**Angular UI development**

Contents

[**Angular UI development** 1](#_Toc43811326)

[Angular topics 3](#_Toc43811327)

[Project setup 3](#_Toc43811328)

[Models 5](#_Toc43811329)

[String interpolation 5](#_Toc43811330)

[Property binding 5](#_Toc43811331)

[View encapsulation 6](#_Toc43811332)

[Local references in templates 6](#_Toc43811333)

[Getting access to the template and DOM with @ViewChild 7](#_Toc43811334)

[Projecting content into components with ng-content 7](#_Toc43811335)

[Component lifecycle 8](#_Toc43811336)

[Lifecycle hooks in action 8](#_Toc43811337)

[How to emit events. 9](#_Toc43811338)

[Directives 10](#_Toc43811339)

[Structural Directives 11](#_Toc43811340)

[Attribute directives 12](#_Toc43811341)

[Generate a directive 12](#_Toc43811342)

[@HostBinding 12](#_Toc43811343)

[[ngSwitch] directive 13](#_Toc43811344)

[Services and dependency injection 14](#_Toc43811345)

[Creating a service 14](#_Toc43811346)

[Angular hierarchical injector 15](#_Toc43811347)

[Cross component communication through EventEmitter 16](#_Toc43811348)

[Angular Routing 16](#_Toc43811349)

[Redirecting and wildcard routes 21](#_Toc43811350)

[Outsourcing the route configuration 21](#_Toc43811351)

[How to define routes 23](#_Toc43811352)

[How to export routes to the app.module.ts 23](#_Toc43811353)

[Using routes 23](#_Toc43811354)

[child routes 23](#_Toc43811355)

[configuring route parameters 23](#_Toc43811356)

[Programmatically navigating to a route 25](#_Toc43811357)

[Assignment 1 27](#_Toc43811358)

Angular topics:

* Components
* String interpolation, property binding, local references
* Directives
* Event Emitters
* Services and Dependency injection
* Routing , Page Navigation
* Observable
* Forms
* HttpModule
* Authentication and route protection
* deployment

# Project setup

Install node: <https://nodejs.org/en/download/>

Install Visual Studio code IDE.

npm install -g @angular/cli.

npm install bootstrap / npm install --save bootstrap@3

npm install --save @angular/http@latest

Create a new angular app:

ng new first-angular-app

cd first-angular-app

make bootstrap configuration change in: angular.json

"styles": [

              "node\_modules/bootstrap/dist/css/bootstrap.min.css",

              "src/styles.css"

            ],

Start the app

ng serve

ng serve --open --port 4201

browse to: <http://localhost:4200/>

Register components in app.module.ts , example: registering HeaderComponent

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { HeaderComponent } from './header/header.component';

@NgModule({

  declarations: [

    AppComponent,

    HeaderComponent

  ],

  imports: [

    BrowserModule,

    FormsModule,

    HttpModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Components

generate a component: ng generate component recipes

generate a component within a component:

ng generate component recipes/recipe-list

ng generate component recipes/recipe-detail

ng generate component recipes/recipe-list/recipe-item

components generated through the ng generate component command are automatically added to the root: app.module.ts

ng generate component shopping-list

ng generate component shopping-list/shopping-edit

# Models

Define a model: recipe.model.ts

Refer the model in component.ts classes

Import a model in the component class and use it in the component template

Model

Declaring a model

Option 1

export class Ingredient {

public name: string;

public amount: number;

constructor(name: string, amount: number) {

this.name = name;

this.amount = amount;

}

}

Option 2

export class Ingredient {

constructor(public name: string, public amount: number) {}

}

# String interpolation

<h4 class="list-group-item-heading">{{ recipe.name }}</h4>

# Property binding

[src]= "recipe.imagePath"

# View encapsulation

The styles ( css ) applies only to the current component and not to the whole app . Declared in component.module.

Example:

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

encapsulation: ViewEncapsulation.Emulated //This is default , css styles restricted only to current component. Other options: Native --> shadow DOM , None --> styles apply to all components

})

encapsulation: ViewEncapsulation.Emulated //This is default , css styles restricted only to current component. Other options: Native --> shadow DOM , None --> styles apply to all components

})

# Local references in templates

Get access to some element in the template ( html ) and then use it either directly in the template or pass it to the typescript code.

