2020

LAZY LOADING OF MODULES IN ANGULAR

CHINMAYI G PANICKER
M1060956

Introduction to Angular

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript.

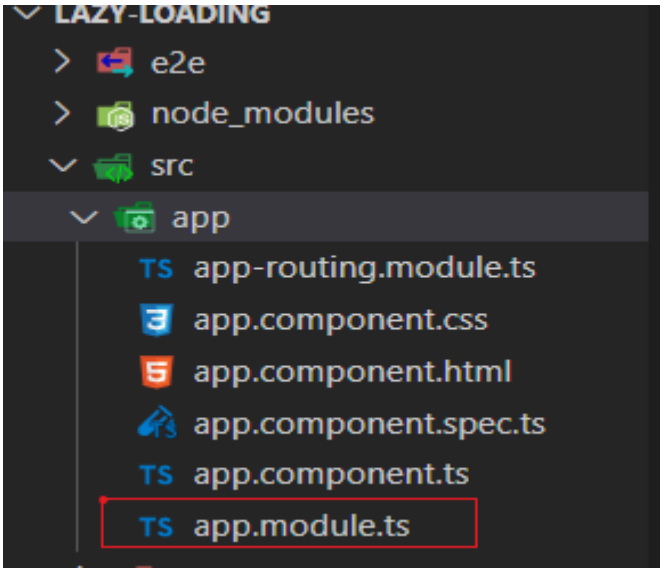
Consider your angular **application** as a building. A building can have N number of apartments in it. An apartment is considered as a **module**. An apartment can have N number of rooms which correspond to **components**. Now each apartment (**module**) will have rooms (**components**), lifts (**services**) to enable larger movement in and out the apartments, wires (**pipes**) to move information around and make it useful in the apartments.

Modules

Angular apps are modular. A module is a way to group components, pipes and services that are related. This modularity is termed as NgModule. Each application will have at least one root module. This root module is called AppModule and is located inside src/app/app.module.ts file as shown below.

The application runs through the Bootstrapping process of this module.

Note*: "Bootstrapping" comes from the term "pulling yourself up by your own bootstraps." Bootstrapping is a technique of initializing or loading our Angular application. The AppModule is also referred as the 'entryComponent' of our Angular application.*



An NgModule is defined by the class with decorator @NgModule().The properties of an NgModule() function are as follows,

- **declarations**: This is an array of components, directives and pipes that are part of the module
- **imports**: This array includes the modules that are used by the components in the module.
- **providers**: This includes services. The services declared in the root module will be accessible in all parts of the application.
- **bootstrap**: This will have the root component which get loaded on to the browser. Bootstrap property will exist only in the root module of the application.

  Here's a simple root NgModule definition.

By default, NgModules are eagerly loaded, which means that as soon as the application starts all the modules are loaded whether or not it is necessary. For larger applications lazy loaded is required.

Concept of lazy loading

- What is lazy loading and why is it important?

Lazy loading is a technique in which non critical resources are loaded only as per user's demand. The importance of lazy loading can be explained through a simple example.

Consider a chat application like WhatsApp. A user can have any number of conversations. Whenever they open the app, the application will display only the latest message in each conversation in the chat list. What if the user is a celebrity and has thousands of conversations? In this case, it will take forever to load the whole conversation list. As a solution to this lazy loading principle is applied. The application loads only the conversation list required as per the screen size. The rest is loaded only when the user scrolls down through chat list.

