

# Security and Exceptions

## Exercise - How to

<b>Outline</b>	<b>2</b>
<b>How to</b>	<b>2</b>
Create a Role	2
Restricting Access on Screens	4
Grant Access to Users	9
Checking Permissions in the Logic	13
Add Cast and Crew Logic	13
Save Movie and Person Logic	21

# Outline

In this exercise, we will add some security features to the OSMDb app. So far, all the Screens can be accessed by everyone, which means that every functionality of the app is also available to everyone.

In this exercise, we will create the concept of the administrator user of the app. We want to make sure that only administrators can add or edit movies or people in the database, as well as add members of the cast and crew to a movie. We will achieve this in several steps:

1. Create the OSMDbAdmin role.
2. Grant the role to a user.
3. Guarantee, on the logic side, that only OSMDbAdmin users can add a member of the cast/crew to a movie.
4. Use Exceptions to display an error message to the user if they do not have the correct permissions.
5. Restrict the access to the AddCastAndCrew Screen to OSMDbAdmin users.
6. Make sure that only Admin users can edit the movie and people's information, but everyone else can at least see the information.

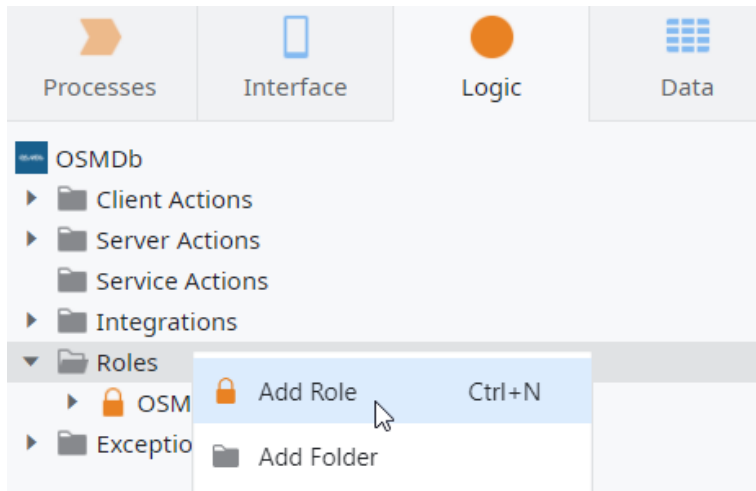
## How to

In this section, we'll describe exercise 7 – *Security and Exceptions*, step by step.

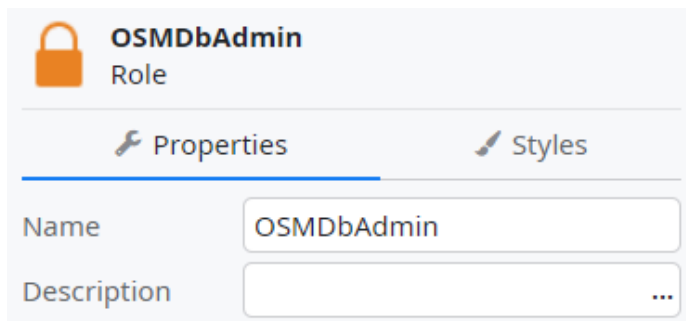
### Create a Role

Creating a new Role in OutSystems is pretty simple and straightforward. In this section, you'll create a new role for the app's admin, the OSMDbAdmin. We will also take the opportunity to check the built-in functions that are created, and rename an existing role to OSMDbUser.

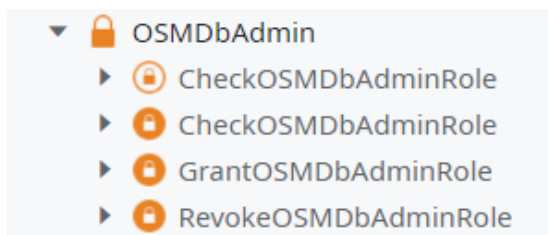
1. In the **Logic** tab, right-click on the **Roles** folder and choose **Add Role**.



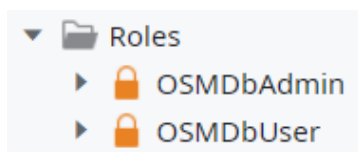
2. Set its **Name** to *OSMDbAdmin*.



And it is as simple as that. Notice that your new role also has built-in functions that will help you check, grant, and revoke permissions in your logic.



3. Rename the **OSMDb** role to *OSMDbUser*. Don't worry about this role now, as you are only going to use it later in the exercises.

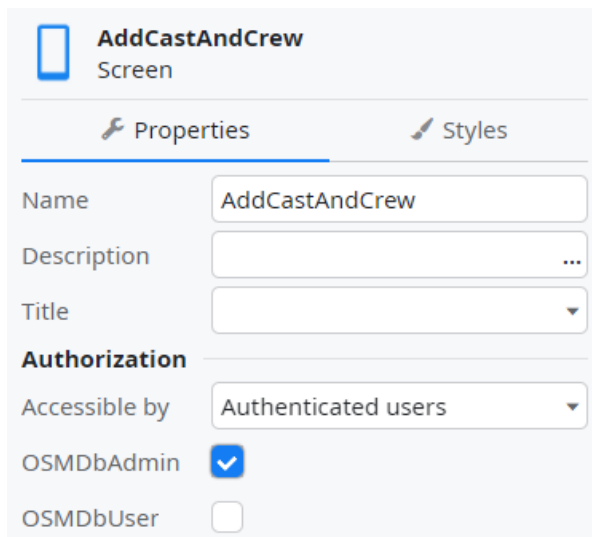


**Note:** The OSMDb role is a role that is automatically created by OutSystems and that has the same name as your app.

## Restricting Access on Screens

Now that we have the OSMDbAdmin role, we can use it to start restricting some functionality. Let's start with the UI. First, we will make sure the AddCastAndCrew Screen will only be accessible by OSMDbAdmin users. Then, the link to the AddCastAndCrew Screen can only be visible to OSMDbAdmin users as well. And finally, all the input fields in the MovieDetail Screen and PersonDetail Screens should be disabled for all users that are not OSMDbAdmin users. This also means that the New Movie and New Person links in the Movie and People Screens should be hidden from all users that are not OSMDbAdmins.

1. In the **Interface** tab, click on the **AddCastCrew** Screen. In its properties, change the **Accessible by** property to **Authenticated users**, and then select the **OSMDbAdmin** role.



