ANGULAR - ROUTING

June 21st, 2023

Objective

- Why Application Needs Routing?
- Difference between Routing on Server and Routing in Browser
- Angular Routing
- Configure Routes for Angular App
- Navigate through code (TypeScript)
- Route Parameters
- Lazy Loading
- Route Guards

Why Routing is Necessary?

- **Help to Navigate from one page to another**
- **Entering a URL in the address bar and the browser navigates to a corresponding page**
- **Show/Hide modules based on the User privileges**

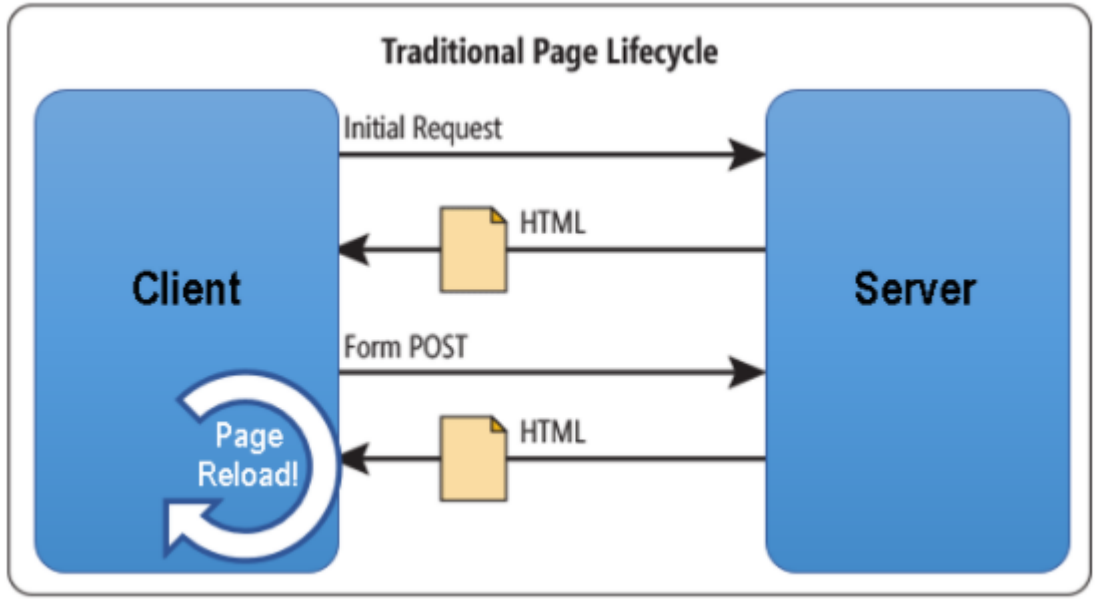
Types of Routing

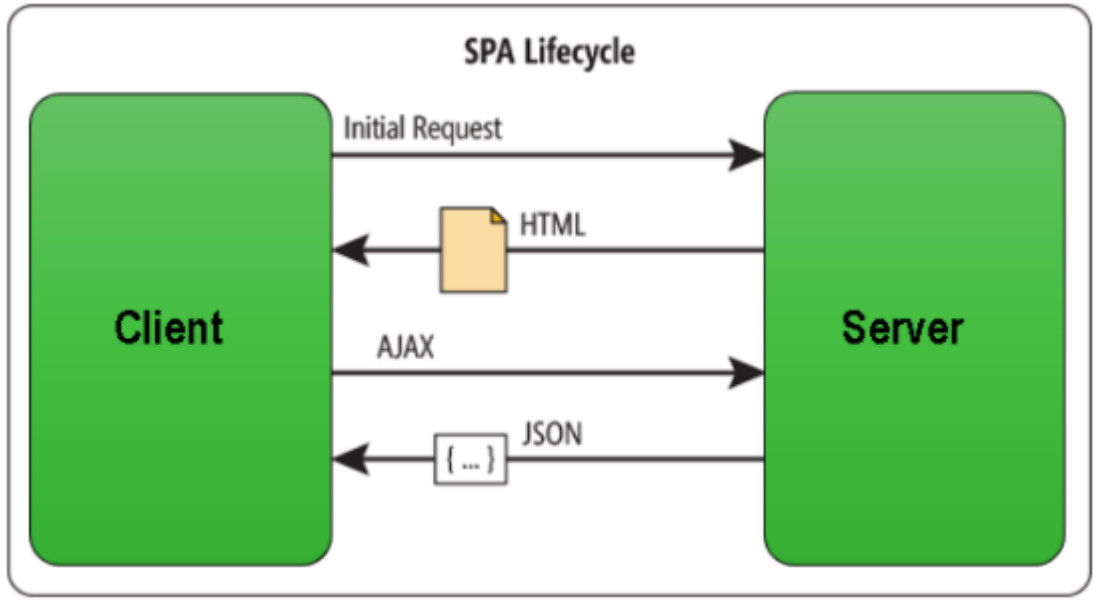
- **Traditional Routing**
  - User clicks a link in the browser
  - Browser sends an HTTP request to server
  - Server reads the URL from the HTTP request and generates an appropriate HTTP response
  - Server sends the HTTP response to the browser

