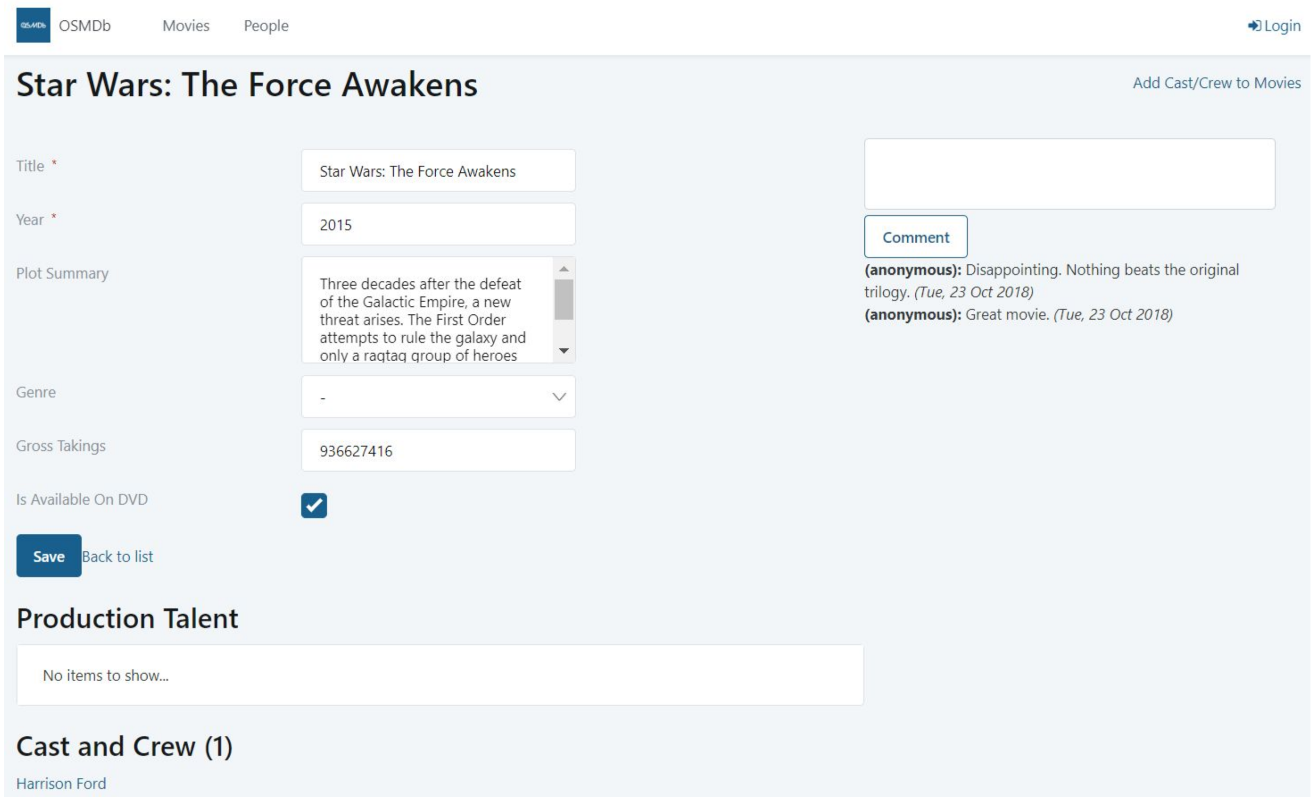
Ajax Interaction Exercise



Star Wars: The Force Awakens

Title *	Star Wars: The Force Awakens
Year *	2015
Plot Summary	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
Genre	-
Gross Takings	936627416
Is Available On DVD	✓

(anonymous): Disappointing. Nothing beats the original trilogy. *(Tue, 23 Oct 2018)*
(anonymous): Great movie. *(Tue, 23 Oct 2018)*

Production Talent

No items to show...

Cast and Crew (1)

Harrison Ford

Introduction

In this exercise Lab we will use the Ajax Submit method for Buttons / Links. So far, all Buttons / Links were just submitting data to the server, using the Submit method. This forced the Screen Preparation to run again and the whole Screen to be rebuilt, after the End of a Screen Action.

To use the Ajax Submit, we will add a new Sidebar to the MovieDetail Screen. That Sidebar will have an Input and a Button to Ajax Submit comments to the database. After the comment is submitted to the database, we will see how to refresh a specific query from the Preparation (without executing the other queries) and specific parts of the Screen (and not the entire Screen). This way, every time a new comment is added, the Comment List is automatically refreshed and the new comment should quickly appear on the Screen.