**AddCastAndCrew**  
Screen

Properties Styles

Name AddCastAndCrew

Description ...

Title

**Authorization**

Accessible by Authenticated users

OSMDbAdmin ☒

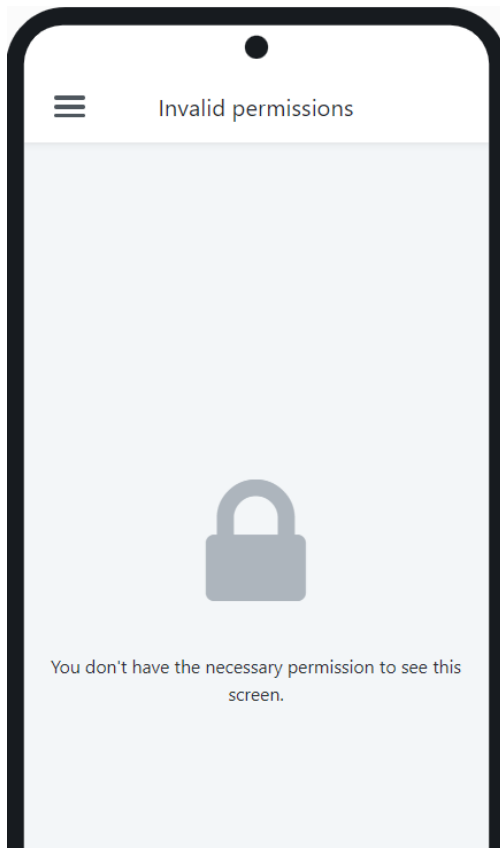
OSMDbUser ☐

Up to this point, the Screen was accessible to everyone. Now, only authenticated users have access to the Screen, specifically, users with the OSMDbAdmin role.

2. Publish the app and open it in the browser.

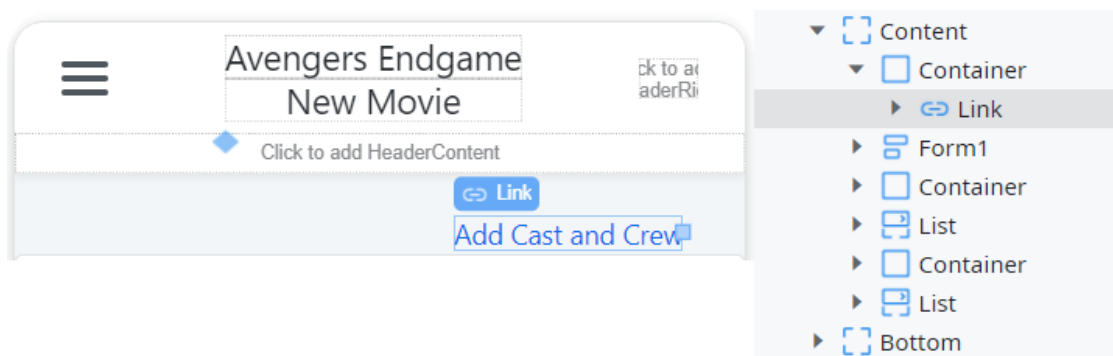


3. Navigate to the AddCastAndCrew Screen. You should see something like this now.



This happens because you do not have an authenticated user with the OSMDbAdmin role yet. And that's why now you see the Invalid Permissions Screen. Don't worry! At the end of the exercise, your user will have all the permissions it needs. But for now, let's keep adding restrictions to the app.

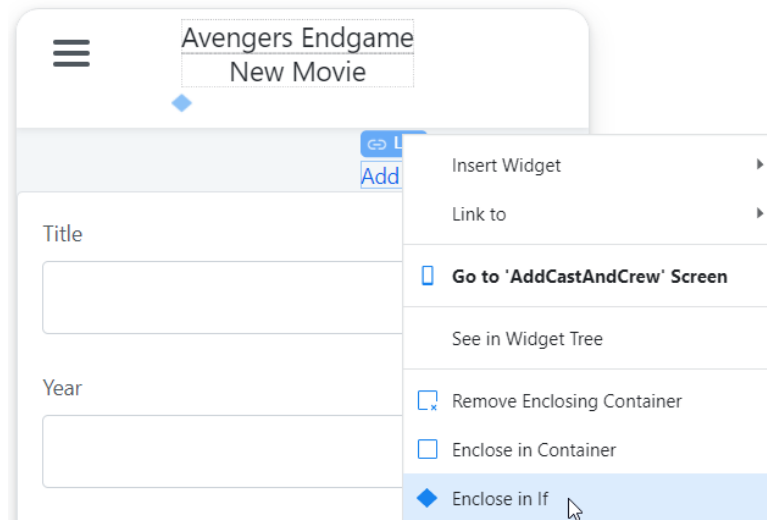
4. Let's hide the Add Cast/Crew link in the MovieDetail Screen for users that are not OSMDbAdmins.
  - a. Double-click the MovieDetail Screen to open it in the preview area.
  - b. Locate the Add Cast/Crew link and select it.



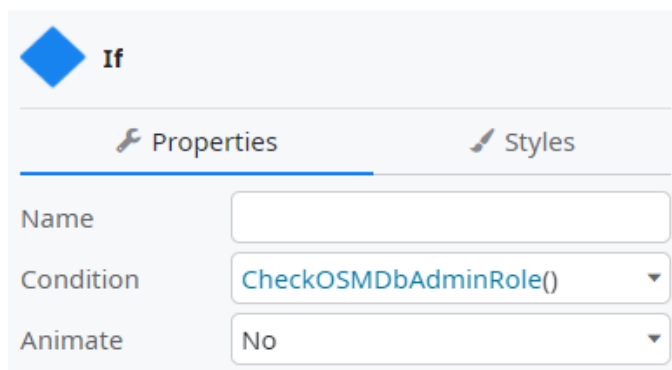
**Note:** When you click on the Link to select it, you may actually select the text inside the Link. That happens because in the preview area, when you click on a widget, it

selects the one “deeper” in the hierarchy, meaning that when I have a text inside a Link, it selects the text. So, use the widget tree to help you select exactly what you want in these cases.

- c. Right-click on the Link and select Enclose in If.

