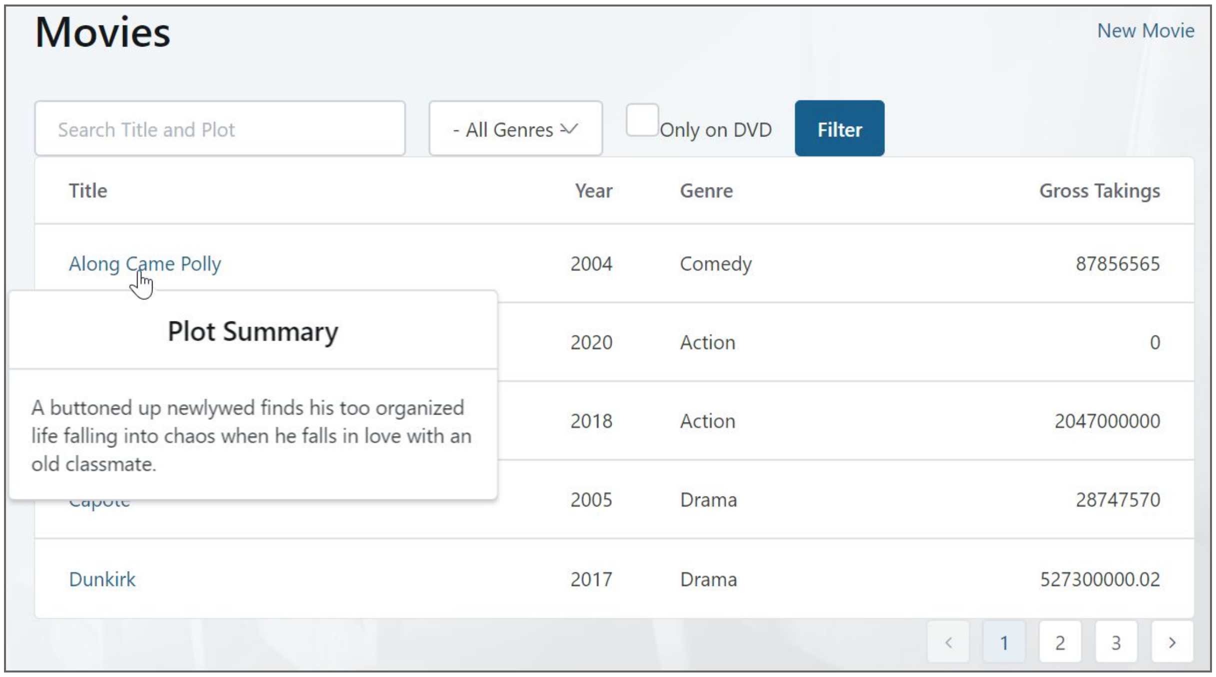
Advanced UI Patterns Exercise



Movies

New Movie

Title	Year	Genre	Gross Takings
Along Came Polly	2004	Comedy	87856565
	2020	Action	0
	2018	Action	2047000000
Capote	2005	Drama	28747570
Dunkirk	2017	Drama	527300000.02

Plot Summary

A buttoned up newlywed finds his too organized life falling into chaos when he falls in love with an old classmate.

Introduction

During this Lab, we will enhance some of the OSMDb application's Screens. In particular, we will work on the Movies Screen, where we will add a couple of new patterns to make it look better and more functional.

First, we will add pagination for the list of movies, to allow us navigating through the list of movies, displaying just a short number of movies per page. Then, we will allow users to dynamically sort the list of movies, using the List_SortColumn widget. The List_Navigation and the List_SortColumn are part of a module called RichWidgets, which has more advanced UI patterns that can be used. This module is already referenced in all UI modules, so the elements will be ready to be used.

Then, in the Movies Screen, we will add a Balloon to display the Plot Summary of a movie, instead of having that column in the Table Records. We will also add the User Initials to be displayed next to the user name in the login section. These are just two of dozens of the patterns available in the OutSystems UI Framework. These patterns are components that can be dragged and dropped to any Screen that implement common UI patterns and functionalities, and are also available in the Widget toolbar.s

In this specific exercise lab, you will:

- Add Dynamic Sorting and Pagination to the Movies Screen
- Add a Balloon and a User Initials pattern to the application

Table of Contents

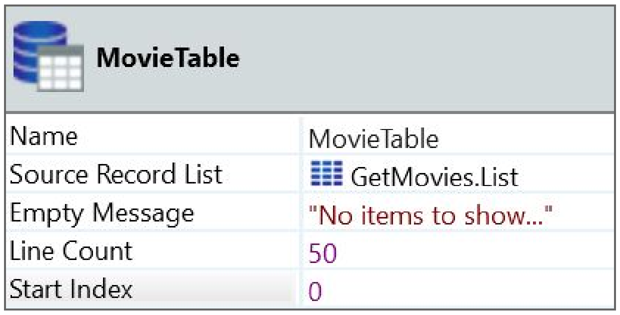
Add RichWidgets patterns to the application

In this exercise lab, we will start by using two elements from the RichWidgets module: List_Navigation, to allow adding pagination and for navigating through those pages in the Table Records that lists all movies, and List_SortColumn, to add a dynamic sorting behavior to the same list of movies. These changes will be done in the Movies Screen.

The RichWidgets module is part of the OutSystems System Components and it is already available in our environments. In particular, these two elements are also referenced by default in any Web Responsive module.

Use the List Navigation in the Movies Screen

At the current state of our application, the MovieTable Table Records in the Movies Screen displays the movies in the database, up to *50*. This number 50 is the default **LineCount** of a TableRecords, meaning that it shows up to 50 movies in the Table Records.



MovieTable

Name	MovieTable
Source Record List	GetMovies.List
Empty Message	"No items to show..."
Line Count	50
Start Index	0

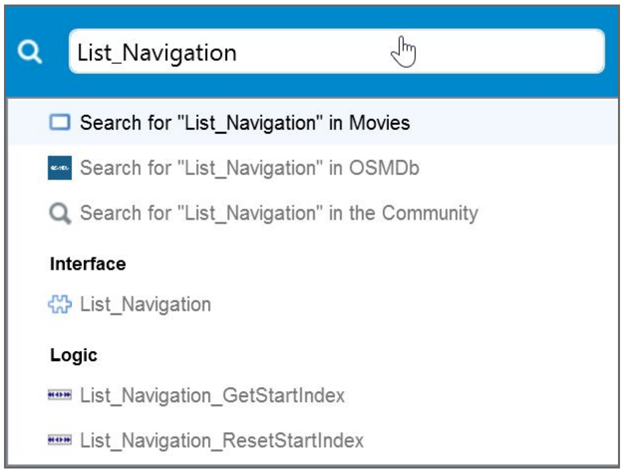
However, with this we have two problems. First, if we have more than 50 movies in the database, only the first 50 would be displayed. Second, even if we change the number to something else bigger, having a static number is not a good idea, because we may not know at design time how many movies are there. So, we will add a **List_Navigation** widget to the Movies Screen, to add pagination and allow that all movies are displayed in different pages of the Table Records. The end-user can then select the page that it wants to see.

