HEXAWARE

Angular – Routing

Session Objective

- To understand and workout the base concept of routing and Navigation using dynamic forms from below topics.
  - Routes
  - RouterModules
  - RouterLink
  - Route Parameter

Routing & Navigation

Routing & Navigation

- When a user enters a web application or website, routing is their means of navigating throughout the application. To change from one view to another, the user clicks on the available links on a page.
- Angular provides a Router to make it easier to define routes for the web applications and to navigate from one view to another view in the application.

Routing

- In web development, routing means splitting the application into different areas usually based on rules that are derived from the current URL in the browser.
- For instance, if we visit the / path of a website, we may be visiting the home route of that website. Or if we visit /about we want to render the “about page”, and so on.

Why Do We Need Routing?

HEXAWARE

- Defining routes in our application is useful because we can:
  - separate different areas of the app;
  - maintain the state in the app;
  - protect areas of the app based on certain rules;

Components of Angular routing

- There are three main components that we use to configure routing in Angular:
  - **Routes** describes the routes our application supports
  - **RouterOutlet** is a “placeholder” component that shows Angular where to put the content of each route
  - **RouterLink** directive is used to link to routes

Routes & Router Module Cont…

- In order to use the router in Angular, import constants from the @angular/router package:
- import { RouterModule, Routes } from '@angular/router';

Routes & Router Module Cont…

- To define routes for our application, create a Routes configuration and then use **RouterModule.forRoot(routes)** to provide our application with the dependencies necessary to use the router

Routes & Router Module Cont…

- The mapping of URLs to Components on the page is done using Route Configuration, it’s an array which can defined as

```
const routes: Routes = [
{ path: '', component: HomeComponent },
{ path: 'search', component: SearchComponent }
];
```


Routes & Router Module Cont…

- The **path property** describes the URL this route will handle.
- The **component property** is the name of the component we want to display when the URL in the browser matches this path.

Routes & Router Module

- We then install these routes into our application by importing RouterModule.forRoot(routes) into NgModule,

```
@NgModule({
 imports: [
 .      .
 RouterModule.forRoot(routes)  ]
  .     .     .
})
class AppModule { }
```


RouterOutlet Directive

- We need to add a directive called router-outlet in the template HTML.
- This directive tells Angular where it should insert each of those components in the route,

```
<router-outlet></router-outlet>
```


Redirects Cont…

HEXAWARE

- There are more ways to configure the routes, for example we might like to change our routes to add some redirects,

```
const routes:Routes = [
 {path: '', redirectTo: 'home', pathMatch: 'full'},
 {path: 'find', redirectTo: 'search'},
 {path: 'home', component: HomeComponent},
{path: 'search', component: SearchComponent}
];
```


Redirects

- The **redirectTo** property describes the path we want to redirect the user to, if they navigate to this URL.
- Now if the user visits the root (empty) URL they are redirected to /home instead.
- For the special case of an empty URL we also need to add the pathMatch: 'full' property so Angular knows it should be matching exactly the empty string and not partially the empty string.

Catch all route Cont…

- We can also add a catch all route by using the path **, if the URL doesn't match any of the other routes it will match this route.

```
const routes:Routes = [
  {path: '', redirectTo: 'home', pathMatch: 'full'},
  {path: 'find', redirectTo: 'search'},
  {path: 'home', component: HomeComponent},
  {path: 'search', component: SearchComponent},
  {path: '**', component: HomeComponent}
 ];
```


Catch all route

- If nothing matches, the HomeComponent is shown
- Now if we navigate to /foo it will show us the welcome message.

Navigating via a routerLink directive

- Navigation can also be controlled by using the routerLink directive in the template,

```
<a class="nav-link"  [routerLink]="['home']">Home</a>
```

- The routerLink directive takes as input the same link params array format that the router.navigate(…) function takes.

routerLinkActive Cont…

- An important feature of any navigation component is giving the user feedback about which menu item they are currently viewing.
- Another way to describe this is giving the user feedback about which route is currently active.

routerLinkActive

- A routerLinkActive directive is associated with a route through a routerLink directive.
- It takes as input an array of classes which it will add to the element it's attached to if it's route is currently active,

```
<a    [routerLink]="['home']"    [routerLinkActive]="['active']">

Home </a>
```


Route Parameter

Route Parameters Cont…

HEXAWARE

The Two Ways to Grab Route Parameters

- **The Snapshot Way**: The router provides us with a snapshot of the current route
- **The Observable/Stream Way**: Since Angular employs Observables heavily, the router also returns an Observable that we can listen to.

Route Parameters

- Add a route parameter ID
- link the route to the parameter
- add the service that reads the parameter.

Declaring Route Parameters Cont…

- The route for the component that displays the details for a specific product would need a route parameter for the ID of that product.

```
export const routes: Routes = [
  { path: '', redirectTo: 'product-list', pathMatch: 'full' },
  { path: 'product-list', component: ProductList },
  { path: 'product-details/:id', component: ProductDetails }
];
```


Declaring Route Parameters

- **:id** in the path of the product-details route, which places the parameter in the path.
- For example, to see the product details page for product with ID 5, use the following
  - URL: localhost:3000/product-details/5

Linking to Routes with Parameters Cont…

- In the ProductList component list of products can be displayed. Each product would have a link to the product-details route, passing the ID of the product:

```
<a *ngFor="let product of products"
  [routerLink]="['/product-details', product.id]">
  {{ product.name }}
</a>
```


Linking to Routes with Parameters

- Alternatively we could navigate to the route programmatically:

```
<div (click)="goToProductDetails(pro)" *ngFor="let pro

of products" >

goToProductDetails(id) {

  this.router.navigate(['/product-details', id]);

}
```


Reading Route Parameters Cont…

- The ActivatedRoute service provides a params Observable which we can subscribe to to get the route parameters

```
export class LoanDetailsPage implements OnInit, OnDestroy {
  id: number;
  private sub: any;
  constructor(private route: ActivatedRoute) {}
  ngOnInit() {
    this.sub = this.route.params.subscribe(params => {
       this.id = +params['id'];
    });
  }
```


Reading Route Parameters

- Grabbing Route Parameters (The Snapshot Way)

```
constructor(private route: ActivatedRoute) {}
ngOnInit() {
 this.id = this.route.snapshot.params['id'];
 }
```


HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com