Lecture 17: Routing in Angular

What is Routing in Angular?

- Enables navigation between different views/components in an SPA (Single Page Application)
- Helps in creating a dynamic and interactive user experience
- Core Concepts:
    - **RouterModule –** Configures the application’s routing
    - **Routes –** Defines mapping between URLs and components
    - **RouterOutlet –** Placeholder for routing views
    - **RouterLink –** Directive to navigate between routes

Setting up Routing

Steps to Add Routing in an Angular App.

1. Import **RouterModule** and **Routes** from **@angular/router.**
2. Define an array of route objects.
3. Add **RouterModule.forRoot(routes)** in the **@NgModule** imports.
4. Use **<router-outlet>** in your **AppComponent** template.

Setting up Routing

```
const routes: Routes = [
  { path: '', component: AppComponent },
  { path: 'about', component: AppComponent },
  { path: '**', component: AppComponent }
];

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

Navigating between Routes

- **Using RouterLink**

```
<nav>
    <a routerLink="/">Home</a>
    <a routerLink="/about">About</a>
</nav>
```

- **Using Router in TypeScript**

```
navigateToAbout() {
  this.router.navigate(['/about']);
}
```

What are Route Guards

- Mechanisms to control access to specific routes
- Implemented as Angular services that return a boolean or observable
- Common Use Cases:
    - **Authentication** (restrict access for unauthorized users
    - **Authorization** (restrict based on roles/permissions
    - **Prevent Navigation** (e.g. unsaved form data)

Implementing CanActivate

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';

@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate {
  constructor(private router: Router) {}

  canActivate(): boolean {
    const isAuthenticated = // logic to check auth state;
    if (!isAuthenticated) {
      this.router.navigate(['/login']);
      return false;
    }
    return true;
  }
}
```

Implementing CanActivateChild

```
@Injectable({
  providedIn: 'root',
})
export class AdminGuard implements CanActivateChild {
  canActivateChild(): boolean {
    const hasPermission = true // logic for permission check;
    return hasPermission;
  }
}
```

Implementing CanLoad

```
@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanLoad {
  constructor(private router: Router) {}

  canLoad(): boolean {
    const isAuthenticated = true // check authentication logic;
    return isAuthenticated;
  }
}
```

```
const routes: Routes = [
  {
    path: 'admin',
    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule),
    canLoad: [AuthGuard],
  },
];
```