# Angular best friends

## Module 2 Exercise 1

### Purpose

The outcome of this exercise is to familiarize ourselves with an Angular folder structure following Angular Style Guide and architectural best practices. During the exercise we’ll create a new Angular application, some feature modules and corresponding routing modules.

### Steps

1. Create a folder anywhere you want called “Module 2 Exercise 1”
2. Open Node command prompt (in newer versions of Node.js the command prompt is integrated in the Windows command prompt so opening it will do it)
3. Switch directories to the new created folder you created in step 1

**Command**

*cd “{your folder path}”*

1. Create an Angular application called “myCrm” using the following command in the Node Command Prompt (same as at step 2)

**Command**

*ng new myCrm*

1. You’ll find the current application structure in *myCrm > src > app*. Take a look at what we have by default with a new Angular application. How many modules do we have? How many components? How are they organized?
2. Run the application to make sure it’s working

**Commands**

*cd myCrm*

*npm install @angular-devkit/core*

*ng serve –open*

After we make sure everything is running we can start think about how we’ll organize features and modules. myCrm > src > app serves as our base directory for organizing features and modules. By default with a new application we get the base app.module.ts but no routing modules associated with it. So we need to think here about how we’ll organize everything.

We’ll start very simple by creating a module for our “static” sections like “About” and “Contact”. We’ll call it “About”.

1. Change directory to src > app
2. Create a new folder called “about”
3. Change directory to the newly created folder
4. Generate a new module called “about-routing”. This will be our routing module

**Command**

ng generate module about-routing –flat

The “—flat” flag prevents us from creating a new folder with the same name as the module. Without the flag, Angular CLI would also create a new folder, which we don’t want.

1. Generate a new module called “about”. This will be our feature module

**Command**

ng generate module about –flat

1. Generate a new component called “about”. Here we might want to have it in a new folder so we won’t use the “flat” flag.

**Command**

ng generate component about

1. Generate a new module called “app-routing”. ATTENTION this module needs to be generated in the “app” folder so we have to change to it first.

**Commands**

cd {your app folder}

ng generate module app-routing –flat

1. In “app-routing.module.ts” delete all code and replace it with the following:

import { NgModule } from '@angular/core';

import { RouterModule, Routes, PreloadAllModules, NoPreloading } from '@angular/router';

import { AppComponent } from './app.component';

const app\_routes: Routes = [

{ path: '', component: AppComponent},

{ path: 'about', loadChildren: 'app/about/about.module#AboutModule' },

{ path: '\*\*', pathMatch: 'full', redirectTo: '/about' } // catch any unfound routes and redirect to home page

];

@NgModule({

imports: [

RouterModule.forRoot(app\_routes, { preloadingStrategy: PreloadAllModules })

],

exports: [

RouterModule

],

declarations: []

})

export class AppRoutingModule { }

1. In “app.module.ts” delete all code and replace it with this:

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

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { AppRoutingModule } from './app-routing.module';

import { AboutModule } from './about/about.module';

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

AboutModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. In “app.component.html” delete all code and replace it with this:

<div style="text-align:center">

<h1>

Welcome to {{ title }}!

</h1>

<img width="300" alt="Angular Logo" src="">

</div>

<router-outlet></router-outlet>

1. In “app-routing.module.ts” delete all code and paste this one instead

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { AboutComponent } from './about/about.component';

const routes: Routes = [

{ path: 'about', component: AboutComponent }

];

@NgModule({

imports: [

RouterModule.forChild(routes)

],

exports: [

RouterModule

],

declarations: []

})

export class AboutRoutingModule {

static components = [ AboutComponent ];

}

1. In “about.module.ts” delete all code and replace it with this:

import { NgModule } from '@angular/core';

import { AboutRoutingModule } from './about-routing.module';

@NgModule({

imports: [

AboutRoutingModule

],

declarations: [AboutRoutingModule.components]

})

export class AboutModule { }

Bow we have a basic setup that we can simply follow when we add new components to existing modules or when we add new feature modules. Let’s continue by adding a new component called “Contact” to the about module and check out how routing works.

1. Make sure to change directory in the “about” folder where also “about..module.ts” is located.
2. Generate a new component named “contact”

**Command:**

*ng generate component contact*

1. In about-routing.module.ts add an import for the new component add add the new component to the static property called “components”:

import { ContactComponent } from './contact/contact.component'

export class AboutRoutingModule {

static components = [ AboutComponent, ContactComponent ];

}

1. In the same about-routing.module.ts add a new route for the “contact” endpoint.

const routes: Routes = [

{ path: 'about', component: AboutComponent },

{ path: 'contact', component: ContactComponent}

];

Notice that the contact route is working properly even if we didn’t add it to app-routing.module.ts. This is because a route to the about module is already defined there so routing requests are sent there if needed.

1. Following the same steps create a customers feature module and wire it up so that everything works fine and we are able to navigate to localhost4200/customers