**Lesson32 Angular custom preloading strategy**

**1-we have 3 methods of the preloading Strategy**

**A- PreloadAllModules //It’s activate the preload**

**B- preloadingStrategy: NoPreloading //It’s enable the lazy loading**

**C- preloadingStrategy: CustomElementRegistry**

**//To apply custom method to enable preload or lazy loading**

**Steps:-**

**1-To apply custom preloading strategy, first we create custom service and apply implements**

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

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

import { PageNotFoundCompComponent } from './components/page-not-found-comp/page-not-found-comp.component';

import { HomeComponentComponent } from './components/MainComponent/home-component/home-component.component';

import { CustomPreloadingServiceService } from './Services/custom-preloading-service.service';

const routes: Routes = [

{ path: 'home', component: HomeComponentComponent },

{ path: '', redirectTo: '/home', pathMatch: 'full' },

//In the preloading strategy type if it’s preloading 🡪 value true , or lazy loading 🡪 value false below route we pass the parameter to detect the

{ path: 'employees',data:{preloading:false} ,loadChildren: './components/Employee/employee.module#EmployeeModule' },

{ path: '\*\*', component: PageNotFoundCompComponent }];

@NgModule({

//we see that we apply custom preloading service

imports: [RouterModule.forRoot(routes,{ preloadingStrategy: CustomPreloadingServiceService })],

exports: [RouterModule]})

export class AppRoutingModule { }

**2-we create service on the folder service and apply code**

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

import { PreloadingStrategy, Route } from '@angular/router';

import { Observable, of } from 'rxjs';

@Injectable({

providedIn: 'root'})

export class CustomPreloadingServiceService implements PreloadingStrategy{

constructor() { }

// PreloadingStrategy abstract class has the following preload()

// abstract method for which we need to provide implementation

preload(route: Route, fn: () => Observable<any>): Observable<any> {

// If data property exists on the route of the lazy loaded module

// and if that data property also has preload property set to

// true, then return the fn() which preloads the module

if (route.data && route.data['preloading']) {

return fn();

// If data property does not exist or preload property is set to

// false, then return Observable of null, so the module is not

// preloaded in the background} else {return of(null);}}