**Angular 16 Mini Application**

I'll guide you through creating an Angular 16 application with product display and selection components using FakeStoreAPI, Bootstrap CSS, routing, and Docker deployment.

**Create the Angular Application**

First, let's set up the Angular project:

ng new angular-product-display --style=scss --routing=true

cd angular-product-display

**Install Required Packages**

npm install bootstrap @ng-bootstrap/ng-bootstrap

**Set Up Bootstrap**

Add Bootstrap to angular.json:

"styles": [

"node\_modules/bootstrap/scss/bootstrap.scss",

"src/styles.scss"

],

"scripts": [

"node\_modules/bootstrap/dist/js/bootstrap.bundle.min.js"

]

**Create Product Interface and Service**

Create src/app/models/product.ts:

export interface Product {

id: number;

title: string;

price: number;

description: string;

category: string;

image: string;

rating: {

rate: number;

count: number;

};

}

Create src/app/services/product.service.ts:

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

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

import { Observable } from 'rxjs';

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

@Injectable({

providedIn: 'root'

})

export class ProductService {

private apiUrl = 'https://fakestoreapi.com/products';

constructor(private http: HttpClient) { }

getProducts(): Observable<Product[]> {

return this.http.get<Product[]>(this.apiUrl);

}

getProduct(id: number): Observable<Product> {

return this.http.get<Product>(`${this.apiUrl}/${id}`);

}

}

**Create Product Components**

Generate components:

ng generate component components/product-list

ng generate component components/product-detail

Update src/app/components/product-list/product-list.component.ts:

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

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

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

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

@Component({

selector: 'app-product-list',

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

styleUrls: ['./product-list.component.scss']

})

export class ProductListComponent implements OnInit {

products: Product[] = [];

isLoading = true;

constructor(

private productService: ProductService,

private router: Router

) { }

ngOnInit(): void {

this.productService.getProducts().subscribe({

next: (products) => {

this.products = products;

this.isLoading = false;

},

error: (error) => {

console.error('Error fetching products:', error);

this.isLoading = false;

}

});

}

viewProduct(id: number): void {

this.router.navigate(['/products', id]);

}

}

Update src/app/components/product-list/product-list.component.html:

<div class="container mt-4">

<h2 class="mb-4">Product List</h2>

<div \*ngIf="isLoading" class="text-center">

<div class="spinner-border" role="status">

<span class="visually-hidden">Loading...</span>

</div>

</div>

<div \*ngIf="!isLoading" class="row">

<div \*ngFor="let product of products" class="col-md-4 mb-4">

<div class="card h-100">

<img [src]="product.image" class="card-img-top p-2" [alt]="product.title" style="height: 200px; object-fit: contain;">

<div class="card-body">

<h5 class="card-title">{{ product.title | slice:0:50 }}...</h5>

<p class="card-text">{{ product.price | currency }}</p>

<p class="card-text"><small class="text-muted">{{ product.category }}</small></p>

</div>

<div class="card-footer bg-transparent">

<button (click)="viewProduct(product.id)" class="btn btn-primary btn-sm">View Details</button>

</div>

</div>

</div>

</div>

</div>

Update src/app/components/product-detail/product-detail.component.ts:

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

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

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

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

@Component({

selector: 'app-product-detail',

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

styleUrls: ['./product-detail.component.scss']

})

export class ProductDetailComponent implements OnInit {

product: Product | undefined;

isLoading = true;

constructor(

private productService: ProductService,

private route: ActivatedRoute

) { }

ngOnInit(): void {

const id = this.route.snapshot.paramMap.get('id');

if (id) {

this.productService.getProduct(+id).subscribe({

next: (product) => {

this.product = product;

this.isLoading = false;

},

error: (error) => {

console.error('Error fetching product:', error);

this.isLoading = false;

}

});

}

}

}

Update src/app/components/product-detail/product-detail.component.html:

<div class="container mt-4" \*ngIf="product">

<div class="row">

<div class="col-md-6">

<img [src]="product.image" class="img-fluid" [alt]="product.title">

</div>

<div class="col-md-6">

<h2>{{ product.title }}</h2>

<p class="text-muted">{{ product.category }}</p>

<p class="h4">{{ product.price | currency }}</p>

<p>{{ product.description }}</p>

<div class="d-flex align-items-center mb-3">

<span class="badge bg-warning me-2">{{ product.rating.rate }} ★</span>

<small class="text-muted">{{ product.rating.count }} reviews</small>

</div>

<button class="btn btn-primary">Add to Cart</button>

<a routerLink="/" class="btn btn-outline-secondary ms-2">Back to Products</a>

</div>

</div>

</div>

<div \*ngIf="isLoading" class="text-center mt-5">

<div class="spinner-border" role="status">

<span class="visually-hidden">Loading...</span>

</div>

</div>

<div \*ngIf="!product && !isLoading" class="alert alert-warning mt-5">

Product not found. <a routerLink="/" class="alert-link">Go back to products</a>

</div>

**Set Up Routing**

Update src/app/app-routing.module.ts:

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

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

import { ProductListComponent } from './components/product-list/product-list.component';

import { ProductDetailComponent } from './components/product-detail/product-detail.component';

const routes: Routes = [

{ path: '', component: ProductListComponent },

{ path: 'products/:id', component: ProductDetailComponent },

{ path: '\*\*', redirectTo: '' }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**Update App Module**

Update src/app/app.module.ts:

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

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

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

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

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

import { ProductListComponent } from './components/product-list/product-list.component';

import { ProductDetailComponent } from './components/product-detail/product-detail.component';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

@NgModule({

declarations: [

AppComponent,

ProductListComponent,

ProductDetailComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule,

NgbModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**Update App Component**

Update src/app/app.component.html:

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<div class="container">

<a class="navbar-brand" routerLink="/">FakeStore</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" routerLink="/" routerLinkActive="active">Products</a>

</li>

</ul>

</div>

</div>

</nav>

<div class="container mt-4">

<router-outlet></router-outlet>

</div>

**Create Dockerfile**

Create Dockerfile in the root directory:

# Stage 1: Build the Angular app

FROM node:18-alpine as build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build --prod

# Stage 2: Serve the app with Nginx

FROM nginx:alpine

COPY --from=build /app/dist/angular-product-display /usr/share/nginx/html

COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

Create nginx.conf in the root directory:

server {

listen 80;

server\_name localhost;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

try\_files $uri $uri/ /index.html;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

**Create .dockerignore**

Create .dockerignore in the root directory:

node\_modules

dist

.git

.gitignore

Dockerfile

.dockerignore

\*.md

**Push to GitHub**

Initialize Git and push to GitHub:

git init

git add .

git commit -m "Initial commit with Angular product display app"

git branch -M main

git remote add origin https://github.com/yourusername/angular-product-display.git

git push -u origin main

**Build and Push Docker Image**

Build the Docker image:

docker build -t yourdockerhubusername/angular-product-display .

Push to Docker Hub:

docker push yourdockerhubusername/angular-product-display

**Run the Docker Container**

Run the container:

docker run -d -p 8080:80 yourdockerhubusername/angular-product-display

Now you can access the application at http://localhost:8080.