List and Detail Screens Exercise

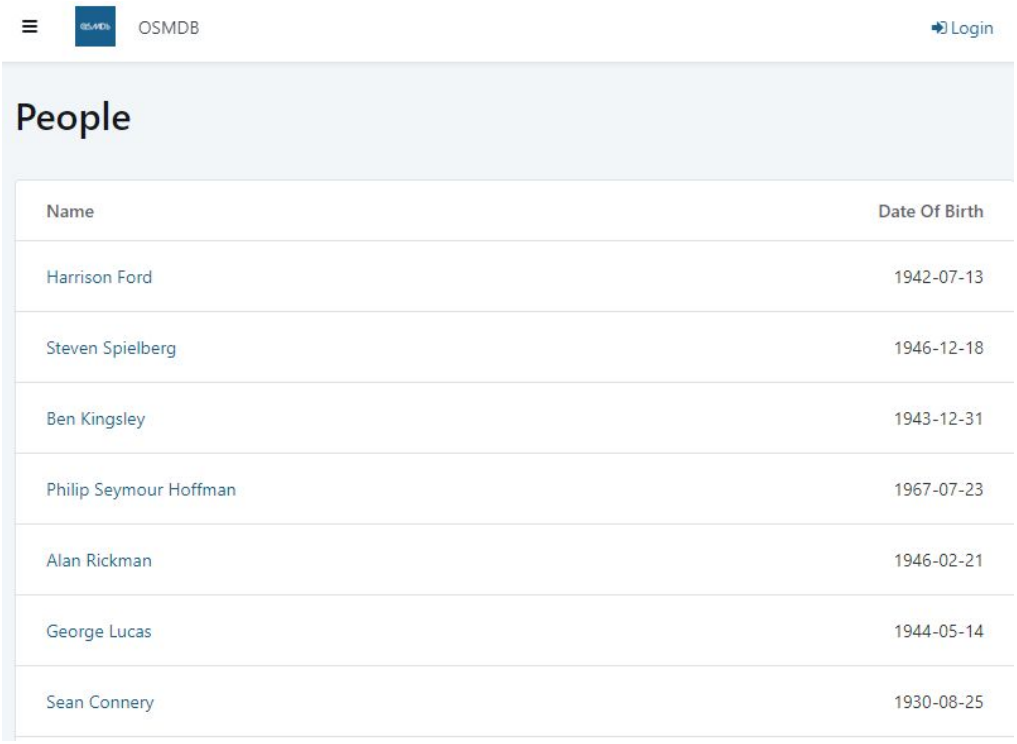


Table of Contents

Introduction

Up until this exercise lab, we already created the application and its modules, and the initial data model with four Database Entities, to model movies and their genres and people and their roles in the movies.

In this Lab, we will start creating the UI of our applications. We will create four Screens, two for movies and two for people. Two Screens will be used for listing the movies and people in the database, while the other two will be to display the detailed information about a specific movie or a specific person.

On top of that, we will also create navigation Links between these Screens, and to the Top Menu of the application, to help the end-user navigate through it.

In this specific exercise lab, we will:

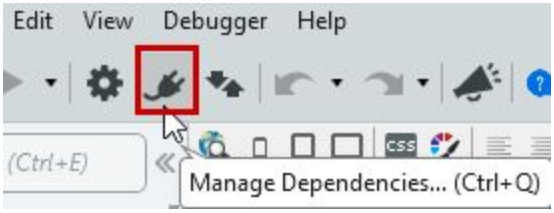
- Reference the Entities created in the core module
- Create list and detail Screens for the main Entities
- Add Screens to the application top Menu
- Publish and test the UI module in the OutSystems web server

Reference Entities from the Core module

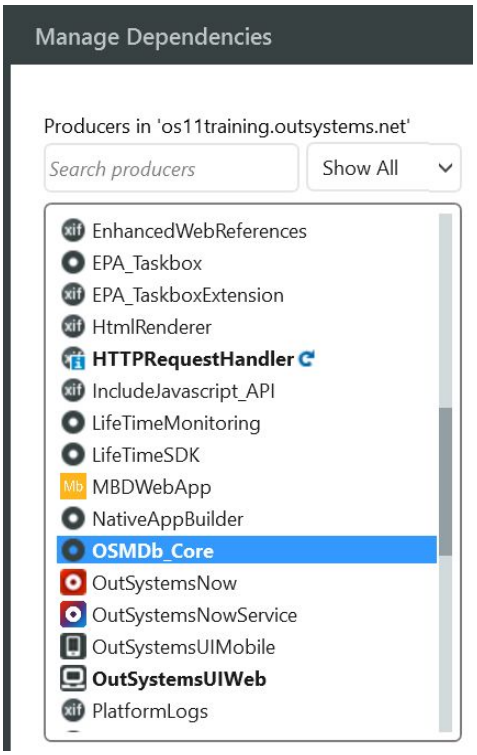
Before starting developing the UI of the application, the Traditional Web module (OSMDb) needs to have access to the Database Entities defined in the Core module.

In this section, we will import the Entities from the Core module to the OSMDb module, to enable retrieving data, as well as adding / removing records from those Entities, from the responsive module. To do that we will use the Manage Dependencies dialog in the OSMDb module.

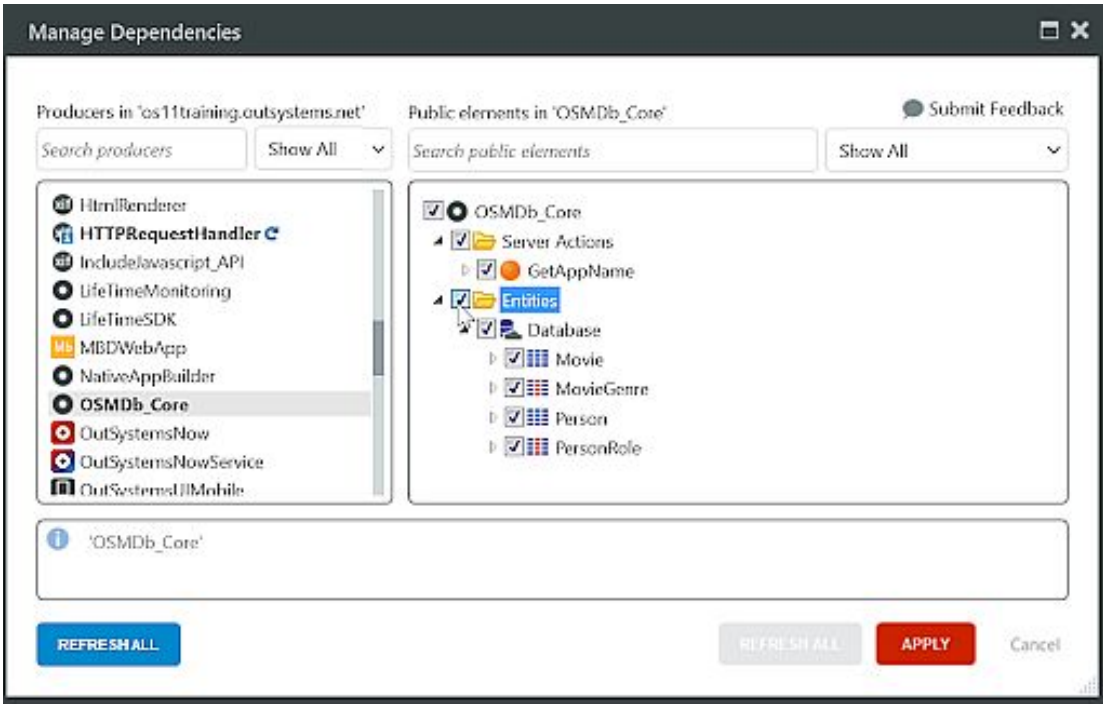
1) In the OSMDb module, open the **Manage Dependencies** dialog by pressing the icon in the toolbar.



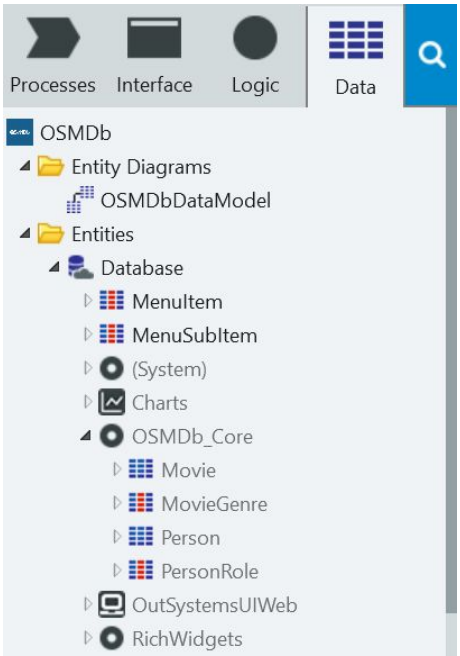
2) In the Manage Dependencies dialog, ensure the option **Show All** is selected in the list of Producers. Then, select the **OSMDb_Core** module from the list.



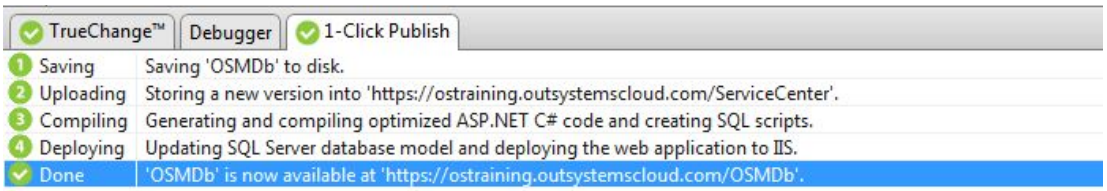
3) In the Public elements list for the core module, you should see the four entities created and made Public in the earlier exercise. Select them all and press **Apply** to close the dialog.



4) Click the **Data** tab in the upper right corner of the workspace to display the Data elements. Notice that the Entities referenced from the core module are now available in the current module.



5) Publish the module using the **1-Click Publish** button so the server commits these dependencies between modules.



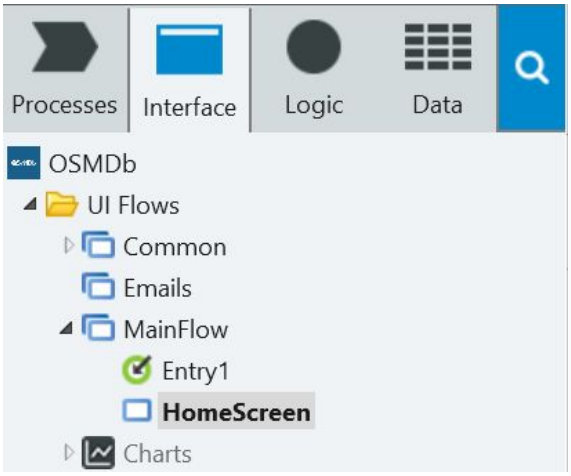
Create the Movies Screen

In this part of the exercise, we will create a Screen to display a list of movies. These movies will be retrieved from the database, in particular from the Movie Entity. This does not mean that the movies can only be fetched from Database Entities defined in Service Studio. It is also possible to fetch it from other sources, such as external databases or web services, if desired.

We will go through several steps to build the Movies Screen, starting by preparing the HomeScreen to actually be the new Movies Screen. Then, we will fetch the movies data from the database, using the Screen Preparation. Finally, to display the list of movies, we will use a **Table Records** Widget in the Screen, with the result of the query in the Preparation as its Source Record List.

1) Use the existing HomeScreen to make it the *Movies* Screen. Change the Screen name and delete the "Hello World" message.

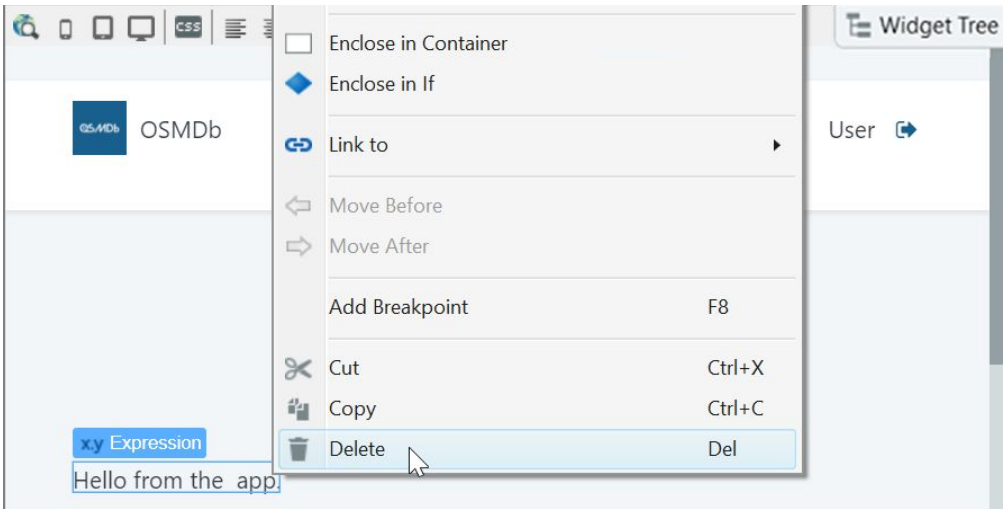
    a) Switch to the Interface tab, in the upper right corner of the workspace and expand the MainFlow.



    b) Click on the HomeScreen and change its **Name** to *Movies*.

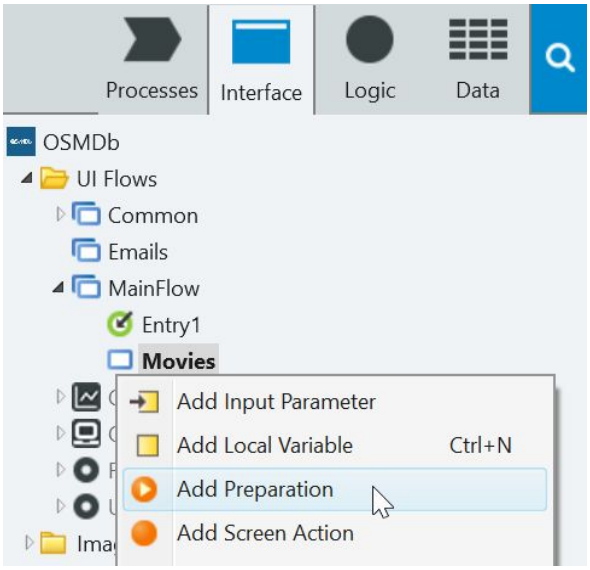
    c) Double-click the Movies Screen to open it in the canvas.

d) Right-click over the “Hello from the OSMDb app” expression in the canvas and then select the **Delete** option.



