

Advance Java

Class notes

By

Mr . Shekhar sir



**SRI RAGHAVENDRA
XEROX**

Software languages Material Available

Beside Bangalore Iyyager Bakery .Opp. CDAC, Balkampet Road,
Ameerpet,Hyd

9951596199

- (1) In java we can develop two types of Applications.
- (i) Standalone Application
 - (ii) Distributed Application
- (2) Distributed Applications are divided into two types-
- (i) Web Application
 - (ii) Remote Application
- (3) Sun Microsystems divided java into 3 parts
- (i) JAVASE
 - (ii) JAVAEE
 - (iii) JAVAME
- (4) JAVASE is used to develop standalone applications.
- (5) JAVAEE is used to develop distributed application.
- (6) A stand alone application means, it is an application installed on one machine and it is non-shareable across network.

For ex - Anti-virus installed on one machine can't be shareable to another machines.

Another example is Eclipse IDE installed on one machine is not shareable to other machines.

(7) A distributed application means, it is an application installed on server machine and shareable from client machines.

for ex - Flipkart application installed on server machine and it is accessible to all the clients in network.

(8) A stand-alone application doesn't follow client/server model, but distributed application follow client server model.

* Q) Why you selected java as your career?

Ans - Many tools and technologies in realtime are developed based on java programming. So, it will be easy to migrate from one technology to another if java is career. so, I selected java.

4 Aug 15

(9) A web application is a server side application. It means, Application runs on server and it is accessible for all clients across network through browser.

(10) A remote application is an application, divided into two parts called client and server programs. server program run on server system and client program runs on multiple client systems.

For ex - ATM applications are remote applications. server program runs on bank system and client program run on multiple ATM centers.

(9) what is difference b/w standalone and web Application ?

Ans 1) A standalone application runs on local machine but a web application runs on server machine.

2) A stand alone application is not accessible to all the clients in network but a web application is accessible to all the clients in the network.

(9) what is the difference b/w a web Application and Remote application ?

Ans In web Application , client is a browser and in remote application client is a java program.

(11) Web Applications are of two types -

(i) static

(ii) dynamic

(i) static applications will concentrate on presentation

(ii) dynamic applications will concentrate on service.

(12) • To develop static applications, we need only client side technologies like HTML, CSS, javascript.

• To develop dynamic applications, we need client side and server side technologies are required.

(13) Some server side technologies are -

(i) CGI (common gateway Interface)

(ii) servlet

(iii) ASP

(iv) JSP

(v) PHP

(vi) cold Fusion etc

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(8) What we can do with advance java technologies?

Ans - We can develop dynamic web Applications (websites) with database support using JDBC, servlet and JSP technologies.

5 AUG 15

Q) Why we need a class?

- Ans- (i) Java Programming language has already provided predefined datatypes but they are not enough by creating the projects.
- (ii) For example, we are creating a project airlines reservation. In this project we need a passenger datatype but in java there is no predefined datatype called passenger.
- (iii) So, we need to create our own datatype called Passenger in the project. For this, we use class.
- (iv) Finally, class is a keyword used for creating user defined datatypes.

Q) Why we need a object?

- Ans- (i) In C language, when we run the program then a program loaded into memory, then executed and then terminated.
- (ii) To run the same program again then we need to load the program into memory again.
- (iii) In OOPL, we can load a class into memory for one time and we can execute it for multiple times, by using objects.

6 Aug 15

Q) Why we need interfaces in Java?

Ans - 1) In client/server Applications of Java, a client application calls methods (services) of server application.

- 2) If a client Application wants to call the methods of server then a client application should know what are the methods exist in server application.
- 3) A server Application tells the client application about what are the methods in server by creating an interface.
- 4) Interfaces provides/ establishes communication b/w client and server application.
- 5) In real time applications, one module calls methods of another module. A communication b/w the two modules are established by using interfaces.

Q) Why we need abstract classes?

Ans - 1) In a Java project, multiple classes can have some methods and some methods can have same logic.

- 2) In order to get reusability, we create a super class and in that we define common logic methods and we make uncommon logic methods as abstract. That super class is an abstract class.

```
abstract class Burger
```

```
    void packing()
```

```
{ == }
```

```
}
```

```
abstract void calculatePrice();
```

```
{ . . . }
```

```
class vegBurger extends Burger
```

```
    void calculatePrice()
```

```
{ == }
```

```
}
```

```
{ }
```

```
class chickenBurger extends Burger
```

```
    void calculatePrice()
```

```
{ == }
```

```
}
```

```
{ }
```

Q) What is the abstract method and the concrete method?

Ans - An abstract method means it contains only header without body but a concrete method is if it contains body also.

Note - 1) In java, if a class contains atleast one abstract method then that class must be an abstract class. But, an abstract class may or may not contain abstract methods.

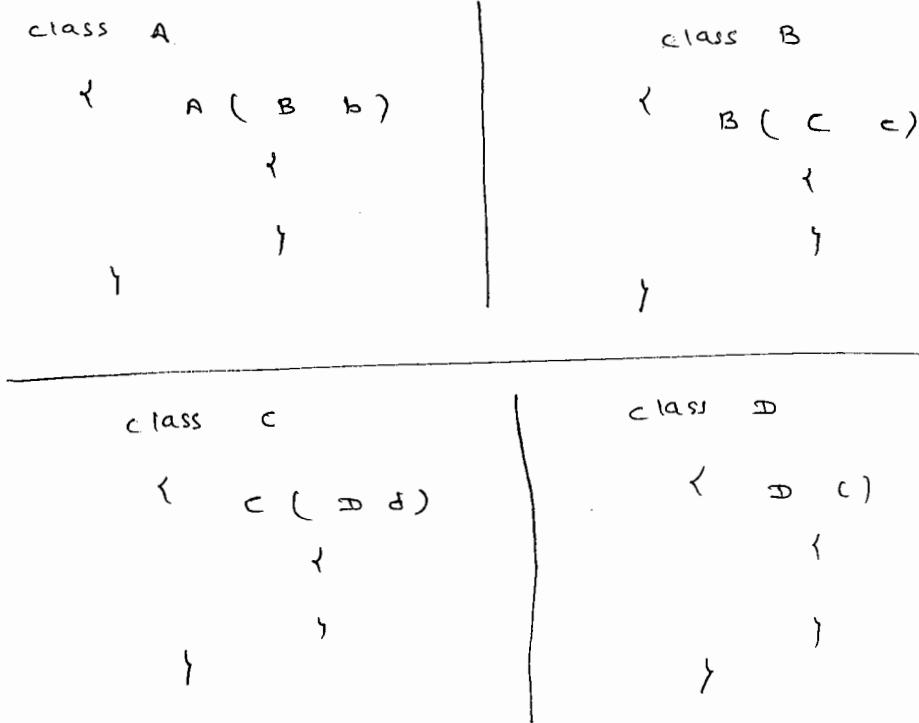
2) In java, we can't create objects for abstract classes and interfaces.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

7 AUG 15 Factory method

- 1) A factory method means, it is a method which produces an object of a class.
- 2) In Java, it is not always easy to create object of a class directly. sometimes we need complex logic to create an object.

For ex —



- 3) If we want to create object of class A then first we need to create D object then C object, then B object and finally A object.

```
D d = new D();  
C c = new C(d);  
B b = new B(c);  
A a = new A(b);
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Avyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 4) If we want to execute A object in multiple classes of the project then we need to repeat the complex logic again and again.
- 5) To solve the above problem we create a factory method in a class and we write the complex logic in that method and then we return the object.

```

public class demo
{
    public A getInstance()
    {
        D d = new D();
        C c = new C(d);
        B b = new B(c);
        A a = new A(b);
        return a;
    }
}

```

factory method

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

- 6) In a project, if we want "a" object ten times we need to call getInstance() for 10 times.

7) Types of Factory methods →

- (i) Instance factory method
- (ii) Static factory method

- (i) Instance factory methods are called by using object name.
- (ii) static factory methods are called by using class name.
- 3) A factory method can return either same class object or any other class object.

Ex -

```

public class Demo
{
    public void m1() // Not a factory method
    {
        System.out.println("m1");
    }

    public int m2() // not a factory
    {
        System.out.println("m2");
        return 10;
    }

    public A m3() // instance factory
    {
        System.out.println("m3");
        return new A();
    }

    public static B m4() // static factory
    {
        System.out.println("m4");
        return new B();
    }

    public Demo m5() // instance factory
    {
        System.out.println("m5");
        return new Demo();
    }

    public static Demo m6() // instance
    {
        System.out.println("m6");
        return new Demo();
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Q) If a method return type is an instance then what that method returns?

Ans - Object of its implementation class

Ex -

```
public interface I1  
{  
}  
  
public class S implements I1  
{  
}  
  
public class Demo  
{  
    public I1 getInstance() // factory method  
    {  
        S s = new S();  
        return s;  
    }  
}
```

10 Aug 15

Q) If a method return type is abstract class then what it returns?

Ans - Object of its subclass

Ex -

```
abstract class X  
{  
}  
  
class Y extends X  
{  
}  
  
public class Demo  
{  
    public X m1() // factory method  
    {  
        Y y = new Y();  
        return y;  
    }  
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q) If return type of a method is class
then what that method returns?

Ans - It returns either an object of the same
class or an object of its sub class.

Ex -

```
class A
```

```
{
```

```
}
```

```
class B extends A
```

```
{
```

```
}
```

```
public class Demo
```

```
{
```

```
    public A m1() // factory method
```

```
{
```

```
    A a = new A();
```

```
    return a;
```

```
(or)
```

```
    B b = new B();
```

```
    return b;
```

```
}
```

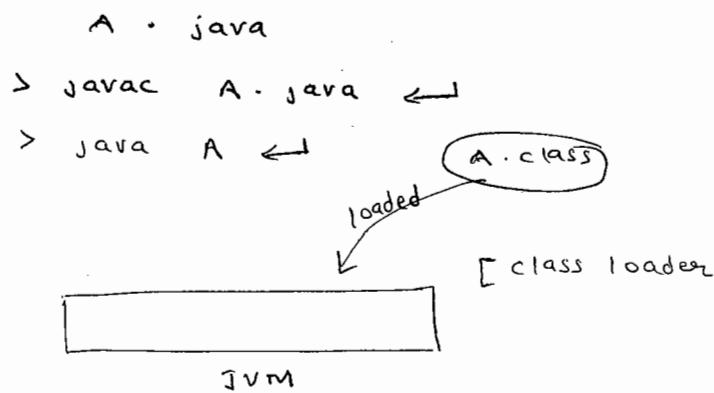
```
/
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

class loading in java

- 1) In java, every class can be executed in JVM only.
- 2) In java, there is a predefined component called class loader, it loads a class into JVM.
- (3) Loading a class means transferring a .class file from secondary memory to main memory (JVM).

FOR EX -



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

In the above diagram java is a tool given by jdk and it calls a class loader and that class loader loads A.class into JVM.

- (4) When we are writing java programs, if we want to load either an userdefined class or a predefined class into JVM then we need to call `forName()`.

`forName()` is a static method provided in "Class"

- (5) In java Class is a keyword, class is a class name.
- (6) Static methods can be called directly by using className, so, the syntax is -
- `Class.forName(" fully qualified className");`
- * (7) In java, a class can be loaded into JVM for maximum one time.
- (8) While loading, a class loader first verifies whether a class is already loaded into JVM or not. If not loaded then it will be loaded and if already loaded then it will be skipped.

LINDRA ALKU
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

For ex -

```
Class.forName(" Oracle.jdbc.driver. OracleDriver");
Class.forName(" Oracle.jdbc.driver. OracleDriver");
Class.forName(" sun.jdbc.odbc.JdbcOdbcDriver");
```

In the above, class loader loads OracleDriver for one time, because in the second line OracleDriver is already loaded so the second line will be skipped.

11 AUG 15

(9) In jdk we have source code of java API and a class file of java API.

(10) source code of java API exist in

jdk\src.zip
source code
of java API

(11) classes of java API exist in

jdk\jre\lib\rt.jar
classes of
java api

(12) A class can be loaded into JVM for only once. We can test whether a class is loaded for only once or not by writing static block in the class.

For example -

```
//A.java
class A
{
    static
    {
        S.O.P. (" I am static block");
    }
}
```

```
// Main.java
class Main
```

```
{    psvm( ) throws Exception
```

```
    {
        Class.forName(" A");
    }
```

```
    Class.forName(" A");
}
```

```
    Class.forName(" A");
}
```

}

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
C:\> javac A.java  
C:\> javac Main.java  
C:\> java Main
```

I am static Block

(13) When a class is loaded into JVM, JVM creates a class object and it will store metadata of a loaded class.

(14) `forName()` returns class object created by JVM. So, `forName()` is a static factory method.

```
Class c = Class.forName("A");  
          ↑           ↓  
It contains      static Factory()  
metadata of A
```

(15) `forName()` throws "ClassNotFoundException".

(16) At the time of loading a class, if that class is not a part of java API then "ClassNotFoundException" is thrown.

(17) "ClassNotFoundException" is a checked exception.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

12 AUG 15 PATH and CLASSPATH

- 1) When we run a batch file (.bat) or executable file (.exe) or command file (.cmd), if operating system doesn't know the location of the file then OS generates not recognized problem.
- 2) In order to solve the above problem we need to add the location of the file to path variable.
- 3) Before executing a file, OS reads the value from path variable and then it will run that file.
- 4) For example - when we are running javac command, we will get javac is not recognized problem. To solve this problem we are adding the location of javac to path variable.

For ex -

> javac A.java ↴

'javac is not recognized as internal
or external command'

> set PATH = c:\program files\java\jdk1.7.0\bin ↴

> javac A.java ↴

compiled successfully

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- ⇒ We can set path and classpath either temporarily or permanently.
- ⇒ When we are loading a class, which is not a part of java API then ClassNotFoundException is thrown.
- ⇒ To solve the exception, we need to add the location of a jar file to the classpath variable.
- ⇒ If we are importing a package, which is not a part of java API then at compile time, java compiler generates error message. To solve this error, we need to add the location of a jar file to classpath variable.

Example -

```

class Main
{
    public static void main (String args[])
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}

> javac Main.java ( success )
> java Main ↵
      ClassNotFoundException

> set CLASSPATH = c:\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar ↵
> java Main ↵
      Executed
  
```

SRI RAHMAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

EX2

// A.java

```

import java.util.*;
import oracle.jdbc.*;
class A
{
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

> javac A.java ←

error: package oracle.jdbc doesn't exist

> set CLASSPATH = c:\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar

> javac A.java ←

(successfully compiled)

- Q) path is related to OS and classpath is related to java.

Jar File

- 1) In java, we compress multiple java classes into a single file called jar file.
- 2) A jar file is similar to zip file.
- 3) In java, jar files are created, to release java related softwares easily.
- 4) To create a jar file, we use jar tool, given by jdk.

c:\sathya> jar cvf test.jar *.* ←

create cvf file
 ↓ ↓
 verbose
 (optional)

All the classes of sathya is loaded in test.jar

13/Aug/15

JDBC

Q) Why we need a database?

Ans- i) we can store the data in the following three-

- (i) Variable
- (ii) File
- (iii) Database

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

2) Variable

- i) we can store the data in a local variable or in an instant variable.
- ii) The data stored in a local variable will be destroyed/ removed when a control comes out of a method.
- iii) The data stored in an instance variable will be destroyed when object is removed from the JVM.
- iv) The data stored in variables is a temporary data.

File

- 1) we use flat files (not containing any language extension e.g. abc.txt) to store the data permanently.
- 2) A flat file is nothing but, it is a file which is not related to a particular programming language.

- 3) For ex → A .java file is not a flat file because it is related to java. But, a .txt file is a flat file because it is not related to any language.
- 4) In terminology the data which is stored permanently is called persistent data.
- 5) with files, we have the following drawbacks –
- i) Data Redundancy
 - ii) Redundancy refers duplication of data.
 - iii) For ex, if a customer is opened a saving account and a current account in a bank then that customer information will be duplicated in saving account file and also a current account file. It means there is a data redundancy.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. GAC, Balkampet Road,
Ameerpet, Hyderabad.

Data inconsistency -

Inconsistency refers to not-reliable data.

For ex, if a customer address is changed in saving account file but not changed in current account file then data inconsistency occurs.

(3) no constraint support -

when we are storing the data into file we can't apply any constraints (condition) of the file.

for ex - when we are storing employees record, to restrict duplicate entries, we need to apply a primary key constraint but in file it is not possible.

(4) Security problem -

A file can be opened and it can be modified by any body. so, there is no security for the data stored in a file.

(5) Non-transactional -

A file is not a transactional resource. It means a file can't be used in making transaction.

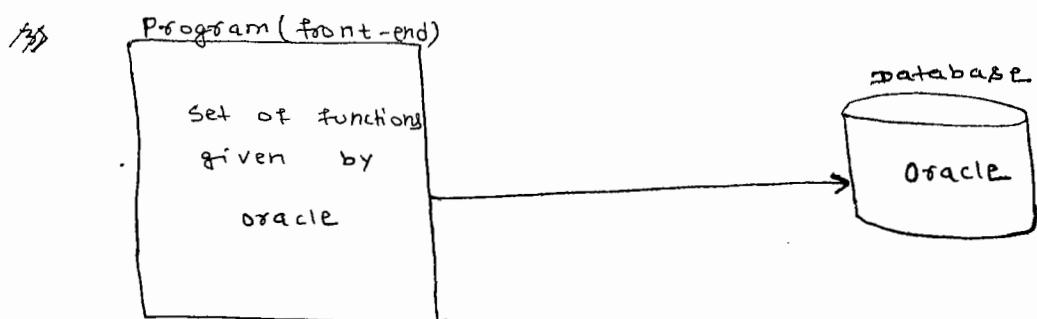
To overcome the above drawbacks of files, database is introduced.

SRI KRISHNENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

14-AUG-15

Q) Why JDBC technology introduced?

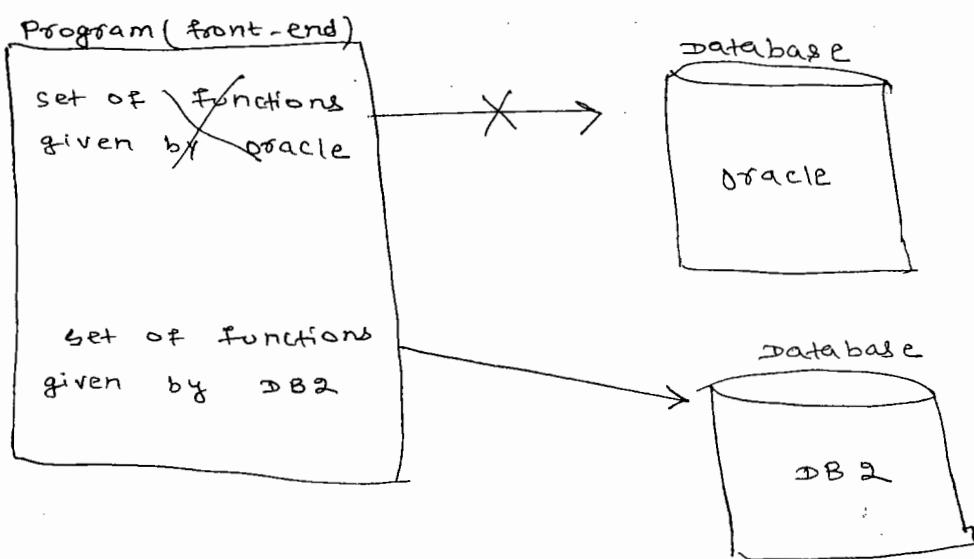
- Ans- 1) In early days, a program (front-end) connected with a database (back-end) by using a set of functions given by database vendor.
- 2) For example, if a program wants to connect with Oracle database then that program used a set of functions given by Oracle Corporation.



- 3) A set of functions given by a database vendor is called Vendor API.
- 4) Then, a problem identified is, if a program wants to connect with another database then we need to modify the program, by replacing previous vendor API with new database vendor API.

SRI RAJAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

For example, if the above program wants to connect with another database DB2 then we need to change the program, by removing Oracle given functions and by adding DB2 (IBM) given functions.



5) In order to make a program as a database independent, microsoft with simba technologies formed a group (ODBC group)

ODBC - Open Database Connectivity

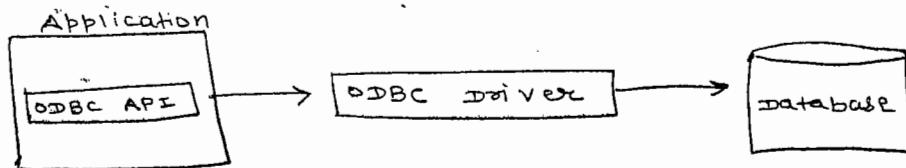
17 Aug 15

6) ODBC group released ODBC API.

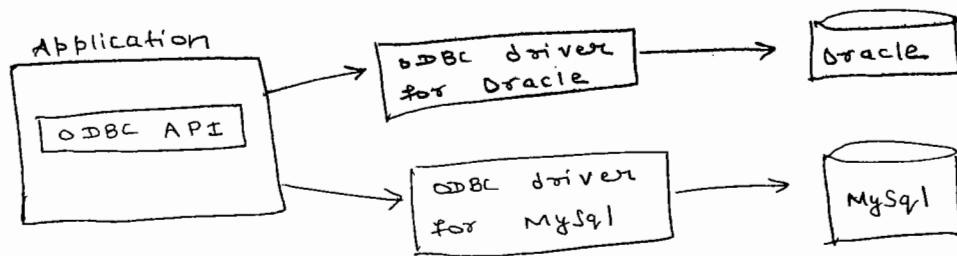
- ODBC API consists a group of common functions created in C language, for making an application as database independent.
- For ODBC functions, implementation will be provided by vendors.
- A vendor is nothing but an organisation, who supports a particular technology.
- For ex, Oracle corporation is at a vendor, it supports ODBC Technology.

3) A program created by a vendor which contains implementation of ODBC API is called ODBC driver.

4) An application connects with ODBC driver and that driver connects with database.



5) For each database there is a separate ODBC driver and with the help of that driver an application can connect with a database.



10) ODBC API has made an application as a database independent.

11) A java application can also use microsoft ODBC API to connect with a database.

12) with java and ODBC combination, sun microsystem identified two major problems -

(i) java is a platform independent language and odbc is a platform dependent API. If java uses ODBC then java program becomes platform dependent but it is against the promise of java "WORA"

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(ii) java program is non-pointers and ODBC functions are in c language with pointers. If java program uses ODBC then non-pointers java program converted to pointers c-program and then it is connected to database. This conversion takes some time. So, a program connects with database slowly.

13) To overcome the above two problems, sun microsystem released JDBC API.

***** Q) What is the difference b/w jdbc and odbc?

Ans- (i) jdbc API is in java language. So, it is platform independent. But, ODBC API is in c-language. So, it is platform dependent.

(ii) jdbc API doesn't contains pointers but odbc API contains pointers.

Q) What is jdbc?

Ans- (i) jdbc is an API (Application programming Interface) for connecting java application with databases in platform independent manner and database independent manner.

(ii) jdbc API contains a set of classes and interfaces given under mainly two packages

- (i) `java.sql*`;
- (ii) `javax.sql*`;

(iii) To connect a java program with a database, the following softwares are needed

- 1) jdk
- 2) driver
- 3) database

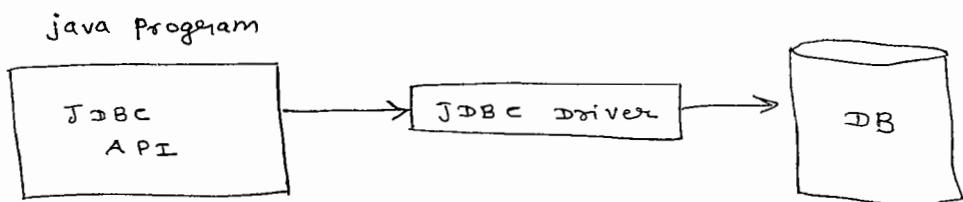
18 Aug 15

jdbc driver

- 1) A jdbc driver is a software, it contains a group of implementation classes for the interfaces of jdbc api.
- 2) A jdbc driver software will be provided by a driver vendor.
- 3) A driver vendor is a company / organisation who creates a jdbc driver product.
- 4) Some list of jdbc driver vendors are -
 - i) Altera software
 - ii) connect software
 - iii) DataDirect Technologies
 - iv) HIT Software
 - v) IBM
 - vi) IDS software
 - vii) MySQL
 - viii) OpenLink
 - ix) Oracle
 - x) simba
 - x) Sybase

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagari Bakery,
Opp. CEG, Balkampet Road,
Ameerpet, Hyderabad.

- 5) In a jdbc driver software , there is a class in which connection logic exist that class is called a driver class .



- 6) A jdbc driver class is a part of jdbc driver software and it is a mediator b/w java program and database

SRI RAHUVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Jdbc Specification

- i) A specification means it is a document .
- ii) A specification contains description about classes and interfaces of jdbc api .
- iii) By reading a programmer can understand about how to develop the applications by using that particular api .
- iv) Mostly sun-microsystem released specification in the form of pdf documents .

Types of jdbc driver

- i) sun microsystem divided jdbc drivers into 4 types , based on their functionality .
- ii) The drivers that are developed with java and with the support of native languages like c or c++ , then they are called partial java drivers .

3) If a driver is developed with 100% implementation in java then it is a pure java driver.

Type1 : JDBC-ODBC Bridge driver

Type2 : Native API partially java driver

Type3 : Net protocol pure java driver

Type4 : Native protocol pure java driver

4) Except type1 driver, the remaining types of drivers are given by vendors.

5) In jdbc, driver vendors are of two types -

i) Database vendor

ii) Third party vendor

6) A database vendor means, a vendor who provided driver software to connect with its own database.

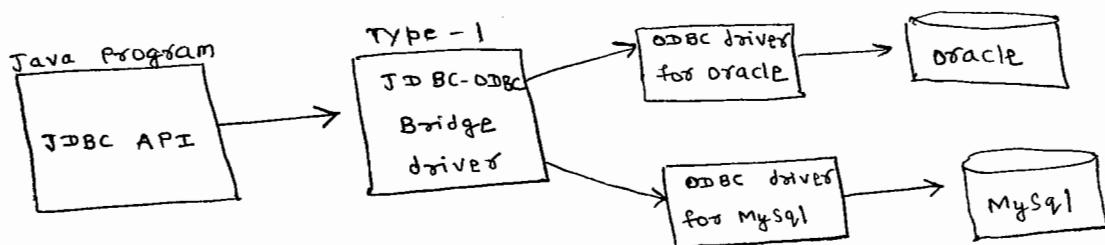
For example, Oracle is a database vendor provided driver for connecting oracle database.

7) A third party vendor is a vendor who provides driver software for connecting with other database.

For example, IDS Software is a vendor provided a driver software for connecting with MS-Access and oracle databases.

19 Aug 15 Type I Drivers - (JDBC-ODBC bridge)

- 1) Type I driver given by sun-microsystem
- 2) Type I driver released by sun, for testing whether a java application is connecting with database or not.
- 3) Type I driver uses appropriate ODBC driver, to connect with a database.
- 4) This type I driver acts as a bridge between jdbc-api and odbc-api.
- 5) This type I driver is installed automatically alongwith jdk software. so, we no need to install separately driver software.



- 6) Type I driver is database independent driver. It means type I driver is for connecting with any database
- 7) ODBC driver is a database dependent driver because we need a separate ODBC driver for each database.
- 8) The ratio b/w ODBC driver and database is 1:n
- 9) The ratio b/w type I driver and database is 1:n

Advantages (PROS)

- 1) Type1 driver is automatically installed along with jdk software. so, we no need to install separately.
- 2) Type1 driver is a database independent driver.

Disadvantages (cons)

- 1) Type1 driver is a platform dependent driver because of ODBC driver.
- 2) Type1 driver is slow because first it connects with ODBC driver then connects with database.
- 3) Java Applets can't use type1 driver to connect with database.
- 10) Type1 driver is removed from java8.

Steps to write a jdbc program :-

- 1) load driver class
- 2) Establish connection
- 3) create statement object
- 4) Execute the commands
- 5) Print the result
- 6) close connection

SRI RAJENDRA XEROX
Software Languages Material Available
Beside Bangalore Material Available
Opp. CDAC, Balkampet Bakery,
Ameerpet, Hyderabad

20 Aug 15

Step 1) A Driver class contains logic to open a connection with database. So, we need to load the driver class into JVM.

2) To load a class into JVM, we need to call static Method forName() and it is given in Class.

3) In java, we can call static method with respect to classname so, syntax of calling forName is —

Class.forName("fully qualified className");

4) sun microsystem provided driver className is

sun.jdbc.odbc.JdbcOdbcDriver
 {
 package name classname

 fully qualified classname
 }

SRI RAHMAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

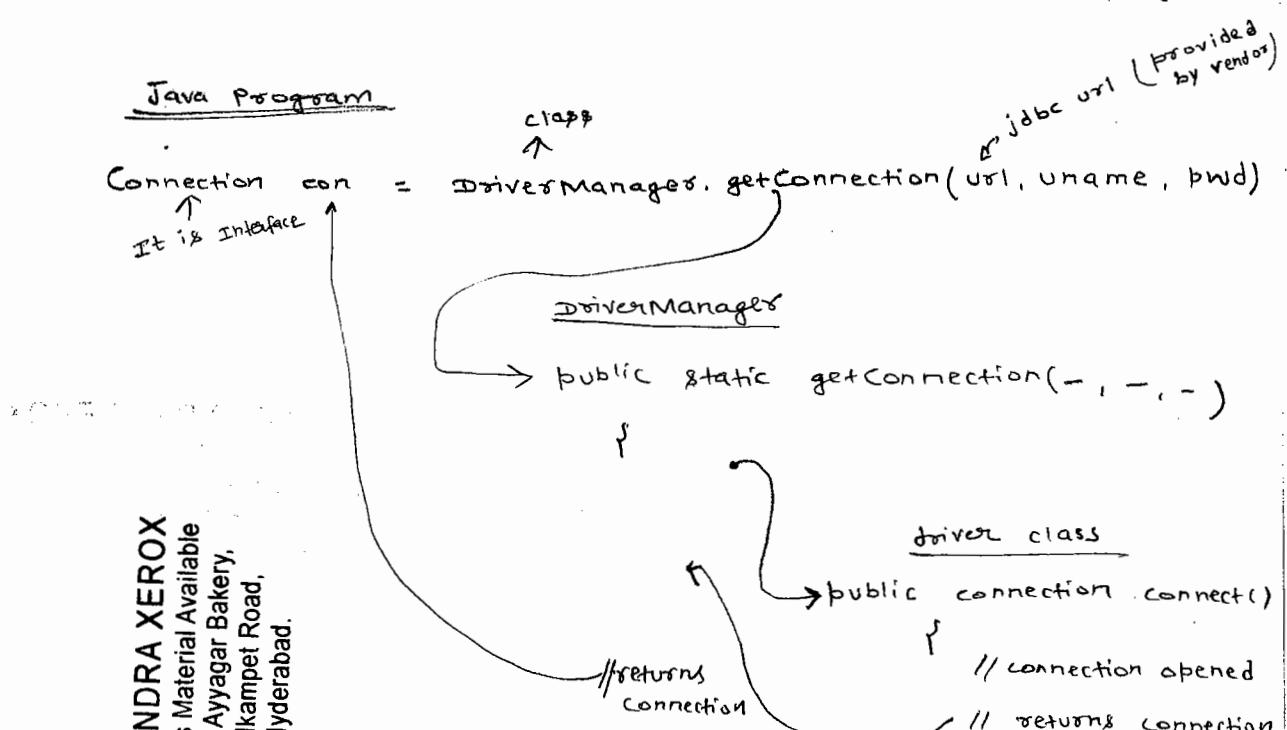
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

5) In Every jdbc driver class there exist static block and it is executed whenever a driver class is loaded.

6) In static block of a driver class, object of the driver class is created and it is registered with DriverManager class.

7) DriverManager class is a predefined class given in java.sql package and it stores object of the driver classes loaded.

- Step - 1) To open a connection with a database, we call `getconnection()` of `DriverManager` class.
- 2) `getconnection()` is a static method, so we can call directly with a `DriverManager` class.
- 3) `getconnection()` internally calls `connect()` of `Driver` class. The `connect()` returns connection object back to `getconnection()` and finally `getconnection()` returns connection object back to java program.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 4) Every Driver class has a separate URL to identify one driver among multiple drivers registered with a driver manager.
- 5) Not A driver URL will be supplied by driver vendor.

Step 3

- i) Statement object provides methods for sending the SQL commands to the database. So, we need to create a statement object.

Interface → Statement stmt = con.createStatement();

Step 4

To execute the command, we have to call one of the following 3 methods —

- (i) executeUpdate()
- (ii) executeQuery()
- boolean ← (iii) execute()

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 1) If we want to execute the Non Select operations on database then we need to call executeUpdate().
- 2) If we want to execute select operation then we need to call executeQuery().
- 3) If we want to execute a non select or select operation then we can call execute().

```
int i = stmt.executeUpdate("insert into student
                           values(101, 'A', 500)")
```

```
ResultSet rs = stmt.executeQuery("select * from student")
```

Step 5 If non select command executed then we will get result as integer and if select command is executed we will get the result as Resultset object. so, we need to point either integer or data (records) stored in Resultset object.

Finally, we need to close the connection with database.

e.g. `con.close();`

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

31 Aug 15

Data source Name (DSN)

- 1) A DSN is a file in which we write information about name and location of a database.
- 2) Type driver uses ODBC driver and ODBC driver reads information from DSN file for opening a connection with the database.
- 3) A DSN file can be created by using a predefined tool called data sources, it is under Administrative tools of the system.
- 4) DSN are of 3 types -
 - (i) System DSN
 - (ii) User DSN
 - (iii) File DSN
- 5) A System DSN is shareable to all user accounts.
- 6) A User DSN is only shareable to current user account of the system.

- ⇒ A File DSN is also like user DSN only but we can store it our require location with extension of the file .dsn.
- b) If we create a System DSN then it will be stored in local machine folder of windows registry.
- c) If we create a User DSN then it will be stored in current user folder of windows registry.

Example1

The following jdbc program creates a table in oracle database by using type1 driver.

//Program1.java

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

class Program1
{
    public static void main(String[] args) throws Exception
    {
        // step-1 : load driver class
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        System.out.println("driver is loaded");

        // step-2 : open connection
        Connection con = DriverManager.getConnection("jdbc:odbc:oracledsn",
                                                     "system", "tiger");
        System.out.println("connection is opened");
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available,
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

// Step-3 : Create Statement object
Statement stmt = con.createStatement();
System.out.println("Statement object created");

// Step-4 : execute command
stmt.executeUpdate("create table student (sid number(s),
sname varchar(10), marks number(s))");

// Step-5 : print output
System.out.println("Table is created");

// Step-6 : close connection
stmt.close();
con.close();
System.out.println("Connection is closed");
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

- i) Create a DSN like the following —
 - (i) Open control panel → Administrative Tool →
 - Data Sources (ODBC) → Select system DSN →
 - Click on Add Button → Select OracleXE →
 - Finish → Enter datasource name **oracledbn**
 - User id **system**
 - OK → OK

Compile and Run Program1.java

```

E:\> javac Program1.java
E:\> java Program1
driver is loaded
connection is opened
Statement object created
table is created
connection is closed

```

22 AUG 15

Absent
(Interview)

- Note) If database is oracle10g then DSN name can be created in windows 7 or windows 8 like the following -

C:\Windows\sysWOW64\odbcad32.exe → click add button → select microsoft odbc for oracle / finish / enter dsn as oracledsn and username as System → OK

- 2) To verify whether a table is created on database or not we need to open sqlplus and type the following command : desc student;

What is sqlplus

- 1) Sqlplus is a client software to connect with oracle database
- 2) It is automatically install along with oracle database

Q~ What are the different types of sql commands -

- Ans- i) DDL (create, alter, truncate, drop)
ii) DML (insert, update, delete)
iii) DRL (select)
iv) TSQL (commit, rollback, savepoint)
v) DCL (grant, revoke)
 ↳ can't be executed from Java Program

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

Example2 the following jdbc program is for inserting into table of oracle database.

// Program2.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
class Program2
{
    public static void main (String [] args) throws Exception
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        System.out.println("Driver is loaded");
        Connection con = DriverManager.getConnection
            ("jdbc:odbc:oracledb", "system", "tiger");
        System.out.println("connection is opened");
        Statement stmt = con.createStatement ();
        System.out.println("Statement object created");
        int i = stmt.executeUpdate("insert into
            student values (10101, 'Sathya', 999)");
        System.out.println(i + " row inserted");
        stmt.close();
        con.close();
        System.out.println("connection is closed");
    }
}
```

Compile and run

```
C:\> javac Program2.java ←
C:\> java Program2 ←
```

```
Driver is loaded
connection is opened
Statement object created
1 row updated inserted
connection is closed
```

Example3: the following jdbc program is updating marks

```
800 for student id 10101  
//Program3.java  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.Statement;  
  
class Program3  
{  
    public static void main(String args[]) throws Exception  
{  
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
        System.out.println("Driver is loaded");  
  
        Connection con = DriverManager.getConnection("jdbc:  
                odbc:oracledsn", "system", "tiger");  
        System.out.println("connection is opened");  
  
        Statement stmt = con.createStatement();  
        System.out.println("statement object created");  
        int i = stmt.executeUpdate("update student  
                set marks = 800 where sid = 10101");  
  
        System.out.println(" " + i + " row updated");  
  
        stmt.close();  
        con.close();  
    }  
}
```

Compile and Run

```
C:\> javac Program3.java
```

```
C:\> java Program3
```

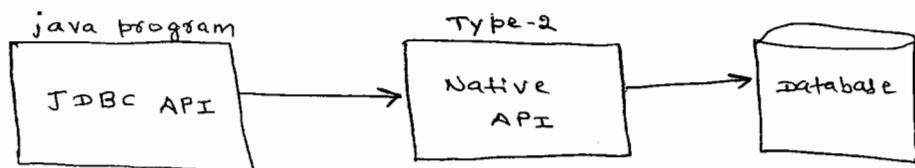
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagiri Bakery,
Opp: CDAC, Balkampet Road,
Ameerpet, Hyderabad.

24 AUG 15

Type 2 (Native API partly Java driver)

This type of drivers will be provided by vendors.

- 1) A type 2 driver is a partial java driver. It means the driver is not 100% implemented in java.
- 2) Type-2 category drivers use Native API of a database, to interact with the database.
- 3) Native - API contains a set of c-language functions to interact with a database.
- 4) Type 2 category drivers doesn't use ODBC drivers. So, there is no need of creating a DSN file while working with type 2 driver.
- 5) A type 2 driver directly connects a java program with a database.



- 6) Each database vendor may not provide a native API to interact with the database.

For ex, Oracle has provided a native API called **OCI** (oracle called interface) and DB2 has provided a native API called **CLI** (call level Interface) but MySQL has not provided native API.

Advantage (Pro)

- 1) Type 2 drivers are faster than type 1 driver, because type 2 drivers doesn't use ODBC driver.

Disadvantages (cons)

- 1) Type 2 drivers are platform dependent driver, because they use native API and native API is a c language API.
- 2) Type 2 drivers are database dependent driver, because native API will be different from one database to another database.
- 3) Java applets can't use type 2 driver to interact with database.

Oracle OCI driver

- 1) It is a type 2 driver product name given by Oracle.
- 2) A driver class name is OracleDriver and package name is oracle.jdbc.driver. so, the fully qualified class name is oracle.jdbc.driver.OracleDriver.
- 3) This oracle OCI driver software is automatically installed along with Oracle software.
- 4) At the time of loading OracleDriver, class NotFoundException will be thrown by the class loader, Because this class is not part of java API.
- 5) To resolve that exception, we need to add a jar file which contains that driver class to the classpath variable.

- 6) In oracle10g database, OracleDriver class exists in ojdbc14.jar and in oracle11g it exists in ojdbc6.jar.

Set classpath = c:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar;

- 7) The url of oracle OCI driver is -

jdbc: oracle:oci@XE
 ↳ service id

- 8) We can find the service id of oracle in the following 3 ways -

(i) open sqlplus and type the following command

SQL> select * from global_name;

(ii) open control panel → Administrative Tools → services → find OracleServiceXE
 ↳ service id

(iii) open c:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN\tename.ora file and find service name.

Example the following jdbc program is for deleting a row from student table using type-2 driver.

1/program4.java

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

class program 4
{
    public static void main( String args[] ) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("driver is loaded");
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:oci:@xe","system","tiger");
        System.out.println("connection is opened");
        Statement stmt = con.createStatement();
        int i = stmt.executeUpdate(" delete from
            student where sid = '901'");
        System.out.println(i + " row deleted");
        stmt.close();
        con.close();
    }
}

```

SRI KRISHNENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

25 Aug 15

Absent
(Interview)

Selecting the data from Table

- 1) To read the data from a table we need to call executeQuery() of statement interface.
- 2) executeQuery() returns ResultSet object. This object stores the now selected from a table.

- 3) A ResultSet object maintain a cursor and initially that cursor will position before the first row.
- 4) To read the rows, we need to move the cursor by calling next(), next() returns boolean. If true then there is a row to read the data and false then there is no more rows.
- 5) To read the data from a row, we need to call getXXX(), it means to read integer getInt(), for string we call getString() ... etc.

ResultSet rs = stmt.executeQuery("Select * from student");

SFR

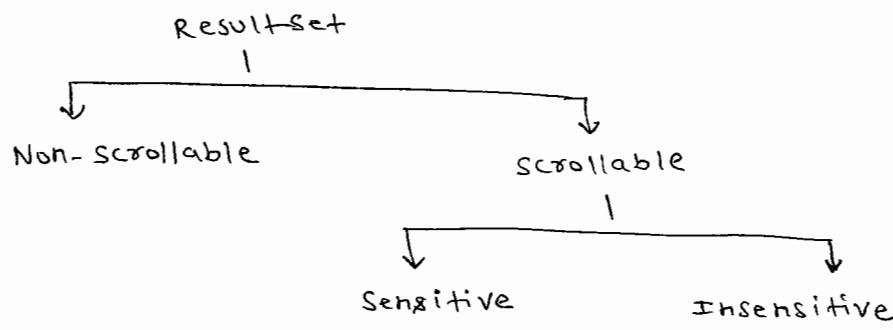
-	-	-
-	-	-
-	-	-

while (rs.next())

```

    {
        System.out.println(rs.getInt(1) + " " + rs.getString(2)
                           + " " + rs.getInt(3));
    }
  
```

- 1) In JDBC, we have two type of Resultset
- 1) Non scrollable / forward only
 - 2) Scrollable



- 2) By default, Resultset object is non-scrollable.
- 3) In non-scrollable Resultset object, a cursor can be moved only in forward direction but in scrollable Resultset, a cursor can be moved in both forward and backward direction.
- 4) One statement object in JDBC can create one Resultset object only. But if another Resultset object is opened, automatically closes a previous Resultset object.

Ex :-

```
ResultSet rs1 = stmt.executeQuery("select * from emp");
```

```
ResultSet rs2 = stmt.executeQuery("select * from dept");
```

while (rs1.next()) → exception

while (rs2.next()) → correct

5) If we want two Resultset object in a program then we need two statement also.

```
Statement stmt1 = con.createStatement();
```

```
Statement stmt2 = con.createStatement();
```

```
ResultSet rs1 = stmt1.executeQuery("Select *  
from emp");
```

```
ResultSet rs2 = stmt2.executeQuery("Select *  
from dept");
```

```
while(rs1.next()); → correct
```

```
while(rs2.next()); → correct
```

6) The ratio b/w a connection and statement object is 1:m, it means a single connection object can create multiple statement object.

7) The ratio b/w statement and Resultset object is 1:1, it means one statement can create one Resultset object only.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Program 5.java

SRI RAJAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

class Program5
{
    public void main( String args[] ) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        oracle.jdbc.driver.OracleDriver driver = new
        oracle.jdbc.driver.OracleDriver();
        DriverManager.registerDriver( driver );
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:oci:@xe", "system",
             "tiger");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select * from
            Student");
        while ( rs.next() )
        {
            System.out.println( rs.getInt(1) + " " + rs.getString(2)
                + " " + rs.getInt(3));
            System.out.println(" = = = = ");
        }
        rs.close();
        stmt.close();
        con.close();
    }
}
```

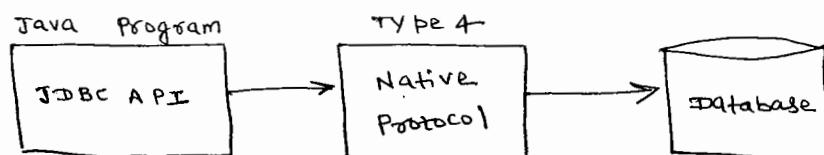
```

E:\> set classpath = C:\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar ;
E:\> javac programs.java
E:\> java programs
10101 A 500
10102 B 600

```

Type 4 Driver (Native Protocol Pure Java driver)

- 1) This category of drivers are 100% implemented in java. So, they are called pure java drivers.
- 2) This type 4 category driver uses native protocol (Database specific protocol), to connect with a database.
- 3) For ex, http is a common protocol to connect with any web server but ttc is native protocol only to connect with Oracle database.
- 4) Apart from native protocol, a type 4 driver uses ip address, port no. and service ID (SID) to interact with a database.
- 5) The ip address, port no., SID are passed to the type 4 driver from the URL of the type 4 driver.



SRI KRISHNENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Advantages (pros)

- 1) Type 4 drivers are faster than remaining types of drivers because type 4 driver doesn't use ODBC driver and native's API.
- 2) Type 4 drivers are platform independent. because they are pure java driver
- 3) Java Applets can use type 4 driver to connect with database.

Disadvantages (cons)

- 1) Type 4 drivers are database dependent.

TYPE	Platform Independent	Database Independent
TYPE 1	NO	YES
TYPE 2	NO	NO
TYPE 3	YES	YES
TYPE 4	YES	NO

Oracle thin driver

- 1) Oracle corporation has released two JDBC drivers products —
 - (i) Oracle OCI driver (type 2)
 - (ii) Oracle thin driver (type 4)
- 2) For both products driver class name is same and URL's are different.
- 3) The URL of oracle thin driver is
"jdbc:oracle:thin:@ipaddress:port:sid"

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

4) IP address is used to identify a system in network and port no. is to identify a server in a system.

27/08/15
Example 6 The following jdbc program is for updating marks of a student using oracle type 4 driver

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;

class programs
{
    public static void main (String args [])
        throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:XE",
             "system", "tiger");

        Statement stmt = con.createStatement();
        int i = stmt.executeUpdate ("Update
            Student set marks = 500 where
            sid = 10101");

        System.out.println (i + " row updated");
        stmt.close();
        con.close();
    }

    compile and run :
    E:\> java programs.java
    E:\> set classpath = c:\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar;
    E:\> java programs
    1 row updated
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Installing MySQL

MySQL database has two editions -

- (i) MySQL enterprise edition (commercial)
- (ii) MySQL community edition (open source)

- 1) visit www.mysql.com/downloads
 - ↓
 - click on community downloads
 - ↓
 - click on download link of mysql community server
 - ↓
 - click on download button of mysql installer MSI
 - ↓
 - click on download MSI installer (253.9 MB)
 - ↓
 - start my download
- 2) A setup file will be downloaded and double click on the file and installation started.
- 3) checked accepted agreement → next → next
→ execute → next → password → next
→ execute → finish
- 4) The connection properties of connector/j driver are -

com.mysql.jdbc.driver → driver class name
jdbc:mysql://localhost:2306/test → URL
root → username
root → password

Example: Following jdbc program for creating a table and inserting in the database MySQL type-4 driver.

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;

class Program
{
    public static void main(String args[]) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("Driver is loaded");

        Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/test", "root", "root",
            "connection is opened");

        Statement stmt = con.createStatement();
        System.out.println("Statement object created");

        int i = stmt.executeUpdate("insert into
            student values " + createTableStudent +
            "(sid int, sname varchar(10), marks int)");
        System.out.println("table created");

        int j = stmt.executeUpdate("insert into
            student values (101, 'ABC', 600)");

        System.out.println(j + " row inserted");

        stmt.close();
        con.close();
        System.out.println("connection is closed");
    }
}
```

SRI RAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 1) In the above program we loaded .com.
- 2) This driver class is not a part of Java API, so ClassNotFoundException is thrown.
- 3) We need to download mysql-connector-java-5.1.6.jar from google.

Compile and Run

E:\> javac program7.java

E:\> java program7

java. ClassNotFoundException

Set class path = c:\mysql-connector-java-5.1.6.jar;

E:\> java program7 ↵

driver is loaded

connection is opened

statement object created

Table created

1 row inserted

connection close

To verify whether table is created and row inserted or not do the following -

Start → program → mysql → mysqlServer 5.6
 → mysql commandline client → enter password
 → connect test;
 Select * from student;

Software Languages Material Available
 Beside Bangalore Ayagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

28 Aug 15 Installing SQLyog

- 1) When we install MySQL software then automatically MySQL command line client s/w is also automatically installed and it is a CLI client software.
- 2) SQLyog is a third party software for connecting MySQL database.
- 3) SQLyog is a GUI client s/w.
- 4) We can download a free trial version of SQLyog from <https://www.webyog.com/product/sqlyog>
- 5) Select either 64 bit or 32 bit version and download.
Enter email id
- 6) A download link will be sent to mail id
then click on download, exe file will be downloaded when we click that link.
- 7) When we install SQLyog then icon created on desktop. Start the SQLyog through that icon.
- 8) Enter the following details

host address	<input type="text" value="localhost"/>
username	<input type="text" value="root"/>
password	<input type="text" value="root"/>
port	<input type="text" value="3306"/>
database	<input type="text" value="test"/>
<input type="button" value="connect"/>	

MR. AVINASH VENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Prepared Statement

- 1) Prepared statement is an interface of java.sql package.
- 2) Prepared statement is a subinterface and its super is Statement interface.
- 3) There are two problems with Statement interface
 - (i) suppose, we are executing same sql command for multiple times, in a database, every time, the command is first compiled and then executed. Because of compile, performance of the application is going to be decreased.
 - (ii) Statement object can only send the text data to the database but it can't send the binary data (images/ Photo) to the database.
- 4) To overcome the above two problems, Prepared statement interface is given.
- 5) In case of Prepared statement, a sql command will be first send to the database for compilation. After that the compiled code of the command will be stored in PreparedStatement object.
- 6) To compile the command only command syntax is required but not the values. so, in the command in place of values we put index parameter (?) .

For ex,

Prepared statement pstmt = con.prepareStatement ("insert
into student values (?, ?, ?)");
Database
✓ compiled
✓ code
Stored

29 Aug 15

- 1) In a sql command we can used only ? symbol.
 - 2) we can't use any other special symbol
 - 3) we can't use (?) symbol to replace tablename, column name in the command, it is only for value
 - 4) we can't use (?) symbol in DDL commands (create, alter, truncate and drop commands)
 - 5) create table abc (id number (?), ~~name~~ varchar2 (?)) → invalid
 - 6) Select sid, ? from student → invalid
 - 7) Update ? set marks = 500 where sid = 10101 → invalid
 - 8) Select * from emp where empno = ? → valid

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example8 the following jdbc program insert 3 row with the dynamic value into the student table using PreparedStatement with database mysql.

```
// program8.java
import java.sql.DriverManager;
import java.sql.Connection;
import java.util.Scanner;
import java.sql.PreparedStatement;

class program8
{
    public static void main(String args[]) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/test","root","root");

        PreparedStatement pstmt = con.prepareStatement
            ("insert into student values (?, ?, ?)");
        Scanner s = new Scanner (System.in)
        for( int i=1 ; i <=3 ; i++)
        {
            System.out.println(" Enter sid");
            int sid = s.nextInt();
            System.out.println(" Enter sname");
            String sname = s.nextLine(); //s.nextLine();
            System.out.println(" Enter marks");
            int marks = s.nextInt();
            pstmt.setInt(1, sid);
            pstmt.setString(2, sname);
            pstmt.setInt(3, marks);
            pstmt.executeUpdate();
        }
    }
}
```

SRI KUCHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

    // set values
    pstmt.setInt(1, sid);
    pstmt.setString(2, sname);
    pstmt.setInt(3, marks);
    int k = pstmt.executeUpdate();
    System.out.println(k + " row inserted");
}

pstmt.close();
con.close();
}
}

```

compile and run

```

C:\> javac program8.java
C:\> set classpath=C:\mysql-connector-java-5.1.6.jar;
      .
C:\> java program8

```

```

Enter sid
102
Enter sname
RAM
Enter marks
500

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Inserting an Image

- 1) An image is a file with extension .gif or .jpg or .bmp. In a database, directly an image file can't be stored but binary data image can be stored.
 - 2) To store the binary data we need blob type column in table (binary large object)
 - 3) for inserting binary data, we must need PreparedStatement of jdbc. To set binary stream of data, we need to call getBinaryStream()
- getBinaryStream(parameter index, FileInputStream object, length of the file);

31 Aug 15

Example 9

```
// ImageInsert.java  
  
class ImageInsert  
{  
    public static void main(String[] args) throws Exception  
{  
        Class.forName("oracle.jdbc.driver.OracleDriver");  
        Connection con = DriverManager.getConnection("jdbc:  
        oracle:thin:@localhost:1521:XE", "system", "tiger");  
        PreparedStatement pstmt = con.prepareStatement("insert  
        into employees values (?, ?, ?)");  
        pstmt.setInt(1, 9999);  
        pstmt.setString(2, "Sunder.P");  
        FileInputStream fis = new FileInputStream("D:/pichai.jpg");  
        File f = new File("D:/pichai.jpg");  
        int len = (int)f.length();  
        pstmt.setBinaryStream(3, fis, len);  
        int k = pstmt.executeUpdate();  
        System.out.println(k + " row inserted");  
        pstmt.close();  
        con.close();  
    }  
}
```

compile & run

```
E:\> javac ImageInsert.java  
E:\> java ImageInsert  
1 row inserted
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Retrieving an Image

- 1) Retrieving an image is nothing but reading binary data of an image from table statement.
- 2) To retrieve an image also we must use preparedstatement of jdbc.
- 3) When we select an image then its binary data result will be stored in Resultset object.
- 4) To read binary data from Resultset, we call getBinaryStream().
- 5) getBinaryStream() reads bytes from Resultset, stores them in InputStream and then returns InputStream object.
- 6) To write the bytes into a file, we need to create a FileOutputStream object.
- 7) We need to read one by one byte from InputStream object till end of the file bytes(-1) and we need to write the bytes into file.

SRI RAJAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

Example 10 // Imageselect.java

class ImageSelect
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "tiger");
        PreparedStatement pstmt = con.prepareStatement(
            "select photo from employees where empno=?");
        pstmt.setInt(1, 9999);
        ResultSet rs = pstmt.executeQuery();
        rs.next();
        InputStream is = rs.getBinaryStream(1);
        FileOutputStream fos = new FileOutputStream(
            "F:/sunder.jpg");
        int k;
        while ((k = is.read()) != -1)
        {
            fos.write(k);
        }
        System.out.println("Image Selected, refer F:/sunder.jpg");
        rs.close();
        pstmt.close();
        con.close();
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

compile and run

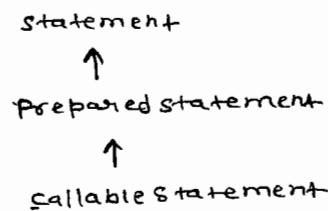
E:\> javac ImageSelect.java

E:\> java ImageSelect

Image Selected, refer F:/sunder.jpg

CallableStatement

- i) CallableStatement is an interface and its super interface is PreparedStatement.



- ii) The difference b/w PreparedStatement and CallableStatement is with the help of PreparedStatement we can only execute the command but with the help of CallableStatement we can execute queries commands and also we can call procedures and functions of a database.

Syntax 1 CallableStatement cstmt = con.prepareStatement ("sql command");

Syntax 2 CallableStatement cstmt = con.prepareCall ("{call profun(?)}

1 Sept 15

- Q) Why procedure or function is created in database?

Ans -

- 1) Suppose in a program, we want to do calculation by reading multiple values from database.
- 2) To read multiple values from database multiple trips are needed b/w program and database.
- 3) When no. of trips are increased then performance will be decreased.

(4) To improve the performance, in database only calculation will be done by creating a procedure or a function and only result will be returned back to the program. So, that no. of trips are reduced and performance will be improved.

Q) When a procedure is created in database?

Ans- After calculations if there are no value to return or there are multiple values to return then a procedure is created.

Q) When a function is created in database?

Ans- After calculations if there is exactly one value to return a function is created.

Q) Java program creates a procedure or function in database?

Ans- No

Q) Java program calls a procedure or function of database?

Ans- Yes (using callable statement)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Syntax for creating procedure in oracle

- 1) CREATE OR REPLACE PROCEDURE **proname** (**parameters if any**) IS

```
    variables;  
begin  
    statements;  
end;
```

Parameters of a procedure are of 3 types -

- (i) in parameter
- (ii) out parameter
- (iii) inout parameter

- (i) in parameters are used to read the input
- (ii) out parameters are used to return the output
- (iii) inout parameters are for both.

- 2) If we don't provide a parameter mode then by default it is in parameter.

Example

The following procedure takes input as employee no. and returns output as employee name and experience.

```
SQL> ed a1 ← [notepad is opened]
```

```

create or replace procedure pro_exp(eno in number,
        empname out varchar2, experience out number)

is
    DOJ DATE;
    today DATE;
begin
    select ename into empname from emp
        where empno = eno;

    select hiredate into DOJ from emp
        where empno = eno;

    select sysdate into today from dual;

    experience := (today - DOJ) / 365;

end;

```

SQL> @ a1
line no.
to compile 15 /

Procedure created

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

2 Sept 15

Example11 The following JDBC program calls the above procedure of database by using CallableStatement

// ProcedureTest.java

```

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.CallableStatement;
import java.sql.Types;

```

```

class ProcedureTest
{
    public static void main(String[] args)
    {
        Connection con = null;
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection("jdbc:
                oracle:thin:@localhost:1521:xe",
                "system", "tiger");

            CallableStatement cstmt = con.prepareCall
                ("?call pro-exp(?, ?, ?)");
            cstmt.setInt(1, 7900);
            cstmt.registerOutParameter(2, Types.
                VARCHAR);
            cstmt.registerOutParameter(3, Types.INTEGER);
            cstmt.execute();
            String str = cstmt.getString(2);
            int i = cstmt.getInt(3);
            System.out.println("Name : " + str);
            System.out.println("Experience : " + i);
            cstmt.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        finally
        {
            try
            {
                con.close();
            }
            catch(Exception e2)
            {
                System.out.println(e2);
            }
        }
    }
}

```

// console will throw exception while closing the connection

SRI RAJAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

compile and run -

E:\> javac procedureTest.java

E:\> java procedureTest

Name : MILLER

Experience : 15

Note -

Types is a class, used to convert database datatypes into java datatypes.

Syntax for creating a function in oracle

- 1) create or replace function functioname(parameters
return returntype is

```
variables;  
begin  
statements;  
end;
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 2) In a function, parameters are IN only
but in a procedure parameters are
IN, OUT, INOUT.

- 3) In a function body return statement is
compulsory but for a procedure body
return statement is not allowed.

the following function of oracle takes input
as employee number and return bonus as
output →

SQL> ed a2

```
Create or replace function fun_bonus(eno number)
return number is
temp number(s);
bonus number(s);
begin
    Select sal into temp from emp where empno=eno;
    if temp <= 2000 then
        bonus := temp * 0.10 ;
    elsif temp > 2000 and temp <= 6000 then
        bonus := temp * 0.20 ;
    else
        bonus := temp * 0.30 ;
    end if ;
    return bonus ;
end ;
```

Save and Exit

SQL> @ed2 ;

16 /

Function created

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example 2

The following jdbc program is for calling the above function created in oracle by using callable statement of jdbc -

```
// FunctionTest.java
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.CallableStatement;
import java.sql.Types;

class FunctionTest
{
    public static void main(String[] args)
    {
        Connection con = null;
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "tiger");
            CallableStatement cstmt = con.prepareCall("{? = call fun_bonus(?)");
            cstmt.registerOutParameter(1, Types.INTEGER);
            cstmt.setInt(2, 7456);
            cstmt.execute();
            int i = cstmt.getInt(1);
            System.out.println(" Bonus = " + i);
            cstmt.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        finally
        {
            try
            {
                con.close();
            }
            catch(Exception e2)
            {
                System.out.println(e2);
            }
        }
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet, Hyderabad.

Compile and Run

E:\> javac FunctionTest.java

E:\> java FunctionTest

Bonus = 3000

Q) Why we need register out parameter?

- Ans - 1) By default, callablestatement assumes that all index parameters as in parameter.
- 2) In order to tell the callablestatement that a parameter is out parameter, we need to call registerOutParameter().

3 sept 15

SCROLLABLE RESULTSET

- 1) In JDBC Resultset are of two types —
- (i) non-scrollable
 - (ii) scrollable
- 2) In a non-scrollable result set, cursor can be moved in a forward direction only.
- 3) In a scrollable Resultset, a cursor can be moved in both forward and backward direction.
- A) Scrollable Resultset are again divided into 2 types —
- (i) sensitive (commercial driver)
 - (ii) insensitive (open source driver)

↳ we are using
this driver

SRI KRISHNENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 5) The difference b/w a sensitive and insensitive type is, sensitive type allows a modification done on the database table but insensitive type doesn't allow the changes done on the database table.
- 6) For example, a result set object is a sensitive type and in database, suppose third row is deleted then in the resultset object also, 3rd row is deleted automatically. If it is insensitive then 3rd is not deleted from the resultset object.
- 7) To create a scrollable resultset object, while creating a statement object we need to pass type and mode parameters.
- 8) Type and mode parameters are integers and their constant values defined in the Resultset interface.
- 9) In interface, all the variables are public static and final so the variable names are represented in uppercase letters.

Type value

10) `public static final int TYPE_SCROLL_INSENSITIVE = 1004;`
`public static final int TYPE_SCROLL_SENSITIVE = 1005;`

mode value

`public static final int CONCUR_READ_ONLY = 1007;`
`public static final int CONCUR_UPDATABLE = 1008;`

11) When creating a Statement object we can pass type and mode parameters like the following -

```
statement stmt = con.createStatement( 1004, 1007 );  
(08)
```

```
statement stmt = con.createStatement
```

```
(Resultset.TYPE_SCROLL_INSENSITIVE,  
Resultset.CONCUR_READ_ONLY);
```

12) If the Resultset is opened in read only mode then we can only read the rows from Resultset but we can't modify. If it is opened in updatable mode then we can read the rows and even we can modify the rows also.

13) A scrollable Resultset object provides better performance than non-scrollable Resultset because in Scrollable Result set we can move the cursor from top or from bottom or directly to a specific record.

14) To move the cursor in forward direction or backward direction or directly to a specific row we call the below methods -

- (1) afterLast()
- (2) previous()
- (3) absolute()
- (4) relative()
- (5) first()
- (6) last()
- (7) beforeFirst()
- (8) next()
- (9) getRow()

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

absolute() — jumps the cursor to specific row either from top or from bottom ~~bottom~~ of the resultset.

For ex, `rs.absolute(8)` moves/jumps cursor to 8th row from top.

SRI KRISHNA VENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

similarly,

`rs.absolute(-8)` moves/jumps cursor to 8th row from bottom.

relative() — jumps the cursor to a specific row from current position of the cursor.

For ex, if the current position of cursor is 12 and if we say `rs.relative(4)` then the cursor will moved to 16th row.
 $(12+4=16)$

Similarly, if we say `rs.relative(-3)` then the cursor will moved to 9th row. $(12-3=9)$

getRow() — used to read the current position of the cursor.

```
int i = rs.getRow(); // i stores cursor position say 9
```

4 Sept 15

Example 13 The following JDBC program is for printing records of emp table in backward direction -

```
// ScrollableTest.java
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;

class ScrollableTest
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("-", "-", "-");
        Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = stmt.executeQuery("select * from emp");
        rs.last();
        while (rs.previous())
        {
            System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " +
                               rs.getInt(3) + " " + rs.getInt(4));
        }
        System.out.println("-----");
        System.out.println(rs.getInt(3));
        System.out.println("-----");
        System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " +
                           rs.getInt(3) + " " + rs.getInt(4));
        System.out.println("-----");
        rs.close();
        stmt.close();
        con.close();
    }
}
```

SKI RAUNAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q) Suppose a Resultset has 5 rows and if we move the cursor to 10th row then what happens ?

Ans - An Exception is thrown with a message exhausted Resultset.

Q) Suppose a row has 4 columns and if we point the 5th column then what happens ?

Ans - An Exception is thrown with a message invalid column index.

Q) Can we create a scrollable Resultset by using PreparedStatement and CallableStatement or not?

Ans - Yes, we need to pass 3 parameters by creating the PreparedStatement or CallableStatement object.

