Angular
(Part – 4)

PROF. P. M. JADAV
ASSOCIATE PROFESSOR
COMPUTER ENGINEERING DEPARTMENT
FACULTY OF TECHNOLOGY
DHARMSINH DESAI UNIVERSITY, NADIAD

Content

- Routing and Navigation
- Default Route and Page Redirection

Routing and Navigation

- The Angular Router enables navigation from one view to the next as users perform application tasks

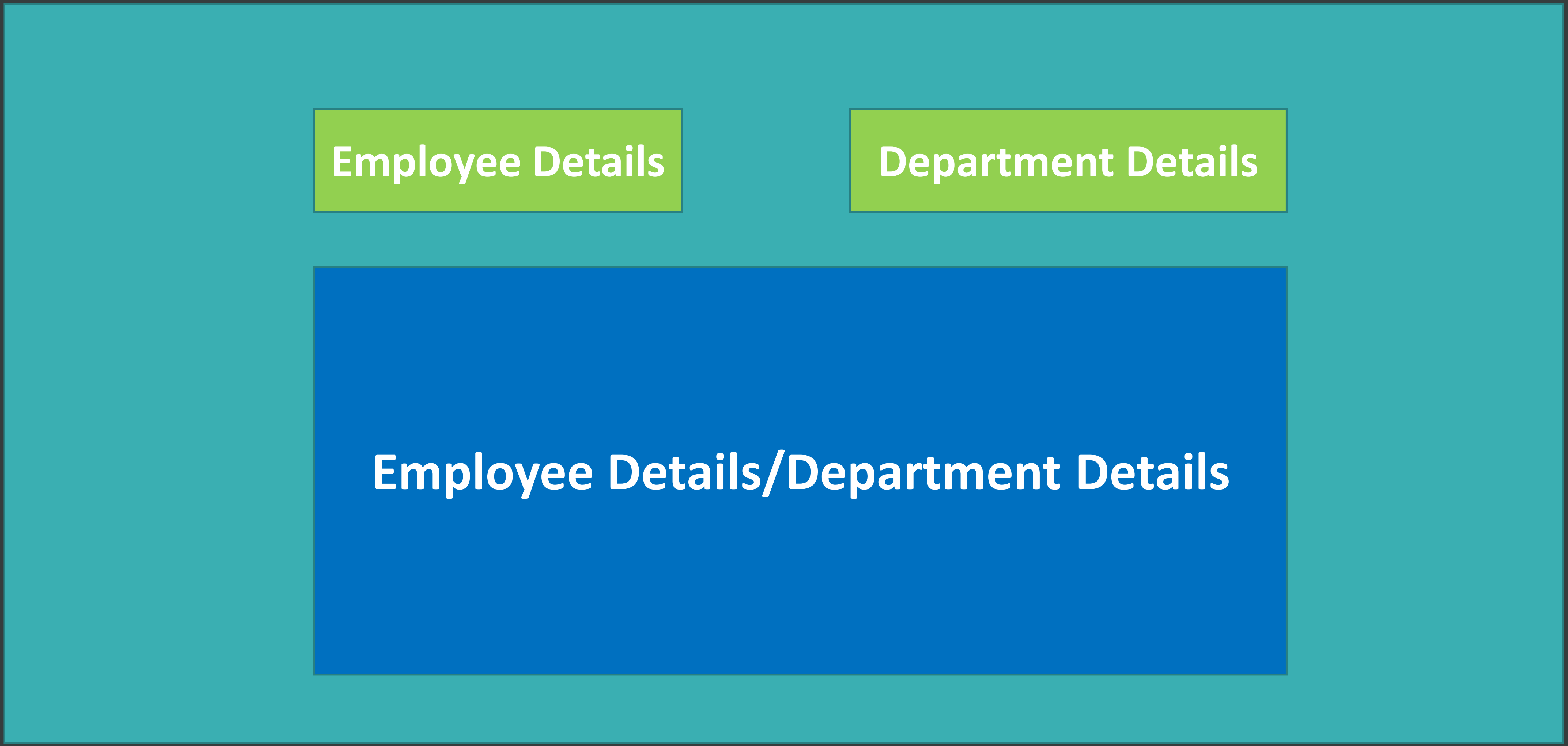
Routing Overview

- It can interpret a browser URL as an instruction to navigate to a client-generated view
- It can pass optional parameters to the 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

Routing Overview

- You can navigate imperatively when the user clicks a button, selects from a drop box, or in response to some other stimulus from any source
- The router logs activity in the browser's history journal so the back and forward buttons work as well.

