ANGULAR

**Angular 7.0**

- It is a framework used to develop client side application made up of typescript

- used to develop single page application

**Why Angular?**

- Angular promotes modular approach and hence application we build will have a clear structure

- Each modules will contain different component(ts files) - we can have lot of reusable code

- Angular provides lots of inbuild features like validation, routing, forms etc

- Angular provides unit testing

Jasmine - is a library used to write unit test cases

Karma - is a test runner used to run test cases

- Google+Microsoft

Angular is maintained by Google, typescript is maintained Microsoft

**Creating Angular Project**

It is difficult to create project from the scratch, angular team creates a base project from angular/cli

>npm install -g @angular/cli

>ng new "projectname"

would to continue stricter - N

would to like to add angular routing - y

Style to add - CSS - U just enter

e2e - end to end testing - Protractor framework

node\_modules - all packages installed to develop angular application

src - app - we create our own components

package.json - contains all packages to build and run angular application

dependencies - which is essential for running our angular application

devDependencies - packages required only at the time of development and we don’t need these packages on our production environment

**Execute/Run angular application**

>npm start or >ng serve

we will launch the angular application in browser using http://localhost:4200/

>ng serve --open

- it will start application and automatically open the browser in http://localhost:4200

>ng serve --open --port 3000

- it will start application and automatically open the browser in <http://localhost:3000>

1. npm start or ng serve - used to start the application at development

2. npm build or ng build - used to start the application in production environment

3. npm test or ng test - used to launch Karma to test jasmine test cases

4. npm lint or ng lint -

Linting used to verify code quality, tslint.json

5. npm e2e or ng e2e - used to e2e testing using protractor

Whenever u create an angular project, by default it creates an App component which is ts file

Whenever we create a component in angular, it creates 4 files

1. app.component.html - design of the component

2. app.component.css - styles for html page

3. app.component.ts - typescript file used to write logic

4. app.component.spec.ts - contains Jasmine unit test case for that particular component

Whenever u start the application using ng serve, first it goes to angular.json from there it goes to main.ts which bootstrap the module called AppModule which bootstrap AppComponent.ts which invoke its related app.component.html, in order to identify this html we use selectors

Whenever you launch your application in browser using http://localhost:4200, by default it always open index.html

Selector - is a custom tag used to display the html file for that component

Decorator - anything starts with @

@NgModule - it will acts as module which contains collection of components

@Component - it is a typescript class which contains logic

export - if we want to use a class or interface in some other program then we use export keyword

template - In case if my html code is very small, we can use template

templateUrl - In case if my html code is very big, we can use templateUrl

styles - In case if my css code is very small, we can use styles

stylesUrl - In case if my css code is very big, we can use stylesUrl

**Interpolation / Data binding** - 3 types

- used to display the data using {{expr}}

1. one way binding - from component class to view

2. one way binding - from view to component class

3. two way binding - from component to view and view to component

All images has to be kept inside asset folder

**Property Binding**

- moves component class property to html element property

- using []

Property Binding for html attribute

- using [attr]

Event Binding

- event flows from html to component

- using ()

Angular Components

- u have to create component inside src/app/

- >ng g c "componentname" => ng generate component "componentname"

**Structural Directive**

- it is responsible for the structure and layout of DOM element

- used to hide or display the things in DOM

1. \*ngIf - to check single condition

2. \*ngFor - used for iteration

3. \*ngSwitch - to check multiple condition

**Angular Pipes**

- It is a typescript class used to display the data in different format

- using |

Build in pipes

1. uppercase

2. lowercase

3. currency

4. date - longDate,shortDate,mediumDate,fullDate,shortTime,longTime,mediumTime,fullTime

5. json - used to display in json format

6. decimal/number -

7. percent

8. slice

9. async pipe - to read asynchrouns data

Custom Pipe - used to create user defined pipe

- ng g p "pipename"

- Implement an interface called PipeTransform and override transform()

2 types of pipes

1. Pure pipe - default, works only when the component is loaded

-- is only called when angular detect a change in the value or parameters passed to pipe

2. Impure pipe - works for every change in the component

To create different prefix for the component, we use

> ng g c sample --prefix tech

**Angular Nested Component**

- One component is accessed inside another component

EmployeeListComponent is parent component, EmployeeCount is child component

@Input - Transfer the data from parent component to child component using property binding

@Output - Transfer the action from child to parent component

Whenever we click the radio button, we want to pass the user action from child to parent component