```
PreparedStatement pstmt = con.prepareStatement  
("command", type, mode);
```

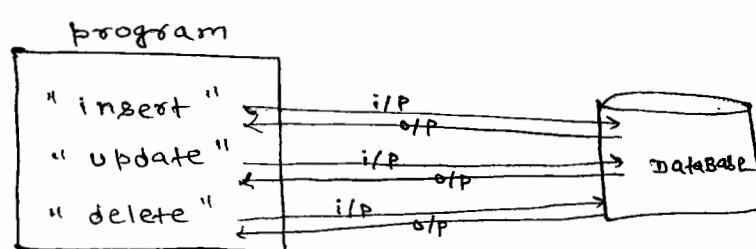
```
CallableStatement cstmt = con.prepareCall ("command", type,  
mode);
```

SRI RAHNAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

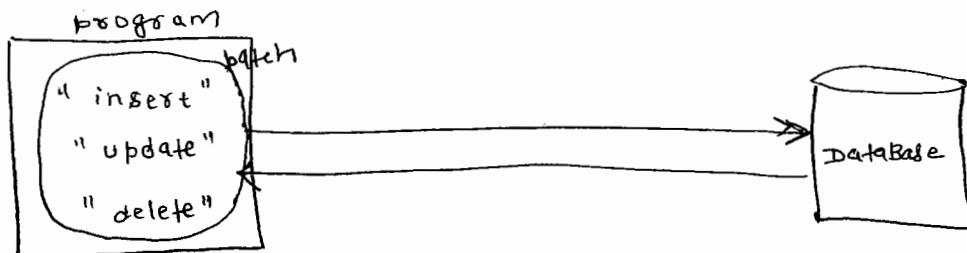
BATCH PROCESSING

- 1) From one jdbc program, we can execute multiple sql commands on a database.
- 2) By default one by one command will be executed so no. of trips b/w java application and database will be increased.
- 3) When no. of trips are increased then performance of an application will be decreased.
- 4) To improve the performance of an application we can add multiple commands to a batch and we can execute that batch in a single trip. This is called batch processing.

without batch processing :



with batch processing



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 5) A rule in batch processing is, Select command is not allowed in batch processing.
- 6) To add SQL command to a batch, we need to call addBatch().
- 7) To execute the batch we need to call executeBatch().
- ** 8) There is no relation b/w ResultSet object and batch processing because select command is not allowed in a batch.
- 9) Suppose, in a batch there are 4 commands, first and second commands are successfully executed and exception is raised in 3rd command then 4th command is cancelled but 1st and 2nd command are already executed, so they are not cancelled. Finally, a Batch Update exception will be thrown.
BatchUpdateException

Sept 15

- 10) Batch processing can be done with PreparedStatement and CallableStatement also.
- 11) If we use PreparedStatement or CallableStatement then we can add same command for multiple time to the batch.

For ex,

```
PreparedStatement pstmt = con.prepareStatement  
("Insert into student values  
( ?, ?, ? )");  
  
pstmt.setInt(1, 101);  
pstmt.setString(2, "A");  
pstmt.setInt(3, 650);  
  
pstmt.addBatch();  
  
pstmt.setInt(1, 102);  
pstmt.setString(2, "B");  
pstmt.setInt(3, 750);  
  
pstmt.addBatch();  
  
int i[] = pstmt.executeBatch();
```

Q: What is the difference b/w Batch processing with statement and batch processing prepared statement.

Ans - With statement object, we can add different commands to the Batch and with the Prepared statement object, we can add same command for multiple time to a batch.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example 14

```
// BatchTest.java

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;

class BatchTest
{
    public static void main(String args[]) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:
            oracle:thin:@localhost:1521:xe", "system", "tiger");

        Statement stmt = con.createStatement();
        stmt.addBatch("update emp set sal=2000 where
                      empno = 7321");

        stmt.addBatch("delete from satya where
                      id = 10101");

        stmt.addBatch("insert into dept values
                      (10, 'ACCOUNTING', 'DELHI')");

        int i[] = stmt.executeBatch();
        System.out.println("Batch completed");

        stmt.close();
        con.close();
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example 15 Example for Batch processing with prepared statement

```
// PreparedStatementBatchTest.java
```

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;

class PreparedStatementBatchTest
{
    public static void main(String args[]) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:xe",
             "system", "tiger");

        PreparedStatement pstmt = con.prepareStatement
            ("insert into dept values (?, ?, ?)");

        pstmt.setInt(1, 20);
        pstmt.setString(2, "RESEARCH");
        pstmt.setInt(3, 10);
        pstmt.addBatch();

        pstmt.setInt(2, 30);
        pstmt.setString(3, "FINANCE");
        pstmt.setInt(4, 20);
        pstmt.addBatch();

        int i[] = pstmt.executeBatch();
        System.out.println("Batch executed");

        pstmt.close();
        con.close();
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

UPDATABLE RESULTSET

- 1) Updatable Resultset is used to perform insert, update and delete operation on table of database , by without typing sql commands.
- 2) Updatable Resultset is an optional feature , so a jdbc driver may or may not support Updatable Resultset feature .
- 3) If we want to make a Resultset as updatable by creating Statement object we need to pass CONCUR_UPDATABLE as mode.
for example

Statement stmt = con.createStatement

(~~ResultSet.TYPE_SCROLL_INSENSITIVE~~
ResultSet.CONCUR_UPDATABLE);

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

- 4) for ex , if we want to delete 3rd row from emp table then we need to delete 3rd row from Resultset object like the following —

rs.absolute(3);
rs.deleteRow();

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Balkampet Road
Ameerpet, Hyderabad.

5) for ex, if we want to update salary of 5th employee then we need to update salary in Resultset like the following -

```
rs.absolute(5); → row  
rs.updateInt(3, 6500); → updated salary  
                           ↓ 3rd column  
rs.updateRow();
```

6) If we want to insert a new record (row) into the table then we need to insert a new record into the Resultset object like the following -

```
rs.moveToInsertRow(); // It moves cursor to empty row  
rs.updateInt(1, 7911);  
rs.updateString(2, "ABCD");  
rs.updateInt(3, 7600);  
rs.updateInt(4, 20);  
rs.insertRow();
```

Note -

When we create updatable Resultset then automatically one empty row is added to the Resultset for inserting the record.

SRI RAHMAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

8 Sept 15
Example 16

The following jdbc program is on updatable
Resultset for inserting a row into emp
table of mysql (oracle not supported)

```
// UpdatableResultSetTest.java

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;

class UpdatableResultSetTest
{
    public static void main(String[] args) throws Execution
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");
        Statement stmt = con.createStatement(ResultSet.TYPE_
                                         SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = stmt.executeQuery("select * from emp");
        rs.moveToInsertRow();
        rs.updateInt(1, 800);
        rs.updateString(2, "MOHAN");
        rs.updateInt(3, 9000);
        rs.updateInt(4, 20);
        rs.insertRow();
        rs.close();
        stmt.close();
        con.close();
    }
}
```

4

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Execute()

- 1) This method is used to execute the both select and non select operation.
- 2) If the command is select then execute() will run the command, stores the records in ResultSet object and then stores the ResultSet object in statement cache and finally returns a boolean value true.
- 3) If the command is non select then execute() will run the command, stores the int value in statement cache and finally returns a boolean value false.

To read the ResultSet object from cache, we need to call getResultSet()

To read int value from cache, we need to call getUpdateCount()

Generally, we call execute() when we want to take the command as input at Runtime.

Example17

The following jdbc program takes sql command as input at runtime and execute that command by calling execute().

// ExecuteTest.java

```

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.Scanner;

public class ExecuteTest {
    public static void main(String args[]) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");
        Statement stmt = con.createStatement();
        Scanner s = new Scanner(System.in);
        System.out.println("Enter your query");
        String query = s.nextLine();
        boolean flag = stmt.execute(query);
        if (flag == true) {
            ResultSet rs = stmt.getResultSet();
            while (rs.next()) {
                System.out.println(rs.getInt(1));
            }
        }
        rs.close();
    }
}

```

```

        else
        {
            int i = stmt.executeUpdate();
            System.out.println(i);
        }
        stmt.close();
        conn.close();
    }
}

```

Compile and Run

```

E:\> javac ExecuteTest.java
E:\> java ExecuteTest
Enter your query:
Update emp set sal = 1000 where empno = 7901
1 row updated

E:\> java ExecuteTest
Select * from emp
101
102
103
104

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Opp. CDAC, Ayyagar Bakery,
 Ameerpet, Hyderabad.

Reading connection properties from properties file

- 1) while writing the jdbc program we are typing driver name, url, username and password directly in the program.
- 2) A drawback is, if we want to change to another driver we need to modify the program, recompile the program, then we need to execute the program.
- 3) To resolve the above drawback we use properties file.
- 4) A properties file is also called a resource bundle and it will store the data key / value pair format.
- 5) A properties file can be created through notepad and it will be saved with extension .properties.

9 Sept 15

- 6) In a jdbc program, to read the data from properties file we need the following steps —
 - (i) create a `FileInputStream` class object, for reading the data from properties file.
 - (ii) create `java.util.Properties` class object.
 - (iii) copy the data from properties file into the `Properties` class object by calling `load()`

(iv) After loading, we need to read the value of each key from Properties class object by calling getProperty()

for example

Save it as- connection.properties (write in notepad)

```
jdbc.driver = oracle.jdbc.driver.OracleDriver  
jdbc.url = oracle.jdbc:oracle:thin:@localhost:1521:XE  
jdbc.uname = system  
jdbc.pwd = tiger
```

Jdbc program (sample code)

```
FileInputStream fis = new FileInputStream("connection.properties");  
Properties props = new Properties();  
props.load(fis);  
String s1 = props.getProperty("jdbc.driver");  
String s2 = " " ("jdbc.url");  
String s3 = " " ("jdbc.uname");  
String s4 = " " ("jdbc.pwd");
```

Key	value/name
jdbc.driver	XX
jdbc.url	XXX
jdbc.uname	XX
jdbc.pwd	XX

```
Class.forName(s1);  
Connection con = DriverManager.getConnection(s2, s3, s4);
```

AVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example 18

```
// PropertiesTest.java

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.io.FileInputStream;
import java.util.Properties;

class PropertiesTest
{
    public static void main (String[] args) throws Exception
    {
        FileInputStream fis = new FileInputStream("connection.properties");
        Properties prop = new Properties();
        prop.load(fis);
        String s1 = prop.getProperty("jdbc.driver");
        String s2 = prop.getProperty("jdbc.url");
        String s3 = prop.getProperty("jdbc.username");
        String s4 = prop.getProperty("jdbc.password");
        // load driver
        Class.forName(s1);
        // connection
        Connection con = DriverManager.getConnection
            (s2, s3, s4);
        // Statement
        Statement stmt = con.createStatement();
        // resultset
        ResultSet rs = stmt.executeQuery("select * from emp");
```

Mr. RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

        while(rs.next())
    {
        System.out.println(rs.getString(1) + " " + rs.getString(2) + " " + rs.getString(3)
                           + " " + rs.getString(4));
    }
    rs.close();
    stmt.close();
    con.close();
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Note: `getString()` is the common method for all but `getInt()` is specific to int value only. Here all means any datatype (like int, float, string, date ... etc)

RESULTSETMETADATA (interface)

- 1) As a programmer, if we don't know about columns information of a table then we can't read the data from the ResultSet object
- 2) To read data properly from ResultSet object then we need metadata of ResultSet object
- 3) we can get the metadata of ResultSet object by calling `getMetadata();`
i.e. `ResultSetMetadata rsmd = rs.getMetadata();`

a) From ResultSetMetaData object we can read the information about columns by calling the below methods —

(i) getcolumnCount() → It reads no. of columns in resultset

(ii) getColumnName(columnIndex) → It reads name of a column for the given index.

(iii) getColumnTypeName(columnIndex) → It reads a column datatype of a column index.

(iv) getColumnDisplaySize(columnIndex) → It reads columnsize of a column index.

etc ...

Example 19

```
//ResultSetMetaDataTest.java

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;

class ResultSetMetaDataTest
{
    public static void main(String args[]) throws Exception
    {
        Class.forName("com.mysql.jdbc.driver");
        Connection con = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/test",
            "root", "root");
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
ResultSetMetaData rsm = rs.getMetaData();
int count = rsm.getColumnCount();
for(int i = 1; i <= count; i++)
{
    System.out.println("column index = " + i);
    System.out.println("column name = " + rsm.getColumnName(i));
    System.out.println("column datatype = " + rsm.getColumnTypeName(i));
    System.out.println("column size = " + rsm.getColumnDisplaySize(i));
    System.out.println("=====");
}
rs.close();
stmt.close();
con.close();

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Compile and run

```

E:\> javac ResultSetMetadataTest.java
E:\> java ResultSetMetadata

```

10 Sept 15

DatabaseMetadata (interface)

- 1) This interface provides methods for reading metadata of a database.
- 2) we can obtain DatabaseMetadata object by calling getMetadata() of connection interface.

Syntax -

```
DatabaseMetadata dbmd = con.getMetaData();
```

- 3) Some of the methods of DatabaseMetadata of interface are -

- (i) getDatabaseProductName() → It returns database name
- (ii) getDatabaseProductVersion() → It returns version of database
- (iii) getMaxColumnsInTable() → It returns max^m no. of columns allowed in a table.
- (iv) getMaxTableNameLength() → It returns max^m no. of characters allowed in a table name.
- (v) getMaxColumnNameLength() → It returns max^m no. of characters allowed in a column name.

```
Example 20 // DatabaseMetaDataTest.java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.DatabaseMetaData;

class DatabaseMetaDataTest {
    public static void main(String args[]) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "tiger");
        DatabaseMetaData dbmd = con.getMetaData();
        System.out.println("Database product name : " + dbmd.getDatabaseProductName());
        System.out.println("Database version : " + dbmd.getDatabaseProductVersion());
        System.out.println("Max" + " column name length" + " column allowed : " + dbmd.getMaxTableNameLength());
        System.out.println("Max" + " table name length : " + dbmd.getMaxTableNameLength());
        System.out.println("Max" + " column in table : " + dbmd.getMaxColumnsInTable());
        con.close();
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ParameterMetadata (interface)

- 1) This interface provides methods for reading information about index parameter used in a PreparedStatement and CallableStatement object.
- 2) We can obtain ParameterMetadata object by calling `getParameterMetadata()` using PreparedStatement or CallableStatement object.

ParameterMetadata pmd = pstmt. getParameterMetadata();
(or)

ParameterMetadata pmd = cstmt. getParameterMetadata();

- 3) Some of the methods of ParameterMetadata are:
 - (i) `getParameterCount()` → returns no. of parameters
 - (ii) `getParameterTypeName(2)` → returns datatype name of second parameter
 - (iii) `getParameterMode(3)` → returns 3rd parameter mode

Example 21

```
// Parameter metadata Test.java

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ParameterMetadata;
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

class ParameterMetadataTest
{
    public void psvm(String[] args) throws Exception
    {
        Connection con = DriverManager.getConnection("jdbc:
            oracle:thin:@localhost:1521:xe", "system", "tiger");

        PreparedStatement pstmt = con.prepareStatement(" update
            emp set = ? where empno = ? ");

        pstmt.setInt(1, 4000);
        pstmt.setInt(2, 7456);
        pstmt.executeUpdate();

        ParameterMetadata pmd = getParameterMetadata();
        System.out.println(" count : " + pmd.getParameterCount());
        System.out.println(" mode of 2nd parameter : " + pmd.getParameterMode(2));
        pstmt.close();
        con.close();
    }
}

```

SRI RAHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

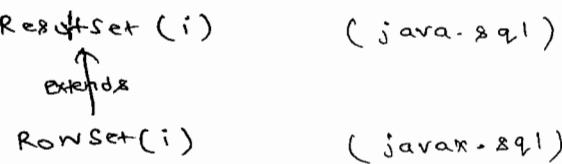
Note

- 1) In the above jdbc program we have not loaded the driver by `Class.forName()` statement. The driver is will be automatically loaded from `ojdbc6.jar` added on the classpath.
- 2) This automatically loading of driver works from `oracle11g`.

1 Sept 15
(my b'day)

RowSet in JDBC

- 1) RowSet is an extension of ResultSet and a Rowset object is used for storing the data selected from a table.



- 2) Rowset is an optional feature, it means a driver vendor may or may not provide implementation (Oracle provided this feature only).
- 3) The following are the differences b/w ResultSet and a Rowset objects -

ResultSet Object	Rowset Object
(i) A ResultSet object is not a Serializable object, so, it can't be transferred across network from one application to another application.	(i) A Rowset object is a serialized object, so, it can be transferred in the network.
(ii) By default a ResultSet object is not scrollable, not updatable. To make it as scrollable and updatable we need to pass type and mode parameters.	(ii) A Rowset object is by default scrollable and updatable. So, we no need to pass type and mode parameters.

(iii) A ResultSet object is always works in connected mode. It means, until we read the data from ResultSet object, connection <u>shouldn't</u> be closed.	(iii) A Rowset object works in both connected and disconnected mode. It means we can read the data from Rowset object even after closing the connection also.
------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------

Q) How to make an object in java as a Serialized object?

Ans- By implementing a class from `java.io.Serializable` interface.

4) Types of Rowset

- 1) JDBCRowset ✓ connected
 - 2) CachedRowset
 - 3) WebRowset
 - 4) JoinRowset
 - 5) FilteredRowset
- } disconnected

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

5) The above 5 types of Rowset interfaces are inherited from Rowset interface.

6) The above 5 types of Rowset interfaces are given in `javax.sql.Rowset` package

7) Oracle ^{Driver} vendor has provided implementation for Rowset interface but MySQL drivers has not provided implementation for Rowset interfaces.

Steps to write a Rowset program

A Rowset program steps differs from normal jdbc program.

- 1) Create an object of implementation class of a Rowset interface.
- 2) Set the properties
- 3) Set the commands
- 4) Execute
- 5) Read the data from a Rowset object
- 6) close Rowset object.

// step-1
Interface → JdbcRowset jrs = new OracleJDBCRowset();
implementation class
of the class
constructor will
load the driver
automatically.
// step-2

jrs.setURL(".....");
jrs.setUsername(".....");
jrs.setPassword(".....");

// step-3

jrs.setCommand("select * from emp");

// step-4

jrs.execute();

// step-5
while(jrs.next())
{
 //
 //
 //
}

// step-6

jrs.close();

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagari Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

12 Sept 15

- 7) In JDBC we have two kinds of Rowset -
 - (i) connected
 - (ii) disconnected
- 8) JDBCRowset is a connected and remaining four are disconnected Rowset.
- 9) JDBCRowset maintains connection until ~~JDBC obj~~ JDBC object is closed.
- 10) The other four types of Rowset closes connection immediately when the command is executed.

Example 21

```
// Program of JDBCRowset

// JDBCRowsetTest.java

import javax.sql.Rowset; JDBCRowset;
import oracle.jdbc.rowset.OracleJDBCRowset;

class JDBCRowsetTest

{
    public static void main(String[] args) throws Exception
    {
        JDBCRowset jrs = new OracleJDBCRowset();
        jrs.setURL("jdbc:oracle:thin:@localhost:1521:xe");
        jrs.setUsername("system");
        jrs.setPassword("tiger");
        jrs.setCommand("select * from emp");
        jrs.execute();
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

while(jrs.next())
{
    System.out.println(jrs.getString(1) + " " + jrs.getString(2) + " " + jrs.getString(3));
    System.out.print(" = = = = = ");
    jrs.absolute(3);
    System.out.println(" The following is a 3rd row... ");
    System.out.println(jrs.getString(1) + " " + jrs.getString(2) + " " +
        jrs.getString(3));
    jrs.close();
}

```

Example 2

```

// program on CachedRowSet
// CachedRowSetTest.java

import javax.sql.rowset.CachedRowSet;
import oracle.jdbc.rowset.OracleCachedRowSet;
class CachedRowSetTest
{
    public static void main(String args[]) throws Exception
    {
        CachedRowSet crs = new OracleCachedRowSet();
        crs.setURL("jdbc:oracle:thin:@localhost:1521:xe");
        crs.setUsername("system");
        crs.setPassword("tiger");
        crs.setCommand("select * from emp");
        crs.execute();
        while(crs

```

SRI KRISHNENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

SOP("Employees in reverse order ...");
crs.last();
while (crs.previous())
{
    SOP(crs.getString(1) + " " + crs.getString(2) + " " +
        crs.getString(3));
}
crs.close();

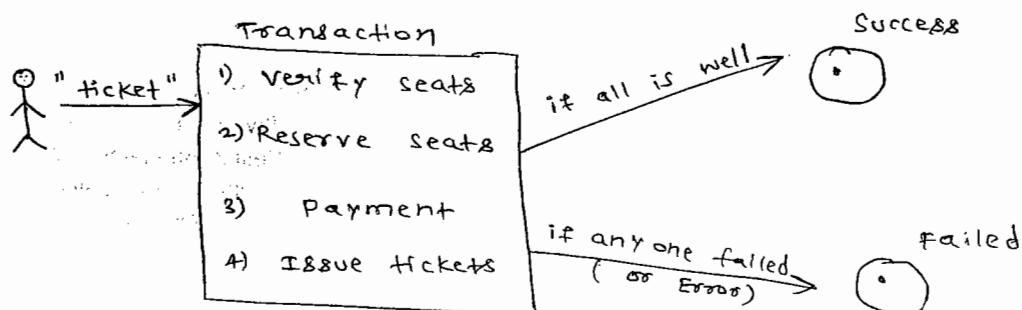
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Transaction Management in JDBC

- 1) A Transaction is a group of operations used to complete a unit of work .
- 2) A Transaction can become success or failed .
- 3) If all operations of a transaction are executed successfully then a transaction becomes success .
- 4) If anyone operation is failed then a transaction is failed .
- 5) A transaction is failed means all the operations of transaction are cancelled .
- 6) For ex, booking a movie ticket is a transaction. In this , a group of operations are verifying the seats , reserving the seats , collecting payment and finally issue the tickets .

In the above operations if any one is failed then transaction failed and if all are executed then a transaction success.



Types of Transaction

- 1) local
- 2) distributed / global / XA
- 3) If all state operations of a transaction is executed on a single database. Then, it is a local transaction.
- 4) If more than one database is involved in a transaction then it is a distributed transaction.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

For ex - Transferring money b/w two accounts of same bank is called local transaction.

Transferring money b/w two accounts of different bank is called distributed transaction.

- 3) JDBC Technology only supports local transaction

- 4) To develop distributed transaction application, we have to use either EJB or Spring Framework.

18 Sept 15

- 5) In JDBC, every ^{SQL} operation executed from java program is a permanent operation because in a JDBC connection, by default autocommit mode is enabled.
- 6) To execute a group of operations as a transaction, autocommit mode must be disabled.
- 7) To disable autocommit mode of connection, we need to call `getAutoCommit()`.
e.g. `con.setAutoCommit(false);`
- 8) If all operations of a transaction are executed without exception then we need to commit a transaction, by calling `commit()`.
- 9) If any exception is occurred during execution of a transaction then rollback a transaction by calling `rollback()`.
- 10) For JDBC transactions, we call the following 3 methods of Connection —
- `getAutoCommit()`
 - `commit()`
 - `rollback()`

Note -

- 1) A transaction can contain insert, update, delete and select commands only.
- 2) Remaining commands like DDL commands are not possible in transaction. ~~are autocommit we can't roll back them~~

Sample code

```

try {
    con.commit();
}

catch ( Exception e )
{
    con.rollback();
}

```

SRI RAJAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Example 23

```

// TransactionTest.java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

class TransactionTest
{
    public (String [] args) throws Exception
    {
        // driver automatically loaded
    }
}

```

```

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

Statement stmt = con.createStatement();

// disable autocommit mode
con.setAutoCommit(false);

try {
    int i = stmt.executeUpdate(" update emp set sal
                                = 9000 where empno
                                = 7456");

    int j = stmt.executeUpdate(" insert into dept
                                values (10, 'Accounting', 'hyd')");

    int k = stmt.executeUpdate(" delete from
                                students where stuid = 101");

    con.commit();
    sop(" transaction committed");
}

catch (Exception e) {
    con.rollback();
    sop(" transaction rollback/ cancelled");
}

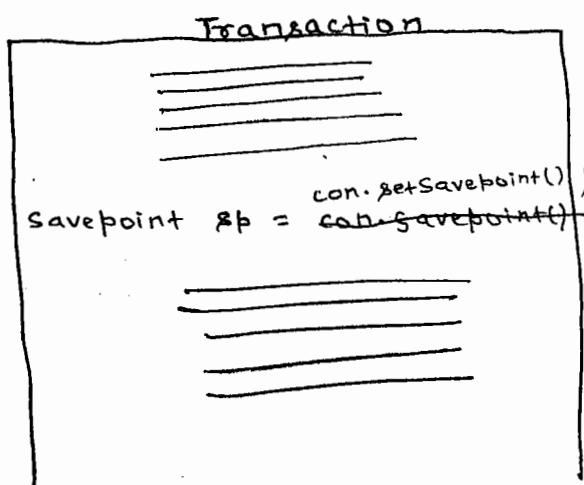
stmt.close();
con.close();
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

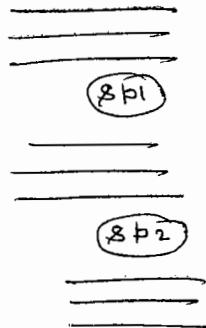
Savepoint

- 1) Savepoint is an interface of java.sql package.
- 2) Implementation class of savepoint will be provided by driver vendor.
- 3) A savepoint is used to logically divide a transaction into two groups.
- 4) If we want to commit a group of operations even though an exception is occurred in another group of operation then we create a savepoint in the middle of two groups of the ~~tran~~ operations of a transaction.
- 5) If an exception is occurred after the savepoint is created then we can rollback only operations after the savepoint, by without cancelling the above operations of savepoint.
- 6) To create a savepoint, we need to call `getSavepoint()` of connection.



Savepoint interface

- 7) A transaction can have multiple savepoint also.
 - 8) If we rollback from first savepoint (sp1) then all operations from sp1 and its below below operations are rollback/ cancelled.



- 9) programatically , if we want to remove a savepoint from middle of the operation then we call `releaseSavepoint()`

Savepoint Example

- i) A Movie-Ticket booking is a transaction with the following operations -

 - (i) verify the seat
 - (ii) Reserve the seat
 - (iii) Payment
 - (iv) Send email
 - (v) Send Message

SKI KAGNENDRA XEROX
Software Languages Material Available
Beside Bangalore Material Available
Opp. CDAC, Balkampet Bakery,
Ameerpet, Hyderabad.

- 2) In the above transaction in the middle of ④th and ⑤th steps a savepoint is created. If an exception is occur by sending a message to mobile then only that message operation is cancelled and the above four method operations are committed and finally a transaction is success.

19 Sept 15

Connecting with MS-Excel

- 1) By using jdbc API we can connect with a work book and we can perform CURD operation on worksheet.
- 2) To connect from a java program to Excel workbook, we need either type1 driver or any vendor supported driver.
- 3) To read the data from a worksheet of MS-Excel we need to follow the below steps -
 - 1) We need to create from a workbook and we need to enter some data in worksheet.
 - 2) We need to create dsn
 - 3) we need to create a jdbc program to read the data from sheet.

- 4) Open Microsoft Excel and enter the following data in sheet1 and save workbook with name products.

Sheet1

Pid	Pname	Price
1101	Sony	7000
1102	LG	5000
1103	Samsung	8000

- 5) To create a dsn, open control panel → Administrative tool → Data sources (ODBC) → System DSN → Add button → select Microsoft Excel driver → Finish.

Enter DSN : xldsn

Click on select workbook button → Select products.xls → unchecked read only → OK → OK → OK

Example 24

//ExcelRead.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
```

class ExcelRead

```
{
    public void main(String[] args) throws Exception
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

SRINIVAS VENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

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

Statement stmt = con.createStatement () ;
ResultSet rs = stmt.executeQuery (" select * from [sheet1$] ");
while ( rs.next () )

{
    sop ( rs.getInt(1) + " " + rs.getString(2) +
          " " + rs.getInt(3));
}

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

```

O/P -

```

C:\> javac ExcelRead.java
C:\> java ExcelRead

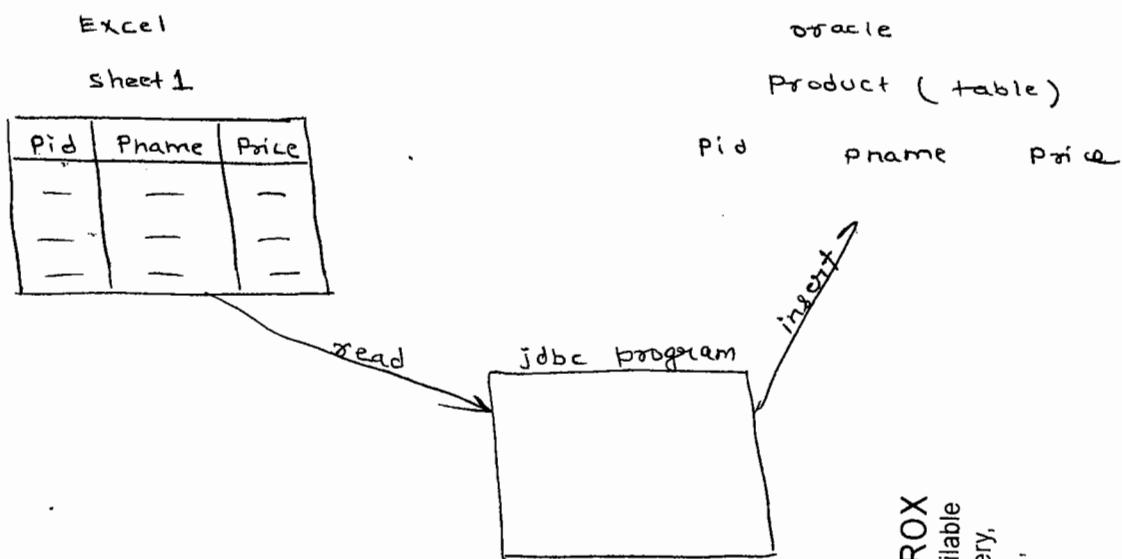
1101 sony 7000
1102 LG 5000
1103 samsung 8000

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Example 25

The following jdbc program is for copying the data from excel worksheet to oracle table.



```
// ExcelOracle.java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

class ExcelOracle
{
    public static void main(String[] args) throws Exception
    {
        // Load Type-1 driver
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        // Type-4 (oracleDriver) automatically loaded.
        Connection con = DriverManager.getConnection ("jdbc:
odbc:xdsn");
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

Connection con1 = DriverManager.getConnection("jdbc:  

        oracle:thin:@localhost:1521:xe",  

        "system", "tiger");  

Statement stmt = con1.createStatement();  

PreparedStatement pstmt = con2.prepareStatement  

        ("insert into product values(?, ?, ?)");  

ResultSet rs = stmt.executeQuery("Select * from [Sheet1$]");  

while (rs.next())  

{  

    pstmt.setInt(1, rs.getInt(1));  

    pstmt.setString(2, rs.getString(2));  

    pstmt.setInt(3, rs.getInt(3));  

    int i = pstmt.executeUpdate();  

}  

System.out.println("Data is copied from Excel to Oracle");  

rs.close();  

stmt.close();  

pstmt.close();  

con1.close();  

con2.close();
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

- Before executing the above program create a table called Product in oracle like the following —

```

CREATE TABLE Product (pid NUMBER(5) PRIMARY KEY  

        BNAME VARCHAR2(10), PRICE NUMBER(5));

```

connecting with a csv file

- 1) By writing a jdbc program we can read a data from a csv file.
- 2) A csv file is a text file, in which the data is entered in a rows and columns format by separating with a comma (,)
- 3) A csv file is created by notepad or edit plus .
- 4) We need to follow the below steps —
 - (i) Create a csv file
 - (ii) Create a dsn
 - (iii) Create a jdbc program to read the data from csv file.
- 5) Open notepad, Enter the data and save it with customers.csv

cid , cname , address
101 , abcd , hyd
102 , dcba , delhi
103 , beda , chennai

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 5) To create a dsn open control panel →
 Administrative tools → Data sources → System DSN →
 Add button → Select Microsoft Access Text driver →
 Finish → Enter DSN : csvdsn → unchecked
 Use current directory → click select directory
 button → select the drive → OK → OK → OK

Example 6

```
// CSVRead.java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

class CSVRead

{
  public static void main (String [] args) throws Exception
  {
    Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection
      ("jdbc:odbc:csvdsn");
    Statement stmt = con.createStatement ();
    ResultSet rs = stmt.executeQuery ("select * from
      customers.csv");
    System.out.println (rs.getInt(1) + " " + rs.getString(2) +
      " " + rs.getString(3));
    rs.close ();
    stmt.close ();
    con.close ();
  }
}
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

compile and run

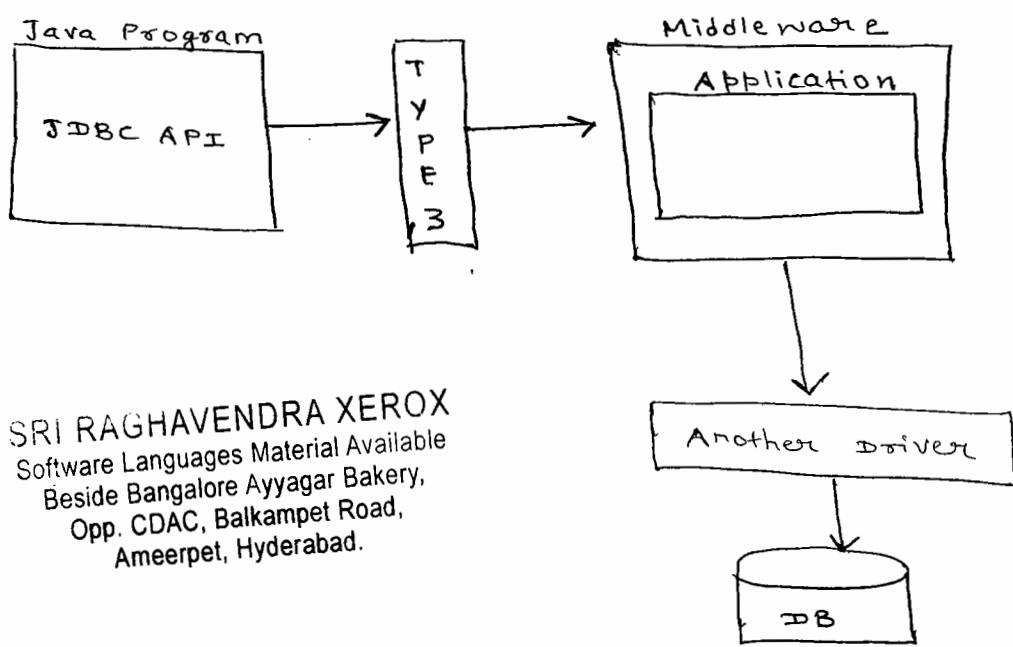
```
E:\> javac CSVRead.java  
E:\> java CSVRead  
101 abcd hyd  
102 dcba delhi  
103 beda chennai
```

27 Sept 15

(sunday topic) Type - 3 Net Protocol pure Java driver

- 1) Type-3 driver is a pure Java driver. It means the driver is 100% implemented in java.
- 2) Type-3 driver doesn't connect a java program with a database. It connects java program with a middle-ware server and middle-ware server connects with database.
- 3) In type1, type2 and type4 drivers the completely driver SW is loaded into client side JVM. So, they increase the burden on client-side JVM. These type1, type2 and type4 are called heavy weight driver.
- 4) Type-3 driver is loaded at client-side JVM only to connect with middle-ware server. So, it is a light weight driver.

- 5) A middleware server contains an Application, it uses another jdbc driver to connect with database.
- 6) Only type-3 driver follows 3 tier Architecture.
The three tiers are -
- i) java Application / program
 - ii) Middleware server
 - iii) database



- 7) While working with type1 or type2 or type4 drivers, if we want to communicate with database across network then in the server system we need to disable the firewall and we need to enable the server port.

- 8) In case of type-3 driver, java program connects with middleware server not with database directly. So, no need to disable the firewall and enable the port.

Advantages (Pos)

- 1) Type3 driver is platform independent and database independent.
- 2) We can use type3 driver in Applets because Applets can only use pure-java driver.
- 3) Type3 drivers are light weight.
- 4) We no need to disable firewall and enable the port.

Drawback (cons)

- 5) We need to separately install middleware server.

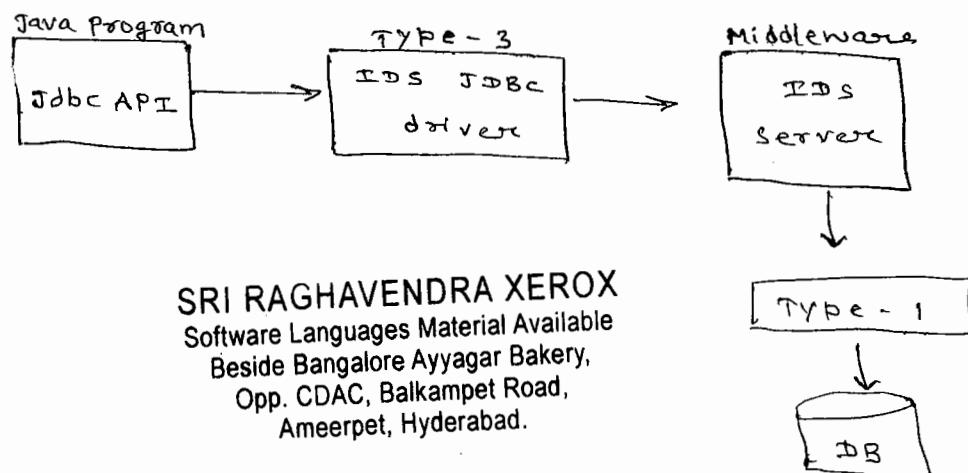
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

IDS JDBC Driver

- 1) It is a type3 JDBC driver, which comes with IDS server.
- 2) We need to install IDS server as a middleware b/w java program and database.
- 3) We can download trial version of IDS server from -

<http://www.idssoftware.com/download.html>

- 4) An .exe will be downloaded and when we install the server then automatically server will be started.
- 5) We can find whether a IDS server started or not from services tool of Administrative tool.



Program

Example 27 In the following jdbc program we are reading the data of excel with type-3 driver

Step 1

- 1) Create a worksheet with the following data by opening excel and save workbook with name as demo

Sno	sname	marks
111	aaa	500
222	bbb	600
333	ccc	700

Step 2

- 1) create a dsn like the following -
- 2) open control panel → Administrative Tools → Data Sources → System DSN → add button → select Microsoft Excel Driver → Finish → Enter dsn : **Excel** → click on select workbook → Select Demo → OK → OK → OK

Step 3

write the jdbc program -

class Type3

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

{ public void main( String [] args ) throws Exception
{
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    Connection con = DriverManager.getConnection
        ("jdbc:odbc:localhost:12/con?dsn='Excel'");
    System.out.println("connection is opened");

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery ("select *"
                                    "from [sheet1$]");

    while( rs.next() )
    {
        System.out.println( rs.getInt(1) + " " + rs.getString(2) + " "
                           + rs.getInt(3));
    }

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

```

Compile and Run

```

D:\> javac Type3.java
D:\> set classpath = c:\classserver\classes\.
D:\> java Type3

```

Types	Database Independent	Platform independent
Type1	Yes	No
Type2	No	No
Type3	Yes	Yes
Type4	No	Yes

VENNDRA XEROX
 Beside Bangalore Material Available
 Opp. CDAC, Balkampet Bakery,
 Ameerpet, Hyderabad.

Datasource interface in jdbc

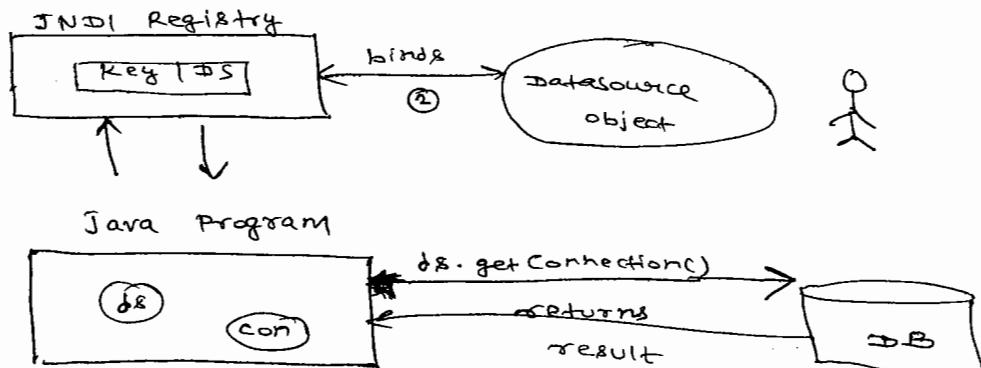
- Q) In how many ways we can get connection in java with database ?

Ans - 2 ways —

- DriverManager class (java.sql.*)
 - Datasource interface (javax.sql.*)
- With the release of JDBC technology, DriverManager class is introduced to get a database connection.
- In the industry, some drawbacks are identified while working with DriverManager class. In order to overcome the drawbacks of DriverManager, Datasource interface is introduced in JDBC 2.0 version.
- Now Today, we have two options to get a Database connection —
- DriverManager class (java.sql)
 - Datasource interface (javax.sql)

- 4) With `DriverManager` class the following drawbacks are identified —
- To get a connection, a programmer has to type connection properties in source code, it means a programmer should remember the connection properties, so, a programmer burden is increased.
 - `DriverManager` class takes around 3 to 4 sec in the network to get the connection with database. So, there is a performance drawback.
 - A connection returned by the `DriverManager` is a Non-reusable connection. It means we can't preserve/store the connection for reusing it in other connections.
- 5) All the above drawbacks of `DriverManager` are solved with `Datasource` mechanism of JDBC.
- How Datasource reduces burden on programmer?
- In a `Datasource` mechanism, work is divided between Administrator and programmer.
 - Administrator creates a `Datasource` object and then binds it with JNDI Registry (Java Naming and Directory interface).
 - A java program connects with registry, retrieves a `Datasource` object from registry and then gets a connection with a database.

4) A programmer is no need to remember connection properties of driver. So, a programmer burden is reduced.



5) A datasource used in distributed Transactions.

Q) In JDBC how many types of connections are there?

Ans - 2 types -

- (i) non-reusable connection (Connection interface)
- (ii) reusable connection (Pooled Connection interface)

DataSource (interface)



(implementation class)

OracleDataSource (class)

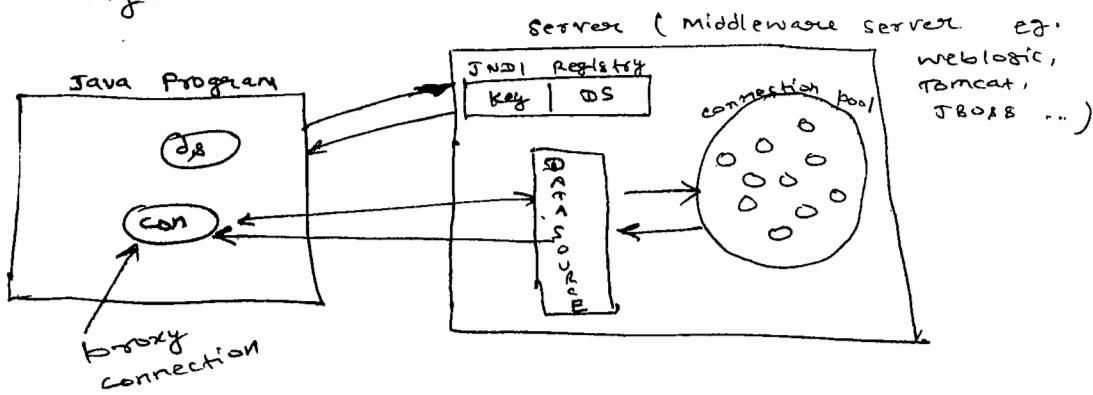
Vendor

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

connection Pooling

- 1) Database connections are expensive objects, if connections are opened and closed again and again then it will increase the burden on database server.
- 2) In order to reuse database connections, instead of recreating them again and again, connection Pooling Technique is used.
- 3) In connection pooling technique, a group of reusable connections are opened and stored in a connection pool.
- 4) With the help of connection pooling, the burden on a database server is reduced.
- 5) In real-time projects a Server Administrator / Team Leader configures a connection pool at server side, in order to create reusable connections, Connection Pool Datasource interface is used.
- 6) A Connection Pool Datasource returns pooled connections (reusable connections).
- 7) In jdbc, we have 2 types of connection
 - (i) Non-reusable connection indicated with Connection interface
 - (ii) Reusable connection, indicated with Pooled Connection interface.

- 8) while creating a connection pool, an administrator at server side will apply some settings like min capacity, max capacity and capacity increment.
- 9) If the no. of clients asking for a connection are more than max^m capacity then the remaining clients (Excess clients) has to wait until one of the connection in pool becomes free.
- 10) A connection pool contains all equal objects, it means, from 1st object to the last object in the pool, they are exactly same. So, we can call the objects in pool as stateless object.
- * 11) A connection pool can store all connections to same database only. It means a pool can't store connections of different databases. For ex, if a connection pool is created for oracle, it stores all connections to oracle only.



- 12) For each connection pool, Administrator configures a datasource object, to get connection from pool by java programs.
- 13) Administrator will store / bind the datasource object in JNDI Registry. A java program first reads datasource object from registry and then it obtains a connection from pool using datasource object.
- 14) In the middle, datasource object creates a logical connection (proxy connection) for a pooled connection and returns that logical connection back to the java program.

Q) How connection object is again going back to pool?

Ans - whenever proxy connection is closed / connection is sitting idle then one event is fired, connection Event call connection Listener informs dg, ds sends real connection back to pool.

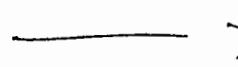
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 15) In our java program, after finishing database work, we close proxy connection. When proxy connection is closed then internally ConnectionEvent occurs and it calls ConnectionListener. The ConnectionListener will inform the Datasource object to send a pooled connection (or) connection back to the pool.
- 16) In order to configure (or) create a connection pool, we need a middleware server like Tomcat (or) Weblogic.

Example

In the following example, we are not creating any connection pool, but we are opening one pooled connection and a proxy connection (logical connection) for the pooled connection.

```
import oracle.jdbc.pool.OracleConnectionPoolDataSource;  
import java.sql.*;  
import javax.sql.PooledConnection;  
  
class PooledConnectionTest  
{  
    public void psuM( String args[] ) throws Exception  
    {  
        →  
    }  
}
```



```
// This class is implementation of Connection PoolDataSource  
interface
```

```
OracleConnectionPoolDataSource ds = new OracleConnectionPool  
DataSource();  
ds.setURL("jdbc:oracle:thin:@localhost:1521:XE");  
ds.setUser("system");  
ds.setPassword("tiger");
```

```
// open pooled connection
```

```
PooledConnection pc = ds.getPooledConnection();
```

```
// open logical connection
```

```
Connection con1 = pc.getConnection();
```

```
Statement stmt = con1.createStatement();
```

```
stmt.executeUpdate("create table emp(id  
number(6));");
```

```
SOP("table created");
```

```
stmt.close();
```

```
con1.close();
```

```
// open another logical connection
```

```
Connection con2 = pc.getConnection();
```

```
Statement stmt2 = con2.createStatement();
```

```
stmt2.executeUpdate("insert into emp values(10);")
```

```
SOP("row inserted");
```

```
stmt2.close();
```

```
con2.close();
```

```
pc.close();
```

```
Y
```

```
X
```

SRI RAJAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

20/9/15
(sunday)
missed topic

Date conversion in database

Inserting a date in database -

- 1) A java program reads a date from either command prompt or from html page in the form of string, but every database accepts date in its own format. so, there is need for conversion.
- 2) Read a date from end user or string.
- 3) By using converter, convert date in string format to java.util.Date object
- 4) convert java.util.Date to java.sql.Date.
- 5) set java.sql.Date to insert command and then execute the command.

Pseudocode

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
java.util.Date d = sdf.parse(str);
    ↴ convert to milliseconds (ms)

long ms = d.getTime();

java.sql.Date d = new java.sql.Date(ms);
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example The following jdbc program will insert employee number, name and DOB to employee table —

```
SQL> create table employee  
      ( empno number(5) primary key,  
        ename varchar(20),  
        dob date  
    );  
  
// DateInsert.java  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.util.Scanner;  
import java.text.SimpleDateFormat;  
  
class DateInsert  
{  
    public static void main(String[] args) throws Exception  
    {  
        // driver loaded automatically (oracle 11g)  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost:  
             1521:xe", "system", "tiger");  
        PreparedStatement pstmt = con.prepareStatement  
            ("insert into employee values  
             (?, ?, ?)");  
        Scanner s = new Scanner(System.in);  
        pstmt.setInt(1, Integer.parseInt(s.nextLine()));  
        pstmt.setString(2, s.nextLine());  
        pstmt.setDate(3, new SimpleDateFormat("yyyy-MM-dd").  
                     parse(s.nextLine()));  
        pstmt.executeUpdate();  
    }  
}
```

VENKATESWARA XEROX
Software Languages Material Available,
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

        System.out.print(" Enter empno");
        int empno = s.nextInt();
        System.out.print(" Enter ename");
        String ename = s.next();
        System.out.print(" Enter DOB (dd/MM/yyyy)");
        String dateInString = s.next();
        //converting string to java.util.Date
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        java.util.Date utilDate = sdf.parse(dateInString);
        //convert java.util.Date to java.sql.Date
        long ms = utilDate.getTime();
        java.sql.Date sqlDate = new java.sql.Date(ms);
        //Set the values to PreparedStatement
        pstmt.setInt(1, empno);
        pstmt.setString(2, ename);
        pstmt.setDate(3, sqlDate);
        int i = pstmt.executeUpdate();
        System.out.println(i + " rows inserted");
        pstmt.close();
        con.close();
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Opp. CDAC, Ayyagar Bakery,
 Ameerpet, Hyderabad.

compile and run -

E:\> javac DateInsert.java

E:\> java DateInsert

Enter empno

1101

Enter ename

SCOTT

Enter DOB (dd/MM/yyyy)

22/04/1970

1 row inserted.

SQL> select * from employee ;

EMPNO	ENAME	DOB
1101	SCOTT	22-APR-70

Note-

- 1) Date format in oracle is dd-MM-yy
- 2) If we execute the above program in MySQL database then date will be inserted as 1970-04-22 because MySQL date format is yyymm - dd

SRI RAGHAVENDRA XEROX
Available
SRI RAGHAVENDRA Material Bakery,
Software Languages Ayyagar Road,
Beside Bangalore Opp. CDAC, Balkampet.
Opp. Amerpet, Hyderabad.

Reading the date from database

- 1) When we select the date from database then we will get date in database format. We can convert it into string format.
- 2) For converting a Date into string Format we use a converter called SimpleDateFormat
- 3) SimpleDateFormat class has two important methods —
 - (i) parse(); // String → date
 - (ii) format(); // Date → String

// DateSelect.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;

class DateSelect
{
    public static void main(String args[]) throws Exception
    {
        // driver automatically loaded
        Connection con = DriverManager.getConnection("","","");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select
            dob from employee where
            empno = 1101");
    }
}
```

SRI RAUNAVENDRA XEROX
SRI RAUNAVENDRA XEROX Available
Software Languages Material Bakery,
Beside Bangalore Ayagar Road,
Opp. CDAC, Balkampet.
Opp. Ameepet, Hyderabad.

```
// move cursor to next row  
rs. next();  
  
java.util.Date utilDate = rs.getDate(1);  
SimpleDateFormat sdf = new SimpleDateFormat  
("EEEE, dd, MMMM yyyy");  
  
String str = sdf.format(utilDate);  
System.out.println(str);  
  
stmt.close();  
con.close();  
}  
}
```

O/P

E:\> javac DateSelect.java

E:\> java DateSelect

Wednesday 22, April 1970

SRI RAHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Web Application -

- 1) A web application is a server side application, it runs on server and it provides the services to multiple clients across the network.
- 2) Another definition is a, web application is a group of passive and active resources.
- 3) Web Applications are of two types -
 - (i) static
 - (ii) dynamic
- 4) A static web Application will produce same response to all the clients.
- 5) A dynamic web Application generates response based on the input given by a client.
- 6) Static web Applications are created by using client side web technologies.
- 7) Dynamic web Applications are created by using client side and server side web technologies.
- 8) Client side web technologies are : HTML, CSS, Javascript.
- 9) Server side web technologies are : Servlet, JSP, ASP, PHP .. etc.

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 10) A passive resource means it is a file which is executed on a browser.
- 11) Active resources means, it is a file executed on server.
- 12) For example, HTML files, images, CSS ... etc are passive resources because all these files are executed on the browser.
- 13) For example, A servlet, JSP or ASP files ... etc are active resources because they are executed on server.

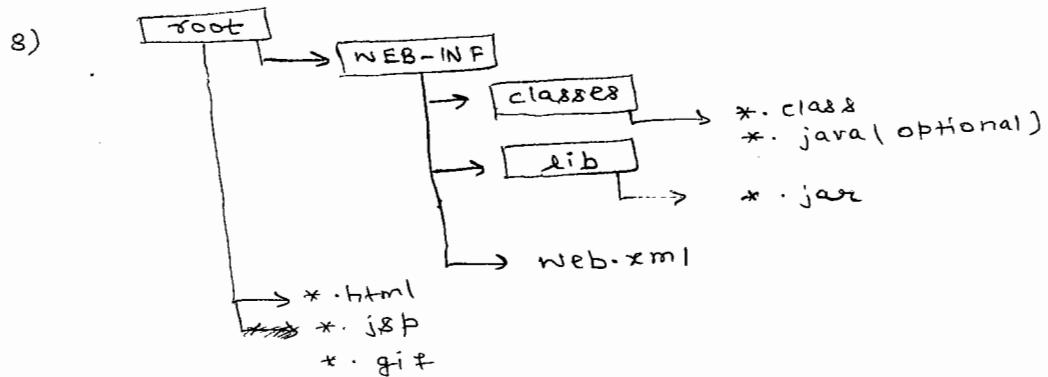
Web Application directory structure

- 1) To create a web application in java, we need to follow a standard directory structure given by Sunmicrosystem.
- 2) If we follow a directory structure then only a server recognize our application as a java web application.
- 3) If we follow the directory structure then our web application becomes server independent.

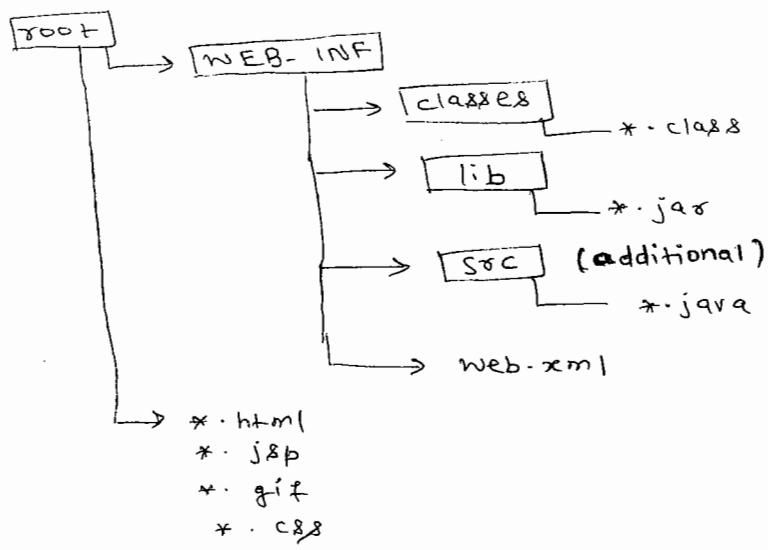
Note - In java we have two types of servers

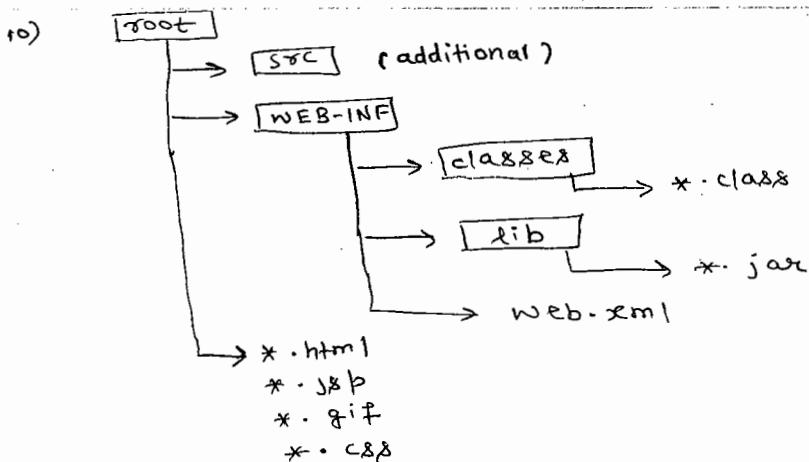
- (i) Web server - For ex, Tomcat, Resin
- (ii) Application server - For ex, Weblogic, JBoss, Glassfish.

- 4) for every web application there must be one and only one root directory. the root directory name can be given as any name.
- 5) under root directory we need to create a sub-directory with name **WEB-INF**
- 6) under WEB-INF we need two sub-directories **classes** and **lib**.
- 7) In WEB-INF sub directory we save deployment descriptors (web.xml)



- 9) we can also add additional folders to the WEB-INF and root directory





Note- ↳ we can't create additional directories under classes and lib directories.

↳ we have to follow directory structure, while developing web applications using frameworks like struts, spring, JSF etc also

JSF : Java Server Pages

8 | Aug | 15

Web container / Servlet container / JSP container
Servlet Engine / JSP Engine

- ↳ In JAVAEE we have 2 types of container
- Web container
 - EJB container
- ↳ A web container has multiple synonyms
- Servlet container
 - JSP container
 - Servlet Engine
 - JSP Engine

- 3) A container means, it is a program, which provides runtime support for other programs.

Ex1

class A

```
{ void m1 ()  
    {  
        }  
    }
```

class B

```
{ psvm ( string k[] )  
    {  
        A a = new A ();  
        a.m1();  
    }  
}
```

i) In this example, we can't directly run class A. Because it doesn't contain main().

ii) class B contains main() and it is providing the runtime support for class A. so, class B is a container.

Ex2

In AWT (Abstract windowing Toolkit), A Frame is a class which provides runtime support for other classes like Button, TextField etc. so, Frame is a container

- 4) In web Application, we create servlets, servlets are classes without main().
- 5) In a server, to load and execute the ~~server~~ servlet classes, there is a predefined program called web container or servlet container.

TYPES OF servlet containers -

- 1) A servlet container is a predefined program provided by vendors.
- 2) servlet containers are 3 types -
 - (i) stand Alone container
 - (ii) In process container
 - (iii) Out process container
- 3) A standAlone container means, a server and container both are developed as a single program.
- 4) A Inprocess container means, server and container both are developed as a separate program but container program also runs within the same process of server.
- 5) A outprocess container means, server and container are separate program and both runs in a separate process.

10 AUG 15

Actually TOMCAT is a container. If we download tomcat from Apache website then tomcat acts as a server also. Tomcat is an example for standalone container.

JBOSS is a server, it uses tomcat as a web container. When JBOSS server started then in the same process tomcat container also runs. So, here Tomcat is a in-process container.

We can install Apache web server and Tomcat server separately and we can integrate both processes by using a plugin. Here, Tomcat is a out process container.

(Q) Why servlet?

Ans-(1) Before servlet technology there was CGI (common gateway interface) technology for creating dynamic web applications.

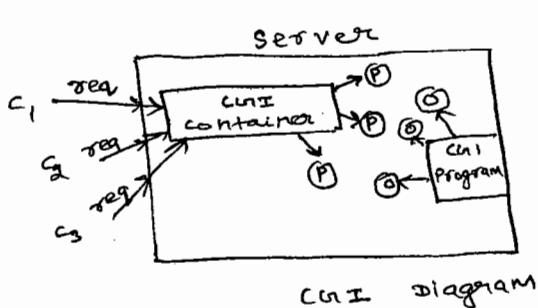
(2) CGI technology was released by NCSA organization

NCSA → National center for supercomputing Application

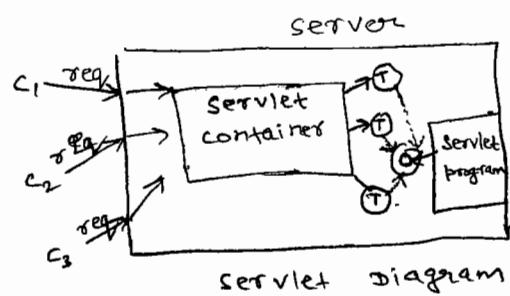
(3) In CGI technology, a server provides CGI container and that container will provide runtime support for CGI programs.

(+) For every request, CGI container starts a process and creates an object for the CGI program.

- (5) If concurrently multiple clients are sending request then CII container starts multiple processes and creates multiple objects for CII program. These multiple processes and objects increases burden on server.
- (6) When burden on server is increased, its performance is decreased and at some point of time server is going to be crashed.
- (7) To overcome drawbacks of the CII technology Sun microsystem introduced servlet technology to develop dynamic web applications.
- (8) Servlet technology follows thread model (light weight), not process model (heavy wt) so, it reduces the burden on server.



P = Process
O = Object



T = Thread
O = Object

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

11 AUG 15

Q) What are the differences b/w CGI and Servlet technology?

Ans -

CGI	Servlet
1) CGI programs are written in PEARL (Practical Extraction and Reporting language). PERL is a platform dependent language. So, CGI is a platform dependent technology.	1) Servlet programs are written in Java. Java is a platform independent language. So, servlet is a platform independent technology.
2) CGI technology follows process model. So, it is a heavy weight technology.	2) servlet follows thread model. So, servlet is a light weight technology.
3) CGI is a non scalable technology. It means, when no. of clients are increasing then it decreases performance of a server.	3) servlet is a scalable technology. It means, even though no. of clients are increasing but it doesn't decrease the performance of a server.

RAVEENDRA XEROX
 1st Floor, Material Available
 Software Languages Ayyagar Bakery,
 Beside Bangalore Road,
 Opp. CDAC, Balkampet Road.
 Opp. CDAC, Hyderabad.

Q- what is a servlet?

- Ans - 1) A servlet is a java class , it runs on server side and it ~~in~~ provides services to all the clients across the network.
- 2) A servlet is a web component , it creates dynamic web pages.
- 3) A servlet is a class , it extends the functionality of a server .

Q) what is the difference b/w an Applet and a servlet ?

- Ans - 1) Applets runs on browser and increases the browser functionality. But, servlets runs on web server and increases server functionality .
- 2) An Applet has a size , so it is visible. But servlet doesn't have size . So , it is not visible on server .

Q) Why we create servlets ?

- Ans - we create servlets , to create dynamic web pages and we create html pages to create static web pages .

How to create servlet ?

- 1) Servlet technology has provided 3 ways to develop the servlet classes in a project -
 - (i) By implementing our class from servlet interface.
 - (ii) By extending our class from ~~GenericServlet~~ generic servlet abstract class.
 - (iii) By extending our class from HttpServlet abstract class.
- 2) Servlet technology contains Servlet API and servlet API contains 3 packages -
 - (i) javax.servlet
 - (ii) javax.servlet.http
 - (iii) javax.servlet.annotation
- 3) The above 3 packages contains interfaces and classes to develop servlets -

Developing servlet class -

1st way -

```
public class MyServlet implements servlet
```

```
{  
    ==  
    }  
    ==
```

SRI RAJAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

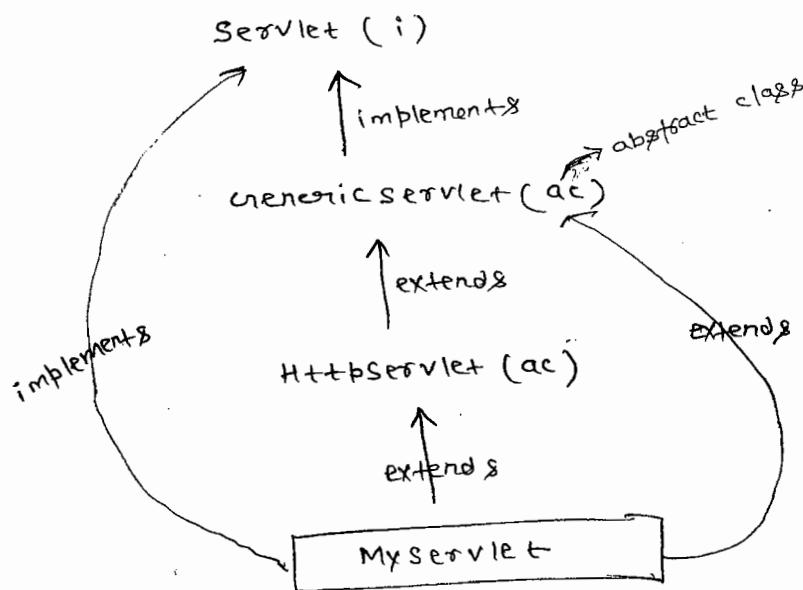
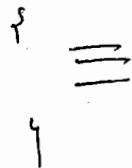
2nd way -

```
public class MyServlet extends genericServlet
```

```
{  
    ==  
    }  
    ==
```

3rd way

```
public class MyServlet extends HttpServlet
```



12 AUG 15

- * Servlet is a class which implements servlet interface directly or indirectly.

- a) Every programmer created servlet class either directly or indirectly implements servlet interface.
- b) generic servlet is an abstract class because it has one abstract method.
- c) HttpServlet is also an abstract class, but it doesn't contain abstract methods.

- 7) In java, if a class contains atleast one abstract method then that class is abstract class. But, an abstract class may or may not contain abstract methods.
- 8) While developing our servlet classes, mostly we extend either GenericServlet or HttpServlet, but we don't implement servlet interface directly because, in servlet interface, there are 5 abstract methods and we need to override all the 5 abstract methods. So, it is a burden on developer.

