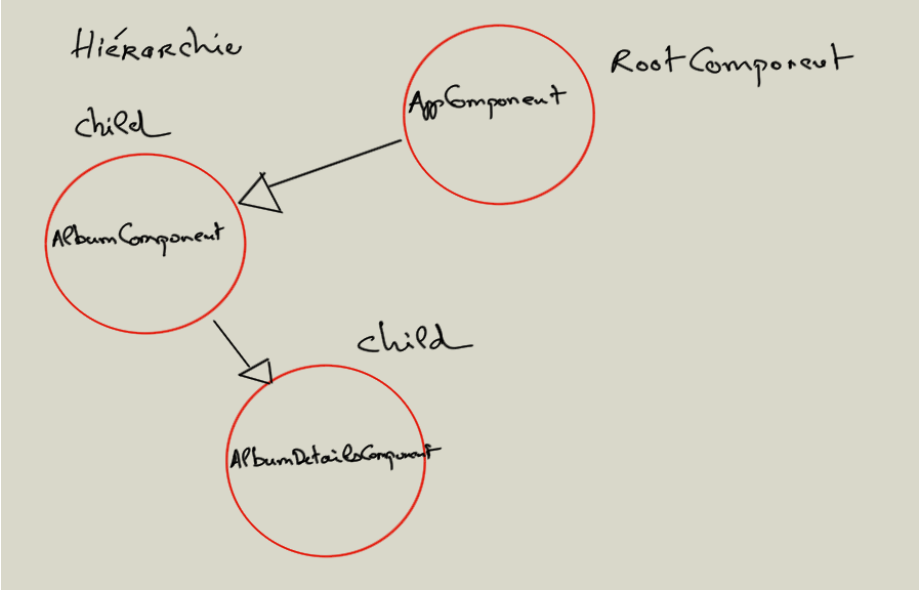
Chapitre 6 Introductions aux Components

Présentation

Nous allons créer deux Components : albums et albums-details.

Un component est une classe TypeScript décorée par le décorateur @Component. Un component est associé normalement à son template et à ses CSS, par défaut isolés des autres components.

Nous allons créer deux Components : AlbumsComponent et AlbumDetailsComponent. Il faudra voir les components comme des éléments hiérarchiques dans la page HTML, le component AlbumsComponent est un enfant du Component AppComponent. Ce dernier étant le Component racine. Notons, mais nous en reparlerons, que ces components sont encapsulés dans l'unique module existant pour l'instant de notre application : AppModule.



Création du component AlbumsComponent

Tapez dans la console la ligne de code suivante. Nous utilisons angular-cli pour créer le component :

```
# la commande classique
ng generate component albums

# la même commande plus consise peut être écrite
ng g c albums
```

Vous devriez voir maintenant le dossier albums créé dans le dossier src, nous

allons travailler dans la classe Component pour importer les données puis les passer à la vue : template HTLM associé.

Angular a automatiquement ajouté à son AppModule ce component:ouvrez le fichier app.module.ts pour vérifier cet import.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { AlbumsComponent } from './albums/albums.component';

@NgModule({
  declarations: [
    AppComponent,
    AlbumsComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Création des données

Nous allons maintenant créer les données d'exemple, dans un fichier pour l'instant. Pour cela nous allons créer un fichier albums.ts avec une classe pour définir nos données :

```
ng generate class Album
```

Ce fichier c'est normalement créé à la racine du dossier src.

Exercice 8

Créez les attributs de la classe Album sachant que nos albums sont constitués dans leur structure par les éléments suivants (vous devez définir chaque attribut avec son type précis, sans préciser la visibilité dans cette classe) :

id	ref	name	title	description	duration	status	url ?	tags ?	like ?
string	string	string	string	string	number	string	string	Array	string

Récupérez la source des mock-albums.ts que vous placerez dans le dossier src/app.

Importez les données dans le component

Nous allons maintenant importer les données dans le component AlbumsComponent.

Les sélecteurs

Dans le component parent AppComponent vous allez placer le sélecteur :

```
<app-albums></app-albums>
```

Il permet d'insérer le component enfant AlbumsComponent dans son component parent.

Dans le component AlbumsComponent (fichier albums.component.ts) vous trouverez le nom du sélecteur à utiliser dans le component AppComponent :

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-albums', // sélecteur à mettre dans le parent
  templateUrl: './albums.component.html',
  styleUrls: ['./albums.component.scss']
})
export class AlbumsComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Dans le fichier app.component.html vous devez alors mettre votre sélecteur pour « câbler » les deux templates parent et enfant :

```
<div class="container-fuild principal">
  <nav class="navbar navbar-expand-lg navbar-light">
  </nav>
<app-albums></app-albums>
</div>
```

Vous devriez voir maintenant le message suivant dans la page principale de l'application : albums works!

Nous allons nous occuper de l'affichage des données à partir du Component AlbumsComponent. Nous devons importer les données dans un premier temps dans ce component.

