Routing in Angular 8

July 21, 2019

Routing

Angular Router is a powerful JavaScript router is built and maintained by the Angular core team that can install from the **package @angular/router**. Routing provides a complete routing library with the possibility of multiple router outlets, different path strategies.

The angular Router is the main part of Angular platform. It allows developers to build single-page applications with multiple views and allow navigation between views.

Angular supports SPA using routing module ngRoute. The routing module acts based on the URL. It works as a browser navigation's bar, and it was navigating between the pages.

- Enter URL in the address bar, and then the browser will navigate to the corresponding page.
- Click on the link to the page and the browser will navigate to the new page.
- Click on the browser on the back or forward, and the browser will navigate backward or forward according to the history pages.

Working of Router:

Whenever a user acts, such as clicking a link which loads a new page in the browser, the Router intercepts the browser behavior, and show and hide the view hierarchy.

If the Router finds the current application, it requires particular functionality, and it can be lazy to load the module on demand.

button works well.

To define navigation rules, we associate navigation path with our component. A path uses a URL- like a syntax that integrates our program data in the same way that template syntax integrates. We can apply program logic to choose views to show or hide, in response to user input and our access rules.

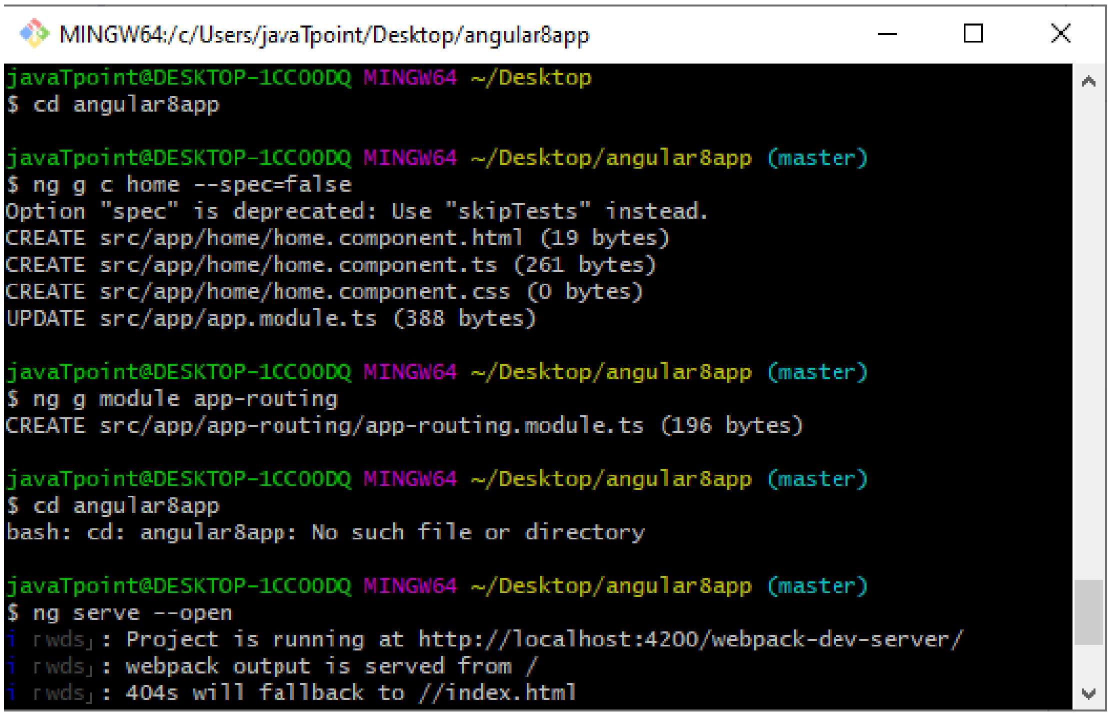
How to Set-up routing in Angular 8?

Routing could be easily added to a project. We were prompt if we have the message, " **Would you like to add angular routing?"(Y/N)**,if we answered by Y**.** The angular Router was set up in our project without having added it manually.

