

Security and Exceptions

Exercise

Outline	2
Hands-on	3
Users and Roles	3
Check Roles in the Screen	3
Disable Inputs for Non-Admins	4
Check Roles in the Logic	5

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:

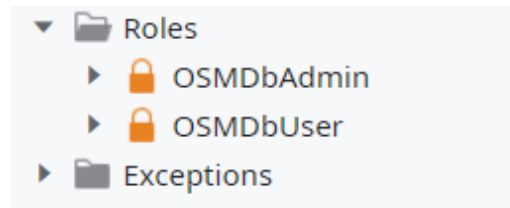
1. Create the OSMDbAdmin role.
2. Grant the role to a user.
3. Guarantee, on the logic side, that only OSMDbAdmin users can add a member of the cast / crew to a movie.
4. Use Exceptions to display an error message to the user if they do not have the correct permissions.
5. Restrict the access to the 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.

Hands-on

In this exercise, we will make our OSMDb app more secure by restricting many of the functionality exclusively to administrators. The main idea is that anyone can access information about movies and people, but only administrators can actually modify it, including the cast and crew of each movie.

Users and Roles

Let's start by defining a new Role for the app's administrators, the **OSMDbAdmin** Role. We also take the opportunity to rename the OSMDb role to **OSMDbUser**.



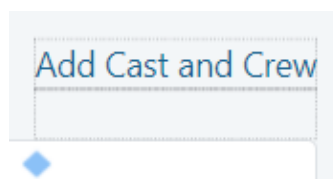
We need to grant the OSMdbAdmin Role to one of the Organization users, which can be done in the Users section in the **ODC Portal**. The **Manage Permissions** option allows defining the user's permissions by granting the OSMdbAdmin role to a user.

Check Roles in the Screen

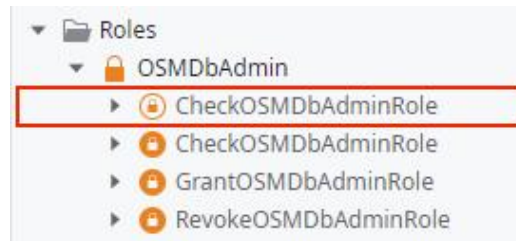
At this point, our app only allows users with the OSMdbAdmin Role to add cast / crew members to a particular movie. This is verified at the logic level, but we can take this one step further. Since a user that is not an administrator does not have permission to perform this operation, they don't need access to the Screen. So we will restrict access to the AddCastAndCrew Screen to users with the OSMdbAdmin role.

Note: In OutSystems, we can easily define which users can access a particular Screen using Roles. However, these Screens and their content exist on the client side, which makes them more vulnerable security-wise. This means that despite all the role restrictions we can create at the Screen level, it is a **best practice to always verify on the server side if a particular operation can be performed**. In this exercise, we will do that in the PersonMovieRole_Create Action by checking, on the server side, if the user has the OSMdbAdmin Role. Following the best practices, this verification is recommended even after restricting access to the Screen.

Now, in the MovieDetail Screen, we have a Link to the **AddCastAndCrew** Screen. This Link should only be visible to users with the OSMdbAdmin role. So, we need to hide it from the other users.



To accomplish this, we need to consider that the **CheckRole Server Action** cannot be used directly on the Screen. To solve this, we need to use the **CheckRole Client Action**.



We can check if the user logged-in has the OSMdbAdmin Role and return an output parameter as the result of this verification. Then, the output can be used to hide the Link when the user does not have the correct Role.

Disable Inputs for Non-Admins

To wrap up the exercise, we want to make sure that the movie and people information in the database can only be created / updated by users with the OSMdbAdmin role. This includes:

1. Form fields can only be edited by OSMdbAdmin users.
2. Save Buttons can only be visible by OSMdbAdmin users.

Hint 1: Each input field has an Enabled property that allows enabling the field only if a certain condition is evaluated as True.

The screenshot shows the configuration panel for an input field named 'Input_Title'. The 'Properties' tab is active, displaying various settings. The 'Enabled' property is set to 'True'.

Properties	
Name	Input_Title
Variable	GetMovieById.List.Curre
Prompt	
Input Type	Text
Max. Length	50
Mandatory	False
Enabled	True

