Phase 4 :

8 classes

Day 1

Testing and deployment

Testing for UI Technologies

Testing Using Jasmine and Karma

Unit Testing with Angular

Grunt : Java Script Task runner

Docker

With Jenkin

Overview of cloud and AWS (EC2 and S3 service)

Day 1

Read a, a 10, 20

Compute sum = a+a 10-20

Write a -10

Testing is use to find the defect or error or bugs in the application.

Unit Testing : Unit testing is use to test the function functionality working or not.

It is a type of white box testing.

Jasmine : Jasmine is a open source framework which provided set of function or API which help to do the unit testing for Client side as well as server side JavaScript code.

Test suite : Test suite is a like container which hold more than one test cases as well as another test suite.

To write the test suite all testing framework provided describe function .

Syntax

describe(“Msg”,callback);

Test case : Test case is use to test the function functionality. To write the test case jasmine provide pre defined function ie it

Syntax

It(“Msg”,callback);

Expectation function : Jasmine framework provided set of expect function which help t check actual and expected output .

expect(expectedOutput).toXXX(actualOutput);

describe(“Msg”,()=> {

it(“Msg”,()=> {

more than one expect

})

It(“msg”,()=> {

More than one expect

})

})

Day 2

Jasmine Provided some hook or life cycle function which will call automatically.

beforeAll(()=> {

it will call before it method : it will call only once

})

beforeEach(()=> {

it will call before each it function. This will call again and again.

})

afterEach(()=> {

it will call after each it function. This will call again and again

})

afterAll(()=> {

it will call after all it function. It will call only once.

})

Jasmine version 4.x version

Jasmine 3.x : Jasmine only provide set of function or api to do testing. Jasmine didn’t provide runner run the application.

Karma :Karma is known a test runner to run Jasmine or any other testing framework.

**Client side scripting testing using Node JS**

Using npm init command create the package.json file

jasmine-core it provide function for testing ie describe, it and more than except

jasmine-browser-runner : it is responsible to run the application on browser.

npm install jasmine-browser-runner jasmine-core -D

**or**

npm install jasmine-browser-runner jasmine-core --save-dev

or

npm install jasmine-browser-runner –D

npm install jasmine-core –D

npx jasmine-browser-runner init

npm is use to install the module

npx is use to execute the module

yarn

To run the jasmine through node js have to execute the command as

Jasmine-browser-runner serve

Or

Npx jasmine-browser-runner serve

Then create the src folder and write all Javacript files inside a folder.

function checkUser(name,pass){

    if(name=="Raj" && pass=="123"){

        return true;

    }else {

        return false;

    }

}

Then create spec files inside spec folder

describe("Login Operation Testing",()=> {

    it("Verification Testing",()=> {

        var result = checkUser("Raj",123);

        expect(result).toBeTrue();

    })

})

Then run the command as

npx jasmine-browser-runner serve : default port number is 8888

npx jasmine-browser-runner server –port=8989 : running on port number 8989

Angular framework internally provided configuration for jasmine testing framework.

Angular internally configure Karma test runner to run the jasmine testing test case on browser.

Angular framework provided Angular utility which help to test angular specific classes.

TestBed is a pre-defined API provided by angular which help to do the testing for angular programs.

describe, it, hook function and except function are same in client side JS, Server side JS and Angular framework.

Create the new project

ng new angular-testing-app

Day 3

16-04-2022

Mock test or fake data testing or proxy or dummy data providing while testing.

When we developing layer architecture application every layer depends upon the another layer.

Component layer depends upon the user – defined service

User-defined service depends upon the http service

http service depends upon backend technologies service.

MVC in backend technologies develop using express js

Controller layer depends upon the repository (database connectivity).

User- defined service create fake object of HttpClient service.

If you call this url for get, post, put and delete you will get this output.

