Angular 2 Routing Part 1

Sang Shin
JPassion.com
"Code with Passion!"

Topics

- What is a Router?
- Handle “user enters a URL into the browser”
- Handle “user clicks a link in a page”
- Handle “user actions on a page”
- Redirection
- Parameters
- Query parameters

What is Router?

What is and Why Routing?

- In SPA (Single Page Application), everything happens in one page
  - One layout page with partial fragments filling up the content area
- Issues that need to be addressed in SPA application
  - How do URLs handled?
  - How should “back” button work?
  - How should linking between pages work?
- Angular routing comes to rescue
  - It helps you in dividing your application into logical views based on URL patterns - in other words, it simulates the navigation behavior of multi-page application
  - It also let you bind different views to different components

Navigation Scenarios

- Router handles navigation from one view to the next
- Navigation is triggered by following user activities – your application should address all these scenarios
  1. User enters a URL directly entered into a browser
  2. User clicks a link on a page
  3. User takes an action on a page - clicks a button, selects from a drop box, etc (Navigation via code)
  4. User selects back or forward custom buttons
  5. User selects browser back or forward buttons (handled by Angular)

1. Handle “User enters a URL Directly into a Browser”

Route Definitions

- A router must be configured with a list of route definitions
  - Create route configuration file (name it “app.routes.ts”) that contains route definitions

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

export const routes: Routes  = [
  {path: 'heroes', component: HeroListComponent},
  {path: 'heroes/:id', component: HeroDetailComponent},
  {path: 'dashboard', component: DashBoardComponent},
  {path: '', redirectTo: 'dashboard', pathMatch: 'full'}
];
```

How Route Definitions are used?

- The router uses these route definitions when the browser URL changes
  - When the browser's location URL changes to match the path in the configuration, an instance of the component is created and its view gets displayed
- The router also uses these route definitions when the code tells the router to navigate along a route path
  - When the application requests navigation to the path, same thing occurs. In addition, browser's address location and history will be updated

Configure Router Module

- Calling *forRoot(routes)* method of the *RouterModule* creates a module with all the router providers and directives

```
import { routes } from './app.routes'

RouterModule.forRoot(routes)
```

Note: Somehow importing of routes in Visual Studio Code kept importing RouterModule instead. Please manually import routes as shown above

Add <router-outlet>

- The router displays views within the bounds of the <router-outlet> tags

```
<h1>Root page</h1>
<router-outlet></router-outlet>
```

Lab: Create simple routing

- Create HomeComponent and UserListComponent
- Create routing configuration file that contains routing definitions

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

- Configure routing module (app.module.ts)

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

- Add <router-outlet> to the root page (app.component.html)
- From the browser go to
  - > http://localhost:4200/home
  - > http://localhost:4200/userlist

2. Handle “User clicks a link”

Create Links with [routerLink]

- When a link is selected, component's view gets displayed in the <router-outlet>

```
<h1>
  Root page - this is always displayed
</h1>
<h4>The following is where component displays its view</h4>
<nav>
   <a [routerLink]="['']">Default</a>
   <a [routerLink]="['home']">Home</a>
   <a [routerLink]="['userlist']">User List</a>
</nav>
<hr>
<router-outlet></router-outlet>
```

Create Links

- Path segment can be absolute or relative path
- You could also use ..
- Add the following links in the template of UserListComponent

```
<h1>User list component</h1>
<a [routerLink]="['../home']">Home</a>
<a [routerLink]="['/userlist']">User list using absolute path</a>
<a [routerLink]="['userlist']">User list using relative path – will not work </a>
```

3. Handle “User action on a page” (Navigation via Code)

3. Navigation via Code

- Navigation triggered in the template

```
template: `
  <h1>User list component</h1>
  <button (click)="navigateToHome()">Go to home</button>
`
```

- Navigation via code

```
  constructor(private router: Router) { }

  navigateToHome(){
      this.router.navigate(['/home']);
  }
```

Redirection

Redirection

- Redirection can be specified in the route configuration

```
export const routes: Routes = [
    // {path: '', component: HomeComponent},
    {path: 'home', component: HomeComponent},
    {path: 'userlist', component: UserListComponent},
    { path:' ', redirectTo:"home", pathMatch: "full"},
    { path: '**', redirectTo: 'home' }
]
```

Parameters

Configure routes with parameters

- Parameters can be specified with :<parameter-name>

```
export const routes: Routes = [
    {path:'home', component: HomeComponent},
    {path:'userlist', component: UserListComponent},
    {path:'user/:id', component: UserDetailComponent},
    {path:' ', redirectTo:"home", pathMatch: "full"},
    {path:'**', redirectTo:"home"}
]
```

Configure router links with parameters

- Parameter can be specified as path segment

```
<nav>
  <a [routerLink]="['']">Default</a>
  <a [routerLink]="['/home']">Home</a>
  <a [routerLink]="['/userlist']">User list</a>
  <br>
  <input type="text" (input)="onInput(id.value)" #id>
  <a [routerLink]="['/user', id.value]">User Detail</a>
</nav>
<router-outlet></router-outlet>
```

How to access parameters – option #1

- You can access parameters from a snapshot (one-time only)

```
export class UserDetailComponent {

  private id: any;

  constructor(private activatedRoute: ActivatedRoute) {
      this.id = activatedRoute.snapshot.params['id'];
  }

}
```

How to fetch parameters – option #2

- Or you can subscribe it

```
export class UserDetailComponent {

  private id: any;

  constructor(private activatedRoute: ActivatedRoute) {
      activatedRoute.params.subscribe(
                (params:any) => this.id = params['id']
      )
  }

}
```

How to fetch parameters – option #2

- When you subscribe, it is usually a good idea to unsubscribe it as a measure for preventing memory leak

```
export class UserDetailComponent implements OnDestroy {

  private id: any;
  private subscription: Subscription;

  constructor(private activatedRoute: ActivatedRoute) {
    this.subscription = activatedRoute.params.subscribe(
      (params:any) => this.id = params['id']
    )
  }

  ngOnDestroy(){
    this.subscription.unsubscribe();
  }
}
```

Query Parameters

Configure router links with query params

- Query parameters can be specified with [queryParams]

```
<nav>
  <a [routerLink]="['']">Default</a>
  <a [routerLink]="['/home']">Home</a>
  <a [routerLink]="['/userlist']"
      [queryParams]="{country:'Korea', city:'Seoul'}">User list</a>
  <br>
  <input type="text" (input)="onInput(id.value)" #id>
  <a [routerLink]="['/user', id.value]">User Detail</a>
</nav>
<router-outlet></router-outlet>
```

How to access query parameters

- You can subscribe it

```
export class UserListComponent {
  country: string;
  city: string;
  constructor(private activatedRoute: ActivatedRoute) {

    activatedRoute.queryParams.subscribe(
              (queryParams: any) => {
                                  this.country = queryParams['country'];
                                  this.city = queryParams['city'];
              }
    )
  }

}
```

How to set the query parameters

- Use JavaScript object

```
export class UserListComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit() {
  }

  goHome(){
   this.router.navigate(['/home'],
                        {queryParams: {'country':'france', 'city':'paris'}});
  }

}
```

Code with Passion!
JPassion.com