# \*\*\*

.

1. How do we organize the files in the project?
2. How do we deliver the project to the customer?
3. How do we resolve the problems like package not do not excite or class not found Exception?

Every project organizers the files in a better manner by create a folder.

Most of the people follow a procedure do organizer the files in the project?

Step1: Create a folder and the folder name must be project name (BMS).

Step2: Create the following folders in the above created project. src = browser files bin\classes-class files lib-library files (jar files) doc-documentation tmp-temporary

We would like to develop 3 java programs in src folders and compile and placed .class files in the bin folder. If we would like to place the .class files into a specific folders we use an option ‘-d’ for javac command.

**Ex**: **Javac –d <destination-folders> sourcefileName**

Javac –d d:\work\bms\bin welcome.java

If always recommended to create the java program with packages. It is not at all recommended to create the java programs with out packages.

To create the packages every company follows their own procedures. Some of the company users the following pattern.

Package domain.companyname.projectname.module (Or)

Package companyname.projectname.modulename

To deliver the project to the customer we can use any of the following.

1. JAR(javar a relive)
2. WAR(web a relive)
3. EAR (enterprise a relive)

We deliver the software in the form of jar if it’s corejava related projects.

All the web based applications (servlets, jsp, structs) will release in the form of WAR files.

The enterprise applications like EJB’s will release in the from of EAR files.

We have only one file or command JAR.EXE to create jar file or war file or ear files.

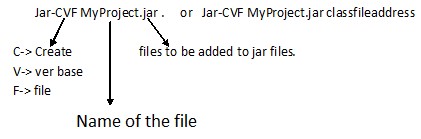
D:\> javac –d

**Procedure to create JAR file**:

Goto a folder where all .class files are available use the following command to create JAR file.

Syntax: **Jar –cvf jarfilename files-to-be-added**

**Ex**:



To extract contents of jar file we use the following command.

**Jar –XFV MyProject.jar**

When we extract the contents of jar file we have found on extra folder META-INF.

Inside this folder we found an extra file MANIFEST.MF

As part of manifest.MF we specify version information of software.

***Creating a WAR file and EAR file***

***Jar -CVF MyProject.war***

***Jar -CVF Myproject.ear***

* The main advantage of java is reusability. In java we can reuse the classes developed by somebody by creating the objects. If we would like to call the project, we must aware of all the methods of that class.
* The javadoc command will help us increasing the doc as per the documentation we can find all the field summary, constructor summary & method summary.

Ex. Javadoc Cone.java

* The java documentation will generate so many HTML files. We can navigate the documentation from index.HTML file.

Javac –d D:\work\lic\bin or D:\work\lic\src\\*.java bin> java info.inetsolv.jee.mb.welcome delivering project.jar file cd work\lic\bin>jar –cvf lib.jar

\bin>cd src

>src> java doc –d

D:\work\lic\doc welcome.java

**Environment Variables**:

These are the variables which will be created for us to create Environment variable. We use keyword set.

**Eg**: D:\>work>set ONE = 1

* To see all the environment variables of our computer we use a keyword “set”.
* To check the value of environment variable we use a command echo

**Ex**: echo% one %

When we run above command. If environment variable one is available we get value of it, if it is not available we get the value as % one %

Variable name

Set one = 1;2;3

Echo % one %

1;2;3

C:\> set one = 1

E:\> echo % one %

Set one = % one % 4,5;6;7

Echo % one % - 1; 2; 3; 4; 5; 6; 7

Set one = % one % ;3;4;5

Echo % one % 1; 2; 3; 4; 5

Set one = 0; % one %

Echo % one % 0; 1; 2; 3; 4; 5

To append new value to the existing environment variable we will use following command.

**Scope**:

Where we can access variable is scope by default the scope of environment variable is until we close session (close the window) procedure to set the environment variable is until permanently.

1. Computer Right click -> properties -> Advanced system settings.
2. Choose an option advanced -> environment variables.
3. In system variables select new button and enter variable name and value.

If we want operating system to search for some files we use an environment variable path.

**Ex**: set PATH=C:\programfiles\java\jdk1.6.0\_29\bin

Path environment variable environment variable is used to search for the files. This variable is used by operating system.

We use the **class path environment variable to search for the .class files** this environment variable to search for the .class files this environment variable is used by JVM.

\bin>jar –CVF MyProject.jar

C:\> set CLASSPATH=C:\MyProject.jar;.; C:\> java bdps.welcome package bdps javac –d d:\work\bin

D:\work\lic\src .java java bdps.welcome

set CLASSPATH=D:\work\lib\bin;.;

When we deliver jar file to customer. He set CLASSPATH to jar file & execute it.

**Ex**:

C:\> set CLASSPATH=C:\MyProject.jar;.; .cmd file

C:\> java bdps.welcome .b (or)

C:\>pause sh for unix

To release project to customers we create jar file if we give only jar file customer is responsible to set the CLASSPATH and type the commands to run the project.

To resolve this problem we develop a CMD file or data file. In this we supply all the DOS commands.

set CLASSPATH=c:\MyProject.jar;.; java info.inetsolv.jee.mb.welcome

pause //run.cmd

creating an object’s INSTANCE

\***API (Application Programming Interface)**:

API is a document. (not a software)

They are two types of API’s are available. They are

1. Public API
2. Proprietary API

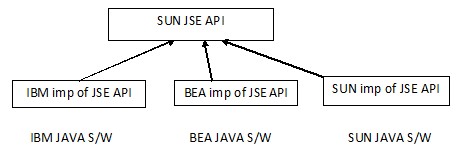
Any body can use the public API. Proprietary API can be used by only that company.

**In java, API document contains set of class and Interfaces.**

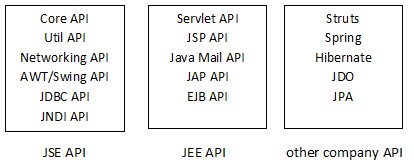
**In case of ‘C’ language, API document contains set of predefined functions**.

Once if the API is released any body can use the API.

The meaning of using API is providing the implementation.



Sun micro system as released JSE API. This is a public API released by sun micro system. Once in the public API is released any body can provide the implementation. We call the implementation has software from the above diagram we have under studied that the java a software released from different companies like IBM, BEA, SUN micro and etc. All these companies have provided the implementations to all the classes and interfaces released by sun micro system company.



JDBC

Java Database Connectivity.

In general, to maintain the data of an organisation the data can be stored in multiple places. They are

1. Files
2. Database server

In a File system there are so many disadvantages to store the data.

They are:

1. File system ***does not provide security***.
2. There is ***limit on the size of the files.***
3. If we store the data into a file, the redundant data will be stored.
4. The main disadvantage is – data sharing.
5. In a file system, the data cannot be accessed from different places.
6. To overcome these problems database system was developed.
7. A database can support – data sharing and data integration.
8. Integration means the data can be stored at only one place. So, all the users can access the data from a single location. This is called data sharing.
9. a java application wants to interact with communication with database server, JDBC API is used.
10. There are so many database servers are available in the market. Some of them all ***Oracle, MYSQL, SAP, IBMDB2, POINT database server and etc***.
11. Whenever we purchase any database server we get two different software’s.

They are:

***Database Server Program(S/W)***

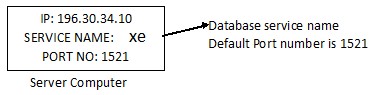
***Database Client Program(S/W)***

**Database Server:**

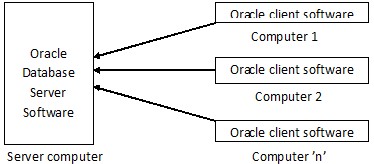
The database server s/w will be installed in server computer. It is the responsibility of database or network administrator to install server software in server computer. Every server computer is connected to a network. We can quickly identify a server computer by using the IP Address of the computer. If we want to develop any program to communicate with the server, then we need the following information’s.

They are

1. ***IP Address***
2. ***Service Name***
3. ***Port Number***



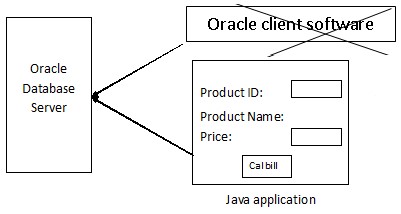
The database client software installed with all the developer computers.



***Service name***is nothing but the name assigned to database server.

***Port number*** is unique communications channel.Every programmer uses one port number.

Instead of oracle client software we are developing the java (programmer) application which interacts with database server.



We are using a java application instead of oracle client program If we want to request the server the client is supposed to type the SQL queries and sent the request to the server. But when we use a java application to send the request to the server, user no need to generate SQL queries. When the user interacts which GUI and the client on the button the java application generates the queries and send to server.

A ‘C’ program would like communicate with database server. If two programs would like communicate each other they must use same protocol. If they use different protocol they will not communicate each other.

Oracle database server is developing based on thin protocol.

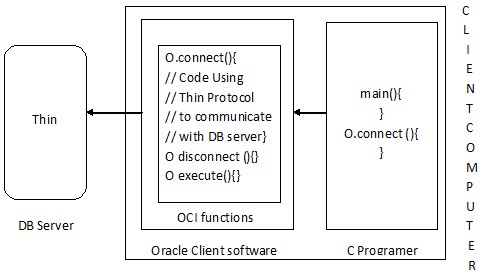
Thin protocol proprietary to Oracle database server.

If we would like to develop a ‘C’ program to communicate with database server. The ‘C’ Program has to send the request by using thin protocol. But Oracle guys are not ready to release thin protocol to outside market.

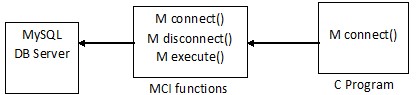


To resolve the above problem Oracle guys are developed OCI functions (Oracle Call Interface) these functions are developed a ‘C’ language. The code inside the OCI functions. Actually interact with database server. Oracle guys are release OCI functions as part if client software.

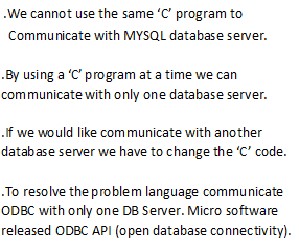
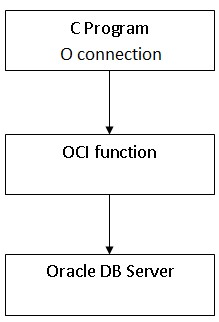
Now we can develop a ‘C’ program to communicate with Oracle database server. By using OCI functions.



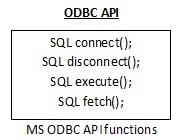
Similarly, MYSQL DB Server releases MCI functions to interact with MYSQL DB Server.



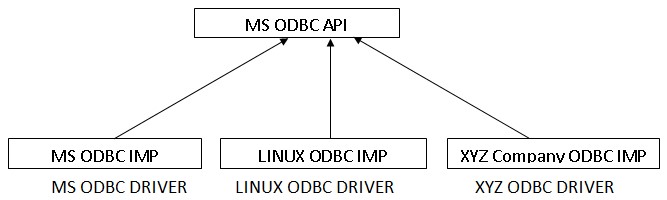
We got a project to develop in ‘C’ language to communicate with Oracle database server. The following architecture is same.



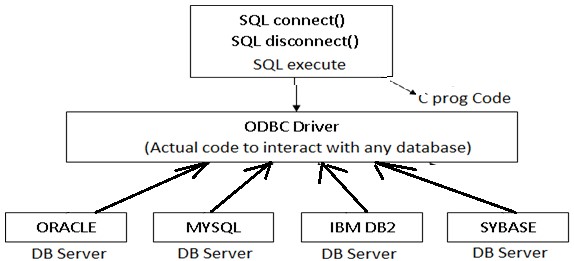
***ODBC API (Open Database Connectivity)*** - contains set of ‘C’ language functions.



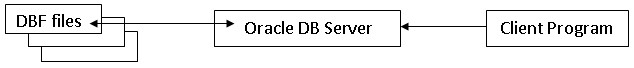
Once in the API’s released anybody can provide the implementation. The implementation is called as ODBC software or ODBC Driver.



***Architecture of ‘C’ program which uses ODBC API.***



Sun Micro System released JDBC API to develop a java program to communicate with any database server without changing the java code. The database server stores the data into files. These files are called as DBF files.



When the client sends a request to the database server, Server takes the request and encrypts the data and store the data into DBF files. If data required, then server program reads the data from DBF files and decrypt the data and send to the clients.

***Oracle Database Server released in the form of two versions. They are:***

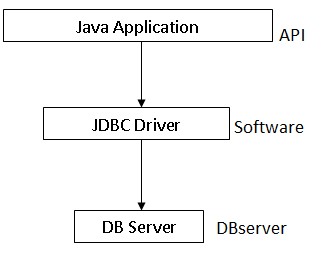
1. ***Enterprise edition (XE)***
2. ***Express edition***

For an enterprise edition the service name and port number of database server are fixed. They are

***Service Name: XE***

***Port No: 1521***

**JDBC Applications Architecture**



From the above architecture, we have to develop the java application which communicate with DB server.

Sun Micro System released two packages for JDBC. They are:

1. ***Java.sql***
2. ***Javax.sql***

***java.sql package***

A package is a collection of classes and interfaces.

**java.sql interfaces**:

1. Driver
2. Connection
3. Statement
4. PreparedStatement
5. CallableStatement
6. ResultSet
7. DatabaseMetadata
8. ResultSetMetadata

**java.sql classes**:

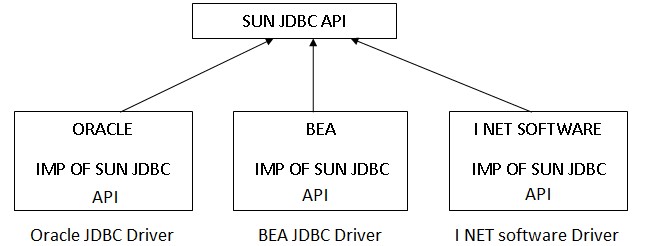
1. DriverManager
2. Types
3. Date

**javax.sql package**

**javax.sql interfaces:**

1. DataSource
2. RowSet

Once the API released, so many people has provide the implementation to JDBC API, we call the implementation as JDBC Driver. We can use any company’s JDBC Driver to develop the application.



JDBC Driver software has been provided in the form of jar files. Jar files are used to develop the java application to communicate with database server.

# JDBC Drivers

|  |
| --- |
| JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:   1. JDBC-ODBC bridge driver 2. Native-API driver (partially java driver) 3. Network Protocol driver (fully java driver) 4. Thin driver (fully java driver) |
|  |

1. **JDBC-ODBC bridge driver**

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.



### **2) Native-API driver**

|  |
| --- |
| The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java. |



### **3) Network Protocol driver**

The Network Protocol driver uses middleware (application server) that converts JDBC calls

directly or indirectly into the vendor-specific database protocol. It is fully written in java.



### **4) Thin driver**

|  |
| --- |
| The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language. |



**procedure to develop a java application**

1. Register the JDBC Driver.
2. Get the Connection from the Database Server.
3. Create the Statement Object.
4. Execute the Query’s.
5. Close the Connection.

Register the Driver:

* It is nothing but a java program, which interact with the database server.
* To register driver, you must create the object for driver interface, ie we can create the object for implementation class or driver interface.
* Here, every database server gives its different implementation class for driver’s interface.

**A class which provides an implementation of JDBC Driver interface is called as Driver class.**

The name of the class varying from Driver to Driver we can find Driver class name in manual.

Import java.sql.\*;

Public class RegisterDriver

{

Public static

ublic static void main(String[] args)throws SQLException{

Driver d = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);

staticmethod driver object or driver class object

System.out.println(“Driver is Register”);

} }

\*The following JDBC program established connection with Oracle DB server.

import java.sql.\*;

public class DatabaseConnection{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

(“**jdbc:oracle:thin:@localhost:1521:xe**”,”system”,”malli”); System.out.println(“got the connection:” + con.getClass());

}

}

If we got the connection object successfully we can say the java program establish with connectivity database server. If we fail to get the connection it throws an Exception java.sql.Exception.

\*Requirement:

Develop a java program to create a table in the database server. The table name must be emp with eno, ename, salary as columns names.

import java.sql.\*; public class CreateTable{

public static void main(String[] args)throws SQLException{

//step1: Register the JDBC Driver

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

//step2: get the connection from the DB Server

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”system”,”malli”);

//step3: create the statement object

Statement stmt = con.createStatement();

//step4: execute the query stmt.executeUpdate(“ create table emp(eno number(3), ename varchar2(15),

salary number(5)”);

//step5: close the connection

Stmt.close();

} }

***Queries in JDBC:***

in JDBC queries classified into two types they are:

1. Select query’s
2. Non-select query

**select queries**

The query’s which is start with select keyword is called as select query.

Ex. **select statement (command)**

**non-select queries**

The query’s which does not start with select keyword are called as non-select query’s.

Ex. **insert, update, delete, create, drop etc.**

To execute the **non-select query’s execute update method is used**

**Syntax**:  **int executeUpdate(query);**

To execute the **select query execute query method is used.**

**Syntax**: **ResultSet executeQuery(query);**

In a project we create the tables before the project starts.On the created tables are performing CURD operations.

***C-Create***

***U-Update***

***R-Retrieve***

***D-Delete***

import java.sql.\*; public class InsertTable{

public static void main(String args[])throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli"); Statement stmt = con.createStatement(); stmt.executeUpdate("insert into emp values(106,'Ravi',18000)");

} }

**Execute update method returns an integer value**. This integer value indicates the number of records affected by the query in the database server. The memory of effected is because of SQL query how many number of records are update, delete and inserted the data.

**Ex**: int no = stmt.executeUpdate(“delete from emp where eno = 101”); System.out.println(no);

**set CLASSPATH=ojdbc14.jar;.;**

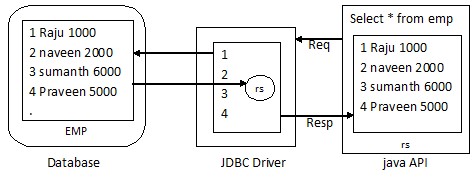
**Requirement**:

To retrieve all the records from a table, we can use the select query. The retrieved data to be displayed to the client.

The select query will be executed by using execute query method. Itreturns the data to a ResutlSet object.

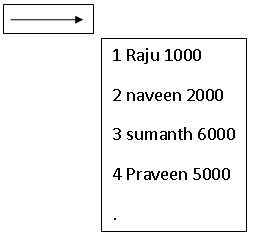
**Ex**: ResultSet rs = stmt.executeQuery(“select \* from emp”);

When we send the query to JDBC driver and it sends the query to data base server. DBS (database server) is executed the query and return the records to JDBC driver. It is the responsibility of JDBC driver to convert data into ResultSet object (rs) and this ResultSet object is return to java application.



When even we got the result set object(rs). It is associated with a ResultSet pointer. **Initially these ResultSet pointer points to invisible record before first row.**

ResultSet pointer



**Methods:**

* 1. **getRow:** This method is used to find where the ResultSet pointer is ponting to.

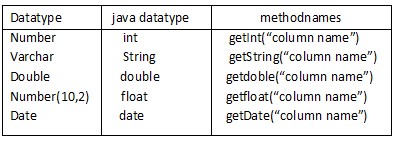
**Syn**: **int getRow()**

* 1. next() It is used to move the ResultSet pointer to the next row.

