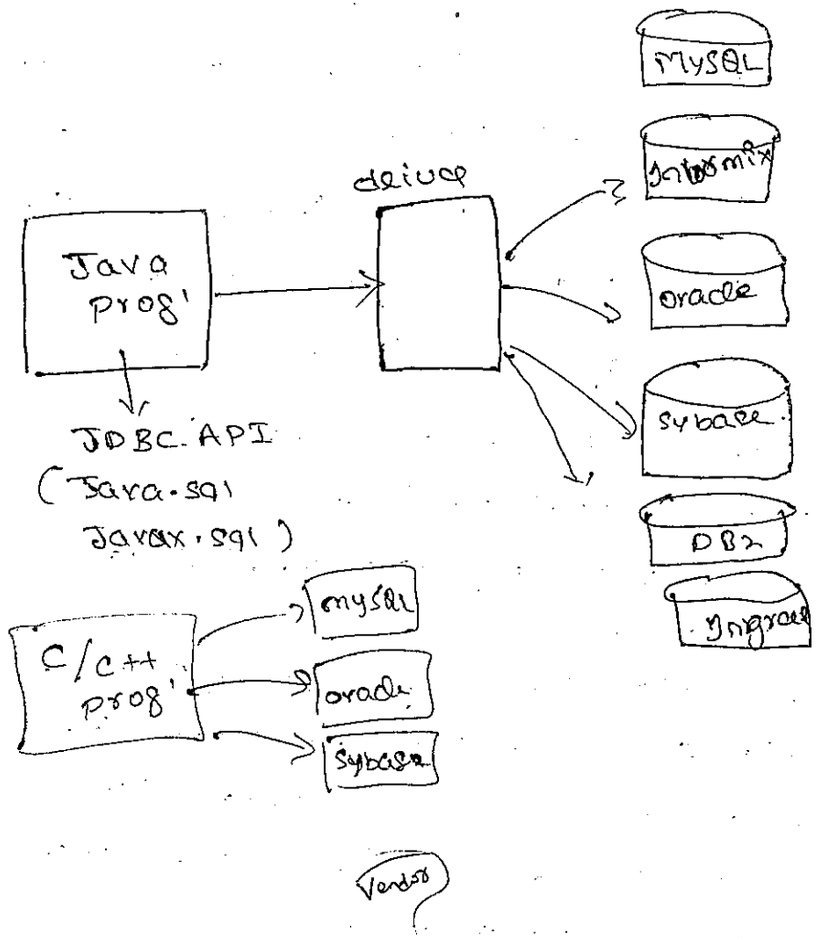
24/06/06

JDBC

Java DataBase Connectivity:



If we want to interact with database from c/c++ prog', then we have to use database related libraries directly in c/c++ prog's. This increases the direct dependency bet" prog' & database & also ↑ the maintanances. These prs degrades flexibility of the sys'. & performance.

Sun analize all these prs & decided to provide portable & unique API to interact with any database, with less direct dependency.

For this Sun as introduce JDBC API. In this calls frm Java prog's will be given to some interface called driver & then driver forwards the calls to particular database.

"Sun" has provided 4 types of JDBC drivers.

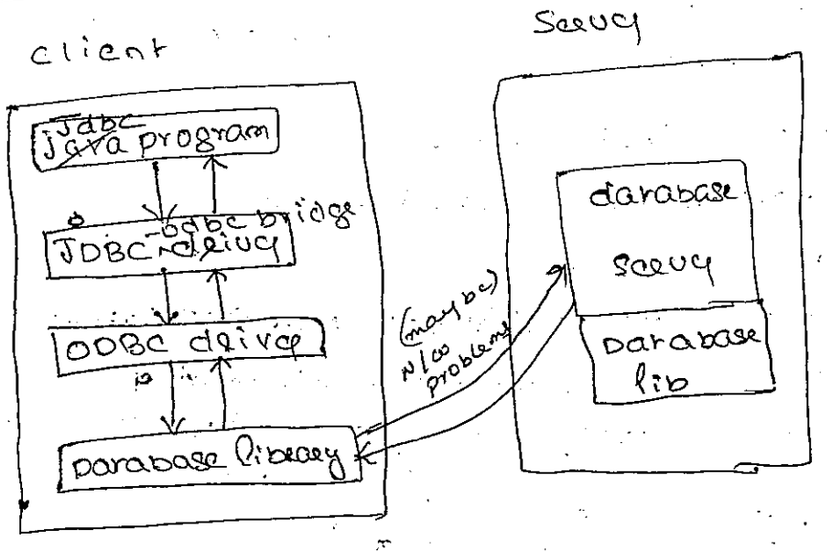
Type 1 : Jdbc-odbc bridge

Type 2 : partial native partial java driver

Type 3 : Net protocal driver

Type 4 : pure java driver.

Jdbc-odbc bridge driver :



when u running Jdbc prog', means u r making the Jdbc call. Jdbc call converts to Odbc call. Odbc call will interact with the vendor specific, client side database library. Na client side database lib. makes a call to database server which is the native call. or native Requestr

jar = collection of packages

Type 1

name: ~~JDbc~~ Jdbc-odbc bridge

vendor: Javasoft+inrepsolv

path : None

classpath: d:\j2sdk1.5\jre\lib\rt.jar;class.

name : sun.jdbc.odbc.JdbcOdbcDriver

url : jdbc:odbc:DSN name (datasource name)

- jdbc → main protocol
- odbc → sub protocol
- DSN name → sub name (unique)

pros:

1. when ur migrating the platform (appln), then we can make use of existing odbc drivers & dsn (datasourcename).
2. Suitable for smallscale applns.

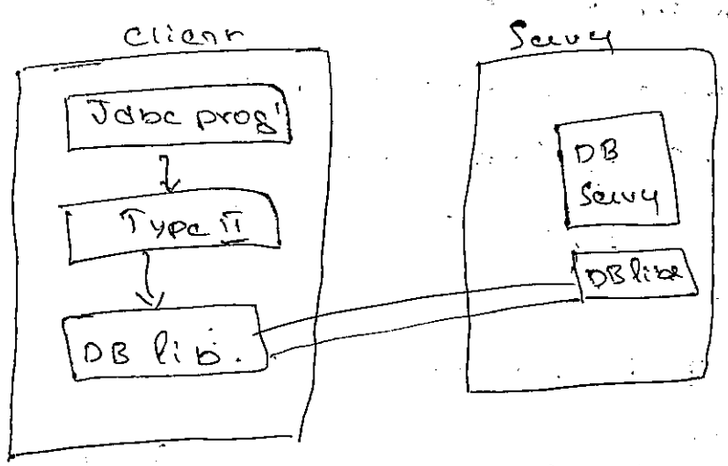
Cons:

1. Type I is limited for windows O.S system.
2. ~~Type~~ When we are using type 1 for conversions has to be done. bcoz of this request processing ~~spee~~ response time will be more.
3. performance of type 1 is less, when u compare with other 3 drivers.

~~Type II~~:

4. we have to install some database lib. in the client m/c also.
5. odbc is developed in c-lang.
6. It is not suitable for Internet applns.

2. Type 2 : partial native partial java driver.



when u make a jdbc call, jdbc call will be converted into native call with the help of client side database library.

name: partial native partial java driver
Vendor: DB Vendor
path : d:\oracle\ora90\bin;
classpath : d:\oracle\ora90\jdbc\lib\classes111.jar;

~~zipped~~ classes111.jar is available in ora90\jdbc\lib
classes111.zip (ora81)

.jar classpath both on client & server systems.

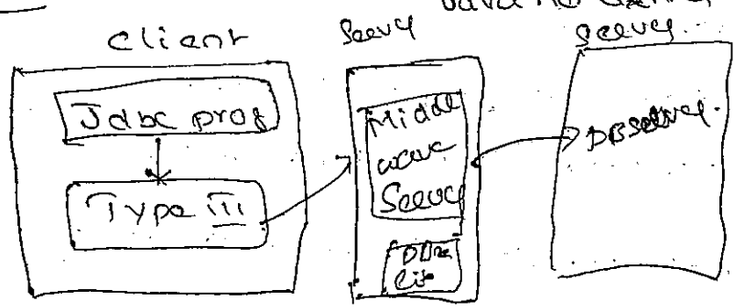
classname : oracle.jdbc.driver.OracleDriver
url : oracle:oci8:@hostname:portnumber:SID
eg: jdbc:oracle:oci8:@hostlocalhost:1521:JAVA

(we can find the info abt hostname, sid and portno. in d:\oracle\ora90\NETWORK\admin\tnsnames.ora

pros : 1. performance of this driver is better than Type1.
2. It is suitable for all O.S.

cons : 1. In this we have to install some lib. in the client m/c, bcoz of this maintenance will be ↑.
2. It is not suitable for internet appln.

Type III : Net protocol driver (or) Java net driver



when u make a jdbc call, that will interact with the middleware server, where the database lib is available & this call will be converted to a native call with using that lib.

name : net protocol driver (or) Java net driver

Vendor : Intersolv

S/w : install IDS server

path : d:\idserver\idss.exe

classpath : d:\idserver\...\idsdriver.jar

classname : ids.driver.IdsDriver

url : jdbc:ids://localhost/ie/conn

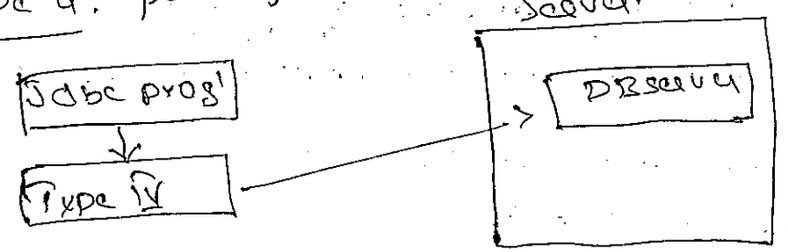
pros :

1. No need to install the db lib in client m/c
2. Suitable for Intranet appln.

cons :

1. Using this type 3, if u don't have a chance to interact with db server directly we have to go through a middle way server. This ↑ request-response time.
2. performance is little bit poor; than type 1 & 2

Type 4 : pure java driver



→ when u make a jdbc call, that will be converted to a native call directly.

Type IV:

name: pure java driver

vendor: DB vendors

software: any oracle s/w

path: d:\oracle\ora90\bin;

classpath: d:\oracle90\jdbc\lib\
classes111.jar;

(On server system)

(Ser the classpath only on server syst)

classname: Oracle.jdbc.driver.OracleDriver

url: jdbc:oracle:thin:@hostname:1521:SID
(Tunique)

Eg: jdbc:oracle:thin:@hello:1521:JAVASESS

performance: Type IV

Speed: Type II

pros: 1. NO need to install db lib in the client m/c

2. It is suitable for internet appln

3. performance is good, among 4 (Type 1, 2, 3, 4)

cons:

1. Speed is less when compared to Type II.

Elapsed Time (request time + processing time + response time) is little bit higher than type 2.

Comparision:

In performance wise Type IV is best,

Speed wise Type II is best.

Type II vs Type IV

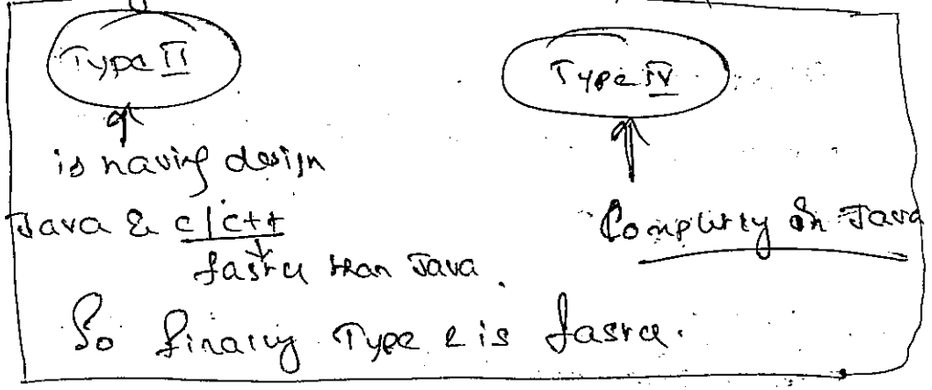
1. Type 2 is not good for stored procedures but for prepared stmts (prepared stmts) where as Type 4 is good for all.

2. Type 2 is not good for internet appln where as Type 4 is good

3. Type 2 is not portable on all OS's where as Type IV

4) Type 2 is not supported by all the db coneccas type 4 supports all.

? Why Type fastr than Type 1?



So finaly Type 2 is fastr.

steps to implements JDBC prog':

1. load the driver class
2. Estrablish the connecn.
3. create the stmr.
4. prepare the Query & execute it.
5. process the results.
6. close the stmr & connecn.

E: for Type 1

1. load the driver class

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
... driver class will be loaded into mem'.
... static blks will be executed.
... this driver will be registered
```

2. Estrablish the connection:

```
Connection
con=DriverManager.getConnection(url,un,pw);

Connection
con=DriverManger.getConnection("jdbc:odbc:
        srDSN","scott","tiger");
                          by default.
```

3. create the stmr.

```
Statement st = con.createStatement();
```

4. prepare the query & execute it

```
String sql = "select * from student";
ResultSet rs=st.executeQuery(sql);
```

5) process the results:

rs contains results ... processing goes here

6) close the stmt & connection.

```
rs.close();
sr.close();
con.close();
```

① Type1 driver & oracle database. (prog)

~~import java.sql.*;~~

class JdbcEx1

① create the table

create table b7student(

```
sno char(5),
sname char(20),
email char(15),
phone char(15));
```

② Insert 2 records.

```
insert into b7student values('s099',
    'sri','sri@sd.com','9999');
insert into b7student values('s0888','vas',
    'vas@
```

③ create the dsncalled sriDSN

1) start → settings → controlpannel
2) click on administrative tools
3) Data Sources (ODBC)
4) clk on add button
5) select oracle driver & click on finish
6) provide:
   dsn name .. sriDSN
   tns servicename ... JAVA8888
   user ID : ~~sakr~~ scott

and click on Test connection.

7) provide password --- tiger & click ok

8) ok ok.

(4) Write the Jdbc program

```
import java.sql.*;
class JdbcEx1
{
  p s v m(String args[])
  {try{
  class.forName("sun.jdbc.odbc.JdbcOdbc
                    Driver");

  Connection
    con = DriverManager.getConnection("jdbc:
        odbc:sriDSN","scott","tiger");

  Statement st = con.createStatement();
  String sql = "select * from b7student";

  ResultSet rs = st.executeQuery(sql);
  while(rs.next())
  { s.o.p(rs.getString(1));
    s.o.p(rs.getString(2));
    s.o.p(rs.getString(3));
    s.o.p(rs.getString(4));
    s.o.p("\n");
  }
  rs.close();
  st.close();
  con.close();
  } catch(Exception e)
    { s.o.p(e);
    }
}
}
```

(5) Compile & Execute the prog:

Compile... javac jdbcEx1.java.

25106106

Q: Extracting classes from oracle.

Using Type 4 with oracle database.

