breaking thought barriers

Angular Training

Session -24

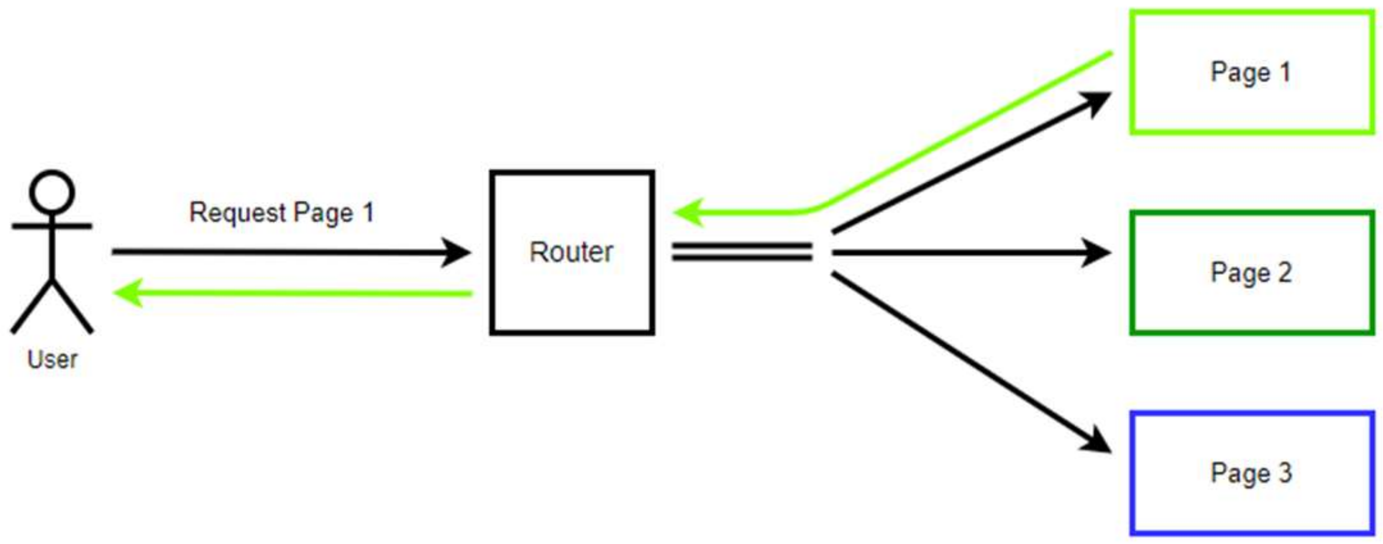
Outlines

Angular Router

What is Angular Routing

- Routing allows you to move from one part of the application to another part or one View to another View.

- In Angular, Routing is handled by the Angular Router Module.



Using Angular Router you can

- Navigate to a specific view by typing a URL in the address bar
- Pass optional parameters (query parameters) to the View
- Bind the clickable elements to the View and load the view when the user performs application tasks
- Handles back and forward buttons of the browser
- Allows you to load the view dynamically
- Protect the routes from unauthorized users using **Route Guards**

Components of Angular Router

Router

- An Angular Router is a service (Angular Router API) that enables navigation from one component to the next component as users perform application tasks like clicking on menus links, and buttons, or clicking on the back/forward button on the browser.

- We can access the router object and use its methods like navigate() or navigateByUrl(), to navigate to a route

Route

- Route tells the Angular Router which view to display when a user clicks a link or pastes a URL into the browser address bar.

- Every Route consists of a path and a component it is mapped to.

- The Router object parses and builds the final URL using the Route

Routes

Routes is an array of Route objects our application supports

RouterOutlet

The outerOutlet is a directive **(<router-outlet>)** that serves as a placeholder, where the Router should display the view

RouterLink

- The RouterLink is a directive that binds the HTML element to a Route.
- Clicking on the HTML element, which is bound to a RouterLink, will result in navigation to the Route.
- The RouterLink may contain parameters to be passed to the route’s component.

