Front-End Infrastructure Implementation

This lesson explains the front-end infrastructure implementation, the modules, components, and goes over the folder structure of the project we'll go over and implement in this chapter.

WE'LL COVER THE FOLLOWING

- Modules & Components
- model Folder
- services Folder
- users Folder
- Components

Modules & Components

We will be implementing the following project from GitHub; so let's start by discussing the different folders/files that we'll be using throughout the project.

In the GitHub project, there is a separate *mean_frontend* folder.

You can see that there are a bunch of files in there, most of which are configuration files. Our focus should be on the *src* folder, where the source code is. This is probably a good moment to introduce the terms *Modules* and *Components*, as they are the main units of every Angular application.

Angular applications are modular, and have their own modular system – `NgModules`. These modules help developers organize applications in blocks of functionality and to have a better separation of concerns.

Every Angular application has at least one module called the *root* module, which can be found in the project with the named – `AppModule`. Since our application is going to be a simple one, we will only use this module; you can find it in the file *app.module.ts* in the *mean_frontend/src/app/* directory.

> Each module can have one or more Angular Component.

If you take a look at the folder structure, you will notice that there is only one component, *users.component.ts*, in the */mean_frontend/src/app/users/* directory. This is for simplicity.

So, if you look in the *app* directory you'll find:

- user
- service
- model

model Folder

Inside the *model* folder, you will find the class that defines the structure of *users* and will help us manipulate the data on the front-end:

```
export class User {

    constructor(
        public _id: string,
        public name : String,
        public age: Number,
        public location : string,
        public blog: string
    ){}

    static CreateDefault(): User {
        return new User('', '', 0, '', '');
    }
}
```

/mean_frontend/src/app/model/user.ts

services Folder

Inside the *services* folder, one can find the file *user.service.ts*, which contains part of the Angular application that is used to communicate with the backend, using the Web API.

More of this implementation will be explained later.

users Folder

And finally, the *users* folder contains two files:

- *users.component.ts*
- *users.component.html*

These two files define the component that was mentioned earlier. Together, these files contain all of the front-end CRUD logic that we will explain in the next lessons.

Components

Let's say a little bit more about Angular components. In general, components are still classes, but there are some rules that need to be followed when implementing them.

Every component class has to have the decorator `@Component`. This way, developers can mark a class as a component and define how it will be created and used.

This decorator has the fields:

- `selector`: defines *HTML tag* of the created component
- `templateUrl`: defines a path to the components HTML.

Alternatively, one can use field templates and define HTML components inline.

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

@Component({
  selector: 'example',
  templateUrl: './example.component.html',
})
export class ExampleComponent implements OnInit {

  constructor() { }

  ngOnInit() { }
}
```

Example

Note that this example component implements the `OnInit` interface. This interface provides one function `ngOnInit()`, which will be called during

interface provides one function, `ngOnInit()`, which will be called during initialization of the component. All of the code regarding initialization of the component should be in this function.

Also, it is very important that the entities defined in these files are incorporated in the *root* module, otherwise the application won't work. Take a look at the *app.module.ts* file:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { HttpModule } from '@angular/http';

import { UserComponent } from './users/users.component';
import { UserService } from './services/users.service';

@NgModule({
  declarations: [
    AppComponent,
    UserComponent,
  ],
  imports: [
    BrowserModule,
    HttpModule,
    FormsModule
  ],
  providers: [
    UserService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

/mean_frontend/src/app/app.module.ts

As you can see, all of the `declarations`, `imports` and `providers` have been mentioned in this file.

In the next lesson, we'll take a look at the backend infrastructure implementation.