```
import java.sql.*;
class JdbcEx2
{
  p s v m (String as[])
  {
    String sno = as[0];
    String sname = as[1];
    String email = as[2];
    String phone = as[3];
    try {
  class.forName("oracle.jdbc.driver.OracleDriver");
  Connection con = DriverManager.getConnection(
        "jdbc:Oracle:thin:@hello:1521:
    JAVASRE", "scott", "tiger");
  Statement st = con.createStatement();
```

~~String sql = "insert into student values(?,?,?,?);~~

```
class.forName("Sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:Type 2
                "SridSN", "Scott", "tiger");
String sql = "insert into student values("+sno+", "+sname+"
        "+email+", "+phone+");
int x = st.executeUpdate(sql);
if (x==1)
  s.o.p("Inserted Successfully");
else
  s.o.p("Not Inserted");
String sql = "select sname, email, from student;"
ResultSet rs = st.executeQuery(sql);
while (rs.next())
  {
    s.o.p(rs.getString(1));
    s.o.p(rs.getString(2));
```

26/06/06

```
        S.O.P(" ");
    }
    rs.close();
    st.close();
    con.close();
}
catch(Exception e)
{
    S.O.P(e.getMessage());
    e.PrintStackTrace();
}
}
```

URL - Uniform resource locator (26/06/06

portno. → contextpath. → to find request

Types of stmts:

We have 3 types in Jdbc

(1) statement

(2) prepared Statement

(3) callableStatement.

(1) Statement:

① Statement st = con.createStatement();

when u creating the stmt, we are not passing any query as parameter.

② ResultSet rs = st.executeQuery(sql);

we are passing the query, while ~~as~~ when ur executing the ~~executeQuery~~ stmts, execute, executeUpdate(), executeQuery. At this time following things happen every time

- queySubmission
- query Compilation
- query Execution.

2) prepared statement:

1. preparedStatement ps = con.prepareStatement (String) query Submission, query Compilation → query.

when ur creating the prepare Stmt, we are passing the query as the parameter.

2. ResultSet rs = ps.executeQuery() → query will be executed

we are not passing any parameter, when ur are ~~are~~ calling execute(), executeUpdate(), executeQuery() mrds.

when u creating the prepareStatement() by passing query as parameter, then query will be submitted to the sql engine. & will be compiled and precompile query will be stored in the database memory (not in JVM mem).

3) CallableStatements : (Just like JDBC mtd), are used to execute the stored procedures available in the db.

* By using stmt & preparedStmt, we are executing one query at a time, but using stored procedures we can write as many as we can.

CallableStatement cs = con.prepareCall(String)

call to the procedure

Eg: "{ call add() }"

submission
compilation

int x = cs.executeUpdate(); → Execn.

when ur creating the callable stmt by passing string as parameter (string is a call to stored procedure "{call add()}"), call will be submitted to the database engine and will be compiled & stored in mem.

when u call execute(), executeUpdate() executeQuery() mtd procedure will be executed & results will be return to client prog.

Difference among execute(), executeUpdate() & executeQuery();

① boolean execute():

returns true, when the query is executed successfully, returns false otherwise.

② int executeUpdate():

returns no. of rows affected with the given query.

	yes	no
insert into student val(,---).	(1)	(0)
update student set bal=0 where bal<=500	(M)	(0)
delete from student where bal=0	(N)	0

select * from student;

③ ResultSet executeQuery():

Use this mrd, for the select srmts when u execute this, no. of rows will be retrieved & will be stored in the ResultSet objects. We can create the resultSet object with the following mrds.

- boolean next()

next mrd returns true, when there is next record, returns false, when there is no next rec.

Initially resultSet object pts to the row, which is before to the first Record. we get the corresponding values of the columns; at pointed row, we can call the following mrds.

rs.getInt(1)/(2) rs.getInt("sno");
rs.getString(2) rs.getString("sname");
rs.getDouble(3) rs.getDouble("bal");

(RS)

	Int	String	bool
1			
2			
3			
4			
5			
6			

Sno	Sname	Semail	pin:bool

getXXX(int)
getXXX(String)

Connection con = DM.getConnection(-,-,-);

class OracleDriverManager extends DM
{
  Connect getConnection(-,-,-)
  {
    con = new OracleConnection();
    return con;
  }
}

class OracleConnection implements Connection
{
  Stmt createStatement()
  {
    return new OracleStatement();
  }
}

Eg3 with type 4 & mysql database:

1) create the table student

MySQL:

DriverManagerName.

className: com.mysql.jdbc.Driver

url : jdbc:mysql://localhost/b7db

:create database b7db
:use b7db
:create
:desc student.

} in mysql.

mysql> show tables
show databases

Ex3: with type 4 & mysql DB

/* 1) start the mysql

a) start → programs → MySql5.0 → MySQL Server → MySql Command line Client

b) give password u can see mysql command prompt. (root)

2) Create the db in mysql

create database b7db;

3) open the database b7db ;

use b7db;

4) create the table student in mysql

create table student(

sno integer,
sname char(15);
phone char(8);
bal double);

5) insert 2 rec

insert into student
values("99", 'sri', '9999', 2000.00);

insert into student
values(08, 'nitai', '0808', 25000.00);

6) write the program */

import java.sql.*;

class JdbcEx3

{ p s v m (String args)→

{ try

{ Class.forName("com.mysql.jdbc.Driver");

/* Driver
d = DriverManager.getDriver ("com.mysql.jdbc
Driver");

DriverManager.registryDriver (d); */

Connection
con = DriverManager.getConnection("
jdbc:mysql://localhost/b7db", "root",
"Javaee");

```
sraremaor sr = con.ceeareStaremenr();
Sring sql = "inseer into sradenr valuey
(11,'sinha','0000', 30000);
inr x = sr.execureOpdare(sql);
if (x==1)
{ s.o.p("insoered succesfully");
}
else
  { s.o.p("unsuccesfull");
  }
String sqll = "seleer * from sradenr";
ResaltSer rs = sr.execureQuery(sqll);
while (rs.naxr())
{ s.o.p(rs.gerInr(1));
  s.o.p(rs.gerString(2));
  s.o.p(rs.gerString(3));
  S.o.p(rs.gerDouble(4));
   s.o.p(" ");
}
rs.close();
sr.close();
con.close();
} carch (Exceerion e)
  {
    s.o.p(e);
  }
}
}
/*
 compin
  Javac Jdbc[x3.Java
8) ser
  classparh = %.classpath%;c:\bali\dbc\
                    mysql.Jar;
```

```
/* Using PreparedStmt & Type4 in MySql DB        27/06/06.
0) install MySql
1) Starting mysql
2) Creating DB
3) Open DB
4) Creating table
5) insert data
6) program */

import java.sql.*;
class Jdbc2x4
{
  psvm(String as[])
  {
    int sno = Integer.parseInt(as[0]);
    String sname = as[1];
    String phone = as[2];
    Double bal = Double.parseDouble(as[3]);

try{
  Class.forName("com.mysql.jdbc.Driver");
  Connection
  con = DriverManager.gerConnection(
    "jdbc:mysql://127.0.0.1/b7db","root",
     "javasree");
  PreparedStatement ps = con.prepareStatement
  ("insert into Student values(?,?,?,?)");
                    → Position of place holder
  ps.setInt(1, sno);
  ps.setString(2, sname);
  ps.setString(3, phone);
  ps.setDouble(4, bal);
  boolean b = ps.execute();
  if(!b)
  {
  s.o.p("insertion successfully");
  }
  else
  {s.o.p("insertion Unsuccessfull");
  }
```

```
PreparedStatement ps1=con.prepareStatement
("update student set bal=?");
ps1.setbal(999)
ps1.setDouble(1,999.98);
int x = ps1.executeUpdate();
S.O.P(x);
PrepareStatement ps2
     = con.prepareStatement("select *
  from student where sname=?");
ps2.setString(1,"nikai");
ResultSet rs=ps2.executeQuery();
while(rs.next())
{
  S.O.P(rs.getInt(1));
  S.O.P(rs.getString(2));
  S.O.P(rs.getString(3));
  S.O.P(rs.getDouble(4));
  S.O.P(" ");
}
rs.close();
ps2.close();
ps1.close();
ps2.close();
con.close();
}
catch(Exception e)
  { S.O.P(e);
  }
}
```

/* Using CallableStatement and Type 4 in Oracle DB.

1) creating table

```
create table mytab
(a number(3),
 b number(5),
 c number(5),
 s number(6),
);
```

2) creating stored procedures

```
create or replace procedure
   myadd(a number, b number, c number)
as s number;
   begin s: a+b+c;
insert into myrab values(a,b,c,s);
 end;
```

6) program */

```
import java.sql.*;
class JdbcExs
{
  p s v m(String asc[])
  { int a = Integer.parseInt(as[0]);
    int b = Integer.parseInt(asc[1]);
    int c = Integer.parseInt(as[2]);
  try {
   class.forName("oracle.jdbc.driver.
                  OracleDriver");
```

```
connection con = DriverManager.getConnection
  ("jdbc:oracle:thin:@127.0.0.1:1521:
    "JAVASRE", "scott", "tiger");
CallableStatement cs = con.prepareCall("
        { call myadd(?,?,?)}");

cs.setInt(1,a);
   cs.setInt(2,b);
   cs.setInt(3,c);
  int x = cs.executeUpdate();
  s.o.p(x);
PreparedStatement ps2 =
 con.prepareStatement("select * from myrab");
 ResultSet rs = ps2.executeQuery();
 while(rs.next())
{ s.o.p(rs.getInt(1));
  s.o.p(rs.getInt(2));
  s.o.p(rs.getInt(3));
  s.o.p(rs.getInt(4));
  s.o.p("");
```

2) Creating stored procedures

```
create or replace procedure
empcal(eid in number, hra in number,
da in number, ta in number, ta in
number, de in number, gross out number,
nersal out number) as

begin
gross:=hra+da+ta;
nersal:=gross-de;
insert into myemp values
(eid,hra,da,ta,de,gross,nersal);
end;

/* program */

import java.sql.*;

class jdbc8x6
{
```

```
p s v m(String as[])
{
  int eid = Integer.parseInt(as[0]);
  int hra =   "     "     (as[1]);
  int ta =    "     "     (as[2]);
  int da =    "     "     (as[3]);
  int de =    "     "     (as[4]);
  int gross = 0; "  "     (as[5]);
  int nersal = 0; "  "     (as[6]);

try {
Class.forName("oracle.jdbc.driva.Oracle
                -Driva");

Connection con = DrivaManager.getConnection(
  "jdbc:oracle:thin:@localhost
       :1521:"JAVA
       sree","scott","tiger");

CallableStatement cs = con.prepareCall
    ("{call empcal(?,?,?,?,?,?,?)}");

cs.setInt(1,eid);
cs.setInt(2,hra);
cs.setInt(3,ta);
cs.setInt(4,da);
cs.setInt(5,de);
```

DataBase MetaData: 29/06/06

This interface is used to find the information abt database.

DataBaseMetaData ref:

```
import java.sql.*;
class Jdbc2x7
{
 public static void main(String args[])
 {
  try{
  Class.forName("oracle.jdbc.driver.
                              OracleDriver");

  Connection con = DriverManager.getConnection
   ("jdbc:oracle:thin:@localhost:1521:JAVA
              SREE","scott","tiger");

  DatabaseMetaData dmd = con.getMetaData();
  S.O.P(dmd.supportStoredProcedures());
                                    );
   S.O.P(
  } closing ... all dbs
 }
```

```
catch(Exception e) { S.O.P(e); }
```

ResultSetMetaData Interface:

The ords in this interface are used to find the information about resultset.

```
import java.sql.*;
class Jdbc2x8{
  p s v m(String args[])
  {  try{
       Class.forName("Oracle.jdbc.driver.
                              OracleDriver");

     Connection con = DriverManager.getConnection
     ("jdbc:oracle:thin:@localhost:1521:JAVA
                    SREE","scott","tiger");

   Statement st = con.createStatement();
    String sql = "Select * frm student";
    ResultSet rs = st.executeQuery(sql);
    closing dbs.
```

```
ResultSetMetaData rmd = rs.getMetaData();
S.o.p(rmd.getColumnCount());
while(rs.next()) {
for(int i=0; i<=rmd.getColumnCount();
                                i++)
  {
    s.o.p(rmd.getColumnName(i));
    s.o.p(rmd.getColumnType(i));
    s.o.p(rmd.getColumnClassName(i));
    s.o.p(" ");
  }
}
```

Scrollable ResultSet :

ResultSets are by default forward only/-. We can make resultset Scrollable by using the following constants in resultset Interface.

1) TYPE_SCROLL_SENSITIVE
2) TYPE_SCROLL_INSENSITIVE

when u make ResultSet scrollable, U can move the cursor in forward direction & backward direction.

1) rs.next() → move the cursor forward.
   ↳ it will check whether the record (row) is there or not. if it is return true & move @ next pr, any way..
   else false
2) previous() → move the cursor back.
3) beforeFirst() / first() / afterLast() / last() } move the cursor from any to Specified one. R/T void.
4) a isFirst() / b isLast() / b isBeforeFirst() / b isAfterLast() } R/T boolean To check whether the cursor is present to Specified position/not.

⑤ booleanAbsolute(i) → Cursor will be moved to specified pos i.

⑥ Relative(i) → it may be +ve or -ve

for relative mtd we can provide +ve and -ve values, when we use +ve values, cursor will be moved in the forward direction; by specified no. of rows from the current position. When we use -ve value, cursor will be moved in backward direct, by specified no. of rows from current position

Eg:

```
import java.sql.*;
class JdbcEx9 {
  p s v m(String args)
  } try {
Class.forName("oracle.jdbc.DriverManager.OracleDriver");
Connection con = DriverManager.getConnection
    ("~~create~~jdbc:oracle:thin:@localhost:1521:JAVASREC","scott","tiger");
Statement st = con.createStatement(),,
  String sql = "Select * from student";
ResultSet rs = st.executeQuery(sql);
  s.o.p("Records in the forwardorder");
while(rs.next())
{
s.o.p(rs.getInt(1));
s.o.p(rs.getInt(2));
s.o.p(rs.getString(3));
}
s.o.p("Records in reverse Order")
while (rs.previous())
{
Statement st = con.createStatement
(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY)
```

```
while(rs.previous())
{ s.o.p(rs.getInt(1));
  s.o.p(" "+rs.getString(2));
  s.o.p(" "+rs.getString(3));
  s.o.p(" "+rs.getDouble(4));
}
s.o.p("Third rec details");
rs.absolute(3);
s.o.p(rs.getInt(1));
s.o.p(rs.getString(2));
s.o.p(rs.getString(3));
s.o.p(rs.getDouble(4));
s.o.p("moving the cursor to before first");
rs.beforeFirst();
s.o.p("Records in reverse Order");
rs.afterLast();
while(rs.previous())
{ s.o.p(rs.getInt(1));
  s.o.p(rs.getString(2));
}
```

Sensitive → Static
Sensitive → dynamic
Insensitive → dynamic

we can scroll using sensitive & insensitive But we can't update.
Insensitive db.
By default resultSets are readable & forward Only. 30/06/06

Updatable ResultSet:
We can divide the resultSets into 2 types:

1. StaticResultSets (or)
   ReadOnlyResultSets
2) Dynamic or UpdatableResultSets.

- Bydefault ResultSets are readOnly.
  To make the result sets updatable, we are using following constants.

1. ResultSet.TYPE_SCROLL_SENSITIVE

2. ResultSet.CONCUR_READ_ONLY

Statement st = con.createStatement(ResultSet.
TYPE_SCROLL_SENSITIVE, ResultSet.
CONCUR_UPDATABLE);

	READ_ONLY	UPDATABLE
FORWARD_ONLY	Default forward read	forward read
SCROLL_INSENSITIVE	scrollable readonly	scrollable readonly
SCROLL_SENSITIVE	scrollable readable	scrollable updatable

Eg: 8

// Updatable and Scrollable resultsets

// Mysql typeIV driver:

/* set classpath=%classpath%;c:\bb\bdbc\mysql-connector-java-3.1.12-bin.jar; */

