**Item.ts**

export class Item {

constructor(public id: number, public desc : string, public price : number,

public quantity : number) {

}

}

**Dbservice.service.ts**

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, of } from 'rxjs';

import { catchError, map, tap } from 'rxjs/operators';

import { ReturnStatement } from '@angular/compiler';

import {Item} from './Item';

const httpOptions = {

headers: new HttpHeaders({ 'Content-Type': 'application/json' })

};

const Url = 'http://localhost:1234/SkillMapperRestApi/item';

@Injectable({

providedIn: 'root'

})

export class DBserviceService {

//private Url = 'http://localhost:1234/SkillMapperRestApi/item';

constructor(public http: HttpClient) { }

getItem(): Observable<Item[]> {

return this.http.get<Item[]>(Url +'/listitems')

.pipe(

tap(product => console.log('fetched items')),

catchError(this.handleError('getItem', []))

);

}

getSingleItem(id: number): Observable<Item> {

return this.http.get<Item>(Url +'/find/item/${id}').pipe(

tap(\_ => console.log(`fetched item id=${id}`)),

catchError(this.handleError<Item>(`getItem id=${id}`))

);

}

addItem (item: Item): Observable<Item> {

console.log("service"+item.id);

return this.http.post<Item>(Url +'/additems', item, httpOptions).pipe(

catchError(this.handleError<Item>('addItem'))

);

}

updateItem(item: Item): Observable<any> {

return this.http.put(Url +'/updateitems', item, httpOptions).pipe(

catchError(this.handleError<any>('updateItem'))

);

}

deleteItem (id:number): Observable<Item> {

const url = `${Url}/deleteitems/${id}`;

return this.http.delete<Item>(url, httpOptions).pipe(

catchError(this.handleError<Item>('deleteItem'))

);

}

private handleError<T> (operation = 'operation', result?: T) {

return (error: any): Observable<T> => {

// TODO: send the error to remote logging infrastructure

console.error(error); // log to console instead

// Let the app keep running by returning an empty result.

return of(result as T);

};

}

}

**App.module.ts**

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { InsertComponent } from './insert/insert.component';

import {DBserviceService} from './dbservice.service'

import {HttpClientModule} from '@angular/common/http';

import { UpdateComponent } from './update/update.component';

import { GetComponent } from './get/get.component';

import { DeleteComponent } from './delete/delete.component';

import {FormsModule, ReactiveFormsModule} from '@angular/forms';

@NgModule({

declarations: [

AppComponent,

InsertComponent,

UpdateComponent,

GetComponent,

DeleteComponent

],

imports: [

HttpClientModule,

FormsModule,

ReactiveFormsModule,

BrowserModule,

AppRoutingModule

],

providers: [DBserviceService],

bootstrap: [AppComponent]

})

export class AppModule { }

**app-routing.module.ts**

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

import { Routes, RouterModule } from '@angular/router';

import {InsertComponent} from './insert/insert.component';

import { DeleteComponent } from './delete/delete.component';

import { UpdateComponent } from './update/update.component';

import { GetComponent } from './get/get.component';

const routes: Routes = [

{ path :'insert', component:InsertComponent,data:{title:'Insert Products'}},

{ path :'update', component:UpdateComponent,data:{title:'Update Products'}},

{ path :'delete', component:DeleteComponent,data:{title:'Delete Products'}},

{ path :'get', component:GetComponent,data:{title:'Retrive Products'}}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**insert.component.ts**

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

import {Item} from '../item';

import {DBserviceService} from '../dbservice.service';

import { FormGroup, FormBuilder, Validators } from '@angular/forms';

@Component({

selector: 'app-insert',

templateUrl: './insert.component.html',

styleUrls: ['./insert.component.css']

})

export class InsertComponent implements OnInit {

items :Item[];

insertForm: FormGroup;

submitted = false;

constructor(public dbservice:DBserviceService,private fb: FormBuilder) { }

ngOnInit() {

this.insertForm = this.fb.group({

id: ['', Validators.required],

desc: ['', Validators.required],

quantity: ['', Validators.required],

price: ['', [Validators.required, Validators.email]],

});

