Seneca
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

BTI425

Web Programming for Apps and Services

Notes	Weekly
Resources	Graded work
Policies	Standards
Professors & addendum	Code examples

Routing in Angular, an introduction

This is a very brief and simple introduction to *routing* in an Angular app. Other notes will provide more coverage.

Recently, you learned about, and implemented routing in a React app. The topic idea is the same in an Angular app - to enable navigation among components, and/or component replacement in a viewport. The implementation is similar enough for you to learn and use within minutes.

Prerequisites, what we need

We need an app that was generated with the `--routing` option, which (among other tasks) creates a source code file `app-routing.module.ts` (and inside is a class named `AppRoutingModule`). The generation task does a few other useful things too, which we'll get to in a moment.

We also need a few other things, which should be obvious:

- multiple components (for navigation/replacement)
- a nav menu (or other workflow control UI)

"Standard" routes

When you learned and implemented React routing, you had to configure an empty route, and a "not found" route. We do that here, too.

Open `app-routing.module.ts` for editing. Locate the declaration for the `routes` constant. Add these route objects:

- The first handles an empty route
- The other handles a "not found" route

```
{ path: '', redirectTo: '/home', pathMatch: 'full' },
{ path: '**', component: NotFoundComponent }
```

Make sure that these two route objects are always at the end of the array.

What's in a *route object*?

A *route object* a (JavaScript) object that conforms to the Route interface.

While the interface documentation shows the members, we must look at the Routes type documentation to learn about the purpose and use of each member.

For beginner scenarios, the most-often used members are:

`path` - a string for the URL segment that follows the leading slash

`component` - a component type (class name)

Soon, you will learn about a couple of other members for some specialty route objects.

Implementing routing

Here's how to implement routing, on a per-component basis. *Do this for each component* that participates in routing.

Open `app-routing.module.ts` for editing. Locate the declaration for the `routes` constant.

Before the "standard" route objects, add a new route object, which looks something like this:

```
{ path: 'home', component: HomeComponent },
```

This route object assumes the following:

- We want to use “home” as the URL segment (the path)
- We have an existing “HomeComponent” class

Configuring a navigation-aware link

In the UI - in a nav menu or in some other link (or link that looks like a button) - we want to navigate to a component. Similar to React, but different. In React, we *replaced* the `a` element with a `Link` element (and its `to` attribute). In Angular, we still use the `a` element, but instead of using the `href` attribute, we use a *replacement attribute* named `routerLink`.

Prepare to edit an existing `a` element. Replace the `href` attribute:

- The new attribute is named `routerLink`
- Its value is a root-relative path (which means that it begins with `/` a slash), that matches the value of the `path` attribute in the route objects (above)

For example:

```
<a routerLink='/home'>Home</a>
```

> *Incidentally, we will cover programmatic routing / navigation in a later topic set.*

More info about routes

This document has a summary of guidance about designing suitable routes in your web API and app.

Summary, checklist

Here’s a summary of the above, which can be used as a quick-start checklist.

1. Ensure that the app was generated with the `--routing` option.

2. Create a component that will participate in routing.

3. Open `app-routing.module.ts` for editing. Add a new route object to the `routes`

constant.

4. If the component must be part of the app's nav menu, add and configure an a element.