9-8-2024

Developer, Tester,ProjectManager

Repository – Project

Tuple – Relation

Merged with master --- Continuous Integration -- gitlab runner,Jenkins

Git clone -u <url> -

Drive –

Rootfolder /etc,/dev

Git - Version Management – Source code management

Gitlab – collaberative development platform / source code management server

To add files to the staging area- git add <filename>

Eg:git add sample.txt

Push changes from local repository to remote repository

Git push <url> <branch>

To understand current locally

Git branch-

Alias name – short name assigned to remote repo url

Git remote add <aliasname> <url>

Eg:git remote add Pankaj <http://gilab.stackroute.in/anil/actalent.git>

git pull -- to pull the changes from remote repo to local repo

git pull <url> <branch>

eg:git pull anil master

to create a new branch locally and change to that

git checkout -b developer1

git init—command to convert an existing folder to git folder

agile -

scrum -- , Kanban,

Product backlog->

CustomerManagement->sprints-> sprint team->sprint backlog->

Order Management->

Scrum Master ->

sprints->

10-8-2024

================================

flow chart,psedocode,

algorithm

start

step1:

stop

graphical representation of algorithm

psedocode -

Programming --- procedure oriented – cobol, basic

Function oriented - c,

object oriented, component oriented

class stock{

int stockCode;

void AddStock(){

int a;

}

Void ShowStock(){}

} //encapsulation

Int main(){

Customer ptr = new Customer(); --- ptr is implicit pointer

Ptr.RegisterCustomer();

stock s = new stokc();

} -- stand alone functions

class Customer {

char name[100];

Void RegisterCustomer(){}

}

Sample.c ->compilation-obj file – machine code-> link+make ->exe file

Obj 100kb- exe -200mb --

St – memory management- pointers to

Encapsulation, abstraction,

Platform independent –

Java runtime environment –

.jar -

12-8-2024

OOA –object oriented analyses --

Identity the data

Identity the process

Identity related data and processs -- consider it as an object

Object is implemented by – class,struct,record,enum,union etc

Stand alone functions -- functions that does not use the global data of any object

Java – pure object oriented -- object/class definitions

Stand alone functions in java are called static

Make, link – combine object files to a single deployable unit . exe

-jar - runtime --

Java – packages – collection of related classes

To create package

Javac -d <destination> <source>

Eg : javac -d . \*.java

To create java documentation

Javadoc -d <destination> <source>

System.out.println();

Building b = new Building();

Building b = 10; - strongly typed language -- truncation of data

String name = new String(“anil”);

String name = “anil”; -- boxing --new String(“anil”); --- auto boxing

Int x=50;

Integer a =20;

System.out.println(name); -unboxing – auto unboxing

float rate = (float)20.5;

variable declaration statement

assignment statement --- syntax

arithmentic statement

decicision making statement

function calling

loop statements

relational statements

input statement

output statement

etc

in java all statements are written inside a function and declaration statements can be inside class also

primitive -- extended/userdefined/complex/composite

byte,short,int,long,float,double,Boolean,char - u

int a; --- value types

String s; --- reference -- 4 byte,8 byte

Operators - arithmethic,relational,logical operators,arithmetic assignment,bitwise operator ,ternary

1010

0010

1000

Loop – exit controlled,

Int count =0;

Do{

System.out.println(count);

Count++;

}while(count<10);

While(){}

For(int count=0;count<10;c++){

}

For(int x in [1,2,3,4]){

X=10;

}

Write a program to display multiplication table of number 5

Arrays - group of variable

Single dimensional, multidimenstiona

If(s==s1)

Int [][]arr[] = new int[2][2][2] ;

Int[][]marks=new int[10][5];

Arrays reference int []xx = null;

findAverage(null);

call by reference and value

Array or class type call reference

Void FindAverage(int []input){

}

FindAverage(null);

Array –

* RealEstate domain -
* Property – Flat,House,Land,Plantation,Office,CommercialSpace etc
* Area,saleType,Location,Rate
* Flat – floor,deposit
* House – bhk,isparking,deposit
* Office –
* Commercial space –
* Relationship – kind of, has a, uses
* Inheritance, composition ,utilization
* DRY – do not repeat yourself
* SOLID –
* Class Driver{
* Public void Driving(){
* C.switchon();
* C.gearchange();
* C.accleration();
* }
* Public void eating(){}
* }
* TS 01 AA 1001 – is a maruthi alto car
* Class servicestation{
* Doservice(Vehicle c){}
* }
* Class vehicle{}
* Class car extends vehicle{}

Servicestation().Doservice(new Car());

Property f = new Flat();

f.accept(); --- binding --- associating function to a class

compile time -- static binding --compiler

runtime --- dynamic binding -- runtime

abstract methods – if used bottom up approach it used for dynamic binding

top down approach – the abstract method will act as a contract

www ,ieee –

interfaces -- to create specification of kind of classes,

interface Vehicl{ void acceleration();void braking();void gearchange();}

UML—JUDE

class BMW implements Vehicle {

}

Interfaces will to implement dynamic binding –

Interfaces are used future referencing – it is a way to allow user of a package class to provide implementation of one or more functionality which will be consumed by the provider.

Interfaces can be used higher level of abstraction --

Abstraction – provide relevant information from users perspective

Same class method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Access specificer | sameclass | Anotherclass in same package | Child class in samepackae | Another class in another package | Child class in another package |
| public | Yes | yes | Yes | yes | yes |
| friendly | yes | yes | yes | no |  |
| protected | yes | Yes | yes | no |  |
| private | yes | no | no | no | No |

another class ,another class method in the same package

another class or method in another package

class Atm implements Customer,Employee{

private Atm(){}

public void checkBalance(){}

public void setDenomination(){}

public static Atm getInstance(){

return new Atm();

}

}

Public interface Employee{ public void setDenomination();}

Public interface Customer{ public void checkBalance();}

Employee e = Atm.getIntance();//new Atm();

Customer c = Atm.getInstance();//new Atm();

e.setDenomination();

c.checkBalnace();

component- beans

1. Introspection --
2. Customization
3. Interaction -

JFrame jf=new JFrame();

Jf.setLocation(12,12);

Jf.setBounds(12,12,300,300);

