Angular Pass Data to Route: Dynamic/Static

14 Comments / 6 minutes of reading / February 1, 2024

← Navigate Between Routes | Angular Tutorial | RouterLinkActive →

Angular allows us to pass data through the route. The route data can be either static or dynamic. The static data use the **Angular route data property**, where you can store arbitrary data associated with that specific route. To pass dynamic data (or an object), we can use the history state object. The Routed Component can then retrieve the dynamic data from the history state object.

Table of Contents

Various ways of passing route data

Angular components can share data in many ways. The parent can communicate with their child using @Input directive. A child can pass data to the Parent using the @Output & EventEmitter. The parent can use the @ViewChild to access the child component. In case of components are unrelated then we can use the Angular Services to Share data between them.

We can also share data between components using the route. Angular can pass **data through the route** in several ways.

1. Using Route Parameter
2. The Query Parameters or Query Strings
3. Using URL Fragment
4. Static data using the data property
5. Dynamic data using the state object

In this article, we look at how to pass static or dynamic data using the Route. i.e. items 4 & 5 in the above list.

Passing static data to a route

We can configure the static data at the time of defining the route. This is done by using the **Angular route data property** of the route. The **route data property** can contain an array of arbitrary string key-value pairs. You can use the static data to store items such as page titles, breadcrumb text, and other read-only, static data

For Example, consider the following route with the data property set

```
1
2 { path: 'static', component: StaticComponent, data :{ id:'1', name:"Angular"}},
3
```

The Angular Router will pass the `{ id:'1', name:"Angular"}` when the `StaticComponent` is rendered. The data value will be located in the data property of the `ActivatedRoute` service

We can then read the data by subscribing to the `activatedroute.data` property as shown below

```
ngOnInit() {
    this.activatedroute.data.subscribe(data => {
       this.product=data;
    })
}
```

Passing Dynamic data to a Route

The option to pass the dynamic data or a user-defined object was added in **Angular Version 7.2** using the state object. The state object is stored in History API

Providing the State value

The state can be provided in two ways

Using routerLink directive

```
<a [routerLink]="['dynamic']" [state]="{ id:1 , name:'Angular'}">Dynamic Data</a>
```

Using navigateByUrl

```
this.router.navigateByUrl('/dynamic', { state: { id:1 , name:'Angular' } });
```

NavigationId

navigationId is a number, which is incremented every time we navigate from one route to another. Angular uses it to identify every navigation. The Router will add a **navigationId** property to the state object. Because of that, we can only assign an object to the State object.

Hence we cannot store primitive types like strings or numbers etc. For example, the following code results in an error because we are passing a string.

```
this.router.navigateByUrl('/dynamic', { state:  'Angular' });
```

You can only assign an object to the state object. The following code is ok.

```
this.router.navigateByUrl('/dynamic', { state: {name: 'Angular' } });
```

Accessing the state value

The state can be accessed by using the `getCurrentNavigation` method of the router (**works only in the constructor**)

```
this.router.getCurrentNavigation().extras.state
```

Or use the `history.state` in the ngOnInit.

```
console.log(history.state)
```

or use the `getState` method of the Location Service. This method is available in Angular 8+

```
import { Location } from '@angular/common';

export class SomeComponent
{
  products:Product[];

  constructor(private location:Location){
  }

  ngOnInit() {
    console.log(this.location.getState());
  }
}
```

Note that you will lose the dynamic data if the page is refreshed.

Passing Data to the Routes Example

Let us build a simple project to demonstrate how to pass data to the route

Passing static data example

static.component.ts

```
1
2  import {Component, OnInit} from '@angular/core';
3  import { ActivatedRoute } from '@angular/router';
4
5  @Component({
6      template: `<h1>Passing Static Data Demo</h1>
7         {{product  | json}}`
8  })
9  export class StaticComponent implements OnInit {
10
11     product:any;
12     constructor(private activatedroute:ActivatedRoute) {
13     }
14
15     ngOnInit() {
16         this.activatedroute.data.subscribe(data => {
17             this.product=data;
18          })
19     }
20 }
21
```

Source Code

The static component gets the static data configured in the route. It subscribes the activatedroute.data property to get the product data as shown above.

Passing dynamic data (or object) example

dynamic.component.ts

```
1
2  import {Component, OnInit, ChangeDetectorRef} from '@angular/core';
3  import { ActivatedRoute, Router, NavigationStart } from '@angular/router';
4  import { map, filter} from 'rxjs/operators';
5  import { Observable} from 'rxjs/observable';
6
7  @Component({
8      template: `<H1>Passing Dynamic Data Demo</H1>
9
10     {{ product | json }}`
11 })
12 export class DynamicComponent implements OnInit {
13
14     product;
15
16     constructor(private router:Router, private activatedRoute:ActivatedRoute) {
17         console.log(this.router.getCurrentNavigation().extras.state);
18     }
19
20     ngOnInit() {
21         //console.log(history.state);
22         this.product=history.state;
23     }
24
25 }
26
```

Source Code

The Dynamic Component gets dynamic data. We use the `history.state` to access the product data. Alternatively, we can use the `this.router.getCurrentNavigation().extras.state` to achieve the same. **Please remember** `getCurrentNavigation` **only works in the constructor**. It will return null if used elsewhere.