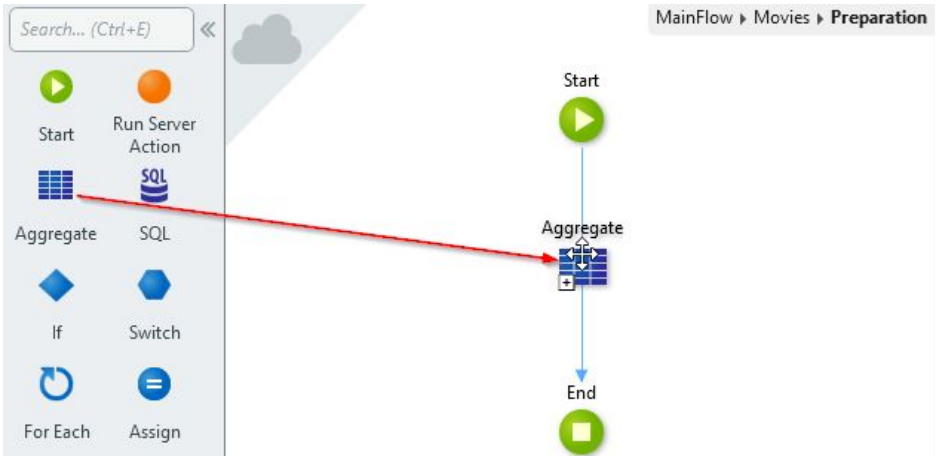
2) Create a **Preparation** in the **Movies** Screen, to fetch all the movies from the Movie Entity. A Preparation is a special Action that runs before the Screen is rendered, so when the Screen is built, it already has the data that it needs.

a) In the Elements area, right-click the Movies Screen and select **Add Preparation**



NOTE: Notice that the Preparation flow opens automatically in the editor.

b) Drag an **Aggregate** to the flow between the Start and the End. Drop the Aggregate when the connection is flashing blue to link it to the flow.

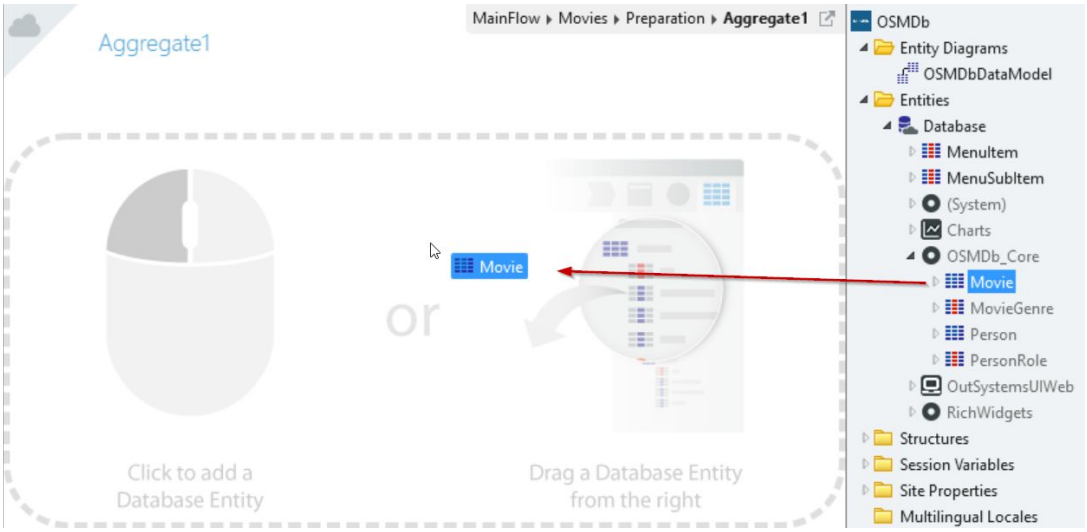


NOTE: Notice that the Aggregate gets connected in the flow and becomes Aggregate1. Also notice that the green 1-Click Publish button and the **green** checkmark on the TrueChange tab have turned into **red**. Hovering Aggregate1 causes a tooltip to appear that indicates that the Aggregate must have at least one Entity to be valid.

---

c) Double-click **Aggregate1** to open it in the Aggregate editor.

d) Switch to the **Data** tab and drag and drop the **Movie** Entity to the canvas. This sets up the Aggregate to get all the movies from the database (Movie Entity).

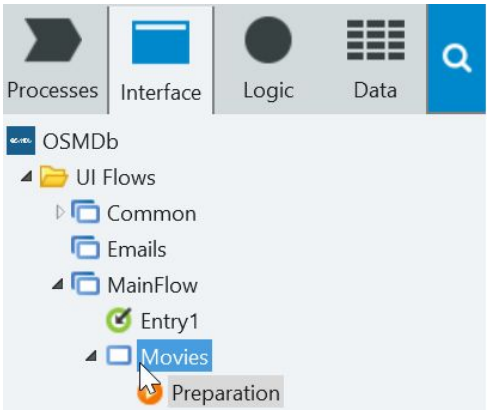


NOTE: After dragging and dropping the Movie Entity to the Aggregate, the Aggregate automatically gets the name GetMovies.

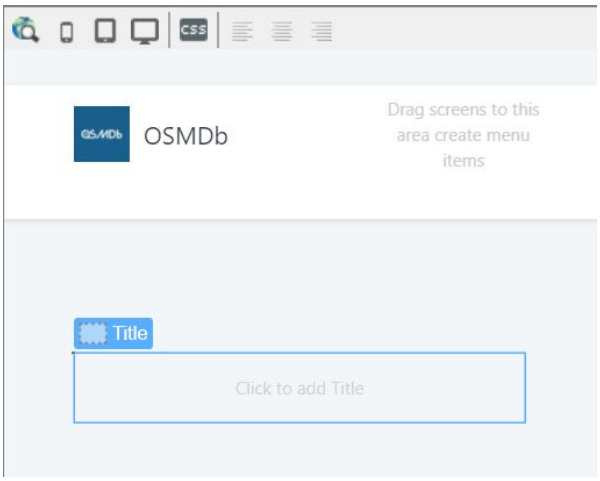
Movie Title	Movie Year	Movie PlotSummary
Star Wars: The Force Awakens	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Orde
Raiders of the Lost Ark	1981	Archaeologist and adventurer Indiana Jones is hired by the US government to find the A
Schindler's List	1993	In Poland during World War II, Oskar Schindler gradually becomes concerned for his Jew
Avatar 2	2020	(unknown)
Indiana Jones and the Last Crusade	1989	When Dr. Henry Jones Sr. suddenly goes missing while pursuing the Holy Grail, eminent
Capote	2005	In 1959, Truman Capote learns of the murder of a Kansas family and decides to write a b
Along Came Polly	2004	A buttoned up newlywed finds his too organized life falling into chaos when he falls in l
Avengers: Infinity War	2018	The Avengers and their allies must be willing to sacrifice all in an attempt to defeat the
Dunkirk	2018	Allied soldiers from Belgium, the British Empire and France are surrounded by the Germ
Star Wars: The Last Jedi	2017	Rey develops her newly discovered abilities with the guidance of Luke Skywalker, who is

3) Now that we have the data, let's create the content of the Movies Screen to display the list of all movies in the database. First, define the **Title** of the Screen (in the UI) as *Movies*. Then, add a **TableRecords** to the Screen, bind it to the **GetMovies** Aggregate, and display the columns for all attributes, except for the Id and IsAvailableOnDVD.

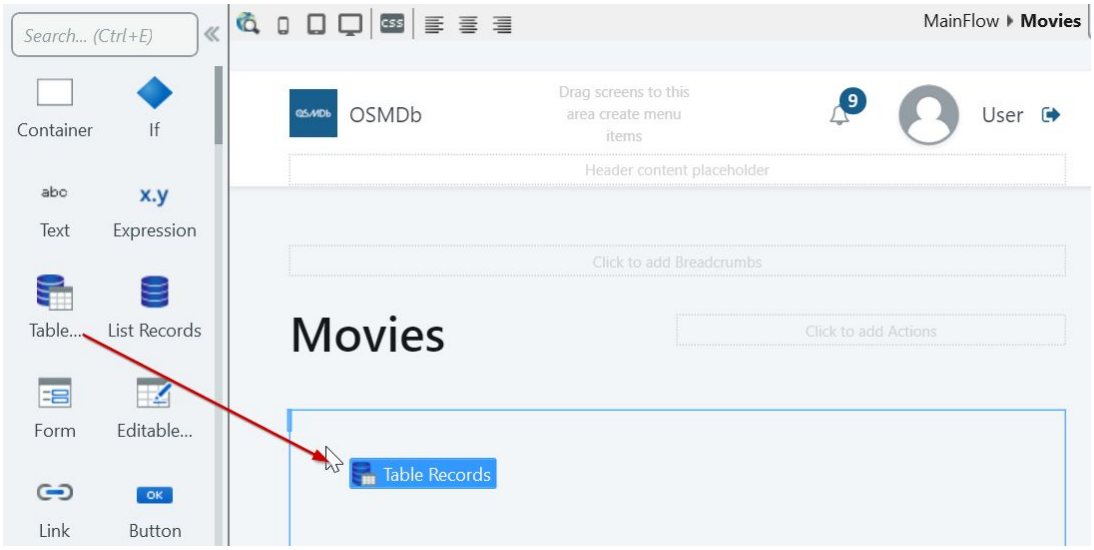
   a) Switch to the Interface tab and double-click the Movies Screen to open it.



b) Type *Movies* in the **Title** area of the Screen.



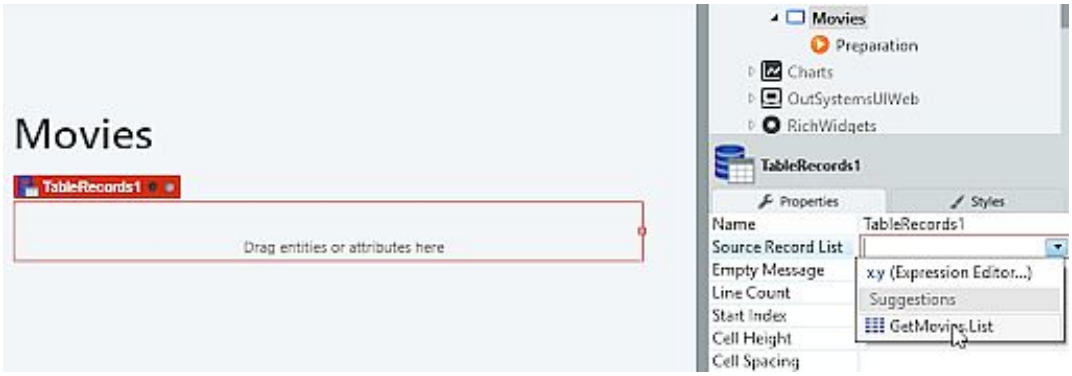
c) Drag and drop a **Table Records** Widget from the Toolbox to the Main Content area of the Screen.



NOTE: Notice that the Table Records becomes TableRecords1 and is highlighted red. The **TrueChange** tab indicates an error because all Table Records require the **Source Record List** property to contain a valid value.

---

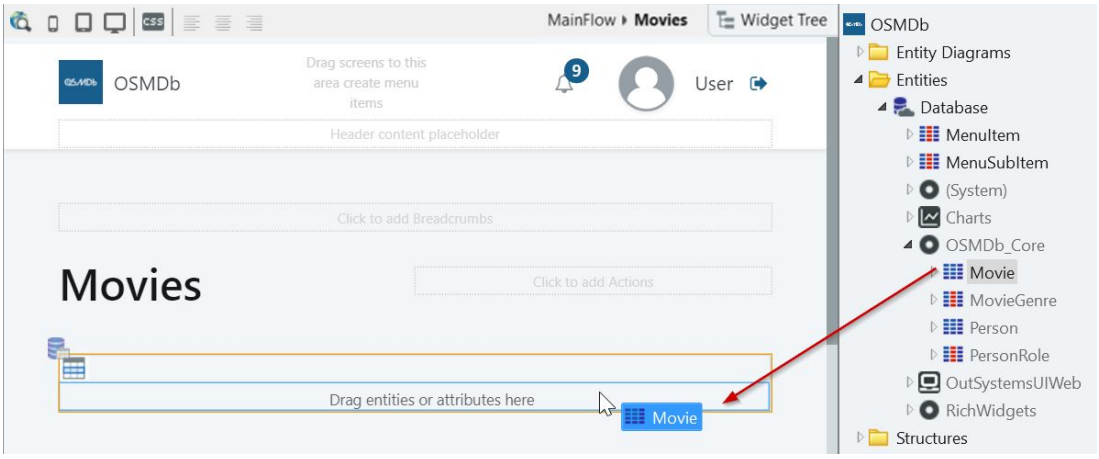
d) In the Properties editor, click the **Source Record List** drop down and select *GetMovies.List* from the suggestions. This defines that the result of the query GetMovies will be the source of the data to be displayed in the TableRecords.



NOTE: Suggestions are a helpful part of the platform, that will highlight some of the variables in the scope, with an appropriate data type for the property being filled in. Often there are several suggestions to choose from. **Always check the suggestion** and consider whether or not it is the correct value for the property. Sometimes the right value may not be on that shortlist and you may need to use the **(Expression Editor...)** to define the value that you want.

---

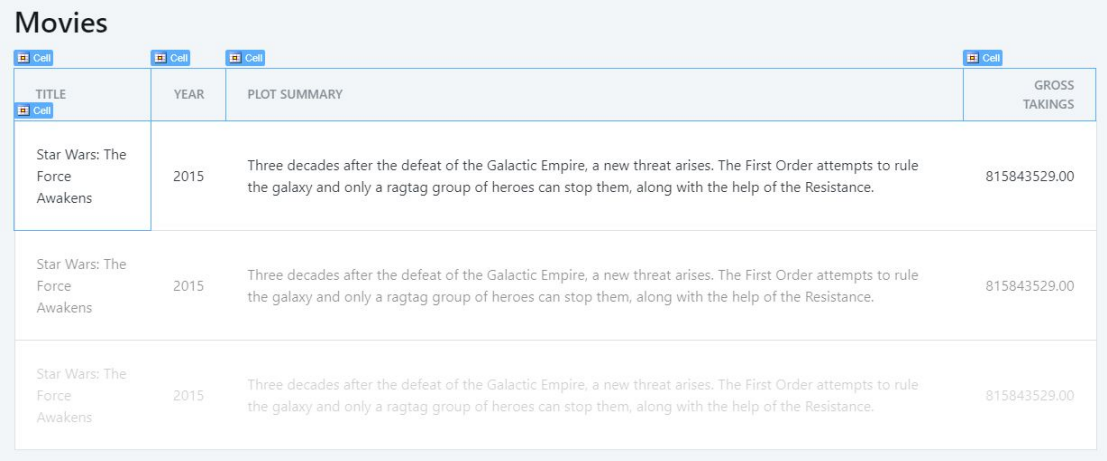
e) Select the Data tab, then drag and drop the Movie Entity onto the Table Records, where it says **Drag entities or attributes here**. This is an accelerator to create the layout for displaying the movie data in the Table Records.



NOTE: Defining a Table Records is a 2-step process. First, the source data for the Widget must be defined, in the **Source Record List** Property. Then, the layout for how the data should be displayed must be defined in the Screen. That is what we did in the previous steps.

---

f) Notice that the Table Records now displays movie data.



NOTE: A TableRecords will by default display the first four attributes of an Entity that is used to configure it, except for the ID. Columns for attributes can be added or removed as necessary. You may not view all the columns if your Screen is not wide enough. If that is the case, try to maximize it.