Example:

<input

      type="text"

      class="form-control"

      #serverNameInput>

(click)="onAddServer(serverNameInput)">Add Server</button>

Module

onAddServer(nameInput) {

    console.log(nameInput.value);

# Getting access to the template and DOM with @ViewChild

Local references fetched through @ViewChild

<input

      type="text"

      class="form-control"

      #serverContentInput

      >

In module

@ViewChild('serverContentInput',{static: true}) serverContentInput: ElementRef ;

  onAddServer(nameInput: HTMLInputElement) {

    console.log(nameInput.value);

    console.log(this.serverContentInput);

    this.serverCreated.emit({

      serverName: nameInput.value,

      serverContent: this.serverContentInput.nativeElement.value

    });

  }

# Projecting content into components with ng-content

Any content placed between the opening and closing tags of the component are placed into the component at the <ng-content> </ng-content> tags.

Example:

app.component.html

<app-server-element

      \*ngFor="let serverElement of serverElements"

      [srvElement]="serverElement">

      <p>

        <strong \*ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>

        <em \*ngIf="serverElement.type === 'blueprint'">{{ serverElement.content }}</em>

      </p>

    </app-server-element>

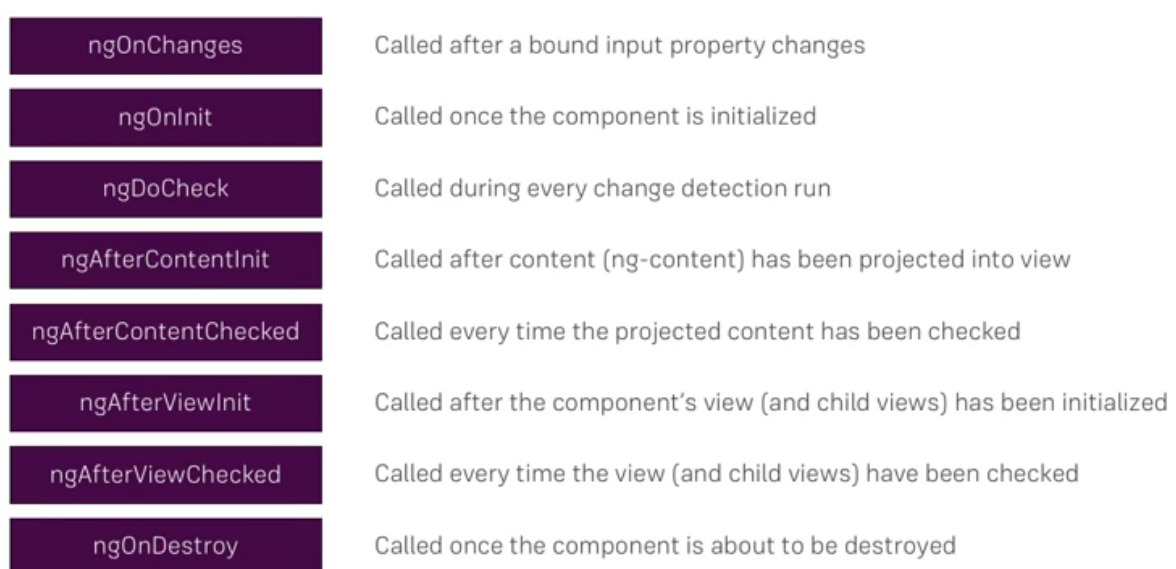
server-element.component.html

<div class="panel-body">

         <ng-content></ng-content>

        </div>

# Component lifecycle



Lifecycle hooks in action:

export class ServerElementComponent implements OnInit, OnChanges, DoCheck, AfterContentInit, AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy

constructor() {

    console.log('constructor called !');

   }

   ngOnChanges(changes: SimpleChanges)  {

     console.log('ngOnChanges called !');

     console.log(changes);

   }

  ngOnInit(): void {

    console.log('ngOnInit called !');

  }

  ngDoCheck(): void {

    console.log('ngDoCheck called !');

  }

  ngAfterContentInit(): void {

    console.log('ngAfterContentInit called !');

  }

  ngAfterContentChecked(): void {

    console.log('ngAfterContentChecked called !');

  }

  ngAfterViewInit(): void {

    console.log('ngAfterViewInit called !');

  }

  ngAfterViewChecked(): void {

    console.log('ngAfterViewChecked called !');

  }

  ngOnDestroy(): void {

    console.log('ngOnDestroy called !');

