VII-Routing-2

PARAMÈTRE DE ROUTE

Dans cette partie, nous allons modifier notre application.
On souhaite à présent créer une fiche par étudiant et accéder à cette fiche en clique sur un étudiant de la liste.
Afin d'afficher une page par élève, il sera nécessaire de créer des liens avec des paramètres:
exemple : ...../presences/1
Dans ce genre de cas, on choisira plutôt de créer une route avec paramètre.

Ouvrons : **`app-routing.module.ts`** et ajoutons ce code

```
const routes: Routes = [
  { path: "presences", component: MonPremierComponentComponent },
  { path: "presences/:id", component: EleveDetailComponent },
];
```

L'utilisation des deux-points : après le slash / indique que la suite du lien correspond à un paramètre

Il nous faut aussi ajouter des modifications à **`service/data.service.ts`**

code > mon-projet > src > app > service > TS data.service.ts > ...

```ts
export class DataService {
  presenceTab = [
    {
      id: 1,
      name: "Emmanuelle",
      status: "Présent",
    },
    {
      id: 2,
      name: "Robin",
      status: "Absent",
    },
    {
      id: 3,
      name: "Dominique",
      status: "Présent",
    },
  ];

  getStudent(id: number) {
    const students = this.presenceTab.find((s) => {
      return s.id === id;
    });
    return students;
  }
}
```

On ajoute un id pour chaque élève

On a également créé une fonction getStudent()
Nous verrons son utilisation plus tard dans le cours

Nous allons ajouter ce lien dans chaque bloc de la liste
Modifier le code de **`mon-premier-component.component.html`**

code > mon-projet > src > app > mon-premier-component > mon-premier-component.component.html > ...

```
1  <h2>Liste de présence</h2>
2  <li class="list-group-item" *ngFor="let student of presenceTab">
3    <p>Etudiant : {{ student.name }} -- Statut : {{ student.status }}</p>
4
5    <a routerLink="/presences/{{ student.id }}">
6      Voir fiche : {{ student.id }} - {{ student.name }}
7    </a>
8
9  </li>
```

On a créé une lien vers la fiche de l'étudiant et on utilise l'interpolation dans la boucle pour insérer le bon paramètre "id" dans chaque routerLink.

Les données proviennent du service. Nous les avons donc importer depuis **mon-premier-component.component.ts**

Il nous faut à présent créer le template de la fiche des eleves
Commençons par créer une nouveau composant : eleve-detail

Ajoutez-y le code suivant: **`eleve-detail.component.html`**

code > mon-projet > src > app > eleve-detail > <> eleve-detail.component.html > ...

```
1  <h2>{{ name }} Details</h2>
2  <p>id: {{ id }}</p>
3  <p>Status: {{ status }}</p>
4
```

eleve-detail.component.ts

code > mon-projet > src > app > eleve-detail > eleve-detail.component.ts > EleveDetailComponent > status

```ts
import { Component, OnInit } from "@angular/core";
import { ActivatedRoute } from "@angular/router";
import { Location } from "@angular/common";
import { DataService } from "../service/data.service";

@Component({
  selector: "app-eleve-detail",
  templateUrl: "./eleve-detail.component.html",
  styleUrls: ["./eleve-detail.component.css"],
})
export class EleveDetailComponent implements OnInit {
  id: string;
  name: string;
  status: string;
  constructor(
    private route: ActivatedRoute,
    private dataService: DataService,
    private location: Location
  ) {}

  ngOnInit(): void {
    const id = this.route.snapshot.params["id"];
    this.id = id;
    this.name = this.dataService.getStudent(+id).name;
    this.status = this.dataService.getStudent(+id).status;
  }

  goBack(): void {
    this.location.back();
  }
}

```

(ligne 16) On a injecté `ActivatedRoute`, importé depuis `@angular/router`, afin de récupérer le fragment `id` de l'URL

(ligne 24-25) Dans `ngOnInit()`, on utilise l'objet `snapshot` qui contient les paramètres de l'URL et on a attribué le paramètre id à la variable name et status

La méthode `getStudent()` que nous avons ajouté dans le service permet de récupérer l'identifiant de l'URL et de l'utiliser pour récupérer l'appareil correspondant

Connaissance Typescript
Puisqu'un fragment d'URL est forcément de type `string`, et que la méthode `getAppareilById()` prend un nombre comme argument, on utilise `+` avant `id` pour changer le type de la variable en nombre.