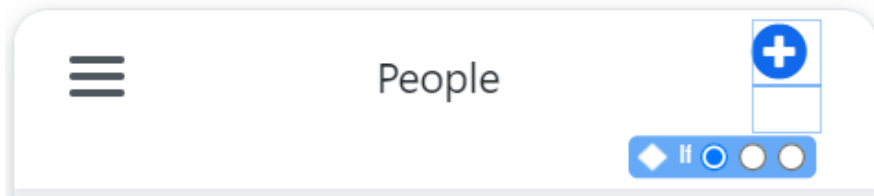


- d. Set the If Condition to CheckOSMDBAdminRole().



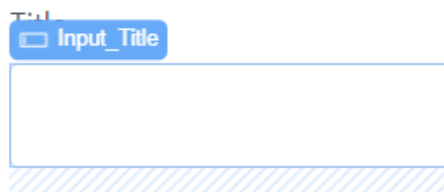
Here, we are leveraging the built-in Client Action that comes with the role. This Check action verifies if the user that is currently logged in has the OSMDBAdmin role. If it does, it shows the link. Otherwise, it shows nothing.

- 5. Apply the same strategy to the **New Movie** Link on the Movies Screen and the **New Person** Link on the People Screen.

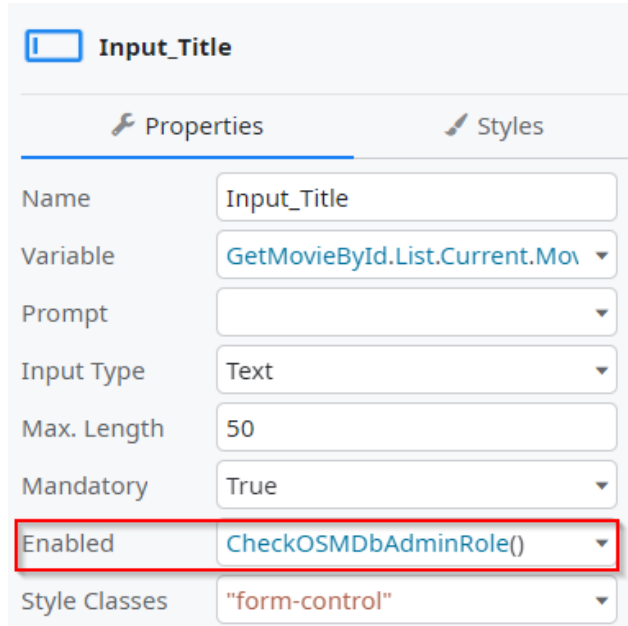


6. The last thing we need to do is to disable the input fields in the MovieDetail and PersonDetail Screens, to make sure that only OSMDbAdmins can edit information. Let's start with the MovieDetail Screen.

- a. Open the **MovieDetail** Screen and select the input field for the **Title**.



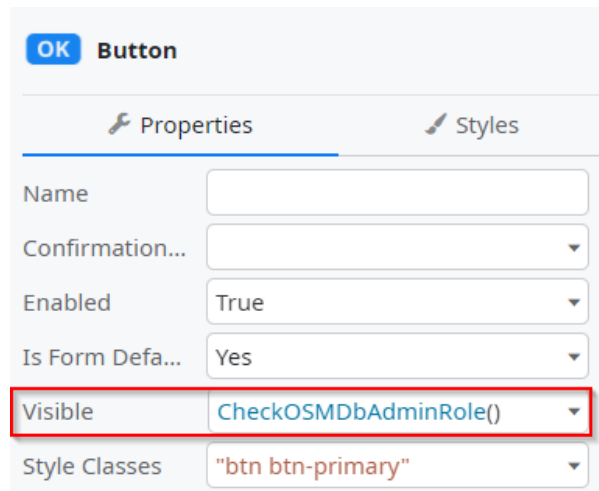
- b. In the properties of the Input, change the **Enabled** property to *CheckOSMDbAdminRole()*.



This property defines in which circumstances the input field is enabled for editing. When set to True, it's always enabled for editing. In this case, it will only be enabled for users with the OSMDbAdmin role.

- a. Repeat the process for the remaining input fields of the Screen.

- b. Last, but not least, select the **Save** Button and set its **Visible** property to `CheckOSMDBAdminRole()`



The screenshot shows the 'Properties' tab of the OutSystems Properties Panel for a 'Button' widget. The 'Visible' property is highlighted with a red box and set to 'CheckOSMDBAdminRole()'. Other properties include 'Name', 'Confirmation...', 'Enabled' (set to 'True'), 'Is Form Defa...' (set to 'Yes'), and 'Style Classes' (set to '"btn btn-primary"').

Property	Value
Name	
Confirmation...	
Enabled	True
Is Form Defa...	Yes
Visible	CheckOSMDBAdminRole()
Style Classes	"btn btn-primary"

7. Now, just repeat the same process for the input fields and Save Button in the PeopleDetail Screen.
8. Publish the app and open it in the browser.



You should be able to navigate to a Person or Movie details page, but the input fields should be disabled.



A mobile application interface for a movie database. The header shows a hamburger menu icon and the title 'Avengers Endgame'. The form contains the following fields:

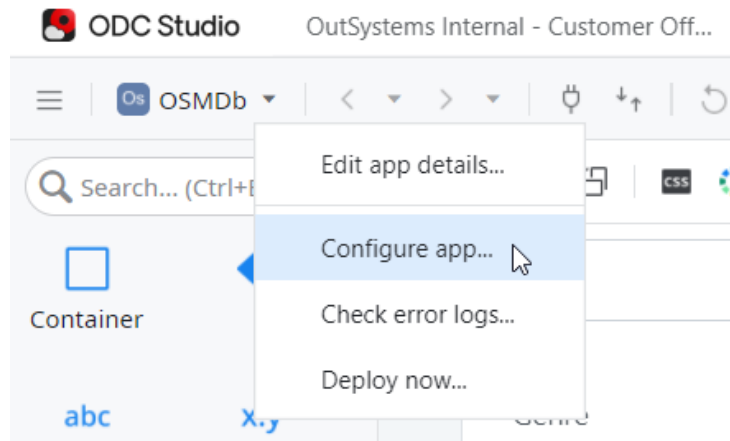
- Title \***: A text input field containing 'Avengers Endgame'.
- Year \***: A text input field containing '2019'.
- Plot Summary**: A text input field containing 'After the devastating events of Avengers:'.
- Genre**: A dropdown menu with the text '(Select a Genre)' and a downward arrow.
- Gross Takings**: A text input field containing '2798000000'.
- Is Available On DVD**: A checkbox that is checked, with a blue checkmark icon.

