



Best Practices

Reactive Master Class Exercise

Table of Contents

Table of Contents	2
Outline	3
Resources	4
Hands-on	5
Creating Homepage Screen	6
Create Homepage Screen	6
Fetch Event and Holiday Data	6
Implement the Home Screen Template	8
Configure the Access to the Screen	9
Test the Screen	10
Applying Best Practices to Home Screen	11
Improve Columns Behaviour	11
Replace Table Empty Message	12
Improve the Aspect of the Homepage	12
Set Content Visibility Priority	15
Showing Data On Demand	16
Fetch Teacher Data	16
Add Teacher Information	16
Control Teacher Information Visibility	17
Add Enrolled Students to the Screen	18
Control Enrolled Students List Visibility	19
(Challenge) Improve Data Fetching on ClassDetail Screen	20
Summary	21

Outline

In this exercise, we will apply some of the Best Practices discussed before and improve the navigation and style of the application.

Resources

The resources we need during this exercise can be found in the *Resources* folder the trainer shared with you.

For this exercise we will need the *empty.jpg*, *events.jpg* and *holidays.jpg* images.

Hands-on

In this hands-on, we will create a new screen to be the Home of our application. We will use some of the discussed Best Practices on this page. We will also change the ClassDetail page to show more details about the Teacher assigned to the class, on-demand.

Creating Homepage Screen

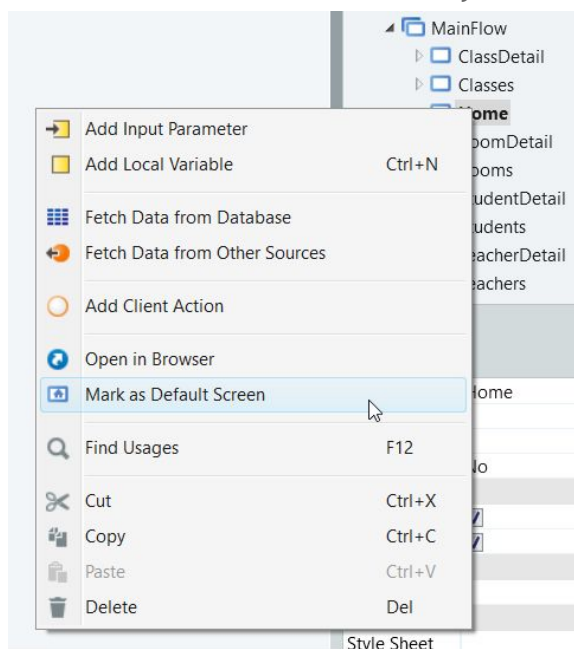
In this section we will create the Homepage screen to show the list of events and holidays.

The process consists of the following steps:

- Create the screen
- Fetch the required data (both lists)
- Prepare the Screen template (using List or Table)
- Configure the access to the screen
- Add an entry into the menu to the screen (optional)

Create Homepage Screen

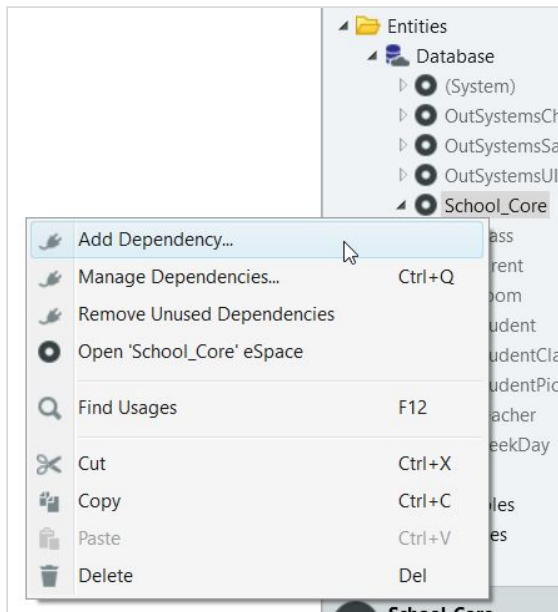
- In your **Interface** Module, add a new screen to **MainFlow** (Interface Layer).
- Select the **Empty** screen template and enter **Home** as the screen name. Press 'Create Screen'.
- Set it to be the **Default Screen** of your module.