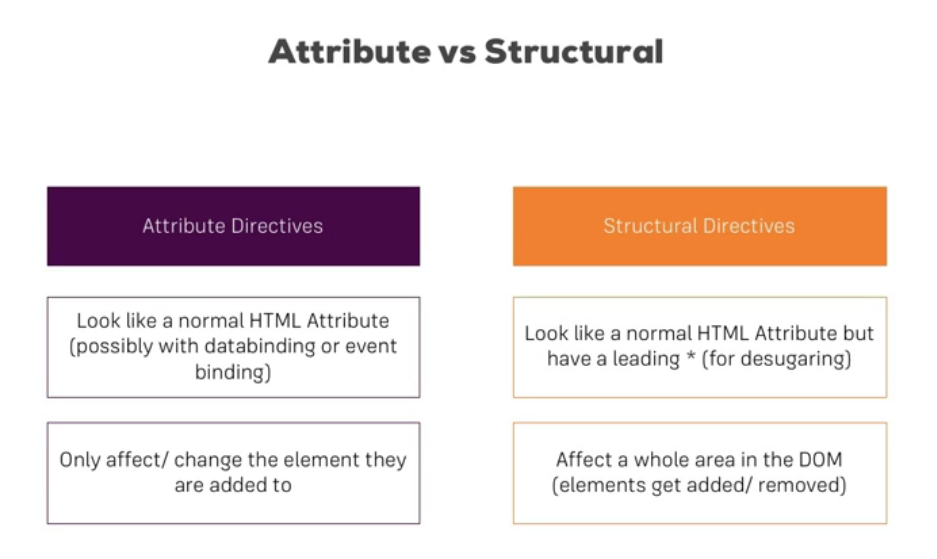
  }

# How to emit events.

How to declare EventEmitter

How to use @Output() , @Input()

# Directives

****

**Example:**

structural directive

\*ngIf

\*ngFor

attribute directive

[ngClass]

[ngStyle]

\*ngFor directive

<div class="row">

    <div class="col-xs-12">

        <a href="#"

        class="list-group-item clearfix"

        \*ngFor="let recipe of recipes">

            <div class="pull-left">

                <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

                <p class="list-group-item-text">{{ recipe.description }}</p>

            </div>

            <span class="pull-right">

                <img

                 [src]= "recipe.imagePath"

                 alt="{{recipe.name}}"

                 class="img-responsive" style="max-height: 50px;">

            </span>

        </a>

        <app-recipe-item></app-recipe-item>

    </div>

</div>

<div \*ngIf="onlyOdd">

          <li

          class="list-group-item"

          \*ngFor="let odd of oddNumbers"

          [ngClass]="{odd: odd % 2 !== 0}"

          [ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"

          >

          {{ odd }}

        </li>

        </div>

## Structural Directives

They change the structure of the DOM. Take a Boolean that equates to true or false. Ex:

<p \*ngIf="serverCreated">Server was created, server name is: {{ serverName }} </p>

serverCreated is a Boolean variable.

Ex:

<p \*ngIf="serverCreated;else noServer">Server was created, server name is: {{ serverName }} </p>

<ng-template #noServer>

    <p>No server was created !</p>

</ng-template>

## Attribute directives

ngStyle

They don’t add or remove the elements. They only change the elements they were placed on.

<p [ngStyle]="{backgroundColor: getColor()}" >The server component with {{ serverId }}  is {{ getServerStatus() }}</p>

 getColor() {

            return this.serverStatus === 'online' ? 'green' : 'red';

        }

Attribute directives

ngClass dynamically add or remove css classes

<p [ngStyle]="{backgroundColor: getColor()}"

   [ngClass]="{online: serverStatus === 'online'}"

>

The server component with {{ serverId }}  is {{ getServerStatus() }}

</p>

## Generate a directive

PS D:\angular-workspace\directives-start> ng generate directive better-highlight

CREATE src/app/better-highlight.directive.spec.ts (261 bytes)

CREATE src/app/better-highlight.directive.ts (159 bytes)

Cover HostListener

define a HostListener for mouseenter and mouseleave

@HostListener('mouseenter') mouseover(eventData: Event) {

      this.renderer.setStyle(this.elRef.nativeElement,'background-color', 'orange');

    }

    @HostListener('mouseleave') mouseleave(eventData: Event) {

      this.renderer.setStyle(this.elRef.nativeElement,'background-color', 'transparent');

    }

## @HostBinding

@HostBinding('style.backgroundColor') backgroundColor: string;

binding to directive properties

 <p appBetterHighlight

      [defaultColor]="'yellow'"

      [highlightColor]="'pink'"

      >Style me with basic directive</p>

@Input() defaultColor: string = 'transparent';

@Input() highlightColor: string = 'orange';

@HostBinding('style.backgroundColor') backgroundColor: string;

@HostListener('mouseenter') mouseover(eventData: Event) {

      this.backgroundColor = this.highlightColor;

    }

    @HostListener('mouseleave') mouseleave(eventData: Event) {

      this.renderer.setStyle(this.elRef.nativeElement,'background-color', 'transparent');

      this.backgroundColor = this.defaultColor;