**Syn**: **boolean next()**

When we call the next method if the next record is available it returns the true value. If the next record is not available it returns the false.

To read the values from specific column we use get row methods. The following diagram shows how to read the values of a specific data type.



import java.sql.\*;

public class RetriveRecords{ public static void main(String[] args)throws SQLException{ DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from emp"); while(rs.next()){

System.out.println(rs.getInt("eno"));

System.out.println(rs.getString("ename"));

System.out.println(rs.getInt("sal"));

}

}

}

Note:

* In the JDBC API whenever an Exception was occurred, then it will throw to **java.sql.SQLException.**
* As part of the JDBC program we are trying to retrieve column which is not available in a ResultSet object. In this scenario java application throws SQLException with **invalid column name as the message.**
* If the ResultSet pointer is pointing to an invisible record after the last row and if we are trying to get the values, then we will get **exausted ResultSet Exception**.
* When we got the ResultSet object and without calling next() method we get an Exception **ResultSet next was not call**.
* Most of the projects we try to select the records based on some criteria.
* To fitter the records based on the criteria, select query will be used with where clause.

**Ex**: ***Retrieve all the records from emp table whose salary is > 3000***

String s=“select \* from emp where salary > 3000”;

ResultSet rs = stmt.executeQuery(s);

***Projection***

**Display selected columns is called as projection.**

Develop a java application to retrieve emp and salary from the emp table.

**Ex**: ResultSet rs = stmt.executeQuery(“select eno,salary from emp”);

***Methods of ResultSet interface***

**Ex**: **getxxx(columnName);**

**getxxx(columnIndex**);

We can get the values based on column index also. The column Index starts with (1) one.

**Ex**: ***System.out.println(rs.getString(1)); System.out.println(rs.getInt(2));***

It always recommended to use getxxx with for getxxx (columnName);

* Whenever we use **DriverManager.getConnection we get a physical connection between java.API and database server**. It always recommended to close the connection other programs cannot use the connections of DBS.
* To close the connection we use a method ***con.close();***when we close the connection all the resources will be immediately released.

\***Assignment**:

1. Retrieve the records from the ResultSet object by using do…while loop and display to the client.
2. Use a for loop to retrieve the records from ResultSet object display the client.

// using with do…while loop import

java.sql.\*; public class DoWhile{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con=

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from emp"); if(rs.next()){ do{

System.out.println(rs.getInt("eno"));

System.out.println(rs.getString("ename"));

System.out.println(rs.getFloat("sal"));

}while(rs.next());

}

else{

System.out.println("There is no Records");

}

} }

import java.sql.\*; public class IfDoWhile{ //using with if condition and do…while public static void main(String[] agrs)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con=

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from emp"); if(rs.next()){ for(; rs.next() ;){

System.out.println(rs.getInt("eno"));

System.out.println(rs.getString("ename"));

System.out.println(rs.getFloat("sal"));

}

}

else{

System.out.println("There are no Records");

}

} }

1. Try to create the table by using executeUpdate and check the return type of exeucuteUpdate. int no = executeUpdate(“Query”);
2. try to execute selectQuery by using executeUpdate and check the return value. ResultSet rs = executeQuery(“Query”);

statements:

1. prepared statement
2. callable statement
3. **PreparedStatement**:

* PreparedStatement is used to send the queries to DBS (Database Server).
* **PreparedStatement improves the performance of java application when compared with statement object**.
* By using PreparedStatement also we can perform CURD operations.

C-create a record

U-update a record

R-retrieve records

D-delete a record

To work:

* 1. Register the JDBC Driver
  2. get the connection from database server.
  3. Create a PrepareStatement object by supplying query as input. These query must contain positional parameters.

**Ex: insert into emp values(?,?,?);**

* 1. Supply the values to positional parameters by using **setxxx()**methods.
  2. Execute the queries by calling executeUpadate() or executeQuery()
  3. Close the connection

\***Requirement**:

Develop a java application to insert a record into emp table by using prepared statement?

import java.sql.\*;

public class preparedst

{

public static void main(String[] args)throws SQLException

{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

String cs="jdbc:oracle:thin:@localhost:1521:xe";

Connection con=DriverManager.getConnection(cs,"system","manager");

String q="insert into employ values(?,?,?,?,?)";

PreparedStatement pst = con.prepareStatement(q);

pst.setInt(1,102);

pst.setString(2,"Raju");

pst.setString(3,"clerk");

pst.setDouble(4,5000);

pst.setInt(5,10);

int no = pst.executeUpdate();

System.out.println(no);

con.close();

}

}

import java.sql.\*;

public class PreparedUpdate

{

public static void main(String[] args)throws SQLException

{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

String cs="jdbc:oracle:thin:@localhost:1521:xe";

Connection con= DriverManager.getConnection(cs,"system","manager");

String q="update employ set sal = ? where ename = ?";

PreparedStatement pstmt = con.prepareStatement(q);

pstmt.setDouble(1,5500);

pstmt.setString(2,"Akash");

int no = pstmt.executeUpdate();

System.out.println(no);

con.close();

}

}

By using PreparedStatement object we can execute any type of Queries that is Insert, Update, Delete and Retrieve records.

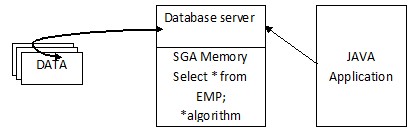
**It is the responsibility of DataBase Server to improve the performance of the same query is sent for the second time.**

When we sent the query second time with a small changes in it , database server treats the query as different query and process all the steps.

Even if then is change we are use of the query database treat as different query.

The following steps will be carried out by the database server when the query is send to it.

1. The database server checks the query. If it is invalid it thrown an exception.
2. Database server checks where there all the records are available or not if not available it will display the error message.
3. Database the query into its understandable language.
4. As part of database server prepares couple of algorithms to get the data from database files.
5. Database stores algorithm with queries in the SGA (System Global Area) memory of database server.
6. Based on the batabase check get the best database server algorithm.
7. It execute the algorithm and get the data from database files and sent to client.



When to choose statement and prepared statement. When even a sending queries UN difficult it not modify then recommended using statement.

**If the same query with the different values are going to DBS. Then recommended to use prepared statement**.

DBS has given the solutions if you sending same query with different values to database server. These problems we use bind variables.

|  |
| --- |
| variable Veno number; print Veno; |

SQL>

**Ex**: select \* from emp where eno :veno := 5; variable veno number; exec := veno := 2;

insert into emp values(:veno, :vename, :vsalary);

\*How to use bind variables in Oracle?

1. We must **create a variable** before we use it. To create a variable we use the command.

**Syntax: variable variablename datatype**

**Ex**: **variable vsalary number;**

1. To **assign the value to variable**

**Syntax**: **exec :variablename := value**

**Ex**: **exec :veno := 2;**

1. To display/print the value of a variable

**Syntax**: **print variablename**

**Ex**: **print veno**

1. The following is an example of bind variable query to **insert the records. Ex: Insert into emp values(:veno, :vename, :vsalary);**
2. The following is an example of **select query from bind variable. Ex: Select \* from emp where eno =:veno**
3. When even the JDBC driver is ‘n’ commands prepared statement. The JDBC driver commands prepared statement into bind variables.
4. When even we call the **setter methods JDBC driver supply the values to bind variables.**
5. **Prepared statement will not improve the performance when the query goes to database server for the first fine. This is because first time database server has to execute all the steps.**
6. In a project we should never use Scott and tiger we always use our user name to login database server by using system user.
7. We have to use the following command to create a new user in database server.
   1. Create user laxmi identified by kusuma
   2. Grant connect, resource to laxmi;

We must follow the following to best practices when we write the JDBC code.

1. Always recommended to practice the query in SQL\*plus and use it a java application.
2. It is always recommended to display what is the query which sends to java application. We can achieve this by the writing the query in a variable and display the query.

\*Develop a JDBC program to get data from keyboard (Buffered Reader) store the data into employee table. Use prepared statement?

import java.io.\*;

import java.sql.\*;

public class PreparedBufferedReader{

public static void main(String arga[])throws IOException,SQLException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter employee number:");

String eno = br.readLine();

System.out.println("Enter employee name:");

String name = br.readLine();

System.out.println("Enter employee salary:");

String salary = br.readLine();

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con=

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

String query = "insert into emp values(?,?,?)";

System.out.println(query);

PreparedStatement pstmt = con.prepareStatement(query); pstmt.setString(1,eno); pstmt.setString(2,name); pstmt.setString(3,salary); pstmt.executeUpdate();

} }

import java.util.\*; import java.sql.\*; public class Statementsanner{ public static void main(String[] args){ Scanner s = new Scanner(System.in); System.out.println("eno"); int eno = s.nextInt(); System.out.println("name");

String name = s.next(); System.out.println("salary"); double salary = s.nextDouble();

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con=

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

Statement stmt = con.createStatement();

String query = "insert into emp values("+eno+",' "+name+" ',"+salary+")"; System.out.println(query); stmt.executeUpdate(query);

}

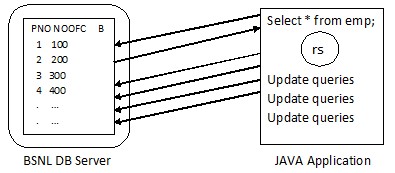
}

By using JDBC we cannot execute the queries which are specific to a particular DBS.

**Callable statement**:

**Callable statements are used to call the procedures from java application.**

Calculate the bill amount to all the customers which are available in customer table.



import java.sql.\*;

public class StudentTotal

{

public static void main(String[] args)throws SQLException

{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

String cs="jdbc:oracle:thin:@localhost:1521:xe";

Connection con = DriverManager.getConnection(cs,"bdps","oracle");

String query = "Select \* from student";

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(query);

while(rs.next())

{

int a = rs.getInt("clang");

int b = rs.getInt("java");

int c = rs.getInt("python");

int tot = a+b+c;

double av=tot/3.0;

int num = rs.getInt("sno");

String UQuery = "update student set total = ?,avgm= ? where sno = ?";

//System.out.println(UQuery);

PreparedStatement pstmt = con.prepareStatement(UQuery);

pstmt.setInt(1,tot);

pstmt.setDouble(2,av);

pstmt.setInt(3,num);

pstmt.executeUpdate();

}

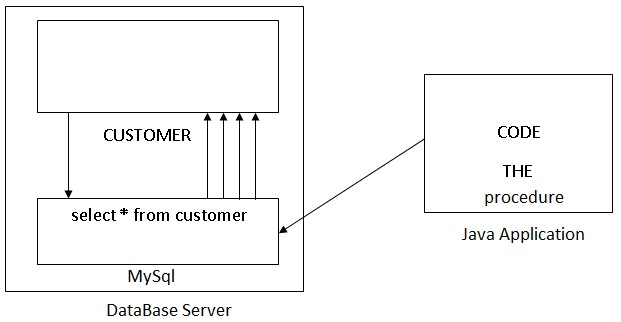
}

}

The disadvantage of the above approach is we are interacting with database server for multiple times throw networks. Because of this we get the performance issue we can resolve this by using processors.

We can write the processor with business logic. Processor in database server run’s the procedure inside database server.

Whenever anybody call the procedure then the database server run’s it to improve the performance of project.



We can use three types of procedures.

1. A procedure which doesn’t take any type of parameters ie without parameters.
2. A procedure which takes input parameter.
3. A procedure input parameter as well as output parameter.

\*How to create procedure in database?

Create or replace procedure Myproc as

Begin

Insert into emp values(101,’kumar’,3000);

End Myproc;

/

\*To execute the Procedure from the client we use a **command**

**Exec MyProc;**

The following procedure which takes input parameter.

1. create or replace procedure MyProc(veno IN number,
2. vname IN varchar2, vsalary in number)
3. as
4. begin
5. insert into emp values(veno, vname, vsalary);
6. end MyProc;
7. /

\*To run the above procedure we use the following command

Eg. **exec MyProc(1,’abc’,2345);**

\*A procedure which take input and output parameters.

create or replace procedure addition(no1 in number, no2 in number,result out number) as

begin

result := no1 + no2;

end addition

/

exec addition;

exec addition(10,20, :result);

print result;

\*To call the above procedure we have to perform the following steps.

Step 1: Before we call the procedure we must register / create variable. (/ slash)

Step 2: Call the procedure by supplying bind variable as input.

Step 3: After procedure is executed we can get the value from out variable and display.

\*Develop a java application to call a procedure which doesn’t take any parameter.

public class CallProcedure{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”kesav”,”kesav cherry”); CallableStatement cstmt = con.prepareCall(“{call MyProc}”); cstmt.Close();

}

}

\*Develop a java application to call the procedure which takes Input parameters. SQL>

create or replace procedure MyProc(Veno in number, Vname in varchar2, Vsalary in number) as begin insert into emp values(Veno, Vname, Vsalary); end MyProc;

/

import java.util.\*; import java.sql.\*; public class CallProcedure{

public static void main(String[] args)throws SQLException{

Scanner s = new Scanner(System.in);

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”kesav”,”kesav cherry”); CallableStatement cstmt = con.prepareCall(“{call MyProc(?,?,?)}”); cstmt.setInt(1,s.nextInt()); cstmt.setString(2,s.next()); cstmt.setInt(3,s.nextInt()); cstmt.execute();

} }

\*Develop a java application to call a procedure which takes input parameter and output parameter?

create or replace procedure addition(Vno1 in number, Vno2 in number, Vresult out number) as begin result := Vno1 + Vno2; end addition;

/

import java.sql.\*; public class CallableStatement{ public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle:jdbc:driver:OracleDriver());

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”kesav”,”kesav cherry”); CallableStatement cstmt = con.prepareCall(“{call addition(?,?,?)}”); cstmt.setInt(1,11); cstmt.setInt(2,22); cstmt.**registerOutParameter(3,Types.INTEGER);** cstmt.execute(); int result = cstmt.getInt(3);

System.out.println(result);

} }

**When we register out parameter it is responsibility of JDBC driver to declare a variable**. One in the procedure is executed the value will store in the variable.In oracle we can use functions. They are two types of function are available in oracle.

1. Predefined functions (or) aggregate functions
2. User defined functions

In oracle we have the following aggregate functions they are

1. count()
2. min()
3. max() 4.sum()

5.avg() and etc.

To call function in oracle we use Queries for example

Select sum(sal) from emp;

Select max(sal) from emp;

Select count(\*) from emp;

\*Develop a java application to find the number of records available a emp table?

import java.sql.\*; public class FindRecords{

public static void main(String[] main)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OralceDriver());

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost”,”kesav”,”kesav cherry”); Statement stmt = con.createStatement();

ResutlSet rs = stmt.executeQuery(“select count(\*) count from emp”); if(rs.next()){

System.out.println(rs.getInt(“count”));

} }

Select \* from user\_source;

Show errors;

create or replace function addition(Vno1 in number, Vno2 in number) return number

as result number; begin result := Vno1 + Vno2; return result; end addition;

/

**Approach1**: To call the above functions use a query to get the values. **SQL**> select addition(10,20) from dual;

**Approach2**: We can call a function by using callable statement;

**SQL**> select addition(10,20) sum from dual; **Ex**: import java.sql.\*; public class FindRecords{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”kesav”,”kesav cherry”); CallableStatement cstmt = con.prepareCall(“{? = call addition(?,?)}”); cstmt.registerOutParameter(1,Types.INTEGER); cstmt.setInt(2,23); cstmt.setInt(3,33); cstmt.execute(); int sum = cstmt.getInt(1);

System.out.println(sum);

}

}

**Types of ResultSets**:

There are two types of ResultSets in java.

They are

1. **Forwardonly ResultSet**
2. **Bi-Directional ResultSet**
3. **forwardonly ResultSet**

A ResultSet which can move only forward direction is called as Forwardonly ResultSet. By default in java we get Forwardonly ResultSet.

1. **Bi-Directional ResultSet?**

A ResultSet which can move Forward and Backward is called as Bi-Directional ResultSet.

Bi-Directional ResultSet can be achieved in both statement and prepared statement object. To get the Bi-Directional ResultSet we have to supply parameters to statement object (or) prepared statement object.

To create Bi-Directional ResultSet for statement object.

**Syntax**: create statement(ResultSetTYPE,ResultSet CONCURRENCY);

Where **ResultSet TYPE** we can supply any of the following three values.

1. TYPE\_FORWARD\_ONLY By Default
2. TYPE\_SCROLL\_SENSITIVE
3. TYPE\_SCROLL\_INSENSITIVE

For **ResultSet ConCurrency** we can supply any of the following two values.

1. CONCUR\_READ\_ONLY By Default
2. CONCUR\_UPDATABLE

**Sensitive and Insensitive ResultSet**

When we develop a java application to retrieve records and get the result to java application and if somebody modify the data in database server, if it got reflected in java application we call it as Sensitive ResultSet.

After modify the data in database server if it is not reflecting in ResultSet object we call it as Insensitive.

When we use CONCUR\_UPDATABLE when even we modify the data in ResultSet object it get reflected in database server.

**Ex**:

import java.sql.\*; public class ResultSetTypeStatement{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement

(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_READ\_ONLY); ResultSet rs = stmt.executeQuery("select \* from emp"); rs.next(); rs.next();

System.out.println(rs.getRow()); rs.previous();

System.out.println(rs.getRow()); rs.previous();

System.out.println(rs.getRow()); con.close();

} }

When we call the previous() method the ResultSet pointer is placed to old recored. **Absoulte()**: This method is used to move the ResultSet pointer to a specific Record/Row.

**Ex**: rs.absolute(5);

When the above line is executed to ResultSet pointer is placed at 5th Row.

We can use Bi-Directional ResultSet in case of prepared statement object also.

**Ex**: PreparedStatement pstmt = con.prepareStatement(“select \* from emp”

,ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_READ\_ONLY);

**Ex**: To develop a sensitive ResultSet program we must follow the follow three steps.

1. Make the ResultSet object as sensitive
2. Use the refreshRow() method
3. In the Query instead of \* we must use column names.

import java.sql.\*; public class ResultSetTypeReadOnly{

public static void main(String[] args)throws Exception{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statementstmt = con.createStatement

(ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_READ\_ONLY); ResultSet rs = stmt.executeQuery("select eno, ename, sal from emp"); while(rs.next()){

//System.out.println("press any key to get the next result");

//System.in.read(); //System.in.read(); rs.refreshRow();

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

System.out.println(rs.getString(3));

}

} }

When ever java program has en counter a refreshRow() it will check the data in database server and the data in ResultSet object same or not. If they are not same refreshRow() will update the data in ResultSet object. **refreshRow() method work only from sensitive ResultSet**.

In the SQL query instead of \* we are specified column names. This in because when ever then the change in specific column the JDBC Driver update the specific column only.

Develop a java application to retrieve the records un updated a specific records in the ResultSet object by using CONCURRENT updatable?

**Ex**:

import java.sql.\*; public class Absolute{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement

(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE); ResultSet rs = stmt.executeQuery("select eno, ename, sal from emp"); rs.absolute(3); rs.updateDouble("sal",22000); rs.updateRow();

} }

import java.sql.\*; public class MoveToInsertRow{ public static void main(String[] args)throws SQLException{ DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement

(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE); ResultSet rs = stmt.executeQuery("select eno,ename,sal from emp"); rs.moveToInsertRow(); rs.updateInt(1,123); rs.updateString(2,"veera"); rs.updateDouble(3,19000); rs.insertRow();

}

}

\*Develop a java application to retrieve the records from emp table and delete the 4th record?