home.component.ts

```
1
2  import { Component } from '@angular/core';
3  import { ActivatedRoute, Router } from '@angular/router';
4
5  @Component({
6   template: `
7      <ul>
8        <li><a [routerLink]="['/static']">Static Data</a></li>
9        <li><a [routerLink]="['/dynamic']" [state]=product>Dynamic Data</a></li>
10     </ul>
11
12     <p>Id :   <input type="text" [(ngModel)]="product.id" > </p>
13     <p>name :<input type="text" [(ngModel)]="product.name" > </p>
14     <button (click)="gotoDynamic()" >Goto Dynamic Component</button>`
15 })
16 export class HomeComponent {
17
18   public product = { id:'1', name:"Angular"};
19
20   constructor(private router : Router) {
21   }
22
23   gotoDynamic() {
24     //this.router.navigateByUrl('/dynamic', { state: { id:1 , name:'Angular' } });
25     this.router.navigateByUrl('/dynamic', { state: this.product });
26   }
27 }
28
```

Source Code

In HomeComponent , we have used routerLink & navigateByUrl to pass the data to the dynamic component. You can also use the form fields to change the data, before passing it to the dynamic route.

app.routes.ts

```
1
2  import { Routes } from '@angular/router';
3
4  import { StaticComponent} from './static.component'
5  import { DynamicComponent } from './dynamic.component';
6  import { HomeComponent } from './home.component';
7
```

```
export const appRoutes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'static', component: StaticComponent, data :{ id:'1', name:"Angular"}},
  { path: 'dynamic', component: DynamicComponent },
  { path: '', redirectTo: 'home', pathMatch: 'full' }
];


```

Source Code

Here the static data is set for StaticComponent using the data property.

app.component.ts

```

import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<div class="container">

  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <div class="navbar-header">
        <a class="navbar-brand" [routerLink]="['/']"><strong> {{title}} </strong></a>
      </div>
    </div>
  </nav>

  <router-outlet></router-outlet>

  </div>`
})
```

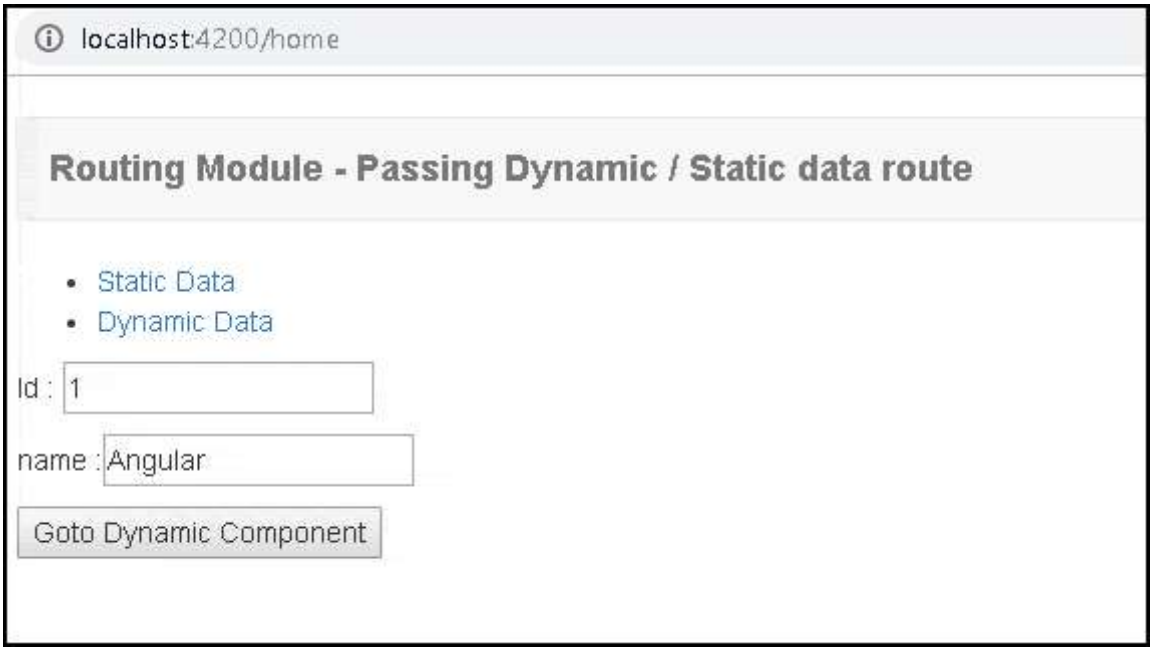
```
export class AppComponent {
  title = 'Routing Module - Passing Dynamic / Static data route';
}

```

Source Code

app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

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

import { AppComponent } from './app.component';
import { StaticComponent} from './static.component'


import { appRoutes } from './app.routes';
import { DynamicComponent } from './dynamic.component';
import { HomeComponent } from './home.component';

@NgModule({
  declarations: [
    AppComponent,StaticComponent,DynamicComponent,HomeComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    RouterModule.forRoot(appRoutes)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Source Code



References

History API

Read More

1. Angular Tutorial
2. Angular Routing
3. Passing Parameters to Route
4. Child Routes/Nested Routes
5. Passing Query Parameters
6. Navigation Between Routes

← Navigate Between Routes | Angular Tutorial | RouterLinkActive →

Related Posts