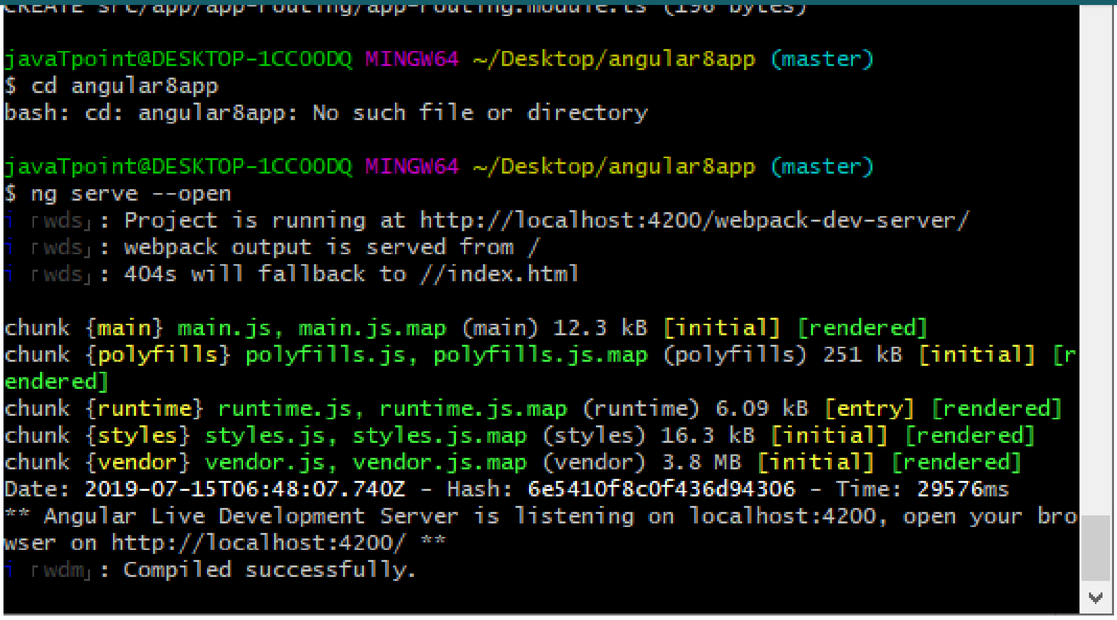
Otherwise, if we have no option like that, then we have to import it manually in our project.

- Firstly open our project where we have to import or add the routing module.
- After that, for creating the module we have to write the command **ng g c home –spec=false.**
- In the same directory, we have to write the command **ng g module app-routing**

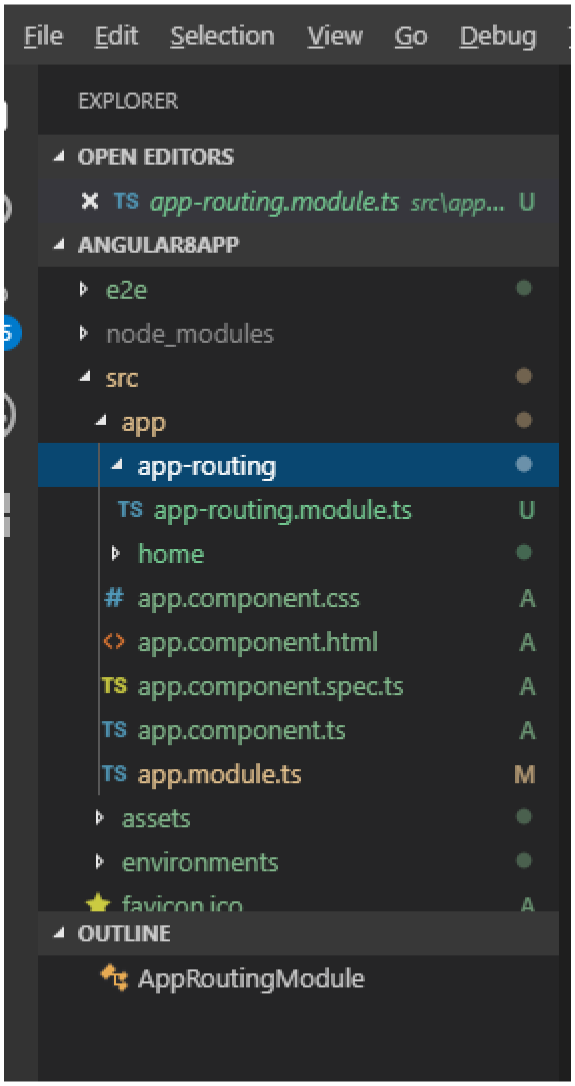
According to the following screenshot.



