|  |  |
| --- | --- |
| **Angular Module Loading (Eager, Lazy and Preloading)** | |
|  | |
|  | A module can be loaded eagerly, lazily and preloaded. Eager loading is loading modules before application starts. Lazy loading is loading modules on demand. Preloading is loading modules in background just after application starts.  In lazy loading and preloading, modules are loaded asynchronously.  The application module i.e. AppModule is loaded eagerly before application starts. But the feature modules can be loaded either eagerly or lazily or preloaded. |
|  | E**ager loading**: To load a feature module eagerly we need to import it into application module using import metadata of @NgModule decorator. Eager loading is useful in small size applications. In eager loading, all the feature modules will be loaded before the application starts. Hence the subsequent request to the application will be faster.  **Lazy loading**: To load a feature module lazily we need to load it using loadChildren property in route configuration and that feature module must not be imported in application module. Lazy loading is useful when the application size is growing. In lazy loading, feature module will be loaded on demand and hence application start will be faster.  **Preloading**: To preload a feature module we need to load it using loadChildren property and configure preloadingStrategy property with RouterModule.forRoot. That feature module must not be imported in application module. When we assign Angular PreloadAllModules strategy to preloadingStrategy property, then all feature modules configured with loadChildren, are preloaded. To preload selective modules, we need to use custom preloading strategy. |
|  |  |
|  | In our demo application we have two feature modules CountryModule and PersonModule. We will create three demo applications. |
|  | Find the technologies being used in our example.  1. Angular 5.2.0  2. Angular CLI 1.7.3  3. TypeScript 2.5.3  4. Node.js 6.11.0  5. NPM 3.10.10 |
|  |  |
|  | Project Structure of Demo Application |
|  | my-app  |  |--src  | |  | |--app  | | |  | | |--country  | | | |  | | | |--country.component.ts  | | | |--country.ts  | | | |--country.module.ts  | | | |--country-routing.module.ts  | | | |  | | | |--country-list  | | | | |  | | | | |--country.list.component.html  | | | | |--country.list.component.ts  | | | |  | | | |--services  | | | | |  | | | | |--country.service.ts  | | |  | | |--person  | | | |  | | | |--person.component.ts  | | | |--person.ts  | | | |--person-routing.module.ts  | | | |--person.module.ts  | | | |  | | | |--person-list  | | | | |  | | | | |--person.list.component.html  | | | | |--person.list.component.ts  | | | | |  | | | |  | | | |--services  | | | | |  | | | | |--person.service.ts  | | |  | | |--address.component.ts  | | |--page-not-found.component.ts  | | |--app.component.ts  | | |--app-routing.module.ts  | | |--app.module.ts  | |  | |--main.ts  | |--index.html  | |--styles.css  |  |--node\_modules  |--package.json |
|  |  |
|  | **Eager Loading**  1. In eager loading module, feature modules are loaded before application start on the first hit. To load a feature module eagerly, we need to import that module in application module i.e. AppModule using imports metadata of @NgModule decorator.  2. When a module is loaded, it loads all the imported modules, configured components, services, custom pipes etc.  3. Modules are loaded in the order they are configured in imports metadata.  4. Eager loading is good for small applications because at the first hit of the application all the modules are loaded and all the required dependencies are resolved. Now the subsequent access to the application will be faster.   Now find the example. In the example we have two feature modules and we will load them eagerly.  Module and routing module for feature 1: |
|  | **country.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  import { CountryService } from './services/country.service';  import { CountryRoutingModule } from './country-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  CountryRoutingModule  ],  declarations: [  CountryComponent,  CountryListComponent  ],  providers: [ CountryService ]  })  export class CountryModule {  constructor() {  console.log('CountryModule loaded.');  }  } |
|  | **country-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  const countryRoutes: Routes = [  {  path: 'country',  component: CountryComponent,  children: [  {  path: 'country-list',  component: CountryListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(countryRoutes) ],  exports: [ RouterModule ]  })  export class CountryRoutingModule { } |