In summary, in this specific exercise lab, we will:

- Create a sidebar on the MovieDetail Screen
- Add an Input section, as well as a list of comments to the sidebar
- Submit comments using Ajax
- Refresh the list of comments using Ajax

Table of Contents

Introduction 2

Table of Contents 3

Add a Comments Sidebar to the MovieDetail Screen 4

Create Screen Action to Add Movie Comments 11

End of Lab 16

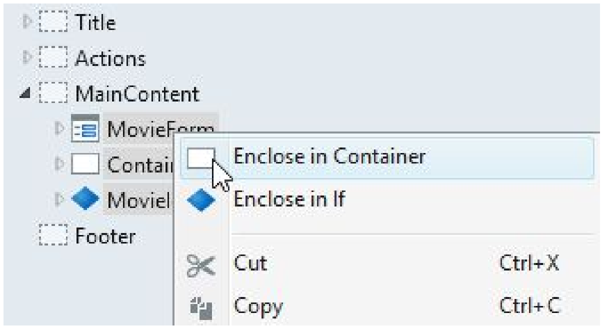
Add a Comments Sidebar to the MovieDetail Screen

We will start this Lab by creating a Sidebar in the **MovieDetail** Screen. We will split the Screen vertically in two sections. The one on the left will keep the Form and the cast and crew info, while the new section on the right will have the option to add comments and read other comments about the movie.

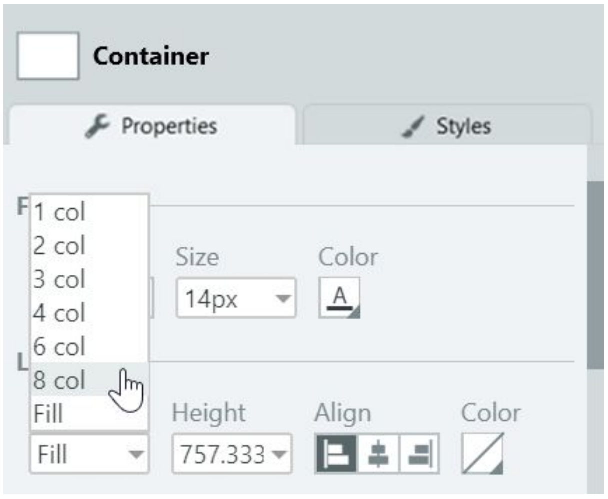
1) Enclose in a **Container** all the main content in the **MovieDetail** Screen with a width of 8 columns. Create a new **Container** with a width of 4 columns to the right of the existing one, to create a sidebar. This sidebar should only be visible when we are editing / viewing an existing movie.

   a) Open the **MovieDetail** Screen and switch to the Widget Tree.

   b) Select everything inside the **MainContent** placeholder and right-click the selection. Select *Enclose in Container* from the pop-up menu.

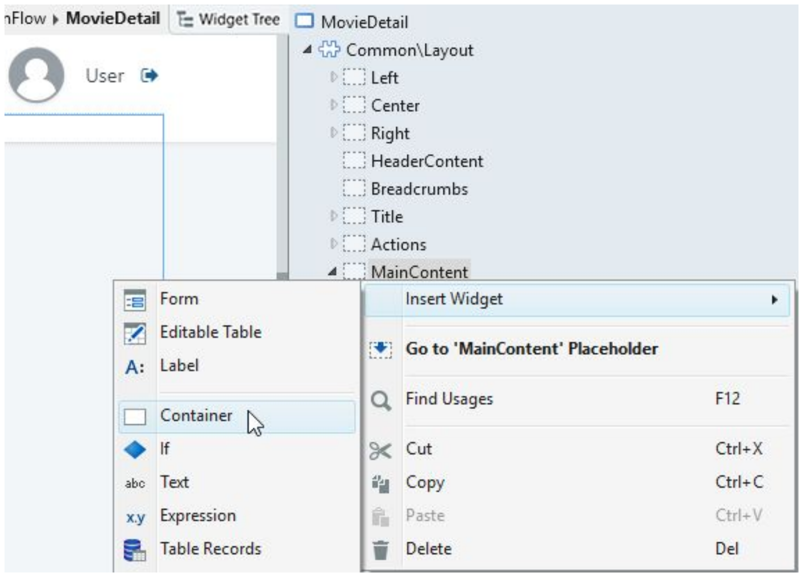


   c) In the Styles Editor, change the **Width** of the Container to *8 col*.

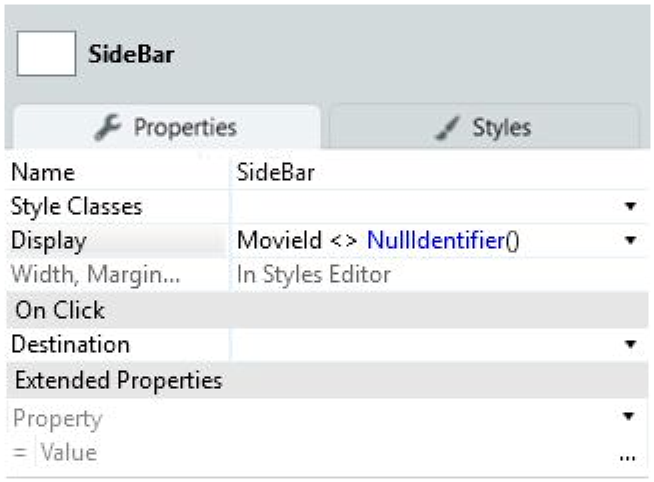


   d) Open the properties of the Container and rename it as *MainContainer*.

e) In the Widget Tree, right-click on the **MainContent placeholder** (not the MainContainer). Select *Insert Widget* from the pop-up menu and then click *Container*.



