**https://github.com/Kaleakash/tcsangulartraining.git**

1. Template Reference

<input #uname type=”text” name=”uname” id=”uname” >

App- component 🡪 parent

Template –ref 🡪 child component

1. Angular Forms
   1. Using Template Driven Form
   2. Using Model Driven Form or Reactive Form
   3. With Validation

#userRef="ngForm"

ngForm is a pre-defined attribute which help to create the reference of form. ngForm attribute is a part of FormsModule

that we have to import in app.module.ts file

HTML5 : 4 properties for validation

required

minlength

maxlength

pattern

1. Angular Service
2. Dependency injection
3. Call REST API (Fake REST API or Using JSON Server we create simple REST API).

ng new angularformswithservice

routing -🡪 yes

style –>css

Angular Service : Service is use to write the business logic. Using Service only we can achieve separation of concern.

Angular Service mainly divided into two types

1. User – defined service
   1. Creating object explicitly
   2. Creating object using DI(Dependency Injection)
2. Pre-defined service

Angular 1.x $http

Angular 2 to 4.2 Http

Angular 4.2 onward HttpClient

HttpClient API is a part of HttClientModule we have to import in

­­app.module.ts file

Template Driven Form Validation code

**<form #formObj="ngForm“ (ngSubmit)="verify(formObj.value)“ nonvalidate>**

**<input type="text" name="user" ngModel required #userRef="ngModel” minlength="2"/>**

**<div \*ngIf="userRef.errors && (userRef.dirty || userRef.touched)">**

**<span [hidden]="!userRef.errors.required">**

**UserName is required**

**</span>**

**<span [hidden]="!userRef.errors.minlength">**

**UserName must be 2 character**

**</span>**

**</div>**

**<input type="submit" [disabled]="!formObj.valid">**

**Model Driven Form Code**

<form [formGroup]="loginForm" (submit)="verify()" nonvalidate>

<input type="text" formControlName="user" required/>

<div \*ngIf="!loginForm.controls.user?.valid && ( loginForm.controls.user?.dirty

|| loginForm.controls.user?.touched)">

<div [hidden]="!loginForm.controls.user.errors.required">

UserName is required

</div>

<div [hidden]="!loginForm.controls.user.errors.minlength">

Min Length must be 2 character

</div>

</div>

<input type="submit" value="submit" [disabled]="!loginForm.valid">

Node JS :

MEAN Stack

Mongo DB

Express Module

Angular Framework

Node JS

Angular Framework 🡪 Retrieve the Product Details from Database

Html --🡪 component --🡪 service --🡪 HttpClient

🡨--Fake Service or json-server

Mongo DB : No SQL Database key-value pairs like a JSON.

Node JS db module which help to CRUD Operation (Create, Read, Update and Delete) on MongoDB.

Express Module : Which help to Create REST API using JavaScript.

Html --🡪 component --🡪user defined service --🡪 HttpClient

Get/post/delete/update --🡪

Express module --🡪 db module ---🡪 Mongo DB

Today Session Plan

http module

express module or framework (third party)

using express module get(), post(), put() and delete()

middleware application.

Get(), post(), put() and Delete()

Client application may browser only for get

Post() : form tag (normal html page)

Delete() and put() : Browser RESTClient plugin.

http or https

req(http/https)---🡪

header

body

Client Server

http:

Get() : Get Resources : Employee Details, Product Details

GetEmployee ById GetProjectById

Post()

: Create Resource : Store Employee Details,

Product Details

Put() : Update the Resource,Employee Salary using Id

Update price using pid

Delete(); : Delete Resource by Id or any property

From Client if you want to pass the data to server using get method

1. Using Query Param
2. Using path param

Query Param

<http://localhost:9090?key1=value&key2=value&key3=value>

<form action=” <http://localhost:9090>” method=”get”>

UserName :<input type=”text” name=”uname”><br/>

<input type=”submit” value=”submit”/>

</form>

Path Param :

<http://localhost:9090/1/Raj/12000>

<http://localhost:9090/a>

<http://localhost:9090/b>

<http://localhost:9090/c>

Client Unix or Git or command base Client

<http://localhost:9090/user/Raj/Deep>

Through browser we can call only get method but post, put and delete.

Through html form we can call get or post with the help of submit button but not put and delete.

Postman client or Chrome REST Plugin

We can test get, post, put or delete.

To check all four method

1. Ajax call
2. Angular
3. React JS
4. Spring boot or JAX\_Rs
5. .net Rest API
6. Any REST client program

Get : Get All Employee Details, GetEmployee By Id

Post : Store Employee Records

Put : Update Employee (Salary, Age, Desg using ID)

Delete : Delete Employee using ID

Insert : CustId,CustName,Age,Phnumber

Update : update Age using Custid

Update Phnumber using CustId

Update Age and Phnumber using CustId

path

**Mongo DB**

<https://www.mongodb.com/try/download/community>

MonoDB :

Input

Process

Output

File base system

Limitation :

Data is not secure

Data Redundancy(Duplicate Records)

DataBase :

MySQL

Oracle

SQL Server

Db2

Table format.

Database :

RDBMS : Schema : Logical entity

Table

C1 C2 C3 C4

Datatype Datatype DataType DataType

Employee

Id Name Salary Age

Number(10) varchar2(10) number(10,2)

MongoDB :

Documents :

MySQL :

Show databases;

Database : collection of tables.

create database databaseName;

use databaseName; switch the database.

Table :

show tables;

MongoDB

show databases;

Or

show dbs;

use databaseName; it is use to create as well as switch to that database.

In MongoDb Table is known as Collection

Show collections;

Insert the records in Collection

db.CollectionName.insert({property:value,property:value});

In Mongo DB Record is known as document.

Retrieve documents from Collection

db.Collection.find();

db.Emp.find().pretty() : display the record in proper format.

db.Emp.find()[indexPosition]: Display specific documents.

db.Emp.find().forEach(printjson):

db.Emp.find()[indexPosition].property

Relationship

Primary key -🡪 Foreign Key

One – to – One

One – to –Many

Mongo DB we can achieve relationship using two ways

1. Embedded Relationship : Within one collection we can write another collection.
2. Linking Relationship

Embedded Relationship

One – to – one using

Person/Employee -🡪 Address

One – to – Many

Person /Employee 🡪 More than one address

\_id : 100, name:”Ravi”,age:21,add:{city:””,”state”}

Linking Relationship

Trainer 1

\_id tName tech

1 Raj Java

2 Raju .net

3 Ramesh Angular

Student many

\_id Sname Age ts\_id (FK)

100 Seeta 21 1

101 Reeta 22 1

102 Meeta 23 2

103 Teeta 24 2

104 Keeta 25 3

105 Leeta 26 3