

Angular

Lab 9

Routing

Exercises

1	LAB SETUP	1
2	GENERATING A NEW ANGULAR PROJECT	1
3	REGISTERING ROUTES TO USE WITH ROUTEROUTLET	1
4	USING ROUTERLINK AND ROUTERLINK ACTIVE TO CONNECT TO AND IDENTIFY ACTIVE ROUTES	1
5	ROUTE REDIRECTING AND WILDCARD ROUTES	2
6	PASSING ROUTE PARAMETERS THROUGH DYNAMIC ROUTING	2
7	PASSING QUERY PARAMETERS	2
8	NESTED / CHILD ROUTES	2
9	PROGRAMMATIC NAVIGATION AND SETTING PAGE TITLE	2

1 Lab setup

2 Generating a new Angular project

3 Registering routes to use with RouterOutlet

4 Using RouterLink and RouterLink Active to connect to and identify active routes

Q1. Create a new component called Articles and provide some suitable HTML to identify its template

Q2. Provide a link to navigate to this page from the root template and also configure this route appropriately.

.

5 Route redirecting and wildcard routes

6 Passing route parameters through dynamic routing

Q1. Building on the ArticlesComponent that was created in the solution for Part 4, add in a String array called `articleNames` in HomeComponent containing a number of random strings, and then bind these strings as a path suffix using the `routerLink` directive to the end of the `/articles` path that matches to ArticlesComponent, in a similar way that was done for the HeroComponent. Provide a suitable route configuration for dynamic routes so that these strings becomes the route parameters when appended to the end of the `/articles` path.

Q2. In ArticlesComponent, retrieve the single route parameter from the path URL using Approach # 2 (The Observables approach) and display in the template.

7 Passing query parameters

Q1. Building on the ArticlesComponent that was created in the solution for Part 4, add in an array of 3 objects called `articleObjects`, where each object contains a single property `articleName` with a random string value of your choosing (for e.g. `{ articleName : 'Nice article' }`). Bind each of the objects in this array using the `*for` directive to an `<a>` tag in the template of ArticlesComponent which links to the `/articles` path, and uses the specific object in the array as a query parameter. You can use the article name as the content of the `<a>` link itself.

Also ensure that you change the routing configuration so that the `/articles` path no longer needs to receive a route parameter.

Q2. In ArticlesComponent, retrieve the single query parameter from the path URL using Approach # 2 (The Observables approach) and display in the template.

8 Nested / child routes

Q1. Generate 2 new child components `sadArticle` and `happyArticle`. Make them nested routes of the ArticlesComponent that was created in the solution for Part 4. Provide `<a>` links to them in the template of ArticlesComponent (there is no need to pass any route or query parameters to them). Ensure that your routing configuration reflects these child components so you can navigate effectively to them and provide a link to the main Articles page from the root template.

9 Programmatic navigation and setting page title

Q1. Building on the ArticlesComponent that was created in the solution for Part 4, add a field to obtain input from the user for an article name (implement with NgModel two way binding in the HomeComponent template). Then provide a button, which when clicked on, navigates to the ArticlesComponent view, passing along the article name as a suffix to the `/articles` path portion. Ensure that you have a suitable route configuration for dynamic route for ArticlesComponent.

Q2. In ArticlesComponent, retrieve the single route parameter from the path URL using Approach # 2 (The Observables approach) and display in the template.