To achieve the above requirement we can a method delete Row.

import java.sql.\*; public class AbsoulteDeleteRow{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement

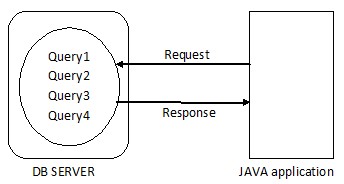
(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE); ResultSet rs = stmt.executeQuery("select eno,ename,sal from emp"); rs.absolute(9); rs.deleteRow();

} }

**Batch Updates: (improve the performance)**

When we use Batch updates in the projects it will improve the performance. We can add multiple queries to Batch object and send Batch object to database server only once. Because of these we can improve the performance.

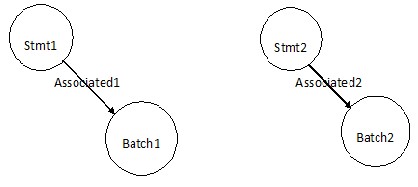
If we use Batch updates we can write Business logic in java applications.



**When ever we create the statement object immediately a Batch object will be created. This Batch object is associated with statement object.**

**Ex**: Statement stmt1 = con.createStatement();

Statement stmt2 = con.createStatement();



**Requirement**:

\*Develop a java application to insert three records by using Batch updates? import java.sql.\*; public class InsertBatchRecords{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry"); Statement stmt = con.createStatement(); stmt.addBatch("insert into emp values(122,'niranjan',16666)"); stmt.addBatch("insert into emp values(124,'sravan',11143)"); stmt.addBatch("insert into emp values(125,'mrudhu',14371)"); stmt.executeBatch();

} }

By using object we can add insert, update and delete queries. By using Batch updates we can not perform select operations.

**Syntax of execute**: int a[] executeBatch(); **Ex**: int a[] = stmt.executeBatch();

The size of integer array is dependent on the size of the Batch object. The integer array contains the values which are got effected by each and every query. We can print these values by using a loop.

**Ex**: for(int i = 0;i<a.length;i++){

System.out.println(a[i]);

}

While working with Batch updates if any query in the batch object is failed. JDBC driver throws java. Sql.BatchUpadeException.

Once if we sent the Batch object to database server we can clear the Batch by using cleared Batch method.

\***Requirement**: insert three records into emp table by using prepared statement?

import java.sql.\*; public class PreparedBatchInsert{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry"); PreparedStatement pstmt = con.prepareStatement("insert into emp values(?,?,?)"); pstmt.setInt(1,133); pstmt.setString(2,"suneel"); pstmt.setDouble(3,15500); pstmt.addBatch(); pstmt.setInt(1,144); pstmt.setString(2,"sudheer"); pstmt.setDouble(3,9876); pstmt.addBatch(); pstmt.executeBatch();

}

}

import java.sql.\*; public class PreparedBsnlBill{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from BSNLCUSTOMER");

PreparedStatement pstmt = con.prepareStatement

("update bsnlcustomer set bamount = ? where phoneno = ?"); while(rs.next()){ int noofc = rs.getInt("noofc"); int phoneno = rs.getInt("phoneno"); double bamount = noofc \* 0.10; pstmt.setDouble(1,bamount); pstmt.setInt(2,phoneno); pstmt.addBatch();

}

pstmt.executeBatch();

} }

Without JDBC driver we cannot develop java application.

Majorly there are three way’s are available to register the driver they are

1. DriverManager.registerDriver
2. By using class.forName

**Ex**: public class RetriveRecords{ public static void main(String[] args){ class.forName(“oracle.jdbc.driver.OracleDriver”);

Connection con = DriverManager.getConnection

(“jdbc:oracle:thin:@localhost:1521:xe”,”kesav”,”kesav chari”);

}

}

**Note**: According to the sun micro system documentation every driver class must be small and standalone. **According to documentation when even the driver class is loaded it should able to create the object is own and register Driver with the help of DriverManager.** According to this statement every driver class developed by the driver vendors must provide static block. The following is a sample code available as part of oracle driver class.

**Ex**: public class oracleDriver Implements Driver{ //provides the implement to the all 6 methods static{

OracleDriver o = new oracleDriver();

DriverManager.registerDriver(o);

}

Sun micro has released different versions of JDBC specification. **Ex**: JDBC 3.0 and JDBC 4.0

When ever a specification is released it is the responsibility of JDBC Driver vendors to provide the implementation generally between the version the changed the names of the java files.

**Ex**: Incase of JDBC 3.0 specification oracle ways are choose ojdbc14.jar in case of JDBC

4.0 implementation they choose ojdbc6.jar.

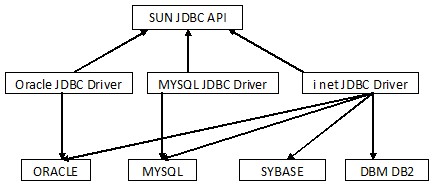
When the new versions of specifications are released then all new interfaces and new methods in the existing interfaces.

As part of JDBC4.0 specification sun micro system has added a feature to register the driver automatically (DriverManager.getConnection).

We can find the specification implementation version as part of META\_INF folders. When we work with JDBC4.0 to register JDBC driver automatically we need to make sure that we set the CLASSPATH to ojdbc6.jar.

Theoretically speaks a driver can communicate with any DB server when we consider oracle JDBC Driver. They have provided the code to communicate only with oracle DB server.

We cannot use oracle JDBC Driver to communicate with Mysql Database Server.



\*uses the enterprise edition;

USERNAME: root

PASSWORD: kesav

\*procedure to work with mysql database server.

Step 1: Login to mysql DBS by using mysql command line client.

Step 2: To work with mysql DBS we have to create DBS’s. The following is a command which is used to create database.

**Ex**: create database database mydb.

Step 3: To use the database we can use a command use my database.

\*Develop a java application to insert a record into emp table of mysql database server? import java.sql.\*; public class MysqlInsertRecords{

public static void main(String[] args)throws Exception {

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

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/mydb"."root","kesav");

Statement stmt = con.createStatement();

int no = stmt.executeUpdate("insert into emp values(999,'Mohan',60000); System.out.println(no);

} }

\*\*\*The problem with above code is we have hard coded. Driver class, URL, USER\_NAME and password because of these reason we are able to insert with only one DB server. If we want to same java program to insert with any database server without changing the code we should able to remove the hard coding.

\*The following an example of java application which can be used to interact with any database server? import java.sql.\*; public class AnyDatabaseServerInsertRecord{

public static void main(String[] args)throws SQLException,ClassNotFoundException{

String drv = System.getProperty(“drv”);

String url = System.getProperty(“url”);

String uname = System.getProperty(“uname”);

String pwd = System.getProperty(“pwd”);

Class.forName(drv);

Connection con = DriverManager.getConnection(url,uname,pwd); Statement stmt = con.createStatement();

int no = stmt.executeUpadate(“insert into emp values(888,’RajuRani’,143); System.out.println(no);

} }

imp: By removing the hard coding and if we would like to use

DriverManager.registerDriver we can use following approach.

public class HardcodingInsertRecords{

public static void main(String[] args)throws Exception{

String drv = System.getProperty(“drv”);

Class c = Class.forName(drv);

Object o = c.newInstance();

Driver d = (Driver)o;

DriverManager.registerDriver(d);

Connection con = DriverManager.getConnection(…..); Statement stmt = con.createStatement();

int no = stmt.executeUpdate(“insert into emp values(1,’Raju’,1000)”); System.out.println(no);

} }

But there is no use of writing DriverManager.registerDriver(),Because by using Class.forName we registering the Driver.

**Transactions**: (performing)

1. **Performing sequence of steps in a single unit is called as Transacion.**
2. Every transaction will have a starting point and ending point.
3. Every transaction is having two states.
   1. Success state
   2. Failure state
4. If all the steps in the transaction are executed successfully then we say transaction is success. If any one step is failed, we can say transaction is failed.
5. Once if one transaction is ended a new transaction will be started. In a project we can have any number of transactions. It is based on the project requirement.
6. Whenever we establish a connection with database server,then starts a transaction.
7. After establish the connection with database server then the user performs any operation. The data will store permanently when we end the transaction.
8. When one transaction is completed it is the responsibility of database server to start another transaction.
9. By default, it is the responsibility of JDBC driver to start a transaction as well as to end the transaction.
10. Whenever we establish connection with database server immediately JDBC driver starts the transaction.
11. Whenever we send a query to database server immediately JDBC driver ends the transaction.
12. When we establish the connection internally JDBC driver uses

***con.setAutoCommit(true);*** or

***con.setAutoCommit(false); and***

***con.rollBack();***”.

\*The following an example of user defined transactions?

import java.sql.\*; public class AutoCommiteInsertRecords{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

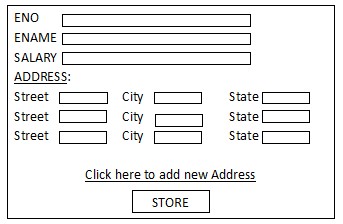
Connection con= DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry"); con.setAutoCommit(false);

Statement stmt = con.createStatement(); stmt.executeUpdate("insert into emp values(5,'Kiran',1111)"); stmt.executeUpdate("insert into emp values(6,'Naaga',2222)"); con.commit();

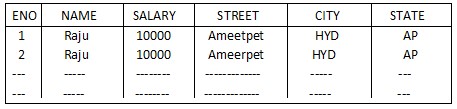
}

}



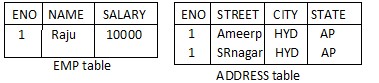
The above application is used to capture employee details and store it in a DBS. Our application should be able to support dealing with multiple address.

To achieve the above requirement we have two designs. Design1: In this design we create only table.



The disadvantage of this approach is we find the duplicate data in the table. It is not recommend using design1.

Design2: In this design we will try to have two tables. They are emp and Address tables.



\*The following example of user defined transactions to store the data into multiple tables?

import java.sql.\*; import java.util.\*; public class InsertTwoTableRecords{

public static void main(String[] args)throws SQLException{

Scanner s = new Scanner(System.in);

System.out.println("Enter the employee number"); int eno = s.nextInt();

System.out.println("Enter the employee ename");

String ename = s.next();

System.out.println("Enter the employee salary"); double salary = s.nextDouble();

System.out.println("Enter the employee street");

String street = s.next();

System.out.println("Enter the employee city");

String city = s.next();

System.out.println("Enter the employee state");

String state = s.next();

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav”,”kesav cherry”); con.setAutoCommit(false);

PreparedStatement pstmt1 = con.prepareStatement("insert into emp2 values(?,?,?)"); pstmt1.setInt(1,eno); pstmt1.setString(2,ename); pstmt1.setDouble(3,salary);

PreparedStatement pstmt2 = con.prepareStatement

("insert into address values(?,?,?,?)"); pstmt2.setInt(1,eno); pstmt2.setString(2,street); pstmt2.setString(3,city);

pstmt2.setString(4,state); pstmt1.executeUpdate(); pstmt2.executeUpdate(); con.commit();

} }

It’s not recommended to use throws as for the project. It’s always recommended to use try and catch blocks. By using try and catch block we can handle the errors.

When JVM executes a piece of code and if it fails JVM check for appropriate error handler. If it is available JVM execute the code if the error handler not available the JVM throws an error to the client (Exception stack trees).

As per as catch block we will provide the code when an exception is raised what code is executed.

There are some scenarios when we want execute the code. When the exception doesn’t occur we provide this code in finally block.

import java.sql.\*; import java.util.\*; public class DBConnectionTwoTables{

public static void main(String[] args)throws Exception{

Scanner s = new Scanner(System.in);

System.out.println("Enter the employee number"); int eno = s.nextInt();

System.out.println("Enter the employee ename");

String ename = s.next();

System.out.println("Enter the employee salary"); double salary = s.nextDouble();

System.out.println("Enter the employee street");

String street = s.next();

System.out.println("Enter the employee city");

String city = s.next();

System.out.println("Enter the employee state");

String state = s.next();

Connection con = null;

PreparedStatement pstmt1 = null;

PreparedStatement pstmt2 = null;

ResultSet rs = null;

con.setAutoCommit(false);

try{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver()); con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","malli"); con.setAutoCommit(false); pstmt1 = con.prepareStatement("insert into emp2 values(?,?,?)"); pstmt1.setInt(1,eno); pstmt1.setString(2,ename); pstmt1.setDouble(3,salary); pstmt2 = con.prepareStatement("insert into address values(?,?,?,?)"); pstmt2.setInt(1,eno); pstmt2.setString(2,street); pstmt2.setString(3,city); pstmt2.setString(4,state); pstmt1.executeUpdate(); pstmt2.executeUpdate();

}catch(SQLException se){ con.rollback();

System.out.println("some code is missing");

}finally{ con.close(); pstmt1.close(); pstmt2.close(); rs.close();

}

}

}

When even we provide user defined transactions. We must connect the transaction in try block. We must provide rollback in side catch block.

import java.sql.\*; public class DBServerConnectType4C{

public static void main(String[] args)throws SQLException{

Connection con = null;

Statement stmt = null; ResultSet rs = null; try{ DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver()); con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","malli"); stmt = con.createStatement();

rs = stmt.executeQuery("select \* from emp");

System.out.println(con);

System.out.println(stmt);

System.out.println(rs);

}catch(SQLException s){

}finally{ con.close(); rs.close(); stmt.close();

}

}

}

\***connection is an interface**:

How come we are able to create object to connection interface?

We are not creating object to connection interface, we are creating a reference variable to connection interface. These reference variables hold an object of a class which provides an implementation of connection interface.

The internal code of implementation class is responsible to really established connection with DB server.

**How to achieve polymorphism by using JDBC?**

Based on the interface like Connection, Statement, ResultSet we are able to achieve polymorphism. If we doesn’t the interface and if we use the implementation class as part of applications. Application will work with one database server.

\***MetaData**:

Data about data is called as MetaData.

In JDBC we have three types of MetaData available. They are:

1. ResultSetMetaData
2. DatabaseMetaData
3. ParameterMetaData

\***ResultSetMetaData**:

ResultSetMetaData is an object which gives more information about ResultSet object. By using ResultSetMetaData object we can find the number of column is available in ResultSet, the names of the columns, the Data types of the columns. **Ex**: ResultSetMetaData rsmd = rs.getMetaData();

\***DatabaseMetadata**:DatabaseMetadata is an object which gives name information about under line Database server. That is it can find the database server, version information and JDBC Driver information.

\***ParameterMetaData**:

ParameterMetaData is an object which gives more information about ParameterMetaData’s of PreparedStatement (possional parameter information is object).

To get ResultSetMetaData object we take the help of ResultSet object. We use a method get MetaData to get ResultSetMetaData object.

The following is example of using ResultSetMetaData object to find number of columns name of the columns and data types.

**Ex**: rs = stmt.executeQuery(“select \* from emp”);

ResultSetMetaData rsmd = rs.getMetaData();

System.out.println(rsmd.getColumnCount());

System.out.println(rsmd.getColumnName(2));

System.out.println(rsmd.getColumnTypeName(2));

System.out.println(rsmd.isSearchable(2));

System.out.println(rsmd.getColumnType(1));

\*Write a sample program to use ResultSetMetaData?

import java.sql.\*; public class RSMDConnection{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav”,”kesav cherry”);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from emp");

ResultSetMetaData rsmd = rs.getMetaData();

System.out.println(rsmd.getColumnCount());

System.out.println(rsmd.getColumnName(2));

System.out.println(rsmd.getColumnTypeName(2)); System.out.println(rsmd.isSearchable(2));

System.out.println(rsmd.getColumnType(2));

} }

\*Write a program to ResultSetMetaData with try and catch block?

import java.sql.\*; public class RSMDTryCatch{

public static void main(String[] args)throws SQLException{

Connection con = null;

Statement stmt = null; ResultSet rs = null; try{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver()); con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav”,”kesav cherry”); stmt = con.createStatement();

rs = stmt.executeQuery("select \* from emp"); ResultSetMetaData rsmd = rs.getMetaData();

System.out.println(rsmd.getColumnCount());

System.out.println(rsmd.getColumnName(2));

System.out.println(rsmd.getColumnTypeName(2));

System.out.println(rsmd.isSearchable(2));

System.out.println(rsmd.getColumnType(1));

}catch(SQLException s){

System.out.println("some proble in the above code");

}finally{ con.close(); stmt.close(); rs.close();

}

} }

Sun micro system as given a class java.sql.types this class doesn’t contain any methods. This class contains only static and final variables. We will never instantiate the type’s class this is because it doesn’t contain any methods. This class is used to map generic SQL types.

\***DatabaseMetaData**:

By using DatabaseMetaData we can find the information about underline Database Server. The information like version number of Database server, JDBC Driver version as well as JDBC specifications also.

import java.sql.\*; public class DatabaseMetaDataConnection{

public static void main(String[] args)throws SQLException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav”,”kesav cherry”);

DatabaseMetaData dbmd = con.getMetaData();

System.out.println(dbmd.getDatabaseMajorVersion());

System.out.println(dbmd.getDatabaseMinorVersion());

System.out.println(dbmd.getDatabaseProductName());

System.out.println(dbmd.getDriverMajorVersion());

System.out.println(dbmd.getDriverMinorVersion());

System.out.println(dbmd.getJDBCMajorVersion());

System.out.println(dbmd.getJDBCMinorVersion());

}

}

Generally we use DatabaseMetaData at the time of installing the software. It will check the underline Database server version is match with required DBS’s or not.

\***ParameterMetaData**:

This object is used to find the information about perameters of PreparedStatement. To work with PerameterMetaData first we must get PreparedStatement object. We use a method getParameterMetaData.

All Driver vendors as not provide the implementation to all the methods of ParameterMetaData.

**Ex**: ParameterMetaData pmd = pstmt.getParameterMetaData(); System.out.println(pmd.getParameterCount());

**Types of Drivers**:

The JDBC drivers are classified into 4 types. They are:

TYPE 1: JDBC-ODBC Bridge Driver

TYPE 2: Native API Driver

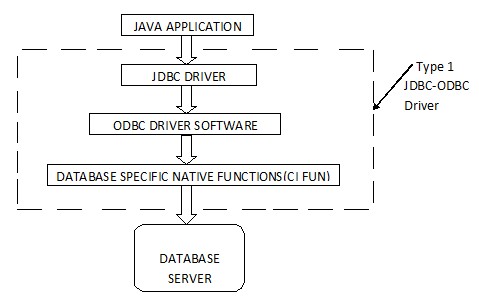
TYPE 3: Network protocol Driver

TYPE 4: Pure java Driver/thin Driver

**JDBC-ODBC Bridge Driver**:

**We can use this driver to interact with any database server. This is the first driver available in the market to interact with DB Servers.**

The following architecture of type1 JDBE-ODBC Driver.



\***Disadvantages of type 1 driver**:

1. If we would like work with type one JDBC driver. Developer/Client as to install software of database to get CI functions we have install ODBC Driver software and JDBC driver software and JDBC driver software. We have to configure JDBC driver to communicate with ODBC Driver.
2. Type1 Driver is platform dependent this is because CI functions are developed in ‘C’ language.
3. MS as released ODBC API to develop a ‘C’ program to interact with any database server. The internal code of the ODBC drivers uses OCI functions(native functions) to communicate with Database servers.
4. If the .DLL’s are not available in ODBC driver software we have install then separately.
5. By default ODBC Driver available with windows operations system.