Bank – library -- package.json{

}

Pom .xml --

Class Ujjwala{

Ujjwala(int x){}

}

Class Bhumika extends Ujjwala{

}

Bhumika b = new Bhumika(); //compilation error

Base class reference variable can store address of child class object

=============================

14-8-2024

Datastructures – shape of data in memory

Arrays - static fixed -- collection --

Dynamic and elastic -

Traversal,add,remove ,show

Exceptions - program will terminate –

Try{} catch{} /finally{}

Persistence -- storing data in to permanent storage device like disk

Java.io;

Txt,binary files

State of an object /instance --- data stored in the instance variables at a particular time

Serialization - --

Text -- InputStream ->ByteInputStream,DataInputStream,FileInputStream,

OutputStream->FileOutputStream

FileReader,FileWriter –

Interface WindowListener{

Void windowClosing(WindowEvent e);

Void windowClosed(WindowEvent e);

Void windowMaximised(windoeweven t);

}

Class windowHandler extends WindowAdaper{

}

mnuOpen.addActionListener(new ActionListner(){

public void actionPerformed(ActionEvent e){

mnuOpenActionPerformed(e);

}

);

16-8-2024

JButton b = new JButton(); - developer A - ActionListener – actionPerformed(ActionEvent e);

b.addActionListener(ActionListenr e){}

b.setLabel(“Save”); // Developer B

FileI

FileReader,FileWriter – string,char

F FileReader(“abc.txt”);

Scanner s = new Scanner(f);

s.nextLine();

fw= new FileWriter(“abc.txt”);

fw.write(“hello world”);

fw.flush();

fw.close();

datastructure -- shape tree,stack,list,binary tree, array,queue

restaurant class –

generic class --- a class which act as template from which new classes and instances can be created

collection framework :- is a set of classes help us to deal with datastructure

Collection ->

List,Map,Set

Interface list implements Collection

ArrayList implements List

Collection c = new ArrayList();

c.

S - single responsibility principle

O – open close principle

L – liskov substitution principle

I - Interface segregation principle

D - dependency inversion principle

Class Quboid{

calculateVolume(){}

}

Class Sphere{

calculateVolumne(){}

}

17-8-2024

Unit – testing --

Testing library --- testing tool

Junit --- JunitRunner - java

Chai --- mocha -- javascript

Jasmine --- karma - angular

NUnit,XUnit,MStest - Testexplorer - dotnet

Databases –

What is a database - organized collection data -- store new data, modify existing data, retrieve data, remove data

Manual – computerized

Filecabinet -- disk files --- DBMS -- Foxpro, dbase,clipper (single user) --- Dr.AstonTate

Human

Dr EF Codd -- Relational Theory/principles /rules 1971 – mathematical programmer in IBM

Larry Ellison – Oracle - RDBMS – null -- Peter Chenn – ER Diagram -

ENIAC- ESDAC ---

RDBMS – software must minimum two tier architecture

Client server—thin client – fat server, fat client - thin server

Relational principles & two tier architecture -- Oracle, MSSQLServer, MySql,Postgres, Ingres,Informix,Pointbase, db/2 etc

Data can be organized in the form of two dimensional arrays called tables (relation)

Rows (tuple),column(attribute), degree, key attribute, domain(group of values)

Structure of table is called schema

Database is a collection of datafiles and log files –

create database actalent

on(

name='actalentdata',

filename='D:\Stackroute\Actalent\data\actalent.mdf',

size = 10,

filegrowth=2,

maxsize=1024

)

log on(

name='actalentlog',

filename='D:\Stackroute\Actalent\data\actalent\_log.ldf',

size = 8,

filegrowth=1,

maxsize=100)

alter database actalent modify file(name='actalentdata',size=12)

alter database actalent add file(

name='actalentdata1',

filename='D:\Stackroute\Actalent\data\actalent1.mdf',

size = 8,

filegrowth=1,

maxsize=500)

alter database actalent remove file actalentdata1

SQL commands – dml,ddl,dcl,tcl,select

Datatypes in sql server

Numeric- tinyint,shortint,int,bigint – float,double,numeric()

Character

Date

Binary

Create table employee(id int,name varchar(40), gender varchar(7) check(gender in(‘male’,’female’,’m’,’f’,’Female’,’Male’,’M’,’F’))

,age tinyint check(age between 1 and 150)

Insert into employee values(1,’dfdf’,’dfd,’200) – integrity -- correctness

Insert into employee values(1,’anil’,’dfd,’200)

Constraints -- rule imposed on column

1.primary key – null, duplicate

2.unique -- duplicate

3.not null \*

4.check – pattern,domain,range %\_

5.foreign key -- refer data in another column- referential integrity

6. user defined –

Table level and column level

Table level means constraint applied after the data type of column during the create table,column level means constraint applied after all columns definition during create table

[constraint <contstraintname> ] <constraint type> [<specification>]

create table visitor(

slno int ,

name varchar(50) not null,

purpose varchar(40),

persontomeet varchar(50),

logintime datetime default getdate(),

phoneno char(10) ,

check(phoneno like '[6-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

)

alter table visitor alter column name varchar(40)

alter table visitor add constraint pkid primary key(slno)

19-8-2024

Composite primary key - cannot be applied in column level

System database --- master,model,tempdb,msdb

Model – any object created in this db will be copied to all database

User databases

Create table vehicle (regno char(10) primary key ,chasisno int unique not null, engineno char(16)unique not null)

Candidate – column which can be selected as the key

Regno – key

Chasisno and engineno are alternate key

|  |  |
| --- | --- |
| key | offset |
| a | 1,4,5,,7 |
| b | 3,4 |
| c | 66,90, |

Index -- clustered index, non clustered

Clustered index stores records –

Empid name designation salary projectid projectname startdate duration

1001 ajay teamlead 45000 p1 skyview 12-3-24 100

1001 ajay teamlead 45000 p2 swiggy 13-3-24 50

1002 satvik programmer 50000 p1 skyview 12

1002 satvik programmer 50000 p2 swiggy 13-3-24 50

Normalization --- is the process of decomposing complex table structure into simple table structures without loosing data to avoid redundancy.