Servlet Life cycle

- 1) Servlet is a java class and every java class has a life cycle.
- 2) In a life cycle of a java class, there are 5 stages
- doesn't exist
 - Instantiated
 - Initialized
 - service
 - destroyed
- 3) A normal java class life cycle is like the following -

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

public class A
{
    private int x;
    public A()
    {
        x = 100;
    }
    public void m1()
    {
        x = 100;
    }
    public void m2()
    {
        x = 100;
    }
}

```

```

public class Main
{
    public static void main(String args[])
    {
        A a = new A();
        a.init();
        a.m1();
        a.m2();
    }
}

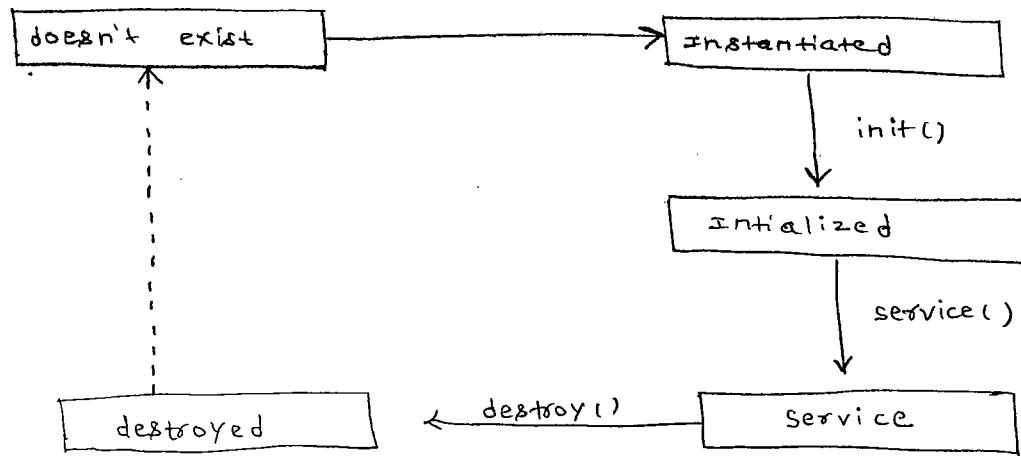
```

doesn't exist ①
Instantiated ②
Initialized ③
Service ④
destroyed ⑤

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

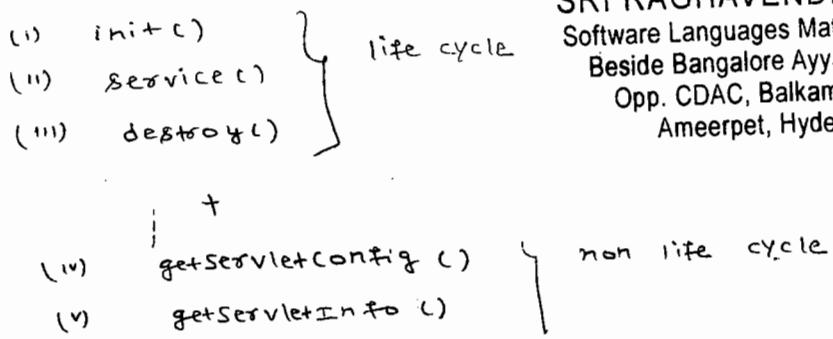
13 AUG 15

- 1) A servlet class life cycle also contains same life cycle stages like an ordinary java class.
- 2) A servlet life cycle will be managed by another predefined program called servlet container.
- 3) Managing the life cycle of servlet means, container creates an object of servlet, calls life cycle methods of servlet and finally destroys that servlet object.



- 4) By default, a servlet container creates an object of servlet class, when a first request is arrived.
- 5) In some applications, we need a servlet object in server immediately when a servlet class is deployed. In this case a servlet container creates servlet class object at a deployment time only.
- 6) After creating the object container calls init() to initialise a servlet object.

- 7) For each request, container calls service() of servlet. This service() will read input, execute the logic and then generates output.
- 8) Finally, when a servlet object is no more required in a server then container will remove that object from server by calling destroy().
- 9) We call init(), service() and destroy() methods as a life cycle methods of servlet.
- 10) These life cycle methods of servlet ~~are~~ is given by javax.servlet.Servlet interface.
- 11) Actually javax.servlet.Servlet interface has 5 abstract methods. We call 3 methods of them as life cycle and remaining two as non life cycle methods -



- 12) A servlet container calls init() and destroy() for once and service() for each request.

Syntax of life cycle methods

(1) public void init (ServletConfig config) throws ServletException

```

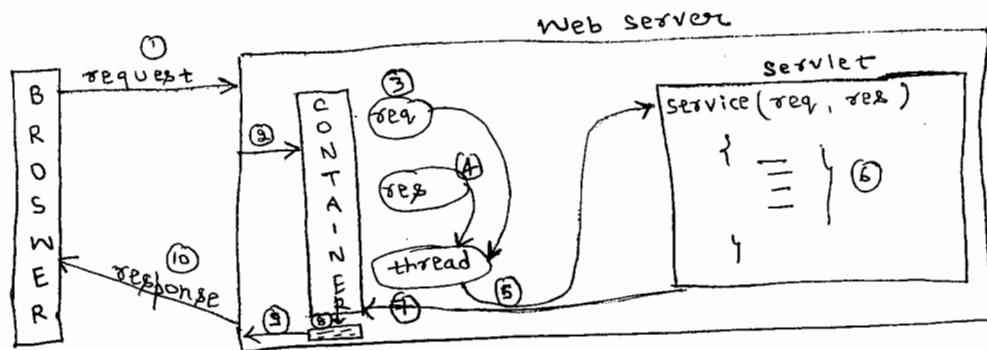
  {
  ===
  }
  
```

(ii) public void service (ServletRequest req , ServletResponse res)
throws ServletException , IOException

(iii) public void destroy ()

Q) How a servlet container calls service () ?

Ans -



- 1) A client sends a request to a servlet from browser.
- 2) A server stops the request and checks whether a request is sent ~~for~~ static or dynamic resource? servlet is a dynamic resource, so, server forward request to container.
- 3) container creates request, response objects and thread also.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 4) Container assigns request and response object to thread.
- 5) Container start the thread . start() calls run() and then run() method then calls service() of servlet by passing request object and response object as parameter.
- 6) Service() reads input from request object , executes the logic and writes the output into response object .
- 7) A container collects response object from servlet .
- 8), 9) A container converts response object into a web-page and then it transfers that web page to server .
- 10) Finally, a server will send a web-page as a response to the browser .

Note:- At server side , a container removes request, response and thread objects .

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

14 AUG 15

Servlet configuration -

- 1) In java terminology, if we write information about a class in another file then it is called configuration.
- 2) In web application, a servlet container will create and manages servlet object.
- 3) In order to tell a servlet container about information of our servlet class, we need to configure a servlet in web.xml.
- 4) we call web.xml as a deployment descriptor file.
- 5) To configure a servlet in web.xml, Sunmicrosystem has provided two tags -
 - (i) < servlet >
 - (ii) < servlet-mapping >
- 6) while configuring a servlet in web.xml, we need to write three names of a servlet -
 - (i) alias name
 - (ii) fully qualified class name
 - (iii) url pattern
- 7) The basic rule of writing web.xml any XML file are -
 - (i) XML is a case sensitive language.
 - (ii) XML file contains only one root tag.
 - (iii) Under the root tag there can be many parent tags and a parent tag can have many child tags.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- (iii) Every open tag should have respective closing tag.
 - (iv) If a parent tag is open and a child tag is opened then at the time of closing first we need to close the child tag then we need to close parent tag.
 - (v) Attribute value must be in either in single quote or in double quotes.
- 8) A servlet container creates object of a servlet and executes life cycle methods. But, by default a container don't know a request is given for which servlet by the client.
- 9) A web Application can contain multiple servlet and a container doesn't know which request is given for which servlet. So, we need to tell the container about this mapping by using three names —
- i) when a request comes from the browser, in that request URL-pattern exists. Now, in the container verifies URL-pattern request, is matched with URL-pattern in web.xml or not

- 2) If matched then container reads alias-name and goes to <servlet> tag and verifies alias name is matched in that tag or not.
- 3) If matched then container will call that servlet class.

For example

```

<web-app>
  <servlet>
    <servlet-name> debit </servlet-name>
    <servlet-class> P1. DebitCardServlet </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> debit </servlet-name>
    <url-pattern> /dc </url-pattern>
  </servlet-mapping>
</web-app>

```

Browser : http://localhost : 8080 / App1 / dc ←

Note -

One web application contains exactly one web.xml file only and filename must be web.xml only.

AVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

15 AUG 15

Apache Tomcat

Type : web server

Vendor : Apache software Foundation

Version : Tomcat 8.0 (compatible with jdk8)

Tomcat 7.0 (compatible with jdk7)

Default : 8080

HTTP port

Location : C:\Program Files\Apache Software Foundation
 \Tomcat 7.0

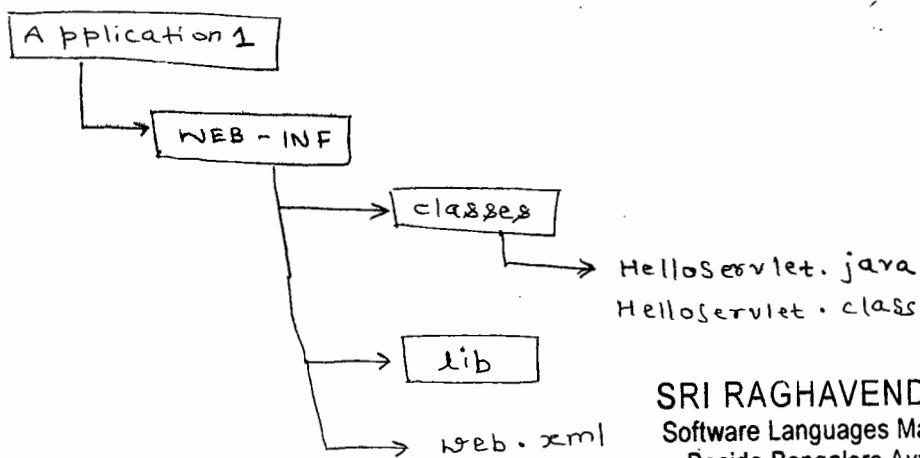
- 1) If oracle database is installed then along with oracle, oracle http service is also installed with port no. 8080.
- 2) If Tomcat server is also installed on the same port 8080 then clash occurs. So, we need to change the tomcat http port.
- 3) To change HTTP port no. of TOMCAT, go to C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf directory and open for server.xml.
- 4) Go to <Connector> tag and change port from 8080 to 2015.
- 5) We can start the tomcat in following 3 ways -
 - (i) Go to Tomcat 7.0\bin directory and start Tomcat7.exe
 - (ii) Go to Tomcat 7.0\bin directory and start (click)
 - (iii) Window start button → All programs → Apache Tomcat7 → Monitor tomcat → A button comes in system icons today → Right click on button → Start service

Note:

- 1) When tomcat server is installed then it will be added to system services and it will be automatically started.
- 2) Go to Administrative tools → services → Select Apache Tomcat and right click and select properties and change start up type as manual.

Steps for creating a servlet program -

Step 1) Create directory structure.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Step 2) Create HelloServlet.java source code -

- (i) We are extending our servlet class from genericServlet.
- (ii) genericServlet is an abstract class and it contains one abstract method call service so, we must override service().
- (iii) In order to write output into response object we need PrintWriter class object.
- (iv) Creating PrintWriter class object, to write an output into response object is a complex task. So, already a factory method is given to return PrintWriter object called getWriter().

(v) We use the following statement to read/get the PrintWriter object -

```
PrintWriter out = response.getWriter();
```

17 AUG 15

Example

```
// HelloServlet.java

import javax.servlet.GenericServlet;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;

public class HelloServlet extends GenericServlet {
    public void service(ServletRequest request,
                        ServletResponse response) throws
    ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<H1> Hello, welcome to servlet </H1>");
        out.close();
    }
}
```

Step 3> Configure servlet in deployment descriptor (web.xml)

```
// web.xml
<web-app>
    <servlet>
        <servlet-name> Hello </servlet-name>
        <servlet-class> HelloServlet </servlet-class>
    </servlet>
</web-app>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
< servlet-mapping >  
  < servlet-name > Hello < /servlet-name >  
  < url-pattern > /servlet < /url-pattern >  
</servlet-mapping>  
</web-app>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Step4 compile the servlet -

- 1) java compiler only knows about java api, but it doesn't know servlet api. so, compiler generates error.
- 2) To resolve the errors we need to add a jar file which contains servlet-api with classpath variable.
- 3) In Tomcat server, servlet-api.jar contains servlet-api so, we need to add the servlet-api.jar to the classpath variable.

```
E:/Application1/WEB-INF/classes> set classpath =  
"C:\Program Files\Apache Software Foundation\Tomcat7.0\  
lib\servlet-api.jar";  
^ current working directory
```

```
E:/Application1/WEB-INF/classes> javac HelloServlet.java
```

Step5 Deploy the Application

- 1) copy the root directory (Application1) into webapps folder.

Location :

C:\Program Files\Apache Software Foundation\Tomcat7.0\
webapps folder. (Hard deployment)

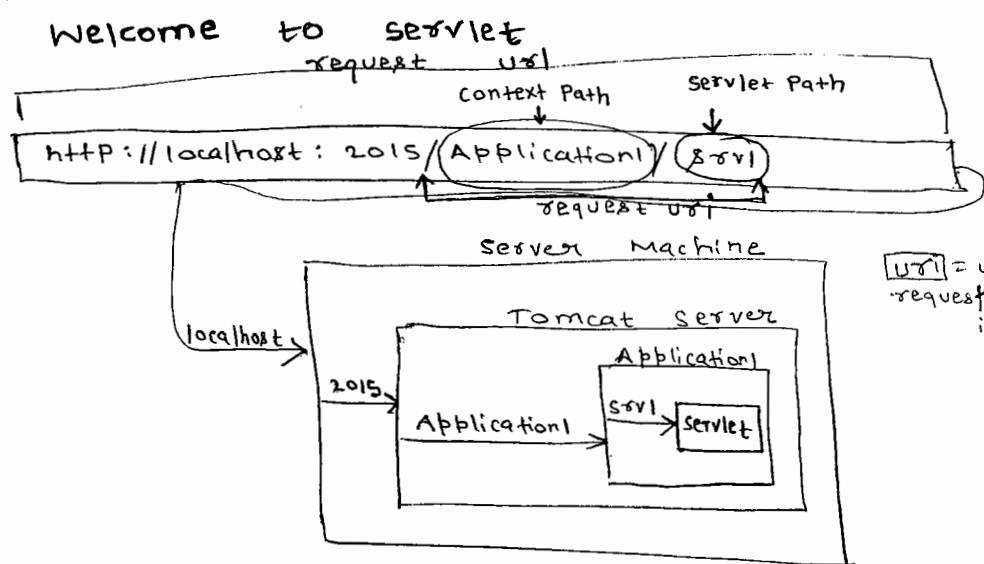
Step 6 start the server

Go to Tomcat 7.0 \bin folder and double click on tomcat7.exe

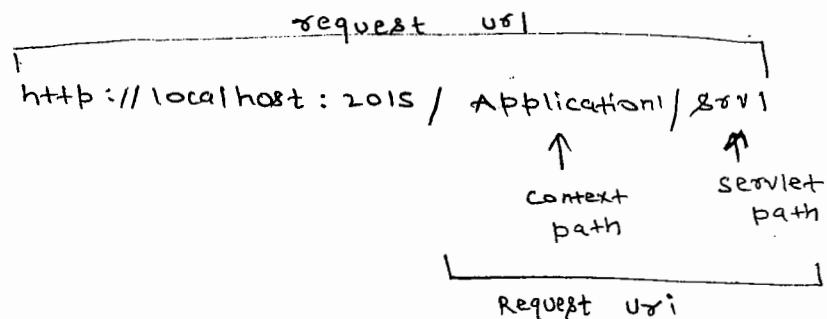
Step 7 Open the browser and type the following URL in address bar

http://localhost:2015/Application1/Serv1 ↵

O/P :



* The different paths in the above request are—



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

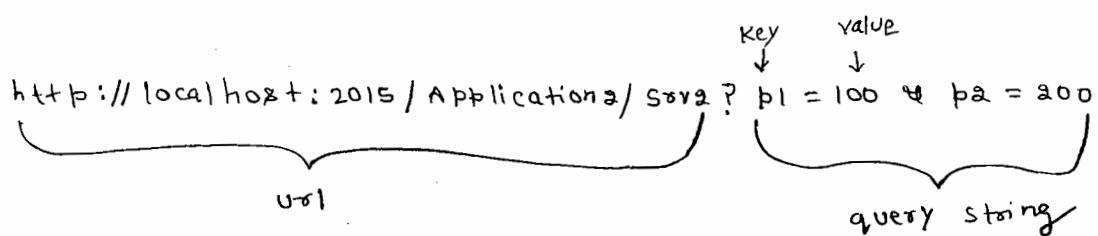
18 Aug 15

sending i/p to a servlet -

- 1) When sending a request to a servlet along with that request we can also send input values to the servlet.
- 2) A servlet reads i/p, processes the request and then generates the output.
- 3) We can add i/p values to a request from browser in two ways -
 - (i) By adding a query string to a url
 - (ii) By creating an <html> page.

Using query string -

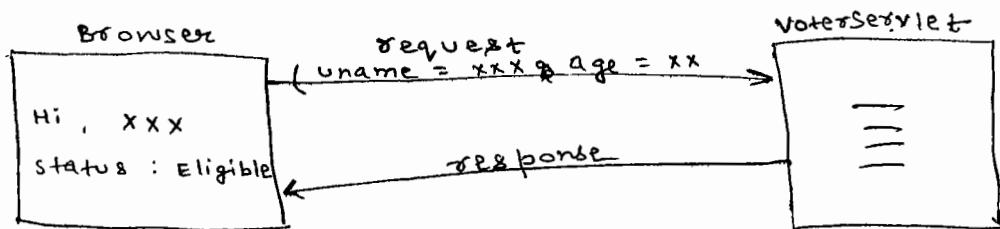
- 1) A query string is a group of parameters appended to the url in address bar.
- 2) A query string can have one or more parameter where each parameter will be in the form Key = value pair.
- 3) A parameter is separated with next parameter by using $\&$ symbol.
- 4) A query string is separated from its url by using $?$ symbol



- 5) When a request is sent to a servlet then servlet container will store the parameters in request object.
- * 6) A request object internally maintains a Map object for storing the parameters.
- 7) To read the value of a parameter from request object, we need to call getParameter() of ServletRequest interface.
- ```
String s1 = request.getParameter("P1");
String s2 = request.getParameter("P2");
```
- \* 8) getParameter() reads value as string format. If conversion needed then we need to apply wrapping.

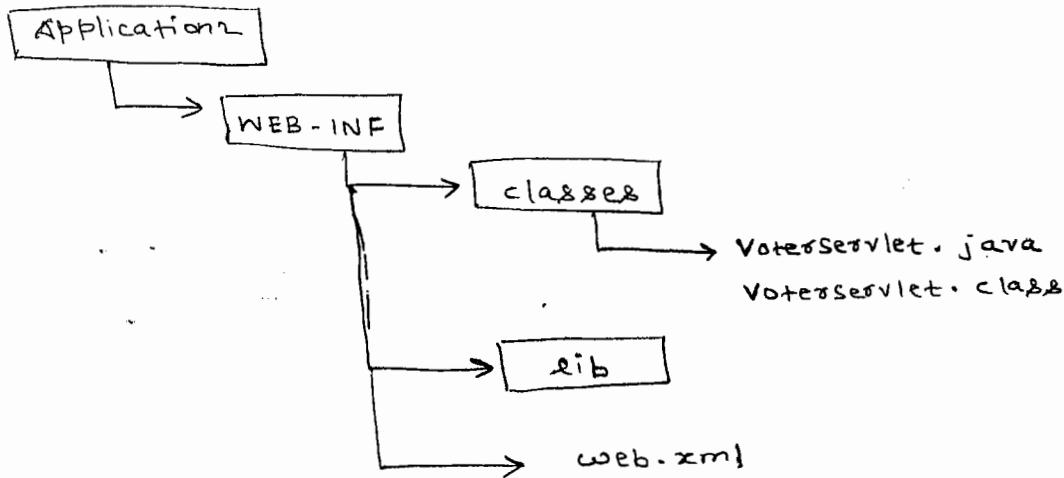
### Example 2

In the following web-application, a servlet accepts two input values username and age. Based on that input a servlet generates the output.



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road.  
Ameerpet, Hyderabad.

## Directory structure ( Applicationz )



//VoterServlet.java

```
import javax.servlet.GenericServlet;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;

public class VoterServlet extends GenericServlet
{
 public void service(ServletRequest request, ServletResponse response)
 throws ServletException, IOException
 {
 String username = request.getParameter("uname");
 String strAge = request.getParameter("age");
 int age = Integer.parseInt(strAge);
 PrintWriter out = response.getWriter();
 if (age >= 18)
 {
 out.println("Hi," + username);
 out.println("Eligible for vote");
 }
 }
}
```

XEROX  
SRI RAJAHAVENDRA Material Bazaar,  
Software Bangalore Alkamperpet, Hyderabad  
Beside CDAC, Ambedkar Road,

```

else
{
 out.println(" Hi, " + username);
 out.println(" not eligible for vote");
}
out.close();
}
}

```

// web.xml

```

<web-app>
< servlet >
 < servlet-name > Voter < /servlet-name >
 < servlet-class > Voterservlet < /servlet-class >
</servlet>
< servlet-mapping >
 < servlet-name > Voter < /servlet-name >
 < url-pattern > /servlet < /url-pattern >
</servlet-mapping>
</web-app>

```

<1> compile the servlet by adding servlet-api.jar in classpath

<2> deploy Applications folder to tomcat7.0/webapps directory

<3> start the server and then open the browser and type the following url in address bar -

http://localhost:2016/Applications/servlet?uname = xyz  
age = 20

SRINIVASAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

19 AUG 15

### HTML to servlet communication -

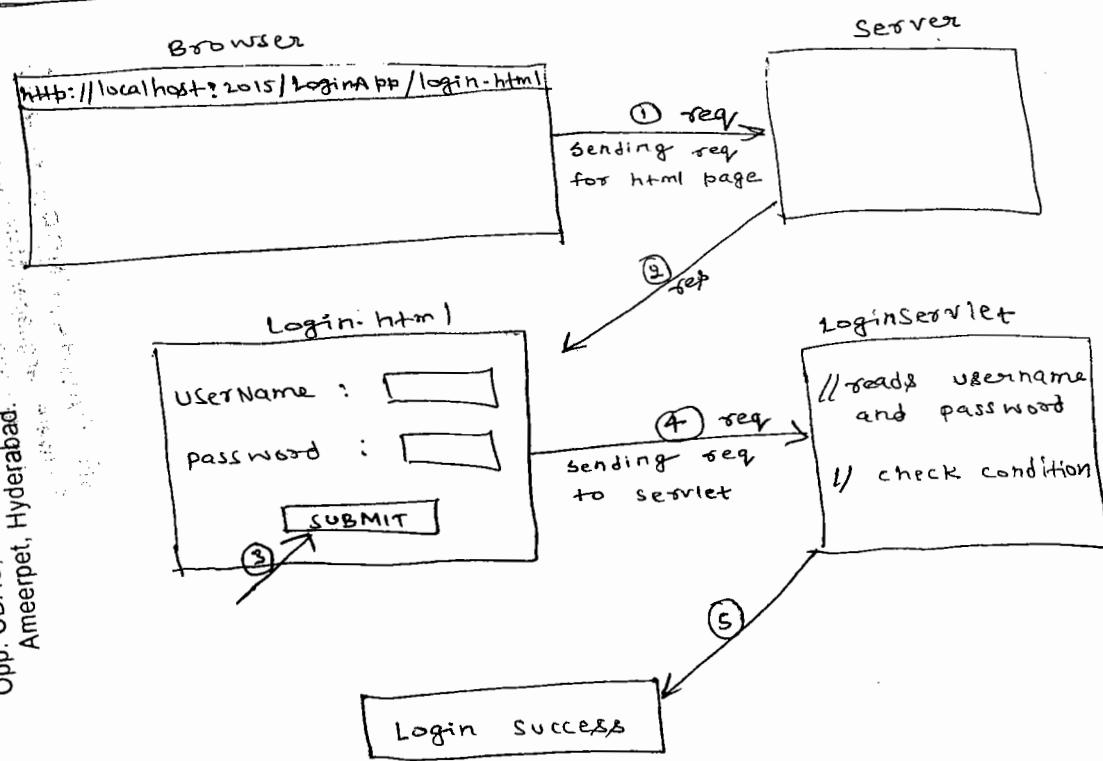
- 1) In this communication, through html page, we send input values of end user to a servlet.
- 2) While creating html page we need to design a form by using form elements, for sending the input values along with a request.
- 3) Some of the form elements are -
  - (i) <input type = "text">
  - (ii) <input type = "password">
  - (iii) <input type = "radio">
  - (iv) <input type = "checkbox">
  - (v) <input type = "button">
  - (vi) <input type = "reset">
  - (vii) <input type = "submit">
- 4) While creating web-applications, html page and servlet class both are stored in server. First html page will be sent from server to browser and that page displays a form on browser.
- 5) When a request is submitted then html page will send the i/p values to the servlet.
- \* 6) Html page is stored on server but it is executed on browser. So, html is a client side technology.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 7) To map a request from html page to a servlet, we need to add a servlet url pattern as a value to the action attribute of <form> tag.

```
<form action = "servlet">
```

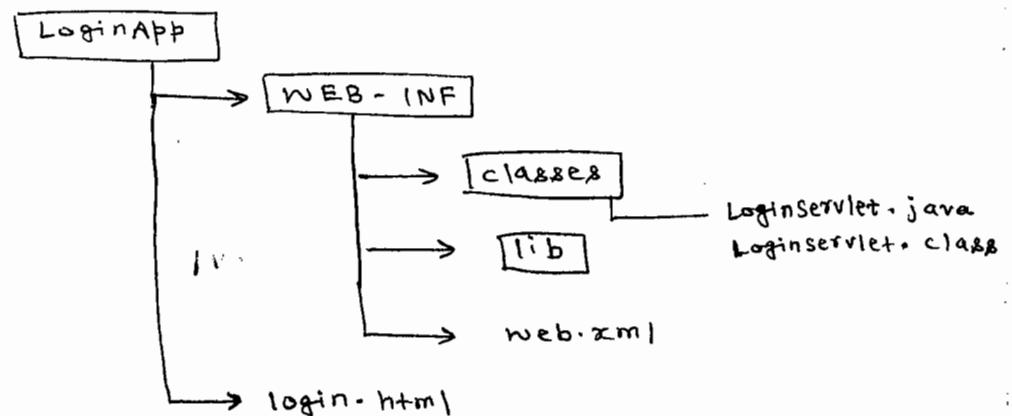
Flow:-



**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 8) In html to servlet communication, from browser, we send request for two times. First time we send request to html page and second time we send request to servlet.

## directory structure



<!-- login.html-->

```
<center>
 <h1>
 <form action = "loginsrv">
 Username : <input type = text name = "uname"
 placeholder = " Enter username">

 Password : <input type = password name = "pwd"
 placeholder = " Enter Password">
 <input type = submit value = "SUBMIT" >
 </form>
 </h1>
</center>
```

```

// Loginservlet.java

import javax.servlet.GenericServlet,
import javax.servlet.GenericServlet ;
import javax.servlet.ServletRequest ;
import javax.servlet.ServletResponse ;
import javax.servlet. ServletConfig ServletConfig ;
import javax.servlet. ServletException ;
import javax. serv

import java.io.IOException ;
import java.io.PrintWriter ;

public class Loginservlet extends GenericServlet
{
 public void init(ServletConfig config) throws
 ServletException
 {
 System.out.println("I am init()") ;
 } // end of init()

 public void service(ServletRequest request,
 ServletResponse response) throws ServletException,
 IOException
 {
 System.out.println("I am service()") ;
 // read username and password
 String str1 = request.getParameter("uname");
 String str2 = request.getParameter("pwd");
 if
 PrintWriter out = response.getWriter();
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

```

if (str1.equals("sathya") && str2.equals("sekhar"))
{
 out.println(" login success ... ");
}
else
{
 out.println(" login failed ");
}
out.close();
// end of service();

public void dispta destroy()
{
 System.out.println(" I am destroy ");
}
// End of destroy()

```

```

<t-web.xml -->
<web-app>
 <Servlet>
 <Servlet-name> login </Servlet-name>
 <Servlet-class> Loginservlet </Servlet-class>
 </Servlet>
 <Servlet-mapping>
 <Servlet-name> login </Servlet-name>
 <url-pattern> /login< /url-pattern>
 </Servlet-mapping>
</web-app>

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

- 9) compile the servlet by adding servlet-api.jar to the classpath
- 10) copy LoginAPP folder in tomcat 7.0/webapps directory
- 11) start the server and then type the following request from browser -

http://localhost:2015/LoginAPP/login.html

20 Aug 15

Note -

- 1) If we write any System.out.println statement in a servlet class then that message will be displayed on server console and also stored in server log file. But, not displayed on browser.
- 2) We can open the log file from C:\Program Files\Apache Software Foundation\Tomcat 7.0\log\catalina.

2015-08-20 .

Early loading a servlet

- 1) By default, a servlet container loads a servlet class into JVM and creates an object for the servlet class, when a first request is arrived to that servlet from browser. This is called lazy loading a servlet
- 2) We can tell the servlet container to create an object of a servlet class at deployment time only, by adding <load-on-startup> tag in web.xml file. This is called early loading a servlet.

- 3) <load-on-startup> tag is a child tag of <servlet> tag.
- \* 4) The default value of <load-on-startup> tag is -1.
- \* 5) <load-on-startup> tag value must be an integer. If it is negative integer then a servlet is lazy loaded and if it is positive integer then a servlet is early loaded.
- 6) If we configure a double or string or a boolean value to <load-on-startup> tag then exceptions are thrown by the servlet container at deployment time.

for example

<servlet>

```

<servlet-name> one </servlet-name>
<servlet-class> First </servlet-class>
<load-on-startup> 10 </load-on-startup>

```

</servlet>

- 7) If <load-on-startup> tag is added for more than one servlet in web.xml then servlet container will create objects based on priority. Low value gets high priority and high value gets low priority.

For ex -

<servlet>

```

<servlet-name> one </servlet-name>
<servlet-class> FirstServlet </servlet-class>
<load-on-startup> 10 </load-on-startup>

```

</servlet>

< servlet >

< servlet-name > TWO </ servlet-name >

< servlet-class > SecondServlet </ servlet-class >

< load-on-startup > 5 </ load-on-startup >

</ servlet >

- 8) In the above, container creates SecondServlet class first and FirstServlet class object next.
- 9) If two servlets contains equal <load-on-startup> value, then ~~order~~ priority will be based on configuration order.

Servlet class	load-on-startup	priority
Servlet 1	5	3rd
Servlet 2	2	2nd
Servlet 3	0	1st
Servlet 4	-10	when first request arrived
Servlet 5	-1	when first request arrived

Q) When container destroys servlet object ?

Ans - 1) When a servlet object timeout period (Waiting time) is exceeded.

- 2) When a web-application is undeployed from server.
- 3) When execution of a web-application is stopped.
- 4) When a server is shutdown.
- 5) When a web-application is reloaded.
- 6) When a server is crashed.

21 Aug 15

Q) Why a Servlet class must be a public class ?

Ans - 1) A Servlet container is a pre-defined class and our servlet class is an user-defined class and both are in different packages.

2) One package class becomes visible to another package class if the class is a public class. So, at the time of creating a servlet class, we need to make our class as a public class.

Q) Can we write a parameterized constructor in a servlet class or not ?

Ans - 1) A servlet container creates a servlet class object, by calling `newInstance()`.

e.g.

```
Class.forName("LoginServlet")::newInstance()
```

- 2) `newInstance()` calls default constructor in the class for creating object.
- 3) If we define only parameterized constructor then ~~`newInstance()`~~ can't `newInstance()` can't create object and it throws an exception.
- 4) If we write a default constructor and parameterized constructor both then there will be no exception so, if we want to write parameterized constructor then we also must a default constructor manually.

SRI RAHMDEVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### configuring a welcome file

- 1) When creating web applications, if we want to send a file (.html file) automatically to the browser, whenever a the request comes to the web Application then we need to configure that file as a welcome file in `web.xml` of a web-application.
- 2) For ex, if we want to configure `login.html` file as a welcome file then we need to configure in `web.xml` like the following —

```
<welcome-file-list>
```

```
 <welcome-file> login.html </welcome-file>
```

```
</welcome-file-list>
```

- 3) while sending a request from browser, we no need to type html file name in address bar.

http://localhost:2015/LoginApp ←

- 4) In web.xml, if more than one file is configured as welcome file then server will search for the files in the sequential order.
- 5) If a file is found then that file will be sent to browser. If no file is found then http status 404 response will be sent to the browser.
- 6) If a web application has index.html page then it will be the default welcome file of that application. so, we no need to configure index.html file has a welcome-file.  
Suppose, we have index.html and login.html and we want to set login.html as a welcome file then we need to configure login.html as a welcome-file.
- 8) If a web-application contains index.html and index.jsp both files then index.html file will be used by the server as a welcome file. If not then index.jsp file will be used by server as a welcome file.

22 Aug 15

### console deployment in Tomcat

- 1) If we directly copy a web-application root directly into tomcat7/webapps folder then it is hard deployment.
- 2) Hard deployment is possible, when tomcat is running on the local machine.
- 3) If the tomcat server is running on a remote machine then hard deployment is not possible.
- 4) By using console deployment, we can deploy either in the local tomcat server or in another system tomcat server running on network.
- 5) For console deployment, we need to follow the below steps -

#### Step (i) create a war file -

A war file is a compressed file of a web application and it must be created under root directory.

```
E:\LoginApp> jar cvf Logincheck.war .
↑
↑
tool
create
verbose (o/p)
file
```

MR RAHUVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Step(ii) Start Tomcat server.

Step(iii) Open Tomcat homepage on browser

http://localhost:2015

Step(iv) Click on Manager App button → Enter Username and Password → Click on LOGIN

Step(v) Go to war file to deploy section, click on browse button and select war file (LoginCheck.war) → Deploy

Step(vi) To check the web application, type the following request.

http://localhost:2015/LoginCheck  
↑  
warfilename

#### Methods to read request parameter

- 1) getParameter()
- 2) getParameterValues()
- 3) getParameterNames()
- 4) getParameterMap()

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 1) getParameter() — is used to read single value of a parameter in the form of string.

```
String str = request.getParameter("uname");
```

- 2) If multiple values are submitted with a single parameter name then by using getParameter() we can read only one value, but we can't read multiple values. So, we need to use getParameterValues()

For ex.

Hobbies :

<input checked="" type="checkbox"/> music
<input type="checkbox"/> art
<input type="checkbox"/> dance
<input checked="" type="checkbox"/> watching TV

**SUBMIT**

Servlet

String hobbies []

= request.getParameterValues("Hobby");

hobbies [0] → music

hobbies [1] → watching TV

24 AUG 15

getParameterNames() - 1) reads only parameter names from request object.

- 2) when a request is submitted from browser then container stores all the parameters in a request object. A request object internally contains a map which stores key and values.

when we call getParameterNames () then it reads only parameter names (keys), stores them in Enumeration object and finally returns Enumeration object.

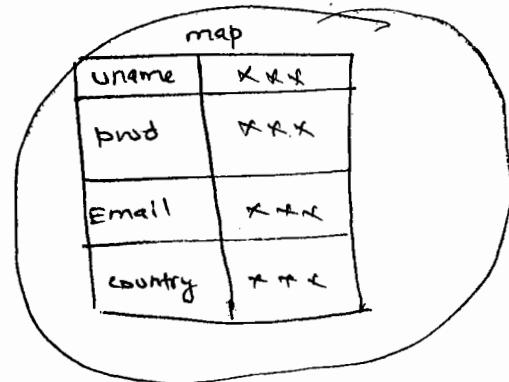
Enumeration is a collection, it maintains a cursor and by default the cursor is positioned before first element.

- 5) we can read one by one elements of enumeration by calling the methods hasMoreElements () and nextElement () .

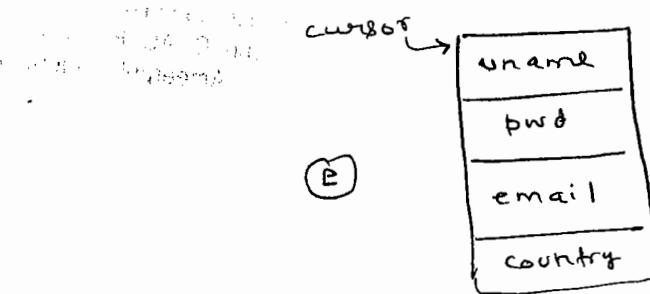
html

userName	<input type="text"/>
password	<input type="text"/>
Email	<input type="text"/>
country	<input type="button" value="▼"/>
<input type="submit" value="SUBMIT"/>	

server  
request ← object



Enumeration e = request.getParameterNames();



while (e.hasMoreElements())

    Object o = e.nextElement();

    String s = (String)o;

}

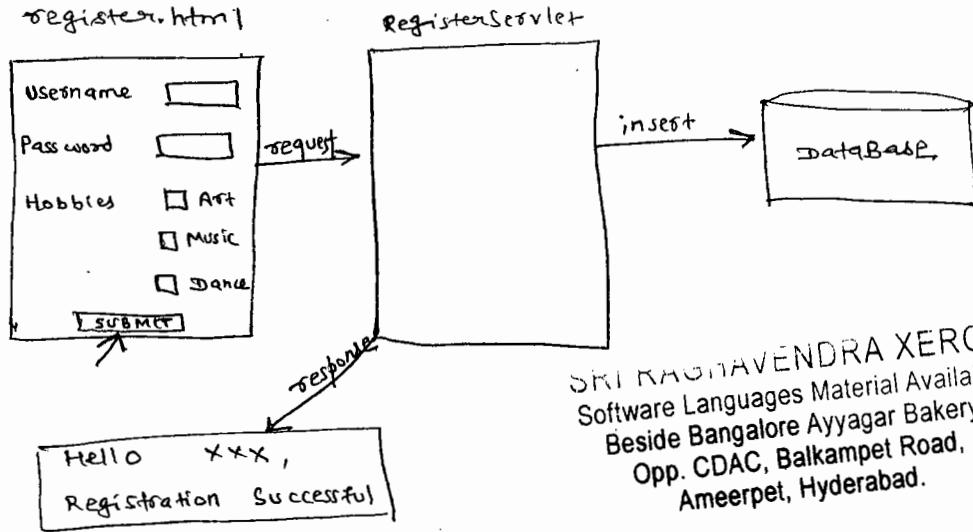
### getParameterMap()

- It is used to read the completely map object from request object. In this map object parameter names and values exist.

Map m = request.getParameterMap();

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Opp. CDAC, Balkampet Bakery,  
Ameerpet, Hyderabad.

## creating a web application in Eclipse IDE



SKI KUMARAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 1) Eclipse is an open source IDE for creating Java and JavaEE applications.
- 2) Eclipse JAVA IDE is only for developing core java applications and Eclipse JAVA EE IDE is for developing both core java and web applications and other framework application also.
- 3) Download eclipse IDE, for JavaEE developer, LUNA package from  
<http://www.eclipse.org/downloads/packages> (32-bit)
- 4) A zip file is downloaded, extract it then eclipse folder is created.

Step 1 - Start the Eclipse IDE.

~~http://www.eclipse.org/downloads/packages~~

⇒ D:\eclipse\eclipse.exe (Run as Administrator)

Step-2) Enter workspace name : E:\WI → OK

Step-3) Click on file menu → new → dynamic Web Project  
→ enter project name : RegisterApp → change  
web module version as 2.5 → finish

Step-4) copy servlet-api.jar and ojdbc6.jar (oracle 11g)  
to webcontent / WEB-INF / lib folder

Step-5) Right click on webcontent folder → new →  
html file → filename → "Register.html" → finish

```
<!-- register.html -->
<center>
<form action = "registerServlet">
 Username : <input type = text name = "uname">

 Password : <input type = password name = "pwd">

 Hobbies : <input type = checkbox name = "hobby"
 value = "Music" > Music

 <input type = checkbox name = "hobby"
 value = "Art" > Art

 <input type = checkbox name = "hobby"
 value = "Dance" > Dance

 <input type = submit value = "SUBMIT" >
</form>
</center>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

step 6 At left view expand java Resources folder  
→ src → Right click on src → new → package → enter name : com.sathyaservlet  
→ finish

step 7 Right click on the package name → new  
→ servlet → Enter class name : RegisterServlet →  
Enter super class : javax.servlet.GenericServlet →  
next → Select url mappings → Edit button →  
change pattern as /registerServlet → finish

25 Aug 15  
Absent  
(Interview)

=====

```
public class RegisterServlet extends GenericServlet
```

```
 public void service(ServletRequest request,
```

```
 ServletResponse response) throws SE, IOE
```

```
 { String str1 = request.getParameter("uname");
```

```
 String str2 = request.getParameter("pwd");
```

```
 String str3[] = request.getParameter("hobby");
```

```
 String str4 = " ";
```

```
 if (str3 != null)
```

```
 {
```

```
 for (int i = 0; i < str3.length; i++)
```

```
 { str4 = str4 + " " + str3[i];
```

```
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

PrintWriter out = response.getWriter();

try {
 Class.forName("oracle.jdbc.driver.OracleDriver");
 Connection con = DriverManager.getConnection("jdbc:
 oracle:oci:@XE", "system", "tiger");

 PreparedStatement pstmt = con.prepareStatement(
 "Insert into userinfo values (?, ?, ?);"

 pstmt.setString(1, str1);
 pstmt.setString(2, str2);
 pstmt.setString(3, str4);

 int i = pstmt.executeUpdate();

 pstmt.close();
 con.close();
 out.println(" Hi ", + str1);
 out.println("
");
 out.println(" Register successful ");

}

catch(Exception e) {
 out.println(" Registration failed ");
}

out.close();
}

```

SRI RAJAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagir Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

Step 8 - click on window menu - show view →  
servers → click on the link → Expand Apache  
→ Select Tomcat 8.0 server → next →  
click on browse button and select  
tomcat 8.0 → finish

Step 9 - open sql plus and create a table  
like the following -

create table userinfo

(  
    uname varchar(15) primary key,  
    pwd varchar(12),  
    hobbies varchar2(20)  
) ;

Step 10 - Right click on registerAPP → Run As →  
Run on server → next → finish

Step 11 - open the browser and type the  
following request .

<http://localhost:2015/RegisterAPP/register.html>

20 Aug 15 HttpServlet - an abstract class of javax.servlet.http pkg

- 1) HttpServlet is a subclass of ~~GenericServlet~~ GenericServlet
- 2) In HttpServlet ~~the~~ class abstract service() of GenericServlet is overridden.
- 3) HttpServlet class included one more service() and we call it as non-life-cycle service(). So, in HttpServlet class there are two service method.

(i) public void service (servletRequest request, servletResponse response) throws ServletException, IOException  
Life cycle method →

(ii) protected void service (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException  
Non life cycle method →

- 4) HttpServlet class included 7 doXXX() methods.
- 5) Among 7 doXXX(), we use only doGet() and doPost() methods.
- 6) In HttpServlet class 7 doXXX() contains logic which is not related to application. They contain logic (dummy) for sending an <sup>405</sup> error as response.

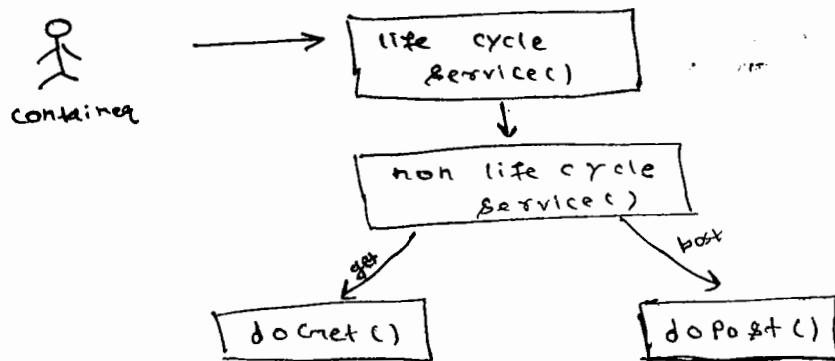
- 7) The 7 doXXX() are not provided as abstract methods, because if they are abstract methods then a developer must override all 7 abstract methods by extending HttpServlet class.

\* \* \* 8) HttpServlet class is a complete class, but

*Explanation ①+②* ~~it~~ it is not a completely defined class so, it is given as a abstract class.

\* Actually, HttpServlet class doesn't contain abstract methods, but class is abstract class.

- g) we extend our class from HttpServlet when we want to define a logic for http get() method request and another logic for http post() method request.
- h) we override doGet() , to define the logic for get () request .
- i) we override doPost() , for post () request .
- j) public class MyServlet extends HttpServlet  
 {  
 public void doGet( request , response ) throws SE,IOE  
 {  
 }  
 ==  
 }  
 public void doPost( request , response ) throws SE,IOE  
 {  
 }  
 ==  
 }  
 }
- k) doGet() and doPost() are not life cycle methods so , they are not directly called by the container .
- l) container calls life cycle service () , then it internally called non life cycle service () then it calls doGet() for a get request and doPost() for a get post request .



Q) what happens, if doPost() is not overridden for a post() request?

Ans - If not overridden then super class doPost() is called and it will send 405 error as a response.

Q) How many init() and service() are there in Generic Servlet?

Ans - Two init() and one service()

Q) How many init() and service() are there in HttpServlet?

Ans - two init() and two service()

GenericServlet  
↑  
HttpServlet

Q) can we send request through POST() by clicking hyperlink?

Ans - No. Because <a> tag doesn't allow method attribute.

By default, get() is used by browser to send the request.

Q) What are the different ways to write logic for creating our servlet class from extending HttpServlet

Ans - public class MyServlet extends HttpServlet

{ opt1 }

    public void service(ServletRequest request,  
                          ServletResponse response) throws  
                          SE, IOE

}

opt2

    public void service(HttpServletRequest request,  
                          HttpServletResponse response) throws  
                          SE, IOE

{

}

opt3

Recommended

    public void doXXX(HttpServletRequest request,  
                          HttpServletResponse response) throws  
                          SE,  
                          IOE

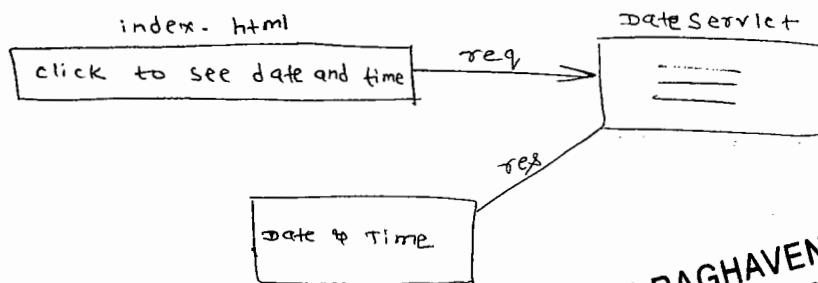
{

}

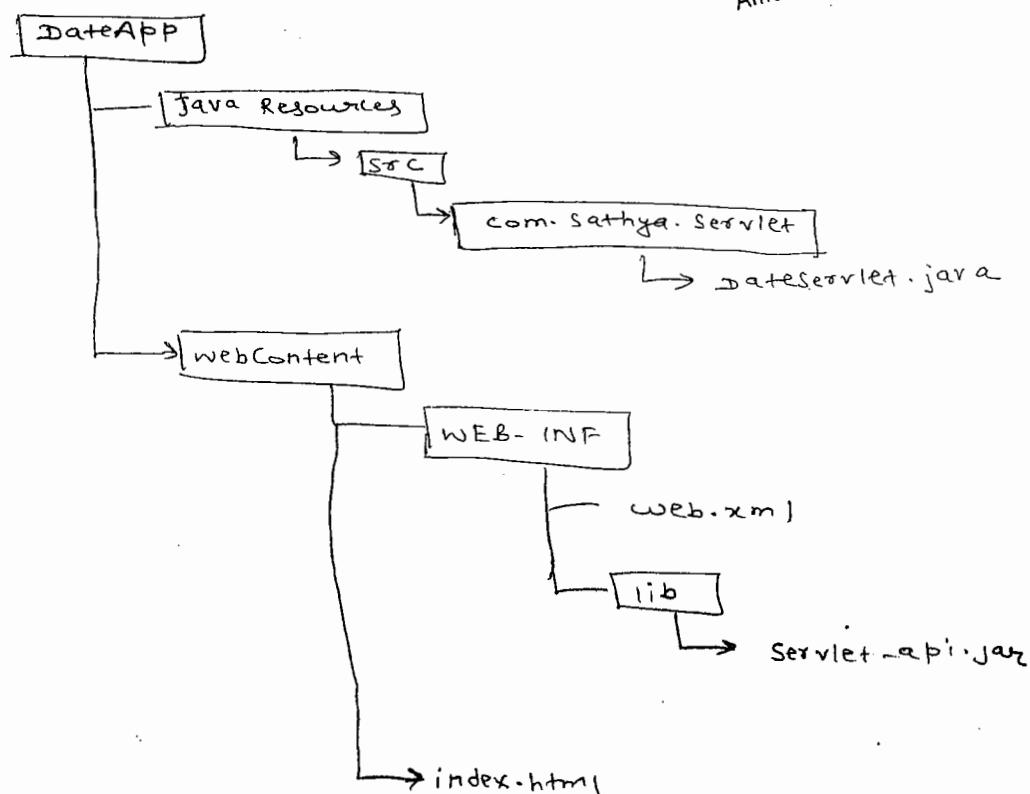
27 AUG 15  
(Interview)

### Example

The following web application displays system date and time and response will be changed for each and every second.



### Eclipse project structure



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### index.html

```
<center>
<h1>
 click to see date and time
</h1>
</center>
```

### Dateservlet.java

```
public class Dateservlet extends HttpServlet
{
 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException
 {
 java.util.Date date = new java.util.Date();
 PrintWriter out = response.getWriter();
 response.setHeader("Refresh", "1");
 out.println(date);
 out.close();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Baikampet Road,  
Ameerpet, Hyderabad.

- 1) In the above servlet program, we have written  
new line of code

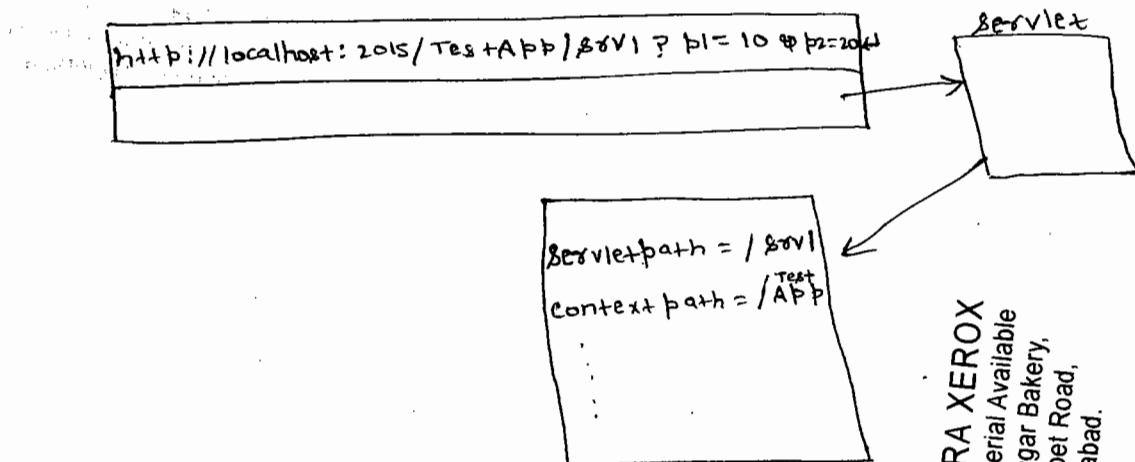
```
response.setHeader("Refresh", "1");
```

- 2) This statement tells the browser that refresh  
response of servlet for every 1 sec.

- 3) A browser automatically sends a request  
to servlet for every one second and  
servlet will display response to the browser

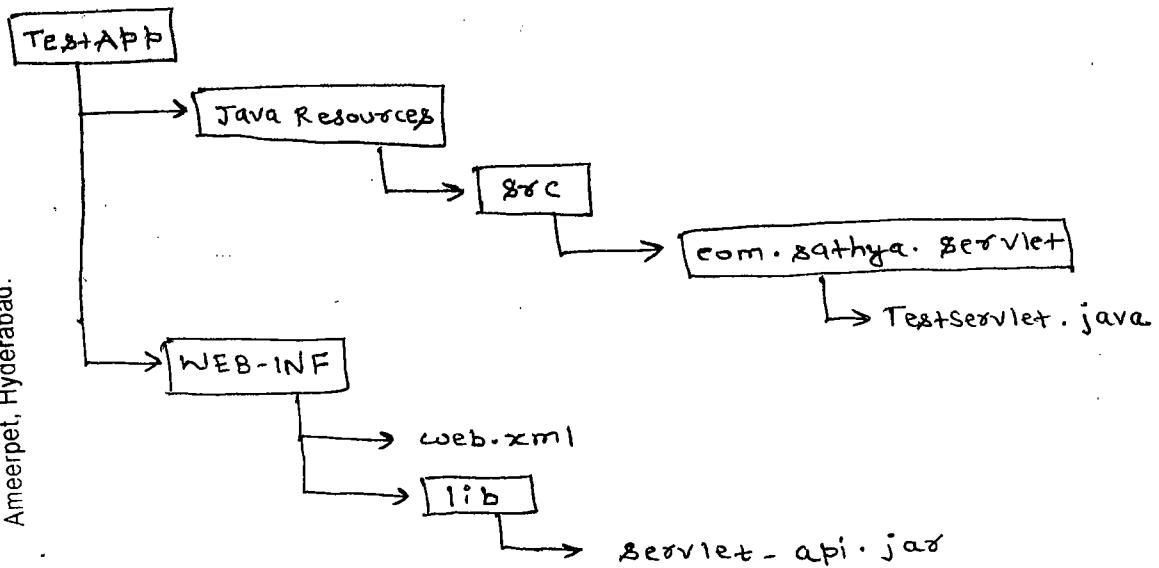
## How to read parts of url ?

- i) we can read different parts of url, by calling the following methods of HttpServletRequest object
- ii) getServletPath() → reads url pattern
- iii) getContextPath() → reads application name
- iv) getRequestURI() → reads context path and servlet path
- v) getRequestURL() → reads url of request
- vi) getQueryString() → reads a query string
- vii) getRemoteAddr() → reads IP address of client system
- viii) getScheme() → reads protocol name ....etc



## Eclipse project structure

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ammeerpet, Hyderabad.



## TestServlet.java

```

public class TestServlet extends HttpServlet
{
 protected void doGet (HttpServletRequest request,
 HttpServletResponse response) throws SE, IOE
 {
 PrintWriter out = response.getWriter ();
 out.println ("servletPath = " + request.getServletPath ());
 out.println ("contextPath = " + request.getContextPath ());
 out.println ("RequestURI = " + request.getRequestURI ());
 out.println ("requestURL = " + request.getRequestURL ());
 out.println ("Protocol name = " + request.getScheme ());
 out.println ("query string = " + request.getQueryString ());
 out.println ("client IP Address = " + request.getRemoteAddr ());
 out.close ();
 }
}

```

The code defines a Java servlet named 'TestServlet' that extends the 'HttpServlet' class. The 'doGet' method prints various request parameters to the response using a PrintWriter. The output includes the servlet path, context path, RequestURI, requestURL, protocol name, query string, and client IP address.

## MIME Type

- 1) MIME stands Multipurpose Internet Mail Extension
- 2) From servlet we can send different types of response to the browser.
- 3) For eg., we can send html type of response or XML type of response or PDF type of response etc
- 4) At the time of sending a response, along with that response, a servlet should also inform the browser about the content type of the response.
- 5) Based on content type, a browser will take support of a software or plugin, to open that file.
- 6) For eg., if content type of response is pdf then browser couldn't directly open the response, so it takes the support of acrobat reader software to open pdf file.

SRI RAHUVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

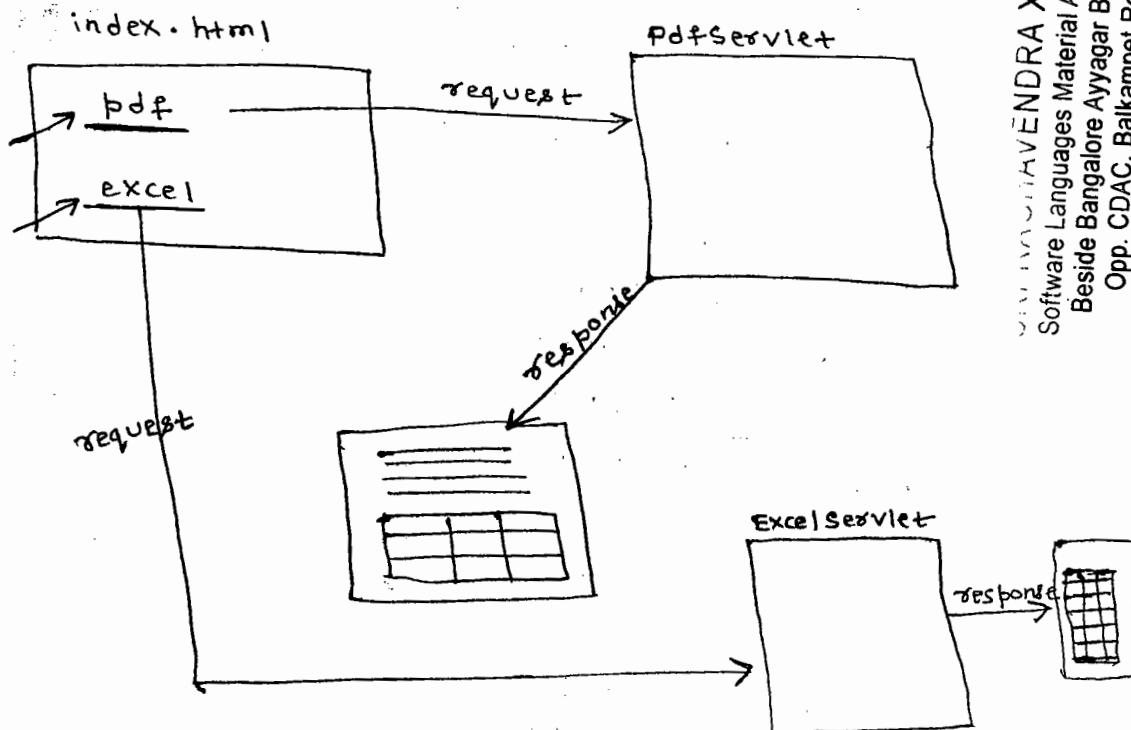
- 7) If content type of response is html or plain or xml then browser can directly open that response.
- 8) The MIME types, also called internet media types or standard for the web application in using any web technologies like servlet, php or asp etc.
- 9) When we developing a servlet program, to set the MIME type for response, we need to call `getContentType()` of `ServletResponse` interface.
- 10) If you don't set MIME type by default MIME type will be set ~~be set~~ as `text/html`.
- 11) Some important MIME type are -
- `text/html` (type/subtype)
  - `text/plain`
  - `text/xml`
  - `application/pdf`
  - `application/msword`
  - `application/ms-excel`
  - `application/vnd.ms-powerpoint`
  - `image/gif`
  - `image/jpeg`
  - `video/x-flv`
  - `video/avi`
  - `audio/mpeg`
  - `video/mpeg`

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Generating output in pdf and Excel formats

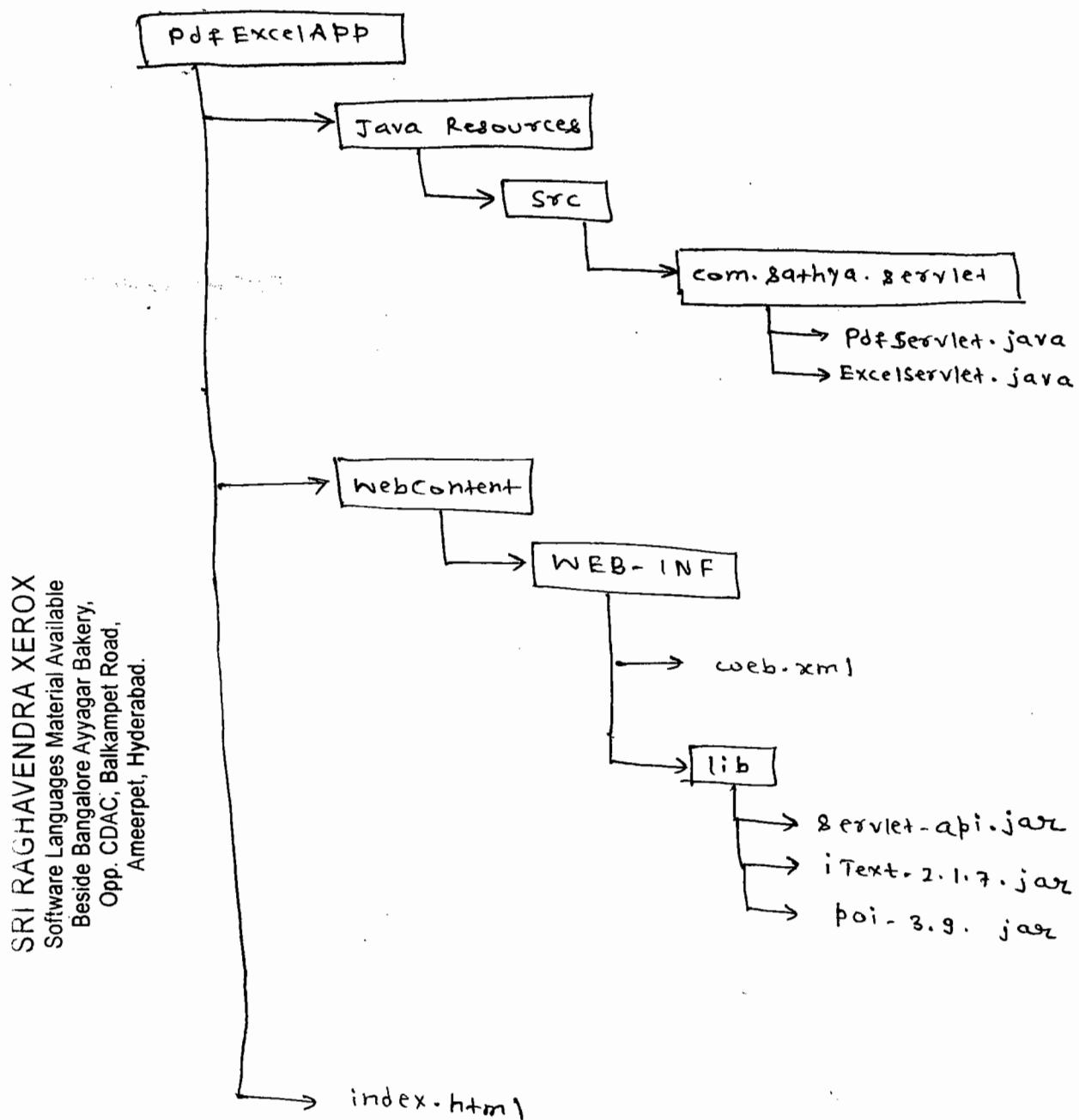
- 1) In servlet api there is no predefined classes or interfaces for generating the outputs in PDF and Excel format.
- 2) To generate the output in PDF document we use third-party api called iText api
- 3) To generate response in Excel format, we use third party api called poi api
- 4) When we use third party api in web application then we need to copy their jars files to lib folder.

## Project flow



## Project Flow

### Eclipse Project Structure



## index.html

<h2>

<a href = " pdfserv "> PDF </a>

<br>

<a href = " excelserv "> Excel </a>

</h2>

## PdfServlet.java

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

```
public class PdfServlet extends HttpServlet
```

```
{
```

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws SE, IOE
```

```
{ // Set content type
```

```
response.setContentType("application/pdf");
```

```
// we want servletOutputStream object
```

```
servletOutputStream sos = response.getOutputStream();
```

```
try
```

```
{
```

```
// create Document object
```

```
Document d = new Document();
```

29 Aug 15  
(Rakshabandhan)

pdfteservlet.java (continued)

```
// PdfWriter class writes elements added to Document
object to FileOutputStream

 PdfWriter.getInstance(d, fos);

// open document

 d.open();

Paragraph p = new Paragraph();
p.add("Java Platform complete Reference");

// create table with 2 columns
Table t = new Table(2);

// row1
t.addCell("Company name");
t.addCell("website");

// row2
t.addCell("IBM");
t.addCell("www.ibm.com");

// row3
t.addCell("Apache");
t.addCell("www.apache.org");

// add paragraph to document
d.add(p);

// add table to document
d.add(t);

// close element
d.close();

} catch(Exception e)
{
 SEP(e);
}
```

SRI RAHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Excelservlet.java

```
package com.sathyas.servlet;
import java.io.IOException;
public class Excelservlet extends HttpServlet {
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws SE, IOException {
 // Set content type
 response.setContentType("application/vnd.ms-excel");
 HSSFWorkbook wb = new HSSFWorkbook();
 Sheet sheet = wb.createSheet("Sheet1");
 Row r1 = sheet.createRow(0);
 Cell c1 = r1.createCell(0);
 c1.setCellValue("company name");
 Cell c2 = r1.createCell(1);
 c2.setCellValue("website");
 Row r2 = sheet.createRow(1);
 Cell c3 = r2.createCell(0);
 c3.setCellValue("oracle");
 Cell c4 = r2.createCell(1);
 c4.setCellValue("http://www.oracle.com");
 Row r3 = sheet.createRow(2);
 Cell c5 = r3.createCell(0);
 c5.setCellValue("IBM");
 Cell c6 = r3.createCell(1);
 c6.setCellValue("http://www.ibm.com");
 ServletOutputStream so8 = response.getOutputStream();
 wb.write(so8);
 so8.close();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### \*\*\* **Servletconfig** object

- 1) **Servletconfig** is an interface of `javax.servlet` pkg.
- 2) An implementation class for **Servletconfig** <sup>interface</sup> ~~class~~ will be provided by vendors
- 3) A **Servletconfig** object means, it is its implementation class object.
- 4) When we are creating a **Servlet**, we can initialise data members of the **Servlet** in a default constructor with static values.
- 5) To initialise the data members with dynamic value, we need a parameterized constructor in a class.
- 6) In a **Servlet** class we can write parameterized constructor along with default constructor but we can't call that constructor, because a **Servlet** object will be created by container but not by programmer.
- 7) As a solution for this problem, in **Servlet** technology life-cycle `init()` is given.
- \* 8) In a **Servlet**, we can initialize data members in constructor and also in the `init()` but difference is, constructor will initialise with static values and `init()` will initialise with dynamic values.

- ✓ 9) The dynamic values to a servlet, we can pass, by configuring init-parameters in web.xml.
- 10) Every init-parameter has a name and value and it can be configured with web.xml, by using <init-param> tag.

#### Syntax

```

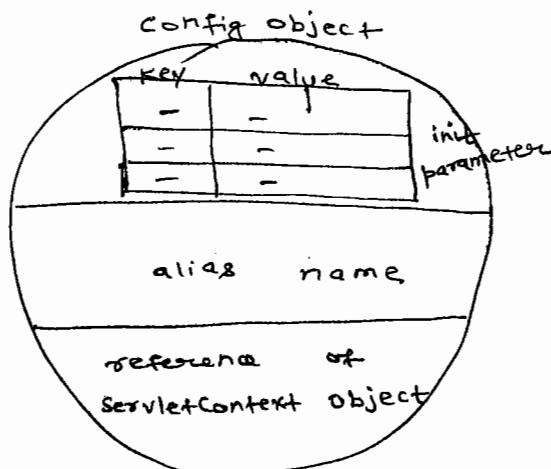
< servlet >
 < servlet-name > *** < /servlet-name >
 < servlet-class > MyServlet < /servlet-class >
 < init-param >
 < param-name > p1 < /param-name >
 < param-value > 100 < /param-value >
 < /init-param >
 < /servlet >

```

31 AUG 15

- 11) A servlet container creates config object after constructor of servlet is executed and before calling the init() of servlet.
- 12) In a config object, container will store the following three information —
- init parameters
  - alias name of servlet
  - a reference of servlet context object

- 13) To store init parameters config object internally has a Map object.



- 14) For each servlet, a servletConfig object is created. So, the ratio is 1:1.
- 15) In servletConfig interface, there are 4 methods to read the data from config object -

(i) getInitParameter(name):

- This method is to read the value of a init-parameter by using its name.

```
String sto = config.getInitParameter(name);
```

(ii) getInitParameterNames():

- This method is used to read the names of all init parameters. This method returns Enumeration object.