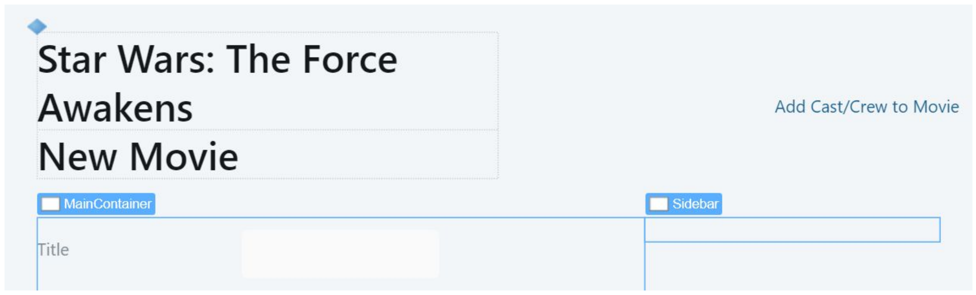
f) Rename this Container *SideBar* and change the Container **Display** property to *MovieId <> NullIdentifier()*. This makes sure that it is only displayed on the page when we are editing / viewing an existing movie.



Name	SideBar
Style Classes	
Display	MovieId <> NullIdentifier()
Width, Margin...	In Styles Editor
On Click	
Destination	
Extended Properties	
Property	
= Value	...

NOTE: The Container Widget has a **Display** property, in which the contents inside the Container will be displayed when the condition defined in the property is evaluated as true.

g) Change the **Width** of the SideBar container to *4 col*. Since the default grid for a responsive module is 12 columns, the **MainContainer** and the **SideBar** will appear side by side on the Screen.



2) Add an Aggregate to the Preparation that returns all the **UserMovieComments** for the current film, including information about the **User** that posted them. They should be sorted from the most to the least recent comment. **Hint:** We will need the *(System)* User Entity.

   a) Open the **MovieDetail** Screen Preparation.

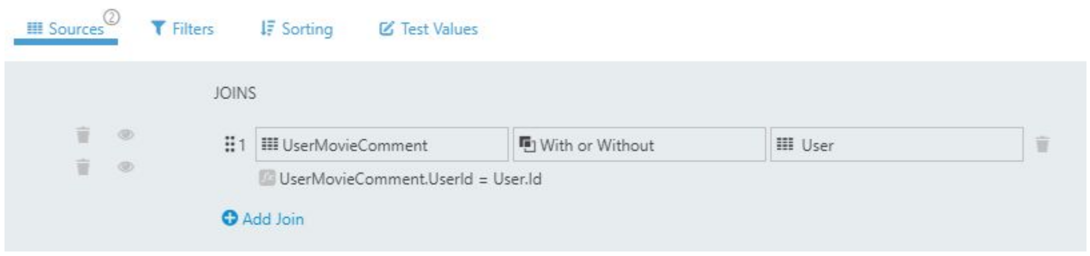
   b) Drag and drop an **Aggregate** before the End statement. Set its **Name** to *GetComments*, and double-click to open it.

   c) Drag and drop both the **UserMovieComment** and the **User** Entities into the Aggregate.

   NOTE: The **User** Entity is inside the *(System)* producer, inside the **Entities** folder (*not* inside the **OSMDb_Core** module).

   ---

   d) In **Sources** confirm that the join type is set to *With or Without*.



UserMovieComment | With or Without | User

UserMovieComment.UserId = User.Id

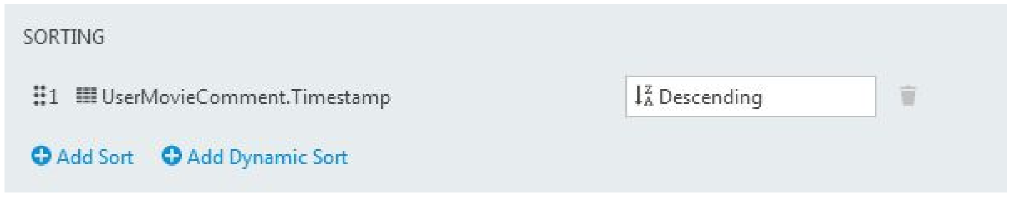
   NOTE: The With or Without will have the effect of allowing users to comment anonymously, since it does not requires that the User is registered in the system. A User registered in the System has a record in the User Entity.

   ---

   e) Go into the **Filters**, and select **+Add Filter** to add the following condition

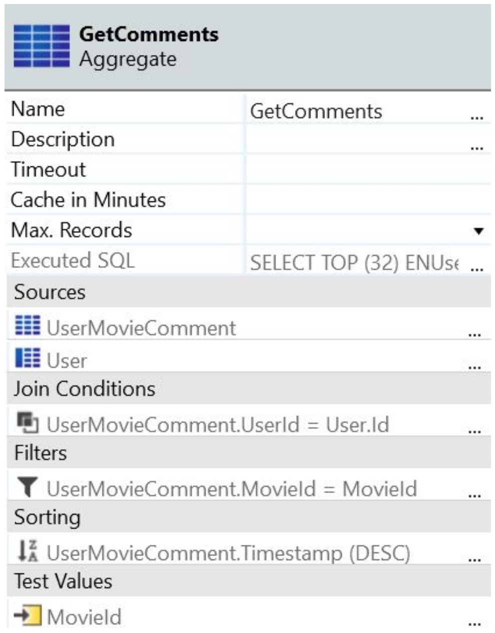
      UserMovieComment.MovieId = MovieId

f) Go into **Sorting** and select **+Add Sort**. Choose the *UserMovieComment.Timestamp* attribute in the dialog, and then select the *Descending* order.


