

Detail Screens

Exercise

Outline	2
Hands-on	3
Movie Detail Screen	3
Create/Update Movies	6
Linking the Screens	7
People Detail Screen	9

Outline

In this exercise, we will continue building the UI of our app. We will build two Detail Screens to allow creating and editing new movies and people.

- MovieDetail Screen
 - a. Screen to display the details of a particular movie, identified by its id, in a Form.
 - b. Make sure that the Screen will allow creating new movies, as well as editing existing ones.
 - c. Build the logic to create/update a movie in the database.
 - d. 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, and a Link at the top of the Screen to allow creating a new movie.
- PeopleDetail Screen
 - a. Screen to display the details of a particular person, identified by their id, in a Form.
 - b. Make sure that the Screen will allow creating new people, as well as editing existing ones.
 - c. Create the logic to create/update a person in the database.
 - d. Add a Link between the PersonDetail Screen and the People Screen to go back to the list when needed. In the People Screen, add a Link between every person to their PersonDetail Screen to display its detailed data and a Link at the top of the Screen to allow creating a new person.

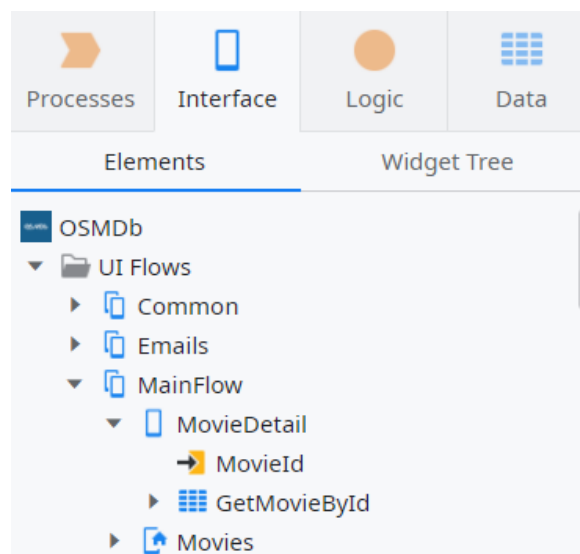
Hands-on

In this exercise, we will continue developing the UI of our app by creating two Detail Screens for movies and people.

Movie Detail Screen

For the MovieDetail Screen, we need to create a new **Empty Screen**, as we did in the first exercise, and call it *MovieDetail*. This new screen should also be accessible to everyone.