Routing Application



Generate an application with routing enabled

```
ng  new  routing-app  --routing
```

Adding Routing Features in existing Angular Project

index.html

```
<!doctype html>
<head>
        <meta charset="utf-8">
        <title>Routing Application</title>
        <base href="/">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
      <body>
            <app-root></app-root>
      </body>
</html>
```

Component Creation for Two Views

1. ng g c employees -it -is --skipTests=true
2. ng g c departments -it -is --skipTests=true

- it (to create inline template)

- is (to create inline styles)

--skipTests=true (not to create .spec file)

AppRoutingModule

- Load and configure the router in a separate, top-level module that is dedicated to routing and imported by the root AppModule
- By convention, the module class name is AppRoutingModule and it belongs in the app-routing.module.ts in the src/app folder

Add the AppRoutingModule

```
ng generate module app-routing  --flat --module=app
```

--flat	puts the file in src/app instead of its own folder
--module=app	to register it in the imports array of the AppModule

src/app/app-routing.module.ts (updated)

```
import { NgModule }                  from '@angular/core';
import { Routes, RouterModule }      from '@angular/router';
import { EmployeesComponent }        from './employees/employees.component';
import { DepartmentsComponent}       from './departments/departments.component';

const routes : Routes = [
   { path: 'employees',     component: EmployeesComponent},
   { path: 'departments', component: DepartmentsComponent}
]
@NgModule({
          imports: [RouterModule.forRoot(routes)],
          exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.module.ts

```
import { AppRoutingModule }       from './app-routing.module';
import { EmployeesComponent } from './employees/employees.component';
import { DepartmentsComponent }  from './departments/departments.component';

@NgModule( {
   declarations: [ AppComponent,   EmployeesComponent, DepartmentsComponent],
    imports :       [BrowserModule, AppRoutingModule,    FormsModule],
    providers:    [EmployeeService],
    bootstrap:   [AppComponent]
})
export class AppModule { }
```

app.component.html

```
<h1>Routing application</h1>
<nav>
        <a      [routerLink]="/employees" routerLinkActive="active">
                Employee Details
        </a>
        <a      [routerLink]="/departments" routerLinkActive="active">
                Department Details
        </a>
</nav>
<router-outlet></router-outlet>
```

inserts the component (as the sibling) that gets matched based on the current browser's URL

routerLink (Directive)

- When applied to an element in a template, makes that element a link that initiates navigation to a route. Navigation opens one or more routed components in one or more <router-outlet> locations on the page.

- Given a route configuration:

```
[{ path: 'user/:name', component: UserCmp }]
```

- the following creates a static link to the route:

```
<a routerLink="/user/bob">link to user component</a>
```

routerLink (Directive)

- You can use dynamic values to generate the link. For a dynamic link, pass an array of path segments, followed by the params for each segment. For example:

```
['/team', teamId, 'user', userName, {details: true}]
```

- generates a link to

```
/team/11/user/bob;details=true
```

Relative Link Paths

- The first segment name can be prepended with /, ./, or ../

- If the first segment begins with /, the router looks up the route from the root of the app.

- If the first segment begins with ./, or doesn't begin with a slash, the router looks in the children of the current activated route.

- If the first segment begins with ../, the router goes up one level in the route tree.

Setting and Handling query params and fragments

The following link adds a query parameter and a fragment to the generated URL:

```
<a      [routerLink]="['/user/bob']" [queryParams]="{debug: true}"
        fragment="education">
                link to user component
</a>
```

The example generates the link:

```
/user/bob?debug=true#education
```

routerLinkActive (Directive)

- Tracks whether the linked route of an element is currently active, and allows you to specify one or more CSS classes to add to the element when the linked route is active.