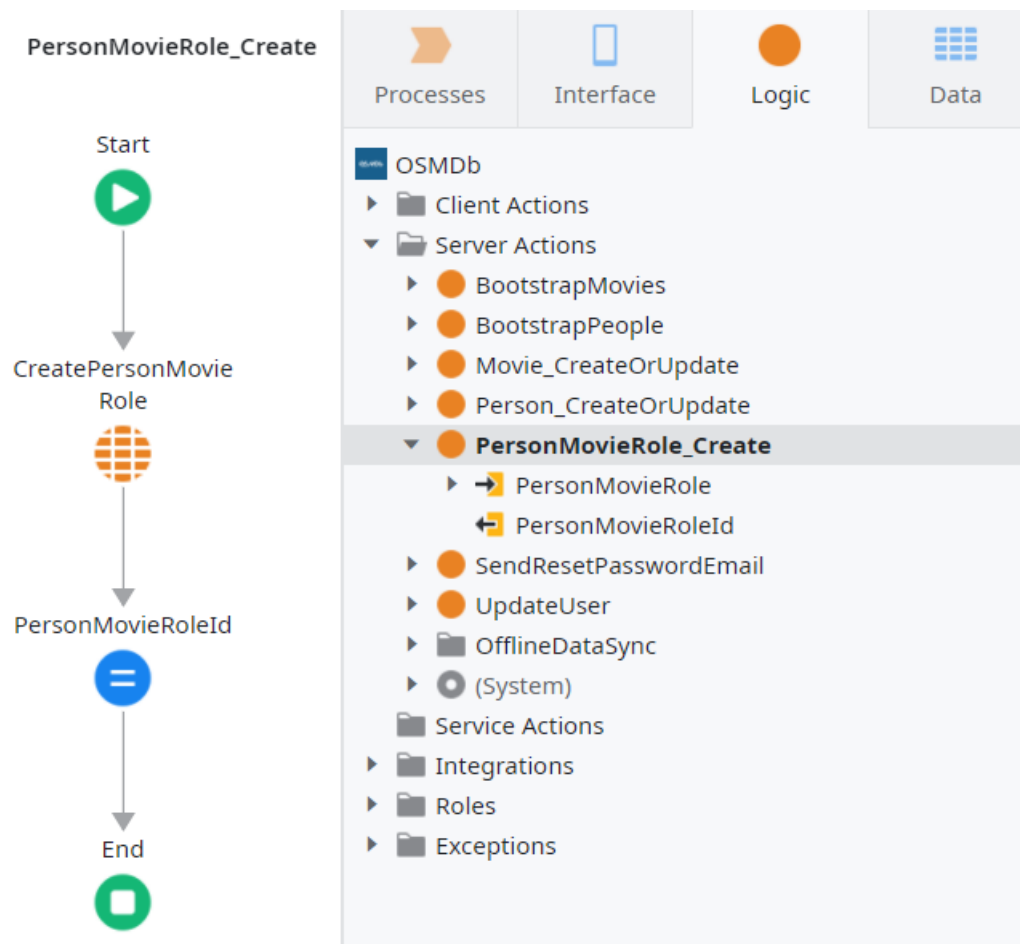
Below the properties, there is an 'Expression Editor...' section and a 'Suggestions' list with 'True' and 'False' options.

Hint 2: Don't forget to leverage the CheckRole Client Action.

Hint 3: Make sure to also apply the verifications to the logic, before adding / updating the movie / person to the database.

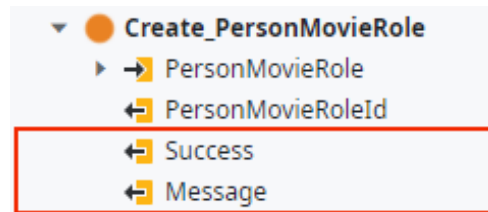
Check Roles in the Logic

We created the logic to add a member of the cast / crew to a movie in the AddCastAndCrew Screen by adding a new PersonMovieRole Entity record. To accomplish that, we created a Server Action.



Now, we want to ensure that the PersonMovieRole record is only created if the user has the OSMDbAdmin Role.

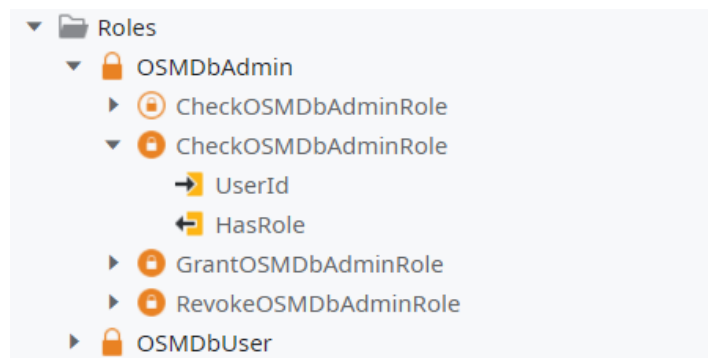
To complete the logic, the Server Action must return an output parameter showing if the operation was successful or not, and a message of success or error, depending on the scenario.



Hint: We will use exception handling to help us do that.

To do so, we'll have to adapt the logic to make sure the success code in the output of the Action is processed and an error / success message is displayed to the user depending on the value of the output.

We will use the built-in Actions created automatically with the Role and pass the result to the Server Action.



Hint: If a CheckRole Action is used without parameters, the Action will check if the user currently logged in has that specific Role.