|  | Module and routing module for feature 2: |
|  | **person.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  import { PersonService } from './services/person.service';  import { PersonRoutingModule } from './person-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  PersonRoutingModule  ],  declarations: [  PersonComponent,  PersonListComponent  ],  providers: [ PersonService ]  })  export class PersonModule {  constructor() {  console.log('PersonModule loaded.');  }  } |
|  | **person-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  const personRoutes: Routes = [  {  path: 'person',  component: PersonComponent,  children: [  {  path: 'person-list',  component: PersonListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(personRoutes) ],  exports: [ RouterModule ]  })  export class PersonRoutingModule { } |
|  |  |
|  | To achieve eager loading, we will import features modules in AppModule using imports metadata of @NgModuledecorator. |
|  |  |
|  | **app.module.ts**  import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppComponent } from './app.component';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  import { CountryModule } from './country/country.module';  import { PersonModule } from './person/person.module';  import { AppRoutingModule } from './app-routing.module';  @NgModule({  imports: [  BrowserModule,  CountryModule,  PersonModule,  AppRoutingModule  ],  declarations: [  AppComponent,  AddressComponent,  PageNotFoundComponent  ],  providers: [ ],  bootstrap: [ AppComponent ]  })  export class AppModule {  constructor() {  console.log('AppModule loaded.');  }  }  The modules will be loaded in the order: BrowserModule -> CountryModule -> PersonModule-> AppRoutingModule.  Find the code for AppRoutingModule. |
|  |  |
|  | **app-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  const routes: Routes = [  {  path: 'address',  component: AddressComponent  },  {  path: '',  redirectTo: '/country',  pathMatch: 'full'  },  {  path: '\*\*',  component: PageNotFoundComponent  }  ];  @NgModule({  imports: [  RouterModule.forRoot(routes)  ],  exports: [  RouterModule  ]  })  export class AppRoutingModule { }  **Output** When we access the application first time, we will get following output in browser console. |
|  | CountryModule loaded.  PersonModule loaded.  AppModule loaded.  Angular is running in the development mode. Call enableProdMode() to enable the production mode.  We can see that before loading AppModule, all imported module in their order has been loaded. |
|  |  |
|  | **Lazy Loading** |
|  | **a. .** When the modules are loaded on-demand, then it is called lazy loading. It is loaded by using loadChildrenproperty in route configuration. In lazy loading, modules are loaded asynchronously. These modules must not be imported in application module otherwise they will be eagerly loaded.  **b.** In route configuration loadChildren property is used as following.  const routes: Routes = [  {  path: 'country',  loadChildren: 'app/country/country.module#CountryModule'  },  ------  ];  **c.** If the application size is growing and there are many feature modules then loading all feature modules eagerly will make application slow. What we can do, is we can load a feature module when it is requested. Such type of module loading is called lazy loading.   Now find the steps to perform lazy loading of feature modules in the previous application which is using eager loading.  **Step-1**: In route configuration of feature modules, change the parent path as empty string ("").  Module and routing module for feature 1: |
|  |  |
|  | **country-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  const countryRoutes: Routes = [  {  path: '',  component: CountryComponent,  children: [  {  path: 'country-list',  component: CountryListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(countryRoutes) ],  exports: [ RouterModule ]  })  export class CountryRoutingModule { } |
|  | **country.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  import { CountryService } from './services/country.service';  import { CountryRoutingModule } from './country-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  CountryRoutingModule  ],  declarations: [  CountryComponent,  CountryListComponent  ],  providers: [ CountryService ]  })  export class CountryModule {  constructor() {  console.log('CountryModule loaded.');  }  } |