    }

Create a custom structural directive and use in the html template

<div \*appUnless="onlyOdd">

## [ngSwitch] directive

in component module.ts

value = 10;

in component html

 <div [ngSwitch]="value">

        <p \*ngSwitchCase="5" [ngStyle]="{backgroundColor: 'lightblue'}">Value is 5</p>

        <p \*ngSwitchCase="10" [ngStyle]="{backgroundColor: 'lightgreen'}">Value is 10</p>

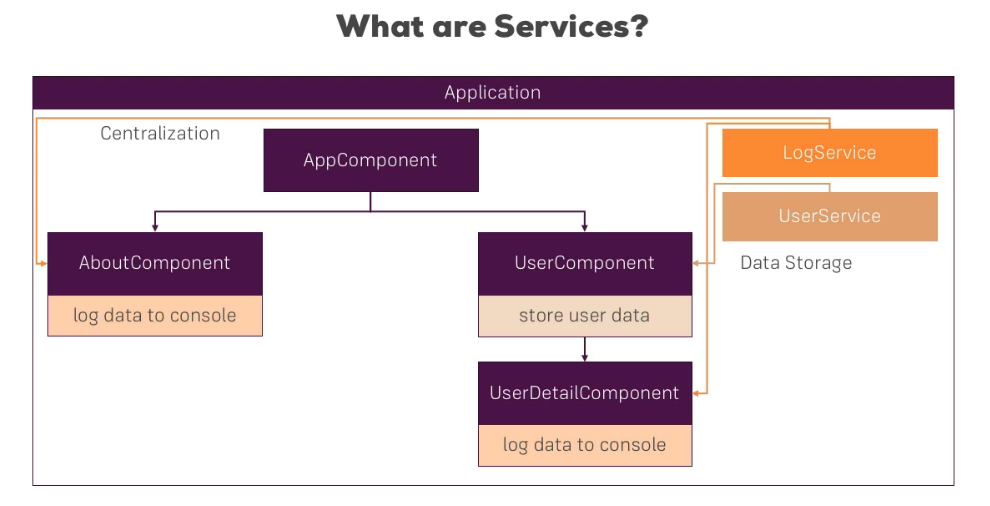
        <p \*ngSwitchCase="100" [ngStyle]="{backgroundColor: 'lightgrey'}">Value is 100</p>

        <p \*ngSwitchCase="1000" [ngStyle]="{backgroundColor: 'skyblue'}">Value is 1000</p>

        <p \*ngSwitchDefault>Value is Default</p>

      </div>

Dropdown open and close using attribute directive



# Services and dependency injection

Creating a service:

export class LoggingService {

    logStatusChange(status: string) {

        console.log('A server status changed, new status: ' +status);

    }

}

Dependency injection for a service:

import {LoggingService} from '../logging.service';

@Component({

  selector: 'app-new-account',

  templateUrl: './new-account.component.html',

  styleUrls: ['./new-account.component.css'],

  providers: [LoggingService]

})

export class NewAccountComponent {

constructor(private loggingService: LoggingService) {}

this.loggingService.logStatusChange(accountStatus);

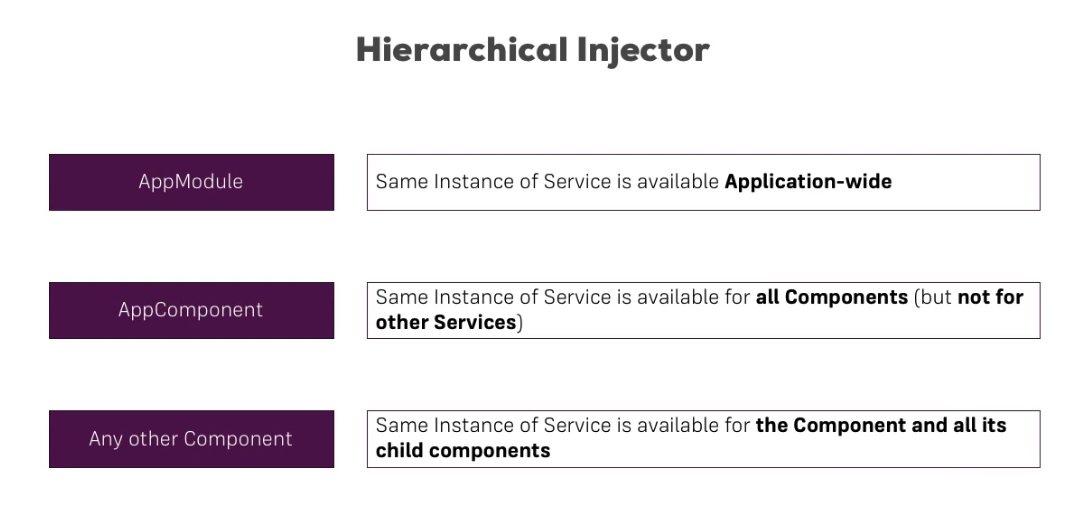
## Angular hierarchical injector

If we inject a dependency to a service inside a component in Angular , angular will create an instance of the service for the component and for all the child components.

