

Sharing modules



Contents

Using components vs services from other modules

More on NgModules

Creating shared modules allows you to organize and streamline your code. You can put commonly used directives, pipes, and components into one module and then import just that module wherever you need it in other parts of your app.

Consider the following module from an imaginary app:

```
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CustomerComponent } from './customer.component';
import { NewItemDirective } from './new-item.directive';
import { OrdersPipe } from './orders.pipe';

@NgModule({
  imports:      [ CommonModule ],
  declarations: [ CustomerComponent, NewItemDirective, OrdersPipe ],
  exports:      [ CustomerComponent, NewItemDirective, OrdersPipe,
                  CommonModule, FormsModule ]
})
export class SharedModule { }
```



Note the following:

- It imports the CommonModule because the module's component needs common directives.
- It declares and exports the utility pipe, directive, and component classes.
- It re-exports the CommonModule and FormsModule.

By re-exporting CommonModule and FormsModule, any other module that imports this SharedModule, gets access to directives like NgIf and NgFor from CommonModule and can bind to component properties with [(ngModel)], a directive in the FormsModule.

Even though the components declared by SharedModule might not bind with [(ngModel)] and there may be no need for SharedModule to import FormsModule, SharedModule can still export FormsModule without listing it among its imports. This way, you can give other modules access to FormsModule without having to import it directly into the @NgModule decorator.

Using components vs services from other modules

There is an important distinction between using another module's component and using a service from another module. Import modules when you want to use directives, pipes, and components. Importing a module with services means that you will have a new instance of that service, which typically is not what you need (typically one wants to reuse an existing service). Use module imports to control service instantiation.

The most common way to get a hold of shared services is through Angular [dependency injection](#), rather than through the module system (importing a module will result in a new service instance, which is not a typical usage).

To read about sharing services, see [Providers](#).

More on NgModules

You may also be interested in the following:

- [Providers](#).
- [Types of Feature Modules](#).

RESOURCES

[About](#)

[Resource Listing](#)

[Press Kit](#)

[Blog](#)

[Usage Analytics](#)

HELP

[Stack Overflow](#)

[Gitter](#)

[Report Issues](#)

[Code of Conduct](#)

COMMUNITY

[Events](#)

[Meetups](#)

[Twitter](#)

[GitHub](#)

[Contribute](#)

LANGUAGES

[简体中文版](#)

[正體中文版](#)

[日本語版](#)

[한국어](#)

Super-powered by Google ©2010-2020. Code licensed under an MIT-style License. Documentation licensed under CC BY 4.0.

Version 10.0.10-local+sha.84d1ba792b.