Angular Documentation Notes

**Architecture Overview**

* Components define views, which are sets of screen elements.
* Components use services which provide specific functionality not directly related to views
* Service providers can be injected into components as dependencies making your code modular, reusable, and efficient
* Components and services are classes, the classes use decorators to mark their type and provide metadata
* A template combines ordinary HTML with Angular Directives and binding markup that allows Angular to modify the HTML
* Angular provides the Router service to help you define navigation paths among views

**Modules**

* Declare the things you create in Angular and group them together, like Java packages
* Declarations is for things you’ll use in your templates: mainly components (views: the classes displaying data)
* Providers is for services (models: the classes getting and handling data)

Code Example on Next Page 🡪

|  |
| --- |
| import { NgModule } from '@angular/core';  import { SomeComponent } from './some.component';  import { SomeDirective } from './some.directive';  import { SomePipe } from './some.pipe';  import { SomeService } from './shared/some.service';  @NgModule({  declarations: [SomeComponent, SomeDirective, SomePipe],  providers: [SomeService]  })  export class SomeModule {} |

* Since Angular 6, services don’t need to be registered in a module anymore. The use of “providers” in a NgModule is now limited to overriding existing services.
* In declarations components are in local scope (private visibility)
* In Providers services are (generally) in global scope (public visibility)
* This means the components you declared are only usable in the current modules. If you want to use them in other modules, you’’ll have to export them

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | import { NgModule } from '@angular/core'; | |  |  | |  | import { SomeComponent } from './some.component'; | |  | import { SomeDirective } from './some.directive'; | |  | import { SomePipe } from './some.pipe'; | |  |  | |  | @NgModule({ | |  | declarations: [SomeComponent, SomeDirective, SomePipe], | |  | exports: [SomeComponent, SomeDirective, SomePipe] | |  | }) | |  | export class SomeModule {} | |

* Service you provided will generally be available/injectable anywhere in your app
* Another thing you do with NgModules in Angular is import other Modules you need.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | * import { NgModule } from '@angular/core'; | |  | import { CommonModule } from '@angular/common'; | |  | import { HttpClientModule } from '@angular/common/http'; | |  |  | |  | import { FeatureModule } from '../feature/feature.module'; | |  |  | |  | @NgModule({ | |  | imports: [CommonModule, HttpClientModule, FeatureModule] | |  | }) | |  | export class SomeModule {} | |

Problem is you need to know why you import these other modules.

Why? Because given the difference of scope between components and services:

* **if the module is imported for components, you’ll need to import it in each module** needing them,
* **if the module is imported for services, you’ll need to import it only once**, in the first app module.

**When to import main Angular modules?**

Modules to import each time you need them

* CommonModule (all the basics of Angular templating: bindings, \*ngIf, \*ngFor…), except in the first app module, because it’s already part of the BrowserModule
* FormsModule / ReactiveFormsModule
* MatXModule and other UI modules
* any other module giving you components, directives or pipes

Modules to import only once

* HttpClientModule
* BrowserAnimationsModule or NoopAnimationsModule
* any other module providing you services *only*.

That’s why with [Angular CLI](https://github.com/angular/angular-cli), CommonModule is automatically imported when you create a new module.

**Mixed NgModules**

How to handle modules with components and services at the same time.

**Lazy-loaded modules**

|  |
| --- |
| const routes: Routes = [  { path: ‘admin’, loadChildren: ‘./admin/admin.modules#AdminModule’ }  ]; |

It will be a different bundle and modules, loaded only on demand by default, its not included in the global scope of your app.

* Services provided in your lazy-loaded module will only be available in this lazy-loaded module