Web Architectures - Delivery 5

Marrocco Simone

January 5, 2023

Contents

1 The assignment

Our task was to create a Front End application that works with the Scottish Parlament API using Angular. There are two pages: the first one shows all the members. When clicking a particular member, a new page opens showing more data about her/him.

2 Tecnology used

The project was created with the **ng** command line interface, using the **ng new** command. To start the application it was used **ng serve**, to create a new component **ng generate component X**, and to build the files to use in production **ng build**.

To write the code it was used VScode because of its compatibility with typescript and the aviable addons. The code is also loaded on a repo, visitable at
https://github.com/Marrocco-Simone/AngularScottishParlamentFE

Github allows the users to serve the content from a repository to a static website, so the best way to look up at the project is by going to
https://marrocco-simone.github.io/AngularScottishParlamentFE/

There were some problem with the Angular build, used later with the Tomcat server: in the **index.html** it is generate a tag **<base href="/">** which does not allow the web browser to find the javascript files if not hosted on a basic url (like **localhost:4200/**) but instead on a sub page (like **localhost:4200/AngularWarExploded/**). The solution is change the tag to **<base href=".">**, but this means it is not possible to just use the builded files directly. For more information, you can look at
https://stackoverflow.com/questions/50858981/html-relative-paths-issue-for-script-src