RouterLinkActive

- RouterLinkActive is a directive for adding or removing classes from an HTML element that is bound to a RouterLink.
- Using this directive, we can toggle CSS classes for active RouterLinks based on the current RouterState

ActivatedRoute

- The ActivatedRoute is an object that represents the currently activated route associated with the loaded Component.

RouterState

The current state of the router includes a tree of the currently activated routes together with convenience methods for traversing the route tree.

RouteLink Parameters array

- The Parameters or arguments to the Route.
- It is an array that you can bind to RouterLink directive or pass it as an argument to the Router.navigate method.

How to configure Angular Router

To Configure the Router in Angular, you need to follow these steps

1. Set the <base href>
2. Define routes for the view
3. Register the Router Service with Routes
4. Map HTML Element actions to Route
5. Choose where you want to display the view

1. Set the <base href>

- The HTML <base> element specifies the base URL to use for all relative URLs contained within a document.
- The Angular Router uses the HTML5 style of Routing (or PathLocationStrategy) as the default option.
- The router makes use of the browser’s history API for navigation and URL interaction.

```html
<base href="/">
```

Angular supports two Location Strategies:

HashLocationStrategy -> http://localhost:4200/#/product
PathLocationStrategy -> http://localhost:4200/product

2. Define the routes

- create an array of route objects. Each route maps the path (URL Segment) to the component

```
const appRoutes={ path: ‘product ', component: ProductComponent }
```

(This route tells angular to render ProductComponent when the user navigates to the URL “/product”)

path: The URL path segment of the route. We will use this value to refer to this route elsewhere in the app

component: The component to be loaded.

3. Register the Routes

- Import the Angular Router from @angular/router library in the root module of the application

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

- Then, install the routes using the **RouterModule.forRoot** method, passing the routes as the argument in the imports array

```
imports: [RouterModule.forRoot( appRoutes )],
```

4. Map Action to Routes

- we need to bind the click event of the link, image, or button to a route. This is done using the routerlink directive

```
<li><a [routerLink]="['product']">Product</a></li>
```

- The routerLink directive accepts an array of route names along with parameters. This array is called a Link Parameters array.

When the application requests navigation to the route “product”, the router looks in the routes array and activates the instance of the component associated with the route “product”, which is ProductComponent. The browser address location & history is also updated to /product

5. Choose where you want to display

Finally, we need to tell the angular where to display the view. This is done using the RouterOutlet directive as shown.

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

1. Create Separate Module for Routing

2. Import Router Module ,Routes In your AppRouting Module
   import { Routes,RouterModules} from '@angular.router'
3. Create a Routes Config

```
 const appRoutes :Routes =[
    {path:'home',component :HomeComponent},
    {path:'user-list',component:UserListComponent},
    {path:' ',redirectTo:'/home',pathMatch:'full'}
];
```

4. Call RouterModule.forRoot() and give it the Routes config

5. Export this module into you **RootModule**

```
@NgModule({
      imports :[RouterModule.forRoot(appRoutes)] ,
      exports :[RouterModule]
})
```

6. Place a **<route-outlet></router-outlet>** tag in your template where you to perform it.

7. Place links that will take your user to those Routes and use **routerLink** attribute to give them links.

Angular Router : Example

Angular Route Params

Angular Route Params

- There are many scenarios, where you need to pass parameters to the route.
- For example, to navigate to the product detail view, we need to pass the id of the Product, so that component can retrieve it and display it to the user. This is where we use the Route Parameters (or Angular Route Params)
- The Route Parameters are a dynamic part of the Route and essential in determining the route.

```
{ path: 'product', component: ProductComponent }
```

The above route match only if the URL is **/product**

To retrieve the product details, our URL should look something like

/product/1
/product/2

How to Pass Parameters to Angular Route

Example : Products and Product Details Component

Defining the Route

```
{ path: 'product/:id', component: ProductDetailComponent }
```

If you have more than one parameter, then you can extend it by adding one more forward slash followed colon and a placeholder

```
{ path: 'product/:id/:id1/:id2', component: ProductDetailComponent }
```

