Chapitre 7 Component détails d'un albums

Nous allons maintenant voir comment communiquer dans un autre sens, entre la vue et le code TypeScript, c'est-à-dire du Template vers le Component (code TypeScript). C'est de l'Event data binding, c'est du one-way également. Il va nous permettre de réagir en fonction d'un événement dans la page Web.

Exercice 10 (détails d'un album)

Dans cette partie nous allons "cabler" les deux Components AlbumsComponent et AlbumDetailsComponent afin qu'ils puissent communiquer entre eux. L'exercice 11 permettra de finaliser l'affichage des détails d'un album c'est-à-dire sa liste de chansons.

L'événement (click) sera placé dans l'élément div.card. Créez un bouton cliquable. Ainsi en sélectionnant un album dans l'application on récupèrera un album dans le TypeScript.

Définissez une méthode onSelect par exemple dans le TypeScript. Faites un console.log pour contrôler que tout marche bien.

Affichage d'un album dans la partie div.col-sm-4.

Créez un autre Component AlbumDetailsComponent. Ce sera un component enfant du component AlbumsComponent. C'est dans ce dernier que vous afficherez le détail d'un album.

```
ng g c album-details
```

Communication du parent vers l'enfant : lorsque vous sélectionnez un album avec la méthode onSelect(album) depuis le component AlbumsComponent vous devez faire passer celui-ci (one-way data binding) au Component enfant AlbumDetailsComponent.

Créez une variabe **selectedAlbum** dans le component AlbumsComponent. Lorsque vous sélectionnez un album, ce dernier est assigné à cette variable.

Nous allons maintenant faire passer cette dernière à notre Component enfant AlbumDetailsComponent.

```
<div class="col-sm-4 video">
    <!-- AlbumDetails communication parent vers enfant -->
    <app-album-details [album]="selectedAlbum" ></app-album-details>
</div>
```

La syntaxe [album] permet de faire passer la variable album au component enfant, c'est une « property binding » c'est du "one-way in".

La propriété «album» sera liée au Component enfant AlbumDetailsComponent comme suit (utilisation du décorateur @Input(), attention à bien importer ce décorateur dans le component enfant) :

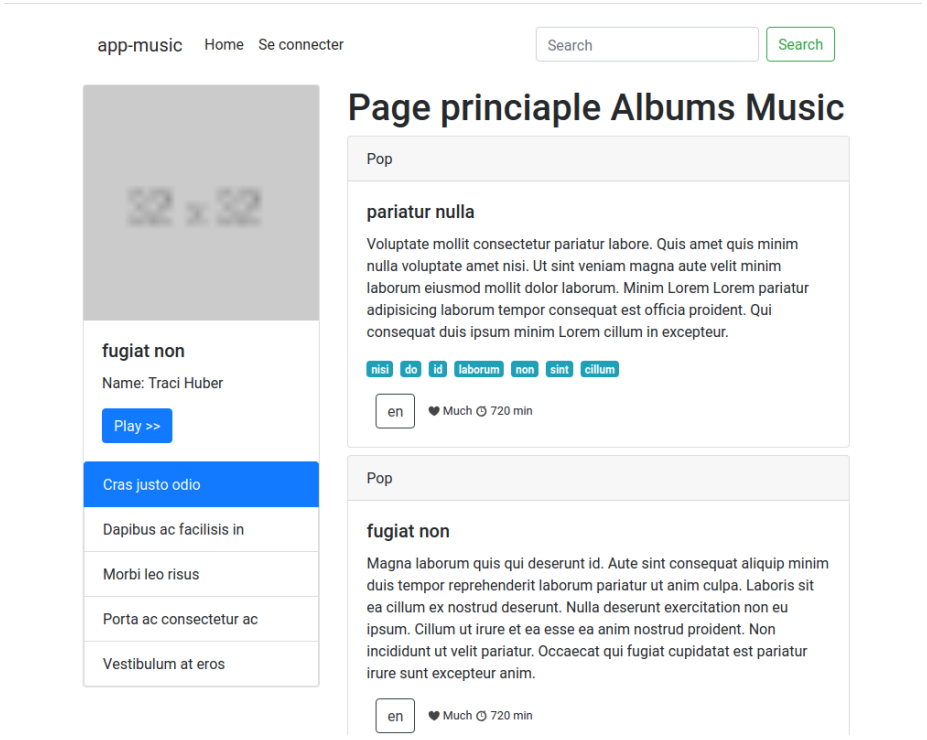
```
// Component enfant
// la classe Input est nécessaire
import { Component, OnInit, Input } from '@angular/core';
import { Album } from '../album';

@Component({
  selector: 'app-album-details',
  templateUrl: './album-details.component.html',
  styleUrls: ['./album-details.component.scss']
})
export class AlbumDetailsComponent implements OnInit {

  // Classe Input permet de récupérer les data de l'enfant
  // album est liée à une entrée [album] du parent dans le sélecteur
  @Input() album: Album;

  constructor() { }

  ngOnInit() {
    console.log(this.album); // pour l'instant c'est undefined ... C'est normal
  }

}
```

Vous pouvez dans la vue du Component AlbumDetailsComponent afficher un album sélectionné de la manière suivante, uniquement pour contrôler que tout est en place, **le pipe json** est à utiliser pour le débogage :

```
{{ album | json}}
```

Maintenant que vous savez comment récupérer une propriété depuis un Component parent vers son enfant, affichez dans un premier temps l'album en respectant le visuel suivant : (vous penserez également à ajouter du CSS pour mettre le bon curseur lorsqu'on survole un album cliquable. Faites bien attention à ce dernier point : les CSS sont propres à un component donné) :

Ne vous occupez pas de l'aspect dynamique de la liste des albums pour l'instant.



Exercice 11 (gestion des titres de l'album)

Récupérez les données dans le component AlbumDetailsComponent afin d'afficher le détail des listes des chansons par album. Chaque album possède un identifiant unique « id ». Utilisez dans cet exercice cette propriété.

Life cycle

C'est dans la méthode ngOnChanges du component AlbumDetailsComponent que l'on va récupérer la propriété album du component parent. La méthode ngOnChanges est déclenchée à chaque fois que l'on clique sur un album dans le component parent lié avec la propriété [album] et à l'instanciation du component.

La liste de chaque album devrait maintenant s'afficher en fonction d'AlbumLists et cela de manière dynamique.