CSM
breaking thought barriers

Angular Training

Session -25

Outlines

Angular Router Guard

Angular Route Guards

We use the Angular Guards to control, whether the user can navigate to or away from the current route.

Guards help us to secure the route or to perform some actions before navigating into a route or leaving the route.

Why Guards

Allowing the user to navigate all parts of the application is not a good idea. We need to restrict the user until the user performs specific actions like login. Angular provides the **Route Guards** for this purpose.

One of the common scenario, where we use Route guards is authentication. We want our App to stop the unauthorized user from accessing the protected route.

Uses of Angular Route Guards

- To Confirm the navigational operation
- Asking whether to save before moving away from a view
- Allow access to certain parts of the application to specific users
- Validating the route parameters before navigating to the route
- Fetching some data before you display the component.

Types of Route Guards

The Angular Router supports Five different guards, which you can use to protect the route

1.CanActivate

2.CanDeactivate

3.Resolve

4.CanLoad

5.CanActivateChild

1 . CanActivate

- This guard decides if a route can be activated (or component gets used).
- This guard is useful in the circumstance where the user is not authorized to navigate to the target component. Or the user might not be logged into the system.
- Use this guard, when we want to check on some condition, before activating the component or showing it to the user.
- This allows us to cancel the navigation.

Use cases for the CanActivate Guard

- Checking if a user has logged in
- Checking if a user has permission

A route can have more than one canActivate guard.

- If all guards returns true, navigation to the route will continue.
- If any one of the guard returns false, navigation will be cancelled.
- If any one of the guard returns a UrlTree, current navigation will be cancelled and a new navigation will be kicked off to the UrlTree returned from the guard.

How to use CanActivate Guard

1. Create a Angular Service.
   - ❑ The service must import & implement the **CanActivate** Interface.
   - ❑ This Interface is defined in the **@angular/router** module.
   - ❑ The Interface has one method i.e. **canActivate**. We need to implement it in our Service.

```
interface CanActivate {
  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable< boolean | UrlTree > | Promise<boolean | UrlTree> | boolean | UrlTree
}
```

- The method gets the instance of the ActivatedRouteSnapshot & RouterStateSnapshot. We can use this to get access to the route parameter, query parameter etc.
- The guard must return true/false or a UrlTree . The return value can be in the form of observable or a promise or a simple boolean value.

```
import { Injectable } from '@angular/core';
import { Router, CanActivate, ActivatedRouteSnapshot,RouterStateSnapshot } from '@angular/router';
@Injectable()
export class AuthGuardService implements CanActivate {
   constructor(private _router:Router ) {
   }
   canActivate(route: ActivatedRouteSnapshot,
            state: RouterStateSnapshot): boolean {
      //check some condition
      if (someCondition)  {
         alert('You are not allowed to view this page');
         //redirect to login/home page etc
         //return false to cancel the navigation
         return false;
      }
      return true;
   }
}
```

2. Update the route definition with the canActivate guard as shown below. You can apply more than one guard to a route and a route can have more than one guard

```
{ path: 'product', component: ProductComponent, canActivate : [AuthGuardService] },
```

CanActivate guard Example

HomeComponent

ContactComponent

ProductComponent

LoginComponent

AuthService

AuthGuardService

Angular CanActivateChild

- The CanActivatechild guard is very similar to CanActivateGuard.
- We apply this guard to the parent route.
- The Angular invokes this guard whenever the user try to navigate to any of its child route.
- This allows us check some condition and decide whether to proceed with the navigation or cancel it.

```
{ path: 'product', component: ProductComponent, canActivate : [AuthGuardService],
    children: [
    {  path: 'view/:id', component: ProductViewComponent  },
    {  path: 'edit/:id', component: ProductEditComponent  },
    {  path: 'add', component: ProductAddComponent }
    ]
 },
```

The ProductComponent displays the list of product. We have attached the canActivate guard to the product route. The canActivate guard blocks the access to the route, if the user is not logged in. This guard protect both the product route and all its children.

- Now, consider the case where we want all users to view the ProductComponent, but only the Admin user can view any of its child routes

- We can create another guard ProductGuardService which implements the canActivate guard and attach it to each of those child routes.

```
{ path: 'product', component: ProductComponent, canActivate : [AuthGuardService],
    children: [
    {  path: 'view/:id', component: ProductViewComponent, canActivate : [ProductGuardService]  },
    {  path: 'edit/:id', component: ProductEditComponent, canActivate : [ProductGuardService]  },
    {  path: 'add', component: ProductAddComponent, canActivate : [ProductGuardService] }
    ]
 },
```

- Another way is to use the CanActivateChild guard and attach it to the product route
- When Angular sees a canActivateChild guard attached to the parent route, it invokes it every time the user tries to navigates to the child route.
- Hence instead of attaching Guard service every child, you can attach it to the parent route.

```
{ path: 'product', component: ProductComponent, canActivate : [AuthGuardService],
canActivateChild : [AdminGuardService],
    children: [
    {  path: 'view/:id', component: ProductViewComponent  },
    {  path: 'edit/:id', component: ProductEditComponent  },
    {  path: 'add', component: ProductAddComponent }
    ]
 },
```

How to Create CanActivateChild Guard

Example :

1. LoginComponent
2. AuthService
3. ProductComponent
4. ProductService
5. Product
6. ProductAddComponent
7. ProductEditComponent
8. ProductViewComponent
9. HomeComponent
10. ContactComponent
11. AuthGuardService

Angular CanDeactivate Guard

- The Angular CanDeactivate guard is called, whenever we navigate away from the route before the current component gets deactivated.

- The best use case for CanDectivate guard is the data entry component. The user may have filled the data entry and tries to leave that component without saving his work. The CanDeactivate guard gives us a chance to warn the user that he has not saved his work and give him a chance to cancel the navigation.

How to use CanDeactivate Guard

1. **Create a CanDeactivate Guard**
2. **Implement CanComponentDeactivate in Your Component**
3. **Configure the Route with the CanDeactivate Guard**

CanDeactivate Example

- First create a component, with a method canExit, which returns a boolean. In the canExit method, you can check if there are any unsaved data, etc. If Yes, then ask for confirmation whether the user wants to leave or not.

- Return true to exit the component else return false to stay in the same component.

Angular Resolve Guard

- The Resolve guard is used to pre-fetch data before the route is activated.
- It helps in resolving data dependencies for a component before the component is displayed.
- This is particularly useful when you need to ensure that certain data is available before rendering a component.

Angular **CanLoad** Guard

- The CanLoad guard is used to prevent the asynchronous loading of feature modules, allowing you to conditionally load modules based on certain criteria.

- This guard is typically used with the Angular Router to control the loading of feature modules before they are requested.