Routing

Mohamed Mukthar Ahmed

Routing

Mohamed Mukthar Ahmed

What is routing?

- In web development, **routing** means splitting the application into different areas usually based on **rules** that are derived from the current **URL** in the **browser**
- For instance
  - If we visit the **/** path of a website, we may be visiting the **home** route
  - If we visit **/about** we want to render the "**about page**", and so on

Mohamed Mukthar Ahmed

186

Why Do We Need Routing?

- Defining routes in our application is **useful** because we can:
  - separate different areas of the app;
  - maintain the state in the app;
  - protect areas of the app based on certain rules;
- Routing lets us define a **URL string** that specifies where within our app a user should be.
- The initial root URL could be represented by http://our-app/
- When the above URL is sent, we could be redirected to our "**home**" route at http://our-app/home
- When we sent the URL http://our-app/about we could access the "**about**" page.

Mohamed Mukthar Ahmed

187

How server-side routing works?

- With **server-side routing**, the **HTTP** request comes in and the **server** will render a different controller depending on the incoming URL
- For instance, with Express.js you might write something like this:

```
var express = require('express');
var router = express.Router();
// define the about route
router.get('/about', function(req, res) {
    res.send('About us');
});
```

- The server that accepts a **request** and **routes** to a controller and the controller runs a specific action, depending on the ***path*** and ***parameters***

Mohamed Mukthar Ahmed

How client-side routing works?

- **Client-side routing**, is very similar in **concept** but different in **implementation**
- With client-side routing we're **not** necessarily making a request to the server on every URL change
- **Angular** apps, we refer to them as "**Single Page Apps**" (**SPA**)
- In **SPA**, the server gives us a single-page and it's our **JavaScript** that renders different pages.

Mohamed Mukthar Ahmed

Clever Hack

- **Client-side routing**, started out with a **clever hack**
- Instead of using a normal server-side URL for a page; in our **SPA**, we use the anchor tag as the client-side URL
- Recall in HTML

```
<!--  lots of page content here  -->
<a name="about"><h1>About</h1></a>
```

- If the URL http://something/#about, the browser would jump straight to that H1 tag that identified by the **about** anchor

Mohamed Mukthar Ahmed

Clever Hack

- The clever move for client-side frameworks used for **SPAs** was to take the anchor tags and use them to represent the routes within the app by formatting them as paths.
- This is what is known as **hash-based routing**

```
<!--  lots of page content here  -->
<a name="about"><h1>About</h1></a>
```

- If the URL http://something/#about, the browser would jump straight to that H1 tag that identified by the **about** anchor

Mohamed Mukthar Ahmed

Writing our first routes

- In **Angular** we configure **routes**
  - by mapping **paths** to the **component** that will handle them.
- Let's create a small app that has multiple routes.
- On this sample application we will have 3 routes:
  - A **home** page route, using the ***/#/home*** path;
  - An **about** page, using the ***/#/about*** path;
  - A **contact** us page, using the ***/#/contact*** path;
- And when the user visits the root path (***/#/***), it will redirect to the **home** path

Mohamed Mukthar Ahmed

192

Components of Angular routing

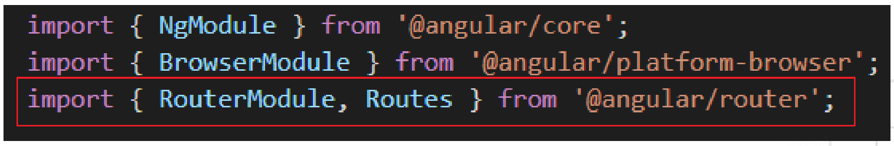
- THREE main components that we use to configure routing
  - **Routes** describes the routes our application supports
  - **RouterOutlet** is a "placeholder" component that shows Angular where to put the content of each route
  - **RouterLink** directive is used to link to routes

Mohamed Mukthar Ahmed

193

Imports

- In order to use the router in **Angular**, we import constants from the **@angular/router** package

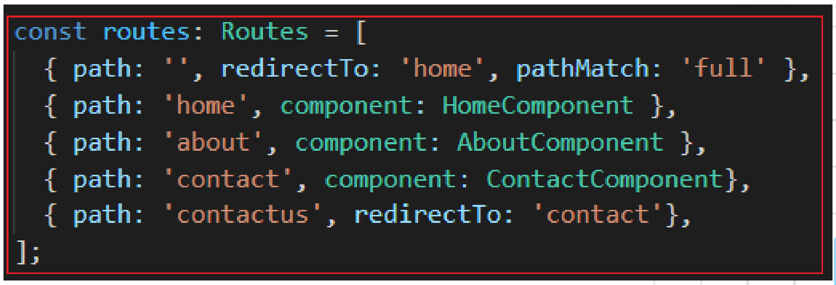


```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';
```

- Now we can define our router configuration.

