

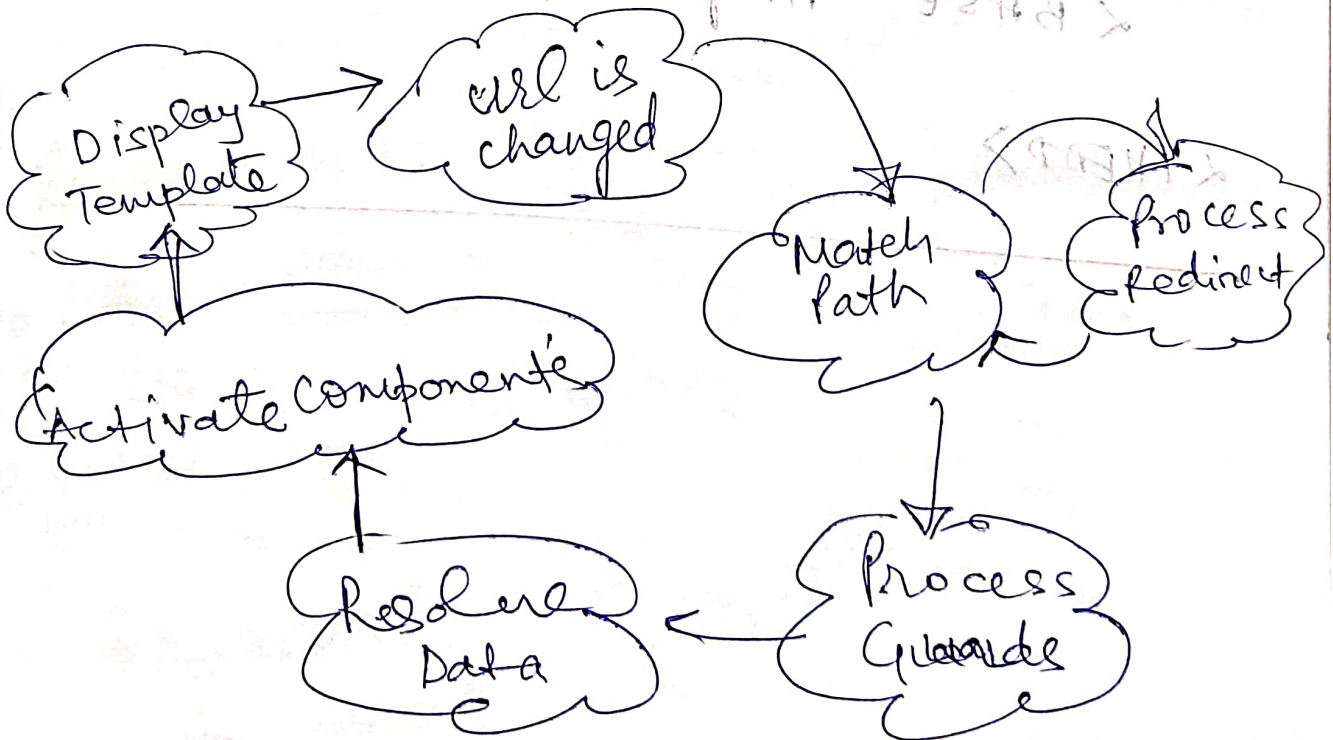
# Angular Routing

- ① child and Secondary routes
- ② route parameters
- ③ styling, animating and reacting
- ④ Route guards and resolve.
- ⑤ Lazy loading

Primary Routes

↓  
child Routes

↓  
Secondary Routes



Router - Outlet  
Life cycle.

## Pre-requisites

- Angular Modules
- Components
- Templates
- Services

## Setting the base path

--base-href /APM/  
path

<base href = "/APM/">

<HEAD>

<BASE href = "/APM/">

<!HEAD>

## Router Module

- ↳ Directives → ① Router link
- ↳ service → ② Router Link Active
- ↳ Configuration → ③ Router Outlet

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

## RouterModule.forRoot()

- register the route service
- used once for the application.

## RouterModule.forChild()

- Does NOT register the router service
- Used in feature modules.

```
@NgModule({
```

```
  imports: [
```

```
    RouterModule.forRoot()
```

```
  ],
```

## Configuring Routes

{ path: 'welcome', component: — }

{ path: '', redirect to: 'welcome',

pathMatch: 'full' }

{ path: '??', components: [ pageNotFound ]

]

Note: Order in routes matters a lot.  
Router uses a first match  
wins strategy.

### Activating a Route with code

this.router.navigate([' /home ']) — standard

this.router.navigate('/home') — shortcut

this.router.navigateByUrl('/welcome')  
complete URL path

It removes the secondary routes

## Secondary Routes

Can be used for → Dashboard  
→ Multi-window user Interface  
→ Notes or comments  
→ Messages.

```
<router-outlet name="popup">  
</router-outlet>
```

---

RouterModule.forChild ( [

```
{  
  path: 'messages',  
  component: MessageComponent,  
  outlet: 'popup'  
},  
)
```

---

```
<a [routerLink]="[outlets:
```

```
  { popup: ['messages'] }  
  ]" > message </a>
```

## Activating Secondary Routes in Code

```
this.router.navigate(['& outlets:  
& pop up: ['messages']??]);
```

## Clearing Secondary routes

```
let routerLink = ''  
outlets = { pop up: null ?? };
```

# Route Guards

- can Activate
- can Activate child
- can Deactivate
- [can Load
- resolve (data Prefetch)

## Guard Processing

- can Deactivate
  - ↳ can Load
  - ↳ can Activate child
    - ↳ can Activate
    - ↳ resolve.

## Building a Guard @ service

auth.guard.ts

```
import { Injectable } from '@angular/core'
{ can Activate } from '@angular/router'
```

```
canActivate(): boolean {
```

```
}
```

## Guarding a Route

```
{  
  path: '/id',  
  component: _____,  
  resolve: { product: ProductResolver,  
    canActivate: [AuthGuard]  
  }  
}
```

CLI

ng g user/auth