At the bottom, there is a blue button with a white checkmark and the text 'Back to Movies'.

## Grant Access to Users

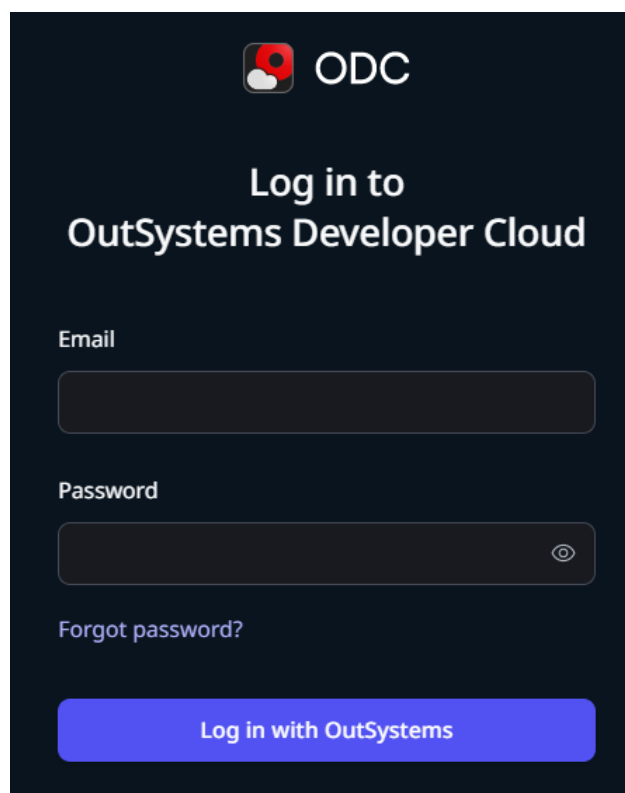
We have many functionalities restricted to the OSMDbAdmin role, but so far we have never logged in or even assigned roles to users. So, let's take some time to do that now.

1. Open the ODC Portal to assign the **OSMDbAdmin** Role to your user.
  - a. In the top right corner of ODC Studio, you can find the icon and name of the app with an option to expand a dropdown. Click on it. Then, on the dropdown, select the option **Configure app...** This will open the **ODC Portal**.

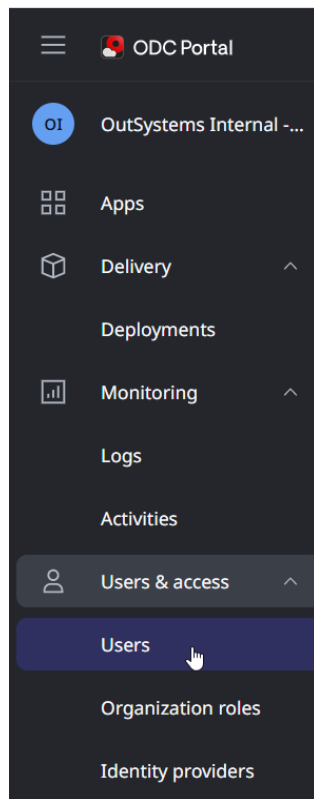


The ODC Portal is where the configurations of the app, among other things, are done, including the user management.

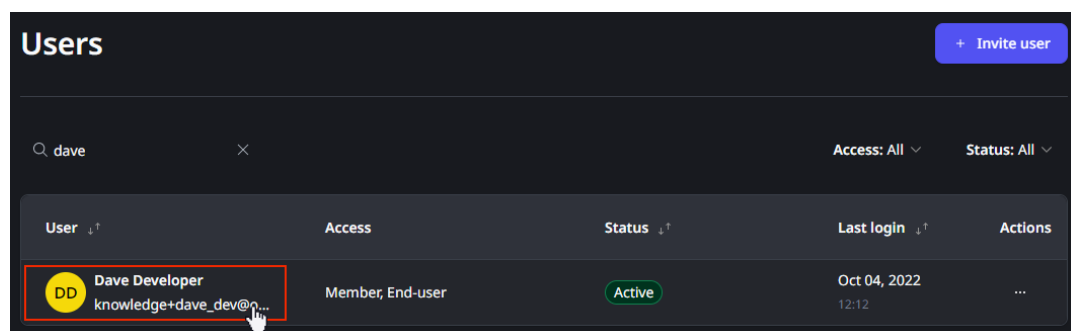
- b. If a login is required, use the same credentials you use to log into the ODC Studio.



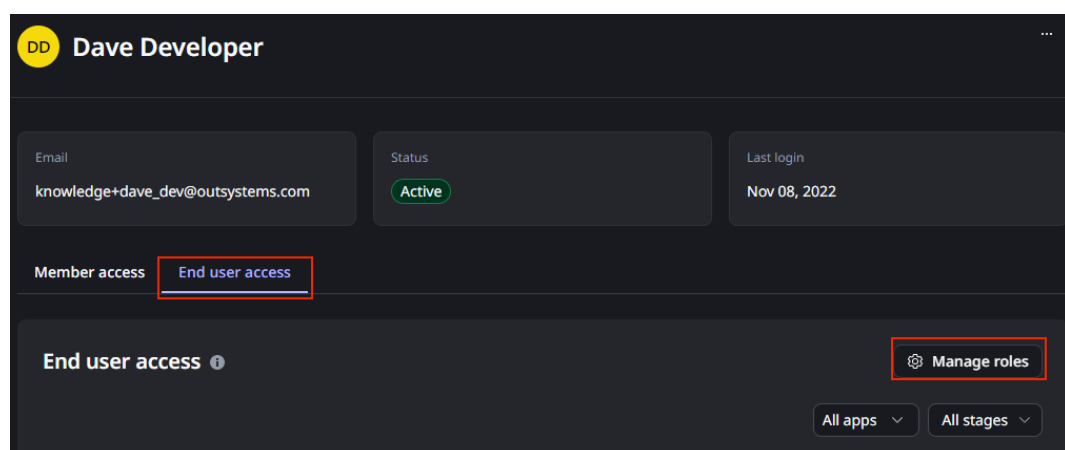
- c. In the ODC Portal, check the left sidebar, find the **Users & access** section and click on the **Users** option to open it.



