

TP 4 : Framework Angular (Routing)

Ce lab se focalise sur le système de routage dans le Framework Angular afin de permettre la navigation à travers les différentes pages de notre application. Angular propose le module *RouterModule* disponible dans la librairie *@angular/router*. Ce module a un objectif simple : permettre d'avoir des URLs compréhensibles qui reflètent l'état de notre application, et déterminer pour chaque URL quels composants initialiser et insérer dans la page.

The image displays three screenshots of an Angular application demonstrating routing:

- Top Left:** A browser window at `localhost:4200/list` showing the "Liste des sessions de formation" page. It lists three sessions: "Web" (MEAN), "Mobile" (Ionic), and "Web" (NodeJS). Each session card includes details like dates, duration, and availability, along with buttons for "Details", "Inscription", and "Partager".
- Top Right:** A browser window at `localhost:4200/admin/list` showing the same "Liste des sessions de formation" page but with an additional sidebar menu on the left containing icons for a list, a plus sign, and a pencil. The session cards now include "Suppression" and "Modification" buttons.
- Bottom Left:** A browser window at `localhost:4200/admin/edit/1` showing the "Ajouter une session de formation" page. It features a form with fields for "Name" (set to Mobile), "Track" (set to Ionic), "Date" (21/06/2018), "Days" (3), "Location" (Paris), and "Participants" (16). An "Enregistrer" button is at the bottom.
- Bottom Right:** A browser window at `localhost:4200/admin/edit/1` showing the "Modifier une session de formation" page. It features a form with fields for "Name" (set to Web), "Track" (set to MEAN), "Date" (jj/mm/aaaa), "Days" (3), "Adress" (Lyon), and "Participants" (0). A "Modifier" button is at the bottom.

Etapes d'implémentations

1. Faire d'abord les modifications nécessaires pour que l'application bootstrap le module *AppModule* par défaut (au lieu d'*AdminModule*). Dans une application Angular, on configure un seul module comme *BrowserModule* (c'est l'*EntryPoint* du notre app).

- a. Importer le module *BrowserModule* dans le fichier *app.module.ts*.

```
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports: [
    BrowserModule,
  ],
```

- b. Modifier le type du module *AdminModule* à *CommonModule* (au lieu de *BrowserModule*)

```
import { CommonModule } from '@angular/common';
@NgModule({
  CommonModule,
  FormsModule,
},
```

- c. Modifier la page *index.html* pour charger par défaut le composant *root* (*AppComponent*) du module *Admin*.

```
<app-root></app-root>
```

- d. Modifier le fichier *main.ts* pour charger le *AppModule* par défaut.

```
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

2. Création et configuration des Routes (table du routage) du module *AppModule*.

- a. Module *AppModule* (Fichier *app.module.ts*)

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { SessionItemComponent } from './session-item/session-item.component';
import { SessionItemListComponent } from './session-item-list/session-item-list.component';
import { InscriptionDisabledDirective } from './inscription-disabled.directive';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
```

```
import {RouterModule, Routes} from '@angular/router';
const appRoutes: Routes = [
  {
    path: 'list',
    component: SessionItemListComponent},
  {
    path: 'admin',
    loadChildren: './admin/admin.module#AdminModule'
  },
  { path: '', redirectTo: '/list', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent }
];
@NgModule({
  declarations: [
    AppComponent,
    SessionItemComponent,
    SessionItemListComponent,
    InscriptionDisabledDirective,
    PageNotFoundComponent,
  ],
  imports: [RouterModule.forRoot(
    appRoutes,
    { enableTracing: true }
  ),
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Commentaires :

On définit une première table de routage au niveau du module *AppModule*. Pour cela, nous allons lister dans le tableau de type *Routes* de notre module, les *Route* de notre application. Une *Route* est l'association d'un *path* (partie d'une URL) et d'un composant. Ce composant sera intégré dans la balise `<router-outlet>` (voir plus tard).

La propriété *loadChildren* indique au routeur de récupérer les routes du Module *AdminModule* lorsque l'utilisateur navigue vers l'espace Admin, puis fusionne les deux configurations de routeur et, enfin, active les composants nécessaires.

- b. Créer le composant *PageNotFound*.
- c. Pour inclure les routes, ajouter la directive *RouterOutlet* dans le composant *AppComponent* (Fichier *app.component.html*).

```
<section>
  <router-outlet></router-outlet>
</section>
```

- d. Créer les liens de navigation avec la directive *RouterLink*. Dans le template HTML du composant *SessionItemListComponent*, ajouter le lien hypertextes Admin (avec l'icône *admin.png*) pour accéder à l'espace administrateur (Fichier *session-item-list.component.html*).

```

<header>
  <h1>Liste des sessions de formation </h1>
  <p class="description">Suivre les sessions que je
  veux participer.</p>
</header>
<section>
  <app-session-item *ngFor="let sessionItem of
  sessionItems" [session]="sessionItem"
  [ngClass]="{'session-web':
  sessionItem.name === 'Web', 'session-mobile':
  sessionItem.name === 'Mobile'}">
  </app-session-item>
</section>
<footer>

  <a routerLink="/admin" routerLinkActive="active">
    
  </a>
</footer>

```

Commentaires :

Contrairement à une application Web classique, il n'est pas possible d'utiliser l'attribut *href* dans vos balises lien `<a>` (cela entraîne un rechargement de la page et une perte de notre contexte JavaScript). Angular nous fournit donc une directive en remplacement : *RouterLink*. Cette directive attendant un tableau contenant un path et des paramètres optionnels (*query paramaters, fragment, ...*).

Une fois nos liens déclarés, il est nécessaire d'indiquer à Angular où sera chargé le contenu du lien. Cela se fait simplement à l'aide de la directive `<router-outlet>` qui va accueillir le contenu de la route.

3. On passe maintenant à la création et la configuration des Routes du module *AdminModule*.

- a. Créer la deuxième table du routage dans *AdminModule* (Fichier *admin.modue.ts*)