1) Add a **List_Navigation** Widget to the **Movies** Screen, and link it to the MovieTable Table Records. Call the widget *Navigation.*

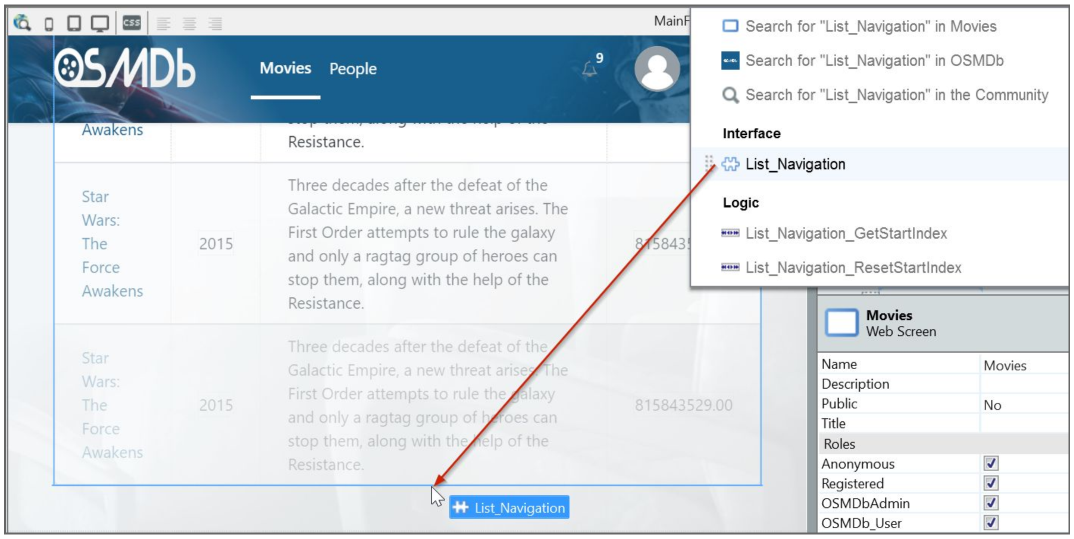
    a) Open the **Movies** Screen.

    b) Click on the magnifying glass on the top right corner, to expand the Search Box.

    c) Search for *List_Navigation*. You will see all the elements in the module with this substring in the name.

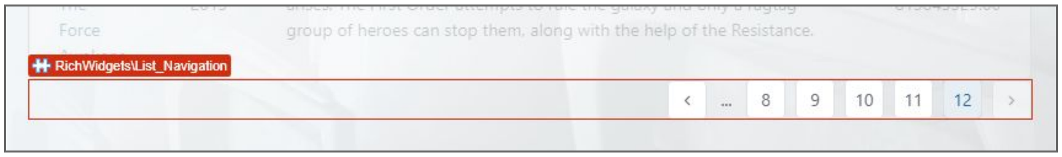

List_Navigation
Search for "List_Navigation" in Movies
Search for "List_Navigation" in OSMDb
Search for "List_Navigation" in the Community
Interface
List_Navigation
Logic
List_Navigation_GetStartIndex
List_Navigation_ResetStartIndex

d) Select the **List_Navigation** element. Drag and drop it, below the Table Records in the Movies Screen.


Movies
People
Search for "List_Navigation" in Movies
Search for "List_Navigation" in OSMDb
Search for "List_Navigation" in the Community
Interface
List_Navigation
Logic
List_Navigation_GetStartIndex
List_Navigation_ResetStartIndex
Movies
Web Screen
List_Navigation

Name	Movies
Description	
Public	No
Title	
Roles	
Anonymous	✓
Registered	✓
OSMDbAdmin	✓
OSMDb_User	✓

e) After dropping the element, you should get something like this:

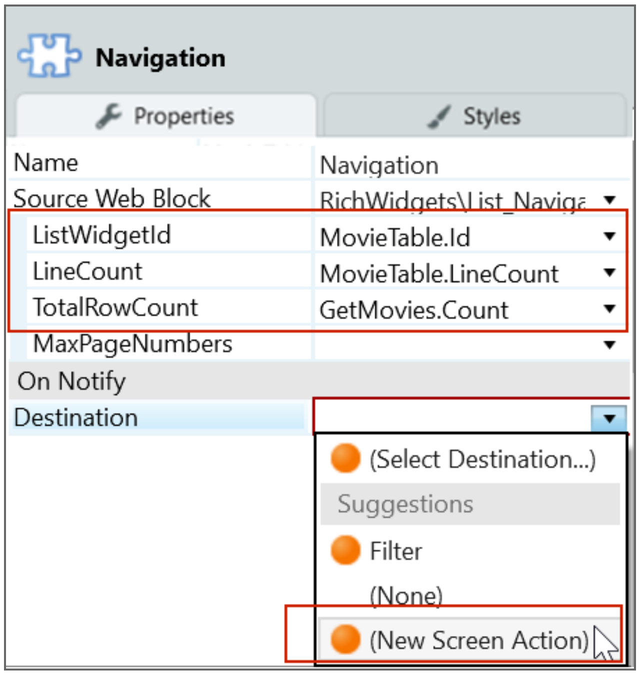


f) Change the **Name** of the List_Navigation Widget to *Navigation*.

g) Set the **ListWidgetId** to the Id of the Table Records, *MovieTable.Id.* This binds the List_Navigation to the TableRecords.

h) Set the **LineCount** to the Line Count field of the Table Records, *MovieTable.LineCount.* This indicates how many movies will be displayed per page. At this point, that number would be *50*.