Routes

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



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

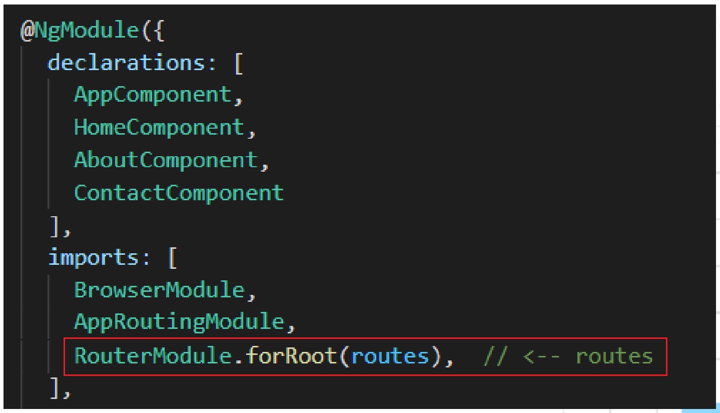
Routes

- Notice a few things about the routes
  - **path** specifies the **URL** this route will handle
  - **component** is what **ties** a given ***route path*** to a component that will handle the route
  - the optional **redirectTo** is used to redirect a given path to an existing route

Mohamed Mukthar Ahmed

Routes

- To define routes for our application
  - then use **RouterModule.forRoot(routes)** to provide our application with the dependencies necessary to use the router.



```
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    RouterModule.forRoot(routes),  // <-- routes
  ],
```

Mohamed Mukthar Ahmed

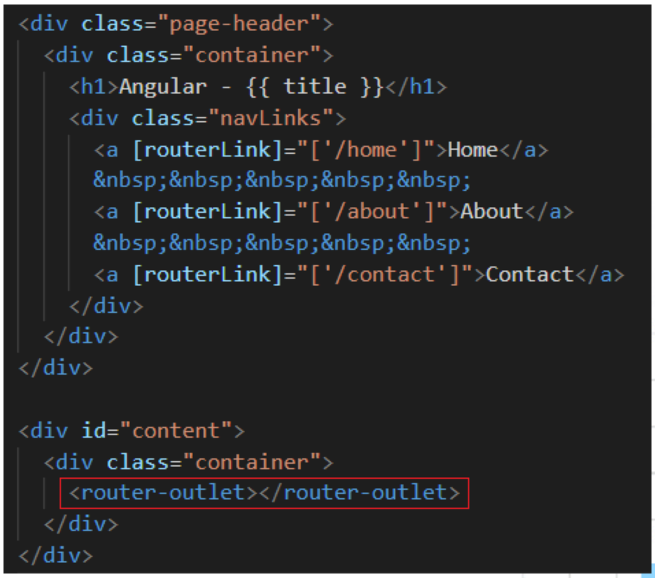
RouterOutlet using <router-outlet>

- When we change routes, we want to keep our outer “layout” template and only substitute the “inner section” of the page with the route’s component
- The **router-outlet** element indicates where the contents of each route component will be rendered.
- Let’s look into the application template for the navigation wrapper of our app.

Mohamed Mukthar Ahmed

198

Navigation Template



```
<div class="page-header">
  <div class="container">
    <h1>Angular - {{ title }}</h1>
    <div class="navLinks">
      <a [routerLink]="['/home']">Home</a>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
      <a [routerLink]="['/about']">About</a>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
      <a [routerLink]="['/contact']">Contact</a>
    </div>
  </div>
</div>

<div id="content">
  <div class="container">
    <router-outlet></router-outlet>
  </div>
</div>
```

Mohamed Mukthar Ahmed

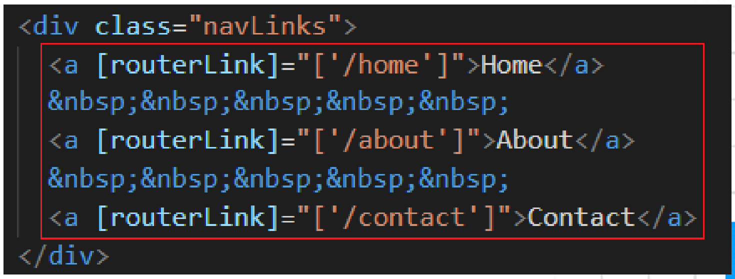
199

Navigation Template

- If we look at the navigation template, you will note the **router-outlet** element right below the navigation menu.
- When we visit **/home**, that's where **HomeComponent** template will be rendered.
- The same happens for the other components.