```
import java.sql.*;
class KJdbc2x10
{
    p s v m(String as[])
    {
        String sname, phone;
        int sno;
        double bal;
        try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getCon-
                ection("jdbc:mysql://localhost/b7db",
        "root", "javasree");
        Statement st = con.createStatement
(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet
    .TYPE_CONCUR_UPDATABLE);
ResultSet rs = st.executeQuery("select * from
                student");
while(rs.next())
{ sno = rs.getInt(1);
```

```
sname = rs.getString(2);
phone = rs.getString(3);
bal = rs.getDouble(4);
s.o.p("..."+sno+"..."+sname+"..."+phone+"..."+
bal);
}
rs.absolute(2);
rs.updateString(2,"srinivas");
rs.UpdateString(3,"aaaaa");
rs.UpdateRow();
rs.absolute(3);
rs.moveToInsertRow();
rs.updateInt(1,111);
rs.updateString(2,"asdv");
rs.updateString(3,"jatg");
rs.updateDouble(4,999);
rs.insertRow();
rs.absolute(3);
rs.deleteRow();
s.o.p("------");
s.o.p("reverse Order");
rs.afterLast();
while(rs.previous())
{ sno = rs.getInt(1);
sname = rs.getString(2);
phone = rs.getString(3);
bal = rs.getDouble(4);
s.o.p("..."+sno+"..."+sname+"..."+phone+"..."+bal);
}
//step 6: cleanup
rs.close();
con.close();
}
catch(Exception e)
{ s.o.p(e);
}
}
```

11/07/06

Batch Updates:

Till now in all the examples using Stmt or prepare stmt, we are submitting one query at a time.

If i want to execute more than one query, I have to issue more than one db calls. This ↑ n/w traffic & degrades as appln performance. My requirement is that, I want to execute more than one sql Query in single db call. This we can do with Batch updates.

38 Q// BatchUpdates, MySql type 4 driver

```
import java.sql.*;
class JdbcEx11
{
    psvm(String args) {
```

```
int sno = 0;
String sname = null;
String phone = null;
double bal = 0.0;
try {
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://locahost/b7db", "root", "javasree");
Statement st = con.createStatement();
String sq1 = "insert into student values (1,'aa','1111',1000)";
String sq2 = "insert into student values (2,'bb','2222',2000)";
String sq3 = "insert into student values (3,'cc','3333',3000)";
String sq4 = "insert into student values (4,'dd','4444',4000)";
```

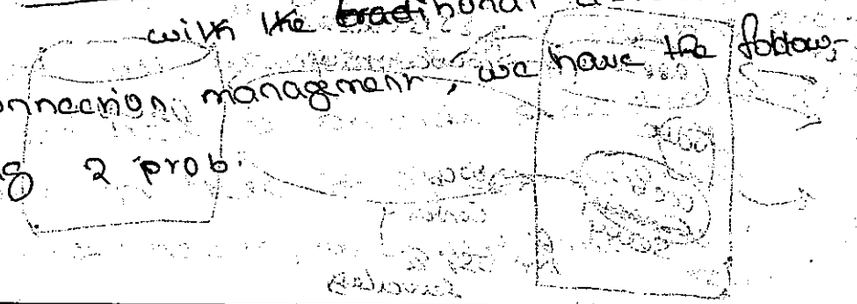
```
sr.addBatch(sql1);
sr.addBatch(sql2);
sr.addBatch(sql3);
sr.addBatch(sql4);

int x[] = sr.executeBatch();
for(int i=0; i<x.length; i++)
{
System.out.println(x[i]+" ");
}
String sql="select * from student";
ResultSet rs=sr.executeQuery(sql);
while(rs.next())
{
sno=rs.getInt(1);
sname=rs.getString(2);
phone=rs.getString(3);
bal=rs.getDouble(4);
System.out.println(sno+" "+sname+" "+phone+" "+bal);
}
// step 6: cleanup
rs.close();
sr.close();
con.close();
} catch(Exception e)
{
System.out.println(e);
e.printStackTrace();
}
```

Advanced Jdbc

1. Using the package java.sql
2. Connection Pooling mechanism:
2. Transaction Management
3. RowSet Mechanism.

1. Connection Pooling

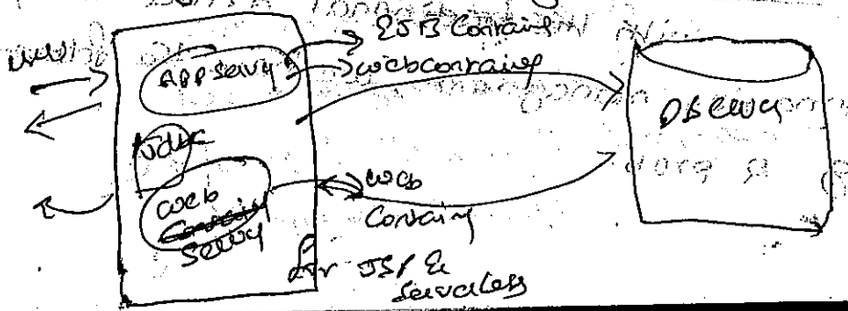
with the traditional driver manager connection management, we have the following 2 prob:



a) we are hardcoding the driver class, url, username/password exactly in the jdbc prog. when i change any one of these details, we have to modify all jdbc programs. This is the maintainance problem.

b) In this everytime we are taking the new connection and after using we are closing the connection. Opening & closing the connection every time is very expensive (cost, time) in distributed systems.

we can solve above 2 problems, using connectionPooling mechanism



ConnectionPooling mechanism is available in any appServers or webServers. AppServers contain 2 containers called EJB & web container. whereas webServer contain web container.

ConnectionPool is a set of connections created by the container when server startup. This set of connections is also called as DataSource.

First we have to configure the connectionPool in any AppServer (or) webServer by providing Driver class name, url, username, pass & Database etc.. we have to provide some dataSource name for these set of Connections & also we have to provide DataSource JNDI name.

(Datasource JNDI name is nothing but a name which is used to bind the DataSource object in the JNDI registry).

We are going to use JNDI naming service to bind & lookup the datasource objects.

Sun has provided JNDI API for the developers to access JNDI reg. The package provided is javax.naming.

DataSource obj will be registered binded in the JNDI registry by the container automatically. As a developer we have to use lookup() to get the datasource obj. & the connection from the datasource obj.

Steps to configure connectionpool

Steps to install weblogic Server (AppServer): 8.1

1) click on platform 8.1.1 - win32
2) click on next
3) select yes & radio button & click on next
4) give the bea home directory as d:\bea & clk on next.
5) Select custom install type and clk on next.
6) de-select integration. and portal & clk on next
7) clk on next
8) clk on next.
9) deselect XMLSpy & Sun quick start & clk on done
10) clk on next

11) click on next

12) click on next

13) provide username & password & clk on next

14) clk on next

15) provide Domain/configuration name & clk on create.

16) click on done

Steps to Configure Stark weblogic

console:

1. Start the weblogic server.

Start → programs → Bea-platform 8.1 → weblogic user-projects → javasrce → Stark server

2) open the browser & type the following url http://localhost:7001/console

3) provide the username & password

4) Steps to Configure connection pools

1) services → Jdbc → connection pools

2) clk on Configure a new JDBC Connection pool... Link

3) select DarabaseType: oracle & clk on deiveType: other

Continue...

4) provide the following details

1) connection pool name: B7pool

2) driver class: Oracle.jdbc.driver.OracleDriver

3) url: jdbc:thin:@localhost:1521:JAVA

4) username: scott

5) password: scottTiger

6) confirmpassw: Tiger

& clk on. test Driver Configuration

5) clk on create & deploy

Steps to configure DataSources:

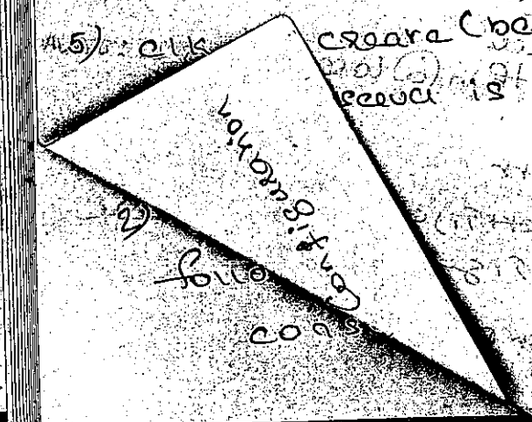
1) services → jdbc → datasources

2) clk on. Configure a new Jdbc DataSource

3) provide the datasource name & datasource jndi name (jdbc/B7pool)

4) select the pool name (B7pool) & click on continue next

5) create (before make sure is checked)



Eg: // CPM Example

import java.sql.*;

import javax.sql.*;

import java.util.*;

import javax.naming.*;

class CPMDemo

{ p s v m (String args[])

{ try {

Properties p = new Properties();

p.put(Context.INITIAL_CONTEXT_FACTORY,

"weblogic.jndi.WLInitialContextFactory");

p.put(Context.PROVIDER_URL, "t3://

localhost:7001");

Context ctx = new InitialContext(p);

Object o = ctx.lookup("jdbc/B7POOL");

```
DataSource ds = (DataSource)
Connection con = ds.getConnection();
Statement st = con.createStatement();
String sql = "select sname,email from
              b7student";
ResultSet rs = st.executeQuery(sql);
while(rs.next())
{
   s.o.p(rs.getString(1)
   s.o.p(rs.getString(2)
   s.o.p(rs.getInt(3));
}
rs.close();
st.close();
con.close();
}catch(Exception e)
{
  e.printStackTrace();
}}}
```

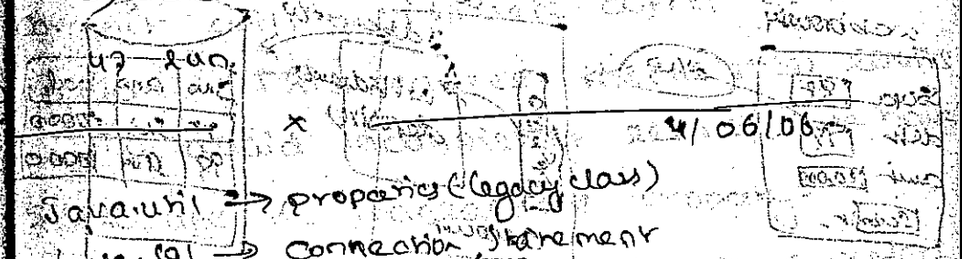
1) run setEnv

(d:\local\ejb-projects\domain\bin>startWebLogic.cmd)

setEnv.cmd

2) compile

3) set classpath=%.classpath%;



→ Context is the runtime environment which is related to security identity - authn.

→ weblogic.jar have other weblogic.JNDI package.

→ JNDI registry will used to bind any type of Object whereas as ONE registry will be used to bind java class object only.

→ JNDI registry will be maintained by server

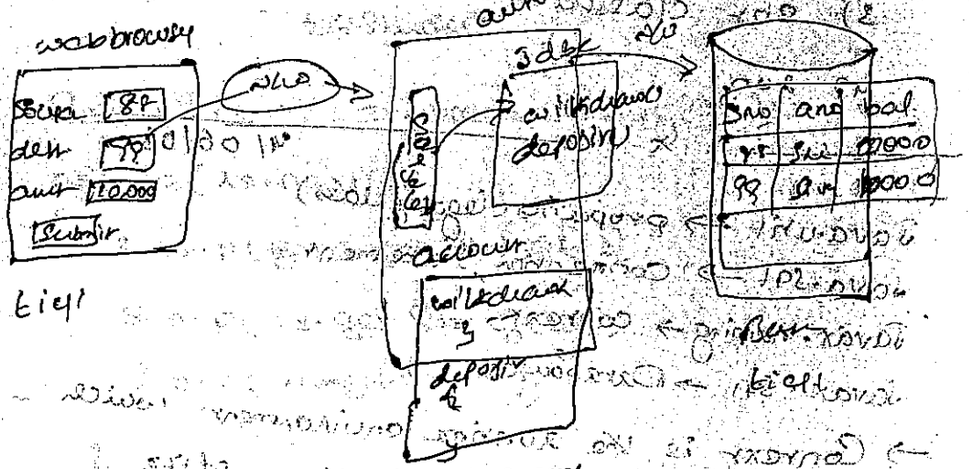
LDAP → it will start automatically when application

is running (weblogic).

LDAP → (light weight Directory access protocol)

→ vendor specified protocol

Transaction:



for Eg: I want to transfer the funds from one account to another account

for this we have to call 2 methods called deposit & withdraw. These 2 operations are update operations. i.e we are going to issue 2 update SQL stmts to the database.

In Jdbc apps by default auto-commit is true. When we issue 2 update commit will be issued after each update. This causes inconsistent results.

To avoid inconsistent results, we have to manage the transactions properly.

def: Transaction is working our performing series of database operations as a single unit.

If all the operations are successful in the unit, then issue the commit. If any one operation is failed issue the rollback.

The following operations we have to do, when managing the transaction with Jdbc code

1) Disable the autocommit

con.setAutoCommit(false);

2) Begin the transaction

con.begin();

3) Write call 2 or 3 of Database Operations

m1();
m2();
m3();

4) Verify that all operations are successful

If any one is failed

If all are successful issue commit

con.commit();

else issue rollback

con.rollback();

① What is the default value of autoCommit?
(True)

② Means we manage transaction with JDBC code? Yes

③ How to manage transactions with JDBC code?

In connection they are provided commands.

1. setAutoCommit()
2. begin
3. commit()
4. rollback();

** ① We have 4 transactional isolation levels?

1. ReadUncommited (READ-UNCOMMITED).
2. ReadCommited
3. Repeatable Read
4. Serializable

We can set the isolation level to any transaction, using the following mtd.

setTransactionIsolation(int);

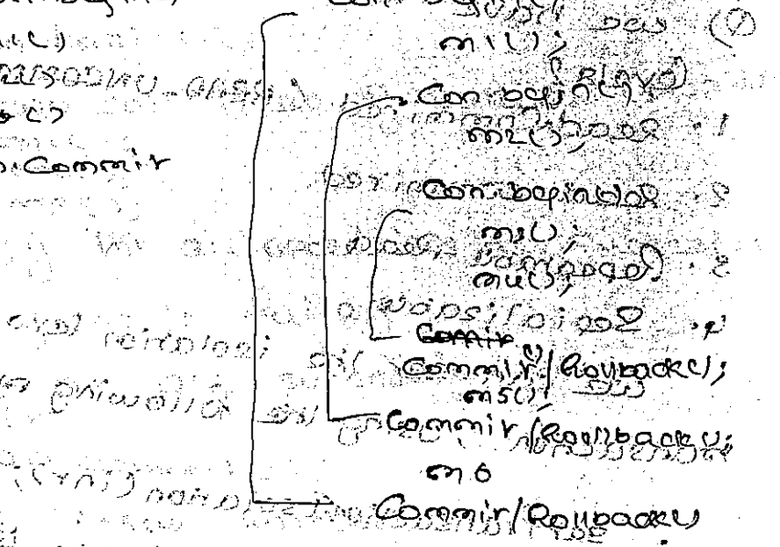
We can get the available isolation level for the given transaction using the following mtd.

getTransactionIsolation();

We have 2 types of transactions:

1. FlatTransactions.
2. NestedTransactions

Eg: Con.begin() Con.commit



Oracle supports only flat transactions.

It doesnot support nested transactions.

AccountClass