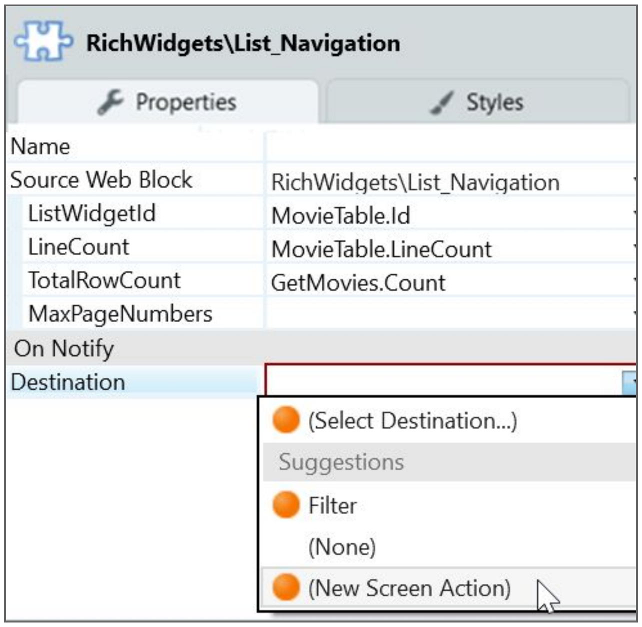
i) Set the **TotalRowCount** to the number of elements returned by the **GetMovies** Aggregate: *GetMovies.Count.* Remember that the GetMovies Aggregate is the **Source Record List** to the Table Records. This will get how many movies are in the database. With this and the LineCount, the List_Navigation determines how many pages are needed to consider all the movies



Name	Navigation
Source Web Block	RichWidgets\List_Naviga
ListWidgetId	MovieTable.Id
LineCount	MovieTable.LineCount
TotalRowCount	GetMovies.Count
MaxPageNumbers	
On Notify	
Destination	

2) The **List_Navigation** Widget requires an *OnNotify* Action to respond to the user changing navigation pages, while using the application. This behavior is similar to the ones in Web Blocks, where the parent needs to respond to some changes made in the Block. In this case, it is the end-user selecting a new page on the navigation. Typically, the Action requires refreshing the Aggregate / SQL Query and the Widgets (Table / List and Navigation).

a) Set the **OnNotify** to a *(New Screen Action)*.



RichWidgets\List_Navigation

Name	
Source Web Block	RichWidgets\List_Navigation
ListWidgetId	MovieTable.Id
LineCount	MovieTable.LineCount
TotalRowCount	GetMovies.Count
MaxPageNumbers	
On Notify	
Destination	

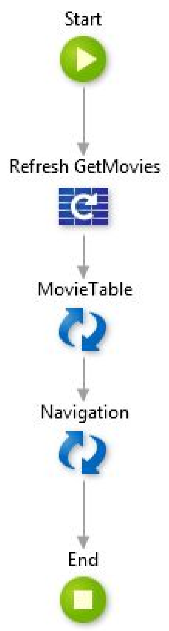
b) Change the Action's name to *OnNotify*.

c) Drag and drop a **Refresh Data** statement to the Action flow and select the **GetMovies** Aggregate in the **Select Data Source** window.

d) Drag and drop an **Ajax Refresh** statement below the **Refresh Get Movies** and select the **MovieTable** in the **Select Widget** window.

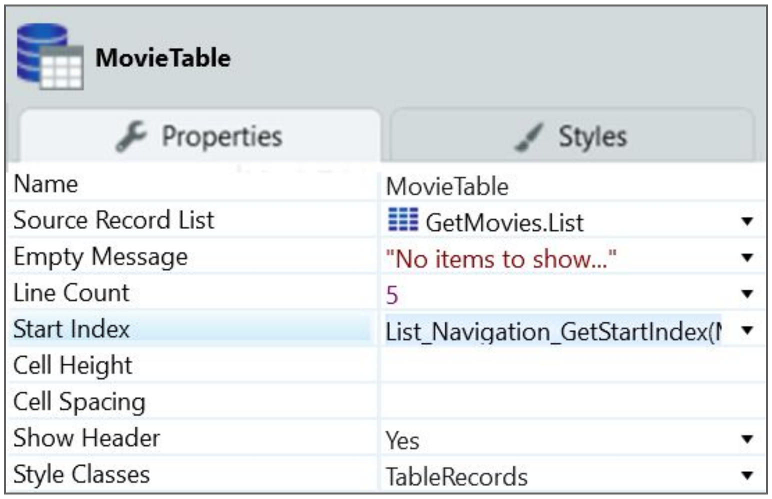
e) Add another **Ajax Refresh** statement and select the **Navigation** Widget.

f) The **OnNotify** Action should look like this



3) To reflect the user changing pages using the List_Navigation, the **Start Index** of the Table Records needs to be set using the **List_Navigation_GetStartIndex** Action. This Action already exists in the RichWidgets and dynamically sets the Start Index of the Table Records. This guarantees that the proper records are displayed, considering the page selected in the List_Navigation by the end-user.

   Set the **Start Index** of the MovieTable to *List_Navigation_GetStartIndex(MovieTable.Id).* Also, change the **LineCount** of the Table Records to *5*, to display 5 records per page.



Name	MovieTable
Source Record List	GetMovies.List
Empty Message	"No items to show..."
Line Count	5
Start Index	List_Navigation_GetStartIndex(M
Cell Height	
Cell Spacing	
Show Header	Yes
Style Classes	TableRecords

NOTE: If the number of records in the Table Records is less than the **Line Count**, the List_Navigation will not appear. Changing the **Line Count** property to a small number allows to test the Navigation Widget, without having to enter more data to the application.

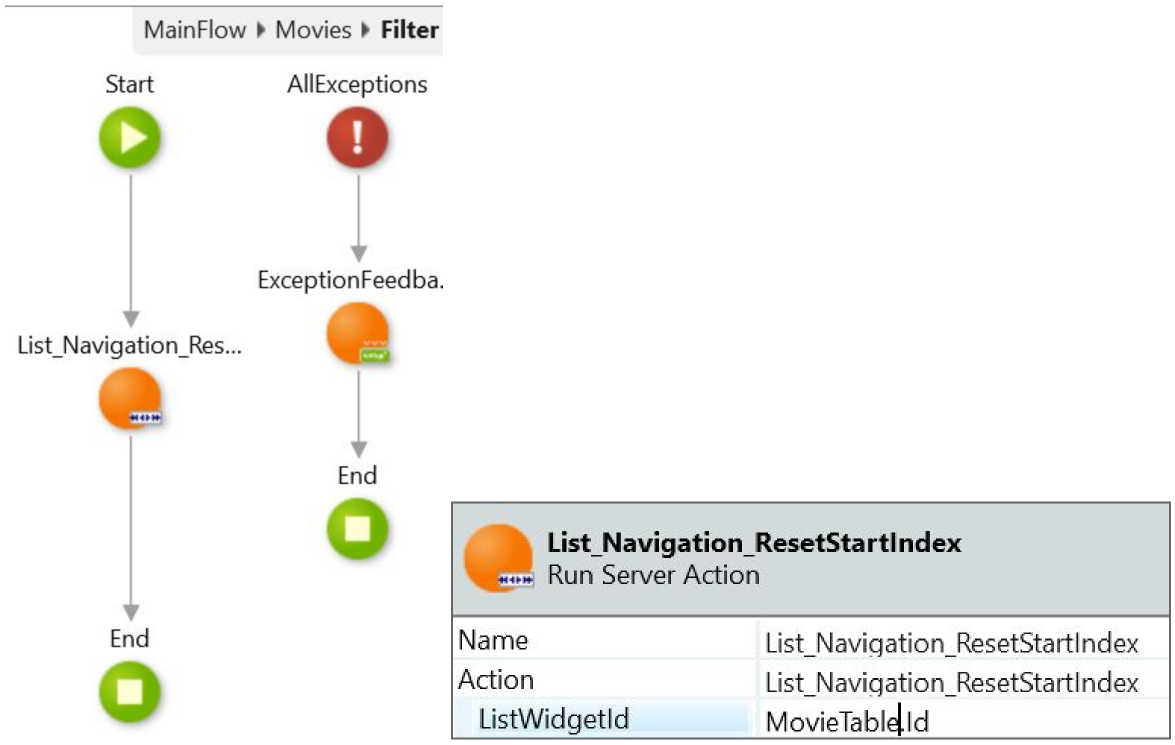
4) Publish and test the changes made to the OSMDb application.

