Angular Services LAB and Homwork

All the statements in the first part of the lab will build on the lab from the previous day. So if that lab is not complete go back and ensure that lab is complete first.

Routing Services Dependency Injection Observables Http

Routing

Add a routing component using the `ng generate module` command at the command prompt.

Add a route for the new component that was created at the end of the last lab. Remove the tag for that component and put in the `<router-outlet>` tag in its place.

At first, navigate manually to the new route to verify it works.

Next, add an anchor tag (`<a>`) with the appropriate property to navigate to the new page.

Repeat the above series of steps by:

- creating a new component
- modifying the html template to display something other than the default message.
- create a route for the new component.
- add a tag/link that goes to the page.

Style the pages links, either as horizontal or vertical items so they resemble a menu.

Finally, create another component that will serve as the application default. Give it some additonal "Welcome to the site" verbiage and make it the default route in the routing system.

Services and DI

Next, create a simple Person class in the src/app folder. Some applications would also create this in a sub-folder called "models" or "domain-classes", it's entirely up to you.

Next, using the CLI and `ng generate`, generate a service class. Also, use the option to the generate command to *NOT* create a test file.

In the Service class, reference the Person class from above. Create a variable of type Person. In the constructor for the service give it some default values.

Next, create a basic method to return the person such as `getPerson()`.

Pick one of the compoents and inject the service. Create a member variable of type Person, and modify the html to use interpolation to display the properties of the person.

Use the `ngOnInit()` method to get the Person from the service.

Observables

Change the service call to return an Obervable rather than a Person object. Change the call in the component to subscribe() to the call rather than being a synchronous call.

To really see the process of Observables without a long web delay, create this function inside the service class:

```
async delay(ms: number) {
    await new Promise(resolve => setTimeout(()=>resolve(), ms));
}
```

and inside of the `getPerson()` method, call the method to delay before returning the value:

```
this.delay(4000);
// OR
this.delay(4000).then(any => {return peep});
```

Notice how the value will automatically display after the waiting period. This simulates a long-running call to get the data.

http

http will be covered as part of the lab below. In that tutorial/walkthrough you will simulate a web call.

2nd part of the lab - go to the Angular.io site and work through the Angular Tour of Heroes app. Be sure to do the CLI-based version of the tutorial.

This covers many of the same concepts as the two labs so far, but also has quite a bit of additional content such as using loops on the client side and doing extensive CSS styling. It will be very beneficial to do this tutorial. Once complete, how would you expand on it?

Go back and change your first application and service to return multiple people and display a list.