- By calling the methods of enumeration we can read each init parameter name and then its value within a while loop.

```
Enumeration e = config.getInitParameterNames();
```

(iii) getServletName() :-

- This method is to read alias name of the servlet.

```
String str1 = config.getServletName();
```

\*\* (iv) getServletContext() :

- This method is to read a reference of ServletContext object.

```
ServletContext ctx = config.getServletContext();
```

How to use config object in service() ?

- 1) servlet container will pass the config object to init() as a parameter, but not to the service().
- 2) we can use config object in service() by getting it in two ways —

way1

- If life cycle init() is overridden then we can store config object in a instance variable and we can use that instance variable in service().

For ex,

```
public class MyServlet extends GenericServlet
{ private ServletConfig sc; // instance variable.
 public void init(ServletConfig config) throws ServletException
 { sc = config;
 }
 public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
 { // Here we can use sc
 }
```

### ways

- i) If we do not override life cycle init() method in a servlet class then container calls init() of super class.
- ii) we can read config object from super class by calling getServletConfig().

For ex,

```
public class MyServlet extends HttpServlet
{
 public void service (request, response) throws
 SE, IOE
 {
 ServletConfig sc = getServletConfig();
 }
}
```

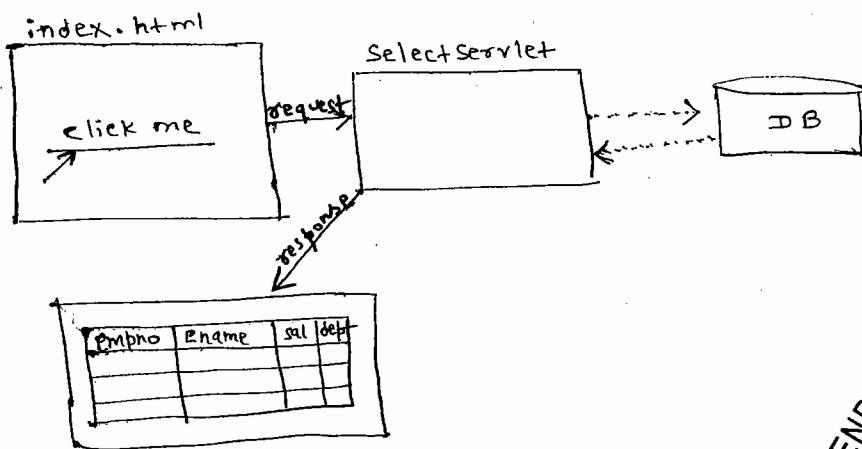
### Note -

In java , we can call a method directly without using object name in another method in the following two cases -

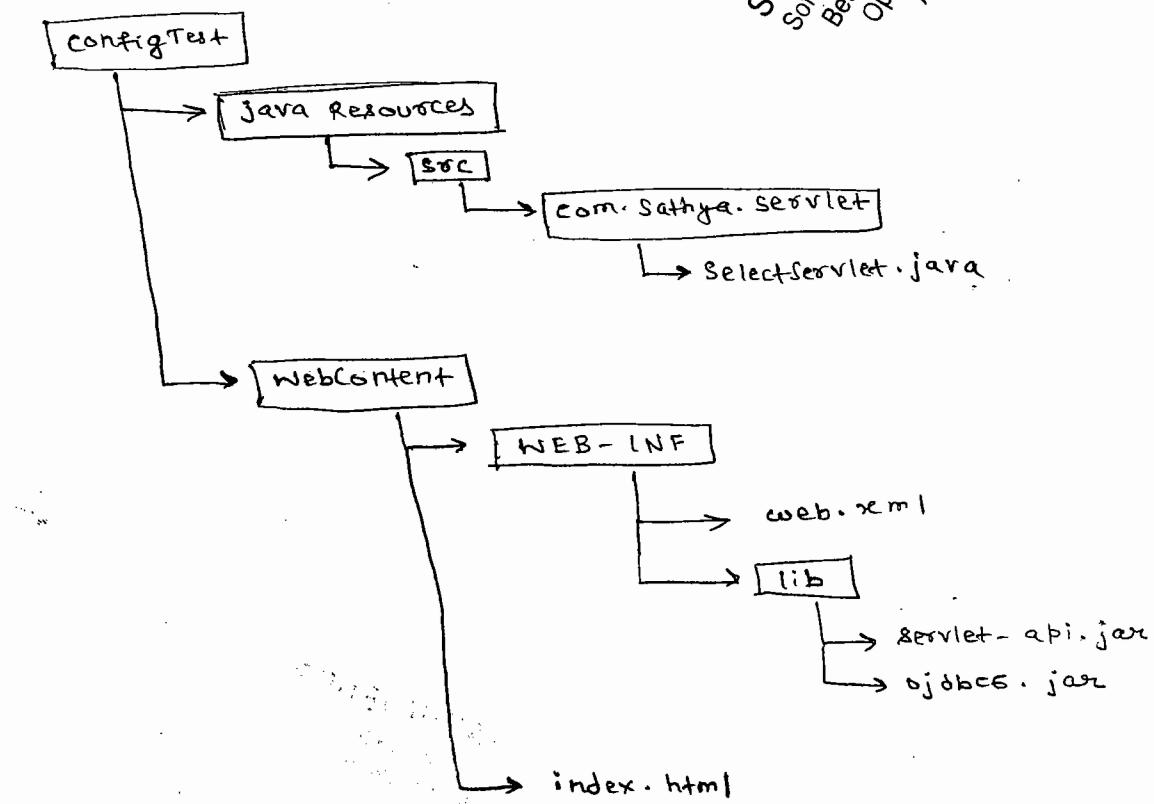
- (i) If both methods are in same class
- (ii) A super class method we can call directly in sub class.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Example



### Eclipse project structure



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

1 sept 15 index.html

```
<! -- index.html -->
<center>
 <h2> click me </h2>
</center>
```

SelectServlet.java

```
public class SelectServlet extends HttpServlet
{
 private String s1, s2, s3, s4; // datamembers
 public void init(ServletConfig config) throws SE
 {
 s1 = config.getInitParameter("driver");
 s2 = config.getInitParameter("url");
 s3 = config.getInitParameter("username");
 s4 = config.getInitParameter("password");
 }
 protected void doget(HttpServletRequest request, HttpServletResponse response)
 {
 try // get mime type throws: SE, IOE {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 Class.forName(s1);
 Connection con = DriverManager.getConnection(s2, s3, s4);
 Statement stmt = con.createStatement();
 ResultSet rs = stmt.executeQuery("Select
 * from emp");
 out.println("<center><table border = 2>");
 out.println("
");
 out.println(" <th> EMPNO </th><th> ENAME </th>
 <th> SAL </th><th> DEPTNO</th>");
 out.println(" </tr> ");
 }
 }
```

```

 out.println("<tr>");

 out.println("<td>" + rs.getString(1) + "</td>");
 out.println("<td>" + rs.getString(2) + "</td>");
 out.println("<td>" + rs.getInt(3) + "</td>");
 out.println("<td>" + rs.getInt(4) + "</td>");

 out.println("</tr>");

 } //while

 out.println("</table></center>");

 rs.close();
 stmt.close();
 con.close();

} //try

catch(Exception e)

{
 out.println("<h1>Sorry, some problem
occurred </h1>");

}

out.close();
}

```

In eclipse IDE while creating servlet, we need to click on add button to add the initialization parameter, automatically, Eclipse IDE adds init-parameters to web.xml

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### web.xml

<init-param>

<param-name> driver </param-name>

<param-value> oracle.jdbc.driver.OracleDriver </param-value>

</init-param>

<init-param>

<param-name> url </param-name>

<param-value> jdbc:oracle:thin:@localhost:1521:xe </param-value>

</init-param>

<init-param>

<param-name> Username </param-name>

<param-value> system </param-value>

</init-param>

<init-param>

<param-name> password </param-name>

<param-value> tiger </param-value>

</init-param>

Q) can we make a servlet class as final or not?

Ans- Yes, because we are not going to extend our ~~class~~ servlet class in any other class.

Ex-

(or)

```
public final MyServlet extends GenericServlet
```

```
public final MyServlet extends HttpServlet
```

SRI RAGHAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Q) can we write user defined methods in a Servlet or not?

Ans - Yes, but container doesn't call user defined methods because they are not life cycle method.

Ex, public class MyServlet extends GenericServlet

```
public void service(-----) throws SE, IOE
{
 sayHello();
}

public void sayHello()
{
}
```

Q - can we write public static void main (String k[]) in servlet class or not?

Ans - Yes, but main() can't be called by container because it is not a life cycle method.

\*\* Q - can we add checked exception additionally to the throws statement, while overriding a method or not?

Ans - No, when we are overriding a super class method in subclass, we can add unchecked exception additionally to throws statement but we can't add checked exceptions.

### Example

class A

{ void m1() throws CNFE.

```
{
 =
}
```

class B extends A

{ void m1() throws CNFE, IOE }

checked  
exception

```
{
 =
}
```

}

O/P : compile time Error

class A

{ void m1() throws CNFE

```
{
 =
}
```

class B extends A

{ void m1() throws CNFE NPE }

```
{
 =
}
```

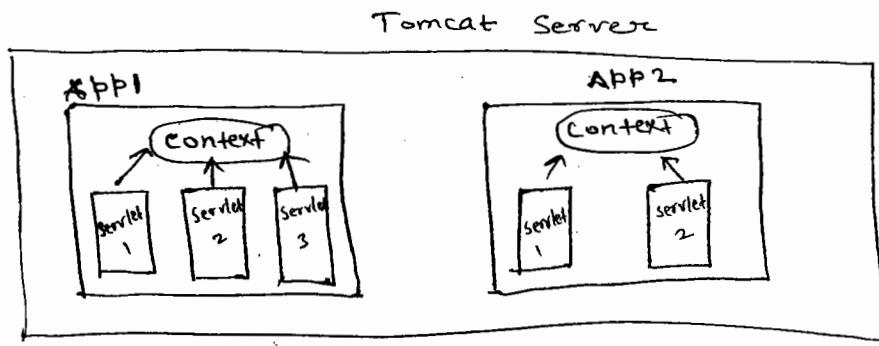
}

O/P : valid

2 Sept 15

### ServletContext interface

- 1) ServletContext is an interface whose implementation will be provided by server vendors.
- 2) ServletContext object means , its implementation class object.
- 3) When a web Application is successfully deployed in a server then immediately servlet container creates a servlet context object.
- 4) ServletContext object is one for one web application.
- 5) All the servlet classes in a web Application will share a single context object.
- 6) The ratio b/w context object and servlet object is 1: n



- 7) In a web Application , if multiple servlets need same initialization values, then from web.xml we pass the ~~init~~ initialization values by configuring context parameters.
- 8) we can also configure init parameters separately for each servlet in web.xml but the size of web.xml file is going to be increased .
- 9) In order to reduce the size of web.xml , we configure context parameters .
- 10) In a web.xml file we can configure a context parameter under the root tag <web-app> like the following -

```

<context-param>
 <param-name> xxx </param-name>
 <param-value> xxx </param-value>

```

```
</context-param>
```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

- i) When a web application is deployed, servlet container reads all the context parameters from web.xml and it will store them in context object.
- ii) A context object internally maintains a map object to store the key-value pairs.
- iii) If a servlet wants to share a value with remaining servlet then that servlet will store that value with a key in the context object. This key value pair is called an attribute.
- iv) In context object, it maintains internally two map object where one map object is for storing parameters and another map object is for storing attributes.
- v) Attributes and parameters both are in key / Value pair format but the difference is parameters are passed from web.xml and attributes are passed from servlet.
- vi) Attributes are used to share the data from one servlet to other servlet in project.
- vii) A servlet context object internally maintains two map objects, one for storing the parameters and other for storing the attributes.

18) To work with attributes we need to call the below methods -

(i) setAttribute (key, value) :- It stores an Attribute in the context object.

```
context.setAttribute("k1", 1000);
```

(ii) getAttribute (key) :- It is to read value of an attribute.

```
Object o = context.getAttribute("k1");
```

(iii) removeAttribute (key) :- It is to remove an attribute from context object.

```
context.removeAttribute ("k1");
```

(iv) getAttributeNames() - To read all attribute names from context object.

```
Enumeration e = context.getAttributeNames();
```

3 Septis

Different ways of obtaining servletContext reference -

- 1) In a servlet program, if we want to read the context parameters or context attributes then we need a reference of ServletContext object in a servlet program -

2) we can get a reference of servletcontext object in two ways -

(i) way 1 :

1) suppose, if we are overriding life cycle init() in a servlet then servlet container calls that init() and it will pass config object to the init().

2) In init(), we can read the reference of the ServletContext object from config object.

for ex:

```
public class Demoservlet extends GenericServlet
{
 private ServletContext context;
 public void init(ServletConfig config) throws SE
 {
 context = config.getServletContext();
 }
 public void service(HttpServletRequest request, HttpServletResponse response)
 throws SE, IOE
 {
 }
}
```

SRI RAJAHAVENDRA XEROX  
Software Languages Material Available  
Opp. CDAC, Balkampet Bakery,  
Ameerpet, Hyderabad.

## way 2

1) suppose, if we don't override life cycle init() in our servlet class then container calls super class init() and container will pass config object as parameter.

2) If we want to read reference of servlet context object from the super class then we need to call getServletContext() of super class.

\* In super class getServletContext() reads reference of context object from config object and then returns that reference.

```
public DemoServlet extends GS
{
 public void service(request, response) throws
 SE, IOE
 {
 ServletContext context = getServletContext();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Note :-

- 1) The following two methods are common methods for config and context object -
  - (i) getInitParameter()
  - (ii) getInitParameterNames()
- 2) To read initParameters, we call the methods with config object.

- 3) similarly, to read context parameter we call the methods context object.

String v = config.getInitParameter("k");

⇒ It reads init parameter value

String v = context.getInitParameter("k");

⇒ It reads a context parameter value

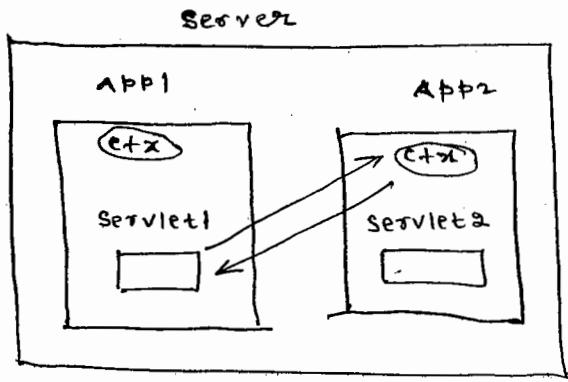
#### Asept 15 Difference b/w config and context

##### ServletConfig

- 1) Every servlet has its own config object. so ratio b/w servlet and config object is 1:1.
- 2) The data of config object is not shareable to another servlet.
- 3) without a servlet object, a config object does not exist because config object is created after creating servlet object.
- 4) A config object can store init parameters, but it can't store attribute.

##### ServletContext

- 1) multiple servlet of web application has single context object. so, ratio b/w context and servlet object is 1:n
- 2) the data of context object is shareable with all servlet of the application.
- 3) A context object can exist without a servlet object because context object is created before creating servlet object.
- 4) context object can store parameters and also attributes.



**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet, Hyderabad.

### Servlet1

```

service();

ServletContext thisContext = getServletContext();
ServletContext thatContext = thisContext.getServletContext("App2");

Object o = thatContext.getAttribute("K");

```

## JBOSS

Type : Application server

Vendor : JBoss Inc [ a division of RedHat ]

Version : 7.1.1.Final [ compatible with JDK 1.7 ]

Website : <http://jbossas.jboss.org/> download

Default http port : 8080

→ Download a zip file and then extract it.  
 JBoss server will be automatically installed

→ JBoss is an Application server and it  
 has Tomcat as a web container /  
 servlet container.

→ By default http port is 8080 and it will clash with oracle http service. So, we need to change http port number like the following —

E:\jboss-as-7.1.1.Final\standalone\configuration\  
standalone.xml

→ In this xml → <socket-binding name="http"  
port="8080"/> 2016

→ While installing jboss, it will not ask any administrator username and password.

→ To work with jboss server we need to add administrator username and password like the following —

Moto E:\jboss-as-7.1.1.Final\bin folder →  
double click on add-user (Run as Administrator) —

What type of user do you wish to add?

- a) Management user
  - b) Application user
- (a) : a

Enter the details of the new user to add realm:  
↔ (press one enter key)

Username : admin

Password : jboss

Reenterpassword : jboss

Is this correct : yes/no

[type yes]

- Start JBoss server by double click on standalone.bat in E:\jboss-as-7.1.1-final\bin folder.
  - To test whether server is started or not type : http://localhost:2016

## Deploying a web Application in jboss 7

- 1) In jboss 7 server we can only do console deployment. so, ~~first~~ first we need to create a war file.

Step1 : E:\LoginAPP > jar cvf Logincheck.war ..

Step 2 : In browser type the request

<http://localhost:2016> ← , click on

## Administration console

user name	admin
password	boss
	OK

- Step 3 : In left view, click on deployments  
→ manage deployments →  
add content button at right side →  
browse button → select war file  
(Logincheck.war) → next → save  
→ Enable → confirm.

- Step 4 : open the browser and send the request as following -

<http://localhost:2016/LoginCheck/login.html>

05 sept 15

## Servlet Exception Handling

- 1) In a servlet at runtime if any exception is occurred then that exception stack trace message will be shown to the end user on browser.
- 2) As end users are unknown about server side technologies, they can't understand the problem occurred in server. so, it is a bad way of developing application.
- 3) Instead of showing an exception to the end user we need to show a message that is understandable to the end user.
- 4) In servlet, if we want to handle the exceptions then we follow 1 of the 2 approaches —
  - ⇒ Programmatic Exception Handling
  - ⇒ Declarative Exception Handling
- 5) Programmatic exceptions of a servlet must be handled programmatically only. because if we don't handle checked exceptions then a servlet class can't be compiled.
- 6) Unchecked exception can be handle in either programmatic or declarative approach.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bus Stand  
Opp. CDAC, Balkampet Road.  
Ameerpet, Hyderabad.

### Programmatic Approach

In this approach we need to write explicitly try and catch block to the code of each servlet of a web application. In a catch block, instead of showing exception on browser we can redirect html page to the browser.

For ex -

```
public class Myservlet extends HttpServlet
{
 public void doget(— , —) throws SE, IOE
 {
 try
 {
 |||
 |||
 catch (Exception e)
 {
 response.sendRedirect("error.html");
 }
 }
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

With programmatic exception handling we have following problem -

- 1) In each servlet of the project we need to add try and catch block. It is burden on the developer.
- 2) If any modification are needed and exception handling logic then we need to modified each servlet.

- 3) If modification are done then ,we need to recompile ,then reload the application . Finally we need to restart the server.

1 Declarative

- 1) Problem of programmatic approach solved can be resolved using declarative approach.
- 2) Instead of adding try and catch block to each Servlet we can add <error-page> to web.xml
- 3) If an exception is thrown by a servlet then container look for error-page appropriate <error-page> tag , it will send the page to the browser as the response.
- 4) we can configure <error-page> under the root tag like the following -

```
<web-app>
 <error-page>
 <exception-type> java.lang.Exception </exception-type>
 <location> /error.html </location>
 </error-page>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

5) we can configure <error-page> for multiple time also in a web.xml

<error-page>

<exception-type> java.lang.NullPointerException </exception-type>  
<location> /sorry.html </location>

</error-page>

<error-page>

<exception-type> java.lang.Exception </exception-type>

<location> /error.html </location>

</error-page>

In declarative approach, we can handle error code also. Handling error-code means instead of sending error response to the browser we can send html page as a response.

Ex: If 404 error is occurred then instead of sending , if we want to send error.html as response then we need to configure like the following -

<error-page>

<error-code> 404 </error-code>

<location> /error.html </location>

</error-page>

7 Sept 15

Note-

- 1) In java we have two types of exceptions -
  - (a) Checked
  - (b) Unchecked
- 2) To find whether an Exception class is checked or unchecked, first we need to verify the Super class. If the Super class is Exception or Throwable then it is a checked Exception and if Super class is RuntimeException then it is a unchecked Exception.

SERVLET COLLABORATION

- 1) A servlet collaboration is also called servlet chaining or servlet to servlet communication.
- 2) Generally, when we are developing web Application it is not possible to develop complete Application logic in a single servlet. We create two or more servlets by dividing the logic.

To provide service to the clients, one servlet collaborates to another servlet.

- 3) For servlet collaboration, there are two techniques are provided in servlet technology -
  - (i) Dispatching
  - (ii) Redirection

## Dispatching

i) Dispatching mechanism is sub-divided into two types —

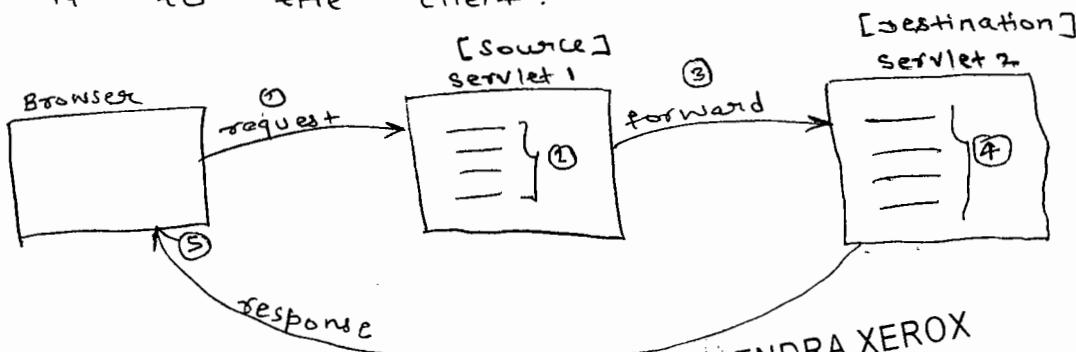
- (i) Forwarding
- (ii) Including

### (i) Forwarding

1) When there is a huge amount of code is required in a servlet then instead of writing the huge code in a single servlet we can divide the code into multiple servlet and we can collaborate the servlet using forwarding technique.

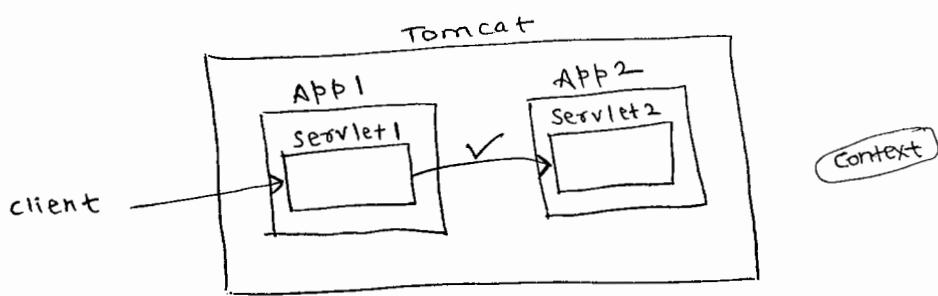
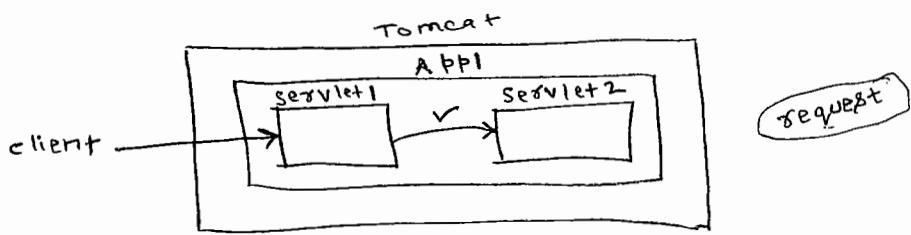
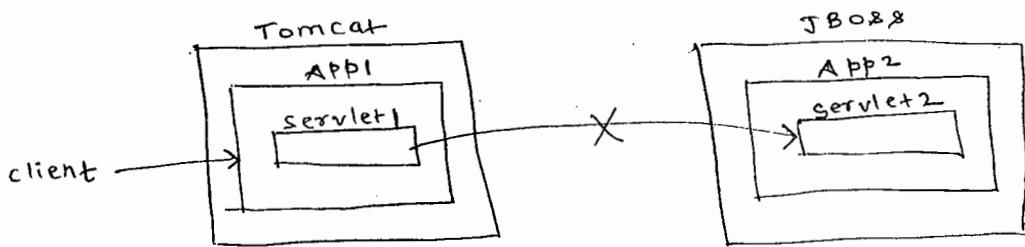
2) If suppose, logic is divided b/w two servlet then we call the 1<sup>st</sup> one as source and 2<sup>nd</sup> one as the destination.

3) When a request comes to a source then it forwards to the destination and after completion of processing, destination will generates response and then sends it to the client.

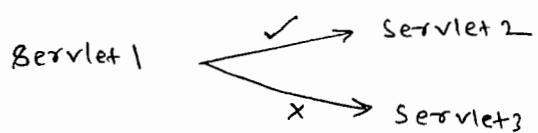


8 Sept 15

- 4) When a servlet-1 to forward request to servlet-2, a browser will not getting information about forwarding. Both the servlet are server side and forwarding is done at server side only.
- 5) A servlet-1 forwards a client request to servlet-2 but it is not a new request. So, a container will not generate another pair of request and response objects.
- 6) If when a servlet forwards a client request to another servlet then alongwith that request, request parameters are automatically forwarded to servlet 2.
- 7) In forwarding, a servlet can forward the request to another servlet / jsp page / html page.
- 8) Forwarding a request is possible, only when both source and destination servlets are exist on same server. If they exist on two different server then forwarding is not possible.



- q) when a request is forwarded from servlet1 to servlet2 then the control is permanently transferred to servlet2 . so , servlet1 can't forward the request again to servlet3 .

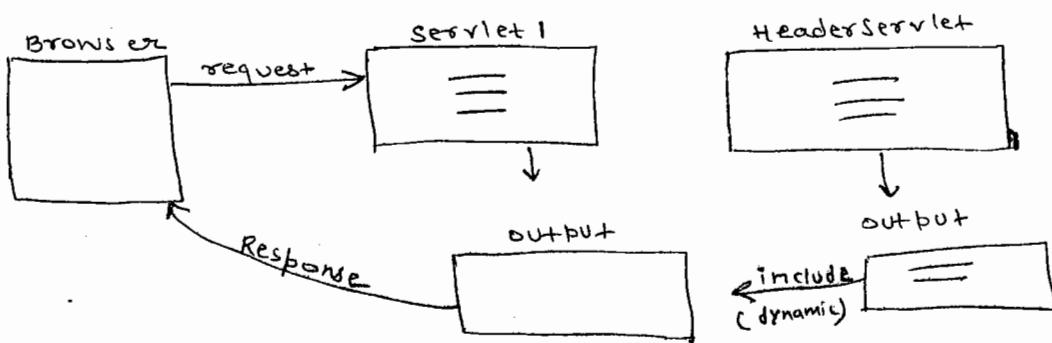


Servlet 1 → Servlet 2 → Servlet 3 ✓

Servlet 1 → Servlet 2 → Servlet 1 → Servlet 3 ✓

### Including

- 1) In a web application, if multiple servlets are having any common output then we create a separate servlet for generating the common output and we include that common output into output of multiple servlet.



- 2) A browser (client) doesn't know about including the response at server side.
- 3) A servlet can include outputs of multiple servlets. For example, a servlet can include outputs of header servlet and footer servlet.
- 4) In including, the destination resource must be a servlet/ jsp page/ html page.
- 5) If source and destination servlets are on two different servers then including is not possible. For including also both the servlets must be on same server.

- 6) In servlet-api, RequestDispatcher (Interface) is provided for forwarding a request and including a ~~the~~ response.
- 7) RequestDispatcher (Interface) has only two methods →
  - (i) forward (ServletRequest request, ServletResponse response)
  - (ii) include (ServletRequest request, ServletResponse response)
- 8) RequestDispatcher is an interface so RequestDispatcher object means its implementation class object.

#### Different ways of obtaining a RequestDispatcher object

##### gsept15 (i) Request.getRequestDispatcher (String path) :

- 1) To obtain RequestDispatcher object, we need to pass request relative path as a parameter to getRequestDispatcher() of request object.
- 2) while constructing request relative path if url directories are having virtual directory names patterns then to move the control back from one directory we need to use a symbol ..
- 3) If url patterns of ~~upto~~ two servlets are containing same directory names at a particular level then we ~~do~~ no need to move the control back from directory.

4) If sub-directory names in two url patterns are same, but parent directory names are not same then we need to move the control back from sub-directory and also parent directory.

Examples -

1) Servlet 1 → /servlet1  
Servlet 2 → /servlet2

RequestDispatcher rd = request.getRequestDispatcher ("servlet2");

2) Servlet 1 → /a/servlet1  
Servlet 2 → /b/servlet2

RequestDispatcher rd = request.getRequestDispatcher  
("a/b/servlet2");

3) Servlet 1 → /aa/bb/servlet1  
Servlet 2 → /aa/cc/servlet2

RequestDispatcher rd = request.getRequestDispatcher  
("../cc/servlet2");

4) Servlet 1 → /a/b/servlet1  
Servlet 2 → /a/b/c/servlet2

RequestDispatcher rd = request.getRequestDispatcher  
("c/servlet2");

5) Servlet 1 : /aa/bb/cc/servlet1  
Servlet 2 : /ee/bb/dd/servlet2

RequestDispatcher rd = request.getRequestDispatcher  
(".../ee/bb/dd/servlet2");

### (ii) context.getRequestDispatcher()

- 1) To obtain a RequestDispatcher object, we need to pass context relative path as a parameter for getRequestDispatcher() of context object.
- 2) context relative path means context path + servlet path. It is also called uri.

Example1 -

```
servlet1 : /servlet
servlet2 : /servlet2
```

```
RequestDispatcher rd = context.getRequestDispatcher
("/APP1/servlet");
```

Example2 -

```
servlet1 : /a/servlet
servlet2 : /a/servlet2
```

```
RequestDispatcher rd = context.getRequestDispatcher
("/APP1/a/servlet2");
```

### (iii) context.getNamedDispatcher()

- 1) To obtain a RequestDispatcher object we need to pass alias-name of destination servlet as a parameter.

Example

```
servlet1 : < servlet-name> sone </servlet-name>
servlet2 : < servlet-name> stwo </servlet-name>
```

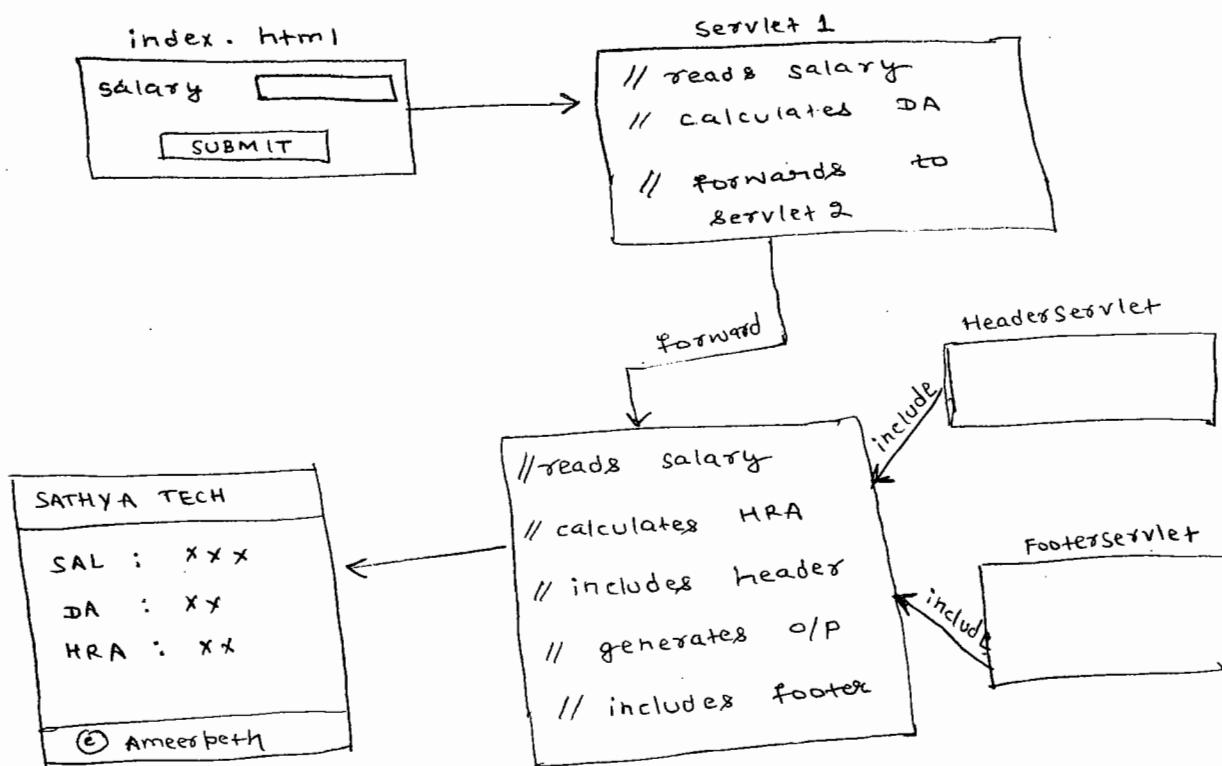
```
RequestDispatcher rd = context.getNamedDispatcher
("stwo");
```

Q) What is the difference b/w `request.getRequestDispatcher()` and `context.getRequestDispatcher()` ?

Ans- 1) `request.getRequestDispatcher()` can be used only when both servlet 1 and servlet 2 are in same web application.

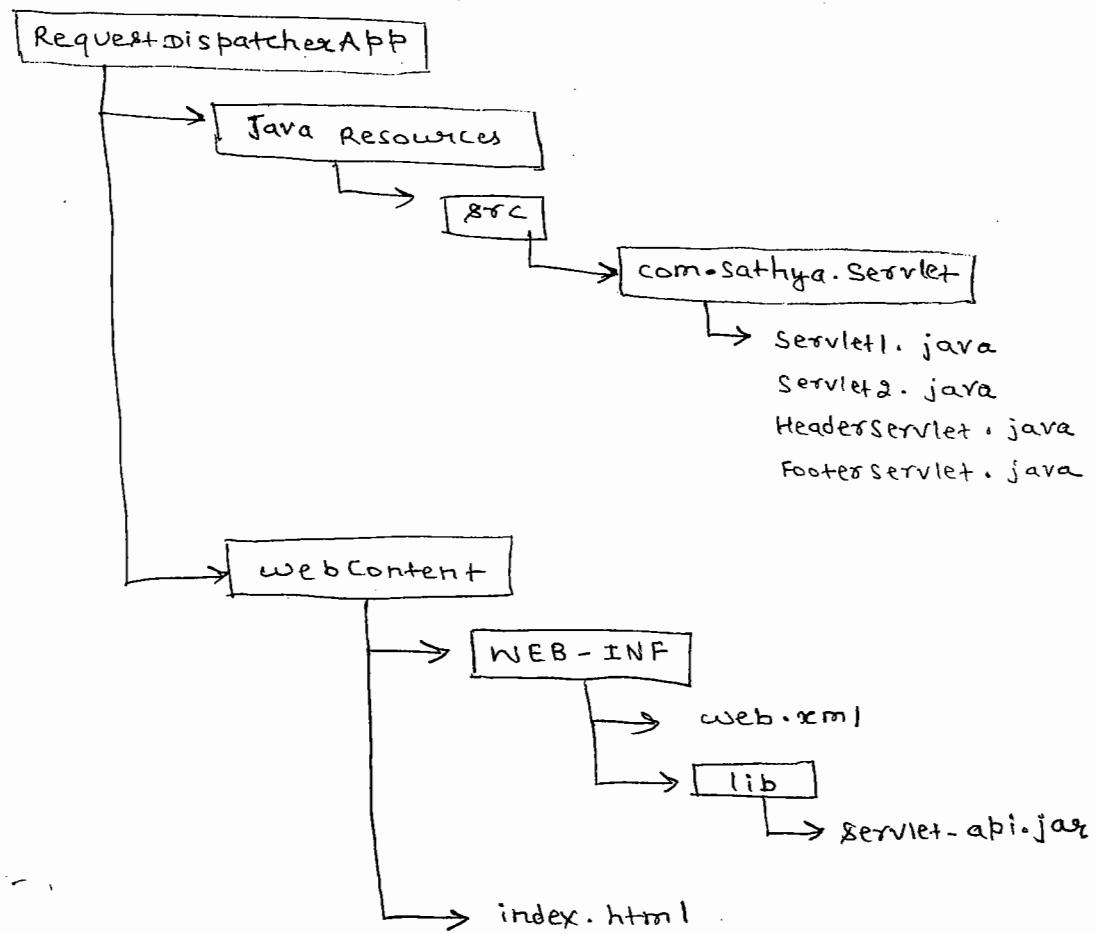
2) If servlet 1 and servlet 2 are into different web application we can use only `context.getRequestDispatcher()`.

### Example



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Eclipse directory structure



### // index.html

```
<center>
<form action = " a/b/srv1">
 Salary : <input type = text name = "t1">

</form>
</center>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

//servlet1.java

public class servlet1 extends HttpServlet
{
 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws
SE,IOE
 {
 // read salary

 int sal = Integer.parseInt(request.getParameter("t1"));

 // calculate da

 double da = sal * 0.10;

 * request.setAttribute("k1", da);

 // obtain RequestDispatcher object

 RequestDispatcher rd = request.getRequestDispatcher
 ("..../c1/gov2");

 rd.forward(request, response);
 }
}

tosept15 //servlet2.java

public class servlet2 extends HttpServlet
{
 protected void doGet (_____, ___) throws IOE, SE
 {
 response.setContentType ("text/html");

 PrintWriter out = response.getWriter();

 // read salary

 int sal = Integer.parseInt (request.getParameter("t1"));

 // read da

 double da = (double) request.getAttribute ("k1");
 }
}

```

```

// find hra
double hra = get * 0.20;
out.println(" <center> <h1> ");
RequestDispatcher rd = request.getRequestDispatcher(".../header");
rd.include(request, response);
out.println("
 ");
out.println(" SAL: " + sal);
out.println("
 ");
out.println(" DA: " + da);
out.println("
 ");
out.println(" HRA: " + hra);
out.println(" <h1> ");
RequestDispatcher rd2 = request.getRequestDispatcher(".../Footer");
rd2.include(request, response);
out.println(" <h1> </center> ");
out.close();
}

}

// HeaderServlet
public class Headerservlet extends HttpServlet
{
protected void doGet(—, —) throws SE, IOE
{
PrintWriter out = response.getWriter();
out.println(" SATHYA TECHNOLOGIES");
}
}

```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Footerservlet

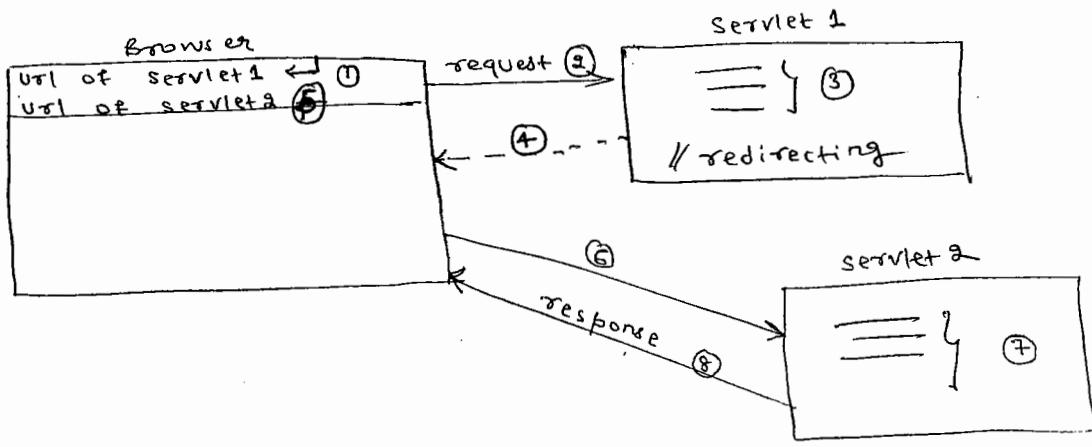
```
public class Footerservlet extends HttpServlet
{
 protected void doget(HttpServletRequest request,
 HttpServletResponse response)
 {
 PrintWriter out = response.getWriter();
 out.println("<copy : \"Ameerpet\"");
 }
}
```

#### Note -

- 1) When creating the servlet of above example, we need to configure the URL pattern like the following -  
  
Servlet 1 : /a/b/srv1  
Servlet 2 : /a/c/srv2  
Headerservlet : /header  
Footerservlet : /footer
- 2) In the above example, we have treated OUT object in servlet 2, header servlet and footer servlet but the three OUT objects are single object (same object only) because response object is same for all the servlet.
- 3) In header servlet if we close OUT object in header servlet then OUT object will close in servlet 2 and footerservlet automatically.
- 4) So, output of footerservlet and servlet 2 will not be printed.

## Servlet Redirection

- 1) Redirecting a request from one servlet to another servlet is also called a servlet chaining or servlet to servlet communication.
- 2) In a redirection process, a servlet will inform the browser about url of the next servlet. So a browser will next automatically send request to new url.
- 3) In this redirection communication, a client (browser) is intimated that your request is redirected from one servlet to another.



- 4) Some examples where we can find redirection are -
  - (i) when a website of a company is changed from old url to new url then redirection is applied. A client types old url and sends the request then website will redirect the request to new website, by informing the browser about the new url of the website.

For example1, if we type www.portal.beamtele.com and if we send the request then that request will be redirected to portal.actcorp.in

#### Example 2

Suppose, we have four servlets in our project and a client must enter into Servlet 1 first before entering into Servlet 2, Servlet 3, Servlet 4. If a client directly send request to Servlet 2, Servlet 3 and Servlet 4 then it is redirected to Servlet 1.

#### Example 3

In online shopping application, after selecting the items for payment through cards, request will be redirected from the online shopping website to payment gateway website.

H-sept

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

11 Sept 15 Difference b/w forwarding and redirection

(My B'day)  
Heavy rain

Forwarding	Redirecting
1) In Forwarding, a request will be forwarded from one servlet to another servlet by without giving any intimation to the client.	1) In redirection, a request will be redirected by giving an intimation to the client.
2) In forwarding, container will not generate another pair of request and response object	2) In redirection, container generates another new pair of request and response objects for destination servlet.
3) In forwarding both source and destination servlet must be on the same server.	3) In redirection, source and destination servlets may be on the same server or may be on the different server.
4) When a request is forwarded then automatically along with the request input values are forwarded.	4) In case of redirection, input values are automatically not redirected.
5) While forwarding, destination must be java enabled resource only (servlet, jsp, html)	5) In redirection, destination can be either java enabled resource or it can be non java resource also (servlet, jsp, php, asp, html etc.)

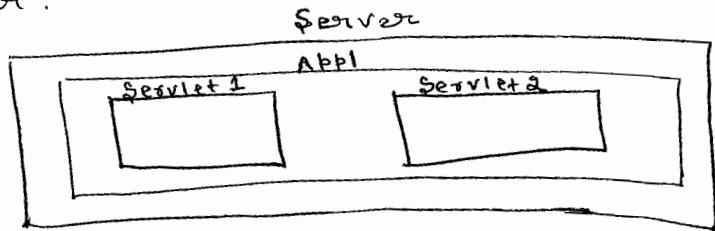
- 5) Forwarding is possible from both genericServlet type and HttpServlet type.
- 6) Redirection is possible  
or only from HttpServlet type.

## SRI RAGHAVENDRA XEROX

Software & Hardware Material Available  
Opposite to Sri Jayagar Bakery,  
Bukampet Road,  
Bukampet, Hyderabad.

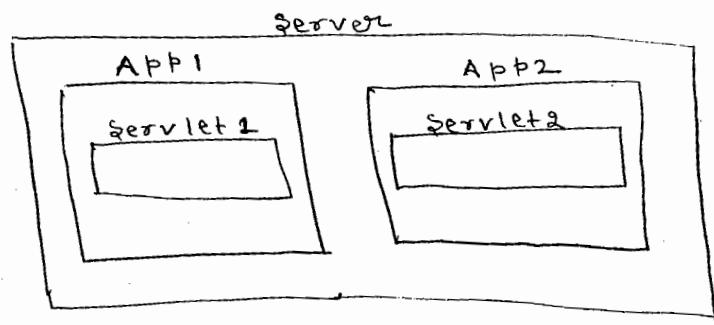
### How to Redirect a Request ?

- 1) To redirect a request we need to call sendRedirect() of HttpServletResponse interface
- 2) If source and destination servlets are resided on same server then we need to pass destination servlet url pattern as a parameter.



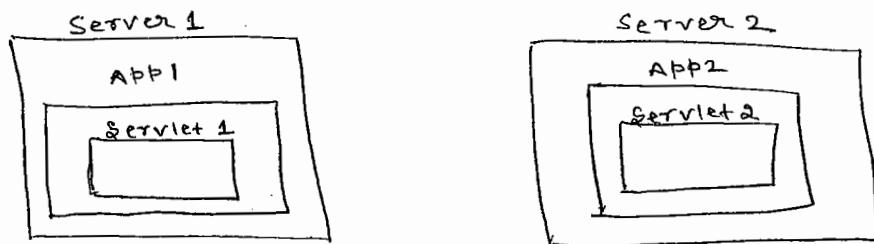
response.sendRedirect("servlet2");

- 3) If source and destination servlets are in two different web application of same server then we need to pass URI of destination servlet as parameter.



response.sendRedirect("/App2/servlet2");

- A) If source and destination servlets are on two different servers then we need to pass url of the destination servlet as a parameter.



`response.sendRedirect("http://localhost:2016/App2/srv2");`

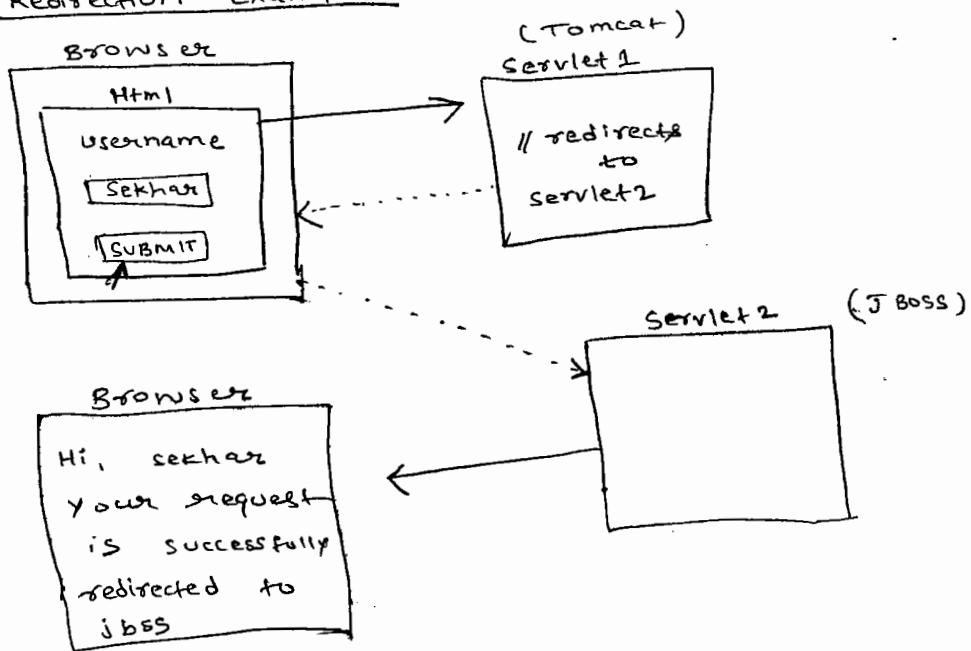
- B) Can we redirect request parameters explicitly?

Ans - Yes, by default request parameters are not redirected but we can explicitly redirect by appending request parameter as a query string.

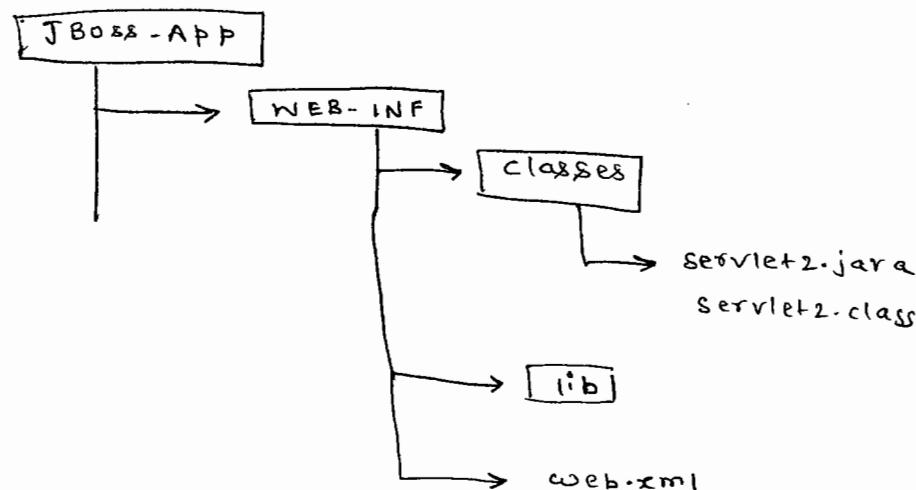
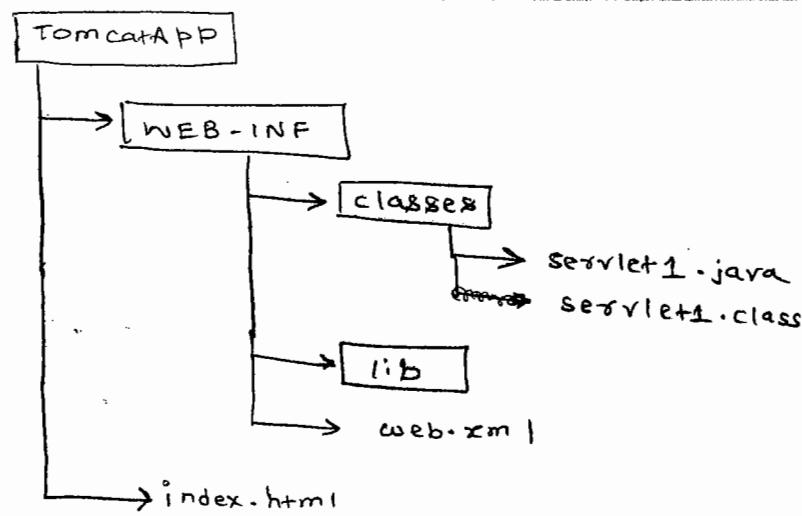
```
String s1 = request.getParameter("uname");
response.sendRedirect("srv2?uname=" + s1);
```

12-sept-15

#### Redirection example



DR. RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.



```

<!-- index.html -->

<center> <form action = "servlet1" method = "post">

 UserName <input type = "text" value = "uname" >

 <input type = "submit" value = "submit" >

 </form>
</center>

```

```

// servlet1.java

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;

public class servlet1 extends HttpServlet
{
 public void doPost(HttpServletRequest request,
 HttpServletResponse response) throws
 SE,
 IOE
 {
 // read Username

 String str = request.getParameter("uname");
 response.sendRedirect("http://localhost:2016/
 jbossApp/srv2?uname
 =" + str);
 }
}

```

### SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### web.xml

<web-app>

<servlet>

<servlet-name> sone </servlet-name>

<servlet-class> servlet1 </servlet-class>

</servlet>

<servlet-mapping>

<servlet-name> sone </servlet-name>

<url-pattern> /srov </url-pattern>

</servlet-mapping>

</web-app>

```

// servlets.java

import javax.servlet.GenericServlet;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;

public class Servlet2 extends GenericServlet
{
 public void service(ServletRequest request,
 ServletResponse response) throws
SE, IOException
 {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 String str = request.getParameter("uname");
 out.println("<h2>");
 out.println("Hi, " + str);
 out.println("
");
 out.println("Your request successfully redirected
to JBoss....");
 out.println("</h2>");
 out.close();
 }
}

```

### web.xml

<web-app>

<servlet>

<servlet-name> stwo

<servlet-name> servt\_swo </servlet-name>

<servlet-class> servlet2 </servlet-class>

</servlet>

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

< servlet-mapping >

< servlet-name > &two </ servlet-name >

< url-pattern > /servlet2 </ url-pattern >

</servlet-mapping >

- 1) compile servlet1.java of tomcat application and servlet2.java of JBoss Application.

C:\TomcatApp\WEB-INF\classes> javac servlet1.java

C:\JBossApp\WEB-INF\classes> javac servlet2.java

- 2) Create war files for tomcat and Jboss separately

D:\> cd TomcatApp

D:\TomcatApp> jar cvf TomcatApp.war .

D:\> cd JBossApp

D:\JBossApp> jar cvf JBossApp.war .

using console deployment process, deploy TomcatApp.war in Tomcat server and JBossApp.war in Jboss server.

- 4) open the browser and type the following request in address bar -

http://localhost:2015/TomcatApp ↵

\*\* Note -

- 1) We can compile a servlet with one server given jar file in classpath and we can execute that servlet in another server.
- 2) For example, we can set weblogic ~~given~~ Server given jar file ( weblogic.jar ) in classpath, we can compile the servlet then we can execute that servlet in Tomcat server.

14 Sept 15 Developing a web application using NetBean IDE (8.0.2)

- 1) NetBeans is an IDE from sun microsystem.
- 2) NetBeans is an open source IDE and it can be downloaded from [www.netbeans.org/download](http://www.netbeans.org/download)
- 3) with NetBean IDE glassfish and tomcat servers are installed automatically.
- 4) glassfish is an Application server, it uses http port as 8080.
- 5) we need to change http port from 8080 to 2020 like the following -
  - (i) open c:\Program Files\glassfish-4.1\glassfish\domains\domain1\config folder

- (ii) copy domain.xml on to the desktop and then open the xml file from the desktop.
- (iii) Find <network-listener port = "8080" and modify the port as "2020"
- (iv) copy domain.xml from desktop back to config folder.

steps -

- 6) (i) Start the netbean IDE by right click on NetBean icon on desktop → run as administrator.

If NetBean icons is not present on desktop then go to

C:\Program Files\NetBeans 8.0.2\bin  
and right click on netbeans application → Select run as administrator

click on file menu → New Project →  
select java web at left side →  
web Application at right side →  
Enter project name : DemoApp →  
select glass-fish server → Enter  
context path : /DemoApp → next  
→ finish

XEROX  
PRINTING MATERIAL Available  
At All Major Bakeries.  
Opp. C.J.S. Bazaar, Ambedkarpet,  
Balkampet, Hyderabad.  
(ii)

(iii) By default index.html page created. delete this file by right click on index.html → delete

(iv) At left navigation, right click on web pages  
→ new → HTML → Enter HTML file name as demo → finish

```
<!-- demo.html -->
```

```
<center>
```

```
<h1>
```

```
<form action = "demosrv" >
```

```
Enter Your Name : <input type = "text"
name = "name" >
```

```
<input type = "submit" value = "click" >
```

```
</form >
```

```
</h1 >
```

```
</center>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

(v) Right click on source packages (at left side) →  
new → servlet → Enter class name : demoservlet  
→ Enter Package → com.sathyas.servlet →  
next → Select Add information to  
deployment descriptor checkbox → Enter  
Servlet name - demo → URL-pattern - /demosrv  
→ finish

```

public class Demoservlet extends HttpServlet
{
 protected void processRequest(HttpServletRequest request,
 HttpServletResponse response) throws SE, IOE
 {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 //read name
 String str = request.getParameter("name");
 out.println("<h1>");
 out.println(" HI " + str);
 out.println("</h1>");
 out.close();
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

(vi) Right click on Project DemoApp →  
 click on Run.

\* 1) Automatically IDE deploy Application,  
 starts a glass fish server, opens  
 browser.

\* 2) Type the following request in address  
 bar —

`http://localhost:2020/DemoApp/demo.html`

## Types of URL patterns

- (i) Exact match
- (ii) Directory match
- (iii) Extension match

(D) Exact match → In this type of URL pattern, servlet container verifies whether URL-pattern of a request is exactly match with URL-pattern of the servlet configured in web.xml or not.

- This type of URL-pattern is used when we want to send the request from one HTML page to one servlet.
- For example,

```
< servlet-mapping >
 < servlet-name > some / < servlet-name >
 < url-pattern > /servlet < /url-pattern >
< /servlet-mapping >
```

### Request :

http://localhost:2015/App1/Serv → invalid

http://localhost:2015/App1/aa/serv → invalid

http://localhost:2015/App1/serv/aa → invalid

http://localhost:2015/App1/serv → valid

15 Sept 15  
(Engineer's day)

(ii) directory match - This type of URL patterns are used when you want to map a request from one group of html pages to one servlet and another group of html pages to another servlet.

- In a directory match url-pattern virtual directories names are used.
- A directory match url-pattern begins with (/) and end with (\*)
- For example, we have ~~an~~ application with two servlet admin servlet and user servlet. If we want to send a request from an admin related pages to Adminservlet and request from all user pages to user servlet then we use directory match url-pattern.

HTML pages } → Adminservlet  
<url-pattern> / proj/admin/\* </url-pattern>

HTML pages } → Userservlet  
<url-pattern> / proj/user/\* </url-pattern>

### For example

< servlet-mapping >

< servlet-name > Some < /servlet-name >

< url-pattern > / catalog / item / \* < /url-pattern >

< servlet-mapping >

can be '0' character  
or any

### Request -

- 1) http://localhost:2015/App1/catalog/item/Sathyaserv //VALID
- 2) http://localhost:2015/App1/Sathyas catalog/item/serv //INVALID
- 3) http://localhost:2015/App1/catalog/item // valid
- 4) http://localhost:2015/App1/catalog/Item) serv // Invalid  
(case-sensitive)

(iii) Extension Match — In a web application, if we want to send request from all html pages of the project to a single servlet then we use extension match.

- Extension match url-pattern begins with \*
- In a url-pattern, if there is no \* then it is a exact match.
- If a url-pattern, ends with \* then it is a directory match.
- If a url-pattern begins with \* then it is called a extension match.

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### For example

< servlet-mapping >

< servlet-name > some </ servlet-name >

< url-pattern > \*.do </ url-pattern >

</ servlet-mapping >

### request :-

- 1) http://localhost:2015/App1/sathyam.do // valid
- 2) http://localhost:2015/App1/a/b/test.do // valid
- 3) http://localhost:2015/App1/do.do.do // invalid
- 4) http://localhost:2015/App1/.do // valid

Q) What is a default servlet?

- Ans - 1) A servlet which is configured with url pattern '/' is called a default servlet.
- 2) For a given request, if no other servlet url-pattern is matched then servlet container will send the request to default servlet.

### Example -

< url-pattern > /servlet1 </ url-pattern > → servlet 1

< url-pattern > /servlet2 </ url-pattern > → servlet 2

< url-pattern > / </ url-pattern > → servlet 3

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Session Management / Session Tracking

- 1) HTTP is the communication protocol b/w a client (browser) and a server.
- 2) From browser HTTP protocol will send the data of the client to the server.
- 3) After sending the data HTTP protocol will remove that data from the protocol's memory (cache).
- 4) If a client is sending a second request then HTTP protocol doesn't remember previous request data. So, HTTP protocol is called stateless protocol.
- 5) In a server, the data sent from browser will be stored in a request object and request is a temporary object it means a request object will be deleted when the response is send back to the browser.
- 6) When a 2nd request comes to the server of a client then that client's first request data is not available in a server. So, it means a server also doesn't remember previous request data of a client. So, by default a server's behaviour is also stateless.

- 7) Session management is a mechanism given to convert stateless behaviour of server and http protocol into stateful behaviour.

16 Sept 15

### Cookies

- i) Session management with cookies works like the following —
  - (i) A client will send the data along with request to the servlet (server).
  - (ii) Servlet reads data from request, creates cookies and stores the data in cookies.
  - (iii) Servlet adds the cookies to response.
  - (iv) Finally, a servlet will send response back to the browser.
- (v) Alongwith response cookies are also come from server and stored on browser.
- (vi) When a client sends second request from browser then browser will automatically adds the cookies to request and it will send request to servlet.
- (vii) Servlet reads previous request values from cookie and current request values from the request object.

- 2) A cookie is an object, it stores data in key / value pair, created on server and send to browser and finally stored on browser.
- 3) In a servlet-api Cookie class is given in javax.servlet.http package for creating cookies in a servlet.
- 4) When creating a cookie key / value both are mandatory and both must be strings only.

Syntax :

```
Cookie c = new cookie(String key, String value);
```

For example

- (i) cookie c = new cookie("user", "sathya"); ✓ (valid)
- (ii) cookie c = new cookie("age", 20); X (invalid)
- (iii) cookie c = new cookie("java"); X (invalid)
- (iv) cookie c = new cookie(); X (invalid)

- 5) Cookies are created in server and to send the cookies to browser we need to add cookie object to response object, by calling addCookie().
- 6) addCookie() is a method of HttpServletResponse interface. so, we can't add cookies to the response object in generic servlet.

- 7) when cookies are send from browser to a servlet in a server, a servlet can read the cookies from request object by calling getCookies().
- 8) getCookies() is a method of `HttpServletRequest` interface. so, we can't get the cookies in generic servlet.
- 9) `getCookies()` returns an array of type `Cookie`.

```
Cookie cookies [] = request.getCookies();
```

### Types of cookies

- 1) In-Memory cookie
- 2) Persistent cookie

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. S.V.C. Balkampet Road,  
Amarapet. H. Bengaluru.

- 1) • An In-Memory cookie means, it is a cookie which stores on browser until that browser is closed.
- In servlet by default every cookie is an in-memory cookie only.
- In-memory cookie will be stored in RAM at client-side. when the browser is closed then automatically the cookies are deleted from RAM.

- 2) • A persistent cookie will be stored in client system hard disk in the form of file.
- The cookies file will be deleted from hard-disk whenever expire time is reached.
  - Before reaching the expire time, even though we close the browser or we shutdown the system but the persistence cookie will not be deleted.
  - In a servlet, if we want to make a cookie as a persistent, we need to set expire time to a cookie in the form of a seconds by calling `setMaxAge()`.

Example,

```
Cookie c1 = new Cookie ("user", "Sekhar");
c1.setMaxAge(1800); // 30 minutes
 ↑
 Seconds
```

- 10) The following are the important methods of cookie class -

(i) getName() → To read the key of a cookie.

```
String key = c1.getName();
```

(ii) getValue() → To read the value of a cookie.

```
String value = c1.getValue();
```

(iii) getValue() → To modify the value of a cookie.

c1.setValue("java"); // "Sekhar"  
"java"  
- After creating a cookie, its key can't be modified but its value can be modified so, we don't have setName().

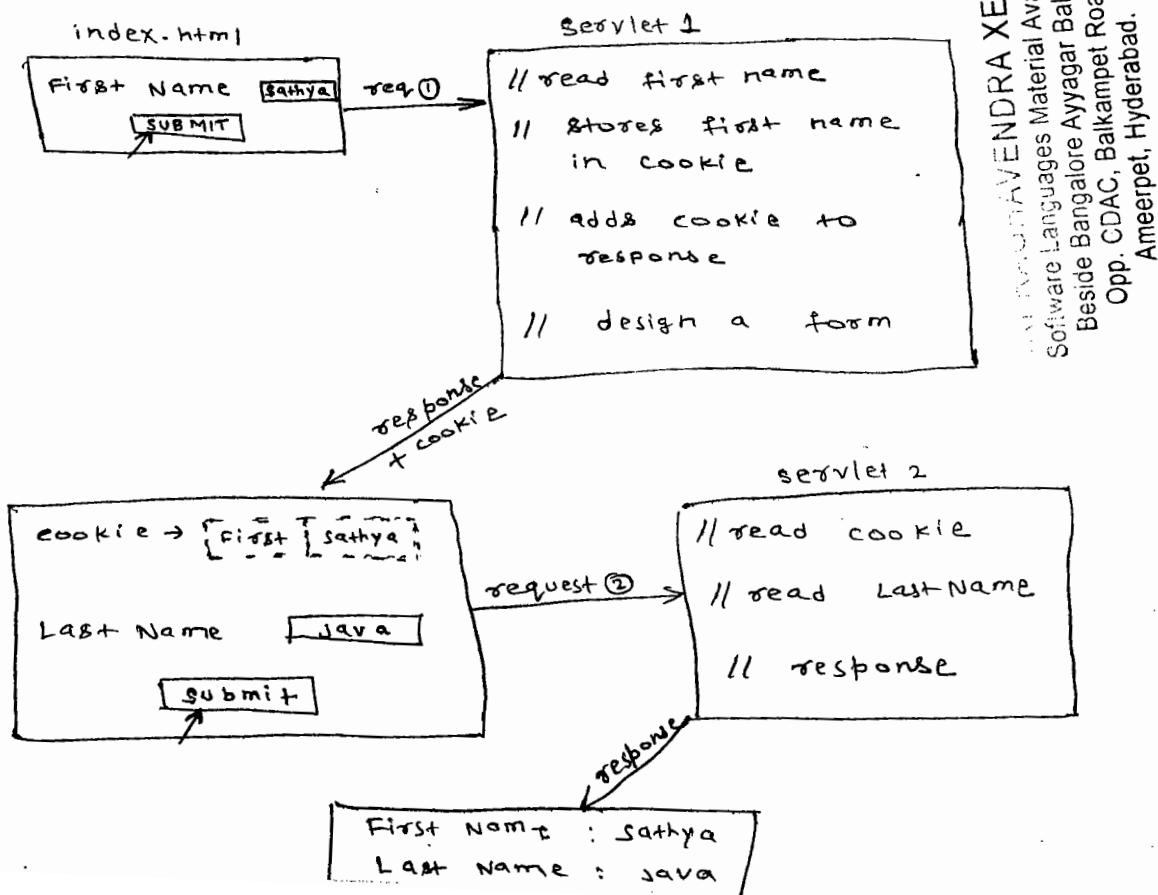
(iv) getMaxAge() → To read expire time of a cookie in seconds.

\* long seconds = c1.getMaxAge();

(v) setMaxAge() → To change expire time of a cookie.

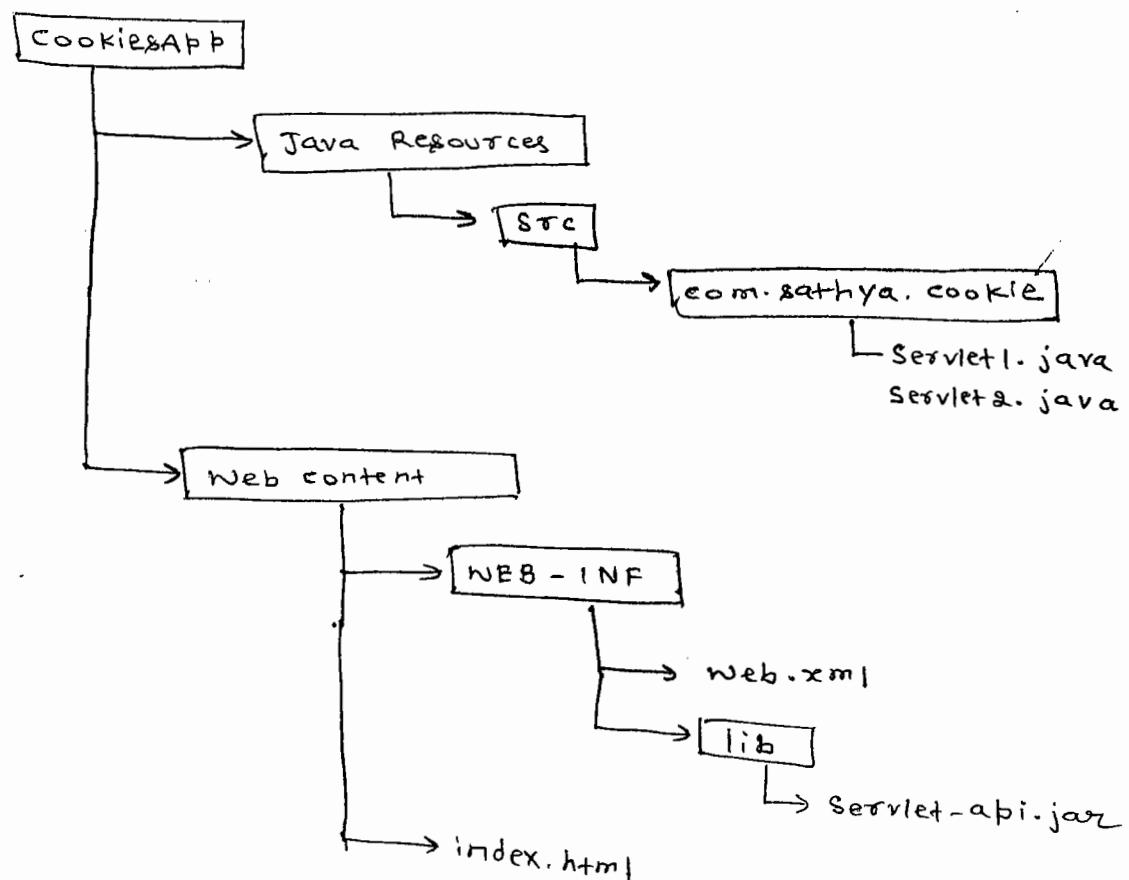
c1.setMaxAge(120);

Example:



18 Sept 15

### Eclipse directory structure



### index.html

```
<center>
<form action = "servlet">
First Name : <input type = text name = "first" >

<input type = submit value = "click" >
</form>
</center>
```

XEROX  
AVENDRA Available,  
Materials, Material Bakery,  
Opp. Lalgudi Jageswari Nagar Road,  
Lalgudi Bangalore, Balkampet Road,  
Opp. C.J.A.C., Hyderabad.  
Opp. Ameerpet, Hyderabad.

SRI RAJAVENDRA XEROX  
Software Languages Material Available  
Beside Bangaliere Ayyagar Bakery,  
Opp. CDAC, Baikampet Road,  
Ameerpet, Hyderabad.