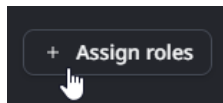
d. Search for your user in the list, then click on it to see its permissions.



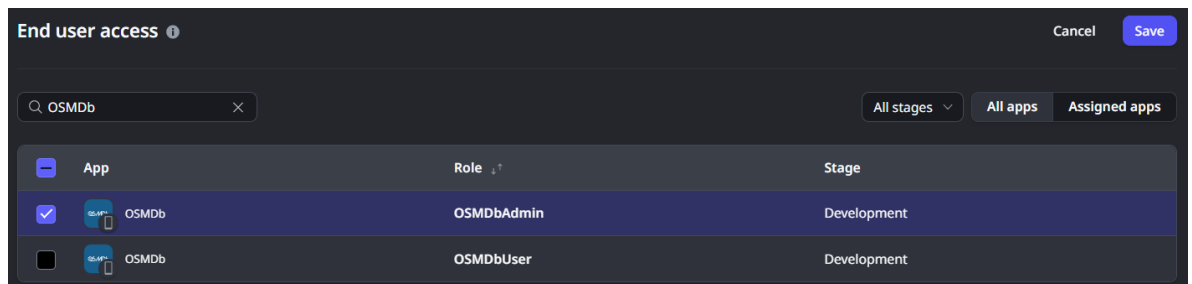
e. In the user detail page, select the **End user access** tab and then click on **Manage roles**.



**Note:** if you don't have any roles assigned to you yet, you will see the option **Assign roles** instead of Manage roles.



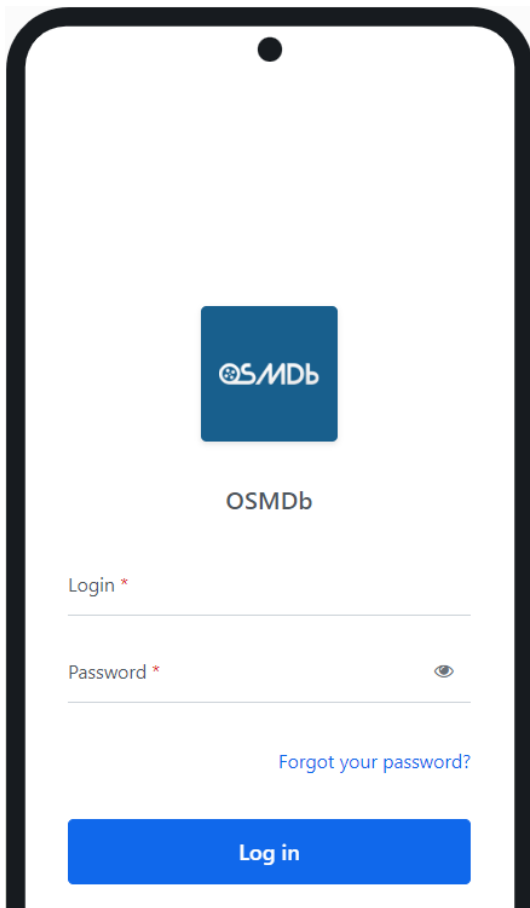
- f. Then, search for your app's name (in case you have more than one app in ODC Studio), check the **OSMDbAdmin Role** (the OSMDbUser role should remain unselected), and click on **Save** when you're done.



2. Back in the ODC Studio, publish the app and open it in the browser.



3. Log in to your application with your user credentials.



4. You should be able to access the OSMDbAdmin functionalities now.

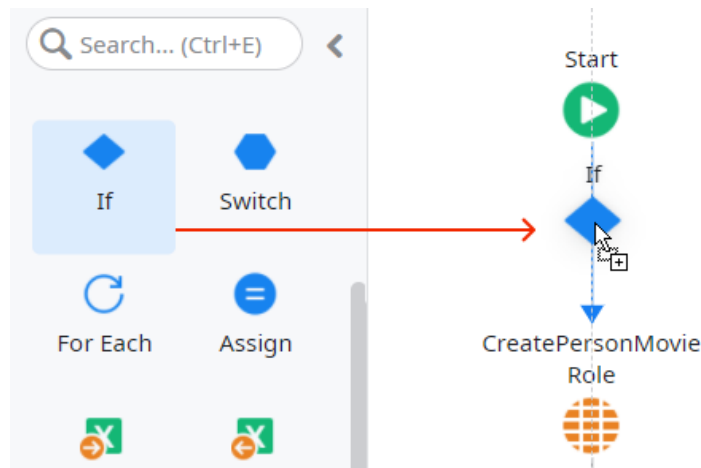
## Checking Permissions in the Logic

After checking the permissions in the UI, it's time to do the same in the logic part. Since we only want to allow OSMDbAdmin users to create/edit information about movies and people, we should also check on the logic of those permissions, before actually performing the changes in the database. The same thing applies to adding a member of the cast and crew to a movie. Let's start with that one!

### Add Cast and Crew Logic

To check permissions when adding a member of the cast or crew to a movie, there are two Actions that we need to work on: the Server Action, **PersonMovieRole\_Create**, and the Client Action, **SaveOnClick**.

1. Change the **PersonMovieRole\_Create** Server Action to ensure that only users with the OSMDbAdmin Role can create the record in the database. Raise a **Not OSMDbAdmin Exception** when the user does not have the OSMDbAdmin Role.
  - a. In the Logic tab, open the **PersonMovieRole\_Create** Action.
  - b. Drag an **If** from the left sidebar and drop it after the Start node.

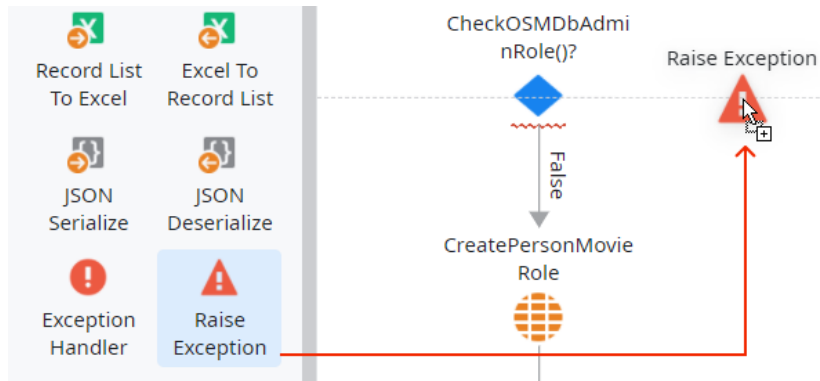


- c. Set the **Condition** of the If to *CheckOSMDbAdminRole()*.