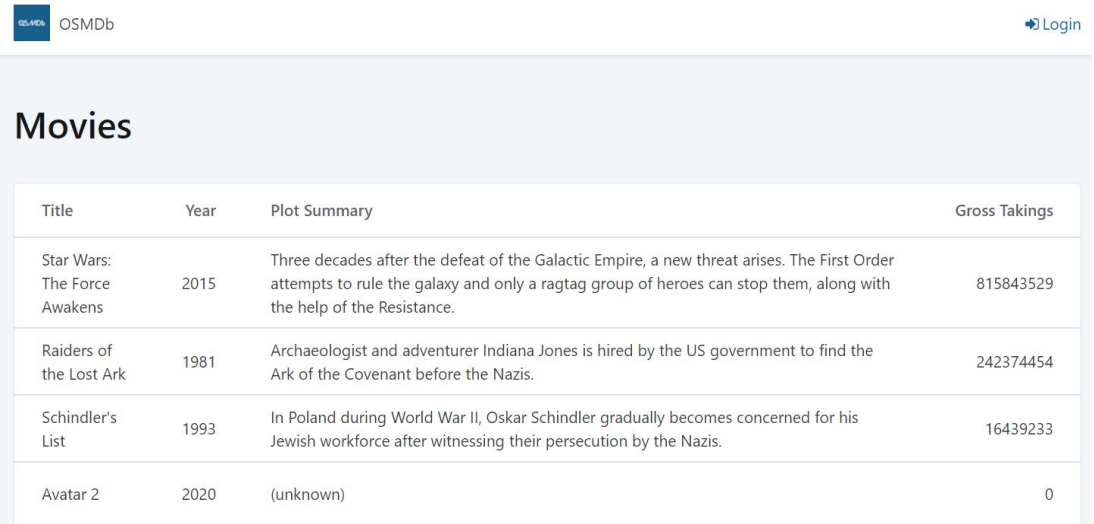
---

4) Publish the new updates to the server and make sure that you can access the Movies Screen and that you see all the movies.

a) Click the **1-Click Publish button**  to publish the module to the server.

b) Verify that the application was published successfully in the **1-Click Publish tab** and click on the **Open in Browser** button  .

c) You should see something like this in the browser.

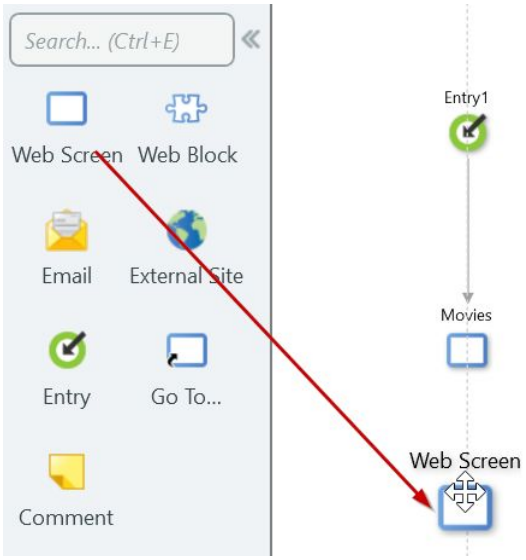


Create the MovieDetail Screen

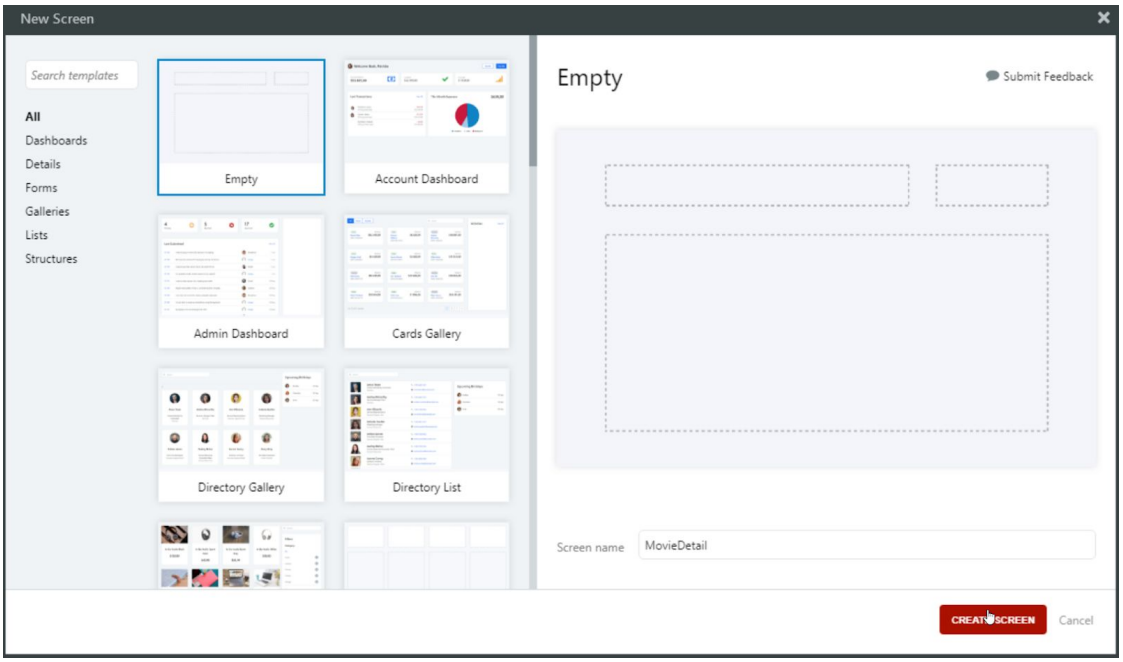
In this section of the lab, we will create a detail Screen, to display the details of a particular movie, using a Form. For that purpose, we need to fetch the movie from the database, in the Preparation of the Screen, using an Aggregate. This Aggregate will be **filtered by the Id of the movie** that we want to see on the Screen. That Id will be an Input Parameter of the Screen.

Add a Link between the MovieDetail Screen and the Movies Screen to go back to the list, when needed. In the Movies Screen, add a Link between every movie to the respective MovieDetail Screen, to display its detailed data.

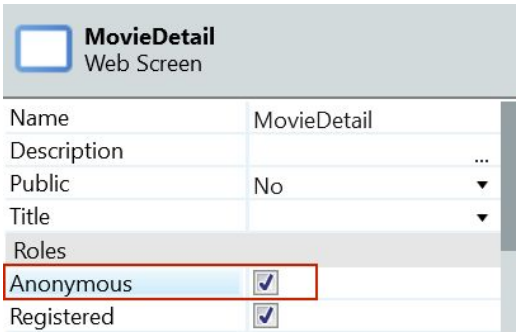
1) Create the MovieDetail Screen using the **Empty** screen template and make it **Anonymous**.
    a) Switch to the Interface tab, and double-click on the **MainFlow** to open it.
    b) Drag and drop a new **Web Screen** below the Movies Screen.



c) In the new window, select the **Empty** Template and name the Screen as *MovieDetail.* Click on the **Create Screen** button.

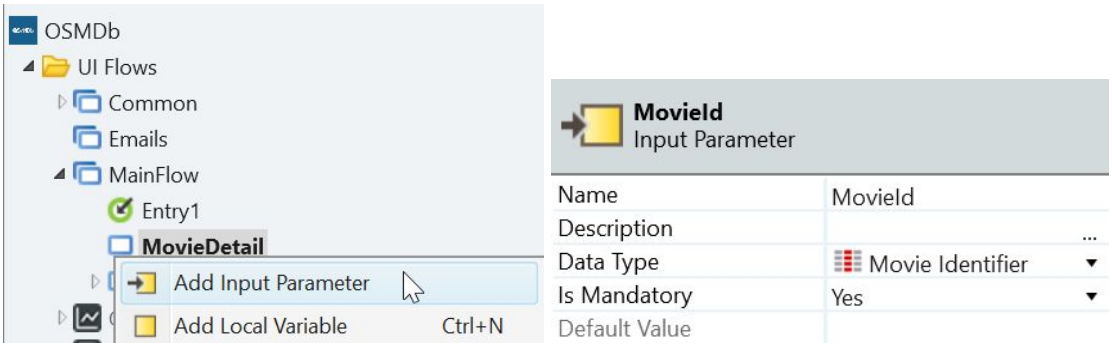


d) Enable the **Anonymous** Role access to the MovieDetail Screen.



2) Since we want to display the details of a movie in the MovieDetail Screen, we need to fetch the specific movie that we want to display. For that, create a **Preparation**, with an Aggregate that gets a movie, filtered by its **Id**. The Movie Id will be an **Input Parameter** of the Screen.

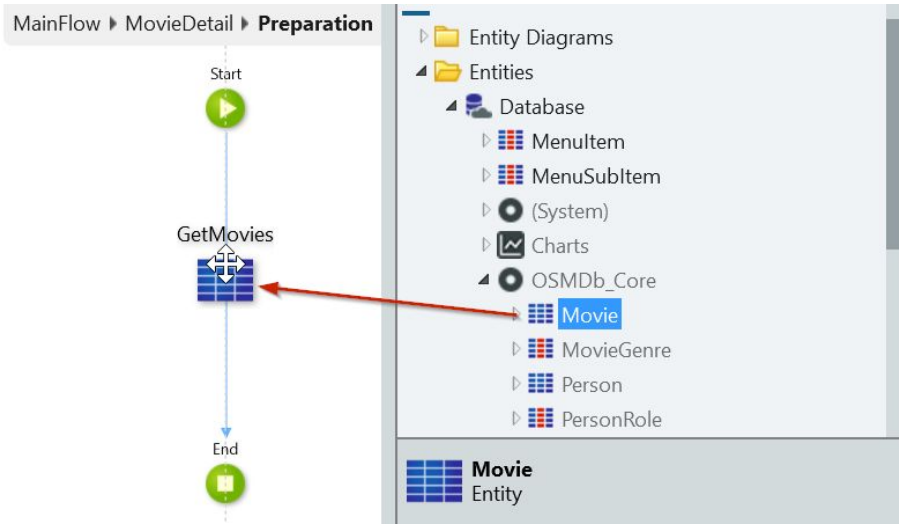
a) In the Elements area, right-click the **MovieDetail** Screen and select **Add Input Parameter** and name it *MovieId*. Verify that the **Data Type** of the Input Parameter changes to *Movie Identifier*, and that **IsMandatory** is set to *Yes*.



NOTE: When a Screen has a mandatory Input Parameter, it is necessary to pass a value for the Input Parameter, in **every request for the Screen to the server**.

b) Right-click the MovieDetail Screen and select **Add Preparation**.

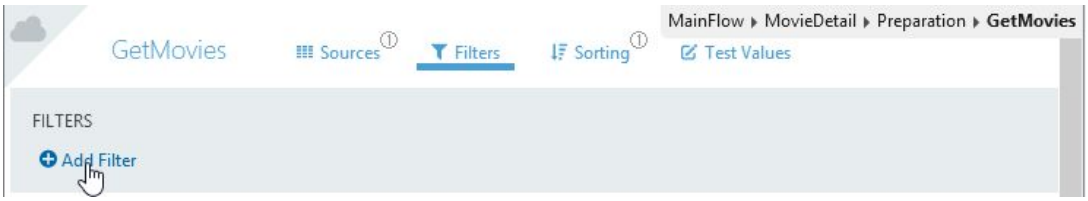
c) Switch to the **Data** tab and drag and drop the **Movie** Entity onto the Preparation flow. This creates an Aggregate to get all the movies from the database. This is an accelerator to create an Aggregate similar to the one in the Movies Screen.



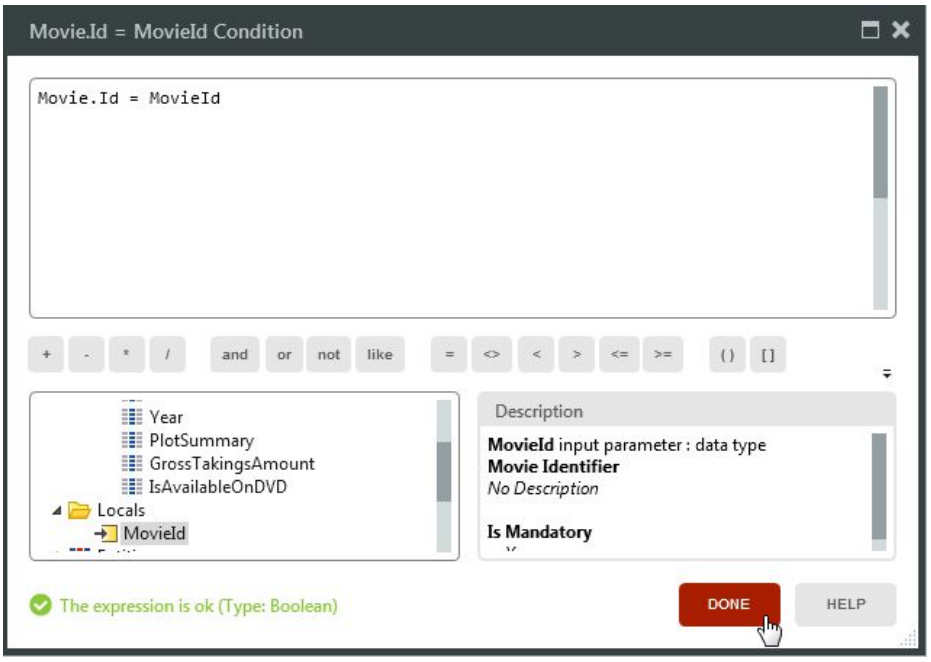
d) Since we just want the movie that matches the Id from the Input Parameter, we need to change the Aggregate. Double-click the **GetMovies** to open the editor.

e) Select the **Filters option** to open the Filters section of the editor.

f) Click **+Add Filter** to add a filter to the Aggregate. This opens the Expression editor to add the filter condition.



g) In the new dialog, set the expression to: *Movie.Id = MovieId*
Close the editor by clicking on the Done button. This will select the movie from the database, whose Id (*Movie.Id*) is equal to the *MovieId* Input Parameter.



NOTE: You can type the expression directly or you can do it by selecting the elements on the Expression Editor's lower left dialog. To do so, expand the Movie and double-click on the Id attribute. Then, click on the = button, expand Locals (variables) and double-click on MovieId.

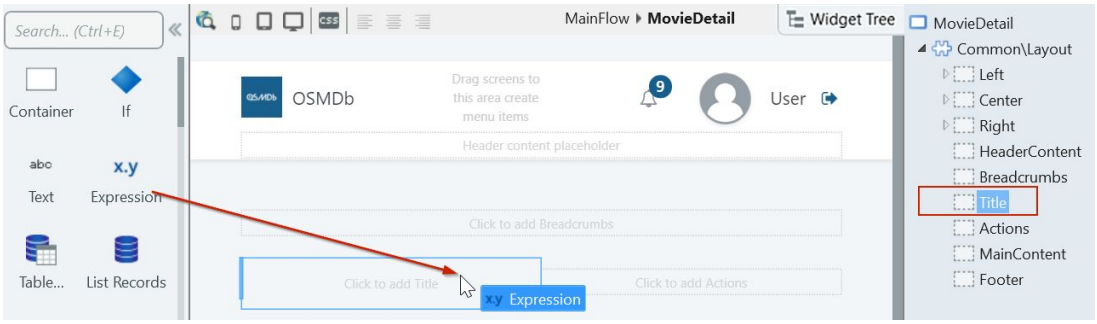
---

h) Notice that the Aggregate's name has changed from **GetMovies** to *GetMovieById*. The sample data also has only one row. Can you understand why?