```
void Transfer()
{
  try
  {
    deposit(5000,88);
    withdraw(5000,77)
    Con.commit();
  }
  catch(Exception e) { Con.rollback(); }
}
```

Accounttable

ano	aname	bal
99	abc	5000
88	xyz	10000
77	efg	2000

deposit()
{
}

withdraw()
{
}

getBalance()
{
}

RowSets Mechanism:

- is nothing but implementation of javax.sql.RowSet interface, which is implementing java.sql.ResultSet.

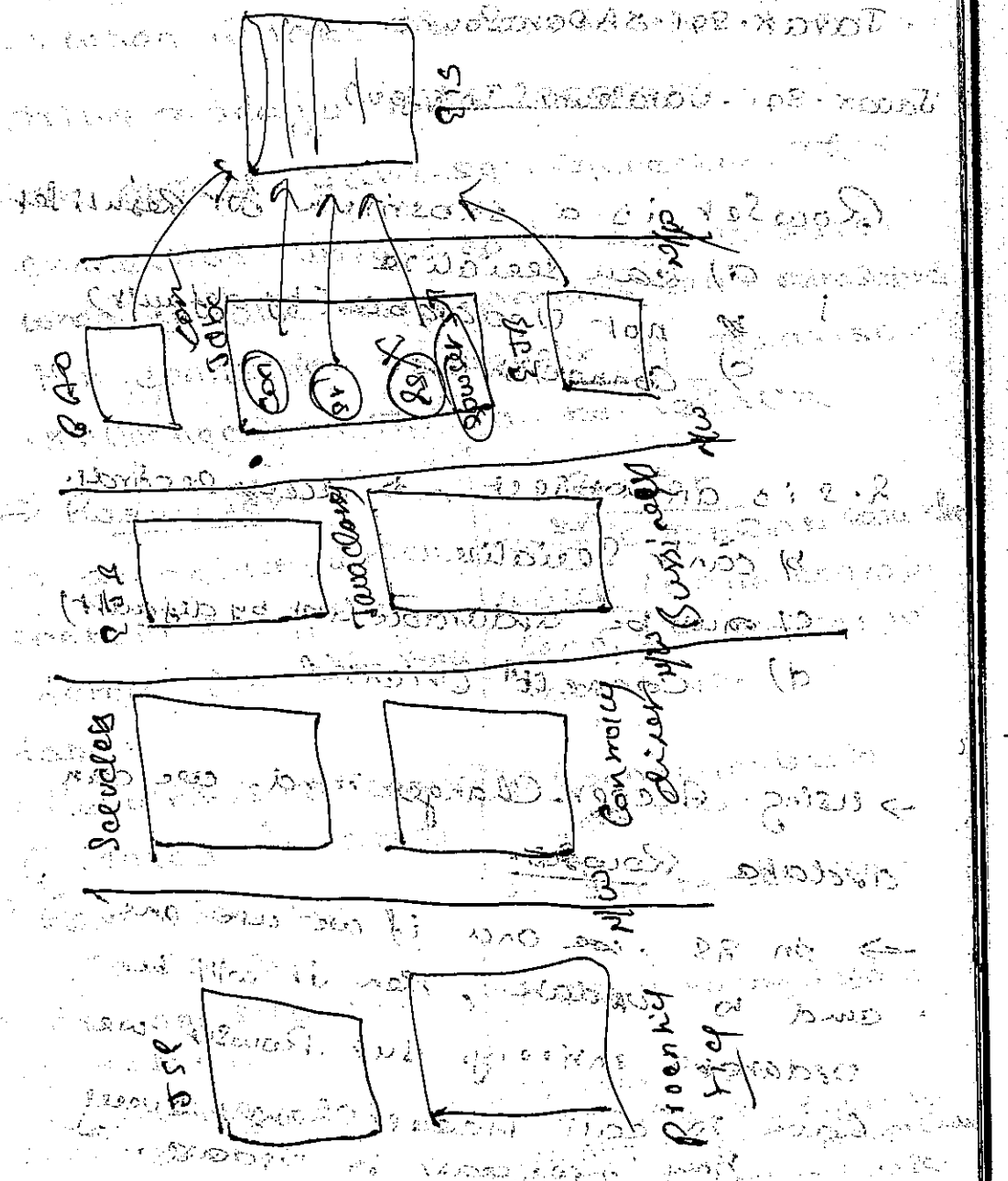
RowSet extends ResultSet implementing in the sense inheriting properties.

we can't serialize ResultSet object. R.S is connected object.

Connection, Statement, ResultSet which are in sql java.sql.* ; are not able to serialize.

we can serialize javax.sql.*RowSet

(ConnectionPooling etc)



Javax.sql.XADataSource

Javax.sql.DataSource (Interface)

RowSet is a subinterface for ResultSet

a) can serialize
b) not updatable (by default)
c) Connectionless

R.S is an object, to access records

b) can't serialize
c) must be updatable (not by default)
d) Connected Oriented

→ using AcceptChanges() method we can update RowSet.

→ In RS, once if we use one cmd to update, then it will be updated entirely. But RowSet we have to call AcceptChanges() every time if we want to update.

→ when u take the connection from driver manager, the connection is an object Java.sql.Connection. This connection object can't be serialize. when u take the connection from datasource then connection is object of javax.sql.XAConnection. This can be serialize

→ ResultSet:
is an object, which contain set of rec fetched from database table.

2) ResultSets are connected Oriented in nature
3) ResultSets can't be serialize
4) are not scrollable by default
5) ResultSet can be updatable

RowSet

RowSet is subclass ResultSet which is also contain set of rec similar to ResultSet.

2) Connectionless in nature.
3) can be Serialize.
4) Scrollable by default.
5) RowSets can't be updatable by default. we should call AcceptChanges() method to refresh RowSet

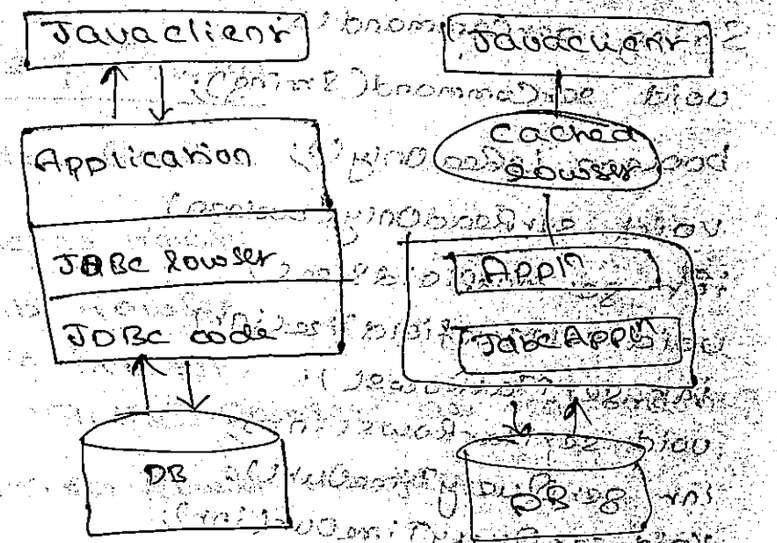
Types of Rowsets:

1. Jdbc Rowsets
2. Cached Rowset
3. Web Rowset
4. Join Rowset

All these Rowsets are implementing the Javax.sql.Rowset interface.

```
interface Rowset extends Resultset
{
    String getUrl();
    void setUrl(String);
    String getDataSourceName();
    void setDataSourceName(String);
    String getUsername();
    void setUsername(String);
    String getPassword();
    void setPassword(String);
    int getTransactionIsolation();
    void setTransactionIsolation(int);
    String getCommand();
    void setCommand(String);
    boolean isReadOnly();
    void setReadOnly(boolean)
    int getMaxFieldSize();
    void setMaxFieldSize(int)
    int getMaxRows();
    void setMaxRows(int);
    int getQueryTimeOut();
    void setQueryTimeOut(int);
    void setType(int)
    void setConcurrency(int)
    void setXXX(int, XXX)
    void clearParameters()
    void execute()
}
```

1. **JdbcRowset**: is a subclass of Rowset interface. It is connection oriented. (almost equals to Resultset).



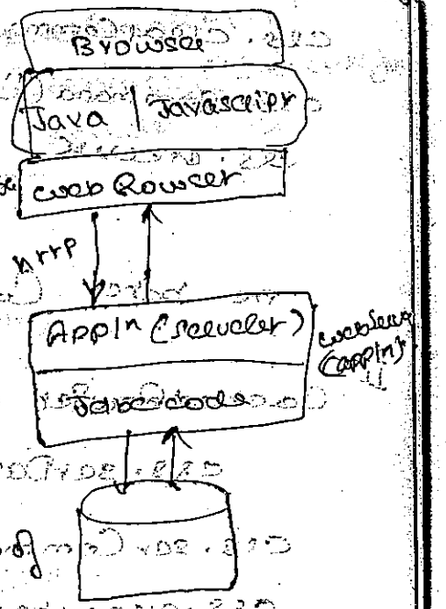
Cached Rowset:

Subclass of Rowset interface & can be serialize. No need to have the Connection to access the CachedRowset for getting data from database & stored all rec in a private cache memory, which we can access without Connection

Java.sun.com/developer/earlyaccess/crs

SDN

Web Rowset:



web Rowset are Connected & serialized But internally uses http protocol to move from one location to another

Join Rowsets: are collection of other Rowsets.

```
CachedRowSet crs = new CachedRowSet();
crs.setURL("jdbc:mysql://localhost/bzdb");
crs.setUsername("root");
crs.setPassword("javaee");
crs.setCommand("select * from student");
crs.execute();
while(crs.next())
{
//
}
```

```
crs.createCommand();
crs.setCommand("select * from student");
crs.execute();

no batch Commands

CachedRowSet crs = new CachedRowSet();
crs.setDataSource("jdbc/BBRoot");
crs.setCommand("select * from student");
crs.execute();
while(crs.next())
{ System.out.println(crs.getString(1));
  System.out.println(crs.getString(2));
}
crs.absolute(3);
crs.updateString("name","Ravi");
=
crs.updateRow();
crs.acceptChanges();
```

```
CachedRowSet crs = new CachedRowSet();
crs.execute(Query("select * from student"));
crs.populate(rs);
crs.execute();
while(crs.next())
  System.out.println(crs.getString(1));
  System.out.println(crs.getString(2));
```

API

SPI: (Service Provider Interface);

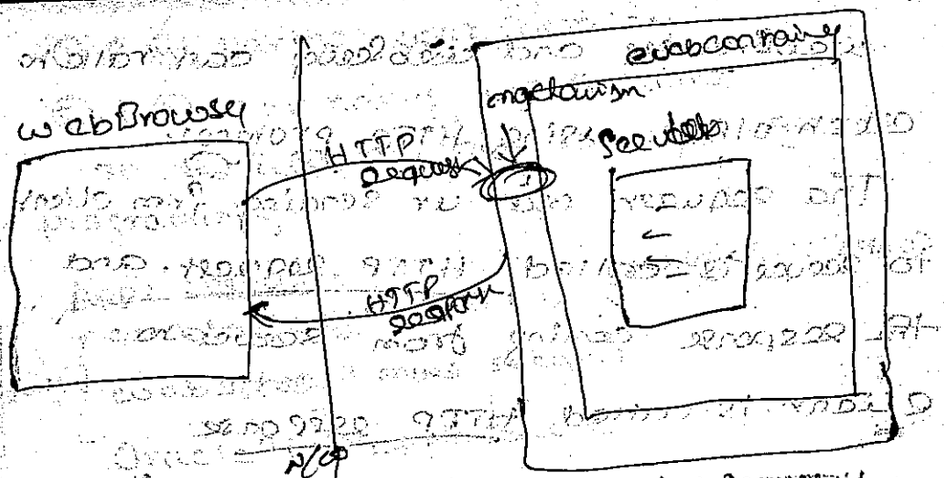
Servlets

Servlet is a technology from sun which is used to develop server side components.

Servlet as a server side component is responsible to receive the request from the client and process the request and send the response back to the client.

web Browser is a client for servlet.

Apache Server → Tomcat ↓ becomes Apache Tomcat ↓ web server.



Tcp → carry data

IP → carry data

WebBrowser is used to send the request and is responsible to raise the response given by the webserver.

Webserver is responsible to receive the request given by the webBrowser and sends the response back to webBrowser.

webBrowser and webServer can talk to each other using HTTP protocol.

The request we are sending from client to server is called HTTP request and the response coming from server to client is called HTTP response

HTTP is a stateless protocol (no "it will not remember our request"). HTTP built upon TCP/IP.

TCP is responsible to control the transmission of the data. HTTP is an underlying protocol which is responsible to carry the data.

Web servers available:

1) Apache Tomcat
2) PWS (personal web server) } micro soft
3) IIS (Internet Information server) }
4) SunDynamics (Sun) not in use

① & ④ having webContainer

② & ③ not having webcontainer.

so ② & ③ are not useful for servlet programming.

App Servers:

- weblogic 8.1 (EJB 2.1)
- websphere (more secure) } buy
- Oracle 9i (appln server & JBoss (opensource) JRun (opensource) Resin (opensource) Pramati (Indian Company) e-matrix Planer (opensource)

• Difference bet appln server & web server -

→ applnServer contains both webcontainer & EJB container where as webserver contains only webcontainer.

→ web server supports servlet & Jsp's only, where as applnServer supports all J2EE technologies.

J2EE

JNDI } apps server
JDBC
servlets] → web server (Container)
JSP
EJB
JMS } apps server
JAAS
JTS
Javamail

Steps to install Apache Tomcat web server: (supports all)

[Go for professional] 98 windows.
web logic doesn't support

Steps:

1) click on Jakarta-tomcat-5.0.16
... r.
... agree
... r
... give the destinat? folder)
... ext

1) Apa...
2) PWS (P...
3) IIS (Intra...
4) netDynamics

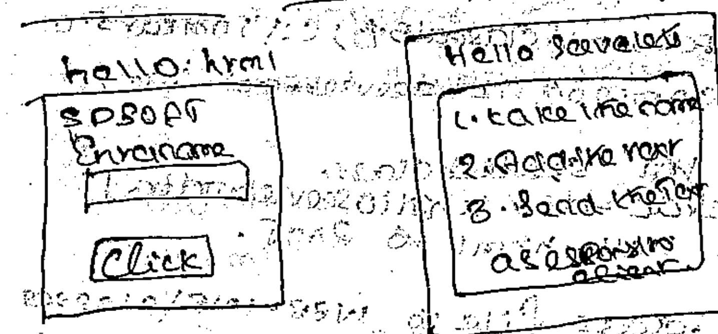
provide port number: 9999
Username : javasree
Password : javasree

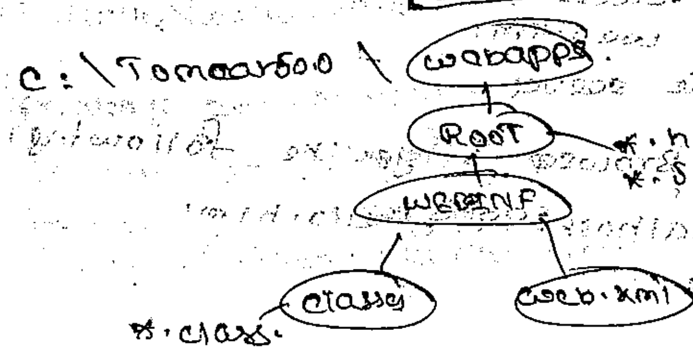
click on next

7) click on install.
8) uncheck all & click on finish.

ASF (apache Software foundat?)
& given the opensources as Jakarta...

First Servlet Example:





① hello.html

② HelloServer.java

③ Update the web.xml

Steps:

1) write hello.html
2) write HelloServlet.java
3) Update web.xml.
4) set the classpath

set classpath=.;classpath;C:\Tomcat\common\lib\servlet-api.jar;

5) compile the servlet class.

javac -d . HelloServlet.java

6) copy the hello.html to ROOT.
7) copy the .class file to WEB-INF\classes
8) update web.xml
9) start the server.
10) open the Browser & type the following.

http://localhost:9999/hello.html

1) write hello.html

