**Docker Swarm:**

**Code:**

**Maven spring code🡪 backend**

**Application properties:**

Server.port=9090

**productBackendApplicatio.java**

package com;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication(scanBasePackages = "com")

public class ProductBackendApplication {

public static void main(String[] args) {

SpringApplication.run(ProductBackendApplication.class, args);

System.out.println("Server Started ....");

}

}

**ProductController.java**

package com.conntroller;

import java.util.ArrayList;

import java.util.List;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.bean.Product;

@RestController

@RequestMapping("product")

@CrossOrigin() // it help to enable to access the resources

public class ProductController {

@GetMapping(value = "allProduct",produces = MediaType.APPLICATION\_JSON\_VALUE)

public List<Product> getAllProduct() {

List<Product> listOfProduct = new ArrayList<Product>();

listOfProduct.add(new Product(1, "TV", 14000, "https://m.media-amazon.com/images/I/71L2wXXLthL.\_SL1500\_.jpg"));

listOfProduct.add(new Product(2, "Computer", 350000, "https://image.shutterstock.com/image-photo/modern-desktop-computer-wireless-keyboard-260nw-55573504.jpg"));

listOfProduct.add(new Product(3, "Laptop", 150000, "https://fdn.gsmarena.com/imgroot/news/22/04/apple-mac-studio-hands-on/-1200w5/gsmarena\_009.jpg"));

listOfProduct.add(new Product(4, "PenDrive",1200, "https://cdn.pixabay.com/photo/2015/07/20/19/50/usb-853230\_\_340.png"));

return listOfProduct;

}

}

**Product.java**

**package** com.bean;

**public** **class** Product {

**private** **int** pid;

**private** String pname;

**private** **float** price;

**private** String url;

**public** **int** getPid() {

**return** pid;

}

**public** **void** setPid(**int** pid) {

**this**.pid = pid;

}

**public** String getPname() {

**return** pname;

}

**public** **void** setPname(String pname) {

**this**.pname = pname;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) {

**this**.price = price;

}

**public** String getUrl() {

**return** url;

}

**public** **void** setUrl(String url) {

**this**.url = url;

}

@Override

**public** String toString() {

**return** "Product [pid=" + pid + ", pname=" + pname + ", price=" + price + ", url=" + url + "]";

}

**public** Product(**int** pid, String pname, **float** price, String url) {

**super**();

**this**.pid = pid;

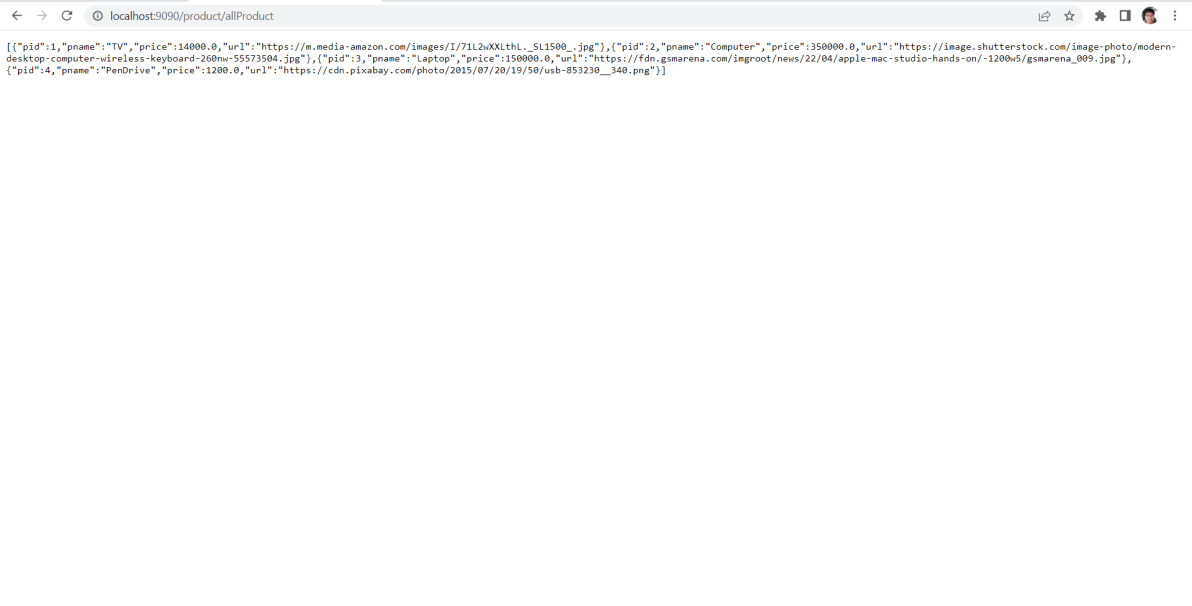
**this**.pname = pname;

**this**.price = price;

**this**.url = url;

}

}



Mvn package

Create a Spring Maven Docker Image command :-

And

**add a Dockerfile code :-**

sudo docker build -t product-backend-app . -f Dockerfile //ubuntu command

docker build -t product-backend-app . -f Dockerfile

Angular Code :- {Frontend]

ng create a

**app.component.html :-**

<app-product></app-product>

**product.component.ts :-**

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

import { Product } from '../product';

import { ProductService } from '../product.service';

@Component({

selector: 'app-product',

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

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

})

export class ProductComponent implements OnInit {

products:Array<Product>=[];

constructor(public ps:ProductService) { }

ngOnInit(): void {

this.loadAllProduct();

