# Angular Best Friends

## Module 6 Exercise 5 – Secondary routes

## Goals

Secondary routes are a great tool to use when you need to develop a certain application part that offers a dashboard like experience. In this exercise we’ll get familiar with how to set up secondary routes and how to activate them.

## Steps

1. Go to the “**Module6Exercise5 > initial**” folder
2. Right click it and open he folder in VS Code
3. In the terminal window run **npm install** to install the needed node modules. This step is needed only when you run the application for the first time.
4. Go to the **app.component.html** file. Notice the div with a container class. In this div we display the content of our routes. What we want to do is rewrite this div so that it will have two column. One for the main router outlet (the one that we used till now) and one for the secondary router outlet. So delete the entire div and replace it with the following:

<div class="container">

<div class="row">

<div class="col-md-10"

[@slideInAnimation]="o.isActivated ? o.activatedRoute : ''">

<router-outlet #o="outlet"></router-outlet>

</div>

<div class="col-md-2">

<router-outlet name="popup"></router-outlet>

</div>

</div>

</div>

Notice that for the secondary router outlet we have defined a name. This is how we will be able to specify in which router outlet our content needs to be placed.

1. No we need to configure the secondary route. In VS Code navigate to “**src > app > messages**” folder and open the **messages.module.ts** file.
2. Add an import to the RouterModule at the top of the file:

import { RouterModule } from '@angular/router';

1. Add the RouterModule to the list of imports. The final list should look like this:

imports: [

SharedModule,

RouterModule.forChild([

{

path: 'messages',

component: MessageComponent,

outlet: 'popup'

}

])

]

Notice that in the route definition we added something new. And that is the “outlet” property. For this property we have specified the name. Note that the name should be an exact match to the name that we have defined in step 4!

1. Now we need to activate the secondary route. To do this let’s go back to the **app.component.html** file. We want to activate the secondary route so that when we click on “Show messages” it should display some messages in the secondary router outlet. (Don’t bother about how messages are generated. In the application there is a service for it. Remember what we studied yesterday!)
2. On line 29 we have a list item that contains the “Show messages” link. Delete that entire <li> tag and replace it with the following:

<li class="nav-item"

\*ngIf="!isMessageDisplayed">

<a class="nav-link"

(click)="displayMessages()">Show Messages</a>

</li>

1. We have defined here one function that we should call when clicking the “Show messages” link. Two lines we above we have an ngIf directive that will cause the li to show in this form only if messages are not displayed yet. So the link was not clicked yet. This means that we need to activate everything in code. To do this, go to the messages folder and open the **messages.service.ts** file.
2. Delete all the code and replace it with the following snippet:

import { Injectable } from '@angular/core';

@Injectable({

providedIn: 'root'

})

export class MessageService {

private \_messages: string[] = [];

isDisplayed = false;

get messages(): string[] {

return this.\_messages;

}

addMessage(message: string): void {

const currentDate = new Date();

this.messages.unshift(message + ' at ' + currentDate.toLocaleString());

}

}

If you make a comparison to what we had previously, we have basically added a private array to hold message strings and a Boolean property that should indicate if the messages should be displayed or not.

1. In the **app.component.ts** file we need to add the method that handles the logic when the “Show messages” link is clicked. Go to line 45, put the cursor after the closing curly bracket and press Enter two times. Then paste the following snippet:

displayMessages(): void {

this.router.navigate([{ outlets: { popup: ['messages'] } }]);

this.messageService.isDisplayed = true;

}

1. Import the message service on the top of the file:

import { MessageService } from './messages/message.service';

1. Add it to the constructor. Your constructor should look like this afterwards:

constructor(private authService: AuthService,

private router: Router,

private messageService: MessageService) {

router.events.subscribe((routerEvent: Event) => {

this.checkRouterEvent(routerEvent);

});

}

1. Under the displayMessages function that we created earlier create a new function to hide messages. Use the following code snippet

hideMessages(): void {

this.router.navigate([{ outlets: { popup: null } }]);

this.messageService.isDisplayed = false;

}

1. We need to finally add a public property for the message displaying options. To do that, go on line and put the mouse cursor after the closing curly brace. Then press two times Enter. And then add the following code snippet:

get isMessageDisplayed(): boolean {

return this.messageService.isDisplayed;

}

1. In **app.component.html**, just below the <li> tag that holds the Show messages link add another li by pasting the following code:

<li class="nav-item"

\*ngIf="isMessageDisplayed">

<a class="nav-link"

(click)="hideMessages()">Hide Messages</a>

</li>

1. Now everything should be fine. Run the application and play around. To add a message to the message logs just click the login link and login with a fake username and password.