Security and Exceptions - Exercise

Table of Contents

Outline

In this exercise, we will add some security features to the OSMDb application. So far, all the Screens can be accessed and operations can be completed with the Anonymous role, meaning that there are no security restrictions. At the end of this exercise, we want to make sure that only Admins can create/update information for movies and people, including adding a cast or crew member to a movie.

We want to do this in several steps:

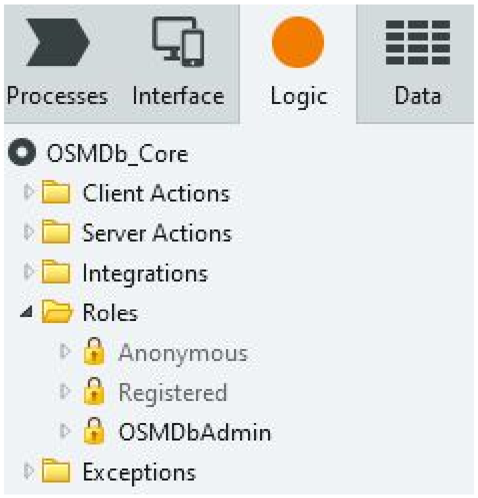
1. Create a OSMDbAdmin role.
2. Create some end-users and grant the role to one of them.
3. Guarantee in the logic to add a member of cast/crew to a movie that only OSMDbAdmin users can do that. Use Exceptions to display an error message to the user if they do not have the correct permissions.
4. Filter the access to the AddCastAndCrew Screen to OSMDbAdmin users.
5. Make sure that only Admin users can edit the movie and people's information.

Hands-on!

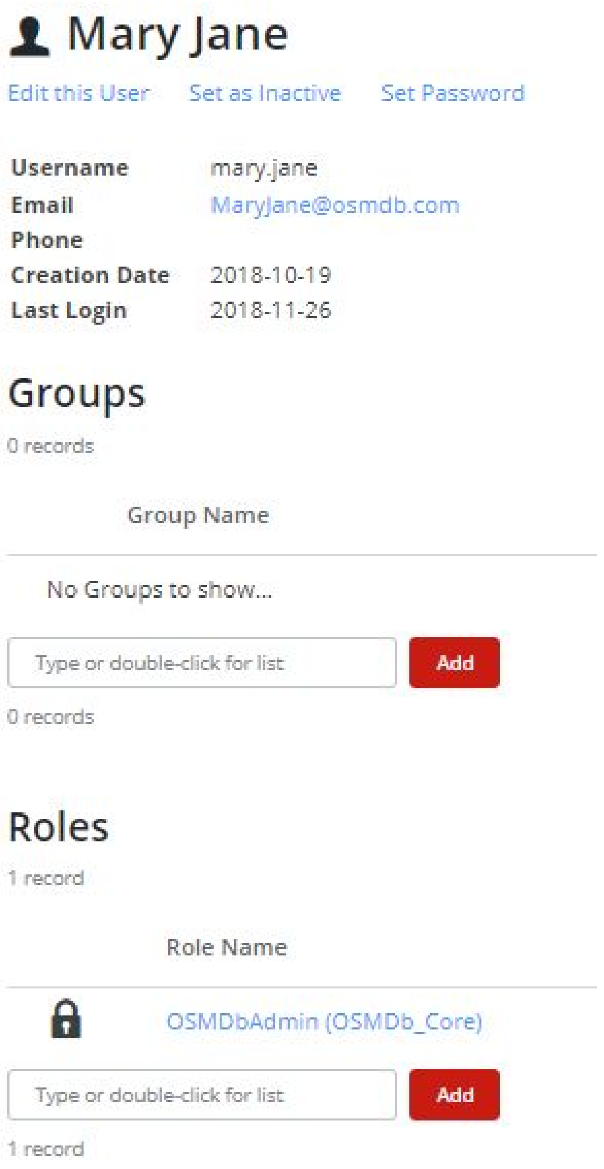
In this exercise, we will make our OSMDb application more secure, by restricting a lot of the functionality to administrators. Basically, the idea is that anyone can look for the information about movies and people, but only administrators can actually change it.

Users and Roles

Let's start by defining a new Role for the administrators of the application, the **OSMDbAdmin** Role, in the OSMDb_Core module. We need to publish the module after adding the Role.



