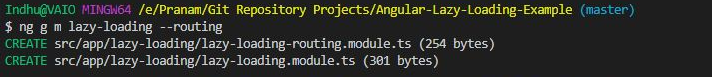
**Steps to implement Lazy Loading in Angular**

1. Create a new Angular project:

ng new angular-lazy-loading

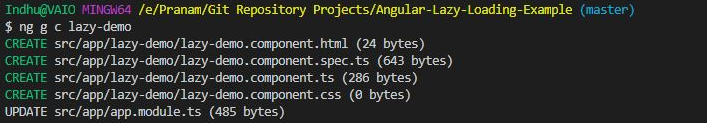
1. Create a module and separate routing file named lazy-loading. The purpose of independent routing is to handle all the components associated with angular lazy-loading module.

**ng g m lazy-loading –routing**



1. Create a component named lazy-demo within the lazy-loading module

**ng g c lazy-demo**



1. Adding a link in the header on whose route we will implement lazy loading.

**app.component.html :**

<li class="nav-item">

<a class="nav-link" [routerLink]="['/lazy-loading']">

Lazy Loading

</a>

</li>

1. Lazy load the component, which will be displayed on the route – /lazy-loading

Make necessary changes in **app-routing.module.ts**. Here we will load the module lazily using **loadChildren**

**app-routing.module.ts :**

{

path: 'lazy-loading',

loadChildren: () => import('./lazy-loading/lazy-loading.module')

.then(m => m.LazyLoadingModule)

},

1. Set up the route in **lazy-loading-routing.module.ts**.

**lazy-loading-routing.module.ts**

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

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

import { LazyDemoComponent } from './lazy-demo/lazy-demo.component';

const routes: Routes = [

{ path: '', component: LazyDemoComponent }

];

@NgModule({

imports: [RouterModule.forChild(routes)],

exports: [RouterModule]

})

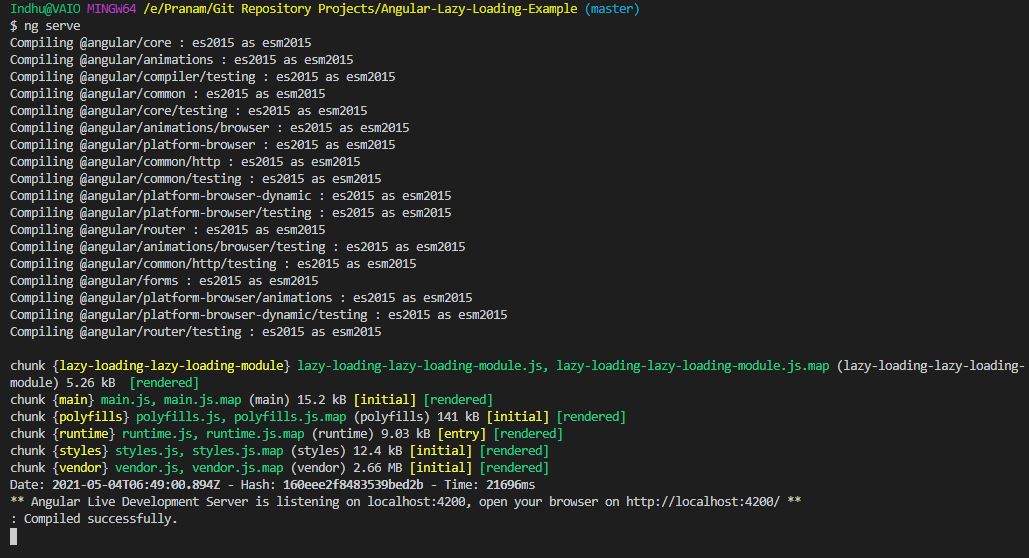
export class LazyLoadingRoutingModule { }

If the above import option gives error, try like this:

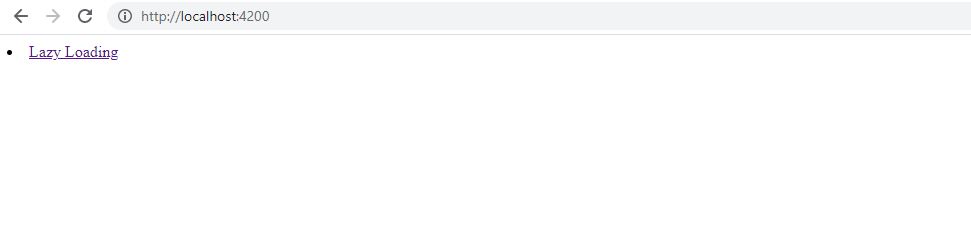
import { LazyDemoComponent } from "E:/Pranam/Git Repository Projects/Angular-Lazy-Loading-Example/src/app/lazy-demo/lazy-demo.component";

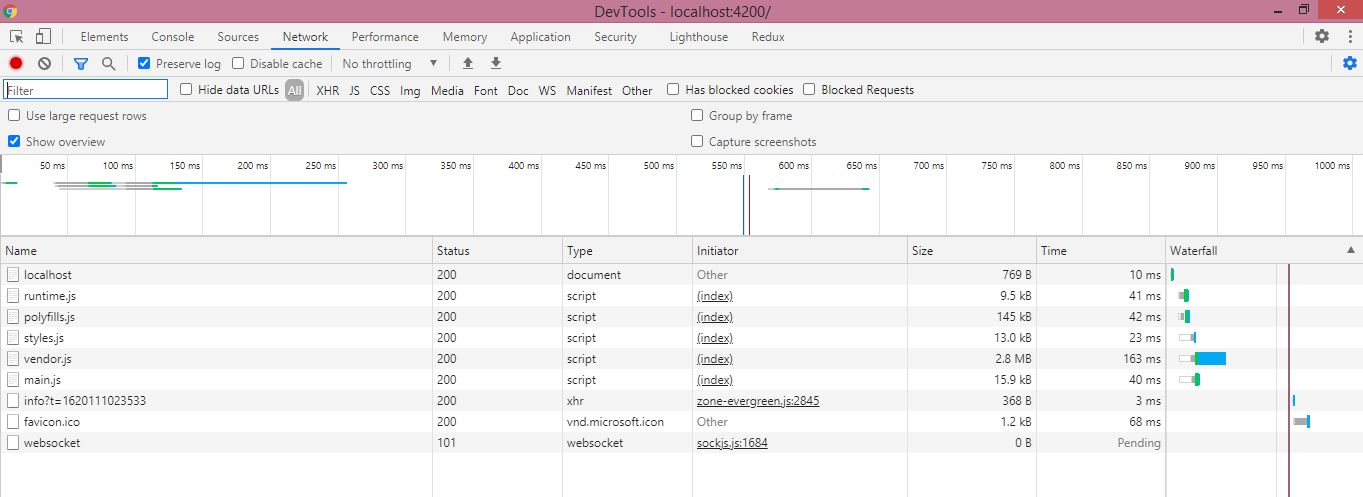
**It’s time to run our application.**

* Go to the terminal and execute “ng serve” command.

