



Angular 16 : Online Class

Routing

By: Sahosoft Solutions

Presented by : Chandan Kumar
 **sahosoft**
Online Learning Platform

Router-outlet



Router-outlet in Angular works as a placeholder which is used to load the different components dynamically based on the activated component or current route state. Navigation can be done using router-outlet directive and the activated component will take place inside the router-outlet to load its content.

To enable routing, we need to use router-outlet into our HTML template like this.

```
<router-outlet></router-outlet>
```

Wildcard Route



Wildcard route to intercept invalid URLs and handle them gracefully.

A *wildcard* route has a path consisting of two asterisks (* *).

It matches *every* URL, the router will select *this* route if it can't match a route earlier in the configuration. A wildcard route can navigate to a custom "404 Not Found" component or redirect to an existing route.

```
{ path: '**', component: PageNotFoundComponent }
```

If the entire router configuration is processed and there is no match, router navigation fails and an error is logged.

Note:

If you add a wildcard route as the first route, no other routes would be reached and the wildcard route would always be matched. As a result, you should always add a wildcard route as the last route in your router configuration.

Redirecting Routes



A redirect route that translates the initial relative URL (") to the desired default path (component-one)

When application starts, it navigates to the empty route by default. We can configure the router to redirect to a named route by default:

A router has no routes until you configure it.

```
export const routes: Routes = [  
  { path='', redirectTo: 'component-one', pathMatch: 'full' },  
  path: 'component-one', component: ComponentOne  
  { path: 'component-two', component: ComponentTwo  
  ];
```

Redirecting Routes



- This route redirects a URL that fully matches the empty path to the route whose path is 'component-one'
- **A redirect route requires a pathMatch property to tell the router how to match a URL to the path of a route. The router throws an error if you don't.**
- For the special case of an empty URL we also need to add the pathMatch: 'full' property so Angular knows it should be matching exactly the empty string and not partially the empty string.

Difference between Href and routerLink



Href

it stands for hyper reference , generally used for making link to switch between multiple pages for simple HTML. href is html anchor tag attribute to navigate to another page. Here a new page will be loaded.

RouterLink

RouterLink is used to achieve the same functionality but angular 2 or above are single page applications, where the page should not reload. RouterLink navigates to New Url and the component is rendered in place of routeroutlet without reloading the page.

routerLink - functionality similar to href but this is in the form of angular2, means to say that routerLink is used in angular 2 for routing purpose and behind the scene routerLink is being converted into href for switch between pages (routes in angular's term)

Difference between [routerLink] and routerLink



[routerLink]

When you use brackets, it means you're passing a bindable property (a variable).

```
<a [routerLink]="routerLinkVariable"></a>
```

So this variable (**routerLinkVariable**) could be defined inside your class and it should have a value like below:

```
export class myComponent {  
  public routerLinkVariable = "/home"; // the value of the variable is string!  
}
```

But with variables, you have the opportunity to make it dynamic right.

```
export class myComponent {  
  
  public routerLinkVariable = "/home"; // the value of the variable is string!  
  
  updateRouterLinkVariable(){  
  
    this.routerLinkVariable = '/about';  
  }  
}
```

Difference between [routerLink] and routerLink



[routerLink]

The other specialty about using brackets specifically for routerLink is that you can pass dynamic parameters to the link you're navigating to:

So adding a new variable

```
export class myComponent {  
    private dynamicParameter = '129';  
    public routerLinkVariable = "/home";  
}
```

Updating the [routerLink]

```
<a [routerLink]="[routerLinkVariable,dynamicParameter]"></a>
```

When you want to click on this link, it would become:

```
<a href="/home/129"></a>
```

routerLink

Where as without brackets you're passing string only and you can't change it, it's hard coded and it'll be like that throughout your app.

```
<a routerLink="/home"></a>
```


Child Routes / Nested Routes

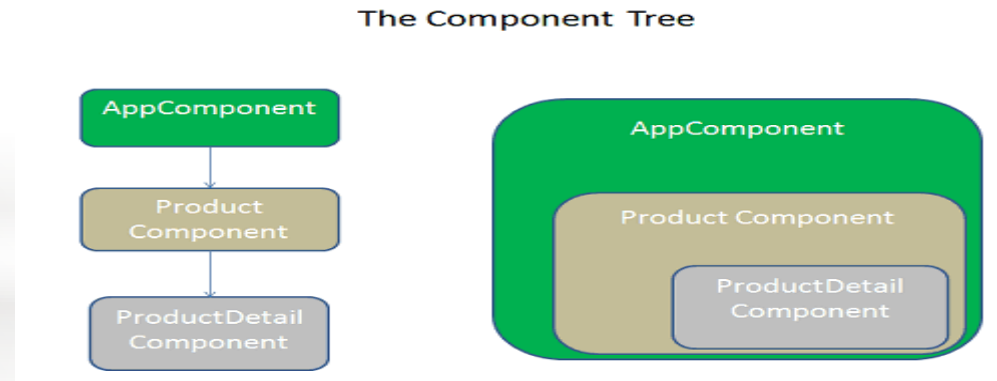


The Angular 2 and above applications are based on the idea of Components. The Components follows a Tree structure, where we have a root component at the top. We can then add child components forming loosely coupled components resembling a Tree.

The Routes in Angular also follows the component tree structure and allows us to define the nested or child routes.