Employees

Empid name designation salary

1001 ajay teamlead 45000

1002 satvik programmer 50000

Projects

projectid projectname startdate duration

p1 skyview 12-3-24 100   
p2 swiggy 13-3-24 50

projectemployee

empid projid

1001 p1

1002 p1

1001 p2

1002 p2

One to one

One to may

Many to many

Join -- very costly

Denormalization – intentional introduction of redundancy

Join -- retrieving data from two or more tables and to join n tables you need to give minimum n-1 conditions

1.cross join

2.equi join

3.outer join

4.self joijn

Syntax: select table1.column1,table2.column1,… from table1 , table2 where table1.column = table2.column

Ansi syntax:

Syntax: select table1.column1,table2.column1,… from table1 join table2 on table1.column = table2.column

Select \* from employee,project --- cartesian product ---

1001 ajay teamlead 45000 p1 skyview 12-3-24 100 1001 p1

1001 ajay teamlead 45000 p2 swiggy 13-3-24 50 1001 p2

1002 satvik programmer 50000 p1 skyview 12

1002 satvik programmer 50000 p2 swiggy 13-3-24 50

1024 – 8kb -540000

10 exa byte - 1 exa byte - 1 million terabyte

Subqueries – is a collections of nested select statements within a query

Single row – string,char,date,miscelenous,arithmetic etc

Multirow – min,max,stddev,count,rank,denserank etch

Select columnname/\*/expression from tablename

select \* from dimproduct d where exists

(select count(distinct listprice) from dimproduct dp

where dp.listprice<= d.listprice)

20-8-2024

Single row functions – function which called for each row of the table

Group function – functions which work on group of records

View – is a database object that stores select statement

Complex view and simple view

\*\*Dml operations on the complex view will not be possible always

View derived multiple tables or view shows computed data, group by clause

create view product\_rank as

select listprice,EnglishProductName,ROW\_NUMBER() over(order by listprice) as rownum,

DENSE\_RANK() over (order by listprice desc) as ranks from dimproduct

select \* from product\_rank where ranks=4

insert into product\_rank(listprice,englishproductname) values(444,'dfdf')

collection sql statements stored inside a .sql file

batch- group sql statements along with programming constructs

network congestion – server overhead

parsing – compile – execute

procedure --- is a database object stores batch

header

batch

create procedure proc1

as

begin

end;

sp\_helptext <procedurename> to view the body

1 \* 5 = 5

2 \* 5 = 10

To create copy of a table

Select column,column into <newtable> from <existingtable>

select \* into products from product\_table()

**table valued function**

alter function product\_table()

returns @product table

(id int ,name varchar(30),price float)

as

begin

insert into @product values (1,'bingo',33.9);

insert into @product values(2,'sunfeast',44);

return

end

rule – a database object stores constraint definitions

create rule agerule

as

@age>0 and @age<=150

create table customers(id int ,age tinyint)

sp\_bindrule 'agerule', 'customers.age','futureonly'

sp\_unbindrule 'customers.age'

insert into customers values(1,120)

cursor -- is a variable stores output of select command, usually used for row by row processing

declare <cursorname> cursor as <select statement>

open <cursorname>

fetch <cursorname> into <variablelist>

close <cursorname>

deallocate <cursorname>

alter procedure updslnocart

as

begin

declare carts cursor for select \* from cart

declare @itemcode int

declare @name varchar(30)

declare @qty int

declare @price float

declare @slno int

open carts -- execute the select command

fetch carts into @itemcode,@name,@qty,@price,@slno

while(@@FETCH\_STATUS=0)

begin

print @itemcode

fetch carts into @itemcode,@name,@qty,@price,@slno

end

close carts

deallocate carts

end;

cursor to update table row by row

alter procedure updslnocart

as

begin

declare carts cursor for select \* from cart for update

declare @itemcode int

declare @name varchar(30)

declare @qty int

declare @price float

declare @slno int

declare @counter int =1

open carts -- execute the select command

fetch carts into @itemcode,@name,@qty,@price,@slno

while(@@FETCH\_STATUS=0)

begin

update cart set slno = @counter where current of carts

fetch carts into @itemcode,@name,@qty,@price,@slno

set @counter = @counter + 1

end

close carts

deallocate carts

end;

truncate – no where clause, it will not invoke delete trigger, it do page deallocations than deleting row

triggers - automatically executing procedure ( to perform automations)

type of triggers – dml, ddl , logon ,logoff, system event

set identityinsert on

create trigger updstock

on <tablname/viewname>

after/insteadof

as

body of the trigger

*orderno generation trigger*

alter trigger insorder

on orders

instead of insert

as

begin

declare @ordno char(10)

declare @name varchar(40)

declare @gross float

select @name =customername,@gross=gross\_amt from inserted

select @ordno = concat('O',datepart(y,getdate()),datepart(mm,getdate()),datepart(dd,getdate()),

isnull(count(ordno),0)+1) from orders

insert into orders values(@ordno,@name,getdate(),@gross)

end

21-8-2024

Console –

GUI -- Web applications --- mobile applications -- tv --- settopbox --- embedded -- ML

Webapplications -- applications deployed/installed on a web server

Web server -- is a machine installed with middleware

Middleware -- is a software manages web request processing(eg:tomcat,apache,glassfish,iis,JBoss,Kestrel,NGINX,Fusion etc)

Web application – need to follow minimum two tier architecture

Client -- Server ( server side script)

Html,css,JS -- Java(servlets),C#(asp.net),C++(CGI),Python,Javascript(Nodejs), Hypertext Preprocessor(php), ColdFusion

Webapps-- (Traditional and Modern)

Navigator -- jvm – applets

C# --- Blazor ---

DART -- Flutter -- react native --- mobile app development frameworks

Client part ---- html,css, javascript(client side)

HTML -- markup -- XML -

* <tag> <tag/>
* <b> hello </b> -- .html
* Format the data

<products> ---tags describe the data

<product id=1>

<name>bingo</name>

<price>300</price>

</product>

<product id=2>

<name>kitkat</name>

<price>30</price>

</product>

</products>

Json{ key:value,key:value}

Web Site -- web app

www -- web 1.0 -- pages used display data --- (no interaction)

web 2.0 --- interaction --

web 3. 0 --- semantic web ---

html 5 --- semantic elements

crawlers --- aesthetic

CSS – cascading style sheet – 3

Unique element – unique style

Same element – same style

Multiple elements same style

Robots.txt -- configures the crawalers

Selector{

Key:value;]

Key:value;

} -- UX

Grid layout, flex layout

Style- inline .class1 {} #id{}

Bootstrap,960.gs,tailwind --- provide style files – style classes

RESPONSIVE – pages adjust appearance according screen size

Mobile first –

Media query – to achieve responsiveness

Bootstrap.css bootstrap.min.css

Cdn - content delivery network server

22-8-2024

Web – interaction -- responding user events,showing proactive messages to the user, or changing appearance of the page. To make pages interactive we scripting languages like javascript,vbscript,oscript,ecmascript,jscript etc

