Angular Routing

Most web applications need to support different URLs to navigate users to different pages in the application. That's where a router comes in.

In traditional websites, routing is handled by a router on the server:

1. a user clicks a link in the browser, causing the URL to change
2. the browser sends an HTTP request to server
3. the server reads the URL from the HTTP request and generates an appropriate HTTP response
4. the server sends the HTTP response to the browser.

In modern JavaScript web applications, routing is often handled by a JavaScript router in the browser.

What is a JavaScript router?

In essence, a JavaScript router does two things:

1. update the web application state when the browser URL changes
2. update the browser URL when the web application state changes.

JavaScript routers make it possible for us to develop single-page applications (SPAs).

An SPA is a web application that provides a user experience similar to a desktop application. In an SPA, all communication with a back end occurs behind the scenes.

When a user navigates from one page to another, the page is updated dynamically without reload, even if the URL changes.

There are many different JavaScript router implementations available.

Some of them are specifically written for a certain JavaScript framework such as Angular, Ember, React, Vue.js and Aurelia, etc. Other implementations are built for generic purposes and aren't tied to a specific framework.

What is Angular Router?

Angular Router is an official Angular routing library, written and maintained by the Angular Core Team.

It's a JavaScript router implementation that's designed to work with Angular and is packaged as @angular/router.

First of all, Angular Router takes care of the duties of a JavaScript router:

- it activates all required Angular components to compose a page when a user navigates to a certain URL
- it lets users navigate from one page to another without page reload
- it updates the browser's history so the user can use the *back* and *forward* buttons when navigating back and forth between pages.
-

In addition, Angular Router allows us to:

- redirect a URL to another URL
- resolve data before a page is displayed
- run scripts when a page is activated or deactivated
- lazy load parts of our application.

we'll learn how to set up and configure Angular Router, how to redirect a URL and how to use Angular Router to resolve todos from our back-end API.

Routing configuration

The whole routing functionality is possible with help of a module called Router module , we need to import it

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

Create a variable which is an array of objects which defines paths and their associated components

```
const routes=[
  {path:"",component: HomeComponent},
  {path:"offices",component: OfficesComponent},
  {path:"newcourses",component: NewcoursesComponent},
  {path:"offers",component: OffersComponent},
  {path:"vclass",component: VirtualclassComponent},
  {path:"reviews",component: ReviewsComponent},
  {path:"cdetails/:ccode",component:CoursedetailsComponent},
  {path:"schedules",component:ScheduleComponent,outlet:"sidebar"}
]
```

In the above configuration except the path called schedules all other paths , the component will be loaded in default router-outlet component , but for schedules it will load the component in a named outlet called sidebar and also you can see the path cdetails has a parameter called ccode, we read it by using activatedRoute object.


```
<nav class="navbar navbar-expand-sm bg-green navbar-dark">
  <!-- Brand/logo -->
  <a class="navbar-brand" href="#">XYZ Courses</a>

  <!-- Links -->
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" routerLink="/">{{"VARS.Home" | translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/offices">{{"VARS.Offices"| translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/newcourses">{{"VARS.Newcourses"| translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/vclass">{{"VARS.Virtualclass" | translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/reviews">{{"VARS.Reviews" | translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/offers">{{"VARS.Offers"| translate}}</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" [routerLink]="[{ outlets: { sidebar: ['schedules'] } }]">
        Schedules</a>
  </li>
  </ul>
```

We place the router-outlet in a place which we want

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

Every time you change the route, the component gets loaded in the router-outler component

