Wednesday, 16 August 2023

22.00

**Step 1: Set Up the Environment**

* I have downloaded from npm (node package manager) and node JS from the official Node.js website
* Installed the Angular CLI globally using the following command :  
    
  npm install -g @angular/cli

**Step 2: Creating a New Angular Project**

* Created a new Angular project by running the following command:

ng newweather-dashboard

* Navigate into the project directory:  
  Cd weather-dashboard

**Step 3: Designing the Dashboard**

Inside the project directory, open the src/app folder and create the necessary components and services models using the Angular CLI.  
  
ng generate component dashboard

ng generate service weather

Designde the layout of your weather dashboard using HTML and CSS. Used Angular's data binding and directives to dynamically display data.

**Step 4: Fetching Weather Data**

* Open the weather.service.ts file (inside the src/app folder) and use Angular's HttpClient module to make API requests. I have used public weather API OpenWeatherMap.
* Typescript Code :

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

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

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root',

})

export class WeatherService {

private apiKey = 'myapikey';

private apiUrl = '<https://api.openweathermap.org/data/2.5/weather>';

constructor(private http: HttpClient) {}

getWeather(city: string): Observable<any> {

const url = `${this.apiUrl}?q=${city}&appid=${this.apiKey}`;

return this.http.get(url);

}

}

**Step 5: Displaying Weather Data**

Open the dashboard.component.ts file and use the WeatherService to fetch weather data.  
  
Typescript Code :

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

import { WeatherService } from '../weather.service';

@Component({

selector: 'app-dashboard',

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

styleUrls: ['./dashboard.component.css'],

})

export class DashboardComponent implements OnInit {

weatherData: any;

constructor(private weatherService: WeatherService) {}

ngOnInit(): void {

this.weatherService.getWeather('CityName').subscribe((data) => {

this.weatherData = data;

});

}

}

Open the dashboard.component.html file and use Angular's data binding to display the weather data.

Html :

<div \*ngIf="weatherData">

<h2>Weather in {{ weatherData.name }}</h2>

<p>Temperature: {{ weatherData.main.temp }} °C</p>

<p>Weather: {{ weatherData.weather[0].description }}</p>

</div>

<div\*ngIf="weatherData"><h2>Weather in {{ weatherData.name }}</h2><p>Temperature: {{ weatherData.main.temp }} °C</p><p>Weather: {{ weatherData.weather[0].description }}</p></div>

**Step6 : Adding the temperature gauge component to display the transitioning of colors**

* Generate a new temperature gauge component using the following command : Ng generate component temperature-gauge
* Design the Temperature Gauge Component

Open the temperature-gauge.component.ts file and define the input property to pass the temperature value to the component:

Open the temperature-gauge.component.html file and design the temperature gauge using HTML and CSS. You can use a simple SVG element to create the gauge.

Open the temperature-gauge.component.css file and apply styles to the gauge:

* Using the temperature gauge component

* Open the dashboard.component.html file and include the <app-temperature-gauge> tag to use the TemperatureGaugeComponent.

**Step 7: Running the Application**

Run the development server using the following command: ng serve   
 Open the browser and navigate to <http://localhost:4200> to see the weather dashboard in action.