Developer 2000, power builder,VB 6

Javascript --- browser & node ---

Javascript is a object based scripting language –

Statement,function,object oriented --- dynamic page ---

DOM manipulation

Var,let, const

Type inferenced -- var --- variable will be hoisted

Higher order functions – functions which expect another function as parameter and returns another function as output

function ddfdf(int a,int b){

}

Synchronous --- function1(){

c.l(‘hello’);

show(); --- synchronous-- blocked ----

c.l(‘end’);

}

Multi threaded --- parallel processing --- time sharing - cpu - os

8086,8088, 80186,10286, 80486,80486 MMX, 80486HT Pentium –

Single threaded,

Non blocking --- asynchronous --- background --

Front end -- backend --- modern web app

Java – socket programming

Web service -- service is a function created in such a way that it can be called through internet

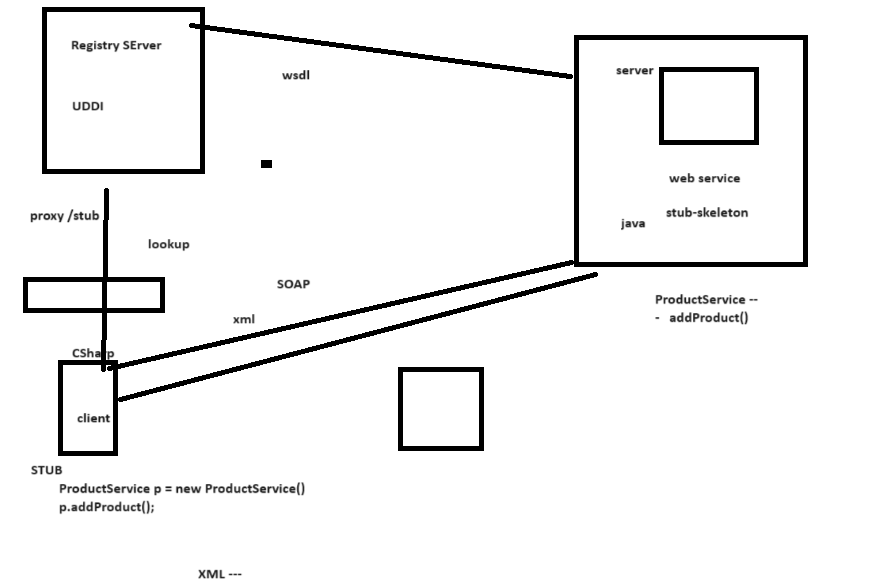
SOAP – simple object access protocol

<service>ProductSErviice

<method>addproduct</method>

</service>

WSDL – contains info about service



Web API --- api means collection of functions --- functions which can be called through internet using http url

<http://abc.com/api/Stock/add> --

<http://abc.com/api/Stock/update>

<http://abc.com/api/Stock/delete>

REST API -- rest api is a kind of web api

<http://abc.com/api/Stock> --

<form action=<http://abc.com/api/Stock> method =”post”>

</form>

Simple rest api – javascript

const http = require('http')

const server = http.createServer((req,res)=>{

    switch(req.method)

    {

        case 'GET':

        res.statusCode = 200;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify({'id':1,'name':'bingo'}))

        break;

       case 'POST':

        res.statusCode = 201;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify({'title':'success','message':'post request received'}))

        break;

      default:

        res.statusCode = 400;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify({'title':'not found','message':'method not supported'}))

    }

    res.end();

})

server.listen(3000,()=>{

    console.log('server started')

})

With json file data

const http = require('http')

const products = require('./data.json')

const server = http.createServer((req,res)=>{

    req.products = products;

    switch(req.method)

    {

        case 'GET':

        res.statusCode = 200;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify(req.products))

       // res.write(JSON.stringify([{'id':1,'name':'bingo'},{'id':2,'name':'sunfest'}]))

        break;

       case 'POST':

        res.statusCode = 201;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify({'title':'success','message':'post request received'}))

        break;

      default:

        res.statusCode = 400;

        res.setHeader('Content-Type','application/json')

        res.write(JSON.stringify({'title':'not found','message':'method not supported'}))

    }

    res.end();

})

server.listen(3000,()=>{

    console.log('server started')

})

What is nodejs -- runtime environment javascript, libraries

Nodjs libraries -- help us develop web applications and web api

Const htpp = require(‘http’)

Const server = http.createServer((req,res)=>{

If(req.method ==’GET’)

{

}

})

server.listen(3000,()=>{ }) ---

200 – ok

201 – created

202 – accepted

203 ---

300 –

400 –bad request

404 – not found

403 – forbidden

402 – payment required

401 – unauthorized

Get – retrieve

Post – insert data, put – modify, delete – delete ,patch - partial modification

Const obj = require(‘./controllers/get-controller’)

Obj.func1();

Document.on(‘on-click’,()=>{})

Express middleware – is a function/s called during all request eg:app.

Angular --- spa application framework

Spa – framework (angular,next),library(react)

Framework – partial implementation of application

npm install -g @angular/cli -- will give angular commands

typescript – pure object oriented and strongly typed language build on the top of javascript

client -- server

angular -- node, dotnet,java

to convert typescript to javascript – install typescript compiler/transpiler – transpilation

npm install -g typescript -- will install the transpiler

tsc filename.ts --- will generate filename.js

access modifiers typescript – private,public and protected

import {ETF,MutualFund} from ‘./StockManagenmen.ts’

angular -- front development framework

install angular – npm install -g @angular/cli – ( latest version will be installed) –

ver 1 - angularjs --- library -

ver 2 – angular 2 – framework -

2-17 -- module based

18 – stand alone components

To create a project ng new <projectname> --no-stanalone

Angular follows component based approach . A component is a chunk html along with css and typescript.