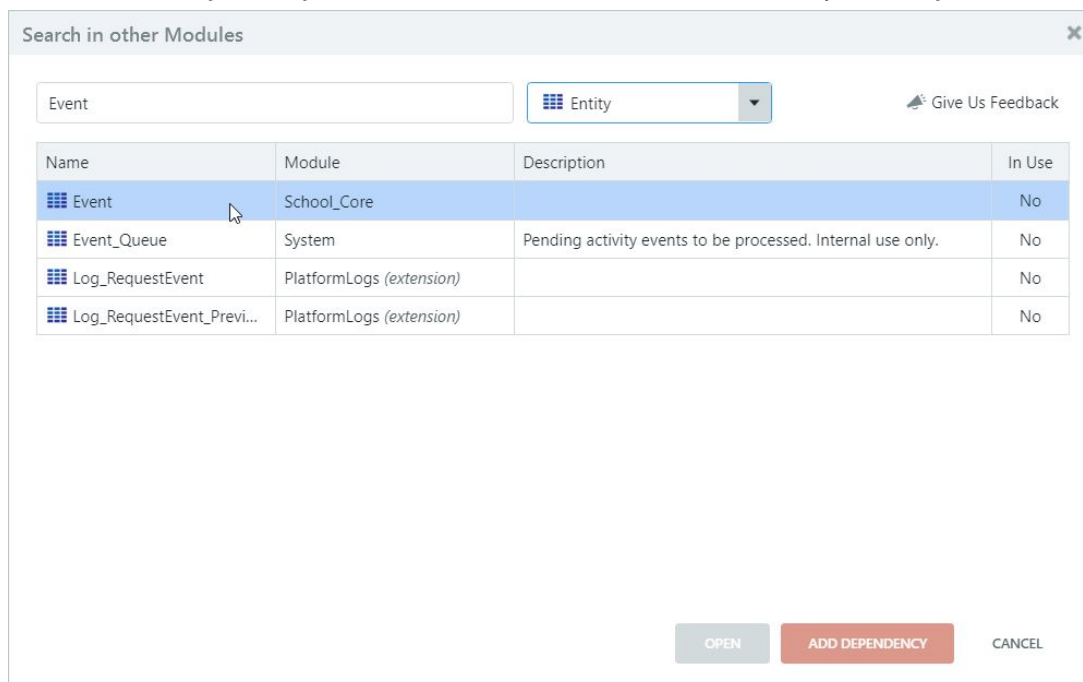
Fetch Event and Holiday Data

- In the **School_Core** module, mark the entities **Event** and **Holiday** as *public* and set the **Expose Read Only** property to *No*. Publish the module.

2. In the **School** module, go to the Data layer and right-click the **School_Core** module. On the context menu, select the **'Add Dependency...'** option.



3. Type **'Event'** on the search box and choose the **Entity** option from the DropDown. Select the **Event** entity from your core module and click the **'Add Dependency'** button.



Note

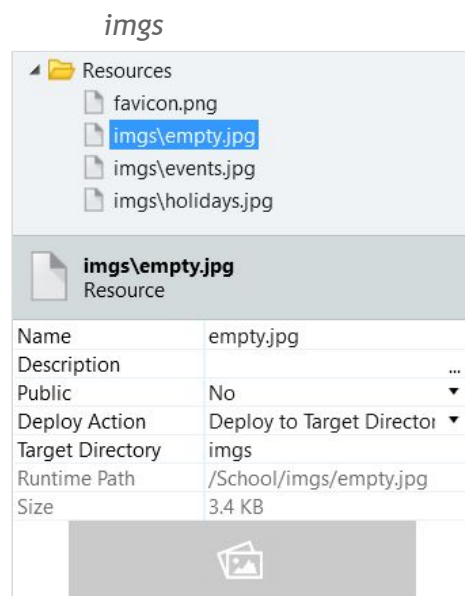
The **'Add Dependency...'** context menu is usually a faster way to add a single dependency to your module than using the **Manage Dependencies**.