```
// servlet1.java

public class servlet1 extends HttpServlet
{
 protected void doGet(HttpServletRequest, HttpServletResponse)
 throws IOException, ServletException
 {
 // read first name
 String str = request.getParameter("first");

 // create a cookie
 Cookie firstNameCookie = new Cookie("firstname", str);

 // set cookie expire time as 1 minute
 firstNameCookie.setMaxAge(60);

 // add cookie to response
 response.addCookie(firstNameCookie);

 // set mime type
 response.setContentType("text/html");

 PrintWriter out = response.getWriter();
 out.println("<center>");
 out.println("<form action = 'servlet1'>");
 out.println("Last Name : <input type = text
name = 'last'>");
 out.println("
");
 out.println("<input type = submit
value = 'click'>");
 out.println("</form> </center>");
 out.close();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```
// servlet2.java

public class servlet2 extends HttpServlet
{
 protected void doGet(_____, ___) throws SE, IOE
 {
 // read cookie(s) from request

 Cookie cookies[] = request.getCookies();

 // set mime type

 response.setContentType("text/html");

 PrintWriter out = response.getWriter();
 if (cookies != null)
 {
 String name = cookies[0].getName();
 String value = cookies[0].getValue();
 out.println("<h1>");
 out.println(name + " : " + value);
 out.println("</h1>");

 out.println("Y");
 }
 else
 {
 out.println("<h1> first name cookie expired </h1>");
 }

 out.println("
");

 // read last name

 String last = request.getParameter("Last");
 out.println("<h1>");
 out.println("last name : " + last);
 out.println(" </h1>");

 out.close();
 }
}
```

### Drawback with cookie

- \* 1) A browser can block the cookies . If blocked then we will get stateless behaviour but not the stateful behaviour.
- 2) A cookie can store only string type of data . Other types of data is not allowed .
- 3) If more no. of cookies are transferred b/w server and client then network traffic will be increased .
- 4) A browser can store approximately 300 cookies of one server . A Huge no. of cookies can't be stored .

### Hidden Fields

- 1) Like cookies , hidden fields are also created to maintain the state (data) of a client .
- 2) In cookies , a major problem/ drawback , a browser can block the cookies . With hidden fields , this problem is going to be solved .
- 3) Like cookies , we store the data in a hidden field and that hidden field is added to the form .

- 4) In case of cookie, a browser can block it but in case of hidden field a browser can't block hidden field. So, definitely we can achieve stateful behaviour.
- 5) A hidden field is also a text field but it is not going to be visible on browser.
- 6) Hidden fields are created by a using html <form> tag.
- E.g. `<input type = "hidden">`
- 7) For ex, if we want to store First Name in a hidden field then the code will be like the following in a servlet —

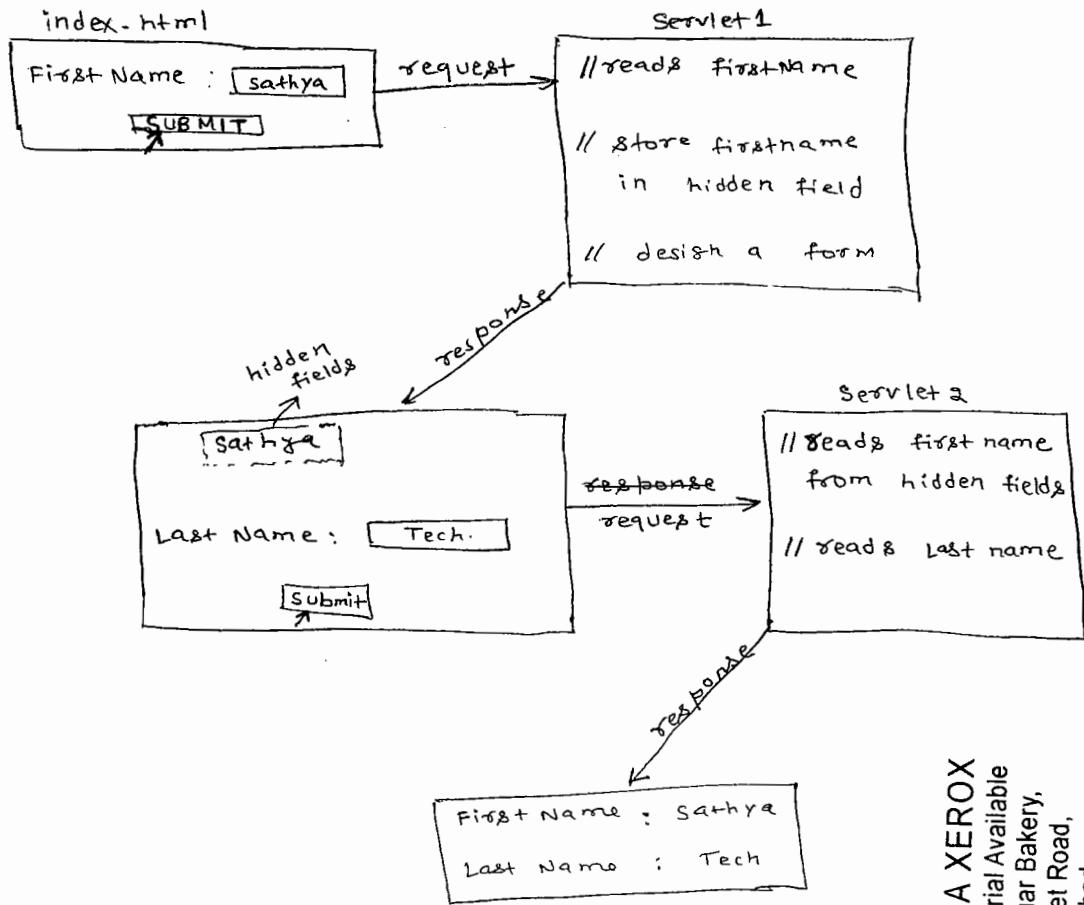
```
String str = request.getParameter("first");
out.println("<form action = 'xyz'>");
out.println(①<input type = "hidden"
 name = 'first'
 value = ①+str+">");
```



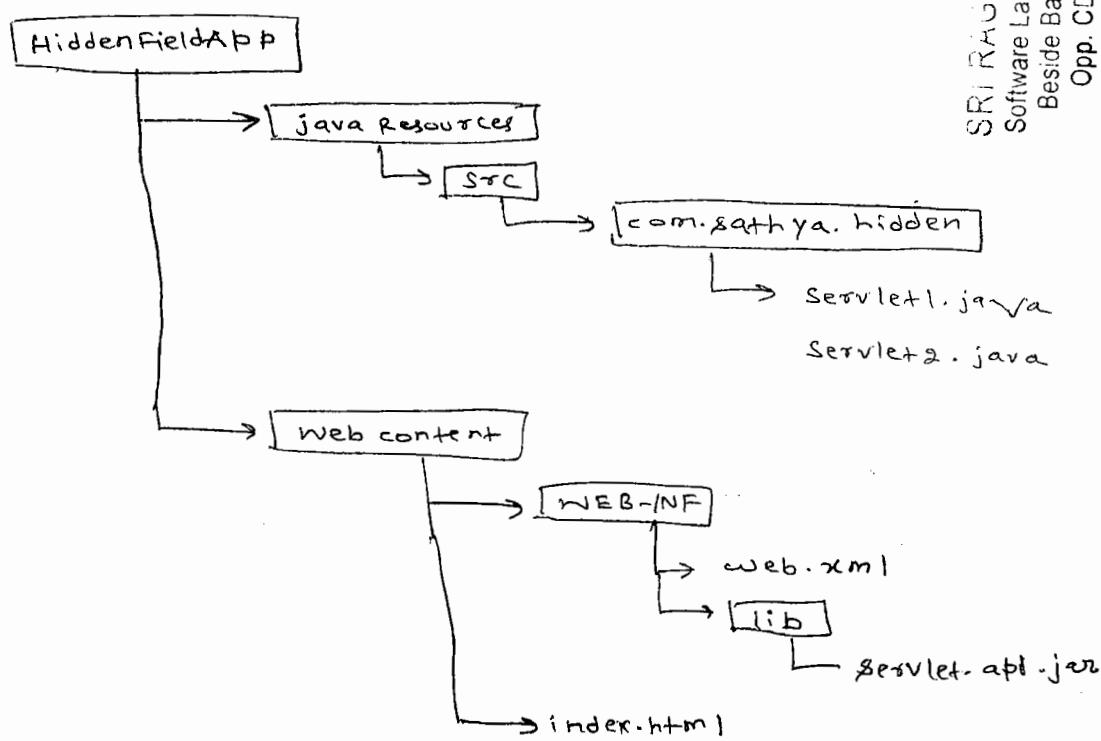
```
out.println("</form>");
out.close();
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Baikampet Road  
Ameerpet, Hyderabad.

19 sebt 15



### directory structure :-



SRI RAMAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## // Servlet1.java

```
public class Servlet1 extends HttpServlet
{
 protected void doGet(_____, ___) throws SE, IOE
 {
 // read the first name
 String str = request.getParameter("first");
 // set mime type
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<center>");
 out.println("<form action ='servlet'>");
 out.println("<input type = hidden name = 'first'"
 value = "+str+">");
 out.println("
");
 out.println("Last Name:
 out.println("<input type = text name = 'last'>");
 out.println("
");
 out.println("<input type = submit value = 'submit'>");
 out.println("</form></center>");
 out.close();
 }
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Aviation Services  
Opp. C.I.T. Engineering College  
Bengaluru - 560034

### //Servlet2.java

```
public class Servlet2 extends HttpServlet
{
 protected void doGet(— , —) throws SE, IOE
 {
 // read firstname and lastname
 String fname = request.getParameter("first");
 String lname = request.getParameter("last");

 // set mime type
 response.setContentType("text/html");

 PrintWriter out = response.getWriter();
 out.println("<center><h1>");
 out.println("First Name : " + fname);
 out.println("
 ");
 out.println(" Last Name : " + lname);
 out.println("</h1> </center> ");
 out.close();
 }
}
```

### Drawback of Hidden Field

- 1) For a hidden field, we can't set the expire time. so, when a browser is closed then automatically hidden fields are removed from the browser.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

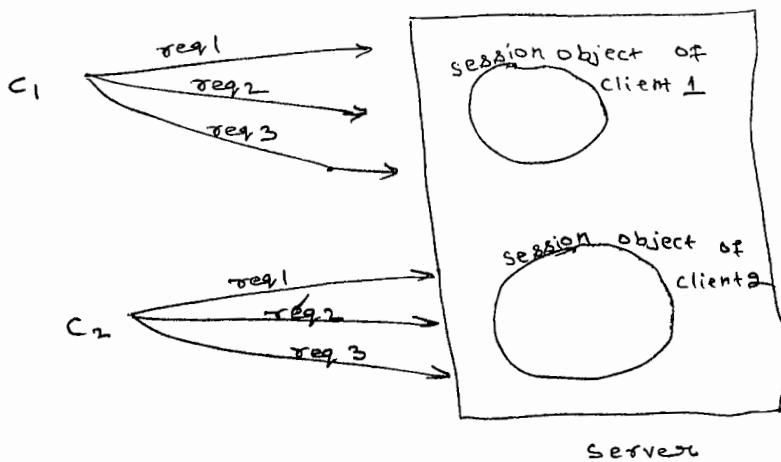
- 2) A hidden field is also a text field, so, we can't store other types of data except string.
- 3) Hidden fields can't be added to the hyperlinks.

### HttpSession API with cookie

- 1) HttpSession is an interface of servlet api, which provides API methods for identifying a user across multiple requests.
- 2) HttpSession technique works like the following—
  - (i) When an user (client) sends request then a server (servlet) verifies whether user is existing or a new user.
  - (ii) If new user then ~~creates~~ server creates a session object and a session id. The session id will be stored in a cookie and a cookie will be added to the response.
  - (iii) with response cookie will be send back to the browser.
  - (iv) If existing user, then server uses existing session object of the client but it doesn't a new session object.

21 sept 15  
(Absent)  
weakness

- 3) Most of the java websites uses HttpSession API with cookie for session management.
- 4) For example, if we disable cookies in a browser then we can not login to facebook and gmail.
- 5) In HttpSession api, for each client a session is started, it means a session object and a session id are created.
- 6) A session object remembers the data of the client, when a client is visiting multiple pages of a website.
- 7) A session id is transferred from server to client, then client to server to identify the client.
- 8) For every client, there will be a session object in the server. so, always number of clients is equal to the no. of session objects.



## Obtaining a session object

- 1) In HttpServletRequest we have two methods to obtain a session object.

(i) `getsession()`  
with  
`getsession(boolean)`

`HttpSession session = request.getSession();`

`HttpSession session = request.getSession(true);`

`HttpSession session = request.getSession(false);`

- 2) `getSession()` and `getsession(true)` works as same.

Both methods returns current session object of request. If there is no session object for request then creates a new one.

- 3) `getsession(false)` returns current session object.

If request doesn't contains a session object then returns null.

## Methods to store/Maintain the data -

- 1) In a session object, data of a particular client is maintained at server.
- 2) The following methods are used for storing or reading or deleting the data from a session object.

- (i) `setAttribute( k, v )`
- (ii) `getAttribute( key )`
- (iii) `removeAttribute( key )`
- (iv) `getAttributeNames( )`

SRI RAJAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### Methods to Read additional information of Session object -

- 1) `getId()` - This method returns session id.  
 A session id is a alphanumeric string.

```
String id = session.getId();
```

- 2) `isNew()` - It returns true for a new session and false for existing session.

```
boolean flag = session isNew();
```

#### Ex -

```

if (flag == true)
{
 sop ("client is a new client");
}

else
{
 sop ("client is a existing client");
}
```

22 Sept 15

④ SetMaxInactiveInterval() -

- (1) This method is used to set the inactive time period for a session.
- (2) If a client is not coming back to the server within the inactive time period then a session of a client will be removed from the server.
- (3) By default, a server will set some inactive time period for every session. When a session is newly started.
- (4) For example, in Tomcat inactive time is will be set as 30 min and in weblogic it will be set as 60 min.
- (5) We can change the default inactive period in two ways -
  - (1) Programmatically
  - (2) Declaratively
- (6) Programmatically means, by calling `SetMaxInactiveInterval()` for example, `session.SetMaxInactiveInterval(10 * 60)`  
 $\uparrow \text{10min}$   
 $\downarrow \text{600sec}$
- (7) Declarative means, by configuring `<session-config>` tag in `web.xml`

```
<session-config>
 <session-timeout> 5min </session-timeout>
</session-config>
```

(8) In programmatic, we can set in seconds  
but in declarative we can set only  
in minutes.

(9) If we set in both ways then always  
programmatic value overrides the declarative  
value.

#### ⑤ getCreationTime()

(1) This method returns session creation time  
in milliseconds.

(2) We convert milliseconds to date format  
by passing milliseconds to date class  
constructor.

```
long ms = session.getCreationTime();
Date d = new Date(ms);
```

#### ⑥ getLastAccessedTime()

(1) This method returns when the a session  
is last accessed by the user in milliseconds.

(2) We need to convert milliseconds to date  
format by passing the milliseconds to  
date class constructor.

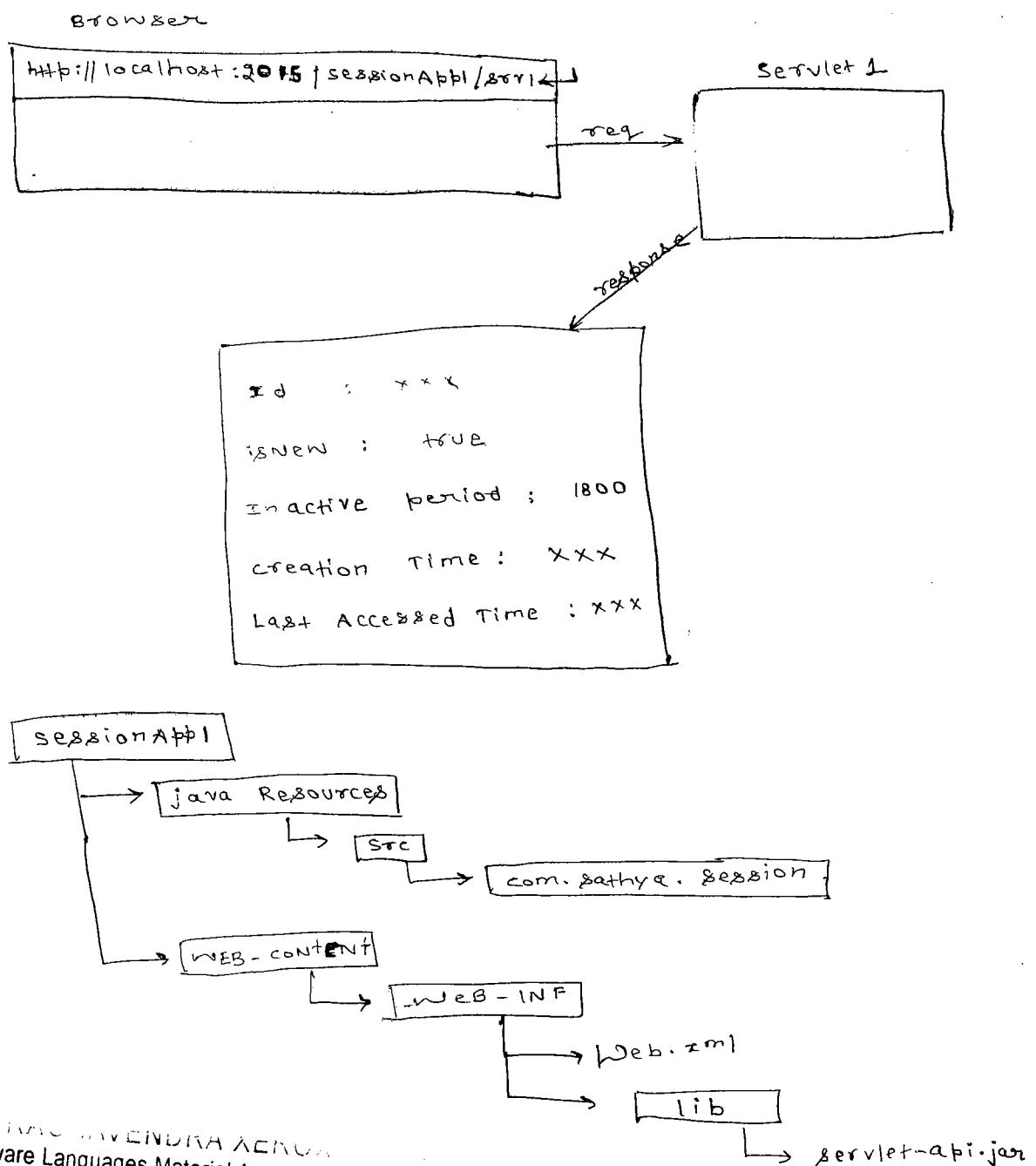
```
long ms = session.getLastAccessedTime();
Date d = new Date(ms);
```

## ⑦ invalidate()

- (ii) This method is to invalidate a session when a client is quitting / exit / logout from a website.

### Example

The following example contains a servlet, which displays a session information on browser.



```

public class Servlet1 extends HttpServlet
{
 protected void doGet(_____, ___) throws SE, IOE
 {
 // get session object
 HttpSession session = request.getSession(true);

 // set mime type
 response.setContentType("text/html");

 PrintWriter out = response.getWriter();
 out.println("Id : " + session.getId());
 out.println("
");
 out.println("isNew : " + session.isNew());
 out.println("
");

 out.println("inactive period :" + session.getMaxInactiveInterval());
 out.println("
");

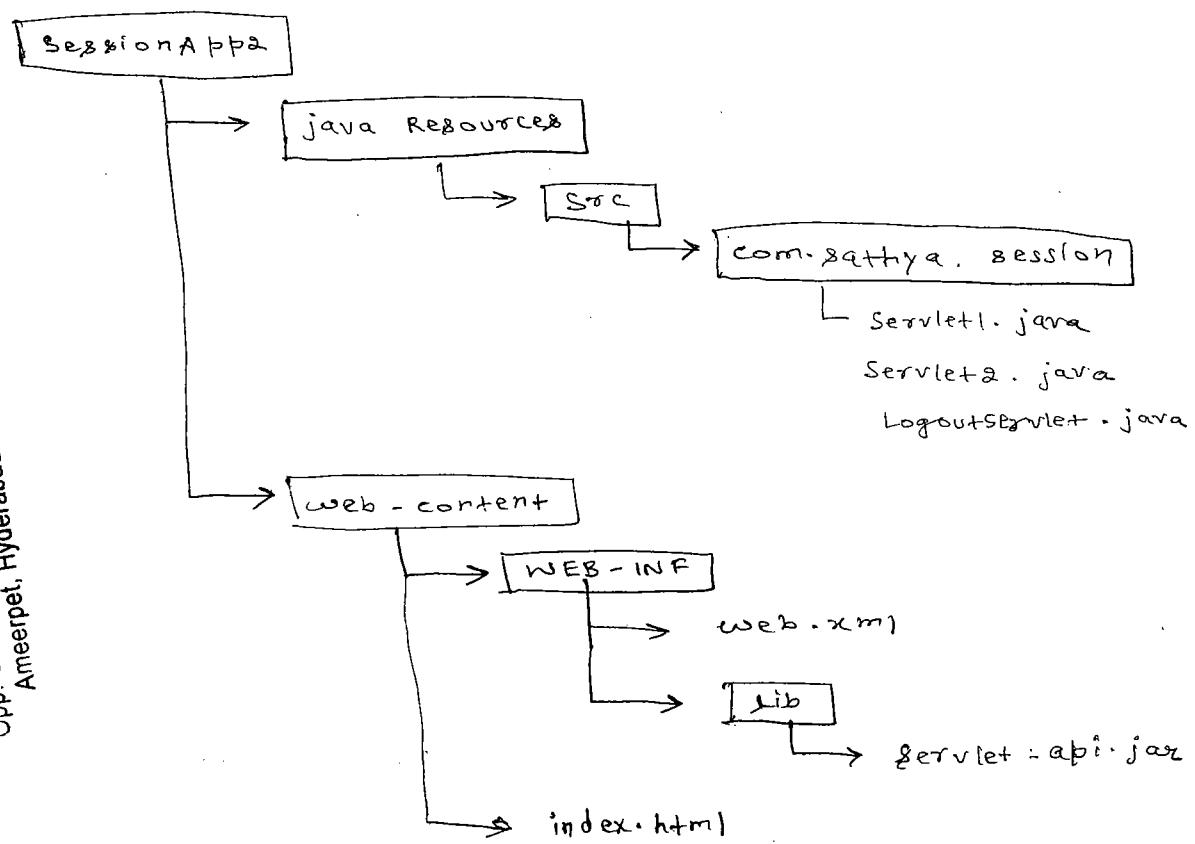
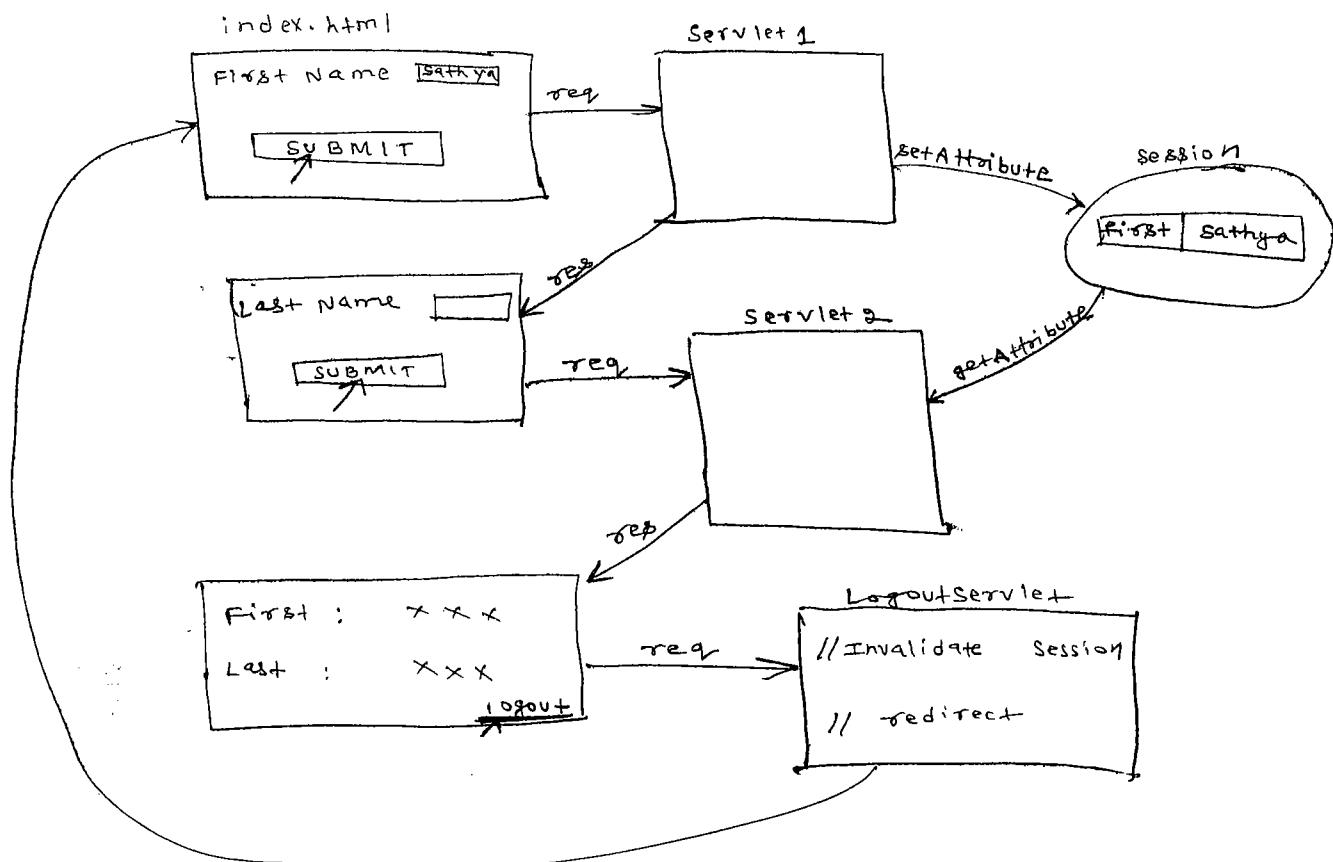
 long ms = session.getCreationTime();
 java.util.Date creationDate = new java.util.Date(ms);
 out.println("creation date : " + creationDate);
 out.println("
");

 long ms2 = session.getLastAccessedTime();
 java.util.Date lastAccessedDate = new java.util.Date(ms2);
 out.println("Last Accessed date = " + lastAccessedDate);
 out.close();
 }
}

```

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### Example 2



**SRI RAGHAVENDRA XEROX**  
Available  
Software Languages Material Bakery,  
opp. Bangalore Ayyagar Bakery,  
Bandra, Mumbai, Maharashtra  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

23 Sept 15

### index.html

```
<center>
<form action = "servlet">
```

```
First Name : <input type = text name = "first" >

<input type = submit value = "click" >
</form>
</center>
```

### Servlet1.java

```
public class Servlet1 extends HttpServlet
{
 protected void doGet(— , —) throws SE, IOException
 {
 // read first name
 String str = request.getParameter("first");
 // get session object
 HttpSession session = request.getSession();
 // store data in session object
 session.setAttribute("first", str);
 // get mime type
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("Your session id :" + session.getId());
 out.println("<center>");
 out.println(" <form action = 'servlet'> ");
 out.println(" Last Name : <input type = text name = 'last'> ");
 out.println("
 ");
 out.println(" <input type = submit value = 'click' > ");
 out.println(" </form> </center> ");
 out.close();
 }
}
```

VENKATESWARA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp CDAC, Balkampet Road,  
Ameerpet, Hyderabad

### Servlet2.java

```
public class servlet2 extends HttpServlet
{
 protected void doGet(HttpServletRequest, HttpServletResponse) throws ServletException, IOException
 {
 // get session object
 HttpSession session = request.getSession();
 // read first name from session object
 String str1 = (String) session.getAttribute("first");
 // read last name from request
 String str2 = request.getParameter("last");
 // set mime type
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("your session id :" + session.getId());
 out.println("<center>");
 out.println("First Name : " + str1);
 out.println("
");
 out.println("Last Name : " + str2);
 out.println("
");
 out.println(" Logout ");
 out.println("</center>");
 out.close();
 }
}
```

SRI KAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Logout servlet

```
public class LogoutServlet extends HttpServlet
{
 protected void doGet(— , —) throws ServletException, IOException
 {
 // get session object

 HttpSession session = request.getSession();

 // invalidate

 session.invalidate();

 // redirect to index.html
 response.sendRedirect("index.html");
 }
}
```

### HttpSession API with URL rewriting

- 1) In HttpSession with cookie, session id will be send in the form of a cookie from server to browser.
- 2) If browser blocks cookies, then we will get stateless behaviour but not statefull behaviour.
- 3) In order to solve the above problem session id will be appended to each url-pattern i.e. coming from server to browser.
- 4) From browser when a user submits a next request then along with url pattern session id is also submitted, with that session id server recognizes a client as a existing client.

VENKRA XEROX  
Copies & Images Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- s) In order to append the session id to url pattern a method is provided in `HttpServletResponse` interface called `encodeURL()`.
- 6) Suppose, if we want to convert the previous application to url rewriting technique then we need to do the following two changes —
- In `servlet1.java`, change the form tag like the following —

```
out.println("<form action = " + response.
encodeURL("srv1") + ">");
```
  - In `servlets.java`, change hyperlink like the following —

```
out.println("<a href = " + response.encodeURL
("logoutsrv") + "> Logout ");
```

\* Q - How can we say that URL rewriting is applied in a website or not ?

Ans - If session id is displayed in address bar then we can say that url rewriting technique is applied in that website.

\* Q - What are the different types of servlet ?

Ans - 2 types —

- generic servlet
- Http servlet

Q What is a difference b/w generic servlet and HttpServlet?

Ans.

Generic Servlet	HttpServlet
(1) Generic servlet is a protocol independent. It means generic servlet accepts all protocol request.	(1) HttpServlet is a protocol dependent. It means, it will accept only http protocol request.
(2) Redirect is not possible	(2) possible
(3) session management with cookies and HttpSession api is not possible	(3) possible

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

24 Sept 15

## Web - logic Server

Type : Application server

Vendor : oracle corporation

Version : 12c (12.1.3)

Website : <http://www.oracle.com/technetwork/middleware/download/index-087510.html>

Default location : C:\Oracle\Middleware\wlserver

Default port : 7001

- 1) A jar file is downloaded with the name fmw\_12.1.3.0.0\_wls.jar
  - 2) Extract the jar file, then a folder a is created. Under that open disk1 folder and double click on install. So, weblogic server will be installed.
  - 3) Type the following command for installing weblogic server.  
C:\> java -jar fmw\_12.1.3.0.0\_wls.jar ↳  
Creating a domain ↳ name of jar file
- 1) In Application server we need atleast one domain for deploying our application.
  - 2) In other Application server like jboss, glassfish etc automatically one domain is created alongwith the server installation.

3) In web-logic server we need to explicitly create a domain.

Step1 :- windows start button → Programs → Oracle → OracleHome → Weblogic Server 12c  
→ Tools → configuration wizard

Step2 :- Select create a new domain and change the name base\_domain with domain\_spm .

Step3 :- click on next button → next →  
Enter Name   
Password   
confirm Password   
→ next → next → next

4) The location of the ~~after~~ domain created is

C:\oracle\Middleware\Oracle\_Home\user\_projects\domains\domain\_spm

#### Hand deployment in web-logic

i) copy a web application root directory into

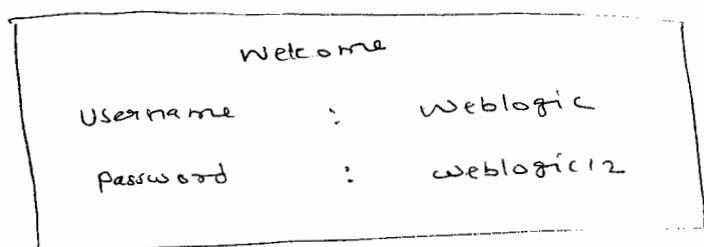
C:\oracle\Middleware\Oracle\_Home\user\_projects\domains\domain\_spm\autodeploy folder

- 2) In domain.xml folder, double click on start weblogic server (run as administrator)
- 3) open the browser and type the following url —  
<http://localhost:7001/Application/htmlName>

### Console deployment in weblogic

Step -

- 1) open Eclipse and right click on sessionApp project → select Export → War file → Browse button and select F: Drive → OK
- 2) Start the weblogic server
- 3) open the browser and type the following request —  
<localhost:7001/console/login/LoginForm.jsp> ↪



- 4) click on deployment at left view → install button → upload your files link → click on choose file button → select war file (sessionApp.war) → next → next → next → finish

- 5) click on logout link.
- 6) open the browser and send the request like the following -

`http://localhost:7001/sessionApp1/8881`

### Difference b/w a web server and Application Server -

Webserver	Application server
1) A webserver doesn't allow to create a domain. so, the settings defined in the server will be commonly applied to all the projects.	1) In application server domains are created. so, settings done in one domain for a project will not be applied to another domain.
2) A webserver has only web container.	2) An Application server has web container and EJB container also.
3) A webserver only executes war files.	3) An application server executes war, jar, ear files.
4) A webserver provides less middleware services. For ex, it doesn't provide transaction and messaging support. Ex → Apache Tomcat, Resin	4) An Application server provides more middleware services. For ex, it provides transaction and messaging. websphere Ex → weblogic, jboss, Glassfish

25 Sept 15

## File Uploading in servlet

- 1) File Uploading is nothing but transferring a file from client system to server system.
- 2) When a file is transferred then browser will add the file name and file data to the request.
- 3) While creating html page, the following changes are needed in design of a form.
  - (i) A browser can upload the file only through post method. So, we need to change method in form tag as post.
  - (ii) By default, browser encrypts data before sending to the server. To tell the browser that don't encrypt a file data, we need to change enctype attribute value as multipart/form-data.
  - (iii) In order to select a file for uploading it a browse button is required on browser. So, we add input tag to the form of type file.

```
<form action = "govi" method = "post"
enctype = "multipart/form-data">
```

Select File : <input type = "file" name = "file">

<input type = "submit" value = "upload" >

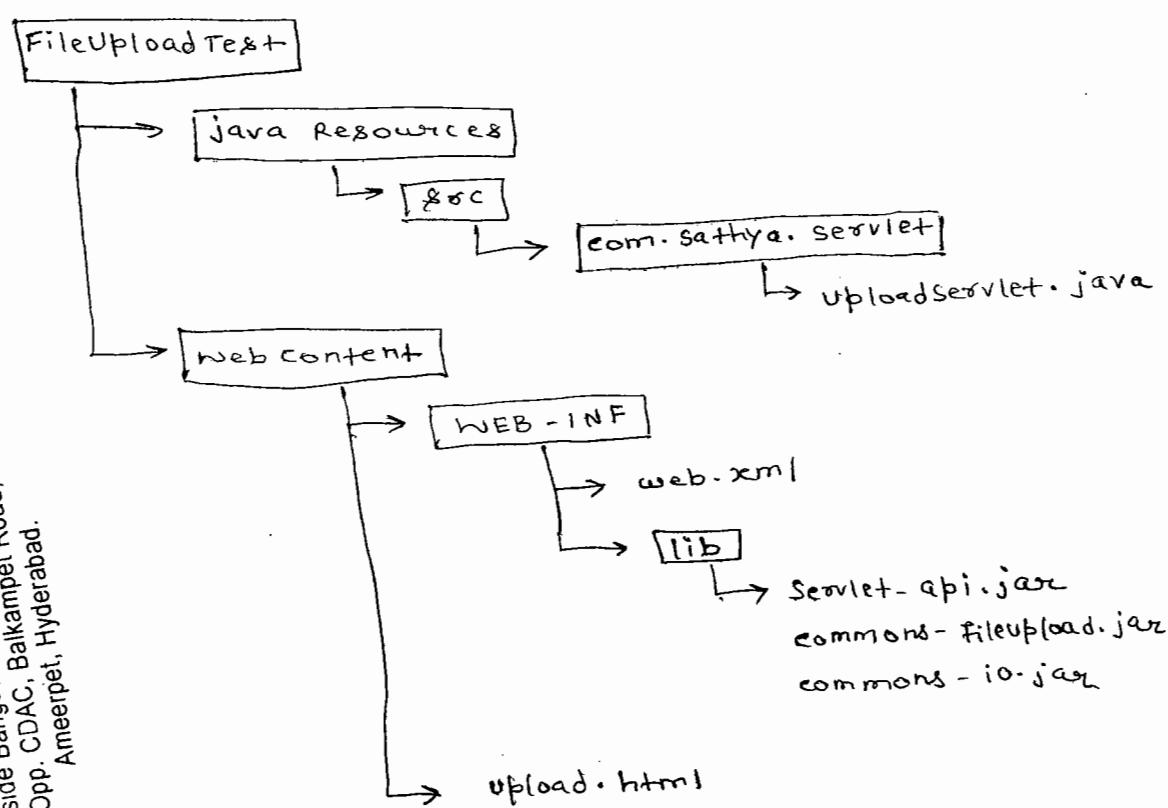
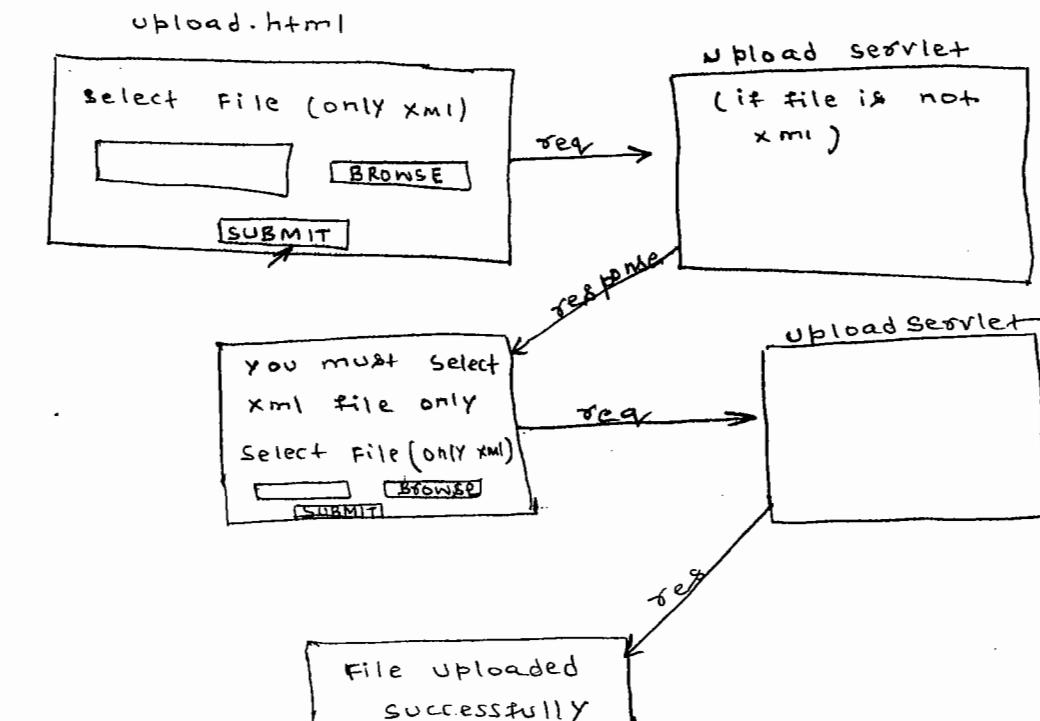
</form>

- 4) In servlet-api, there are no predefined classes or interfaces provided, to read the uploaded filename and filedata. So, we take the support of 3rd party api to read uploaded file and its data from the request.
- 5) We can use third party api like —
- commons - fileupload api (from Apache vendor)
  - javazoom api ... etc
- 6) In all the real time applications we use commons - fileupload api of Apache
- 7) In a Servlet program, we need to create the following two classes object —
- (i) DiskFileItemFactory → class
  - (ii) ServletFileUpload → class
- 8) While creating ServletFileUpload class object, we need to pass DiskFileItemFactory class object as parameter.
- ```
DiskFileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
```
- 9) In ServletFileUpload class we have a parseRequest() and this method reads each item/value from request, creates a FileItem object, stores FileItem objects in a list and finally returns list.

List list = upload.parseRequest(req);
 contains FileItem objects

as per this

Example



upload.html

```
<center>
<form action = "uploadServlet" method = "post"
      enctype = "multipart/form-data">
  Select a File (only xml) :
  <input type = file name = "file"> <br>
  <input type = submit value = "submit" >
```

uploadServlet.java

```
public class uploadServlet extends HttpServlet
{
    protected void doPost( — , — ) throws SE, IOE
    {
        PrintWriter out = response.getWriter();
        DiskFileItemFactory itemFactory = new DiskFileItemFactory();
        ServletFileUpload
        ServletFileUpload upload = new ServletFileUpload(itemFactory);
        try
        {
            List list = upload.parseRequest( request );
            Iterator it = list.iterator();
            while( it.hasNext() )
            {
                FileItem fi = (FileItem) it.next();
                if( ! fi.getContentType().equals("text/xml") )
                    throw new Exception();
                if( fi.isFormField() == false )
                {
                    String fileName = fi.getName();
                    File file = new File("F:/"+fileName);
                    fi.write(file);
                    out.println("U & file uploaded successfully");
                }
            }
        }
```

```

        catch ( Exception e )
        {
            out.println(" <center> <font color = 'red' >
                You MUST select XML file only !! </font> ");
            RequestDispatcher rd = request.getRequestDispatcher
                ("upload.html");
            rd.include ( request, response );
        }
        out.close();
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Q- How many files can be uploaded at a time to a server ?

Ans - i) It is not based on the no. of file.
 It depends on size.

ii) For ex, Tomcat doesn't allow to upload a file or files whose size exceeds 10 MB.
 It will be different from one server to another.

Differences b/w GET and POST methods

GET	POST
1) GET is not a secure method because it will show a form data in address bar.	2) POST is a secure method because it doesn't show a form data in address bar.
2) GET method doesn't support file uploading.	2) POST method supports file uploading.
3) GET method can't send huge amount of data at a time to the server. Approximately 1 KB of data only it can send at a time.	3) POST method can send large amount of data at a time.
4) GET is designed for getting resources (files) from server.	4) POST method is designed for posting the data to the server.
5) GET method is called idempotent method.	5) POST method is called non-idempotent method.

SRI RAGHAVENDRA XEROX

Notes

Idempotent method means, a request send to the server with that method will not do any changes to the data in server.

For ex, by using get() if we send the request to the server to get html page from server to browser then this request will not modify any data at server.

Non-idempotent method means, a request send to the server with this method will do modification with data at server.

For ex, if we send username and password through post() then in server the previous username and password values are replaced with new values.

28 Sept 15 Request and Response Headers

- 1) When a request is send from a browser to a server then a browser creates a request document and it will send it to server.
- 2) When a server is sending response then server prepares document and it will send it to the browser.

- 3) A request and Response document contains two parts called header and body.
- 4) A sample request document is created, when request send to home.html page from browser will be like the following -

request document

```

GET /APP1/home.html HTTP/1.1
Host : localhost : 2015
User-Agent : Mozilla/5.0
Accept-Language : en-US
Connection : keep-alive
  
```

The diagram shows a rectangular box divided into two horizontal sections. The top section is labeled "Header" with an arrow pointing to the first four lines of the request. The bottom section is labeled "Body" with an arrow pointing to the empty space below "Connection".

- 5) When request send through get(), body part of the request will be always empty.
- 6) A server prepares response document like the following

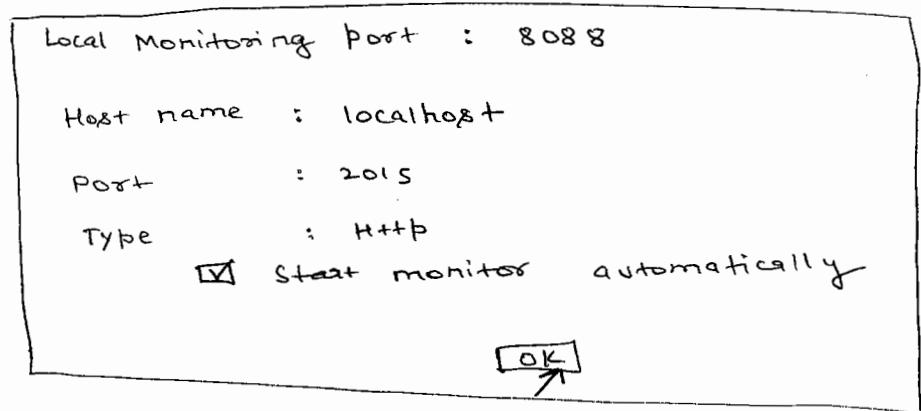
response document

```

HTTP/1.1 200 OK
Host : localhost : 2015
Content-Type : text/html
Content-Length : 87 (bytes)
Date : MON 28 SEP 2015
<html>
  <body>
    ==
  </body>
</html>
  
```

The diagram shows a rectangular box divided into two horizontal sections. The top section is labeled "Header" with an arrow pointing to the first five lines of the response. The bottom section is labeled "Body" with an arrow pointing to the "html" and "body" tags.

- 7) we can observe a request document and response document transferred b/w browser and server through TCP/IP Monitor in Eclipse.
- 8) In order to track the request and response document, we need to send the request to server through TCP/IP Monitor.
- 9) A request goes from browser to TCP/IP monitor and monitor will forward the request to server.
- 10) We can add the TCP/IP monitor in eclipse like the following -
- (i) click on window menu → show view → other → Expand debug → select TCP/IP Monitor
- (ii) click on view menu → show Header
- (iii) click on view menu → properties → add button



- 11) By sending the request from browser to server we need to use port no. as 8088

Servlet thread Safety

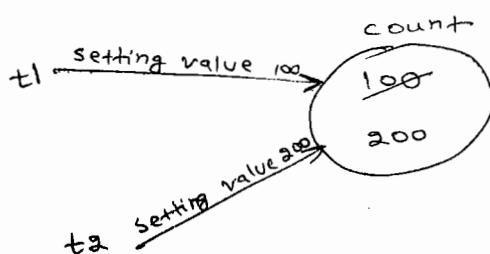
1) Every servlet is a class which follows single instance (Object) and multiple threads model. It means one object of servlet class can be shared by multiple threads.

* 2) By default a servlet is not a thread-safe. Because it can have instance variable or static variable and if one thread changes instance or static variable value then it affects on the remaining concurrent thread sharing that servlet object.

3)

```
public class MyServlet extends HttpServlet
{
    private int count; // instance variable
    public void service( request, response ) throws
        SE, IOException
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



When t2 set value as 200 then it affects on t1 also. So, a servlet is not thread safe by default.

29 Sept
Aman Arun
B'Day

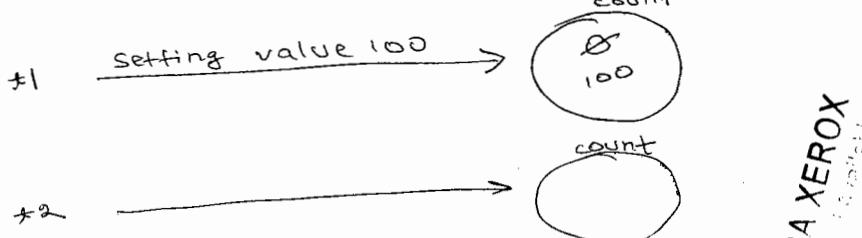
- 4) we can make a servlet object as thread safe object in 3 ways -

Way 1

- 1) If possible avoid instance variable and static variables in a servlet class and in place of them create local variable in service()
- 2) For local variables, memory is allocated to each thread separately. So, if one thread changes a variable value then it doesn't affect on remaining concurrent threads.

For ex,

```
public class MyServlet extends GenericServlet
{
    public void service( req, res ) throws SE, IOE
    {
        int count = 0; // local variable
    }
}
```



ways

- 1) sometimes, it is not possible to avoid instance variable in servlet class. Because same variable is required in multiple methods.
- 2) In servlet-api, singleThreadModel interface is provided(Marker) in javax.servlet package. We need to implement our class from singleThreadModel interface.
- 3) Implementing singleThreadModel interface is nothing but making service() of a servlet as synchronized method.
- 4) A servlet container allows only one thread to access the servlet object at a time. so, a servlet object becomes thread safe.
- 5) A problem in this approach is waiting time for the next thread will be increased, so that an application performance will be decreased.

for ex,

```
public class Myservlet extends GenericServlet  
    implements SingleThreadModel  
{ public void service( req, res ) throws SE, IOE  
    {  
        }  
    }  
}
```

ways

- 1) suppose, service() has 10 lines of code and in that thread safety problem is occurring because of 5 lines of code.
- 2) If we make service() as a synchronized method then a next thread has to wait until previous thread executes all the 10 lines of code.
- 3) To reduce the waiting time, we can use a synchronized block in service().

for ex,

```
public class MyServlet extends GenericServlet
{
    public void service( req, res ) throws SE, IOE
    {
        // 1
        // 2
        // 3
        synchronized( this )
        {
            // 4
            // 5
            // 6
            // 7
            // 8
        }
        // 9
        // 10
    }
}
```

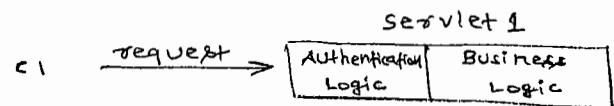
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

30 Sept 15

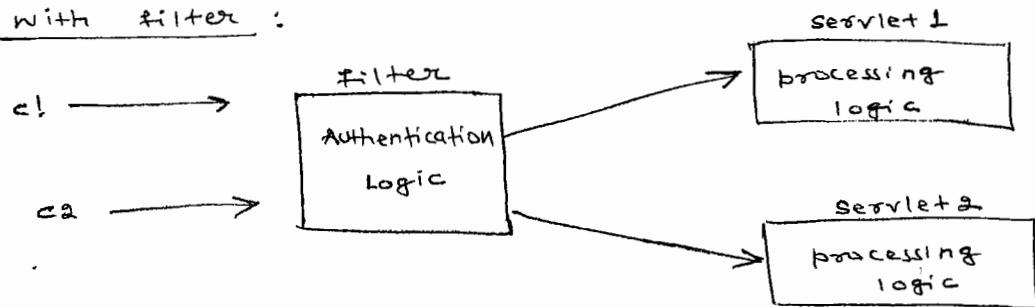
Filters in Web Application

- 1) Filter is a new component added in servlet version 2.3
- 2) A filter is a class which looks like a servlet and it contains pre-processing or post-processing or both logics of a servlet.
- 3) A filter intercepts a request and a response and executes pre-processing logic from request and post-processing logic on response.
- 4) The logic defined in a filter is called a services logic and the logic defined in servlet is the business logic.
- 5) A benefit of creating filters is, we can modularize a service and we can apply that service for multiple servlets through a filter.
- 6) For ex, we have two servlets with different business logics and having common authentication logic.
- 7) In each servlet, we need to define authentication logic separately if we don't use a filter but if we use filter then we can define authentication logic in filter then we can apply that filter for both the servlets.

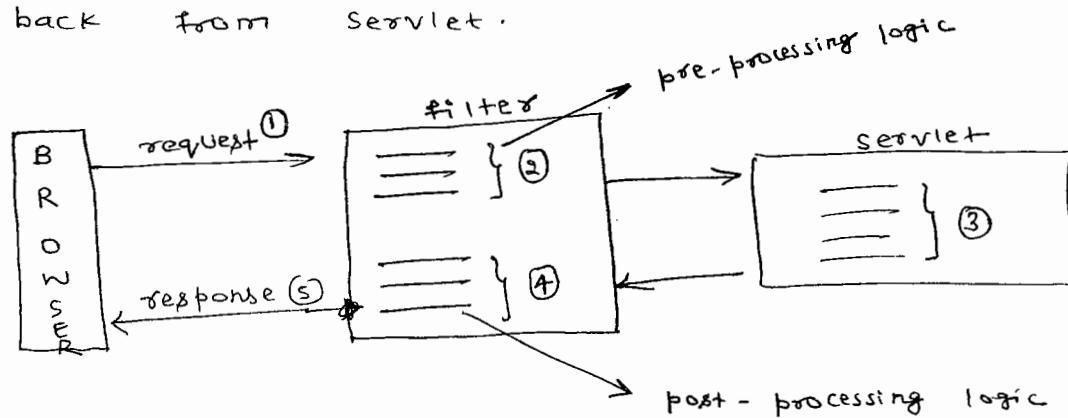
without filter :



with filter :

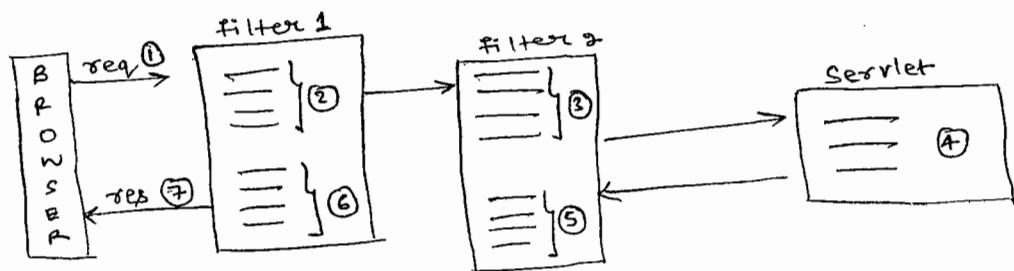


- 8) Pre-processing logic defined in a filter is executed when a request is going to a servlet and Post-processing logic of filter is executed when response is coming back from servlet.



SKI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- g) A Web Application can have multiple filters also. Pre-processing logics of filter is executed in 1 to n order and post processing logic of filters are executed from n to 1 order.



How to create a Filter?

- 1) To create a filter, we need to implement our class from Filter interface.
- 2) Filter interface has 3 abstract methods. So, we must override all the 3 abstract methods.
- 3) The three abstract methods are called life cycle methods of filter.

```

public class MyFilter implements Filter {
    public void init( FilterConfig fc ) {
    }

    public void doFilter( ServletRequest request ,
        ServletResponse response , FilterChain chain ) throws
        SE, IOE
    }

    public void destroy() {
    }
}

```

- (4) `init()` and `destroy()` methods are called once and `doFilter()` is called for each request.
- (5) A Filter is almost equal to a servlet because both life cycles are managed by container.
- (6) We define both pre and post processing logics in `doFilter()`.
- (7) To separate pre and post processing logics in the middle we call `doFilter()` of FilterChain object.

```
public void doFilter(HttpServletRequest, HttpServletResponse, Chain)
```

}

$\equiv \} \text{ pre}$

`chain.doFilter(req, res);`

$\equiv \} \text{ post}$

}

(8) `doFilter()` is given in two interfaces —

(i) `Filter(i) → doFilter(req, res, chain)`

(ii) `FilterChain(i) → doFilter(req, res)`

(9) `doFilter()` of `FilterChain` is used to call a next filter. If next filter doesn't exist then it calls servlet.

Filter - configuration

- 1) A filter - configuration looks like servlet configuration only.
- 2) we can configure a filter in web.xml with two parent tags -
 - (1) <filter>
 - (2) <filter-mapping>
- 3) A filter <url-pattern> will be same as a servlet <url-pattern> because a filter should be executed in the middle when a request is going to a servlet.

4) for example,

```
<filter>
  <filter-name> authentication </filter-name>
  <filter-class> AuthenticationFilter </filter-class>
</filter>

<filter-mapping>
  <filter-name> authentication </filter-name>
  <url-pattern> /servlet </url-pattern>
</filter-mapping>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Baikampet Road,
Ameerpet, Hyderabad.

- 5) we can attach same filter for multiple servlets - we need to configure multiple servlet's <url-pattern> to the filter.

```

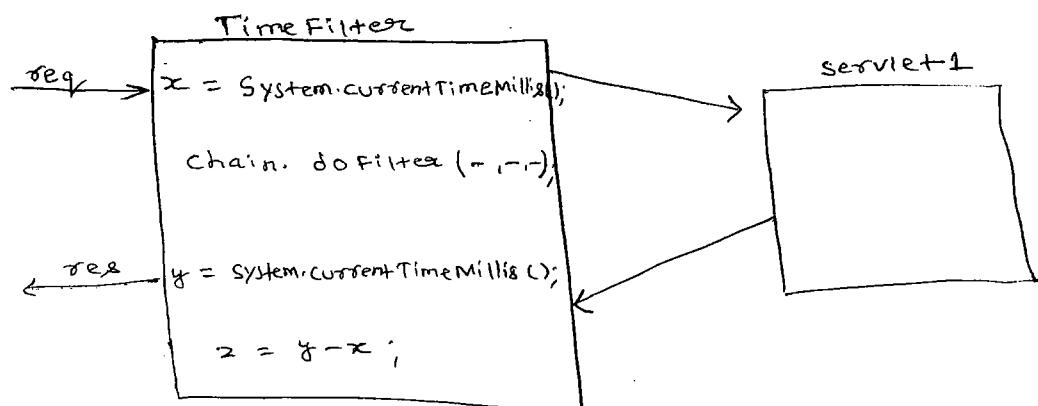
<filter-mapping>
  <filter-name> authentication </filter-name>
  <url-pattern> / s0v1 </url-pattern>
  <url-pattern> / s0v2 </url-pattern>
</filter-mapping>

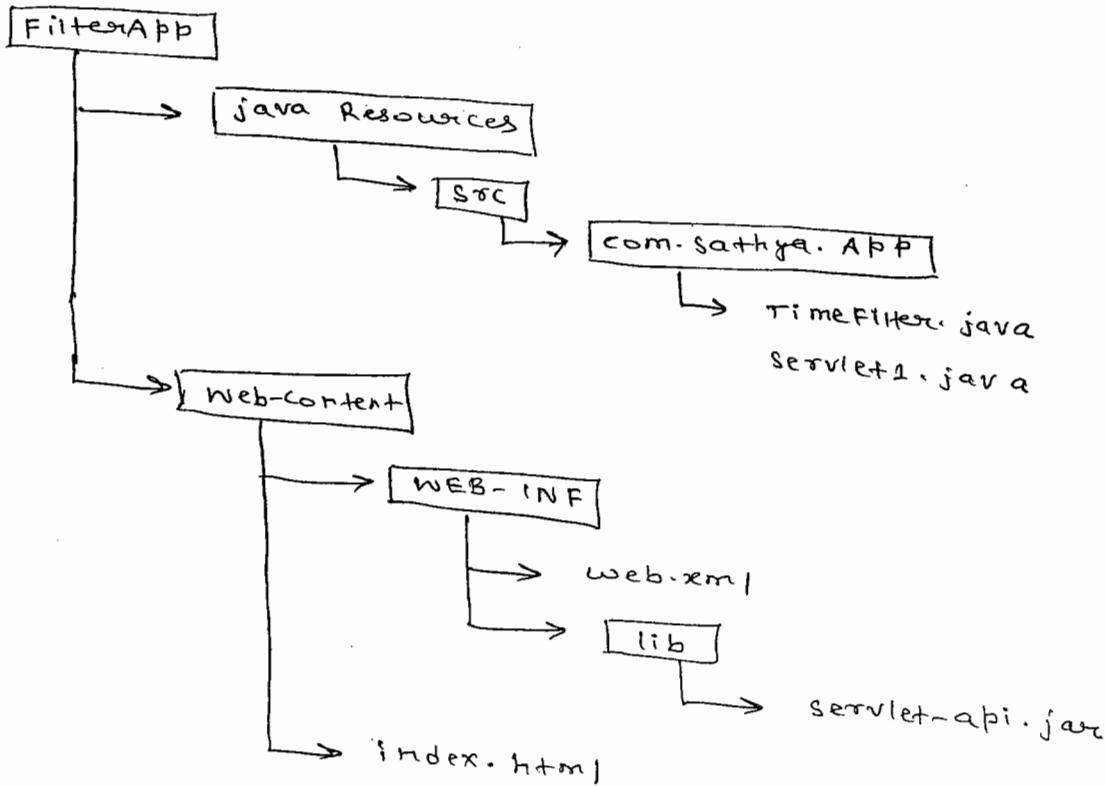
```

- 6) A servlet container creates a filter object at deployment time only.
- 7) In filter configuration, there is no need of `<load-on-startup>` tag. suppose, if we write `<load-on-startup>` tag then container ignores that tag.

Oct 15

- 1) In this example, we are creating a TimeFilter which calculates a servlet Execution time in milliseconds





// index.html

```
<center>
<a href = "servlet1" > click </a>
</center>
```

// servlet1.java

```
public class servlet1 extends HttpServlet
{
    protected void doGet(_____, ___)
    {
        try
        {
            Thread.sleep(15000);
        }
        catch (Exception e)
        {
        }
    }
}
```

SRINIVASENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

11 TimeFilter.java

```
public class TimeFilter implements Filter
{
    public void destroy()
    {
        // ...
    }

    public void doFilter(_____, ____, _____) throws SE, IOE
    {
        long x = System.currentTimeMillis();
        chain.doFilter(request, response);
        long y = System.currentTimeMillis();
        long z = y - x;

        PrintWriter out = response.getWriter();
        out.println("Milliseconds = " + z);
        out.close();
    }

    public void init(FilterConfig config) throws SE
    {
        // ...
    }
}
```

Servlet Listeners

- 1) Listener are introduced in servlet technology at Version 2.4.
- 2) with Servlet Listeners we can develop efficient Web Applications.

- 3) For some specific points in the execution of a web application, servlet container fires Events.
- 4) Based on Events, if we want to run some code in web application then we need to take the support of Servlet Listener.
- 5) A servlet container fires events for request, session and context objects.
- 6) Servlet & Listeners are divided into two categories
- Life cycle listeners — executed by the container when an object is created or destroyed
 - Attribute listeners — executed by the Listener when an attribute is added, removed or replaced.