***To create only workspace***

Ng new <workspacename> --create-application=false

Eg:ng new workspace1 –create-application=false

To create project inside workspace

Cd <workspace folder>

Ng generate app <projectname>

Eg: ng generate app app1

To run the project from workspace

Ng serve <projectname>

Eg:- ng serve app1

Eg:- ng serve app1 - - port 4300 – will run the app on port 4300 instead of the default port

--- modules,component,directive,decorators

* Angular core – part of angular development kit

Components html file contains partial html for the page

Data binding : -- associating template elements to variables/object in the typescript file

One way binding-- showing data from the variable/object to template element

Or

Sending data from template element to variable / object

{{}} -- interpolation

[value] =”<variable>” -- property binding

(click)=”handler()” --event binding

[(ngModel)]="name" – banana syntax

27-08-2024

Routing -- loading components in the placeholder area

1.import router module

2.configure the route table

3.perform routing

Services -- classes which can be injected to other

Component interaction -- passing data from parent to child and vice versa

Parent to child - achieved @input decorotor

28-8-2024

Component – is a collection

@Injectable({

Providedin:’root’

})

Class myservice{}

Json-server -- sample rest api peform CRUD on json file

Npx json-server stocks.json

Interacting with rest api from angular app— HttpClient

HttpClient – in built class to interact with rest api , this class is present in ‘angular/commons/http’ module

DTO pattern -

Observable -- is present in rxjs package – is a kind of promise

[http://localhost:3000/stocks/1,{}](http://localhost:3000/stocks/1,%7b%7d)

first create edit component

add edit component as child route for showstock

then route edit component on click of edit button

29-8-2024

Reactive programming – uses publish subscribe method

Observable, BehaviouralSubject , Signals

promise is eager while observable is lazy

promise can return only single value – observable can emit multiple values

promise asynchronous – observable can be s asynchronous

observable can release memory through unsubscribe method

* - rest api -- using httpclient.get,post method
* Constructor(private bhumika:HttpClient){
* This.bhumika.post(url,object)
* Observable result = bhumika.post(‘http://localhost:3000/stocks’,s)
* Result.subscribe((data)=>{ }) --- call back method will be called when it is success
* Var methods ={
* Next:()=>{},
* Error:()=>{}
* }
* Observable result = bhumika.get(‘http://localhost:3000/stocks’)
* Result.subsribe(methods);
* Or
* Result.subscirbe({next☹)=>{},error☹)=>{}});
* Bhumika.get<Stock[]>>(‘dfdf’);
* Route guard – disabling routing of components based condition
* Steps
* 1.create a guard -- ng g guard <name>
* 2.add the validation code in the guard class
* 3.enable the guard in the route table

Reactive Forms –

30-8-2024

Components,directives,services and module

Lazy loading of module – use loadchildren function instead of component key int the route of the app.module

{ path: 'admin', loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule) }

Admin/vieworder

ng generate module admin --route admin --module app.module

pipes – to format date

syntax:

data | <pipe> [:param1:param2 …]

token -- get(‘ ’,{header:{‘content-type:’appic;ation/json’,authentication:’bearer’+token}}

* Interceptors - object that will intercept web request and inject additional request values

Testing – component testing – e2e testing

Component testing: ng test

E2e testing: ng e2e

To add karma configuration file

ng generate config karma

react --- is a library for creating single page applications

meta – third – party framework – Gatsby,next,

components -- js - function,class

ts- class, function

redux, context api

lifting state up—pushing state down

2-9-2024

React – library

Diffing --- comparing domtree with virtual dom

Cdn—

Npx

Class components -- state - entire page state ---

One component’s State->props

Parent component can send state to child compo

App

Return(

<Home/>

)

Home

Return( <>

<AddDish/>

<ShowDish/>

Adddish

Return(<>

<comp1/>

<comp2/>

Hooks --- will import state management to functions

3-9-2024

Axios.get,post,put,delete

Base url store in axios.config file

Post(‘/stocks’, s), useEffect()

Routing --- loading components on demand

1.install react-router-dom package

2.in the root component import BrowserRouter, Routes,Route

3 . in the component from which you want enable the routing import Link compoent or navigate method

Different ways to route – LINK component or navigate method

To use navigate method import useNavigate hook

4-09-2024

Routing

Redux - realm

Context api –

Createcontext() - create the context instance

useContext--- will return the context object address to subscribing components

Angular--

parent to child -- create a variable in the child component to receive the data . mark it with @input decoroator, the variable name becomes property of the component

class showbooking{

@Input () bookings = []

}

Class addbookig{

bookins = []

addHandler(){ this.bookigs.push(anil)

}

Html

<inut type=text/>

<app-showbooking [bookings]=”bookins”/>

Child to parent- declare an eventemitter type object , and mention the type of data in the declaration time eg: new EventEmitter<Array<Stock>>() and mark it with @Output Decorator

And call the event emitter object.emit() method to pass the data

class showbooking{

@Output ( ) sendbooking = new EventEmitter<Array<booking>>()

Onbuttonhandler(){

This.sendbooking.emit([]);

}

}

Parent

Class receiver{

onReceive(data:any){

}

<div>

<app-showbooking (sendbooking)=onReceive($event)/>

5-9-2024

Package -- webpack

Next.js --- is a framework made on the top of react, uses server side rendering by default

Npx [create-next-app@14.1](mailto:create-next-app@14.1) <projectname>

Npm run dev ---

Routeparams, validation etc

Dotnet

What is dotnet - is a name of initiative from Microsoft to reduce the complexity of n tier applications.

As part of this initiative Microsoft developed

1..Net Framework - is a software contains tools and platforms for dotnet application development

2.dotnet languages -- vb.net,c#.net,c++.net,f#.net,java.net etc

3. runtime (CLR) -- execution environment for dotnet project

CLS -- common language specification

CTS -- common type system

Dim a as integer - vb.net --- 1010110110

Int a; - c# -- 1010110110

Output of compilation is called assembly – similar jar file in java, inside assembly MSIL

And the extension of this file .dll , .exe

Standard,universal,xamarin

.net Framework -1.0,2.1,3.1,3.5,4.0,4.8 – targeting on windows os

.net core framework -2016 - 1.0 ,2.1,3.1 -- cross platform

.Net 5, .Net 6, .Net 8 -- c# language ver 12

C# is pure object oriented -- does not support multiple inheritance

Dotnet new console -n <,projectname> - to create a console project

Dotnet new list --- will give the list of project types

Language interoperability --- one language program can use classes written other dotnet language

//top level statement

//byte,sbyte,short,ushort,int,uint,long,ulong,decimal,char,bool,float,double

// value type

//0-255(1byte) -byte

//-128 - 127 (1byte)

//   ??,\_,?