Defining the Navigation

```
<a [routerLink]="['/Product', ‘2’]">{{product.name}} </a>
```

OR

```
<a [routerLink]="['/Product', product.productID]">{{product.name}} </a>
```

You can also use the **navigate method of the router object**

```
goProduct() {
    this.router.navigate(
        ['/products'. product.productID] }
    );
}
```

Retrieve the parameter in the component

- Finally, our component needs to extract the route parameter from the URL.
- This is done via the ActivatedRoute service from the angular router module to get the parameter value

ActviatedRoute

- The ActivatedRoute is a service, which keeps track of the currently activated route associated with the loaded Component.

- To use ActivatedRoute, we need to import it into our component

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

- Then inject it into the component using dependency injection

```
constructor(private _Activatedroute:ActivatedRoute)
```

- There are two properties that ActviatedRoute routes provide, which contain the Route Parameter.

  1. **ParamMap**
  2. **Params**

1. ParamMap

❑ The Angular adds the map of all the route parameters in the ParamMap object, which can be accessed from the ActivatedRoute service

❑ The ParamMap has **three methods**, which makes it easier to work with the route parameters.

   a. get method retrieves the value of the given parameter.
   b. getAll method retrieves all parameters
   c. has method returns true if the ParamMap contains a given parameter else false

2. Params

The Angular ActviatedRoute also maintains the Route Parameters in the Params array. The Params array is a list of parameter values, indexed by name.

Reading the Route Parameters

There are two ways in which you can use the ActivatedRoute to get the parameter value from the ParamMap object.

1. Using the Snapshot property of the ActivatedRoute

2. Subscribing to the paramMap or params observable property of the ActivatedRoute

Using Snapshot

- The snapshot property returns the current value of the route.
- It does not contain any observable. Hence if the value changes after you retrieve the values, you will not be notified of it.
- The snapshot contains both paramMap & params array.
- You can use any of them to read the value of id.
- Code to read the value from the paramMap object.

```
this.id=this._Activatedroute.snapshot.paramMap.get("id");
```

- Code to read the router parameter from the params array.

```
this.id=this._Activatedroute.snapshot.params["id"];
```

Using Observable

The ActivatedRoute also contains the paramMap & params observable. We can subscribe to it and listen for changes.

The paramMap observable emits the paramMap object, while the params observable emits the params array.

The following code subscribes to the paramMap observable. We use the get method to read the value of id.

```
this._Activatedroute.paramMap.subscribe( paramMap => {
   this.id = paramMap.get('id');
});
```

The code to subscribe to the params observable

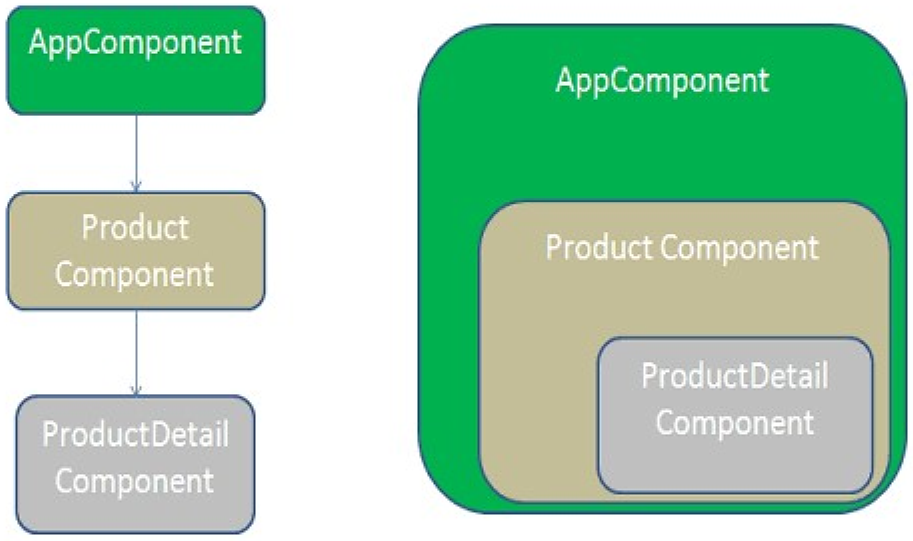
```
this._Activatedroute.params.subscribe( params => {
   this.id = params['id'];
});
```