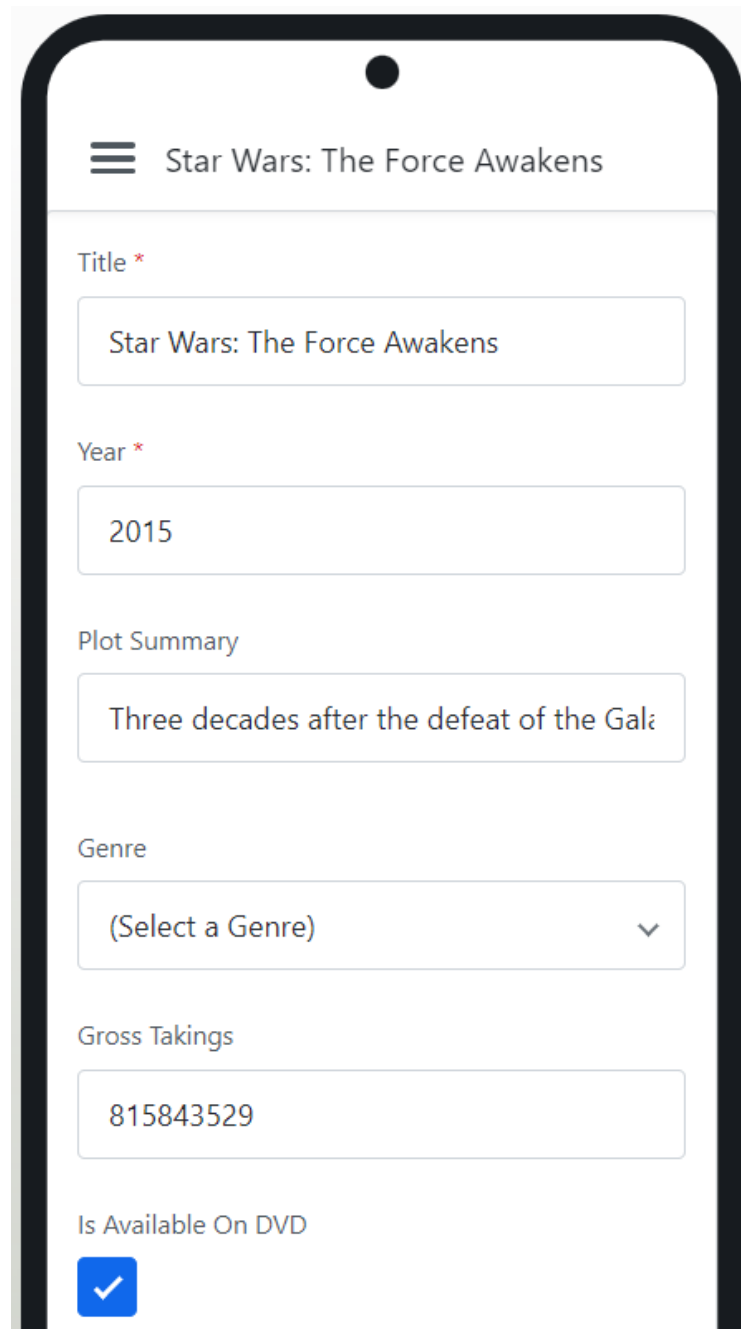
This Screen will display the details of a particular movie. So, we should start by fetching the data for a particular movie, using an Aggregate. To do so, we need to distinguish which movie we want to display, so we need to pass that information to the MovieDetail Screen through an **Input Parameter** of the type **Movie Identifier** and use it to only fetch the data related to that movie.



For the UI part, we need to use a Form. The Form will have a field for every attribute of the Movie Entity, except the Id. It will also automatically create a Save button that we will deal with later. In the **Title** section of the Screen we should have the following behavior:

- If the Id is *NullIdentifier()*, it should display *New Movie*.
- Otherwise, it should display the Title of the movie.

In the end, the Screen should look like the following screenshot:



The screenshot shows a mobile app interface for editing movie details. At the top, there is a hamburger menu icon and the title 'Star Wars: The Force Awakens'. Below this, the form contains several fields: 'Title *' with the value 'Star Wars: The Force Awakens', 'Year *' with the value '2015', 'Plot Summary' with the value 'Three decades after the defeat of the Gal...', 'Genre' with a dropdown menu showing '(Select a Genre)', 'Gross Takings' with the value '815843529', and 'Is Available On DVD' with a checked checkbox.

Star Wars: The Force Awakens

Title *

Star Wars: The Force Awakens

Year *

2015

Plot Summary

Three decades after the defeat of the Gal...