SORTING

1 UserMovieComment.Timestamp — Descending

g) Looking at its properties, the GetComments Aggregate should look like this


GetComments
Aggregate

Name	GetComments
Description	
Timeout	
Cache in Minutes	
Max. Records	
Executed SQL	SELECT TOP (32) ENUse
Sources	
UserMovieComment	
User	
Join Conditions	
UserMovieComment.UserId = User.Id	
Filters	
UserMovieComment.MovieId = MovieId	
Sorting	
UserMovieComment.Timestamp (DESC)	
Test Values	
MovieId	

3) Add a **Local Variable** *MovieComment* (of type **UserMovieComment**) to the Screen. Add to the **SideBar** an **Input** (bound to the **Comment** attribute of the MovieComment Local Variable) and a **Button** to submit the comments to the database. This will be the way that end-user will be able to submit comments about a movie.

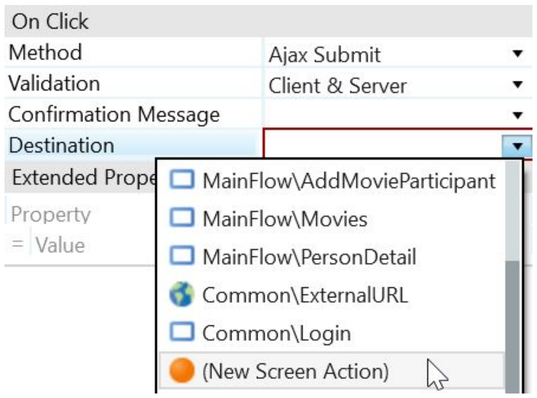
a) Right-click the **MovieDetail** Screen and select the *Add Local Variable*. Define the **Name** as *MovieComment* and set its **Data Type** to *UserMovieComment*.

b) Drag and drop an **Input** Widget into the **SideBar** Container. Set its **Name** to *Input_Comment* and its **Variable** to *MovieComment.Comment*. This attribute of the UserMovieComment record will hold the actual textual comment that the users submit.

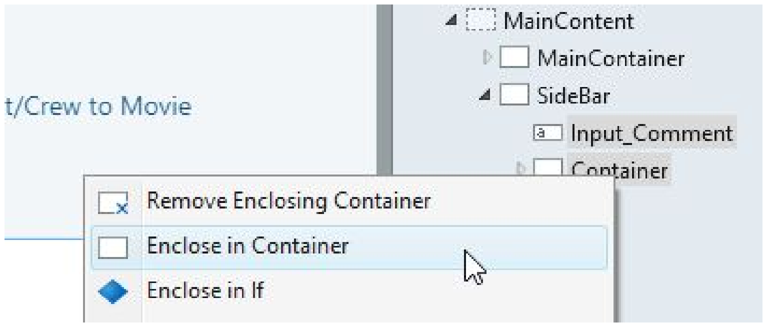
c) Increase the **Text Lines** property to *2* so there's more typing space for comments.

d) Drag and drop a **Button** just below the **Input_Comment**, making sure it's still inside the **SideBar** Container. Set its **Label** to "*Comment*" and its **Validation** property to *Client & Server*.

e) Change the Button's **Method** to *Ajax Submit*. For its **Destination** property, select *(New Screen Action)*.

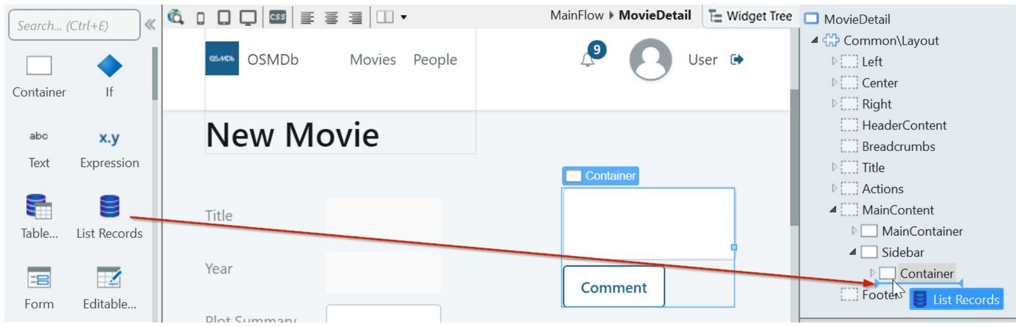


f) In the Widget Tree of the MovieDetail Screen, select both the Input and the Button inside the **SideBar** Container, right-click on the selection and choose the option *Enclose in Container*. This will help organize this section of the Screen later.