(Start control panel administrative tools data source)

1. Type1 JDBC Driver is developed by only sun micro system. No body else has developed type1 JDBC Driver. Sun micro system has integrated type1 JDBC Driver with java software only. We no need to install type1 JDBC Driver saperatly.
2. When we observe the .DLL file of oracle we have found that internally it is use it ODBC API functions. The function survive SQL connect, SQL disconnect, SQL execute and etc. these functions internally communicate with OCI functions.

\***Procedure to configure ODBC Driver**:

* Open ODBC data source administrative window.
* Choose system DSN tab and click on add button.
* From the list of available ODBC drivers choose the required driver (Oraclexe).
* From the driver configure window provide data source name, description, userid and click on “OK”.

\*The following is an example of using type1 Database connection Driver? import java.sql.\*; public class InsertRecords{

public static void main(String[] args){

DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());

Connection con = DriverManager.getConnection

(“jdbc:odbc:myoracleds”,”system”,”malli”);

Statement stmt = con.createStatement();

stmt.executeUpdate(“insert into emp values(1,’abc’,234)”);

} }

When we use type1 driver the JDBC Driver calls has to be translated to ODBC calls has be translate to JDBC calls. Because of these reason we see a performance impact. This driver is slowest in all the Drivers.

This is not at all recommended to use develop web applications.

\***Advantage of type1 Driver**:

1. People say by using type1 Driver we can interact with any Database server. This is because already ODBC Drivers are available to all database servers.
2. If we would like to use any SQL to interact with by using type1 JDBC Driver. The Driver is not available in the last of available drivers. We are installed it seperatly and configure ODBC Driver.

\***Requirement**:

Develop a java program application to read the data from micro soft access file system by using type1 driver?

import java.sal.\*; public class RetriveRecords{ public static void main(String[] args){

DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());

Connection con = DriverManager.getConnection(“jdbc:odbc:mysqlds”,”system”,”malli”);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery(“select \* from emp”);

} }

We can develop a java application to interact with excel spread sheet.

When we are using excel sheet the sheet name as tread as table name. the first row is treated as columns names. import java.sql.\*; public class RetriveRecords2{ public static void main(String[] args){

DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());

Connection con = DriverManager.getConnection("jdbc:odbc:myds"," "," ");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from [sheet1$]"); while(rs.next()){

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

System.out.println(rs.getString(3));

}

}

}

\*\*\*Moving the data from one database server to another database server is called as data migration. We can develop a java program to migrate the record from Excel to Oracle.

import java.sql.\*; public class RetriveRecords3{ public static void main(String[] args){

DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());

Connection excelcon = DriverManager.getConnection("jdbc:odbc:myds","","");

Statement excelstmt = excelcon.createStatement();

ResultSet rs = excelstmt.executeQuery("select \* from [emp$]");

Connection oraclecon = DriverManager.getConnection

("jdbc:odbc:oracleds","kesav","kesav cherry");

PreparedStatement pstmt = oraclecon.prepareStatement

("insert into emp values(?,?,?,)"); while(rs.next()){ pstmt.setString(1,rs.getString(1)); pstmt.setString(2,rs.getString(2)); pstmt.setString(3,rs.getString(3)); pstmt.addBatch();

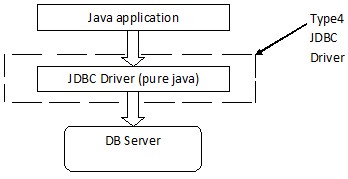
}

pstmt.executeBatch();

} }

\***Type4 Driver**:(pure java Driver/thin Driver)

The following is architecture of type4 JDBC Driver.



**Advantages of type4 driver**:

* This Driver is development in pure java. Because of those we can say this Driver is platform independent.
* This is fastest Driver in all the Drivers. This is because Driver directly communicates with database server.
* We need to install any client software.

**Disadvantage of type4 driver**:

They are some people who say download in a specific driver is a disadvantage of type4 Driver.

**JNI:**

JNI stands for Java Native Interface JNI is an API which is used to develop a java program to call the ‘**C**’ language functions.

The following is an example code to call ‘C’ language functions from java.

**Ex**: public class HelloWorld{ public native void SayHello(){ static{

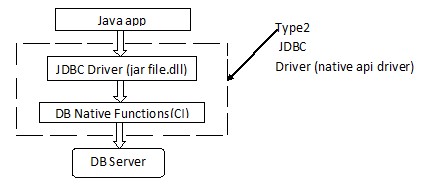
System.loadLibrary(“Hello”);

}

public static void main(String[] args){ HelloWorld h = new HelloWorld() h.SayHello();

} }

\***Type2 Driver (Native API Driver)**: The following is architecture of Type2 JDBC Driver.



Oracle people have placed all the .DLL which is required to interact with OCI functions in the bin folder. As part of oracle10g the name of the DLL which uses OCI functions is OCI JDBC.dll.

Oracle guys are clubbed the classes of T4 Driver and T2 Drivers and placed in “ojdbc14.jar”.

The following example of T2 Driver to retrieve the data from Database Server.

**Ex**:

import java.sql.\*; public class Type2Driver{

public static void main(String[] args)throws Exception{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:oci:@localhost:1521:xe","kesav","kesav cherry");

PreparedStatement pstmt = con.prepareStatement("insert into emp2 values(?,?,?)"); pstmt.setInt(1,3); pstmt.setString(2,"kkk"); pstmt.setDouble(3,66666); pstmt.executeUpdate(); pstmt.executeBatch(); pstmt.addBatch();

}

}

\***Disadvantages of Type2 Driver**:

This Driver not purely written in java. This is platform dependent.

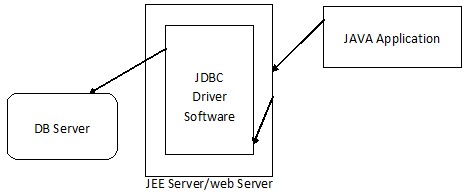
Like Type1 Driver it has to translate the java language calls to ‘C’ language calls.

Client is supposed to install client software (DB client software) when ever there is a change in DB Server.

Type2 Driver gives better performance when compared with Type1 Driver.

\***Type3 Driver (Net work Protocol Driver)**:

To store the data into database server you are using the data types like BLOB (Binary long object) or CLOB data types (Character long object). To store the raw data like images we use BLOB data type to store the character data we are using CLOB data type. The following the architecture of T3 Driver:



\***Requirement**:

Develop a java application to store image into database server or file?

SQL> create table empimg (eno number(5), name varchar2(20), image blob);

import java.sql.\*; import java.io.\*; public class StoreImageT3{

public static void main(String[] args)throws SQLException,FileNotFoundException{

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

PreparedStatement pstmt = con.prepareStatement("insert into empimg values(?,?,?)"); pstmt.setInt(1,4); pstmt.setString(2,"anc");

File f = new File("Jellyfish.jpg");

FileInputStream fis = new FileInputStream(f); pstmt.setBinaryStream(3,fis,(int)f.length()); pstmt.executeUpdate();

} }

\*Generate the FileOutputStream code?

import java.sql.\*; import java.io.\*; public class StoreImage1{

public static void main(String[] args){

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from empimg"); rs.next();

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

FileOutputStream fos = new FileOutputStream("one.jpg"); fos.write(getBytes(3));

} }

\***Date**:

We have a data type a “date” to deal with dates in the project.

By default oracle as supply the data format date-month-year (DD-MM-YY).

SQL> create table emp(eno number(5), name varchar2(20), doj date);

**Ex**: insert into emp values(1,’Raju’,sysdate); // this current system data display insert into emp values(2,’Naveen’,’05-Feb-12’);

As part of AWT/Swings generally when the user will be write the code to store the raises a data into Database server.

**Ex**: J Button.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){ try{

//JDBC code to store in DB Server }catch(SQLException s){ s.printStactTrace();

}

}

# \*\*\*JNDI\*\*\*

**JNDI**: (Java Naming and Directory Interface)

We use JNDI to develop a java application to interact with directory servers.

Similar to Database servers we have directory servers. Directory servers also use to store the data.

As part of Database server we store the data in the form of relational records as part of directory server. We store the data in the form of objects.

\***Difference between Database server and Directory servers**:

Directory servers give the best performance if we want to the store the data once and retrieve the data for multiple times.

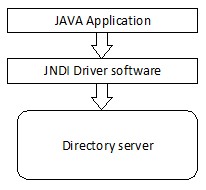
As part of Database servers we can store huge amount of data. Directory servers are not meant to store huge amount of data. They can store only small amount of data. (Yahoo, Google and etc…)

To interact with database servers where using query language. To interact with directory server we have to use the predefined functions/methods.

Directory servers can’t store the data permanently. These servers are meant to store the data temporally. In most of the project we use both database servers and directory servers.

In most of the projects we write the java code to store the data into Database server permanently. Once if data into data is stored in server we write another java program to retrieve required data represent in the form of object and store it in directory server.

JNDI API is mainly used to develop a java application to interact with directory servers. The following is architecture of JNDI.



They are so many directory servers are available some of them all.

1. LDAP (Light weight Directory Access Protocol) from open source apache.
2. ADS (Active Directory Server) from Micro soft
3. NDS (Novel Directory Server)
4. DNS (Domain Naming Server)……etc.

We no need to install directory servers separately. Now way days all the JEE servers are integrated directory servers. When we install JEE servers we get the directory server also. They following some of the JEE servers.

**Ex**:

* 1. Weblogic
  2. JBOSS
  3. Tomcat
  4. Websphere
  5. Resin

When install weblogic server it got install in a home directory called as BEA.

To work with weblogic server we must configure the weblogic server. That meaning of weblogic server is placing all the required files in a folder.

To work with weblogic server we have to configure the weblogic server (or) we have to create a Domain.

The meaning of creating a domain is create the folder and copy all the required files into it.

\***Procedure to create domain in weblogic server**:

Start the weblogic server configuration wizard (start -> All Programs -> oracle weblogic -> weblogic10gR3 -> tools -> click on configuration wizard).

The above step launches a window how’s name is oracle weblogic configuration wizard.

From this select an option creates a new weblogic domain and click on NEXT.

Select an option generates a domain configured automatically and click on NEXT.

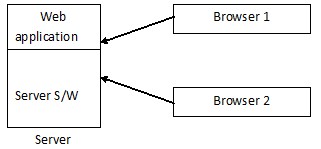
Enter the username and password which act as on administrator.

Select the development mode and available JDK. Choose the customization as NO and click on create BUTTON.

To start the weblogic server oracle guys are provide CMD files and SH files. We use start weblogic CMD to start the server.

Now a days people are not give separate client software the client software is integrated with the server in the form of web based application.

The advantage of this approach is we now to install the client software separately. We can access the client software by taking the help of BROWSER. To access oracle client software where using the following URL. **Ex**: <http://localhost:8080/apex/>



The following is URL which is used to access admin console of web logic server. **Ex**: <http://localhost:7001/console/>

The default port number of weblogic server is 7001 and the default server name is “admin”. We can change the default port number and server name. We can use customization option to use our own port number as well as our own server names. In the projects most of the time we change the settings according to our project requirements.

Sun micro system as released JNDI API to develop a java application to communicate with directory server. Sun micro system as released all the classes and interfaces as part of javax.naming package. The most important interface is context. The most important class is initial context. Initial context class provide the implementation of context interface.

Context

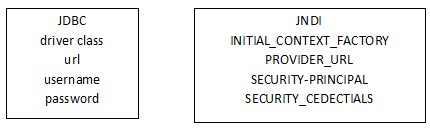
Interface

Initial context

Class

As a programmer we are responsible to develop a java application to create an object and store the object into directory server Update, Delete and Retrieve objects directory server.

To develop a JNDI application we have to supply the following four values.



The following are the steps to develop JNDI Application.

1. Create Hash table.
2. Store the details in the Hash table using keys and values.
3. Get the connection to directory server by supplying Hash table object as input.
4. Call the methods bind/unbind/rebind/lookup methods to perform the operations with Directory server.

\*Develop a JNDI application to store a string object into directory server. To store an object in the directory server we use a method bind.

**Syntax**: void bind(key, Object)

**Ex**: import java.util.\*; import javax.naming.\*; public class StoreObject{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context ic = new InitialContext(h); String name = "Raju"; ic.bind("uname",name);

}

}

The bind() method convert the object into super class object and stored it in directory server. In the above example string object is stored into directory server. By converting it into super\_class\_object object.

\*The following example demonistrate how to retrieve the data from directory server.

import java.util.\*; import javax.naming.\*; public class RetrieveObject{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context c = new InitialContext(h);

Object o = c.lookup(“uname”);

String s = (String)o;

System.out.println(s);

} }

With key we are using to searching if it is not available. It got a error message NameNotFoundException.

\*We using a method rebind to update a Record a directory server.

import java.util.\*; import javax.naming.\*; public class UpdateRecord{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv"); Context c = new InitialContext(h); c.rebind(“uname”,”Naveen”);

} }

\*Unbind method is used to delete the Record a directory server.

import java.util.\*; import javax.naming.\*; public class DeleteRecord{ public static void main(String[] args) throws NamingException{ Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

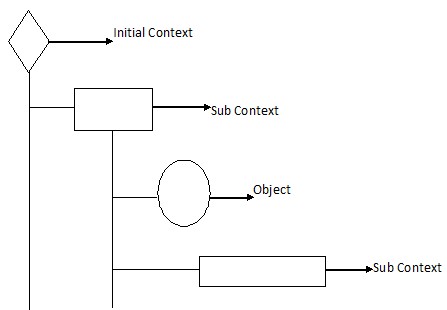
h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv"); Context c = new InitialContext(h); c.unbind(“uname”);

} }

It is not recommended to store data directory into initial context. This is because of if we store directory the search operation takes time. It always recommended store the data into sub context (sub context is like a folder).



\*To create the sub context use a method.

import java.util.\*; import javax.naming.\*; public class CreateSubContext{

public static void main(St~~ring[] args) thro~~ws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv"); Context c = new InitialContext(); c.createSubcontext(“Btech”);

c.createSubcontext(“mca”);

} }

\*To create the sub context inside the sub context we can create the separate dot.

import java.util.\*; import javax.naming.\*; public class CreateSubContext1{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context c = new InitialContext();

c.createSubcontext(“Btech.1styear.mech”);

c.createSubcontext(“Btech.1styear.it”);

c.createSubcontext(“Btech.1styear.csc”);

} }

\*if we want to create a sub context in another sub context we need to make sure that the base sub context is available.

import java.util.\*; import javax.naming.\*; public class CreateSubContext{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv"); Context c = new InitialContext(); c.createSubcontext(“btech”);

c.createSubcontext(“btech.1styear”);

c.createSubcontext(“Btech.1styear.mech”);

c.createSubcontext(“Btech.1styear.it”);

c.createSubcontext(“Btech.1styear.csc”);

} }

\*To store an object into a specific context we have to specify absolute context path.

import java.util.\*; import javax.naming.\*; public class CreateBind{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv"); Context ic = new InitialContext();

ic.bind(“betch.1styear.csc.ramesh”,”abc”);

} }

\*To remove sub context we use a method destroy sub context.

import java.util.\*; import javax.naming.\*; public class CreateBind{

public static void main(String[] args) throws NamingException{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context c = new InitialContext();

c.destroySubcontext(“betch.1styear.csc”);

} }

\*\*\*\***Connection pool**:

When we develop a java application to get the connection from Database server. By using DriverManager.getConnection we always get physical connections.



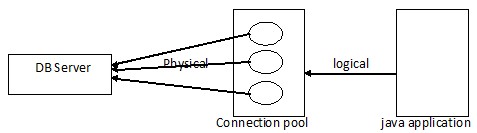
If we got a physical connection and we are not closing the connection after we above work the other application can not use the same connections.

Connection pool is java program which manages the connections of Database server.

Connection pool contains set of connections.

There are so many connection pool programs are available some of them are.

1. DBCP (Database connection pool)
2. C3p0 connection pool
3. Weblogic connection pool
4. JBOSS connection pool
5. Tomcat connection pool ……. Etc.



Generally connections pool program is released in the form of jar files. The jar file contains set of class.

The meaning of using connection pool program is creating the object class and supply Driver class, url, username, password and initial capacity.

\*The following is an example of using DBCP connection pool program. import org.apache.commons.dbcp.\*; import java.io.\*; import java.sql.\*; public class DBC3P0{

public static void main(String args[])throws IOException{ BasicDataSource bds = new BasicDataSource(); bds.setDriverClassName(“oracle.jdbc.driver.OracleDriver”); bds.setUrl(“jdbc:oracle:thin:@localhost:1521:xe”); bds.setUsername(“system”); bds.setPassword(“malli”); bds.setInitialSize(3);

Connection con1 = bds.getConnection();

System.out.println(con1);

System.in.read();

System.in.read();

Connection con2 = bds.getConnection();

System.out.println(con2);

System.in.read();

System.in.read();

Connection con3 = bds.getConnection();

System.out.println(con3);

System.in.read();

System.in.read();

Connection con4 = bds.getConnection();

System.out.println(con4);

System.in.read();

System.in.read();

Connection con5 = bds.getConnection();

System.out.println(con5);

System.in.read();

System.in.read();

}

}

\***Using C3P0 connection pool**:

The following is an example of using c3p0 connection pool. import com.mchange.v2.c2p0.\*; import java.sql.\*; public class DBc3p0Connection{

public static void main(String[] args)throws Exception{

ComboPoolDataSource cpds = new ComboPoolDataSource(); cpds.setDriverClass(“oracle.jdbc.driver.OracleDriver”); cpds.setJdbcUrl(“jdbc:oracle:thin:@localhost:1521:xe”); cpds.setUser(“system”); cpds.setPassword(“malli”); cpds.setInitialPoolSize(3);

Connection con = cpds.getConnection();

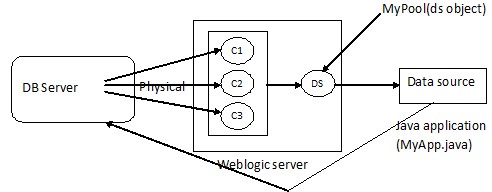
System.out.println(“Connection:” +con);

System.in.read();

System.in.read();

} }

\***Procedure to work with weblogic connection pool**:



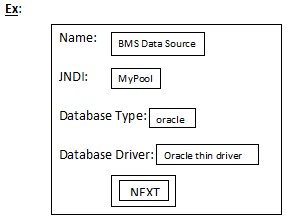
\***Procedure to configure weblogic connection pool**:

As part of the admin console go to Domain Structure services JDB

Data Source.

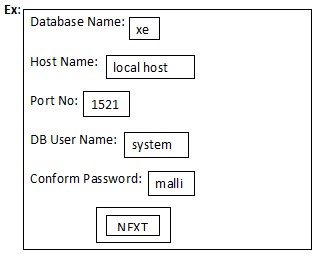
|  |
| --- |
| NEW |

This will list out all available Data sources. To create the new Data Source select button. Supply the following details for JDBC Data Source properties.



Name is used to identify that available Data Source. JNDI name is used to store the data source object into directory server. DB type and DB driver are used to which DB Server it connects.

It will display the transaction options by default “one-phase commit” option are coming. Supply the following connection properties.