If we want to achieve mock for HttpClient API we have to do the DI for HttpTestingController API.

to check the testing coverage we have to run the below command.

ng test --no-watch --code-coverage

or

ng test –code-coverage

Testing for backend

Node Js testing

Create the folder : Server Side JS testing – Node JS

Then create the package.json file using npm init

Then install two dependencies

npm install jasmine –D

npm install jasmine-node -D

npx jasmine init : This command is use to create the spec folder

**Operation.js**

function add(a,b){

    var sum = a+b;

    return sum;

}

module.exports = {add};

**OperationSpec.js**

var obj = require("../src/Operation");

describe("Operation Testing ",()=> {

    it("Addition testing ",()=> {

        var result = obj.add(10,20);

        expect(30).toBe(result);

    })

})

To run the test case we have to run the command as

npx jasmine

Or

jasmine (but before this commands you have to install jasmine using npm command

Ie npm install jasmine –g

17-04-2022

Testing for Express JS using Jasmine

If we want to test the REST API or http protocol methods then you can use the supertest in one the third party library with Jasmine.

Create the folder : Express JS testing with jasmine

Create the package.json file using npm init command.

Then install two dependencies

npm install express

npm install jasmine –D

npm install jasmine-node -D

npm install supertest -D

npx jasmine init : This command is use to create the spec folder

Docker : Docker is an advanced OS Virtualization software platform that makes it easier to create, deploy and run the application in a Docker Container.

Virtualization : Virtualization help to run the application or software in Virtual machine. It will create the virtual version of our resources like a server, database or application/tools.

Virtualization let us divided a system into a series of separate section, and each one acting as a distinct individual system. That environment is known as virtual machine.

16 GB RAM

VM 🡪 Cent OS : 4 GB 50 GB hard disk

10 VM os we want to run

VM -🡪 2GB

Docker : Docker is Containerization

Virtualization is a abstract version of physical machine or device.

Containerization is the abstract version of an application.

Docker container : Docker container is also known as Docker engine or run time environment.

Running instances of Docker images turn or run the actual application. A container includes an application and of its all dependencies.

Container are deployed application bundled with all necessary dependencies and configuration files.

A Docker container “contains” everything which help to run our application in any environment. But that environment must be present Docker engine.

Docker Image : The file system and configuration of our application which are used to create the container or Docker images are the source code for our container.

DockerFile : A Docker file is a blue print or set of instruction that defined how our images must be build.

Application up or running

Container

Image

File (Docker file )

Docker Registry : Docker registry help to store the images as well as publish the our images.

Docker hub : Docker hub is like a Git hub. Which help to store the images in their registries.

In Docker hub we can create two types of registered one is private and more than one public.

After installation successfully please check the

docker --version : this command display the Docker version

sudo docker –-version : in Virtual Lab

docker images

docker pull hello-world : This command is use to pull the image from docker hub

docker run hello-world

hello-world : running C application

busybox : tiny unix os

alpine : linux os

We will create our own method to run the application.

We want to create the image to run the application we have to take the help OS image like

Busybox or alpine in that image installed necessary software and run the application.

To create the image we have to take the help of Docker file. The file name must be Dockerfile without extension.

Create the folder

Simple image to display message

Dockerfile

FROM busybox:latest

CMD ["echo","Welcome to Simple Docker Image"]

docker build –t my-busybox-111 . –f Dockerfile

docker run imagename

Image to run the node js program

Create the folder – image to run node js program

Simple.js file

console.log("Welcome simple node js program using docker")

let a =10;

let b =20;

let sum = a+b;

console.log("Sum of two number is "+sum);

function sayHello(name){

    return"Welcome "+name;

}

console.log(sayHello("Ravi"));

**Dockerfile**

FROM node:17-alpine

COPY simple.js .

CMD ["node","simple.js"]

docker build –t my-node-111 . –f Dockerfile

docker run my-node-111

image to run the Express JS application

create the package.json file using npm init command

**npm install express**

**app.js**

let express = require("express");

let app = express();

let employees = [

    {id:100,name:"Ravi",age:21},

    {id:101,name:"Ramesh",age:22},

    {id:102,name:"Rajesh",age:23},

]

app.get("/",(req,res)=> {

    res.send("Welcome to Simple Express JS runnning through docker")

})

app.get("/user/:name",(req,res)=> {

    let name = req.params.name;

    res.send("Welcome user to express with docker "+name)

})

app.get("/employees",(req,res)=> {

    res.json(employees);

})

app.listen(9090,()=>console.log("Server running on port number 990"))

