**Chapter 8: Routing & Navigation**

**Angular Routing** is a way to navigate between views in a single-page application (SPA) — without reloading the browser.

It mimics the behavior of multiple pages while everything actually runs in one HTML file.

**How to Set Up Routing (Angular 17+ Standalone)**

**1. Define your routes in a separate file (e.g. app.routes.ts)**

import { Routes } from '@angular/router';

import { HomeComponent } from './home.component';

import { ProductDetailComponent } from './product-detail.component';

import { NotFoundComponent } from './not-found.component';

export const routes: Routes = [

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

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

{ path: 'product/:id', component: ProductDetailComponent },

{ path: '\*\*', component: NotFoundComponent }

];

**2. Import routes in main.ts**

import { bootstrapApplication } from '@angular/platform-browser';

import { provideRouter } from '@angular/router';

import { AppComponent } from './app/app.component';

import { routes } from './app/app.routes';

bootstrapApplication(AppComponent, {

providers: [provideRouter(routes)]

});

**Navigation Basics**

**Template Navigation: routerLink**

<a routerLink="/home">Home</a>

<a [routerLink]="['/product', 42]">Product 42</a>

**Navigation Area: <router-outlet>**

<router-outlet></router-outlet>

**Active Link Styling: routerLinkActive**

<a routerLink="/home" routerLinkActive="active">Home</a>

**Route Parameters**

**1. Route Path with Parameter**

{ path: 'product/:id', component: ProductDetailComponent }

**2. Accessing Parameters**

constructor(private route: ActivatedRoute) {

this.route.paramMap.subscribe(params => {

console.log(params.get('id'));

});

}

**Query Parameters**

**Example: /product/42?highlight=true**

this.route.queryParamMap.subscribe(query => {

const highlight = query.get('highlight') === 'true';

});

**Redirection & Wildcard Routing**

**A. Redirect to another route**

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

* **pathMatch: 'full'** means match the whole empty path.
* **prefix** is default but generally used for route guards, not redirects.

**B. Wildcard route (\*\*)**

Handles any unknown or invalid URL.

{ path: '\*\*', component: NotFoundComponent }

Use this to create a **custom 404 page**.

**Nested Routes (Children)**

**Use Case: Admin Panel with sub-pages**

{

path: 'admin',

children: [

{ path: '', component: AdminDashboardComponent },

{ path: 'users', component: AdminUsersComponent },

{ path: 'settings', component: AdminSettingsComponent }

]

}

**Access as:**

* /admin
* /admin/users
* /admin/settings

You can have <router-outlet> inside the AdminComponent to render child routes.

**Full Working Example**

**app.routes.ts**

export const routes: Routes = [

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

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

{ path: 'product/:id', component: ProductDetailComponent },

{

path: 'admin',

children: [

{ path: '', component: AdminDashboardComponent },

{ path: 'users', component: AdminUsersComponent },

{ path: 'settings', component: AdminSettingsComponent }

]

},

{ path: '\*\*', component: NotFoundComponent }

];

**Programmatic Navigation**

You can also navigate using the Router service:

constructor(private router: Router) {}

goToProduct(id: number) {

this.router.navigate(['/product', id], { queryParams: { highlight: true } });

}

**Tips for Beginners**

* Always place the wildcard route \*\* **at the end**.
* Use **pathMatch: 'full'** for redirecting empty paths.
* Use **standalone components** for route targets in Angular 17+ apps with CLI 19.
* Use **routerLinkActive** to indicate which nav item is currently active.
* Use **ActivatedRoute** to read both route parameters and query strings dynamically.