4. Repeat (2.) and (3.) to add a dependency to the **Holiday** entity.

5. In the **Home** screen, add a new aggregate to it to fetch the data from the **Event** entity.
6. Do the same, now for the **Holiday** entity.

Implement the Home Screen Template

1. On the **School** module **Data** layer, add as resources the images empty, events and holidays (from the resources folder). Do as follow:
 - a) Select the Resources folder and press **Ctrl+N**, or right-click the Resources folder and select the **Import Resource** option.
 - b) Find the image to add as a resource, select it and click in **Open**.
 - c) Select the option **"Add As Resource"**.
 - d) Select the three images and change the property **Deploy Action** from *Do Nothing* to *Deploy to target Directory*.
 - e) Set the **Target Directory** property to:



2. Open the **Home** screen **Canvas**.
3. Drag a **Columns2** pattern to the **MainContent** placeholder.
4. Follow the next steps to add a list of events to the screen.
 - a) Add a **Container** to the **Column1** placeholder of the **Columns2** pattern.
 - b) Set the Container's **Style Classes** property to:

"card"

- c) Drag the GetEvents aggregate to inside the Container.
- d) Remove the “No Items To Show...” Text that was automatically added. **Do not** remove the IF pattern or the content of the False Branch.
- e) Set the **Show Header** property of the Table to *No*.
- f) Make the Table to have a single column, with the Event Name above, the Date in the middle and the Location at the bottom of the same cell.
- g) Make sure that all information is left aligned.
- h) Add a **Text** pattern above the List. Replace its text by **Events**. Set its Style Classes property to:

"heading3"
- i) Add an IF pattern above the “Events” Text pattern. Set its condition to:

GetEvents.IsDataFetched
- j) In the True Branch, add an **Image** pattern. Set its **Type** property to *External URL*.
- k) Set the image URL property to:

"/School/imgs/events.jpg"

Note

Replace “School” by the name of your module (*School_<your initials>*).
The image will not appear before you publish the module and, eventually, close the screen canvas and open it again.







5. Repeat the (3.) to add the list of holidays. Pay attention to the following:
 - a) The list must be placed at the second column.
 - b) The information to show must be **Name**, **Date** and **Type**.
 - c) The image to use is the **holidays.jpg**.
6. Publish the module.



Configure the Access to the Screen

1. Select the Home screen in the Interface Layer
2. Check the Anonymous option in the Roles section of the Screen properties.
3. Publish the Module.

Test the Screen

1. Open your application in the browser.
2. You should see a screen similar to the next image.

 School
  Teachers
  Students
  Rooms
  Classes
  Login

Events

Name ↕	Location ↕	Date ↕
Open day		15 Sep 2020
Seminary Future XXX Century	Atlanta auditorium	25 Oct 2020
Sign Language Workshop	Forum Star	1 Jan 1900

1 to 3 of 3 items

Holidays

Date ↕	Name ↕	Type ↕	Details ↕
1 Jan 2020	New Year's Day	National holiday	
14 Feb 2020	Valentine's Day	Observance	
25 Feb 2020	Carnival / Shrove Tuesday	Optional Holiday	
19 Mar 2020	St. Joseph's Day	Municipal Holiday	Santarém
19 Mar 2020	Father's Day	Observance	
20 Mar 2020	March Equinox	Season	
29 Mar 2020	Daylight Saving Time starts	Clock change/Daylight Saving Time	
10 Apr	Good Friday	National holiday	

Applying Best Practices to Home Screen

In this section, we will apply a few best practices to the Home Screen.

We will move the content of our “columns” to a Columns2 pattern. One of the advantages of this pattern is to allow adapt the columns depending on the device. For example, we may choose to have a single column in Phone, while keeping two columns in Tablets and Desktop.