```
<html>
<head>
<title>SOSOFT</title>
</head>
<body bgcolor="silver">
<h1 align="center">SOSOFT</h1>
Enter Name: <br>
<form action="/sri.do">
<input type="text"
name="myname"/><br>
<input type="submit" value="greeting"/>
</form>
</body>
</html>
```

2) HelloServlet.java

```
package com.javasrc;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

public class HelloServlet extends
HttpServlet

{

public void service(HttpServletRequest
req, HttpServletResponse res) throws
ServletException, IOException

{

String str = req.getParameter("my name");

str = "<h1>Hello !" + str + "Welcome to
Servlet Programming...";

//sending the output

res.setContentType("text/html");

PrintWriter out = res.getWriter();

out.println(str);

out.close();

}

}

the following tags in web.xml

<servlet-name>hello</servlet-name>

<servlet-class>com.javasrc.HelloServlet</
servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>hello</servlet-name>

<url-pattern>/sri.do</url-pattern>

</servlet-mapping>

Start the servlet

Browser supports: html
Javascript
Jscript
VBscript
Javaprogr

SERVLET

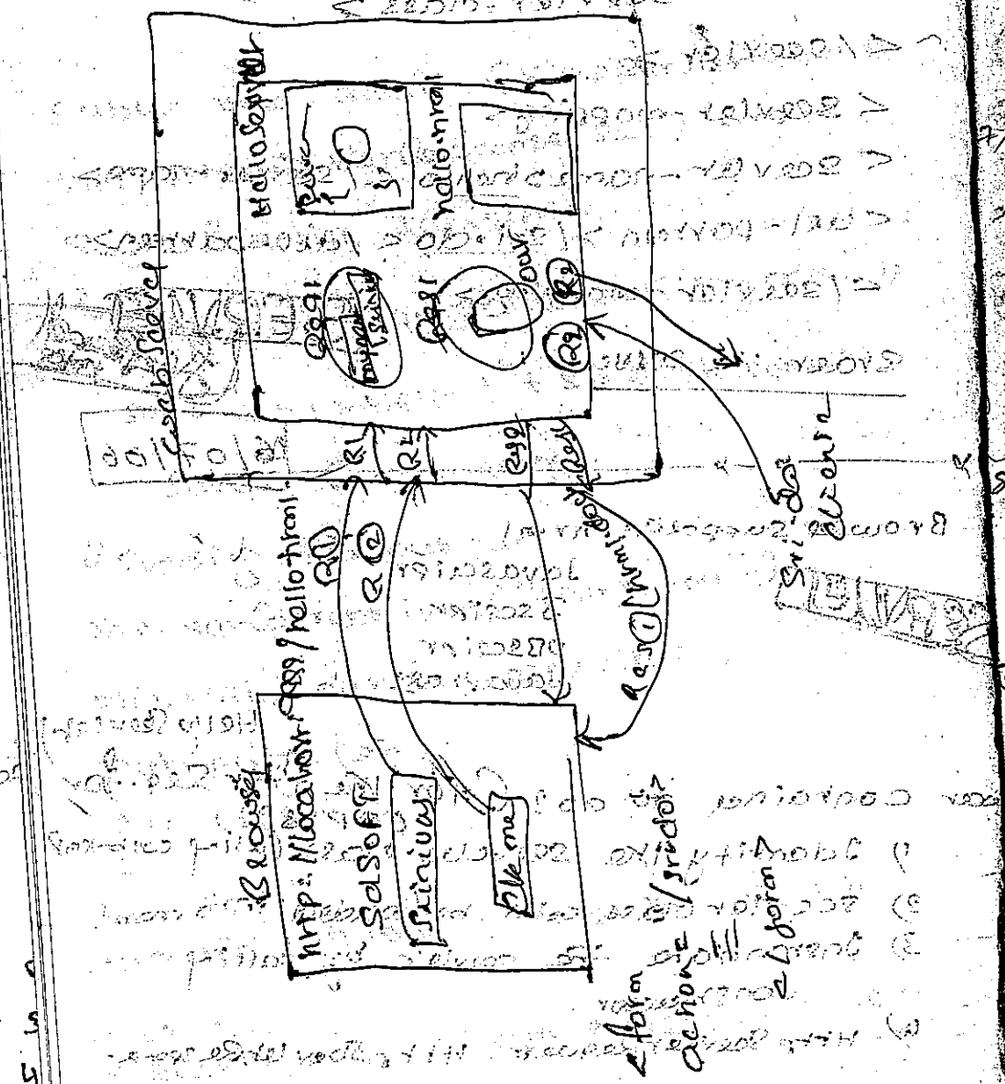
SERVLET

06/07/06

ear containing do? (for the first req. for HelloServlet)

1) Identify the servlet class (using web.xml)
2) servlet class will be loaded into ram
3) Instantiate the servlet by calling constructor.
4) HttpServletRequest, HttpServletResponse

5) service() will be invoked.



5) service() method is in class HttpServlet.

6) one instance is created for servlet for n-no. of requests.

7) After sending 'our' to webserver from browser then immediately request & response objects are destroyed immediately. ('our' is only the url of response).

8) we can't do anything on container objects which are in container (container will take care of all things).

9) n-no. of users can access servlet prog at a time. (Here different users different requests), servlet instance is not synchronized. it is not thread safe.

we are not placing data in instance of servlet, but data is in "request obj" (req. obj. is synchronized).

b) Second onwards:

1) Identify the servlet class
2) no need
3) no need
4) HttpServletRequest, HttpServletResponse
5) Service() method will be invoked

Servlet Life Cycle methods:

[Sun API has only three methods.]

1) init()
2) Service()
3) destroy()

1) init(): → will be called by the container only once in the life cycle.

→ only once it will be called.

→ init() method is used to initialise the servlet instance. init() method will be called by the container only once in the life cycle.

Q) Can I write the constructor inside a servlet?

A) yes, but no use.

Q) can I initialise servlet instance with constructors instead of init() method?

A) No. instance will be created by the container, we don't have chance to pass values using constructors.

2) Service() method:

Service() method contains the code to process the client request. Service method will be called by the container for each request.

3) destroy(): destroy() method is used to destroy() the service instance. (when we shut down the server). Before destroy the instance, it will call the cleanup code, (to close the database).

Sun has provided 2 packages for Servlet

Server developers

1. javax.servlet
2. javax.servlet.http

3. javax.servlet package.

javap javax.servlet.Servlet > servlet.txt

interface Servlet {

void init(ServletConfig) throws ServletException;

ServletConfig getServletConfig();

void service(ServletRequest, ServletResponse)
throws ServletException, IOException;

String getServletInfo();

void destroy();

}

abstract class GenericServlet

implements Servlet, ServletConfig, Serializable

GenericServlet();

void destroy();

String getInitParameter(String);

Enumeration getInitParameterNames();

ServletConfig getServletConfig();

ServletContext getServletContext();

String getServletInfo();

void init(ServletConfig) throws S.E;

void init() throws S.E;

void log(String);

void log(String, Throwable);

String getServletName();

}

JVM specification (Logical)

the physical rep of JVM is JRE

[Compiler, JRE, & all tools in JDK tool kit]
(JIT) JRE OK tool kit

Jar file: Jar → Java archieve

Jar file contains packages & package contains .class files.

If a want to use any .class file which is in the package which is in a Jar file we have to set the classpath for the Jar file.

servletapi.jar:

jar file contains package like javax.servlet, javax.servlet.http)

classes111.jar:

This jar file contains Oracle implemented Jdbc driver class. i.e "oracle.jdbc.driver.OracleDriver"

MySql.jar:

which contains mysql implemented Jdbc driver class, i.e "com.mysql.jdbc.driver"

rt.jar: J2SDK1.5/→J2RE/lib/rt.jar.

tools.jar: J2SDK1.5/../lib

weblogic.jar:

This jar contains the package called weblogic.jndi. which contains the class called weblogic.jndi.WLInitialContextFactory

abstract class HttpServlet extends GenericServlet implements Serializable

public HttpServlet();

protected void doGet(HttpServletRequest, HttpServletResponse) throws

protected long getLastModified(HttpServletRequest);

protected void doHead(HttpServletRequest, HttpServletResponse) thr se, ie

protected void doPost(HttpServletRequest, HttpServletResponse) thr se, ie

doPut(HttpServletRequest, HttpServletResponse);

doDelete(" , ");

doOption(

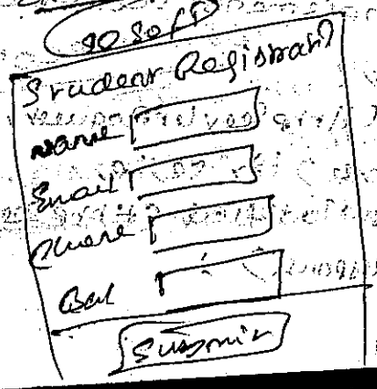
doTrace(

protected void service(HttpServletRequest, HttpServletResponse);

public void service(ServletRequest, ServletResponse)

Eg: Exception

registry.html



RegistrySevlet.java

→ get the data from client

→ Send in back to client

msg "Registration Successful"

Registration Successful
with the following info

name: s
Email: s@com
Phone: 9999
Bal: 0

7/07/06

Servlet API:

packages

Javax.servlet

Javax.servlet.http

Javax.servlet.Servlet (I)

GenericServlet (AC)

ServletRequest (I)
ServletResponse (I)
ServletContext (I)
ServletConfig (I)
ServletException
UnavailableException
RequestDispatcher
*SingleThreadModel (not in use)

HttpServlet
HttpServletRequest (I)
HttpServletResponse (I)
*** HttpSessions } imp
*** Cookies

ServletContext interface

1) ServletContext getContext(String);
2) RequestDispatcher getRequestDispatcher(String);
3) RequestDispatcher getNamedDispatcher(String);
4) Servlet getServlet(String) throws S.E
5) Enumeration getServlets();
6) Enumeration getServletNames();
7) String getServerInfo();
8) String getInitParameter(String);
9) Enumeration getInitParameterNames();
10) Object getAttribute(String);
11) Enumeration getAttributeNames();
12) void setAttribute(String, Object);
13) void removeAttribute(String);
14) String getServletContextName();

ServletConfig Interface

1) String getServletName();
2) ServletContext getServletContext();
3) String getInitParameter(String);
4) Enumeration getInitParameterNames();

5) ~~HttpServlet abstract class~~

6) interface HttpServletRequest
extends ServletRequest {

authentication types:
a) BASIC_AUTH
b) FORM_AUTH
c) CLIENT_CERT_AUTH
d) DIGEST_AUTH

security purpose

* String getAuthType();
* Cookie[] getCookie();
* String getHeader(String);
* Enumeration getHeaders(String);
Enumeration getHeaderNames();
* int getIntHeader(String);
* String getMethod();
* String getPathInfo();
* String getContextPath();
* String getRemoteUser();
* boolean isUserInRole(String); } security purpose

* java.security.Principal
getUserPrincipal();
* String getRequestedSessionId();
* String getServletPath();
* HttpSession getSession(boolean);
* HttpSession getSession();

boolean
isRequestedSessionIdValid();
boolean
isRequestedSessionIdFromCookie();
boolean
isRequestedSessionIdFromURL();

7) interface HttpServletResponse
extends ServletResponse {
a) void addCookie(Cookie);
b) boolean containsHeader(String);
c) String encodeRedirectURL(String);

String encodeURL(String);

String encodeURI(String);

String encodeRedirectURL(String)

String sendRedirect(String)

void setContentType(String);

PrintWriter getWriter();

public interface SingleThreadModel {

HttpRequest Headers:

1. HostName:
2. URL
3. Cookie
4. ConnectionAlive
5. ConnectionTimeOut
6. RemoteUser

HttpRequest: containing 2 parts

Header & Body

→ information about client & Serv.

like hostname, URI, User, Connection timeout.

HttpResponse also having 2 parts

Header & Body.

from Client to Server, Header only we can

Requests only

GET

POST

Delete

option

Trace

put

head

when we use GET mrd (by HttpRequest)

http://sdsoftindia.com/login?un=abc&pw=abc

(data is attached to the url).

So no data in body. body is empty in http Request. (size of url is 2000KB) size limited.

If we use post; data is placed (using variables) in body of Http Request. Data is not attached to url.

: Security purpose (post is better)

: If we don't have data use get() mrd:

→ Get & Post are 2 mrds used to send the req from client to server.

Query String

ServletContext:

ServletContext is an object which contains the info abt entire web appln.

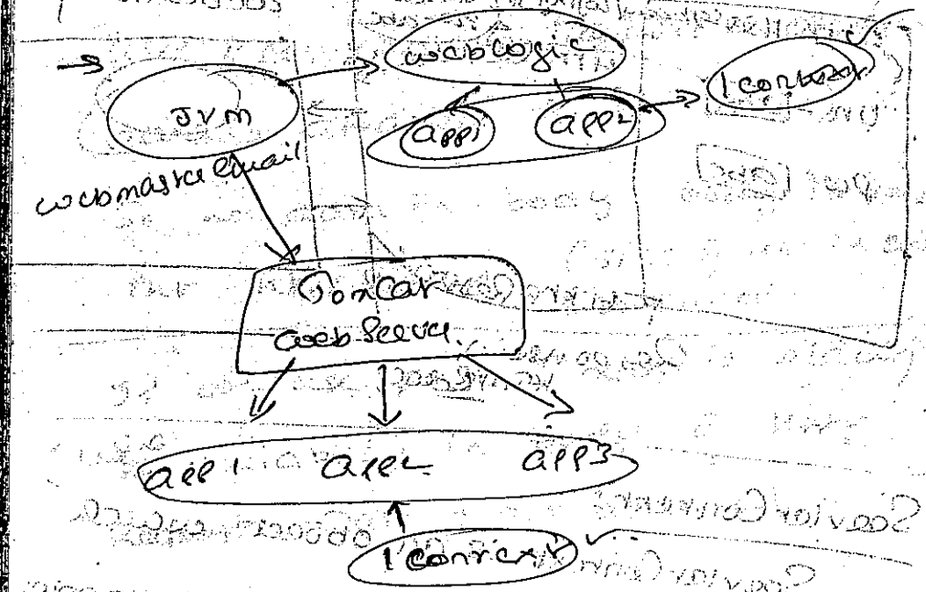
ServletContext object will be created by the container at server startup.

