Angular 16 : Online Class

Route Guards

Presented by : Chandan Kumar

By: Sahosoft Solutions

Route Guards

Being web application developers, we are aware how a Server checks the permissions for the user on the navigation and returns the result Telling whether a user can access the resource or not.

In the same way, we need to achieve this on client-side. Route Guards are the solution for that. We can use the route guards to control whether the user can navigate the route or not.

Angular's route guards are interfaces which can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class which implements the given guard interface.

In the Angular application in which authentication and authorization is required to navigate a route, the role of Angular route guard comes into the picture.

There are five different types of guards and each of them is called in a particular sequence.

The router's behavior is modified differently depending on which guard is used.

Route Guards

Guard Types

There are 5 types of Route Guards we use in Angular.

CanActivate
This helps to decide whether the route can be activated or not.

CanActivateChild
This helps to decide whether child routes can be activated or not.

CanDeactivate
This helps to decide whether the route can be deactivated or not. Generally, this is used to warn a user if they are navigating from the component.

Resolve
This performs route data retrieval before any activation of the route.

CanLoad
Checks to see if the user can navigate to the module which is lazily loaded.

Route Guards

Guard Types

In Short:

- ✓ CanActivate guard (e.g. it checks route access).
- ✓ CanActivateChild guard (checks child route access).
- ✓ CanDeactivate guard (prompt for unsaved changes).
- ✓ Resolve guard (pre-fetching route data).
- ✓ CanLoad guard (check before loading feature module).

Before starting to implement the Guards, let's have some basic information about route guards.

- ❑ Guards are always implemented as service that needs to be provided so we always make them *@Injectable* while creation.
- ❑ Guards always return true or false telling you if access is allowed or not
- ❑ Guards can also return the Observables or Promises which can resolve into the Boolean value finally and will be returned.
- ❑ Every Route guard must be imported to the application or Root Module
- ❑ As the Route guards are services they must be registered in the Providers section of the Application module.

Route Guards

To sum up we can add the Route Guards in Following ways

- ✓ To Restrict Access or to warn before the navigation we can use the Route Guards depending on our need.
- ✓ We can use the Interface canActivate,canDeactivate,canActivateChild for that purpose.
- ✓ Route Guards are injectable classes which can accept service dependency which can further be called and determine if we want to load the Route or not.

We can use the Angular CLI command
ng g g [guard name]

? Which interfaces would you like to implement?
>( ) CanActivate
( ) CanActivateChild
( ) CanLoad

Route Guards Parameters

ActivatedRouteSnapshot
This is the future route which will be activated when the guard condition is passed.
RouterStateSnapshot
This the future router state which will be activated if the guard is passed.

Route Guards

CanActivate Route Guard

Generally, in our application, we have a need where we want to check if the user is logged in or not. In that case, we can decide if we want to activate that route for that user or not.

If the **CanActivate guard** returns false then the user is not able to navigate to route further where want to navigate. Angular provides CanActivate Route for that.

CanActivate decides whether we can navigate to a route or not. It is used to redirect to login page to require authentication.

CanActivate is interface and have method **canActivate()** that is used for authentication.

CanActivate is an Angular interface. It is used to force user to login into application before navigating to the route.

Route Guards

CanActivate Route Guard

CanActivate interface has a method named as canActivate() which has following arguments.

ActivatedRouteSnapshot:
This is the future route which will be activated when the guard condition is passed.
RouterStateSnapshot:
This the future router state which will be activated if the guard is passed.

canActivate() returns boolean value, UrlTree or Observable or Promise of Boolean or UrlTree value.

Before Angular 7.1, route guards can only return a **boolean**, **Promise<boolean>** or **Observable<boolean>** (asynchronous boolean objects) to tell the router if the route can be activated or not.

But now, you can also return an **UrlTree** variable which provides the new router state (route) that should be activated.
According to the Angular docs an UrlTree is a data structure that represents a parsed URL.
Note: You can create an UrlTree by calling the parseUrl() or createUrlTree() method of the Router object.

Route Guards

CanActivateChild Route Guard

This guard checks the access on child routes.

CanActivateChild decides whether we can navigate to child routes or not. It is used to decide link access on the basis of authorization. It is possible that those links accessible to ADMIN role, will be not be allowed to USER role.

CanActivateChild is interface and have method **canActivateChild()** that is used for authorization.

CanActivateChild is an Angular interface to guard child routes. Suppose a user has been authenticated but not authorized to visit the child routes, so child routes can be guarded using CanActivateChild.

Route Guards

CanDeactivate Route Guard

Sometimes, we have a condition where we want the user to confirm if he really wants to navigate from the component. We can use **CanDeactivate** route guard for this.