List of Life cycle Listener

When event fired	Event class	Listener Interface	Methods of Listener
request created / destroyed	Servlet Request Event	ServletRequestListener	(i) requestInitialized() (ii) requestDestroyed()
Session created or session destroyed	HttpSession Event	HttpSessionListener	(i) sessionCreated() (ii) sessionDestroyed()
context created or context destroyed	ServletContextEvent	ServletContextListener	(i) contextInitialized() (ii) contextDestroyed()

2 Oct 15

List of attribute listeners

When event fired	Event class	Listener Interface	Methods of Listener
① attribute added (or) removed (or) replaced in request	ServletRequestAttribute Event	ServletRequestAttribute Listener	(i) attributeAdded() (ii) attributeRemoved() (iii) attributeReplaced()
② attribute added (or) removed (or) replaced in session	HttpSessionBinding Event	HttpSessionAttribute Listener	(i) " (ii) " (iii) "
③ attribute added (or) removed (or) replaced in context	ServletContextAttribute Event	ServletContextAttribute Listener	(i) " (ii) " (iii) "

configuring a Listener

- 1) A Listener class is created by implementing a Listener interface.
- 2) A servlet container automatically creates an object of Listener class and invokes a respective method when an event is occurred.
- 3) We need to configure a listener in web.xml by using <listener>.

e.g.

```
<listener>
```

```
  <listener-class> MyListener </listener-class>
```

```
</listener>
```

- 4) A listener object is created at deployment time. So, there is no <load-on-startup> to the listener. If we write this then we can get exception.
- 5) Each listener class is configured separately with <listener> tag.

```
<listener>
  <listener-class> XXZ </listener-class>
  <listener-class> XY </listener-class>
</listener>
```

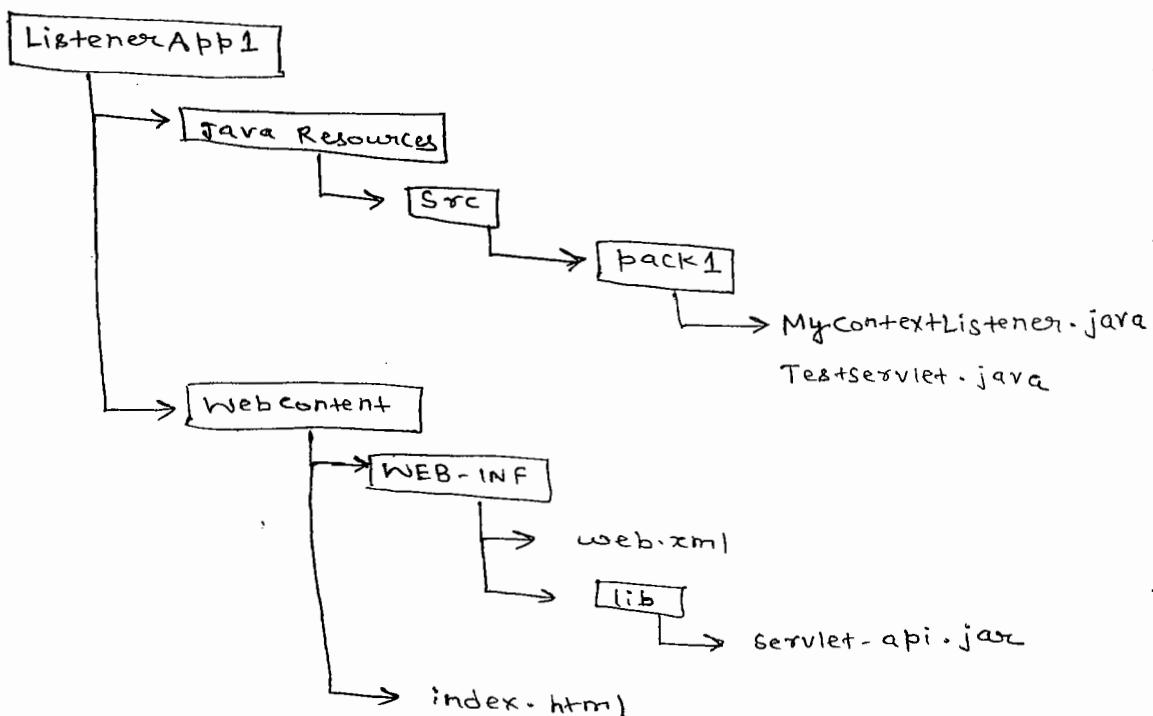
H
N
V
A
L
I
B

```
<listeners>
  <listener-class> XX </listener-class>
  <listener>
    <listener-class> XY </listener-class>
  </listener>
</listeners>
```

N
A
L
I
B

Example1

In this web application, we are finding deployment time, undeployment time and execution time of a web-application through `ServletContextListener`.



index.htm

```
<a href = "servlet"> click here </a>
```

// TestServlet.java

```
public class TestServlet extends HttpServlet
{
    protected void doGet( — , — )
    {
        PrintWriter out = response.getWriter();
        out.println("Hello");
        out.close();
    }
}
```

```

// MyContextListener.java

public class MyContextListener implements servletcontext  

servletconfig  

    Listener
{
    Date d1;
    Date d2;
    long x,y;
}

public void contextInitialized(ServletContextEvent arg0)
{
    d1 = new Date();
    x = System.currentTimeMillis();
}

public void contextDestroyed(ServletContextEvent arg0)
{
    d2 = new Date();
    y = System.currentTimeMillis();
    long z = (y-x)/1000;
    System.out.println("Deployment time : "+d1);
    System.out.println("Undeployment time : "+d2);
    System.out.println("Executed for : "+z+" sec");
}

```

- Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.
- 1) Right click on ProjectName → export → WAR file
→ browse → Select webapps folder of Tomcat
→ finish
 - 2) Start the Tomcat server from outside of the IDE. Open the browser and type the following request —

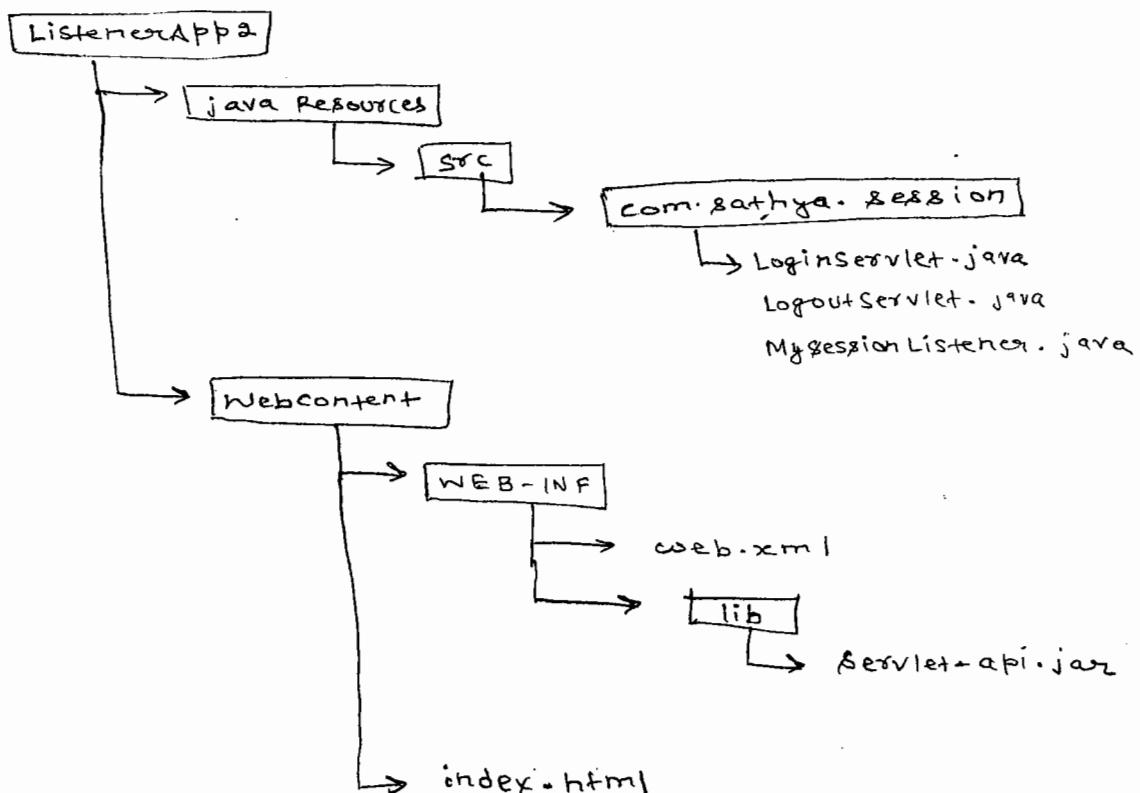
http://localhost:2018/listenerApp1/

- 3) click the hyperlink , so that we get the response from servlet.
- 4) open Tomcat Manager , then Stop ListenerAPP1 application
- 5) on Server console we can find this following output -

Deployment time : Fri Oct 02 18:36:27 IST 2015
 Undeployment time : Fri Oct 02 18:37:54 IST 2015
 Executed For : 87 seconds

30ct 15

Example2 The following example is for counting the number of sessions running in server for a Web Application with the help of HttpSessionListener



// index.html

```
<a href = "login.srv" > login </a>
```

// Loginservlet

```
public class Loginservlet extends HttpServlet  
{  
    protected void doget( — , — ) throws SE, IOE  
    {  
        response. setContentType( "text/html" );  
        // get session  
        HttpSession session = request. getSession();  
        PrintWriter out = response. getWriter();  
        out. println( " Your session activated . . ." );  
        out. println( "         out. close();  
    }  
}
```

// LogoutServlet

```
public class LogoutServlet extends HttpServlet  
{  
    protected void doget( — , — )  
    {  
        // getSession  
        HttpSession session = request. getSession();  
        // invalidate  
        session. invalidate();  
        response. sendRedirect( " index.html" );  
    }  
}
```

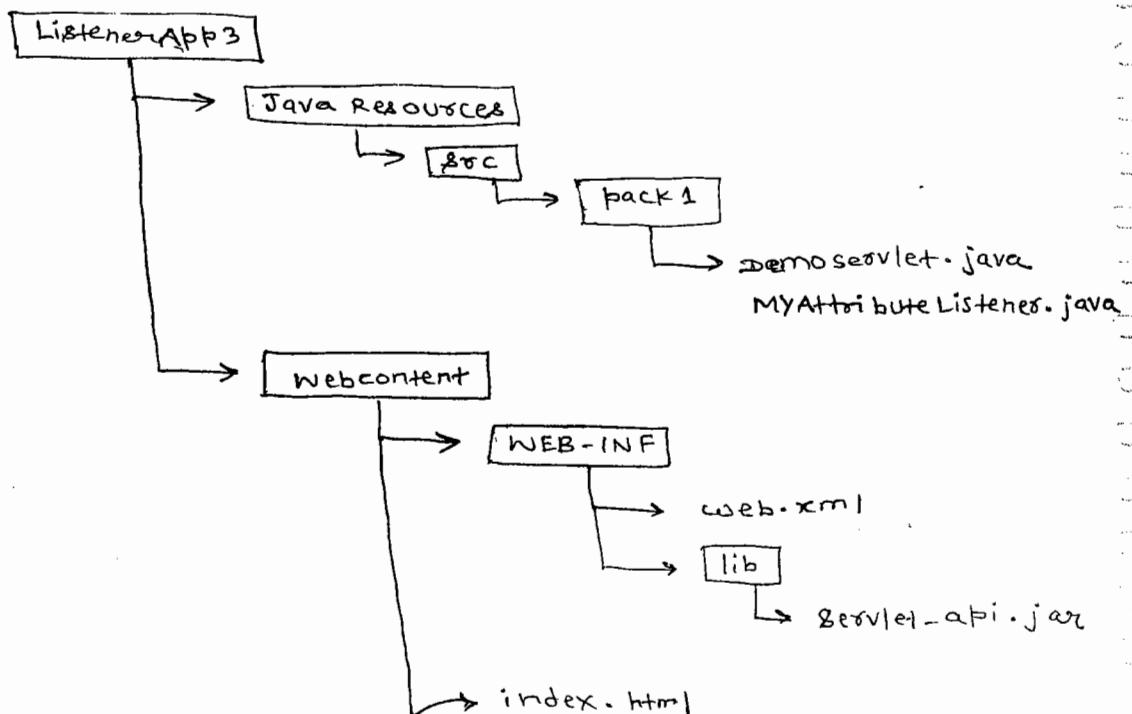
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

// MySessionListener

```
public class MySessionListener implements HttpSessionListener  
{  
    private int count = 0;  
    public void sessionCreated(HttpSessionEvent arg0)  
    {  
        count++;  
        System.out.println("At present no. of session " + count);  
    }  
    public void sessionDestroyed(HttpSessionEvent arg0)  
    {  
        count--;  
        System.out.println("At present no. of session " + count);  
    }  
}
```

Example 3

In this example, we count how many attributes are added and removed from the request object by using `ServletRequestAttributeListener`



// index.html

< a href = "demo&sv" > click here

// MyAttributeListener.java

```
public class MyAttributeListener implements ServletRequestAttributeListener
{
    public static int count1, count2;

    public void attributeAdded (ServletRequestAttributeEvent arg0)
    {
        count1++;
    }

    public void attributeRemoved (ServletRequestAttributeEvent arg0)
    {
        count2++;
    }

    public void attributeReplaced (ServletRequestAttributeEvent arg0)
    {
    }

    public static int getCount1 ()
    {
        return count1;
    }

    public static int getCount2 ()
    {
        return count2;
    }
}
```

Er. S. Venkateswara XEROX
Engineering Material Available
Chp. 5th, Kryzstar Bakery,
Ameerpet, Hyderabad.

```
// demoservlet.java

public class demoservlet extends HttpServlet
{
    protected void doGet( HttpServletRequest request,
                          HttpServletResponse response )
    {
        // add attributes to request

        request.setAttribute("k1", 1000);
        request.setAttribute("k2", 2000);
        request.setAttribute("k3", 3000);

        // remove attribute

        request.removeAttribute("k2");

        // read the counts from listener

        int c1 = MyAttributeListener.getCount1();
        int c2 = MyAttributeListener.getCount2();

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        out.println("Attributes added to request : " + c1);
        out.println("<br>");
        out.println("Attribute removed from request : " + c2);
        out.close();
    }
}
```

http://localhost:2015/ListenerExApp3 ↵

Attributes added to request : 3
Attributes removed from request : 1

Q. 5 Ques 15

Q) Difference b/w HttpSessionAttributeListener and HttpSessionBindingListener ?

Ans - (i) When the attributes are added to the session or removed from session or replaced in a session then container automatically generates HttpSessionBindingEvent.

(ii) To handle HttpSessionBindingEvent, in servlet technology two listeners are provided -

- (i) HttpSessionAttributeListener
- (ii) HttpSessionBindingListener

(iii) If we want to execute some logic when attribute added or removed or replaced in a session then we need to create a class by implementing HttpSessionAttributeListener.

For ex

```
public class MyListener implements HttpSessionAttributeListener
```

```
{
```

```
    public void attributeAdded(HttpSessionBindingEvent e)
```

```
{
```

```
    ==
```

```
}
```

```
    public void attributeRemoved(HttpSessionBindingEvent e)
```

```
{
```

```
    ==
```

```
}
```

```
    public void attributeReplaced(HttpSessionBindingEvent e)
```

```
{
```

```
    ==
```

```
}
```

```
}
```

XEROX
Copier Available
Atayagar Bakery,
Opposite Ayyagari Road,
Balkampet Road,
Ameerpet, Hyderabad.

(iv) If we want to execute some logic of the java class when an object of that java class is added as an attribute to the session or when an object is removed from the session then we need to implement the java class from HttpSessionBindingListener.

(v) suppose, if we want to execute some logic of Employee class, when its object is added as an attribute to the session or when it is removed from the session then we need to create that Employee class like the following —

```
public class Employee implements HttpSessionBindingListener  
{  
    public void valueBound (HttpSessionBindingEvent e)  
    {  
        //  
    }  
    public void valueUnbound (HttpSessionBindingEvent e)  
    {  
        //  
    }  
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

HttpSessionActivationListener

- 1) In real time, a session object will be shifted from one server to another server in a distributed cluster.
- 2) When a session object is shifted then a java class object added to the session is also shifted from one server to another.
- 3) At the time of shifting a java class object if we want to execute any logic before an object is going to be shifted and after an object is objected then we need to implement that java class from HttpSessionActivationListener.
- 4) In HttpSessionActivationListener, there are two abstract methods —
 - (i) SessionWillPassivate (HttpSessionEvent e)
 - (ii) SessionDidActivate (HttpSessionEvent e)

(i) SessionWillPassivate (HttpSessionEvent e)
will be called before shifting an object

(ii) SessionDidActivate (HttpSessionEvent e)
is called after an object is shifted

Example

```

public class Employee implements HttpSessionActivationListener, Serializable
{
    public void sessionWillPassivate (HttpSessionEvent e)
    {
        =
    }

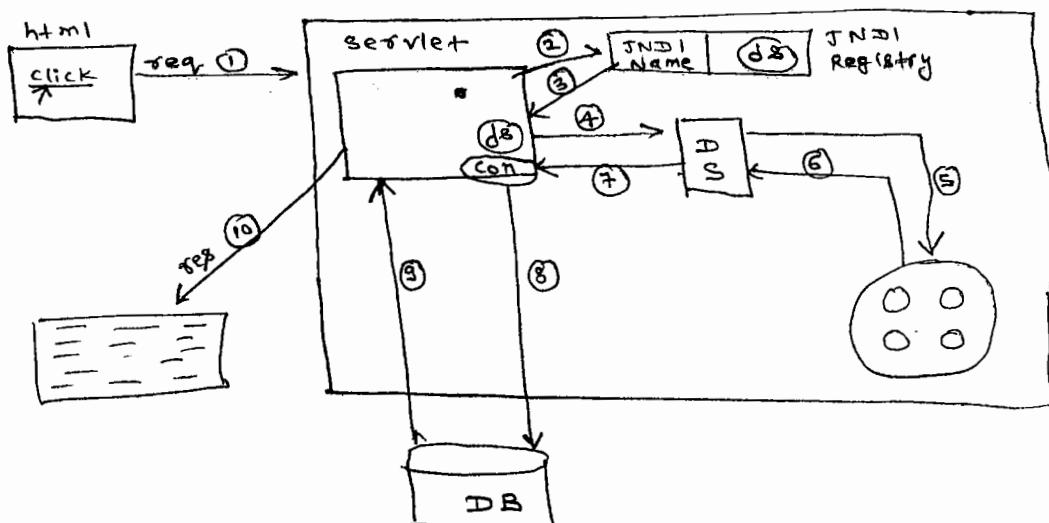
    public void sessionDidActivate (HttpSessionEvent e)
    {
        =
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Servlet with connection pooling

- 1) In the following example, a servlet reads connection from connection pool configured in weblogic server and reads data from database and displays the data on browser.



Configuring a connection pool in weblogic server

- 1) Start the weblogic server
- 2) Open browser and type the following request -
`http://localhost:7001/console`
- 3) Expand services at left side → Select Datasources
→ New button → Generic datasource →
Enter the following details -

Name

JNDI Name

Data Base type

Next → next → next

- 4) Enter the following connection properties —

Database Name :

Host Name :

port :

Database Username :

Password :

confirm Password :

→ next

5) click on Test configuration button → next →
→ select Admin server → finish

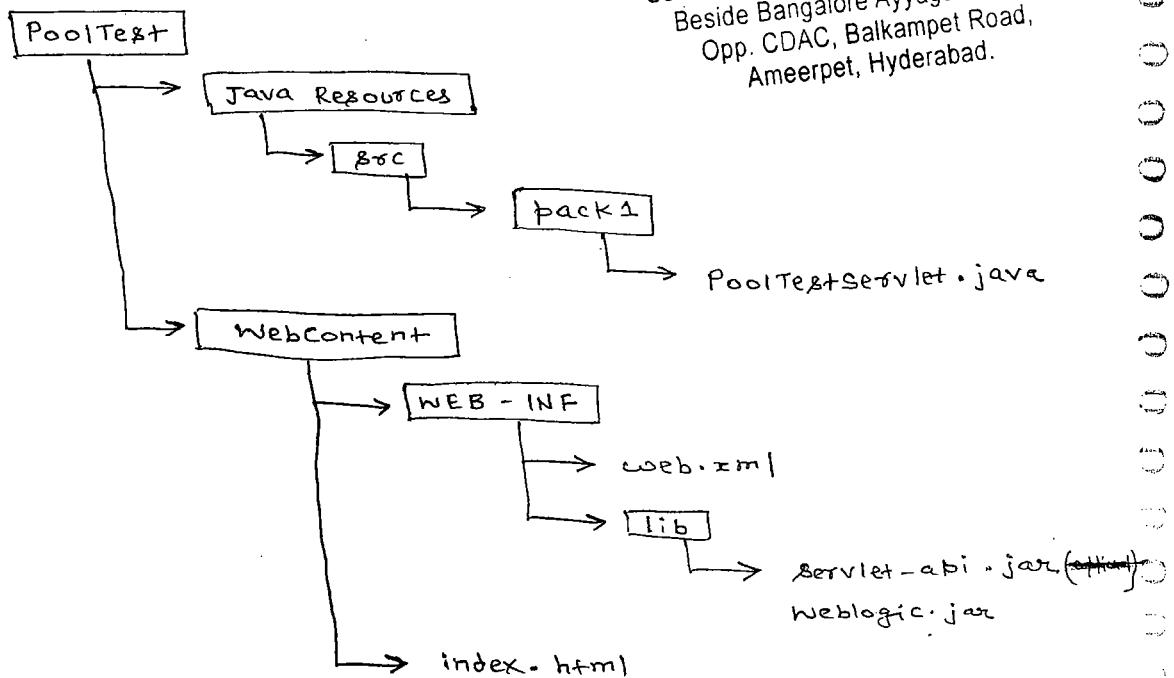
Note -

If we want to see the connection pool settings then click on test data source → connection tag → find the details.

6/10/15

Example

Directory Structure



// index.html

```
<a href = " pooltry" > click me </a>
```

// PoolTestServlet.java

```
public class PoolTestServlet extends HttpServlet
{
    DataSource db;
    public void init(ServletConfig config) throws SE EE
    {
        try
        {
            Properties prop = new Properties();
            // get initial properties
            prop.put("java.naming.factory.initial", weblogic.jndi.WLInitialContextFactory);
            prop.put("java.naming.provider.url", "t3://localhost:7001");
            InitialContext ctx = new InitialContext(prop);
            Object o = ctx.lookup("test");
            db = (DataSource) o;
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
    // init()
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    {
        try
        {
            Connection con = db.getConnection();
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from emp");
            response.setContentType("text/html");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

SRI RAMA RAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

PrintWriter out = response.getWriter();
while(rs.next())
{
    out.println(rs.getInt(1) + " " + rs.getString(2)
    + out.println("<br>"); + " " + rs.getInt(3));
    out.close();
    rs.close();
    stmt.close();
    con.close();
}
catch(Exception e)
{
    System.out.println("Exception : " + e);
}
}

```

Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

- 1) In the above servlet we have written logic for obtaining datasource object from registry under init() and the remaining database logic under doget()
- 2) To get a connection from connection pool for multiple times, we need a mediator object called a datasource from registry.
- 3) If once datasource object is retrieved from registry then we can use that datasource object to get connection for multiple times. so, we have defined the registry logic in init() method.

4) To open a connection with registry, we need initial context object. This class reads jndi properties from properties object and opens a connection with a registry.

Execution steps

- 1) Right click on the project name → export → war file → browse → select R:/ drive → finish.
- 2) Start the weblogic server → open browser and open Admin console of web logic by typing the following request —
`http://localhost:7001/console/`
- 3) click on deployment on left side → install button → upload your file → browse button and select the war file → next → next → next → finish
- 4) open the browser and type the following url —
`http://localhost:7001/PoolTest`

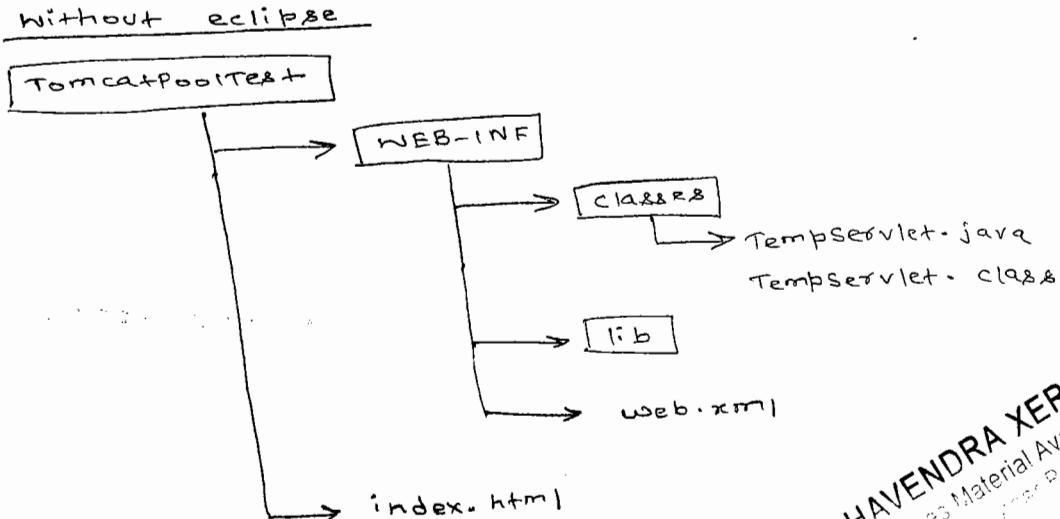
connection pooling with Tomcat

- 1) To create a connection pool in Tomcat, we don't have any GUI support.
- 2) To configure a connection pool with oracle database, we need to write the following <Resource> tag in context.xml under Tomcat7 /conf directory -

```
<Resource name = " jdbc/myoracle"
auth = " container"
type = " javax.sql.DataSource"
driverClassName = " oracle.jdbc.OracleDriver"
url = " jdbc:oracle:thin:@(127.0.0.1):1521:xe"
username = " system" password = " tiger"
maxActive = " 20" maxIdle = " 10"
maxWait = "-1" />
```

- 3) copy ojdbc6.jar to Tomcat7 / lib folder.

without eclipse



SRI RAGHAVENDRA XEROX
Material Available
for Reference Only
www.sriraghavendraacademy.com

// index.html

```
<a href = " servlet " > click me </a>
```

// TempServlet.java

```
import javax.servlet.GenericServlet;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import javax.servlet.ServletConfig;
import java.io.IOException;
import java.io.PrintWriter;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import javax.naming.Context;
import javax.naming.InitialContext;
```

```
public class TempServlet extends GenericServlet
{
    DataSource ds;

    public void init ( ServletConfig config ) throws SE
    {
        try
        {
            Context initContext = new InitialContext ();
            Context envContext = (Context) initContext.
                lookup ("java: / comp / env ");
            outer registry "key"
            ds = (DataSource) envContext.lookup ("jdbc /
                inner registry myoracle ");
            key "key"
        }
        catch ( Exception e )
        {
            System.out.println ( e );
        }
    }
}
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```
public void service ( ServletRequest request , ServletResponse response )
throws SE, IOException
```

```
{  
try
```

```
Connection con = ds.getConnection();
```

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

```
ResultSet rs = stmt.executeQuery ("Select * from emp");
```

```
response.setContentType ("text/html");
```

```
PrintWriter out = response.getWriter();
```

```
while ( rs.next() )
```

```
{  
out.println (rs.getInt(1) + " " + rs.getString(2)  
+ " " + rs.getInt(3));
```

```
out.println ("<br>");
```

```
rs.close();
```

```
stmt.close();
```

```
con.close();
```

```
out.close();
```

```
}
```

```
catch (Exception e)
```

```
{  
System.out.println (e);
```

```
}
```

```
}
```

```
.
```

SRI KRISHNENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

web.xml

```
< web-app >
  < servlet >
    < servlet-name > temp < /servlet-name >
    < servlet-class > TempServlet < /servlet-class >
  < /servlet >

  < servlet-mapping >
    < servlet-name > temp < /servlet-name >
    < url-pattern > /servlet/temp < /url-pattern >
  < /servlet-mapping >

  < resource-ref >
    < res-ref-name > jdbc/myoracle < /res-ref-name >
    < res-type > javax.sql.DataSource < /res-type >
    < res-auth > container < /res-auth >
  < /resource-ref >

< /web-app >
```

SRI KAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Servlet (3.0) annotation

- 1) To reduce configuration in web.xml we got annotation.
- 2) Servlet annotation are introduced in servlet 3.0 and even in javax.servlet.annotation package.
- 3) Every annotation is internally special interface and its implementation will be provided by server vendors.
- 4) In Tomcat, servlet annotation are supported from tomcat 7 onwards.

<u>Tag</u>	<u>Annotation</u>
① <Servlet> & <Servlet-mapping>	= @WebServlet
② <filter> & <filter-mapping>	= @WebFilter
③ <listener>	= @WebListener

- 6) @WebServlet
This annotation is added on top of the servlet class.

For ex:

```
@WebServlet(value = "/servlet", name = "sone")
```

```
public class Myservlet extends GenericServlet
```

```
}
```

CHIRAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

example 2

```

@WebServlet( value = "/srv1", name = "fone"
loadOnStartup = 1, initParams = {
    @WebInitParam( name = "p1", value = "100"),
    @WebInitParam( name = "p2", value = "200")
})

```

```
public class MyServlet extends GenericServlet
```

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- ⊕ If we want to add a single url-pattern to a servlet then we need to choose value element.
- ⑧ If we want to add multiple url-patterns then in place of value we need to use urlPatterns element.

Ex

```
@WebServlet( value = "/srv1")
```

```
@WebServlet( urlPatterns = {"/srv1", "/srv2"} )
```

g) @WebFilter

Ex1

```
@WebFilter( value = "/srv1", name = "fone" )
```

```
public class MyFilter implements Filter
```

{

====

}

```
@ WebFilter( urlPatterns = { "/servlet1" , "/servlet2" } ,  
           name = "fone" )
```

```
public class MyFilter implements Filter
```

{
=

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

10) @WebListener

```
<listener>
```

```
<listener-class> XYZ </listener-class>
```

```
</listener>
```

@WebListener

```
public class MyListener implements ServletContextListener
```

{
=

Note-

- 1) For the tags `<context-param>`, `<session-config>`,
`<welcome-file-list>` there are no
annotations because these tags are used
for application level settings.
- 2) annotations are not used for application
level settings.

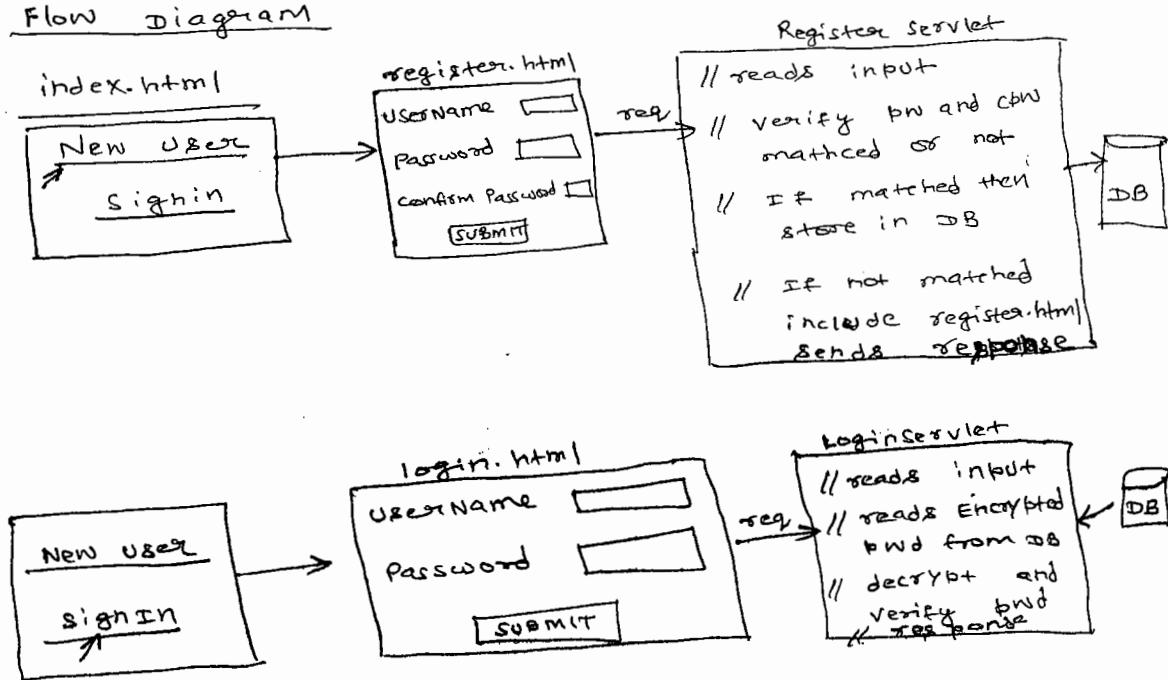
— X — X —

Password Encryption (Miscellaneous Example)

- 1) In real-time applications, passwords are not stored in database in plaintext format because, database users can see the password and they can misuse the password.
- 2) A password is encrypted in application and then encrypted password will be stored in database.
- 3) If there is any need to verify password then application reads encrypted password and then decrypts the password then verify the password.
- 4) For encrypting and decryption a password, either we can manually define the logic or we can take the support of a third party provided code.
- 5) JBCrypt is a third party api for encrypting and decrypting a password. We can download a zip file of JBCrypt from <http://www.mindrot.org/projects/jBCrypt/>

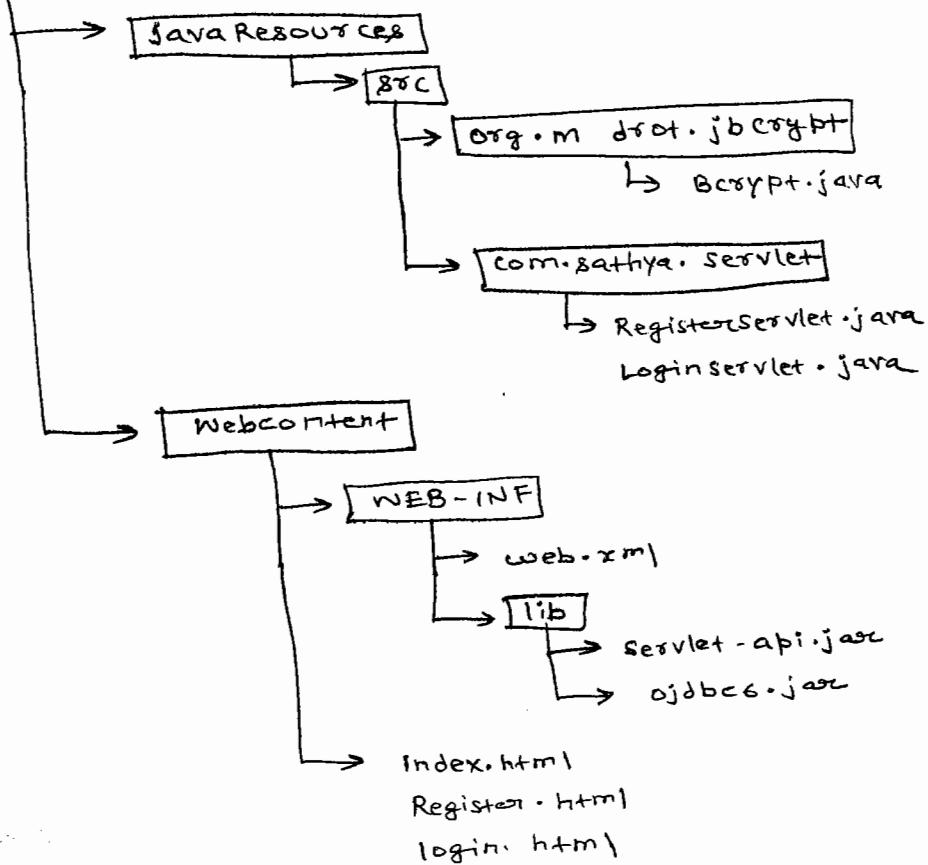
- 6) Download `JBCrypt-0.4.zip` and then extract it. After extraction, in `src` folder we can find `Bcrypted.java` and we need to copy this file to our application.
- 7) In `Bcrypted.java`, we have two required static methods called `hashpw()` and `checkpw()`
- 8) `hashpw()` encrypts a password and `checkpw()` decrypts a password.

Flow diagram



SHRI KAVINDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Password Encrypt



```
<!-- index.html -->
```

```
<center>
```

```
<a href = "register.html"> New user </a> <br>
```

```
<a href = "login.html"> sign in </a>
```

```
</center>
```

```
<!-- register.html -->
<center>
<form action = "register&sv" method = "post">
    userName: <input type = "text" name = "uname"> <br>
    password: <input type = "password" name = "pwd"> <br>
    confirm Password <input type = "password"
        name = "cpwd">
    <input type = submit value = "&copy;">
</form>
</center>
```

```
<!-- login.html -->
<center>
<form action = "login&sv" method = "post">
    userName <input type = text name = "uname"> <br>
    password <input type = password name = "pwd">
    <input type = submit value = "login">
</form>
</center>
```

Software Languages ENDKA XEROX
Beside Bangalore Material Available
Opp CDAC, Balkampet Road,
Ameerpet, Hyderabad.

RegisterServlet.java

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
public class RegisterServlet extends HttpServlet {
    public void doGET(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException {
        String username = req.getParameter("uname");
        String password = req.getParameter("pwd");
        String confirmPassword = req.getParameter("cpwd");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        if (!password.equals(confirmPassword)) {
            out.println("<center><font color='red'>
                confirm password and Password
                are not matched </font></center>");
            RequestDispatcher rd = request.getRequestDispatcher(
                "register.html");
            rd.include(req, res);
        } else {
            // encrypt password
            String encryptPassword = Bcrypt.hashpw(password,
                Bcrypt.gensalt());
            try {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:XE",
                    "system", "tiger");
            }
        }
    }
}
```

```

Prepared Statement pstmt = con.prepareStatement
        (" insert into login values(?,?)");

pstmt.setString(1, username);
pstmt.setString(2, encryptPassword);
int i = pstmt.executeUpdate();
pstmt.close();
con.close();
out.println(" User Registered");
out.println("<a href = index.html> click here </a>");
} catch( Exception e )
{
    SEP(e);
    out.close();
}
}

```

// Loginservlet.java

```

public class Loginservlet extends HttpServlet {
public void doGet( --, -- ) throws SE, IOE {
{
    String username = request.getParameter("uname");
    String password = request.getParameter("pwd");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String encryptedPassword;
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
            (" jdbc:oracle:thin:@localhost:1521",
             "system", "tiger");
    }
}

```

SKI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

Preparedstatement pstmt = con.prepareStatement(
    "select password from login
     where Username = ?");

pstmt.setString(1, username);

ResultSet rs = pstmt.executeQuery();

if (rs.next() == true)
{
    String encryptedPassword = rs.getString(1);
}
else
{
    out.println("password / username invalid");
}

rs.close();
pstmt.close();
con.close();

```

```

boolean flag = Bcrypt.checkpw(password,
                               encryptedPassword);

```

```

if (flag == true)
{
    out.println("login success");
}
else
{
    out.println("login failed");
}

out.close();

```

```

catch (Exception e)
{
    System.out.println(e);
}

```

SQL > Create table login (username varchar(50),
 primary key, password varchar(50));

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Digitized by srujanika@gmail.com

* Java Server Pages (JSP)

- 1) We have 2 types of web applications -
 - (i) static
 - (ii) dynamic
- 2) In early days all the web applications are static, it means they used to produce same content to all the users.
- 3) In the next, today there is a requirement of dynamic web application.
- 4) In order to develop dynamic web applications, server side technologies are introduced.
- 5) A first server side technology to develop the dynamic web application was CGI technology.
- 6) As CGI technology faced some problems, in order to overcome Sun-microsystems introduced servlet technology.

Why JSP ?

- 1) Actually servlet is a good technology to develop dynamic web application but a huge code is required to produce dynamic output.
- 2) In order to reduce the coding burden Microsoft introduced ASP technology. In this technology logic is written within the tags of the page.

- 3) ASP has reduced coding burden and provided an easy way for developing the dynamic web application. So, the developers in industry migrated from Servlet technology to ASP.
- 4) In order to attract the developers back to Sun-microsystem technologies, SunMicrosystem released a similar technology called JSP.
- 5) Servlet and JSP are two complementary technologies.
- 6) JSP is an enhancement to servlet technology but not replacement.
- 7) In today's web application we create a very few servlet and a more no. of JSP to develop the web application.

Differences b/w Servlet and JSP

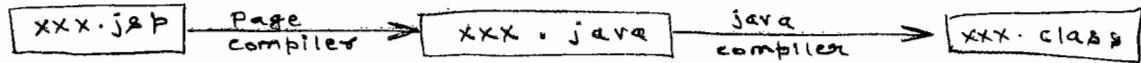
Servlet	JSP
1) Servlet is a program, it needs a lot of java code.	1) JSP is a page, it needs a less java code.
2) In a servlet we write the content in logic.	2) In JSP we write the logic in content.

Out.println(" Hi");
 ↓
 logic ↓
 content

- | | |
|-------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 3) A servlet is a class so, if any modifications are done then we need to recompile, reload and then we need to restart the server. | 3) JSP is a page, so we don't need recompilation, reloading or restart of the server. |
| 4) servlet technology has not given separation of business and presentation logic. | 4) JSP technology has provided a way for separating business and presentation logic |
| 5) In servlet, we have only 3 implicit variables (object), the remaining we need to get explicitly. | 5) In JSP, we have 9 implicit objects. |
| 6) In servlet technology we have only 3 scopes for sharing the data in the project. | 6) In JSP, there are 4 scopes for sharing the data. |

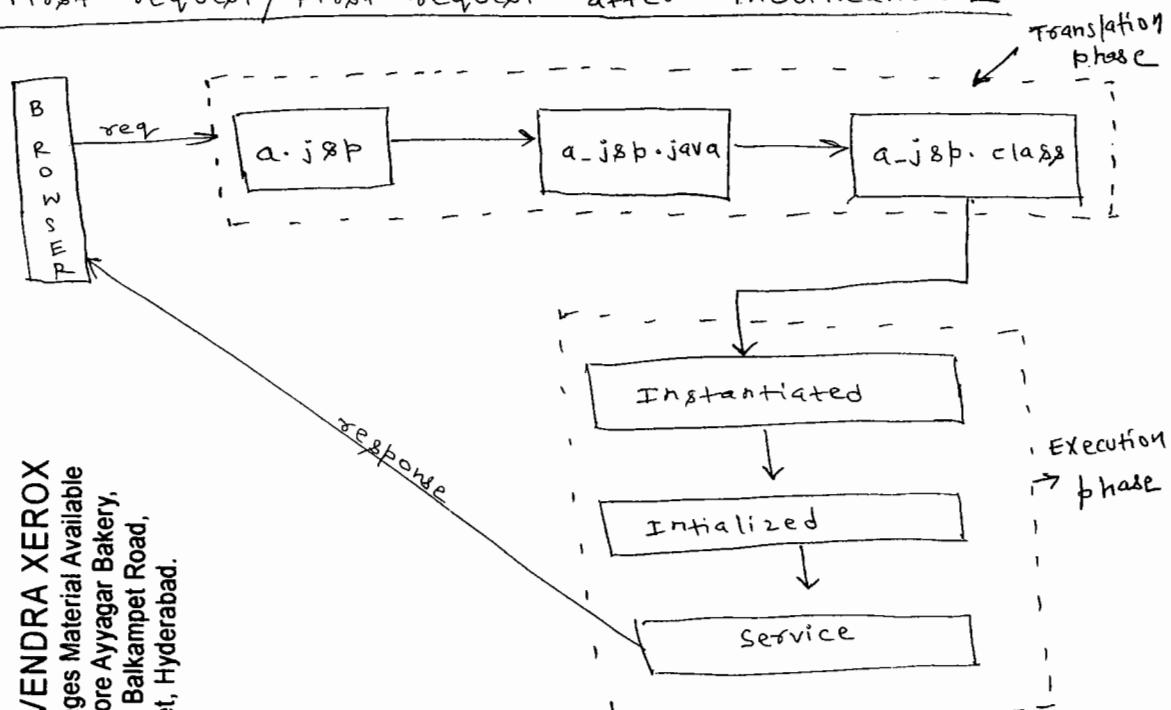
AVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 1) JSP Architecture contains two phases —
 - (i) Translation phase
 - (ii) Execution phase
- 2) Every JSP page is translated into equivalent servlet at server.
- 3) A JSP container has a page compiler. It will translate a JSP page into an equivalent servlet.
- 4) A page compiler in Tomcat is Jasper and a page compiler in weblogic is jstl.
- 5) For ex, if JSP file is a.jsp then Jasper component translates a.jsp as a-jsp.java and jstl component translates a.jsp as --a.java.
- 6) Translation phase of a JSP page is nothing but, translating a .jsp file as .java and then compiling it as a .class.
- 7) In a Translation phase two compilers are used —
 - (i) Page compiler
 - (ii) Java compiler

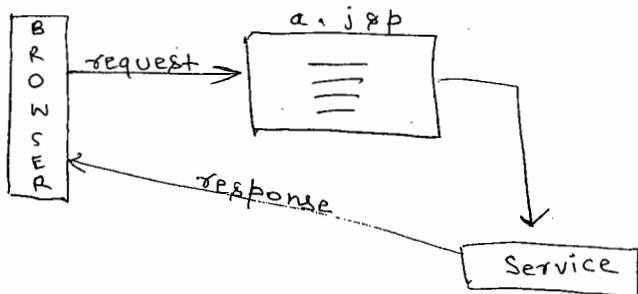


- 8) In execution phase, a servlet is instantiated then initialized and then its service is executed.
- 9) When a first request comes to a jsp page or when a first request comes after modification of the page then a jsp container (web container) first starts translation phase then starts execution phase.
- 10) For the remaining request to a jsp page container will only call the service.

First request / First request after modification -



Remaining request



Q) What happens if a server is restarted, after executing a jsp for several times?

Ans - 1) When a server is shutdown then automatically a servlet object is destroyed but .java and .class files of a jsp are not deleted from the server.

2) After restarting the server when we send the request to the same jsp page then container creates an object for existing servlet class then initialized it and then call its service. It means container again start execution phase but not translation phase.

request	Translation			Execution		
	• JSP	• Java	• class	Instantiation	Initialization	Service
request1 →		✓	✓	✓	✓	✓
request2 →		✗	✗	✗	✗	✓
--- JSP Page Modified ---						
request3 →		✓	✓	✓	✓	✓
--- server restarted ---						
request4 → (request1 after restarting)		✗	✗	✓	✓	✓

*) How a JSP container recognizes whether a JSP page is modified or not?

Ans - A JSP container internally uses some file comparison tool. This tool will verify previous timestamp of a page and a current timestamp of a page both are same or not. If same then container directly executes service if not same then container starts translation phase followed by execution phase.

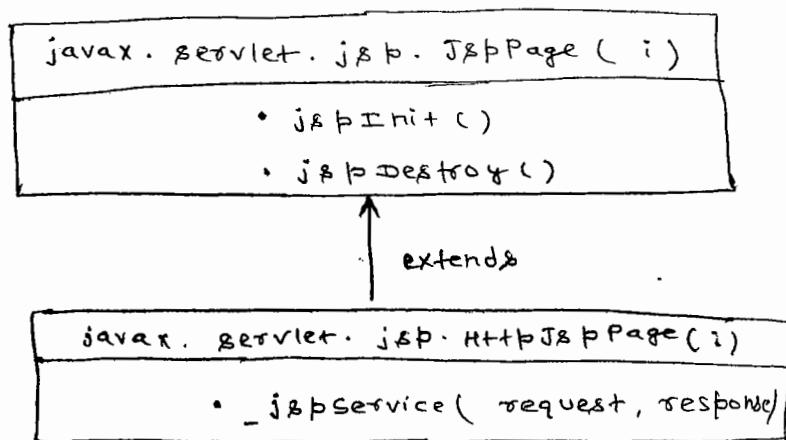
10 Oct 15

JSP Life cycle methods

- 1) Like a servlet life cycle, jsp life cycle also contains 3 methods —
 - (i) `jspInit()`
 - (ii) `_jspService(request, response)`
 - (iii) `jspDestroy()`
- 2) The above 3 life cycle methods of jsp are given in `HttpJspPage` interface of `javax.servlet.jsp` package.
- 3) Actually, `HttpJspPage` interface extends `JspPage` interface and 2 life cycle methods `jspInit()` and `jspDestroy()` are inherited from `JspPage` interface.

4)

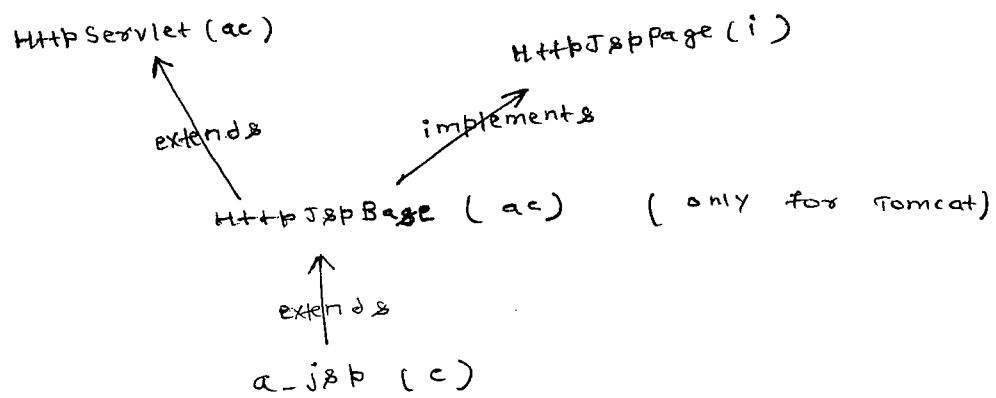
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



- 5) When a jsp page is translated into equivalent servlet page compiler will extend that servlet class from a superclass provided by the container.

- 6) For ex, in case of Tomcat server, jasper is a page compiler, it extends a servlet class from HttpServletPage (abstract class) provided by Tomcat.
- 7) Every web container provides a super class for jsp equivalent servlet and that super class extends HttpServlet and implements HttpServletPage interface.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



- 8) A container provided super class overrides two life cycle methods of jsp called jspInit() and jspDestroy().
- 9) The remaining life cycle method - jspService() will be overridden by page compiler at translation time.
- 10) A container provided super class calls jsp life cycle methods from servlet life cycle methods. So, a container calls servlet life cycle methods and they call jsp life cycle methods.

Q) Why `_jspService()` started with underscore?

Ans - It is an indication to the programmer that a programmer should not override `_jspService()` in a JSP page. But, a programmer can override `jspInit()` and `jspDestroy()`.

JSP tags

- 1) A JSP page is mix of HTML tags and JSP tags.
- 2) HTML tags are used for producing static content and JSP tags are used for producing dynamic contents.
- 3) JSP tags are categorized into 4 types -
 - (i) Scripting tags.
 - (ii) Directive tags
 - (iii) Action tags
 - (iv) Custom tags

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Scripting tags

- 1) Script means, it is a code written in a page.
- 2) Scripting tags are used to insert java code at different places of JSP page.

3) Scripting tags are of 3 types -

- (i) Scriptlet tag
- (ii) Expression tag
- (iii) Declaration tag

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(i) Scriptlet tag

- 1) This tag is used to insert some java code into a jsp page which should be executed for each request.
- 2) we can write a scriptlet tag in the form of html tag or in the form of xml tag.

For example

a.jsp

```
<%  
    count++;  
%>
```

a.jsp

```
<jsp:scriptlet>  
    count++;  
</jsp:scriptlet>
```

- 3) At translation time, a java code written under scriptlet tags will be inserted into - jpservice (req, res) of servlet. So, a scriptlet tag code is executed for each request.

a.jsp

<%

count++;

%>

a-jsp.java

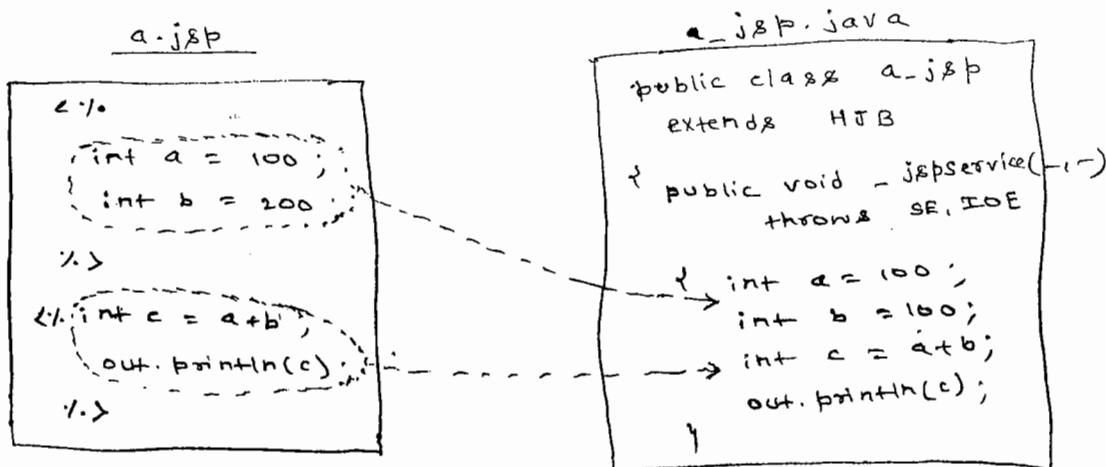
```
public class a-jsp extends  
HttpJspBase  
{  
    public void _jpservice(~)  
        throws SE, IOE  
    {  
        count++;  
    }  
}
```

4) we can write any number of scriptlet tags in a jsp page. At a translation time the code of all scriptlet tag will be inserted into - jspService().

5) - jspService() is executed for each request . So , scriptlet tags are executed for each request .

12 Oct 15
(sandeep's b'day)

6) we can declare variables in a scriptlet tag . Those variable will become as local variables to the - jspService() at translation time .



Q) can we define methods in a scriptlet tag?

Ans- NO, if we define a method in a scriptlet tag then at translation time that method is inserted into `- jspService(-,-)` method . But in java we can't define a method in another method . So , we can't define the methods under scriptlet tag .

a. j&p

≤ γ

```
public void m1() {
```

二

۲۷

a -j & p . java

public class a.jsp extends
HJB

1

```
public void -jstService( -,- )  
throws SE, IOE
```

public void mix)
t = invalid

1

Q) can we make variables
scriptlet tag as private

created in
or public?

Ans - No, local variables can't
the only one modifier
local variable is final.

be private or public
allowed for the

a. i&F

17

```
private int x = 100;  
public int y = 200;
```

-1-

— i g p . j a r a

public class a-j & p
extends HJB

```
    public void _jipService()  
        throws SE, IOE
```

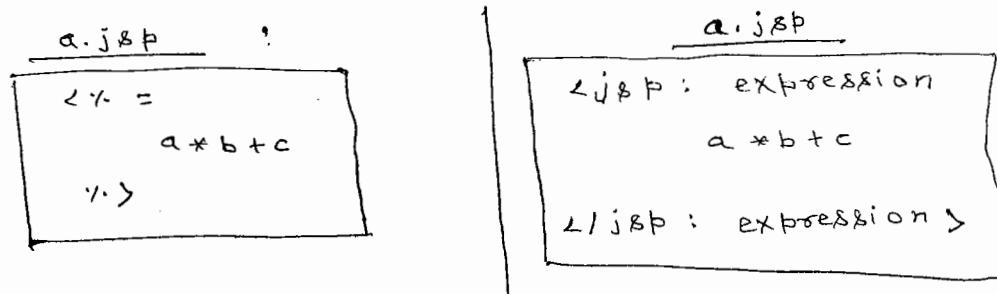
```
{ private int x = 100;  
public int y = 200;
```

۹

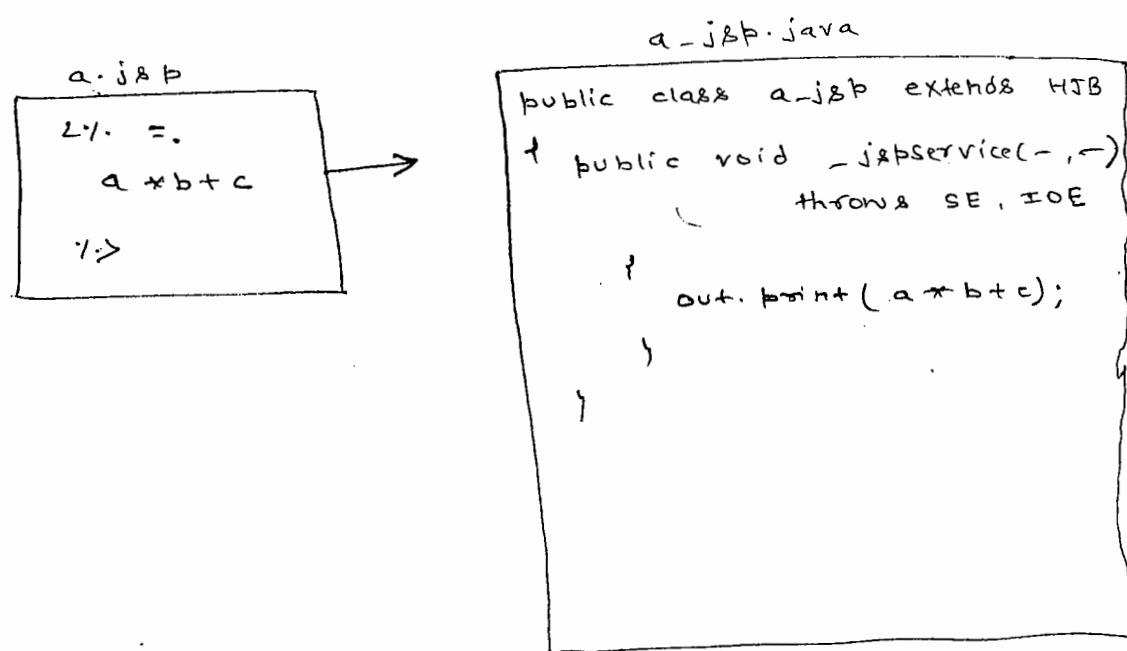
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Basaveshwara Bangalore Ayyappa Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Expression tag

- 1) Expression tag is used to write expression in JSP.
- 2) Expression tag evaluates an expression and displays its output on the browser.
- 3) we can write expression tag in two ways with html syntax or xml syntax.



- 4) At translation time, page compiler converts expression tag into out.print statement and inserts this statement into `_jpservice()` so, the result of the application will be printed for each request on the browser.



5) EX1

$\langle \gamma. = d = a * b + c \gamma. \rangle$

↓

out.print($d = a * b + c$); // INVALID

EX2

$\langle \gamma. = a * b , a + b \gamma. \rangle$

↓

out.print($a * b , a + b$); // INVALID

EX3

$\langle \gamma. = a + b ; \gamma. \rangle$

↓

out.print($a * b ;$); // INVALID

EX4

$\langle \gamma. = a + b + " " + c * d \gamma. \rangle$

↓

out.print($a + b + " " + c * d$); // VALID

EX5

$\langle \gamma. = \text{out.print}(a * b) \gamma. \rangle$

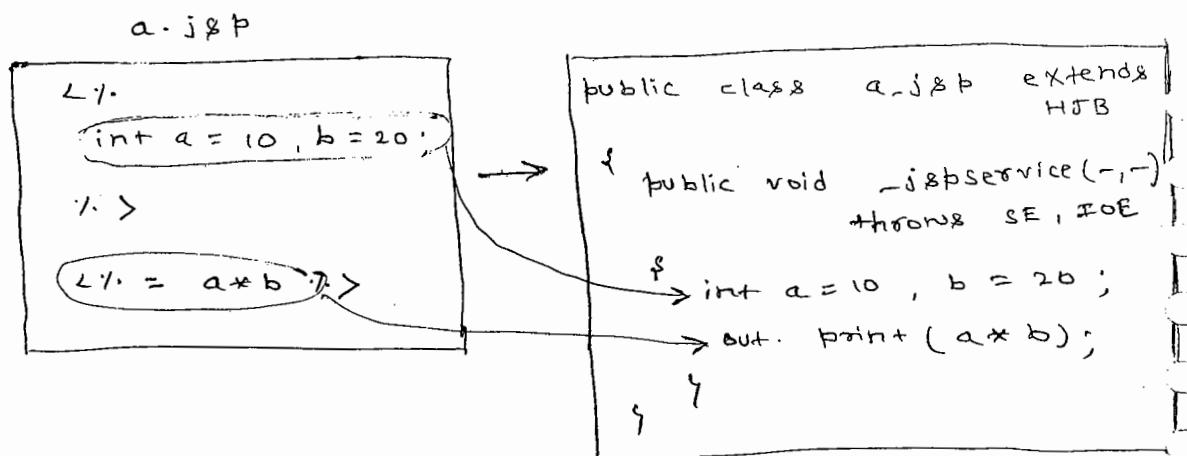
↓

out.print(out.print($a * b$)); // INVALID

SRI RAHULAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

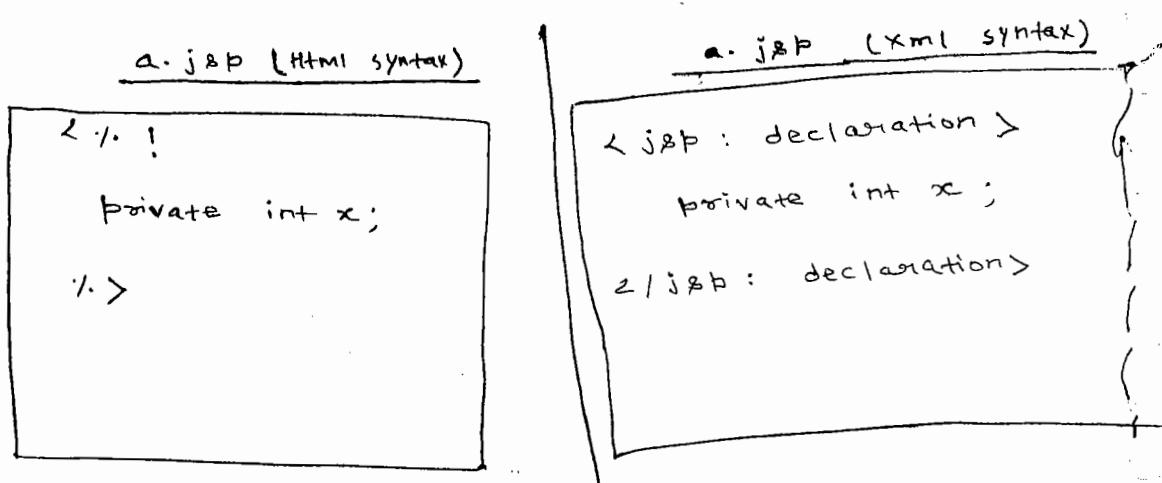
Q) can we use variables of scriptlet tag in expression tag or not?

Ans - Yes, because both scriptlet tag and expression tag are inserted into `_jspService()` at translation time.

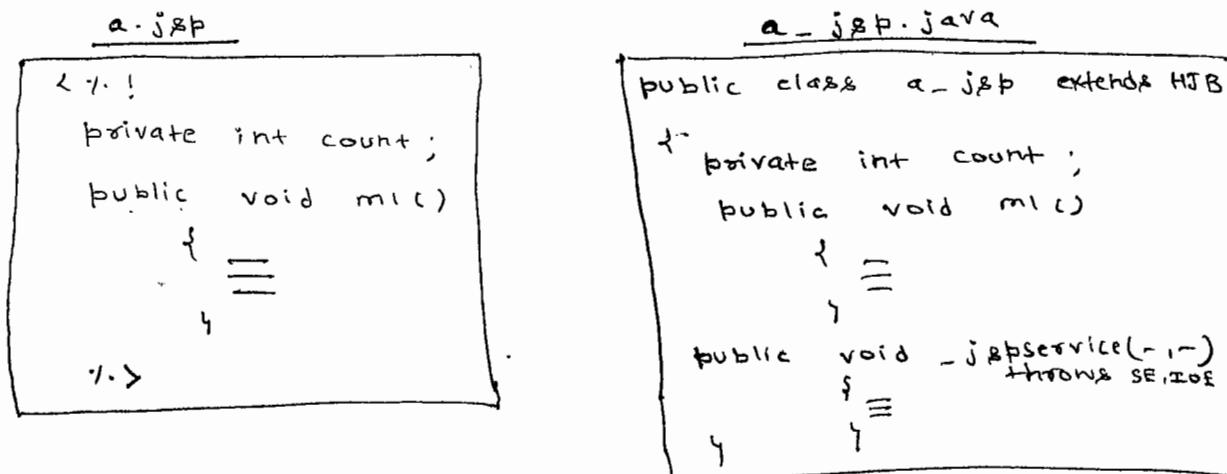


declaration tag

- 1) This declaration tag is used to create instance variable and also to define methods in a jsp page.
- 2) We can write a declaration tag in html syntax or in xml syntax.



- 3) The code of declaration tag will be inserted into class at translation time.



13 Oct 15

- 4) Implicit objects of JSP are not allowed in a `<declaration>` tag because declaration tag goes code goes to out of `_jpservice()`.

- 5) Implicit objects of JSP are allowed only in `<scriptlet>` tag and `<expression>` tag because the code of these two tags goes to `_jpservice()`.

For ex:-

```
<%!
public void sayHello()
{
    Not allowed
    {
        out.println("Hello");
    }
}
%>
```

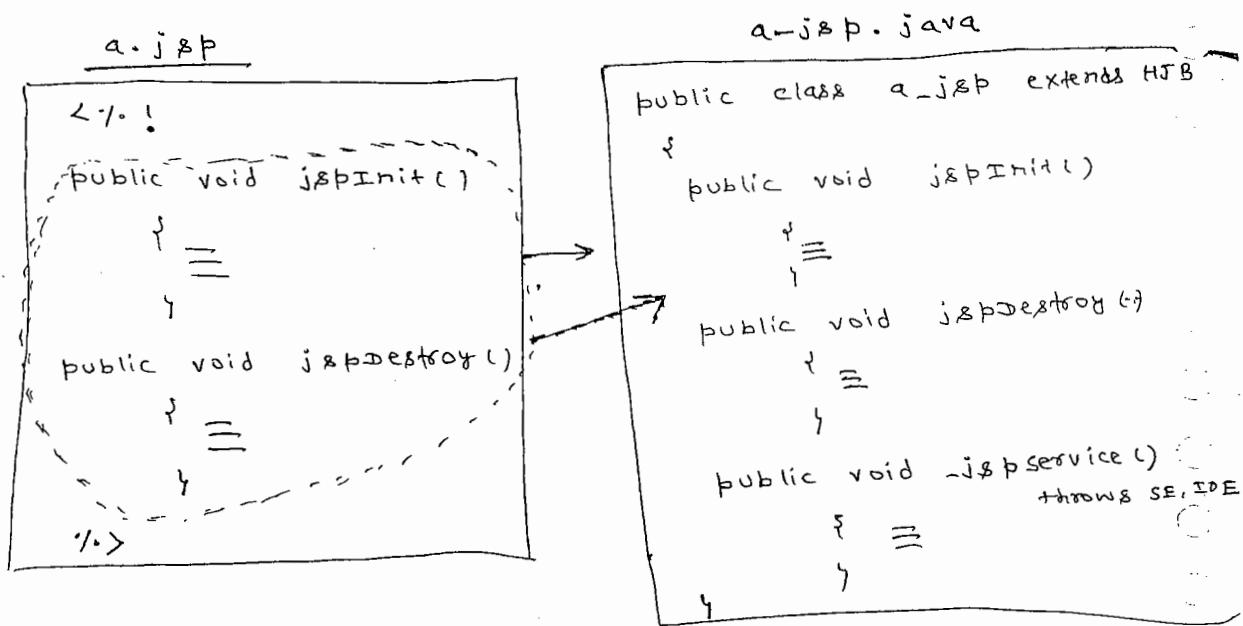
A bracket on the right side of the code is labeled "Declaration tag".


```
2) <%
    out.println("Hello");
    %> }
```

An arrow points from the word "out" in the first line to the word "out" in the second line, with the text "Allowed" written below it. A bracket on the right side of the code is labeled "Scriptlet tag".