1. Create a property called selectedRadioButtonValue in employeecount component which keep track of value of radiobutton

selectedRadioButtonValue: string="All";

2. Create a custom event countRadionButtonSelectionChanged with @Output which used EventEmitter class. So whenever we select radiobutton, the value is passed to the event handler method

@Output()

countRadioButtonSelectionChanged: EventEmitter<string> = new EventEmitter<string>();

3. Create a function which raise our custom event which uses emit() to raise the event which takes the selectedRadioButtonValue which keep tack of value selected

4. To perform 2 way binding, we import FormsModule in app.module.ts using [(ngModel)]

5. whatever selectedRadioButtonValue from count component transfer to employeelist component, and we want to store that selected gender in a property called selectedEmployeeCountRadioButton

We can’t write 2 structural directives on the same element, instead we have to go for <ng-container>

**Life cycle Hooks**

Whenever Angular creates, destroys or render something in the component internally it calls some methods called as life cycle hook method

1. constructor

- constructor will at execute at booting time, only once and used for initialization

2. ngOnChanges

- after constructor immediately priority goes to ngOnChanges

- Whenever change is detected in @Input property automatically invokes ngOnChanges

3. ngOnInit

- after ngOnChanges priority goes to ngOnInit

- called as first life cycle hook, used to write some business logic

4. ngDoCheck

- whenever change is detected in component properties

- after ngOnInit automatically it invokes ngDoCheck

5. ngAfterContentInit

- whenever framework identifies the memory for the component, only once

6. ngAfterContentChecked

- whenever particular memory reserved for component

7. ngAfterViewInit

- whenever the component successfully loaded into that memory, only once

8. ngAfterViewChecked

- Whenever data populated successfully from the component

9. ngOnDestroy

- before killing component by destroy

**To use bootstrap**

1. we have install bootstrap package

> npm install bootstrap@4 jquery --save

2. configure js packages in angular.json

"scripts": [

"./node\_modules/jquery/dist/jquery.min.js",

"./node\_modules/bootstrap/dist/js/bootstrap.min.js"

],

3. configure css in styles.css

@import "~bootstrap/dist/css/bootstrap.css";

**Angular Routing**

- used to develop Single Page application(i.e.) multiple pages on single view

1. In order to do routing, all the routing configuration has to be done inside app-routing.module.ts using const routes[]

the const route holds an array because it contains multiple routes.

Each route is JS object which contains path and its related component

const routes: Routes = [

{ path: '', component: HomeComponent },

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

{ path: 'servers', component: ServersComponent }

];

Where we want to load our component when we click the hyperlink using <router-outlet> directive

2. [routerLink] - we can provide the link

By using href, each time it contact the server to do processing,

3. Whenever we click each tab, it is active with the help of css class="active" instead routes provide a directive called "routerLinkActive" which has bootstrap class called active

But always home is active,

home=/, servers=/servers, users=/users

Empty path is part of all path, so we dont want home to marked active in all time. So we use "routerLinkActiveOptions" which is JS object with one configuration called "exact:true", which tells angular if exact full path is matching it makes to be active

4. If we want to do the routing programmatically (i.e.) through components, then we have to go for navigate() method

5.Passing parameter in the url using :

const appRoutes: Routes = [

{ path: '', component: HomeComponent },

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

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

{ path: 'servers', component: ServersComponent }

];

/users/1/Ram

6. Fetching the parameter from url and print in UserComponent using ActivatedRoute class which contains snapshot property

7.Redirecting

- If we give some wrong path, then it has to redirect to some common component

const appRoutes: Routes = [

{ path: '', component: HomeComponent },

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

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

{ path: 'servers', component: ServersComponent },

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

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

];

8. Passing query parameter (i.e.) any parameter transferred between ? and &

fragment - anything transfer with #

const appRoutes: Routes = [

{ path: '', component: HomeComponent },

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

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

{ path: 'servers', component: ServersComponent },

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

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

{ path: '\*\*', redirectTo: '/not-found'} //only in last line

];

9. Retreive query parameter and fragment using ActivatedRoute

constructor(private route:ActivatedRoute){}

ngOnInit(){

console.log(this.route.snapshot.queryParams);

console.log(this.route.snapshot.fragment);

