Here’s a complete breakdown of the **Angular Router lifecycle**, with a practical guide to using **Route Guards** and **Resolvers**.

**🔄 Angular Router Lifecycle (Step-by-Step)**

Whenever navigation occurs via router.navigate() or a router link ([routerLink]), the **router follows this sequence**:

| **Phase** | **Description** |
| --- | --- |
| 1. **NavigationStart** | Navigation is triggered. |
| 2. **Route Recognized** | Router matches the route. |
| 3. **Guards Check** | Runs CanActivate, CanActivateChild, and CanDeactivate guards. |
| 4. **Resolvers Run** | Runs Resolve<T> before component loads. |
| 5. **Component Instantiation** | Component is created and ngOnInit() is called. |
| 6. **NavigationEnd** | Finalized successfully, emits NavigationEnd event. |

**✅ Route Guards**

**🔐 Purpose:**

* Control access to routes (auth, role-based, unsaved changes, etc.)

**🔸 CanActivate – guard before entering route**

// auth.guard.ts

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 isLoggedIn = !!localStorage.getItem('token');

if (!isLoggedIn) {

this.router.navigate(['/login']);

return false;

}

return true;

}

}

**🧭 Using Guards in Routes**

// app-routing.module.ts

import { AuthGuard } from './guards/auth.guard';

const routes: Routes = [

{

path: 'dashboard',

component: DashboardComponent,

canActivate: [AuthGuard]

}

];

**✅ Route Resolvers**

**📦 Purpose:**

* Pre-fetch data **before** the component is loaded.

// user.resolver.ts

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

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

import { Observable, of } from 'rxjs';

import { UserService } from './user.service';

@Injectable({ providedIn: 'root' })

export class UserResolver implements Resolve<any> {

constructor(private userService: UserService) {}

resolve(): Observable<any> {

return this.userService.getUserDetails(); // API call

}

}

**Use the Resolver in Routes:**

const routes: Routes = [

{

path: 'profile',

component: ProfileComponent,

resolve: {

user: UserResolver

}

}

];

**Read Resolved Data in Component:**

// profile.component.ts

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

export class ProfileComponent implements OnInit {

user: any;

constructor(private route: ActivatedRoute) {}

ngOnInit(): void {

this.user = this.route.snapshot.data['user'];

}

}

**🧠 Summary: Guard vs Resolver**

| **Feature** | **Guard (CanActivate)** | **Resolver** |
| --- | --- | --- |
| Runs **before** route | ✅ | ✅ |
| Controls access? | ✅ | ❌ |
| Fetches data? | ❌ | ✅ |
| Blocks navigation if fails? | ✅ | ❌ (unless exception occurs) |

**✅ Optional: Other Guards**

| **Guard Type** | **Use Case** |
| --- | --- |
| CanActivateChild | Guard child routes |
| CanDeactivate | Prevent leaving route with unsaved changes |
| CanLoad | Control loading of lazy modules |
| CanMatch (Angular 15+) | Conditional route match before loading |