|  | Module and routing module for feature 2:  **person-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  const personRoutes: Routes = [  {  path: '',  component: PersonComponent,  children: [  {  path: 'person-list',  component: PersonListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(personRoutes) ],  exports: [ RouterModule ]  })  export class PersonRoutingModule { } |
|  | **person.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  import { PersonService } from './services/person.service';  import { PersonRoutingModule } from './person-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  PersonRoutingModule  ],  declarations: [  PersonComponent,  PersonListComponent  ],  providers: [ PersonService ]  })  export class PersonModule {  constructor() {  console.log('PersonModule loaded.');  }  } |
|  | **Step-2**: Use loadChildren property to load feature modules in application routing module i.e. AppRoutingModule.  **app-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  const routes: Routes = [  {  path: 'country',  loadChildren: 'app/country/country.module#CountryModule'  },  {  path: 'person',  loadChildren: 'app/person/person.module#PersonModule'  },  {  path: 'address',  component: AddressComponent  },  {  path: '',  redirectTo: '',  pathMatch: 'full'  },  {  path: '\*\*',  component: PageNotFoundComponent  }  ];  @NgModule({  imports: [  RouterModule.forRoot(routes)  ],  exports: [  RouterModule  ]  })  export class AppRoutingModule { }  **Step-3**: Remove all references of feature modules from application module i.e. AppModule.  **app.module.ts**  import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppComponent } from './app.component';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  import { AppRoutingModule } from './app-routing.module';  @NgModule({  imports: [  BrowserModule,  AppRoutingModule  ],  declarations: [  AppComponent,  AddressComponent,  PageNotFoundComponent  ],  providers: [ ],  bootstrap: [ AppComponent ]  })  export class AppModule {  constructor() {  console.log('AppModule loaded.');  }  } |
|  |  |
|  | When we access the application first time, AppModule will be loaded eagerly. We will get log in browser console as following.  AppModule loaded.  Angular is running in the development mode. Call enableProdMode() to enable the production mode.  When we click on country, then CountryModule will be loaded lazily. Find the log.  CountryModule loaded.  When we click on person, then PersonModule will be loaded lazily. Find the log.  PersonModule loaded. |
|  | **Preloading** |
|  | **1.** In preloading, feature modules are loaded in background asynchronously. In preloading, modules start loading just after application starts.  **2.** When we hit the application, first AppModule and modules imported by it, will be loaded eagerly. Just after that modules configured for preloading is loaded asynchronously.  **3.** Preloading is useful to load those features which are in high probability to be visited by user just after loading the application.  **4.** To configure preloading, angular provides preloadingStrategy property which is used with RouterModule.forRootin routing module. Find the code snippet. |
|  |  |
|  | @NgModule({  imports: [  RouterModule.forRoot(routes,  {  preloadingStrategy: PreloadAllModules  })  ],  ------  })  export class AppRoutingModule { } |
|  | **5.** To configure preloading features modules, first we will configure them for lazy loading and then using Angular in-built PreloadAllModules strategy, we enable to load all lazy loading into preloading modules.  **6.** Using PreloadAllModules strategy, all modules configured by loadChildren property will be preloaded. The modules configured by loadChildren property will be either lazily loaded or preloaded but not both. To preload only selective modules, we need to use custom preloading strategy.  **7.** Once we configure PreloadAllModules strategy, then after eager loading modules, Angular searches for modules applicable for preloading. The modules configured by loadChildren will be applicable for preloading. We will take care that these feature modules are not imported in application module i.e. AppModule.  **8.** We can create custom preloading strategy. For this we need to create a service by implementing Angular PreloadingStrategy interface and override its preload method and then configure this service with preloadingStrategy property in routing module. To select a module for custom preloading we need to use dataproperty in route configuration. data can be configured as data: { preload: true } for selective feature module preloading. |
|  | For the demo, here we will use in-built preloading strategy i.e. PreloadAllModules strategy. Find the example.  Module and routing module for feature 1: |