```

import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { SessionItemComponent } from '../session-item/session-
item.component';
import { SessionItemListComponent } from '../session-item-
list/session-item-list.component';
import { SessionAddFormComponent } from '../session-add-
form/session-add-form.component';
import { SessionEditFormComponent } from '../session-edit-
form/session-edit-form.component';
import { AdminComponent } from '../admin/admin.component';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';
const adminRoutes: Routes = [
  {
    path: '',
    component: AdminComponent,
    children: [

```

```

    { path: 'add', component: SessionAddFormComponent },
    { path: 'edit/:id', component:
SessionEditFormComponent },
    { path: 'list', component: SessionItemListComponent
},
    { path: '', redirectTo: 'list', pathMatch: 'full' }
  ],
}
];
@NgModule({
  imports: [RouterModule.forChild(adminRoutes),
    CommonModule,
    FormsModule
  ],
  declarations: [
    SessionItemComponent,
    SessionItemListComponent,
    SessionAddFormComponent,
    SessionEditFormComponent,
    AdminComponent
  ],
  providers: [],
  bootstrap: [AdminComponent]
})
export class AdminModule { }

```

- b. Créer une barre de navigation dans le composant *AdminComponent*. Ajouter la directive *RouterOutlet* et l'ensemble des liens nécessaires en utilisant la directive *RouterLink*. (Fichier *admin.component.html*)

```

<nav>
  <a routerLink="./list" routerLinkActive="active">
    
  </a>
  <a routerLink="./add" routerLinkActive="active">
    
  </a>
  <a [routerLink]="['./edit', 1]" routerLinkActive="active">
    
  </a>
</nav>
<section>
  <router-outlet></router-outlet>
</section>

```

Commentaires :

La directive `[routerLink]="['./edit', 1]"` signifie qu'on va modifier la première session par défaut. D'où le paramètre *id* (comme indiqué dans la route *edit* dans la question) va recevoir la valeur 1 par défaut. Pour faire cela, vous devez implémenter les étapes **c** et **d** suivantes.

- c. Ajouter la fonction *getSession()* dans le service *FakeSessionItemService* qui permet de retourner une session.

```
getSession(id: number) {
  return SESSIONITEMS[id - 1];
}
```

- d. Modifier le Fichier TS du composant *SessionEditFormComponent* (fichier *session-edit-form.component.ts*) afin de récupérer la première session de formation. La récupération de la session à travers son ID sera implémentée dans la méthode *ngOnInit()* en appelant la fonction *getSession(ID)* du service *FakeSessionItemService*.

```
import { Component, OnInit } from '@angular/core';
import { Session } from '../session';
import { FakeSessionItemService } from '../fake-session-item.service';
@Component({
  selector: 'app-session-edit-form',
  templateUrl: './session-edit-form.component.html',
  styleUrls: ['./session-edit-form.component.css']
})
export class SessionEditFormComponent implements OnInit {

  session: Session ;
  tracks = ['MEAN', 'Angular', 'NodeJS', 'Android', 'Swift', 'Xamarin'];

  constructor(private sessionItemService:
FakeSessionItemService) { }
  ngOnInit() {
    this.session = this.sessionItemService.getSession(1);
  }
  editSession(sessionItem) {
    console.log(sessionItem);
  }
}
```

4. Tester toutes les routes du module AppModule (list, pagenotfound) et AdminModule (list, add, et edit)
5. Cette question s'intéresse au routage avec transfert des paramètres dans la directive RouterLink. Dans la liste de formations affichée dans l'espace Admin, le clic sur le lien *Modification* permet de modifier les données relatives à la session de formation.
 - a. Modifier le template HTML du composant *SessionItemComponent* en ajoutant la directive *RouterLink* : associer la route *edit* au lien *Modification* et spécifier l'ID de la session comme paramètre.

```
<a [routerLink]="['../edit', session.id]"
routerLinkActive="active" class="inscrire">
  Modification
</a>
```

- b. Modifier le Fichier TS du composant *SessionEditFormComponent* (fichier *session-edit-form.component.ts*) afin de récupérer la session qu'on veut mettre à jour à partir de l'ID reçu. La récupération de la session à travers son

ID sera implémentée dans la méthode *ngOnInit()* en appelant la fonction *getSession(ID)* du service *FakeSessionItemService*.

```
import { Component, OnInit } from '@angular/core';
import { Session } from '../session';
import { FakeSessionItemService } from '../fake-session-
item.service';
import { ActivatedRoute } from '@angular/router';
import { SessionHttpService } from '../session-http.service';
import { Observable } from 'rxjs';
@Component({
  selector: 'app-session-edit-form',
  templateUrl: '../session-edit-form.component.html',
  styleUrls: ['../session-edit-form.component.css']
})
export class SessionEditFormComponent implements OnInit {
  id;
  private sub: any;
  session: any;
  tracks = ['MEAN', 'Angular',
    'NodeJS', 'Android', 'Swift', 'Xamarin'];
  constructor(private route: ActivatedRoute, private
sessionItemService: FakeSessionItemService,
    private sessionhttp: SessionHttpService) { }
  ngOnInit() {
    this.sub = this.route.params.subscribe(params => {
      this.id = params['id']; // (+) converts string 'id' to a
number
    });
    console.log('Session ID ' + this.id.toString());
    this.session = this.sessionItemService.getSession(this.id);
  }
  editSession(sessionItem) {
    console.log(sessionItem);
  }
}
```