The component into which the dependency has been injected as well as all the child components and their child components will receive the same instance of the service.

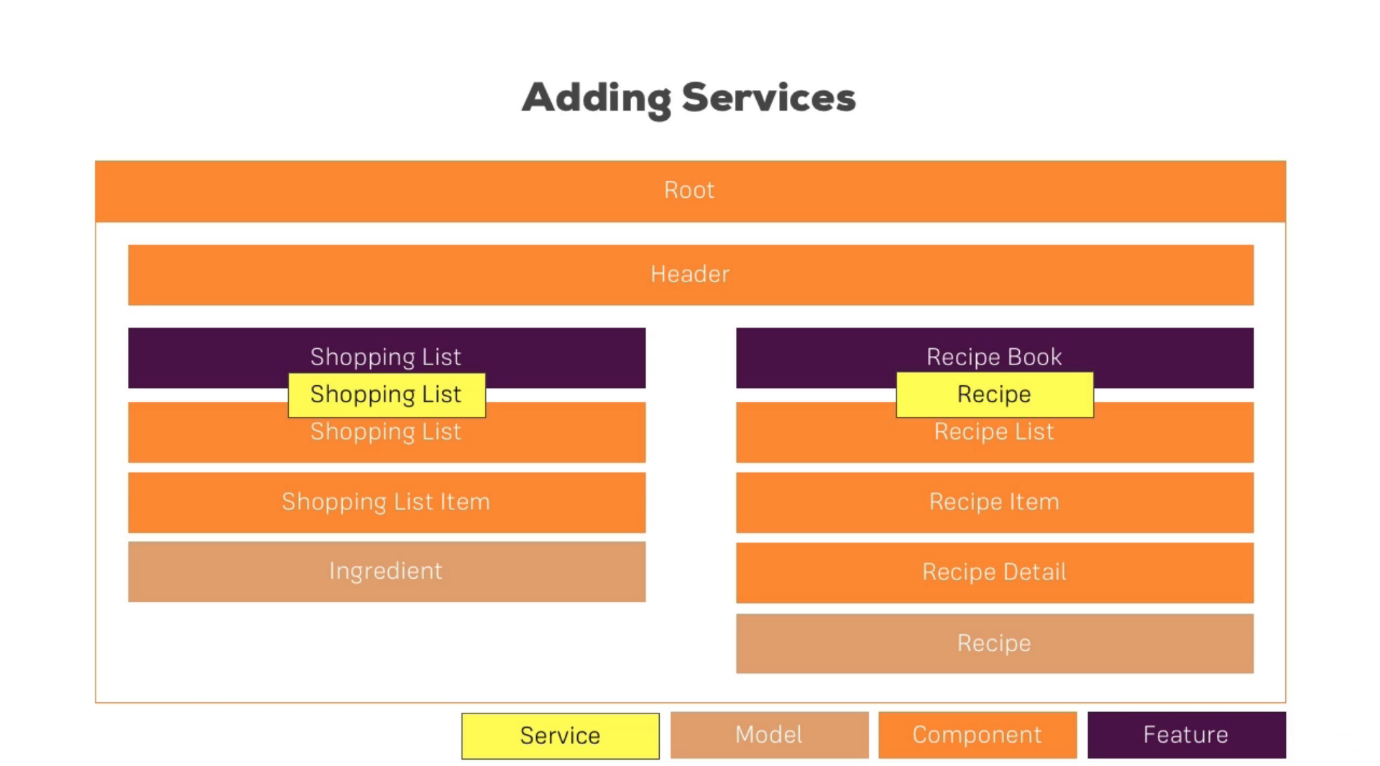
The reference to the Service in the providers should only be done once , here at the AppModule level. But in all the child modules , it should still be there as import and in constructor , but not in the providers.

providers: [AccountsService]



# Cross component communication through EventEmitter

Setting up Services



# Angular Routing

npm install --save @angular/router

Declare the routes in app.module.ts

const appRoutes: Routes = [

  {path: '', component: HomeComponent },

  {path: 'users', component: UserComponent },

  {path: 'servers', component: ServersComponent }

];

Register the routes:

imports: [

    BrowserModule,

    FormsModule,

    RouterModule.forRoot(appRoutes)

  ]

Add this directive to the app.component.html. To render the selected components

<router-outlet></router-outlet>

This marks the place in the document where the angular-router will load the component of the currently selected router.