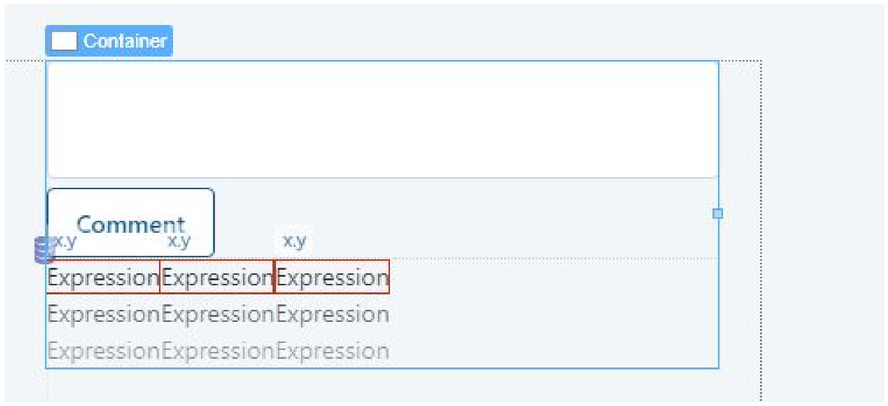


4) Add a **List Records** that shows all the movie comments fetched in the Preparation. Each comment should be displayed in a different row, with the user name (in **bold**), followed by the comment itself, and ending in the comment timestamp (in *italic*).

a) Drag and drop a **List Records** below the Container with the Input and the Button, but still inside the **SideBar**. Set its **Name** to *CommentList*, **Source Record List** to *GetComments.List* and its **Empty Message** to "No comments yet..."

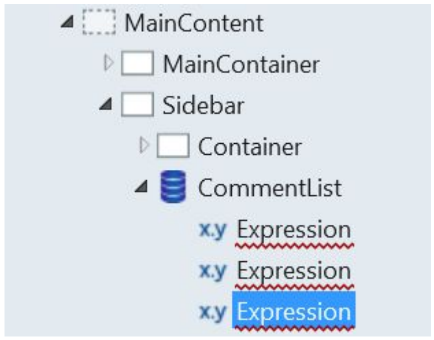


b) Drag three **Expressions** inside the List Records, side by side, like this


Container
Comment
x.y
x.y
x.y
ExpressionExpressionExpression
ExpressionExpressionExpression
ExpressionExpressionExpression

The **first** Expression is going to display the name of the person that made the comment, the **second** one is going to display the comment and the **third** one is going to display the time when the comment was made.

c) Ensure all three are really inside the List Records and not after it. Using the Widget Tree, the **SideBar** Container should look like this



d) For the poster name prefix, set the **Value** of the first Expression to

If (CommentList.List.Current.UserMovieComment.UserId <> NullIdentifier(), CommentList.List.Current.User.Name, "(anonymous)") + ": "

This checks if the user commenting has an Id different than *NullIdentifier()*. If it has, it's because it exists in the User Entity, so we display its name. Otherwise, we display *(anonymous)*.

e) Set the Expression's **Example** property to *Jane:*

f) Set the first Expression to **bold** by changing to the Styles Editor and selecting the Bold shortcut button

g) For the comment itself, set the **Value** of the second Expression to

CommentList.List.Current.UserMovieComment.Comment

h) Set the Expression's **Example** property to *I liked this flick!*.

i) For the comment timestamp, set the **Value** of the third Expression to:
" (" + FormatDateTime(CommentList.List.Current.UserMovieComment.Timestamp, "ddd, dd MMM yyyy") + ")"

NOTE: The **FormatDateTime()** function formats the Date type passed in as a parameter to be displayed, in a more human-friendly way.

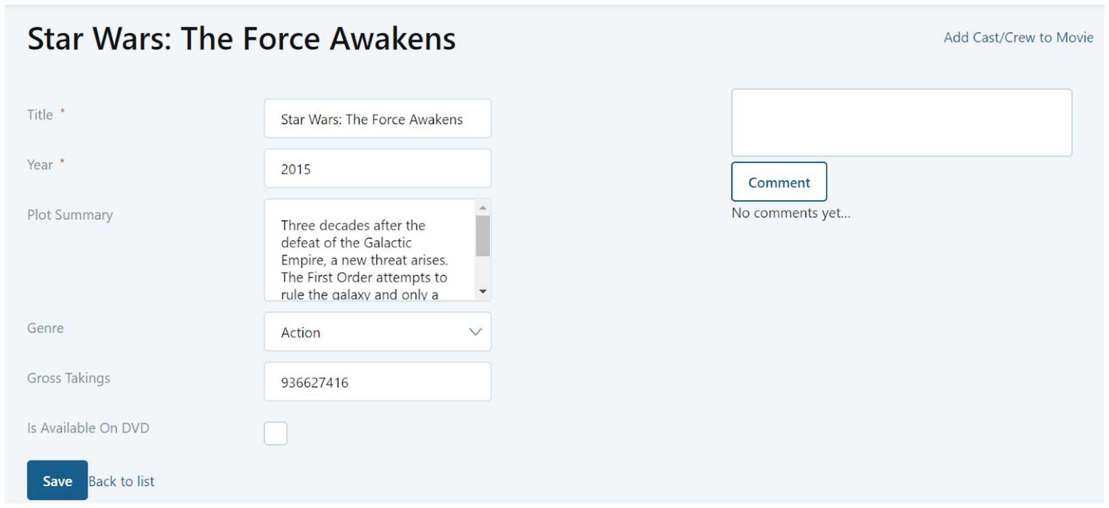
---

j) Set the Expression's **Example** property to *(5 mins ago)*

k) Set the Expression to *italic* by changing to the Styles Editor and selecting the Italic shortcut button.

