

Infinite Scrolling and Sorting

Exercise

Outline	2
Hands-on	3
Infinite Scrolling	3
Sorting	5

Outline

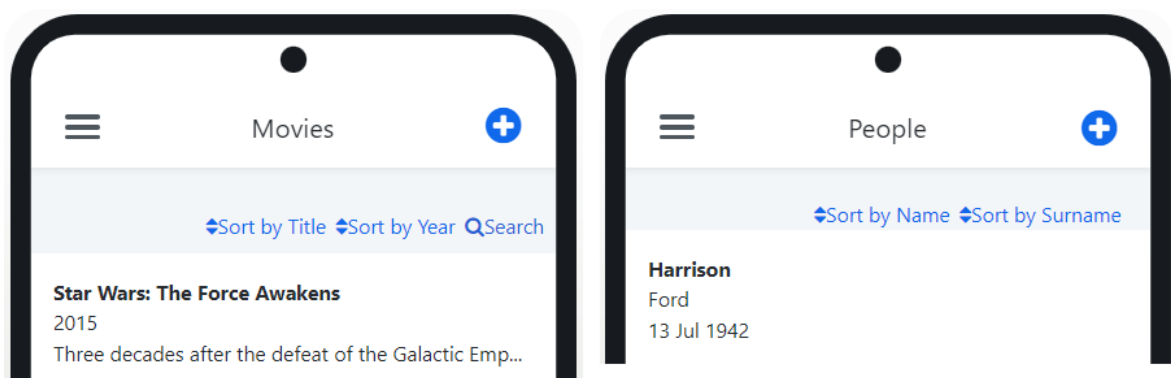
In this exercise, we will add the OnScrollEnding functionality to the Movies and People Screen of the OSMDb application.

Our application can have multiple movies and multiple people in the database, which can lead to a long List, with a lot of records to display in the Movies and the People Screen. To improve the user experience, we will add the onScrollEnding functionality, so only 7 records will be fetched at first, and with further scroll down movements done by the end-users, 5 new records are fetched and displayed on the Screen, per scroll.

This functionality is very common in mobile apps to avoid fetching a large number of records at once, which leads to the notion of perceived performance, where the user immediately sees a part of the data, and by request can fetch other portions of the data.

Then, we will also add dynamic sorting capabilities to the Movies and People Screens. For the Movies, we want to allow end-users with a click on a Link, to sort the movies by title or by year. For the People, we want to support the same behavior for the Name and Surname of the Person.

At the end, both Screens should look like the following screenshots, with the Links right above the Movies and People Lists used to trigger the sorting.



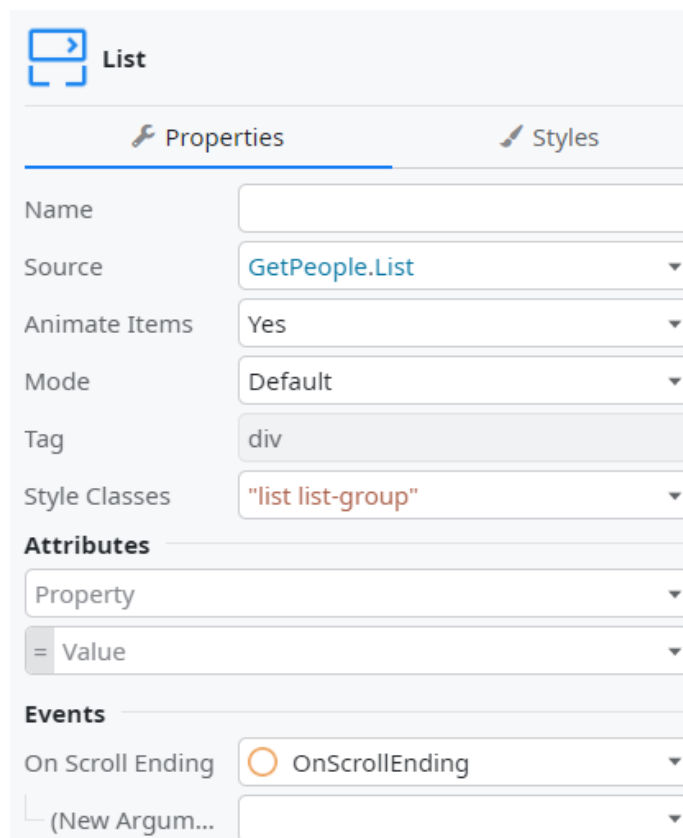
Hands-on

Infinite Scrolling

Let's start by implementing the Infinite Scrolling behavior in the People Screen. This functionality is not directly visible in the UI, so no work has to be done on that end, but it requires a couple of logic tweaks to implement it.

To create the infinite scrolling behavior we need to:

- Limit the GetPeople Aggregate to only fetch a maximum number of records. A MaxRecords variable can be used for that purpose.
- A Client Action needs to be defined on the Screen and associated with the OnScroll Event of the List.



