

# List and Detail Screens - Exercise

## Table of Contents

<b>Outline</b>	<b>1</b>
<b>Hands-on</b>	<b>2</b>
Movie List and Detail Screens	3
Movies Screen	3
MovieDetail Screen	4
Create/Update Movies	5
Linking the Screens	8
People List and Detail Screens	9
Menu	10

## Outline

In this exercise, we will start building the UI of our application. We will build four Screens, two List Screens to list the movies and the people in the database, and two Detail Screens to allow creating and editing new movies and new people.

- Movies Screen
  - a. Screen to display the list of movies in the Movie Entity.
  - b. The list of movies should be displayed in a tabular layout, with the title, year, plot summary and the gross amounts.
- 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.
- People Screen
  - a. Screen to display the list of people in the Person Entity.
  - b. The list of people should be displayed in a tabular layout, with the name, surname and date of birth.
- PeopleDetail Screen
  - a. Screen to display the details of a particular person, identified by its 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 the respective 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 start developing the UI of our application by creating two list screens and two detail Screens, for movies and for people. This will be done in the OSMDb module, which is our UI module.

Before we start, we must not forget that the data model was defined in the OSMDb\_Core module, so the first thing to do would be to reference the Entities in the OSMDb module, using the Manage Dependencies option.

---

**NOTE:** If the Entities are not visible when trying to reference them from the OSMDb module, make sure the OSMDb\_Core module is published, or that the Entities are set to Public. If they are not, we need to change them and publish the module again.

---

## Movie List and Detail Screens

Now that we have the Entities available in the OSMDb module, it's time to create the Screens for the Movies. Let's start by the Movies Screen.

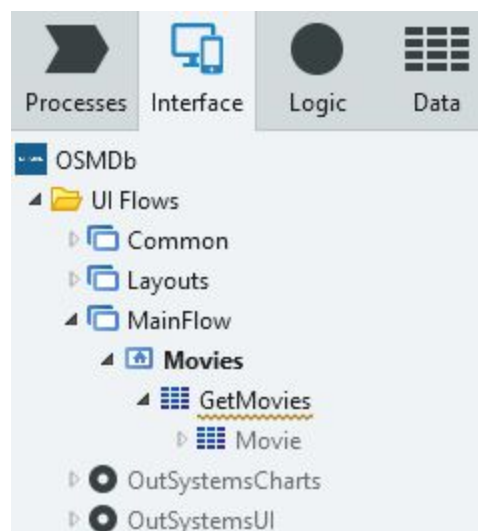
### Movies Screen

To use the Screen already created, let's rename the **HomeScreen** to *Movies* and delete the Expression to display the static message.

To build a list Screen, we need two things:

1. Fetch data from the database.
2. Build the UI with the data we want to display, fetched in the previous step.

To fetch the data from the database, create a new Aggregate that fetches all the Movies.



Then, we need to create a Table that will display the Title, Year, Plot Summary and Gross Takings Amount of the movies being fetched from the Aggregate. Select the correct widget from the widget toolbar on the left, drag it to the Screen and follow the Service Studio hints to complete the table!

At the end, the Screen should look like this:

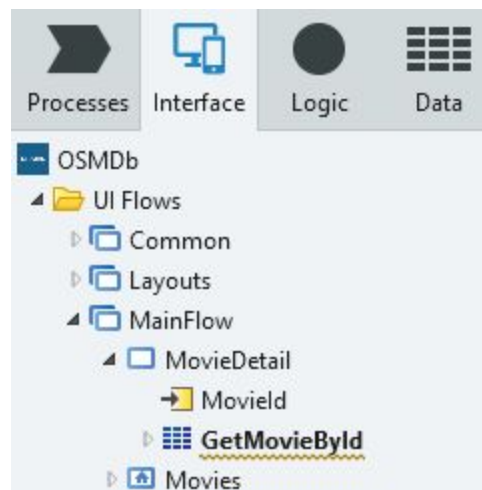
## Movies

Title ↕	Year ↕	Plot Summary ↕	Gross Takings Amount ↕
Star Wars: The Force Awakens	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.	\$815,843,529.00
Raiders of the Lost Ark	1981	Archaeologist and adventurer Indiana Jones is hired by the US government to find the Ark of the Covenant before the Nazis.	\$242,374,454.00
Schindler's List	1993	In Poland during World War II, Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.	\$16,439,233.00
Avatar 2	2020	(unknown)	\$0.00
Indiana Jones and the Last Crusade	1989	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.	\$197,171,806.00

