Angular | Lecture 4

Marina Magdy

Agenda

- Recap last lecture points
- Routing
- Route Configuration
- Navigation
- Parameterised Routes
- Router Guard

Routing

Routing

- In a single-page app, you change what the user sees by showing or hiding portions of the display that correspond to particular components, rather than going out to the server to get a new page. As users perform application tasks, they need to move between the different views that you have defined.

- To handle the navigation from one view to the next, you use the Angular Router. The Router enables navigation by interpreting a browser URL as an instruction to change the view.

Routing

The advantages of an SPA are:

- Can be faster. Instead of making a time-consuming request to a far away server every time the URL changes the client app updates the page much faster.
- Less bandwidth required. We don’t send over a big HTML page for every URL change, instead we might just call a smaller API which returns just enough data to render the change in the page.
- Convenience. Now a single developer can build most of the functionality of a site instead of splitting the effort between a front-end and server-side developer.

How to use routing in Angular

Route Configuration

https://angular.dev/guide/routing/common-router-tasks

There are fundamental building blocks to creating a route :

- **Set up a Routes array for your routes**

```
export const routes: Routes = [];
```

- **Define your routes in your Routes array**

```
const routes: Routes = [
      { path: 'first-component', component: FirstComponent },
      { path: 'second-component', component: SecondComponent }
];
```

Route Configuration

- Add your routes to your application

```
<a routerLink="/first-component">First Component</a>
```

- Make sure to import `RouterLink` in the array of imports of the component your are using the routerLink.

```
@Component({
 …
 imports: [RouterLink],
})
```

Route Configuration

- **Next, update your component template to include <router-outlet>.**

  The RouterOutlet is a directive from the router library that is used like a component. It acts as a placeholder that marks the spot in the template where the router should display the components for that outlet.

  This element informs Angular to update the application view with the component for the selected route, it acts as a placeholder that Angular dynamically fills based on the current router state.

Navigation

To navigate to other component will use :

- From template using Router link on <a> tag with 2 ways :
  - `<a class="nav-link" routerLink="/login">login</a>`
  - `<a class="nav-link" [routerLink]="['/details' , id ]">login</a>`

Navigation

Not Found Page

The two asterisks, **, indicate to Angular that this routes definition is a wildcard route. For the component property, you can define any component in your application. Common choices include an application-specific PageNotFoundComponent, which you can define to display a 404 page to your users; or a redirect to your application's main component. A wildcard route is the last route because it matches any URL.

```
{path: '**', component:PageNotFoundComponent }
```

Navigation

Setting the page title

Each page in your application should have a unique title so that they can be identified in the browser history. The Router sets the document's title using the title property from the Route config.

```
{
   path: '',
   component:HomeComponent,
   title: 'Home component',
},
```

Navigation : routerLinkActive

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.

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="/user/bob" r**outerLinkActive="active-link"**>Bob

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

You should import `RouterLinkActive` In the array of imports of the component.

Navigation : routerLinkActiveOptions [Self-Study]

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

This will apply the style to the exact matched route.

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

Navigation

To Navigate from TS file you can use navigate method from router service.

- You need to inject router in constructor first

```
constructor( private router: Router  ) {}
```

- Then use navigate method to pass the path you want to navigate to

```
this.router.navigate(['/games']);

this.router.navigate(['/games' , id ]); // In case of using parameterized routes
```

Navigation : Nested Routes [Extra]

As your application grows more complex, you might want to create routes that are relative to a component other than your root component. These types of nested routes are called child routes. This means you're adding a second <router-outlet> to your app, because it is in addition to the <router-outlet> in AppComponent.

For Example you can add child-a, and child-b component to componentX for example. Here, so componentX has its own <nav> and a second <router-outlet> in addition to the one in AppComponent.

Navigation：Nested Routes [Extra]

ComponentX implementation.

```
<p>ComponentX content!</p>

<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>
```

Navigation : Nested Routes [Extra]

And you need to define the paths in the routes array

```
{
    path: 'parentX',
    component: ComponentX,
    children: [
      {
        path: 'child-a', // child route path
        component: ChildAComponent,
      },
      …
   ]}
```

Parameterised Router

Navigation : Parameterised Routes

Sometimes we need part of the path in one or more of our routes (the URLs) to be a variable, a common example of this is an ID for example : /movie/1

We will learn to :

- Configure parameterised routes in our route definition object.
- Components can be notified with what the parameter values are when the URL gets visited.
- Have optional parameters in routes.