Int a;

Var a = “dflkdjf”

a = 10;//wrong

Dynamic a;

a = 90; --int

a =”satvik”

a =9.0

object,string -- reference type

arrays -- array names are reference type variable

single dimentional,multi(rectangular,jagged)

int []num = new int[size];

two dimensional

float [,] weather = new float[4,4]; // two dimensional rectangular

weather[0,0] =44’

float [][]weather = new float[3][];

weather[0] = new float[44];

weather[1]=new float[3];

weather[2]=new float [5];

weather[1][0] =88;

weather[0][43] =

class,struct,record

[<access modifier> < modifier>] class <clasname>{

}

Eg: public static class System{

}

**Access modifiers** :-public internal protected private private protected,protected internal, file

class yes [yes] no no no no yes

Fields/properties yes yes yes [yes] yes

Methods yes yes yes [yes] yes yes yes

Modifiers: abstract, static, sealed, transcient, volatile, synchronized, virtual

Internal - accessible only in the current namespace

Public - accessible everywhere

6-9-2024

protected Class sample{

Int age = 90;

Void display(){

Console.WriteLine($” your age is {age}”

}

}

Char ch = ‘a’;

Decimal amount;

“”” hello “Mr” satvik”””

Auto properties

8-09-2024

API- node.js

Minimal API --- rest api with one or two end points

Functional programing – passing functions as params

App. Middleware function –

traditional - ( web forms app) - using .net framewok

mvc –

.net core ---

razor pages –

Mvc -slots.aspx ----

Asp engine --

Public void page\_preload(){

Response.wrie(“<html>”);

}

Public void page\_load(){

}

Public void button1\_clicke(){

}

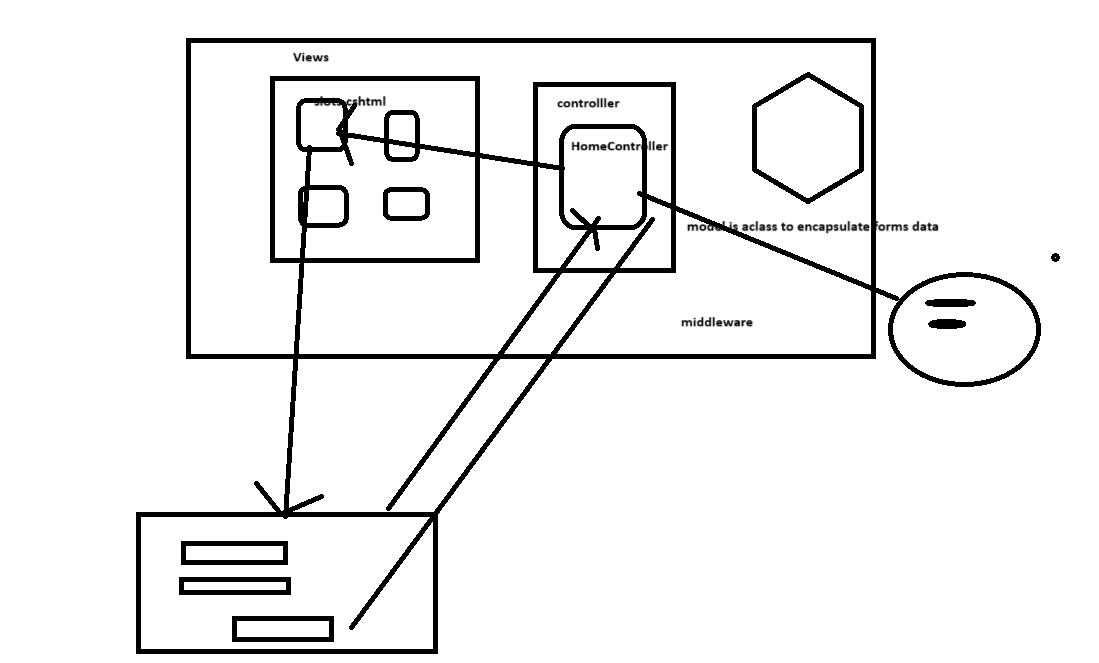
}

Click - event – post back-

Controller -- action methods --- will return view

In mvc application – we sent request to action methods of controller

Model is a class encapusulate the data



9-09-2024

Func f = ()=>{ };

Action <int>

Predicate

Public void Showdata(Func<string,List> param)

{

}

Custom delegate types

Delegate void MyDelege(int,int); -- delegate

MyDelegate m = (x,y)=>{//code;}

Mvc api --

Model,controller, views –

System.Collections –

System.Collections.Generic;

Stack s= new Stack(); //hetreogenous

Queue<Store> s\_queue = new Queue<Store>();//homogenous

// Hashtable hash = new Hashtable();

Dictionary<string, Store> hash = new Dictionary<string, Store>();

List <Store>array = new List<Store>();

LinkedList<Store> list = new LinkedList<Store>();

IEnumerable<Store> stores = new LinkedList<Store>()

Attributes - provide metadata (additional data) about classes,methods,properties etc

[attribtename]

Eg: [ApiController]

<http://localhost:4445/ajay>

dyanamic dependency- depependcy injection

new StoresController(new StoreListRepository());

Database Interaction – AdO.net, Entity Framework

DbContext – class –

Connection string=”server=localhost; initial catalog=jeanstation; user id=sa; password=pass@123;trustedservercertificate=true;”

Object Relational mapping – creating tables in the same structure of model class

* Migration
* Add Microsoft.entityframeworkcore.tools
* Then open tools menu->select nugetpackagemanager console

Type: add-migration <name>

Then- update-database

10-09-2024

-- the below property will not be converted to a column in the table during migration

[NotMapped]

Public int age;

Formik

Npm I formic

import axios from 'axios';

import { useFormik } from 'formik'

import React from 'react'

const AddHotel = () => {

    const formik = useFormik({

        initialValues:{mobile:'',latitude:'',longitude:'',address:''},

        onSubmit:(values)=>{

            alert(JSON.stringify(values))

            axios.post()

        }

    });

    return (

        <div class="row justify-content-center">

            <div class="col-md-5">

                <div class="card">

                    <h2 class="card-title text-center">Hotel Onboarding</h2>

                    <div class="card-body py-md-4">

                        <form  onSubmit={formik.handleSubmit}>

                            {/\* <div>

                                <input type="text" value={formik.values.mobile} onChange={formik.handleChange} />

                            </div> \*/}

                            <div class="form-group">

                                <input type="text" class="form-control"

                                id="mobile" name='mobile'

