Activity 3: Part 2

Ryan Coon

CST-391

Professor Bobby Estey

July 16, 2023

Markdown URL: <https://github.com/rcoon1/CST-391/blob/main/Topic%203/activity3part2.md>

**Screenshots:**

A screenshot of a computer

Description automatically generated

Here you can see the initial application page running.

A group of people standing in front of a table

Description automatically generated

Here is the GCU Homepage actively clicked from a button within the application.

A screenshot of a music collection

Description automatically generated

Here is the create album form.

A screenshot of a music collection

Description automatically generated

Here is information inputted into the create album form.

A screenshot of a music collection

Description automatically generated

Here is the result of creating a new album and submitting the form.

A screenshot of a computer

Description automatically generated

Here is the result of clicking on the about button showing a pop up box with the version number.

**Research:**

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

import exampledata from '../../data/sample-music-data.json';

import { Artist } from '../models/artists.model';

import { Album } from '../models/albums.model';

import { Track } from '../models/tracks.model';

//set injectable to get to all services in project

@Injectable({ providedIn: 'root' })

export class MusicServiceService {

private readonly artists: Artist[] = [];

private readonly albums: Album[] = [];

constructor() {

this.createArtists();

this.createAlbums();

}

//create artists constructor, shows The Beatles in list as default

private createArtists(): void {

this.artists.push(new Artist(0, 'The Beatles'));

}

//create albums constructor, will show beatles list as default

private createAlbums(): void {

exampledata.forEach((data: any) => {

if (data.artist === 'The Beatles') {

const tracks = data.tracks.map((trackData: any) => new Track(trackData.id, trackData.number, trackData.title, trackData.lyrics, trackData.video));

const album = new Album(data.id, data.title, data.artist, data.description, data.year, data.image, tracks);

this.albums.push(album);

}

});

}

//will get artists from data set and return them to the view in a list

public getArtists(): Artist[] {

return this.artists;

}

//will get the albums in dataset, and return them to the view in a list

public getAlbums(artist: string): Album[] {

return this.albums;

}

//will get albums that were searched for and return them to the view in a list

public getAlbum(artist: string, id: number): Album | undefined {

const album = this.albums.find((a) => a.Artist === artist && a.Id === id);

if (album) {

const tracks = album.Tracks.map((track) => new Track(track.Id, track.Number, track.Title, track.Lyrics, track.Video));

return new Album(album.Id, album.Title, album.Artist, album.Description, album.Year, album.Image, tracks);

}

return undefined;

}

//this is where we create a new album and push it into the data set

public createAlbum(album: Album): void {

this.albums.push(album);

}

//here we update albums within the data set

public updateAlbum(album: Album): void {

const index = this.albums.findIndex((a) => a.Id === album.Id);

if (index !== -1) {

this.albums.splice(index, 1, album);

}

}

//constructor to delete an album from within the data set

public deleteAlbum(id: number, artist: string): void {

const index = this.albums.findIndex((a) => a.Id === id);

if (index !== -1) {

this.albums.splice(index, 1);

}

}

}