## MovieDetail Screen

For the MovieDetail Screen, we need to create a new **Empty Screen**, like we did on the first exercise, call it *MovieDetail*, and set it as **Anonymous**.

This Screen will need to display the details of a particular movie. So, we should start with fetching the data for a particular movie, using an Aggregate. To do that 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 type **Movie Identifier** and use it to just 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.

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

The screenshot shows a web application interface for OSMDb. At the top left is the OSMDb logo, and at the top right is a 'Login' link. The main heading is 'Star Wars: The Force Awakens'. Below this is a form with the following fields:

- Title \***: A text input field containing 'Star Wars: The Force Awakens'.
- Year \***: A text input field containing '2015'.
- Plot Summary**: A text area containing 'Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.'
- Gross Takings Amount**: A text input field containing '815843529'.
- Is Available On DVD**: A checkbox that is checked.

At the bottom left of the form is a blue 'Save' button.

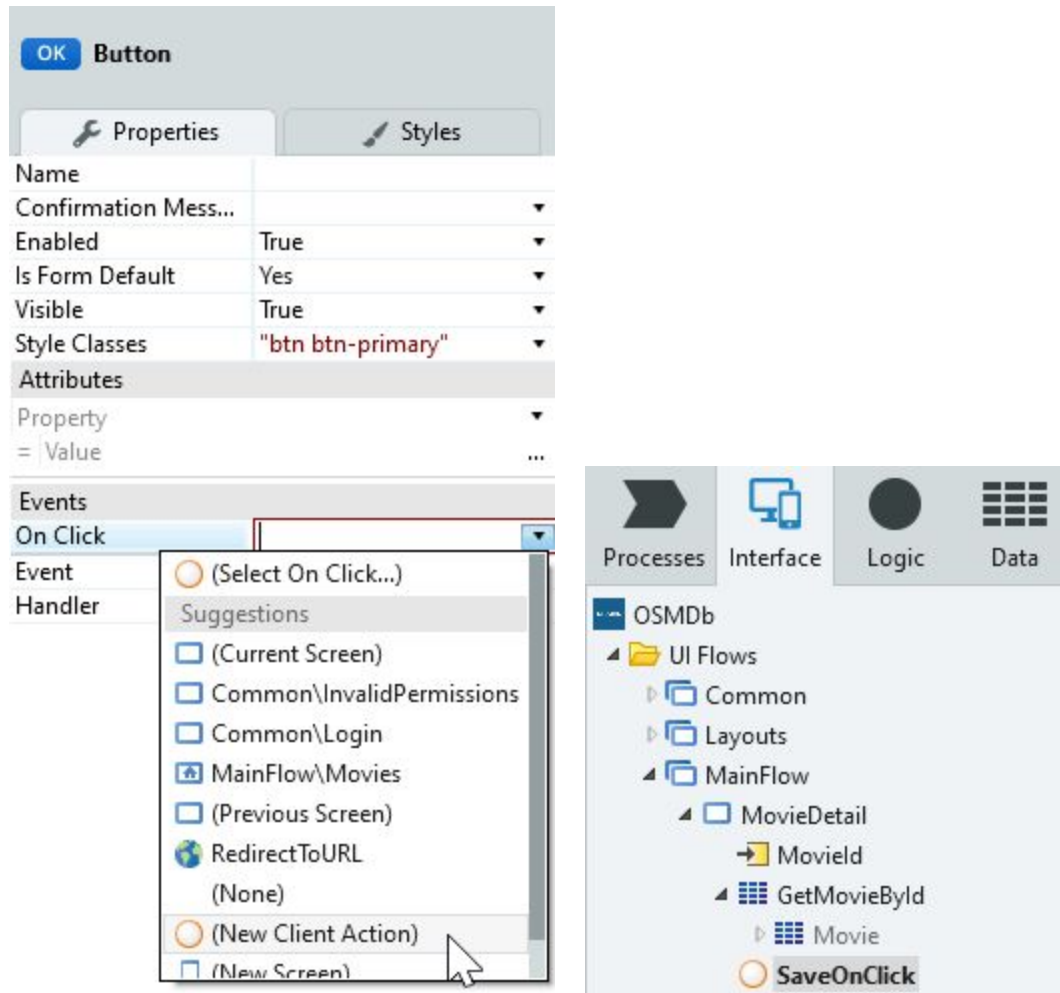
At this point it will not be possible to navigate to the MovieDetail Screen in the browser, or even publish the module, since there are errors in the application. Use the screenshot as a visual reference.

