JOHN BRYCE
Leading in IT Education
a matrix company

Angular Routing

Advanced Routing with @angular/router

Our Goal

JOHN BRYCE
Leading in IT Education
a matrix company

We will learn advanced routing in Angular application, while creating the following apps:

- Home page - our app will have a homepage with textual data
- About page - we will have about page with textual data
- Todo pages - our app will have a link for a todo list page displaying the list of tasks from the server
- Task detail - clicking an item from the todo list will move us to a todo item page where we display more items on the list
- Admin - there will be admin area available for logged in users

Our Goal

JOHN BRYCE
Leading in IT Education
a matrix company

While creating our app, we will cover the following:

- Configure routing
- **Router-outlet**
- Handling links
- Transferring information on the URL
- Router hooks
- Route guards
- Lazy loading
- Server side

Web App Routing

JOHN BRYCE
Leading in IT Education
a matrix company

- User types a url in the address bar- we need to direct him to the proper resource he is asking
- User presses a link- we need to pass him to the proper page and change the url
- User press back and forth- we need to move him to the proper page
- User reloads the app- we need to transfer the proper resource
- User search our app- we need to add url query param (upon reload display the same search results)
- User add filters- we need to adjust the url with proper query params (upon reload display the same filtering)
- User navigates to a resource- we need to add the proper matrix param
- We need to add the proper slug for SEO
    - **/tasks/1/buy-groucery/**

@angular/router

JOHN BRYCE
Leading in IT Education
a matrix company

- **@angular/router** is the Angular package that helps us deal with routing
- To install:
  - **npm install @angular/router --save**
- **@angular/router** is installed when we start a **@angular/cli** project
- We need to add the proper module to our **imports** array

<base>

JOHN BRYCE
Leading in IT Education
a matrix company

- We need to instruct the router how to build the urls for the app
- The urls will be appended to the **href** attribute of the base tag
- In most cases our app will be in the root url so we should add
  - **<base href="/">**
- **@angular/cli** already placed default base tag with href set to the root

Router Service

JOHN BRYCE
Leading in IT Education
a matrix company

- There is a singleton router service for Angular application that uses routing
- The job of the router service is to match routes and load the proper component according to the match
- By default there is no routes defined

RouterModule

JOHN BRYCE
Leading in IT Education
a matrix company

- To add the **RouterModule** you have to static **creation** functions
  - **forRoot**
  - **forChild**
- The creation function requires us to pass first argument that describes the routes
- For each module we create (root or feature module) we will add the routing in seperate module
- For our **AppModule** we will create **AppRoutingModule** and will wrap the routing logic in it's own module
- For our **AppRoutingModule** we will define the following pages:
  - **HomePageComponent**
  - **AboutPageComponent**
  - **Error404PageComponent**

RouterOutlet Directive

JOHN BRYCE
Leading in IT Education
a matrix company

- This directive control where the router needs to place the component after it has a match
- We will place the directive in the **app.component.html**
- We can place common things to all routes in the **app.component.html**
- Let's place the directive and check if our routing works

Navigation

JOHN BRYCE
Leading in IT Education
a matrix company

- Let’s add navigation bar to our app
- We can navigate by using the **routerLink** directive
- We can navigate by injecting the **Router** service and navigating by code
- Let’s add a button to the home page and navigate by code to the about page
- **routerLink** can also be dynamic

Routing for Feature Module

JOHN BRYCE
Leading in IT Education
a matrix company

- If a feature module should add routing then you should place an additional routing module for it
- In our todo application we created a **TodoModule**
- The todo application should have a home page
- The todo application should have the url **/tasks** which displays a list of all the tasks
- The todo application should have a route of **/tasks/:id** which displays detail about a single todo task.
- Let's create the todo module with the task list component
- We will create a route that will lead to a page displaying the list
- We will also create a page that displays a single component
- We will need to create a service which query the service for the list and for a single task

Nested Routing

JOHN BRYCE
Leading in IT Education
a matrix company

- similar to what we did with the navigation bar we can nest additional routes and create a parent child routing hierarchy
- The component part of the route is optional
- you can place **children** array to nest routes
- if the parent route has component in it then the children will be nested in the parent **router-outlet** directive
- you can also live path as empty string if you want to deal with the index route

Matrix Params

- The tasks list component will have to pass matrix param to the detail route
- To define a matrix param in the **Route** object you place in the path the following syntax
  - **/tasks/:id**
- You can inject the **ActivatedRoute** and grab the matrix params from there
- The **ActivatedRoute** holds the route associated with the current component

Persist complex state in URL

JOHN BRYCE
Leading in IT Education
a matrix company

consider the following case:

- you are building a search page
- the search page contains a component with a complex bar with filtering and sorting
- Another component is in charge of displaying the data
- How can the 2 components pass the data?
- Can we pass the state without creating extra coupling?
- Can we utilize the fact that the query params arrive with an observable?

Load Feature Routes - Lazy Loading

JOHN BRYCE
Leading in IT Education
a matrix company

- We use **loadChildren** to point to our feature module
- Angular will load those components lazily only when the user navigates to that page which will optimize performance

Search Tasks

JOHN BRYCE
Leading in IT Education
a matrix company

- In our above the task list we want to implement a search from the task list
- The rest server supports filtering by search string
  - https://nztodo.herokuapp.com/api/task/?format=json&search=**<search-query>**
- In our task service we will add a method to search the list of tasks
- We will add a search component
- The search component will contain an input to search that will alter the url and add a search query param
- The list component will list for the search param and apply the youtube filter to optimize the request

Router Lifecycle

JOHN BRYCE
Leading in IT Education
a matrix company

- A routing operation goes through different lifecycle stages
- The **Router** has a read only property **events: Observable<Event>**
- The events is an **Observable** stream which will pulse an **event** before lifecycle stage
- You can subscribe to that stream and by the type of the event know which lifecycle stage is on
- Lets subscribe for the **NavigationEnd** event and log the route we are going to

Router Lifecycle Events

- **NavigationStart**
- **RouteConfigLoadStart -** lazy loading route config of lazy module
- **RouteConfigLoadEnd**
- **RoutesRecognized -** when routes are recognized
- **GuardsCheckStart**
- **ChildActivationStart**
- **ActivationStart**
  - **GuardsCheckEnd**
  - **ResolveStart**
  - **ResolveEnd**
  - **ActivationEnd**
  - **ChildActivationEnd**
  - **NavigationEnd**
  - **NavigationCancel**

Router Hooks

JOHN BRYCE
Leading in IT Education
a matrix company

- In the **Route** config you can connect hooks to the router lifecycle
- The hooks are injectable classes that implement an interface
  - **CanActivate**
  - **CanActivateChild**
  - **CanDeactivate**
  - **Resolve**
  - **CanLoad**

Server Side Implications

- Although we are writing a SPA we still need to take care of cases the user reloads our app
- The server needs to SSR in different paths not just the root path
- If we are not implementing SSR we can just server the **index.html** on every route
- We can distinguish between downloading a static resource and a web route if the path the user enters contains a dot

Router Summary

JOHN BRYCE
Leading in IT Education
a matrix company

- Good routing conventions are important part of our web application
- Now our routing also controls the code the user download to the browser and we can control this with lazy loading which is an important feature of routing
- We also learned about the proper way to keep our app modular and routes modular