Implementation of lazy loading

Step 1:

Let's generate two modules and components for understanding lazy loading using the command as shown below.

```
E:\Angular\lazy-loading>ng g m home
CREATE src/app/home/home.module.ts (190 bytes)

E:\Angular\lazy-loading>ng g m lazy
CREATE src/app/lazy/lazy.module.ts (190 bytes)
```

```
E:\Angular\lazy-loading\src\app>ng g c lazy/lazy-demo
CREATE src/app/lazy/lazy-demo/lazy-demo.component.html (24 bytes)
CREATE src/app/lazy/lazy-demo/lazy-demo.component.spec.ts (643 bytes)
CREATE src/app/lazy/lazy-demo/lazy-demo.component.ts (286 bytes)
CREATE src/app/lazy/lazy-demo/lazy-demo.component.css (0 bytes)
UPDATE src/app/lazy/lazy.module.ts (276 bytes)

E:\Angular\lazy-loading\src\app>ng g c home/home
CREATE src/app/home/home/home.component.html (19 bytes)
CREATE src/app/home/home/home.component.spec.ts (614 bytes)
CREATE src/app/home/home/home.component.ts (267 bytes)
CREATE src/app/home/home/home.component.css (0 bytes)
UPDATE src/app/home/home.module.ts (258 bytes)
```

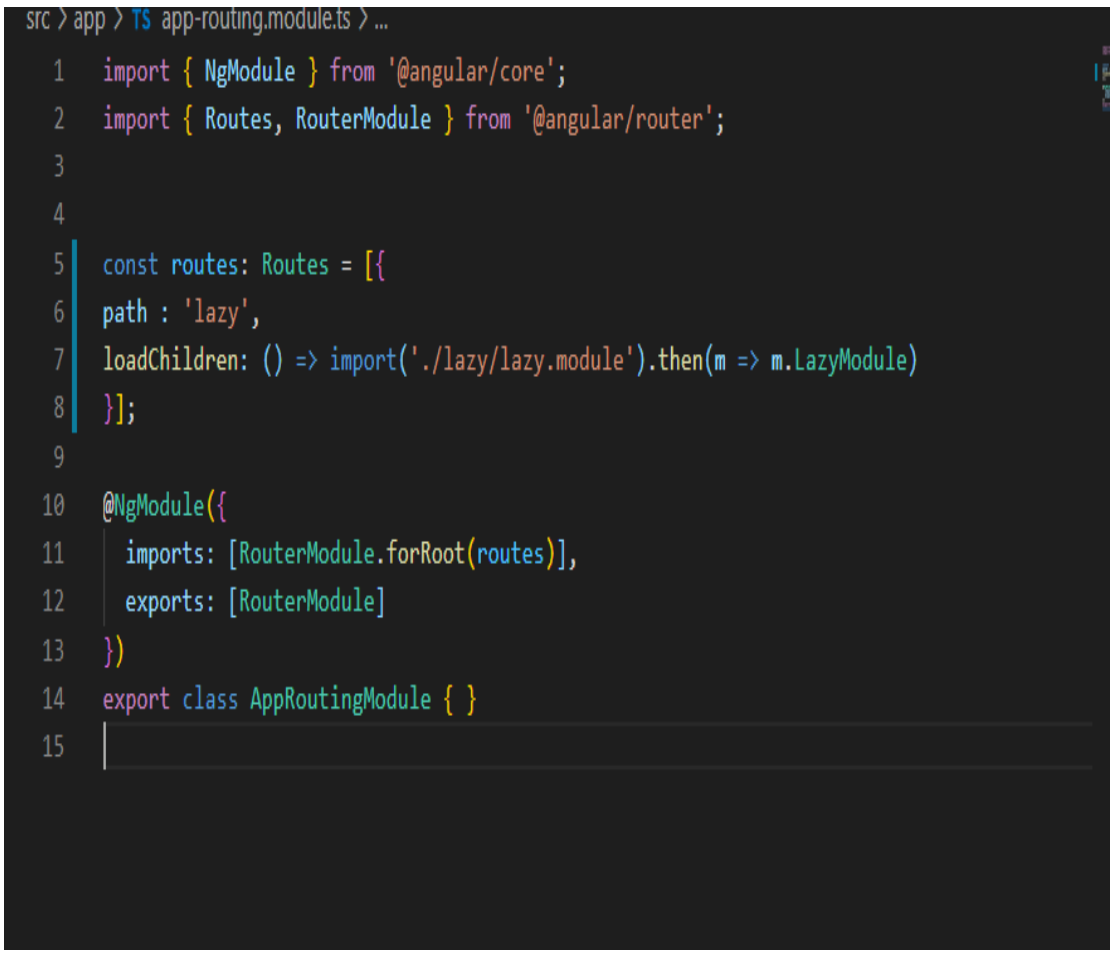
After executing the command, the @angular/cli automatically import the component in the lazy.module.ts file.