   a) Click the **1-Click Publish** button to publish the application, and access it.

   b) Navigate to the **Movies** Screen.

   c) Determine if the List_Navigation is working properly. Select the second page and then apply the search filter by searching for *Capote*. You will notice that no movies are displayed. This happens because we are in the second page of the navigation, but now only one movie is displayed. Click on page *1* and make sure that the movie appears.

5) Reset the Navigation when applying the search filter, to make sure it always starts from the first page. Use the *List_Navigation_ResetStartIndex* Action in the **Filter** Screen Action of the Movies Screen to reset it. Make sure that the application works properly after publishing the changes.



List_Navigation_ResetStartIndex
Run Server Action

Name	List_Navigation_ResetStartIndex
Action	List_Navigation_ResetStartIndex
ListWidgetId	MovieTable.Id

Add Dynamic Sorting in the Movies Screen

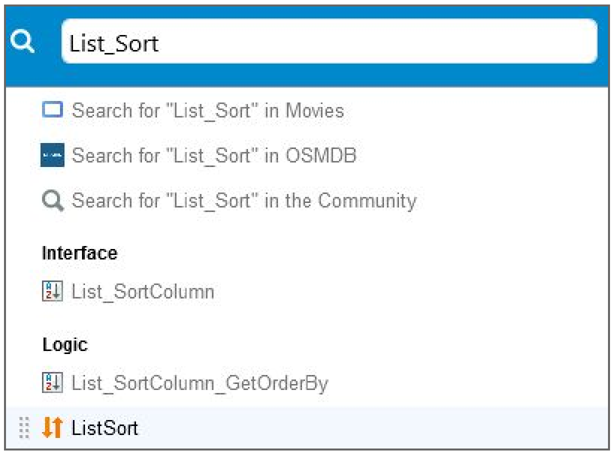
In the Movies Screen, we want to allow the end-users to sort the data in the table at runtime, by clicking on the respective column title in the Table Records. This requires a dynamic sorting in the Aggregate that fetches the movies, since only the end-user will be able to choose the sorting at runtime. To do that, we will use the List_SortColumn element from the RichWidgets, and some other Actions to help us.

1) Add a **List_SortColumn** to the Table Records in the **Movies** Screen, to enable sorting by movie title, year and gross takings.

   a) Go back to the **Movies** Screen.

   b) Click on the magnifying glass, to expand the Search box.

   c) Search for *List_Sort*. You will see all the elements in the module with this substring in the name.



List_Sort

Search for "List_Sort" in Movies

Search for "List_Sort" in OSMDB

Search for "List_Sort" in the Community

Interface

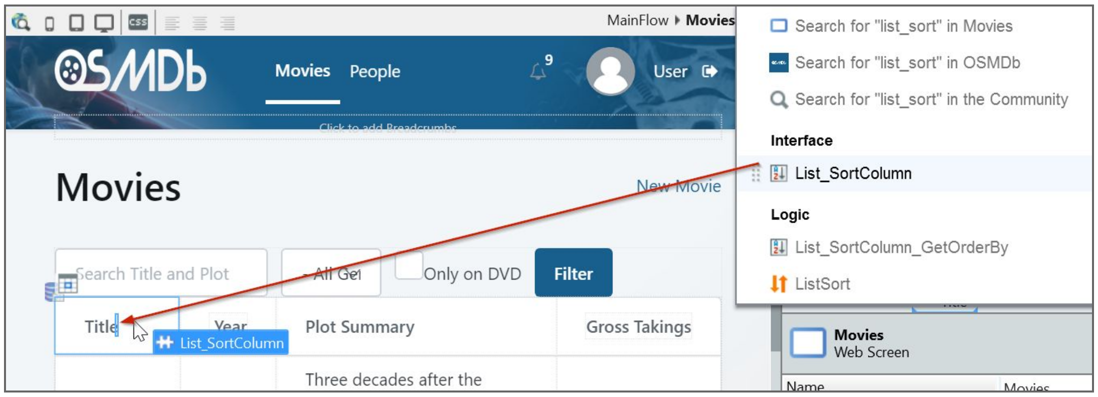
List_SortColumn

Logic

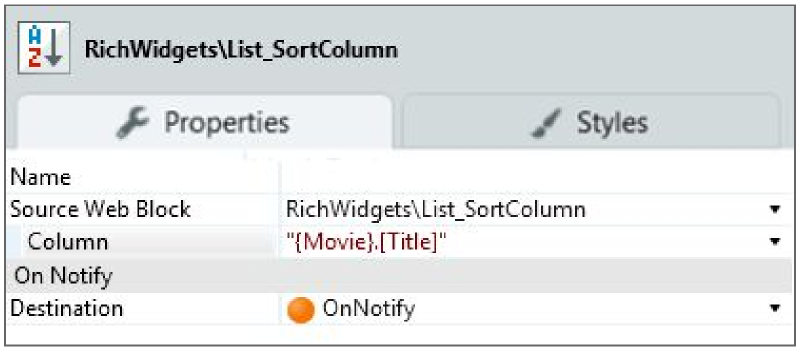
List_SortColumn_GetOrderBy

ListSort

   d) Select the **List_SortColumn** element. Drag and drop it in the Header row, inside the **Title** column.



e) Select the dragged List_SortColumn and fill the missing properties. Set the **Column** to *"{Movie}.[Title]"* and the **Destination** to the *OnNotify* Action.



