What is Routing in Angular?

Routing in Angular is used to navigate between different views (components) in a Single Page Application (SPA). It allows users to move between pages without reloading the entire application.

Key Features of Angular Routing:

- Maps URLs to components.
- Allows navigation without reloading the page.
- Supports lazy loading for performance optimization.
- Can pass parameters between routes.
- Enables route guards for authentication & authorization.

---

2. Why Do We Need Routing?

1. **Single Page Application (SPA):**
   - In SPAs, the content changes dynamically without reloading the full page.
2. **Better User Experience:**
   - Improves performance and reduces unnecessary page reloads.
3. **SEO & Bookmarking:**
   - Routes enable deep linking and help in bookmarking pages.
4. **Modular Structure:**
   - Helps in organizing code efficiently by dividing it into separate views.

---

3. Setting Up Routing in Angular

Generate Components

Create different components for navigation:

```
ng g c home
ng g c about
ng g c contact
```

Configuring Routing in Angular

Step 1: Import `RouterModule` and Define Routes

Modify `app-routing.module.ts`:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

const routes: Routes = [
  { path: '', component: HomeComponent }, // Default route
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Step 2: Add `<router-outlet>` in `app.component.html`

```
<h1>Angular Routing Example</h1>
<nav>
  <a routerLink="/">Home</a> |
  <a routerLink="/about">About</a> |
  <a routerLink="/contact">Contact</a>
</nav>
<hr>
<router-outlet></router-outlet>
```

`router-outlet` is a placeholder where the routed components will be displayed.

Step 3: Add Routing Module in `app.module.ts`

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Now, you can navigate between Home, About, and Contact pages without reloading!

Redirect route

Angular routing allows one Path to be redirected to another. There is an option to set the redirection path to redirect. The route is as follows -

```
const routes: Routes = [

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

];
```

Here,

If the actual Path matches an empty string, the redirect is set as the redirect path.

Wildcard route

The wildcard route will match any path. It is built using ** and will be used to handle non-existing paths in the application. It is called if the second Path does not match by placing a wildcard route at the end of the configuration.

The sample code is below -

```
const routes: Routes = [
  { path: 'about', component: AboutComponent },
  { path: '',   redirectTo: '/about', pathMatch: 'full' },
  { path: ‘**', component: PageNotFoundComponent },  // Wildcard route for a 404 page
];
```

If a non-existent page is called, the first two routes fail. But, the last wildcard route will succeed, and PageNotFoundComponent will be called.

Navigating Programmatically

We can navigate dynamically using `Router`.

Example in `about.component.ts`:

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-about',
  template: `<button (click)="goToContact()">Go to Contact</button>`
})
export class AboutComponent {
  constructor(private router: Router) {}

  goToContact() {
    this.router.navigate(['/contact']);
  }
}
```

Passing Parameters in Routes

1. Create Components

```
ng generate component home
ng generate component product
ng generate component page-not-found
```

---

2. Configure Routing

Modify `app-routing.module.ts` to define a route with a parameter.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { ProductComponent } from './product/product.component';
import { PageNotFoundComponent } from './page-not-found/page-not-
found.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'product/:id', component: ProductComponent }, // Route parameter
  { path: '**', component: PageNotFoundComponent }, // 404 page
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

---

3. Create Links with Route Parameters

Modify `home.component.html` to navigate with dynamic route parameters.

```
<h2>Home Page</h2>
<ul>
  <li><a [routerLink]="['/product', 101]">Product 101</a></li>
  <li><a [routerLink]="['/product', 202]">Product 202</a></li>
  <li><a [routerLink]="['/product', 303]">Product 303</a></li>
</ul>
```

4. Capture Route Parameters in the Component

Modify `product.component.ts` to read the parameter from the URL.

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.css']
})
export class ProductComponent implements OnInit {
  productId: number = 0;

  constructor(private route: ActivatedRoute) {}

  ngOnInit() {

    // Capture route parameter from URL

      this.productId = Number(this.route.snapshot.paramMap.get('id'));
    }
/*
 ngOnInit() {
    // Capture route parameter from URL
    this.route.paramMap.subscribe(params => {
      this.productId = Number(params.get('id'));
    });
  }
*/
}
```

Limitation of snapshot : If the route parameter changes without reloading the component, snapshot won't detect it.

---

5. Display the Captured Parameter

Modify `product.component.html` to show the received parameter.

```
<h2>Product Details</h2>
<p>Product ID: {{ productId }}</p>
<a routerLink="/">Go Back</a>
```

Lay Loading

1. Generate a Feature Module and a Component

Generate the **user module** with routing:

- Create a `user` module.

- Generate a `user-routing.module.ts` for routing.
- Create a `UserProfileComponent`.

3. Configure Lazy Loading in `app-routing.module.ts`

Modify `app-routing.module.ts` to load the `UserModule` **lazily**.

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

const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'user', loadChildren: () => import('./user/user.module').then(m
=> m.UserModule) }, // Lazy loading
  { path: '**', redirectTo: '/home' } // Wildcard route for unmatched paths
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

4. Define Child Routes in `user-routing.module.ts`

Modify `user-routing.module.ts` to set up child routes for the **UserModule**.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { UserProfileComponent } from './user-profile/user-
profile.component';

const routes: Routes = [
  { path: 'profile', component: UserProfileComponent }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class UserRoutingModule { }
```

5. Modify `user.module.ts`

Ensure that `UserModule` imports `UserRoutingModule` so it works with lazy loading.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { UserProfileComponent } from './user-profile/user-
profile.component';
```

```
import { UserRoutingModule } from './user-routing.module';

@NgModule({
  declarations: [UserProfileComponent],
  imports: [
    CommonModule,
    UserRoutingModule
  ]
})
export class UserModule { }
```

6. Modify `app.component.html`

Add navigation to the lazily loaded `UserModule`.

```
<h2>Lazy Loading Example</h2>
<nav>
  <a routerLink="/">Home</a> |
  <a routerLink="/user/profile">User Profile</a>
</nav>
<router-outlet></router-outlet>
```

7. Modify `user-profile.component.html`

Display a message to confirm that the `UserProfileComponent` is loaded.

```
<h3>Welcome to the User Profile Page!</h3>
<a routerLink="/">Go Back to Home</a>
```

Assignment : Online Learning Platform

Scenario:

You are building a **simple online learning platform** where users can navigate between different sections like **Home, Courses, and Profile** using **Angular Modules and Routing**.

Requirements:

1. **Create an Angular app** with separate modules for `Home`, `Courses`, and `Profile`.
2. **Implement routing** so users can navigate between pages.
3. **Use RouterLink** to navigate without refreshing the page.
4. **Pass a dynamic Course ID** as a route parameter.
5. **Protect the Profile page** with a simple authentication guard.

Expected Features

1. Users can **navigate** between Home, Courses, and Profile.
2. Clicking a **course** shows detailed information.
3. **Profile page is protected**, requiring a login.
4. **Lazy Loading** improves performance.
5. **404 Page** displays for unknown routes.