Angular Tutorial

Installation and Setup

npm install -g @angular/cli

ng new project-name

Links: https://angular.dev/tutorials/first-app
Links: https://angular.dev/tools/cli
Links: https://www.youtube.com/@Angular/playlists

Links: https://www.youtube.com/playlist?list=PL1w1q3fL4pmj9k1FrJ3Pe91EPub2_h4jF

Application Flow

[Point1]: Inside the index.html, we call the <app-root></app-root>. The index.html is auto linked to the app component.

[Point2]: The app.component.html will be the root of the application, it will contain three section the header, the body which will be a router routing different application and a footer

```
<!--Header-->
<app-header></app-header>
<!--Wrapper-->
<div class="wrapper">
<router-outlet></router-outlet>
</div>




<!--Footer-->
<app-footer></app-footer>
```

[Point3]: The app.component.ts will contain all the necessary imports for the selector used in the app.component.html

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { HomeComponent } from "./home/home.component";
import { HeaderComponent } from './layout/header/header.component';
import { FooterComponent } from './layout/footer/footer.component';

@Component({
selector: 'app-root',
standalone: true,
templateUrl: './app.component.html',
styleUrl: './app.component.scss',
imports: [RouterOutlet, HomeComponent,HeaderComponent,FooterComponent ]
})
export class AppComponent {
title = 'tutorial';
}
```

[Point4]: Create a file called types.ts in the src folder, then define this interfaces

```
import { HttpContext, HttpHeaders, HttpParams } from "@angular/common/http";

export interface Options {
headers?: HttpHeaders | { [header: string]: string | string[] };
observe?: 'body';
context?: HttpContext;
params?: HttpParams | { [param: string]: string | number | boolean |
ReadonlyArray<string | number | boolean> };
reportProgress?: boolean;
responseType?: 'json';
withCredentials?: boolean;
transferCache?: boolean;
}


export interface Products{
items:Product[];
total:number;
page:number;
perPage:number;
totalPages:number;
}


export interface Product{
id:number;
name:string;
price:string;
description:string;
image:string;
quantity:number;
rating:number;
category:string;
createdAt:string;
updatedAt:string;
deletedAt:string;
}


export interface PaginationParams{
[param: string]: string | number | boolean | ReadonlyArray<string | number |
boolean>
page:number;
perPage:number;

}
```

[Point5]: Create 2 services file to serve the api and the products
ng generate service api
ng generate service products

api.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Options } from '../../types';

@Injectable({
providedIn: 'root'
})
export class ApiService {

constructor(
private httpClient: HttpClient
) { }


get<T>(url: string, options: Options): Observable<T> {
return this.httpClient.get<T>(url, options) as Observable<T>;
}


//<T> means the api can accept any type unless uniquely defined like
//get<string> means the api accept only string






}
```

products.service.ts

```
import { Observable } from 'rxjs';
import { ApiService } from './api.service';
import { Injectable } from '@angular/core';
import { PaginationParams, Products } from '../../types';

@Injectable({
providedIn: 'root'
})
export class ProductsService {

constructor(private apiService:ApiService) { }


getProducts = (url:string, params:PaginationParams):Observable<Products> => {
return this.apiService.get(url, {
params,
responseType: 'json',
})
}



}
```

[Point6]: In your app.config.ts, add the configuration to use the services

```
import { ApplicationConfig } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
providers: [provideRouter(routes), provideHttpClient()]
};
```

[Point7]: Create a Home component ng generate component home

[Point8]: In your app.routes let the initial loading page point to the HomeComponent

```
import { Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';

export const routes: Routes = [
{
path: '',
component: HomeComponent
}


];
```

Project Flow

[Point 9]: Create a product component ng generate component product

[Point 10]: Since the Home.component.html is the root of the application, I will call the <app-product></app-product> selector in the home component and this is where things get interesting,

```
<div class="column">
<div class="row wrap gap-1 space-between">
<app-product *ngFor="let productData of productsDict"
[productData]="productData"></app-product>
</div>
</div>
```

```
Now let me explain it step by step. The productsDict is from the api Data define
in the home.component.ts

import { Component } from '@angular/core';
import { ProductsService } from '../services/products.service';
import { Product, Products } from '../../types';
import { ProductComponent } from '../component/product/product.component';
import { CommonModule } from '@angular/common';

@Component({
selector: 'app-home',
standalone: true,
imports: [ProductComponent,CommonModule],
templateUrl: './home.component.html',
styleUrl: './home.component.scss'
})
export class HomeComponent {

constructor(
private productService : ProductsService
){}

productsDict: Product[] = []


ngOnInit(){
this.productService.getProducts("http://localhost:3000/clothes",{page:0,
perPage:5}).subscribe((data:Products)=>{
this.productsDict = data.items

/*
Notes:

this.productsDict is a dictionary initialized as empty
from the Typescript type Products, the item field is define as an array of
products
items:Product[];

this.productsDict = data.items, the items field used as an interface to capture
the items from the backend is assigned to productDict

We subscribe to an observable

when i use data.[something], i can get all the field defined in the interface
Products
Which are
items:Product[];
total:number;
page:number;
perPage:number;
totalPages:number;
Because from here data:Products, data is of type Products. If i use any, i cant
get the fields

*/
})
}




}
```

simply put, the productDict is captured from the api, data.items is an array of product

Now let move to the productData.

The productData is just a variable defined in the product.component.ts which has the interface of Product as a type,

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

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

@Component({
selector: 'app-product',
standalone: true,
imports: [],
templateUrl: './product.component.html',
styleUrl: './product.component.scss'
})
export class ProductComponent {
@Input() productData!:Product;
}
```

```
[productData]="productData"  This represent the @Input
```

Install primeng, [npm install primeng], this has a prebuilt component that we can use in our application

Now watch here
1. From the product.component.ts, do this

```
import { Component, EventEmitter, Input, Output } from '@angular/core';
import { Product } from '../../../types';
import { RatingModule } from 'primeng/rating';
import { FormsModule } from '@angular/forms';

@Component({
selector: 'app-product',
standalone: true,
imports: [RatingModule,FormsModule],
templateUrl: './product.component.html',
styleUrl: './product.component.scss'
})
export class ProductComponent {
@Input() productData!:Product;
@Output() productOutput: EventEmitter<Product>= new EventEmitter<Product>();

ngOnInit() {
this.productOutput.emit(this.productData)
}
}
```

the @Input allows you to accept data into the component defined here

```
<app-product *ngFor="let productData of productsDict"
[productData]="productDat"

></app-product>
```

the @Output helps to emit variables outside the component

```
<app-product *ngFor="let productData of productsDict"
(productOutput)="onProductOutput($event)"

></app-product>
```

In the home.component.ts, there is a function known as

```
onProductOutput(product: Product){
```

```
console.log(product,"Nice one")
}
```

Jungle Studies