3 The final website

The CSS is aviable in the source code

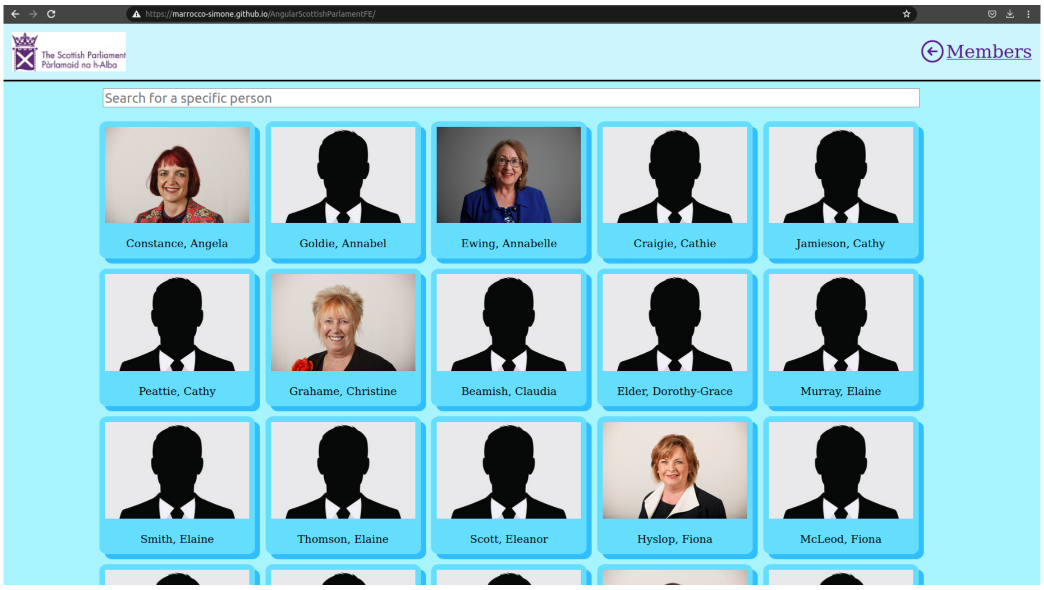
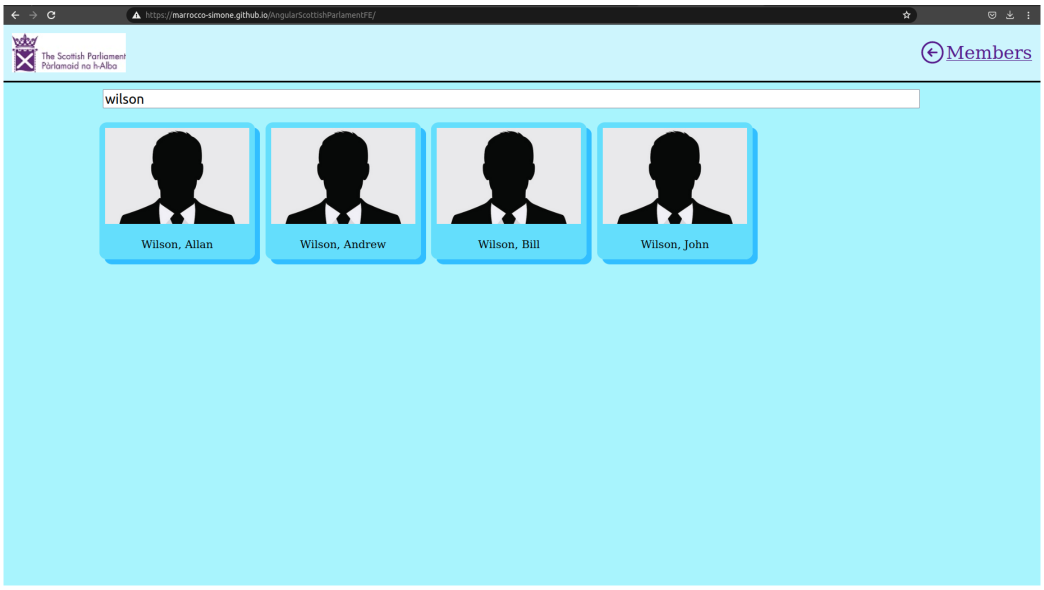
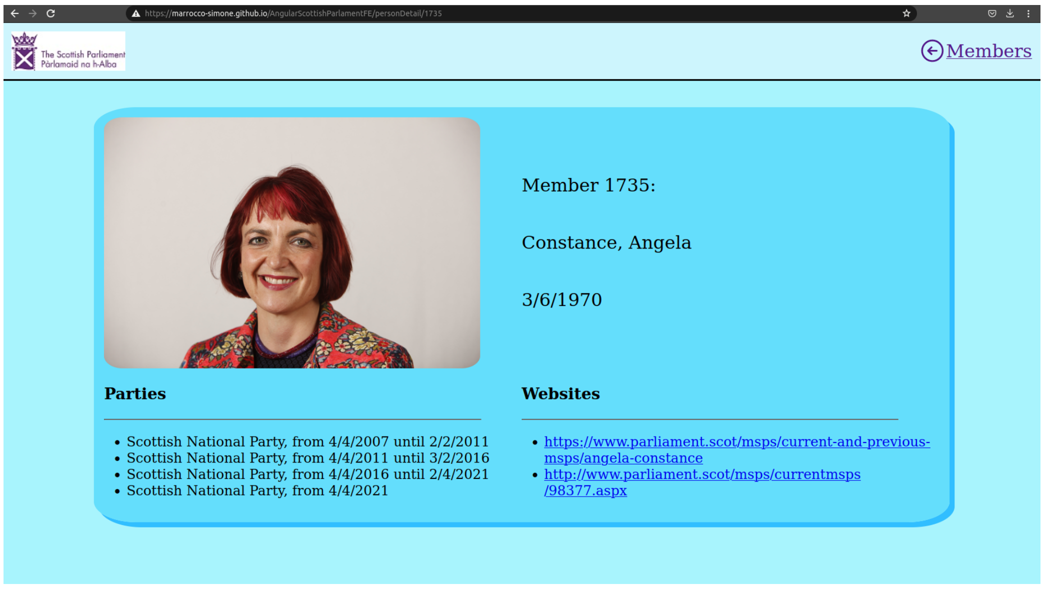


Figure 3.1: Main Page



https://marrocco-simone.github.io/AngularScottishParlamentFE/
The Scottish Parliament Pàrlamaid na h-Alba
Members
wilson
Wilson, Allan
Wilson, Andrew
Wilson, Bill
Wilson, John

Figure 3.2: Searching for a member


https://marrocco-simone.github.io/AngularScottishParlamentFE/personDetail/1735
The Scottish Parliament
Pàrlamaid na h-Alba
Members

Member 1735:

Constance, Angela

3/6/1970

Parties

- Scottish National Party, from 4/4/2007 until 2/2/2011
- Scottish National Party, from 4/4/2011 until 3/2/2016
- Scottish National Party, from 4/4/2016 until 2/4/2021
- Scottish National Party, from 4/4/2021

Websites

- https://www.parliament.scot/msps/current-and-previous-msps/angela-constance
- http://www.parliament.scot/msps/currentmsps/98377.aspx

Figure 3.3: Person Detail Page

4 App component

```
<app-header></app-header>
<router-outlet></router-outlet>
```

Code 1: App component html

The HTML of the top component is quite simple: we load an header on top and we inject the router, which will choose what component to load.

4.1 Header component

```
<header>
  <img src="https://data.parliament.scot/Content/Images/SP_145_50.jpg" />
  <a routerLink="/">
    <svg></svg>
    Members
  </a>
</header>
```

Code 2: Header component constructor

In the header, we load the scottish parlament logo taken from their website and an hyperlink redirecting to the first page, enriched with an svg (not shown here). The redirect is done with the router.

4.2 Router

```
const routes: Routes = [
  { path: "", component: MainComponent },
  { path: "personDetail/:personId", component: FullPersonComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Code 3: Router component module

In the router we define two pages: the main page, located on /, uses the main component. The second page, showing the full person details, uses a dynamic link with the person ID to load the requested member (we will see later how to recover in the component this id from the url).

5 API data and services

We have three services, used to get the members, their parties and their websites. Each service has a constructor, which calls async functions to fetch the data from the API server, and some functions that work with the data they hold.

The member service is called on the main component constructor. The rest of the data, about the people websites and parties, is loaded only when it is needed, as the client clicks on one of the members to go to the detail page. It is possible to verify when the data is fetch by pressing f12, since console logs are added to show that it works as intended (if you're looking on the github page, remember to turn off the warning messages)

Unfortunately, there is no API to get only a single person data, so we need to get the full list for everyone. However, this means that after the first click the rest of them are going to load instantly.

```
export class WebsitesService {
  private websites!: Website[];

  private async getWebsites() {
    console.log('fetching websites');
    let res = await fetch(websitesApiUrl);
    this.websites = await res.json();
  }

  constructor() {
    this.getWebsites();
  }

  getWebsiteUrls(personId: string): string[] {
    if (!this.websites?.length) return [];
    return this.websites
      .filter((w) => `${w.PersonID}` == personId)
      .map((w) => w.WebURL);
  }
}
```

Code 4: Websites API Service

```
export class MembersService {
  members: Member[] = [];

  private async getMembers() {
    console.log('fetching members');
    let res = await fetch(membersApiUrl);
    this.members = await res.json();
  }

  constructor() {
    this.getMembers();
  }

  getSingleMember(personId: string) {
    return this.members.find((m) => `${m.PersonID}` == personId);
  }

  getProfilePicture(member: Member) {
    let url = member.PhotoURL
      ? member.PhotoURL
      : 'assets/blank-profile-picture.jpg';
    return url;
  }

  getBirthDate(member: Member) {
    if (!member.BirthDate) return '';
    try {
      let date = new Date(member.BirthDate);
      return `${date.getDay()}/${date.getMonth()}/${date.getFullYear()}`;
    } catch (e) {
      return '';
    }
  }
}
```

Code 5: Members API Service

```
export class PartiesService {
private parties!: Party[];
private members_parties!: MemberParties[];

private async getParties() {
  console.log('fetching parties');
  let res = await fetch(partiesApiUrl);
  this.parties = await res.json();
}


```