                                placeholder="Mobile" value={formik.values.mobile}

                                onChange={formik.handleChange}/>

                            </div>

                            <div class="form-group">

                                <input type="text" class="form-control"  placeholder="Latitude"

                                id="latitude"

                                name='latitude'

                                value={formik.values.latitude} onChange={formik.handleChange}/>

                            </div>

                            <div class="form-group">

                                <input type="text" class="form-control"  placeholder="Longitude"

                                id="longitude"

                                value={formik.values.longitude} onChange={formik.handleChange}/>

                            </div>

                            <div class="form-group">

                                <textarea class="form-control"  placeholder="Address"

                                id="address"

                                value={formik.values.address} onChange={formik.handleChange}></textarea>

                            </div>

                            <div class="d-flex flex-row align-items-center justify-content-between">

                                <a href="#">Login</a>

                                <button class="btn btn-primary" type="submit">Create Account</button>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    )

}

export default AddHotel

=================

11-9-2024

============================

ADO.Net -- is an api to interact with databases

Supports – connected architecture, disconnected architechture

Connected - 1. Establish connection,execute the query, close the connections.

Package-: System.Data.SqlClient, Microsoft.Data.SqlClient(to interact with sql server)

Or System.Data.OracleClient,

Classes involved is: Connection,Command,DataReader-

SqlConnection,OracleConnection

Partial class- one class definition can be splitted to multiple files

=========================

12-9-2024

==========================

Activex Data Objects.net (ado.net)

Connected,

Disconnected -- CRUD operations can be performed without a database connection

DataSet,DataTable,SqlDataAdapter,SqlCommandBuilder

1.Establish connection and fetch data from table and fill to Dataset DataTable -- use Fill function of Adapter

2.close the connection

3.do the crud operation in datatable

4.update the changes to the database -- use Update function of Adapter

Class DataSet{

DataTable []Tables = new DataTable[100];

}

Class Datatable{

DataRow []Rows =new DataRow[100];

}

Exception- a errorneous situation happens during runtime , during exception program will terminated abnormally.

Exception Handling --- expect/for see statements that may generate exception, and put those statements inside a try block,

Write handler block- typically this is written inside a catch block/ catch function

Certainly! Let’s dive into the **Dispose pattern** in C# and why it’s essential for managing resources.

In the world of software development, we often deal with resources—things like memory, file handles, database connections, or other external entities. Some of these resources are managed automatically by the garbage collector (GC), but others—called **unmanaged resources**—require explicit handling. That’s where the Dispose pattern comes in.

1. **What Is the Dispose Pattern?**
   * The Dispose pattern is a structured approach to releasing unmanaged resources. It’s commonly used for objects that implement the IDisposable interface.
   * When you work with file handles, registry handles, wait handles, or pointers to blocks of unmanaged memory, the garbage collector can’t reclaim those resources. So, we need a way to explicitly release them.
   * The pattern ensures that resources are cleaned up appropriately, even if an object is no longer needed.
2. **How Does It Work?**
   * When you implement the Dispose pattern, you provide an IDisposable.Dispose method. Consumers of your type call this method when the object and its associated resources are no longer needed.
   * The Dispose method should release unmanaged resources immediately. It’s essential to make it **idempotent**, meaning you can call it multiple times without causing issues.
   * By following this pattern, you gain control over when resources are released, rather than relying solely on the GC’s timing.
3. **Safe Handles: A Better Approach**
   * Instead of implementing a finalizer (which has some drawbacks), consider using System.Runtime.InteropServices.SafeHandle objects.
   * A SafeHandle wraps an IntPtr that identifies an unmanaged resource (like a handle or file descriptor).
   * It ensures that the resource is released once and only once—either when the SafeHandle is explicitly disposed or when all references to it are dropped and it’s finalized.

Remember, the Dispose pattern is your friend when dealing with unmanaged resources. It’s like the Marie Kondo of memory management: “Does this resource spark joy? If not, Dispose of it!” 😄

If you have any more questions or need further clarification, feel free to ask! 🌟

Throw – used to throw user defined exceptions

Class NoSlotAvailableExceptin:Exeption{

}

Post method to upload image

[HttpPost]

public IActionResult Post([FromForm] Dish value)

{

try

{

if (value.DishImage == null || value.DishImage.Length == 0)

return BadRequest("No file uploaded.");

var originalFileName = Path.GetFileName(value.DishImage.FileName);

var filePath = Path.Combine(\_basePath, originalFileName);

if (!Directory.Exists(\_basePath))

{

Directory.CreateDirectory(\_basePath);

}

using (var stream = new FileStream(filePath, FileMode.Create))

{

value.DishImage.CopyToAsync(stream);

}

}

catch { }

logger.LogWarning("Dish controller post method is called");

DishDTO dishDTO = new DishDTO();

dishDTO.Name = value.Name;

dishDTO.DishImage = "http://localhost:5212/images/"+value.DishImage.FileName;

DishData.dishes.Add(dishDTO);

return Ok();

}

===========================================

13-09-2024

===========================================

Multi Threaded Programming –

Process(functions) – Thread

Mp4 -- audio ,video

Main(){

Showvideo(); 1

Playaudio();1

}

Timesharing -- mainframe --

Param - 10000 ---

Pit --

Optimization-- rolling -- divide bigger loops into smaller loops

Unrolling – making smaller loops into bigger loops

Fault Tolerance

Request Threshold --

Load balancing

Scaling -- Vertical and horizontal

Micro Service --- divide complex api to simple api’s in such a way that performance can be improved and fault tolerance to minimum.

Gateway Pattern - -

Access Token is an object in encrypted format contains data called as payload.