Now, we need to add some end-users to our environment and grant the OSMDbAdmin Role to one of those users. A minimum of two users should be created, so at least one will not have any role associated with it. We can do that in the Users application (*https://<environment>/Users*).


Mary Jane

Username	mary.jane
Email	MaryJane@osmdb.com
Phone	
Creation Date	2018-10-19
Last Login	2018-11-26

Groups

0 records

Group Name
No Groups to show...

0 records

Roles

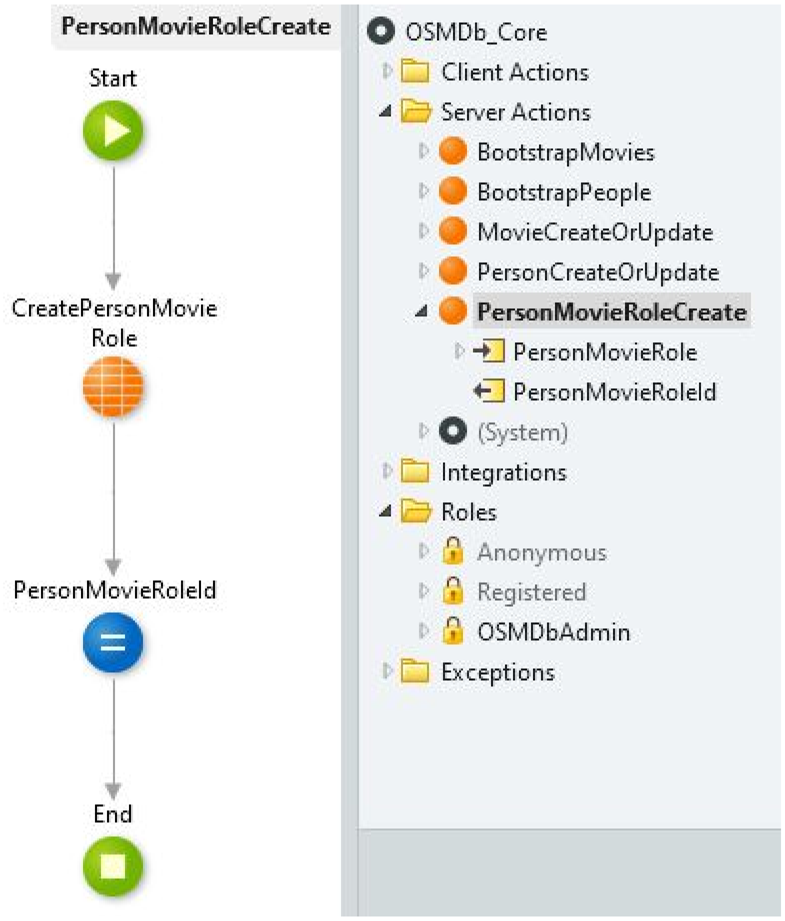
1 record

	Role Name
	OSMDbAdmin (OSMDb_Core)

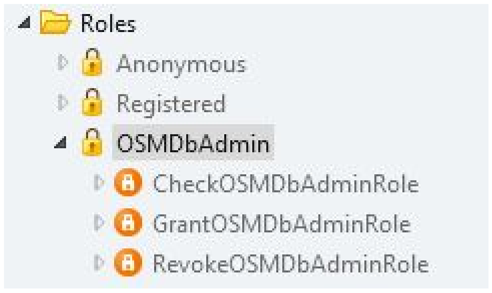
1 record

Check Roles in Logic

In the AddCastAndCrew Screen we created some logic to add a member of the cast/crew to a movie, by adding a new record of the PersonMovieRole Entity. To accomplish that, we created a Server Action in the Core module, since the Entities are exposed as read-only to the OSMDb module.



Now, we want to make sure that the PersonMovieRole record is only created if the user has the OSMDbAdmin Role. If we look at the role, it has three Server Actions created automatically, including the **CheckRole** Action, which verifies if a user has this particular role.



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

To complete the logic, we want the Action to return in an output parameter if the operation was successful or not and a message of success or error, depending on the scenario. To help us do that, we will use exception handling.

On the OSMDb module, 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.

