**1. Define Routes**

TypeScript

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

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

import { ProductListComponent } from './product-list/product-list.component';

import { ProductDetailComponent } from './product-detail/product-detail.component';

const routes: Routes = [

{

path: 'products',

component: ProductListComponent,

children: [

{ path: ':id', component: ProductDetailComponent }

]

}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

* **Parent Route:**
  + path: 'products': Defines the parent route for the product section.
  + component: ProductListComponent: Specifies the component to be displayed for the parent route.
  + children: An array to define child routes within the parent route.
* **Child Route:**
  + path: ':id': Defines a child route with a parameter (id) for displaying product details.
  + component: ProductDetailComponent: Specifies the component to be displayed for the child route.

**2. ProductListComponent (product-list.component.html)**

HTML

<ul>

<li \*ngFor="let product of products">

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

</li>

</ul>

<router-outlet></router-outlet>

* **router-outlet:** This directive within the parent component renders the child component (ProductDetailComponent) when a child route is navigated to.1

**3. ProductDetailComponent (product-detail.component.ts)**

TypeScript

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

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

@Component({

selector: 'app-product-detail',

template: `

<h2>Product Details</h2>

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

{/\* Display product details here \*/}

`

})

export class ProductDetailComponent implements OnInit {

productId: number;

constructor(private route: ActivatedRoute) {}

ngOnInit() {

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

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

});

}

}

**Explanation:**

* This example demonstrates a master-detail view where ProductListComponent acts as the master view and ProductDetailComponent as the detail view.
* When a user clicks on a product in the ProductListComponent, the URL changes to /products/{id}, and the ProductDetailComponent is displayed with the corresponding product details.
* The router-outlet within ProductListComponent is crucial for rendering the child component.

**Key Considerations:**

* **URL Structure:** Carefully plan your URL structure to ensure it is user-friendly and reflects the application's navigation hierarchy.
* **Data Flow:** Consider how data will be passed between parent and child components (e.g., using services, route parameters, or data sharing techniques).
* **Testing:** Write unit tests for your components and routing configurations to ensure they function correctly.

This example provides a basic implementation of child routing. You can further enhance it by adding features like lazy loading, guards, and more complex routing scenarios to suit your specific application needs.