Yallaling Goudar

Understanding the Purpose of Routing in Angular

Yallaling Goudar

Feb 4, 2020 • 7 Min read • 12,932 Views

Feb 4, 2020 • 7 Min read • 12,932 Views

Languages Framework... Angular

Introduction

60

- Introduction
- Overview of Angular Routing
- Router Concepts
- Configuring the Router
- Router Links
- Route Parameters
- Accessing Route Parameters
- Conclusion

Introduction

Routing in Angular helps us navigate from one view to another as users perform tasks in web apps. In this guide you will learn about Angular router's primary features.

Overview of Angular Routing

Some examples of actions users take while navigating in an app include:

- Entering a URL in the address bar and the browser navigates to a corresponding page
- Clicking links on the page and the browser navigates to a new page
- Clicking the browser's back and forward buttons and the browser navigates backward and forward through the history of pages

Angular router can interpret a browser URL as an instruction to navigate to a client-generated view. It can pass optional parameters along to the supporting view component that help it decide what specific content to present. You can bind the router to links on a page, and it will navigate to the appropriate application view when the user clicks a link.

Router Concepts

The Angular router is in its own library package, @angular/router. Import what you need from it as you would from any other Angular package. It is an optional service that presents a particular component view for a given URL.

File name: **app.module.ts**

typescript

```
1  import { RouterModule, Routes } from '@angular/
```

Configuring the Router

Let's look at how to configure an Angular router. The following example creates five route definitions, configures the router via the `RouterModule.forRoot() method, and adds the result to the` AppModule`'s imports array.

File name: app.module.ts

typescript

```typescript
const appRoutes: Routes = [
  { path: 'company', component: CompanyCompo
  { path: 'employee/:id',      component: Em
  {
    path: 'employees',
    component: EmployeeListComponent,
    data: { title: 'Employees List' }
  },
  { path: '',
    redirectTo: '/employees',
    pathMatch: 'full'
  },
  { path: '**', component: PageNotFoundCompo
];

@NgModule({
  imports: [
    RouterModule.forRoot(
      appRoutes,
      { enableTracing: true }
    )
  ],
  ...
})
export class AppModule { }
```

Router Links

After configuring the routes, the next step is to decide how to navigate. Navigation will happen based on user actions such as clicking a hyperlink, etc.

Use the `RouterLink` directive to the anchor tag for navigation, as shown below.

File name: app.module.ts

typescript

```typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  styleUrls: ['./app.component.css'],
  templateUrl: './app.component.html'
})
export class AppComponent {
  title ='Employee application' ;
}
```

Filename: app.component.html

typescript

```typescript
<h1>{{title}}</h1>
<nav>
    <a [routerLink]='["/employeedashboard"]' r
    <a [routerLink]='["/employees"]' routerLir
</nav>
<router-outlet></router-outlet>
```

In the above example, we created hyperlinks and a `routerLink` directive and specified the paths to navigate. If a user clicks on the employee dashboard, it will navigate to

class to the link when it is clicked to make it look like an active link (`active` is a CSS class defined in `app.component.css` that changes the link color to blue in this case).

`router-outlet` is the place where the output of the component associated with the given path will be displayed. For example, if a user clicks on **Employees**, it will navigate to `/employees`, which will execute the `EmployeesComponent` class as mentioned in the configuration details, and the output will be displayed in the `router-outlet` class. To navigate programmatically, we can use the `navigate()` method of the `router` class. Inject the `router` class into the component and invoke the navigate method as shown below.

typescript

```
this.router.navigate([url, parameters])
```

Route Parameters

Parameters passed along with a URL are called *route parameters*. Generate an employee dashboard component using the following CLI command.

typescript

```
import { Component, OnInit } from '@angular/
```

```
@Component({
  selector: 'app-employeedashboard',
  templateUrl: './employeedashboard.componen
  styleUrls: ['./employeedashboard.component
})
export class EmployeeDashboardComponent impl
  employees: Employee[] = [];
  constructor(
    private router: Router,
    private employeeService: EmployeeService
  ngOnInit() {
    this.employeeService.getEmployees()
      .subscribe(employees => this.employees
  }
  gotoDetail(employee: Employee) {
    this.router.navigate(['/employeedetail',
  }
}
```

Import the `router` class from `@angular/router` module . Inject it into the component class through a constructor . The `this.router.navigate()` method is used to navigate to a specific URL programmatically. The `Navigate()` method takes two arguments: the path to navigate and tthe route parameter value to pass. Here the path will be `employeedetail/<employee_id>`.

Accessing Route Parameters

To access route parameters, use the `ActivatedRoute` class. In the example

`EmployeeDetailComponen`t. When the **employees** link is clicked, it navigates to `EmployeesComponent`.

Filename: employee-detail.component.ts

typescript

```typescript
import { Component, OnInit, OnDestroy } from
import { ActivatedRoute, ParamMap } from '@a
import { switchMap } from 'rxjs/operators';
import { Observable } from 'rxjs';
import { Employee } from '../employee/employ
import { EmployeeService } from '../employee
@Component({
  selector: 'app-employee-detail',
  templateUrl: './employee-detail.component.
  styleUrls: ['./employee-detail.component.c
})
export class EmployeeDetailComponent impleme
  employee: Employee;
  error: any;
  sub: any;
  constructor(private employeeService: Emplc
  ngOnInit() {

    this.sub = this.route.paramMap.pipe(swit
      this.employeeService.getEmployee(+para
  }
  ngOnDestroy() {
    this.sub.unsubscribe();
  }
  goBack() {
    window.history.back();
  }
}
```

Import the `ActivatedRoute` class to access route parameters . Inject the `ActivatedRoute` class into the component class through a constructor . The

parameters. It has the `switchMap()` method to process the route parameters. `ParamMap` has a `get()` method to fetch a specific parameter value .

Conclusion

In this guide, we have explored how to use routing in Angular. We have also seen how to use routing in an app using `routerlink` and routing parameters.

You can learn more about Angular in my guide Using the Async Pipe in Angular.

Government (/industries/government)

Gift of Pluralsight (/gift-of-pluralsight)

View Pricing (/pricing)

Contact Sales (/product/contact-sales)

Skill up for free (/product/skills/free)

Skill IQ (/product/skill-iq)

Iris (/product/iris)

Authors (/authors)

Professional Services (/product/professional-services)

Technology Index (/tech-index)