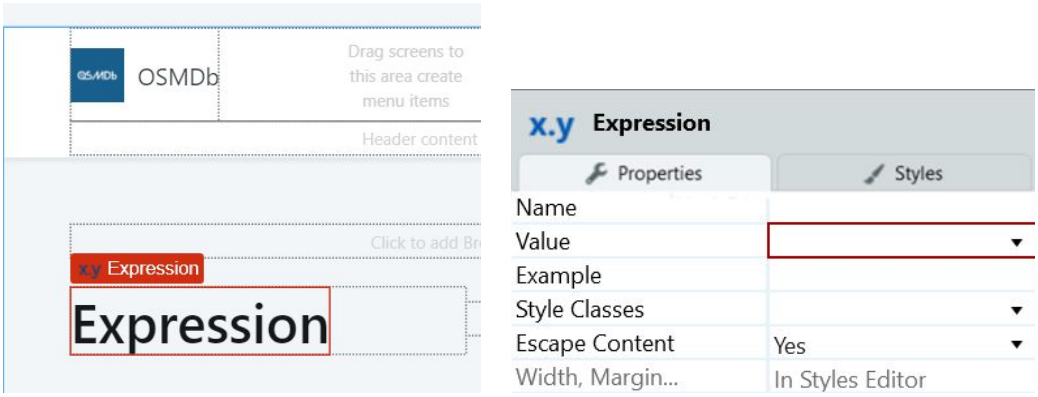
3) Set the **Title** of the MovieDetail Screen to be the name of the movie fetched by the **GetMovieById**. This way, it will be clear which movie we are looking into, when seeing the MovieDetail Screen

a) Switch to the Interface tab and double-click the **MovieDetail** Screen to open it.

b) Drag an **Expression** to the **Title** area and hold it there. Notice that the Elements area changes to display the **Widget Tree** and the Title area is flashing.



c) Expressions must have a valid **Value** set. Double-click the Expression or the Value property to open the editor.



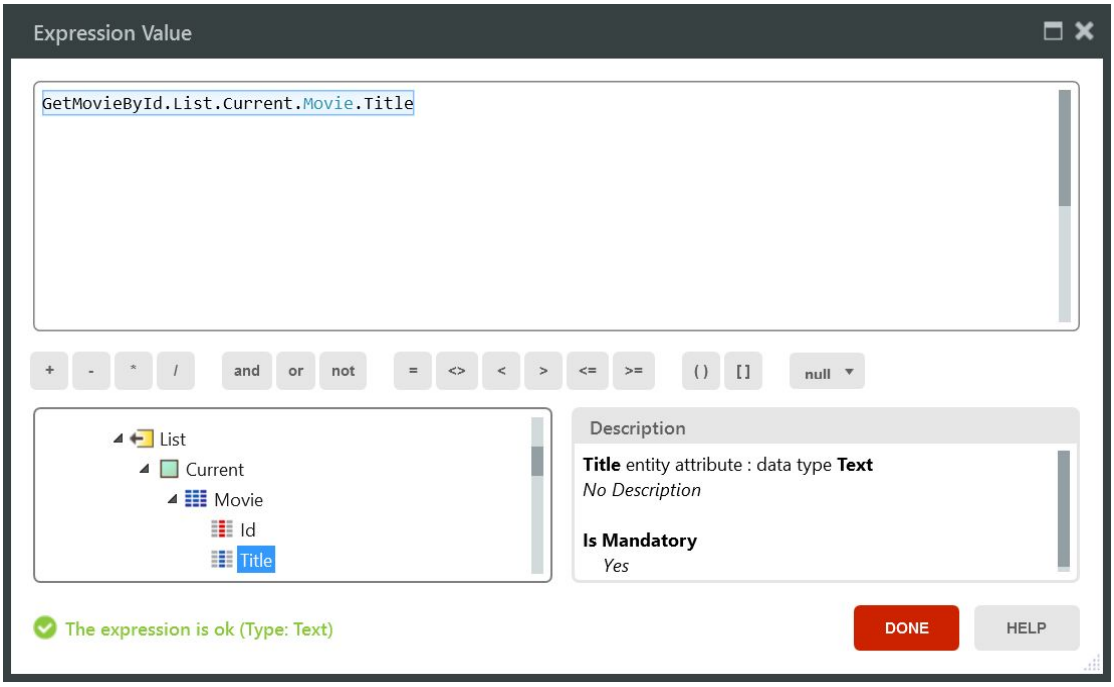
d) Double-click on the Value property of the expression to open the Expression Value dialog.

e) In the Expression Value dialog, enter the following expression:

GetMovieById.List.Current.Movie.Title

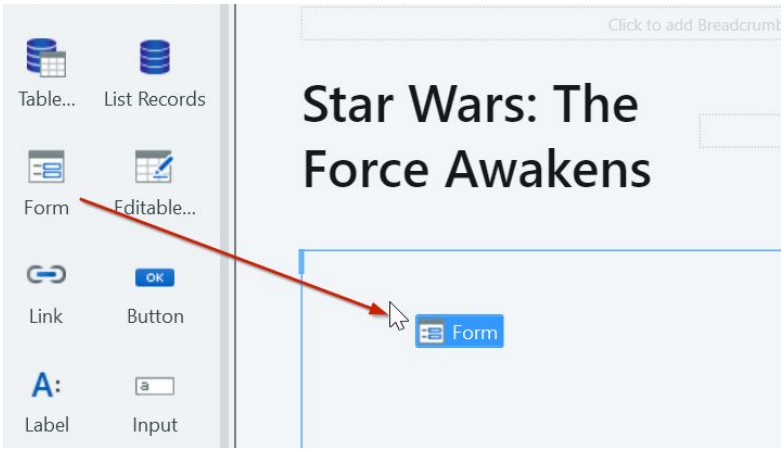
This expression uses the title of the movie returned by the Aggregate, defined in the Preparation. Since it is only one, we can use the **Current** property to access it, which by default points to the first element on the output. Click on **Done** to exit.

NOTE: To add the previous expression, you can expand the Preparation folder, on the lower left dialog, and select the *GetMovieById* > *List* > *Current* > *Movie*, from the Aggregate output. Then, double-click the Title attribute.

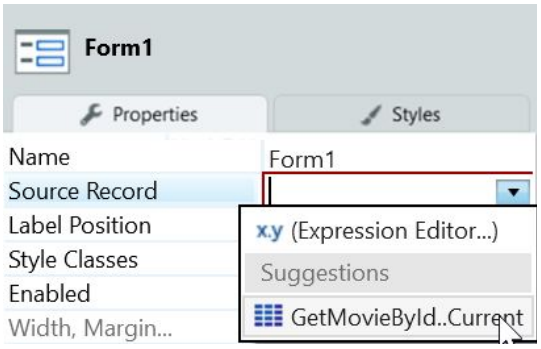


4) Create the Form to display the movie's details, using as source the Aggregate created in the previous step. All the Movie attributes should be displayed, except the Id. Make it *8* columns wide.

    a) Drag and drop a **Form** Widget to the MainContent area of the MovieDetail Screen.



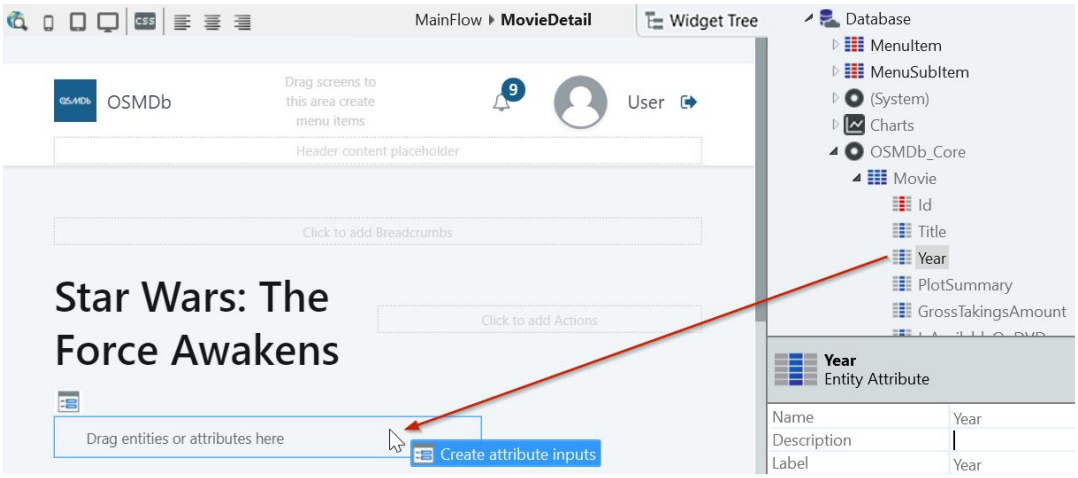
b) In the properties editor of the Form, click on the **Source Record** dropdown and select *GetMovieById.List.Current*. This represents the movie record fetched from the **GetMovieById** Aggregate, binding the Form to the output of the Aggregate.



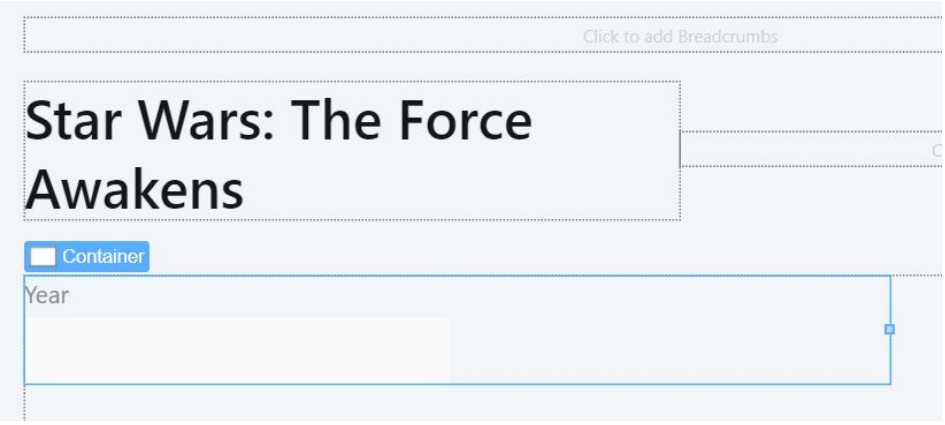
NOTE: Every Form must have a **Source Record** set, otherwise you get an error.

c) Drag the endpoint of the right side of the Form to **8 col** to make it wider.

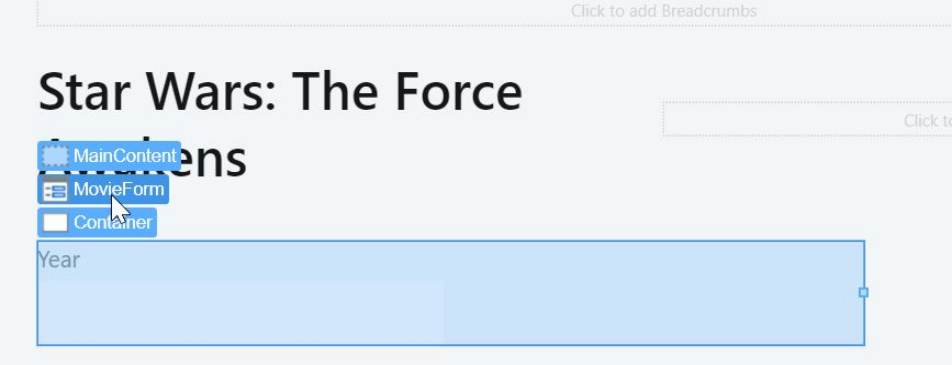
d) Open the Data tab, expand the Movie Entity and drag and drop the **Year** attribute to the Form.



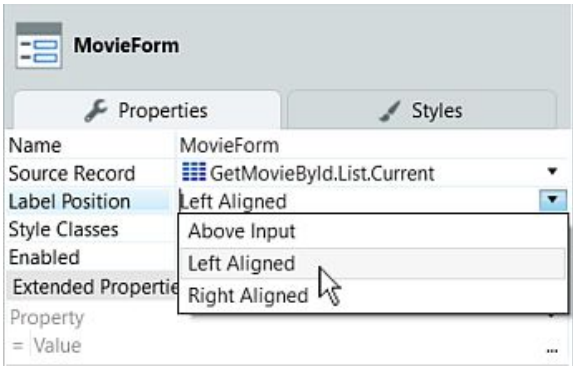
Notice that a **Container** was created. The Container has a **Label** and an **Input** (a text input field) Widgets, that corresponds to the movie *Year*. Besides, the text input field was placed below the Label widget. However, we would like to place the text input field next to the Label.



e) In the canvas, mouse over the Container element and then select the MovieForm.

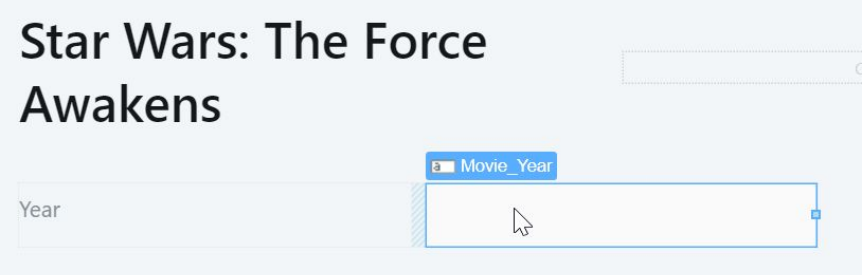


f) Change the value of the Label Position property of the MovieForm to *Left Aligned*.

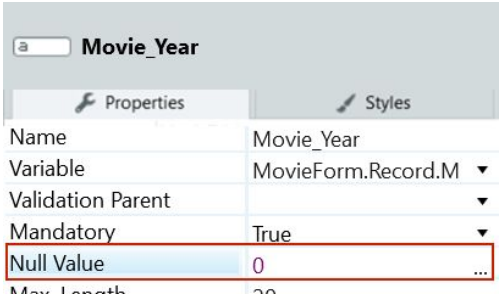


NOTE: At this point, the module has a warning stating that no inputs are submitted to the server. For now, this Screen will be used only for displaying movie data, so this warning can be ignored. We will address it later in the Labs.