This guard is used when a user makes some changes in a form and by mistake clicks on a menu to navigate to some other route. In this case, it can be implemented to prompt a user that there are some unsaved changes, do you want to navigate away from this page or not.

Angular 16 : Online Class

“Resolve” Route Guards

Presented by : Chandan Kumar

By: Sahosoft Solutions

sahosoft
Online Learning Platform

Resolve Route Guards

Angular provides Resolve interface with resolve method declaration. To create a Angular Resolve guard, we need to create a class by implementing Resolve interface.

Resolve guard is used in the scenario where before navigating to any route we want to ensure whether there is data available or not. If there is no data then it has no meaning to navigate there. It means we have to resolve data before navigating to that route. Here comes the role of Angular Resolve guard.

To use Resolve guard we need to create a class by implementing Resolve interface and define resolve method. The resolve method can return Observable or Promise or a synchronous value.

Before Angular 6 version:

After creating resolver class we need to configure resolver class in providers metadata of @NgModule decorator in application module and then we need to configure our resolver class in route configuration using resolve property of Angular Route interface.

Resolve Route Guards

Resolve Interface

To create a Resolve route guard, we need to create a class implementing Angular Resolve interface.

Find the Resolve interface structure from Angular doc.

```
interface Resolve<T> {
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<T> | Promise<T> | T
}
```

It has a method named as resolve with arguments ActivatedRouteSnapshot and RouterStateSnapshot.
The method resolve can return Observable or Promise or a synchronous value.

Resolve interface is imported from @angular/router API.

Angular 16 : Online Class

Named and Multiple Router-Outlets (Auxiliary Routes) (secondary routes)

By: Sahosoft Solutions

Presented by : Chandan Kumar

Named and Multiple Router-Outlets(Auxiliary Routes)

Router Outlet

The Router outlet is a placeholder that gets filled dynamically by Angular, depending on the current router.

1. we'll see advanced uses of the <router-outlet> component such as named, multiple outlets and auxiliary routing.
2. How to Create a Named Router Outlet?
3. We can create a named Router outlet using the name property of the <router-outlet> component:

```
<router-outlet></router-outlet>
<router-outlet name="outlet1"></router-outlet>
<router-outlet name="outlet2"></router-outlet>
```

Named and Multiple Router-Outlets(Auxiliary Routes)

Create Multiple Router Outlets

You can have multiple outlets in the same template:

```
<router-outlet></router-outlet>
<router-outlet name="sidebar"></router-outlet>
```

The unnamed outlet is the primary outlet.

Except for the primary outlet, all other outlets must have a name.

Named and Multiple Router-Outlets(Auxiliary Routes)

Auxiliary Route/Secondary routes

A component has one primary route and zero or more auxiliary routes.Auxiliary routes allow you to use and navigate multiple routes. To define an auxiliary route you need a named router outlet where the component of the auxiliary route will be rendered.

Named outlet will open as secondary route within a (). Suppose we have opened a route with unnamed outlet and then we visit a route that will open in named outlet, we will observe the path on browser address bar that both path exists there. The named outlet path is called secondary routes and will open in () . Secondary routes will be appended with unnamed router outlet path.

We will see below types of named outlet:

1. Single named outlet
2. Two different named outlet
3. Contains parameter in named outlet

Angular 16 : Online Class

Various ways of passing data to route

By: Sahosoft Solutions

Presented by : Chandan Kumar

sahosoft
Online Learning Platform

Various ways of passing data to route

The Angular can pass data to Route in several ways.

- Using Route Parameter
- The Query Param(Query String)
- Static data using the data property
- Dynamic data using state object

Passing static data to a route

The static data is configured at the time of configuring the route. This is done by using the **Angular route data property** of the route. The route data property can contain an array of string **key-value** pairs. The static data can be used to store items such as **page titles**, **breadcrumb text**, and other read-only static data.

Passing Dynamic data to a Route

The option to pass the dynamic data or a user-defined object was added in the **Angular Version 7.2** using the state object.

Providing the State value

The state can be provided in two ways

Using routerLink

Using navigateByUrl

RouterLinkActive

The **RouterLinkActive** is a directive for **adding** or **removing** classes from an HTML element that is bound to a **RouterLink**. Using this directive, we can toggle CSS classes for active Router Links based on the current **RouterState**. The main use case of this directive is to **highlight** which route is currently active. You can either make the font bold or apply some background color.

The **RouterLinkActive** Directive is applied along with the **RouterLink** directive. The right-hand side of **RouterLinkActive** contains a **Template expression**. The template expression must contain a **space-delimited** string of **CSS classes**, which will be applied to the element when the route is active.