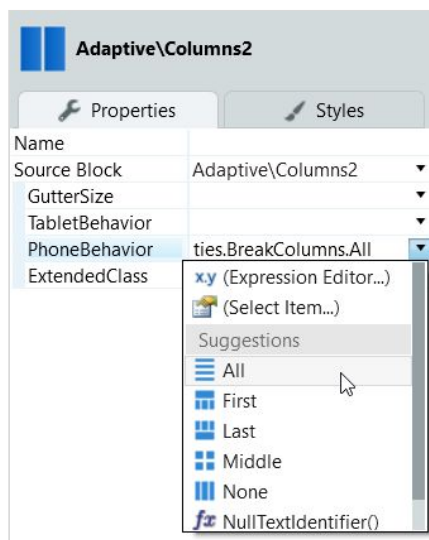
Following the advice to show first the most important data, we will set the column images to appear only after the data of the lists were loaded. This approach can be used in many different situations.

To avoid the page content to “jump” when the images become visible after the data is fully loaded, we will use a wildcard image that will be visible from the beginning. While this does not seem necessary right now because of the browser cache, doing it will allow us to replace the images in the future without causing jumps to the content.

In order to improve the interface aspect, we will replace the Table message, for when no records were found, to a card.

Improve Columns Behaviour

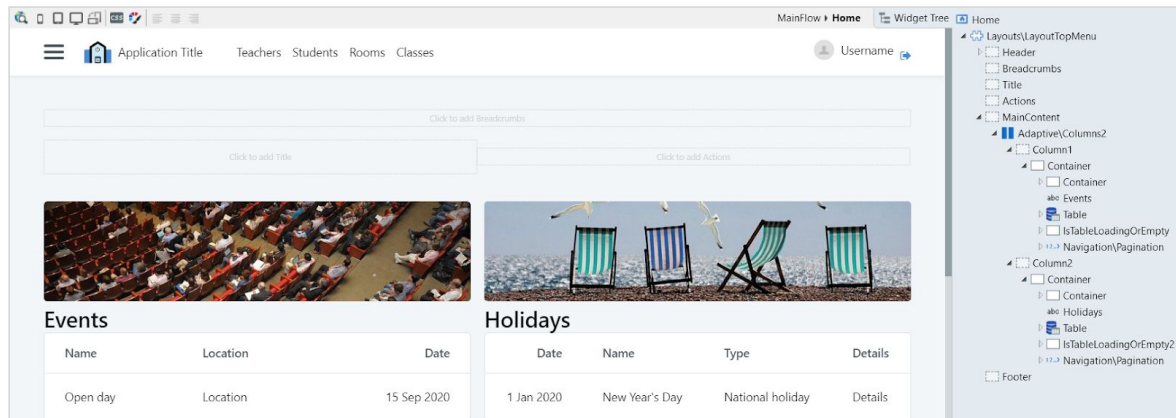
1. On the Home Screen, add a Columns2 to the MainContent placeholder. Set the pattern Phone Behaviour property to *All*.



Note

The Break Columns Phone Behavior set to All will define that on a Phone device, we will break the columns into a single one, so they will become rows, instead.

2. Move the two containers that make from columns to the **Columns2** placeholders (**Column1** and **Column2**). Change the **Containers Width** property to *Fill*.



3. Publish the Module.
4. Open the application on your computer and from your mobile to see the difference.

Replace Table Empty Message

1. Locate and delete the container with the “No items to show...” message on both columns.
2. Add a new **BlankSlate** pattern to the **True Branch** of the *IsEmpty* IF widget of the left column.
3. Change the BlankSlate Icon to:

frown-o

4. Click the Content BlankSlate Content placeholder and type:

There are no events to show...

5. Copy the BlankSlate to the **True Branch** of the *IsEmpty* IF widget of the right column.
6. Replace the text at the content placeholder of the right column BlankSlate to:

There are no holidays to show...