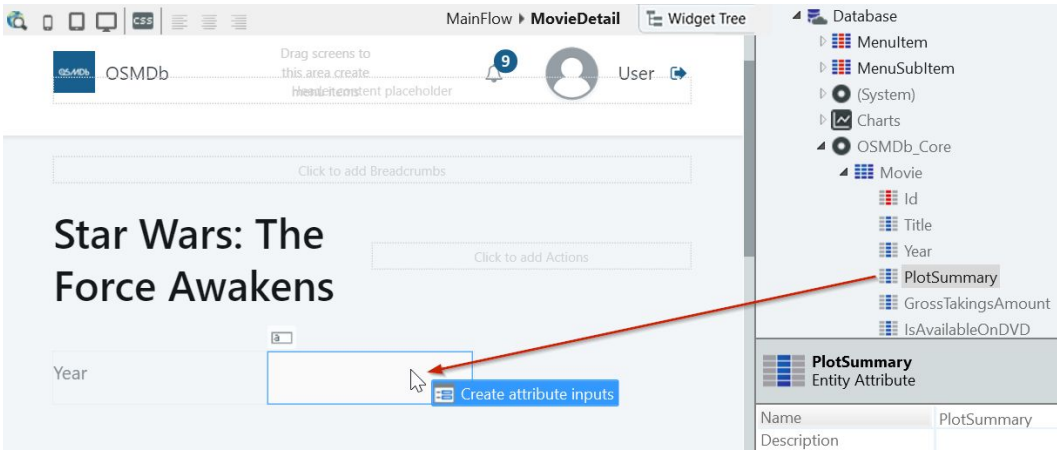
g) In the canvas, select the **Movie_Year** Input Widget.



h) Change its **Null Value** property to *0*. This means that if an end-user does not specify any year, the Form will hold *0* for the Year.



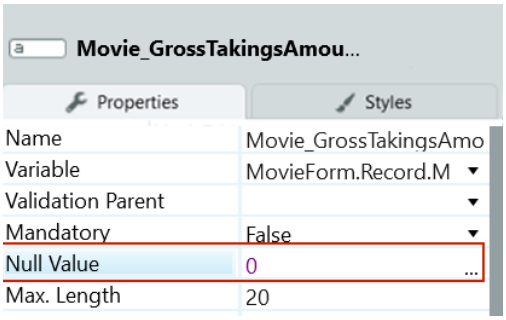
i) Switch to the Data tab and drag and drop the **PlotSummary** attribute of the Movie Entity into the **MovieForm**.



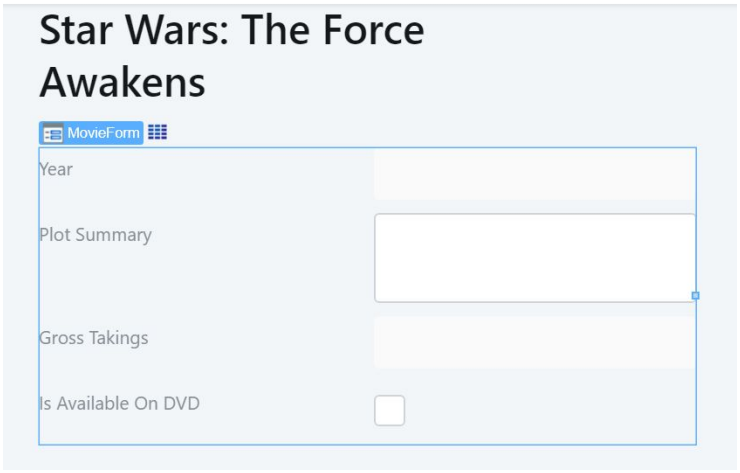
NOTE: Attributes dropped over an existing attribute are placed just below it.

j) Also drag and drop the **GrossTakingsAmount** and **IsAvailableonDVD** attributes to the Form.

k) Set the **Null Value** property of the Gross Takings Input widget to *0*.

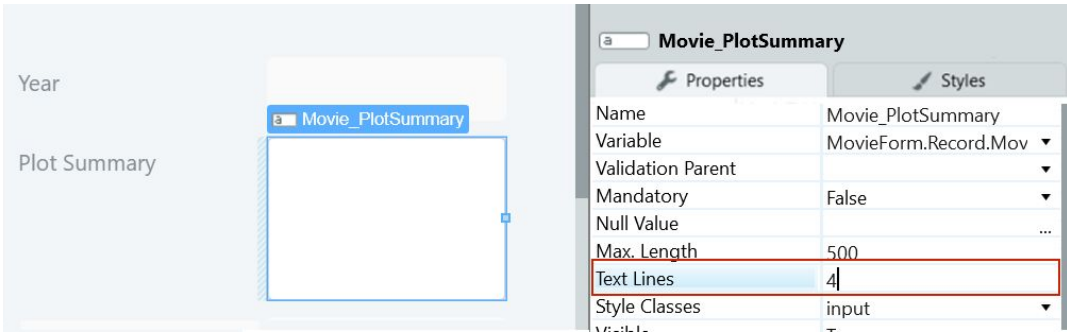


l) The MovieForm should look like this



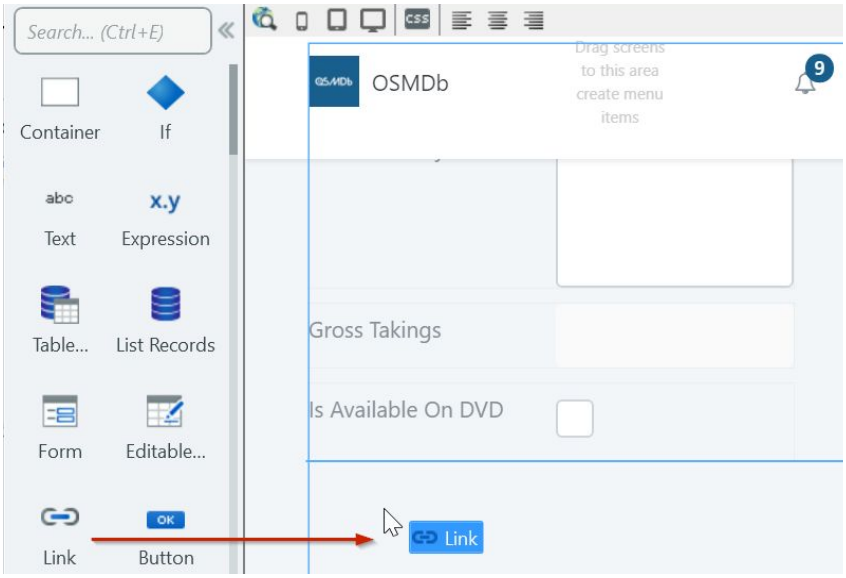
Notice that the Input for IsAvailableOnDVD is a **Check Box**. This is due to this attribute being defined as a *Boolean*. Also, notice that the Input for PlotSummary is, by default larger than the Year or GrossTakingsAmount Input. This is due to this attribute being defined with a **Maximum Length** of *500*.

m) Increase the size of the **Movie_PlotSummary** Input, by setting its **Text Lines** to *4*.

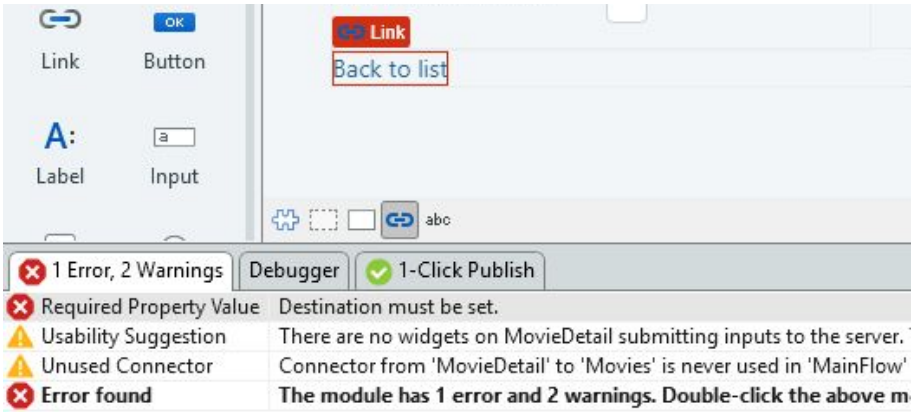


5) Create a **Link** to navigate back to the Movies Screen below the Form, to allow the user to navigate back to the **Movies** Screen.

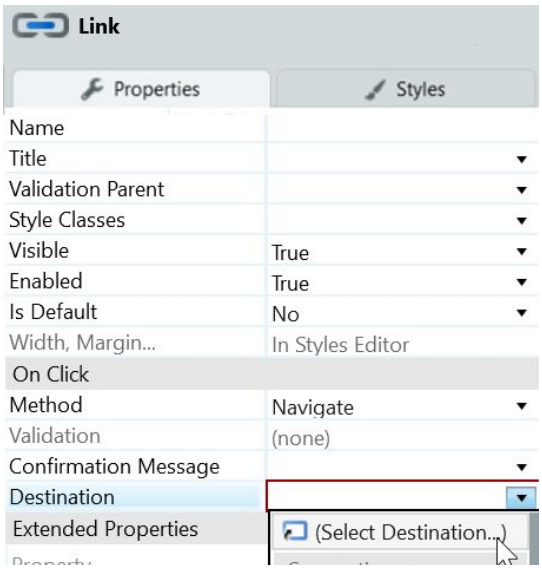
   a) Drag and drop a **Link** below the MovieForm.



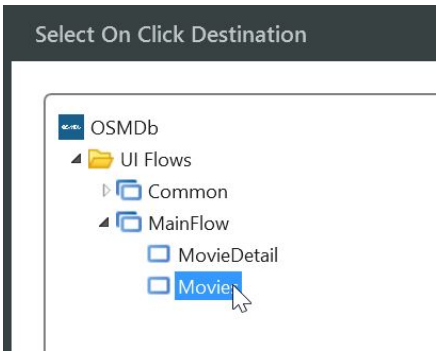
   b) Without moving the input focus, type in *Back to list* inside the Link. Notice that the **TrueChange** panel indicates there is an error. Links must have a **Destination** set for when users click on them. Double-click the 'Required Property Value' error. Focus will move to the location of the error.



c) Click on *(Select Destination...)* on the dropdown or simply double-click the **Destination** property.



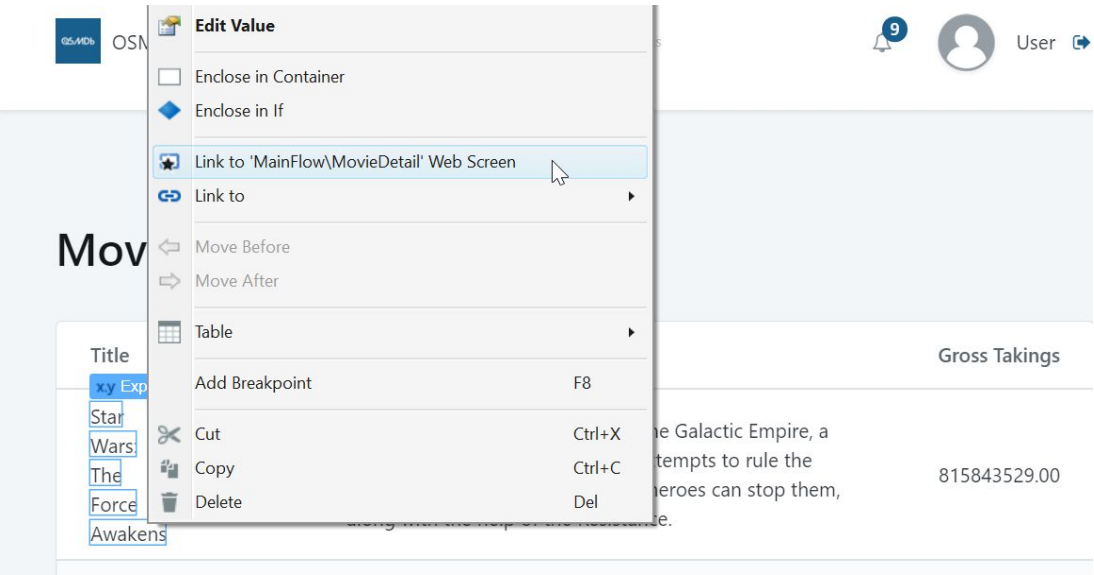
d) In the **(Select Destination)** dialog, expand the **MainFlow** and double-click the **Movies** Screen. This way, when a user clicks on the Link, it will navigate to the Movies Screen.



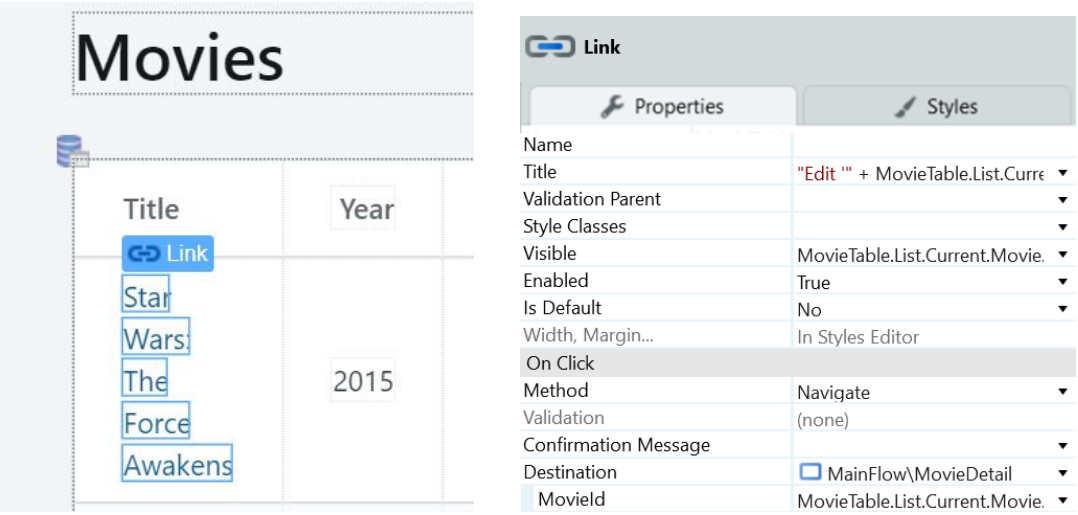
6) In the Movies Screen, add a Link to every movie in the Table Records, to open the respective MovieDetail Screen. Pass the corresponding value to the Movie Id attribute in the new Link.

a) Switch to the **Interface** tab and, in the Elements area, double-click the **Movies** Screen to open it.

b) In the **MovieTable**, right-click the Expression in the **Title** column and select the option Link to *MainFlow\MovieDetail Web Screen*.