JWT- {header:dfdflj,payload:ldkfjk,signature:ldfkdjlfj}

Encrypted object--- secret/salt

DSFDSFADSFASFA8&^\*^\*^HFTf.$^&%(\*IFJG(\*(&FJGJ^%&%yhffu&%fhf.UYYUT5776575&^%&%\*$^$

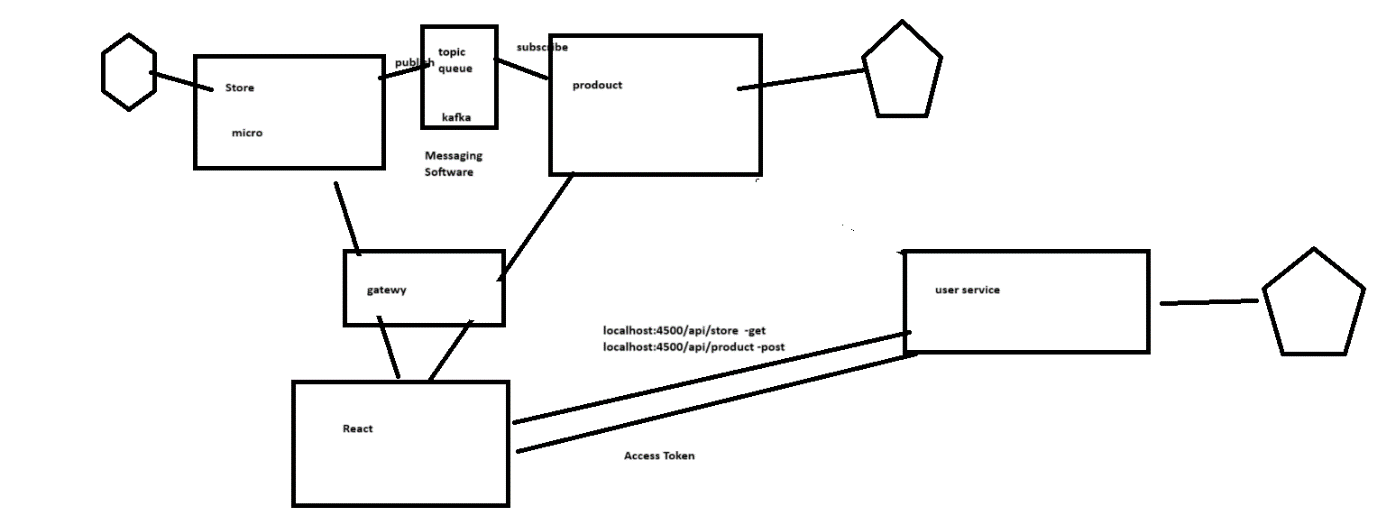
{}

Messaging server-- installed messaging software--

Kafka, Redis,JMS,WindowsMessaging,

SignalR --- WebSocket

Micro Service - architecture



API – Documentation -- a set of html pages that provide info about api end points available

Nuget package—swashbuckler –

Reflection ---API

Unit testing – C#

Test exection tool ----- testing library

Test explorer -mstest,NUnit,Xunit

================================================

14-09-2024

================================================

From s in stores

Join p in products

On s.storeid equals p.storeid

Select new {storied=s.storeid ,sname=s.storename, pname=p.name,price=p.price}

NoSQL database- stores data in a schema less manner{json}

Cassandra, CosmosDb, MongoDb, DynamoDb etc

Databases->collections->documents

Db.products.find({“price”:{$gte:100}}

============================

09-2024

====================

File Operations , Distributing Application(Docker), virtual machines

Reading and writing to files, working with file systems etc

System.IO; text files, binary files --- json files – .Net 6

Distributing applications - Assembly -- exe,.dll

Virtualisation -- vmware, virtual box - hypervisor

Bare metal --- no host operating system required

Containerization:

ContainerEngine:-OpenShift,Docker etc

Create a Dockerfile

Inside the docker file add steps to create image

Create the image – using docker build eg: ***docker build -t store-api:ver1 .***

Test the image using docker run command eg: ***docker run -p 8080:8080 store-api:ver1***

Running docker image is called container

To view the list of containers – ***docker ps***

To view list of containers including stopped – ***docker ps -a***

Stop docker container: docker stop <containerid/name> eg: ***docker stop 2dd***

Start stopped container: docker start <containerid/name> eg: ***docker start 222***

Remove images: docker rmi <imagename> eg: ***docker rmi store-api:ver1***

====================

17-09-2024

=======================

**Components for Route Guarding in React**

***ProtectedRoute.jsx***

import React, { useEffect, useState } from 'react'

import { Navigate, Outlet } from "react-router-dom";

import axios from "axios";

export default function ProtectedRoute() {

const [isAuthenticated, setIsAuthenticated] = useState(null);

useEffect(() => {

axios.post('http://localhost:9000/isauthenticated', {}, {

headers: {

Authorization: localStorage.getItem('token')

}

}).then(res => setIsAuthenticated(res.data.isAuthenticated))

.catch(err => setIsAuthenticated(false));

}, [])

if (isAuthenticated == null) {

return <div></div>;

} else {

return isAuthenticated ? <Outlet /> : <Navigate to="/login" />;

}

}

---------------------

**App.js**

**-------------------------------------------------**

import { BrowserRouter, Routes, Route, Navigate, } from "react-router-dom";

import Register from "./Register";

import Header from "./Header";

import Home from "./Home";

import Login from "./Login";

import Profile from "./Profile";

import ProtectedRoute from "./ProtectedRoute";

function App() {

return (

<div>

<BrowserRouter>

<Header />

<Routes>

<Route path="/" element={<Home />} />

<Route element={<ProtectedRoute />}>

<Route path="/profile" element={<Profile />} />

</Route>

<Route path="/login" element={<Login />} />

<Route path="/register" element={<Register />} />

</Routes>

</BrowserRouter>

</div>

);

}

export default App;

File.Delete(“abc.txt”);

File.Copy(“source”,desti);

File.IsExits(),File.IsDirectory(),

File.MoveTo(sourcefile,destinationfolder);

etc

StreamReader and StreamWriter

Serialisation- storing state of an object to a file

Deserialsation-create object back into memory from file

Store s = JsonSerialiser.Deserialise<Store>(file);

Virtualisation --- is a way to distribute applications size and performace

Containerization:- another way distribute application, we need a container engine(open shift,docker) and container engine will provide multiple os environments

Docker hub -- it s a platform to publish container images, so users can pull the image and run

Docker build -t imagename:tag <location of docker file>

Network --- volumes

Docker-Compose -- Docker Compose is a tool for defining and running multi-container applications

C:docker-compose up -- docker-compose down

**Deployment to Cloud**

---------------------

On Premises -- server available locally need configure and expose application outside the network

Cloud Deployment --- deploy apps to a server provided by companies (Azure – Microsoft,AWS,DigitalOcean), Filezilla, putty,

Pre configured servers – are provided

IasS,PaaS,SaaS,Serverless - Azure

Infrastructure as a Service – Virtual machine

Platform as a Service --- AppService

Software as a Service – Microsoft Office 365,SQL Server Database

Serverless ----- Azure Functions

Azure account—subscription

Portal.azure.