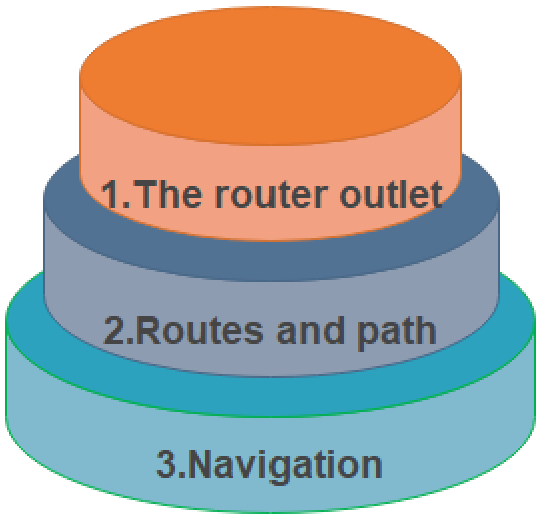
```
javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop
$ cd angular8app

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ ng g c home --spec=false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.ts (261 bytes)
CREATE src/app/home/home.component.css (0 bytes)
UPDATE src/app/app.module.ts (388 bytes)

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ ng g module app-routing
CREATE src/app/app-routing/app-routing.module.ts (196 bytes)

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ cd angular8app
bash: cd: angular8app: No such file or directory

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ ng serve --open
i ｢wds｣: Project is running at http://localhost:4200/webpack-dev-server/
i ｢wds｣: webpack output is served from /
i ｢wds｣: 404s will fallback to //index.html
```



```
CREATE src/app/app-routing/app-routing.module.ts (196 bytes)

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ cd angular8app
bash: cd: angular8app: No such file or directory

javaTpoint@DESKTOP-1CC00DQ MINGW64 ~/Desktop/angular8app (master)
$ ng serve --open
i ｢wds｣: Project is running at http://localhost:4200/webpack-dev-server/
i ｢wds｣: webpack output is served from /
i ｢wds｣: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 12.3 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 251 kB [initial] [r
endered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.09 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.3 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.8 MB [initial] [rendered]
Date: 2019-07-15T06:48:07.740Z - Hash: 6e5410f8c0f436d94306 - Time: 29576ms
** Angular Live Development Server is listening on localhost:4200, open your bro
wser on http://localhost:4200/ **
i ｢wdm｣: Compiled successfully.
```



included are standard view routing, nested child routes, named routes, and route parameters. The concept related to the Router are:



The Router-outlet

The Router-outlet is a directive accessible from the router library where the Router inserts the component and gets matched based on the current browser's URL. We can add multiple outlets in ou angular application, which enables us to implement advanced routing scenarios.

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

Routes And Paths

Routes are definitions comprised of a path and component attributes. The path mention to the part of the URL that determines a unique view that can be displayed, and refer to the Angular component that needs to be associated with a path.

In a component, each route maps a URL path.

The path could take a wildcard string (**). The Router selects this route if the called URL doesn't match the explained routes. It can be used for displaying a **"Not Found"** view or redirecting to a specific view if there is no match.

Example:

```
{path:'contacts', component: ContactListComponent}
```

Route Params

To creating the routes with parameters is a common feature in web apps. The angular Router allows us to access parameters in different ways.

We can create a router parameter using the colon syntax.

```
{path:‘contacts/:id’, component: contactDetailcomponent}
```

Route Guard

A route guard is the feature of the Angular Router which allows the developer to run logic if a router is requested, and it is based on the logic. It allows and denies the user access to the route. We can add a route guard by implementing the CanActivate interface available from **@angular/router** package. It **can activate()** method which holds the logic to allow and deny access to the route.

Example:

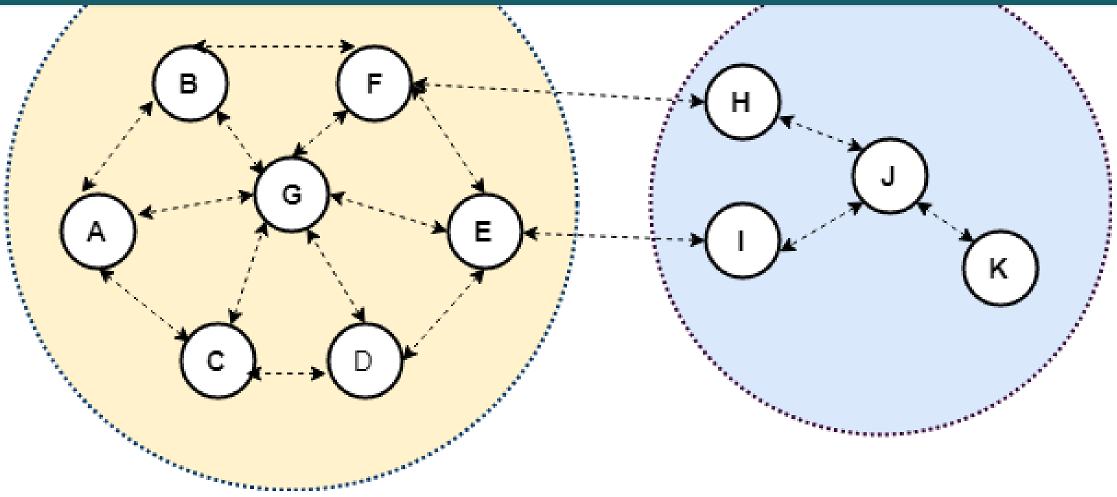
```
Class MyGuard implements CanActivate {
can activate () {
return true;
}
}
```

Navigation Directive

The Angular Router provides the router link directive to create the navigation links. This directive generates the path associated with the component to navigate.

For example:

```
<a[routerLink]= " ‘/contacts' ">Contacts</a>.
```



Router Path

← Previous

Next →

by

Angular 8

Hill Climbing Algorithm

Angular 8 Directives

Latest Tutorials

Angular 8 Tutorial

AI Tutorial

Machine Learning Tutorial

Selenium Tutorial

Hadoop Tutorial

Operating System Tutorial

DBMS Tutorial