Now that we're finished, let's fix the errors in the application.

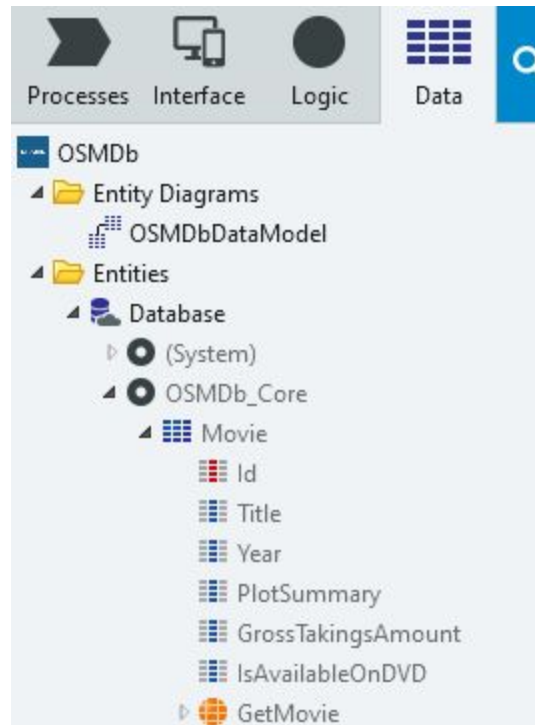
## Create/Update Movies

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

Let's 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.

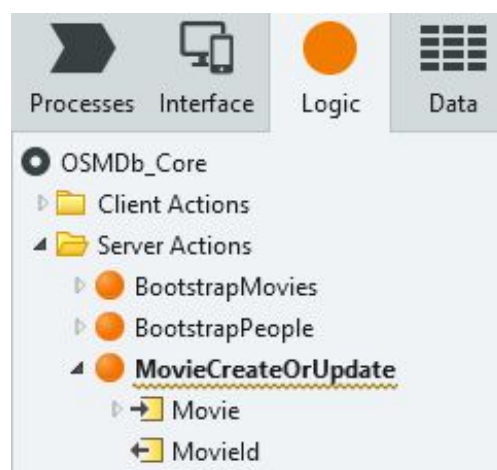


This SaveOnClick Action should have the logic to create/update a movie in the database. However, we have a problem! The Entities are exposed as read only to this module, so we only have the Get Entity action available. Which is not a bad thing!



Exposing the Entities as read only is actually a good development practice, since this way, we make sure that no module that references the Entity has access to logic that can directly change the information in the database.

So, to solve this problem, we need to go back to the Core module and create a Server Action to Create/Update a Movie in the Database. 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.



Let's set the Action as **Public**, reference it in the OSMDb module and use it to complete our logic in the MovieDetail Screen.