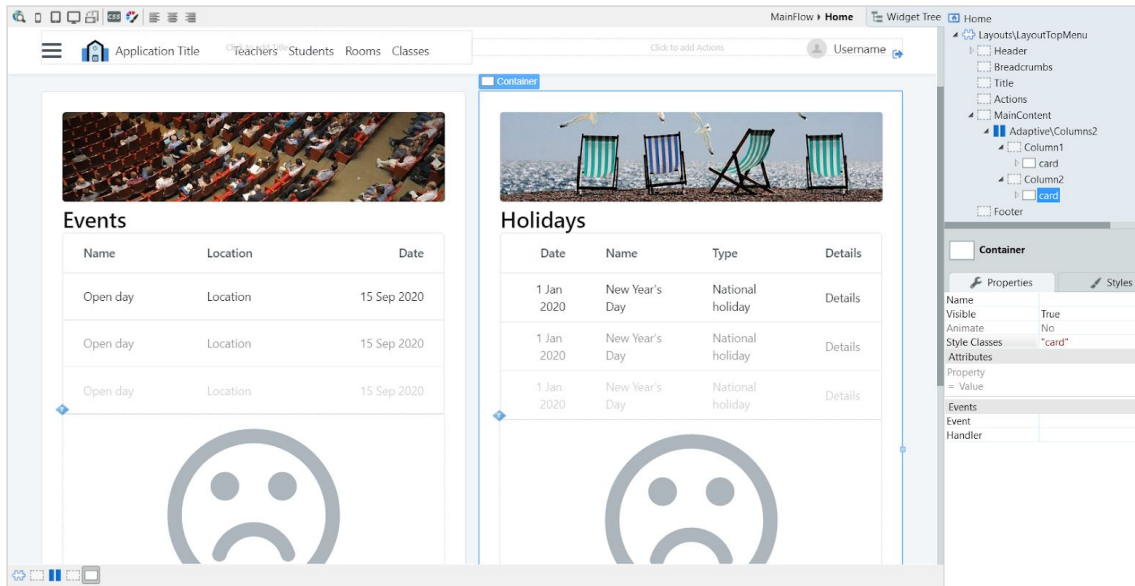
7. Publish the Module.

Improve the Aspect of the Homepage

1. Select each container that encloses the columns and add the following class to the Style Classes property:

“card”

2. Notice the changes to the columns.



3. Set Maximum Number of records to 10, for both tables. Do as follow:

- Change the Default Value property of the MaxRecords local variable to 10.
- Delete the MaxRecords2 variable.
- Select the GetHolidays aggregate and change the Max. Records property to MaxRecords.
- Select the Pagination pattern used with the Holidays table and change the MaxRecords property value to MaxRecords.

4. Set each Table **Show Header** property to *No*.

5. Adjust the columns of the Events Table. Do as follow:

- On the Events Table, move the Location expression to the side of the event expression. Both will stay in the same column.
- Enclose the Location expression in a container.
- Delete the column where the Location was placed.

6. Adjust the columns of the Holidays Table. Do as follow:

- On the Holidays Table, delete the Details column.
- Move the expression with the Type to the right of the expression with the Name of the Holiday. Both will stay in the same column.
- Enclose the expression with the Type in a container.

d) Delete the column where the Type was placed.

e) Move the Date column to be the last column.