Which one to use? snapshot or observable

Example

Angular Child Routes / Nested Routes

The Component Tree



How to Create Child Routes / Nested Routes

1. Define the Routes

```
{ path: 'product', component: ProductComponent },
{ path: 'product/:id', component: ProductDetailComponent },
```

- ✓ To make ProductDetailComponent as the child of the ProductComponent, we need to add the children key to the product route, which is an array of all child routes as shown below

```
{ path: 'product', component: ProductComponent,
    children: [
      { path: 'detail/:id', component: ProductDetailComponent }
    ]
 }
```

- ✓ The child route definition is similar to the parent route definition. It has a path and component that gets invoked when the user navigates to the child route.

Display the component using Router-outlet

- The components are always rendered in the <RouterOutlet> of the parent component.
- For ProductDetailComponent the parent component is ProductComponent and not the AppComponent

```
<h1>Product List</h1>
<div class='table-responsive'>
   <table class='table'>
      <thead>
         <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Price</th>
         </tr>
      </thead>
      <tbody>
         <tr *ngFor="let product of products;">
            <td>{{product.productID}}</td>
            <td><a [routerLink]="['detail',product.productID]">{{product.name}}
</a> </td>
            <td>{{product.price}}</td>
         </tr>
      </tbody>
   </table>
</div>
```

Display the component using Router-outlet

```
<h1>Product List</h1>
<div class='table-responsive'>
    <table class='table'>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Price</th>
            </tr>
        </thead>
        <tbody>
            <tr *ngFor="let product of products;">
                <td>{{product.productID}}</td>
                <td><a [routerLink]="['detail',product.productID]">{{product.name}} </a> </td>
                <td>{{product.price}}</td>
            </tr>
        </tbody>
    </table>
</div>

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

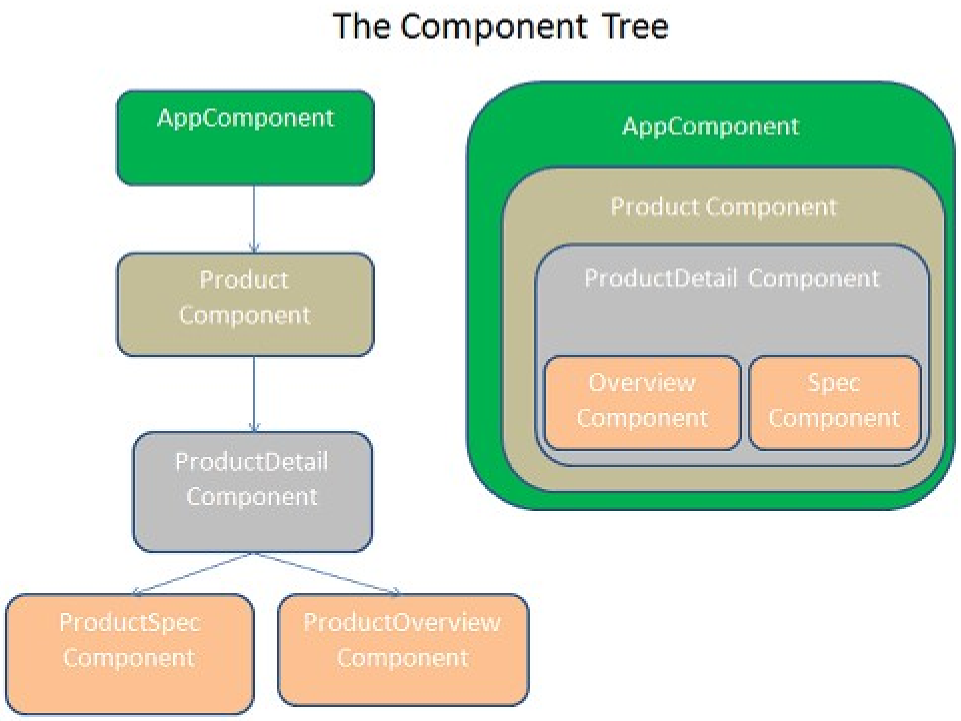
Using the Subscribe method to retrieve the parameters in child routes