When we fill the above form it will display driver class url, username and password. To check whether. The details are valid or not we use on option “Test Configuration” Associate the connection pool program with admin server and click on FINISH button. To change the initial capacity and capacity increment select the Data Source connection pool tab.

When we Run the above connection pool program it is acquired five connections and the information about the connections is stored in Data Source object and it is stored a Directory server.

\*The following java program get the connection from connection pool.

import java.util.\*; import javax.naming.\*; import java.sql.\*; import javax.sql.\*; public class DBConnect2{

public static void main(String[] args) throws Exception{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context c = new InitialContext(h);

Object o = c.lookup(“mypool”);

DataSource ds = (DataSource)o;

Connection con = ds.getConnection();

System.out.println(con);

System.in.read();

System.in.read();

} }

// set CLASSPATH=wlclient.jar;.;

// set CLASSPATH=c:\bea\wlserver\_10.3\server\lib\weblogic.jar;.;

// echo %CLASSPATH%

// echo %PATH%

//c:\bea\user\_projects\domains\mydomain\bin\setDomainEnv.cmd

// c:\bea\user\_projects\domains\mydomain\cd\

To set the class path to “weblogic.jar;.;” we can use “set DomainEnv.cmd”.

\*Retrieve the Records program?

import java.util.\*;

import javax.naming.\*; import java.sql.\*; import javax.sql.\*; public class RetrieveRecords1{

public static void main(String[] args) throws Exception{

Hashtable h = new Hashtable();

h.put(Context.INITIAL\_CONTEXT\_FACTORY,"weblogic.jndi.WLInitialContextFactory");

h.put(Context.PROVIDER\_URL,"t3://lacalhost:7001/");

h.put(Context.SECURITY\_PRINCIPAL,"admin");

h.put(Context.SECURITY\_CREDENTIALS,"inetsolv");

Context c = new InitialContext(h);

Object o = c.lookup(“mypool”);

DataSource ds = (DataSource)o;

Connection con = ds.getConnection();

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery(“select \* from emp”); while(rs.next()){

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

System.out.println(rs.getString(3));

}

} }

By default weblogic server is uses java1.6 some times we may get error message saying that “unsupported class version” as part of connection pool program. The problem this is to compile the java program by using one version and Run the program another version.

To resolve this problem after “setDomainEnv.cmd” is executed recompile the program and run it.

If the maximum capacity connection pool is ‘10’ and trying to get more number of connection to we getting an Exception is “PoolLimitSqlException”.

When we use the connection pool also it’s mandatory that we must close the connection. In con.close() method if it is a logical connection it will be return to connection pool.

# \*\*\*SERVLETS\*\*\*\*

The technologies likes SERVLETS and JSP(java server program) are used to developed web based applications.

By using java we can develop the following four types of applications.

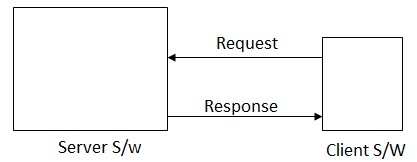
1. Standalone applications
2. Web based applications
3. Applets applications
4. Mobile applications

**Standalone Applications**

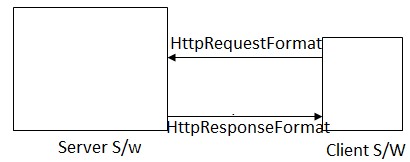
All the applications which we have developed as of now are standalone application. Standalone application will have a “main()” method. Standalone applications run on their own. They are dependent on other program. Whereas web based application are dependent on “Servers”.

**Disadvantages**:

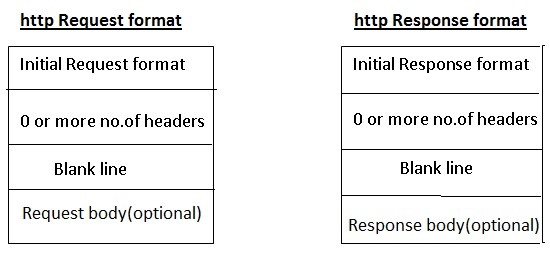
1. As the standalone application Run’s on client computer, so, we need to install it in all the computers.
2. We will have a lot of maintained related problems as part of the project. That is when ever to change the project we have placed with in all the client computers.
3. A Standalone application consumes the resources of client computer (Hard disk, Processor and etc).
4. Standalone applications are person independent only some fellows know how to install and settings the required configuration.
5. To develop the web based applications meagerly two programs are required.
   * 1. Server 2.Client
6. The client sent to request the server takes the request and processes it sent the response to client.



1. There are so many server software’s are available. some of them are **Weblogic, Tomcat, Websphere, Jboss, resign and glash fish etc.**
2. There are so many client software’s are available some of them are **IE (Internet Explorer), Chrome, Firefox, Opera, Safari and Netscape etc.**
3. As a developer we are responsible to develop a program which Run’s inside the server.
4. To develop the web based applications we required “http” protocol. (Server and client are same Protocol using with http)
5. http protocol is divided into two parts. They are:
   1. http Request format
   2. http Response format



The following are http Request format and Response format.



**Initial Request Line**:

Initial Request Line is divided into following three parts. They are:

|  |  |  |
| --- | --- | --- |
| **Method** | **Requested Resource** | **Protocol/Version** |

**Method**: The method indicates what operation has to be carried out by the server.

The following are method of http protocol.

They are:

**GET, PUT, POST, DELETE, TRACE, HEAD, LOCATE** // Server Methods

**Request Resource:** This is indicating the resource which has to be proceeding by the resource may be available or may not available.

**Protocol/Version**: Indicates which version of HTTP protocol is used by client.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GET | /four.html | HTTP/1.0 |  | GET | /four.html | HTTP/1.1 |

**Header**: The Header is used to send extra information about client to server. The following is header format.

|  |  |
| --- | --- |
| Header-Name: | Value |
| (Send to extra information) | (0 or more) |

The following are the more important of two headers.

1. User-Agent
2. Accept-Language

**User-Agent**: Indicates which browser program has sent to request to server.

To represent a next line character we use CR LF.

We can represent CR and LF by using \r\n characters.

**Accept-Language**: Header is use to specify what language is acceptable by the client based on the Accept-Language server will send the output to the client.

**Initial Response Line**:

Initial Response line is divided into three parts.

|  |  |  |
| --- | --- | --- |
| Protocol/Version | Status Code | Status Message |

**Protocol/Version**: indicates the protocol used by the server.

**Status Code:** indicates the status code of the Request send by the server.

HTTP protocol guys are given couple of status code.

100 to 199 Information

200 to 299 Successes

300 to 399 Re-Direct

400 to 499 Request Resources not available

500 to 599 Failed to execute Request Resource

**Status Message:** every status go to will have status message.

**Ex**: 200 – Ok

404 – Not Found

* When the server send 2xx status code we can understand that server is able to process. The request sends by the client properly.
* When we get 404 or 4xx we can understand that the Request Resources is not available in server. When we get 5xx the Resource is available with server but failed to process the resource.
* 3xx is indicates re-direction to other servers. An HTTP Response header indicates server sending extra information to client (server client).
* The most important header send by the server is “Context type: text/html”. Based on the context type client will render output to the user. It is the responsibility of the client to rend/display output to the client.
* Method indicates which operation has to be carried out by the server.

**Get()**:This is the default method send by client to server. When the server receives the get request server process the request and send the response to client.

In case of get request if the client want to send the data to server. The data gets appended to URL and it will be send to server.

**Ex**: . . . GET /two.html?uname=raju&pwd=xyz HTTP/1.1

**POST():** Post() method is also used to execute the resource in the server.

When we use post request the browser capture the data and the add it to request body and send to server.

**Ex**: post /two.html HTTP/1.1

Accept-language: en-us

uname=raju & pwd=mypassword

**What is different between GET and POST methods?**

* In case of get() method request the data will be appended to url and send to server.
* Who’s as from the post request the data will be appended to request body.
* It’s not recommended to use get() to transfer sensitive data.
* This is a limitation of 1024Kb size to transfer the data by using get() method. This is no limitation of transferring the data by using post() method.

**PUT():** Put() method is used to place a resource into server. Because of security reason nun of the server supports this method.

**DELETE():** This method is used to delete the resource from server. Because of security reasons non of the server supports this methods.

**TRACE AND LOCATE()**: These methods are used to search for the files inside the server. Because of the security reasons none of the server vendors supports these methods.

**HTTPS**: Protocol is used to transfer the data between client and server in a security mode. When we use https protocol the data will be encrypted and it will be send it to server.

**HEAD()**: This method is used to specify only the header portion send by the server to client generally. We use this as part of request paches between the server.

**Ex**: 1xx information

3xx redirect

**PROTOCOLS**: These are two types of protocol classifieds. They are:

1. State less protocol
2. State full protocol

**State less protocol**: A state less protocol can’t remember all the conversations which is happening with client.

**Ex**: HTTP Protocol

**State full protocol**: A state full protocol remembers all the conversation. Which is happening with the client?

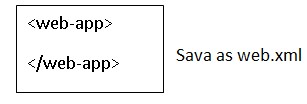
**Ex**: smtp, sntp, tcp api and etc.

**Procedure to develop weblogic application**:

1. Create a folder who’s folder name must be project name (there should not spaces).

**Note**: we call this folder as project context/web context.

1. Inside the above created folder we create a folder with the name “WEB-INF” (must be write the only in capital letters).
2. Inside the “WEB-INF” cerate classes and lib folders.
3. Create a file whose file name must be “web.xml” (inside WEB-INF).



\***Project Structure**:

Webapps //folder

|  |  |  |
| --- | --- | --- |
|  |  | |
| W | EB-INF //folder  Classes //folder |
|
|  | |  |

~~l~~ib //folder web.xml //text document

1. By using java we can develop both static and dynamic web-based applications.
2. We can develop two types of web-based applications. They are:

.What are static web-based applications?

* 1. Static web-based applications
  2. Dynamic web-based applications

1. **When the client send the request to the server, if the server is sending same output for multiple times. Then we call as static web-based applications.** To develop static web-based application we use **HTML, java script and images**.
2. **If the output is getting change the every time we call the applications as dynamic web-based applications** we use **servlets, JSP’s** to develop dynamic web-based applications.
3. **“WEB-INF”** folder is called as a **“private”** folder, because the file in this folder can be accessible only by the server.
4. **web.xml file is called as “Deployment descriptor”**.
5. **All the html files of our project must be placed inside “project and outside WEBINF” folder.**
6. Once if we are ready with the project, we must deploy the project.
7. \*\*what is the meaning of Deployment?

Placing the project in a server specific folder is called as “Deployment”.

1. **By default tomcat server uses 8080 port number in tomcat we use webapps folder as the Deployment folder.**
2. To deploy a project it’s the recommended to starts the server and place the project in webapp (deployment) folder.

<http://localhost7001/inetsolv/one.html>or web.html

1. To deploy a project in weblogic server we use a folder auto deploy (weblogic server).
2. Most of the times we need to read deploy the project in the server.

**What is the re-deployment?**

Removing the existing project and deployment the project is called as re-deployment. In case of java web-based applications the web applications are called as portable. We can deploy the project on any server.

In windows operating system because security reasons they are provided a UAC (User Access Controls) controls



**Note**: Re-Start

* In tomcat when we see an error message address already in use or JVM\_Bind the problem is because of some other program is already is using the same port number by change the port number of tomcat server we can re-solve it.
* We can **change the port number** of tomcat server manually by going to **“server.xml”** file. We can find this xml file as part of confi folder.
* Tomcat server is released in the form of .zip file also. To run the tomcat S/W from the zip file are use “startup.batch”.
* Tomcat S/W is dependent on an Enviroment variable.
* Servlet API is used to develop web based application.
* The following are the most Important “**packages**” to develop servlets. They are:

1. **Javax.servlet**
2. **Javax.servlet.http**

The following are the most important interfaces and classes of javax.servlet packages.

**Javax.servlet**:

**Interface**:

1. **Servlet**
2. **ServletRequest**
3. **ServletResponse**
4. **ServletConfig**
5. **ServletContext**

**Classes**:

**GenericServlet**

The following are the most important interfaces and classes of javax.servlet.http packages.

**Javax.servlet.http**:

**Interfaces**:

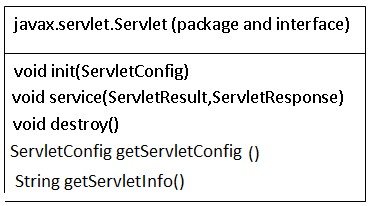
1. **HttpServletRequest**
2. **HttpServletResponse**

**Classes**:

**HttpServlet**

**Servlet**

The servlet is a java program which provides the ‘**implementation**’ of servlet interface directly or indirectly. The following is UML diagram of servlet interface.



**Procedure to develop first servlet program**:

**Step 1**: Develop a class which provides the implementation of Servlet Interface.

public class FirstServlet implements Servlet

{ ServletConfig config;

public void init(ServletConfig config)

{ this.config = config;

System.out.println(“we are in FirstServlet init() method”);

}

public void service(ServletRequest request, ServletResponse response)

{ System.out.println(“we are in FirstServlet service() method”);

}

public void destroy()

{ System.out.println(“we are in destroy() method”);

}

public ServletConfig getServletConfig()

{ return config; }

public String getServletInfo()

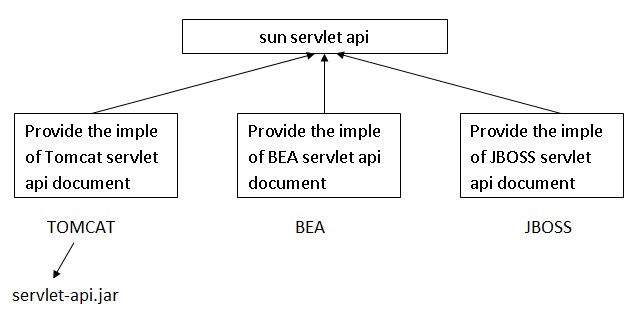
{ return “this is my First servlet”; // return “fs”;

}

}

**Step 2**: compile the FirstServlet program.

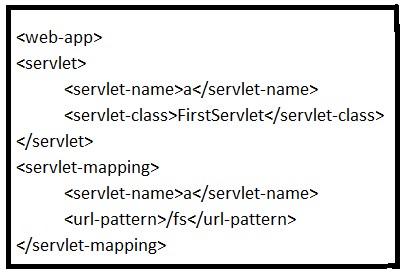
* To compile the servlet program we have set the class path to a jar file which provides the implementation of servlet api.
* The name of the jar files vary between company to company. In case of tomcat server the name of the jar file which provides to implementation of “**servlet-api.jar**”.



**Step 3**: create a web-based application.

**Step 4**: configure the servlet into above web-based application.

**Step 4.1**: copy all servlet related .class files into classes folder. **Step 4.2**: configure servlet into web.xml file.



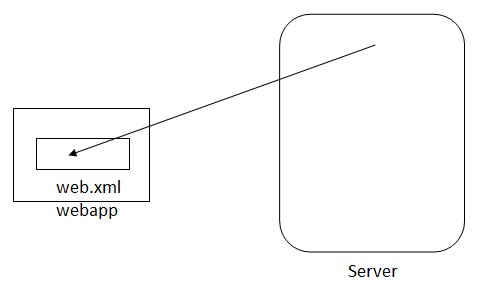
**Step 5**: Deploy the project in any server.

**Step 6**: The following **url** which is used to access the first servlet.

<http://localhost:8000/webapp/fs>

**Step 7**: It is mandatory that every servlet must be configure in “**web.xml**” file. When we deploy the project server reads the contents from ‘**web.xml**’ file.

**Step 8**: when we deploy any web-based application, then server will search for web.xml file. If the server found the xml file, then it will read the contents from web.xml file and store it in JVM’s memory.



**Note:**

When we observe the above behavior of tomcat server, If it doesn’t found web.xml it’s not displaying any error message, Whereas the web logic server has displayed an error message invalid application type.

**Parsers**

To read the contents from xml files are required “**parser**” programs. In java there are two types parsers are available.

1. **SAX parser**
2. **DOM parser**

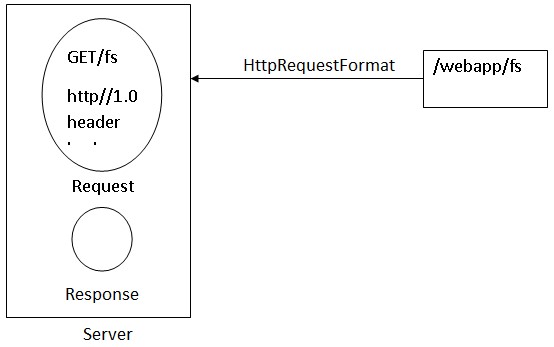
**SAX parser:**

When deploy a project if the xml file is valid xml file then server reads the contents and store it in JVM’s memory. If the xml file is invalid SAX parser program throws an error message. When we develop a project, server will not create any servlet objects.

|  |
| --- |
|  |

Tomcat Server

When the client sends the request to server, server creates two objects. They are request object and response object. Now the server opens http request format and store the data into request object.



It is the responsibility of server to remove request and response objects after finishing the processing of resource.

httpRequestFormat

httpResponseFormat

Tomcat Server Client

After the client has sent the request to server, server opens request and get the requested resource. Now the server checks is there any url pattern configured in web.xml file for that requested resource.

If it is available if does the remaining work. If it is not available in web.xml file server sends httpResponseFormat by using a status code you.

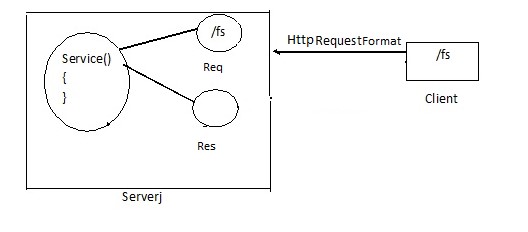
If the resource is available in web.xml file, server gets the servlet class name. Now the server checks is there any servlet object is available in the server. If not available it is the responsibility of server to create servlet object.

The server will try to load the class from class’s folder. If the class file is not available in the classes folder, server throws an Exception java.lang.classnotFoundException.

If the class is loaded successfully server creates first servlet object and call the init().

**Note**: init() will be called only once when the servlet object is created.

Now the server will status the Exception service() to Execute the service() server supply request and response objects as parameters.



After the server has executed service() if it is the responsibility of server to send the response back to client. When the server sends the response to client, server removes request and response objects. The FirstServlet object remains same in the server. It is the responsibility of server to remove servlet object. In the following two scenarios server will remove servlet objects.

1. When we stop the server.
2. When we undeploy the project.

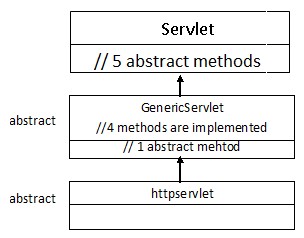
When we got an Exception illegal access specifiers the problem is because the servlet class is not a public class or the constructor is not a public. If we got all Exception classxxxservlet is not a servlet. The problem is because of the class doesn’t provide the implementation of servlet interface.

* **JDK is used by all the developers to develop the program and test it(JDK contains javac, java, javadoc, jar and etc).**
* **JRE is use to run any java based applications. Most of the time custormers install only JRE to run java based applications.**

When we got an Exception saying “unsupported class version” the problem is because of our java compiler version and JRE version used by the servers.

