WEB ARCHITECTURES
Assignment 5

Riccardo Gennaro

January 6, 2023

1 Introduction

1.1 Problem statement

This aims at developing a web application via Angular framework. More specifically, the application uses the Scottish Parliament APIs to retrieve information, in JSON format, about the parliamentarians, and offers to a user

- the **list of the politicians**, including names and personal photos. If the photo is not available a default one is shown.
- for a given politician, the ID, some personal information (name, date of birth) together with the parliamentarian's parties and websites are displayed.

Since Angular is used to develop single-page applications, we change the state of the app using Angular Routing.

2 Implementation

Following, a description of the workflow of the application.

2.1 Connection to the web application

Upon connection to the web-app, the user is redirected to route-path `[base]/list` with component `MspListComponent`. The redirection is defined in the exported Routes in `app.routes.ts`.

2.2 List display

2.2.1 MspListComponent - BackEnd

Redirecting to `[base]/list` triggers `MspListComponent#ngOnInit()`. This method retrieves the MsSP (Members of the Scottish Parliament) data via API

'https://data.parliament.scot/api/members'. In particular, ngOnInit() calls service method MspService#getMspEntries() to retrieve the observable bound to the GET request to the API. Successively, we subscribe to the returned observable. On next subscription, we put the retrieved data in attribute MspListComponent#mspEntries.

2.2.2 MspListComponent - Frontend

In file msp-list.component.html, the data is presented via a grid containing components of type MspGridItem that receive in input an MspEntry. These components are generated through iteration in directive *ngFor.

2.2.3 MspGridItem - BackEnd

Given the mspEntry got through the @Input decorator, this component expose to the previously retrieved data about the iterated parliamentarian. Furthermore, two methods are implemented

- **getImage()**, used to return the PhotoURL in order to display the parliamentarian photo;
- **onItemClick()**, used to change route to [base/msp-details/:id]. This function is called when the component is clicked at frontend.

2.2.4 MspGridItem - FrontEnd

The grid is filled with clickable mat-cards displaying the MSP's name and photo via interpolation.
At this point the list is fully rendered.

2.3 MSP details display

Clicking the card of an MSP will call MspGridItem#onItemClick() changing to route path [base/msp-details/:id], instantiating MspDetailComponent.

2.3.1 MspDetailComponent - BackEnd

Redirecting to [base/msp-details/:id] triggers
MspDetailComponent#ngOnInit(). At this point, the MSP ID is retrieved from the route parameters, and, via a pipe, we get the personal data, the parties and the websites of the MSP.
During the implementation of this method, nested subscriptions were avoided since they cause memory leaks and make the code unmanageable. To solve the problem of this pattern, while still managing the asynchronous actions, a pipe with multiple rjsx#map is used to concatenate the different requests used to retrieve the needed information.

The service methods used to retrieve these information are similar to the one described in 2.2.1. Worth mentioning is `MspService#getMspEntry(id:number)` that can throw an error caused by a not found MSP ID. This error is handled via a `rxjs#catchError` in the pipe. The handling consist in redirecting to route path `[base]/list`.

2.3.2 MspDetailComponent - FrontEnd

This components contains three `mat-card`

- **Personal Details**, that shows data from component `MspPersonalDetailComponent`
- **Parties**, that shows data from component `MspPartyDetailsComponent`
- **Websites**, that shows data from component `MspWebsitesDetailsComponent`

Striving to maintain high modularity, resulted in a high number of components.

The backend of these three components do not contain any notable code. The data is shared from parent to children via Input() decorations, the data is renderend in the frontend via interpolation.
At this point, the MSP details are fully rendered.