Genre

(Select a Genre) ▼

Gross Takings

815843529

Is Available On DVD

☒

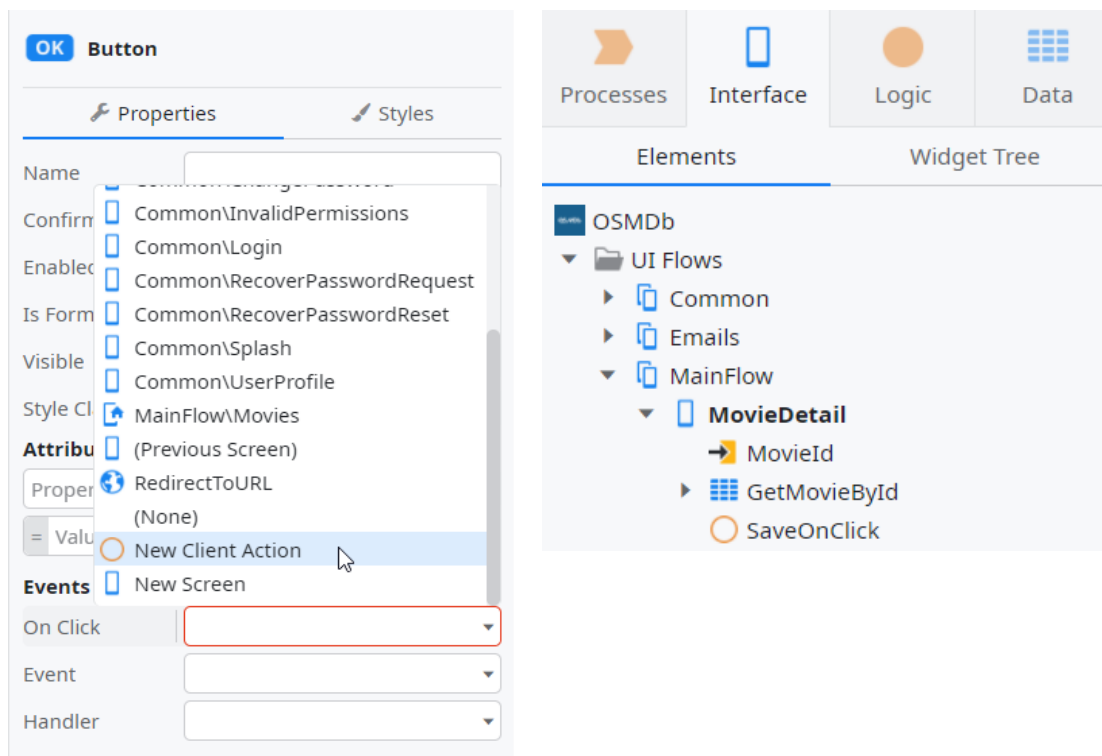
At this point, it will not be possible to navigate to the MovieDetail Screen in the browser or even publish the app, since there are errors in the app. Use the screenshot above as a visual reference.

Now that we have concluded the visual part, let's fix the errors in the app and develop the logic part to create or update movies.

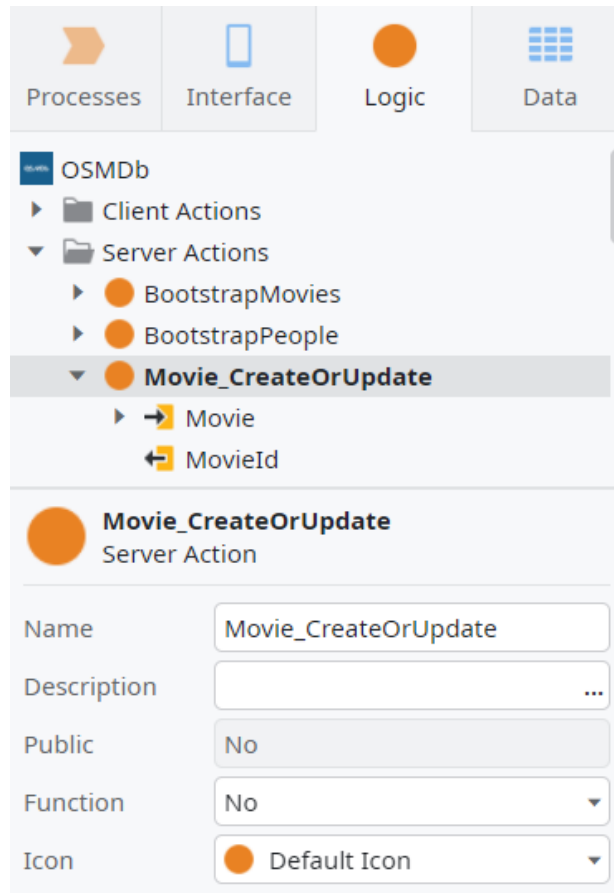
Create/Update Movies

When data was added to the Form, OutSystems automatically created a new Save button. This happens because we have a Form with input fields in it. It is very common to also have a Button or a Link to allow end-users to save and submit the information to the database. And that's what we want as well for our OSMDb app. So let's build the logic for that.