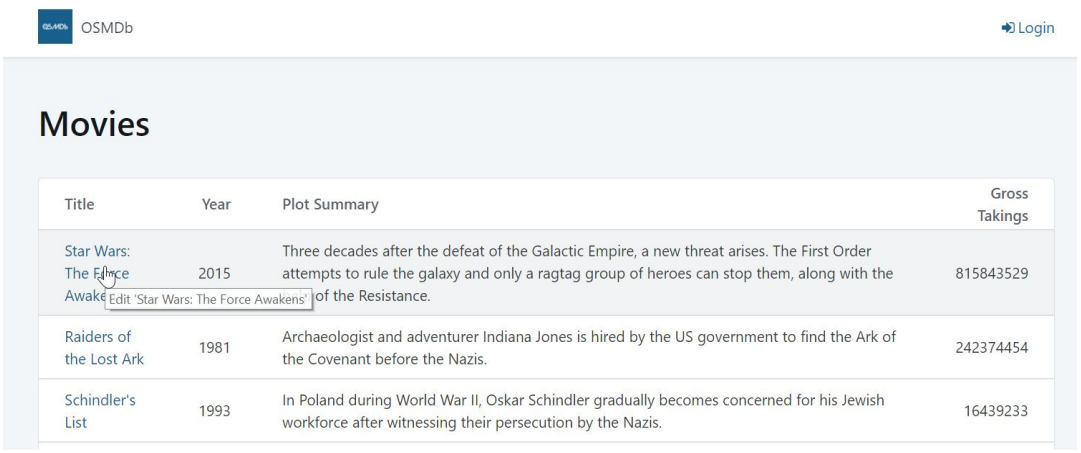
The icon associated with the option 'Link to MainFlow\MovieDetail Web Screen' had a star. This means that it is an accelerator. The Link is created automatically, and set to **Navigate** to the **MovieDetail** Screen, with the **MovieId** Input parameter.



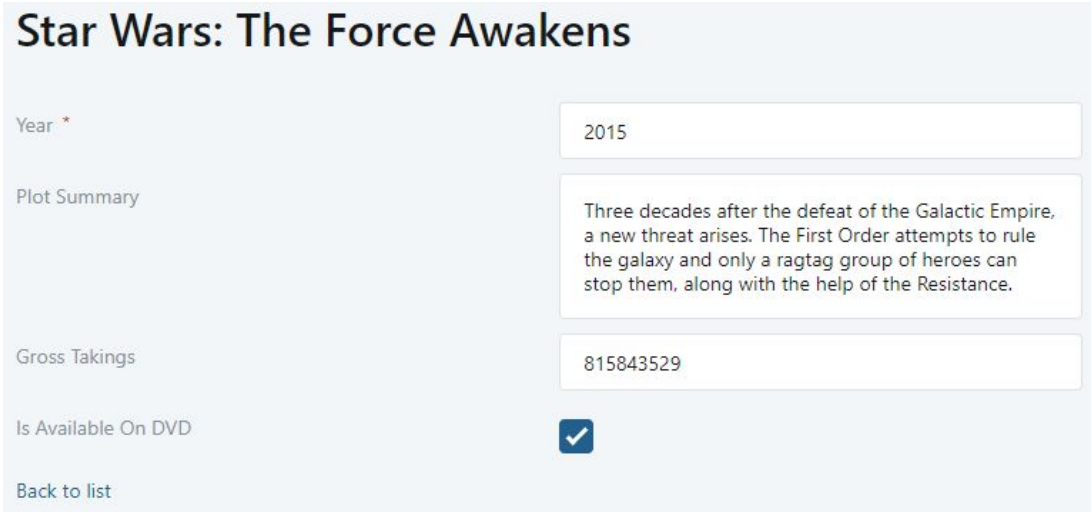
7) Publish the new updates to the server and let's test the updates made to the OSMDb application.

a) Click the **1-Click Publish button** (1) to publish to publish the module to the server and verify that the module was published successfully in the **1-Click Publish tab**.

b) Click on the **Open in Browser** button . The **Movies** Screen should be displayed.



c) Click on a movie title in the Table, like 'Star Wars: The Force Awakens'. This will open the **MovieDetail** Screen for the movie selected. You will be able to see the Form you created, with the information of the movie.

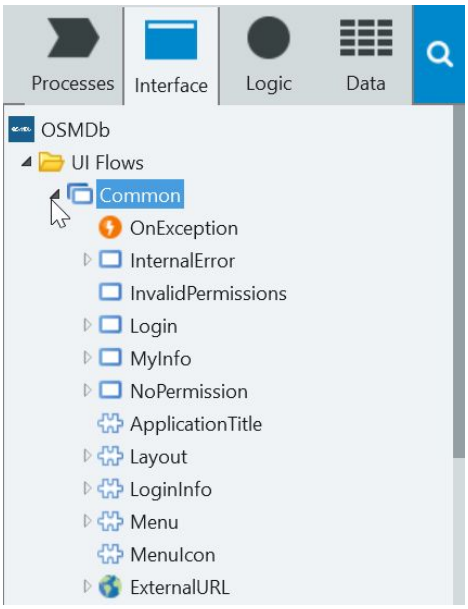


d) Click the **Back to list** Link to return to the **Movies** Screen.

Add an Entry to the Application Menu

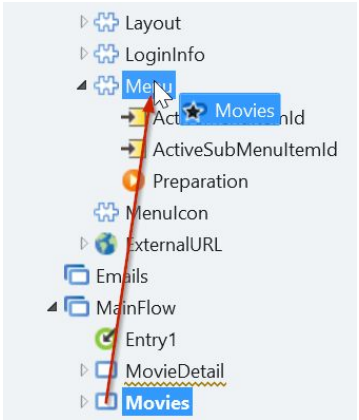
Now that we have the **Movies** Screen built, we will add it to the **Menu** of the application. The top Menu will allow users to navigate to important areas of the application, from wherever they may be.

1) Switch to the **Interface** tab and, in the Elements area, click on the triangle to the left of the **Common** Screen flow to expand it.



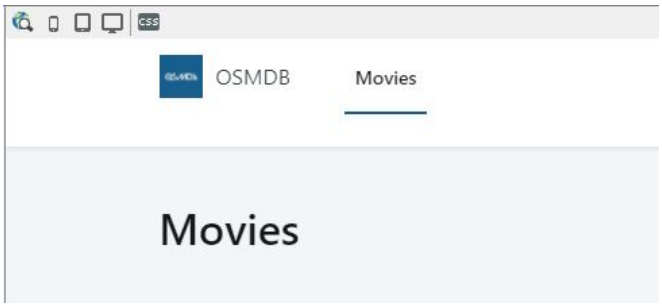
Notice a few elements inside this flow shaped as puzzle pieces. These are called **Web Blocks** and will be covered in more detail later. There is one particular Block element called **Menu**. This Block is automatically added at the top of every Screen in this module.

2) Drag the **Movies** Screen over the **Menu** Block and hold it there.



Notice that the dragged Screen changes to a **Link**, with a star in it, indicating an accelerator.

3) Drop the **Movies** Screen into the **Menu** Block.
4) Double-click any Screen in the **MainFlow** to open it.
5) Notice the **Movies** tab that now exists on the header.
6) After publishing these changes, clicking this Menu item in the application will send users to the **Movies** Screen.

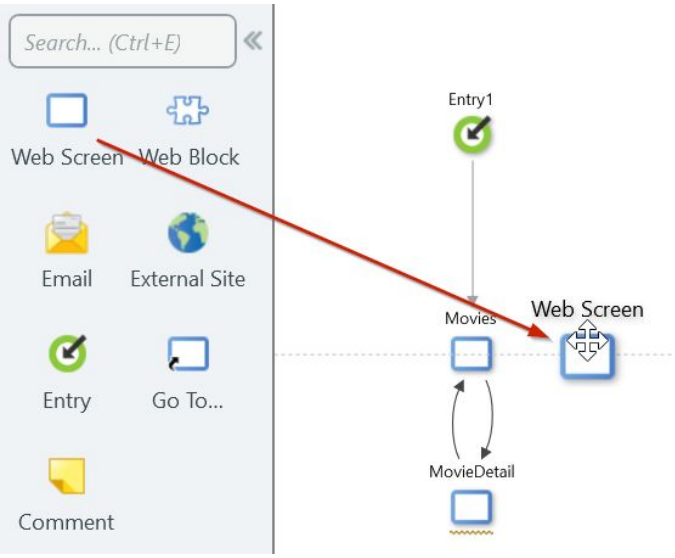


Create the People Screen

In this part of the exercise, we will create a Screen to display a list of people. This information will be retrieved from the database, in particular from the Person Entity.

1) Create the **People** Screen and add a **Preparation** action to fetch all people from the database. Set the Screen with **Anonymous** access.

    a) Switch to the **Interface tab** and double-click the **MainFlow** to open it. Drag and drop a new **Web Screen** next to the **Movies** Screen.

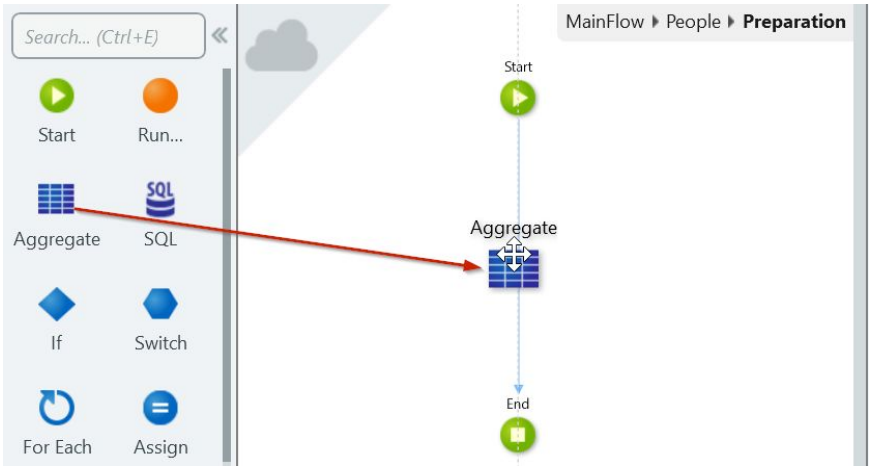


    b) In the new window, select the **Empty** Template and name the Screen as *People.* Click on the **Create Screen** button.

    c) In the properties of the People Screen, enable the **Anonymous** Role access to it.

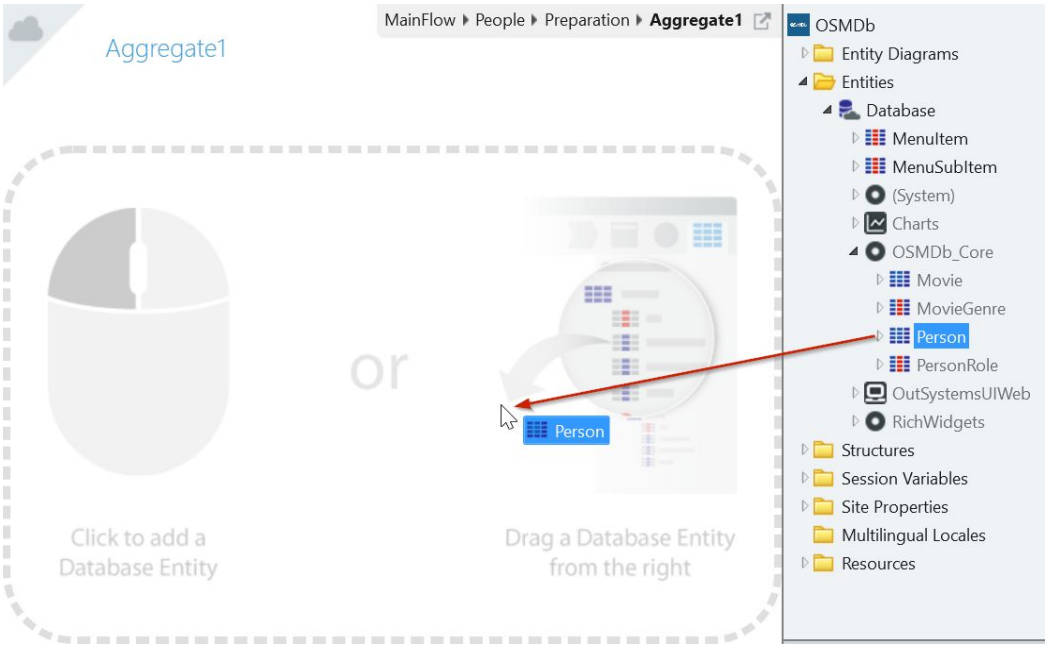
    d) Create a **Preparation** that will retrieve the people from the database. Right-click the **People** Screen and select **Add Preparation**.

e) Drag an **Aggregate** on to the connection between the Start and the End.



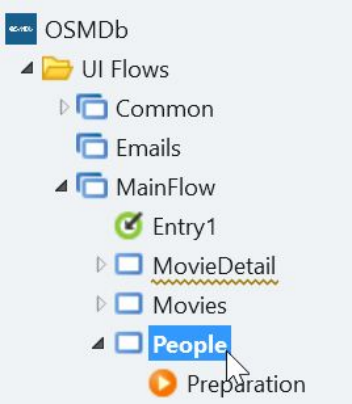
f) Double-click **Aggregate1** to open it in the Aggregate editor.

g) Select the **Data** tab and drag and drop the **Person** Entity to the canvas. You should see a few people on the preview area of the Aggregate.

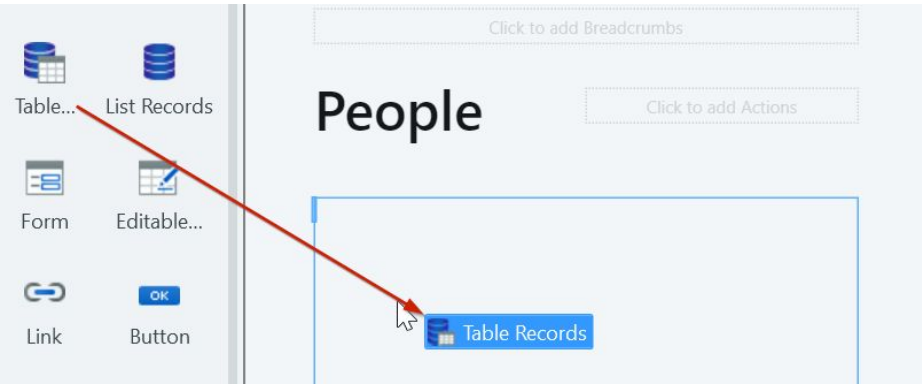


2) Define the content of the **People** Screen. Set the **Title** to 'People' and add a **Table Records** to display basic People data: Name, Surname and Date of Birth.

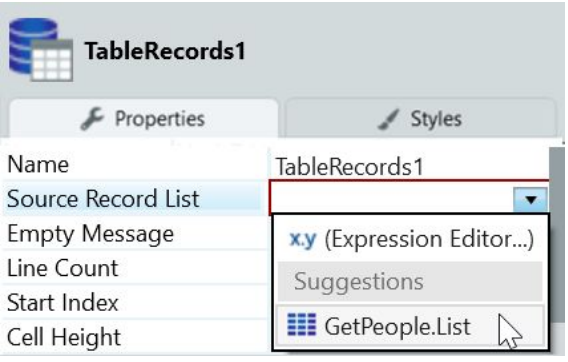
a) In the Interface tab, double-click the People Screen to open it in the canvas.



