1. Create a file in the app folder called hero.service.ts.

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

@Injectable()

export class HeroService {

}

1. Add a getHeroes method stub.

@Injectable()

export class HeroService {

getHeroes(): void {} // stub

}

1. move the mock data to its own file.
   1. Cut the HEROES array from app.component.ts and paste it to a new file in the app folder named mock-heroes.ts. We copy the import {Hero} ... statement as well because the heroes array uses the Hero class.

import { Hero } from './hero';

export const HEROES: Hero[] = [

{id: 11, name: 'Mr. Nice'},

{id: 12, name: 'Narco'},

{id: 13, name: 'Bombasto'},

{id: 14, name: 'Celeritas'},

{id: 15, name: 'Magneta'},

{id: 16, name: 'RubberMan'},

{id: 17, name: 'Dynama'},

{id: 18, name: 'Dr IQ'},

{id: 19, name: 'Magma'},

{id: 20, name: 'Tornado'}

];

1. back in app.component.ts where we cut away the HEROES array, we leave behind an uninitialized heroes property:

heroes: Hero[];

1. Back in the HeroService we import the mock HEROES and return it from the getHeroes method. Our HeroService looks like this:

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

import { Hero } from './hero';

import { HEROES } from './mock-heroes';

@Injectable()

export class HeroService {

getHeroes(): Hero[] {

return HEROES;

}

}

1. Import the HeroService in AppComponent

import { HeroService } from './hero.service';

1. Importing the service allows us to reference it in our code. How should the AppComponent acquire a runtime concrete HeroService instance?

We could create a new instance of the HeroService with new like this:

heroService = new HeroService(); // don't do this

But this is a bad idea for several reasons!

Instead, we will **Inject** the HeroService

* 1. We add a constructor that also defines a private property.
  2. We add to the component's providers metadata.

app/app.component.ts (constructor)

constructor(private heroService: HeroService) { }

Refresh the app and check out the error in concole.

EXCEPTION: No provider for HeroService! (AppComponent -> HeroService)

We have to teach the injector how to make a HeroService by registering a HeroService **provider**. Do that by adding the following providers array property to the bottom of the component metadata in the @Component call.

providers: [HeroService]

1. add a method in AppComponent as given below. Although we don’t really need a dedicated method to wrap one line, we write it anyway.

getHeroes(): void {

this.heroes = this.heroService.getHeroes();

}

1. But when and where to call the above method to get data?

The ngOnInit Lifecycle Hook

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

export class AppComponent implements OnInit {

ngOnInit(): void {

this.getHeroes();

}}

1. Update the HeroService with this **Promise-returning** getHeroes method:

getHeroes(): Promise<Hero[]> {

return Promise.resolve(HEROES);

}

1. Update the getHeroes() in app.component.ts

getHeroes(): void {

this.heroService.getHeroes().then(heroes => this.heroes = heroes);

}