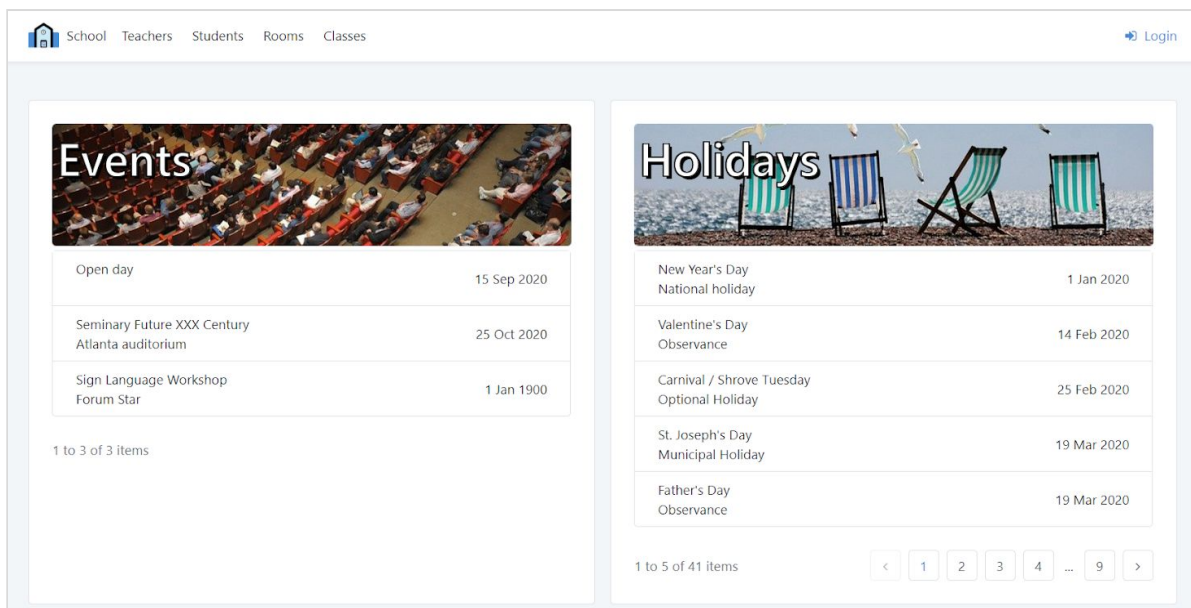
7. Apply the following CSS to the Home screen Style Sheet:

```
.card > div {
    position: relative;
}

.card > div > .heading3 {
    color: white;
    font-size: 48px;
    position: absolute;
    top: 5px;
    left: 5px;
    -webkit-text-stroke: 1px black;
    text-shadow:
        3px 3px 0 #000,
        -1px -1px 0 #000,
        1px -1px 0 #000,
        -1px 1px 0 #000,
        1px 1px 0 #000;
}
```

8. Publish the Module.

9. Open your application and notice the changes.



The screenshot shows a web application interface with a navigation bar at the top containing links for School, Teachers, Students, Rooms, and Classes, and a Login button. The main content area is divided into two columns. The left column is titled 'Events' and features a background image of a large audience in a theater. It contains a table with three items:

Event Name	Date
Open day	15 Sep 2020
Seminary Future XXX Century Atlanta auditorium	25 Oct 2020
Sign Language Workshop Forum Star	1 Jan 1900

Below the table, it says '1 to 3 of 3 items'. The right column is titled 'Holidays' and features a background image of beach chairs. It contains a table with six items:

Holiday Name	Date
New Year's Day National holiday	1 Jan 2020
Valentine's Day Observance	14 Feb 2020
Carnival / Shrove Tuesday Optional Holiday	25 Feb 2020
St. Joseph's Day Municipal Holiday	19 Mar 2020
Father's Day Observance	19 Mar 2020

Below the table, it says '1 to 5 of 41 items' and includes a pagination control with buttons for '<', '1', '2', '3', '4', '...', '9', and '>'.

Set Content Visibility Priority

1. Enclose the Events image in an IF. Set its condition to:

GetEvents.IsDataFetched

2. Add a new Image pattern to the **False** branch of the IF. Set its Type to External URL and the URL to:

`"/School/imgs/empty.jpg"`

3. Repeat the process to the Holidays column.
4. Publish the module.

Showing Data On Demand

In this section we will change the **ClassDetail** screen to show, on demand, the detailed data of the Teacher assigned to the class and the Students enrolled.

Fetch Teacher Data

1. In the **ClassDetail** screen, add a new aggregate to it.
2. Go to the **Data** Layer and drag the **Teacher** Entity to the **Aggregate** canvas.
3. Notice the name of the aggregate changed to **GetTeachers2**.
4. Filter the **GetTeachers** Aggregate by the **TeacherID** attribute of the **GetClassById** Aggregate.
5. Change the name of the Aggregate to:

GetAssignedTeacher

6. Change the property **Fetch** of the **GetAssignedTeacher** Aggregate to *Only on demand*.
7. Publish the module.

Add Teacher Information

1. Open the **ClassDetail** screen, on your **Interface** Module.
2. Add a **Columns2** pattern to the **MainContent** placeholder and move the **Form** to the first column. Remember to adjust the **Form** width to fill the column.
3. Add a **Container** to the second column. Set its bottom margin to *10px*. add the *heading3* class to it.
4. Click the **Container** and type:

Assigned Teacher

5. Add a **CardSectioned** pattern to the second column, after the **Container**.
6. Add an expression to the **Title** placeholder of the pattern and set its value to:

GetAssignedTeacher.List.Current.Teacher.Name

7. Add a **Container** to the **Content** placeholder of the pattern.
8. Add a **AlignCenter** pattern to the **Container**.
9. Add an **Icon** to the **AlignCenter**. Set its name to:

envelope

10. Add an **Expression** after the **Icon**. Set its left margin to (*Auto*) and set its **Value** property to:

GetAssignedTeacher.List.Current.Teacher.Email

11. Add another **Container** to the **Content** placeholder of the pattern. Set its top margin to *10px*.

12. Add a **AlignCenter** pattern to the **Container**.

13. Add an **Icon** to the **Container**. Set its name to:

phone

14. Add an **expression** after the **Icon**. Set its left margin to (*Auto*) and set its **Value** property to:

GetAssignedTeacher.List.Current.Teacher.Phone

15. Enclose both the **Container** and the **CardSectioned** pattern into a **Container** and add the following class to it:

"card content_centered"

16. Add the following CSS to the Interface module Theme Style Sheet:

```
.centered_content {
    text-align: center;
}
```

17. Publish the Module.

Control Teacher Information Visibility

1. Create a local variable of type Boolean and name it:

ShowTeacherInformation

2. Enclose the content inside the **card Container** (**heading3 Container** + **CardSectioned** pattern) inside the second column into an **IF** widget. Set the IF's condition to:

ShowTeacherInformation

3. Add a **Button** to the **False Branch** of the **IF**. Set its label to:

Show Teacher's Details

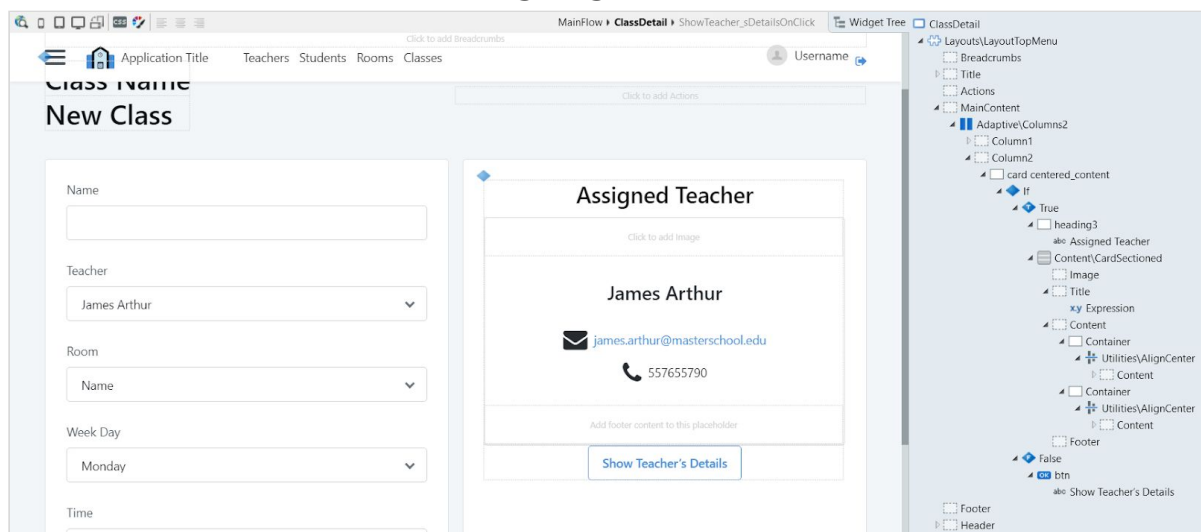
- Set it's visibility to:

ClassId <> NullIdentifier()

- Set the **Button's OnClick** destination to (New Client Action).
- Add a **RefreshData** statement to the action's flow to refresh the **GetAssignedTeacher** Aggregate.
- Add an **Assign** statement to the action's flow, after the **RefreshData** statement, and set the following assignment:

ShowTeacherInformation = True

- Your screen should be like the following image.



- Publish the Module.
- Open your application, add some rooms and a class and test the functionality to see the assigned teacher information.