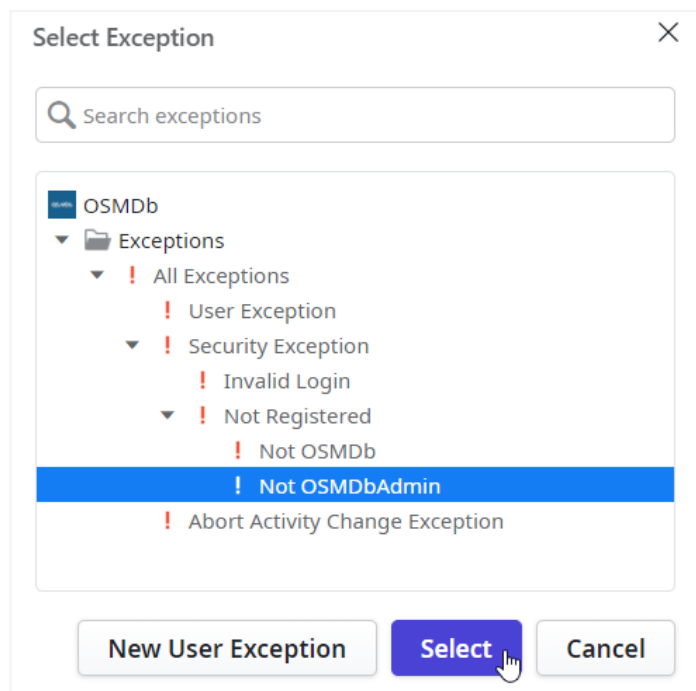
It seems that we're using the same Action here, but we're not. This is the Check Server Action, while in the UI we used the Check Client Action. Their goal is the same, but in reality, they are two different actions.

**Note:** When the CheckRole Action is called without any input parameter, as we did here, the verification is done considering the user that is currently logged in.

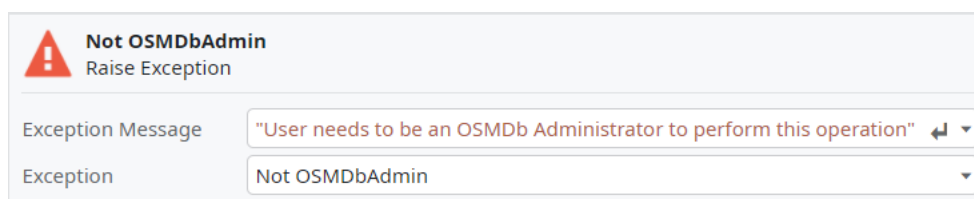
- d. If the user does not have the role, then an Exception should be raised. Drag a **Raise Exception** statement and drop it next to the If.



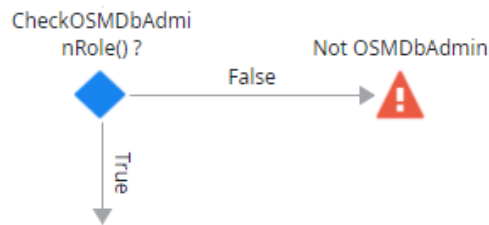
- e. In the new window, choose the **Not OSMDbAdmin Exception**.



This Exception was created automatically when the role was created. Set the **Exception Message** property to *"User needs to be an OSMDb Administrator to perform this operation."*

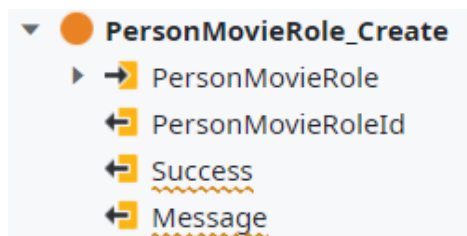


- f. Connect the If to the Raise Exception to create the True branch.  
 g. Right-click on the If and select **Swap Connectors**.

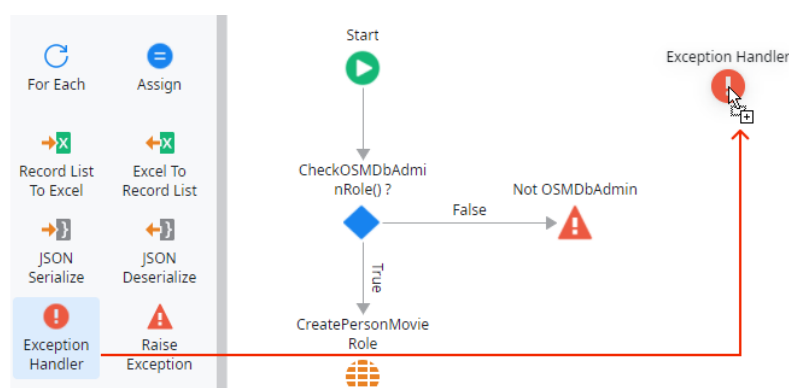


You can swap the connectors of an If to change between the True and False branches. In this example, if the user does not have the role, the condition will be evaluated as False and the Action flow will end, raising an exception.

2. The PersonMovieRole\_Create Action needs to have two extra Output Parameters, one indicating if the operation was successful and one with a message based on the outcome of the operation. Also, we will handle the Exception thrown and create the respective logic to set that the logic was not successful and define the respective message.
  - a. Add a new **Output Parameter** to the PersonMovieRole\_Create Action and call it **Success**. Make sure its **Data Type** is set as *Boolean*.
  - b. Add another Output Parameter to the Action and call it **Message**. Make sure its Data Type is set as *Text*.

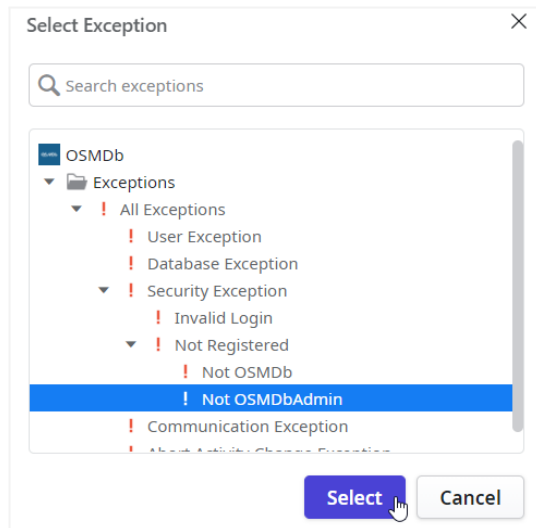


- c. Drag an **Exception Handler** and drop it next to the main flow of the Action.

