Angular Routing and Navigation

February 23, 2025

What We'll Cover:

1. Setting Up Angular Routing (ng generate module app-routing --routing)
2. Defining Routes in app-routing.module.ts
3. Using <router-outlet> and <a routerLink>
4. Passing Parameters in Routes (e.g., /details/:id)
5. Lazy Loading Modules for Performance

Setting Up Angular Routing

- **Concept**: Routing lets users navigate between pages
- **How**: Use Angular CLI to set it up
- **Command**:

```
ng generate module app-routing --routing
```

Output: Creates app-routing.module.ts

Setup:

- Imports RouterModule
- Configures routes

Key Point: Routing is optional but essential for multi-page apps

Defining Routes

- **Where**: app-routing.module.ts
- **Syntax**: Array of route objects
- **Example**:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { TasksComponent } from './tasks/tasks.component';
const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'tasks', component: TasksComponent }];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

Explanation:

- '' = Default route (home)
- 'tasks' = Tasks page

Note: Assumes standalone components are imported elsewhere

Using <router-outlet> and <a routerLink>

- **<router-outlet>**: Placeholder for route content
- **<a routerLink>**: Navigation links
- **Example (App Component)**:

```
@Component({
  standalone: true,
  imports: [RouterModule],
  template: `
        <nav>
        <a routerLink="/">Home</a> |
        <a routerLink="/tasks">Tasks</a>
        </nav>
        <router-outlet></router-outlet>
  `
})
export class AppComponent {}
```

Output: Click "Tasks" → Shows TasksComponent

Key Point: No page reload—single-page app magic!

Passing Parameters in Routes

- **Concept**: Send data via URL (e.g., /details/:id)
- **Define Route**:

```
{ path: 'details/:id', component: TaskDetailComponent }
```

Link:

```
<a routerLink="/details/1">Task 1</a>
```

Access Parameter:

```
import { ActivatedRoute } from '@angular/router';
@Component({
  standalone: true,
  template: `<p>Task ID: {{ id }}</p>`
})
export class TaskDetailComponent {
  id: string;
  constructor(route: ActivatedRoute) {
        this.id = route.snapshot.paramMap.get('id') || '';
  }
}
```

Use Case: Show details for a specific item

Lazy Loading Modules

- **Concept**: Load features only when needed
- **Why**: Faster initial load, better performance
- **Setup**:

```
const routes: Routes = [
  { path: '', component: HomeComponent },
  {
      path: 'tasks',
      loadChildren: () => import('./tasks/tasks.module').then(m => m.TasksModule)
  }
];
```

Tasks Module:

```
@NgModule({
  imports: [RouterModule.forChild([{ path: '', component: TasksComponent }])],
  declarations: [TasksComponent]
})
export class TasksModule {}
```

Key Point: loadChildren delays loading until route is accessed

Putting It Together

- **Mini-Project**: Task app with routing
- **Routes**:

```
const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'tasks', component: TasksComponent },
  { path: 'task/:id', component: TaskDetailComponent }
];
```

App Component:

```
@Component({
  standalone: true,
  imports: [RouterModule],
  template: `
        <nav>
        <a routerLink="/">Home</a> |
        <a routerLink="/tasks">Tasks</a>
        </nav>
        <router-outlet></router-outlet>
  `
})
export class AppComponent {}
```

Tasks Component:

template: `<a routerLink="/task/1">See Task 1</a>`

Demo: Navigate between home, tasks, and task details

Key Takeaways

- Routing enables multi-page navigation
- Define routes in app-routing.module.ts
- Use <router-outlet> for content, <a routerLink> for links
- Pass parameters with :id in paths
- Lazy loading boosts performance with loadChildren