l) Click the **1-Click Publish** button to publish the application, and access it using your browser.

m) Navigate to the **MovieDetail** Screen, by selecting an existing movie. The Screen should look like this



Star Wars: The Force Awakens

Add Cast/Crew to Movie

Title *	Star Wars: The Force Awakens
Year *	2015
Plot Summary	Three decades after the defeat of the Galactic Empire, a new threat arises. The First Order attempts to rule the galaxy and only a
Genre	Action
Gross Takings	936627416
Is Available On DVD	☐

No comments yet...

Create Screen Action to Add Movie Comments

In this part of the Lab, we will create a Screen Action that will add movie comments to the database (to the UserMovieComment Entity). After adding the comment, we need to refresh the Screen to show the new movie comments, immediately after adding them, using the Ajax Submit method.

1) Implement the **Comment** Screen Action. The Action should add a new record to the **UserMovieComment** Entity.

    a) Open the **Comment** Screen Action of the **MovieDetail** Screen.

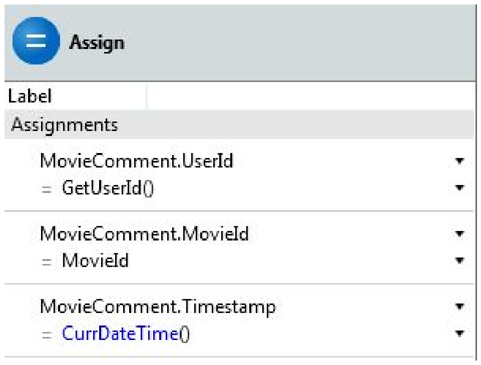
    NOTE: Since the Screen is only expecting the **Comment** text from the user, we need to assign the remaining **UserMovieComment** attributes to the Local Variable **MovieComment,** before using it to create the record in the database.

    ---

    b) Drag and drop an **Assign** statement to the flow the Comment Action.

    c) Add the following three assignments to the Assign statement

    MovieComment.UserId = GetUserId()
    MovieComment.MovieId = MovieId
    MovieComment.TimeStamp = CurrDateTime()



Assign

Label	
Assignments	
MovieComment.UserId	= GetUserId()
MovieComment.MovieId	= MovieId
MovieComment.Timestamp	= CurrDateTime()

    This way we assign the User Identifier, the Movie Identifier and the Timestamp of the Comment to complete the record information.

    NOTE: **GetUserId()** is a function that returns the identifier of the logged in user. When access is anonymous, it returns **NullIdentifier()**.

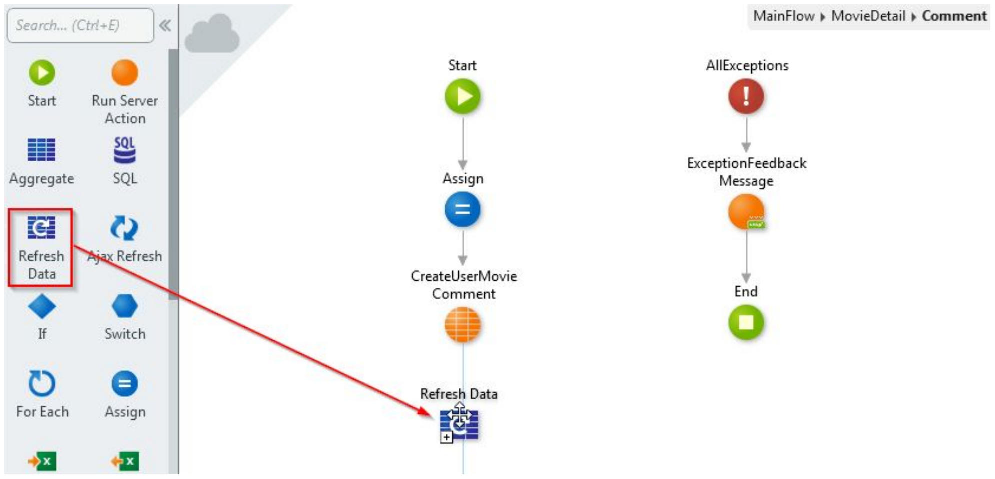
    ---

    d) Drag and drop a **Run Server Action** statement after the Assign. Choose *CreateUserMovieComment* from the **Select Action** dialog.