Q) can we override jspInit() and jspDestroy() in a jsp page?

Ans- Yes, we can override in <declaration> tag of jsp.



*** Q) can we override servlet life cycle methods in a jsp page?

Ans- 1) NO, the reason is when a jsp page translated into servlet it's super class overridden servlet life cycle methods with final modifier.

2) In a super class, if a method is final then in subclass it can't be overridden. So, we can't override servlet life cycle methods in jsp page.

Q) can we write an interface or a class in a jsp page?

Ans - yes, we can define interface or a class in a jsp under declaration tag.

a.jsp

2.0 !

```
interface MyInter  
{  
}  
  
class MyClass implements  
    MyInter  
{  
}  
/*>
```

a.jsp.java

```
public class a.jsp extends JSP  
{  
    interface MyInter  
    {  
    }  
  
    class MyClass implements  
        MyInter  
    {  
    }  
  
    public void jspService() throws ServletException  
    {  
    }  
}
```

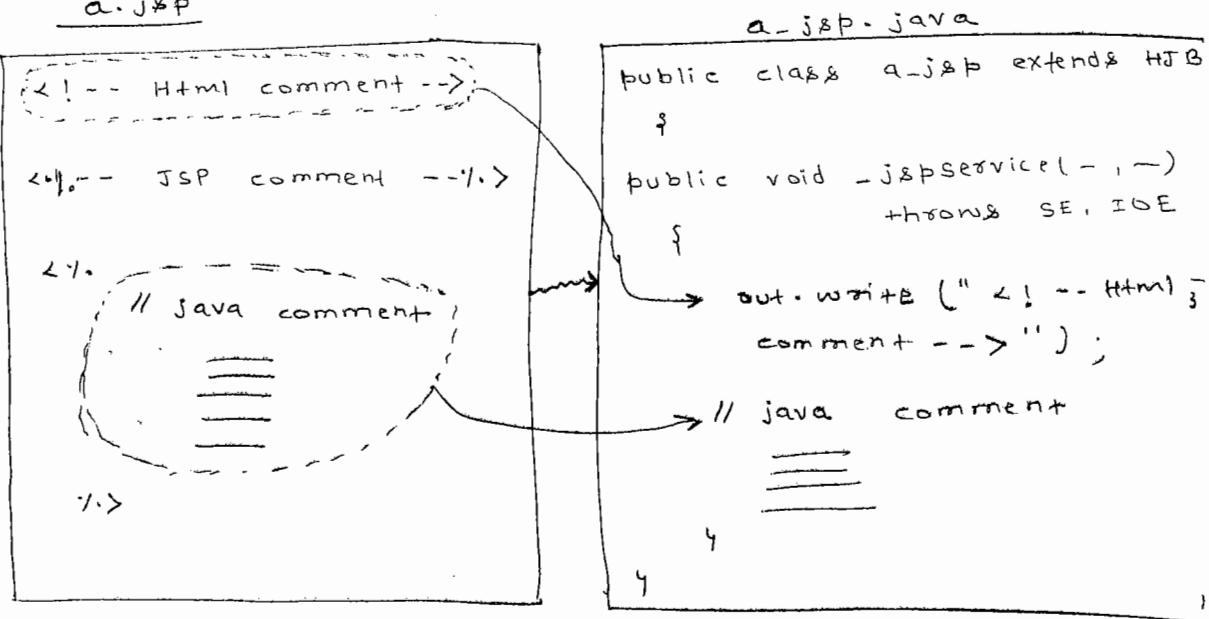
Note

- 1) In java we can define a class in a class
- 2) we can define interface in a interface.
- 3) we can define a class in a interface.
- 4) we can define an interface in a class
- 5) we can't define a method in a method.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

JSP comments

- 1) In a JSP page, we can write 3 types of comments —
 - (i) HTML comments
 - (ii) JSP comments (Invisible / Hidden comments)
 - (iii) Java comment
- 2) A JSP page contains HTML tag also. So, we can write HTML comments.
- 3) In JSP page we can write java code in scriptlet tag and in declaration tag. So, we can write java comments.
- * 4) In expression tag we can't write the comments.
- 5) When a JSP page is translated into a servlet we can see HTML comments and java comment in the servlet but we can't see JSP comment. They are called hidden comments or invisible comments.
- 6) If we write HTML comment then that comment is converted to out.write statement and this statement is inserted into `_jspService(--)`



Q) Which type of comments will come along with response to the browser ?

Ans - Html comments .

SRI RAHMDEVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

JSP configuration

- 1) A JSP page is optional to configure in web.xml .
- 2) suppose , if we want to configure then a jsp is configured like a servlet only . ~~The only difference~~
- 3) The only difference is , in place of < servlet-class > tag we need to ~~type~~ write < jsp-file > tag .

For eg -

< servlet >

< servlet-name > xxx < /servlet-name >

< jsp-file > /a.jsp < /jsp-file >

</servlet>

< servlet-mapping >

< servlet-name > xxx < /servlet-name >

< url-pattern > /jstl < /url-pattern >

</servlet-mapping >

a) If a jsp page is configured then we can send request from browser in 2 ways -

(i) By typing the filename

(ii) By typing the url pattern

b) If the jsp page is not configured then we can send request only by typing file name.

request : (if configured)

http://localhost:2015/App1/a.jsp

(or)

http://localhost:2015/App1/jstl

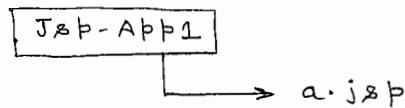
request : (if not configured)

http://localhost:2015/App1/a.jsp

Example 1

- 1) We are creating a JSP page which counts the no. of request.

without Eclipse IDE



14/10/15
Bhaiya B'day

```

<!-- a.jsp -->
<%-- This is a first jsp page --%>
<%!
    //instance variable
    private int count = 0;
%>
<%
    count++;
%>
<%> The count is : <%= count %> </%>
  
```

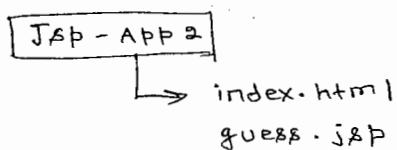
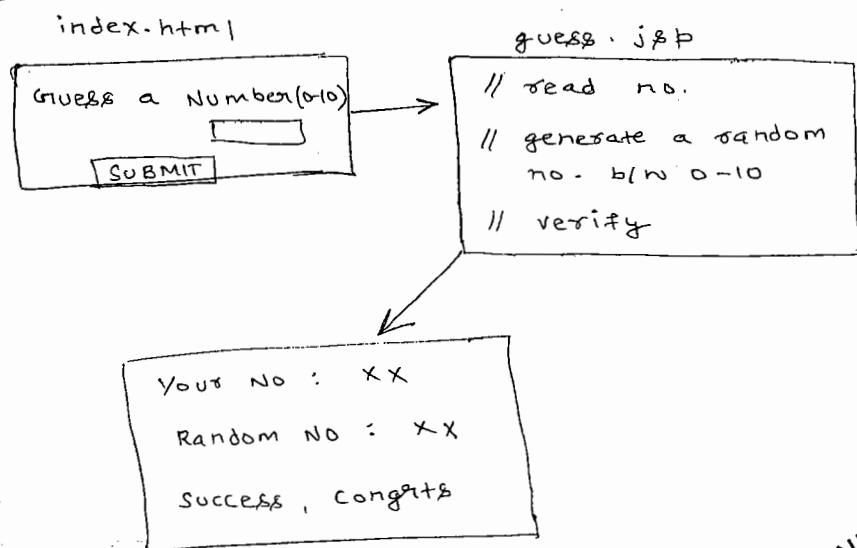
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Deploy the application in Tomcat / webapp folder
- Start the server
- Open the browser and type the following request
`http://localhost:2015/JSP-APP1/a.jsp`

Note- We can find the servlet and its class files at the following location.

Tomcat 7.0 / root / catalina / localhost / JSP-APP1 / org / apache / jsp

Example 2



```

<!-- index.html -->
<center>
    <form action = "guess.jsp" >
        guess a number (0-10) :
        <input type = text name = "t1" > <br>
        <input type = submit value = "submit" >
    </form>
</center>

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

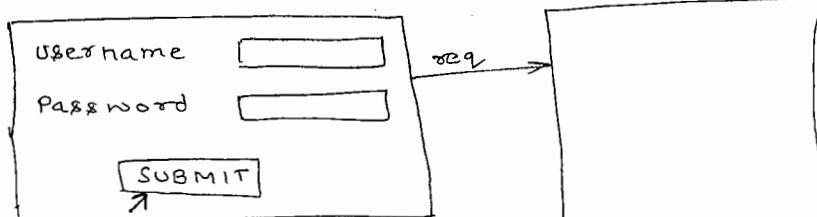
```
<!-- guess.jsp -->

<%
    // read input value
    int i = Integer.parseInt(request.getParameter("t1"));
    // create Random class object
    java.util.Random random = new java.util.Random();
    // read random no. b/w 0 to 10
    int j = random.nextInt(10);
    out.println(" Your no. :" + i);
    out.println("<br>");
    out.println(" Random No. :" + j);
    if (i == j)
    {
        out.println(" success. congrats");
    }
    else
    {
        out.println(" sorry. try again");
        out.println("<a href = index.html> click </a>");
    }
%>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Example 3

login.html



login.jsp

Login successful

jsp-App3

login.html
login.jsp

<!-- login.html -->

<center>

<form action = "login.jsp" >

 username : <input type = "text" name = "uname" >

 password : <input type = "password" name = "pwd" >

 <input type = "submit" value = "submit" >

</form>

</center>

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
< !-- login.jsp -->  
2: <%  
    // read username and password  
  
    String s1 = request.getParameter("uname");  
    String s2 = request.getParameter("pwd");  
    int i = check(s1, s2);  
  
    if (i == 1)  
        out.println("Login Success");  
    else  
        out.println("Login Failed");  
%>
```

```
2: !
```

```
public int check (String str1, String str2)  
{  
    if (str1.equals("sathya") && str2.equals("java"))  
        return 1;  
    else  
        return 0;  
}
```

```
%>
```

SRI KAGAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Directives in JSP

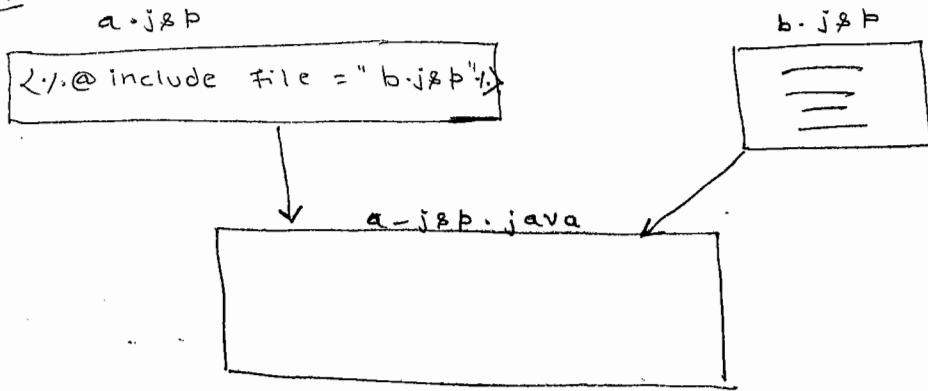
- 1) Directives allows programmer to give a direction to the page compiler.
- 2) with the help of directives we can't write any java logic and we can't display any output on browser.
- 3) There are 3 types of directives
 - (i) include
 - (ii) page
 - (iii) taglib
- 4) In a jsp page we can write a directive in html syntax or in xml syntax.
- 5) $\text{L} \cdot \text{y. } @\text{directivename}$ attributes $\text{y. } >$ \rightarrow HTML syntax
 $\text{Ljsp: directive. directivename}$ attributes $/> \rightarrow$ XML syntax

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

include directive

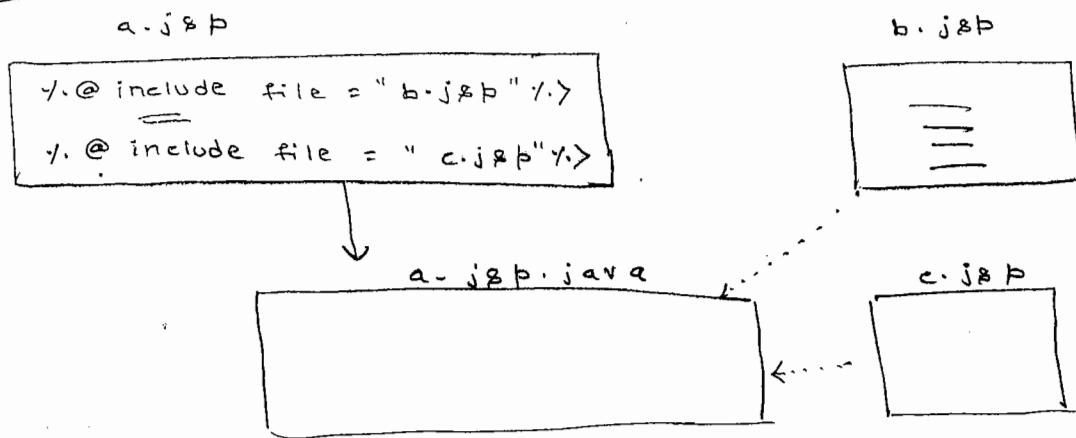
- 1) we use this directive to include source code of another page into the same servlet when translating the current page.
- 2) include directive has only one attribute called file.
- 3) include directive includes source code of another page so it is called static including.

Example 1

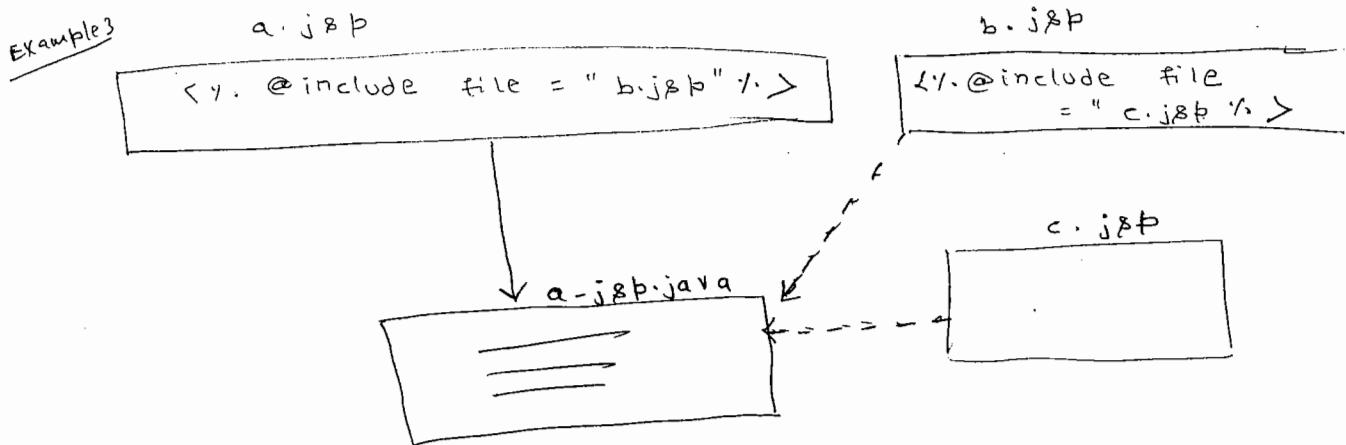


SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Opp. CDAC, Ayyagar Bakery,
Ameerpet, Hyderabad.

Example 2



Example 3



15 Oct 15 Page directive :-

i) import :

- 1) This attribute is used to import one or more packages in a jsp.
- 2) At translation time, page compiler adds all the imports on top of the servlet.
- 3) In a jsp page, a package can be imported either on top of the jsp or in the bottom or in the middle of page.
(At any place)

Example1

a.jsp

```
<%@page import = "java.util.*" %>
```

|||

Example2

a.jsp

```
<%@page import = "java.util.*" %>
```

|||

Example3

a.jsp

```
<%@page import = "java.sql.*" %>
```

|||

```
<%@page import = "java.util.*" %>
```

Example4

a.jsp

```
<%@page import = "java.sql.*", java.util.* %>
```

Example5

a.jsp

```
<%@page import = "java.sql.*"  
import = "java.util.*"
```

```
%>
```

SRI RAGHAVENDRA XEROX
Material Available
Software Languages Ayyagar Bakery,
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

2) Session

- 1) This attribute tells page compiler, whether to add session object to the servlet at translation time or not.
- 2) The default value of session attribute is "true", it means session object is added to the servlet.
- 3) If session = "false" then at translation time, session object will not be added to the servlet.
- 4) we can repeat session attribute in a jsp page but each time its value must be same.

Example 1

a.jsp

```
<%@ page session = "true" %>
=====
<%@ page session = "false" %>
```

Exception

Example 2

a.jsp

```
<%@ page session = "false" %>
=====
<%@ page session = "false" %>
```

valid

Note

If session = "false" then implicit object session is not allowed to use in that jsp page.

Ex-

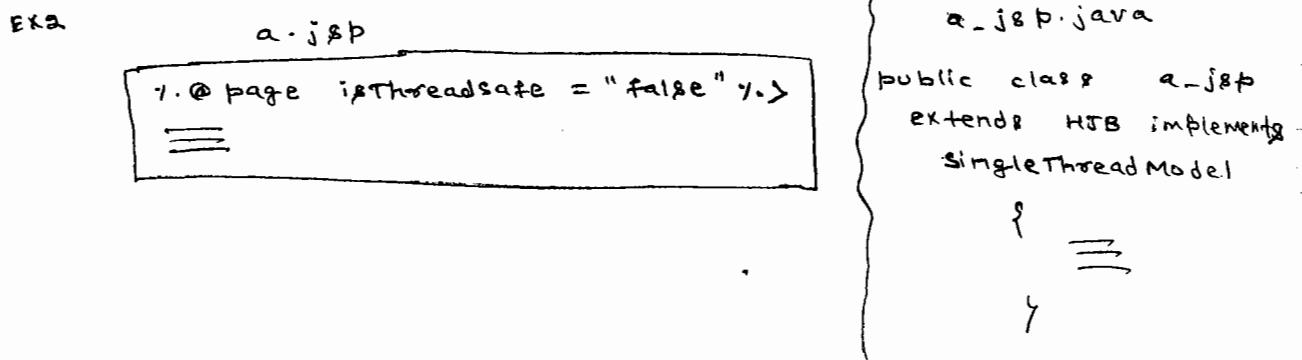
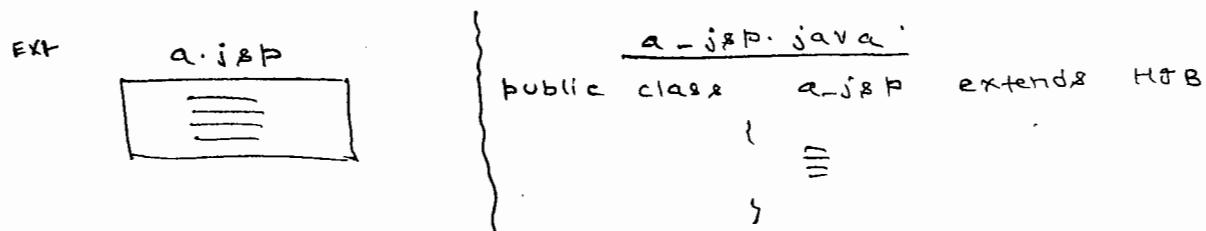
a.jsp

```
<%@ page session = "false" %>
<%
    String str = session.getId();
    ↓
    implicit object
    (not allowed)
%>
```

3) isThreadSafe

- 1) This attribute tells page compiler whether translated servlet ~~should~~ should implement SingleThreadModel interface or not.
- 2) The default of this attribute is true.
- 3) If `isThreadSafe = "true"` then page compiler doesn't implement the `Servlet` class from `singleThreadModel`.
- 4) If `isThreadSafe = "false"` then page compiler implements `Servlet` from `singleThreadModel`.
- 5) If servlet implements `singleThreadModel` then only one thread is allowed to access that servlet object at a time.

isThreadsafe :



4) errorPage

- 1) This attribute tells page compiler, to add the code for forwarding a request if an exception is occurred in current jsp page.
- 2) A request is forwarded to the given error page only if exception is occurred otherwise not forwarded.

Ex. jsp

```
<%@ page errorPage = "ex.jsp" %>
```

==

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

5) iSErrorPage

- 1) This attribute tells page compiler that whether a jsp can act as an error page or not.
- 2) Default value of this attribute is false. It means, a jsp page is not acting as an error page.
- 3) We can make a jsp page as an error page by setting true to the iSErrorPage attribute.
- 4) If iSErrorPage = "true" then only Exception object (implicit object) can be used in a jsp.

Ex. jsp

Ex1

```
<%@ page iSErrorPage = "true" %>
```

exception

allowed

EX 2

```
<%@ page isErrorPage = "false" %>
```

exception

not allowed

6) contentType :

1) In JSP we can set MIME type of response in two ways -

- (i) By using contentType attribute
- (ii) By using setContentType()

2) The default value of contentType attribute is text/html.

(i) <%@ page contentType = "text/xml" %>
(or)

<%

response.setContentType ("text/xml");

%>

17/10/15

7) Buffer

1) In JSP out is an implicit object and out object is created for JspWriter class.

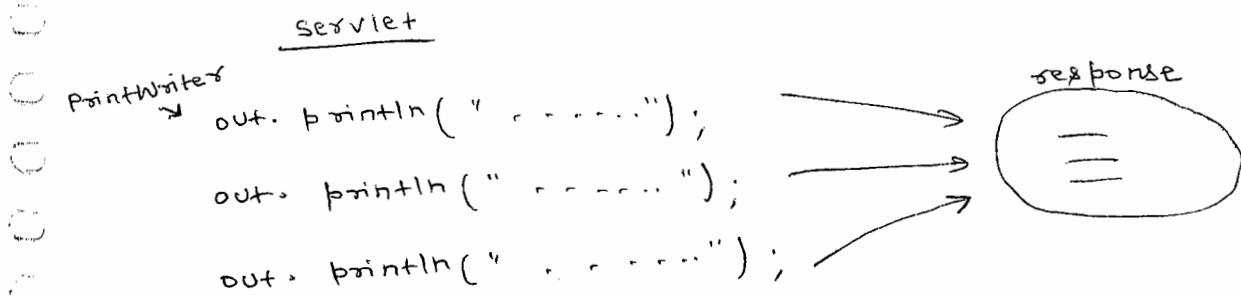
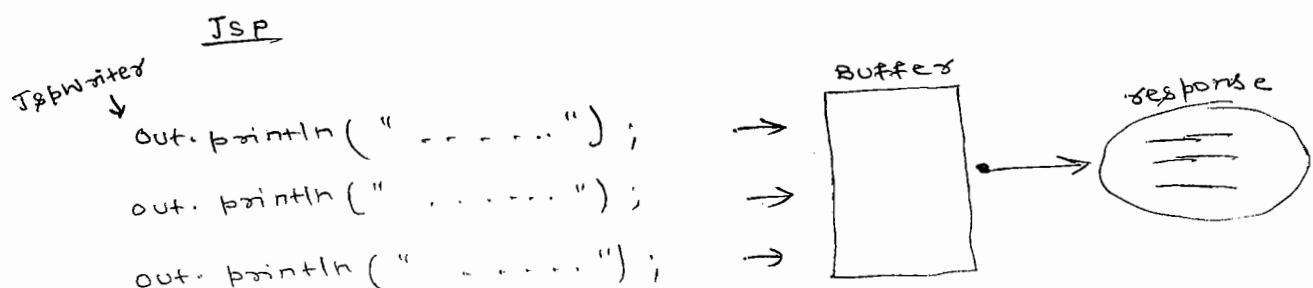
2) When out object is created in JSP then automatically a buffer of size 8 KB is also allocated to that out object.

3) When writing some response data to the response object, out object of JSP will write the data to the buffer. Later the data is transferred to the response object.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 4) The buffer attribute of page directive is used to either increase or decrease a buffer size or buffer can be removed.
- 5) In servlet technology, we use PrintWriter class object to write the response^{data} to the response object. with PrintWriter class object buffer is not created.
- * 6) The difference b/w JspWriter and PrintWriter classes is JspWriter class object manages buffer but PrintWriter class object doesn't manage buffer.
- 7) JspWriter = PrintWriter + Buffer
- 8) we can make JspWriter class is equal to PrintWriter class by removing the buffer.

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Outside Bangalore Ayyagar Bakery,
 Opp. CTAC, Barkampet P.O.,
 Ameepet, Hyderabad.



- g)
- 1) `page buffer = "12 KB" // increased`
 - 2) `page buffer = "4 KB" // decreased`
 - 3) `page buffer = "none" // removed`

8) autoFlush :

- 1) This attribute is used to either to enable or to disable autoflush mode of buffer.
- 2) The default value of this attribute is true. It means autoflush mode is enabled.
- 3) If we want to disable autoflush mode then autoFlush = "false"
- 4) If autoFlush mode enabled then container will take care about flushing the buffer at the appropriate times.
- 5) If autoFlush mode is disabled then we need to manually flush the buffer by writing out.flush();
- 6) Flushing a buffer means transferring the data from buffer to the response object.
- 7) If autoFlush mode is disabled and we call flush() after buffer reaches to full then BufferOverflowException will be thrown. so, it is always recommended to enable autoFlush mode.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

g) language

- i) This attribute tells page compiler about language used in the scripting tags of jsp.
- ii) As of now, we can write only java ~~coding~~ code in scripting tag . so, the default value and the only possible value is java.

L1. @page language = "java" %>

h) info

- i) This attribute is used to write description of a page.
- ii) The description also looks like a comment only but it is a readable comment.
- iii) If we add info attribute then at translation time `getServletInfo()` will be added to the servlet class.
- iv) In a jsp page we can read the value of info attribute by calling `getServletInfo()`

a.jsp

%> @page info = "This is sample page" %>

%>
String str = getServletInfo();
%>

a.jsp.java

```
public class a.jsp.java extends HttpServlet  
{  
    public String getServletInfo()  
    {  
        return "this is sample page";  
    }  
}
```

11) extends

- 1) When a JSP page is translating, page compiler extends a servlet class from a super class provided by the container.
- 2) For ex, in Tomcat, the super class is HttpJspBase.
- 3) Suppose, if we want to create a super class and if we want to tell the translators to extend our superclass then we need to use extends attribute.
- 4) Creating a super class will be burden on a programmer. So, this extends attribute is very rarely used.

For ex:

```
package com.sathyajsp;
public class MyHttpBase extends HttpServlet
    implements JspPage
```

{
==
y

a.jsp

```
<%@ page extends = "com.sathyajsp.MyHttpBase" %>
```

a-jsp.java

```
public class a_jsp extends MyHttpBase
```

{
==
y

13) isELIgnored

- 1) This attribute tells page compiler whether Expression language (EL) is enabled or disabled in a jsp page.
- 2) The default value of this attribute is "false". It means EL is enabled.
- 3) If we set "true" then EL is disabled.
- 4) If EL is enabled then translator translates EL statements into equivalent java code otherwise not translated.

18 Oct 15

Attribute name

import
session
isThreadSafe
errorPage
isErrorPage
contentType
buffer

default value

no
true
true
no
false
text/html
8 KB

CDAC INSTITUTE NENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Action tags in JSP

- 1) Action tags are used for following communication -
 - (i) JSP to JSP or servlet or html
 - (ii) JSP to Applet
 - (iii) JSP to Bean class
- 2) There are mainly six action tags with three sub action tags in JSP -
- 3) Action tags of JSP can be written in XML syntax only.

Main Action tags

- (1) <jsp: forward>
- (2) <jsp: include>
- (3) <jsp: plugin>
- (4) <jsp: useBean>
- (5) <jsp: setProperty>
- (6) <jsp: getProperty>

Sub Action tags

- 1) <jsp: param>
- 2) <jsp: attrs>
- 3) <jsp: fallback>

<jsp: forward>

- This action tag is used for forwarding a request from JSP page to another JSP page or servlet or html page.

SRI RAVENDRA KEROU
Software Languages Material Available
Beside Bangalore Ayagapet Road,
Mysore, Karnataka, India
Mobile: 98452 42345
Opp. CDAC, Balakampet Road,
Ayagapet, Mysore
Amerepet, Hyderabad

<jsp:forward>

a.jsp $\xrightarrow{\text{forward}}$ b.jsp

<jsp:forward page = "b.jsp" />

a.jsp $\xrightarrow{\text{forward}}$ servlet

<jsp:forward page = "servlet" />

a.jsp $\xrightarrow{\text{forward}}$ b.html

<jsp:forward page = "b.html" />

- 2) When forwarding a request we can add one or more parameters to the request by adding a .sub action tag <jsp:param>
- 3) If any request parameters are send from the browser then they are automatically forwarded along with the request. To add additional parameters while forwarding a request we use <jsp:param> sub action tag.
- 4) For ex, from html page on browser, if five parameters are send to a.jsp and from a.jsp we want to forward two more additional parameters to b.jsp then we need to add the two parameters with <jsp:forward> action tag.

a.jsp

```
<jsp:forward page = "b.jsp" />
```

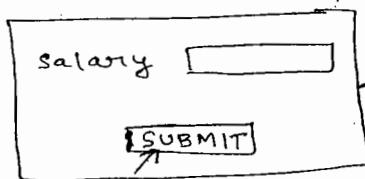
```
<jsp:param name = "p1" value = "100"/>
```

```
<jsp:param name = "p2" value = "200"/>
```

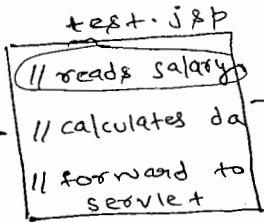
```
</jsp:forward>
```

19 Oct 15

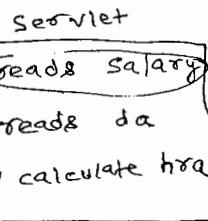
index.html



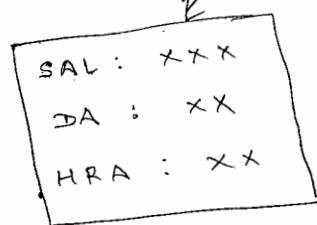
req



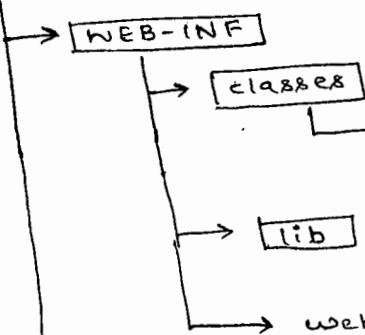
~~response~~



~~req~~ response



"JSP-Apps"



HRAServlet.java
HRAServlet.class

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

index.html

<center>

```

<form action = "test.jsp">
    salary : <input type = text name = "sal">
    <br>
    <input type = submit value = "submit" />
</form>
</center>

```

test.jsp

<%>

```

    int sal = Integer.parseInt(request.getParameter("sal"
        .trim()));

```

```
    double da = sal * 0.10;
```

<%>

<jsp:forward page = "hraserv" />

<jsp:param name = "pl"

value = "<% = da %>" />

</jsp:forward>

HRAServlet.java

```

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;

```

SRI RAMA AVENDRA XEROX
 Available
 SRI RAMA AVENDRA XEROX
 Bakery,
 Software Lab, 2nd Floor, Meggi Road,
 Beside Dabur, Barkampet Road.
 Opp. Ameerpet, Hyderabad.

```

public class HRAServlet extends genericServlet
{
    public void service( ServletRequest request,
                         ServletResponse response )
        throws SE, IOE
    {
        //read salary
        int sal = Integer.parseInt( request.getParameter("sal") );
        // read da
        double da = Integer.parseInt( request.getParameter("pl") );
        // calculatehra
        double hra = sal * 0.20 ;
        // set mime type
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(" salary : " + sal);
        out.println(" <br> ");
        out.println(" da : " + da);
        out.println(" <br> ");
        out.println(" hra : " + hra);
        out.close();
    }
}

```

XEROX
 SKI RAVENDRA
 Languages Material Available
 Software Bangalore Ayyagar Bakery,
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```
<!-- web.xml -->

<web-app>

<servlet>

<servlet-name> hra </servlet-name>
<servlet-class> HRAServlet </servlet-class>
</servlet>

<servlet-mapping>

<servlet-name> hra </servlet-name>
<url-pattern> /hra&gt; </url-pattern>
</servlet-mapping>

</web-app>
```

→ compile the servlet so that ".class" file
is created.

→ deploy the application folder Jsp-Apps in
webapps folder of Tomcat.

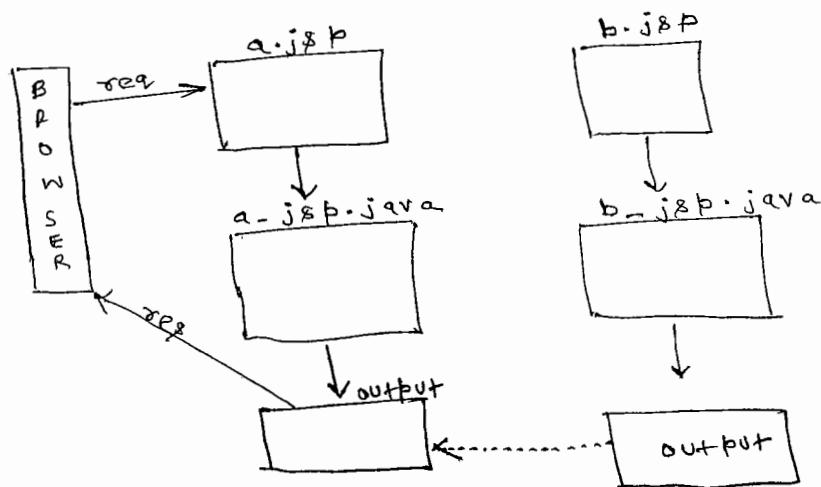
→ start the Tomcat server and type the
following request from the browser -

localhost : 2015 / Jsp-Apps |

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<jsp: include>

- 1) This action tag is used for including response of another jsp / servlet / html into current jsp response of current jsp.
- 2) This action tag includes the response at runtime. so, it is dynamic including.
- 3) suppose, if a.jsp includes b.jsp then for both a and b separate servlets are created at translation time and included at runtime.



20/10/15

- 1) In JSP we have both include directive and include action, for including one page to another. the differences b/w these two are like the following -

include directive

- 1) In include directive source page code of another page is included, so, it is static including.
- 2) include directive can't include source code of servlet.
- 3) we can't pass expression tag as a value to the file attribute of include directive.
- 4) If any changes are made in the source code in the included page then the changes are not included into the source page servlet.

include directive can be added to the jsp in html syntax or in xml syntax.

include action

- 1) In include action output of another page is included at runtime, it is called dynamic including.
- 2) include action can include output of servlet also.
- 3) we can pass expression tag as a value to the page attribute of include action.
- 4) If any changes are made in the included page then the changes are reflected in output.
- 5) include action can be added to the jsp only in xml syntax.

For ex -

<%

String str = "b.jsp";

%>

<%@ include file = " %. = str %>" %> // invalid

<jsp:include page = " %. = str %.>" %> // valid

→ while including the o/p of another page in current page, request is temporarily transferred to the destination page. With the request, we can send parameters to the included page by adding `<jsp:param>` sub action tag.

`<jsp:include page = "b.jsp" %>`

`<jsp:param name = "p1" value = "100" %>`

`<jsp:param name = "p2" value = "200" %>`

`</jsp:include %>`

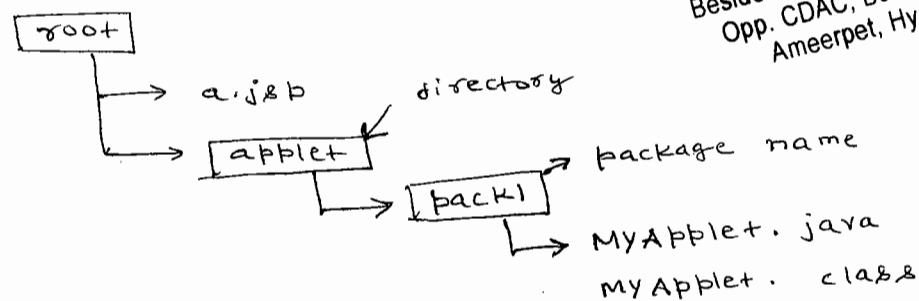
- 4) `<jsp:include>` action tag has page attribute and also flush attribute.
- 5) flush attribute tells the container to clear the buffer before including the output of another jsp page.
- 6) The default value of flush attribute is false. To flush the buffer we can set `flush = "true"`.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<jsp:plugin>

- 1) This action tag integrates a jsp with applet.
- 2) When a jsp page wants to show the response on browser in graphics then jsp integrates with an applet.
- 3) <jsp:plugin> action has the following attributes -
 - (i) type = "applet" (fixed)
 - (ii) code = "applet classfile"
 - (iii) codebase = "directory of applet class"
 - (iv) width = "xxx"
 - (v) height = "xxx"

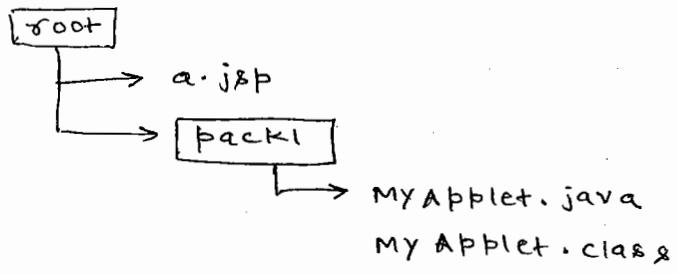
Example:



a.jsp

```
<jsp:plugin type = "applet"  
           code = "pack1.MYApplet.class"  
           codebase = "applet"  
           width = "800"  
           height = "800" />
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



a.jsp

```

<jsp:plugin type = "applet"
             code = "pack1.MYApplet.class"
             height = "300"
             width = "300" />

```

21/10/15 4) <jsp:plugin> action tag has two sub

action tag -

- (i) <jsp:params>
- (ii) <jsp:fallback>

5) These two action sub tags can be used with <jsp:plugin>

6) <jsp:params> contains atleast one

<jsp:param> tag to pass a parameter to an applet.

7) <jsp:fallback> sub action tag is used to define an alternate message to display on browser, if an applet is not displayed on the browser.

For ex

```
<jsp:plugin type = "applet"  
            code = "MyApplet.class"  
            codebase = ".."  
            width = "300"  
            height = "300"  
  
<jsp:params>  
  <jsp:param name = "p1" value = "10"/>  
  <jsp:param name = "p2" value = "20"/>  
</jsp:params>  
  
<jsp:fallback> Unable to load plugin  
</jsp:fallback>  
  
</jsp:plugin>
```

<jsp:useBean>

- 1) We create a java bean in a JSP Application, if any common java code exist in multiple JSP pages.
- 2) If multiple JSP pages has any common java code then to get the reusability, we separate the java code into java bean class.

- 3) A java bean class is a java class which follows the following rules -
- (i) class must be public
 - (ii) class must contain a public default constructor.
 - (iii) private properties must contain either getter, setter or both methods.
 - (iv) class can atmost implement either serializable or ~~External~~ Externalizable.
- 4) <jsp: useBean > action tag either reads a java bean object from a given scope if exists or creates a new object for the java bean class and stores that object in the given scope.
- 5) jsp technology has defined four scopes where each scope is used to share the data across multiple pages..
- (i) page
 - (ii) request
 - (iii) session
 - (iv) application

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Linda Bangalore Ayyagar Bakery,
Upp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Syntax

```
<jsp:useBean id = "objectname"  
class = "fully qualified classname"  
scope = "scope name"/>
```

- 6) When `<jsp:useBean>` tag is finding for a bean object in the given scope then id acts as a key.
- 7) When creating new a object for a Bean class id acts as object name.
- 8) To use any java class object in a JSP page, that class must be stored within a package.
- 9) The default scope of a bean is a page.

For ex

```
<jsp:useBean id = "obj"  
class = "pack1.TextBean"/>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

27/10/15

<jsp : setProperty >

- 1) This action tag will set value to a bean property by calling setter method of the property.
- 2) For ex, we have a bean called LoginBean with property uname and pwd. A jsp page will set the values to both the properties like the following -

```
<jsp : setProperty name = "obj1"  
                    property = "uname"  
                    value = "sathya" />
```

```
<jsp : setProperty name = "obj1"  
                    property = "pwd"  
                    value = "java" />
```

- 3) The value of name attribute must be same as the value of id attribute of `<jsp : useBean>` tag.
- 4) A jsp page can directly set a request parameter value to a property, by using param attribute.

```
<jsp : setProperty name = "obj1"  
                    property = "uname"  
                    param = "username" />
```

Annotations:

- objectname → points to obj1
- bean property → points to uname
- request parameter name → points to username

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDA Colony, Balkampet Road,
Ameerpet, Hyderabad.

- 5) name and property attributes are mandatory.
we can use either param or value but
not both at a time.
- 6) If request parameter name are same as
bean properties names then we can
write a special value to the property
called "*" .
- 7) For example , request parameter names are
uname , pwd and bean property
names are also uname , pwd then
we can set request parameter values
to bean properties by property = "*" .

SRI RAGHAVENDRA XEROX : SetProperty

Software Languages Material Ave
Chennai 600090
Tamil Nadu
India

```
name = "obj1"  
property = "*" />
```

- 8) A bean has 3 properties and there are
only two request parameters , whose
names are matched with bean properties
then we can set the ^{value to} third property
with value attribute .
- 9) suppose , request parameter names are
uname , pwd and bean property names
are uname , pwd , email then we
can set the value like the following -

<jsp: setProperty

name = "obj1"

property = "*" />

<jsp: setProperty name = "obj1"

property = "email"

value = "abc@gmail.com" />

<jsp: getProperty>

1) This action tag reads value of a bean property by calling getter method.

2) This action tag will print a bean property value on browser.

3) This <jsp: getProperty> action tag has only two attributes -

(i) name

(ii) property

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

E.g.

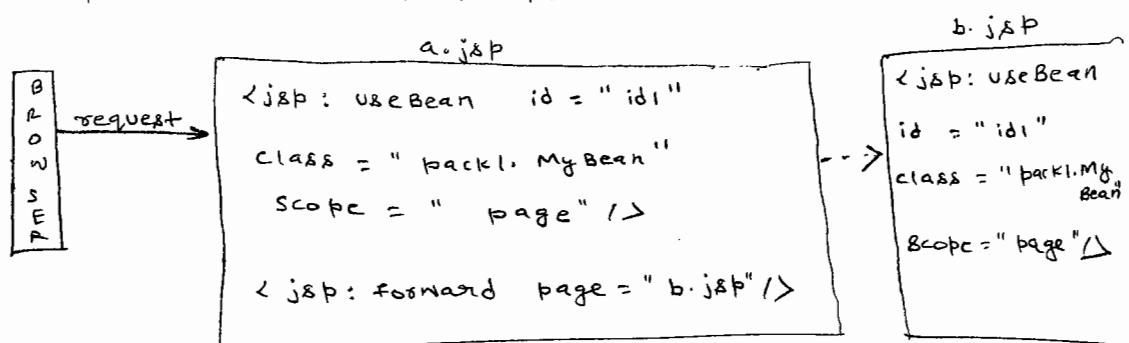
<jsp: getProperty name = "obj1"
property = "uname" />

4) We can't write property = "*" in

<jsp: getProperty> action.

Bean Scopes

- 1) Page scope :
- 2) The bean object stored in Page scope is not shareable with other pages.
- 3) JSP container creates a separate page scope for each JSP page.
- 4) Page scope is shareable within the page. So, it is local scope.



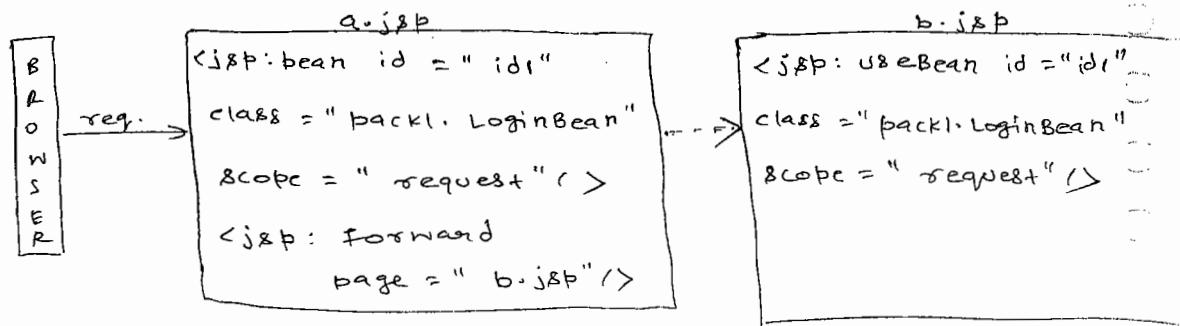
- 4) In the above diagram, Bean class object is created in a.jsp and also another object is created in b.jsp because page scope of a.jsp is not shareable with b.jsp.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Request scope

- 1) A bean object stored in request scope shareable with pages participating in the same request.
- 2) JSP container creates a request scope map and the data stored in that map is shareable with other pages when a request is forwarded to the other pages.

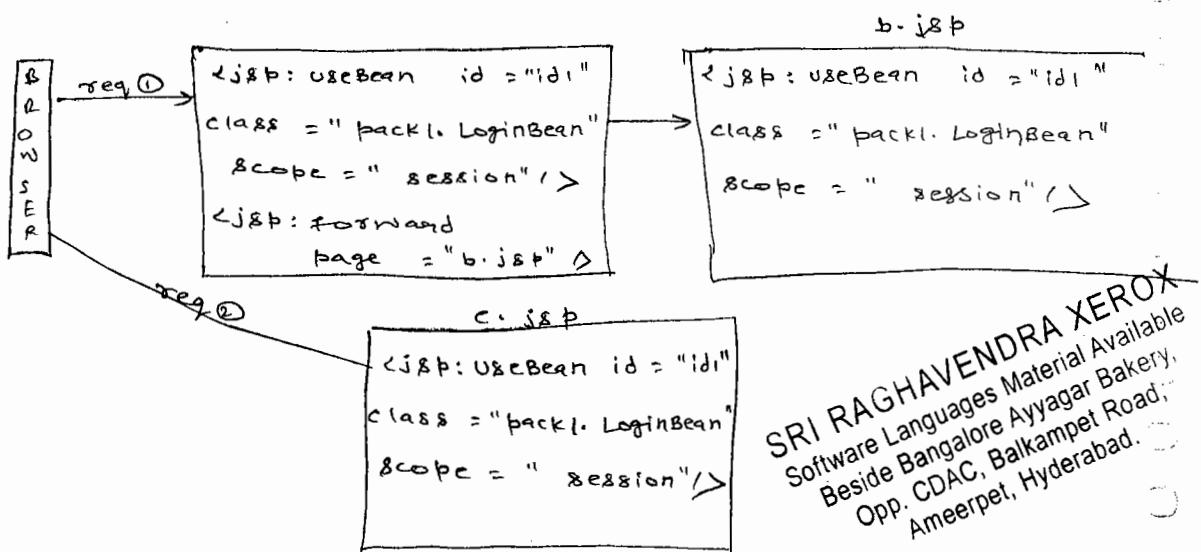
- 3) A JSP container creates one request scope map for each request separately.



- 4) In the above diagram, bean class object is created in **a.jsp** and the same object is shared in **b.jsp** also.
- 5) Because same request is forwarded to **b.jsp**

3.) Session scope

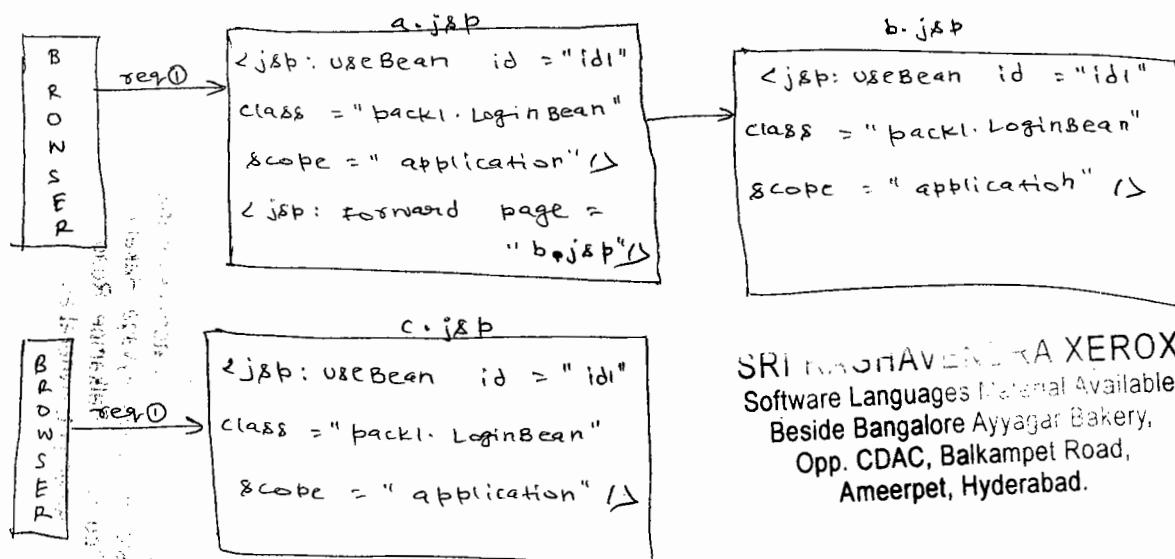
- 1) A bean object stored in session scope is sharable in multiple requests of the same client.
- 2) JSP container creates a session scope map object, for each client separately and the data stored in that map object is sharable in multiple requests within session of that client.



3) In the above program, Bean object is created in a.jsp and the same object is shared in b.jsp and c.jsp because same client has send a 2nd request to the c.jsp.

4) Application scope

- 1) A Bean object is stored in application scope is shareable to all the client in all the pages across the entire application.
- 2) JSP container creates an application scope map object and makes its data as shareable to the entire application.

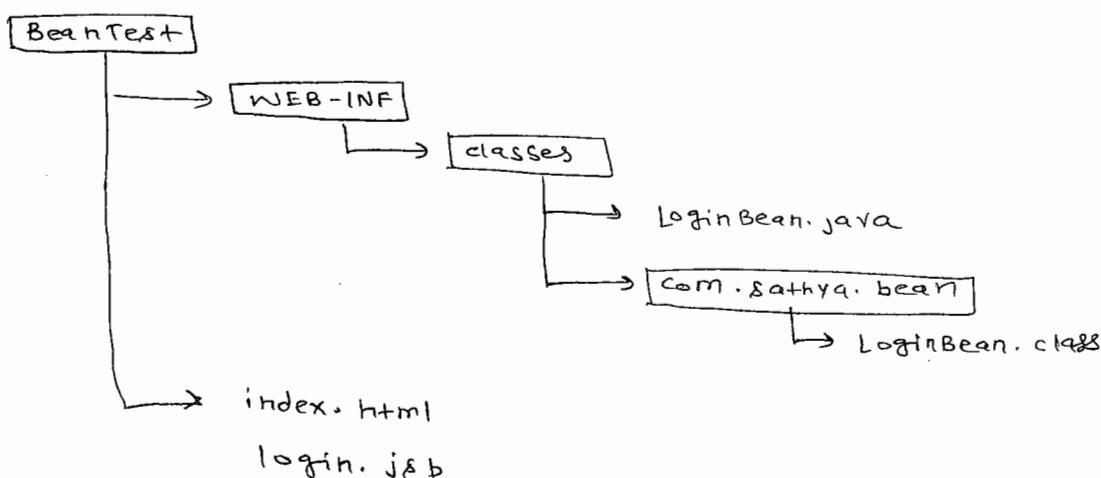


SRI MUNI HAVE A XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 3) In the above diagram bean object is created in a.jsp and the same object is shared in b.jsp and c.jsp also because application is a global scope and data is shareable in all the jsp pages (o*) entire application.

Example

- 1) In this example, a jsp page uses `<jsp:useBean>` and `<jsp:setProperty>` actions.
- 2) JSP page calls check() of the bean to verify input values are valid or not



// LoginBean.java

```

package com.sathyas.bean;

public class LoginBean {
    private String uname;
    private String pwd;
    // setters & getters

    private boolean check() {
        if (uname.equals(pwd))
            return true;
        else
            return false;
    }
}
  
```

SRI KRISHNENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```
<!-- index.html -->  
<center>  
  <form action = "login.jsp" >  
    Username <input type = "text" name = "uname"  
              placeholder = "username" > <br>  
  
    Password <input type = "password"  
              name = "pwd"  
              placeholder = "password" > <br>  
    <input type = "submit"  
          value = "submit" >
```

login.jsp

```
<jsp:useBean id = "loginBean"  
             class = "com.sathya.bean.LoginBean"  
             scope = "page" />
```

```
<jsp:setProperty name = "loginbean"  
                  property = "*" />
```

</hi>

RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

<%>
boolean flag = loginBean.check();
if (flag)
    out.println(" Success ");
else
    out.println(" Failed ");
%>
</h1>

```

compile & run

E:\j&p\Beantest\WEB-INF\classes >

javac -d . javac LoginBean.java

- Deploy Beantest in webapps directory of tomcat.
- Start Tomcat server and open a browser and type the following request —

http://localhost:2015/Beantest/

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Amerpet, Hyderabad.

29/10/15

Implicit Objects in JSP

1) Implicit objects are also called implicit reference variable.

2) There are 9 implicit object in JSP -

name	type
request	HttpServletRequest (i)
response	HttpServletResponse (i)
session	HttpSession (i)
out	Writer (c)
exception	Throwable (c)
config	ServletConfig (i)
application	ServletContext (i)
page	Object (c)
pageContext	PageContext (ac)

3) In a JSP page, session object is allowed to use if session is enabled in the page.

4) In a JSP Page session is enabled by default because, session attribute is by default true

5) exception object is allowed, if a page is acting as an error page.

6) To act a page as an error page we need to set isErrorPage attribute as "true"

7) In JSP technology we call an object application and it is same as context object in Servlet Technology.

8) Page object refers current object. Page object is same as this.

9) We can find the following statement in `-jspService()`, after a JSP translated to a Servlet

```
final java.lang.Object page = this;
```

10) pageContext object is a special implicit object in JSP and given for the following two purposes —

(i) To store remaining implicit objects of a page in a single pageContext object.

(ii) To store attributes in either page scope or request scope or session scope or application scope.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beeje Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Attributes in JSP

- 1) A JSP can share its data with other JSP pages through attributes.
- 2) Attribute is a key/value pair of data.
- 3) To share the data attribute must be stored in an appropriate scope.
- 4) In JSP we can store an attribute in one of the four scope —
 - (i) page
 - (ii) request
 - (iii) session
 - (iv) application
- 5) An attribute can be stored in page scope, only using pagecontext object.

For ex,

```
pageContext.setAttribute("k1", "java");  
(or)
```

```
pageContext.setAttribute("k1", "java", 1);  
(or)
```

```
pageContext.getAttribute("k1", "java",  
pageContext.PAGE_SCOPE);
```

- 6) In Pagecontext class, four public static final int variables are provided, to indicate four scopes in JSP —

```
(i) page public static final int  
PAGE_SCOPE = 1;
```

- (ii) public static final int REQUEST_SCOPE = 2;
- (iii) public static final int SESSION_SCOPE = 3;
- (iv) public static final int APPLICATION_SCOPE = 4;

7) we can store an attribute in request scope using request object or pagecontext.

For ex -

```
request.setAttribute("k2", "JSP");
(oo)
```

```
pageContext.setAttribute("k2", "JSP", 2);
(oo)
```

```
pageContext.setAttribute("k2", "JSP", PageContext.REQUEST_SCOPE);
(oo)
```

8) To store an attribute in session scope
we can use session or pagecontext object

For ex -

```
session.setAttribute("k3", "JSP");
(oo)
```

```
pageContext.setAttribute("k3", "JSP", 3);
(oo)
```

```
pageContext.setAttribute("k3", "JSP", PageContext.SESSION_SCOPE);
(oo)
```

9) To store an attribute in application scope
we can use either application or pagecontext object .

For ex

```
application.setAttribute(" K4 ", " JSP " );  
                                (OS)  
pageContext.setAttribute(" K4 ", " JSP ", 4 );  
                                (OS)  
pageContext.setAttribute(" K4 ", " JSP ", pageContext.  
                                APPLICATION_SCOPE );
```

30/10/15

- 10) We can read an attribute from page scope
only by using pagecontext object .

For ex :

```
Object o = pageContext.getAttribute(" K1 " );  
Object o = pageContext.getAttribute(" K1 ", 1 );  
Object o = pageContext.getAttribute(" K1 ",  
                                pageContext.PAGE_SCOPE );
```

- 11) By we can read an attribute from request
scope for this we can use either
pagecontext object or request object .

For ex :

```
Object o = pageContext.getAttribute(" K2 ", 2 );  
Object o = request.getAttribute(" K2 " );
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(2) To read an attribute from session scope, we can use either pagecontext object or session object.

For ex -

```
Object o = pageContext.getAttribute("k3", 2);
```

```
Object o = session.getAttribute("k3");
```

(3) To read an attribute from application scope, we can use either pagecontext object or application object.

For ex -

```
Object o = pageContext.getAttribute("k4", 4);
```

```
Object o = application.getAttribute("k4");
```

* Q- What is the difference b/w getAttribute() and findAttribute() of PageContext object?

Ans- getAttribute() will only search for the key under given scope. If not exist then returns null.

But, findAttribute() will search for the key in high level scope also. If not found in high level scope also then returns null.

For example

```
Object o = pageContext.getAttribute("k2", 2);
```

It will search for the key only in request scope.

```
Object o = pagecontext.getAttribute("k2", 2);
```



It will search for the key in request scope, then session scope, then application scope.

Example

- 1) we are creating three jsp pages a.jsp, b.jsp, c.jsp. we forward the request from a.jsp to b.jsp and we send a separate request to c.jsp.
- 2) we are finding which scope attribute are shareable b/w the two request and b/w the two clients by using this example.

AttributeText

↳ a.jsp

b.jsp

c.jsp

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
<!-- a.jsp -->
```

↳

```
pagecontext.setAttribute("k1", "A");  
pagecontext.setAttribute("k2", "B", 2);  
pagecontext.setAttribute("k3", "C", 3);  
pagecontext.setAttribute("k4", "D", 4);
```

↳

```
<jsp:forward page = "b.jsp" />
```

```
<!-- b.jsp -->  
<%  
out.println(pageContext.getAttribute("k1"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k2"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k3"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k4"));  
%>
```

```
<!-- c.jsp -->
```

```
<%
```

```
out.println(pageContext.getAttribute("k1"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k2"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k3"));  
out.println("<br>");  
out.println(pageContext.getAttribute("k4"));  
%>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Browsers

req1: http://localhost:2015/AttributeTest/a.jsp ←

O/P :

null

B

C

D

req : http://localhost:2015/AttributeTest/c.jsp ↪

O/P :

null
null
c
d

Browser

req : http://localhost:2015/AttributeTest/c.jsp ↪

O/P :

null
null
null
d

CUSTOM tags in JSP

- 1) A custom tag is a userdefined tag.
- 2) Already jsp technology has provided predefined jsp tags as common to all the developers. Apart from the predefined tags if any new tags are required then the developer can create own tags with logic and a developer can use them in multiple jsp pages of that project. The own tags created are called custom tags.

31/10/15

3) To create a custom tag, we need to follow the below steps -

- (i) we need to define the logic of the custom tag in a java class. It is called a tag handler.
- (ii) we need to configure one or more custom tags details in a tld file (tag library descriptor).
- (iii) we need to configure location of each tld file with a uri in web.xml.
- (iv) Finally, we need to import tag library uri into the jsp page.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Creating a Tag handler

- 1) A taghandler is a java class and the jsp container will create an object of tag handler and calls its methods at runtime.
- 2) we can create a taghandler either by extending TagSupport or by extending BodyTagSupport classes.

- 5) TagSupport and BodyTagSupport are the classes given by JSP API in javax.scarlet.jsp.tagext.
- 4) If you want to define the logic for a custom tag beginning starting and ending points then we need to extend TagHandler from TagSupport.
- 5) If you want to define the logic for a custom tag starting point, for ending point and before control enters into the body and when after executing the body then we need to extend a TagHandler from BodyTagSupport.
- 6) For ex, if we want to define logic for a custom tag start point then we need to override doStartTag(). Similarly to define a logic for custom tag end point we need to override doEndTag().

For example

```
public class HelloTagHandler extends TagSupport
{
    public int doStartTag() throws JspException
    {
        // ...
    }

    public int doEndTag() throws JspException
    {
        // ...
    }
}
```

- 7) doStartTag() can return one of the following two variable values.
- (i) SKIP_BODY
 - (ii) EVAL_BODY_INCLUDE
- 8) If a custom tag contains body then we must return EVAL_BODY_INCLUDE. If there is no body then return SKIP_BODY.
- 9) doEndTag() returns one of the following two variable values.
- (i) SKIP_PAGE
 - (ii) EVAL_PAGE
- 10) After the end of custom tag, if remaining JSP page exists then we must return EVAL_PAGE. If not exist then SKIP_PAGE.

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

Creating a tld file

- 1) A tld file looks like an xml file and we configure one or more custom tags in that file.
- 2) In an application we can divide custom tags into multiple groups and we can configure each group of tags in a separate tld file.

7) For ex.

None.tid

<taglib>

< tlib -version>1.2 </tlib -version>

<jsbt-version> 1.3 </jsbt-version>

<tag>

<name> *** </name>

<tag-class> **** </tag-class>

<body-content> empty/jsp </body-content>

4/ tag

</taglib>

SRI RAGHAVENDRA XEROX

SRI RAGHAVEN Software Languages Material Available
Bakery

Software Languages ...
Beside Bangalore Ayyagar Bakery
Opposite Kasturba Gandhi Road

Opp. CDAC, Balkampet Road

Ameerpet, Hyderabad.

2/11/15 configuring tld file

- 1) we need to configure a location of a tld file in web.xml
 - 2) with location, we also need to configure uri . JSP container reads location of tld file from web.xml. Then it will store the information of tags configure in tld file unique name called uri.
 - 3) Each tld file created in a web application must be configured in web.xml with <taglib> element.

// web.xml

```
<web-app>
  <jsp-config>
    <taglib>
      <taglib-uri> http://sathya.com/mytags </taglib-uri>
      <taglib-location> /WEB-INF / one.tld </taglib-location>
    </taglib>
  </jsp-config>
</web-app>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

importing uri in Jsp

- 1) If we want to write custom tags in jsp page then we need to import uri configured in web.xml .
- 2) jsp technology has provided <%@taglib> directive for importing uri .
- 3) Each custom tag can be written in a jsp page in xml syntax format .. so , a prefix is required for custom tag .

```
<%@taglib uri = "http://sathya.com/mytags"
prefix = "sathya" %>
```

(i) custom tags are divided into four types —

(ii) `<sathyas: hello />` \Rightarrow empty tag

(iii) `<sathyas: hello user="abcd" />` \Rightarrow with attribute

(iv) `<sathyas: hello >`

=====

\Rightarrow with body

`</sathyas: hello>`

(v) `<sathyas: hello>`

`<sathyas: hai>`

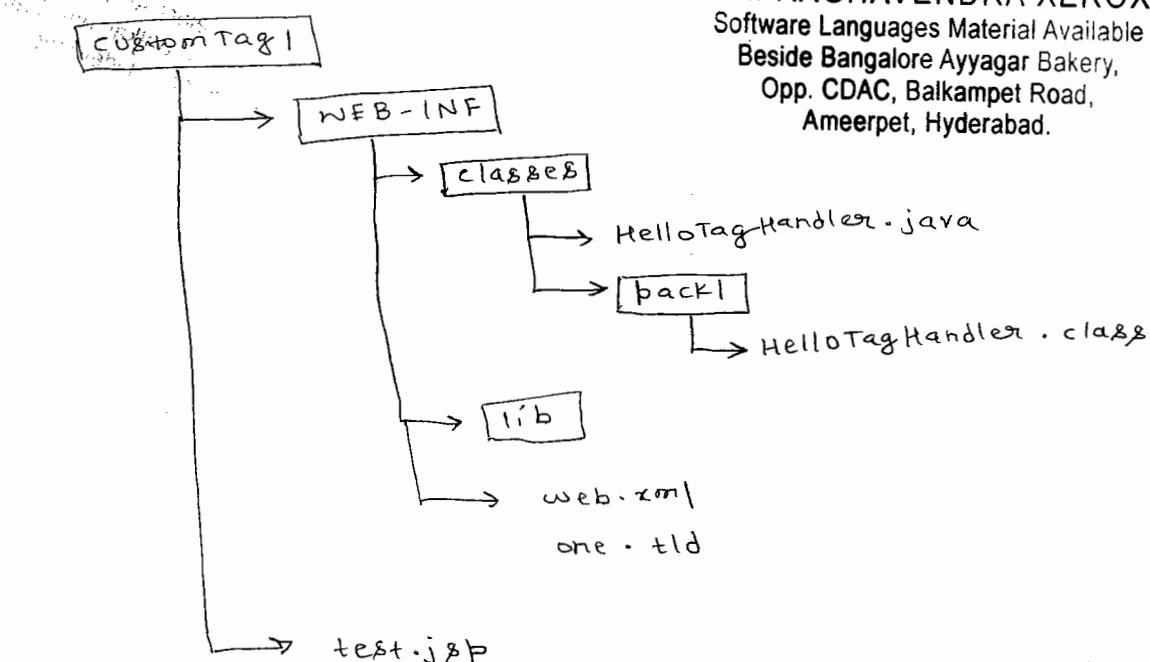
`</sathyas: hai>`

\Rightarrow with nested tag

`</sathyas: hello>`

Example

The following custom tag example creates an empty custom tag



// HelloTagHandler.java

```
package pack1;

import javax.servlet.jsp.tagext.TagSupport;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;

public class HelloTagHandler extends TagSupport
{
    public int doStartTag() throws JspException
    {
        try
        {
            JspWriter out = pageContext.getOut();
            out.println("<h1> Welcome </h1>");
        }
        catch (Exception e)
        {
            System.out.println("Error : " + e);
        }
        return SKIP_BODY;
    }

    public int doEndTag() throws JspException
    {
        try
        {
            JspWriter out = pageContext.getOut();
            out.println("<h1> Bye </h1>");
        }
        catch (Exception e)
        {
            System.out.println("Error : " + e);
        }
        return EVAL_PAGE;
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 1) In the above program, pageContext object came from super class.
- 2) When TagHandler object is created, immediately container calls setPageContext () of super class, by passing pageContext object as parameter.
- 3) super class stores pageContext object in a global variable pageContext. so, we can use super class pageContext variable (public variable in subclass).