NOTE: A few exercises ago we created the wrapper actions to create/update records in the database. That was done in the first place because the Entities were exposed as read-only. By creating these wrappers and making them public, we can actually filter which functionality we want to expose to the other modules. For instance, there is no way at this point to delete a movie or person from the OSMDb module.

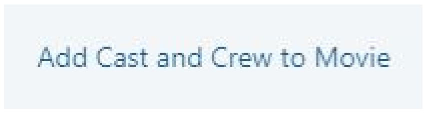
Here, we can see another very relevant use case of these wrappers, which is the chance of adding additional logic, besides a call to the Entity Action. Right now, when this Action is used in a module, it will always perform the verification for the OSMDbAdmin Role, guaranteeing that only users with the right access will be able to add the record to the database.

Restrict Access to Screens for Non-Admins

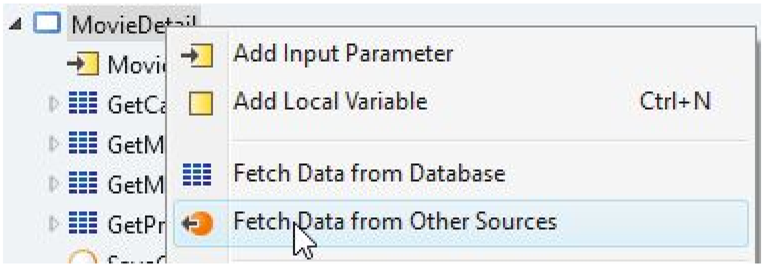
At this point, our application only allows users with the OSMDbAdmin Role to add a cast/crew member to a particular movie. This is verified at the level of the logic, but we can actually do more than that. If a user that is not an administrator cannot perform this operation, then we can only grant access to the AddCastAndCrew Screen to users with the OSMDbAdmin role.

NOTE: In OutSystems, we can easily define which users (using their Roles) can access a particular Screen. 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 do at Screen level, it is always a **best practice to verify, on server-side, if a particular operation can actually be performed**. In this exercise, we did that in the PersonMovieRoleCreate Action, by checking if the end-user has the OSMDbAdmin Role. This verification is still valid and recommended, despite the new Screen access restriction we just added in this section.

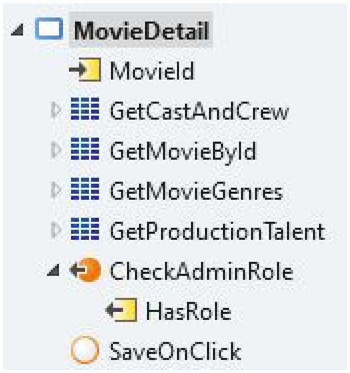
Now, in the MovieDetail Screen we have a Link to the **AddCastAndCrew** Screen. This Link should only be visible to users with the OSMDbAdmin role. So, we need to hide it from the other users.



To accomplish this, we need to consider that the **CheckRole** Action is a Server Action, thus we cannot use them directly on the Screen. To solve this, we need a Data Action.



These Actions run at the Screen level, in parallel with the Aggregates, and it is a Server Action. This means that we can check if the logged in user has the OSMDbAdmin Role and return as output parameter the result of this verification. We can then use that output to hide the Link when the user does not have the right Role.

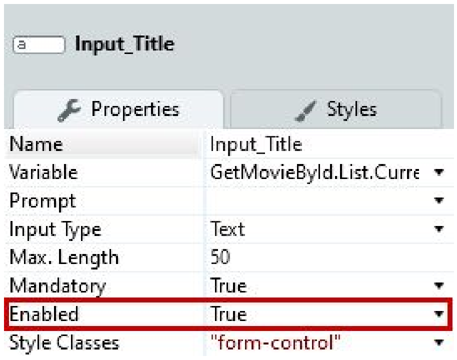


Disable Inputs for Non-Admins

To finalize 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 Enable property that allows to only enable the field if a certain condition is evaluated to True.



Input_Title

Properties | Styles

Name	Input_Title
Variable	GetMovieById.List.Curre
Prompt	
Input Type	Text
Max. Length	50
Mandatory	True
Enabled	True
Style Classes	"form-control"

Hint 2: Don’t forget to leverage the Data Actions if you need to use the CheckRole Action.

Hint 3: Don’t forget to also apply these verifications on the logic, before adding/updating the movie/person in the database.