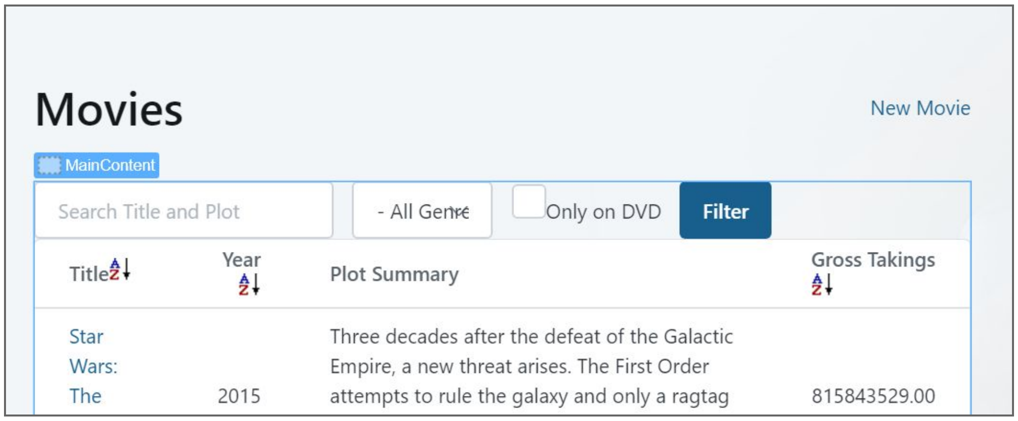
Name	
Source Web Block	RichWidgets\List_SortColumn
Column	"{Movie}.[Title]"
On Notify	
Destination	OnNotify

f) Repeat the same process for the columns **Year** and **Gross Takings**. In this case, the **Column** must be filled with *"{Movie}.[Year]"* and *"{Movie}.[GrossTakingsAmount]"* respectively. The **On Notify Destination** to be the *OnNotify* Action.

NOTE: The **Column** property is filled with the Entity's name and the Attribute's name, respective to the column being sorted, using the syntax **"{EntityName}.[AttributeName]"**. You should follow this syntax, even if the name of the column in the Table Records is different.

---

g) The final look of the Screen in the canvas should be similar to



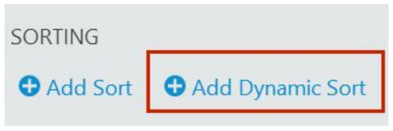
Title	Year	Plot Summary	Gross Takings
Star Wars: The	2015	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag	815843529.00

2) To make sure that the sorting works, it is not enough to add these List_SortColumn elements to the MovieTable Table Records. It is also necessary to sort the Aggregate that is the source of the Table Records, to make sure that the movies are returned in the chosen order. For that purpose, we need to add a dynamic sorting to the **GetMovies** Aggregate, using the Action **List_SortColumn_GetOrderBy**. Make the Action get the

sorting from the MovieTable Table Records and make sure that by default the sorting is done by Title.

a) Open the **GetMovies** Aggregate in the **Movies** Screen Preparation.

b) In the **Sorting** tab, select the option *Add Dynamic Sort*.



c) Add the following expression and click **Done**:
List_SortColumn_GetOrderBy(MovieTable.Id, "{Movie}.[Title]")

This gets the sort to be made from the MovieTable, depending on the choice made by the end-user. If no option is made, the default sorting will be done by movie title.

d) Click the **1-Click Publish** button to publish the application, and check that the sorting works.

NOTE: The expressions that fill the **Column** property of the **List_SortColumn** are stored in **Session Variables**. If you make a mistake in the syntax, you will get an error, but, fixing the problem in Service Studio is not enough. You need to clear the browser cookies or logout and login the Session, so that the Variables are reset and set to the new (and correct) expressions.

Add OutSystems UI patterns to the application

In this part of the exercise, we will add some other patterns to the Movies Screen. These patterns are part of the OutSystems UI (https://www.outsystems.com/outsystems-ui/) Framework. These UI Patterns are available for web and mobile and we will use two in this lab: the User Initials and the Balloon.

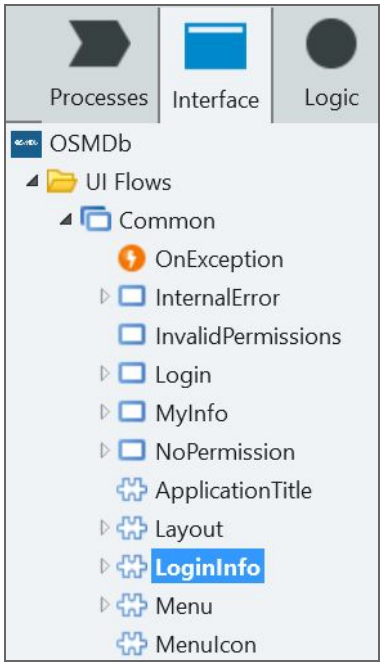
OutSystems UI Web is available on the Forge, but it is already part of the installation of the platform, meaning that it is already referenced in any new Web Application you may create, including our OSMDb. Apart from that, the UI Patterns are also available in Service Studio, in the toolbox on the left, next to the other widgets that we already know. It is just a matter of dragging and dropping them on the Screen, and set their properties to make them work fine.

Add the User Initials to the Login part of the application

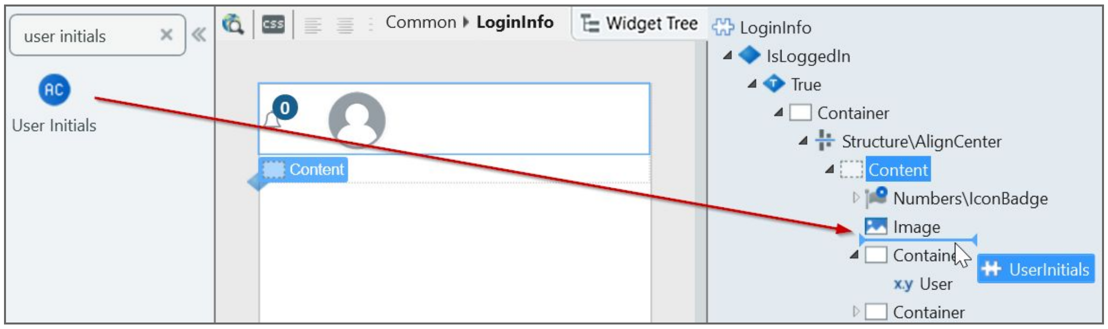
This first OutSystems UI pattern we are going to use is the User Initials. Basically, this patterns gets a name as input and displays the user initials in a nicely manner. For instance, for our user Mary Jane, the user initials would be M J.

1) Create a User Initials in the **LoginInfo** Block, under the **Common** UI Flow, before the Username.

    a) Expand the **Common** UI Flow under the Interface tab, and open the **LoginInfo** Web Block.



