PW 2019 – Sebastien Pittion – TP 2 : Connexion à TMDB

Aide (Installation)

Terminer d'abord le TP 1.

S'assurer que le code est bien présent sur un dépôt Git distant.

Ensuite il faudra implémenter la communication avec l'API TMDB (version 3) pour récupérer et afficher des vraies données de films. Une clé d'API a priori utilisable est épinglée sur Slack (**ebb02613ce5a2ae58fde00f4db95a9c1**), autrement il est possible de s'en procurer une en créant un compte.

Pour commencer, créer un fichier nommé **tmdb.ts** sous **src/** dans le projet. Ce fichier doit exporter une constante nommée **apiKey** contenant la clé d'API que vous souhaitez utiliser. Ce fichier doit être ignoré dans Git (fichier **.gitignore**) et donc être recréé manuellement sur chaque PC ayant cloné le projet.

Dans votre composant **Home** (TypeScript), en partant du principe que vous avez les interfaces suivantes,

```
export interface Movie {
  backdrop_path: string;
  id: number;
  overview: string;
  poster_path: string;
  release_date: string;
  title: string;
}

interface TMDBReponse {
  results: Movie[];
}
```

codez la méthode suivante permettant d'interroger l'API TMDB de façon générique en se basant sur le service HttpClient fourni par Angular ainsi que les Promise JavaScript et async/await :

```
private async askTMDB(api: string, params: object): Promise<Movie[]> {
  const { results } = await this.http.get<TMDBReponse>(
    `https://api.themoviedb.org/3/${api}/movie`,
    { params: { api_key: apiKey, ...params } }
  ).toPromise();
  return results;
}
```

Attention : ne pas oublier d'injecter (injection de dépendances) le **HttpClient** sous le nom **http** ni d'ajouter le **HttpClientModule** au bon endroit et d'importer votre constante **apiKey**…

Codez maintenant une méthode nommée searchMovies, dont la signature est la suivante,

```
private searchMovies(search: string): Promise<Movie[]>
```

celle-ci appellera la méthode **askTMDB** en passant les paramètres nécessaires pour interroger respectivement l'API search de TMDB.

Modifiez maintenant le typage de votre liste de films pour que cette dernière soit **Promise<Movie[]>** au lieu de simplement **Movie[]**.

De fait, vous devez également modifier la méthode **getMovies** (implémentée au TP1) pour assigner à cette liste :

- Soit une Promise déjà résolue contenant une liste vide (si la valeur de la barre de recherche contient moins de 3 caractères), **Promise.resolve([])**.
- Soit un appel à la méthode **searchMovies** en passant en paramètre la valeur de la barre de recherche.

Enfin, pour pouvoir afficher le contenu d'une **Promise** côté template (HTML), il vous faudra utiliser le AsyncPipe fourni par Angular aussi bien sur **ngFor** qu'**ngIf**.

Attention : le cas du **ngIf** nécessitant une syntaxe un peu particulière, je vous la donne et la commenterai pour tout le monde.

```
*ngIf="!(movies | async)?.length"
```