|  | **country.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  import { CountryService } from './services/country.service';  import { CountryRoutingModule } from './country-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  CountryRoutingModule  ],  declarations: [  CountryComponent,  CountryListComponent  ],  providers: [ CountryService ]  })  export class CountryModule {  constructor() {  console.log('CountryModule loaded.');  }  }  **country-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { CountryComponent } from './country.component';  import { CountryListComponent } from './country-list/country.list.component';  const countryRoutes: Routes = [  {  path: '',  component: CountryComponent,  children: [  {  path: 'country-list',  component: CountryListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(countryRoutes) ],  exports: [ RouterModule ]  })  export class CountryRoutingModule { } |
|  | Module and routing module for feature 2: |
|  | **person.module.ts**  import { NgModule } from '@angular/core';  import { CommonModule } from '@angular/common';  import { ReactiveFormsModule } from '@angular/forms';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  import { PersonService } from './services/person.service';  import { PersonRoutingModule } from './person-routing.module';  @NgModule({  imports: [  CommonModule,  ReactiveFormsModule,  PersonRoutingModule  ],  declarations: [  PersonComponent,  PersonListComponent  ],  providers: [ PersonService ]  })  export class PersonModule {  constructor() {  console.log('PersonModule loaded.');  }  } |
|  | **person-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { PersonComponent } from './person.component';  import { PersonListComponent } from './person-list/person.list.component';  const personRoutes: Routes = [  {  path: '',  component: PersonComponent,  children: [  {  path: 'person-list',  component: PersonListComponent  }  ]  }  ];  @NgModule({  imports: [ RouterModule.forChild(personRoutes) ],  exports: [ RouterModule ]  })  export class PersonRoutingModule { } |
|  | Now find the AppModule and AppRoutingModule.  **app.module.ts**  import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppComponent } from './app.component';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  import { AppRoutingModule } from './app-routing.module';  @NgModule({  imports: [  BrowserModule,  AppRoutingModule  ],  declarations: [  AppComponent,  AddressComponent,  PageNotFoundComponent  ],  providers: [ ],  bootstrap: [ AppComponent ]  })  export class AppModule {  constructor() {  console.log('AppModule loaded.');  }  }  **app-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router';  import { PreloadAllModules } from '@angular/router';  import { AddressComponent } from './address.component';  import { PageNotFoundComponent } from './page-not-found.component';  const routes: Routes = [  {  path: 'country',  loadChildren: 'app/country/country.module#CountryModule'  },  {  path: 'person',  loadChildren: 'app/person/person.module#PersonModule'  },  {  path: 'address',  component: AddressComponent  },  {  path: '',  redirectTo: '',  pathMatch: 'full'  },  {  path: '\*\*',  component: PageNotFoundComponent  }  ];  @NgModule({  imports: [  RouterModule.forRoot(routes,  {  preloadingStrategy: PreloadAllModules  })  ],  exports: [  RouterModule  ]  })  export class AppRoutingModule { } |
|  |  |
|  | We can see in the AppRoutingModule that we are using PreloadAllModules strategy for preloading. The module configured by loadChildren i.e. CountryModule and PersonModule will be preloaded.  **Output** When we hit the application for first time, we can see following logs in browser console.  AppModule loaded.  Angular is running in the development mode. Call enableProdMode() to enable the production mode.  CountryModule loaded.  PersonModule loaded. |
|  |  |
|  | Components and Services used in Demo Application |
|  | **country.ts**  export class Country {  constructor(public countryId:number, public countryName:string,  public capital:string, public currency:string) {  }  } |
|  | **country.service.ts**  import { Injectable } from '@angular/core';  import { Observable } from 'rxjs/Observable';  import { of } from 'rxjs/observable/of';  import { Country } from '../country';  const COUNTRIES = [  new Country(1, 'India', 'New Delhi', 'INR'),  new Country(2, 'China', 'Beijing', 'RMB')  ];  let countryList$ = of(COUNTRIES);  @Injectable()  export class CountryService {  getCountries(): Observable<Country[]> {  return countryList$;  }  } |