3 Deployment

3.1 List current MsSP list

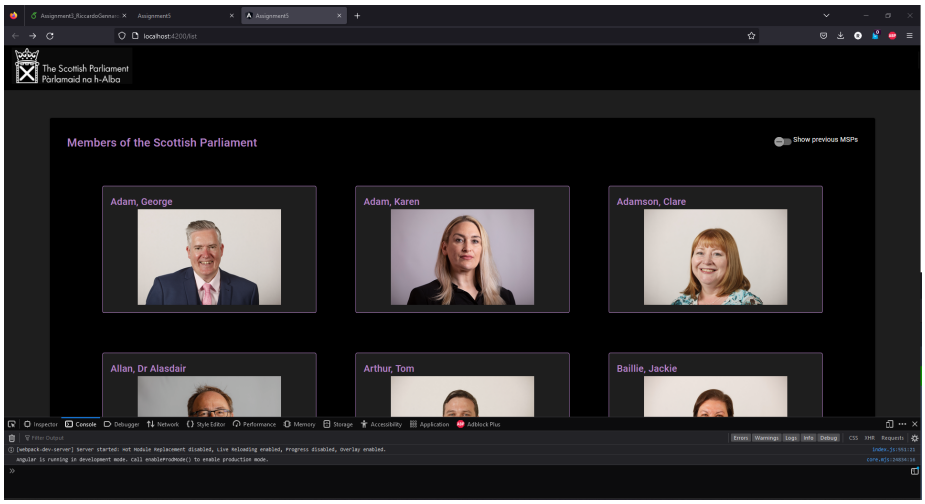

Figure 1: Rendered upon opening

3.2 Current MSP display

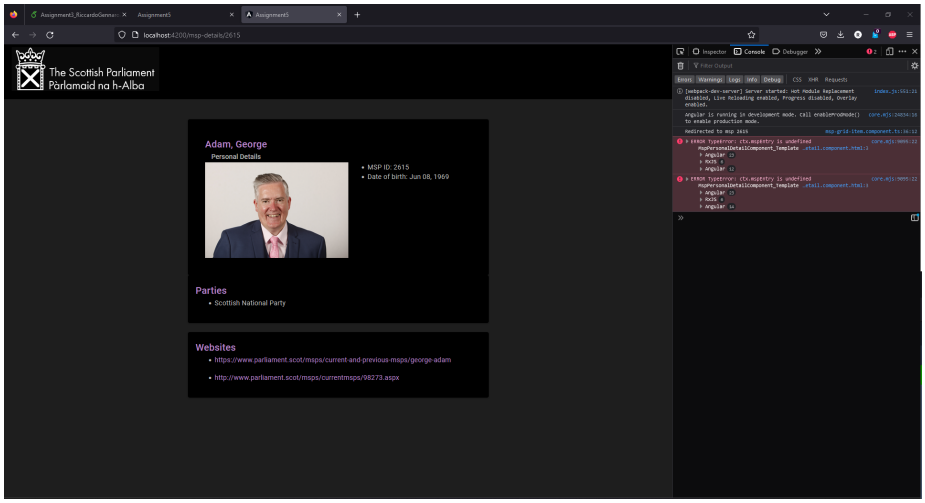

Figure 2: Rendered after clicking on Adam, George card

3.3 List all MsSP list

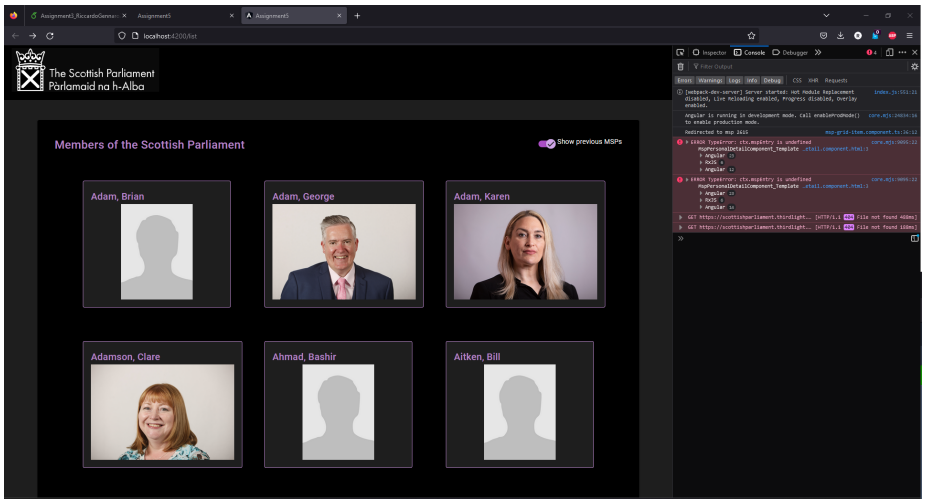

Figure 3: Rendered after checking the slide toggle

3.4 Display non-current MSP

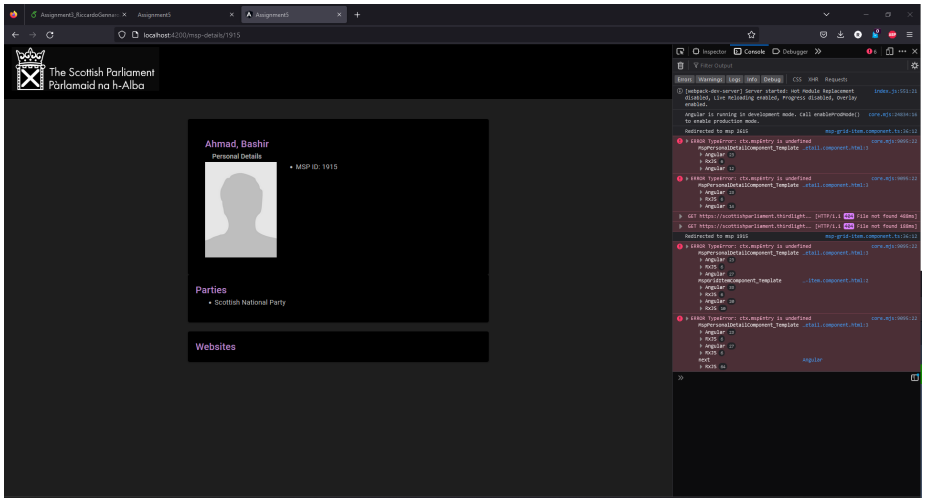

Figure 4: Rendered after clicking on Adam, Brian card

3.5 List all MsSP list - Tomcat

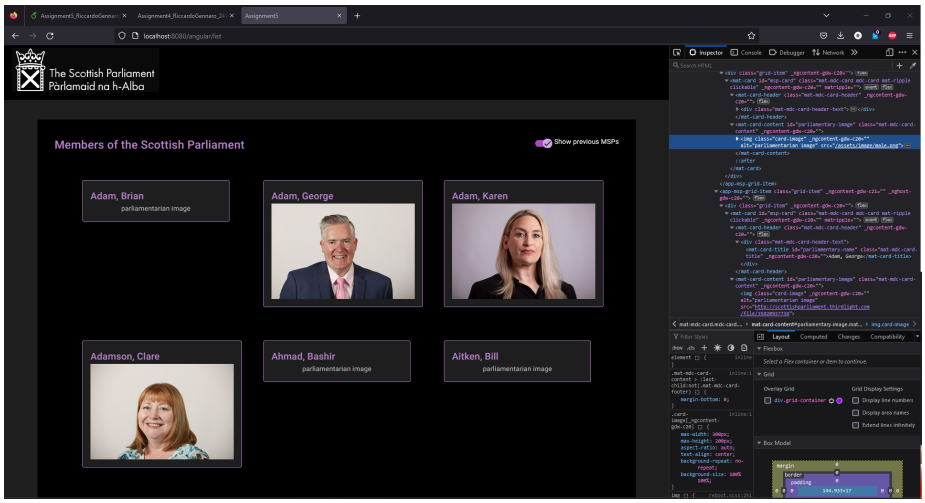

Figure 5: App deployed in Tomcat is the same that the one deployed using command `"ng serve"`. Only one difference - see Comments

4 Comments and notes

For some reasons, when the project is deployed in Tomcat, the default images are not rendered. As you can see from the project structure, and from the image above, even if the default images are in the same folder as the logo of the parliament, and the path of the default images are correctly interpolated, the images cannot be rendered.

This problem is not present when running the project with command `"ng serve"`.

Another problem can be seen at Figure 2. Upon opening the details of a MSP, a TypeError will be thrown with message `"ctx.mspEntry is undefined"`. I tried to locate the source of this error but I was unable to fix it.