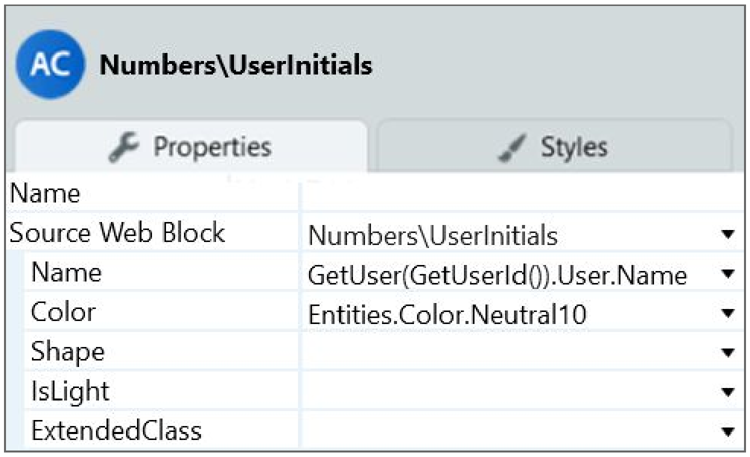
b) Drag and drop a User Initials to the Block, before the Container with the username.



c) Set the **Name** property, under the Source Web Block, of the User Initials to get the name of the user currently logged in: *GetUser(GetUserId()).User.Name*

This Expression is using the **GetUser** Entity Action, which receives the user id of the user currently logged in, and then get its name.

d) Set the **Color** property to the one you prefer. In this example, we will set it to Black (*Neutral10*).

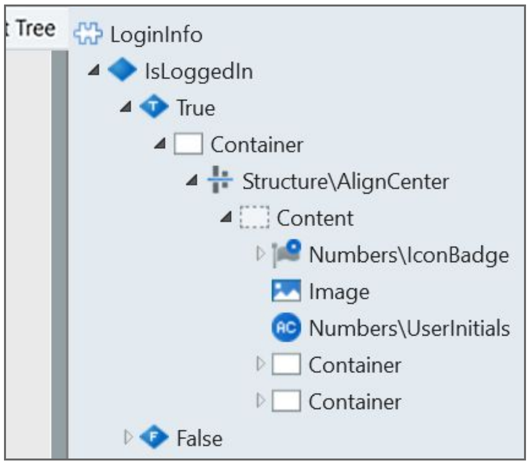


Numbers\UserInitials

Name	
Source Web Block	Numbers\UserInitials
Name	GetUser(GetUserId()).User.Name
Color	Entities.Color.Neutral10
Shape	
IsLight	
ExtendedClass	

2) Delete the other elements on the LoginInfo Block, to make sure that only the initials and user name appear.

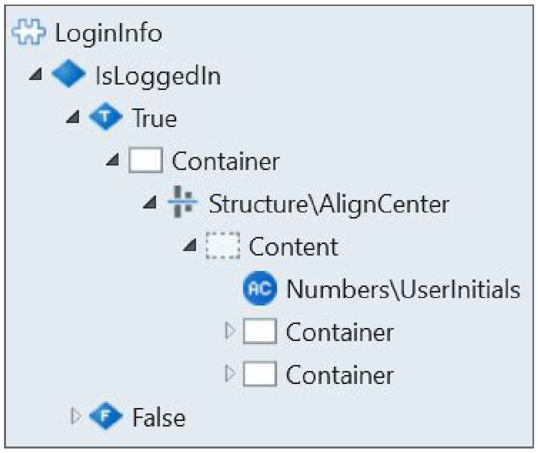
a) In the LoginInfo Block, open the widget tree.



```
Tree  LoginInfo
        IsLoggedIn
          True
            Container
              Structure\AlignCenter
                Content
                  Numbers\IconBadge
                  Image
                  Numbers\UserInitials
                  Container
                  Container
          False
```

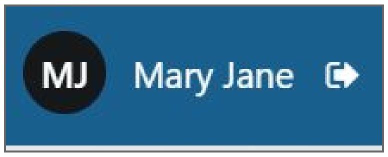
b) Delete the IconBadge and the Image before the User Initials.

c) The Web Block should look like the following screenshot



```
LoginInfo
  IsLoggedIn
    True
      Container
        Structure\AlignCenter
          Content
            Numbers\UserInitials
            Container
            Container
    False
```

d) Publish the module and make sure that the new login look and feel appears correctly.



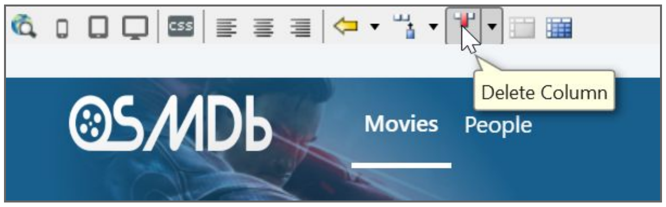
Add a Balloon to the MovieTable in the Movies Screen

To end the exercise lab, we will add a Balloon pattern to the MovieTable Table Records, to display the plot summary of the movie, when the movie title is hovered by the mouse. To make the application consistent, we will remove the plot summary from the Table Records, and will add the movie genre instead.

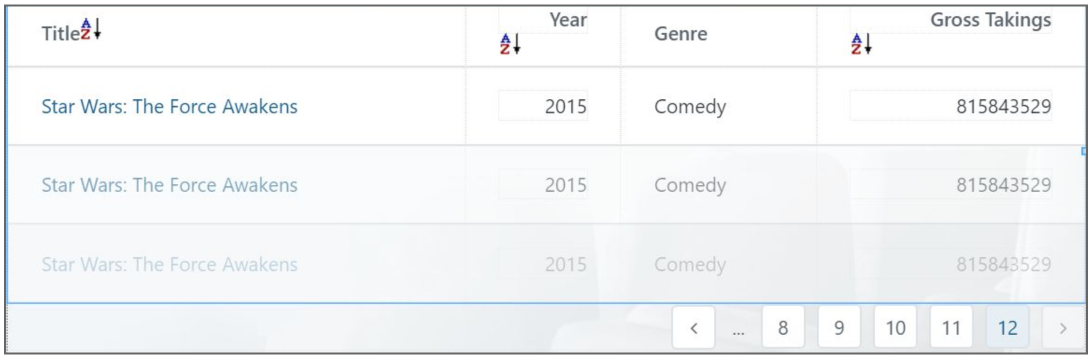
1) Let's start by changing the Table Records, to include the movie genre, instead of the plot summary of the movies.

   a) Open the **Movies** Screen, and select the Plot Summary column in the Table Records.

   b) Delete the column by selecting the **Delete Column** option on the top of the canvas. Don't forget that these options only appear when selecting the Table Records.