To resolve unsupported class version, we used to make that java compiler version and JRE version of server are same. **Init(), service(), destroy() are called as servlet life cycle methods**. In servlets the business logic will be written as part of service() any code which has to be Executed only one time when the server create servlet object is provide in the init() method.

Any code which has to be executed only once at the time of servlet object is removed. Then we provide the code in destroy() method. To simplify the development of servlets sun micro systems has given predefined classes. They are generic servlet, httpservlet. The following diagram shows the relationship or inheritance between these classes.



Different people says there are 3 ways are there to develop the servlets. According to sun micro system it is always recommended to use http servlet class to develop the servlet. If we develop the servlets based on http servlet we can remove the redundant code of init, destroy, getServletConfig, getServletInfo and etc.

**To develop a servlet based on httpservlet:**

public class FirstServlet extends HttpServlet

{

public void service(HttpServletRequest request, HttpServletResponse response)

{

System.out.println(“we are in service() method of FirstServlet”);

}

}

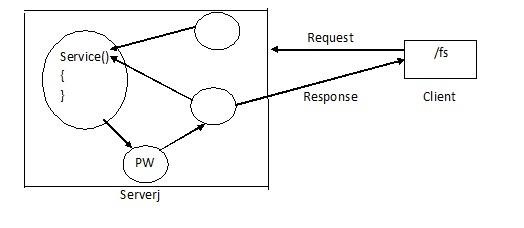
**Servlet Applications**

The servlets can be used to develop two types of applications. They are:

1. Static Servlet
2. Dynamic Servlet

As part of the servlets when we use System.out.println() it is executed on the server and display the message on the server. But for the web-based applications we have to send the output to client.

It is the responsibility of server to take the data from response object and convert it into httpResponseFormat and sent to client. To send the output to client we have to add the data to response object. To add the contents to response object we take the help of **print writer object**. The following diagram shows how output will be sent to client.



public class FirstServlet extends HttpServlet

{

public void service(HttpServletRequest request, HttpServletResponse response)

{

PrintWriter p = response.getWriter();

p.print(“welcome to”);

p.write(“servlets”);

}

}

The following is the servlet which send html contents to the client. To send the html context to the client we have to write html tags as part of write() method or print() method.

public class WelcomeServlet extends HttpServlet

{

public void service( ------- )

{

PrintWrite out = response.getWriter(); out.Println(“<html>”); out.Println(“<tittle>MyProc</title>”); out.Println(“<body>BDPS COMPUTERS</body>”); out.Println(“</head></html>”);

}

}

The above servlet is trying to send html content to the client. Every time when the client send the request it is displaying same output. These type of applications are called as **static web based** applications.

By using servlet also we can develop static applications as well as Dynamic applications. By using servlets it is not recommended to develop static web based applications. It’s always recommended to develop Dynamic web based applications. Instead of writing **printwriter** object we can use **“Servlet.OutputStream”**. It is recommended to use ServletOutputStream to send binary data.

EX. ServletOutputStream out = response.getOutputStream();

out.Println(“BDPS”);

The following are **disadvantages** of developing static servlets.

1. It takes huge amount of time for developing the project.
2. If you want to change the project we need to identify the code and do the necessary changes. Because of this reason we get maintenance related problems.
3. It occupies lot of resources of server. Because of all these reasons it’s not recommended to develop static servlets.

Instead of developing a servlet which send the static content recommended to develop html files to send the static content.

**Dynamic web-based application – display current date and time.**

public class DateAndTimeServlet Extends HttpServlet

{

public void Service( -------- )

{ Date d = new Date();

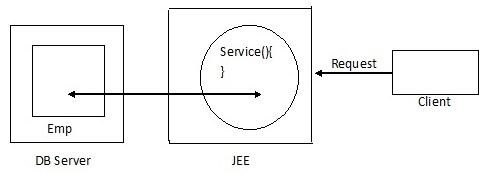
PrintWriter out = response.getWriter();

out.Println(d);

}

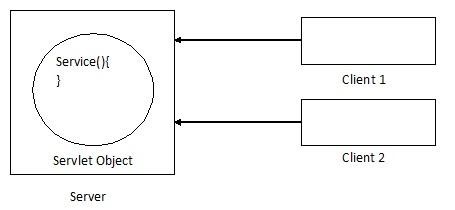
}

**Develop a servlet which retrieve the data from data base server and send the output to the client.**



When multiple clients try to access one servlet only one servlet object will be created.

This servlet object is shared between multiple clients.



When we got an error message like “HTTP status 405-HTTP method GET is not supported” we can understand that service() method is not available.

**The following is the servlet code which retrieve the data from data base server and display to the client**.

public class RetrieveRecordsServlet Extends HttpServlet

{

public void service( ------- )

{ try

{

Driver d = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(d);

String cs="jdbc:oracle:thin:@localhost:1521:xe";

Connection con = DriverManager.getConnection(cs,"subbu","lucky");

Statement stmt = con.createStatement();

ResultSet rs = stmt.ExecuteQuery(“select \* from product”);

PrintWriter out = response.getWriter();

}

}

When we deploy a project and if the server want to create the objects or serverwant to use the classes it will check in the following order.

1. Server checks for class files in classes folder.
2. If it is not available in classes folder server checks for project lib folder.
3. If it is not available in project lib folder, server check in server lib folder. If it is not available in these entire places server throws an Exception NoClassDefFound.

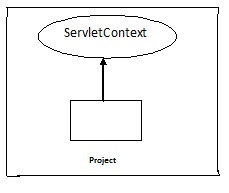
Develop a servlet and create a jar file and deploy the project and test it. In the above Retrieve Data Servlet we are hard coded driver class, url, username and password. Because of this reason the servlet able to communicate with only one Data base server. We would like to remove the hard coding from above servlet.

To secure the hard coding we are try to use **command line arguments and system properties**. But these will **not work in servlets**. This is because we can’t supply command line arguments or system properties to the server.

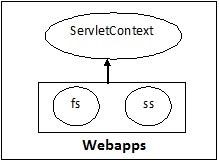
As the server is running the servlet application we can’t use command line arguments or system properties.

**ServletContext, ServletConfig objects are used to remove hard coding. These objects are managed by servers(creation and removing the objects).**

**ServletContext**: If it is the responsibility of server to create ServletContext object. It is the responsibility of server to remove ServletContext object. Server creates the **ServletContext object at the time of project is deployed. Per project we will have only one ServletContext object**.

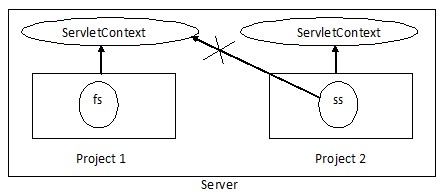


In a server we will have multiple ServletContext objects. This is based on number of projects available in the server. We can store the data into ServletContext object. The data stored in the ServletContext object can be accessible by all the resources of that project.



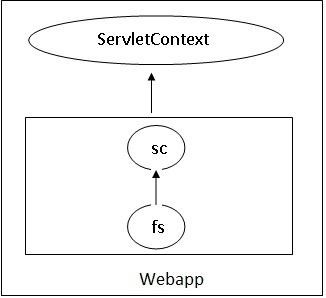
From the above diagram if we store the data into ServerContext object. FS and SS servlets can access the data.

If we store the data into one ServletContext object the other project can’t access the data from current project.

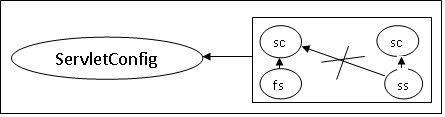


**ServletConfig**:

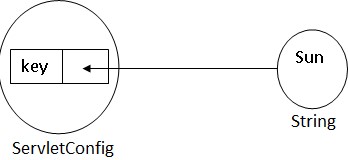
It is the responsibility of server to create ServletConfig object. It is the responsibility of server to remove ServletConfig object. Server is creating the ServletConfig object when ever the servlet object is created. Server removes the ServletConfig object when ever servlet object is removed.



In a server we will have N no.of ServletConfig objects. It’s all based on the no.of servlets. If we store the data into ServletConfig object only that servlet can access it other servlets can’t access the data from this ServletConfig object.



As a developer we can’t write the code to creae ServletContext object and ServletConfig object. To use these objects we have to provide the code to get the reference of these objects. To store the data into all object we required key values. This is useful to store the data as well as to retrieve the data from any object.



GenericServlet class provides the implementation of two interfaces servlet and ServletConfig. In this class there is a method gerServletConfig() which return ServletConfig object. class GenericServlet implements Servlet, ServletConfig{ ServlerConfig getServletConfig(){ return config;

}

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

}

**Procedure to use ServletConfig object**:

1. Store the data into ServletConfig object by using init parameter as shown below.

<web-app>

<servlet>

<servlet-name>ts</servlet-name>

<servlet-class>TestServlet</servlet-class>

<init-param>

<param-name>uname</param-name>

<param-value>bms</param-value>

</init-param>

</servlet>

-- - - - - - - - -

</web-app>

1. To read the data from ServletConfig we use a method getInitParameter(). public class TestServlet implements HttpServlet{ public void Service( ------ ){

ServletConfig config = getServletConfig();

System.out.println(config.getClass());

String uname = config.getInitParameter(“uname”);

String pwd = config.getInitParameter(“pwd”); PrintWriter out = response.getWriter(); out.println(uname); out.println(pwd);

}

}

If the data is not available in the ServletContext object with the given key if returns null value, to read the value from ServletConfig object we use the method gerInitParameter().

**Syntax**: String getInitParameter(key);

If the specified key is available in ServletConfig object we get the value in the form of string. If the key is not available we get null value. To specify multiple InitParameters we must use multiple InitParameters tags as show below.

<servlet>

<servlet-name>ts</servlet-name>

<servlet-class>TestServlet</servlet-class>

<init-param>

<param-name>uname</param-name>

<param-value>bms</param-value>

</init-param>

<init-param>

<param-value>pwd</param-value>

<param-value>abc</param-value>

</init-param>

</servlet>

We have developed another servlet to read the data from TestServlet config object. When we Excecutes this program we got the null value. If we doesn’t know the key values we can get the key names available in ServletConfig object by using getInitParameterNames().

The following is an example of get the key names and display to the client. public class TestServlet Extends HttpServlet{ public void Service( ----- ){

PrintWriter out = response.getWriter()

ServletConfig config = getServletConfig(); Enumeration e = config.getInitParameterNames(); while(e.hasMoreElements()){ out.println(e.nextElement() + “\n”);

}

}

}

Procedure to use ServletContext object.

1.To store the data into ServletContext object we use the tag <context-param> in web.xml file as shown below.

<web-app>

<context-param>

<param-name>uname</param-name>

<param-value>bms</param-value>

</context-param>

<context-param>

<param-name>pwd</param-name>

<param-value>abc</param-value>

</context-param>

<servlet></servlet>

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

</web-app>

1. Get the ServletContext objects. There are 2 ways are available to get the ServletContext object.
   1. By taking the help of ServletConfig object.
   2. We can call a method directly from GenericServlet class.

**Ex**: ServletConfig config = getServletConfig();

ServletContext application = config.getServletContext();

OR

ServletContext application = getServletContext();

1. To read the values from context object we use a method getInitParameter(). public class TestServlet Extends HttpServlet{ public void Service( ------- ){

PrintWriter out = response.getWriter();

ServletConfig config = getServletConfig();

ServletContext application = config.getServletContext();

String uname = application.getInitParameter(“uname”);

String pwd = application.getInitParameter(“pwd”); String url = application.getInitParameter(“url”); out.println(uname); out.println(pwd); out.println(url);

}

}

We have developed another servlet to read the data from ServletContext object. In this servlet instead of service() method we are use a method doGet() method.

public class AnotherServlet Extends HttpServlet{ public void doGet(HttpServletRequest request, HttpServletResponse response){

// code to get the data from the ServletContext and display to the client

}

}

When we configure the above servlet we are use the url pattern as \*.a. <servlet-mapping>

<Servlet-name>as</servlet-name>

<url-pattern>\*.a</url-pattern><servlet-mapping>

Develop a servlet to retrieve the data from DB server and display the output to the client. To achieve this we required to use weblogic connection pool and test it in both tomcat and weblogic servers. We would like to develop a servlet which send the following output.

First Line

Secod Line

To achieve the above requirement we have used println() as part of service() method. public void doGet( ---- ){

PrintWriter out = response.getWriter(); out.Println(“First Line”); out.println(“Second Line”);

}

When we run the above application we got different outputs in different browsers. As the browser understand any HTML tag as part of the response we have added <br>. out.Println(“First Line <br>”);

When the above line is Executed IE is able to process the <br> where as other browsers are displanying <br> directly to the client. To resovle the above problem we have to specify context type.

**Syntax**: void setContextType(String type);

Type attribute specify what type of context is sent by server to client. The following are some of the ContextTypes. text/html , text/xml, application/pdf, application/excel, ---------- etc.

**ContentType are also called as MIME types** to send the xml file to the client we have specify the ContextType as text/xml. To send the error to the client we use a method response.SendError() method.

**Ex**: response.SendError(555,”some problem”);

When an Exception is occurred, the servlet control goes to catch block. If we doesn’t write any code in the catch block use all see a blank server. It is not recommended to display the blank server to the user. Always we have to display an error message to the client. To achieve this we use a method response.SendError().

**Ex**: response.SendError(543,e.getMessage());

It is always recommended to use SendError() in catch block. Because of competation b/w the browser and every browser will release lot of features. The server like IE and Firefox use browser cache mechanism.

Now days all the modern browsers are associated with browser cache.

Request

Client/Response

Response

Data is stored

Server Browser Cache

When the server is sending output to the client by default the expire data to the context is set as none. If express is none browser will never send the second request to the server. This is because before sending the request to server browser check whether the data is available in the cache or not. If it is available and if it is not Expired browser will not send therequest to server.

**Ex**: public class RRServlet Extends HttpServlet{ public void Service( ----- ){ response.setDataHeader(“Expires”,System.currentTimeMillis()+7200000);

----------

}

}

To resolve all the browsers cache related problems we use a header “cache-control”. When by using this header we can make sure that browser doesn’t catch the data. The advantage of this approach is every time. When the client send the request the request go to server. **Ex**: public class public void response.setHeader(“Cache-Control”,”no-cache”);

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

}

}

When ever we develop any web-based application the first two lines of web-based application must be content type and setHeader.

**Form Based Applications**: Most of the web developer’s responsibility is to develop form based applications. **FormBased App’s are responsible to capture the data from the user and send the data to server**. To work with form based applications we should know how we call the resources available in server.

**Senario-1**: We can call the servlet **by typing the url in the browser**.

**Senario-2**: From the form we can call servlets **by using submit button**.

**Ex**: <form action = “/webapp/ws”>

<input type = “submit” value = “callsevlet”/> </form>

When the user click on submit button browser get action attribute value and converted it into HttpRequestFormat and send to server.

**Senario-3**: **The Anchor tag** following is a tag which is use to call a servelt from Anchor tag. <a href = “/webapp/ws”>call servlet </a>

**Using user defined buttons**: In HTML we can use **user defined buttons**. When we use these buttons we have to provide the javascript to submit the form. The following is an example.

<script>

Function doWork(){ window.document.form[0].action = “/webapp/ws”; window.document.form[0].submit(); }

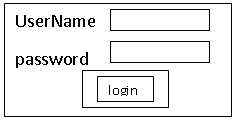
</script>

<form onclick = “doWork()”>

<input type = “button” value = “call servlet”/>

</form>

**Prototype**: **Prototype is a like a dummy project**. When we deliver a prototype to a customer by looking at the prototype customer understand now the final project looks like. Before the project coding starts we develop the prototype and deliver to customer. The following is the prototype of the form based application which we want to develop.



To implement the above requirement we have to develop two programs. They are:

1. Program to display the form.
2. Program to get the data from the client and perform the operations.

The following are the two approaches to implement the above requirement.

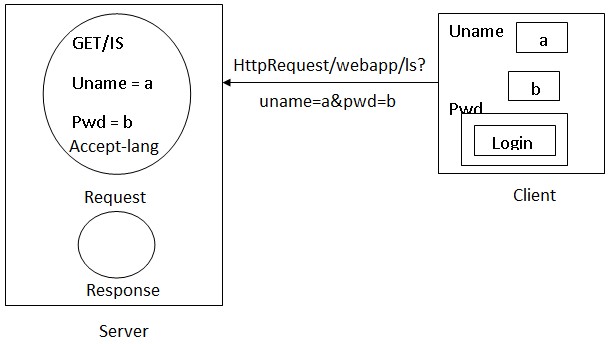
**Ap-1**:

1.DisplayFormServlet - DisplayForm

|  |  |
| --- | --- |
| 2.ProcessFormServlet | - get the data from the client and communicate |
| **Ap-2**: | with DB. |
| 1.login.html | -DisplayForm |
| 2.ProcessFormServlet | -get the data from the client and communicate |
|  | with DB. |

When the user files the form and click on submit button the following steps will be carried out.

1. Browser captures the data and stores the data into the variables which are allocated to the fields.
2. Browser gets action attribute value. Based on the data the url as shown below. /webapp/ls?uname=a&pwd=b
3. Now the browser creates httprequestformat and sends the request to server.
4. Now the server creates request and response objects. It is the responsibility of server to read all the data and place it in request object. When we supply this request object as input to servlet it can read all the details.



To implement the above requirement we are implemented the following two programs.

//Login.html

<html>

-------

<form action = “/webapp/ls”/> username:<input type = “text” name = “uname”/><br> password:<input type = “password” name = “pwd”/><br>

<input type = “submit” value = “Login”/><br>

</form>

</html>

The following is a servlet which captures the data form.

public class LoginServlet Extends HttpServlet{ public void Service( ------ ){

String uname = request.getParameter(“uname”);

String pwd = request.getParameter(“pwd”); String address = request.getParameter(“address”); response.getContentType(“text/html”); PrintWriter out = response.getWriter(); out.Println(uname+”<br>”); out.Println(pwd+”<br>”); out.Println(address+”<br>”);

} }

**To transfer the data between client and server**

**We can use either a get() method or post() method to transfer the data between the client and server.**

**get() – the data will be appended to url.**

**post() – the data will be appended to request body.**

**Most of the projects uses the post() to transfer secured data**.

The following is a syntax of request.getParameter

**String getParameter(fieldname);**

It returns a null value if the specified field name is not available.

A form based application to get the data from a drop down. //w3schools.com

As part of the servlets the methods like doGet( ) and doPost() methods. **doGet() – It will be Executed whenever we send a get request.**

**doPost() – It will be Executed whenever send the post request**.

**To support all the methods we use service()**.

We have implemented the form based application based on application. In this we are used a servlet to send the form output to the client. The following code demonstrate how do we read the values of the form without aware of field names.

Example:

**public class LoginServlet extends HttpServlet**

**{ public void Service( ----- )**

**{**

**PrintWriter out = response.getWriter();**

**Enumeration e = request.getParameterNames(); while(e.hasMoreElements())**

**{**

**String key =(String) e.nextElement();**

**out.Println(key);**

**out.Println(request.getParameter(key));**

**}**

**}**

**}**

The following code is used to get the map object which contains both keys ans values. public class LoginServlet Extends HttpServlet{ public void Service( ----- ){

PrintWriter out = response.getWriter();

Map map = request.getParameterMap();

Set keys = map.keySet(); Iterator i = key.iterator(); while(e.hasnext()){ object o = i.next(); String s = (String)o; out.println(s);

}

} }

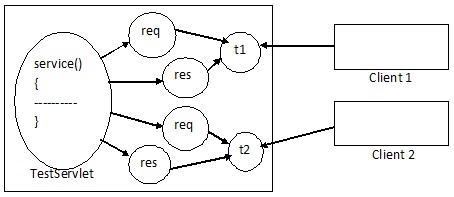
Note:

A web-based application will be accessed by any no.of clients at the same time. If we develop a servlet at the same time two different clients can send the request to the server.

