Activating Routes with RouterLink in Angular

Introduction

Routing in Angular helps navigate from one view to another as users perform tasks in web apps. In this guide you will learn about Angular router's primary feature RouterLink and how to use routerLink in your Angular apps.

Angular Routing

You can configure an app's routes as an array of objects and pass them to Angular's RouterModule.forRoot.

File name: app.routes.ts

1

2

3

4

5

6

import { RouteModule, Routes } from '@angular/router';

const routes: Routes = [

{ path: 'component-one', cmponent: ComponentOne }, { path: 'component-two', cmponent: ComponentTwo }

];

export const routing = RouterModule.forRoot(routes);

typescript

Then, import your routing configuration in the root of your app.

1

2

3

4

5

6

7

8

9

10

11

import { routing } from './app.routes';

@NgModule({

imports: [ BrowserModule, routing],

declarations: [

AppComponent,

ComponentOne,

ComponentTwo

],

bootstrap: [ AppComponent ]

})

export class AppModule {}

typescript

By default, the app navigates to the empty route. You can redirect it to one of the named routes.

File name: app.routes.ts

1

{path: '', redirectTo: 'component-one', pathMatch: 'full'}

typescript

The pathMatch property, which is required for redirects, tells the route how it should match the URL provided in order to redirect to the specified route. Since pathMatch: full is provided, the route will redirect to component-one if the entire URL matches the empty path('').

Linking Routes in HTML

To add links to one of the routes, use the routerLink directive in HTML. This directive accepts an array. The first parameter is the name of the route, and the second parameter is the parameters that you want to pass with the route.

1

<a [routerLink]="['component-one', routeParams]"></a>

html

If you use the routerLink directive without the brackets, you'll need to pass the route as a string.

1

<a routerLink="/component-one"></a>

html

You can also change the route programmatically in Angular.

1

this.router.naigate(['/component-one']);

typescript

The <router-outlet> </router-outlet> acts as a placeholder for components. Angular dynamically adds the component for the route to be activated into this.

Accessing Route Parameters in Angular

The router provides two different methods to grab the route parameters. They are:

Snapshot

The ActivatedRoute service provides a snapshot of the current route.

1

2

3

4

5

6

7

8

9

10

11

12

13

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

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

@Component({

...

})

export class SampleComponent implements OnInit {

constructor(private route : ActivatedRoute) {}

ngOnInit() {

this.myParam = this.route.snapshot.params.myParam;

}

}

typescript

This is simple and enough. But since Angular reuses the components for performance, this method won't work for some cases, for example, if you navigating from /product/1 or /product/2. In that case you need to use the Observable/Stream method.

Observable/Stream

This method uses observables, and when you navigate from one route to another route the observable will pass along the new data. Angular v4+ routers provide ParamMap, which is an observable you can use.

1

2

3

4

5

6

7

ngOnInit() {

// subscribe to the parameters observable

this.route.paramMap.subscribe(params => {

console.log(param.get('myParam'));

this.myParam = params.get('myParam');

})

}

typescript

If you are working with Angular versions prior to v4, you can grab route parameters with params instead of paramMap.

1

2

3

4

5

6

7

ngOnInit() {

// subscribe to the parameters observable

this.route.params.subscribe(params => {

console.log(params.get('myParam'));

this.myParam = params.get('myParam');

});

}

typescript

Using a Child Route in Angular

While some routes may only be accessible and viewed within other routes, it may be appropriate to create them as child routes, for example, if you have a product route with an individual ID and want to navigate to other routes to see its overview or specification.

The children property in the routes accepts an array of child routes. You can use the same directive to hold the child component.

File name: app.routes.ts

1

2

3

4

5

6

7

8

9

export const routes: Routes = [

{ path: 'product-details/:id', component: ProductDetails,

children: [

{ path: '', redirectTo: 'overview', pathMatch: 'full',

{ path: 'overview', component: Overview },

{ path: 'specification', component: Specification }

]

}

}

typescript

Query and Route Parameters

The key difference between query parameters and route parameters is that route parameters are essential to determining a route, whereas query parameters are optional. The optional parameters are equal to /product-list?page=2 where as route parameters are /product-list/2.

1

2

3

4

5

// Pass query params in router link

<a [routerLink]="['product-list']" [queryParams]="{ page: 2 }"> Go to Page 2</a>

// query parameters programmatically

this.router.navigate(['/product-list'], { queryParams: { page: pageNum } });

typescript

To read the query parameters, subscribe to queryParams in the ActivatedRoute.

1

2

3

4

5

ngOnInit() {

this.sub = this.route.queryParams.subscribe(params => {

this.page = +params['page'] || 0;

});

}

typescript

Conclusion

In this guide, we have explored how to activate routes in Angular with routerLink. We have also seen how to link routes in HTML, access parameters using child parameters, and the difference between query parameters and route parameters.

You can learn more about Angular in my guide [Understanding the Purpose of Routing in Angular](https://app.pluralsight.com/guides/understanding-purpose-routing-angular).