- **JavaScript routing**
  - Update the web application state when the browser URL changes
  - Update the browser URL when the web application state changes.

Single Page Application Behavior

- When a user navigates from one page to another, the page is updated dynamically without reload, even if the URL changes.





Routing - Angular

- Angular Router performs the following steps in order:
  - Reads the browser URL the user wants to navigate to
  - Figures out which router state corresponds to the URL
  - Runs the guards that are defined in the router state
  - Activates the Angular components to display the page

- **Enable Routing:**
  - **router service**: the global Angular Router service in our application

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

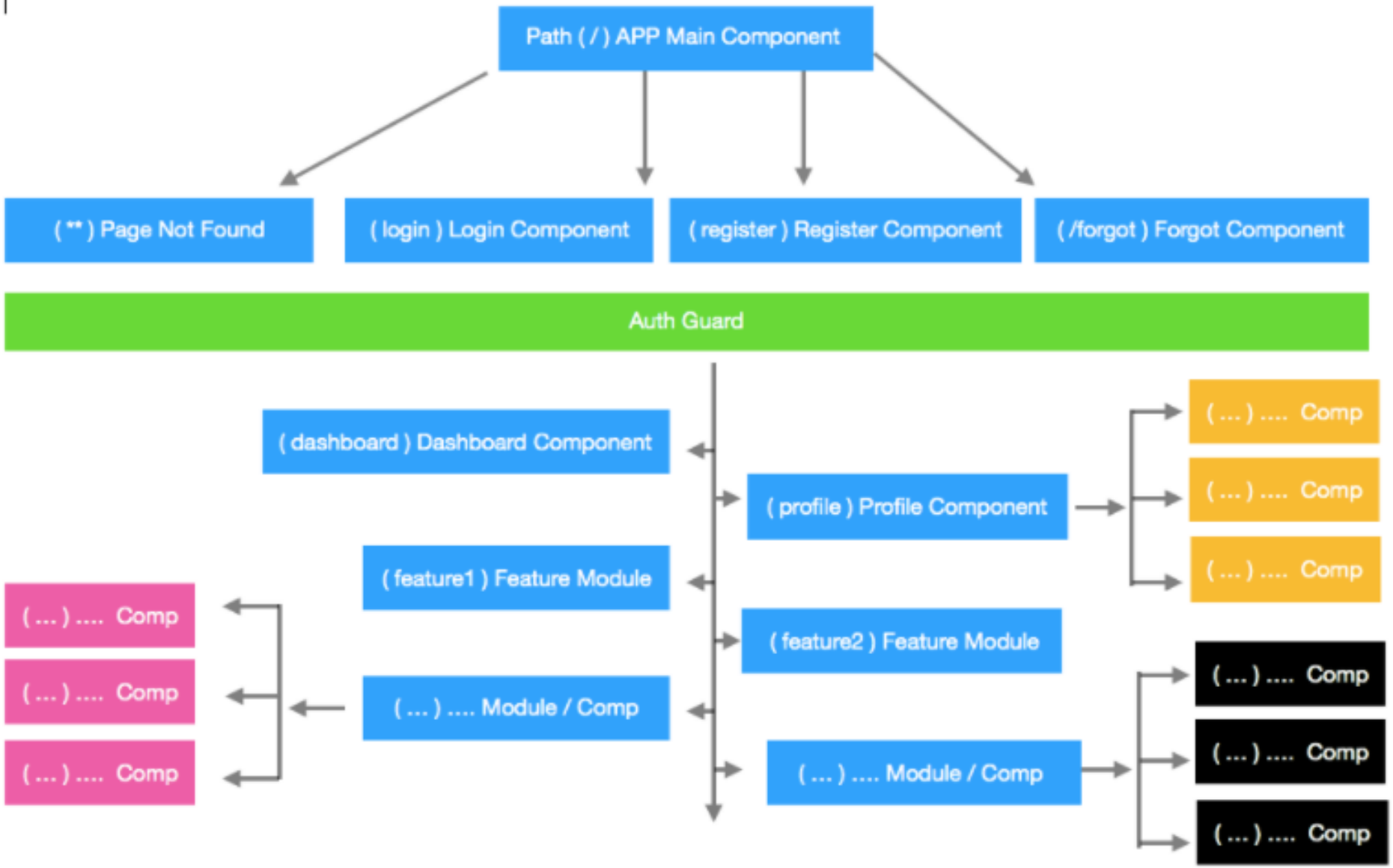
  - **router configuration**: definition of all possible router states our application can be in

```
{ path: 'company', component: CustomerComponent },
{ path: 'employee/:id',      component: CustomerDetailComponent }
```

