Contents(Record)

* Routing Configuration



**Routing Configuration**

* **Defining a basic route**
  1. Import **RouterModule** and **Routes** into your routing module.

**import { BrowserModule } from '@angular/platform-browser';**

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

**import { AppRoutingModule } from './app-routing.module'; // CLI imports AppRoutingModule**

**import { AppComponent } from './app.component';**

**@NgModule({**

**declarations: [**

**AppComponent**

**],**

**imports: [**

**BrowserModule,**

**AppRoutingModule // CLI adds AppRoutingModule to the AppModule's imports array**

**],**

**providers: [],**

**bootstrap: [AppComponent]**

**})**

1. Define your routes in your **Routes** array.

**const routes:** [**Routes**](https://angular.io/api/router/Routes) **= [ { path: 'first-component', component: FirstComponent }, { path: 'second-component', component: SecondComponent }, ];**

1. Add your routes to your application.

**export class AppModule { }**

**<h1>Angular Router App</h1>**

**<!-- This nav gives you links to click, which tells the router which route to use (defined in the routes constant in AppRoutingModule) -->**

**<nav>**

**<ul>**

**<li><a routerLink="/first-component" routerLinkActive="active" ariaCurrentWhenActive="page">First Component</a></li>**

**<li><a routerLink="/second-component" routerLinkActive="active" ariaCurrentWhenActive="page">Second Component</a></li>**

**</ul>**

**</nav>**

**<!-- The routed views render in the <router-outlet>-->**

**<router-outlet></router-outlet>**

* **Getting route information**
  1. Import **ActivatedRoute** and **ParamMap** to your component.

**import { Router, ActivatedRoute, ParamMap } from '@angular/router';**

* 1. Inject an instance of **ActivatedRoute** by adding it to your component’s constructor.

**constructor(**

**private route: ActivatedRoute,**

**) {}**

* 1. Update the **ngOnInit()** method to access the ActivatedRoute and track the parameters and activate route information.

**import {Component} from '@angular/core';**

**/\* . . . \*/**

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

**import {Observable} from 'rxjs';**

**import {map} from 'rxjs/operators';**

**/\* . . . \*/**

**@Component({**

**/\* . . . \*/**

**})**

**export class ActivatedRouteComponent {**

**constructor(route: ActivatedRoute) {**

**const id: Observable<string> = route.params.pipe(map(p => p.id));**

**const url: Observable<string> = route.url.pipe(map(segments => segments.join('')));**

**// route.data includes both `data` and `resolve`**

**const user = route.data.pipe(map(d => d.user));**

**}**

**}**

* **Setting up wildcard routes**
  1. To set up a wildcard route, add the following code to your **routes** definition.

**{ path: '\*\*', component: <component-name> }**

* **Displaying a 404 page**
  1. To display a 404 page.

**const routes: Routes = [**

**{ path: 'first-component', component: FirstComponent },**

**{ path: 'second-component', component: SecondComponent },**

**{ path: '\*\*', component: PageNotFoundComponent }, // Wildcard route for a 404 page**

**];**

* **Setting up redirects**
  1. To set up a redirect, configure a route with the **path** you want to redirect from, the **component** you want to redirect to, and a pathMatch value that tells the router how to match the URL.

**const routes: Routes = [**

**{ path: 'first-component', component: FirstComponent },**

**{ path: 'second-component', component: SecondComponent },**

**{ path: '', redirectTo: '/first-component', pathMatch: 'full' }, // redirect to `first-component`**

**{ path: '\*\*', component: PageNotFoundComponent }, // Wildcard route for a 404 page**

**];**

* **Nesting routes**
  1. When we want to relative to a component other than you root component. So, these types of **nested** routes are called **child** routes.

**<h2>First Component</h2>**

**<nav>**

**<ul>**

**<li><a routerLink="child-a">Child A</a></li>**

**<li><a routerLink="child-b">Child B</a></li>**

**</ul>**

**</nav>**

**<router-outlet></router-outlet>**

* 1. A child route is like any other route, in that it needs both a **path** and a **component**.

**const routes: Routes = [**

**{**

**path: 'first-component',**

**component: FirstComponent, // this is the component with the <router-outlet> in the template**

**children: [**

**{**

**path: 'child-a', // child route path**

**component: ChildAComponent, // child route component that the router renders**

**},**

**{**

**path: 'child-b',**

**component: ChildBComponent, // another child route component that the router renders**

**},**

**],**

**},**

**];**

* **Setting the page title**
  1. Each page in your application should have a unique title so that it can be identified in the browser history.

**const routes: Routes = [**

**{**

**path: 'first-component',**

**title: 'First component',**

**component: FirstComponent, // this is the component with the <router-outlet> in the template**

**children: [**

**{**

**path: 'child-a', // child route path**

**title: resolvedChildATitle,**

**component: ChildAComponent, // child route component that the router renders**

**},**

**{**

**path: 'child-b',**

**title: 'child b',**

**component: ChildBComponent, // another child route component that the router renders**

**},**

**],**

**},**

**];**

**const resolvedChildATitle: ResolveFn<string> = () => Promise.resolve('child a');**

* 1. You can also provide a custom title strategy by extending the **TitleStrategy**.

**constructor(private readonly title:** [**Title**](https://angular.io/api/platform-browser/Title)**) {**

* **Accessing query parameters and fragments**
  1. First, import the following members in the component you want to navigate form.

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

**import { Observable } from 'rxjs';**

**import { switchMap } from 'rxjs/operators';**

* 1. Next inject the activated route service.

**constructor(private route: ActivatedRoute) {}**

* 1. In component one.

**heroes$: Observable<Hero[]>;**

**selectedId: number;**

**heroes = HEROES;**

**ngOnInit() {**

**this.heroes$ = this.route.paramMap.pipe(**

**switchMap(params => {**

**this.selectedId = Number(params.get('id'));**

**return this.service.getHeroes();**

**})**

**);**

**}**

* 1. In component two.

**hero$: Observable<Hero>;**

**constructor(**

**private route: ActivatedRoute,**

**private router: Router ) {}**

**ngOnInit() {**

**const heroId = this.route.snapshot.paramMap.get('id');**

**this.hero$ = this.service.getHero(heroId);**

**}**

**gotoItems(hero: Hero) {**

**const heroId = hero ? hero.id : null;**

**// Pass along the hero id if available**

**// so that the HeroList component can select that item.**

**this.router.navigate(['/heroes', { id: heroId }]);**

**}**

* 1. You can also pass the **routerLink** on params.