**Every JEE server contains a thread pool. This thread pool program will be started whenever the server is started**. Whenever a client sends the request to server picses a thread from thread pool and creates request and response objects. These objects are handover to service().

After the server has finished its work server will removes request object and response object and it return thread to thread pool.

The following diagram shows how two different clients are working on one servlet objects.



We have observed that a different server follows different algoriths to pick a thread from thread pool.

**When we have a parameterized constructor, class.forName() will fail to create an object by throwing an Exception java.lang.InstantiationException**.

**Ex**:

**public class TomcatServer**

**{ public static void main(String args[])**

**{**

**Class c = class.forName(args[0]);**

**Object o = c.newInstance();**

**System.out.println(o.getClass());**

**}**

**}**

We can’t create a parameterized constructor as part of servlets. We can provide only default constructor. Because of this reason sun micro system is not recommending to use constructors in the servlet. As an alternative to the constructor sun micro systems has provided init().

**As part of the init() we have to provide the code which has to be Excecuted only once when the object is created**.

The following is the code which we found is every server. How they creating the objects and calling the methods. public class TomcatServer{

public static void main(String args[]){

Class c = class.forName(args[0]); //class Name is read from web.xml file

Object o = c.newInstance(); Servlet s = (Servlet)o; s.init();

s.service();

s.destroy();

}

}

**When the server create servlet objects always server call the init(). First server check whether init() available in our class or not. If not available server goes to super class and check for the methods.**

By using our own techniques we can make sure that server call parameterized constructor from default constructor.

**Ex**:

public class TestServlet implements Servlet{

public TestServlet(){ this(10);

System.out.println(“TestServlet object is created default”);

}

public TestServlet(int dummy){

System.out.println(“TestServlet object is created”);

}

}

**When the server is creating servlet object**:

**Scenario-1**: When the client sends the request to the server for the first time server is creating servlet object.

**Scenario-2**: We can use load-on-startup. We are configure<load-on-startup> in web.xml file as show below.

<web-app>

<servlet>

<servlet-name>a</servlet-name>

<servlet-class>FirstServlet</servlet-class>

<load-on-startup>5</load-on-startup>

</servlet>

--------

</web-app>

The <load-on-startup> takes a +ve value as input. This +ve value indicates the priority to create the servlet object.

**Ex**: <web-app>

<servlet>

<servlet-name>a</servlet-name>

<servlet-class>FirstServlet</servlet-class>

<load-on-startup>5</load-on-startup>

</servlet>

<servlet>

<servlet-name>b</servlet-name>

<servlet-class>SecondServlet</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

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

</web-app>

We have observed that for two different servlets we have given the same priority. When we deploy the project in TomcatServer he has created the object to both FirstServlet and SecondServlet. But TomcatServer is using an algorithm based on the name of the servlet (based on alphabetical order).

In case of weblogic server we have observed that it is creating servlet object based on the order which we have specified in development descriptor. We are used load.on.startup with the negative value as shown below.

<load-on-startup>-5</load-on-startup>

When we deploy the above project we have observed that server is not creating servlet object. We have observed that same servers are removing the old servlet object and creating new servlet object when ever the .class file is modified and send the request to server.

We have observed this behavior in weblogic server. In the Tomcat server even if we modify the .class file it is not pickingup the lastest .class file until and unless we restart the server.

**Conclusion**: According to SunMicro systems servlet specification it’s all the responsibility of server to create servlet object when ever it is needed. All the above scenarios we have observed in different servers.

When we are developing a project we follow the phases of software life cycle. That is requirements, analysis, design, coding and testing. In a project we will carryout multiple types of testing they are:

1. Unit Testing
2. Integeration Testing
3. System Testing (or) QA Testing
4. UAT Testing (User Acceptence Testing)

It is the responsibility of developer to perfrom Unit Testing and Integration Testing. The tester performs system level testing. It is the responsibility of end user to perform User Acceptence Testing. In a project we maintain different type of servers. They are:

1. Development Server
2. Integretion Server
3. QA server (Quality Assurance)
4. Production server

A development computer is used by a developer to develop the code and to perform unit testing. Integration server is used to perform only Integration Testing. QA server is used by the Tester to perform system level testing.

Production server is used by the client for the final deployments of the project as well as to perform UAT testing.

How do we deploye the project in Integration QA production servers?

To release web based applications we create war files and deploy the project.

Procedure to create a war file and deploy the project.

1. Goto a folder where the project is available(C:\work\webapps).
2. Use the following command to create a war file.

>jar-cvf lic.war

1. Copy pastes the war file inside server specific folder.

When we deploy a war file server creates a folder with the war file name and copy the contents of war file into that folder.

**Approach 3**: We can deploy the projects through server admin console. Every server will have an admin console through that we deploy the projects. In case of Tomcat Server we have an option WAR file and deploy the project.

In case of weblogic server as part of the domain structure we have an option to deploy to the project. In weblogic server also upload the WAR file and deploy the project. There are 3 methods available in servlet which act as servlet life cycle methods. They are:

* 1. init()
  2. service()
  3. destroy()

Generally we write any code which has to be executed only once when the servlet object is getting created inside init(). As part of the destroy() method we will provide the code which has to be executed only once at the time of removing the servlet object. As part of the service() we provide the code which has to be executed repeatedly. The following is an example of using init(), service() and destroy() to get the connection only once and use it.

public class RetrieveRecordsServlet Extends HttpServlet{ Connection con; public void init(ServletConfig config){ try{

DriverManager; con.DriverManager;

System.out.println(“connection got”); }catch(Exception e){ e.printStackTrace();

}

}

public void destroy(){ try{ con.close();

}catch(Exception e){ e.printStackTrace();

}

}

public void Service( -------- ){ try{

Statement stmt = con.

ResultSet rs = stmt.ExecuteQuery(“select \* from emp”); printWriter while(rs.next()){ out.println(rs.getString(1)); out.println(rs.getString(2)); out.println(rs.getString(3))+”<br>”);

}

}catch(Exception e){ e.printStackTrace();

}

}

}

It’s dangeros to use instance variable in servlets. It’s always recommended to remove instance variables from servlets. It’s recommended to use only local variables.

When we use instance variables in servlet we encounter lot of process. If multiple clients send the request to at the same time we are trying to develop the following servlet with out variable as instance variable.

public class TestServlet Extends HttpServlet{ PrintWriter out; public void service( ----- ){ out = response.getWriter(); out.Println(“First Line<br>”); try{

System.out.println(“sleeping”);

Thread.sleep(10000);

System.out.println(“okup”);

}catch(Exception e){

}

out.println(“Second Line”);

}

}

When we test the above servlet by sending multiple requests at the some time from different browsers we have observed unexpected outputs in the browsers. It’s always recommended to avoid using instance variables in servlet. It’s always recommended to use only local variables in servlets.

**Can we call destroy() from service()?** // yes we can call.

When we call the destroy() from the service() every time server executes service() it will call the destroy(). Because of this servlet will not remove servlet object.

public class TestServlet implements Servlet{

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

Public void service(ServletRequest request, ServletResponse response){

System.out.println(“we are in service()”);

Destroy();

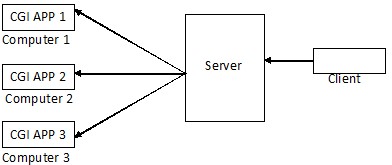
}

}

**Before sun micro system introduce servlets there is a technology CGI(Common Gate way Interface). To develop web based applications.**

There are so many draw backs are available in CGI. They are:

1. CGI applications are platform dependent.
2. The major disadvantage of CGI applications is they run in the computer which are associated with servers. Because of these reasons we find lot of performance issues.

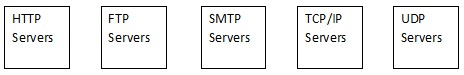


In case of CGI applications when the client send the request server starts a new process.

This process runs the CGI applications separatly.

Sun micro system has introduced servlet API to resolve all the problems of CGI applications. There are so many protocols available in the market. Some of them are HTTP, FTP, SMTP, TCP/IP ,UDP and etc.

Based on these protocols there are so many servers are develop.

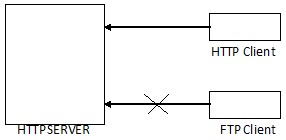


\*\***What is HTTP server?**

**A Server which are built based on HTTP protocol are called as HTTP Server**.

**Tomcat, weblogic, websphare, JBOSS**, ------ are called as HTTP Servers.

The initial idea of sun micro system is to run the servlets on all the servers. That is HTTP servers, FTP servers, SMTP servers, TCP/IP servers and UDP servers etc. An FTP client can’t communicate with HTTP servers.



**Sun micro system want to release multiple packages to support different protocols.**

They are:

Javax.servlet.http (all the classes and interfaces belongs to http protocol).

Javax.servlet.ftp

Javax.servlet.smtp

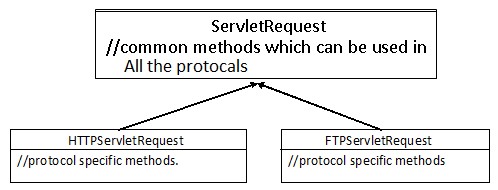
Javax.servlet.tcp/ip

Javax.servlet.udp

-------- etc.

All the common functionalities between all the protocols is placed in a package javax.servlet package.

We call all the classes and interfaces which are part of javax.servlet as protocol independent methods.



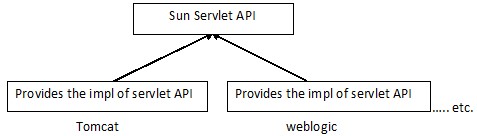
……… etc.

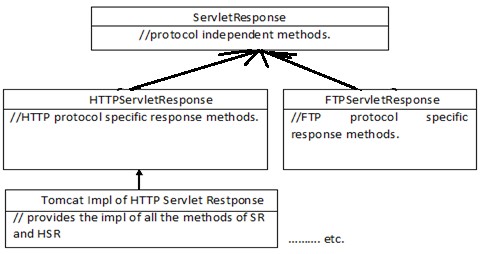
**\*Can we run the servlets on any server?**

**We can run the servlets on any server, if they provide the implementation to servlet.API**. If the server not provide the implementation to servlet.API ,we can’t run the servlets on that server .

We can’t run the servlets on HTTP web server, IIS server. This is because this server doesn’t provide the implementation to servlet.API.

If we want to run the servlets on any server, they have to provide the implementation to servlet.API.





**The meaning of creating request object and response objects are , creating an object to a class which provides the implementation of HTTPServletResponse and HTTPServletResponse.**

The names of these classes are varied from server to servers. Theoretically speaks we can deploy the servlets on any HTTPServers.

As there are there ways to develop the servlet which is the best way either servlet or generic servlet or HTTPServlet. It’s not at all recommended to develop the servlet based on servlet interface. This is because we have to provide lot of redundant code in the project.

The following is the internal code of generic servlet class. public abstract class GenericServlet implements Servlet{

ServletConfig config;

Public void init(ServletConfig config){ this.config = config;

}

public void destroy(){ config = null;

--------

}

The following Servlet is implements based on generic servlet to find method name sent by the client, url and useragent header value. public class TestServlet Extends GenericServlet{

public void service(ServletRequest request, ServletResponse response){

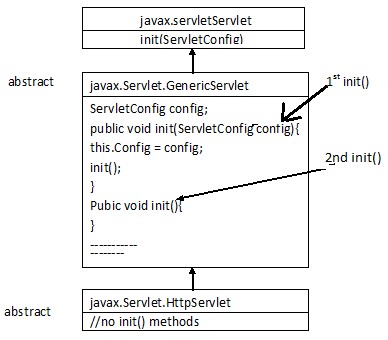
HTTPServletRequest req = (HTTPServletRequest)request;

HTTPServletResponse res = (HTTPServletReaponse)response; PrintWriter out = response.getWriter(); out.println(request.getMethod()); out.println(request.getRequestURL()); out.println(request.getProtocol()); out.println(request.getHeader(“user Agent”)); out.println(request.getHeader(“Accept-Language”));

}

}

The following diagram shows the over all picture of init() methods in servlet interface, GenericServlet and HttpServlet.



when we develop the servlets based on Httpservlet and if we would like to carry out and work only one time we have to use init().

It’s always recommended to use/overside second init() as part of servlets. The following example demonstrates the use of 2nd init().

public class TestServlet Extends HttpServlet{ public void init(){

System.out.println(“we are in 2nd init()”);

}

public void service(HTTPServletRequest request, HTTPServletResponse response){

}

}

When we deploy the above project, Server is creating TestServlet object and then server calls 1st init() method. Now the server opens TestServlet object and check whether the 1st init() is available or not. If it is available it will Execute, if it is not available in TestServlet, server check in HttpServlet, if it available execute it, if not available, server goto GenericServlet class and check for 1st init() method. As the 1st init() is available in GenericServlet it will initialize instance variable config and it will call 2nd init() method. Now the JVM open TestServlet class and check for 2nd init() method. As it is available and get Executed. It’s not recommended to use 1st init() as part of servlets which uses HttpServlet.

public class TestServlet Extends HttpServlet{ public void init(ServletConfig config){

System.out.println(“we are in 1st init() method”);

System.out.println(“doing some work for one time”);

}

public void service( ---- ){

System.out.println(“we are in service() method”);

ServletConfig config = getServletConfig();

System.out.println(config);

String drr = config.getInitParameter(“drr”);

System.out.println(drr);

}

}

When we run the above servlet program we got the NullPointerException this is because of the following reasons.

1. When the server has created TestServlet object it has initialize instance variable config of GenericServlet to Null value.
2. The server has called 1st init() method server has Executed 1st init() method of TestServlet. It has not executed 1st init() method of GenericServlet. Because of this reason it has not initialized config object of GenericServlet.
3. Because of this reason when we call getServletConfig() method returns Null value.
4. The best solution for the above problem is use 2nd init() method. Instead of using 2nd init() in the above TestServlet we have used 1st init() with an instance variable ServletConfig.

**public class TestServlet extends HttpServlet**

**{**

**ServletConfig config;**

**public void init(ServletConfig config)**

**{ this.config = config; }**

**public void service( HTTPRequest request, HTTPResponse response )**

**throws Exception**

**{ System.out.println(config); }**

**}**

The above code resolves the problem. But we have found lot of redundant code.

**Instead of using the 2nd init() method by using super.init(config).**

**public class TestServlet extends HttpServlet**

**{**

**public void init(ServletConfig config)**

**{ super.init(config); }**

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

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

**}**

Generally in a project all the project information is maintain in an xml file. As part of the projects when ever we deploy the project we should be able to create the object to the servlet and read the contents from configuration file and store it in JVM’s memory.

When ever we url deploy the project we should be able to remove the contents from JVM’s memory. The contents which is storing JVM’s memory should be able to accessable in service(). Our project should be able to read the configuration when ever we deploy the project.

public class ActionServlet Extends HttpServlet{ public void init(){

ServletConfig config = getServletConfig();

String filename = config.getInitParameter(“config”);

System.out.println(“Reading the contents from” +filename+ “and store in JVM”);

}

public void service(HttpServletRequest request, HttpServletResponse response){ System.out.println(“using the contents”);

}

public void destroy(){

System.out.println(“Remove the contents”);

}

}

To achieve our requirement we have use the following information in web.xml file.

<web-app>

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>ActionServlet</servlet-class>

<init-param>

<param-name>config</param-name>

<param-value>/WEB-INF/lms.config.xml</param-value>

</init-param>

<load-on-startup>2</load-on-startup>

</servlet>

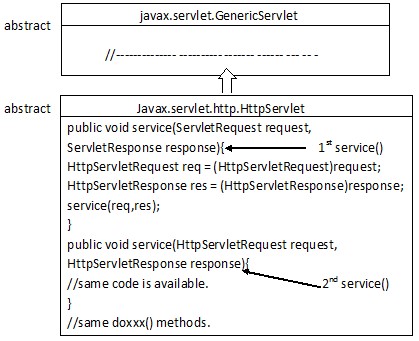
<servlet-mapping>

<servlet-name>action</servlet-name>

<url-param>\*.do</url-param>

</servlet-mapping></web-app>

The following is the same piece of code available as part of HttpServlet.



It’s always recommended to develop the servlets based on 2nd service() method. The advantage of this approach is we no need to typecast ServletRequest, ServletResponse into HttpServletRequest and HttpServletResponse. **Ex**: public class TestServlet Extends HttpServlet{

public void service(HttpServletRequest request, HttpServletResponse response){ System.out.println(“we are in 2nd service()”);

}

}

The following steps are carried out by the server when the client send the Request.

1. Server always call the 1st service() method. Now the server open TestServlet and check whether 1st service() is available or not. If it is not available it will goto HttpServlet and Execute the 1st service().
2. The internal code of 1st service() method typecast ServletRequest, ServletResponse into HttpServletRequest and HttpServletResponse and call 2nd service() method is available or not. If it is available it will execute.

While we are developing the project some times we should be able to process only either post request or get request. In this scenario we over ride doGet() or doPost() methods.

It is the responsibility of HttpServlet 2nd service() to call appropriate doxxx().

**Ex**:

public class LogicServlet Extends HttpServlet{ public void doGet(Http------ ){

PrintWriter out = response.getWriter();

System.out.println(“can’t process your request”);

}

public void doPost(Http------ ){

PrintWriter out = response.getWriter();

System.out.println(request.getParameter(“uname”));

System.out.println(request.getParameter(“pwd”));

}

}

When do we use 2nd service() or doGet() or doPost() methods?

If the application has to more for all the requests then we need to over ride 2nd service(). If the application has to work only for a specific method then we use either doGet() or doPost().

In a project some times we want to use doGet() and doPost() methods instead of 2nd service() and we would like to remove redundant code.

**Ex**:

public class LogicServlet Extends HttpServlet{ public void doGet(Http------ ){ process(request,response);

}

public void doPost(Http -----){ process(request,response);

}

public void process(HttpServletRequest request, HttpServletResponse response){ PrintWriter out = response.getWriter(); out.println(“from process() <br>”); out.println(request.getParameter(“uname”)); out.println(request.getParameter(“pwd”));

}

}

We can say there are ‘n’ number of ways are there to develop the servlets.

**Ex**: some body has developed a class which inherits the properties of HttpServlet and make the class as abstract.

public abstract class InetsolvServlet Extends HttpServlet{

}

Any body can develop the servlets based on the above InetsolvServlet. Because of this reason we say there are ‘n’ number of ways are there to develop the servlet. Generally in every project we will try to develop our own servlets. These servlets we will try to use our own methods.