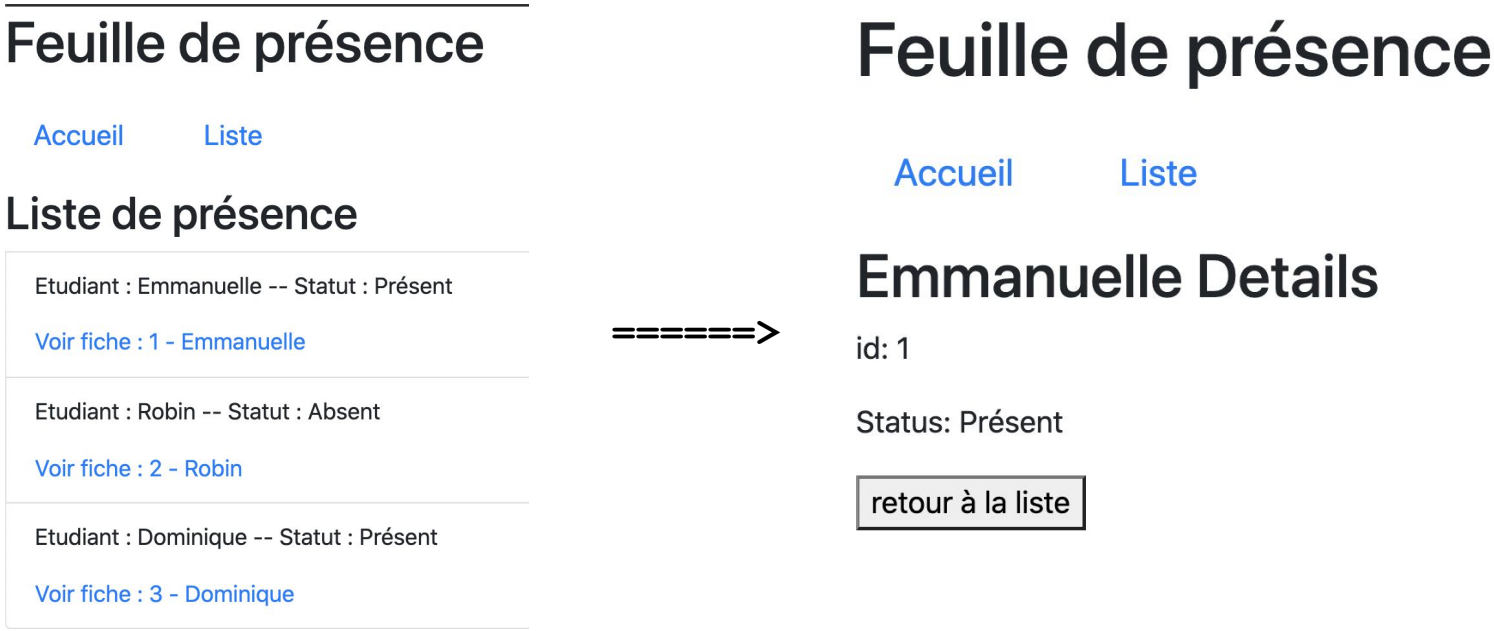
(ligne 3) Le location est un service d'angular pour interagir avec le navigateur. Nous allons l'utiliser pour revenir à la vue où nous avons navigué.
Pour cela nous avons importé Location depuis @angular/common

(ligne 18) Nous avons également besoin d'injecter "Location" dans le constructeur

(ligne 28) On utilise une Location dans une fonction qu'on insérera dans un bouton pour l'actionner

Dans **`eleve-detail.component.html`**, ajouter un bouton utilisant la fonction goBack() pour revenir à la page d'avant

```
<button (click)="goBack()">retour à la liste</button>
```



REDIRECTION

En cas d’erreur, il sera parfois nécessaire de rediriger un utilisateur, vers une page 404 par exemple

Dans le cas de notre application, on aimerait que tous liens non existant soit redirigé vers la page contenant la liste

Pour cela, ils nous faut ajouter une redirection dans **`app-routing.module.ts`**

```
6  const routes: Routes = [
7    { path: "presences", component: MonPremierComponentComponent },
8    { path: "presences/:id", component: EleveDetailComponent },
9    { path: "**", redirectTo: "/presences" },
10 ];
```

A présent, quand on entrera un chemin dans la barre de navigation qui n'existe pas, nous seront redirigé vers l’url /presences du site soit la liste des étudiants.