# Angular Best Friends

## Module 6 Exercise 4 – Child routes

## Goal

The goal of this exercise is to better understand child routes. So we’ll play around with this type of routes

## Steps

1. Go to the “**Module6Exercise4 > 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.

### Child routes

1. We want to add a tabbed layout for the Edit product component so that we can navigate between editing product details and editing tags. So let’s go to the “**src > app > products”** folder and open the **product.module.ts** file.
2. Notice that we have here the previously configured routes. This is also the place where we need to add the child routes as an array in the parent route definition. The easiest way to achieve this is by deleting all the code in the file and replace it with the following:

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

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

import { ProductListComponent } from './product-list.component';

import { ProductDetailComponent } from './product-detail.component';

import { ProductEditComponent } from './product-edit/product-edit.component';

import { ProductEditInfoComponent } from './product-edit/product-edit-info.component';

import { ProductEditTagsComponent } from './product-edit/product-edit-tags.component';

import { ProductResolver } from './product-resolver.service';

import { SharedModule } from '../shared/shared.module';

@NgModule({

imports: [

SharedModule,

RouterModule.forChild([

{ path: 'products', component: ProductListComponent },

{

path: 'products/:id',

component: ProductDetailComponent,

resolve: { resolvedData: ProductResolver }

},

{

path: 'products/:id/edit',

component: ProductEditComponent,

resolve: { resolvedData: ProductResolver },

children: [

{

path: '',

redirectTo: 'info',

pathMatch: 'full'

},

{

path: 'info',

component: ProductEditInfoComponent

},

{

path: 'tags',

component: ProductEditTagsComponent

}

]

}

])

],

declarations: [

ProductListComponent,

ProductDetailComponent,

ProductEditComponent,

ProductEditInfoComponent,

ProductEditTagsComponent

]

})

export class ProductModule { }

1. Notice that to the “edit” route definition we have added a “children” property which is of type array. In this array we have added the desired child routes.
2. Now we have to place everything in the edit component. So go to the **product-edit.component.html** template. Delete the entire html markup and paste the following instead:

<div class="card">

<div class="card-header">

{{pageTitle}}

</div>

<div class="card-body"

\*ngIf="product">

<div class="wizard">

<a [routerLink]="['info']">

Basic Information

</a>

<a [routerLink]="['tags']">

Search Tags

</a>

</div>

<router-outlet></router-outlet>

<div class="row mb-2">

<div class="col-md-4">

<button class="btn btn-primary mr-3"

style="width:80px"

type="button"

title="Save your entered data"

[disabled]="!isValid()"

(click)="saveProduct()">

Save

</button>

<button class="btn btn-outline-secondary mr-3"

style="width:80px"

type="button"

title="Cancel your edits"

[routerLink]="['/products']">

Cancel

</button>

<button class="btn btn-outline-warning"

style="width:80px"

type="button"

title="Delete this product"

(click)="deleteProduct()">

Delete

</button>

</div>

</div>

</div>

</div>

<div class="alert alert-danger"

\*ngIf="errorMessage">{{errorMessage}}

</div>

1. A few words on what we have done. On the page we had a form. What we wanted to do is to delete the form and replace it with two different tabs. One for editing the product details and one for editing the tags. And we chose to place here a router outlet tag. Now child routes will display their correspondent components here.
2. Now we have to find away to pass data between parent route and child route. To do this we can simply subscribe to the router data coming from the parent route. To do this go to the **product-edit-info.component.ts** file. Delete all the code and replace it with the following snippet:

import { Component, OnInit, ViewChild } from '@angular/core';

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

import { NgForm } from '@angular/forms';

import { Product } from '../product';

@Component({

templateUrl: './product-edit-info.component.html'

})

export class ProductEditInfoComponent implements OnInit {

@ViewChild(NgForm) productForm: NgForm;

errorMessage: string;

product: Product;

constructor(private route: ActivatedRoute) { }

ngOnInit(): void {

this.route.parent.data.subscribe(data => {

if (this.productForm) {

this.productForm.reset();

}

this.product = data['resolvedData'].product;

});

}

}

Take a look at it. In the NgOnInit function we have subscribed to the data coming from the parent route.

1. We should do the same things for the component that handles tags. So go to the product-edit-tags.component.ts file. Delete all the code and replace it with the following snippet:

import { Component, OnInit } from '@angular/core';

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

import { Product } from '../product';

@Component({

templateUrl: './product-edit-tags.component.html'

})

export class ProductEditTagsComponent implements OnInit {

errorMessage: string;

newTags = '';

product: Product;

constructor(private route: ActivatedRoute) { }

ngOnInit(): void {

this.route.parent.data.subscribe(data => {

this.product = data['resolvedData'].product;

});

}

// Add the defined tags

addTags(): void {

if (!this.newTags) {

this.errorMessage = 'Enter the search keywords separated by commas and then press Add';

} else {

const tagArray = this.newTags.split(',');

this.product.tags = this.product.tags ? this.product.tags.concat(tagArray) : tagArray;

this.newTags = '';

this.errorMessage = '';

}

}

// Remove the tag from the array of tags.

removeTag(idx: number): void {

this.product.tags.splice(idx, 1);

}

}

Most of the things remained the same. We just subscribed to the parent route data in the ngOnInit function.

1. Now run the application and play around. When you click on the Edit button you should be forwarded to the component where the product details will be displayed in a form as part of a child route / component. There should be another tab for tags. When you click save, all the data will be saved.