**Ex**:

public class abstract BMSServlet Extends HttpServlet{ public void init(){

doOneTimeWork();

}

public void doOneTimeWork(){ public void service(HttpServletRequest request, HttpServletResponse response){ performWork(request,response);

}

abstract public void performWork(HttpServletRequest request, HttpServletResponse response)

}

The following is the FirstServlet Example based on BMSServlet. public class FirstServlet Extends BMSServlet{ public void performWork(HttpServletRequest request, HttpServletResponse response){ printWriter out = response.getWriter(); out.println(“we are in performwork()”); }

}

Most of the times we develop the servlet classes with package names while we are configuring the servlet in web.xml file which contain the packages we must specify absolute class name.

<servlet> //>javac –d .FirstServlet.java

<servlet-name>fs</servlet-name>

<servlet-class>info.inetsolv.FirstServlet</servlet-class>

</servlet> //pdf.coreservlet.com

Once if a develop a project for http protocol we can deploy the some project on https protocol also. We need to enable https option in the server (SSL option).

# \*\*\*JSP\*\*\*

JSP stands for java server pages and it is use to develop web based applications.

1. Generally we place all the JSP programs inside project folder. Every JSP program ends with an extension “.JSP “. We no need to configure JSP as part of web.xml file. We can send the request to the JSP’s directly from the browser.

**Ex**: <http://localhost:8000/webapp/one.jsp>

1. There are so many advantages are available with JSP’s when compared with servlet. The following are some of the advantages of JSP’s when compared with servlets.
   1. JSP’s improve the productivity of the developer (we can develop the JSP projects quickly when compared with servlets).
   2. We can develop JSP’s without the package of java.
   3. By using JSP’s we can sprat out business logic and presentation logic.
2. When we deploy the project and client has sent to request to JSP the server has converted JSP program into corresponding servlet.
3. An Apache guy has released HttpServer. This server can process HTML pages, images and JavaScript. This server can’t run servlet’s and JSP’s.

\*What is a Servlet Container?

Servlet container is a program which provides the implementation of Servlet-API. It is the responsibility of Servlet Container to create Servlet object and call the life cycle methods of servlet.

\*What is JSP container?

A JSP container is a program which provides the implementation of JSP-API. It is the responsibility of JSP container to run JSP programs. A Tomcat Server is a collection of HttpServer, Servlet Container and JSP container.

|  |
| --- |
| HTTP Server |
| Servlet Container |
| JSP Container |

// Tomcat Server (web server)

Today’s server are aggregated into two ways. They are:

1. Web Servers
2. Application Servers

Most of the web servers run servlets and JSP’s. Application server run servelts and JSP’s as well as they can run EJB application also.

|  |
| --- |
| HTTP Server |
| Servlet Container |
| JSP Container |
| EJB Container |
| Directory Server |
| Security Prelims |

Web logic Server Application Server

Tomcat is called as Web Server, Web logic, Web Sphere, JBoss are called as Application Servers.

An application server provides the complete implementation to JEE.API. Where as web servers can provide the partial implementation of JEE.API.

When we develop the JSP programs it is the responsibility of JSP container to convert JSP compiler to convert to run the JSP programs. The following steps will be carried out when a client send a request to server.

Client

Server

One.jsp

JSPC

Servlet code

javac

Servlet byte codes

/one.jsp

One.jsp

//one.jsp.class

The server will carry out all the above the steps when the client has sent the request to the JSP for the 1st time. When the client has sent the request 2nd time directly server execute the servlet object of the corresponding JSP.

Sun micro systems have inherited so many features of JSP programming language and develop JSP applications. Theoretically speaks we can develop JSP applications without writing java class.

According to the discussion as of new we have assented that we can develop the projects quickly and deliver to the customer easily based on JSP’s. No doubt servlets gives setter performance when compared with JSP’s.

Even though servlet’s give better performance when compared with JSP’s. We can’t measure the performance difference with human being eye. But when we consider the project cost we had a huge difference between servlet’s and JSP’s.

In most of the projects we use both servlets and JSP’s. If you want to execute huge amount to business logic we go with servlets. In the entire remaining scenario’s we use JSP’s.

It is the responsibility of the server to call the JSP compiler when the client sends the request to JSP. We can call JSP compiler manually. Generally we do this work to understand the internal code of generated servlet.

**Procedure to call web logic JSP compiler**:

1. Set the class path to multiple jar files by using “setDomainEnv.cmd” (JSPC program available inside web logic.jar).
2. Use the following command to generate the servlet.

>java weblogic.jspc -**keepgenerated** one.jsp

To place the source code(.java file)

By default web-logic JSP compiler removes .java programs. To retail the .java programs we use an option –keepgenerated. The following are the rules of JSP compiler which are available in JSP specifications.

1. Every JSP compiler must generate a servlet and place it in .java program. Because of the above rule different servers are following different naming conversions in writing the servlet name (.java file).

For Web-logic they are using the file name like –One.java

For Tomcat they are using One-jsp.java

1. Every generated servlet must be place inside a package. Different servers used different package names. (org.apache.jsp).
2. Every generated servlet must import 3 different packages.
   1. import javax.servlet.\*;
   2. import javax.servlet.http.\*;
   3. import javax.servlet.jsp.\*;
3. According to JSP specification every JSP compiler has to develop their own servlet. In that servlet we must over write 2nd init() and 2nd service() method. 2nd service() method must be call JSPService() method.

The following is the internal code of Tomcat JSP compiler. public abstract class HTTPJSPBase extends HTTPServlet{ public void init(){ public void service(HTTPServletRequest request, HTTPServletResponse response){ -JSPService(request, response);

}

abstract public void –JSPService(HTTPServletRequest request,

HTTPServletResponse response);

}

1. When the JSP compilers develop a Servlet for the corresponding JSP we must declare the class as final.
2. According to JSP specifications the parameters of –JSPService() method must be request and response only.

The following is the generated code of Tomcat JSP compiler.

package org.apache.jsp; import javax.servlet.\*; import javax.servlet.http.\*; import javax.servlet.jsp.\*; public final class One\_JSP extends HttpJspBase{ public void –JspService(HttpServletRequest request, HttpServletResponse response){

// standard local variables

}

}

When we observed the generated servlet of Tomcat and web-logic we have found extra methods. These methods are specific to the servers. These methods are not declare as part of JSP specification.

When the client send the request to server, server executes –JSPService() method.

In windows-7 OS if we are not able to see the generated servlet then we need to disable set of JSP elements. A JSP program contains set of JSP elements. The following are the elements of JSP. They are given below:

1. Template Text
2. Script let
3. JSP Expressions
4. JSP Declarations
5. JSP Directions
6. JSP Action Tags
7. JSP Custom Tags
8. EL Expressions(Expression Language)

\***Template Text**: This Element is used to send output to the client. We can write the Template TextElement.directly in the JSP without any special symbols.

**Ex**: First Line //TemplateText Line1 Second Line //TemplateText Line2

When we use Template Text the content will be placed inside out.Print() method. When we run the above JSP we got the following servlet.

public class Final one\_jsp extends HttpJSPBase{ public void –JSPService(…….){ out.Println(“First Line”); out.Println(“Second Line”);

}

}

The following is an example of sending a form to the client by using JSP’s.

<form>

User Name:<input type = “text” name = “uname”/><br>

Password:<input type = “Password” name = “pwd”/><br>

<input type = “submit” value = “Login”/>

</form> //One.jsp

**Scriplets**:

Scripts are used to write java code in JSP’s. The following is the syntax of Scriplet. <% //start scriplet

Java code

%> //ending script

**Ex**:

<%

int a = 10; System.out.println(a);

%> //One.JSP

When the JSP complier encounter a Scriplet directly placed inside –JSPService() method. When we are using Scriplets in the JSP. We must follow all the rules in the java. If we don’t follow all the rules of java the JSP compiler will fail in converting .java program into .class file and JSP compiler displays Errors in Browser.

Welcome to JSP’s

<%

int a = 10;

System.out.println(a);

%>

JSP compiler follows set of rules in converting JSP program into corresponding Servlet program.

**Implicit variables**:

In JSP’s we can use some variables without we declaring it. The variables which can be used in JSP’s without declaring we call them as implicit variables or implicit objects. The following are of implicit variables available in the JSP.

1. request
2. response
3. pageContext
4. session
5. application
6. config
7. out
8. page
9. Exception

**Response**: following is an example of using response object directly in the JSP.

<% response.setContextType(“text/xml”); out.println(“<student>”); out.println(“<sno>|</sno>”); out.println(“</student>”);

%>

We can send the error message directly to the client.

<%

O response.sendError(543,”abc”);

%>**Request**:

Request implicit variable can be used directly in the JSP script without we declaring it.

request ----------------- HttpServletRequest

<% out.println(request.getMethod()+”<br>”); out.println(request.getRequestURI()+”<br>”); out.println(request.getProtocol()+”<br>”); out.println(request.getHeader(“User Agent”)+”<br>”); out.println(request.getHeader(“Accept-language”)+”<br>”); %>

**Req**: Develop the following form based application by using JSP’s.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  |   User Name:  Password:   |  | | --- | | Login |   % |  | The following details are you have Entered.  User Name: system  Password: malli |

<

String uname = request.getParameter(uname);

String pwd = request.getParameter(pwd);

%>

The following details are you have entered <br> User Name: <%

out.println(uname);

%>

<br>

Password: <%

out.println(pwd);

%>

**Out**: We can out variable directly in the JSP scriplet without we declaring it.

out --JSPWriter

JSPWriter is a class which is available in javax.servlet.jsp package.

In servlet we are used printWriter to send the output to the client. In JSP we use JSPWriter.

\*What is the difference between PrintWriter and JSPWriter?

Every JSPWriter is associated with 8KB of internal Buffer. Where as PrintWriter doesn’t associated with any Buffer.

**Ex**: We can use both print and write() is an out variable.

<% int a = 10; int b = 20; out.print(“A value is:”+a); out.write(“B value is:”+b);

%>

**XML**:

xml is used to discuss the data while writing the xml file we must follow all the

rules of xml. We can read the content of xml file if it is a valid xml file. If it is invalid xml file we can’t read the content from xml.

1. If we want to develop a xml file we required either dtd file or xml sequence file. These documents contains all the tags which has to be used by xml developer.
2. To resolve the xml problems in servlets sun micro systems has given a dtd file. This is used by all the server developers as well as servlet developers to write xml file.
3. Sun micro systems has released servlets dfd document. All the servers have used this dfd and develop parser program. As developers we can follow the dfd file and create web.xml file.
4. Because of security reasons we can configure JSP’s into web.xml file.

When ever we write any xml file based on dfd document we must specify in web.xml file.

<?\_xml version = “1.0”?>

<! DocTYPE web-app PUBLIC “web.xml” “[http://java.sun.com/dfd/web-app\_2\_3.dtd”](http://java.sun.com/dfd/web-app_2_3.dtd)><web-app>

</web-app> //web.xml

To resolve all the security related problems in JSP’s we can configure the JSP in web.xml file. The following is the configuration we use to configure a JSP into web.xml file.

<web-app>

<servlet>

<servlet-name>a</servlet-name>

<jsp-file>/WEB-INF/one.jsp</jst-file>

<servlet>

<servlet-mapping>

<servlet-name>a</servlet-name>

<url-pattern>/a</url-pattern>

<servlet-mapping>

</web-app>

**Note**: when we are writing <jsp-file> the path of the JSP must start with “/” .

1. If we don’t want to allow any body to access the JSP directly recommended to place inside WEB-INF folder.
2. We can use <load-on-startup> as part of web.xml file for a jsp.

<servlet>

<servlet-name>a</servlet-name>

<jsp-file>/WEB-INF/jsp/one.jsp</jsp-file>

<load-on-startup>4</load-on-startup></servlet>

1. To a JSP when we use load-on-startup and when we deploy the project immediately server will convert jsp into corresponding servlet and the servlet.class will be loaded into JVM’s memory and it creates the servlet object.

**Config**: Config variable is belongs to servletConfig data type. By using servletConfig we can remove the hard coding.

Config---- servletConfig

<%

String drv = config.getInitParameter(“drv”); out.print(drv);

%> servletConfig is an object used by servlet container to pass the information during initialization.

Once if we get the ServletConfig object, we can find the name of the servlet which is configured in web.xml file by using a method getServletName() method. **Ex**:

<%

String name = config.getServletName(); oput.println(name);

%>

As part of the url pattern if it end with .jsp, jsp compiler directly processes the jsp program.

**Application**: application ----- ServletContext

1. Application implicit variable can be used directly in the jsp.
2. ServletContext object contains set of methods which are used to interact with ServletContainer. By using these methods we can find the information about ServletContainer.
3. When ever sun micro system releases a new version of API. It is the responsibility of server vender to provide the implementation to the latest API. For example when servlet 2.5 is released Tomcat guys has provided the implementation and integrated it in Tomcat 6.0 the latest version of servlet API is 3.0 this version is integrated in Tomcat 7.0.

\*What is the different between servlet-API 2.5 and servlet-API 3.0? The servlet-API 3.0 supports processing annotations.

**Session**:

We can use session implicit variable directly in the jsp. Session implicit variable is holding Http Session object. Session ---- HttpSession **Page**:

We can use page implicit variable directly in the JSP.

Page ----- Object

Page implicit variable hold the currently executes Servlet object for the corresponding JSP.

**Exception**:

The implicit variable can be used only in Error pages. When we try to use this variable in a JSP which is not an Error page we get an Error message. Exception ----- Throwable **PageContext**:

We can use PageContext object directly in the jsp.

PageContext --- PageContext

1. Whenever -jspService() method is Executing it creates PageContext object. Whenever –jspService() method Execution is completed it removes PageContext object.
2. By using PageContext implicit variable we can get any other implicit variable object.

**Ex**: application = PageContext.getServletContext(); In the jsp program also we can use comments.

1. We can follow HTML style of comments in jsp.

**Ex**: <!- - -

<%

out.println(“we are in scriplet”);

%>

--- >

1. jsp style of writing comments.

<% ----

code

----%>**JSP Declarations**:

JSP declarations are used to create instance variables, instance methods and static variables and static methods.

**Syntax**:

<%!

// instance variables

// static variables

// instance methods

// static methods

%>

When we run the JSP program which is having JSP declarations the JSP compiler directly place the contents inside the class.

**Ex**:

<%!

int a;

public void method(){

}

%> // One.jsp (JSPC)

public final class one\_jsp Extends HttpJSPBase{

int a; public void method(){

}

public void \_JSPService(------){

// implicit variables }

} // One.servlet

If we write the JSP declaration with set of methods we need to call the methods from \_JSPService() method.

**Ex**: <%!

public void methodOne(){

System.out.println(“we are in methodOne()”);

}

%>

<%

methodOne();

%>

The implicit variables of JSP can’t be accusable in JSP declarations. **Ex**:

<%!

Public void methodOne(){

out.println(“we are in methodOne()”);

}

%>

According to sun micro system the implicit variables can’t be accusable in JSP declarations. This is because the local variables of one() method can’t be accusable in another method.

As a java programmer we can follow our own techniques to use the local variables of one() method in another method.

**Ex**:

<%!

JSPWriter out; public void methodOne(){

out.println(“we are in methodOne()”);

}

%>

<%

this.out = out; methodOne();

%>

The following are three life cycle methods of JSP.

1. JSPInit()
2. \_jspService()
3. jspDestroy()

**Ex**:

<%

public void JSPInit(){

System.out.println(“we are in JSPInit()”);

}

public void jspDestroy(){

System.out.println(“we are in jspDestroy()”);

}

%>

<%

System.out.println(“we are in \_jspService()”);

%>

When the corresponding servlet object for the jsp is created, server call JSPInit() method. Every time the client send the request it Execute \_jspService() method. When the project is un-deploy server Executes jspDestroy() method.

In case of jsp when ever we modify the jsp and send the request. The Existing servlet object will be removed and create new servlet object.

When the server create servlet object for a corresponding jsp program?

**Scenario**:

1. When the client send the request to jsp for the 1st time server create the 1st servlet object and call the life cycle method.
2. When we modify the jsp and when the client send the request to the server if already servlet object is available it removes the old servlet object and creates new servlet object [it will call jspDestroy()].

**JSP Expressions**: jsp Expressions simplifies the use of java Expressions in jsp.

Getting the value from variable and display to the client is called as java Expressions. **Ex**:

<%

int a = 10; out.println(a);

%>

Performing arithmetic, relational, logical operations on a variable one also called as java Expression.

**Ex**:

<%

int a = 10; int b = 20; out.println(a+b);

%>

In a project we use some supporting classes every supporting class must be placed inside a package other wise server will not recognize the supporting classes.

Displaying an object to the client is also considered as java Expressions.

Calling the methods an object and send the output in the client is also called as java Expressions.

**Ex**:

<%

Infodreamsoft.Emp e = new infodreamsoft.Emp(); out.println(e); or (e.toString()); or (e.Eno());

%>

**Syntax**:

<% = javaExpressions %>

The following is an Example of jsp Expressions.

<%

int a = 10; int b = 20;

%>

<% = a + b %> or <% = a%>

The following is the code how jsp compiler convert jsp Expressions. <% = a + b%>

JSPC

Out.println(a + b);

<% info.inetsolv.Emp e = new info.inetsolv.Emp(); e.setEno(“1”);

e.setEName(“Raju”);

%>

<% = e%>

<% = e.getEno() %><br>

The following jsp get the data from the form and display to the client.

|  |
| --- |
| The following details Eno….  Ename….  Salary…. |

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | |  | |  |   Eno  Name  Salary |

Eno:<% = request.getParameter(“Eno”)%><br> Ename:<% = request.getParameter(“Ename”)%><br>**JSP Directive**:

JSP directive is an instruction given to JSP compiler. Based on the instruction which we are given the JSP compiler converts JSP code into corresponding java code. There are 3 directives are available. They are:

1. page directive
2. include directive
3. taglib directives

**syn:**

<%@ directive attribute1 = “ “ attribute2 = “ “%> The following are the attributes of page directive.

1. language
2. import
3. info
4. contentType
5. buffer
6. autoFlash
7. isErrorPage

**Page Directives :**

JSP page directive is used to define the properties applying the JSP page, such as the size of the allocated buffer, imported packages, and classes/interfaces, defining what type of page it is, etc. The syntax of JSP page directive is as follows:

<%@page attribute = "value"%>

**Differentproperties/attributes :**  
The following are the different properties that can be defined using page directive :

**import**: This tells the container what packages/classes are needed to be imported into the program.  
**Syntax**:

<%@page import = "value"%>

**contentType**:

This defines the format of data that is being exchanged between the client and the server. It does the same thing as the setContentType method in servlet used

**yntax**:

<%@page contentType="value"%>

**info**: Defines the string which can be printed using the ‘getServletInfo()’ method.

**buffer**: Defines the size of the buffer that is allocated for handling the JSP page. The size is defined in Kilo Bytes.

**language**: Defines the scripting language used in the page. By default, this attribute contains the value ‘java’.

**isELIgnored**: This attribute tells if the page supports expression language. By default, it is set to false. If set to true, it will disable expression language.

**errorPage**: Defines which page to redirect to, in case the current page encounters an exception.

**Include directive :**

JSP include directive is used to include other files into the current jsp page. These files can be html files, other sp files etc. The advantage of using an include directive is that it allows code re-usability.  
The syntax of an include directive is as follows:

**<%@include file = "file location"%>**

**Taglib Directive :**

The taglib directive is used to mention the library whose custom-defined tags are being used in the JSP page. It’s major application is JSTL(JSP standard tag library).

<%@taglib uri = "library url" prefix="the prefix to

identify the tags of this library with"%>