  - **router outlet**: location in the DOM where Angular Router can place activated components.

```
<nav>
   <a [routerLink]='["/Customer"]' routerLinkActive="active">Customer</a>
   <a [routerLink]='["/CustomerDetail"]' routerLinkActive="active">Customer Details</a>
</nav>
<router-outlet></router-outlet>
```

Angular Routing Continued..



Navigate through Code

- **Import Router in component**

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

- **Redirect Method**

```
export class AppComponent {

  constructor(private router: Router) { }

  navigateToFirst() {
    this.router.navigate(['first'])
  }
  navigateToSecond() {
    this.router.navigateByUrl('/second')
  }

}
```

- Both methods are similar the only difference is that the **navigate** method takes an array and works as the URL, and the **navigateByUrl** method takes an absolute path.

```
    this._router.navigate([‘customers’,1])
    this._router.navigateByUrl(‘/customers/1')
```

FIS

Route Parameters

- The Route parameters are a dynamic part of the Route and essential in determining the route

```
{ path: ‘customer', component: CustomerComponent }
```

- If we want to pass a parameter through route

```
{ path: ‘customer-detail/:id', component: CustomerDetailsComponent }
```

- Navigation

```
<a [routerLink]="['/customer-detail', ‘2’]">John Mathew</a>
```

- Retrieve the parameter in the component

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

Lazy loading

- **on-demand loading -** loads the Modules only on a need basis instead of loading all the modules at the same time.
- **Lazy loading works at the module level**

```
ng g module  Admin --route Admin --module app.module

{path: "admin", loadChildren:'./admin/admin.module#AdminModule’}

loadChildren: () => import('./admin/admin.module').then(m => m. AdminModule)
```

- When the user navigates to the admin section, then AdminModule loads the routes and components of the AdminModule
- The lazy loaded module loads only for the first visit of the URL, it will not load when we revisit that URL again.

Route Guards

- Angular Guards is used to control, whether the user can navigate to or away from the current route

- **Types of Route Guards**

  CanActivate - if a route can be activated

  CanDeactivate - if the user can leave the component

  Resolve - This guard delays the activation of the route until some tasks are complete

  CanLoad - The CanLoad Guard prevents the loading of the Lazy Loaded Module.

  CanActivateChild - This guard determines whether a child route can be activated

Route Guard continued..

Steps to create Route Guards:

- Create a Route Guard as Service
- Implement the Guard Method in the Service
- Register the Guard Service in the Root Module and update the required routes

```
ng g service AuthGuard

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

export class AuthGuardService implements CanActivate {

  constructor(private _route:Router) { }

  canActivate(route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean  {
    console.log("canActivate");
    console.log('You don't have enough privileges to delete customer')
    alert('You don't have enough privileges to delete customer')
    this._route.navigate(['add-customer']);
    return false;
  }
}
```

Route Guard continued..

```
canActivate(route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): boolean  {

  if(state.url.indexOf('view-customer') > -1){
    let userNm = localStorage.getItem('userName');
    if(userNm === 'Admin'){
      console.log("you have permissions");
      return true;
    }
    else{
      console.log('You dont have enough privilges to view this page')
      this._route.navigate(['add-customer']);
      return false;
    }
  }
}
```