PARTIE 6: LE ROUTEUR

LES URLS DANS UNE SPA

- Notre application ne fonctionne pour le moment qu'avec une seule URL: http://localhost:4200/
- Pour rappel, des URLs différentes sont utiles: partage d'un lien vers le détail d'un exercice donné, historique dans le navigateur, ...
- Le routeur permet la navigation en interprétant l'URL d'un navigateur comme une instruction pour modifier la vue
- C'est un service optionnel intégré à Angular, comme `HttpClient`

NAVIGATION

- Dans une application Angular, la navigation s'effectue grâce à un système de **routes** à définir
- Une route est une association entre un composant et une URL
- Lorsque l'URL est modifiée, le **routeur** effectue le rendu du composant demandé

PREMIÈRE ROUTE

- Dans le répertoire `app`, recherchez le fichier `app.routes.ts` qui exporte le tableau de **routes**
- Créez la route vers la "page" d'accueil

```
import { Routes } from "@angular/router";
import ...
export const routes: Routes = [
    {
        //http://localhost:4200
        path:'',
        component: Exercises ,
        title: 'Exercices - Accueil' //texte dans l'onglet
    }
]
```

LE SERVICE `Router`

- Comme `HttpClient`, le service `Router` doit être ajouté au niveau de l'application.
- Le tableau `routes` est transmis pour configurer le routeur.

```
import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    ...
    provideRouter(routes)
    ...
  ]
};
```

LA DIRECTIVE `RouterOutlet`

- dans le **template** de `App`, cette directive indiquera où afficher le composant correspondant à la route actuelle
- `RouterOutlet` doit être ajouté dans les imports

```
<main>
  @for (message of tabMessages(); track message.id) {
    <iut-alert [message]="message" (closeAlert)="removeAlert($event)" />
  }
  <router-outlet />
</main>
```

Une **directive** est un sélecteur (élément ou attribut) angular qui permet de manipuler le DOM de manière dynamique et réactive.

PREMIÈRES ROUTES

- Ajoutez une route pour gérer les routes non définies avec un nouveau composant `PageNotFound`

```
import { Routes } from "@angular/router";
import ...
export const routes: Routes = [
    {
        path:'',
        component: Exercises ,
        title: 'Exercices - Accueil' //texte dans l'onglet
    },
    /* insérer ici toutes les autres routes */

    { /* la dernière du tableau */
        path: '**',
        component: PageNotFound,
        title: 'Exercices - Page inexistante'
    }
]
```

LAZY LOADING D'UN COMPOSANT

- Il est possible de demander à angular de créer un fichier javascript (*chunk*) séparé pour ce qui concerne un composant particulier.
- Ce fichier javascript ne sera chargé que si la route correspondante est utilisée.

```
{
  path: '**',
  loadComponent: () =>
    import('./page-not-found/page-not-found')
  .then(m => m.PageNotFound),
  title: 'Exercices - Page inexistante'
}
```

LA DIRECTIVE `RouterLink`

- Pour créer des liens internes à notre application, il faut utiliser la balise HTML **<a>** avec `routerLink` qui s'utilise comme un attribut

```
<a routerLink="/">Revenir sur la page d'accueil</a>
```

Si un chemin débute par /, la nouvelle URL est calculée à partir de l'URL de base de l'application. Sinon, elle est calculée à partir de l'URL courante.

La directive `routerLink` signifie que la navigation est interceptée et gérée par le routeur côté client. Elle ne provoque donc ni requête au serveur, ni rechargement complet de la page.

NAVIGUER PROGRAMMATIQUEMENT

- Il est possible de naviguer depuis le code typescript en utilisant les méthodes `navigateByUrl` ou `navigate` du service `Router`.

```
private readonly router = inject(Router);
...

  this.router.navigateByUrl('/')
  // ou this.router.navigate(['/']);
```

`Router` est un service à injecter. `RouterOutlet` et `RouterLink` sont des directives à importer.

EXERCICE

- Créez un lien dans le composant `Exercises` vers une nouvelle "page" (composant `CreateExercise` et route associée de *path* 'create'). Elle permettra de créer un nouvel exercice.
- Dans un premier temps, créez juste dans le *template* de ce composant un titre "Ajouter un exercice" et un lien "Retour sur la page d'accueil".

ROUTES DYNAMIQUES (1/2)

- Les paramètres de routes (`:nomParam`) permettent de créer des routes dynamiques
- Nous souhaitons à présent ajouter un lien pour chaque exercice qui permette de naviguer vers une nouvelle "page" Pour le modifier. Cette "page" correspond à un nouveau composant `EditExercise` dont le *template* affiche juste, pour le moment, le titre "Modifier un exercice".
- Pour chaque exercice, il existera donc une URL spécifique pour sa modification.
- Le chemin correspondant sera du type `edit/:id`

```
{
  path: 'edit/:id',
  component: EditExercise,
  title: 'Exercices - Modification'
}
```

ROUTES DYNAMIQUES (2/2)

- La valeur associée à `routerLink` peut comporter des parties dynamiques pour associer à chaque item d'exercice un lien vers la page pour le modifier.

```
<!-- code dans exercices.html -->
<a [routerLink]="['/edit', exo.id]">
    Modifier
</a>
```

ou

```
<!-- code dans exercices.html -->
<a [routerLink]="`/edit/${exo.id}`">
    Modifier
</a>
```

UTILISER UNE ICÔNE

```
<!-- code dans exercices.html -->
<a [routerLink]="['/edit', exo.id]" aria-label="Modifier l'exercice">
    <i class="fa fa-edit" aria-hidden="true" title="Modifier l'exercice"></i>
</a>
```

- L'icône est obtenu en utilisant *fontawesome*. Pour rappel, si vous n'avez pas encore programmé la suppression d'un article (entraînement `HttpClient`):

```
npm install --save-exact @fortawesome/fontawesome-free@7.0.0
```

```
/* dans app/style.css */
/* You can add global styles and also import other style files */
@import '@fortawesome/fontawesome-free/css/all.css';
```

RÉSULTAT

Avec un peu de style :

```css
a:has(.fa-edit) {
  background-color: lightgrey;
  border: solid 2px black;
  padding: 0.1rem 0.3rem 0.05rem;
  border-radius: 15%;
  margin: 0.2rem;
}
```



PARAMÈTRE DE ROUTE ET `input`

- Depuis Angular 16, et comme avec Vue.js, il est possible de lier automatiquement des `inputs` aux paramètres de route de même nom.
- Cela nécessite une configuration du router

```
// dans les providers de l'application
provideRouter(routes, withComponentInputBinding())
```

```
// code dans edit-exercise.ts
readonly id = input.required(); // la valeur provient de la route
```

EXERCICE

Faire en sorte que le titre du *template* de `EditExercise` affiche maintenant "Modifier l'exercice d'identifiant X", avec X la valeur de l'identifiant de l'exercice pour lequel la modification est souhaitée, quand on est sur une page de modification.