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

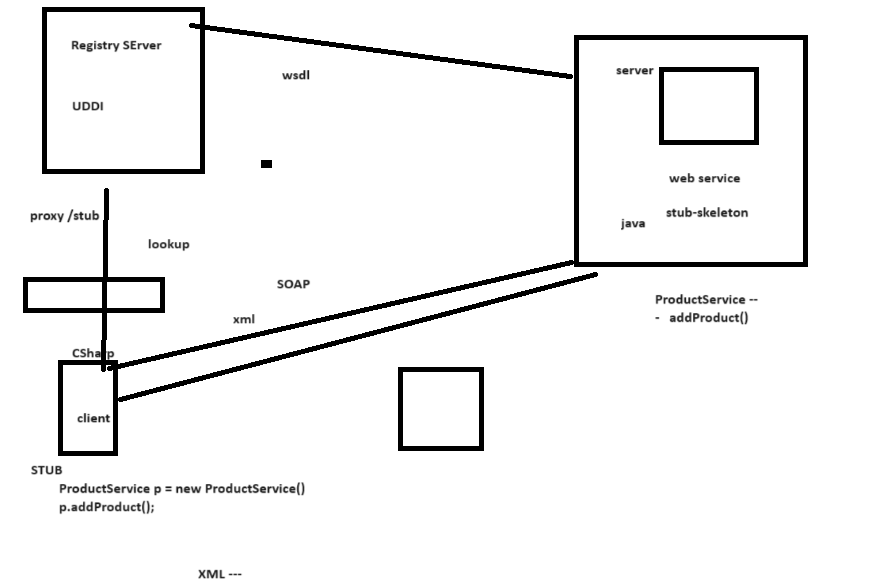
SOAP – simple object access protocol

<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')

})