```
  private async getMembersParties() {
    console.log('fetching member parties');
    let res = await fetch(membersPartyApiUrl);
    this.members_parties = await res.json();
  }

  constructor() {
    this.getParties();
    this.getMembersParties();
  }

  private getPartyName(party_id: number): string {
    if (!this.parties?.length) return "";
    return this.parties.find((mp) => mp.ID == party_id)?.ActualName ?? "";
  }

  getPersonParties(personId: string): string[] {
    if (!this.parties?.length || !this.members_parties?.length) return [];
    let member_parties = this.members_parties.filter((mp) => `${mp.PersonID}` == personId);
    let parties_names: string[] = [];

    for (let pmp of member_parties) {
      let party_name = this.getPartyName(pmp.PartyID);
      if (!party_name) continue;

      if (pmp.ValidFromDate) {
        let date = new Date(pmp.ValidFromDate);
        party_name += `, from ${date.getDay()}/${date.getMonth()}/${date.getFullYear()}`;
      }
      if (pmp.ValidUntilDate) {
        let date = new Date(pmp.ValidUntilDate);
        party_name += ` until ${date.getDay()}/${date.getMonth()}/${date.getFullYear()}`;
      }

      parties_names.push(party_name);
    }

    return parties_names;
  }
}
```

Code 6: Parties API Service

6 Main component

In the main component, we make full use of the power of angular in creating dynamic pages with very little code.

The page has two main parts: the search bar and the grid of members.

The search bar is made of an input tag where we intercept the key-pressed event and save the text value in a variable.

For the grid of members, we cycle through each one and show it if the member name contains the searched value. Of course, if the bar is empty all the members will be shown, since every string contains the null one. The way of showing a single member is leaved to another component, to which we pass the member data.

```
export class MainComponent {
  inputSearchValue = '';

  constructor(public membersService: MembersService) {}

  onKeyPressed(event: any) {
    this.inputSearchValue = event.target.value;
  }
}
```

Code 7: Main component

```
<div class="minimized-people">
  <input
    (keyup)="onKeyPressed($event)"
    placeholder="Search for a specific person"
  />
  <div *ngFor="let m of membersService.members">
    <app-minimized-person
      *ngIf="
        m.ParliamentaryName.toLocaleLowerCase().includes
(
          inputSearchValue.toLocaleLowerCase()
        )
      "
      [member]="m"
    ></app-minimized-person>
  </div>
</div>
```

Code 8: Main html

7 Minimized Person component

With the member we get as an input from the parent component, this one creates an html link to its detail page, showing the profile picture and the name of the person. The URL for the picture is given from the member service, which returns the blank profile picture image if there is no personal picture.

```
export class MinimizedPersonComponent {
  @Input()
  member!: Member;

  constructor(public membersService: MembersService) {}
}
```

Code 9: Minimized Person component

```
<a [routerLink]="['/personDetail', member.PersonID]"
class="minimized-person">
  <img src={{membersService.getProfilePicture(member)}}/
>
  <p>{{member.ParliamentaryName}}</p>
</a>
```

Code 10: Minimized Person html

8 Full Person component

In the full page, we use all the three services (two of which will be initialized when this component is loaded for the first time). In the constructor we get the member ID from the URL and search for the corresponding member. If there is none, the client gets redirected to the main page.

```
export class FullPersonComponent {
  member!: Member;
  personId!: string;

  constructor(
    route: ActivatedRoute,
    router: Router,
    public membersService: MembersService,
    public partiesService: PartiesService,
    public websiteService: WebsitesService
  ) {
    route.params.subscribe((params: Params) => {
      this.personId = params['personId'];
      let member = membersService.getSingleMember(this.
personId);
      if (!member) {
        router.navigate(['/']);
        return;
      }
      this.member = member;
    });
  }
}
```

Code 11: Full Person component

```html
 <div class="full-person">
   <img src="{{ membersService.getProfilePicture(member) }}" />

   <div class="full-person-description">
     <p>Member {{ member.PersonID }}:</p>
     <p>{{ member.ParliamentaryName }}</p>
     <p>{{ membersService.getBirthDate(member) }}</p>
   </div>

   <div class="full-person-extra-data">
     <h3>Parties</h3>
     <hr>
     <ul>
       <li *ngFor="let party of partiesService.getPersonParties(personId)">{{ party }}</li>
     </ul>
   </div>

   <div class="full-person-extra-data">
     <h3>Websites</h3>
     <hr>
     <ul>
       <li *ngFor="let website of websiteService.getWebsiteUrls(personId)">
         <a [href]="website" target="_blank">{{ website }}</a>
       </li>
     </ul>
   </div>
 </div>
```

Code 12: Full Person html