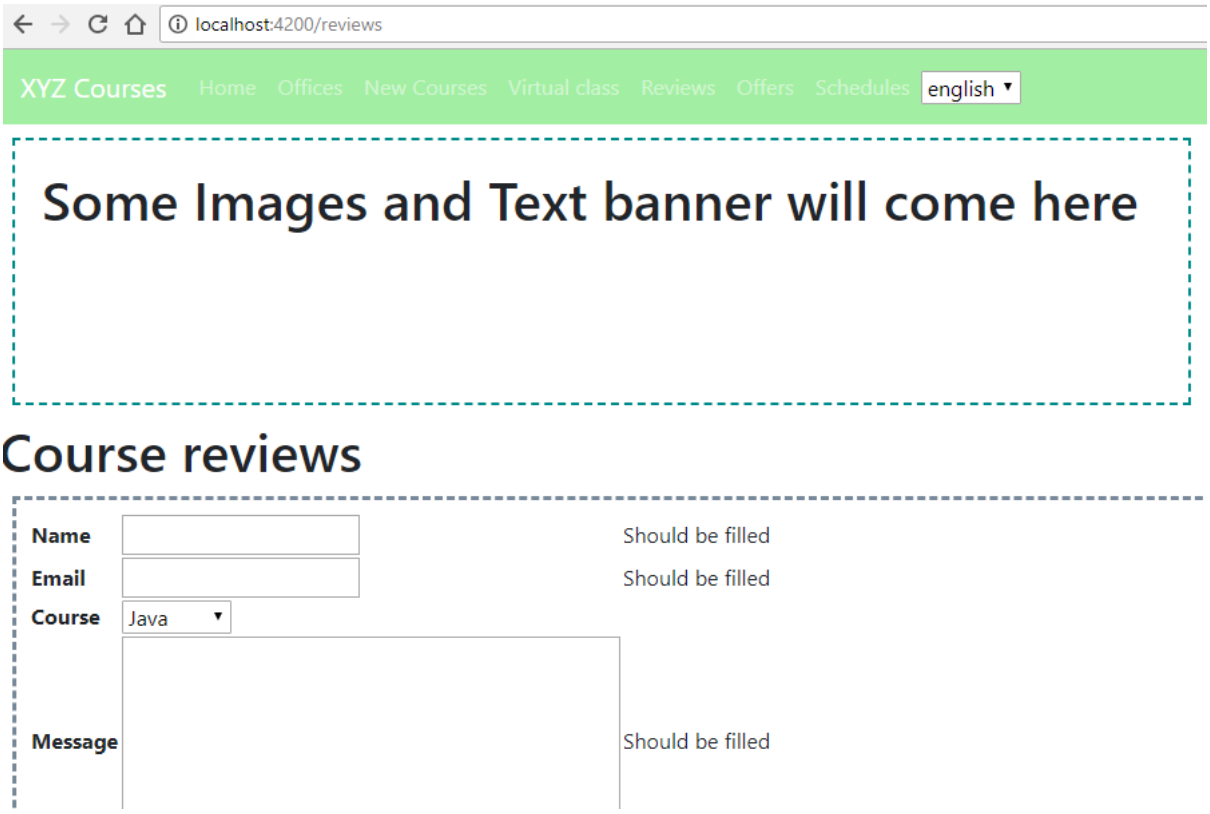
And the named outlet is configured as follows

```
<router-outlet name="sidebar"></router-outlet>
<enquiryform></enquiryform>
```

To handle the parameters we get the parameter the following way

```
import { Component, OnInit } from '@angular/core';
import {ActivatedRoute} from '@angular/router'

@Component({
  selector: 'app-coursedetails',
  templateUrl: './coursedetails.component.html',
  styleUrls: ['./coursedetails.component.css']
})
export class CoursedetailsComponent implements OnInit {
courseCode:string;

  constructor(private route:ActivatedRoute) {
    route.params.subscribe(params=>
      this.courseCode=params['ccode']);

  }

  ngOnInit() {
  }

}
```

And the routing feature works as follows


localhost:4200/reviews
XYZ Courses
Home
Offices
New Courses
Virtual class
Reviews
Offers
Schedules
english
Some Images and Text banner will come here
Course reviews
Name
Should be filled
Email
Should be filled
Course
Java
Message
Should be filled