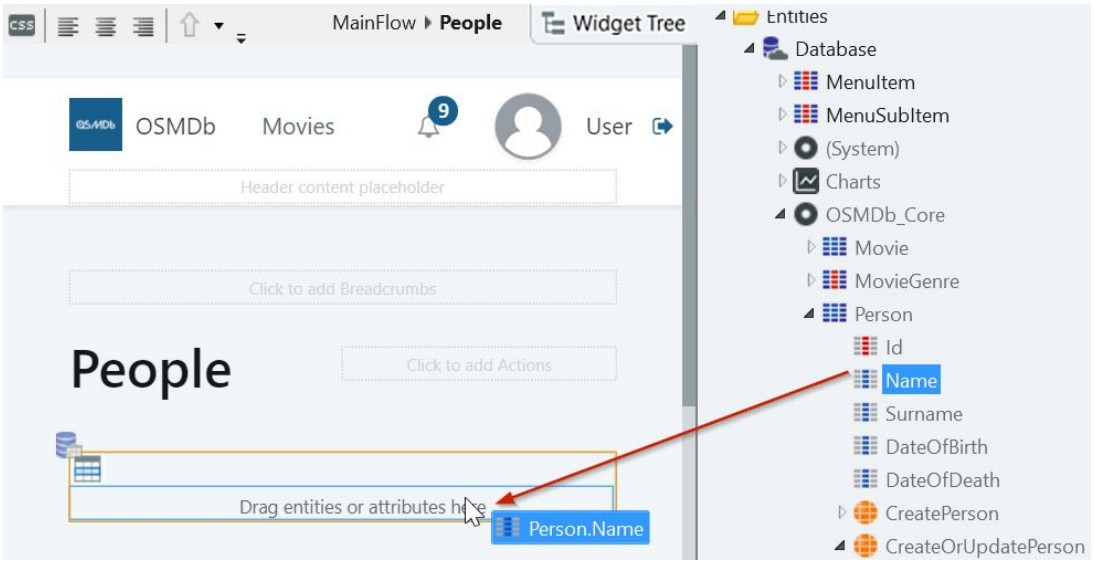
b) Enter *People* in the Title area of the Screen and drag and drop a **Table Records** Widget from the toolbox to the **MainContent** area.



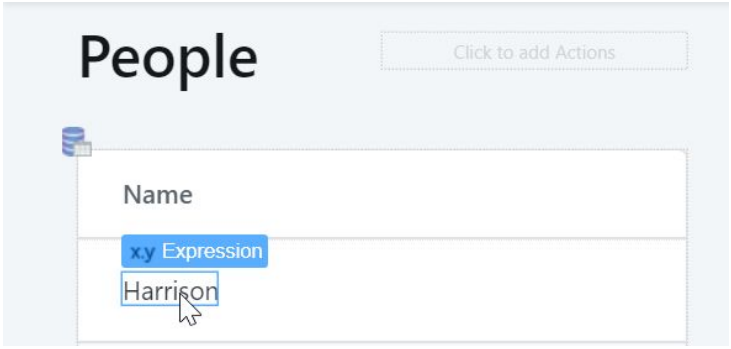
c) In the Properties editor, click the **Source Record List** drop down and select *GetPeople.List* from the Suggestions, to bind it with the Aggregate in the Preparation.



d) Select the **Data** tab, expand the **Person** Entity and then drag and drop the **Name** attribute on to the Table Records.



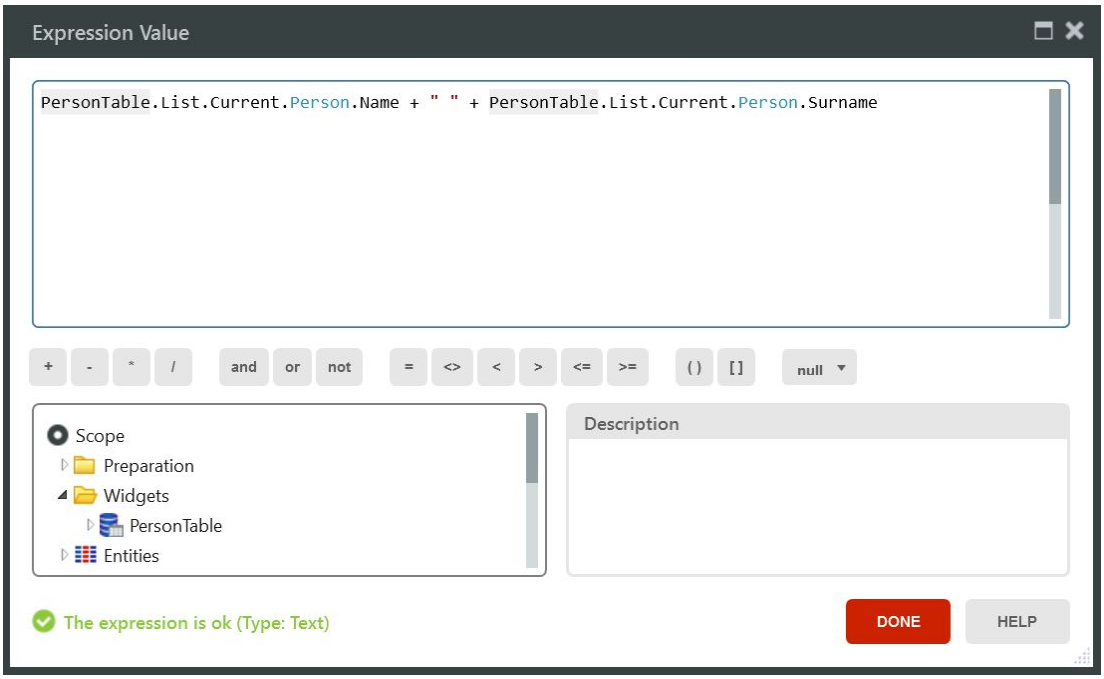
e) Double-click the **Expression** Widget in the **Name** column to open the Expression editor.



f) In the **Expression Value** dialog, notice the initial value for the expression simply displays the current person's Name.

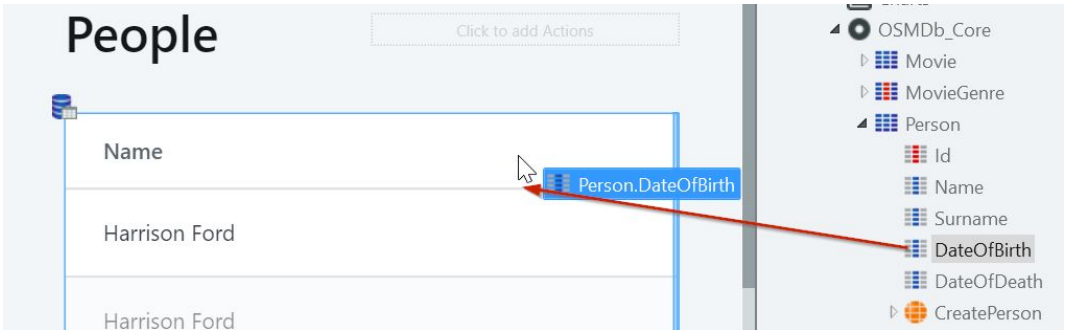
g) Improve the Expression to combine the current person's **Name** with his/her **Surname**, by changing the value to:

PersonTable.List.Current.Person.Name + " " + PersonTable.List.Current.Person.Surname



h) Click **Done** to dismiss the editor.

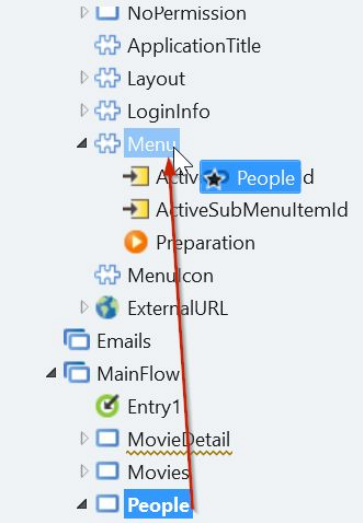
i) Drag and drop the **DateOfBirth** attribute onto the Table Records Widget.



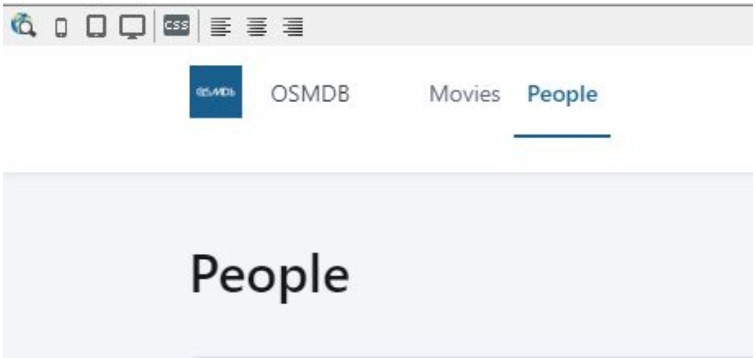
3) Create a Menu entry for the People Screen.

a) Switch to the Interface tab.

b) Drag the **People** Screen over the **Menu** Block and drop it there.



c) Notice that a **People** tab appears next to the **Movies,** in the **Menu** area of all the Screens in this module.

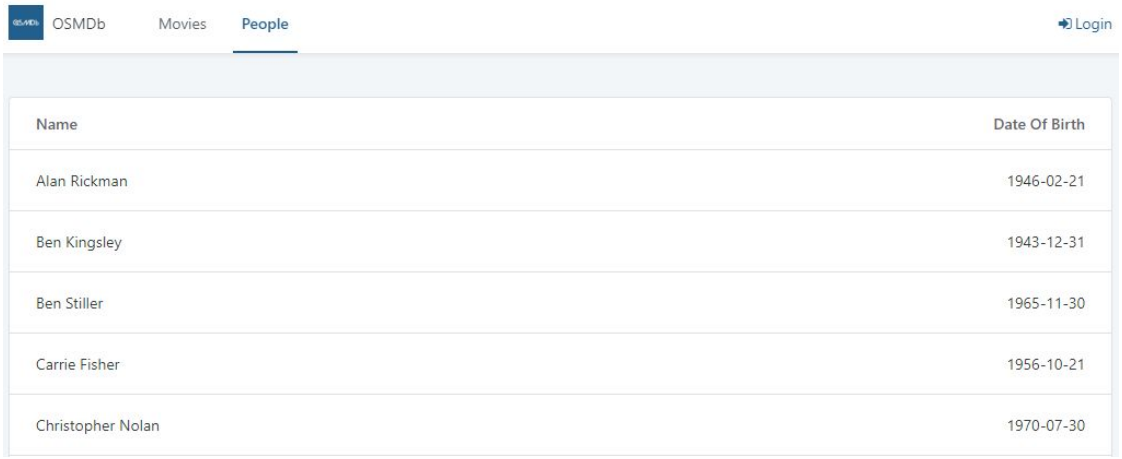


4) Publish the updates made to the server

a) Click the **1-Click Publish button** to publish to publish the module to the server and verify that the module was published successfully in the **1-Click Publish tab**.

b) Click on the **Open in Browser** button . The **Movies** Screen should be displayed. The **Movies** Screen should be displayed.

c) Use the top menu tab links to navigate between the Movies and People areas. Notice how the tabs colors change to highlight the selected Screen.

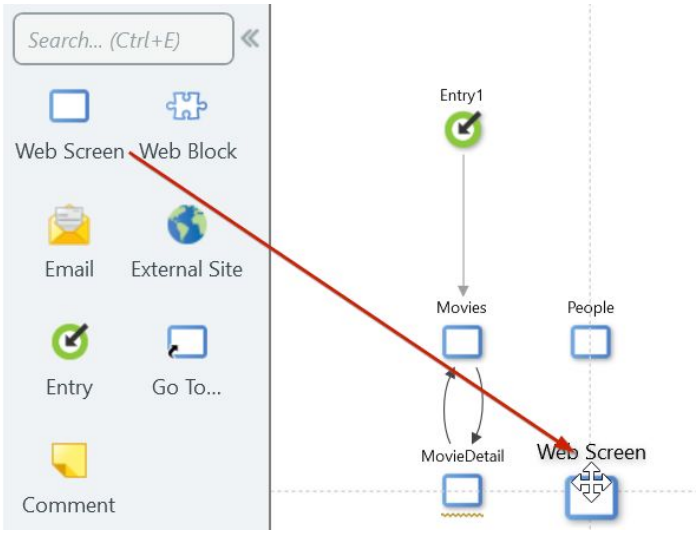


Create the PersonDetail Screen

Similarly to what we did with movies, we will create a detail Screen, to display the details of a particular person, using a Form. For that purpose, we need to fetch the person's information from the database with an Aggregate, in the new Screen Preparation. This Aggregate will be filtered by the Id of the person that we want to see on the Screen. That Id will be an Input Parameter of the Screen.

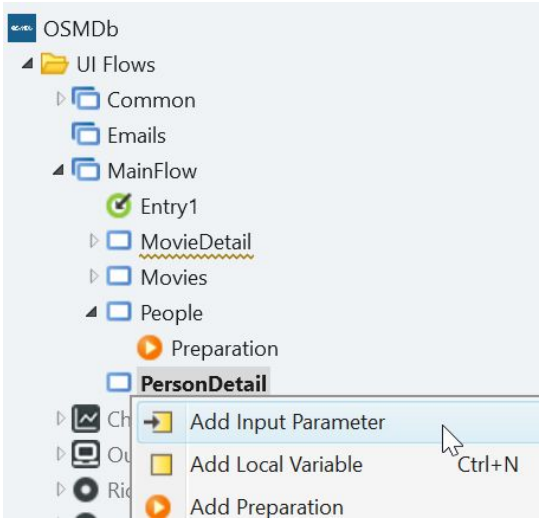
Add the *Back to list* Link from the PersonDetail to the People Screen. In the People Screen, we will also add a Link in every Person to the PersonDetail Screen.

1) Create the *PersonDetail* Screen using the **Empty** Screen template and make it **Anonymous**.
    a) Switch to the **Interface** tab and open the **MainFlow**.
    b) Drag and drop a new **Web Screen** below the **People** Screen.

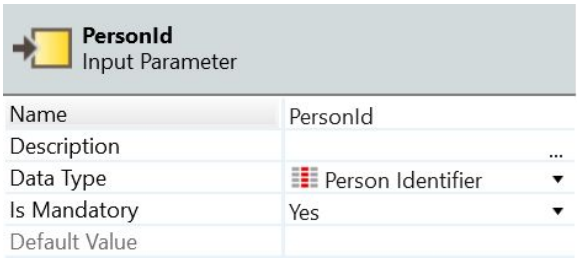


    c) Choose **Empty Screen**, enter *PersonDetail* for the **Name** of the Web Screen and enable **Anonymous** access to it.
2) Since we want to display the details of a person in the PersonDetail Screen, we need to fetch the specific person data. For that, create a **Preparation**, with an Aggregate filtered by its Id. The Person Id will be an **Input Parameter** of the Screen.

a) In the Interface tab, right-click on the **PersonDetail** Screen and select **Add Input Parameter**.