|  | **country.list.component.ts**  import { Component, OnInit } from '@angular/core';  import { Observable } from 'rxjs/Observable';  import { CountryService } from '../services/country.service';  import { Country } from '../country';  @Component({  templateUrl: './country.list.component.html'  })  export class CountryListComponent implements OnInit {  countries$: Observable<Country[]>  constructor(private countryService: CountryService) {}  ngOnInit() {  this.countries$ = this.countryService.getCountries();  }  } |
|  | **country.list.component.html**  <h3>Country List</h3>  <p \*ngFor="let country of countries$ | async">  {{country.countryId}}. {{country.countryName}}  - {{country.capital}} - {{country.currency}}  </p>  **country.component.ts**  import { Component } from '@angular/core';  @Component({  template: `  <h2>Welcome to Country Home</h2>  <a [routerLink]="['country-list']" routerLinkActive="active">View Country List</a>  <router-outlet></router-outlet>  `  })  export class CountryComponent {  }  Code for person feature module:  **person.ts**  export class Person {  constructor(public personId:number, public name:string, public city:string) {  }  }  **person.service.ts**  import { Injectable } from '@angular/core';  import { of } from 'rxjs/observable/of';  import { Observable } from 'rxjs/Observable';  import { Person } from '../person';  const PERSONS = [  new Person(1, 'Mahesh', 'Varanasi'),  new Person(2, 'Ram', 'Ayodhya'),  new Person(3, 'Kishna', 'Mathura')  ];  let personList$ = of(PERSONS);  @Injectable()  export class PersonService {  getPersons(): Observable<Person[]> {  return personList$;  }  } |
|  | **person.list.component.ts**  import { Component, OnInit } from '@angular/core';  import { Observable } from 'rxjs/Observable';  import { PersonService } from '../services/person.service';  import { Person } from '../person';  @Component({  templateUrl: './person.list.component.html'  })  export class PersonListComponent implements OnInit {  persons$: Observable<Person[]>  constructor(private personService: PersonService) {}  ngOnInit() {  this.persons$ = this.personService.getPersons();  }  }  **person.list.component.html**  <h3>Person List</h3>  <p \*ngFor="let person of persons$ | async">  {{person.personId}}. {{person.name}} - {{person.city}}  </p>  **person.component.ts**  import { Component } from '@angular/core';  @Component({  template: `  <h2>Welcome to Person Home</h2>  <a [routerLink]="['person-list']" routerLinkActive="active">View Person List</a>  <router-outlet></router-outlet>  `  })  export class PersonComponent {  }  Now find other files.  **address.component.ts**  import { Component } from '@angular/core';  @Component({  template: `  <h3>ADDRESS</h3>  <p><b> Article: Angular Module Loading </b></p>  <p><b> Category: Angular </b></p>  <p><b> Website: CONCRETEPAGE.COM </b></p>  <div>  <a [routerLink]="['/location']">Find Location</a>  </div>  `  })  export class AddressComponent {  }  **page-not-found.component.ts**  import { Component } from '@angular/core';  import { Location } from '@angular/common';  @Component({  template: `<h2>Page Not Found.</h2>  <div>  <button (click)="goBack()">Go Back</button>  </div>  `  })  export class PageNotFoundComponent {  constructor(private location: Location) { }  goBack(): void {  this.location.back();  }  } |
|  | **styles.css**  .parent-menu ul {  list-style-type: none;  margin: 0;  padding: 0;  overflow: hidden;  background-color: #333;  }  .parent-menu li {  float: left;  }  .parent-menu li a {  display: block;  color: white;  text-align: center;  padding: 15px 15px;  text-decoration: none;  }  .parent-menu li a:hover:not(.active) {  background-color: #111;  }  .parent-menu .active{  background-color: #4CAF50;  }  .parent-container {  padding-left: 10px;  }  **app.component.ts**  import { Component } from '@angular/core';  @Component({  selector: 'app-root',  template: `  <nav [ngClass] = "'parent-menu'">  <ul>  <li><a routerLink="/country" routerLinkActive="active">Country</a></li>  <li><a routerLink="/person" routerLinkActive="active">Person</a></li>  <li><a routerLink="/address" routerLinkActive="active">Address</a></li>  </ul>  </nav>  <div [ngClass] = "'parent-container'">  <router-outlet></router-outlet>  </div>  `  })  export class AppComponent {  }  **Run Application**  To run the application, find the steps.  **1.** Download source code using download link given below on this page.  **2.** Use downloaded **src** in your Angular CLI application. To install Angular CLI, find the [link](https://angular.io/guide/quickstart).  **3.** Run **ng serve** using command prompt.  **4.** Access the URL [**http://localhost:4200**](http://localhost:4200) |