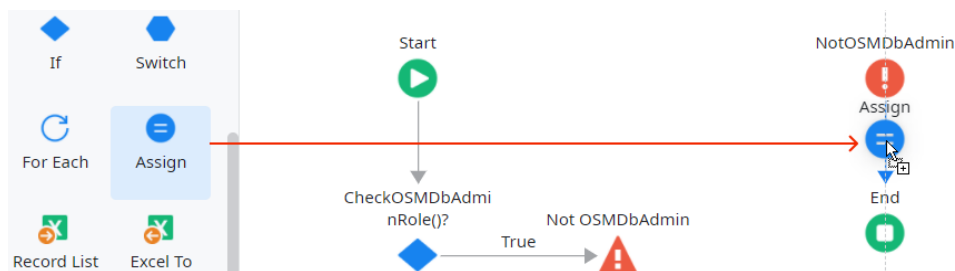




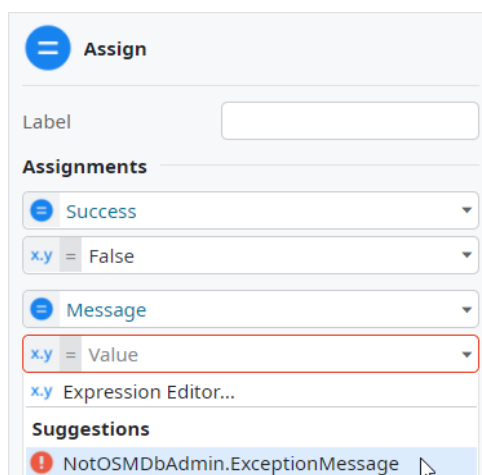
- d. In the new window, select the **Not OSMDbAdmin** Exception.



- e. Drag an **Assign** and drop it on the Exception flow.



- f. Define the following assignments:



**Note:** When we define the Message on the Raise Exception node, that message can be used directly by referencing the Exception and the property name (ExceptionMessage).

- g. Select the Assign right before the End node on the main Action flow and add the following assignments:

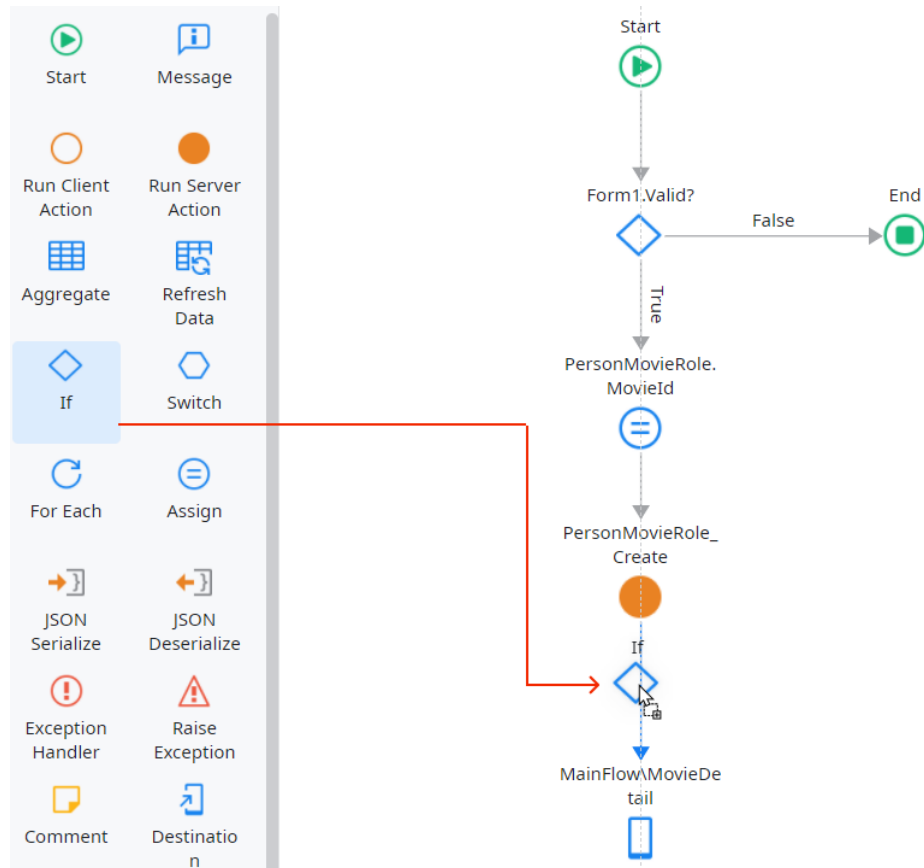
*Success = True*

*Message = "Record created successfully"*

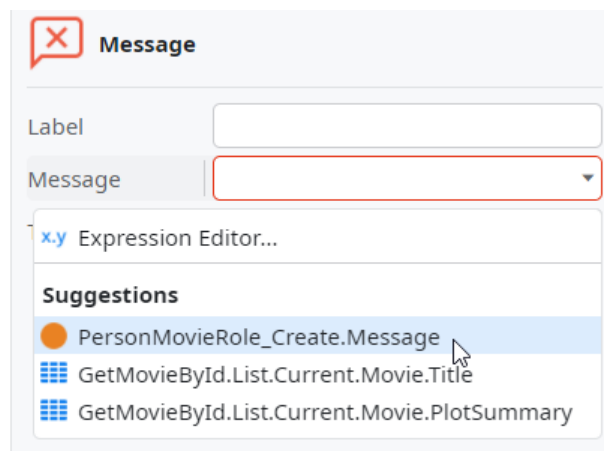
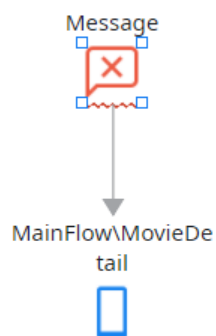
Assignments	
	PersonMovieRoleId
x.y	= CreatePersonMovieRole.Id
	Success
x.y	= True
	Message
x.y	= "Record created successfully"

At this point, the Server Action returns information stating if the operation was successful or not. Also, remember that, for security purposes, it is extremely important to perform these validations server-side.