Add Enrolled Students to the Screen

- Add a new **Container** to the **Container** in the second column on the **ClassDetail** screen. It should be placed after the **IF** widget.
- Click on the new **Container** and type:

Enrolled Students

- Set the **Container** margin top to *20px* and the **Style Classes** to:

heading3

- Drag the **Student** Entity to below the new **Container**.

5. Change the name of the **Aggregate** from `GetStudentClasses` to *GetEnrolledStudents*.
6. Change the **GetEnrolledStudents** **Fetch** property to *Fetch on demand*.
7. Add a new **Filter** to the **GetEnrolledStudents** **Aggregate**:

Class.Id = ClassId

8. Remove the **Class** column on the **Table** that was created.
9. Enclosed in a **Container** the three elements added (**Table**, the **IsTableLoadingOrEmpty Container** and the **Pagination**). Add a top margin of *10px* to the **Container**.
10. Replace the Table Empty message the same way as it was done in the previous section. Use the following message:

There are no Students assigned to this class...

Control Enrolled Students List Visibility

1. Add a new local variable of type Boolean and name it:

ShowAssignedStudents

2. Enclose the Container with the Enrolled Students feature in an IF and set the following condition:

ShowAssignedStudents

3. Add a Button to the False Branch of the IF. Set its label to:

Show Enrolled Students

4. Set it's visibility to:

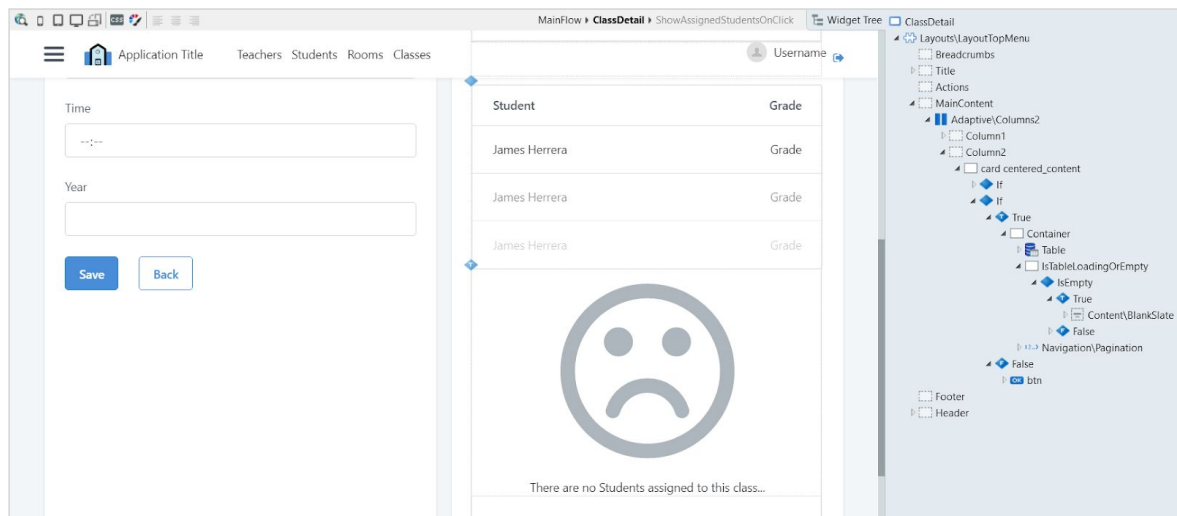
ClassId <> NullIdentifier()

5. Set the Button's On Click destination to (New Client Action).
6. Add a RefreshData statement to the action's flow to refresh the GetEnrolledStudents Aggregate.
7. Add an Assign statement to the action's flow, after the RefreshData statement, and set the following assignment:

ShowEnrolledStudents = True

8. Publish the Module.

9. Your screen should be like the following image.



Note

At this moment, there are no Students enrolled and no way to enroll a student in a class. In a future assignment we will implement the functionality in this screen.

(Challenge) Improve Data Fetching on ClassDetail Screen

As a challenge, improve the data fetching on ClassDetail, using a Data Action to fetch all data that is required immediately when the screen is requested by the user, replacing the Aggregates that fetch the data.

Summary