```
ngOnInit() {
   this.sub=this._Activatedroute.params.subscribe( params => {
      this.id = params['id'];
      let products=this._productService.getProducts();
      this.product= products.find (p =>  p.productID == this.id);
    });
  }
```

We need to unsubscribe when the component is destroyed so as to stop the memory leakage

```
ngOnDestroy() {
   this.sub.unsubscribe();
}
```

Nesting Children’s under a child



Defining the child Route

```
export const appRoutes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'contact', component: ContactComponent },
  { path: 'product', component: ProductComponent,
      children: [
        { path: 'detail/:id', component: ProductDetailComponent,
            children : [
                { path: 'overview', component: ProductOverviewComponent },
                { path: 'spec', component: ProductSpecComponent },
                { path: '', redirectTo:'overview', pathMatch:"full" }
            ]
        }
      ]
    },
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: '**', component: ErrorComponent }
];
```

Mapping the action to View

```
<h1>Product Details Page</h1>

product : {{product.name}}
price : {{ product.price}}

<ul class="nav navbar-nav">
   <li><a [routerLink]="['overview']">OverView </a></li>
   <li><a [routerLink]="['spec']">Specification </a></li>
</ul>

<router-outlet></router-outlet>

<p>
   <a class='btn btn-default' (click)="onBack()">Back to Product List </a>
</p>
```

Query Parameters in Angular

Query Parameters in Angular allow us to generate URLs with query strings. They allow us to pass optional parameters like page number, sorting & filter criteria to the component.

Query params (or Query Parameters) are key-value pairs that appear to the right of the ? in a URL. Each query parameter is separated by &.

```
/product?page=2&filter=all
```

Which one to use? Route Parameters or Query Parameters?

- The best practice is to use Route Parameters to identify a specific resource or resources. For example a specific product or group of products.

```
//All Products
/products

//specific Product
/product/:id
```

- Use query parameters to sort, filter, paginate, etc. For example sort products based on name, rating, etc. Filter based on price, color, etc.

```
//All Products sorted on rating
/products?sort=rating

//All Products sorted on rating with color=red
/products?sort=rating &color=red

//All Products sorted on rating with color=red & second page
/products?sort= rating&color= red&page=2
```

The route parameters determine the route. The angular Router Module determines which component to load using them. Hence they are required.

Query Parameters are optional. Hence if you do not require the value then use the query parameter

Adding Query Parameters

The Query parameters are not part of the route. Hence you do not define them in the routes array like route parameters.

There are two ways in which you can pass a Query Parameter to Route

1. Using routerlink directive
2. Using router.navigate method.
3. Using the router.navigateByUrl method

Using routerlink Directive in Template

We use the queryParams property of the routerlink directive to add the query parameter. We add this directive in the template file.

```
<a [routerLink]="['product']" [queryParams]="{ page:2 }">Page 2</a>
```

The router will construct the URL as

```
/product?page=2
```

We can pass multiple Query Parameters

```
<a [routerLink]="['products']"
                  [queryParams]="{ color:'blue' , sort:'name'}">Products</a>
```

```
/products?color=blue & sort=name
```

Using router.navigate method in Component

```
goTo() {
    this.router.navigate(
        ['/products'],
        { queryParams: { page: 2, sort:'name'} }
    );
}
```

Using the router.navigateByUrl method in the component

We can also use the **navigateByUrl** method of the router.
navigateByUrl expects an absolute URL, Hence we need to build our query string programmatically.
Also,the navigateByUrl method does not have queryParamsHandling option.

```
this.router.navigateByUrl('product ? pageNum=2');
```

Reading Query Parameters

Reading the Query parameters is similar to reading the Router Parameter