**Dockerfile**

# pull the image

FROM node:latest

# create the directory or folder in os

RUN mkdir /usr/src/app

# point OR refer to working directory

WORKDIR /usr/src/app

# copy package.json file in working directory

COPY package.json /usr/src/app

# install all depencies in os image it will create node\_module folder base upon dependencies provided in package.json file

RUN npm install

# copy app.js file in os image

COPY app.js /usr/src/app

# run the application

CMD ["node","app.js" ]

**docker build –t my-express-111 . –f Dockerfile**

if image contains to run the web application we have to expos the port

docker run –p 9090:9090 imageName

right side 9090 is actual port number

left side 9090 is expos port number, expose port number can be same or different.

docker run –p 9191:9090 imageName

docker run –p 9292:9090 imageName

docker run –d –p 9393:9090 imageName/imageId

To remove the images

docker rmi imageName/imageId

To check the running container : Every image like with one container.

docker ps This command is use to provide running container details

docker ps –a This command display all container available in our machine.

We can stop and star the container

docker start containerId

docker stop containerid

remove the container

doker rm containerId

docker rm containerId –f

24-04-2022

To deploy the web application on Docker environment we are using one of the open source server

Ie nginx

First create the html, css and JS pages

index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        h2{

            background-color: coral;

            color: darkgray;

            font-size: 30pt;

        }

    </style>

</head>

<body>

    <h2>Welcome to Simple Web Application deploy using docker</h2>

</body>

</html>

Dockerfile

FROM nginx

COPY index.html /usr/share/nginx/html

Create the image for web application using nginx server

docker build -t my-webapp-111 . -f Dockerfile

nginx default port number is 80

docker run -d -p 80:80 my-webapp-111

Creating image for angular project

ng new angular-docker-app

routing -🡪 no

styling 🡪 css

Once development done you have to build the project using command as

ng build

Dockerfile

FROM nginx

COPY ./dist/angular-docker-app/ /usr/share/nginx/html

docker build -t my-angular-111 . -f Dockerfile

Now we have to publish the image in Dockerhub. Anyone can pull the image and run the application in their machine.

Before push we have to create the tag for that image. Tag is just like a identity for that image.

docker tag imageName dockerhubaccountId/imageName:version

Or

docker tag imageName dockerhubaccountId/imageName:latest

docker tag my-angular-111 akashkale/my-angular-111:1.0

docker push dockerhubaccontid/imageName:1.0

Docker compose : it is use to run the multiple container using one file.

Docker compose is the toolkit provided by Docker to build, ship and run multiple container application.

To run the multiple container using Docker compose we have to create the docker compose file with extension .yml

**docker-composet.yml**

version: '3.1'

services:

  my\_image:

    image: akashkale/my-angular-111:1.0

    ports:

      - "8181:80"

  lalit-image:

    image: bewithlalit/docker101tutorial:latest

    ports:

      - "8282:80"

docker-compose up

CI and CD tool : Continuous integration and continues delivery or deployment

Manager ---- Project Skeleton : it push into remote repository.

Dev1 pull code push the code

Login page

Dev2 pull code push the code remote

Application page

Dev3 pull code push the code

Feedback

Jenkin : Jenkin is a open source automation server tool base upon the Java Technologies.

In Jenkin we can add the plugin base upon our requirement.



Grunt :

While developing the any application we have to do lot of task manually like CSS file compression. Unit testing, converting ts to js, build deployment etc.

Grunt is known as JavaScript task runner.

Webpack is one of the type of task runner.

If we want to execute any task through grunt we have to install the grunt cli.

Create the project folder.

npm install –g grunt-cli : this dependencies install globally which help to run the

run task runner.

first create the package.json file using the command as npm init

To configure the task runner we require grunt module

npm install grunt -D

or

npm install grunt --save-dev

to configure all task in grunt we have to create the Gruntfile

create the folder with name

GruntProjectWithPre-definedTask

Create the package.json file

npm install grunt –D

JavaScript minification ?

JS Minification also known as minimization. It is a process of removing all unnecessary files, or character or code from a JavaScript, html, css without altering source code.

Grunt provided lot pre-defined modules to do minification for the JS project.

npm install grunt-contrib-clean –D

npm install grunt-contrib-copy -D

npm install grunt-contrib-cssmin –D