We need to remove all the selectors of the induvial components

Invoke the routes from a navigation panel.

However , this will reload the full app , and is not the ideal way

<ul class="nav nav-tabs">

        <li role="presentation" class="active"><a href="/">Home</a></li>

        <li role="presentation"><a href="/servers">Servers</a></li>

        <li role="presentation"><a href="/users">Users</a></li>

      </ul>

Use the below notation , to add the router links to the navigation bar , so that the whole app doesn’t load. routerLink is an angular-router directive.

<ul class="nav nav-tabs">

        <li role="presentation" class="active"><a routerLink="/">

          Home</a></li>

        <li role="presentation"><a routerLink="/servers">

          Servers</a></li>

        <li role="presentation"><a [routerLink]="['/users']">

          Users</a></li>

      </ul>

Enable the selected routerlink to be active in the nav bar

The routerLinkActiveOptions is to ensure that the / ( root ) path is not always active in the nav bar.

<ul class="nav nav-tabs">

        <li role="presentation"

        routerLinkActive="active"

        [routerLinkActiveOptions]="{exact: true}"

        ><a routerLink="/">

        Home</a></li>

        <li role="presentation"

        routerLinkActive="active"

        [routerLinkActiveOptions]="{exact: true}"

        ><a routerLink="/servers">

        Servers</a></li>

        <li role="presentation"

        routerLinkActive="active"

        [routerLinkActiveOptions]="{exact: true}"

        ><a [routerLink]="['/users']">

        Users</a></li>

      </ul>

Programatically navigate to routers

in template:

<button class="btn btn-primary" (click)="onLoadServers()" >Load Servers</button>

in typescript

inject in constructor

constructor(private router: Router) { }

invoke through the method

 onLoadServers() {

    // navigate to the servers page

    this.router.navigate(['/servers']);

  }

Passing parameters to routes

app.module.ts

{path: 'users/:id/:name', component: UserComponent },

const appRoutes: Routes = [

  {path: '', component: HomeComponent },

  {path: 'users', component: UserComponent },

  {path: 'users/:id/:name', component: UserComponent },

  {path: 'servers', component: ServersComponent }

];

Fetching route parameters

component.html

<p>User with ID {{ user.id }} loaded.</p>

<p>User name is {{ user.name }}</p>

component.ts

constructor(private route: ActivatedRoute) { }

ngOnInit() {

    this.user = {

        id: this.route.snapshot.params['id'],

        name: this.route.snapshot.params['name']

    };

  }

url: <http://localhost:4200/users/1/max>

Fetching Route parameters reactively

changes in html

<a [routerLink]="['/users', 10, 'Anna']">Load Anna</a>

Changes in module.ts

if the route.params.subscribe is not added , upon clicking the Anna link , the url in the address bar will change but the new data on the page will not load .

This is for a scenario where a component may be reloaded from within itself with new parameter

ngOnInit() {

    this.user = {

        id: this.route.snapshot.params['id'],

        name: this.route.snapshot.params['name']

    };

    this.route.params

        .subscribe(

        ( params: Params) => {

          this.user.id = params['id'];

          this.user.name = params['name'];

        }

        );

  }

passing query parameters and fragments

in app.module.ts

{path: 'servers/:id/edit', component: EditServerComponent }

in html

 <a

        [routerLink]="['/servers', 5, 'edit']"

        [queryParams]="{allowEdit: '1'}"

        fragment="loading"

        href="#"

        class="list-group-item"

        \*ngFor="let server of servers">

        {{ server.name }}

      </a>

the resulting url in browser:

<http://localhost:4200/servers/5/edit?allowEdit=1#loading>

passing query parameters and fragments programmatically

in html

<button class="btn btn-primary" (click)="onLoadServer(1)" >Load Server 1</button>

in module.ts

onLoadServer(id: number) {

    this.router.navigate(

      ['/servers', id, 'edit'],

      {queryParams: {allowEdit: '1'}, fragment: 'loading'}

    );

  }

Setting up child ( nested ) Routes

in app.module.ts

const appRoutes: Routes = [

  {path: '', component: HomeComponent },

  {path: 'users', component: UserComponent },

  {path: 'users/:id/:name', component: UserComponent },

  {path: 'servers', component: ServersComponent, children: [

    {path: ':id', component: ServerComponent},

    {path: ':id/edit', component: EditServerComponent }

    ]

  }

];

Add a <router-outlet></ router-outlet> to the parent component under which the child routes have been defined. The child routes are mapped to child components as well.

