Form Validations - Exercise

Table of Contents

Outline

In this exercise, we will focus on adding some business logic to our application, as we validate the forms that we have already created. This will be done in the MovieDetail and PersonDetail Screens to avoid saving invalid information in the database.

At the end of this exercise, we want to guarantee that the following rules are followed:

- A movie's gross takings can never be a negative number.
- A movie cannot have any gross takings amount, if it wasn't released yet.
- A person's birth date cannot be in the future.
- A person's date of birth cannot be later than the person's date of death.

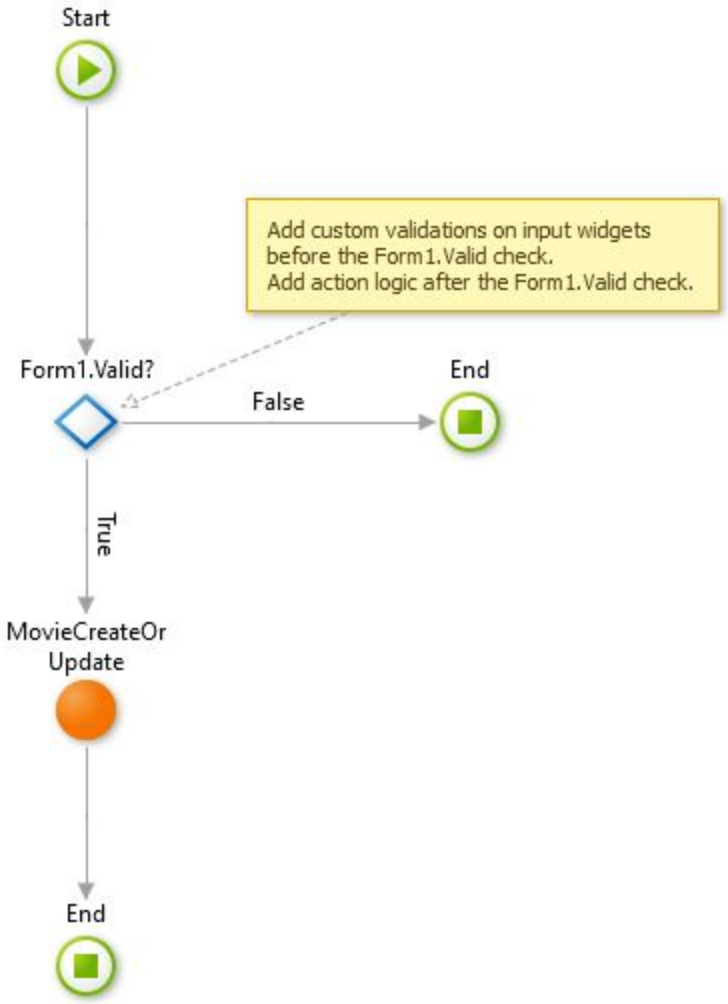
If a movie or person was successfully added to the database, then a feedback message should be displayed on the screen.

Hands-on

In this exercise, we will go back to the MovieDetail and PersonDetail Screens to implement these Form validations. As we know, our applications should limit to the most that invalid data is added to the database. OutSystems already helps with the built-in validations, but it is up to

the developers to guarantee that some other rules are followed as well. With that in mind, we will add some validations to our logic to guarantee that our rules are followed.

On both Screens, we already have a Client Action called SaveOnClick that is triggered when the Save button is clicked. Those Actions hold the logic to add a movie or a person to the database, depending on the Screen we are on. When opening one of those Actions, we can see that automatically the platform adds an If statement in the flow which checks to see if the Form is valid or not.



This guarantees that if the built-in validations (mandatory fields and wrong data types) fail, then the movie or person is not created / updated in the database.

Now, we want to add our own custom validations in both Screens.

MovieDetail Screen

The first thing we want to assure is that no movie in the database has a negative gross takings amount.

If this scenario occurs, then the movie cannot be created / updated in the database and the following message should appear to the end-user:

Gross takings amount cannot be negative.

Next, we want guarantee in the MovieDetail Screen that a movie cannot have a gross takings amount, if it wasn't released yet. In other words, movies with a release date in the future can not have a gross takings amount greater than zero. Just like in the previous scenario, if this happens the movie cannot be created / updated in the database and the following message should appear to the end-user:

The movie cannot have any gross takings since it was not yet released.

Hint: To check if a movie has a release date in the future, the built-in function *CurrDate()* can be helpful to give us the current date. Also the *Year()* function gives us the year of a given date.

When everything goes well, make sure that the user gets a feedback message indicating success.

PersonDetail Screen

In the PersonDetail Screen we also want to add two validations.

First, we want to make sure that a person's birth date cannot be in the future. When a user adds a birth date that does not meet this requirement, the person cannot be created / updated and the following message should appear:

Date of birth cannot be in the future.

Then, we need to guarantee that a person cannot have their date of death prior to their date of birth, meaning a person cannot have died if they have not been born yet. When a date breaking this validation rule is sent to the server to be saved in the database, the following message should appear to the user:

The date of death cannot be prior to the date of birth

Just like with the movies, when the validations are successful and the person is added to the database, a feedback message indicating success should be displayed to the user.

Don’t forget to test the validations with real data to make sure everything works!