}

// convenience getter for easy access to form fields

get f() { return this.insertForm.controls; }

onSubmit() {

this.submitted = true;

// stop here if form is invalid

if (this.insertForm.invalid) {

return;

}

// display form values on success

alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.insertForm.value, null, 4));

}

add(id:number,desc:string,price:number,quantity:number): void {

console.log(id);

id=id;

desc=desc.trim();

price=price;

quantity=quantity;

this.dbservice.addItem({id,desc,price,quantity} as Item)

.subscribe(item => {

//this.items.push(item);

console.log('inside add');

});

}

}

**Insert.component.html**

<p>insert works!</p>

<h2>Item insertion</h2>

<form [formGroup]="insertForm" (ngSubmit)="onSubmit()">

<div>

<label>itemId:</label>

<input #id onfocus="this.value=''" formControlName="id" placeholder = "enter id" [ngClass]="{ 'is-invalid': submitted && f.id.errors }">

<div \*ngIf="submitted && f.id.errors" class="invalid-feedback">

<div \*ngIf="f.id.errors.required">id is required</div>

</div>

</div>

<div>

<label>description:</label>

<input #desc onfocus="this.value=''" formControlName="desc" placeholder = "enter description" [ngClass]="{ 'is-invalid': submitted && f.desc.errors }" >

<div \*ngIf="submitted && f.desc.errors" class="invalid-feedback">

<div \*ngIf="f.desc.errors.required">Description is required</div>

</div>

</div>

<br/>

<div >

<label>price:</label>

<input #price onfocus="this.value=''" formControlName="price" placeholder = "enter price" [ngClass]="{ 'is-invalid': submitted && f.price.errors }">

<div \*ngIf="submitted && f.price.errors" class="invalid-feedback">

<div \*ngIf="f.price.errors.required">Price is required</div>

</div>

</div>

<br/>

<div >

<label>quantity:</label>

<input #quantity onfocus="this.value=''" formControlName="quantity" placeholder = "enter quantity" [ngClass]="{ 'is-invalid': submitted && f.quantity.errors }">

<div \*ngIf="submitted && f.quantity.errors" class="invalid-feedback">

<div \*ngIf="f.quantity.errors.required">quantity is required</div>

</div>

</div>

<br/>

<button (click)="add(id.value,desc.value,price.value,quantity.value);">

Add

</button>

</form>

**update.component.ts**

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

import {Item} from '../item';

import {DBserviceService} from '../dbservice.service';

import { Router } from '@angular/router';

import { FormGroup, FormBuilder, Validators } from '@angular/forms';

@Component({

selector: 'app-update',

templateUrl: './update.component.html',

styleUrls: ['./update.component.css']

})

export class UpdateComponent implements OnInit {

items :Item[];

updateForm: FormGroup;

submitted = false;

constructor(public dbservice:DBserviceService,private fb: FormBuilder) { }

ngOnInit() {

this.updateForm = this.fb.group({

id: ['', Validators.required],

desc: ['', Validators.required],

quantity: ['', Validators.required],

price: ['', [Validators.required, Validators.email]],

});

}

// convenience getter for easy access to form fields

get f() { return this.updateForm.controls; }

onSubmit() {

this.submitted = true;

// stop here if form is invalid

if (this.updateForm.invalid) {

return;

}

// display form values on success

alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.updateForm.value, null, 4));

}

update(id:number,desc:string,price:number,quantity:number): void {

console.log(id);

id=id;

desc=desc.trim();

price=price;

quantity=quantity;

this.dbservice.updateItem({id,desc,price,quantity} as Item)

.subscribe(item => {

//this.items.push(item);

console.log('update');

});

}

}

**Update.component.html**

<h2>Item Updation</h2>

<form [formGroup]="updateForm" (ngSubmit)="onSubmit()">

<div>

<label>itemId:</label>

<input #id onfocus="this.value=''" formControlName="id" placeholder = "enter id" [ngClass]="{ 'is-invalid': submitted && f.id.errors }">

