Routing

What is Routing? Why Do We Need Routing?

Routing means splitting the application into different areas usually based on rules that are derived from the current URL in the browser.

Defining routes in our application is useful because we can:

- separate different areas of the app
- maintain the state in the app
- protect areas of the app based on certain rules

Configure Routing in Angular

There are three main components that we use to configure routing in Angular:

- **Routes** describes the routes our application supports
- **RouterOutlet** is a "placeholder" component that shows Angular where to put the content of each route
- **RouterLink** directive is used to link to routes

Guarding your Routes

You add guards to the route configuration to handle these scenarios-

- When the user is not authorized to navigate to the target component.
- Maybe the user must login (authenticate) first.
- Maybe you should fetch some data before you display the target component.
- You might want to save pending changes before leaving a component.
- You might ask the user if it's OK to discard pending changes rather than save them.

A guard's return value controls the router's behavior:

- If it returns true, the navigation process continues.
- If it returns false, the navigation process stops and the user stays put.

The router supports multiple guard interfaces:

- **CanActivate** to mediate navigation to a route.
- **CanActivateChild** to mediate navigation to a child route.
- **CanDeactivate** to mediate navigation away from the current route.
- **Resolve** to perform route data retrieval before route activation.
- **CanLoad** to mediate navigation to a feature module loaded asynchronously.

Thank You!