<div class="col-xs-12 col-sm-4">

    <router-outlet></router-outlet>

    <!--<button class="btn btn-primary" (click)="onReload()">Reload Page</button>

    <app-edit-server></app-edit-server>

    <hr>

    <app-server></app-server> -->

  </div>

## Redirecting and wildcard routes

ng generate component page-not-found

app.module.ts

{ path: 'not-found', component: PageNotFoundComponent },

  { path: '\*\*', redirectTo: '/not-found'}

const appRoutes: Routes = [

  {path: '', component: HomeComponent },

  {path: 'users', component: UserComponent, children: [

    {path: ':id/:name', component: UserComponent }

      ]

  },

  {path: 'servers', component: ServersComponent, children: [

    {path: ':id', component: ServerComponent},

    {path: ':id/edit', component: EditServerComponent }

    ]

  },

  { path: 'not-found', component: PageNotFoundComponent },

  { path: '\*\*', redirectTo: '/not-found'}

];

## Outsourcing the route configuration

create a new component

app-routing.module.ts

Remove the appRoutes from the app.module.ts and place them in this new module.

const appRoutes: Routes = [

  {path: '', component: HomeComponent },

  {path: 'users', component: UserComponent, children: [

    {path: ':id/:name', component: UserComponent }

      ]

  },

  {path: 'servers', component: ServersComponent, children: [

    {path: ':id', component: ServerComponent},

    {path: ':id/edit', component: EditServerComponent }

    ]

  },

  { path: 'not-found', component: PageNotFoundComponent },

  { path: '\*\*', redirectTo: '/not-found'}

];

@NgModule({

  imports: [

    RouterModule.forRoot(appRoutes)

  ],

  exports: [RouterModule]

})