```

//tld file
<taglib>
<tlib-version> 1.0 </tlib-version>
<jsp-version> 1.3 </jsp-version>
<tag>
  <name> hello </name>
  <tag-class> pack1>HelloTagHandler </tag-class>
  <body-content> empty </body-content>
</tag>
</taglib>

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

// web.xml

<web-app>

<jsp-config>

<taglib>

<taglib-uri> http://sathya.com/tags/ </taglib-uri>

<taglib-location> /WEB-INF/one.tld </taglib-location>

</taglib>

</jsp-config>

</web-app>

// test.jsp

<%@taglib uri = "http://sathya.com/tags"
prefix = "sathya" %>

<center>

A sample message

<sathya: hello />

Again sample message

</center>

SRIRAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Compilation

```
D:\customTag1\WEB-INF\classes> set classpath =  
C:\program files\Apache software foundation\  
Tomcat 7.0\lib\jsp-api.jar ; .
```

```
D:\customTag1\WEB-INF\classes> javac -d .  
HelloTagHandler.java
```

Deployment

- 1) Deploy the application in Tomcat server and start the server.
- 2) Open the browser and type the following request —

`http://localhost:2015/customTag1/test.jsp`

3 NOV 15

Adding attributes to custom tag

- 1) If we want to define attributes to the custom tag then there are two changes needed:-① In taghandler , create a variable with the same name as attribute name. Define setter and getter methods to that variable.
② In a tld file , configure attribute tag
- 2) Suppose , if we want to add an attribute called user to the custom tag Hello then we need to do the following changes to the above application —

(i) In HelloTagHandler.java , add the following code -

```
private String user;  
public void setUser( String user )  
{ this.user = user;  
}  
public String getUser()  
{ return user;  
}  
//doStartTag()  
//doEndTag()
```

(ii) open 'one.tld' file and add the following

"<attribute>" tag after <body-content>tag -

```
<attribute>  
  <name> user </name>  
  <required> true </required>  
  <value> runtimeexpression value <#exprvalue> true </#exprvalue>  
  </attribute>  
</tag>  
</taglib>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 1) If <required> ~~is~~ is true then attribute is mandatory for the tag . If it is false then attribute ~~is~~ is optional in the tag .
- 2) If <expsvalue> is true then runtime values are allowed to the attribute . If it is false then only static value is allowed .
- 3) In test.jsp , ~~def~~ write custom tag like the following -
`<sathy : hello user = "RAM" />`

Extending BodyTag support

- 1) If we want to execute logic at startTag , endTag , before body and after body of a custom tag then we need to extend a taghandler from body tag support .
- 2) If we want to execute a logic only at startTag and endTag then we extend TagHandler from TagSupport .
- 3) The logic of before body need to be define in doInitBody () and the logic after body define in doAfterBody .

< sathya : welcome > → doStartTag ()

• → doInitBody ()

==== Y body

• → doAfterBody ()

< /sathya : welcome > → doEndTag ()

4) doInitBody () returns void .

doAfterBody () returns int .

5) doAfterBody () returns one of the
following two int variables -

(i) EVAL_BODY AGAIN

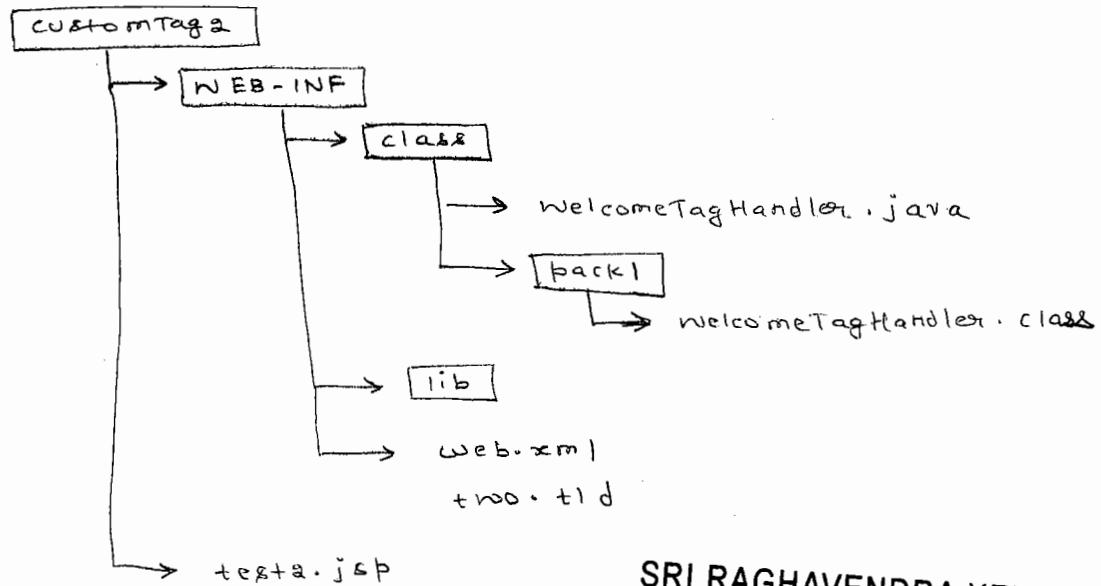
(ii) SKIP_BODY

6) If we want to execute body again
then we return EVAL_BODY AGAIN otherwise
we return SKIP_BODY .

Example

In this example we are creating a
customed tag welcome with body and
we repeat the body execution for
three times

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



// WelcomeTagHandler.java

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

```

package pack1;

import javax.servlet.jsp.tagext.BodyTagSupport;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;

public class WelcomeTagHandler extends BodyTagSupport
{
    int count = 0;

    public int doStartTag() throws JspException
    {
        JspWriter out = page.context.getOut();
        out.println(" welcome");
    }

    catch( Exception e )
    {
        return EVAL_BODY_INCLUDE;
    }
}
  
```

```

public int doAfterBody() throws JspException
{
    count++;
    if (count == 3)
        return SKIP_BODY;
    else
        return EVAL_BODY_AGAIN;
}

public int doEndTag() throws JspException
{
    try
    {
        JspWriter out = pageContext.getOut();
        out.println("Bye");
    }
    catch (Exception e)
    {
        return SKIP_PAGE;
    }
}

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagari Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

tmoo.tld

```
<taglib>  
<lib-version>1.2 </lib-version>  
<jsp-version> 1.3 </jsp-version>  
<tag>  
  <name> welcome </name>  
  <tag-class> pack1.WelcomeTagHandler </tag-class>  
  <body-content> jsp </body-content>  
</tag>  
</taglib>
```

web.xml

```
<web-app>  
<jsp-config>  
<taglib>  
  <taglib-uri> http://abcd.com/mytag </taglib-uri>  
  <taglib-location> /WEB-INF/ (tmoo.tld) </taglib-location>  
</taglib>  
</jsp-config>
```

```
</web-app>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Baker,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

// test2.jsp

```
<%@taglib uri = "http://abcd.com/mytags" prefix = "s"%>

<hr>
<center>
<s:welcome>
    <font color = 'red'> sample message </font>
<s:welcome>
<h1> A sample message after custom tag </h1>
</center>
```

1) compile TagHandler class
2) Deploy Application in Tomcat and start the server
3) open the browser and type the following request -

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

http://localhost:2015/customTag2/test2.jsp
o/p :

Welcome
sample message
sample message
sample message
BYE

4 NOV 15 JSP Expression Language (EL)

- 1) When working with JSP scripting elements, the JSP developers in industry have identified the following problems —
 - (i) When we are writing more no. of scripting element in jsp page then they are killing readability of the jsp page.
 - (ii) Scripting Element makes debugging of a jsp page as difficult.
 - (iii) With more no. of scripting element java code is also increased in a jsp page.
- 2) As a solution for the above problem Sun microsystem responded with expression language.
- 3) Expression language is an internal part of jsp and sun microsystem called it as a language because they given a new syntax to write the statements, operators and keywords.
- 4) Ext statement of expression language starts with $\$ \{$ Expression $\}$

5) we can use expression language statements as alternate for scripting element so that we can reduce the java code within a jsp page.

6) For ex,

scriptlet :

<%

Object o = request.getAttribute("key");

String value = (String)o;

out.println(value);

%>

EL statement :

`$ { requestScope.key }`

∴ The above ~~scriptlet~~ EL statement is equal to the scriptlet

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

EL operators

① Arithmetic : + - * / %

② Relational : < (lt) , > (gt) , <= (le) ,

③ logical : >= (ge) , == (eq) , != (ne)

and (and)

or (or)

! (not)

`$ { 10 gt 4 }` → true

`$ { 'battya' eq 'java' }` → false

EL implicit objects -

- 1) pageScope
- 2) requestScope
- 3) sessionScope
- 4) applicationScope
- 5) param
- 6) paramValues
- 7) cookie
- 8) pageContext ... etc

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

- 1) pageContext is a common implicit object in JSP and Expression language.
- 2) In Expression language, we use pageContext to call ~~other~~ implicit objects of JSP in EL statement.
- 3) We can't call java methods and we can't use java variables in EL statements.

Ex1- \${pageScope.getAttribute("key") } //wrong

Ex2- <%!
int number = 100;
>
\$ {number} //wrong

- a) We can use key of an attribute in EL statement because an attribute key is not a java variable.

Ex 2.1.

```
pageContext.setAttribute("k1", "java", 4);
```

%>

```
$ {applicationScope.k1} // correct
```

- b) suppose, if we want to call implicit object request of jsp in EL statement then we can use pageContext object like the following -

```
$ {pageContext.request.method}
```

↓ ↓ ↓
(implicit object) implicit calls
 of EL object of getMethod()
 jsp //getter

- c) If we want to use implicit object session of jsp in EL statement then we can use pageContext object like the following -

```
$ {pageContext.session.id}
```

↓ ↓ ↓
implicit object implicit calls
 of EL object of getId()
 jsp jsp



5 NOV 15

- 7) param is an implicit of EL, used to read the value of request parameter.
- 8) Suppose, \${param.uname} statement reads value of request Parameter uname, this EL statement is same as `request.getParameter("uname")`.
- 9) paramValues implicit objects is used to read the multiple values of the parameter.
- 10) Suppose, \${paramValues.hobby} statement reads the multiple values of request parameter hobby. This EL statement is same as `request.getParameterValues("hobby")`;
- 11) A jsp container will process EL statements based on the value of isELIgnored attribute of page directive.
- 12) The default value of isELIgnored is "false". It means Expression language is enabled.
- 13) If we want to disable expression language statement we need to set isELIgnored = "true"

Q) What is the difference b/w \${pagescope.k1} and \${k1} ?

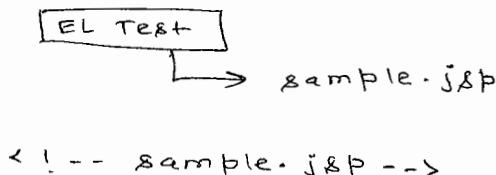
Ans - \${pagescope.k1} statement will only search for the key under pagescope, if not found then it prints a white space on browser.

\${k1} statement will search for the key from pagescope to application scope. If not found then displays/print a white space on browser.

Note →

- 1) In EL, null value will be converted to a white space

Example



SRI RAGHAVENDRA
Software Languages Material
Beside Bangalore Ayyagar Ba
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
<!-- sample.jsp -->  
<!-- @ page is ELIgnored = "false" -->  
<--
```

```
pageContext.setAttribute("k1", "A");  
pageContext.setAttribute("k2", "B", 2);  
pageContext.setAttribute("k3", "C", 3);  
pageContext.setAttribute("k4", "D", 4);
```

```
-->
```

```
</h1>
```

```
$ {pagescope.k1} <br>
```

```
$ {requestScope.k2} <br>
```

```
$ {sessionScope.k3} <br>
```

```
$ {sessionScope.k4} <br>
```

```
$ {k5} <br>
```

```
$ {applicationScope.k4}
```

```
</h1>
```

http://localhost:2015/ELTest/sample.jsp
o/p : A

B

C

D

JSP Standard Tag Library (JSTL)

- 1) In custom tag programming, JSP developer are not satisfied with the way of developing custom tags because it increase the burden on the jsp developers.
- 2) To avoid java code from jsp and to help JSP developers ~~to make~~ in order to make jsp as completely tag based, SunMicroSystem provided a group of predefined tags with the name of JSTL.
Sun MicroSystem
Publishing Java API
Standard Tag Library
- 3) sun divided JSTL tags into 'Five' sub libraries -
 - ① core tags
 - ② xml tags
 - ③ sql tags
 - ④ function tags
 - ⑤ Formatting tags

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

core tags

- 1) This tag library provides tags to define EL variables, conditions, iterations, redirection etc.

- 2) In a JSP page we mix and use JSTL tags and EL statements in a page.
- 3) If you want to use JSTL core tags in a JSP page then we should import a standard URI of core tags library into JSP page using taglib directive.

```
<!--@taglib uri = "http://java.sun.com/jsp/jstl/core"
prefix = "c"-->
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(i) <c: set> -

- 1) This tag is used to define EL variable with scope.
- 2) By using this `<c:set>` tag we can set a static value or runtime value to EL variable.
- 3) By default, each ~~EL~~ EL variable is string type.

Ex:

```
① <c:set var = "gtr" value = "pathya"
scope = "page" />
```

```
② <c:set var = "age" value = "26" />
(default scope is page)
```

③ <c:set var="str1" value="\${param.uname}" scope="page"/>

④ <c:set var="str2" value="\${param.pwdy"/>

(ii) <c:out>

- 1) This tag displays value of EL variable.
- 2) We can print the value of EL variable, by without using <c:out> tag also.

Ex:

① \${str1}

② <c:out value="\${str1}"/>

③ <c:out value="\${str1}" default="abcd"/>

- 3) The use of <c:out> tag is we can print default value if the given variable doesn't exist in a scope.

6/11/15 (iii) <c:remove>

- 1) This tag removes EL variable from a given scope.

<c:remove var="i" scope="request"/>

(iv) <c:if>

- 1) This tag is to define a condition within a jsp page.

<c:if> :

<c:if test="condition">

=====
</c:if>

For ex

```
<c:set var = "$1" value = "${param.username}" />
<c:if test = "${1 eq 'sathya'}">
<c:out value = "username is valid" />
</c:if>
```

(v) <c:choose>, (vi) <c:when>, (vii) <c:otherwise>:

- i) These core tags are used to implement switch case or elif, else statements in jsp.

Ex

```
<c:choose>
  <c:when test = "${color eq 'red'}">
    =
  </c:when>
  <c:when test = "${color eq 'green'}">
    =
  </c:when>
  <c:otherwise>
    =
  </c:otherwise>
</c:choose>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(IX) <c:forEach>

- 1) This core tag can be used as a for loop or a foreach loop.
- 2) With the help of this tag we can execute some code for fixed no. of times or we can execute some code over a collection.
- 3) To execute a code for fixed no. of times then we add attributes begin, end, then step.
- 4) To execute a code over a collection, we use items attribute.

Ex 1:

```
<c:forEach var = "i" begin = "1"  
          end = "10" step = "1" >
```

====

```
</c:forEach>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Ayan
Beside Bangalore Ayyagur Bank,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Ex 2:

```
int x[] = { 10, 20, 30, 40, 50 };
```

```
<c:forEach var = "k" items = "$x" >
```

====

```
</c:forEach>
```

(X) <c:forEachTokens>

- 1) This tag is used to execute a code repeatedly for each token of string.

For ex

```
<c:set var = "str"
      value = "This is a sample message"/>
```

```
<c:forTokens var = "t" items = "$str" />
```

====

```
</c:forTokens>
```

(x1)

<c:out>

- This tag is used to format or prepare a URL. The prepared URL will be stored in a variable.

a.jsp

```
<c:out var = "s" value = "/b.jsp"/>
```

```
<c:param name = "u"
```

```
value = "$param.uname"/>
```

</c:out>

~~/App1/b.jsp?u=sathyra~~

The above <c:out> tag prepares a value

/App1/b.jsp?u=sathyra and stores it in s variable.

- If the destination page is in another application then we need context attribute also.

```
<c:out var = "s" value = "/b.jsp"
       context = "/App2"/>
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Baker,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

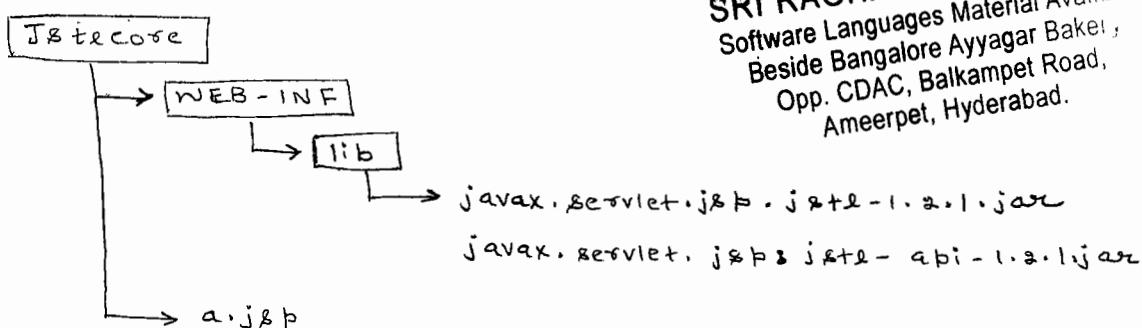
(xii) <c:redirect>

1) This tag redirects a request to another page.

```
<c:redirect url = " b.jsp" />
```

```
<c:redirect url = " $sby" />
```

Example



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Baker,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```

<!-- a.jsp -->
<!--@taglib uri = "http://java.sun.com/jsp/jstl/core"
prefix = "c" %>

<hi>
<c:set var = "str" value = "sathya" />
<c:out value = "${str}" /> <br>
<hi>
<c:if test = "${str ne null}">
  <h2><c:out value = "str is not null" /> </h2>
</c:if>
<br>
<hi>
<c:choose>
  <c:when test = "${str eq 'sathya'}">
    <c:out value = " str value is sathya" />
  </c:when>
<c:otherwise>
  <c:out value = " str value is not sathya" />
</c:otherwise>

```

```

</c:choose>
<br>
<c:forEach var="i" begin="1" end="5">
    <c:out value="Value of i is ${i}" /> <br>
</c:forEach>

<c:set var="line" value="This is a line of
message" />
<c:forTokens var="x" items="${line}" delim="" />
<c:out value="${x}" /> <br>

</c:forTokens>
<h1>
<br>
<c:url var="s" value="/b.jsp">
    <c:param name="u" value="sathy" />
</c:url>
<c:out value="${s}" />
</h1>

```

http://localhost:2015/jstlcore/a.jsp

sathy

str is not null

str value is sathy
 Value of i is 1
 Value of i is 2
 Value of i is 3
 Value of i is 4
 Value of i is 5

This
 is
 a
 line
 of
 message

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

/jstcore

✓ b.jsp; sessionid = 6CABEBCC6E29F8EA77A5A6D0DBE4E222P
u = sathyam

SQL Tags

- 1) From a jsp page if you want to connect with database and if you want to perform any operations then we can write jdbc code under `<scriptlet>` tag.
- 2) To reduce jdbc code within a jsp page, we can use sql tags of JSTL
- 3) To use sql tags of jstl in a jsp page, first we need to imports its uri like the following —

`<taglib uri = "http://java.sun.com/jsp/jstl/sql" />`

`prefix = "sql" />`

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Colony,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<sql: setdatasource >:

This tag is used to set connection properties.

```
<sql: setDataSource var="db"  
driver = "xxxxx"  
url = "xxxxx"  
username = "xxx"  
password = "xxx" />
```

(ii) <sql: update>

- 1) This tag is used to perform non select operations on database.
- 2) If the command is a dynamic command then its value can be set through <sql:param> tag.

Ex -

```
<sql:update var = "i" datasource = "${ds}">
```

```
delete from emp where deptno = ?
```

```
<sql:param value = "${param.dno}" />
```

```
</sql:update>
```

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(iii) <sql: query> :

- 1) This tag is only to execute select operation.
- 2) If the command contains any parameters then the values can be set with <sql:param> tag.

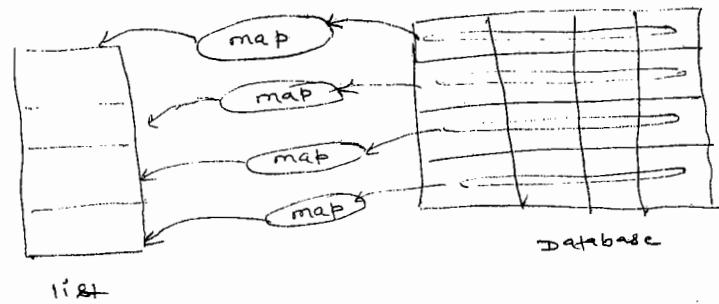
```
<sql:query var = "rs" datasource = "${ds}">
```

```
select * from emp
```

```
</sql:query>
```

- 3) JSTL has provided a predefined property called rows which returns all the selected rows stored in a variable in the form of a java.util.List object.

- 4) In `java.util.List` object, Map objects are stored. Internally, for each row a map object is created.



- 5) We can read each map object of list through a core tag of JSTL `<c:foreach>`

```

<c:foreach var = "k" items = "${rows->row$}">
    <c:out value = "${k.empno}" />
    <c:out value = "${k.ename}" />
    <c:out value = "${k.sal}" />
    <c:out value = "${k.deptno}" />

```

`</c:foreach>`

6) <sql:transaction>

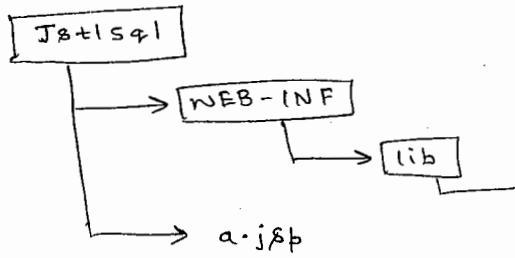
- i) This tag is used to execute multiple operations as one transaction.

```

<sql:transaction>
    <sql:update var = "i" datasource = "${ds}">
        xxxx
    </sql:update>
    <sql:update var = "j" datasource = "${ds}">
        xxxx
    </sql:update>
</sql:transaction>

```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyag. Bazaar,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.



javax.servlet.jsp-jstl-1.2.1.jar
 javax.servlet.jsp.jstl-api-1.2.1.jar
 jdbc.jar

<!-- a.jsp -->

<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql"
 prefix = "sql" %>

<%@ taglib uri = "http://java.sun.com/jsp/jstl/core"
 prefix = "c" %>

<sql: setDataSource var = "ds"
 driver = "oracle.jdbc.driver.OracleDriver"
 url = "jdbc:oracle:thin:@localhost:1521:xe"
 user = "system" password = "tiger"/>

<sql:update var = "i" datasource = "\${ds}">

update emp set sal = 9000 where empno = 7900

</sql:update>

<h1><c:out value = "\${i}" rowUpdated ...></h1>

<sql:query var = "ds" datasource = "\${ds}">

select * from emp

</sql:query>

<h2>

<c:forEach var = "m" items = "\${rows}">

<c:out value = "\${m.empno}" />

<c:out value = "\${m.ename}" />

<c:out value = "\${m.sal}" />

<c:out value = "\${m.deptno}" />

</c:forEach>

</h2>

http://localhost:2015/jstl.jsp/a.jsp ←

1 row updated

123	ANDY	7000	10
124	JAMES	8123	20
125	PIHU	7500	30

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q: How to precompile a jsp ?

Ans - 1) Pre-compiling a jsp means generating .java and .class file as ready in the server for a jsp page before a first request comes from browser.

2) The advantage of pre-compiling a jsp is a first client can also get faster response.

3) we can precompile a jsp by sending jsp-precompile as a parameter along with a request.

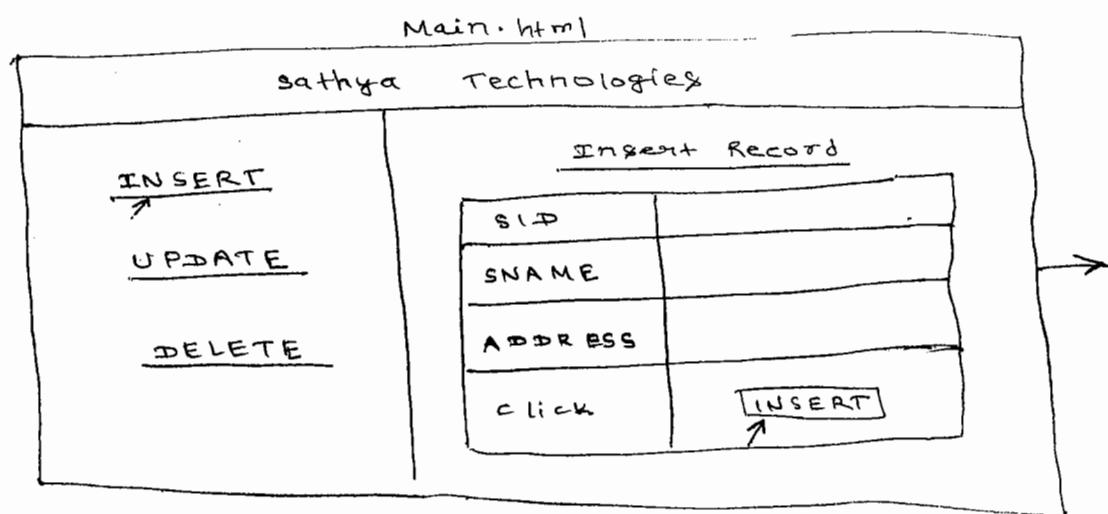
http://localhost:2015/App1/a.jsp?jsp-precompile ←
(or)

http://localhost:2015/App1/a.jsp?jsp-precompile=true ←

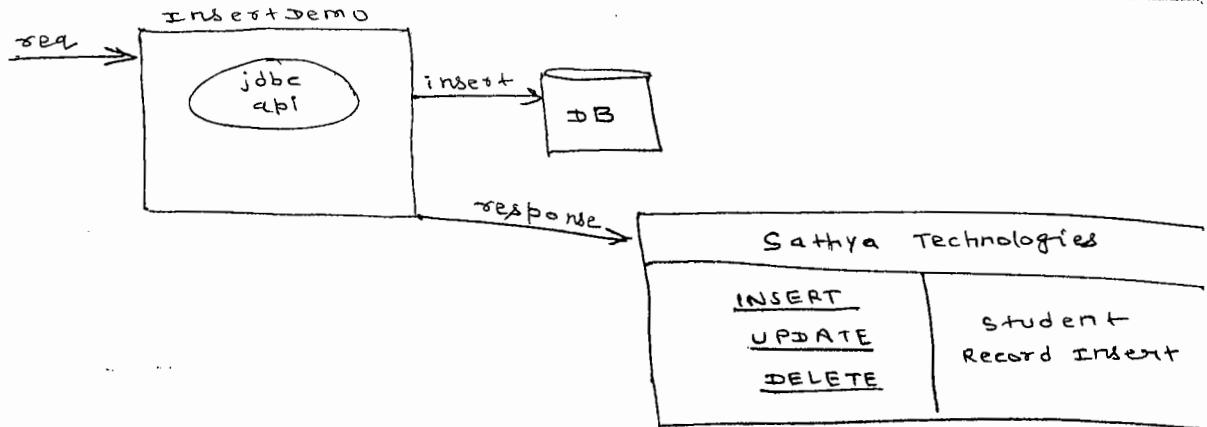
invalid { http://localhost:2015/App1/a.jsp?jsp-precompile=yes ←
 http://localhost:2015/App1/a.jsp?jsp-precompile=no ←

A sample project to perform crud operation

- 1) This project contains html, javascript, jsp, servlet and jdbc.
- 2) In this project a .js file is created to perform client-side validations.
- 3) we used <frameset> tag of html to divide a page into multiple frames.
- 4) After validations, if validations are success then a request is submitted to servlet and servlet executes database operation through jdbc code.
- 5) A sample flow of this application is like the following -



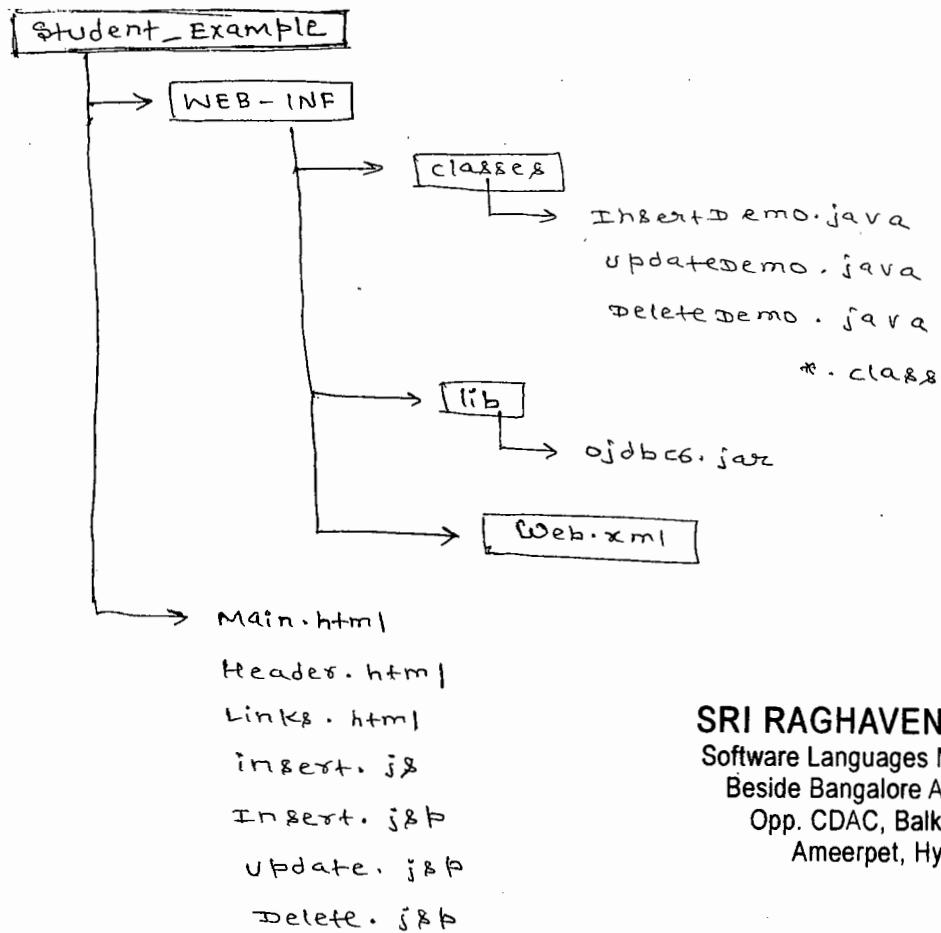
SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.



- 6) The validations applied through javascript are -
 - (i) stu_id must be entered and it must be a numeric value.
 - (ii) stu_name must be entered
 - (iii) stu_address must be entered
- 7) we have added external javascript file to insert.jsp and update.jsp and we have written internal javascript code to delete.jsp
- 8) Before we execute this project we need to create in oracle database like the following —

```
create table student_info (stu_id number(5)
                           primary key, stu_name varchar2(10),
                           stu_add varchar2(10));
```

SRI RAGHAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Baker
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

10NOV15
Absent

WEB - SECURITY

- 1) A web application is a group of resources and it may contain some static resources and some dynamic resources.
- 2) A web application resources can be ~~secured~~ either secured or non-secured.
- 3) A non-secured resource is accessible to all the clients. It is public resource.

- 4) A secure resource is only accessible to authorized client. They are called protected resources.
- 5) Providing security to a web resource is nothing but authenticating and authorizing a client, before accessing a particular resource.
- 6) Authentication is nothing but verifying the username and password, and authorization is nothing but checking the access permission of authenticated client.
- 7) If username and password are valid then it is not possible to access every resource, access permission is also required.
- 8) For ex, If we show a flight ticket and id proof then we will get a boarding pass. It means a passenger is authenticated.
- 9) with that boarding pass, we don't have any permission to sit in any flight. We are authorized to sit only in a particular flight.
- 10) In a web application to provide security for a servlet or a JSP, we have two ways —

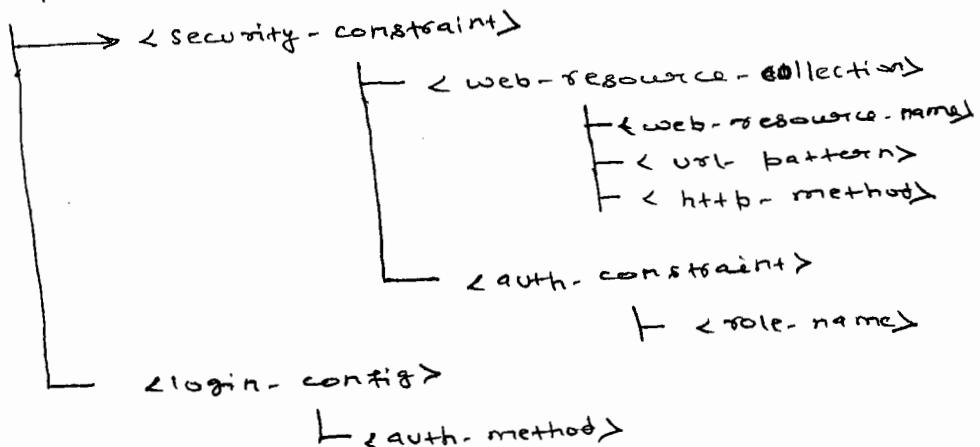
- 1) programmatic security
 - 2) declarative security
- 11) In programmatic security, we can define to filter ~~manually~~ ^{manually} for authentication and authorization.
- 12) When request send to a secure servlet then authentication filter verifies username and password then authorization filter checks for permissions. If authorized then that request is allowed and send to servlet.
-
- ```

graph LR
 Browser[Browser] --> AF[Authentication Filter]
 AF --> AF[Authorization Filter]
 AF --> SS[Secure Servlet]

```
- 13) Every authenticated user may not be authorized user, but every authorized user is an authenticated user.
- 14) A problem in programmatic security is, a programmer has to manually define logic in filters and a programmer has to configure filters manually in web.xml.
- 15) In declarative security, a programmer used to configure <security> tags in web.xml. Automatically servlet container will authenticate and authorize a user. So, declarative security is better than programmatic security.

- 16) In web.xml, we need to configure security tags in the following order to get declarative security

<web-app>



- 17) To provide declarative security to web resource, we need to select one of the four authentication models —

- (i) Basic Authentication Model      (iii) Form based Authentication Model  
(ii) Digest Authentication Model    (iv) Client certification or SSL

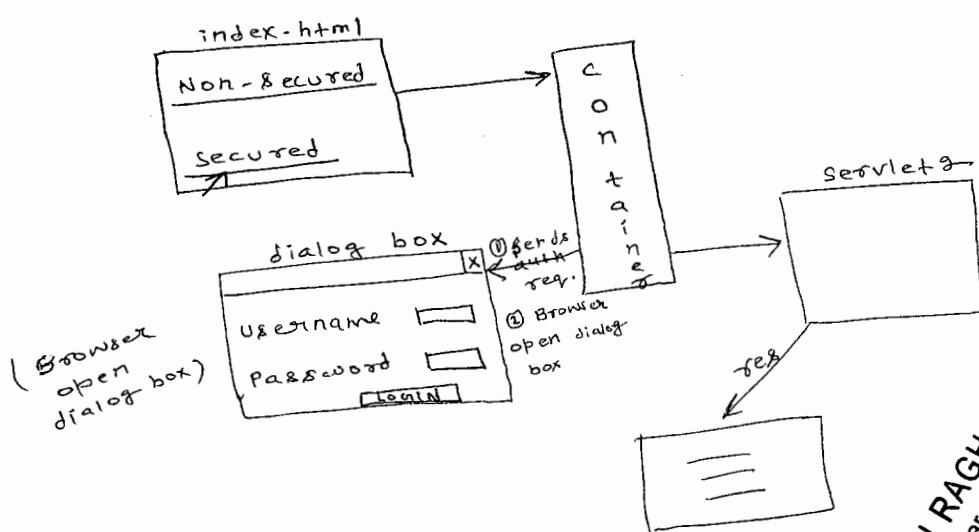
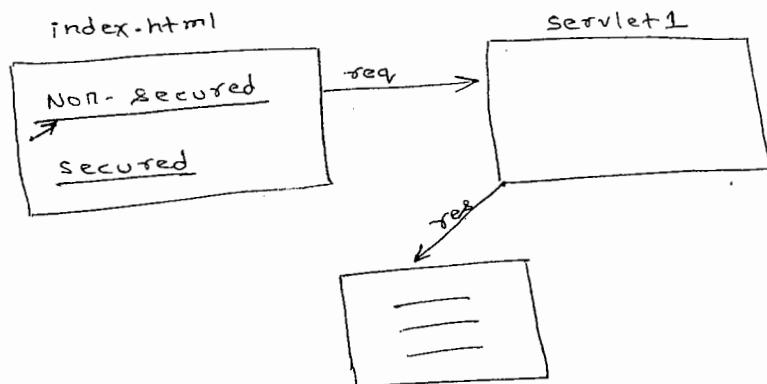
#### Basic Authentication Model

- This web-application authentication models works like the following —

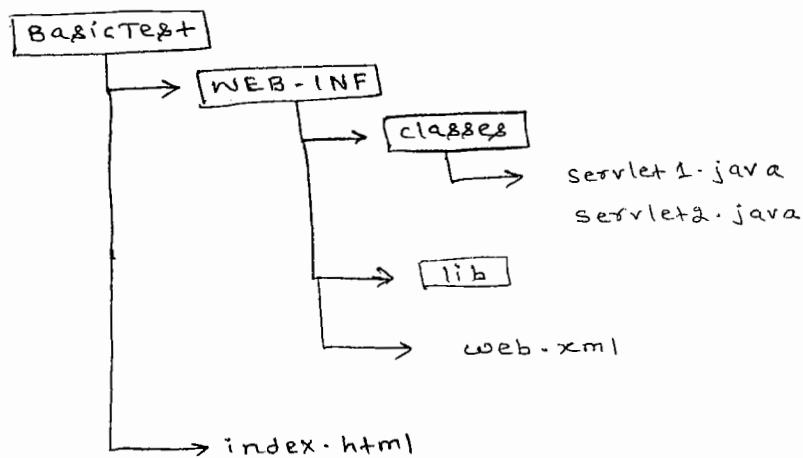
- 1) A client (end user) will send request to secure servlet from browser.
- 2) A servlet container will tell the browser that authentication is required and BASIC is auth. model.
- 3) A browser opens dialog box to enter the username and password. Browser will send username and password to container. Container verifies username and password. If not valid again step ②, ③ are required.
- 4) If valid then a client is allowed to access that servlet.
- 5) A best example for BASIC Authentication model is Tomcat Manager. We click on Tomcat Manager App button in Tomcat Homepage then a request is sent to Manager servlet. But it is secure servlet so, Tomcat container tell browser, the authentication req. and browser opens dialog box for enter uname and pwd.

18 Nov 15

Example



Directory structure



```
<!-- index.html -->
<h1>
 <center>
 Non secured

 Secured
 </center>
</h1>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet, Hyderabad,

```

// servlet1.java

import javax.servlet.*;
import java.io.*;

public class servlet1 extends genericServlet
{
 public void service(ServletRequest request, -

 ServletResponse response) throws

 SE, IOException
 {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<h1> Response from Non

 secured servlet </h1>");
 out.close();
 }
}

```

// servlet2.java

```

import javax.servlet.*;
import java.io.*;

public class servlet2 extends genericServlet
{
 public void service(ServletRequest request, -

 ServletResponse response) throws

 SE, IOException
 {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<h1> Response from Secured servlet

 </h1>");
 out.close();
 }
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Baker  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

web.xml

```
<web-app>
 <servlet>
 <servlet-name> s1 </servlet-name>
 <servlet-class> servlet1 </servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name> s1 </servlet-name>
 <url-pattern> /servlet1 </url-pattern>
 </servlet-mapping>

 <servlet>
 <servlet-name> s2 </servlet-name>
 <servlet-class> servlet2 </servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name> s2 </servlet-name>
 <url-pattern> /servlet2 </url-pattern>
 </servlet-mapping>

 <!-- security configuration -->
 <security-constraint>
 <web-resource-collection>
 <web-resource-name> abc </web-resource-name>
 <url-pattern> /servlet2 </url-pattern>
 <http-method> GET </http-method>
 </web-resource-collection>
```

```
<auth-constraint>
 <role-name> student </role-name>
```

```
<(auth-constraint)>
```

```
</security-constraint>
```

```
<login-config>
```

```
 <auth-method> BASIC </auth-method>
```

```
<(login-config)>
```

```
<web-app>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Note-

- 1) we need to configure role-name with one or more username and passwords in tomcat-users.xml of tomcat/conf folder.

```
<user username = "RAM" password = "ROM"
 roles = "student" />
```

```
<user username = "suresh" password = "Ramesh"
 roles = "student" />
```

```
<user username = "ABC" password = "XYZ"
 roles = "customer" />
```

```
</tomcat-users>
```

- 2) compile the servlets and deploy the application in Tomcat and then start the server

- 3) Type the following url —

http://localhost:2015/BasicTest

## 2) Form based Authentication model —

- 1) In this authentication model, servlet container will send a login page to the browser, when a user sends request to the secured resource.
- 2) If username and password is not valid then container will send error page to the browser.
- 3) The login page and error page should be defined by application developer and the pages must be configured in web.xml like the following —

```
<login-config>
```

```
 <auth-method> FORM </auth-method>
```

```
 <form-login-config>
```

```
 <form-login-page>/login.html </form-login-page>
```

```
 <form-error-page>/login_failed.html </form-error-page>
```

```
 </form-login-config>
```

```
</login-config>
```

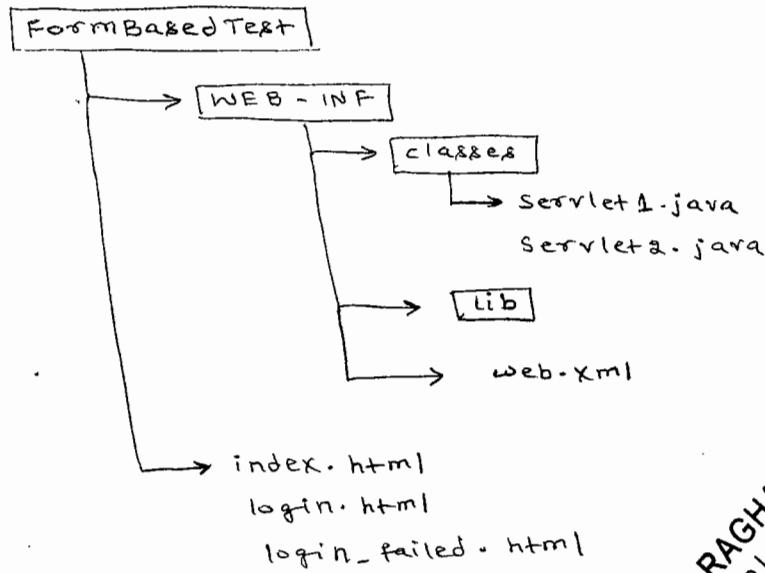
SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 4) In form based authentication, when a login page submitted then a predefined servlet (security servlet) will check the username and password. so, while designing the form, form action and username, password textbox name are predefined.

- 5)   
 i) form action must be j\_security\_check.  
 ii) name of username textbox must be j\_username  
 iii) name of password textbox must be j\_password

Example

directory structure



Error code

403 : Authorization Failed
401 : Authentication Failed

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery  
 O.P.C. CDAC, Balkampet, Bangalore  
 Tel: 080-25422222

// login.html

```

<html>
<body>
<center>
 <h2> Please enter your username and password </h2>
 <p>
 <form method = "post" action = "j_security_check">
 <table border = 1>
 <tr>
 <td> Username : </td>
 <td> <input type = "text" name = "j_username"> </td>
 </tr>
 <tr>
 <td> Password </td>
 <td> <input type = "password" name = "j_password"> </td>
 </tr>
 </form>
 </p>
</center>
</body>
</html>

```

```
<tr>
<td colspan = 2 align = right>
<input type = submit value = "submit"> </td>
</tr>
</table> <(form)> <(center> </body> </html>
```

### //login-failed.html

```
<h1> Your credentials are not valid </h1>
```

```


```

```
 Enter Again
```

→ In web.xml, change `<login-config>` tag like the following -

```
<login-config>
```

```
<auth-method> FORM </auth-method>
```

```
<form-login-config>
```

```
<form-login-page> /login.html </form-login-page>
```

```
<form-error-page> /login-failed.html </form-error-page>
```

```
</form-login-config>
```

```
</login-config>
```

```
</web-app>
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Using Database for authentication

- 1) For authenticating a user, Tomcat container reads usernames, password and the roles from Tomcat-users.xml . It is called in-memory authentication .
  - 2) A problem with in-memory authentication is suppose if a new user is added or existing user is deleted then everytime we need to restart the server because, a container can read tomcat-users.xml at startup time only .
  - 3) We can solve the above problem by using database for maintaining usernames , password and their roles . This is called database authentication .
  - 4) In database authentication , a new user can be added or existing user can be removed , by without restarting the server .
  - 5) To add database authentication to a web-application security , do the following changes —
    - i) create the following tables and insert sample records in mysql
- Table-1
- ```
create table users ( user_name varchar(15) not null,
                     userpass varchar(15) not null
                   );
```

Table 3

```
create table user_roles (user_name varchar(15) not null,  
                        role_name varchar(15) not null,  
                        primary key (user_name, role_name)  
);
```

```
SQL> insert into users values ('James', 'coding');
```

```
SQL> insert into users values ('Amit', 'java');
```

```
SQL> commit;
```

```
SQL> insert into user_roles values ('James', 'student');
```

```
SQL> insert into user_roles values ('Spring', 'FAM');
```

```
SQL> commit;
```

(ii) open server.xml of Tomcat and add the
following <Realm> tag

```
</Realm>
```

```
<Realm
```

```
className = "org.apache.catalina.realm.JDBCRealm"
```

```
driverName = "com.mysql.jdbc.Driver"
```

```
connectionURL = "jdbc:mysql://localhost:3306/test"
```

```
connectionName = "root"
```

```
connectionPassword = "root"
```

```
userTable = "users" usernameCol = "user_name"
```

```
userCredCol = "user_pass"
```

```
userRoleTable = "user_roles"
```

```
roleNameCol = "role_name" />
```

```
<Host name - - - - ->
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(iii) copy mysql jar file to the lib folder of Tomcat

6) start the Tomcat server and execute either BASIC Test or FormBasedTest from the browser.

SSL Authentication / client certificate Authentication

- 1) SSL Authentication is not for verifying the Username, password and the Role. It is for strongly encrypting data and transferring b/w browser and server.
- 2) SSL authentication uses RSA algorithm for encrypting and decrypting the data.
- 3) In SSL authentication, first a certificate will be send to the browser by server.
- 4) A browser verifies whether the server to whom it wants to connect is same or not after that a browser accepts a server certificate.
- 5) If certificate is accepted by browser then only browser and server can communicate.

- 6) If certificate is accepted, server will send encrypted data to the browser and browser side it is decrypted with the help of key's available in certificate.
- 7) A browser will send encrypted data to the server and at server side it will be decrypted.
- 8) The data transfer b/w browser and server with https protocol. so, it is not possible to hack the data in the middle.
- 9) If you want authentication and authorization of user alongwith SSL then we can SSL with basic or SSL with FORM based.
- 10) For realtime applications, a trusted certificate authority generates a security certificate for an organisation.
- 11) Some trusted 3rd party certificate authority are -
(i) Verisign (ii) B&B (iii) thawte etc.
- 12) For java applications, we can generate a security certificate (untrusted) by using Keytool command of jdk.

14-NOV-15

(i) In a web application, if you want to enable SSL authentication, the following changes are required —

(ii) we need to generate a certificate with keytool like the following —

```
3:1> keytool -genkey -alias abcd -keyalg RSA  
-keystore F:/cert/keystore
```

Enter keystore password : mosling

Re-enter new password : mosling

What is your first and last name ?

[Unknown] : a b

What is the name of your organizational unit?

[Unknown] : sathya

What is the name of your organization?

[Unknown] : sathya tech

What is the name of your city or Locality?

[Unknown] : hyd

What is the name of your state or province?

[Unknown] : ts

What is the two-letter country code for this unit?

[Unknown] : in

If CN = "a. b" OU = sathya, L=hyd, ...

[No] : yes

Enter key password for abcd

(Return it same as keystore password)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- (ii) Open server.xml in Tomcat / conf directory and then remove comments for <connector> tag with port 8443 and add the 2 attributes to that <connector> tag -
- keyStoreFile = " F:/cert.keyStore"
keyStorePass = " moshing"
- (iii) open web.xml of FormBasedTest application and add the following tag after closing </auth-constraint>
- <user-data-constraint>
<transport-guarantee> CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
- (iv) start the tomcat service and then send the application from browser <http://localhost:2015/FormBasedTest>.
- (v) click on secured link then request is redirected to <https://localhost:8443/FormBasedTest> and container will send a certificate to Browser.
- (vi) If the certificate accepted then server will send login page back to the browser.

Security Annotation

- 1) with the help of security annotation we can avoid <security-constraint> tag from web.xml
- 2) <login-config> tag cannot be removed from web.xml because we don't have annotation.
- 3) The two security annotations are —
 - (i) @ServletSecurity
 - (ii) @HttpMethodConstraint
- 4) @HttpMethodConstraint can be written in @ServletSecurity and @ServletSecurity is added on top of servlet class.

For ex -

```
@ServletSecurity( httpMethodConstraints = @HttpMethodConstraint( value = "POST" , rolesAllowed = "student" ))
```

```
@WebServlet( value = "/govi" , name = "govi" )  
public class servlet1 extends genericServlet  
{  
    public void service( req, res ) throws SE, IOE  
}
```

≡

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Baker
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

→ security is applied to servlet, when POST method request is sent from browser.

Ex2

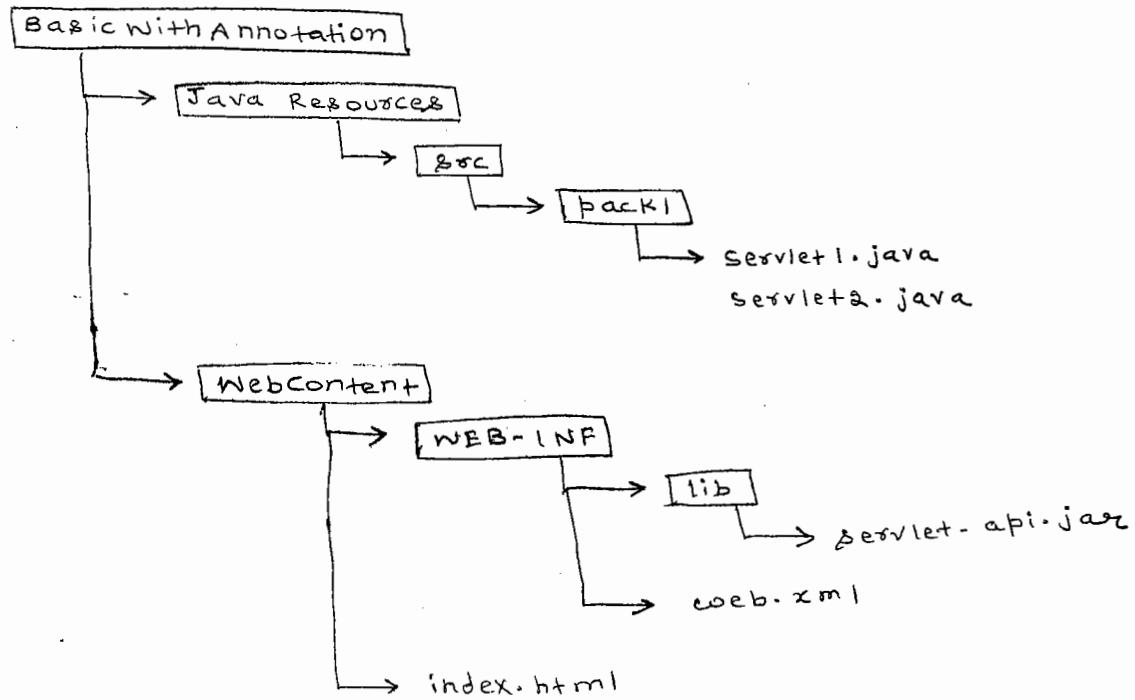
```
@ServletSecurity (httpMethodConstraints =  
    { @HttpMethodConstraint (value = "GET",  
        rolesAllowed = "student"  
    ),  
  
    @HttpMethodConstraint (value = "POST",  
        rolesAllowed = "student"  
    )  
}  
  
@WebServlet (value = "/servlet", name = "some")
```

```
public class Servlet1 extends GenericServlet  
{  
    public void service (req, res) throws SE, IOE  
    {  
        ==  
    }  
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

→ security is applied to Servlet1 when the request is sent from browser with GET() or POST().

16 NOV 15



```
//servlet1.java
package pack1;

@WebServlet(value = "/servlet1", name = "sone")
public class Servlet1 extends GenericServlet
{
    public void service(req, res) throws SE, IOE
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

//Servlet2.java

```
package pack1;

@WebServlet(value = "/servlet2", name = "servlet2")
@ServletSecurity ( httpMethodConstraints =
    {
        @HttpMethodConstraint ( value = "GET",
            rolesAllowed = "student" )
    }
}

public void service( req, res ) throws SE, IOE
{
}
```

// web.xml

```
<web-app>
```

```
  <login-config>
```

```
    <auth-method> BASIC </auth-method>
```

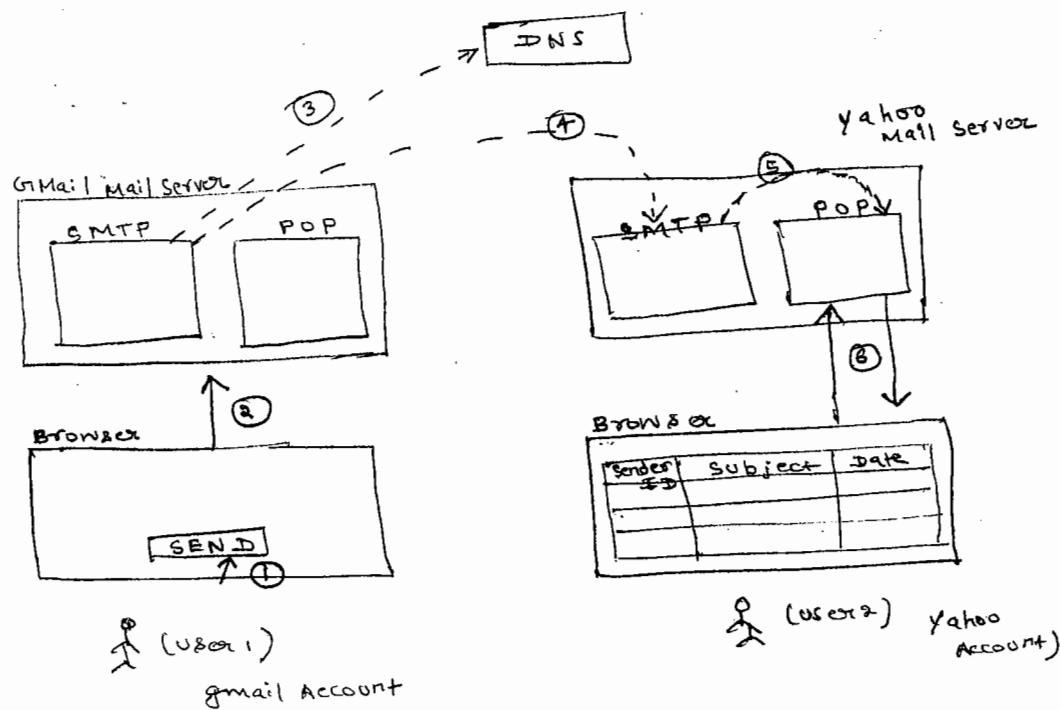
```
  </login-config>
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Servlet with Java Mail API

How Email works ?

AVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



- 1) User1 composes a mail and sends it to the receiver (another account).
- 2) A request will be send from browser to SMTP server of Gmail Mail server. Here Message object is created.
- 3) SMTP server verifies whether receiver is also on same mail server or not. If not then it connects to domain name server (DNS) to find the location of another mail server in network.

- 4) SMTP server sends message to SMTP server of Yahoo mail server.
- 5) SMTP server of Yahoo will transfer the message to POP server. POP server will store the message in inbox folder of that account in the form of a file.
- 6) When user connects to POP server by entering the mail id and password then POP server will read first lines of the file stored in inbox folder and returns them as hyperlinks to the browser.

→ When we send the email to destination account then that email will be stored in inbox folder as a file but it will not be stored in database.

→ Every mail server has two servers internally —

(i) SMTP (simple mail transfer protocol)
/ outgoing server

(ii) POP (Post office protocol) / incoming server

- Java Mail API is a part of JEE, released by sunMicrosystem for sending an email from java application and also for receiving an email through java Application.
- Mostly we use Java Mail API in realtime project only for sending an email to destination account but not for receiving a mail.
- Java Mail API contains two package -
- (i) javax.mail
 - (ii) javax.mail.internet

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

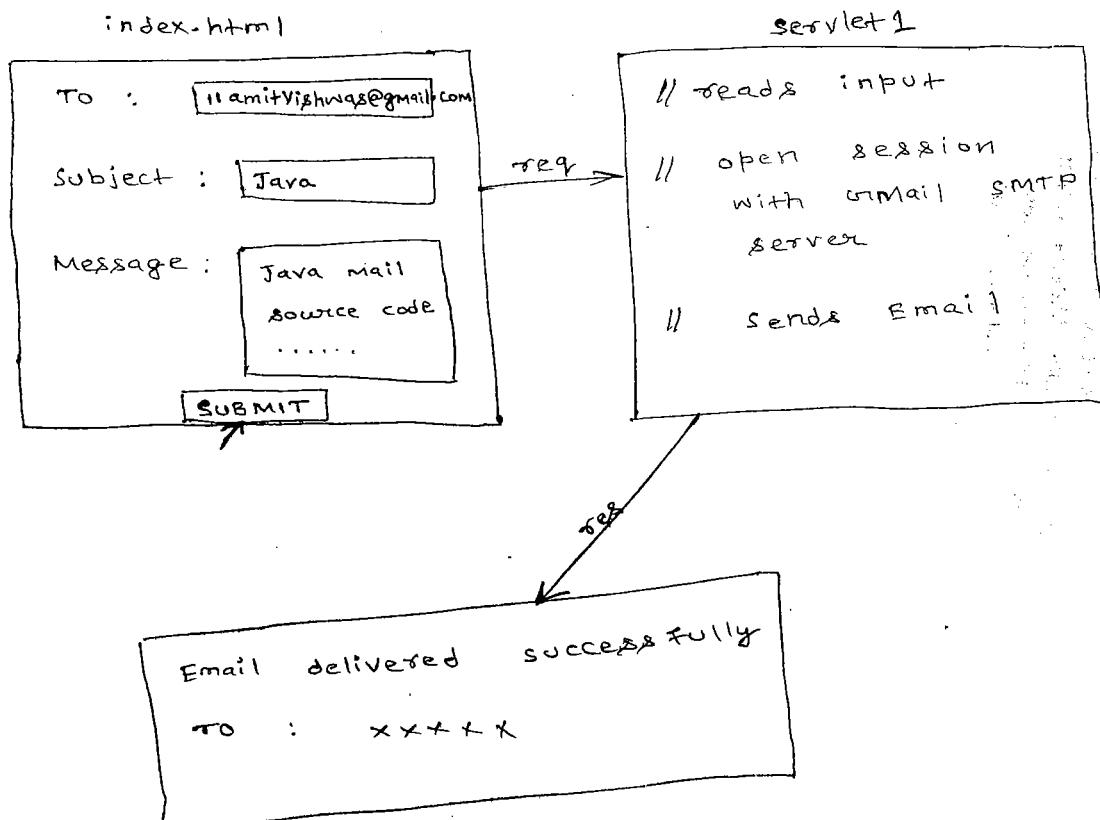
- The classes and interfaces are provided in above two packages can be used in any type of java application. It means we can use in a console application (or) a web application (or) desktop application (or) remote application.
- If it is non web Application then we need to add mail.jar and activation.jar to classpath. If it is web Application then we need to add them to lib directory.

17 NOV 15

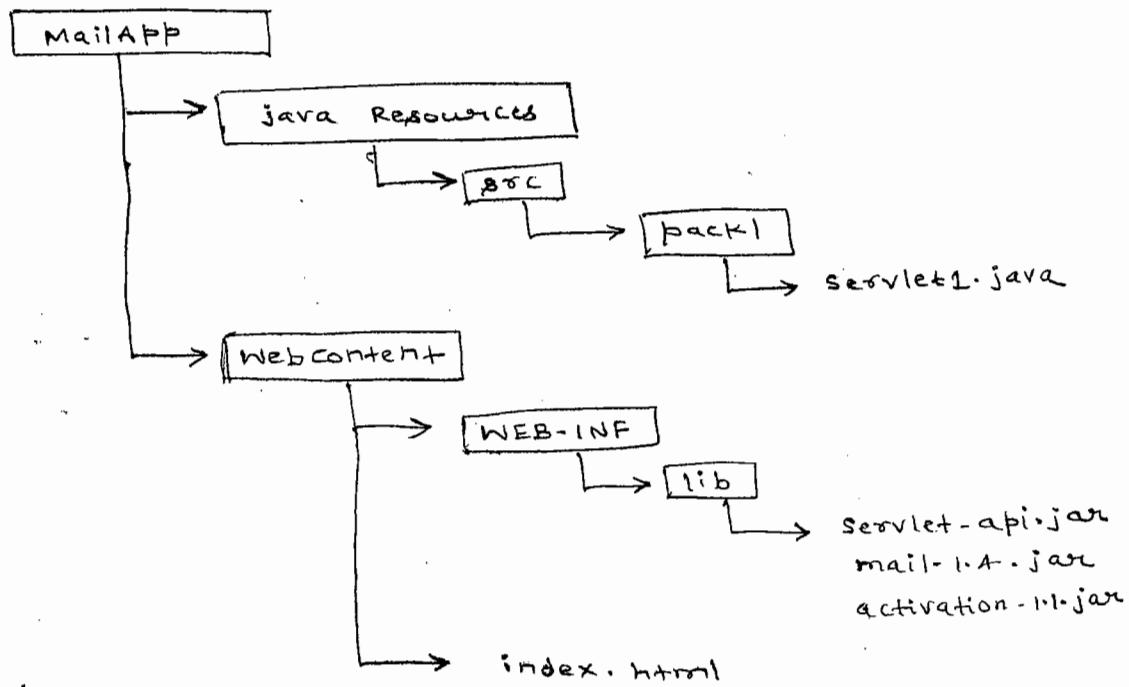
Steps to send an Email from java program -

- 1) set the mail server properties from SMTP
- 2) open a session with mail server
- 3) create a message object and set the message properties
- 4) send message

Flow



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



KRISHNAVENDRA XEROX
 Software Languages Material Available
 Beside Bangalore Ayyagar Bakery,
 Opp. CDAC, Balkampet Road,
 Ameerpet, Hyderabad.

index.html

```

<center>
<h1> Enter Details ... </h1>
<hr>

<form action = "servlet1" method = "post" >
  To : <input type = text name = "to" > <br>
  Subject : <input type = text name = "subject" > <br>
  Message : <textarea rows = 5 cols = 40
                name = "message" >
    </textarea>
    <input type = submit value = "SEND" >
</form>
</center>
  
```

Servlet-1.jar

```
@WebServlet(value = "/gsv1", name = "gsvie")
```

```
public class Servlet1 extends HttpServlet
```

3

```
protected void doPost(HttpServletRequest request,  
                      HttpServletResponse response) throws  
SE, IOException
```

// read input values from html

```
String to = request.getParameter("to");
```

```
String subject = request.getParameter("subject");
```

```
String msg = request.getParameter("message");
```

48tcp-1

```
Properties mailServerProperties = new Properties();
```

```
mailServerProperties.put("mail.smtp.host", "smtp.gmail.com");
```

(“mail.smtp.port”, “587”);

```
("mail.smtp.auth", "true")
```

SOP (" mail server properties have been setup

successfully").

18tep- 2

// get the session object

```
Session session = session.getInstance(mailServerProperties)
```

Anonymous
implementation

```
new javax.mail.Authenticator()
{
    protected PasswordAuthentication getPasswordAuthentication()
    {
        return new PasswordAuthentication("hamitrishwas@gmail.com", "Java");
    }
};
```

// Step-3

```
// create a default MimeMessage object  
Message message = new MimeMessage(session);  
  
// Set To : header field of the header  
message.setRecipient(Message.RecipientType.TO,  
                     new InternetAddress(to));  
  
// Set Subject : header field  
message.setSubject(subject);  
  
// Now set the actual message  
message.setText(msg);  
message.setFileName("D:/aku.pdf"); // for attachment  
// (if any)  
// optional
```

// Step 4

```
Transport t = session.getTransport("smtp");  
t.send(message);  
  
response.setContentType("text/html");  
  
PrintWriter out = response.getWriter();  
  
out.println("<h1> Email successfully sent  
to : " + to + "</h1>");
```

catch(Exception e)

{ e.printStackTrace(); }

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Sending Email with Attachment

- After setting subject to the message object, we need to add the following code -

~~Part 1~~

```
MimeBodyPart part1 = new MimeBodyPart();  
part1.setText(msg); -> text  
  
MimeBodyPart part2 = new MimeBodyPart();  
part2.setfilename("D:/aku.pdf"); -> attachment
```

~~Multipart~~

```
MimeMultipart multipart = new MimeMultipart();  
multipart.addBodyPart(part1);  
multipart.addBodyPart(part2);  
message.setContent(multipart);  
  
Transport t = session.getTransport("smtp");  
t.send(message);
```

X

Faculty :- Sekhar sir

Email : srinivas8919@gmail.com

Contact : 9885340306

FB group : Java sekhar

Student :-

Email : 11amitvishwas@gmail.com