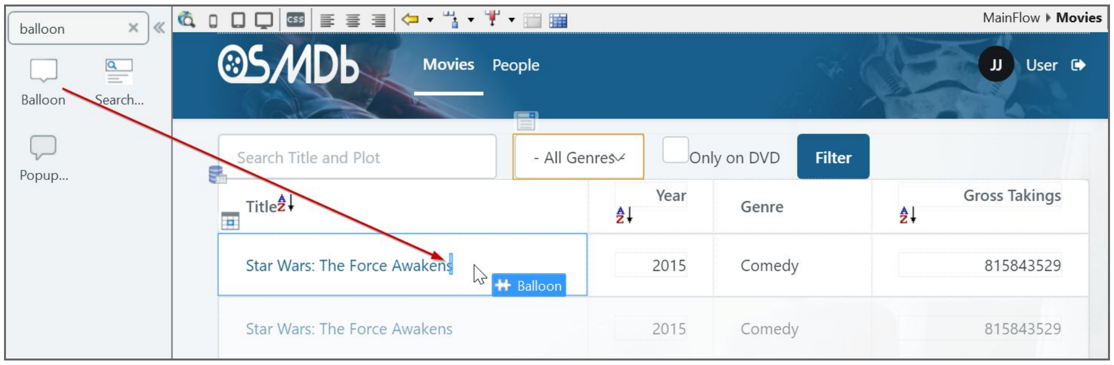
   c) Switch to the Data tab in Service Studio, expand the **Movie** Entity (under the OSMDb_Core module), drag the **GenreId** attribute and drop it between the Year and the Gross Takings on the TableRecords.



Title	Year	Genre	Gross Takings
Star Wars: The Force Awakens	2015	Comedy	815843529
Star Wars: The Force Awakens	2015	Comedy	815843529
Star Wars: The Force Awakens	2015	Comedy	815843529

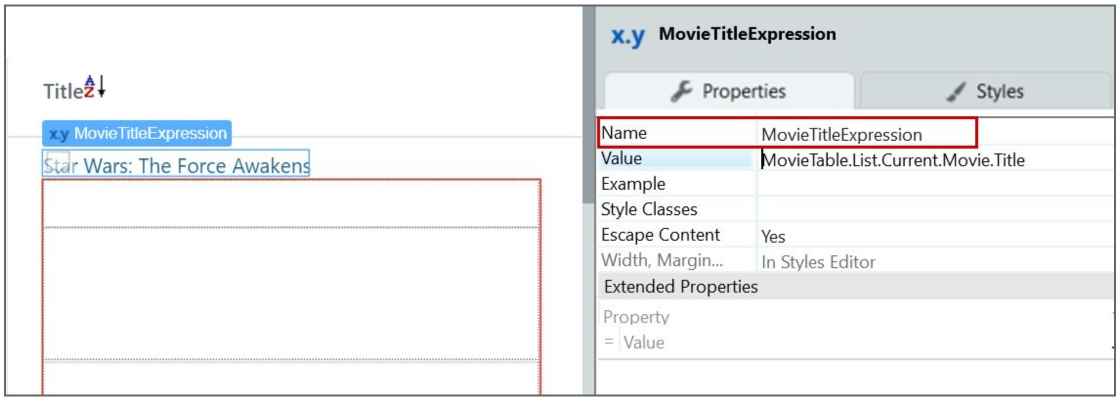
2) Now we will add a Balloon next to the movie title in the Table Records and make sure it will display the Plot Summary of the movie.

   a) Search for balloon in the Widget Toolbar and drag and drop the Balloon pattern next to the movie title, in the Table Records.



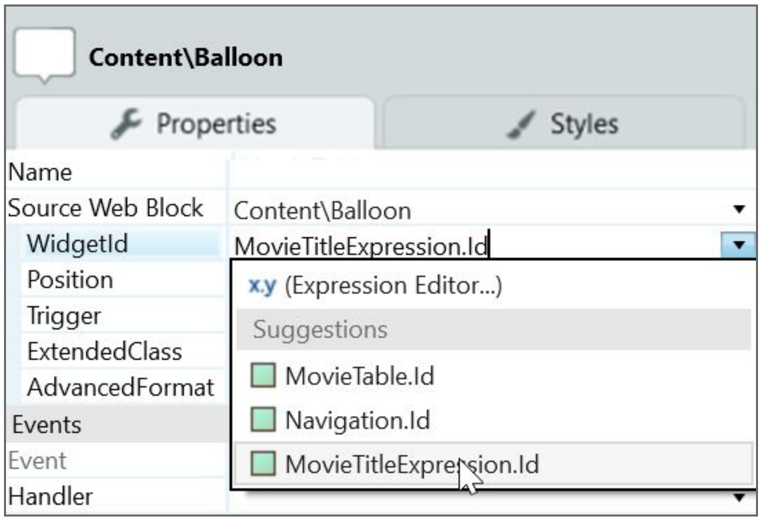
Title	Year	Genre	Gross Takings
Star Wars: The Force Awakens	2015	Comedy	815843529
Star Wars: The Force Awakens	2015	Comedy	815843529

b) The **Widget Id** property of the Balloon is asking for the widget to which the pattern will be associated with. That widget is the Expression that displays the movie title, so it must have a Name to be referenced. Set the **Name** of the Expression to *MovieTitleExpression*. Note that here we are naming the Expression with the Title, not the Balloon.



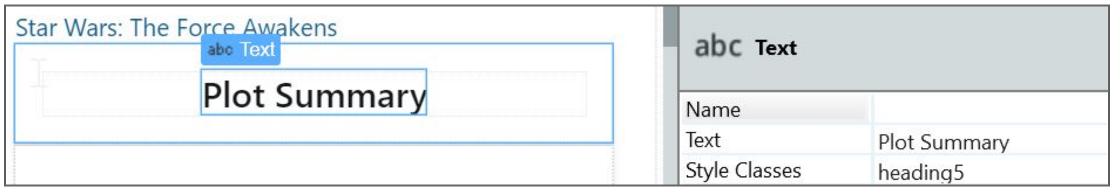
Name	MovieTitleExpression
Value	MovieTable.List.Current.Movie.Title
Example	
Style Classes	
Escape Content	Yes
Width, Margin...	In Styles Editor
Extended Properties	
Property	
= Value	

c) Set the **WidgetId** property of the Balloon to *MovieTitleExpression.Id*



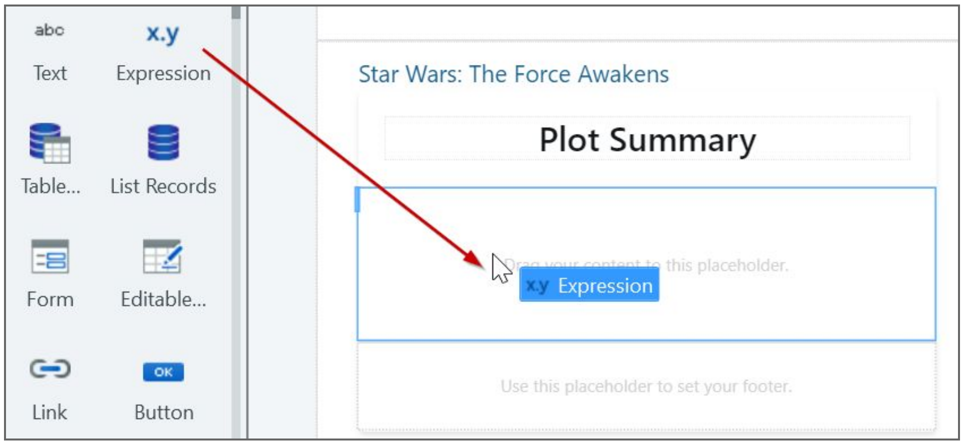
Name	
Source Web Block	Content\Balloon
WidgetId	MovieTitleExpression.Id
Position	
Trigger	
ExtendedClass	
AdvancedFormat	
Events	
Event	
Handler	

d) Inside the Balloon, type *Plot Summary* in the **Title placeholder**. Give it a *heading5* Style Class and center it.

