**Section14 Module in Angular**

**Notes: -**

**1-modules are the container that hold other resources such as**

**Component, Directives, Services,etc.**

**2-Angular analyze NgModules to understand your application and its features**

**3-Angular modules define all building blocks your app uses : component , directives , services ,etc…**

**4-by default there is core modules but can splitted into multiple modules**

**5-core module feature are included in angular modules such as form module to load them only when needed**

**6-you can’t use feature / building block without including it in module**

**(In previous project we have two modules: App.module.ts, App-routing.module.ts (They are using @NgModule))**

**Parts: -**

**A-Declarations: which used to import component, directive, pipes**

**B-Import: which used to import general module such as Form Module, reactive Form Module, another module**

**C-Providers: which used to import services, interceptor**

**D-Bootstrap: which used to startup component to initialize your app**

**E-entryComponent: which used to import component that created at runtime Dynamic component declaration**

**7-any service must apply @Injectable({providedIn:’root’})**

**8-in app-routing.module.ts we will use import and export in order to import appRoutes and exports to export the RouterModule as below**

**(Every module works on its own, so you have to export it to use into another module)**

**//so, in approuting.ts we export the RouterModule**

**@NgModule({imports:[RouterModule.forRoot(appRoutes)],exports:[RouterModule]})**

**export class AppRoutingModule{}**

**//in the app.module.ts we import it as below**

**imports: [AppRoutingModule],**

**Working with multiple Modules**

**Notes: -**

**1-we will split the big main module into spitted modules called feature module**

**(We have two sub modules such as product module, order module, app module)**

**(It will allow to use any resource in any module into the target module)**

**(We will create three module, AUTH Module, recipe Module, shopping List Module, Shared Module)**

**2-you have to export the Module and component and directive and pipes unless service only on provider’s section**

**3-in the shared module we group all the shared module and shared directives and components into this module and inject**

**ng generate module app-routing --flat --module=app**

**ng generate module <name> [options]**

**Lazy Loading Approach**

**Notes: -**

**1-it will load the module once its needed, so instead of make eager loading, we can apply lazy loading in order to load the demand module**

**(This will optimize and make loading faster)**

**2-in our example we have to set all services on the app.module.ts because we are using subject in order to communicate between different component on different modules**

**3-on the app-routing.module.ts we are using loadchildren() in order to apply loazy loading plus remove the sub module in import of the app.module.ts**

**{path:'recipes',loadChildren:() => RecipesModule},**

**{path:'shopping-list',loadChildren:() => ShoppingModule},**

**{path:'auth',loadChildren:() => CoreModule}**

**//or the below syntax**

**{path:'recipes',loadChildren:('./recipes/recipes.module.ts')},**

**{path:'shopping-list',loadChildren:('./shopping/shopping.module.ts')},**

**{path:'auth',loadChildren:('./modules/core/core.module.ts')},**

**On the child-routing we are update the routing by set empty on the base parent routing**

**const appRoutes:Routes = [**

**{path:'',component:RecipesComponent,canActivate:[AuthGuard],children:[**

**{path:'',component:RecipeStartComponent},**

**//you have to set all routing without parameters before the routing with parameter**

**{path:'new',component:RecipeEditComponent},**

**{path:':id',component:RecipeDetailsComponent,resolve:[RecipeResolver]},**

**{path:':id/edit',component:RecipeEditComponent,resolve:[RecipeResolver]}]}];**

**@NgModule({**

**//we change to forchild on the sub modules**

**imports:[RouterModule.forChild(appRoutes)],**

**exports:[RouterModule]})**

**Preloading Approach**

**Notes: -**

**1-the bad performance of lazy loading it will make delay until load the target module , so you can use mix approach is preloading which load the important modules and then apply lazy loading to target module**

**(To solver this issue we apply preloading which preload all modules as possible)**

**@NgModule({**

**imports:[RouterModule.forRoot(appRoutes,{preloadingStrategy:PreloadAllModules})],**

**exports:[RouterModule]})**

**Service And Modules**

**Notes: -**

**1-we can apply services through the following: -**

**A-App Module: service available app-wide**

**(By use root injector (should be the default))**

**(By set only @Injectable ({providedIn: 'root',}))**

**B-App Component or other components: service available in component tree**

**(By use component specific injector)**

**(Use if service is only relevant for component tree)**

**C-Eager-loaded module: service available app-wide**

**(Use root injector)**

**(Such as import recipesService, shoppingListService, authService staticly on app.mdoule provider section)**

**(Avoid this)**

**D-Lazy-loaded module: service available in loaded module**

**(Use child injector)**

**(Use if service should be scoped to loaded module)**

**Ahead of time (AVOT) vs Just in Time (JIT) Compilation**

**Notes: -**

**1-you code and template in angular includes syntax only angular understands e.g ngIf**

**, component, directive which is not understand by the compiler browser**

**2-so the typescript compiler compiles your code to JavaScript**

**3-Angular Compiler: automatically included in built code, compiler template syntax to JavaScript DOM instructions and contains two compilation**

**A-Just in time compilation: -**

**Angular template compiler runs in browsers at runtime, render your page**

**(The size is not small, and takes some time to compilations)**

**B-Ahead of time AOT compilation**

**Angular template compiler runs during build process (before the app is deployed)**

**(It will optimize the performance and speed of execution and size of bundle)**

**(it will some extra checks that the Just in time compilation not checks it to make best performance)**

**4-by using the following command**

**Ng build –prod //build application in few files and size by using AOT compilation**

**4-it will build your app inside dist folder and you see decrease your build as possible**

