Capgemini

Angular 4

Routing

Routing

Routing means loading sub-templates depending upon the URL of the page.

We can break out the view into a layout and template views and only show the view which we want to show based upon the URL the user is accessing.

Routes are a way for multiple views to be used within a single HTML page. This enables you page to look more "app-like" because users are not seeing page reloads happen within the browser.

Defining routes in application can:

- Separate different areas of the app
- Maintain the state in the app
- Protect areas of the app based on certain rules

AngularJS Routes

AngularJS routes enable us to create different URLs for different content in our application.

Having different URLs for different content enables the user to bookmark URLs to specific content.

In Angular 4 routes are configured by mapping paths to the component that will handle them.

For instance, let consider an application with 2 routes:

- A main page route, using the /#/home path;
- An about page, using the /#/about path;
- And when the user visits the root path (/#/), it will redirect to the home path.

Routing Setup

To implement Routing to Angular Application

Import RouterModule and Routes from ‘@angular/router’

```
import { RouterModule, Routes} from '@angular/router';
```

Define routes for application

```
const routes: Routes = [ { path: 'home', component: HomeComponent }];
```

- Install the routes using RouterModule.forRoot(routes) in the imports of NgModule

```
imports: [ BrowserModule,  RouterModule.forRoot(routes)]
```

Components of Angular 4 routing

There are three main components are used to configure routing in Angular

Routes

- Describes the routes application supports

RouterOutlet

- A “placeholder” component that gets expanded to each route’s content

RouterLink

- Directive is used to link to routes

Routes

To define routes for application, create a Routes configuration and then use RouterModule.forRoot(routes) to provide application with the dependencies necessary to use the router.

- path specifies the URL this route will handle
- component maps to the Component and its template
- optional redirectTo is used to redirect a given path to an existing route

```
const routes: Routes = [
    { path: '', redirectTo: 'home', pathMatch: 'full' },
    { path: 'home', component: HomeComponent },
    { path: 'about', component: AboutComponent },
    { path: 'contact', component: ContactComponent },
    { path: 'contactus', redirectTo: 'contact' },
];
```

RouterOutlet

The router-outlet element indicates where the contents of each route component will be rendered.

RouterOutlet directive is used to describe to Angular where in our page we want to render the contents for each route

```
@Component({
selector: 'my-app',
template:`<div class="container">
      <router-outlet></router-outlet>
      </div>`
})
```

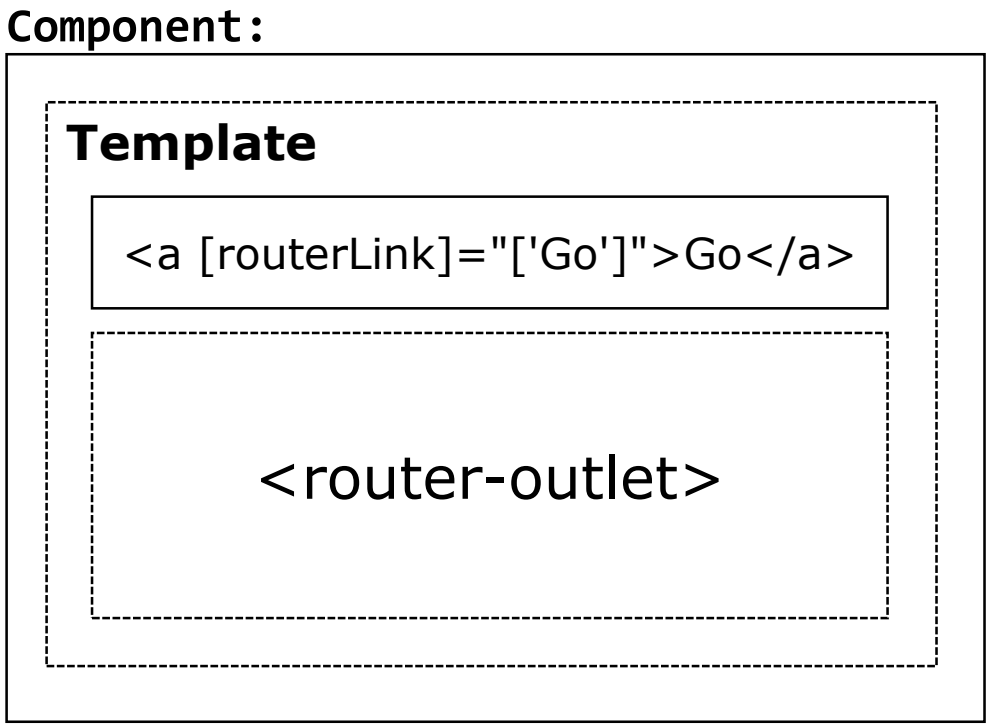
RouterLink

It generates link based on the route path.

routerLink navigates to a route

```
<div>
   <a [routerLink]="['Home']">Home</a>
   <a [routerLink]="[`About']">About
Us</a>
</div>
```

RouterOutlet & RouterLink



HTML : <a [routerLink]="['Go']">Go

Code : router.navigate(['Go']);

Routing Strategies

The way the Angular application parses and creates paths from and to route definitions is now location strategy.

HashLocationStrategy ('#/')

PathLocationStrategy (HTML 5 Mode Default)

```
//import LocationStrategy and HashLocationStrategy
import {LocationStrategy, HashLocationStrategy} from
'@angular/common';

//add that location strategy to the providers of NgModule
providers: [
  { provide: LocationStrategy, useClass: HashLocationStrategy }
]
```

Route Parameters

Route Parametes helps to navigate to a specific resource. For instance product with id 3

- /products/3

route takes a parameter by putting a colon : in front of the path segment

- /route/:param

To add a parameter to router configuration and to access the value refer the code given below

```
const routes: Routes =([
     { path:'/products/:id', name:`Product',
component:ProductComponent }
])

/*To access the parameter value */
routeParams.get('id')
```

ActivatedRoute

In order to access route parameter value in Components, we need to import ActivatedRoute

```
import { ActivatedRoute } from
'@angular/router'
```

inject the ActivatedRoute into the constructor of our component

```
export class ProductComponent {
  id: string;

  constructor(private route: ActivatedRoute) {
    route.params.subscribe(params => { this.id = params['id'];
});
  }
}
```

Demo

RoutingDemo