Step 2:

Next step is to configure the routes in the app-routing.module.ts file. Create a new route and it should use the loadChildren property to load the lazy module instead of component. By using this property, the module will only be loaded when the user triggers the route.

We have to add route for the component of the lazy module in lazy.module.ts. After the configuration the files app-routing.module.ts and lazy.module.ts should look like as given below

- app-routing.module.ts



- lazy.module.ts

```
import { Routes, RouterModule } from '@angular/router';
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LazyDemoComponent } from './lazy-demo/lazy-demo.component';

const LAZY_ROUTES: Routes = [{
path: '',
component: LazyDemoComponent
}];

@NgModule({
  declarations: [LazyDemoComponent],
  imports: [
    CommonModule, RouterModule.forChild(LAZY_ROUTES)
  ]
})
export class LazyModule { }
```

- **forRoot() and forChild()**

We can see RouterModule.forRoot(routes) in the imports array of AppRoutingModule .When this method is called Angular instantiates the Router class globally. The forRoot configures all the routes passing to it and gives access to router directives, registers the Router Service. The method is used only once in the application, inside app-routing.module.ts.

Router.forChild(routes) is used the other way around. It contains directives and the given routes but not the Router Service. This method lets angular know that the route list is for providing additional roues and is intended for feature modules.

Verification of Lazy-loading

Step 3:

Let’s verify lazing loading by running our application. Run the application using the ‘ng serve’ command.

Open the Chrome Menu ➡ More Tools ➡ Developer Tools.

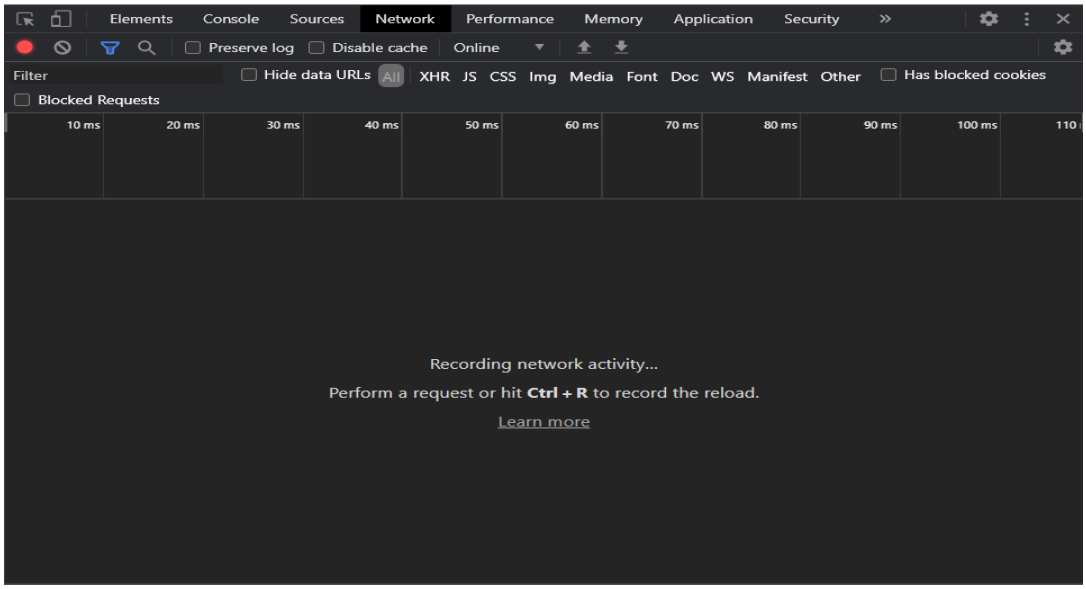
Open the Network tab in the Chrome Dev tools.