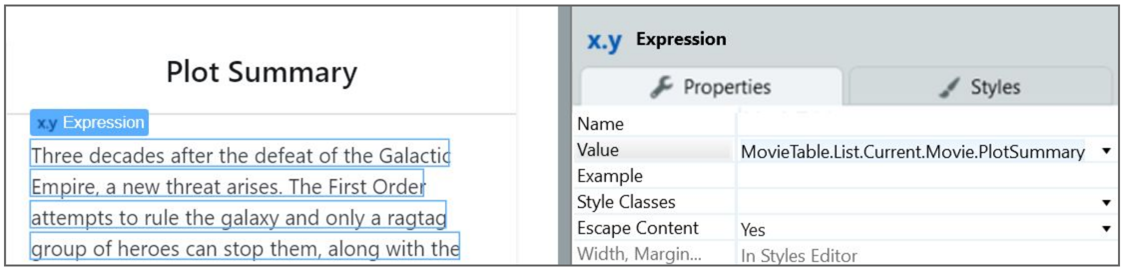


Name	
Text	Plot Summary
Style Classes	heading5

e) Drag an Expression to the **Content** placeholder of the Balloon.


Text
Expression
Table...
List Records
Form
Editable...
Link
Button
Star Wars: The Force Awakens
Plot Summary
Drag your content to this placeholder.
x.y Expression
Use this placeholder to set your footer.

f) Set the Expression’s value to *MovieTable.List.Current.Movie.PlotSummary*

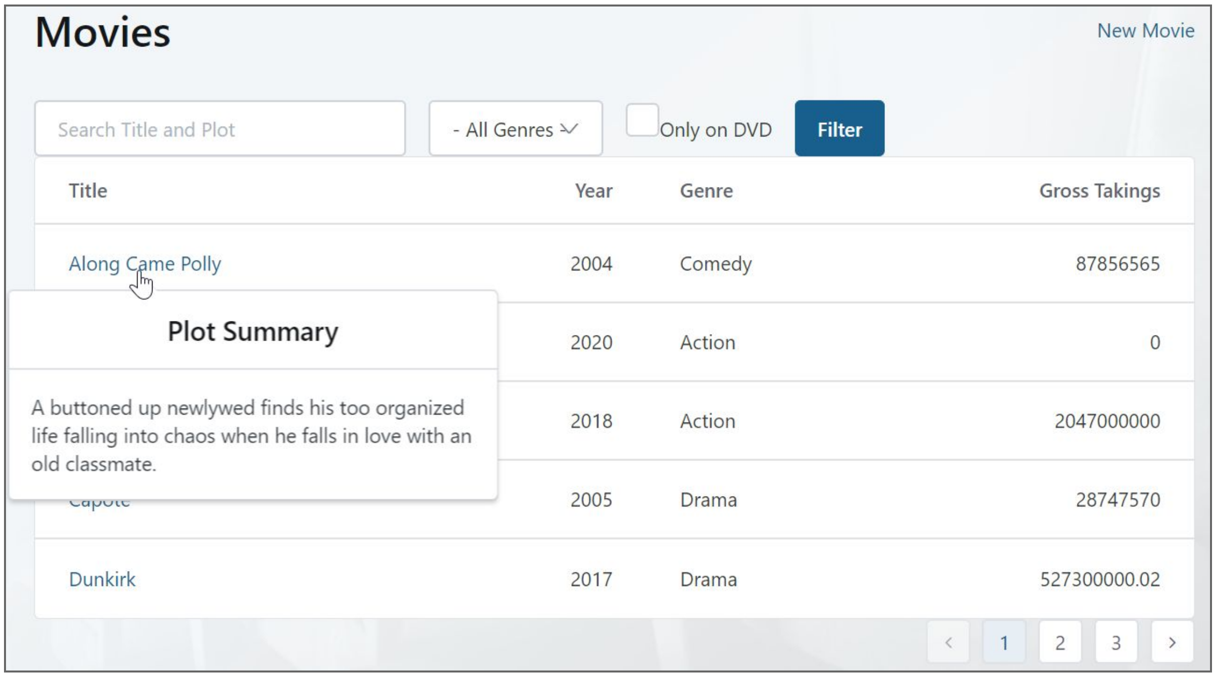


Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a ragtag group of heroes can stop them, along with the

Name	
Value	MovieTable.List.Current.Movie.PlotSummary
Example	
Style Classes	
Escape Content	Yes
Width, Margin...	In Styles Editor

This gets the Plot Summary of the movie, just like we did before in the Table Records directly. But now, it will appear in the Balloon.

3) Publish the module and test the application in the browser. Make sure that the Balloon appears for all movies, with the respective Plot Summary. Also note that some movies may not have genres, since we didn’t add it to them yet. Feel free to do that to make the Movie Table look more complete.



Movies

New Movie

Search Title and Plot | - All Genres | Only on DVD | Filter

Title	Year	Genre	Gross Takings
Along Came Polly	2004	Comedy	87856565
	2020	Action	0
	2018	Action	2047000000
Capote	2005	Drama	28747570
Dunkirk	2017	Drama	527300000.02

Plot Summary

A buttoned up newlywed finds his too organized life falling into chaos when he falls in love with an old classmate.

End of Lab

During this Lab, we used more advanced UI components, from the OutSystems UI Framework and the RichWidgets module (which is part of the System Components).

First, we added a navigation and dynamic sorting to the list of Movies, to allow easily navigating through all the movies, and also sort the list by what the end-user chooses at runtime.

Then, we added two patterns from the OutSystems UI Framework: a Balloon to the list of movies in the Movies Screen, to display the plot summary, and a User Initials, to display the initials of the user next to its name, in the Login Section.

The options were endless to make the application look nicer. Feel free in your own time to enhance it, by using more patterns available in the OutSystems UI Framework, or in the RichWidgets module.

This concludes the OSMDb application. There's much more things that could be done, so if you want to, just imagine it and do it!