of that Widget.
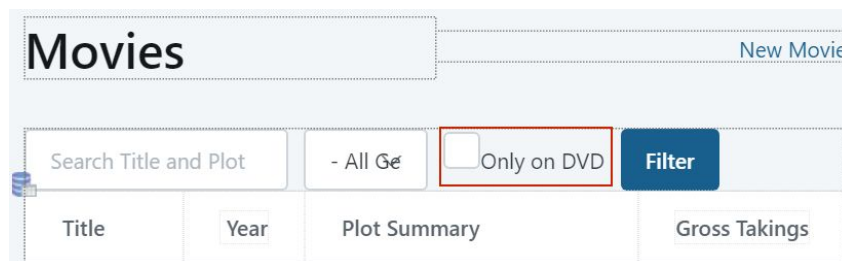
g) Now we need to adjust the **GetMovies** Aggregate to support these new filters. Select the MovieTable in the Movies Screen and click on the small Aggregate icon to open the Aggregate.

h) In the Filters tab, press **+Add Filter** to add a second condition

MovieGenreId = NullIdentifier() or Movie.GenreId = MovieGenreId

NOTE: Remember that, if the user selects an option of the **Special List**, the **Variable** (that tracks "normal" options) will end up with *NullIdentifier()*.

The condition above validates if the user selected '*- Any Genre -*' or an actual movie genre. In the first case, all the movie genres will be included in the output, otherwise only the movies **(Movie.GenreId)** with the chosen movie genre appear.

i) Press **+Add Filter** again to add a third condition

> *IsAvailableOnDVD = False or Movie.IsAvailableOnDVD = True*

NOTE: The condition above validates if the user did not tick the Check Box, or if it ticked it to search for the movies available on DVD. In the first case, all the movies (available on DVD or not) are returned, while in the second case, only the movies available on DVD will appear in the result.

j) The **Filters** area of the Aggregate should look like this
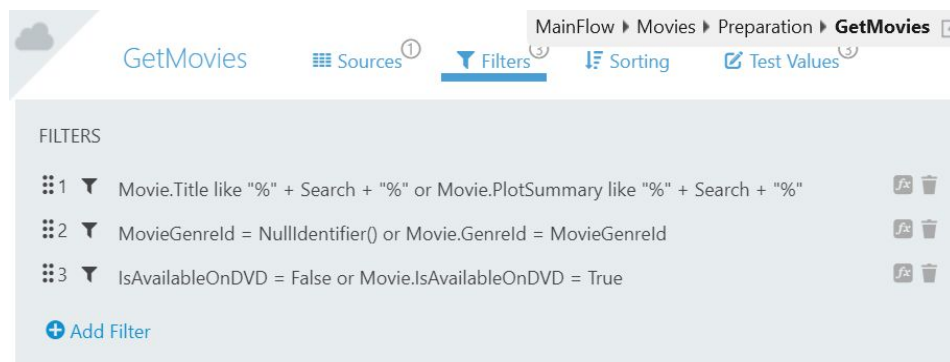
k) Publish the module using the **1-Click Publish** button.

2) Test the filters in the **Movies** Screen, by searching for a movie title or plot, movie genre or if it is available on DVD.

a) When no filters are applied, make sure that all the movies appear in the respective Table Records.

NOTE: This is the main reason why some filters compare the variables associated with the input to the respective Null Value: *IsAvailableOnDvd = False* and *MovieGenreId = NullIdentifier()*. This condition guarantees that if these two fields are not filled, then the conditions in the filters are always true and return every record in the table.

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

> *Along Came Polly*
> *Fight Club*

c) Maintaining the previous filter, select **Drama** in the Combo Box. At least the following movie should appear:

> *Fight Club*

d) Clear all the filters and select the Check Box, to search for the movies available on DVD. All the movies should appear except for **Avatar 2** and **Avengers: Infinity War**. This is just considering the original list of movies, without any changes that you may have done during testing.

e) Uncheck the Check Box and select the genre **Horror** in the Combo Box. Verify that no movie appears (unless you added a Horror movie to the list).

End of Lab

In this exercise, we added a new Screen to add participants to a movie. This Screen can be accessed by a Link in the MovieDetail Screen. In this new Screen, the users can select a person and a role for that movie.

Then, to display that information, we added a Table and a List to the MovieDetail Screen. The Table displays all the directors and producers of the movie, while the List displays the actors and the crew members.

Finally, we extended the search filter to search movies by genre and by availability on DVD.