Example

Consider the following Component Tree



Child Routes / Nested Routes



Example

```
const routes: Routes = [  
  { path: '', redirectTo: 'dashboard', pathMatch: 'full' },  
  { path: 'dashboard', component: DashboardComponent },  
  { path: 'about', component: AboutComponent },  
  { path: 'contact', component: ContactComponent },  
  
  {  
    path: 'student',  
    children: [  
      { path: '', component: StudentComponent , pathMatch: 'full' },  
      { path: 'studentdetails', component: StudentdetailsComponent },  
      { path: 'studentregistration', component: StudentregistrationComponent },  
    ],  
  },  
  
  { path: '**', component: PagenotfoundComponent },  
];
```

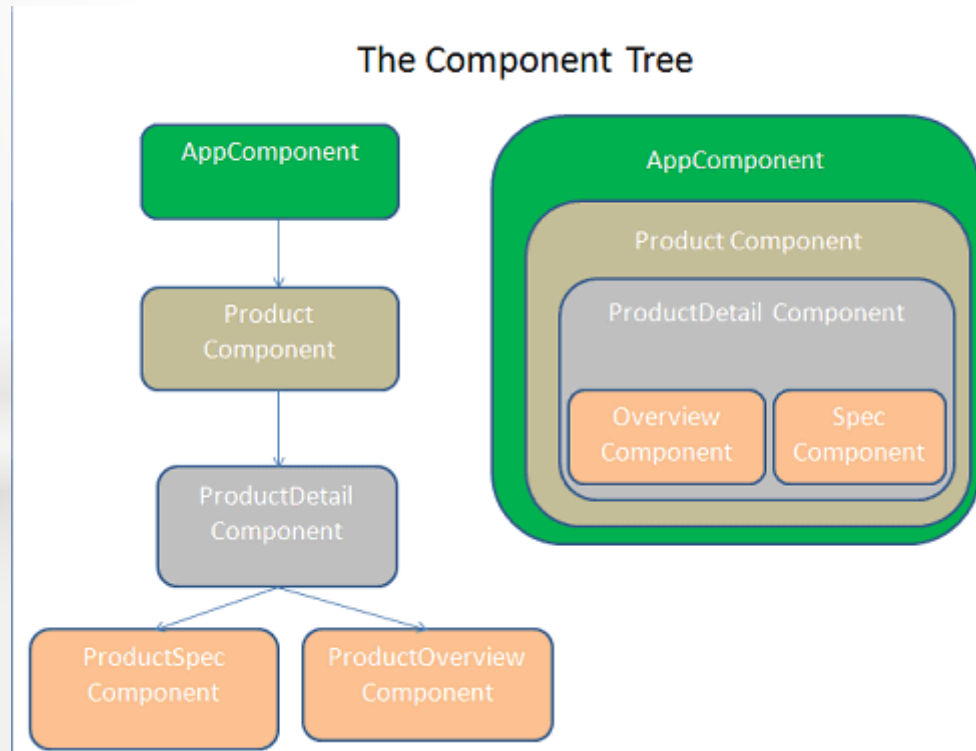


Nested Routes (Nesting Children's under a child)

Nesting Children's under a child

We can add child routes to a child route.

For Example, What if we want to show Product Overview & Specification under the Product Details Page. Our Component Tree is as shown below



Parameterize Route



A route can receive the parameters either using a **snapshot property** of the ActivatedRoute or by **subscribing to its property param**.

The Angular Router provides two different methods to get route parameters:

1. **Using the route snapshot**
2. **Using Router Observables**

Example:

- `route.snapshot.param`
- `route.snapshot.paramMap`

- `route.param.subscribe`
- `route.paramMap.subscribe`

Parameterize Route



The Snapshot Way:

The Snapshot Way: The router provides us with a snapshot of the current route.

The Angular Router provides an **ActivatedRoute** that we can inject into our classes. This will be the main tool we'll use to grab route parameters. We inject the ActivatedRoute into this component via the constructor.

The snapshot approach fails if we are navigating from one component back to the same component. The reason is that Angular needs to figure out whether it needs to initialize the new component or if it can reuse the same component. If we are navigating back to the same component, Angular reuses the same component. This is why our view is not getting updated.

The Observable/Stream Way:

The Observable/Stream Way: By subscribing to its property param. Now there is a property paramMap that allows you to get a particular parameter by using the method get() .

We are getting the updated value even on navigation on the same component. You can see the URL value is getting changed and also making the view updated. Clicking on any record will give you the updated view.

Parameterize Route



1. When to use `route.snapshot.paramMap` ?

If you intend not to update your URL parameter within the same component you are accessing it, then you can use the snapshot.

As the name suggests, the parameter would only be accessed once, when the component loads. Hence, it won't be updated, even if you change its value from within the same component.

```
ngOnInit() {  
  this.obj = this.route.snapshot.paramMap.get("anyValue")  
}
```

2. When to use `route.paramMap.subscribe` ?

If you intend to update the URL parameter within the same component, then you have to use a subscription.

The good news is that it works just like any other subscription in your Angular app.

This technique is only useful if you plan to change the URL parameters within the current route.

```
ngOnInit() {  
  this.route.paramMap.subscribe(params => {  
    this.obj = params.get("anyValue")  
  })  
}
```