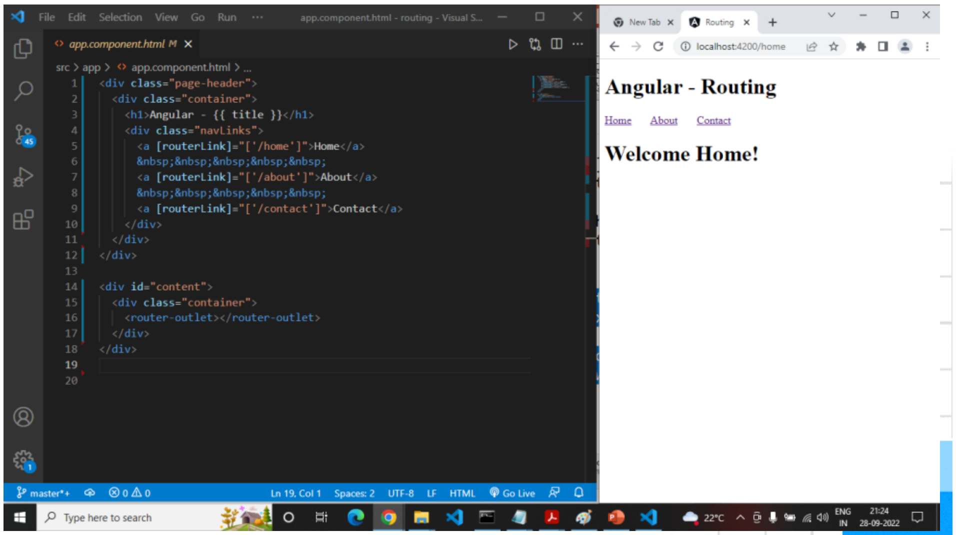
RouterLink using [routerLink]

- Now that we know where route templates will be rendered, how do we tell Angular to navigate to a given route?
- Angular provides the **routerLink** directive to link routes to components.
- The **routerLink** uses a **special syntax** - **Remember**



```
<div class="navLinks">
  <a [routerLink]="['/home']">Home</a>
  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
  <a [routerLink]="['/about']">About</a>
  &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
  <a [routerLink]="['/contact']">Contact</a>
</div>
```

Running The App



```
<div class="page-header">
  <div class="container">
    <h1>Angular - {{ title }}</h1>
    <div class="navLinks">
      <a [routerLink]="['/home']">Home</a>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
      <a [routerLink]="['/about']">About</a>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
      <a [routerLink]="['/contact']">Contact</a>
    </div>
  </div>
</div>

<div id="content">
  <div class="container">
    <router-outlet></router-outlet>
  </div>
</div>
```

localhost:4200/home

Angular - Routing

Home About Contact

Welcome Home!

Mohamed Mukthar Ahmed

202

Route Parameters

- In our apps we often want to navigate to a specific resource.
- For instance, say we had a news website and we had many articles. Each article may have an ID, and if we had an article with ID 3 then we might navigate to that article by visiting th
- To add a **paramete** guration, we specify the route path like this:

How can we retrieve the parameter for a given route?

```
const routes: Routes = [
  { path: 'product/:id',
    component: ProductComponent },
];
```

Mohamed Mukthar Ahmed

203

ActivatedRoute

- In order to use route parameters, we need to first import **ActivatedRoute**

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

- Next, we inject the **ActivatedRoute** into the constructor of our component.

```
const routes: Routes = [
 { path: 'product/:id',
             component: ProductComponent }
];
```

Mohamed Mukthar Ahmed

204

ActivatedRoute

- Then when we write the **ProductComponent**, we add the **ActivatedRoute** as one of the constructor arguments:

```
export class ProductComponent {
 id: string;

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

Mohamed Mukthar Ahmed

205

Wildcard Route

- To setup a **404 page** follow these steps:
  - Create a **PageNotFoundComponent**
  - Design its template.
  - Then in the routes include the **WILDCARD** route for 404 request

```
const routes: Routes = [
 { path: 'home', component: HomeComponent },
 { path: 'about', component: AboutComponent },
 { path: 'contact', component: ContactComponent },
 //Wild Card Route for 404 request
 { path: '**', pathMatch: 'full',
      component: PageNotFoundComponent },
]
```

Mohamed Mukthar Ahmed

206

Q&A

Mohamed Mukthar Ahmed

207