LE ROUTING AVEC IONIC

Le composant Router d'angulaire permet la navigation de page en page dans une application Ionic.

Le routeur d'Angular étant capable d'interpréter n'importe quelle URL permet :

- De faire passer des données d'une page à une autre via l'URL
- De lier une URL à une page de l'application et rediriger l'utilisateur directement vers la page de l'application au clic
- De faire de la redirection lors des événements sur les boutons ou les éléments d'un formulaire.

INTEGRATION DANS LE PROJET IONIC

Le service pour la gestion des routes se trouve dans la bibliothèque ***@angular/router***. Il devra être importé dans tous les projets Ionic comme suit :

import { RouterModule, Routes } from '@angular/router';

Le fonctionnement du service ressemble à celui des navigateurs, quand l'URL de l'application change, le routeur essaie de trouver une correspondance afin de déterminer le composant (page) à afficher.

Les routes sont créées dans les fichiers d'extension ***routing-module.ts*** et enregistrées grâce à la méthode ***RouterModule.forRoot***() ou ***RouterModule.forChild***()

Chaque route mappe une URL à une page. Le slash de début sur l'URL pourra être omis. Le routeur se chargera de générer l'url au bon format.

L'ordre d'enregistrement des routes est important. En effet le routeur utilise la première correspondance de route trouvée.

Par convention, nous enregistrerons toujours en premier lieu les URL statiques, sans paramètre, suivie par la route associée à l'url vide (la route par défaut).

Ajoutons une page à notre projet qui va permettre à l'administrateur d'ajouter un plat.

 Créons la nouvelle page de puis Ionic cli

```
PS D:\Enseignements\ESP\IONIC\Sources\app> ionic generate page ajouter
> ng.cmd generate page ajouter --project=app
CREATE src/app/ajouter/ajouter-routing.module.ts (351 bytes)
CREATE src/app/ajouter/ajouter.module.ts (479 bytes)
CREATE src/app/ajouter/ajouter.page.html (126 bytes)
CREATE src/app/ajouter/ajouter.page.spec.ts (668 bytes)
CREATE src/app/ajouter/ajouter.page.ts (260 bytes)
CREATE src/app/ajouter/ajouter.page.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (538 bytes)
[OK] Generated page!
```

- Supprimons la ligne suivante dans le fichier ***app-routing*** afin d'intégrer manuellement notre route au niveau du tab :

```
{
    path: 'ajouter',
    loadChildren: () => import('./ajouter/ajouter.module').then( m => m.AjouterPag
eModule)
  }
```

- Enregistrons la route de notre page dans le fichier sous-système de routage associé aux plats ou au niveau tab-***routing.module*** après l'avoir déplacer dans le sous-dossier sur les plats :

```
{
    path: 'ajouter',
    loadChildren: () => import('./ajouter/ajouter.module').then( m => m.AjouterPag
eModule)
  }
```

AJOUT DE PARAMETRE A UNE ROUTE

Pour associer une donnée arbitraire à une route, il faut l'enregistrer au niveau de la route.

Nous allons créer dans notre projet une page pour voir modifier les informations sur un plat.

Pour cela après avoir générer la page avec la commande :

ionic generate page plats/modifier

Nous allons mettre à jour la route associée comme suit au niveau du routing

```
const routes: Routes = [
  {
    path: '',
    component: PlatsPage,
  },
  {
    path: 'ajouter',
    loadChildren: () => import('./ajouter/ajouter.module').then( m => m.AjouterPageModule)
  },
  {
    path: 'modifier/:id',
    loadChildren: () => import('./modifier/modifier.module').then( m => m.ModifierPageModule)
  }
];
```

Dans cet exemple, pour accéder à la page de modification, nous serons obligés de fournir l'id du plat. Nous allons revenir dans la suite sur la récupération de cette information depuis le composant.

NAVIGATION ENTRE COMPOSANT

L'idée est de permettre à l'utilisateur de naviguer sur les différentes pages de l'application.

Nous allons essayer à partir de la page d'affichage des plats de donner la possibilité à l'utilisateur de cliquer sur un bouton pour ajouter un plat.

DEPUIS LA VUE HTML

Pour cela, nous allons ajouter un bouton au niveau de la page comme suit :

```
<ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button routerLink="/tabs/plats/ajouter">
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>
```

La directive ***routerLink*** au niveau de la balise ***ion-fab*** avec l'url renseignée permet au routeur de faire référence aux autres associées à d'autres pages.

DEPUIS LA CLASSE COMPONENT

Sur chaque ligne au niveau de la liste des plats ajoutons un événement au clic de la ligne afin de rediriger l'utilisateur vers la page de modification. Pour cela, nous allons ajouter la directive click sur les items de liste au niveau de la page d'affichage des plats comme suit :

```
<ion-list-item>
    <ion-item *ngFor="let plat of plats" (click)="modifier(plat.id)">
        <ion-label> {{plat.nom}} </ion-label>
        <td> {{plat.prix}} </td>
    </ion-item>
  </ion-list-item>
```

Puis allons définir ce que la fonction ***modifierPlat*** fait. Cette fonction prend en entrée l'identifiant du plat et devra rediriger l'utilisateur vers la page de modification :

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { Plats } from '../models/plats';

@Component({
  selector: 'app-plats',
  templateUrl: 'plats.page.html',
  styleUrls: ['plats.page.scss']
})
export class PlatsPage {

  plats = [
    new Plats(
      1,
      'Tiép',
      500,
      'test'
    ),
    new Plats(
      2,
      'Mafé',
      1000,
      'test'
    )
  ]
  constructor(
    private router: Router
```

```
  ) { }

  modifier(id: number): void {
    this.router.navigateByUrl('/tabs/plats/modifier/' + id);
  }
}
```

Ici, nous avons utilisé le service ***Router*** pour faire la redirection vers la page de modification des informations sur un plat. La route associée étant ***/tabs/plats/modifier/:id*** , on a fait recours à la méthode navigateByUrl en donnant les paramètres attendus en entrée.

A la place, on pouvait utilisé, la méthode navigate comme suit :

```
    this.router.navigate(['/tabs/plats/modifier', id])
```

RECUPERATION DES INFORMATIONS SUR UNE ROUTE [i]

Lors d'une navigation, après avoir appliqué les redirections, le routeur crée un **RouterStateSnapshot**.

RouteStateSnapshot est une structure de données immuable représentant l'état du routeur à un moment particulier dans le temps.

L'URL de la route ainsi que les paramètres transmis sont disponible dans un service du routeur nommé ActivatedRoute.

ActivatedRoute fournit un accès aux **observables** url, params, data, queryParams et fragments.

Pour l'utiliser, il faudra importer le service depuis ***@angular/router*** et demandé à angular d'injecter le service dans le composant en y faisant référence via un attribut privée comme suit :

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-modifier',
```

```
  templateUrl: './modifier.page.html',
  styleUrls: ['./modifier.page.scss'],
})
export class ModifierPage implements OnInit {

  id: number;
  constructor(
    private stateRoute: ActivatedRoute
  ) {
      this.id = Number.parseInt(this.stateRoute.snapshot.paramMap.get('id'));
   }

  ngOnInit() {
  }

}
```

La récupération du paramètre devient simple et se fait par invocation de l'attribut snapshot du service

```
    this.stateRoute.snapshot.paramMap.get('id');
```

[i] https://vsavkin.com/routeur-angular-comprendre-létat-du-routeur-5e15e729a6df