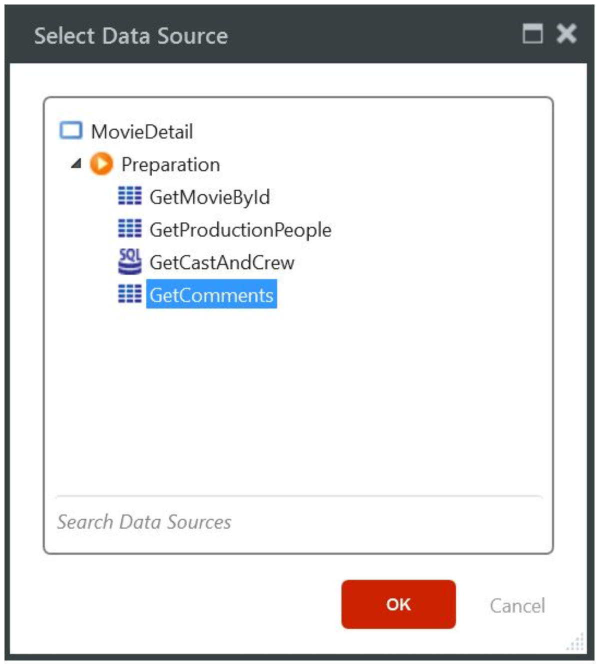
    e) Set the **Source** argument for **CreateUserMovieComment** to *MovieComment*.

2) Still in the Comment Action, after adding the **UserMovieComment** to the database, we need to refresh the Screen to make sure that the new comment appears. To accomplish that, we want to leverage the Ajax Submit method, where the Preparation does not run again. We need to use Ajax Refresh and the Refresh Data nodes appropriately to re-run and refresh only what we want.

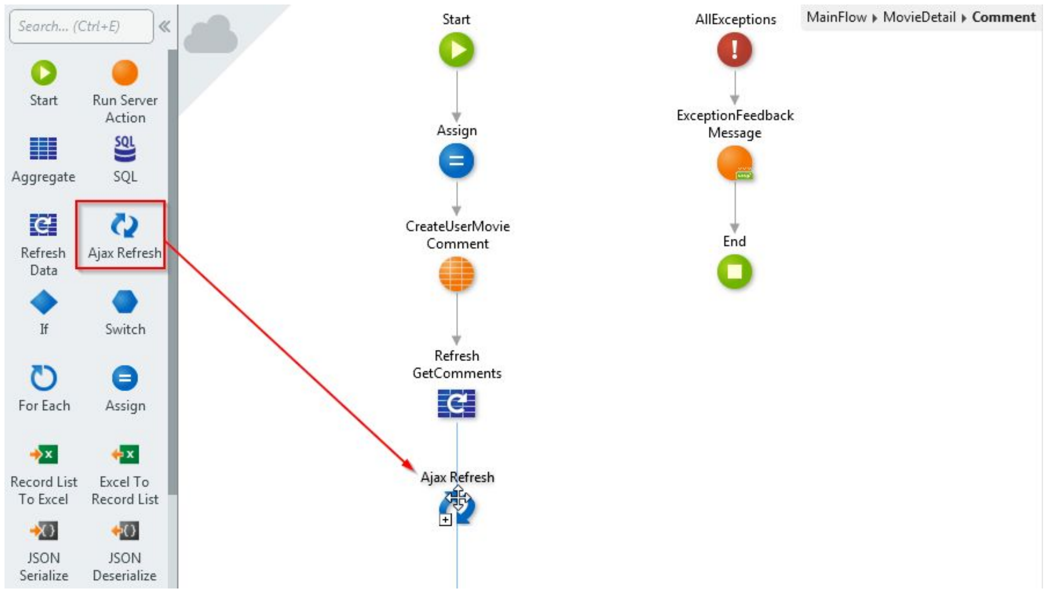
   a) Drag and drop a **Refresh Data** statement after the **CreateUserMovieComment**.



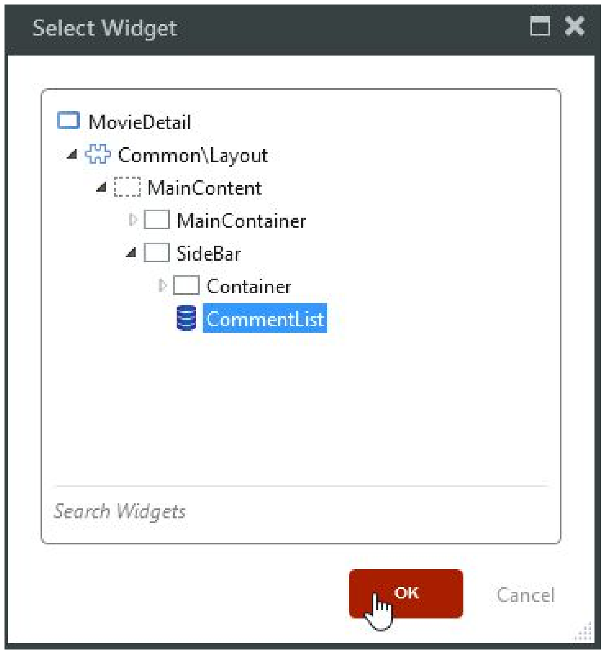
   b) Choose *GetComments* from the **Select Data Source** dialog. This will be the Aggregate we will refresh.



c) Drag and drop an **Ajax Refresh** statement after the Refresh Data.

