Module 11

What to learn

Understanding forRoot and forChild Concept

Practice Exercise

Practice 1

Understanding forRoot and forChild in Angular Routing

Imagine you are developing a multi-module Angular application for an online bookstore. You have two main modules: **BooksModule** and **UsersModule**. Your task is to set up routing for these modules using the **forRoot** and **forChild** methods. Create a routing configuration where:

- The **AppRoutingModule** uses **forRoot** to define the main routes for the application.
- The **BooksModule** uses **forChild** to define routes specific to book-related components.
- The **UsersModule** also uses **forChild** for user-related components.

Ensure that the main application can navigate to the books and users sections seamlessly.

Practice 2

Creating a Shared Module with forRoot

In your online bookstore application, you want to create a **SharedModule** that contains common components, directives, and pipes that can be used across different modules. Your task is to implement this module using the **forRoot** method. The **SharedModule** should:

- Export a common component, such as **HeaderComponent**, which will be used in both the **BooksModule** and **UsersModule**.
- Provide a service, **AuthService**, that manages user authentication and should be a singleton across the application.

Make sure to demonstrate how to import this **SharedModule** in the main application module and in the feature modules.

Practice 3

Lazy Loading with forChild

As part of optimizing your online bookstore application, you decide to implement lazy loading for the **BooksModule**. Your task is to configure the routing such that:

- The **BooksModule** is loaded only when the user navigates to the '/books' route.
- Use the **forChild** method in the **BooksRoutingModule** to define the routes for book-related components.
- Ensure that the main application module does not import the **BooksModule** directly, but instead uses the router to load it lazily.

Provide the necessary code snippets to demonstrate this lazy loading configuration.

Practice 4

Guarding Routes with forChild

In your online bookstore application, you want to protect certain routes in the **UsersModule** to ensure that only authenticated users can access them. Your task is to implement route guards using the **forChild** method.
The **UsersRoutingModule** should:

- Define routes for user profile and order history, which should be protected by an **AuthGuard**.
- Use the **forChild** method to set up the routes, ensuring that the guard is applied correctly.

Provide the implementation details for the **AuthGuard** and how it integrates with the routing configuration.

Practice 5

Testing forRoot and forChild Implementations

After implementing the routing for your online bookstore application, you want to ensure that everything works as expected. Your task is to write unit tests for the routing configurations. Specifically, you should:

- Test that the main application routes are correctly defined using **forRoot**.
- Test that the child routes in the **BooksModule** and **UsersModule** are correctly defined using **forChild**.
- Verify that lazy loading works as intended and that guards are applied correctly.

Provide examples of the test cases you would write to validate these routing configurations.

Assignment 1

Understanding forRoot and forChild Concepts in Angular Routing

In this assignment, you will create a multi-module Angular application that demonstrates the use of the `forRoot` and `forChild` methods in routing. The application will consist of a main module and a feature module, each with its own routing configuration.

Functional Requirements

Your application should include the following:

- A main module that uses `forRoot` to define the primary routes.
- A feature module that uses `forChild` to define additional routes specific to that module.
- Navigation between the main module and the feature module through a navigation bar.

UI Control IDs

Define the following UI elements with unique IDs:

- Navigation bar: `nav-bar`
- Link to Feature Module: `link-feature-module`
- Feature Module Header: `feature-header`

Functional Flow

The user should be able to:

- Load the application and see the navigation bar.
- Click on the link to the Feature Module, which should navigate to the feature module's route.
- View the header of the Feature Module displayed on the UI.

Validation Elements

Include the following validation elements:

- Success message upon successful navigation: `success-message` with the text 'Navigation Successful!'
- Error message if navigation fails: `error-message` with the text 'Navigation Failed!'

Business Logic

Implement the following business logic:

- Ensure that the feature module can only be accessed if the user is authenticated. If not authenticated, display the error message.
- Use a service to manage the authentication state.

Initial Data

Bootstrap the application with the following initial data:

- User authentication state (true/false).

Selectors for Test Cases

Ensure that all control IDs are specified for automated test case generation:

- For navigation: `nav-bar`, `link-feature-module`
- For messages: `success-message`, `error-message`

By completing this assignment, you will gain a deeper understanding of how to implement routing in Angular applications using `forRoot` and `forChild` methods.

Online Reference

No online Reference