Angular 6 Training Course

Exercise E-di

- **Dependency injection (DI)** allows us to separate our project into **loosely coupled** parts.
- The **connections** between the parts are clearly defined.
- Each part is testable in **isolation**.
- Mock data can be **injected** into components.
- This example passes a **service** and an **object** between different parts of the project using DI.

Setup

- Duplicate **c-compose** and rename the folder **e-di**.
- Rebuild the project using NPM.

```
npm install
ng serve --open.
```

Create a data service

- The fruit data is hard-coded in the main component as an array. We can move it into a **service**.
- We will pass an instance of the service into the main component constructor using DI.
- Use the Angular-CLI to generate a new service.

```
ng generate service service/data --dry-run
ng generate service service/data
```

- Note: this command does not update **src/app/app.module.ts**.
- We need to manually add the service to the **NgModule providers** array.

```
import { DataService } from './service/data.service';
providers: [DataService]
```

- The Angular CLI creates a minimal service class with an empty constructor. The class is wrapped in an **Injectable** annotation.

```
@Injectable()
export class DataService {
    constructor() {}
}
```

- Move the fruit array into the service, and create a getter method:

```
    fruit = [ .... ];

    getFruit() {
        return this.fruit;
    }
```

- Import this service into the shop component:

```
import { DataService } from "./service/data.service";
```

- Pass the service into the constructor.

```
constructor( private ds:DataService ) { .. }
```

- Call the getFruit method of the service.

```
this.fruit = ds.getFruit();
```

- Review the Injector Graph in the **Augury** Chrome extension.
- *Exercise : add a boolean parameter to getFruit which filters in only items that are in-stock.*

Inject data objects directly

- You can pass simple JS data objects around using DI.
- Create a file **config/app.config.ts** that holds **project configuration** information.
- This object contains two properties.
- The **provide** property is the name that this object is referred to in DI.

- Its data is held as an object in the **useValue** property.

```
// config/app.config.ts

export const Config = {
    provide:"config",
    useValue : {
        shop : "Southwold Organics",
        addr : {
            street : "14 Dolphin..",
            postcode : "IP18 4HZ"
        }
    }
}
```

- Import it into **app.module.ts** and add it to the list of **providers**.

```
 import { Config } from './config/app.config';
 providers: [ DataService,Config ]
```

- This object can be injected into the constructor.

```
    import { Inject } from '@angular/core';

    constructor( @Inject("config") private config ,
    private ds:DataService )
```

- Refactor the constructor to use this object.

```
this.shop = config.shop;
this.addr = config.addr;
```

- *Review the DI in the Augury Chrome extension.*

Use HTTP to read the data from a JSON file.

- Create a JSON file in the assets folder: **assets/data/fruit.json**
- Note the JSON data needs to be stringified.
- Add the Angular HTTP module to app.module.ts

```
import {HttpClientModule} from '@angular/common/http';
imports: [BrowserModule,HttpClientModule]
```

- Import the Angular HTTP client into the service.

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

- Inject the HTTP client into the service constructor.

```
constructor( private http: HttpClient ) {}
```

- Change getFruit to use the HTTP get() method to read the JSON file.

```
    path : "assets/data/fruit.json";

  getFruit() {
      return this.http.get( this.path );
  }
```

- The main component already calls getFruit but this code will cause a runtime error.

```
this.fruit = ds.getFruit();
```

- The HTTP get method returns an **Observable**, an object for handling an asynchronous stream of data.
- We need to subscribe to this stream in order to read the data.

```
this.ds.getFruit().subscribe( fruit => this.fruit = fruit );
```

- *Observables will be covered in more detail later in the course.*