Navigation : Parameterised Routes

First we need to configure this id in routes array

```
const routes: Routes = [
 { path: movie/:id', component: MovieComponent} (1)
];
```

The path has a variable called id, we know it’s a variable since it begins with a **colon :**

Navigation : Parameterised Routes

To go to this parameterised route we will be from html or ts

From html :

```
[routerLink]="[‘game-details’, game.id]”
```

From ts:

```
this.router.navigate(['movie', idValue]);
```

Navigation : Parameterised Routes

To get this parameter in component will user ActivatedRoute

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

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

Navigation : Parameterised Routes

With Angular 16 and 17 some updates.

you want to pass information from one component to another. For example, consider an application that displays a shopping list of grocery items. Each item in the list has a unique id. To edit an item, users click an Edit button, which opens an EditGroceryItem component. You want that component to retrieve the id for the grocery item so it can display the right information to the user.

https://angular.dev/guide/routing/common-router-tasks#getting-route-information

Navigation : Parameterised Routes

Use a route to pass this type of information to your application components. To do so, you use the withComponentInputBinding feature with provideRouter.

withComponentInputBinding: Enables binding information from the Router state directly to the inputs of the component in Route configurations.

```
providers: [
  provideRouter(appRoutes, withComponentInputBinding()),
]
```

Navigation : Parameterised Routes

Then update the component to have an Input matching the name of the parameter.

```
@Input() id?: string;
```

This will returns the id value that defined in the route

Navigation : Parameterised Routes [Extra]

To pass query params to URL you can bind [queryParams] on anchor tag.

```
<a [routerLink]="['/game' , id]" [queryParams]="{category:
'action'}">Game</a>
```

Or from TS using Navigate

```
this.router.navigate([`game` ,  id ] , { queryParams : { category
: 'action' } })
```

To read the query params in component use @Input

```
@Input('name') name?: string;
@Input('age') age?: string;
```

Navigation

withViewTransitions

With this minimal setup, you can instantly enhance your application's navigation with smooth fade-in and fade-out animations

```
providers: [
    provideRouter(
      [...], // Your route configurations
      withViewTransitions() // Enable View Transitions
    ),
  ],
```

Guard [Extra]

Navigation : Route Guard

Generate a guard :

ng generate guard auth-guard

if you have error during create guard please add - -implements CanActivate to the generate command

Add this guard to routes config :

```
{
    path: 'movie',
    component: MovieDetailsComponent,
    canActivate: [AuthGuardGuard]
},
```

Navigation : Route Guard

In your guard check if user have permission to view page or not:

```
// In Guard File
if (this.isLoggedIn) {
  return true;
}
window.alert("You don't have permission to view this page");
return this.router.navigate(['']) || false;
```

Navigation : Can Deactivate [Self Study]

