Angular Modules

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- Angular modules
- Angular module vs JavaScript module
- How to launch an application

Angular Module

Angular 2 Module

- Angular introduced new modularity system called Angular modules or NgModules
- Every Angular app has at least one module, the root module, conventionally named AppModule
- An Angular module, whether a root or feature, is a class with an @NgModule decorator

Example Root Module (app/app.module.ts)

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

@NgModule({
  imports:      [ BrowserModule, FormModule, HttpModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],   // This is really not needed since nobody imports this module
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Example Module

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MineComponent } from './mine.component';

@NgModule({
  imports: [
    CommonModule
  ],
  exports: [
    MineComponent
  ],
  declarations: [MineComponent]
})
export class MineModule { }
```

Properties of NgModule decorator

- imports
  - Other modules whose exported classes are needed by component templates declared in this module
- providers
  - Creators of services that this module contributes to the global collection of services; they become accessible in all parts of the app.
- declarations
  - The view classes that belong to this module. Angular has three kinds of view classes: components, directives, and pipes.
- exports
  - The subset of declarations that should be visible and usable in the component templates of other modules

BrowserModule and CommonModule

- BrowserModule
  - The root application module (AppModule) of almost every browser application should import BrowserModule from @angular/platform-browser.
  - BrowserModule provides services that are essential to launch and run a browser app.
- BrowserModule vs CommonModule
  - BrowserModule also re-exports CommonModule from @angular/common which means that component in the AppModule module also have access to the Angular directives every app needs such as NgIf and NgFor
  - Other modules should import CommonModulle not BrowserModule

Angular Module vs JavaScript Module

Angular Modules vs JavaScript Modules

- JavaScript also has its own module system for managing collections of JavaScript objects. It's completely different and unrelated to the Angular module system
  - These are two different and complementary module systems
  - You use them both to write your Angular 2 application
- In JavaScript, each file is a module and all objects defined in the file belong to that module
  - A JavaScript module declares some objects to be public by marking them with the *export* key word
  - Other JavaScript modules use *import* statements to access public objects from other modules.

```
import { NgModule }     from '@angular/core';
import { AppComponent } from './app.component';


export class AppModule { }
```

Angular Libraries

- Angular ships as a collection of JavaScript modules
  - You can think of them as library modules.
- Each Angular library name begins with the @angular prefix
- You install them with the npm package manager and import parts of them with JavaScript import statements
- For example, import Angular's Component decorator from the @angular/core library like this:

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

- You also import Angular modules from Angular libraries using JavaScript import statements:

```
import { BrowserModule } from '@angular/platform-browser';
```

How to Launch an Application?

How to launch an application

- Launch an application by bootstrapping its root module
- During development you're likely to bootstrap the AppModule in a *main.ts* file like this one

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```

Lab: Create a new module

- Create a new module to an existing application – it also creates HelloComponent
  - ng g module hello
- Study Hello module (hello.module.ts)
  - Observe that CommonModule is imported
- Export HelloComponent in the Hello module
- Import the Hello module to the Root module
- Use HelloComponent

Lab: Move a custom pipe to a its own module

- Move an existing component into its own module
- Move a custom pipe (the one you created in previous lab) to its own module

Code with Passion!
JPassion.com