1 context per 1 JVM (per webserver)

1 JVM - 1 Tomcat - 5 appln - then 1 context only 1

1 JVM - 2 servers - 3 (1 ser) appln } then → 1 context
- 2 (2 ser) appln → 2 context

→ web components are nothing but JSPs & Servlets.

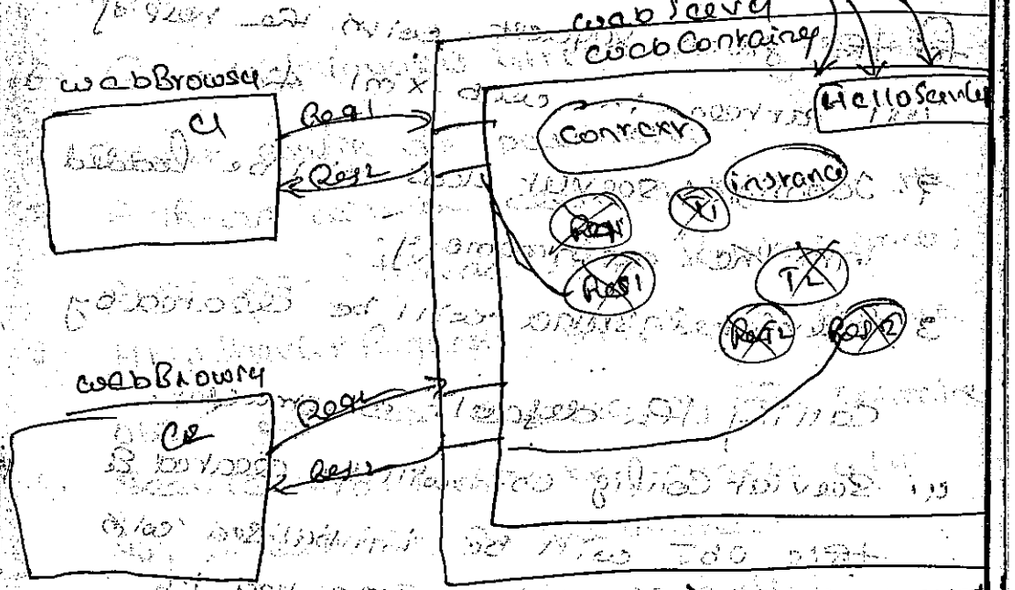


placing the web components in global access place is nothing but 'context'

def: context is a place where we can place the information which can be accessed from all the components of webapplns deployed in one server.

ServletConfig: is an obj, which is created by the container when u send a first req. to a servlet.

each servlet will have its own ServletConfig object. (private data of servlet)



(Req, Res & threads) are destroyed after response completed.

1rst Request to the server

→ when u give the following things will be done by the Conraing

1. Conraing identifies the servlet class for the given request with the help of url pattern in web.xml file
2. Identified servlet class will be loaded into the main memory.
3. Servlet instance will be created by calling the default constructor.
4. ServletConfig obj will be created & this obj will be initialised with the any (parameters specified in servlet tag in web.xml
5. ServletContext obj will be associated with servlet config.

Note: ServletContext already created by the conraing at server startup & is initialized with context parameters specified in the web.xml.

6. Conraing invokes init() mrd by passing servlet config as parameter.
7. A thread will be created to service the given req. (thread is for Servlet instance)
8. HttpServletRequest, HttpServletResponse objs will be created by the conraing
9. Service mrd() will be invoked with req & response as parameters (when we have GenericServlets remove Http in HttpServletRequest/Response)
10. The req will be processed & response data will be placed into the o/p stream,

20/10/23

// example to display the header info.

```
package roca.ex2;
import java.io.*;
import java.util.*;
import java.servlet.*;
import java.servlet.http.*;
public class Demo extends HttpServlet {
public void servlet(HttpServletRequest req,
    HttpServletResponse res)
{ res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  out.println("Header info <br>");
  Enumeration e = req.getHeaderNames();
  while(e.hasMoreElements()) {
    String n = e.nextElement().toString();
    String v = req.getHeader(n);
    out.println("<br>"+n+"...."+v);
  }
}
```

1) accept .. image/gif,
image/x-xbitmap, image/jpeg,
image/pipeg,
2) application/vnd.ms-powerpoint,
application/vnd.ms-excel,
application/msword, */*
3) referer .. http://localhost:9999/show.html
4) accept-language ... en
5) accept-encoding ... gzip, deflate
6) user-agent .. Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.0;
V2-IE6)
7) host .. localhost:9999
8) connection ... keep-alive
9) cookie

out.println("
 protocal: "+req.getProtocol());
out.println("
 Server name"+req.getServerName());

→ interface ServletRequest {
String getProtocol();
String getServerName();
int getServerPort();
String getRemoteAddr();
String getRemoteHost();
int getRemotePort();
}

Servlet Context Vs Servlet Config

10/07/06

LBS: (load balancing server)

Servlet Context	Servlet Config
1) is an object, which is created by the container at server start up.	1) is an object, which is created by the container, when u send a first req.
2) for one or more web apps, which are deployed as one unit in server, which is running under one JVM will have only one servlet context	2) Each servlet has its own servlet config.
3) context obj will be used by all the JSPs & servlets in the appln.	3) Config obj will be used by the servlet for which it is defined.
4) The following tags will be used to specify the context parameters in web.xml [<web-app>] <context-param> <param-name>webmail </param-name> <param-value>sri@sd.com</param-value> </context-param>]	4) The following tags will be used to specify the config parameters in web.xml

SVN / CVS

```
<web-app>
<context-param>
<servlet>
<servlet-name>Hello</servlet-name>
<servlet-class>Com.Sree</servlet-class>
<init-param>
<param-name>city</param-name>
<param-value>Blore</param-value>
<init-param>
</servlet>
```

→ we can't update web.xml at runtime. It's static

Context

```
ServletConfig sc = getServletConfig();
ServletContext sctx = sc.getServletContext();
sctx.getInitParameter("webmail");
```

(or)

```
String mail = getServletConfig().getServletContext().getInitParameter("webmail");
```

Config

```
getServletConfig().getInitParameter("city");
```

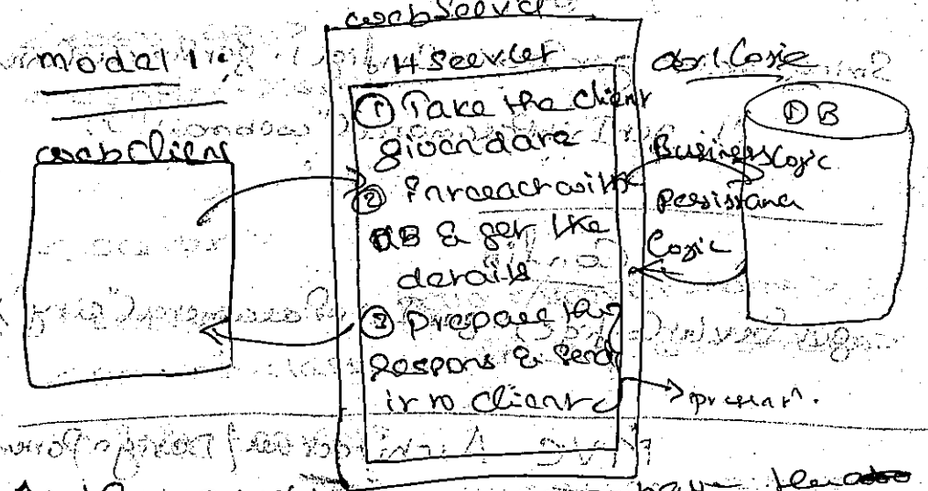
MVC Architecture / Design Pattern

[Model - View - Controller]

MVC is a defacto standard for developing web based appln. Using the following three models.

1. model 1 architecture
2. model 2 or MVC architecture.

Model 1:



In the server, above code have two following:

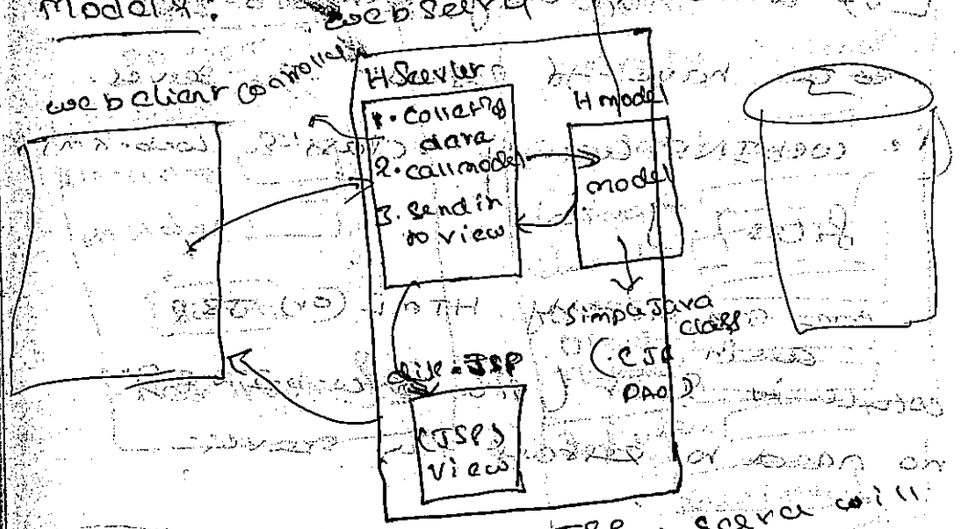
persistance logic — DB connectn code

presentation logic — code used to display the data to the client

to avoid code duplicate in persistance logic & maintenance in presentation logic.

we can go for different servers for diffn Projs.

Model 2: → webserver → (next advance) becomes JSP



[no need to update JSP - server will take care of ir.]

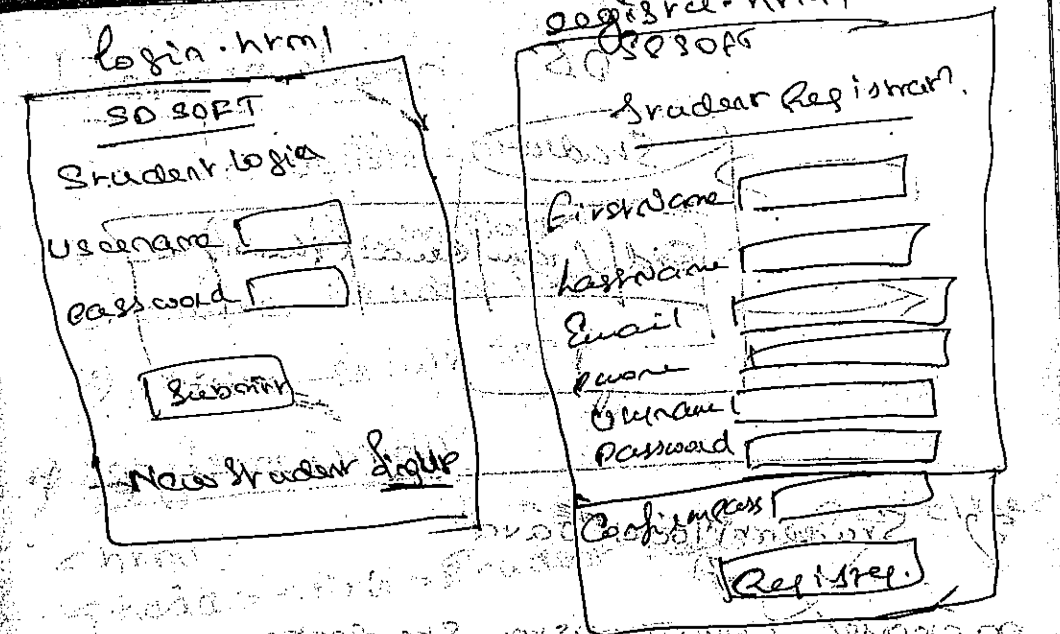
① Advantages of MVC 2 model:

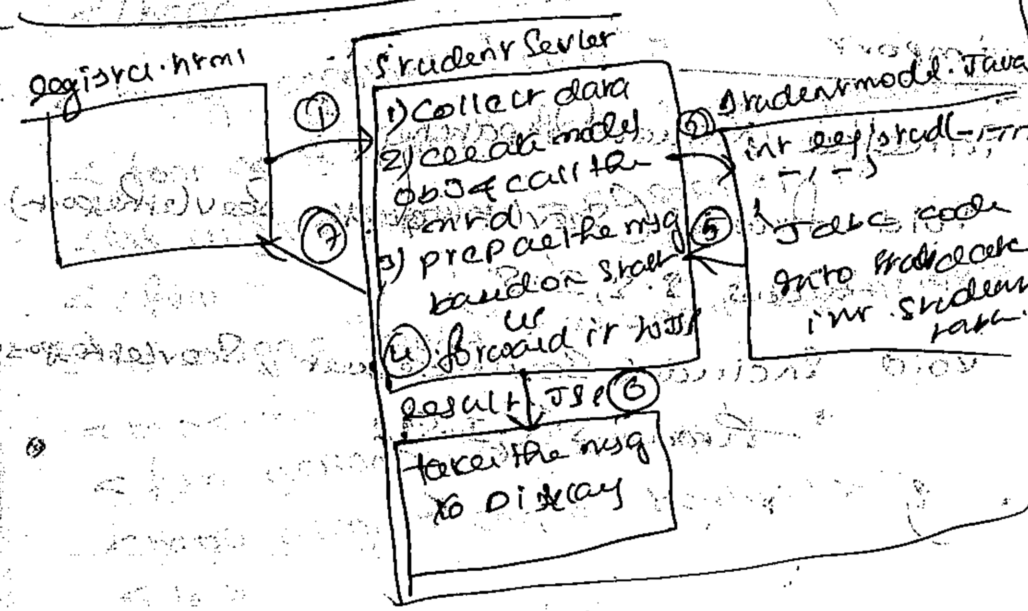
In MVC presentation logic is separate & placed in a separate component called JSP. So I can modify the JSP any time JSP doesn't require any recompilat & redeploy. container will take care of ir.