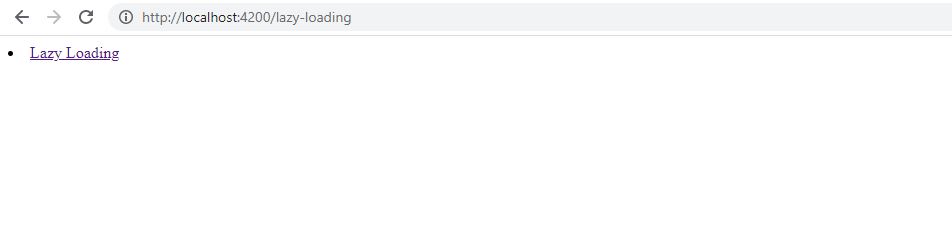


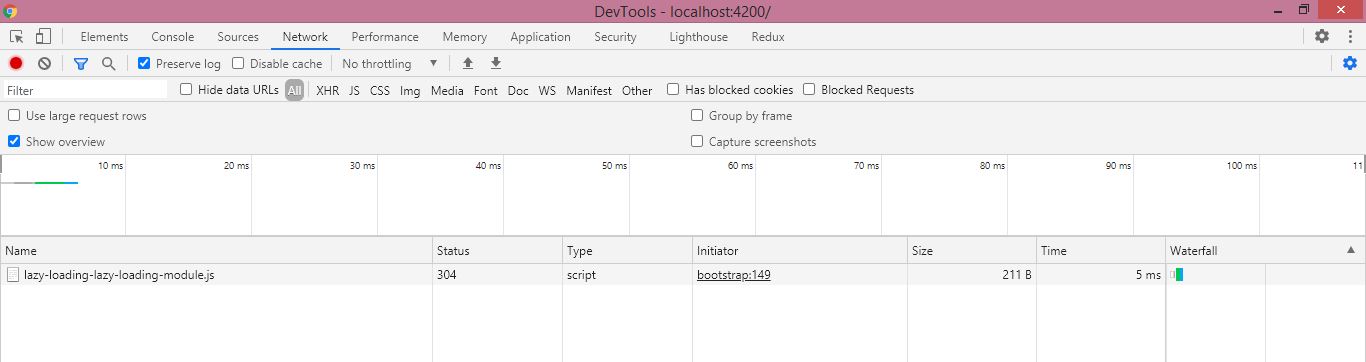
* Now go to the browser <https://localhost:4200>





You will observe that **main.js**is being served on refreshing the browser. And the **Lazy Loading** module is loaded only on hitting the **route /lazy-loading**.





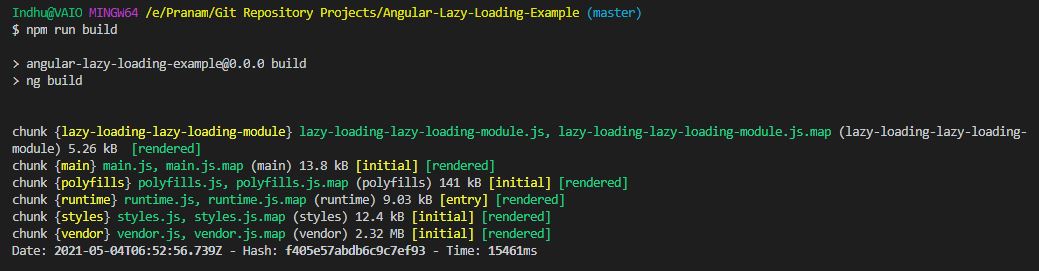
**How to verify Lazy loading in Angular?**

**Verify lazy loading in Angular:**

Run the following command to generate build:

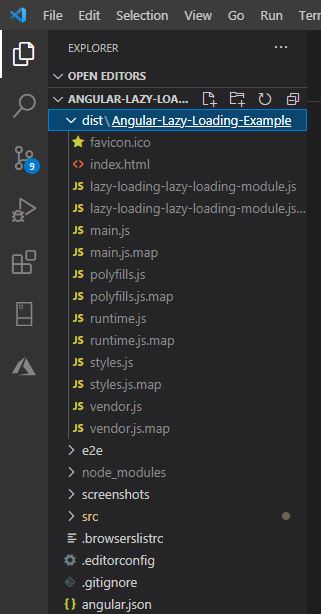
**npm run build**

And you’ll notice something like this –



The above image verifies that a separate chunk is generated for the lazy loading module.

Another way to verify is to open the dist folder of your project. There you will notice a separate file for the module which uses **Lazy Loading**.



**Conclusion:**

**Advantages:**

1. Minimizes start up time of the application.
2. Application consumes less memory because of on-demand loading.
3. Unnecessary request to server is avoided.

**Disadvantages:**

1. Firstly, the extra lines of code, to be added to the existing ones, to implement lazy load makes the code a bit complicated.
2. Additional code when integrating with JavaScript
3. User experience may be affected. For example, backtracking may not be possible if the page structure is not optimal.

**Tips:**

1. If possible, do not lazy load routes that are most frequently visited by your users.
2. Keep your chunk sizes as small as possible (Only import what you need and keep shared modules as compact as possible).
3. Use router states to show a loader or animation when a lazy loaded route is being fetched so they do not think your application lags. You can even handle cases when fetching the lazy loaded route fails e.g. Show a toast notification and redirect to a previously loaded route.