}

loadAllProduct(){

this.ps.loadAllProduct().subscribe({

next:(result:any)=>this.products = result,

error:(error:any)=>console.log(error),

complete:()=>console.log("completed"),

})

}

}

**product.component.html :-**

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

import { Product } from '../product';

import { ProductService } from '../product.service';

@Component({

selector: 'app-product',

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

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

})

export class ProductComponent implements OnInit {

products:Array<Product>=[];

constructor(public ps:ProductService) { }

ngOnInit(): void {

this.loadAllProduct();

}

loadAllProduct(){

this.ps.loadAllProduct().subscribe({

next:(result:any)=>this.products = result,

error:(error:any)=>console.log(error),

complete:()=>console.log("completed"),

})

}

**Product.ts :-**

export class Product {

    constructor(public pid:number,

        public pname:string,

        public price:number,

        public url:string

        ){}

}

**Product.service.ts :-**

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

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

import { Observable } from 'rxjs';

import { Product } from './product';

@Injectable({

  providedIn: 'root'

})

export class ProductService {

  constructor(public http:HttpClient) { }

  loadAllProduct():Observable<Product[]>{

    return this.http.get<Product[]>("http://localhost:9090/product/allProduct");

  }

}

**App.module.ts** :-

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

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

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

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

import { ProductComponent } from './product/product.component';

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

@NgModule({

  declarations: [

    AppComponent,

    ProductComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,HttpClientModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

}

ng build

Command to create a image :-

**Dockerfile code :-**

FROM nginx

COPY dist/product-frontend/ /usr/share/nginx/html

docker build -t product-frontend-app . -f Dockerfile

Create a Docker compose to create a Docker Swarm ,and docker interconnected :-

**docker-compose.yml Code :-**

version: "3"

services:

product-backend:

image: product-backend-app

ports:

- "9090:9090"

networks:

- product-management-system

product-frontend:

image: product-frontend-app

ports:

- "80:80"

depends\_on:

- product-backend

networks:

- product-management-system

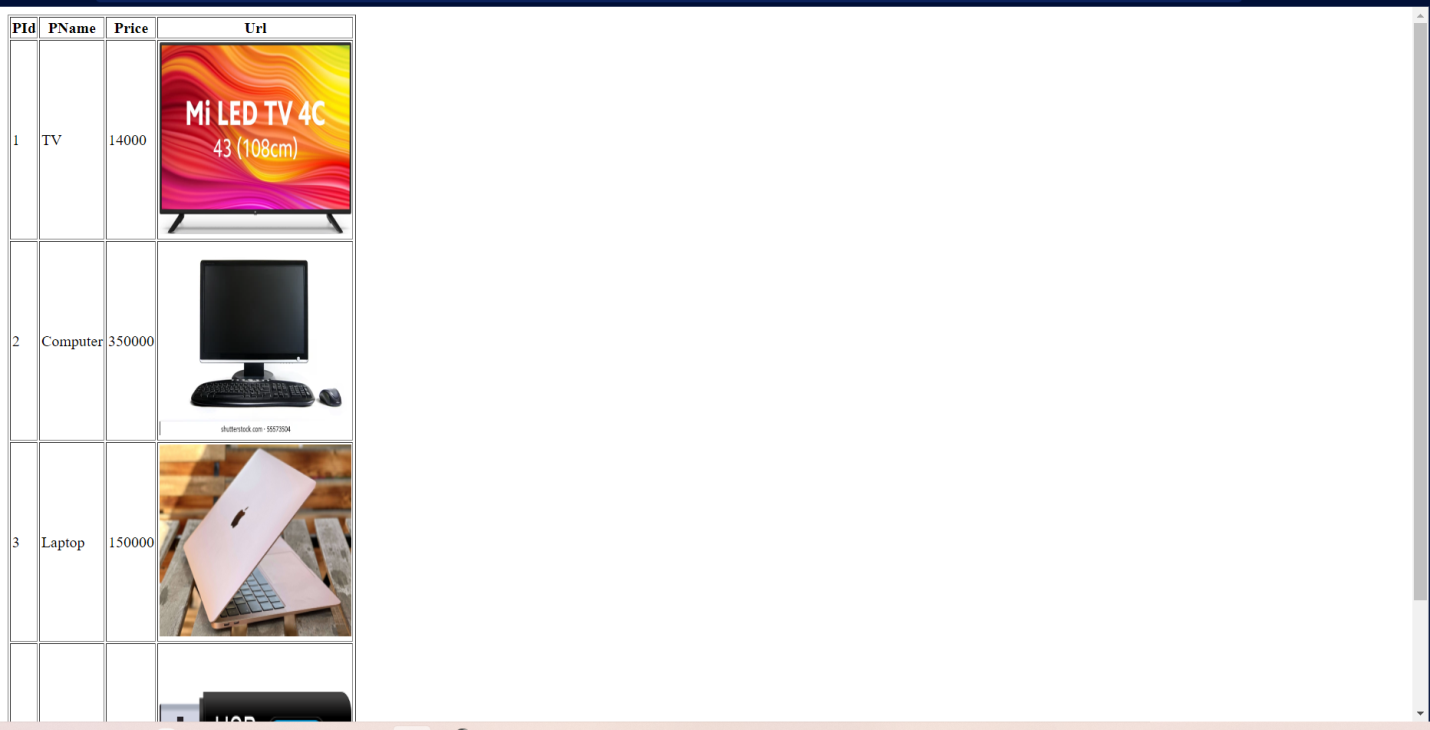
networks:

product-management-system

Start Docker -compose :-

docker-compose up

**screenshot**

****