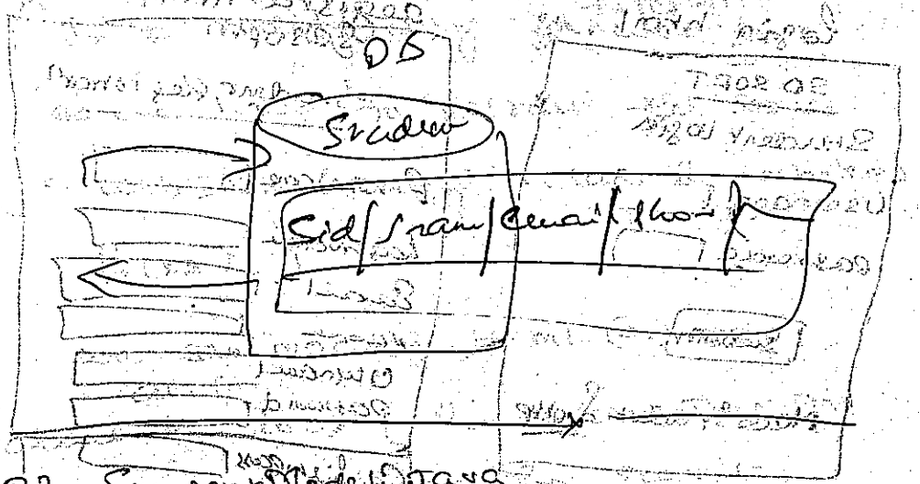
[If we modify any files in web-INF we have to restart the server. (i.e web-INF having class & web.xml files).]

we can modify HTML (or) JSP which are in root (not to web-INF) no need to restart the server.

In model2, we are separating business logic & persistance logic and placing in a java software component call model. This model can be reused in more than one controllers.







2) StudentModel.java

```
package com.javatra.Student;

import ;

interface RequestDispatcher {

    void forward(ServletRequest, ServletResponse)
        throws IOE, SE;

    void include(ServletRequest, ServletResponse)
        throws IOE, SE;

}
```

FirstMVC Example

1) write register.html
2) write StudentModel.java
3) write StudentServlet.java
4) Write result.jsp.

1) register.html

```
<html>
<head><title>Student Registration</title>
</head>
<body bgcolor="silver">
<font color="red" face="monotypecorsiva">
<h1 align="centre">SD SOFT</h1>
</font>
<h2 align="centre">student Registration</h2>
<br><br>
<form action="/register.do">
<table align="centre" border=1>
<tr>
<td>firstName</td>
```

```
<td> inputrype = "text" name="fname" </td>
</tr>
<tr>
<td> LastName </td>
<td> inputtype="text" name="lname" ></td>
</tr>
<tr>
<td> Email </td>
</td> inputtype="text" name="email" ></td>
</tr>
<tr>
<td> phone </td>
<td> inputrype="text" name="phone" ></td>
</tr>
<tr>
<td> Username </td>
<td> inputtype="text" name="uname" ></td>
</tr>
<tr>
<td> password </td>
<td> <input type="password" name="pass" ></td>
</tr>
<tr>
<td> Confirm Password </td>
<td> input type="password" name="pass" ></td>
</tr>
<tr>
<td colspan="2" align="center"><input
type="submit"
value="Registra" ></td>
</tr>
</table>
</form></body></html>
```

(2) StudentModel.java

```
package com.javatpt.student;
import java.sql.*;
public class StudentModel {
public int registraStudent(String fn,
String ln, String em, String ph, String un,
String pw) {
```

```
int x = 0;
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/db",
"javasree");
PreparedStatement ps = con.prepareStatement
("insert into student values(?,?,?,?,?)");
ps.setInt(1,99);
ps.setString(2,fn);
ps.setString(3,ln);
ps.setString(4,pw);
ps.setString(5,ph);
ps.setString(6,os);
ps.setString(7,pw);
x = ps.executeUpdate();
ps.close();
con.close();
}catch(Exception e)
{ s.o.p(e);
}
return x;
}
}//end class.
```

3) StudentServlet.java

```
package com.javasree.student;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class StudentServlet extends HttpServlet
{ public void service(
    HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException{
```

```
String msg = "problem in registration";
String fn = null, ln = null, nam = null;
String ph = null, un = null;
String pw = null, cpw = null;
// Collect the client data;
fn = req.getParameter("fname");
ln = req.getParameter("lname");
em = req.getParameter("email");
ph = req.getParameter("phone");
un = req.getParameter("uname");
pw = req.getParameter("pass1");
cpw = req.getParameter("pass2");
if(pw.equals(cpw))
{
StudentModel sm = new StudentModel();
int x = sm.registerStudent(fn, ln, em,
ph, un, pw);
if(x == 1)
{
msg = "Registration completed Successfully";
}
}
req.setAttribute("info", msg);
RequestDispatcher rd = req.getRequestDispatcher
("result.jsp");
rd.forward(req, res);
}
}
```

(4) result.jsp

```
<html>
<body>
<font color = "red" face = "monotypecorsiva">
<h1 align = "center"> so soft </h1>
</font> <hr>
```

<%

String msg=request.getAttribute("info").

toString();

out.println(msg);

%>

</body>

</html>

Steps

1) set the classpath for servlet-api.jar

Eg mysql.jar ; javac -d . *.java

2) compile

3) copy into tomcat.

4) update web.xml.

create table Student

(snoa registep,

fname char(10),

lname char(15),

email char(10),

phone char(10),

uname char(10), pword char()

create table Student(sno register,

sname char(10));

11/07/06

ANT (opensource)

Ant is a build tool for Java based applns.

(Another need Tool) Ant is an opensource built tool distributed from Apache (ASF (apache s/w foundation)).

s/w required

1) apache-ant-1.6.5.zip

2) unzip the file

3) we get a folder called apache-ant-1.6.5. which contains bin, lib, docs folder etc.

4) bin contains ant.exe

lib contains for of jar files. some of them are (ant.jar), ant-antlr.jar, ant-apache-bcel.jar, apache-ant-log4j.jar, ant-apache-apache-ant-bsf.jar, ant-apache-oro-jar, ant-apache-regexp.jar, ant-javamail.jar.

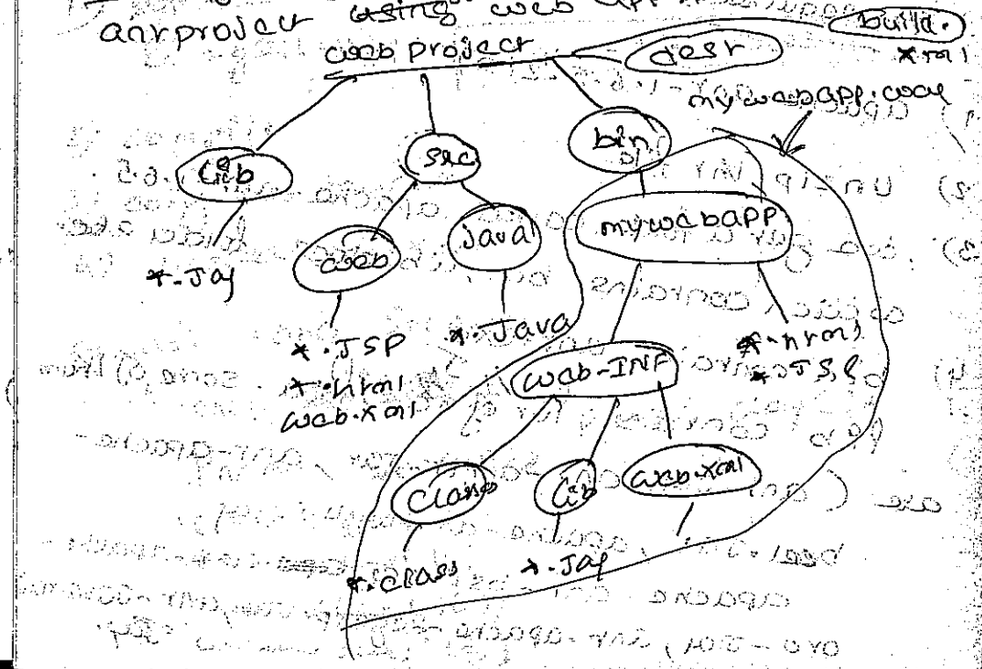
ant-junit.jar; ant-swing.jar; ant-weblogic.jar; ant-commons-logging.jar

e.r.c) Set the path & classpath for ant

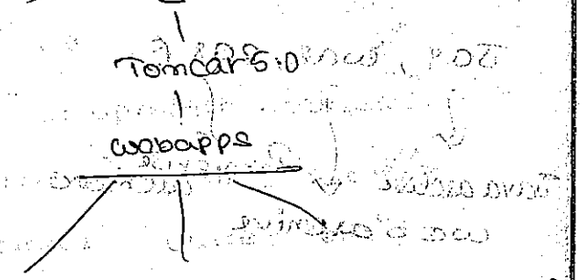
set path = %path%; c:\bz\apache-ant-1.6.5\bin;

set classpath = %classpath%; c:\bz\apache-ant-1.6.5\lib;

Directory Structure: (folder structure) for ant project using for web apps



.war: (web archive)



Tasks (for web based apps)

1. create bin\mywebapp folder.
2. create WEB-INF
3. create classes
4. create Dest.
5. copy jsps & htmls
6. copy web.xml
7. copy the lib folder.
8. compile the Java files
9. create war file.
10. copy the war to tomcat/webapps

→ In Java we have 3 types of archives

Jar, war, ear

Java archive, web archive, Enterprise archive

Jar contains EJB components & supporting files for EJB components

war contains web components & its supporting files

- Ejng files

Ear = war + Jar + appln.xml

→ To automate the tasks for webapplns we have to write a build.xml.

Built.xml is a project (aar project) which contains n-no. of targets.

Build.xml

```
<?xml version="1.0"?>
<project name="webproject" default="all">
<property name="srcdir" value="src"/>
<property name="bindir" value="bin"/>
<property name="libdir" value="lib"/>
<property name="destdir" value="dest" />
<property name="webdir" value="${srcdir}/web"/>
<property name="javadir" value="${srcdir}/java"/>
<property name="mywebappdir" value=
${bindir}/mywebappdir" />
<property name="WEB-INFdir" value="${mywebapp
dir}/
WEB-INF" />
<property name="classdir" value="${web-infodir}/
classes"/>
<property name="tomcar" value="C:/Tomcat5.0/webapps"/>
C:/Tomcat5.0/webapps/"
```

```
top      attribute   value
 ↑          ↑          ↑
<target name="clean">
    <delete dir="${bindir}"/>
    <delete dir="${destdir}"/>
</target>
<target name="create">
                          classesdir
    <mkdir dir="${appdir}"/>
    <mkdir dir="${destdir}"/>
    <mkdir dir="${webinfdir}/lib"/>
</target>
<target name="copy">
                       mywebapp
    <copy todir="${appdir}">
    <fileset dir="${webdir}">
        <include name="**/*.jsp"/>
        <include name=
        <exclude name="**/*.xml"/>
    </fileset>
    </copy>
<target name="copy">
    <copy todir="${webINFdir}">
    <fileset dir="${webdir}">

    <include name="**/*.xml"/>
    <exclude name="**/*.html"/>
    <exclude name="**/*.jspf"/>
    </fileset>
    </clean>

<target name="copy">
    <copy todir="${libdir}">
    <fileset dir="${lib dir}">
    <include name="**/*.jar"/>
    </copy>
</copy> </target>

<target name="compile">
    <javac srcdir="${javadir}"
           destdir="${classesdir}"
    classpathref="myclasspath"/>          Compiling
</target>                                 ali
```

```
<target name="createwar">
    <jar destfile="${builddir}/${appname}.war"
         basedir="${builddir}/${appname}">
    </jar>
</target>

<target name="copywar">
    <copy todir="${myfloor}">
        <fileset dir="${builddir}">
            <include name="**/*.war" />
        </fileset>
    </copy>
</target>
```

creating war file

(1) 7 targets

createall	clean
copy	copywar
compile	createwar

```
<path id="myclasspath">
    <fileset dir="${libdir}" />
</path>
```

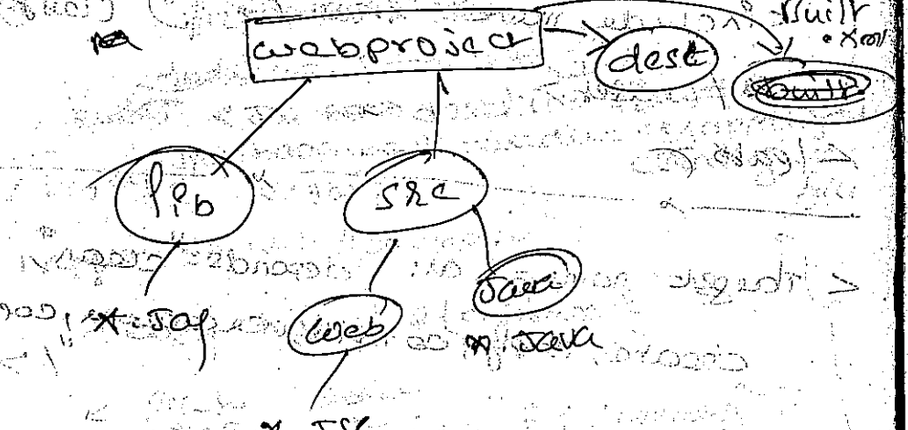
to server classpath

```
<include name="*.jar" />
</fileset>
</path>
```

```
<target name="all" depends="clean"
    create, copy, compile, createwar, copywar
</target>
</project>
```

E:\b7\Servlet\webproject> ant

http://localhost:8080/Mywebapp/req.html



after completion save & run ant

(set path & classpath) Ant &

Run Ant

http://localhost:8080/Mywebapps/reg.html

use ✓ in web.xml

12/07/06

Log4J

Log4J is opensource, java base login framework distributed from apache group;

framework: reusable, semi-completed architecture

- Logging mle can be useful for many scenarios for e.g: login the database operations into a logfile
- Login the customer information in the appln.

s/w required: 1) Download login-log4j-1.2.13.zip

2) extract the zip

Create a new java project.

Set the classpath for log4j-1.2.13.jar

login - log4j -> dist -> lib -> log4j-1.2.13.jar

steps to use log4j framework:

1. get the logger object.

Logger log = Logger.getLogger();

Logger log = Logger.getLogger(Hai.class);

Logger is available in package called

(javap) org.apache.log4j.Logger (10.EXE)

class Logger extends Category {

Logger(String);

static Logger getLogger(String);

static Logger getLogger(Class);

static Logger getRootLogger();

}

public class Category extends Object

{

protected String name;

protected Volatile Level level;

protected Volatile Category parent;

void addAppender(Appender);

void assertLog(boolean, String);

void closeNestedAppenders();

public void debug(Object);

public void error(Object);

public void fatal(Object);

public void warn(Object);

public void info(Object);

public void info(Object, Throwable);

public void debug(

public void error(

public void fatal(

public boolean getAdditivity();

```
Enumeration getAllAppenders();
Appender getAppender(String);
public Level getEffectiveLevel();
public final String getName();
public final Category getParent();
public final Level getLevel();
public final Level getPriority();
public final log(Priority, Object, Throwable)
public void log(priority, object);
void removeAllAppenders();
void removeAppender(Appender)
void removeAppender(String)
public void setAdditivity(boolean)
public void setLevel(level)
public void setPriority(Priority)
}
```

2) invoke the "configurator"

```
BasicConfigurator.configure()
```

(or)

```
PropertyConfigurator.configure("log4j.properties");
```

[default appender]

```
import org.apache.log4j.*;
public class Hai
{
  static Logger log = Logger.getLogger(Hai.class);
  public static void main(String args[])
  {
    BasicConfigurator.configure();
    log.info("starting"); log.debug("start");
    s.o.p(" Something here ");
    log.info("ending");
    log.error("testing");
  }
}
```

Eg: import org.apache.log4j.*;

public class Hello

{

static Logger log =

Logger.getLogger(Hello.class);

public static void main(String args[])

{

PropertyConfigurator.configure("log4j.properties");

log.info("Starting...");

log.debug("Starting...");

log.info("Something here");

log.info("Ending...");

log.debug(error("Exiting..."));

}

}

sequence

① category

② appender

③ layouts

↳ Conversions.pattern.

file:

direct log messages to stdout

log4j.rootLogger = DEBUG, stdout → root logger name

↳ specify the level & name for root logger

log4j.appender.stdout.layout = org.apache.log4j.