export class AppRoutingModule {

}

Add this module to the imports section of the app.module.ts

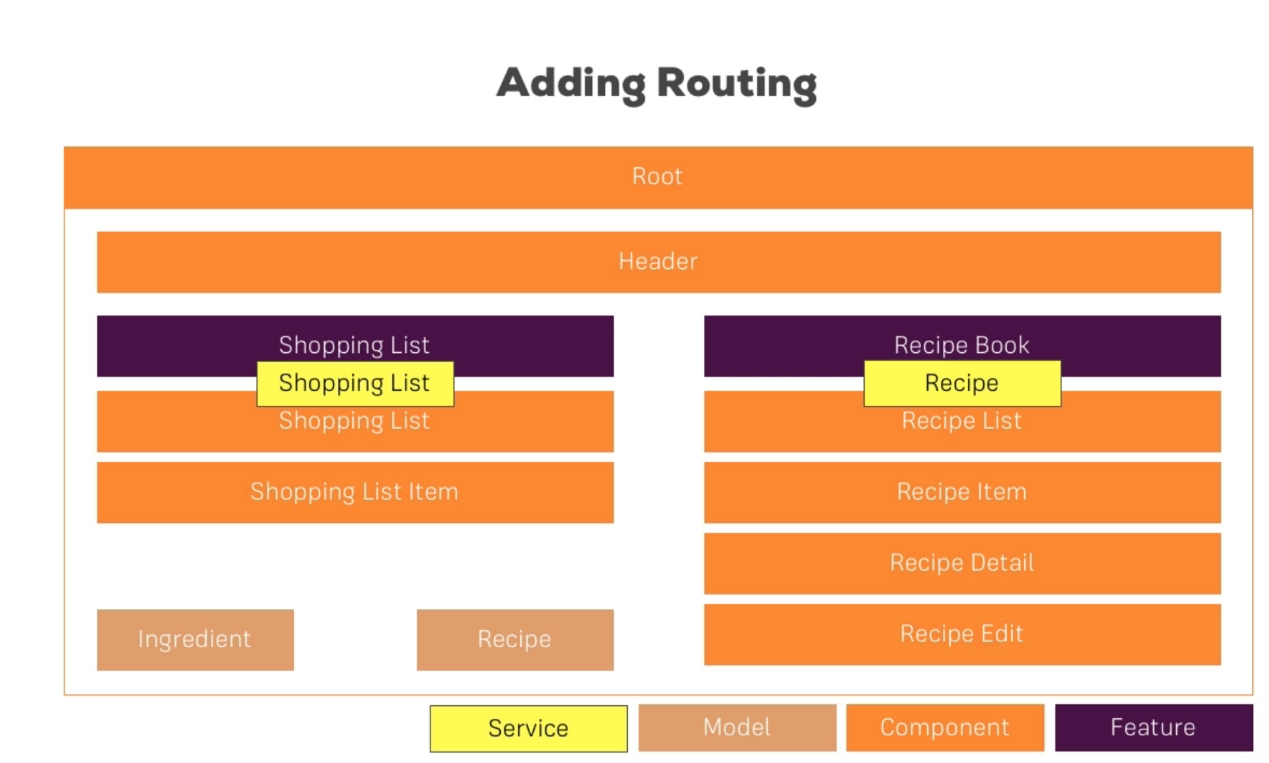
 imports: [

    BrowserModule,

    FormsModule,

    AppRoutingModule

  ],



## How to define routes

## How to export routes to the app.module.ts

## Using routes

## child routes

## configuring route parameters

converting a route parameter to number from string.

this.id = +params['id'];

child route in app-routing.module.ts

{ path: 'recipes', component: RecipesComponent, children: [

  { path: '', component: RecipeStartComponent },

  { path: ':id', component: RecipeDetailComponent }

] }

dynamic parameter

{ path: ':id', component: RecipeDetailComponent }

in the respective typescript module.

call ActivatedRoute in the constructor

constructor(private recipeService: RecipeService,

              private route: ActivatedRoute

        ) { }

Listen to the changes on the route for params and if the param changes , invoke call to the recipe service to change the recipe value.

  ngOnInit(): void {

      this.route.params

        .subscribe(

           (params: Params) => {

              this.id = +params['id'];

              this.recipe = this.recipeService.getRecipe(this.id);

          }

        );

  }

The recipe property in the module.ts is used in the associated html template using string interpolation. ex:

<h1>{{ recipe.name}}</h1>

   <div class="col-xs-12">

        {{ recipe.description}}

    </div>

passing dynamic parameters to links

in the module of the component which needs to be displayed based on the dynamic parameter passed , declare a Input field which will be set by the parent.

export class RecipeItemComponent implements OnInit {

  @Input() recipe: Recipe;

  @Input() index: number;

from the parent component html:

<div class="col-xs-12">

        <app-recipe-item

        \*ngFor="let recipeEL of recipes; let i = index"

        [recipe] = "recipeEL"

        [index]="i"></app-recipe-item>

    </div>

adding the routerLink to the component html which will display the data based on the dynamic param

<a style="cursor: pointer;"

   class="list-group-item clearfix"

   [routerLink]="[index]"

   >

    <div class="pull-left">

        <h4 class="list-group-item-heading">{{ recipe.name }}</h4>

        <p class="list-group-item-text">{{ recipe.description }}</p>

    </div>

    <span class="pull-right">

        <img [src]="recipe.imagePath" alt="{{recipe.name}}" class="img-responsive" style="max-height: 50px;">

    </span>

</a>

Highlight an active link. Use routerlinkActive in the component html

<a style="cursor: pointer;"

   class="list-group-item clearfix"

   [routerLink]="[index]"

   routerLinkActive="active"

   >

## Programmatically navigating to a route

route in app-routing.module.ts

const appRoutes: Routes = [

{ path: '', redirectTo: '/recipes', pathMatch: 'full' },

{ path: 'recipes', component: RecipesComponent, children: [

  { path: '', component: RecipeStartComponent },

  { path: 'new', component: RecipeEditComponent },

  { path: ':id', component: RecipeDetailComponent },

  { path: ':id/edit', component: RecipeEditComponent }

] },

{ path: 'shopping-list', component: ShoppingListComponent },

];

Define a button click event in the component html

<div class="col-xs-12">

        <button class="btn btn-success" (click)="onNewRecipe()">New Recipe</button>

    </div>

define the code for route navigation in the component typescript module

constructor(private recipeService: RecipeService,

              private router: Router,

              private route: ActivatedRoute

    ) {

   }

onNewRecipe() {

        this.router.navigate(['new'], {relativeTo: this.route});

  }

Need the **Router** module from @angular/router to use the navigate method.

Need the **ActivatedRoute** module from the @angular/router to get a handle on the current route

Need the **Params** module from the @angular/router to get process the parameters on the route

Defines the path to go to relative to the current route

this.router.navigate(['new'], {relativeTo: this.route});

# Assignment 1

Create an angular app with three components:

* Home
* Devices
* Headends

Setup routes on a navbar , so that you can navigate from one component to another .

The page should not load when the router links are clicked.

Show data on the component html templates using:

string interpolation

set html element attributes using property binding

pass data to component typescript module using local references

using services as a in memory data store: ( for ex: service with an array of devices or headends )

Call the service using dependency injection in the component typescript module and show in component html template using structural directives ( \*ngFor )