**Lesson25 Angular modules explained**

**Notes:-**

1-In simple terms an Angular Module is a class decorated with @NgModule decorator.

2-**(Angular Modules which is represent as app.modules.ts):-**is a mechanism to group components, directives, pipes and services that are related to a feature area of an angular application.

For example, if you are building an application to manage employees, you might have the following features in your application.

|  |  |
| --- | --- |
| **Application Feature** | **Description** |
| Employee Feature | Deals with creating, reading, updating and deleting employees |
| Login Feature | Deals with login, logout, authenticate and authorize users |
| Report Feature | Deals with generating employee reports like total number of employees by department, top 10 best employees etc |

To group the components, directives, pipes and services related to a specific feature area, we create a module for each feature area. These modules are called **feature modules**.

|  |  |
| --- | --- |
| **Module Type** | **Description** |
| Root Module | Every Angular application has at least one module, the root module. By default, this root application module is called AppModule. We bootstrap this root module to launch the application. If the application that you are building is a simple application with a few components, then all you need is the root module. As the application starts to grow and become complex, in addition to the root module, we may add several feature modules. We then import these feature modules into the root module. We will discuss creating feature modules in our upcoming videos |
| Core Module | The most important use of this module is to include the providers of http services. Services in Angular are usually singletons. So to ensure that, only one instance of a given service is created across the entire application, we include all our singleton service providers in the core module. In most cases, a CoreModule is a pure services module with no declarations. The core module is then imported into the root module (AppModule) only. CoreModule should never be imported in any other module. We will discuss creating a core module in our upcoming videos |
| Shared Module | This module contains reusable components, directives, and pipes that we want to use across our application. The Shared module is then imported into specific Feature Modules as needed. The Shared module might also export the commonly used Angular modules like CommonModule, FormsModule etc. so they can be easily used across your application, without importing them in every Feature Module. We will discuss creating a shared module in our upcoming videos |
| Routing Modules | An angular application may also have one or more routing modules for application level routes and feature module routes |

**Well, there are several benefits of Angular Modules.**

|  |  |
| --- | --- |
| **Benefit** | **Description** |
| Organizing Angular Application | First of all, Modules are a great way to organise an angular application. Every feature area is present in it's own feature module. All Shared pieces (like components, directives & pipes) are present in a Shared module. All Singleton services are present in a core module. As we clearly know what is present in each module, it's easier to understand, find and change code if required |
| Code Reuse | Modules are great way to reuse code. For example, if you have components, directives or pipes that you want to reuse, you include them in a Shared module and import it into the module where you need them rather than duplicating code. Code duplication is just plain wrong, and results in unmaintainable and error prone code. We will discuss creating a Shared module and how it can help us reuse code in our upcoming videos |
| Code Maintenance | Since Angular Modules promote code reuse and separation of concerns, they are essential for writing maintainable code in angular projects |
| Performance | Another great reason to refactor your application into modules is performance. Angular modules can be loaded either eagerly when the application starts or lazily on demand when they are actually needed or in the background. Lazy loading angular modules can significantly boost the application start up time. We will discuss lazy loading modules in our upcoming videos |

**As we have already discussed an Angular module is a class that is decorated with @NgModule decorator. The @NgModule decorator has the following properties.**

**1-declarations**

**2-bootstrap**

**3-providers**

**4-imports**

**5-exports**