chapitre 9 les services

Création d'un service

Le code que nous avons écrit dans les components pour traiter les données et les injecter dans les vues (template) doit être factorisé dans un service. En effet, on doit toujours appliquer les principes SOLID lors du développement d'une application et séparer les rôles. Un component est une sorte de contrôleur, il ne doit pas avoir trop de responsabilités. Le rôle d'un componenent est d'organiser les données et de les passer à la vue uniquement.

Nous allons donc créer un service pour le traitement logique des données. Puis nous injecterons par dépendance ce dernier dans les components. Angular faisant alors une instance unique du service.

Créez le service :

```
ng generate service album
```

AlbumService

Le décorateur @Injectable défini dans la classe indique également que cette classe peut recevoir d'autre(s) service(s) par injection de dépendance. Le paramètre provideIn permet de préciser si vous souhaitez un service général ou spécifique à un module en particulier.

```
import { Injectable } from '@angular/core';

// Une classe injectable est un service et peut recevoir d'autre(s) service(s)
@Injectable({
  providedIn: 'root' // injecter de manière globale
})
export class AlbumService {

  constructor() { }
}
```

Exercice 13

Créez maintenant dans le service album.service.ts les méthodes suivantes :

- getAlbums()„ elle retournera tous les albums.
- getAlbum(id: string), elle retournera un album.
- getAlbumList(id : string), elle retournera la liste d'un album

Une fois le code écrit dans le service, il faudra importer celui-ci dans les components concernés.

Injectez le service dans le component :

Un service sera injecté de manière privée, c'est de la responsabilité du component de l'utiliser pour mettre en place la préparation des données avant de les passer à la vue (template). Une vue ne devrait pas avoir à utiliser directement le service, c'est pour cela que vous mettrez ce dernier en « private ».

Refactorisez votre code pour ne passer que par votre service pour récupérer vos données.

Vous ne devez plus avoir de logique de traitement des données dans vos components.

Récupérez dans la méthode ngOnInit du component AlbumsComponent les albums.

Rappelons que la méthode ngOnInit est appelée à l'initialisation du component, elle est donc adaptée pour faire passer les données à la vue (template).

Exercice 14.1

Retournez les albums par ordre décroissant de durée.

Utilisez la fonction native de JS« sort » pour ordonner les albums et travaillez dans la méthode getAlbums du service.

Exercice 14.2

Créez une méthode count. Elle retournera le nombre d'albums. Contrôlez le bon fonctionnement de cette méthode en faisant un console.log dans le component AlbumsComponent.