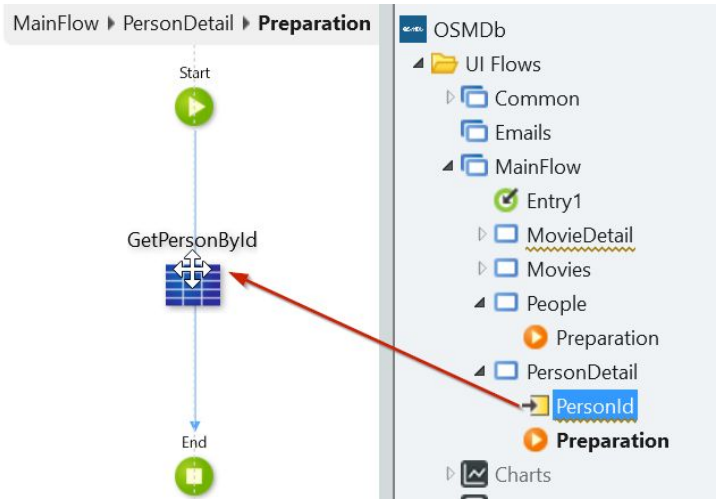


b) Enter *PersonId* for the name of the Input Parameter. Verify that the **Data Type** for the Input changes to **Person Identifier**.



c) Right click **PersonDetail** and select **Add Preparation**.

d) Drag and drop the **PersonId** parameter from the **PersonDetail** Screen in the flow of the Preparation. Notice that an Aggregate named **GetPersonById** was automatically created, with the filter by Person Identifier already created.

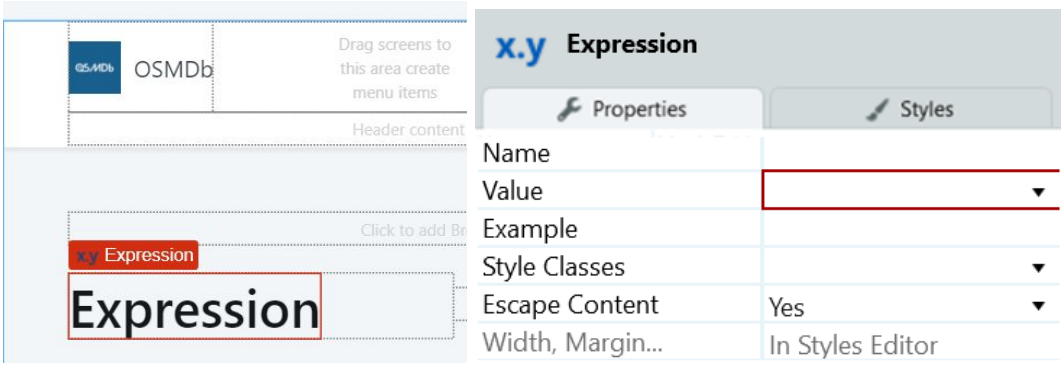


3) Set the **Title** of the PersonDetail Screen to be the name of the person fetched by the GetPersonById Aggregate. This way, it will be clear which person we are looking into, when seeing the PersonDetail Screen.

a) Open the **PersonDetail** Screen.

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

c) Double-click the Expression OR the **Value** label to open the editor.



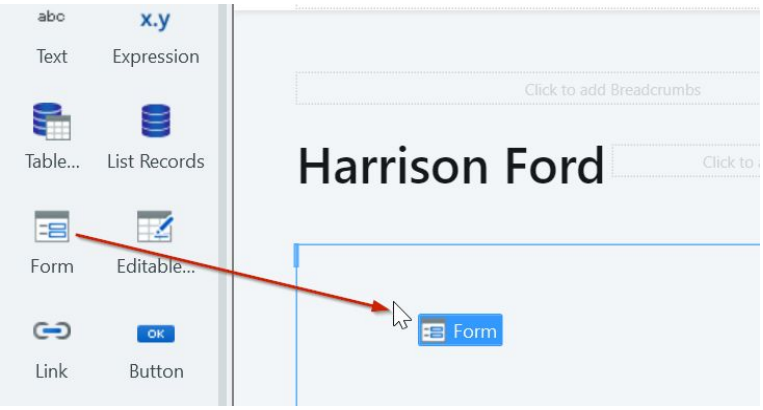
d) In the **Expression Value** dialog, select or enter the value:

GetPersonById.List.Current.Person.Name + " " + *GetPersonById.List.Current.Person.Surname*

This expression uses the Name and the Surname of the person fetched in Aggregate defined in the Preparation. Since it is only one, we can use the Current to access it.

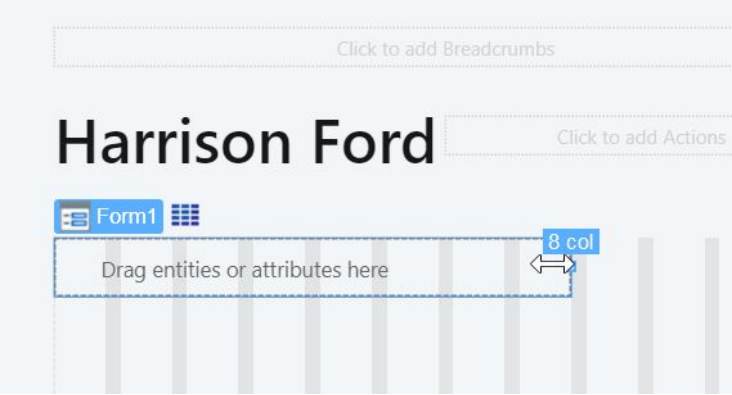
4) Create the Form to display the person's details, using as source the Aggregate created in the previous step. All the attributes should be displayed, except the Id. Make it 8 columns wide.

   a) Drag and drop a **Form** Widget to the **MainContent** area.

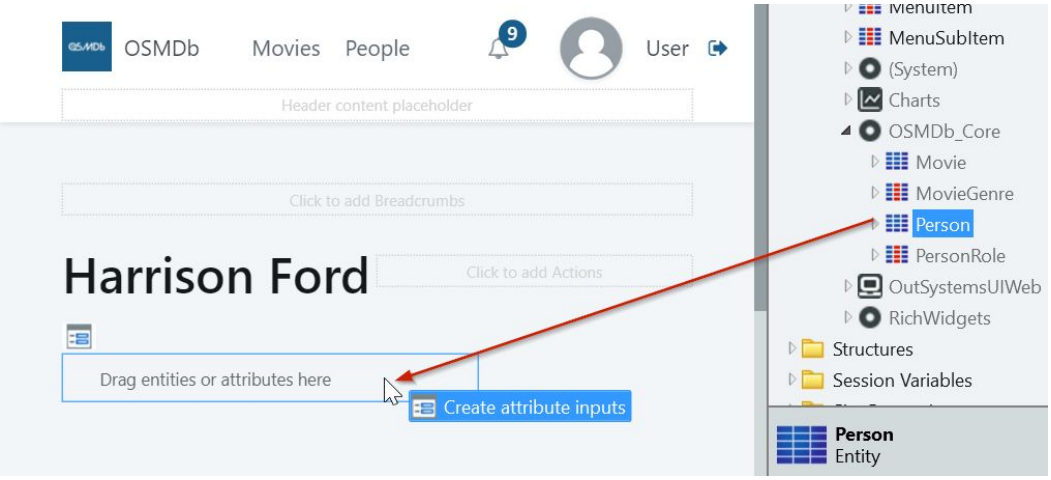


   b) In the properties editor of the Form, click on the **Source Record** dropdown and select *GetPersonById.List.Current*

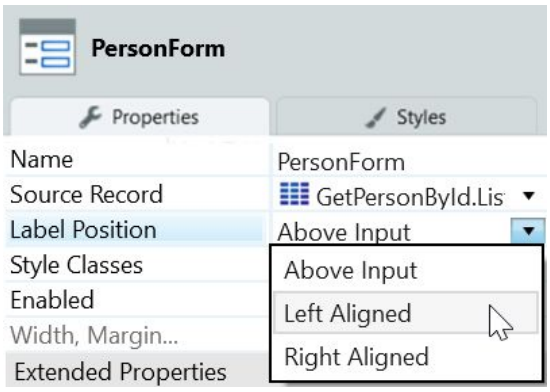
   c) Drag the endpoint of the right side of the form to *8 col* to make it wider.



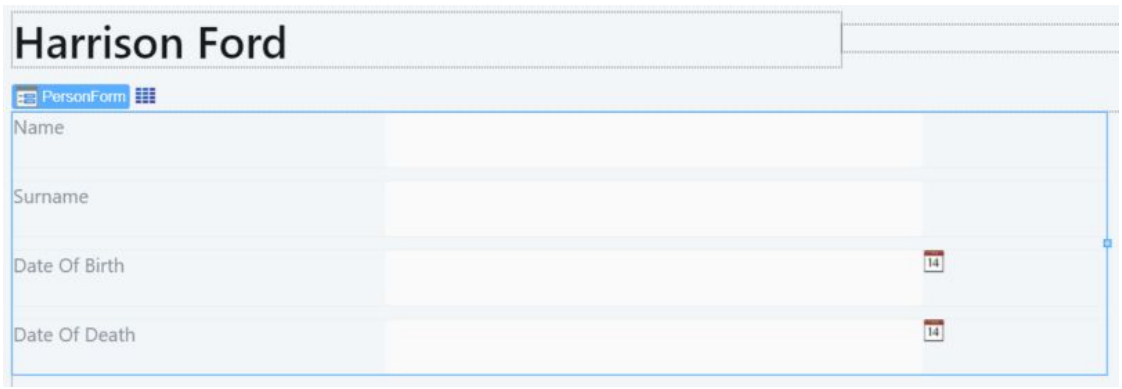
d) Switch to the **Data** tab and drag and drop the **Person** Entity to **Form1**. Notice that dragging a full Entity into a **Form** automatically generates a **Container**, with a **Label** and an **Input**, for every attribute of the Entity dragged.



e) Notice that the Form's name has changed from **Form1** to *PersonForm*. Change the value of the **Label Position** property of the PersonForm to *Left Aligned*.

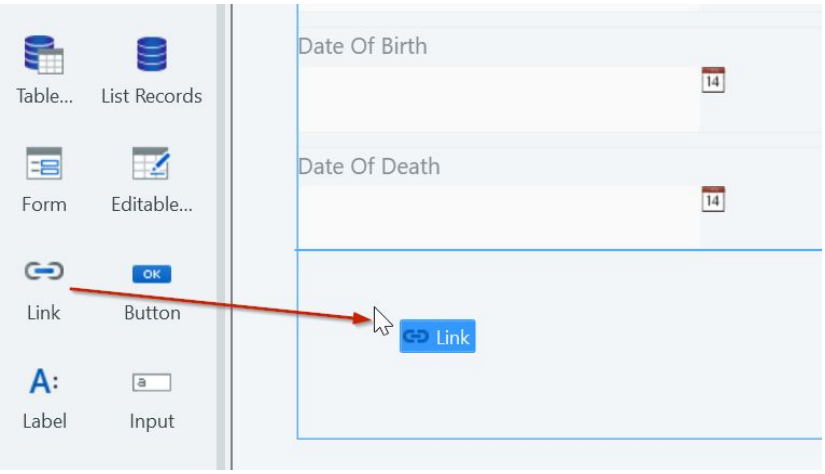


f) The PersonForm should look like this



5) Create a **Link** to navigate back to the People Screen below the Form.

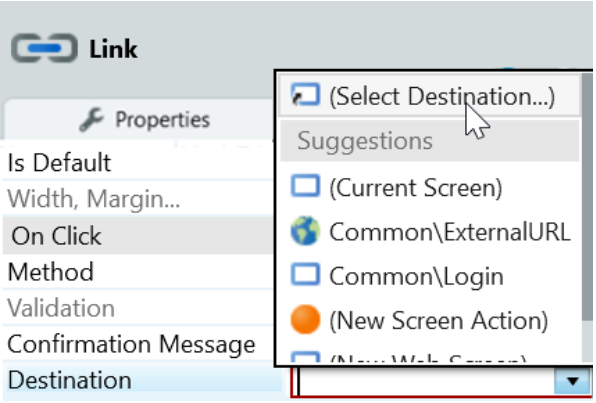
   a) Drag and drop a **Link** below the **PersonForm**.



   b) Without moving input focus, type in *Back to list* inside the Link.

   c) In the canvas, mouse over the **Back to list** text and then select the Link element.

   d) In the Link properties, click on *(Select Destination...)* on the dropdown or simply double-click the Destination property.



   e) In the **(Select Destination)** dialog, expand the MainFlow and double-click the People Screen.

6) Now we want to navigate from the People Screen to the PersonDetail Screen, so that we can click on a person name and access the new PersonDetail Screen to edit its information.

   a) In the Interface tab, double-click on the People Screen to open it.

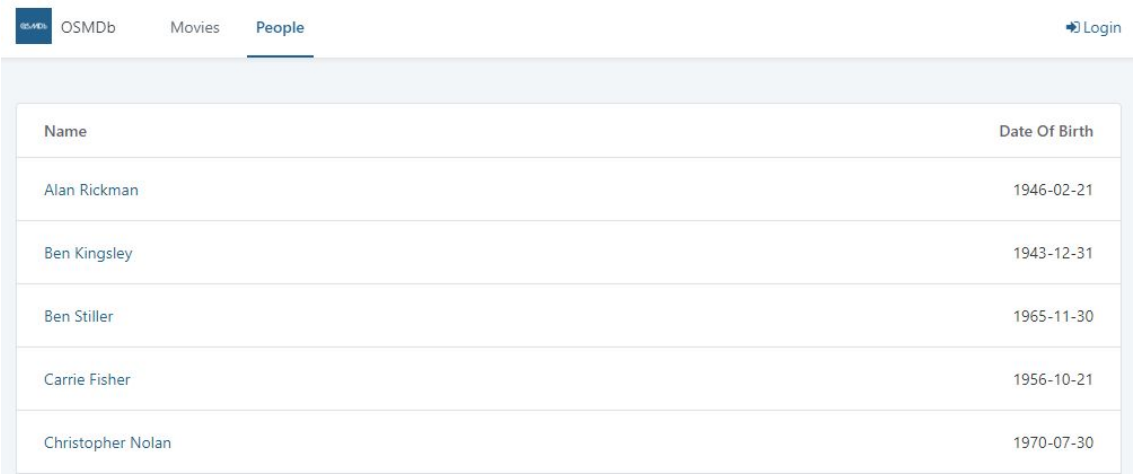
   b) In the **PersonTable**, select the name of the person, inside the **Name** column, and select the option *Link to MainFlow\PersonDetail Web Screen*.

7) Publish the new updates to the server and let's test the updates made to the OSMDb application.

   a) Click the **1-Click Publish button**  to publish the module to the server and verify that the module was published successfully in the **1-Click Publish tab**.

   b) Click on the **Open in Browser** button  The **Movies** Screen should be displayed.

   c) Navigate to the **People** Screen. It should now display links to the **PersonDetail** Screen.



   d) Click on a name displayed in the People Screen and check if the navigation to the **PersonDetail** Screen happens properly.

   e) Click the **Back to list** Link in order to return to the **People** Screen.

End of Lab

In this lab, we created the list and detail Screens for the movies and for the people in the database.

We started to create those Screens by defining their Preparations, with the Aggregates that fetched the relevant data to be displayed on each Screen.

Then, using Table Records and Forms, we defined the layout to display the information in the browser. To help end-users navigate the app, we also created Links between those Screens, as well as Links to the Screens in the Top Menu.

Now we have a simple app where we can see all the movie and people data, using the Entities created in the previous Lab.