<div \*ngIf="submitted && f.id.errors" class="invalid-feedback">

<div \*ngIf="f.id.errors.required">id is required</div>

</div>

</div>

<div>

<label>description:</label>

<input #desc onfocus="this.value=''" formControlName="desc" placeholder = "enter description" [ngClass]="{ 'is-invalid': submitted && f.desc.errors }" >

<div \*ngIf="submitted && f.desc.errors" class="invalid-feedback">

<div \*ngIf="f.desc.errors.required">Description is required</div>

</div>

</div>

<br/>

<div >

<label>price:</label>

<input #price onfocus="this.value=''" formControlName="price" placeholder = "enter price" [ngClass]="{ 'is-invalid': submitted && f.price.errors }">

<div \*ngIf="submitted && f.price.errors" class="invalid-feedback">

<div \*ngIf="f.price.errors.required">Price is required</div>

</div>

</div>

<br/>

<div >

<label>quantity:</label>

<input #quantity onfocus="this.value=''" formControlName="quantity" placeholder = "enter quantity" [ngClass]="{ 'is-invalid': submitted && f.quantity.errors }">

<div \*ngIf="submitted && f.quantity.errors" class="invalid-feedback">

<div \*ngIf="f.quantity.errors.required">quantity is required</div>

</div>

</div>

<br/>

<button (click)="update(id.value,desc.value,price.value,quantity.value);">

Update

</button>

</form>

**Get.component.ts**

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

import { DBserviceService } from '../dbservice.service';

import {Item} from '../item';

import { Router } from '@angular/router';

import { Observable } from 'rxjs';

@Component({

selector: 'app-get',

templateUrl: './get.component.html',

styleUrls: ['./get.component.css']

})

export class GetComponent implements OnInit {

data: Item[] = [];

constructor(private dbservice:DBserviceService, private router: Router) { }

ngOnInit() {

this.dbservice.getItem()

.subscribe((res: any) => {

this.data = res;

console.log(this.data);

}, err => {

console.log(err);

});

}

delete(id:number): void {

console.log(id);

id=id;

this.dbservice.deleteItem(id)

.subscribe(res => {

//this.items.push(item);

this.router.navigate(['/get']);

console.log('item deleted');

});

}

}

**Get.component.html**

<h2>Item List</h2>

<table>

<thead>

<tr>

<th>ItemId</th>

<th>Description</th>

<th>Price</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let items of data ">

<td>{{items.id}}</td>

<td>{{items.quantity}}</td>

<td>{{items.price}}</td>

<td>{{items.desc}}</td>

<td><button> <a routerLink="/update">Update</a></button>

<button (click)="delete(items.id);">

Delete

</button>

</td>

</tr>

</tbody>

</table>

**Delete.component.ts**

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

import {Item} from '../item';

import {DBserviceService} from '../dbservice.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

selector: 'app-delete',

templateUrl: './delete.component.html',

styleUrls: ['./delete.component.css']

})

export class DeleteComponent implements OnInit {

data: Item[] = [];

constructor(public dbservice:DBserviceService,private router: Router){}

ngOnInit() {

}

delete(id:number,desc:string,price:number,quantity:number): void {

console.log(id);

id=id;

desc=desc.trim();

price=price;

quantity=quantity;

this.dbservice.deleteItem(id)

.subscribe(res => {

//this.items.push(item);

this.router.navigate(['/get']);

console.log('item deleted');

});

}

}

**Delete.component.html**

<p>delete works!</p>

<table>

<thead>

<tr>

<th>ItemId</th>

<th>Description</th>

<th>Price</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let items of data ">

<td>{{items.id}}</td>

<td>{{items.quantity}}</td>

<td>{{items.price}}</td>

<td>{{items.desc}}</td>

<td>

<button (click)="delete(id.value,desc.value,price.value,quantity.value);">

Delete

</button>

</td>

</tr>

</tbody>

</table>