3. In the **SaveOnClick** Client Action of the AddCastAndCrew Screen, use the output parameters of the **PersonMovieRole\_Create** Action to create a logic to display feedback messages to the user.
- a. Open the **SaveOnClick** Action flow in the AddCastAndCrew Screen.
  - b. Drag and drop an **If** below the **PersonMovieRole\_Create** Action.

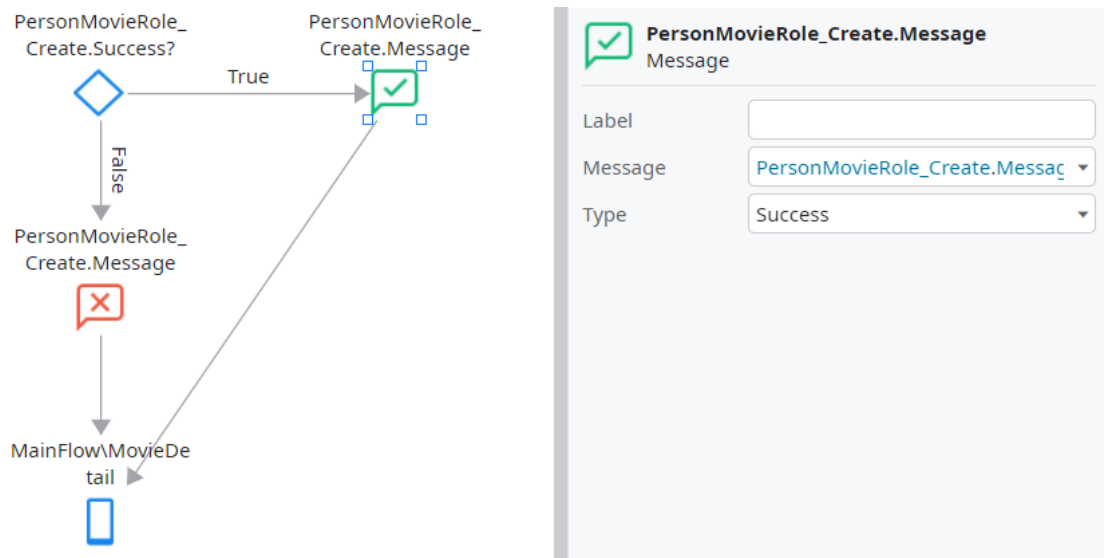


- c. Set the **Condition** of the If to the output of the PersonMovieRole\_Create Action: *PersonMovieRole\_Create.Success*
- d. Drag a **Message** and drop it below the If. Set the **Message** property to the PersonMovieRole\_Create Message output and the **Type** to *Error*.

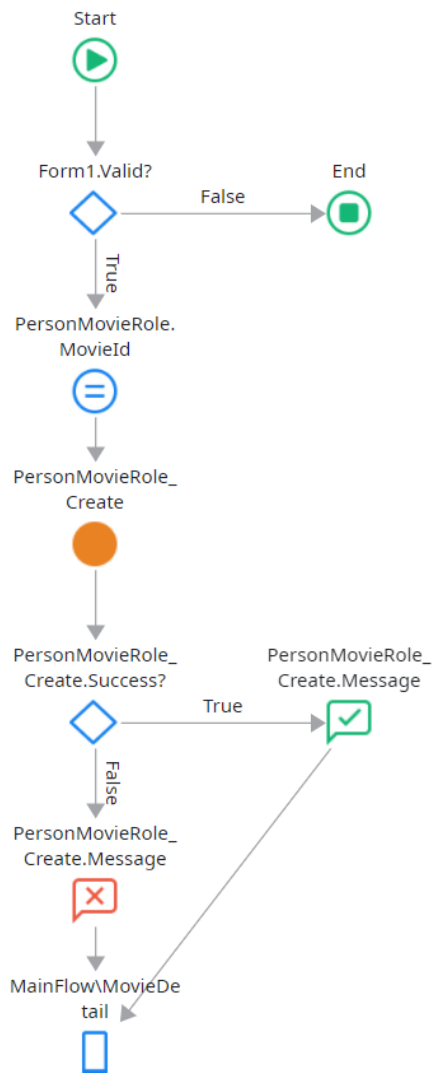


- e. Drag a **Message** and drop it to the right of the If and connect them. Connect the new Message to the MovieDetail Screen Destination.

- f. Set the **Message** property to be the output parameter of the PersonMovieRole\_Create Action and the **Type** to *Success*.



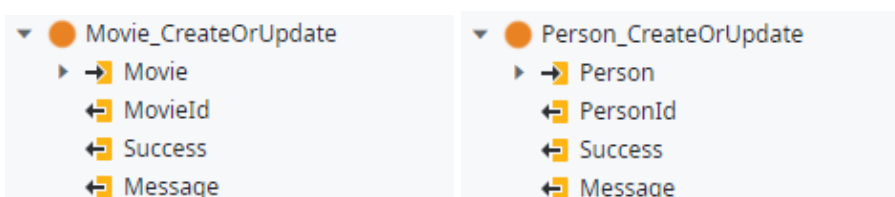
The flow should look like the following screenshot:



## Save Movie and Person Logic

To end the exercise, we need to apply the same logic that we did for adding the cast and crew in the logic to save movies and people in the database. You can do it at this point, right?

1. Repeat the exact same steps for the **Movie\_CreateOrUpdate** and the **Person\_CreateOrUpdate** Actions. Don't forget to use the Raise Exception and the Exception Handler flow in both Actions to define the logic for success and for when the information is not successfully saved in the database.



2. Also, repeat the same logic to the SaveOnClick Actions in the **MovieDetail** and **PersonDetail** Screens. The main difference that you need to worry about is deleting the success messages you already had in the logic, and replacing them with this new logic where the Messages are returned by the Server Actions.
3. Publish the app and open it in the browser.



This part of the logic is not easy to test, since you will not see the Save button if you're not an OSMDbAdmin already. However, think about how an attacker can manipulate the page and make the buttons visible. This way, you are protected with client-side and especially server-side validations, to guarantee that only people with the OSMDbAdmin role can save information in the database. This is something that you need to worry about in all your apps. Do not focus just on the UI!