Select the Button in the MovieDetail Screen and set its **OnClick** behavior to a new Client Action. As we know, Client Actions live as well in the scope of a Screen, so the Screen will have a **SaveOnClick** Action that will be triggered when the Button is clicked in the browser. Cool and practical, isn't it?



This **SaveOnClick** Action should have the logic to create/update a movie in the database. In the Logic tab, we need to create a Server Action to create the server-side logic. And here we have access to the Entity Actions we need! The Action should expect a Movie record and return the Id of the movie created/updated in the database.



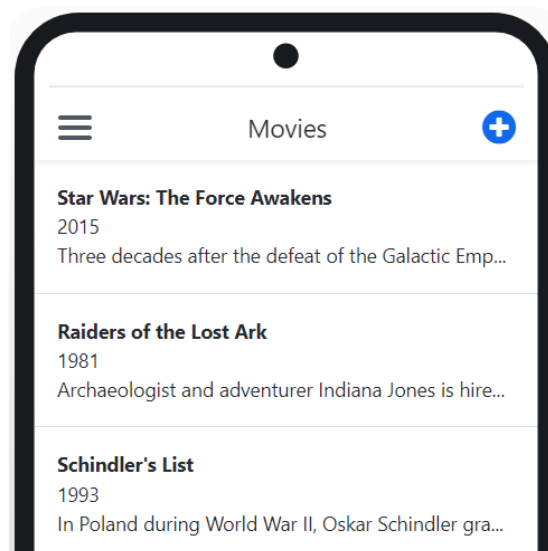
Note: This Server Action directly calls the *CreateOrUpdate* Entity Action to perform the logic we need (besides assigning the output with the movie's Id).

We can also add additional logic to the Action before or after we create/update the movie in the database. For example, imagine that we just want to create/update the record if the user has a certain role. We can easily add that logic before calling the Entity Action. This allows us to have more control over our data and secure it the way we want to.

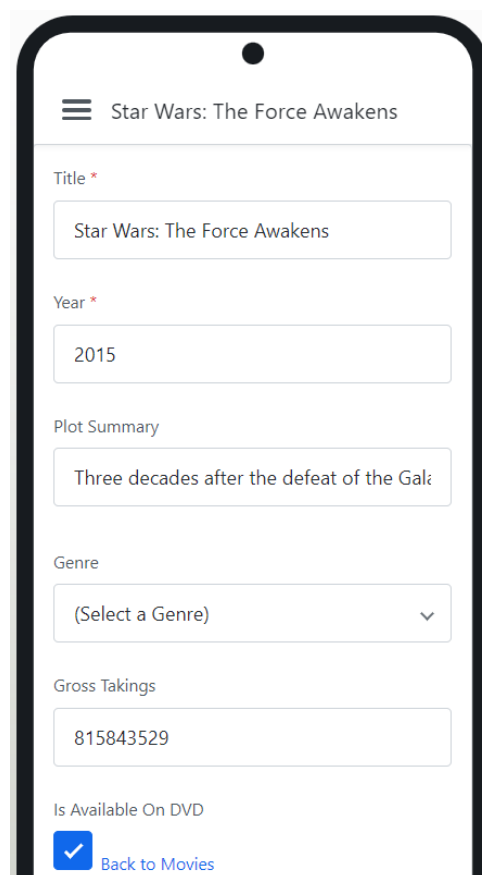
Linking the Screens

When we publish and open it in the browser, we still cannot navigate from the Movies Screen to the MovieDetail Screen. So, let's create a **Link** between the Movies Screen and the MovieDetail Screen using the title of the Movie. The MovieDetail Screen has an input parameter that expects to receive a value, so we need to pass the **Id of the selected movie**.

Also, we want to be able to create a new movie. So, on the top right corner of the Screen, let's add an icon, the plus circle icon, and link it to the MovieDetail Screen. This time, since we're creating a new movie, the value for the Id should be *NullIdentifier()*.



We're almost done with the movies. Now, we just need to create a way for the end-user to return to the Movies Screen, when on the MovieDetail Screen.



People Detail Screen

We're done with the movies. Now, we want to create a similar UI for the data in the People Entity! Let's do it!