The screenshot shows the configuration panel for a 'List' widget. It has two tabs: 'Properties' and 'Styles'. The 'Properties' tab is active, showing the following settings:

- Name: (empty text field)
- Source: GetPeople.List (dropdown menu)
- Animate Items: Yes (dropdown menu)
- Mode: Default (dropdown menu)
- Tag: div (text field)
- Style Classes: "list list-group" (text field)

Below the properties are sections for 'Attributes' and 'Events'.

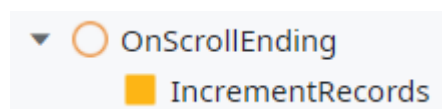
Attributes

- Property: (dropdown menu)
- = Value: (dropdown menu)

Events

- On Scroll Ending: OnScrollEnding (radio button and dropdown menu)
- (New Argum...: (dropdown menu)

- Define a Local Variable on the Client Action, to control how many more records are to be fetched with each scroll.



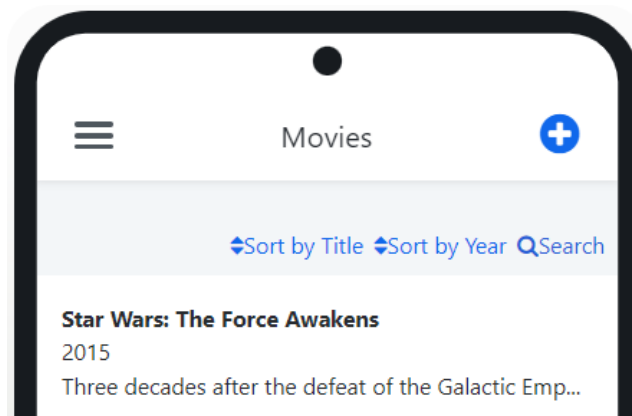
The screenshot shows the configuration for the 'OnScrollEnding' event. It has a dropdown menu with 'OnScrollEnding' selected and a radio button next to it. Below it is a text field labeled 'IncrementRecords'.

- Prepare the logic in the Client Action to do the following things:
 - Check if the data was already fetched, which is useful on the first access to the Screen (there's no point in doing an infinite scrolling with no data). The `IsDataFetched` property of the Aggregate can be useful here.
 - Check if there is more data to be fetched. At this point, using the `Length` property of the List output of the Aggregate (`GetPeople.List.Length`) can be useful, since it stores how many records were fetched by the Aggregate (limited by the `MaxRecords` property). If there is, proceed. If there is not, stop the flow.
 - Increment the `MaxRecords` number with the `IncrementRecords` value, so that the Aggregate fetches more records next time.
 - Re-execute the `GetPeople` Aggregate to fetch more data.

When it's done, the same process should also be repeated for the Movies Screen. Try to select the option *New Infinite Scroll Action* and see what happens!

Sorting

Now, to define the Sorting on both Screens, we need to first add some UI elements to the Screens. Starting with the Movies Screen, we need to create two links right next to the Search Sidebar Link to Sort by Title, and to Sort by Year.



These Links should have as Destination a Client Action, *SortOnClick*, which expects an Input Parameter, *SelectedSort* (Text). This Action will have most of the logic necessary to implement the sorting. On the first Link, we must pass to the Action (through the Input Parameter) that the sorting criterion is the Name of the Movie. To do so, we need to use a special nomenclature: `{Entity}.[Attribute]`. So, in our example: `{Movie}.[Title]`. This nomenclature will be used by the Aggregate to properly sort the information being fetched

from the database. On the second link, Sort by Year, we need to define that the sorting criterion is the Year attribute of the Movie Entity, following the same logic.

The screenshot shows a configuration window for a 'Link' element. It has two tabs: 'Properties' and 'Styles'. The 'Properties' tab is active. Below the tabs, there are several input fields: 'Name' (empty), 'Confirmation ...' (dropdown), 'Enabled' (set to 'True'), 'Visible' (set to 'True'), and 'Style Classes' (dropdown). Below these is an 'Attributes' section with 'Property' (dropdown) and 'Value' (dropdown). At the bottom is an 'Events' section with 'On Click' (set to 'SortOnClick') and 'SelectedSort' (set to '{Movie}.[Title]').

Besides the Action, we will need a Local Variable, *ListSort* (Text), which will hold the sorting criterion being applied to the list of movies.

Inside the SortOnClick Action we need to implement the logic for sorting.

- Check if the sort selected by the user (SelectedSort) is different from the sort currently being applied on the List (ListSort).
 - If it is, then the flow should proceed with changing the current sort to the one recently selected by the user.
 - If not, then it means that the user clicked on the same Link twice in a row. Here, we can add the change of direction on the sort, changing from ascending to descending. To do that, we need to add to the sort criterion the keyword *DESC*, such as: *{Movie}.[Title] DESC*.
- Re-execute the GetMovies Aggregate at the end of the flow.
- Add a new Dynamic Sort to the GetMovies Aggregate using the ListSort variable.

Then, define the same functionality in the People Screen, to sort by Name or Surname.