↓ specifying the appender

PatternLayout

FileAppender log4j.appender.stdout.File

specify the file name of file appender

log4j.appender.stdout.layout = org.apache.

specify the layout log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern

%-d{ABSOLUTE} %-5p %-c{1}:%L - %m%n

13|07|06

log4j has mainly 3 components

1. category
2. Appenders
3. Layouts

Category: we can log the msgs to a log based on following 5 priority level.

1. Debug
2. WARN
3. INFO
4. ERROR
5. FATAL

Use debug to log the debugging msgs these debuging msg should not displayed in production time.

Use WARN to log the warning msgs, after logging warning msgs application will continue without any problems warning are nothing our suggestion (or) recommendation.

Use Info, to log the msgs to understand the flow of the application & things (events happening on the application.

Use ERROR to log the error messages, after logging the error messages program will continue with the alternative way provided.

Use FATAL to log the critical messages! after logging the fatal messages appln will be abnormally terminated.

(2) Appenders:

Appender is a destination for the log messages. some appenders available are

1. Console appender
2. File appender
3. SMTP appender
4. JMS appender
5. Database appender.

(3) Layouts:

Layout is a format of the logging messages. some available layouts are

logs are

① simple layout ② pattern layout ③ Html layout
④ xml layout

simple layout uses a simple format like priority level followed by message & newline

Eg:- %.p - %.m%.n

when we use simple layout no need to specify the conversion pattern property for the layout.

Pattern layout: using this we can log the message in different formats whenever we use this we have to specify the conversion pattern.

Conversion pattern property value contains two ~~this~~ any one is literal & sec is conversion character

Conversion Character

%.m - Ourputs the original message.

%.n - ourputs newline character

%.p - ourputs the priority level

%.d - ourputs the date.

we can specify the dates with some formats also

%.d{ dd-MM-YYYY HH:MM:SS,SSS }

%.d{ ABSOLUTE }

%.d{ RELATIVE }

%.C - Ourputs the complete category of msg.

Eg: %.c — com.javatree.packl.

%.1{} → packl

%.2{} → javatree.packl

%.c — ourputs the fully qualified name of the class

%.L — ourputs the line number from where the log message was issued.

%.M - o/p's the mrdname from where the msg was issued

%.F - o/p the filename " "

%.r -

Login.html 14/08/09

```
<html>
<head> <title> Student login </title> </head>
<body bgcolor="silver">
<font color="red" face="monotypecorsiva">
<h1 align="centre"> SDSoft </h1> </font>
<h2 align="centre"> student login </h2>
<br><br>
<form action="login.do">
<table align="centre" border=1>
<tr>
<td> User Name </td>
<td> input type="text" name="uname" </td>
</tr> <tr>
<td> password </td>
<td> input type="password" name="pass"
</td>
</tr>
<tr>
<td colspan="2" align="centre"> input
type="submit" value="login" </td>
</tr>
```

</body> New User

Password Assistance</body>

</html>

loginServlet.java

```
package com.javastep.student;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.log4j.*;
public class LoginServlet extends HttpServlet
{
static
Logger log = Logger.getLogger(LoginServlet.class);
public void init(ServletConfig sc)
{
String path = sc.getServletContext().getRealPath("/");
String y = sc.getInitParameter("logfile");
String x = path + y;
PropertyConfigurator.configure(x);
log.info("log4j is configured in init");
}
public void service(HttpServletRequest req,
HttpServletResponse res) throws
IOException, ServletException
{
log.info("service invoked");
String msg = "problem in login";
String pw = null, un = null;
// collect the client data
un = req.getParameter("uname");
pw = req.getParameter("pass");
log.info("client submitted data");
log.info("username" + un);
log.info("password" + pw);
```

```
StudentModel sm=new StudentModel();
log.info("calling verify student");
int x = sm.verifyStudent(un,pw);
log.info("verifyStudent returned "+x);
if(x==1)
{
  msg = "Login Completed Successfully";
}
log.info(msg);
}
req.setAttribute("info",msg);
log.info("forwarding the result to jsp");
RequestDispatcher rd = req.getRequestDispatcher
      ("result.jsp");
rd.forward(req,res);
log.info("end of the service...");
}
}
```

StudentModel

```
package com.javasree.student;
import java.sql.*;
import org.apache.log4j.*;
public class StudentModel
{ Logger log = Logger.getLogger(StudentModel
                                .class
public int registerStudent(String fn,String ln,
        String em,String un,String pw)
{ int x=0;
  try{
  Class.forName("com.mysql.jdbc.Driver");
  Connection con = DriverManager.getConnection
  ("jdbc:mysql://localhost/b7db","root",
        "javasree");
  PreparedStatement ps = con.prepareStatement
        ("insert into student values(
        "?,?,?,?");
```

```
ps.setInt(1, id);
ps.setString(2, fn);
ps.setString(3, ln);
ps.setString(4, em);
ps.setString(5, ph);
ps.setString(6, un);
ps.setString(7, pw);
x = ps.executeUpdate();
ps.close();
con.close();
}catch(Exception e)
{
System.out.println(e);
}
return x;
}
public int verifyStudent(String un,String
pwd)
{
int x=0;
try{
Class.forName("com.mysql.jdbc.Driver");
log.info("driver loaded");
Connection con=DriverManager.ser
Connection("jdbc:mysql://localhost/
bzdb","root","javasra");
log.info("connection established");
preparedStatement ps=con.prepareStatemnr
("select * from student where
uname=? and pword=?");
log.info("ps created");
ps.setString(1,un);
ps.setString(2,pwd);
ResultSet rs=ps.executeQuery();
if(rs.next())
{
x=1
```

```
log.info("Student found" +x
}
else {
log.info("Student not found" +x);
rs.close();
ps.close();
con.close();
} catch(Exception e) {
e.p(); log.error("
}
return x;
}
} // end class.
```

```
### direct log messages to file ###
log4j.rootLogger = DEBUG, sri
log4j.appender.sri=org.apache.log4j.
FileAppender
log4j.appender.sri.file=d:\Student.log
log4j.appender.sri.layout=org.apache.
log4j.PatternLayout
log4j.appender.sri.layout.ConversionPattern
=%-d-1.5p%.c[3:%L-%m%n
<target name="compile">
<javac srcdir="${srcdir}"
<copy todir="${classesdir}"
<fileset dir="${javadir}"
<include name="**/*.property"/>
<exclude name="**/*.java"/>
</fileset>
```

<classpath>

<pathelement

path = "${libdir}/mysql.jar;${libdir}/

server-api.jar;${libdir}/log4j.jar" />

</classpath>

</javac>

</target>

loginServlet </servlet-class>

web.xml

<web-app>

<servlet>

<servlet-name> registra </servlet-name>

<servlet-class> com.javastra.student.

RegistraServlet </servlet-class>

</servlet>

<servlet>

<servlet-name> login </servlet-name>

<servlet-class> com.javastra.student.

LoginServlet </servlet-class>

<init-param>

<param-name> logfile </param-name>

<param-value> /WEB-INF/classes/student.

log </param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name> registra </servlet-name>

<url-pattern> /registra.do </url-pattern>

</servlet-mapping>

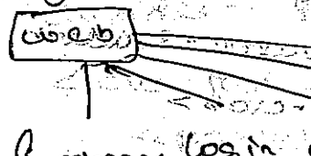
<servlet-mapping>

<servlet-name> login </servlet-name>

<url-pattern> /login.do </url-pattern>

</servlet-mapping>

</web-app>



files required

1) login.html
2) loginServlet.java
3) StudentModel.java
4) result.jsp
5) web.xml
6) log4j.property
7) build.xml
8) student.log

Session Management 15/07/06

* Generic Servlet is protocol independent.

- Http servlet extends Generic Servlet.

Q) Why GenericServlet is an abstract class?
GenericServlet is an abstract class

Q) Why HttpServlet is an abstract class?

Q) Only GenericService method is public, & other Http service doXxx() methods are protected.

Q) When I extend http servlet is it mandatory to override either Generic service (or) http service (or) doXxx methods().

Q) What will happen when I override GenericService in my servlet. (HelloServlet)?

Q) What will happen when I override Http service in my servlet?

Q) What will happen when I override doXxx() methods in my servlet?

Q) What will happen when I override service & doPost methods in my servlet with the http request send as post.
(Ans) service method will be invoked.

Q) for the scenario (8) what I have to do to invoke doPost()?
We have to call the doPost() inside the service.

```
void service(HttpServletRequest req, HttpServletResponse res)
{
    if(req.getMethod().equals("POST"))
        doPost(req, res);
    else
        doGet(req, res);
}

void doGet(—, —)
{
}

void doPost(—, —)
{
    // ...
}

void anyOtherProcessor(—, —, —)
{
    // ...
}
```

```
void doGet(—, —)
{
    anyOtherProcessor(—, —, —);
}

void doPost(—, —)
{
    anyOtherProcessor(—, —, —);
}
```

Q) as a servlet developer, suggest the solution for the following scenario?

(A) U don't know the request and type whether it is get or post. U have to handle any one in ur servlet. tell me the sample code in ur servlet?

sol: one sol. is above code.

```
(B) void serProcessor( )
    {
    }
    void doGet( )
    {
        serProcessor( );
    }
    void doPost( )
    {
        serProcessor( );
    }
```

Q) I am sending the req. calls get and but dopost only is overridden inside the servlet. wat will happen?

given an error

U have to override doget also.

Q) I am sending the req. with post and but i have overridden only service and?

U can

Q) wat is the use of init and?

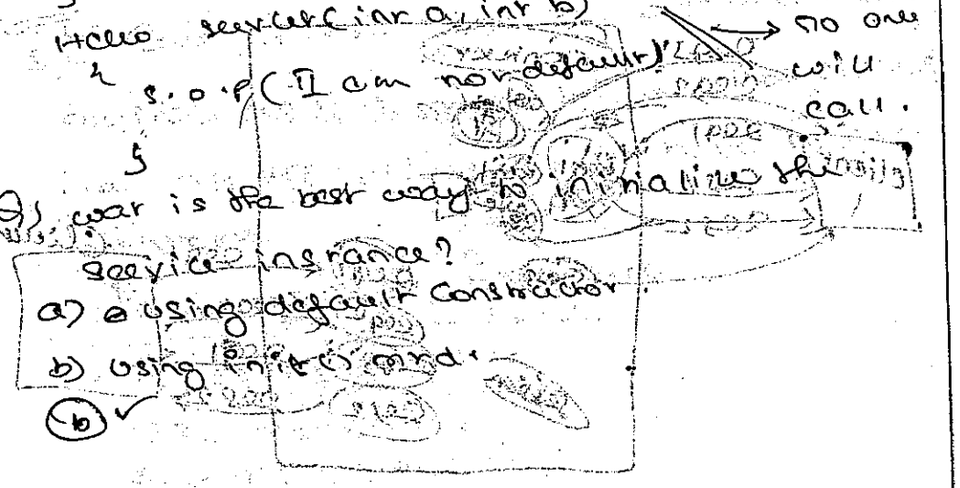
Q) can i do the initialization with the constructor?

yes we can do with default constructor. but we can't overload the constructor.

Q) can i overload service inside the servlet

class Helloservlet extends Genericservlet

```
{ Helloservlet( )
  { s.o.p("I am default"); }
  Helloservlet(int a, int b)
  { s.o.p("I am non-default"); }
}
```



Q) wat is the best way to initialize the service instance?

a) using default constructor

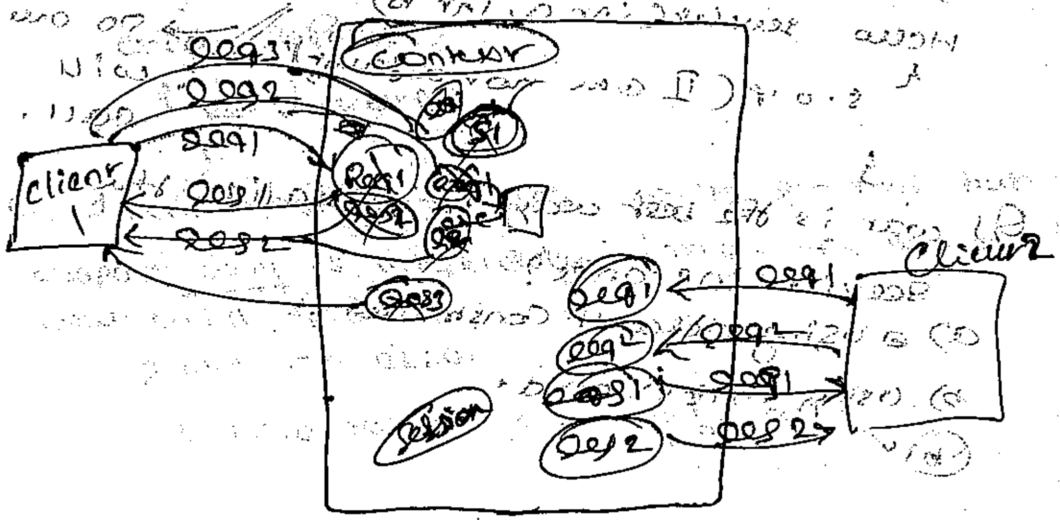
b) using init() and

(B) ✓

Q) can i create obj. for servlets
yes
Q) can i call service (or) init (or) destroy mtd
with obj which i created?

Q) can i call the destroy mtd in the service
Q) when i call, really destroy mtd will be called or not, if destroy mtd is called
the instance will be destroyed or not
Q) can i call init() mtd inside the service
when i call init() really reinitialize the instance?

Session Management



There are 3 scopes in servlets

1. request scope
2. session scope
3. context scope.

1. Request object can be accessed with in one req & response only.

2) session obj can be accessed across multiple requests by the single client.
each client will have his own session

3) Context obj can be accessed across multiple req & responses by multiple clients
i.e all the clients will have only one context.

Parameters vs Attributes

- Types of parameters
  1) Req. parameters
  2) config parameters
  3) context parameters

→ Storing the parameters will be done by JVM container, we can take by calling the getAttribute() method.

Request parameters will be stored by the container by taking client given data.

We can get by calling the getParameter() method.

Config parameters will be stored by the container by reading init params within the server tag from web.xml file.

We can get Config parameters by calling the getInitParameter() method.

Context Parameters will be stored by the container by reading the Context Params from web.xml file. We get by calling getInitParameterNames()

Attributes

Types: ① req. attribute
② session attribute
③ Context attributes

We have to store the attributes explicitly in req, session & context objects. Container won't set the attributes like parameters. The following 3 objects (req, session, context) are provided in the web container

void setAttribute(String, Object)
Object getAttribute(String)
void removeAttribute(String)
Enumeration getAttributeNames()