**NOTE:** At this point, this new Server Action directly calls the CreateOrUpdate Entity Action to actually perform the logic that we need (besides assigning the output with the Id of the movie), which may seem strange at first! However, by creating these wrapper Actions and expose them as Public has two advantages, compared to exposing directly the Entity with write permissions:

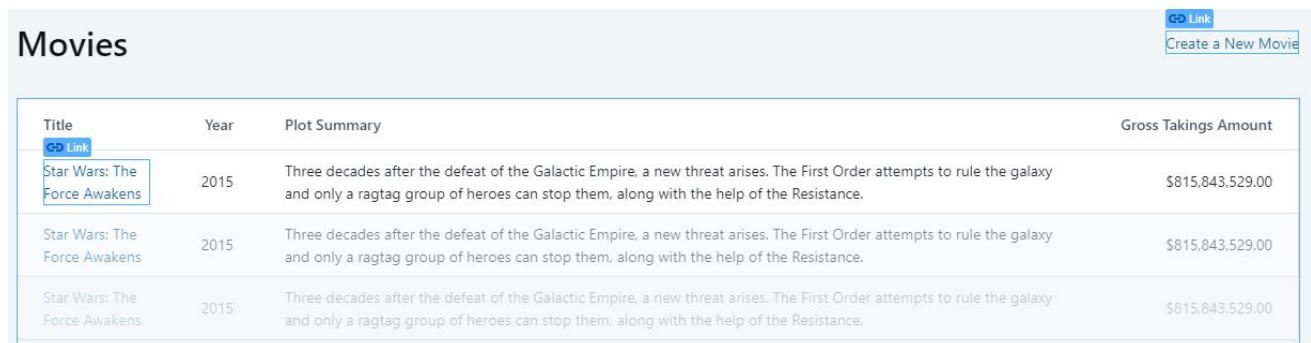
1. We only expose the logic that we want in the module. For instance, at this point it's not possible to delete a movie from the database, from the OSMDb module.
2. We can add additional logic to the Action, before or after we actually create/update the movie in the database. As an example, imagine that 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 on our data and secure it the way we want to.

## Linking the Screens

Now the application has no errors, but when we publish and open it in the browser, we still cannot navigate from the Movies Screen to the MovieDetail Screen.

So, in the Movies Screen, let's create a **Link** from the title of the Movie to the MovieDetail Screen. The MovieDetail Screen is expecting a value for its input parameter, so we need to set it to 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 some text, *Create a New Movie*, and link it to the MovieDetail Screen. This time, since we're creating a new movie, the value for the Id should be **NullIdentifier()**.



Title	Year	Plot Summary	Gross Takings Amount
<a href="#">Link</a> Star Wars: The Force Awakens	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.	\$815,843,529.00
Star Wars: The Force Awakens	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.	\$815,843,529.00
Star Wars: The Force Awakens	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the help of the Resistance.	\$815,843,529.00

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



OSMDB Login

## Star Wars: The Force Awakens

Title \*

Year \*

Plot Summary

Gross Takings Amount

Is Available On DVD

☒

[Back to Movies](#)

## People List and Detail Screens

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!

OSMDB Movies People Login

## People Create a New Person

Name ↕	Surname ↕	Date Of Birth ↕
Harrison	Ford	13 Jul 1942
Steven	Spielberg	18 Dec 1946
Ben	Kingsley	31 Dec 1943
Philip Seymour	Hoffman	23 Jul 1967
Alan	Rickman	21 Feb 1946

 OSMDB [Movies](#) [People](#) [Login](#)

## Philip Seymour Hoffman

Name \*

Philip Seymour

Surname \*

Hoffman

Date Of Birth \*

07/23/1967

Date Of Death

02/02/2014

Save

[Back to People](#)

## Menu

To complete the exercise, drag and drop the Movies and People Screen to the Menu element, under Common. This will make sure that you will have a menu entry for both list screens when navigating in the browser. By clicking on a menu entry, the application will navigate to the respective Screen.

