**1. Define Routes with Parameters**

* In your app-routing.module.ts:

TypeScript

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

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

import { UserDetailComponent } from './user-detail/user-detail.component';

const routes: Routes = [

{ path: 'user/:id', component: UserDetailComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

* path: 'user/:id' defines a route with a parameter named id.

**2. Access Route Parameters in the Component**

* In your user-detail.component.ts:

TypeScript

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

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

@Component({

selector: 'app-user-detail',

template: `

<h2>User Details for ID: {{ userId }}</h2>

`

})

export class UserDetailComponent implements OnInit {

userId: number;

constructor(private route: ActivatedRoute) { }

ngOnInit() {

this.route.paramMap.subscribe(params => {

this.userId = +params.get('id');

});

}

}

* ActivatedRoute: This service provides access to information about the current route, including parameters.
* paramMap: An Observable that emits the route parameters as a ParamMap object.
* params.get('id'): Retrieves the value of the id parameter.
* +params.get('id'): Converts the parameter value to a number.

**3. Navigate to the Route with Parameters**

* In your app.component.html or any other component:

HTML

<a [routerLink]="['/user', 123]">View User 123</a>

* [routerLink]="['/user', 123]" creates a link to the /user/123 URL.

**Key Points:**

* Route parameters are dynamic parts of the URL that can be used to pass data between components.
* Use the ActivatedRoute service to access route parameters within your components.
* Route parameters can be used to fetch data from a server, display specific items, and more.

This example demonstrates how to use parameterized routing in Angular to create dynamic and data-driven applications. You can extend this concept to handle multiple parameters, optional parameters, and more complex routing scenarios.

Example:

Certainly, let's explore a more practical example of parameterized routing with API calls, implementing a master-detail view.

**1. Define Routes**

TypeScript

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

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

import { UsersComponent } from './users/users.component';

import { UserDetailComponent } from './user-detail/user-detail.component';

const routes: Routes = [

{ path: '', redirectTo: '/users', pathMatch: 'full' },

{ path: 'users', component: UsersComponent },

{ path: 'users/:id', component: UserDetailComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

* path: '': Redirects the root URL (/) to the /users route.
* path: 'users': Displays a list of users.
* path: 'users/:id': Displays details for a specific user, where :id is the user's ID.

**2. UsersComponent (users.component.ts)**

TypeScript

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

import { UserService } from './user.service'; // Import your user service

@Component({

selector: 'app-users',

template: `

<ul>

<li \*ngFor="let user of users">

<a [routerLink]="['/users', user.id]">{{ user.name }}</a>

</li>

</ul>

`

})

export class UsersComponent {

users: any[] = [];

constructor(private userService: UserService) {}

ngOnInit() {

this.userService.getUsers().subscribe(users => {

this.users = users;

});

}

}

**3. UserDetailComponent (user-detail.component.ts)**

TypeScript

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

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

import { UserService } from './user.service';

@Component({

selector: 'app-user-detail',

template: `

<h2>User Details</h2>

<p>ID: {{ userId }}</p>

<p>Name: {{ user?.name }}</p>

<p>Email: {{ user?.email }}</p>

`

})

export class UserDetailComponent implements OnInit {

userId: number;

user: any;

constructor(private route: ActivatedRoute, private userService: UserService) {}

ngOnInit() {

this.route.paramMap.subscribe(params => {

this.userId = +params.get('id');

this.userService.getUserById(this.userId).subscribe(user => {

this.user = user;

});

});

}

}

**4. UserService (user.service.ts)**

TypeScript

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

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })

export class UserService {

constructor(private http: HttpClient) {}

getUsers(): Observable<any[]> {

return this.http.get<any[]>('/api/users');

}

getUserById(id: number): Observable<any> {

return this.http.get<any>(`/api/users/${id}`);

}

}

**Explanation:**

* **UsersComponent:**
  + Fetches a list of users from the UserService.
  + Displays a list of user names with links to their detail pages.
* **UserDetailComponent:**
  + Retrieves the userId from the route parameters.
  + Uses the UserService to fetch the user details based on the ID.
  + Displays the user information.
* **UserService:**
  + Provides methods to fetch a list of users and a single user by ID.

**Key Improvements:**

* **Master-Detail View:** Demonstrates a common use case for parameterized routing.
* **API Calls:** Includes example API calls using HttpClient.
* **Data Flow:** Shows how to fetch and display data based on route parameters.

This example provides a more comprehensive understanding of how to use parameterized routing with API calls in Angular to build dynamic and data-driven applications. Remember to adjust the API endpoints and data structures according to your specific needs.