```
<a routerLink="/user/bob" routerLinkActive="active-link">Bob</a>
```

- Whenever the URL is either '/user' or '/user/bob', the "active-link" class is added to the anchor tag. If the URL changes, the class is removed.

- You can set more than one class using a space-separated string or an array:

```
<a routerLink="/user/bob" routerLinkActive="class1 class2">Bob</a>
<a routerLink="/user/bob" [routerLinkActive]="['class1', 'class2']">Bob</a>
```

routerLinkActive (Directive)

- To add the classes only when the URL matches the link exactly, add the option exact: true:

```
<a routerLink="/user/bob" routerLinkActive="active-link"
        [routerLinkActiveOptions] = "{ exact: true }"> Bob </a>
```

- To directly check the isActive status of the link, assign the RouterLinkActive instance to a template variable:

```
<a routerLink="/user/bob" routerLinkActive #rla="routerLinkActive">
        Bob {{ rla.isActive ? '(already open)' : ''}}
</a>
```

routerLinkActive (Directive)

- You can apply the RouterLinkActive directive to an ancestor of linked elements. For example, the following sets the active-link class on the <div> parent tag when the URL is either '/user/jim' or '/user/bob'.

```
<div routerLinkActive="active-link" [routerLinkActiveOptions]="{exact:
true}">
        <a routerLink="/user/jim">Jim</a>
        <a routerLink="/user/bob">Bob</a>
</div>
```

Default Route and Page Redirection

```
const routes : Routes = [
        { path: '', redirectTo:   'employees', pathMatch: 'full'},
        { path: 'employees',      component: EmployeesComponent},
        { path: 'departments', component: DepartmentsComponent},
        { path: '**',                          component: DefaultPageComponent}
        // ** should be the last one
]
```

pathMatch Attribute

- pathMatch = 'full' results in a route hit when the *remaining*, unmatched segments of the URL match ''.
- In this example, the redirect is in a top level route so the *remaining* URL and the *entire* URL are the same thing.
- The other possible pathMatch value is 'prefix' which tells the router to match the redirect route when the remaining URL begins with the redirect route's prefix path (e.g. a/b/c has prefix a/b).
- This doesn't apply to this example because if the pathMatch value were 'prefix', every URL would match ''

Route Parameters

```
ng g c employee_details –is –it
```

Route Parameters

```
const routes : Routes = [
        { path: '', redirectTo: 'employees', pathMatch: 'full'},
        { path: 'employees', component: EmployeesComponent},
        { path: 'departments', component: DepartmentsComponent},
{ path: 'employees/:id', component: EmployeeDetailsComponent},
        { path: '**', component: DefaultPageComponent}
        // ** should be the last one
]
```

employees.component.html

```
<table>
    <tr    (click)="onEmpSelect(emp)"
           *ngFor="let emp of empList">
     <td> {{    emp.id                }}    </td>
     <td> {{    emp.name              }}    </td>
     <td> {{    emp.designation       }}    </td>
    </tr>
</table>
```

employees.component.ts

```
import { Router } from '@angular/router';
…
export class EmployeesComponent implements OnInit {
  empList : IEmployee[ ]

  constructor(  private empService : Employee2Service,
                private router: Router) {   }
  ngOnInit() {
        this.empService.getEmployee().subscribe(data => this.empList = data)
  }
  onEmpSelect(emp: IEmployee) {
        this.router.navigate(['/employees', emp.id])
  }
}
```

Employee-details.component.ts

```
template: ' <p> Employee id = {{ empId }}  </p> '
                                ...
export class EmployeeDetailsComponent implements OnInit {
  public empId

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
 this.empId = parseInt(this.route.snapshot.paramMap.get('id'))
  }
}
```

References

- https://angular.io/guide/router