}

ng g c department-overview

ng g c department-contact

children - To add child routes

relativeTo - used to maintain parent path

pathMatch- Whenever we redirect we have to include pathMatch- 2 values

1. Full - only when entire url is blank then redirect to /departments

2. prefix -- applicationicationy the redirect if prefix of url begins with empty path

{path:' ',redirectTo:'/departments',pathMatch:'prefix'}

/employees

**Angular Forms**

HTML template is used to collect data and that data will be transferred to the component class and from component class the collected data will be sent to server through services

Angular HTML - Component - Services - Server(Spring boot applicationication)

Angular forms - used to get input from user - 2 types

1. Template driven form

2. Reactive/model driven form

**1. Template driven form**

- Much of code and logic resides in HTML page

- Heavily relay on two way binding using [(ngModel)] directive, so whatever value given in textbox, we can get in component class

- Angular provides ngForm directive to track the entire form information

-Bulky HTML code and minimum component code

Drawbacks

- Doesn’t allow unit testing because much of code are written in HTML, only way to test the logic to run e2e testing

- When it comes in handling complex forms

1. Adding basic HTML form

- create basic html form with name, email, phone textbox, select option, radio button, checkbox and submit button with bootstrap classes

2. Binding data with ngForm

- import FormsModule in app.module.ts

- To track the entire form we use ngForm directive with reference variable name (i.e.) userform

- To track each form control we use ngModel, along with ngModel we have to use name attribute

3. Binding data to the model class

- When u create a form, we have to create model class related to the property of the form

- We create the object of user class

- we want to bind the user object to the form using two way binding using [(ngModel)]

4. Form Validation - Angular provides 3 classes

1. ng-untouched - When u load the form for first time, u have not yet visited any form fields, then angular application ng-untouched

ng-touched - If u visit the form control by clicking, then angular application ng-touched

2. ng-pristine - when u load the page for first time,if we not change the value of form controls, then angular application ng-pristine

ng-dirty - if we change the value of form controls, then angular application ng-dirty

3.ng-valid - if all form control values are valid, then angular application ng-valid

ng-invalid - if all form control values are not valid, then angular application ng-invalid

In order to use this classes, we can use ngModel which provided with properties like untouched,touched,pristine,dirty,valid,invalid

By creating reference variable for ngModel, we can access those properties

5. Display the validation error message

- Bootstrap4 provides class called is-invalid to check the form-control is invalid

- Bootstrap provides class called d-none (do not show error message)

1. Name field - required

6. Submitting the form

- For submit button u have to provide an event called (ngSubmit)

**2. Reactive/Model driven form**

- much code and logic are in component class

- No two way binding instead we need to react to user inputs to update the value. So Angular provides methods to update the form control values from component class

- Reactive forms are well suited for complex scenario

- We can generate dynamic form fields

- Inbuild validation

- Custom validation/cross field validation - password and confirm password

- Reactive form allows unit testing since more logic are written in component class

Steps

1. Adding basic html form using bootstrap class
2. Import the module ReactiveFormsModule in app.module.ts
3. The form is represent as model in component class using FormGroup (represent entire form it is like ngForm) and FormControl (represent each form control it is like ngModel) class
4. Associate this model in component class with the html form using directives called formGroup and formControlName, we can communicate between component to html
5. Setting values programmatically without user interaction (ie) through component using setValue(). If we want to populate the value only for few form controls in that case we use patchValue()
6. By creating multiple FormControl instance manually can be repeative. To avoid this angular provides "FormBuilder" class which provides methods to generate form controls
7. Inject FormBuilder in ur constructor
8. FormBuilder provides group() to create different form controls
9. Unlike template driven forms, in reactive forms all the validation rules are applicationicationied in the component class, by using Validators class, we used Validators.required, Validators.minLength(4)
10. Cross field validation between password and confirmpassword

- create a function PasswordValidator which takes form control as argument and returns either JS object or null

- AbstractControl which returns the form control

1. Dynamic form fields

- In Reactive, we use FormArray class to create dynamic form fields

1. Define alternateEmails as FormArray in form model with any number of controls

2. Create the getter method to access alternateEmails

3. create a method called addAlternateEmail() which push new FormControl into FormArray

4. Add a button in HTML to call addAlternateEmail()

HtML (to get input from user) - Component class(data from html goes to component) - service (communicate with server with the data)

Angular Service

- It is injectable class which is used to communicate with the server using HttpClient class present HttpClientModule

- ng g s "servicename"

- @Injectable indicates it is service