From the figure below we can see that no modules are loaded at first.

LAZY LOADING

- Lazy load link

home works!



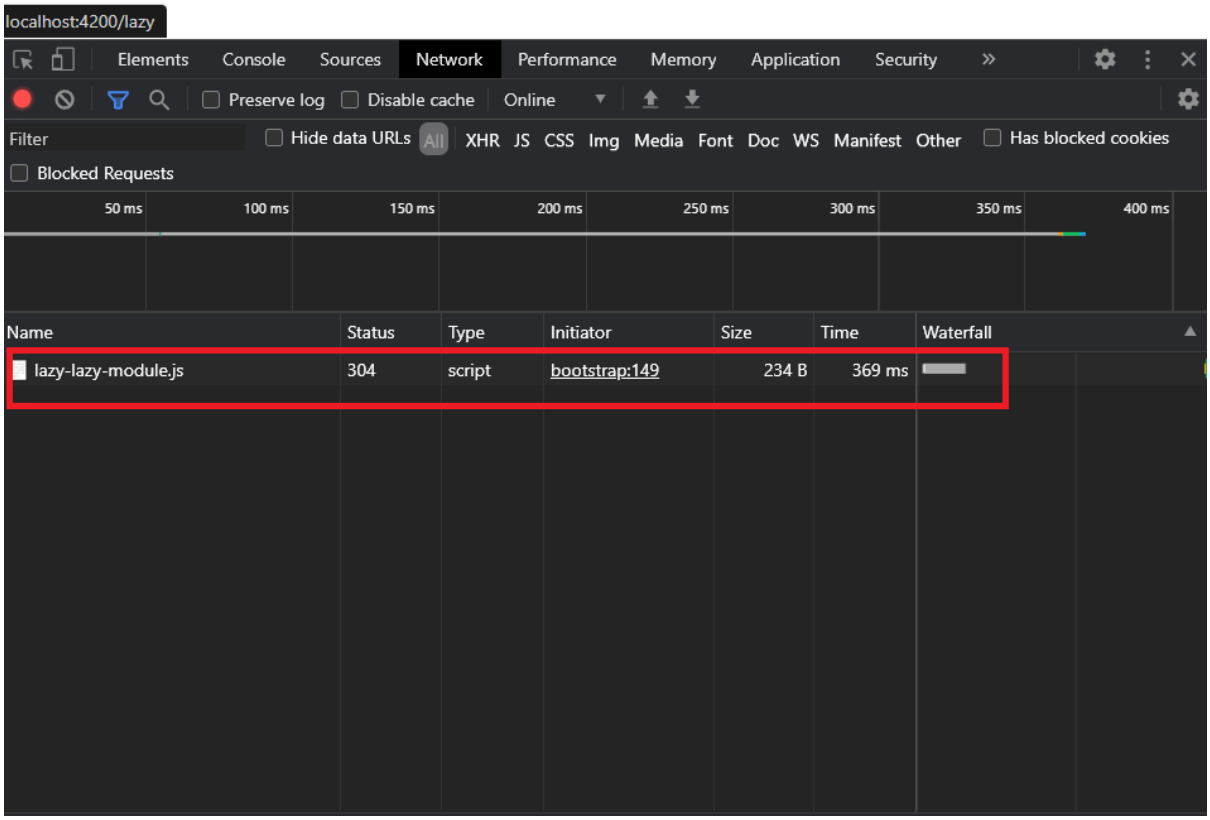
Now let’s see what happens when we click on the ‘Lazy Load’ link

LAZY LOADING

- Lazy load link

Lazy loading works

home works!



From the figure we can see that when the lazy load link is clicked the lazy module is loaded by the browser.

Conclusion

By using the lazy loading principle, we can improve the initial loading time of our application and makes the overall experience of the user better.