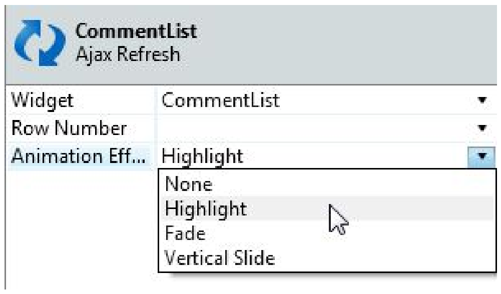


d) Choose *CommentList* from the **Select Widget** dialog, to force the CommentList to be rendered again and display the new data.



NOTE: The steps above are the normal pattern when you want to refresh content on the Screen after an Ajax Submit: (1) refresh the data source of the Widget(s), followed by (2) Ajax Refreshing the Widget(s) that display the data.

e) Set the **Animation Effect** property to *Highlight*.



Widget	CommentList
Row Number	
Animation Eff...	Highlight

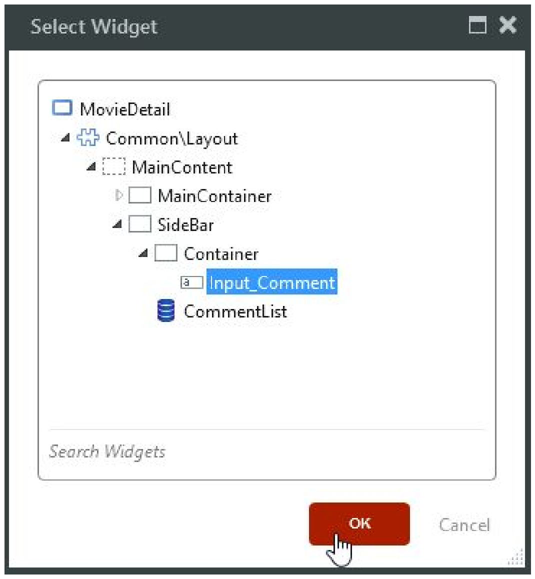
NOTE: The **Animation Effect** allows selecting what visual cue the user will get when the selected Widget is refreshed on the Screen. *Highlight* is a good option here, but feel free to select *Fade* or *Vertical Slide* if you want.

f) Drag and drop an **Assign** statement after the **Ajax Refresh** and add the following assignment to it

MovieComment.Comment = ""

g) Drag and drop another **Ajax Refresh** statement after this Assign.

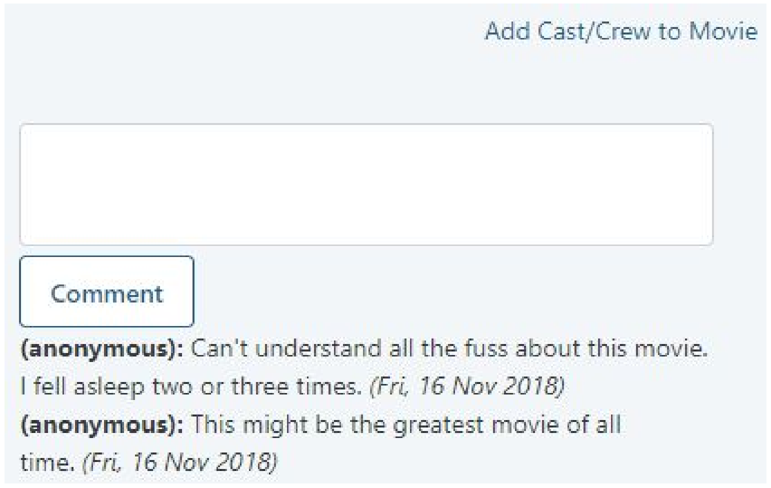
h) Choose *Input_Comment* from the **Select Data Source** dialog.



NOTE: The reason for these last three steps may not appear obvious. Since we want to clean up the comment Input, to prepare it for another comment submission, it was necessary to (1) reset the data source of that input, and then (2) refresh that Widget so that it gets cleared.

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

j) Navigate to the **MovieDetail** Screen by selecting an existing movie. Type in a couple of comments on the sidebar. They should look something like this



(anonymous): Can't understand all the fuss about this movie. I fell asleep two or three times. *(Fri, 16 Nov 2018)*
(anonymous): This might be the greatest movie of all time. *(Fri, 16 Nov 2018)*

End of Lab

In this exercise Lab, we created a sidebar on the MovieDetail Screen, with an Input section and a list of comments. This sidebar also has the option of adding new comments to the movies, which identifies the user that submitted (if not anonymous) and the time of submission.

To accomplish that we used the Ajax Submit on comments, avoiding refreshing the entire Screen and just the parts that we need to refresh. In this particular case, after adding a comment, we just need to refresh the Input and the list of comments, as well as the Aggregate that fetches all the comments from the database. For that purpose, we used the Ajax Refresh and Refresh Data statements.