Interpolation

L'interpolation permet de définir des données dans le TypeScript que l'on peut par la suite afficher dans le template à l'aide des doubles accolades {{ ... }}. C'est du data-binding one-way (TypeScript vers le template). Créez un titre à la page « titlePage » dans le AlbumsComponent puis utilisez-le dans le template :

```
import { Component, OnInit } from '@angular/core';

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

  titlePage: string = "Page princiaple Albums Music";

  constructor() { }

  ngOnInit() {
  }

}
```

Dans le template albums.component.html écrivez :

```
<h1>{{ titlePage }}</h1>
```

Affichage des données albums

Il faut maintenant importer les données et le type Album dans le component AlbumsComponent. Voyez le code suivant, (le fait d'appeler ALBUMS directement dans l'attribut albums permet la récupération de toutes les données) :

```
import { Component, OnInit } from '@angular/core';

// Importez la définition de la classe et les albums
import { Album } from '../album';
import { ALBUMS } from '../mock-albums';
```

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

  titlePage: string = "Page principale Albums Music";
  albums: Album[] = ALBUMS;

  constructor() { }

  ngOnInit() {
  }

}
```

Exercice 9

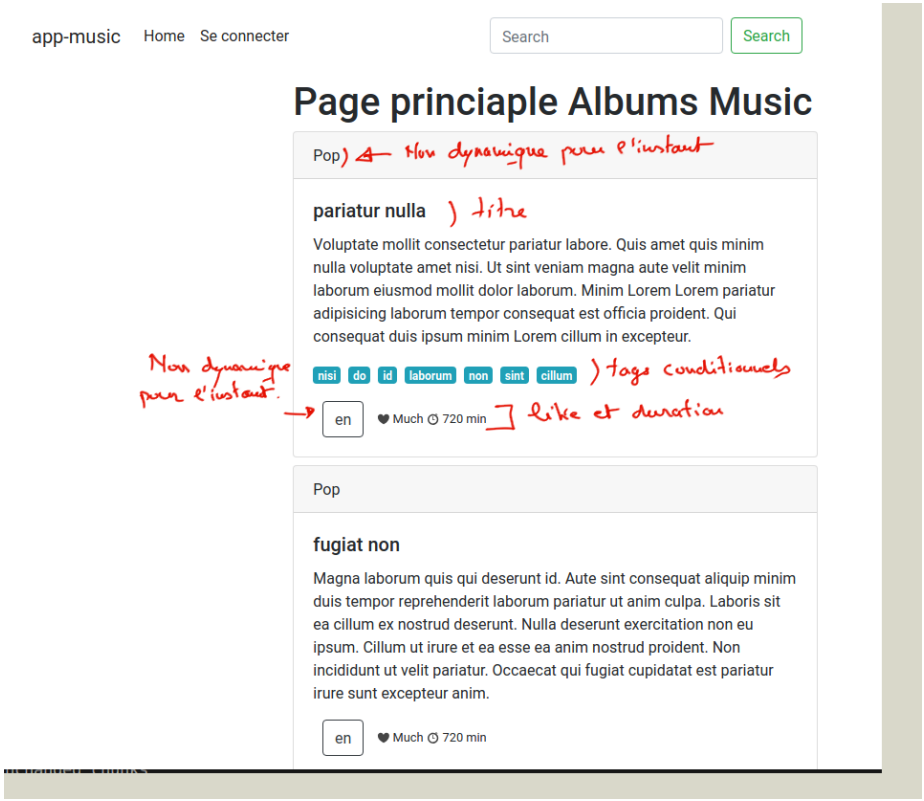
Vous allez maintenant, en vous aidant de la directive structurelle *ngFor (voir un exemple ci-dessous) et en vous aidant du code HTML dans le fichier app.html, afficher dans le div.col-sm-8 music les différents albums.

Voici un exemple d'affichage :la variable items contiendrait des éléments de type { id:number ; name : string ; } par exemple, alors il faudrait écrire dans le template :

```
<ul *ngIf="items" class="list-inline">
  <li class="list-inline-item" *ngFor="let item of items">
    <span class="badge badge-info">{{ item.id }}</span> {{ item.name }}
  </li>
</ul>
```

Pour l'affichage ou non du like et de la durée de l'album vous utiliserez la directive *ngIf. Reportez-vous à la documentation en ligne : documentation directives.

Faites la même chose avec les albums. Voici le résultat en ligne que vous devriez obtenir. Attention, parfois vous avez des tags facultatifs dans les données. Affichez ces derniers comme dans l'exemple qui suit :


app-music
Home
Se connecter
Search
Search
Page principale Albums Music
Pop
Non dynamique pour l'instant
pariatur nulla
titre
Voluptate mollit consectetur pariatur labore. Quis amet quis minim nulla voluptate amet nisi. Ut sint veniam magna aute velit minim laborum eiusmod mollit dolor laborum. Minim Lorem Lorem pariatur adipisicing laborum tempor consequat est officia proident. Qui consequat duis ipsum minim Lorem cillum in excepteur.
Non dynamique pour l'instant.
nisi
do
id
laborum
non
sint
cillum
tags conditionnels
en
Much
720 min
like et duration
Pop
fugiat non
Magna laborum quis qui deserunt id. Aute sint consequat aliquip minim duis tempor reprehenderit laborum pariatur ut anim culpa. Laboris sit ea cillum ex nostrud deserunt. Nulla deserunt exercitation non eu ipsum. Cillum ut irure et ea esse ea anim nostrud proident. Non incididunt ut velit pariatur. Occaecat qui fugiat cupidatat est pariatur irure sunt excepteur anim.
en
Much
720 min