Help on Assignment - 5

Would you need to use routing to display the contact list, add a contact, and edit the contact part of the application?

The simple answer is you are welcome to use Routing as used in the "Tour of Heroes" tutorial but you can accomplish this without needing to use routing.

Follow the following example to create this application without having to use routing:
https://stackblitz.com/angular/apkgqbqxrlp

This is an example that Angular.io provides for Forms. But, it uses a simple capability that you would need in the Contact Manager application that you are building.

In this example, you could see two buttons – Name Editor and Profile Editor. As you click on each of them the relevant component is displayed. This example does not use any routing.

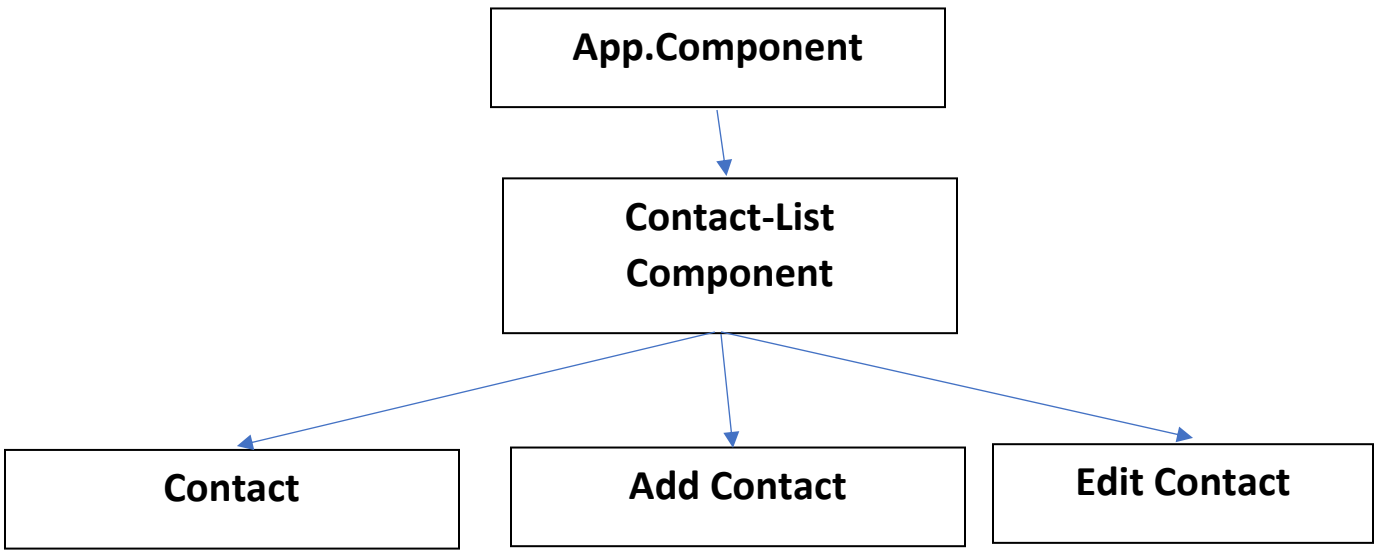
If you look at the app.component.ts you would see that both the custom tags for these components are included but with the *ngIf directive meaning display the component only when the condition is true.

Follow this example, for providing the ability to toggle between different components for the Contact Manager application that you are building.

For your Contact Manager application, you can create the following components to support the features we need in the application

- Contact-List – represents the list of contacts
- Contact – represents a contact
- AddContact – to add a new contact
- EditContact – to edit an existing contact.

The following could be the hierarchy of the components:



In here, AppComponent serves as the root component. Its child is Contact-List. And Contact-List acts as a parent for the following child components – Contact, AddContact, and EditContact.

Contact-List would keep an array of Contacts

The Html of **Contact**-List will have the custom tags for the three child tags nested inside them:

```
<div *ngIf=”..”>
<app-contact *ngFor=”…….” *ngIf=”….”></contact>
</div>
<div *ngIf=”..”>
<app-add-contact *ngIf=”…..”></app-add-contact>
</div>
<div *ngIf=”..”>
<app-edit-contact *ngIf =”…..”></app-edit-contact>
</div>
```

So, just like in the example from stackblitz, all the custom tags are embedded and which component will display is controlled by *ngIf directive whose value can be updated by the specific button that the user clicks. The reason I had to provide a div tag for each of the component selectors was that Angular does not allow two structural directives that start with * in the same tag.

The other issue is regarding how you account for the data communication between the components. There are two possible approaches:

Use Input and Output bindings to communicate data between the parent (contact list) and its child components. But if the siblings have to communicate, then it becomes tedious as you would need to set up a chain of input-output binding from a sibling to parent and then from parent to the other sibling.

1. As an example, when contact is displayed with its edit and delete buttons and the user clicks on the edit button, we want to pass data (contact information that the user wants to edit) to its sibling edit-contact component. With this approach, we would have to pass it to the parent (Contact-List) first using Output binding first, and then using input binding the edit-component will receive the account information from the parent.

As you can see it is a tedious approach, a better approach would be to use a Service.

2. In this better approach, a Service class can be created named ContactService (Please refer to the lecture on Angular Services) that will now contain the Contacts array and provide methods to addContact( ), editContact( ), and deleteContact( ). Then you could inject the service into each of the components through their constructor. This approach reduces greatly the amount of data communication you need between the components. The later part of the Angular-Services lecture demonstrates the cross-communication between sibling components.