**Using Route Parameters**

Say we are creating an application that displays a product list. When the user clicks on a product in the list, we want to display a page showing the detailed information about that product. To do this you must:

* add a route parameter ID
* link the route to the parameter
* add the service that reads the parameter.

**Declaring Route Parameters**

The route for the component that displays the details for a specific product would need a route parameter for the ID of that product. We could implement this using the following Routes:

export const routes: Routes = [

{ path: '', redirectTo: 'product-list', pathMatch: 'full' },

{ path: 'product-list', component: ProductList },

{ path: 'product-details/:id', component: ProductDetails }

];

Note :id in the path of the product-details route, which places the parameter in the path. For example, to see the product details page for product with ID 5, you must use the following URL:localhost:3000/product-details/5

**Linking to Routes with Parameters**

In the ProductList component you could display a list of products. Each product would have a link to the product-details route, passing the ID of the product:

<a \*ngFor="let product of products"

[routerLink]="['/product-details', product.id]">

{{ product.name }}

</a>

Note that the routerLink directive passes an array which specifies the path and the route parameter. Alternatively we could navigate to the route programmatically:

goToProductDetails(id) {

this.router.navigate(['/product-details', id]);

}

**Reading Route Parameters**

The ProductDetails component must read the parameter, then load the product based on the ID given in the parameter.

The ActivatedRoute service provides a params Observable which we can subscribe to to get the route parameters (see [Observables](https://angular-2-training-book.rangle.io/handout/observables/)).

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

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

@Component({

selector: 'product-details',

template: `

<div>

Showing product details for product: {{id}}

</div>

`,

})

export class LoanDetailsPage implements OnInit, OnDestroy {

id: number;

private sub: any;

constructor(private route: ActivatedRoute) {}

ngOnInit() {

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

this.id = +params['id']; // (+) converts string 'id' to a number

// In a real app: dispatch action to load the details here.

});

}

ngOnDestroy() {

this.sub.unsubscribe();

}

}

# Defining Child Routes

When some routes may only be accessible and viewed within other routes it may be appropriate to create them as child routes.

For example: The product details page may have a tabbed navigation section that shows the product overview by default. When the user clicks the "Technical Specs" tab the section shows the specs instead.

If the user clicks on the product with ID 3, we want to show the product details page with the overview:

localhost:3000/product-details/3/overview

When the user clicks "Technical Specs":

localhost:3000/product-details/3/specs

overview and specs are child routes of product-details/:id. They are only reachable within product details.

Our Routes with children would look like:

export const routes: Routes = [

{ path: '', redirectTo: 'product-list', pathMatch: 'full' },

{ path: 'product-list', component: ProductList },

{ path: 'product-details/:id', component: ProductDetails,

children: [

{ path: '', redirectTo: 'overview', pathMatch: 'full' },

{ path: 'overview', component: Overview },

{ path: 'specs', component: Specs }

]

}

];

Where would the components for these child routes be displayed? Just like we had a <router-outlet></router-outlet> for the root application component, we would have a router outlet inside the ProductDetails component. The components corresponding to the child routes of product-detailswould be placed in the router outlet in ProductDetails.

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

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

@Component({

selector: 'product-details',

template: `

<p>Product Details: {{id}}</p>

<!-- Product information -->

<nav>

<a [routerLink]="['overview']">Overview</a>

<a [routerLink]="['specs']">Technical Specs</a>

</nav>

<router-outlet></router-outlet>

<!-- Overview & Specs components get added here by the router -->

`

})

export default class ProductDetails implements OnInit, OnDestroy {

id: number;

constructor(private route: ActivatedRoute) {}

ngOnInit() {

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

this.id = +params['id']; // (+) converts string 'id' to a number

});

}

ngOnDestroy() {

this.sub.unsubscribe();

}

}

Alternatively, we could specify overview route URL simply as:

localhost:3000/product-details/3

export const routes: Routes = [

{ path: '', redirectTo: 'product-list', pathMatch: 'full' },

{ path: 'product-list', component: ProductList },

{ path: 'product-details/:id', component: ProductDetails,

children: [

{ path: '', component: Overview },

{ path: 'specs', component: Specs }

]

}

];

Since the Overview child route of product-details has an empty path, it will be loaded by default. The specs child route remains the same.

[View Example with child routes](https://plnkr.co/edit/MqNv6RyQvzsiZTp0Dkpf?p=preview)

[View Example with route params & child routes](https://plnkr.co/edit/xFL7q0HeTGBPQT1ZiMnI?p=preview)

View examples running in full screen mode to see route changes in the URL.

## Accessing a Parent's Route Parameters

In the above example, say that the child routes of product-details needed the ID of the product to fetch the spec or overview information. The child route component can access the parent route's parameters as follows:

export default class Overview {

parentRouteId: number;

private sub: any;

constructor(private router: Router,

private route: ActivatedRoute) {}

ngOnInit() {

// Get parent ActivatedRoute of this route.

this.sub = this.router.routerState.parent(this.route)

.params.subscribe(params => {

this.parentRouteId = +params["id"];

});

}

ngOnDestroy() {

this.sub.unsubscribe();

}

}

[View Example child routes accessing parent's route parameters](https://plnkr.co/edit/7stoOP3oEl7dqwsgBgu9?p=preview)

View examples running in full screen mode to see route changes in the URL.

## Links

Routes can be prepended with /, or ../; this tells Angular where in the route tree to link to.

| **Prefix** | **Looks in** |
| --- | --- |
| / | Root of the application |
| none | Current component children routes |
| ../ | Current component parent routes |

Example:

<a [routerLink]="['route-one']">Route One</a>

<a [routerLink]="['../route-two']">Route Two</a>

<a [routerLink]="['/route-three']">Route Three</a>

In the above example, the link for route one links to a child of the current route. The link for route two links to a sibling of the current route. The link for route three links to a child of the root component (same as route one link if current route is root component).