Angular

Lecture 5

Agenda

- Routing
- Route Configuration
- Navigation
- Parameterised Routes
- Nested routes
- Router Guard
- Modules

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.

Route Configuration

There are fundamental building blocks to creating a route :

- Import the `AppRoutingModule` into AppModule and add it to the imports array.
- Import `RouterModule` and Routes into your routing module.
- Define your routes in your Routes array into your routing module.
- Add your routes to your application using routerLink and update your component template to include <router-outlet>. This element informs Angular to update the application view with the component for the selected route.

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 [routerLink]="'/login'">Login</a> or [routerLink]="['/login']"`

Navigation

To navigate to other component will use :

- From .ts file using navigate :
  - `import { Router } from '@angular/router';`
  - Create instance from router service (`private route:Router`)
  - `this.route.navigate(['/login']);`

Navigation

- **Add child routes to a parent one :**

```
{ path: "artist/:artistId", component: ArtistComponent,

children: [

{ path: "", component:ArtistTrackListComponent }]

},
```

- **Add general route for not found routes :**

```
{ path: "**", component: HomeComponent } or {path: '**',
component:PageNotFoundComponent }
```

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

Home

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]="[‘movie’, movie.id]”
```

From ts:

```
this.router.navigate(['movie', {id:idValue}]);
```

Navigation : Parameterised Routes

To get this parameter in component will user ActivatedRoute

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


constructor(private route: ActivatedRoute) {
   this.route.params.subscribe( params => console.log(params) );
}
```

Navigation : Route Guard

Generate a guard :

```
ng g guard auth-guard
```

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:

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

Navigation : Can Deactivate [Self Study]

A third 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.

Modules

- Feature modules are NgModules for the purpose of organizing code.
- As your app grows, you can organize code relevant for a specific feature. This helps apply clear boundaries for features. With feature modules, you can keep code related to a specific functionality or feature separate from other code.

Resources :

https://angular.io/guide/feature-modules

Modules

- **To make a feature module**

```
ng generate module AuthModule
```

- **To create component within module:**

```
ng generate component AuthModule/Login
```

- Import Feature Module into app module in imports
- Tbo be able to use login component in app module , you first have to export the loginComponent in the AuthModule

Modules

Shared modules

Creating shared modules allows you to organize and streamline your code. You can put commonly used directives, pipes, and components into one module and then import just that module wherever you need it in other parts of your app such as forms module or any shared module or component.

You should add shared module to each module you create to be able to use this shared modules and components in it.

Add components and modules you want to share in feature or shared modules to exports array.

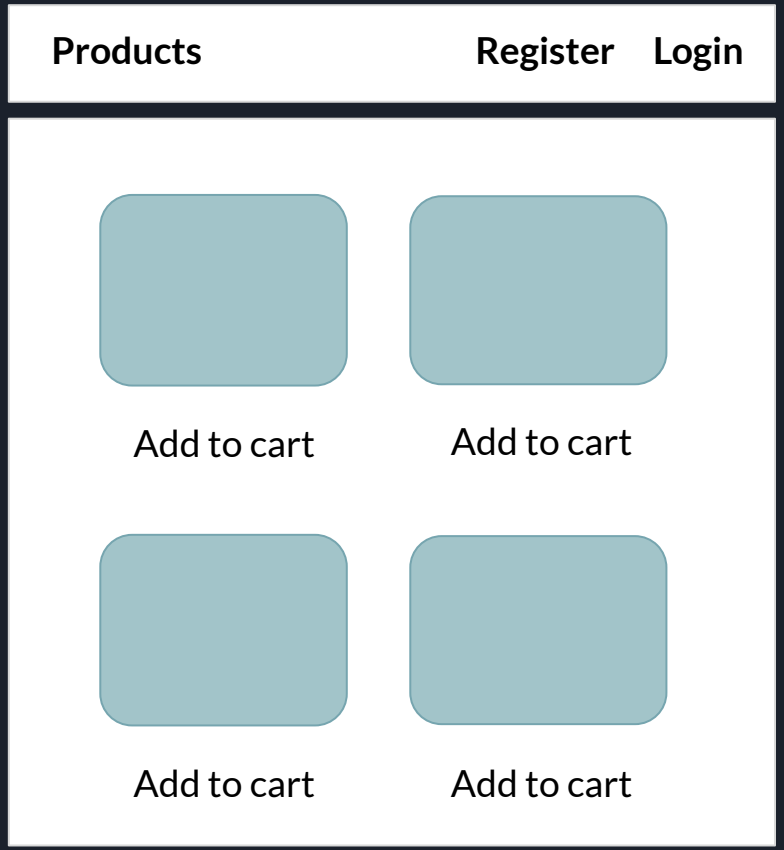
Thank you

Lap

Products

Continuing on the previous task

- Navbar will contain
  - Products page route and will be the default when open the project
  - Route for login
  - Route for register
- Create a Not Found page
- Create a new route called protected route and make it private and can be visited only if user logged in [Static for now]
- Create a can deactivate option to prevent user from leaving the registered page if user entered any data.**[Bonus]**



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.
- Separate the app to modules [Shared , Products , Auth Modules]