academeez

Dependency Injection

5. Providers

Providers

academeez

- The **Providers** are used to register data with a certain injector
- The **Providers** array can be specified on a Component, Directive, Module
- If the **Providers** is on a Component or Directive it will add a node to the element injector tree.
- The **Providers** in a module will either create a new node to the module injector tree (if the module is lazy loaded), or will add the registered data to the nearest node in the tree.
- The **provide** key is the name of the key to register the value in the Map
- We can specify items in the **Providers** array in different ways, we will cover them and their use cases in this lesson.

ClassProvider

- The **Providers** array will accept a class
- The class is a shortcut for the **useClass** syntax
- It Means it will register in the injector under the key of the class type a class instance (creating the class instance is lazy)

```
providers: [
    PigletService,
    { provide: SweetnessService, useClass: SweetnessService}
  ],
```

ValueProvider

academeez

- Usually used when the injector needs to supply a value and not an instance
- Usually registered using an **InjectionToken**
- The **InjectionToken** can have a generic type for type safety of the value
- The **InjectionToken** gets a string used for debugging and exceptions

```
const NUMBER_42: InjectionToken<number> = new InjectionToken(
  'The meaning of life'
)

providers: [
    {provide: NUMBER_42, useValue: 42}
  ],
```

ExistingProvider

academeez

- Upon asking for the key from the injector it will be resolved by asking the injector for another key
- In this example asking for **NUMBER_42** will ask the injector to supply what is registered under then key **PigletService**

```
providers: [
    {provide: NUMBER_42, useExisting: PigletService}
  ],
```

FactoryProvider

- There are times when we will need to create the service when it is requested
- The **FactoryProvider** allows us to provide a function that will be called when we request the service
- You can use the **deps** to inject other services to the factory method
- The **multi** will specify that the injector will hold an array of values for that key

```
providers: [
    {
      provide: TODO_LIST,
      deps: [HttpClient],
      useFactory: (http: HttpClient) => {
        return http.get('https://...')
      }
    }
  ],
```

InjectionToken and useValue

- It is a common pattern to use InjectionToken as a way to pass configuration to a service
- The following is an example of a service that will get a host url as configuration from the DI

```
const HOST_URL: InjectionToken<string> = new
InjectionToken(
  'for configuring our service'
)

providers: [
    {
      provide: HOST_URL,
      useValue: 'https://academeez.com/api'
    }
  ],
```

```
@Injectable({providedIn: 'root'})
export class PigletService {
  constructor(
    private _http: HttpClient,
    @Inject(HOST_URL) private _host: string
  ) { }

  sendRequest = () => {
    return this._http.get(`${this._host}/hello`);
  }
}
```

Summary

- There are various ways in which we can register a key and a value to an Injector
- The registration to an Injector is done with the **Providers** array

academeez

Thank You