Advanced Lab Exercise: Product Listing Application

Objective

To build a product listing application using Angular that demonstrates a comprehensive understanding of different data binding techniques.

For a more complex Angular data binding lab exercise, we'll incorporate a small project that leverages all forms of data binding (interpolation, property binding, event binding, and two-way binding) within a single application. This exercise will guide learners through building a simple product listing application with a search feature and the ability to add new products dynamically.

Requirements

1. Angular CLI installed on your machine.
2. Basic knowledge of Angular, including components, modules, and services.
3. Basic understanding of TypeScript and HTML.

Setup

1. Create a new Angular project: **`ng new product-listing-app`**.
2. Navigate to the project directory: **`cd product-listing-app`**.
3. Serve your application to ensure it's set up correctly: **`ng serve`**.
4. Open your project in a code editor.

Exercises

Exercise 1: Setting Up the Application Structure

1. Generate two components: **`ProductsList`** and **`AddProductForm`**.
```
ng generate component ProductsList
ng generate component AddProductForm
```
2. Generate a service: **`ProductsService`**.
```
ng generate service Products
```
3. In **`ProductsService`**, define a method to get products (mock data for this exercise) and a method to add a new product.
4. In the **`AppComponent`**, use both **`ProductsList`** and **`AddProductForm`** components.

Exercise 2: Displaying Products

1. In **`ProductsService`**, create a mock products array with sample product objects (each having properties like **`id`**, **`name`**, **`price`**, and **`description`**).
2. Implement a method in **`ProductsService`** to return this products array.

3. In `ProductsList` component, use the service to fetch products and display them using *ngFor, demonstrating property binding.

Exercise 3: Adding a New Product

1. In `AddProductForm` component, create a form with fields for product name, price, and description. Use two-way binding with `ngModel` for form inputs.
2. Implement a method to submit the new product form, which calls a service method to add the new product to the products array.
3. After adding a product, use event binding to emit an event to the parent component (`AppComponent`) to refresh the product list.

Exercise 4: Implementing Search Functionality

1. Add a search input field in the `ProductsList` component.
2. Use two-way binding on the search input to bind it to a property in the component.
3. Implement filtering logic in `ProductsList` to dynamically filter the displayed products based on the search input, demonstrating the use of two-way binding and pipes for filtering.

Exercise 5: Styling

1. Apply CSS styles to your application to make it visually appealing.
2. Ensure that the product list is nicely formatted, and the form has a clear, user-friendly layout.

Conclusion

By completing this lab exercise, you will have created a functional Angular application that incorporates various data binding techniques to interact dynamically with data. This application includes a service for managing data, components for displaying and adding data, and incorporates user input to filter and add new products.