Prerequisite: **Forms**

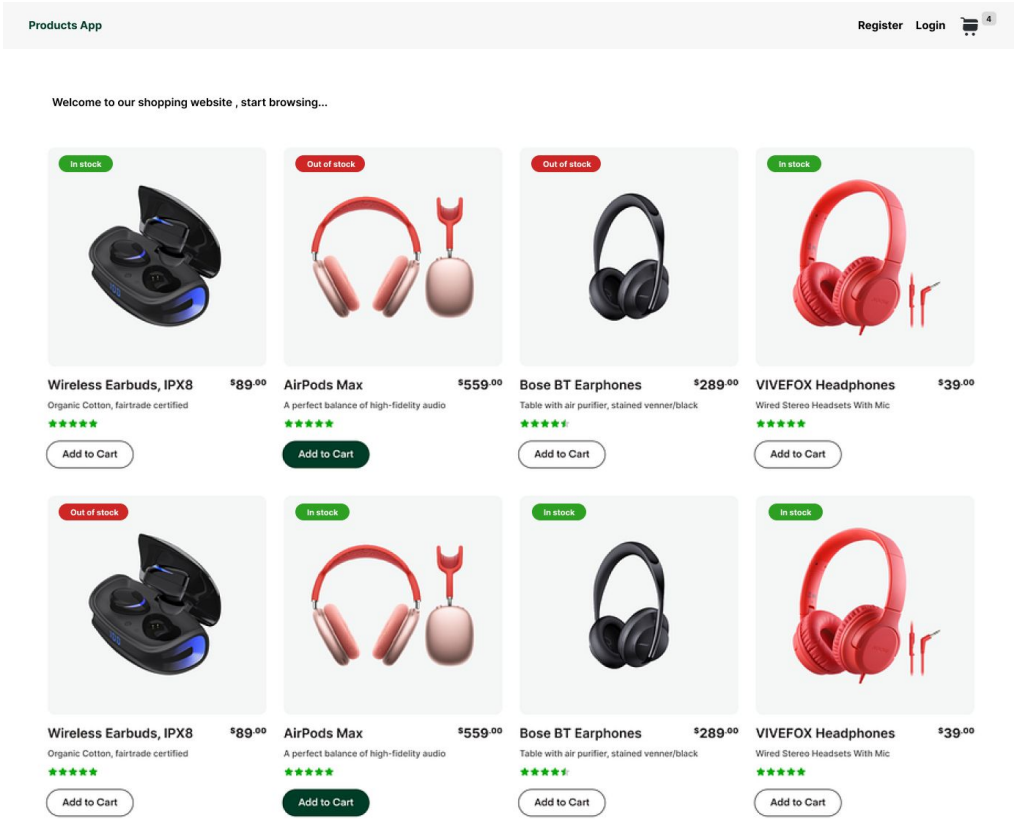
Another type of guard we can add to our application is a CanDeactivate guard which is usually used to warn people if they are navigating away from a page where they have some unsaved changes.

Thank you

Lab

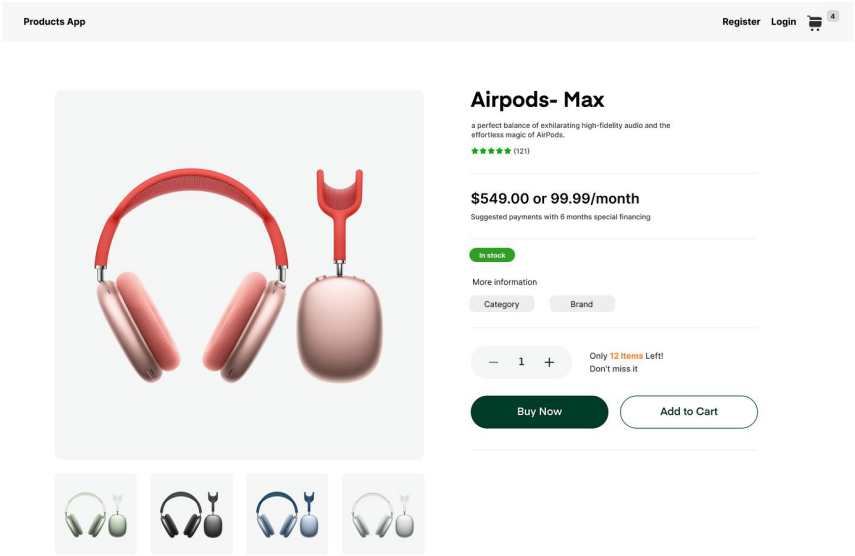
Products

- Create the routes for each page.
- Navbar will contain
  - Products page route and will be the default
  - Route for login
  - Route for register
  - Route for cart
- The active link should be colored with different color [like orange color for products as it’s the active route]
- Create a Not Found page



Products

- Create a product details component.
- When user click on any product card from the list he should be redirected to details page with the route
  - /product/:id [using dynamic routes]
- Filter the array of products with the product id you have from the route to get the single product data
- Pass the product details to the html to show data to user [Product images , title, category, brand, rating, description].



Products

- **[Bouns]** create a custom pipe to calculate the price based on the discountPercentage if exists and show the price before and after discount.